



Instituto Superior de Engenharia

Politécnico de Coimbra

DEPARTAMENTO DE / DEPARTMENT OF
INFORMÁTICA E SISTEMAS

Desenvolvimento de um protótipo de um Futebol GPT

Relatório de Estágio para a obtenção do grau de Mestre em
Informática

Especialização em Desenvolvimento de Software

Autor

Andreia Filipa Almeida Graça

Orientador

Ana Cristina Costa Oliveira Alves

Supervisor na empresa WIT Software

Nuno Carvalho

Coimbra, Setembro de 2024



INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

RESUMO

Nos últimos anos, a Generative AI tem-se destacado como um dos campos mais inovadores, observando-se significativos avanços em diversas áreas: informática, *marketing* ou saúde. A popularidade da tecnologia tornou-se mais evidente com o lançamento do *chatbot* ChatGPT da OpenAI em 2022. Em paralelo, o futebol permanece ao longo dos séculos como o desporto mais popular do mundo, somando milhões de adeptos que consomem informações de forma constante para se manterem atualizados. Ao fundir estes dois mundos, surge uma proposta promissora materializada no estágio curricular denominado SportGPT. Este projeto visa o estudo e desenvolvimento de um GPT do futebol (semelhante a um ChatGPT, mas centrado somente nessa temática), um protótipo que permite aos utilizadores realizarem perguntas futebolísticas e obterem respostas em tempo real, atualizadas e precisas. O desenvolvimento será realizado em linguagem *python*, utilizando LLMs (*Large Language Models*) da OpenAI e a uma base de dados de grafos (Neo4J). A interface da aplicação *mobile* será construída em Flask. Os LLMs são modelos treinados com enormes quantidades de informações com fim a compreender e gerar texto em linguagem natural, responder a perguntas ou outras tarefas linguísticas. Convém ressaltar que os LLMs enfrentam desafios notáveis como geração de respostas imprecisas / incorretas (conhecidas como alucinações) e ainda desatualização dos dados. Contudo, ao recorrer a uma base de conhecimentos interna com dados de futebol e descartando os conhecimentos do LLM poderá ser possível mitigar esses problemas, garantindo a fiabilidade das respostas fornecidas. Assim, o projeto SportGPT pretende ser uma ferramenta robusta e precisa para os entusiastas e curiosos do futebol.

ABSTRACT

In recent years, Generative AI has emerged as one of the most innovative fields, with significant advancements in various sectors such as computing, marketing, and healthcare. The technology's popularity became more evident with the launch of OpenAI's ChatGPT in 2022. Meanwhile, football has remained the world's most popular sport for centuries, attracting millions of fans who constantly consume information to stay updated. By merging these two worlds, a promising proposal has emerged, materialized in the internship project called SportGPT. This project aims to study and develop a football-specific GPT (similar to ChatGPT but focused solely on this theme), a prototype that allows users to ask football-related questions and receive real-time, updated, and accurate answers. The development will be carried out in Python, using OpenAI's Large Language Models (LLMs) and a graph database (Neo4J). The mobile application interface will be built using Flask. LLMs are models trained on vast amounts of data to understand and generate natural language text, answer questions, and perform other linguistic tasks. It is worth noting that LLMs face notable challenges, such as generating inaccurate or incorrect responses (commonly referred to as hallucinations) and having outdated data. However, by relying on an internal knowledge base focused on football data and discarding unrelated information from the LLM, it may be possible to mitigate these issues, ensuring the reliability of the answers provided. Thus, the SportGPT project aims to be a robust and accurate tool for football enthusiasts and curious users alike.

AGRADECIMENTOS

Os dez meses mais desafiantes e intensos chegaram ao fim, e, por isso, gostaria de expressar o meu profundo agradecimento a todos os que me apoiaram e ajudaram a alcançar esta etapa.

Em primeiro lugar, agradeço às pessoas da WIT Software envolvidas no estágio curricular, especialmente ao João Nabais, com quem interagi diariamente, por toda a ajuda e tempo dedicados para me auxiliar. Queria agradecer também à Mylena Dias, por toda a ajuda, por me ouvir e pela força que me deu para continuar. Agradeço igualmente à Ana Pires, do departamento dos recursos humanos, pelo valioso *feedback* que me forneceu ao longo das apresentações mensais na empresa, o qual contribuiu para o crescimento da minha confiança em falar em público. Manifesto ainda a minha gratidão a todos os outros que dedicaram o seu tempo para me ajudar, responder a perguntas e oferecer explicações. Obrigada à WIT Software pela confiança e oportunidade e evolução académica.

De seguida, gostaria de expressar a minha gratidão à minha orientadora do ISEC, a professora Ana Alves, pela constante disponibilidade para me apoiar e ouvir nas diversas dificuldades que enfrentei ao longo do percurso.

De coração, agradeço à minha família, especialmente aos meus pais e à minha avó materna, por me permitirem dedicar cinco anos da minha vida a estudar, aprender e construir o meu futuro. Por todo o esforço e sacrifício em prol de um futuro melhor. Expresso a minha mais sincera gratidão. Sem o apoio deles, nada disto seria possível.

Agradeço também às minhas amigas, Liliana, Ana Margarida e Beatriz, e em especial, ao meu namorado, pelo apoio incondicional, por me ouvirem e me ajudarem a superar as minhas inseguranças.

Por fim, quero agradecer a mim mesma, pela resiliência, por não desistir, por me desafiar constantemente e, apesar das muitas provas e desafios, ter conseguido completar este percurso sem baixar os braços.

ÍNDICE

Resumo	I
Abstract.....	II
Agradecimentos	III
Índice de Figuras	VII
Índice de Tabelas	IX
Lista de Acrónimos e Siglas.....	XI
1 Introdução	1
2 Plano de Trabalhos	3
3 Conceitos Fundamentais.....	5
3.1 Generative AI	5
3.2 Generative Adversarial Network.....	9
3.3 Large Language Models (LLMs)	11
3.3.1 Tokenização	13
4 Estado da Arte	19
4.1 Trabalho Relacionado.....	19
4.2 Conceitos.....	22
4.3 Modelos de Bases de Dados	23
4.3.1 Base de Dados SQL	23
4.3.2 Bases de Dados Não Relacionais.....	25
4.3.3 Discussão.....	30
4.4 OpenAI	35
4.4.1 GPT-1	35
4.4.2 GPT-2.....	36
4.4.3 GPT-3	37
4.4.4 GPT-3.5 Turbo.....	37
4.4.5 GPT-4	37
4.4.6 GPT 4-Turbo	38
4.4.7 GPT-4 vs GPT-4 turbo	39
4.5 Alucinações de LLM.....	40
4.6 Retrieval-Augmentend Generation (RAG) Applications.....	44
4.6.1 Comparação entre utilização de Graph RAG e outras melhorias do LLM [125] :.....	49

Desenvolvimento de um protótipo de um Futebol GPT

4.6.2	Vantagens das RAG Apps:.....	50
4.6.3	Desvantagens / desafios das RAG Apps:.....	50
5	Tecnologias.....	51
6	Metodologia.....	55
7	SportGPT.....	57
7.1	Análise de Requisitos.....	57
7.1.1	Requisitos Funcionais.....	57
7.1.2	Requisitos Não Funcionais.....	57
7.2	Fontes de Dados.....	58
7.3	Arquitetura.....	58
7.3.1	Escolha do Tipo da Base de Dados.....	62
7.4	Integração com Frameworks.....	63
7.5	Diagramas.....	68
7.6	Desenvolvimento.....	70
7.6.1	Lista De Funcionalidades Aplicação Mobile.....	73
7.6.2	WebApp.....	76
7.6.3	App Mobile.....	78
7.6.4	Scripts.....	79
7.6.5	Previsão de Resultados Futebolísticos.....	80
7.6.6	Text-To-Speech.....	83
7.6.7	Speech-To-Text.....	84
7.7	Limitações do Chat.....	85
7.8	Resultados.....	85
7.9	Conclusão.....	86
8	Football Trivia.....	87
8.1	Análise das várias Contas X.....	87
8.2	Análise de Requisitos.....	88
8.2.1	Requisitos Funcionais.....	88
8.2.2	Requisitos Não Funcionais.....	88
8.3	Diagramas.....	89
8.4	Arquitetura.....	91
8.5	Desafios.....	97
8.5.1	Geração de Imagens com GenAI.....	103

8.6	Conclusão.....	105
9	Football Quiz	107
9.1	Análise de Requisitos	107
9.1.1	Requisitos Funcionais	107
9.1.2	Requisitos Não Funcionais.....	108
9.2	Diagramas	108
9.3	Arquitetura	109
9.4	Desenvolvimento	111
9.5	Resultados	113
9.6	Problemas.....	114
9.7	Conclusão.....	115
10	Conclusões e Trabalhos Futuros.....	117
	Referências bibliográficas	119
	Anexos e/ou Apêndices	I
A.	Proposta de Estágio	I
B.	Comparação de Ferramentas e Aplicações de GenAI/ Futebol.....	V
C.	Esquema da Base de Dados Neo4J	VIII
D.	Mockups	XIV
E.	Descrição dos Mockups da Aplicação Mobile do Projeto SportGPT ..	XVII
F.	Autenticação dos utilizadores na aplicação mobile do projeto SportGPT	XIX
G.	Descrição dos Casos de Uso do SportGPT.....	XX
H.	Descrição Detalhada dos Casos de Uso do projeto Football Trivia ..	XXIII
I.	Descrição Detalhada dos Casos de Uso do projeto Football Quiz	XXVI
J.	Flow Relativo à apresentação das imagens dos jogadores	XXIX
K.	Flow relativo à apresentação das imagens das competições e das equipas	XXIX
L.	Análise das Informações apresentadas no projeto FootballQuiz	XXXII
M.	Análise das Contas da Rede Social X.....	XXXII
N.	Curiosidades e respetivas queries para o Projeto Football Trivia	XXXII
O.	Análise da Evolução do SportGPT	XXXII
P.	Previsão de Resultados Futebolísticos Machine Learning.....	XXXIII
Q.	Fontes de Dados Analisadas.....	XXXIII
R.	Registo de Interações.....	XXXIII

ÍNDICE DE FIGURAS

Figura 3-1 - Tokenização de uma frase de exemplo	16
Figura 3-2 - Tokenização de uma frase de exemplo 02	16
Figura 3-3 - Tokenização de uma frase de exemplo 03	16
Figura 4-1 - Exemplo de Base de Dados <i>Key-Value</i>	26
Figura 4-2 - Exemplo de Base de Dados <i>Document</i>	27
Figura 4-3 - Exemplo de Base de Dados <i>Graph</i>	28
Figura 4-4 - Taxonomia das técnicas de mitigação de Alucinação dos LLMs[106]	45
Figura 7-1 - Arquitetura do SportGPT.....	59
Figura 7-2 - Diagrama de Casos de Uso.....	69
Figura 7-3 - Diagrama de Sequência	69
Figura 7-4 - Exemplo de Avatar	76
Figura 7-5 - <i>Web App</i> SportGPT	76
Figura 7-6 - Geração da Resposta	77
Figura 7-7 - Apresentação da resposta.....	77
Figura 7-8 - Implementação da Funcionalidade <i>Text-to-Speech</i>	84
Figura 7-9 - Implementação da Funcionalidade <i>Speech-to-Text</i>	84
Figura 8-1 - Diagrama de Casos de Uso Football Trivia.....	89
Figura 8-2 - Diagrama de Sequência Football Trivia	90
Figura 8-3 - Arquitetura do Projeto Football Trivia	91
Figura 8-4 - <i>Queries</i> com Ids iguais.....	92
Figura 8-5 - UI da web page do Fotball Trivia	93
Figura 8-6 - Introdução do nome do Jogador	94
Figura 8-7 - Introdução do nome da equipa	94
Figura 8-8 - <i>Query</i> relativa a equipas com o mesmo nome	95
Figura 8-9 - Localização das equipas.....	95
Figura 8-10 - Escolha da região da equipa	95
Figura 8-11 - Continentes a que pertencem as competições.....	96
Figura 8-12 - Regiões onde as Competições ocorrem	96
Figura 8-13 - Código de frontend relativo à dropdown das competições.....	97

Figura 8-14 - Output das curiosidades das temporadas, num cenário de múltiplas chamadas ao LLM	99
Figura 8-15 - Output das curiosidades das temporadas, num cenário de uma só chamada ao LLM.....	99
Figura 8-16 - Exemplos de imagens geradas pelo DALL-E 2: a) e b) acerca de Cristiano Ronaldo; c) Jogadores do Chelsea	103
Figura 8-17 - Exemplos de imagens geradas pelo DALL-E 3: a) Resultados dos último jogos do Chelsea e b) Últimos golos de Lionel Messi; c) Últimos golos de Cristiano Ronaldo.....	104
Figura 8-18 - Chamada ao DALL-E para geração de imagens.....	104
Figura 9-1 - Diagrama de Casos de Uso do Football Quiz.....	108
Figura 9-2 - Diagrama de Sequência do Football Quiz	109
Figura 9-3 - 1º Momento da Arquitetura do Football Quiz.....	110
Figura 9-4 - 2º Momento da Arquitetura do Football Quiz.....	111
Figura 11-1 - <i>Mockups</i> da <i>App</i> SportGPT: a) Início da aplicação; b) Ecrã de Registo; c) Ecrã de Registo com dados de exemplo d) Ecrã de Escolha da Equipa favorita do Utilizador e) Ecrã indicativo da Equipa Escolhida pelo Utilizador, com recurso à barra de pesquisa	XIV
Figura 11-2 - <i>Mockups</i> da <i>App</i> SportGPT: f) Ecrã de Login; g) Ecrã de Login cujos dados inseridos demonstram ser incorretos; h) Ecrã Principal da Aplicação após Registo/ Login bem sucedidos; i) Ecrã dedicado às predictions de resultados futebolísticos	XV
Figura 11-3 - <i>Mockups</i> da <i>App</i> SportGPT: j) Ecrã Inicial do Chat; k) Ecrã do <i>Chat</i> após se iniciar a escrita de uma questão futebolística h) Ecrã de <i>Chat</i> ao longo da conversação	XVI
Figura 11-4 - <i>Endpoint</i> responsável pelas curiosidades dos Jogadores	XXIX
Figura 11-5 - Output das curiosidades do jogador juntamente com a imagem	XXIX
Figura 11-6 - <i>Endpoint</i> responsável pelas curiosidades das Equipas.....	XXX
Figura 11-7 - <i>Output</i> das curiosidades das equipas com logótipo desta	XXX
Figura 11-8 - <i>Endpoint</i> responsável pelas curiosidades das Competições	XXX
Figura 11-9 - <i>Output</i> das curiosidades das Competições com imagem	XXXI

ÍNDICE DE TABELAS

Tabela 2-1 - Plano de Trabalhos.....	3
Tabela 3-1 - Comparação entre IA e GenAI	8
Tabela 3-2 - Tabela Comparativa de LLMs.....	13
Tabela 3-3 - Estudo da Tokenização por Idioma.....	14
Tabela 3-4 - Comparação dos Tipos de Tokenização.....	15
Tabela 4-1 - Descrição das Transações ACID.....	23
Tabela 4-2 - Características da base de dados do tipo Chave-Valor	26
Tabela 4-3 - Características de bases de dados do tipo Documento.....	28
Tabela 4-4 - Características de bases de dados do tipo Grafo	29
Tabela 4-5 - Características de bases de dados do tipo Column	30
Tabela 4-6 - Tabela Comparativa entre bases de Dados de Grafos.....	31
Tabela 4-7 -Cláusulas da Linguagem Cypher	33
Tabela 4-8 - Tabela Comparativa de GPTs.....	38
Tabela 4-9- Comparação de GPT 4 com o GPT 4-Turbo.....	40
Tabela 5-1 - Tabela de Ferramentas e Linguagens Utilizadas	51
Tabela 7-1 - Descrição dos Papéis envolvidos.....	60
Tabela 7-2 - Resultados Obtidos na Framework Llama Index.....	64
Tabela 7-3 - Resultados obtidos da framework LangChain	66
Tabela 7-5 - Descrição das Variáveis presentes no dataset	80
Tabela 7-6 - Resultados do primeiro teste.....	82
Tabela 8-1 - Temáticas das Curiosidades da Rede Social X.....	87
Tabela 8-2 - Temáticas das Queries construídas	91
Tabela 8-3 - Tempos de Execução do fluxo	97
Tabela 8-4 - Tempos de execução do fluxo com apenas 1 chamada ao LLM.....	98
Tabela 8-5 - Tempo de execução de queries das Competições	100
Tabela 8-6 - Tempo de execução de queries das Temporadas	100
Tabela 8-7 - Tempo de execução de queries dos Jogadores	101
Tabela 8-8 - Tempo de execução de queries das Equipas.....	102
Tabela 8-9 - Tempo de execução do fluxo com chamadas concorrentes ao LLM	102

Tabela 8-10 - Tempos de Execução com utilização de Concorrência.....	103
Tabela 9-1 - Utilização dos LLMs	112
Tabela 9-2 - Estudo da Veracidade das Informações Apresentadas no Quiz	113
Tabela 9-3 - Análise das Opções apresentadas nas Questões do Quiz	114
Tabela 7-4 - Armazenamento dos Dados do Utilizador.....	XIX

LISTA DE ACRÓNIMOS E SIGLAS

ACID	<i>Atomicity, Consistency, Isolation, Durability</i>
ANSI	<i>American National Standards Institute</i>
API	<i>Application Programming Interface</i>
BSON	<i>Binary JSON</i>
CPU	<i>Central Processing Unit</i>
CRUD	<i>Create, Read, Update, Delete</i>
CSV	<i>Comma-separated values</i>
DNS	<i>Domain Name System</i>
EUA	Estados Unidos da América
GANs	<i>Generative Adversarial Network</i>
GENAI	<i>Generative AI</i>
GPT	<i>Generative Pre-trained Transformer</i>
GQL	<i>Graph Query Language</i>
HTML	<i>HyperText Markup Language</i>
IA	Inteligência Artificial
IBM	<i>International Business Machines Corporation</i>
ISO	<i>International Organization for Standardization</i>
JSON	<i>JavaScript Object Notation</i>
KNN	<i>K-Nearest Neighbors</i>
LaMDA	<i>Language Model for Dialogue Applications</i>
LLM	<i>Large Language Model</i>
MAE	<i>Mean Absolute Error</i>
MB	<i>Megabyte</i>
ML	<i>Machine Learning</i>
MSE	<i>Mean Squared Error</i>
PDF	<i>Portable Document Format</i>
PLN	Processamento de Linguagem Natural
QA	<i>Question Answering</i>
RAM	<i>Random Access Memory</i>
RAG	<i>Retrieval-augmented generation</i>
RDBMS	<i>Relational Database Management System</i>
RLHF	<i>Reinforcement Learning with Human Feedback</i>
RMSE	<i>Root Mean Squared Error</i>
SGBD	Sistema de Gestão de Base de Dados Relacional
SQL	<i>Structured Query Language</i>
SVM	<i>Support Vector Machine</i>
TTS	<i>Text-to-Speech</i>
UEFA	<i>Union of European Football Associations</i>
UUID	<i>Universally Unique Identifier</i>
VAR	<i>Video Assistant Referee</i>
VAE	<i>Variational Autoencoder</i>
XML	<i>Extensible Markup Language</i>

1 INTRODUÇÃO

O futebol, originário no século XIX na Inglaterra, demonstra ser um dos desportos mais populares ao redor do mundo, contando com uma rica e vasta história ao longo dos anos [1]. Não deve ser considerado apenas um jogo em campo, onde 2 equipas de 11 jogadores cada, competem para marcar golo na baliza da equipa adversária. Este desporto tem a capacidade de transcender e ultrapassar barreiras e fronteiras não só culturais como também geográficas. Não é apenas um desporto, mas sim uma manifestação tanto cultural como social que interliga pessoas e culturas. O futebol tem a capacidade de promover a interação humana, não só dentro do próprio campo como fora dele. Não se limita meramente à competição desportiva, mas também propicia a construção de identidades culturais e nacionais. As rivalidades entre clubes, nações e seleções estão bastante enraizadas ao longo de décadas, mas esta picardia demonstra ser saudável, e associada a patriotismo e sentimento de identidade [2]. No fundo, deixa de ser somente um desporto de destreza física, mas uma interligação de identidades: um fenómeno sociocultural de relevância incontestável. Os adeptos do futebol são bilhões ao redor do mundo, e consomem todos os tipos de informação para estarem atualizados [3]. Nos últimos anos tem-se observado uma crescente propensão para o desenvolvimento de aplicações desportivas nas casas de apostas, promovendo conseqüentemente o elevado consumo de dados futebolísticos por parte de fãs e entusiastas.

Noutra vertente encontra-se presente o ChatGPT [4], um modelo de linguagem lançado pela OpenAI [5] no final de 2022. De forma célere, ganhou popularidade, 2 meses após o seu lançamento já contava com mais de 100 Milhões de utilizadores. O ChatGPT foi desenvolvido para ser uma ferramenta de processamento de linguagem natural, destacando-se pela incrível capacidade de compreender e gerar texto em linguagem semelhante à humana através de modelos GPT (*Generative Pre-Trained-Transformer*). Consegue interpretar com precisão a semântica de frases e perguntas, fornecendo respostas coerentes e contextuais consoante o texto / questão introduzida. Ao ser treinado com milhões de informações de diversos âmbitos, tem a aptidão para responder às mais variadas perguntas sobre as mais variadas temáticas. A versatilidade é comprovada ao conseguir desempenhar inúmeras tarefas, por exemplo: gerar texto, traduzir idiomas ou mesmo formular código em múltiplas linguagens de programação[6]. Esta ferramenta demonstra ser deveras revolucionária, acendendo as luzes da ribalta para os "LLMs" (*Large Language Models*) praticamente desconhecidos até então, impulsionando a pesquisa e os desenvolvimentos nesta área [7].

Entende-se que a combinação destes 2 mundos é vantajosa: por um lado encontra-se o futebol com fãs interessados em consumir várias informações e por outro o ChatGPT com a sua capacidade de gerar conteúdo textual de forma sagaz e

original. Desta forma é possível antever uma proposta promissora ao unir as 2 vertentes: desenvolver uma ferramenta de GenAI (Generative AI) com capacidade de processamento de dados históricos futebolísticos. Para contornar a problemática dos LLMs, como o GPT, a nível da ausência de dados atuais e alucinações, optou-se pelo uso de uma base de conhecimentos interna contendo informações de futebol. Para complementar esta ideia e apresentá-la aos utilizadores foi desenvolvida uma aplicação *mobile*, para a realização de questões relacionadas com futebol e receber respostas em tempo real através da utilização de PLN (Processamento de Linguagem Natural) e LLM (um modelo GPT). O modelo terá acesso tanto à dúvida introduzida pelo utilizador como informações relativas ao esquema da base de dados (descrições e particularidades). Com base nestas informações será gerada uma *Cypher Query* (linguagem de consulta utilizada pela base de dados Neo4J) a executar na base de dados de grafos utilizada. O retorno obtido será alvo de uma nova chamada ao LLM, para torná-la mais humanizada e atrativa aos olhos do utilizador, promovendo uma conversação mais natural e fluída, onde as respostas dadas deverão ser coerentes, diretas e precisas.

Este relatório encontra-se dividido em 10 capítulos. No capítulo 2, é descrito o plano de trabalhos traçado ao longo do estágio curricular. O capítulo 3 aborda conceitos fundamentais necessários para o entendimento do relatório, enquanto o capítulo 4 apresenta o estado da arte, reunindo as aplicações e ferramentas estudadas. Nos capítulos 5 e 6, são descritas, respetivamente, as tecnologias utilizadas no estágio e a metodologia seguida. Os capítulos 7, 8 e 9 detalham o desenvolvimento das três aplicações desenvolvidas. Por fim o capítulo 10 apresenta as conclusões finais e as perspetivas de trabalho futuro.

2 PLANO DE TRABALHOS

Neste capítulo será apresentado o plano de trabalhos desenvolvido ao longo do estágio desde a fase de pesquisa e estado da arte, desenvolvimento, otimização e refinamento. Na Tabela 2-1 é descrito de forma detalhada o plano de trabalhos.

Tabela 2-1 - Plano de Trabalhos

	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul
Tarefas											
T1	■	■									
T2		■	■								
T3			■	■	■	■	■	■	■		
T4					■	■	■				
T5					■	■	■				
T6							■	■	■		
T7									■	■	
T8								■	■	■	
T9									■	■	
T10										■	■
T11		■	■	■	■	■	■	■	■	■	■

T1 – Estado da Arte e Tecnologia – pesquisa acerca da área de inteligência artificial e ferramentas e tecnologias a utilizar no projeto. Assim bem, como estudo do mercado para entender aplicações e ferramentas semelhantes já existentes e como nos posicionamos no âmbito;

T2 – Investigação de fontes de dados futebolísticas para enriquecimento dos dados já detidos pela empresa;

T3 – Implementação do Servidor Aplicacional – desenho da solução servidor e a sua implementação. Inclui uma *interface* GPT para receber *prompts* do utilizador. Deve ter a capacidade de gerar conteúdo textual em tempo real acerca de futebol;

T4 – Procura e desenvolvimento de novas funcionalidades para tornar a *app* mais atrativa e inovadora, de forma que se imponha e diferencie no mercado;

T5 – Realização de previsão de resultados de jogos, com *Machine Learning*;

T6 – Implementação da Aplicação *Mobile* em Flutter com uma *interface* de conversação, integrada com o servidor aplicacional, para demonstração das capacidades do GPT do futebol;

T7 – Otimização e refinamento do servidor e aplicação;

T8 – Análise exaustiva de contas da rede social X a nível de curiosidades futebolísticas;

T9 – Construção do servidor da página *web* de curiosidades futebolísticas;

T10 – Construção de um *Quiz* de Futebol;

T11 – Procura de informação e escrita do relatório final.

Ao longo do estágio foram efetuadas algumas mudanças ao plano inicial de trabalhos. Durante o primeiro mês de desenvolvimento do servidor da aplicação futebolística, novembro, foi necessário dedicar algum tempo a nível de introdução dos dados na base de dados Neo4J: construção de *scripts* e o tempo necessário para a inserção. O desenvolvimento do código do servidor começou somente na última semana desse mês. Entre o final do mês janeiro e o início de março, o desenvolvimento do servidor foi temporariamente suspenso para dar lugar a estudos da integração de *frameworks* no projeto: Llama Index [8] e LangChain [9] assim bem como para realização de previsões de resultados de jogos de futebol através da utilização de ML (*Machine Learning*). Durante esses meses foram realizadas apenas pequenas alterações no servidor como: correção de *flow* e implementação de funcionalidades de interação por voz (na realização de perguntas e receção de respostas). No plano de trabalhos inicial era esperado que em fevereiro fosse iniciada a implementação da aplicação *mobile*, no entanto, devido à pausa no servidor esta etapa foi adiada para o mês seguinte. No início de março foi realizada uma reunião com uma *designer* da WIT Software com objetivo de lhe apresentar o conjunto de funcionalidades a serem implementadas na aplicação, para a construção de *mockups*. No início desse mês houve alguns problemas a nível do *budget* associado à conta da OpenAI, durante esse período foi desenvolvida uma *app mobile* em Flutter, apenas com as funcionalidades básicas de *chat*, com fim de estudar a linguagem e para utilização em futuras demonstrações. O *feedback* relativo à construção dos *mockups* só foi obtido em abril, momento a partir do qual foi iniciado o desenvolvimento da aplicação oficial e final do projeto SportGPT. Nos últimos dias do mês de abril foi tomada a decisão de reestruturar esse projeto inicial, sendo necessário redefinir o plano de trabalhos para o novo desenvolvimento: as curiosidades futebolísticas. Para além deste segundo projeto ainda foi requerida a construção de um terceiro, o Football Quiz.

3 CONCEITOS FUNDAMENTAIS

Neste capítulo serão apresentados os conceitos fundamentais que apoiam o desenvolvimento do projeto. Esta base teórica é crucial para entender os conceitos explicados ao longo de todo o documento.

3.1 Generative AI

A evolução da IA trouxe consigo uma enorme revolução: a Generative AI [10], [11]. Este “ramo” da IA tem habilidade de conceder aos sistemas a capacidade de geração de conteúdo criativo como: imagens, áudios e texto. Os modelos mais conhecidos atualmente são: o GPT, o Bard/ Gemini e o DALL-E. Neste documento foi realizada uma análise e um estudo acerca do desenvolvimento do modelo GPT, o modelo utilizado no projeto SportGPT.

O primeiro projeto de GenAI conhecido, foi desenvolvido em 1960 por Joseph Weizenbaum: o *chatbot* ELIZA [12]. No entanto, constatou-se que os primeiros projetos que utilizavam esta tecnologia fundamentavam-se em regras, limitando a sua habilidade. Este obstáculo devia-se a um léxico restrito, escassez de contexto, e excessiva dependência a padrões, tornando os modelos difíceis de personalizar e escalar. Somente em 2014, com os avanços na área de GANs (*Generative Adversarial Network*) [13], estes modelos e tecnologias ganharam destaque e notoriedade. Após este marco os modelos de Deep Learning, possibilitaram a aprendizagem automática dos modelos, análise de textos, classificação de imagens ou mesmo transcrição de áudio. Os modelos de GenAI têm capacidade de combinar vários algoritmos de IA para gerar e processar conteúdo, como técnicas de PLN para gerar texto, pois é necessário transformar letras, palavras e pontuação em classes, entidades e ações representáveis por meio de vetores. De seguida, são utilizadas redes neuronais para gerar novos conteúdos como resposta a *prompts*. Estes podem ser em forma de texto, vídeo, notas musicais, ou quaisquer entradas que o sistema consiga processar [14]. As redes neuronais revelam-se adequadas a todo o processo dada a sua projeção para imitar o funcionar do cérebro humano, e a capacidade de aprender regras e encontrar padrões nos conjuntos de dados fornecidos. Desta forma, a GenAI é reconhecida pela sua grande capacidade de interpretar conteúdo já existente e através deste gerar novo conteúdo, fornecer respostas mais assertivas, criar representações de veras realistas do ser humano ou resumir e traduzir enormes textos.

Dada a enorme versatilidade, esta ferramenta pode ser utilizada em várias profissões para agilizar trabalhos [15]. Por exemplo, as finanças conseguem deter mais atenção nas transações para traçar um histórico de cada indivíduo e assim construir sistemas mais robustos a nível de deteção de fraude. Em advocacia a ferramenta pode auxiliar no elaborar e na interpretação de contratos, documentos e evidências, acelerando todo o processo envolvido. A nível fabril, existe a

possibilidade de desenvolver sistemas com câmaras de *raio-x* para distinguir peças defeituosas, as possíveis causas e reduzir custos associados [14].

Dispersas pela Internet e acessíveis a todos encontramos várias ferramentas de GenAI [16]:

- Para geração de texto: ChatGPT [4], [17] Jasper [17], Gemini [18];
- Para geração de imagens: DALL-E [19], Stable Diffusion [20];
- Para geração de música: Musenet [21];
- Para geração de código: OpenAI Codex [22], Github Copilot [23];
- Para síntese de voz: Listnr [24], Podcast.ai [25].

Apesar desta avultada notoriedade, a tecnologia também acarreta preocupações. Um dos problemas que mais tem preocupado a sociedade são as *deepfakes*: forma eficaz e traiçoeira de distorcer a realidade [26]. É possível inserir em vídeos, pessoas e discursos que nunca proferiram, ou substituir/ adicionar elementos em fotografias, aparentando falsas situações. Geralmente, as informações apresentadas são tão realistas que se torna extremamente árduo detetar imprecisões ou preconceitos e avaliar o viés relativamente a dados reais, pois a manipulação dos dados é muito sofisticada. Outro entrave é utilização abusiva e indevida destas tecnologias, pois pode resultar em novas formas de plágio que ignoram criadores de conteúdo, artistas, ou podem mesmo ferir suscetibilidades com informações incorretas e discriminatórias. As questões éticas relativas à GenAI sempre existiram, desde o início e continuarão em estudo no futuro. Um exemplo deste inconveniente ocorreu em 2019 com o *chatbot* Tay lançado pela Microsoft Corporation no Twitter. Após 16h da divulgação, o *chatbot* foi desativado, uma vez que publicou *tweets* ofensivos na plataforma, sendo substituído posteriormente pelo Zo [27]. A incrível criação de conteúdos criativos também provoca uma atitude passiva e conformista dos utilizadores, pois estes tendem a aceitar as informações como verídicas sem apurar a veracidade e origem. No entanto, esta capacidade de produção rápida de conteúdos pode colocar em risco várias profissões criativas.

Ainda que o auxílio e desenvolvimentos que a IA (Inteligência Artificial) nos proporciona ser inegável, é reconhecido que estes também originam preocupações e restrições de diversas naturezas [28]. Os algoritmos de IA são treinados com milhões de dados para conseguirem executar as tarefas para as quais foram desenvolvidos com precisão e eficácia. Em certa parte, pode ser considerado uma vantagem, dado o poder de adaptabilidade a qualquer área. Mas também tem uma potencial adversidade: as informações utilizadas durante o treino. Os algoritmos podem refletir preconceitos e discriminações presentes nesses dados. Os *chatbots*, ao serem alimentados por IA, podem disseminar a desinformação e propagar essas rejeições. Ao ter a capacidade de processar enormes volumes de dados, como informações de redes sociais, existe a possibilidade da privacidade e a segurança dos dados serem postas em causa ao serem empregues sem qualquer autorização. A

OpenAI, utilizou dados de artigos de *blog*, *reviews* de comentários em *sites*, para o treino dos algoritmos. Apesar de se encontrarem públicas, estas informações foram utilizadas sem consentimento dos próprios utilizadores. Desta forma torna-se fundamental garantir a proteção de dados e direitos humanos a par dessa inovação, desenvolvendo regulamentações claras e diretrizes para reger estas implementações e promover o uso responsável da tecnologia. Os algoritmos de IA enfrentam uma barreira relativa à sua capacidade de extrapolar conhecimento além do âmbito para o qual foram treinados, tornando-os ineficientes no executar de funções em cenários imprevistos.

Atualmente um assunto que tem preocupado consideravelmente a população é o impacto da GenAI no mercado de trabalho [29]. À medida que os modelos e algoritmos são aperfeiçoados, tarefas outrora realizadas por humanos, estão a ser automatizadas por máquinas, promovendo o despedimento em massa. Como se sabe, a utilização de robôs para a realização de tarefas repetitivas de forma eficiente e económica já é uma realidade em múltiplos setores. Torna-se relevante ressaltar que apesar do contínuo desenvolvimento de “máquinas inteligentes”, estas nunca terão a capacidade de substituir a 100% o ser humano. Este consegue colocar compaixão e criatividade nas suas soluções de uma forma única e intransponível. A nível de empresas a utilização de IA pode implicar custos adicionais como: utilização de infraestruturas, manutenção, adaptação e possivelmente profissionais qualificados. Os economistas do Goldman Sachs, uma instituição financeira, lançaram um relatório em março de 2023 onde abordaram o cenário preocupante da ascensão da IA a par do desemprego. Os economistas estimam que 300 Milhões de empregos, cerca de 18% do trabalho global, poderão ser informatizados e automatizados nos próximos anos [30]. Até ao momento, em toda a Europa e nos EUA (Estados Unidos da América), cerca de 2/3 dos empregos atuais já possuem algum tipo de automação e 1/4 pode ser completamente realizado por IA [31].

A opacidade dos modelos de GenAI e IA pode representar um grande desafio em áreas e profissões com tarefas críticas (como a medicina), pois em situações de erros torna-se difícil saber sobre quem recai a responsabilidade. Assim é fundamental desenvolver e aprimorar os modelos para que sejam mais transparentes e forneçam *insights* e informações acerca do caminho percorrido até à obtenção dos resultados.

Para auxiliar e assegurar uma boa utilização de modelos de GenAI é importante adotar e seguir boas práticas que tenham em conta a precisão, eficácia e transparência. Uma abordagem passa por classificar de forma clara o conteúdo, verificar a qualidade das informações geradas através de ferramentas externas ou aprender os pontos fortes e fracos da tecnologia. Nos próximos anos ambiciona-se que a tecnologia continue a crescer, proporcionando significativos avanços nas mais variadas áreas: tradução, criação de novos conteúdos (texto, vídeos, música, *design*), ou mesmo descoberta de novos medicamentos e tratamentos [32].

Para uma melhor compreensão entre a temática da IA e de GenAI, foi realizada uma comparação para compreender as principais semelhanças e diferenças.

A evolução da IA tem desempenhado um papel fundamental na inovação de múltiplos setores como a área da saúde, *marketing* ou agricultura, promovendo avanços tecnológicos significativos e descobertas surpreendentes. No entanto, é importante reconhecer as diferentes formas em que a IA se manifesta à medida que evolui.

A Inteligência Artificial dita “tradicional” caracteriza-se pela capacidade de aprender e tomar decisões com base nos dados, procurando responder a um conjunto específico de entradas de acordo com regras, como referido por Anantha Sharma em [33]. Por exemplo, num jogo de xadrez entre um humano e um computador, a máquina, tem conhecimento de todas as regras, realiza os próprios movimentos e prevê os do humano de acordo com uma estratégia pré-definida. Isto é, o sistema não cria nem “inventa” novas formas de jogar o jogo, apenas seleciona as estratégias com as quais foi programado. A IA também é amplamente empregue em sistemas de recomendação, por exemplo, a nível de músicas, onde o sistema realiza uma análise exaustiva ao histórico de visualizações do utilizador conjuntamente com outros dados relativos às suas preferências. De acordo com essas informações, recomenda músicas que vão de encontro às inclinações de cada pessoa. O modelo aprende com padrões de visualizações passadas, e adapta as recomendações consoante as interações contínuas. Se o utilizador ouviu ou pesquisou recentemente por música pop o sistema irá sugerir mais músicas desse estilo.

A GenAI, é uma forma de manifestação de IA, que produz conteúdos criativos através de conteúdos já existentes [10], [11]. Imaginemos que um aluno precisa de escrever uma fábula, e já tem ideia da lição que quer transparecer, e dos animais que quer colocar na história, mas está com bastante dificuldade em escrever. A GenAI, de acordo com todas as informações que o aluno detém, possui a capacidade de o auxiliar na escrita da composição. Este é apenas um dos muitos exemplos da sua utilização, pois ao ser treinada com enormes volumes de dados consegue aprender os padrões subjacentes nos dados e gerar novas conteúdos: imagens, músicas ou textos.

A principal diferença entre estas 2 vertentes passa pelas suas aptidões e aplicações, a IA tradicional destaca-se em tarefas específicas como: análise de dados ou realização de previsões. A GenAI, por sua vez, é mais propensa para a criação de conteúdos criativos. Na Tabela 3-1 é estabelecido um paralelismo entre as 2 tecnologias.

Tabela 3-1 - Comparação entre IA e GenAI

	<i>GenAI</i>	<i>Inteligência Artificial</i>
<i>Principal Objetivo</i>	Criação de conteúdo original, respostas a questões, protótipos, músicas, textos e imagens	Automatizar tarefas, análise de dados, reconhecimento de padrões de acordo com um conjunto de regras

Desenvolvimento de um protótipo de um Futebol GPT

<i>Treino</i>	Redes Neurais e <i>Transformers</i> (modelo de <i>Deep Learning</i> que revolucionou o campo do PLN) [34]	Redes neurais convolucionais, para processamento de sequências e redes recorrentes e aprendizagem supervisionada, não supervisionada e por reforço
<i>Tomada de Decisão</i>	Criação e tomada de decisão de forma inovadora e probabilística	Determinada por parâmetros pré-definidos
<i>Geração de conteúdos</i>	Geração de novos conteúdos	Automatização de tarefas
<i>Dados de Treino</i>	Dados estruturados e não estruturados	Estruturados e rotulados
<i>Volume de dados</i>	Enormes volumes de dados	Grandes quantidades de dados, mas inferior que em GenAI
<i>Fontes de Dados</i>	Dados recolhidos de várias fontes <i>online</i>	Dados recolhidos de fontes privadas e confiáveis
<i>Modo de Operação</i>	Através de um <i>prompt</i> , permite que um utilizador passe informações que auxiliem na geração de conteúdo	Segue um conjunto predefinido de regras para processar dados e produzir resultado
<i>Aplicações</i>	Âmbitos criativos e de resolução de problemas: tarefas de PLN, geração de conteúdos e sistemas de recomendação	Âmbitos de análise e previsões: Processamento de dados baseados em regras e resultados predeterminados

3.2 Generative Adversarial Network

As *Generative Adversarial Network* (GANs), foram descobertas em 2014 por Ian Goodfellow e pela sua equipa e são as metodologias mais utilizadas na área de GenAI [35]. Estas redes são compostas por duas redes neurais “rivais” (que competem entre si, onde o ganho de uma implica a perda do outro), um gerador (de novos exemplos e amostras) e um discriminador (capaz de aprender a distinguir o teor gerado, do real). As duas redes são treinadas em conjunto, e à medida que o gerador gera novos conteúdos (imagens, sons ou texto a partir de um espaço de entrada aleatório), o discriminador aperfeiçoa a distinção da veracidade dos dados gerados. Enquanto o treino decorre, o gerador e o discriminador ficam cada vez mais hábeis.

O treino ocorre em 2 etapas[36], [37]: o Treino do discriminador, onde este é treinado com conjuntos de dados reais e falsificados (gerados pelo gerador)

aprendendo a distingui-los. E o Treino do Gerador: este tem com objetivo enganar o discriminador, ajustando os seus pesos para produzir dados que pareçam o mais reais possível e consequentemente que o discriminador os classifique como reais. Este mecanismo é executado de forma repetitiva, onde a cada iteração os modelos são refinados continuamente até que o conteúdo produzido seja indiscriminável do já existente. Pode-se estabelecer um paralelismo entre as GANs e o processo de escrita literária, onde o escritor representa o gerador e cria histórias. O crítico literário corresponde ao discriminador, e ao deter entendimento acerca dos vários elementos que compõem uma narrativa, avalia a qualidade destas. Na “fase de treino”, o discriminador é exposto tanto a histórias de elevada qualidade como de baixa qualidade, com fim a aprender os padrões de escrita. Por sua vez, o escritor, de acordo com as críticas efetuadas pelo discriminador, ajusta os enredos, tornando-os mais coerentes e alinhados com os padrões.

No entanto, há a realçar que apesar das amostras concebidas pelas GANs serem de alta qualidade e a velocidade de geração ser alta, a diversidade demonstra ser insuficiente, revelando serem mais vantajosas para produção de informações em domínios específicos.

Para além das GANs existem outros modelos generativos com capacidade de formar poderosos modelos[10], [11]:

Modelos de difusão – também conhecidos por modelos probabilísticos de difusão com eliminação de ruído, são comumente utilizados a nível de processamento de imagem e visão computacional para remoção de interferências. Este modelo é composto por duas etapas: a difusão direta e a reversa. A 1.^a etapa pressupõe a adição de ruído gaussiano a amostras tornando-as ruidosas. A 2.^a etapa, parte dessa amostra ruidosa e tenta reverter o ruído, procurando recriar a amostra original. Desta forma novos dados podem ser gerados através do processo de remoção de ruído. Pode fazer-se uma analogia entre este modelo e uma fotografia de uma paisagem tirada num dia de nevoeiro onde as condições atmosféricas provocam vários pontos de ruído na imagem: variações de intensidade de brilho ou de cor que diferem da imagem real. A remoção de ruído pode melhorar a qualidade da imagem através da utilização de um modelo de difusão. O modelo iria assumir que a imagem introduzida seria a versão ruidosa da imagem verdadeira. Durante o processo de difusão, o ruído da imagem seria gradualmente reduzido, produzindo uma imagem “*denoised*” mais precisa e esteticamente agradável, onde os pontos de ruído foram gradualmente eliminados. Contudo, apesar de ser possível treinar uma quantidade infinita de camadas, leva mais tempo a treinar do que os modelos VAE (*Variational Autoencoder*), pois o processo de *reverse sampling* é lento [38].

VAEs - *Variational Autoencoders* - consistem em 2 redes neuronais denominadas *encoder* e *decoder*. Ao receber uma entrada, o *encoder* converte-a numa representação menor. Esta preserva as informações mais relevantes dos dados, permitindo que o *decoder* reconstrua os dados de entrada originais, descartando em simultâneo a

informação irrelevante. As 2 redes trabalham em simultâneo com fim a aprender uma representação simples dos dados. Apesar de gerarem imagens rapidamente, estas não são tão detalhadas como os modelos de difusão [38]. É possível assemelhar este modelo ao famoso jogo do “*pictionary*”: um jogador (*encoder*) tenta capturar o máximo de informações relevantes da palavra a ser adivinhada concentrando-as num desenho. Outro jogador (*decoder*) com base nesse mesmo desenho interpreta-o e tenta deduzir a palavra que representa.

3.3 Large Language Models (LLMs)

Large Language Models são modelos de IA desenvolvidos com intuito de assimilar e processar linguagem natural. Estes modelos, baseados em *Transformers* [39], são treinados com enormes quantidades de dados de texto para adquirir conhecimentos do mundo e conseqüentemente executar várias tarefas, tal como: relações e extração de padrões em textos. Os modelos são especializados no processamento e geração de textos, muitas vezes indistinguíveis daqueles redigidos por seres humanos. Desta forma, demonstram-se úteis na escrita, tradução e resumo de textos. Neste processo, não pode ser considerada a presença de inteligência, trata-se essencialmente de cálculos matemáticos e probabilísticos executados em alta velocidade, com objetivo de prever a próxima palavra numa sequência. Têm aplicação em diversos cenários: *chatbots*, tradução automática, assistentes virtuais ou mesmo criação de conteúdos. Os LLMs são um marco significativo na área da IA, permitindo a existência de interações mais naturais entre o ser humano e máquinas. Encontram-se em contínua evolução para que sejam fornecidas soluções mais sofisticadas e inteligentes [40].

Os modelos mais conhecidos atualmente são o DALL-E, o ChatGPT e o Bard (agora Gemini). Os 3 modelos têm capacidade de gerar textos, código, realizar traduções, no fundo, criar conteúdos criativos. Mas apesar de realizarem tarefas semelhantes, têm as suas próprias características, pontos fortes e fracos.

DALL-E – modelo criado pela OpenAI em janeiro de 2021, para gerar imagens criativas a partir de *prompts* de texto [19]. O nome do modelo é deveras peculiar, pois é composto por “WALL-E”, a personagem da Pixar, que representa um robô solitário que vive num mundo pós-apocalíptico com habilidade para criar objetos a partir de achados. Esta aptidão remete para a perícia do modelo de criar imagens inovadoras a partir de descrições textuais. Existe também a referência a Salvador Dalí, o famoso pintor mestre do surrealismo, reconhecido pela sua arte peculiar de extravagância e incomum, remetendo para a capacidade do modelo de criar imagens invulgares, desafiando os limites da realidade. O modelo utiliza uma versão do GPT-3 *Transformer*, com cerca de 12 bilhões de parâmetros, para realizar a interpretação de *prompts* de linguagem natural e gerar imagens. O modelo no seu treino teve por base um enorme conjunto de dados de imagens, o que lhe concedeu a capacidade de recriar objetos já existentes de forma realista, bem como criar seres, objetos e cenários totalmente irrealistas, elevando o patamar da criatividade.

GPT – O modelo GPT foi treinado com milhões de dados para entender e responder a solicitações humanas de maneira precisa, interpretando o contexto das interações para fornecer respostas relevantes. Utiliza uma arquitetura de *Transformers* útil para PLN [39]. Ficou conhecido pelo *chatbot* ChatGPT lançado pela OpenAI a 30 de novembro 2022, baseado em Deep Learning. O seu nome combina “*chat*”, o que remete para a sua capacidade de conversação, com “GPT”, sigla de *Generative Pre-Trained Transformer*, uma arquitetura de LLMs. Este *chatbot* utiliza um algoritmo baseado em redes neuronais que permite estabelecer diálogos naturais e fluídos com utilizadores através do processamento de um enorme volume de dados. O *chatbot* tem capacidade de criar e traduzir textos, responder a perguntas, gerar código entre outras tarefas criativas. Após o seu lançamento ganhou rapidamente popularidade: 1 Milhão de utilizadores em 5 dias, contando atualmente com cerca de 1,6 bilhões de entradas.

Bard / Gemini – o Bard é um modelo de linguagem lançado pela Google AI em março de 2023, baseado no modelo de linguagem LaMDA (*Language Model for Dialogue Applications*) [41], direcionado para a geração de texto [18]. Foi lançado como uma resposta ao GPT da OpenAI e tem a capacidade de gerar conteúdo criativo como: texto, traduções, código, através de *prompts* de texto.

O modelo foi treinado com uma ampla gama de dados textuais e código, assim bem como dados da *web*. Ao ter uma estrutura com base em *Transformers* [39] e redes neuronais recorrentes, revela-se útil no processamento de dados de forma sequencial como tradução de textos para um determinado idioma. Destaca-se pela sua capacidade de raciocínio complexo em diferentes tipos de dados, sendo particularmente eficiente em análise e interpretação profunda de dados. No seu treino são utilizadas técnicas de aprendizagem supervisionada e não supervisionada, e ao recorrer a dados rotulados e não rotulados aprimorou a sua capacidade de compreensão e interpretação de padrões complexos [42]. O Bard é acessível em vários países e tem o inglês como principal idioma suportado e revela-se gratuito.

Decorridos 9 meses, a 6 de dezembro de 2023 foi lançado o Google Gemini, um modelo de capacidade superior a nível de processamento de tarefas complexas, entendimento lógico e com suporte para diferentes tipos de *prompts*: texto, imagens, áudio ou vídeo. A arquitetura do modelo baseia-se em *Transformers*, proporcionando a interpretação de diferentes solicitações de forma eficaz. O Gemini distingue-se pela sua habilidade de lidar simultaneamente com grandes quantidades de dados em vários formatos, sejam textos, imagens ou áudios [42]. Através da capacidade de capturar semânticas e nuances presentes nos textos introduzidos, o modelo consegue fornecer respostas adequadas ao contexto e revela-se versátil para diferentes utilizações. É através da DeepMind [43] que o Gemini se encontra acessível ao redor do mundo, podendo a sua disponibilidade estar sujeita a restrições regionais [44].

Desenvolvimento de um protótipo de um Futebol GPT

Na Tabela 3-2 é apresentada uma comparação entre os três LLMs estudados, como forma de entender as principais diferenças e utilizações.

Tabela 3-2 - Tabela Comparativa de LLMs

LLM			
<i>Ano de Lançamento</i>	5 Janeiro,2021	30 Novembro, 2022	Março, 2023 Dezembro, 2023
<i>Empresa</i>	OpenAI	OpenAI	Google AI
<i>Principal Funcionalidade</i>	Geração de Imagens através de texto	Geração de texto	Geração de texto
<i>Pontos Fortes</i>	Gera imagens realistas e criativas	Gera texto com qualidade e em vários formatos	Gera informações com base em informações recentes
<i>Pontos Fracos</i>	Dificuldade em interpretar certas descrições textuais, ou na geração de conteúdos	Dados não atualizados em tempo real. Pode comer ambiguidades ou erros	Privacidade dos dados, inconsistências nas respostas

A escolha do LLM a utilizar está bastante dependente das funcionalidades e dos requisitos específicos do projeto em mãos. Se o objetivo for geração e apresentação de imagens inovadoras e de acordo com determinados contextos, o DALL-E poderá ser a melhor opção. No caso de ser primordial a geração de textos e conteúdos, as melhores opções serão o ChatGPT (ou API – *Application Programming Interface* - do GPT) ou o Gemini.

3.3.1 Tokenização

Os LLMs da OpenAI, também frequentemente chamados GPT, têm a capacidade de processar textos utilizando *tokens*: segmentos de texto que retratam seqüências de caracteres, que ocorrem frequentemente nos dados de treino [45]. Esta decomposição de grandes cadeias de texto em unidades inferiores, posteriormente mapeadas para reconstruir o texto original, é denominada Tokenização [46][47]. Na página da OpenAI está disponível uma ferramenta que auxilia na contabilização dos *tokens* num trecho de texto: o Tokenizer, estando otimizado para suportar diferentes naturezas de textos, em diferentes idiomas e domínios [48]. Ao fazer um pedido à API de um LLM (por exemplo, o GPT-4), o *prompt* antes de ser processado será alvo

de um processo de tokenização, onde o texto é decomposto em palavras, caracteres ou mesmo sub-palavras, (irá depender do tipo de tokenização utilizado). No caso dos modelos mais recentes do GPT, geralmente é feita uma separação em sub-palavras, onde os vocábulos são divididos em partes menores, por exemplo: a palavra “correria” iria ser dividida em 3 *tokens*: cor, rer, ia [49]. A esta contabilização poderão ainda ser adicionados *tokens* especiais que assinalem o início e o fim de um texto, permitindo que o sistema inicie e termine a sua análise a partir de pontos específicos. Cada *token* pode representar uma sub-palavra, um carácter, um espaço à esquerda ou à direita, e é mapeado consoante um número inteiro único, conforme a lista de vocabulário do próprio modelo. A posição em que o *token* ocorre também é codificada para que o modelo entenda a ordem das palavras. Estudos indicam que em média, 1 *token*, corresponde a cerca de 4 caracteres em inglês, mas é importante referir que a tokenização depende intrinsecamente do idioma utilizado [50], tal como apresentado na Tabela 3-3.

Tabela 3-3 - Estudo da Tokenização por Idioma

<i>Idioma</i>	<i>Expressão</i>	<i>Divisão dos tokens</i>	<i>Contabilização de tokens</i>
<i>Português</i>	Olá, como estás?	[“Ol”, “á”, “,”, “como”, “est”, “ás”, “?”]	7 <i>tokens</i>
<i>Inglês</i>	Hello, how are you?	[Hello, “,”, “how”, “are”, “you”, “?”]	6 <i>tokens</i>
<i>Espanhol</i>	¿Hola como estas?	[“¿”, “Hola”, “como”, “estas”, “?”]	5 <i>tokens</i>
<i>Francês</i>	Salut comment vas-tu?	[“Sal”, “ut”, “comment”, “vas”, “-t”, “u”, “?”]	7 <i>tokens</i>

Durante a fase de tokenização também pode ocorrer a extração de pontuação prescindível, caracteres especiais ou mesmo normalizar as minúsculas e maiúsculas com intuito de tornar o texto mais coeso. Através de diferentes métodos de tokenização, cada palavra pode ser dividida em diferentes *tokens*, afetando a forma como o modelo processa e compreende o texto. A palavra “corrida” pode ser dividida em 3 *tokens* se utilizarmos a tokenização por subpalavra / caracteres, mas se usarmos tokenização por palavra corresponderia a apenas 1 *token*. A tokenização é um processo bastante utilizado em ferramentas e aplicações da atualidade, tal como a nível de tradução, onde o sistema segmenta a frase original em segmentos menores (palavras) e traduz cada um para o idioma destino. Outro exemplo são os assistentes virtuais, pois estão intrinsecamente dependentes da tokenização que é realizada, uma vez que a voz do utilizador necessita de ser convertida em texto, posteriormente dividido para que o sistema possa processar e “entender” a solicitação. Pode-se extrapolar esta questão para análise de sentimentos expostos pelos utilizadores, através da identificação de semânticas positivas ou negativas: “**gosto** muito deste jogador”, “**odeio** este treinador”.

Abaixo encontram-se descritos 3 possíveis tipos de tokenização [49]:

Tokenização por palavra - o texto é dividido em unidades menores, palavras. Por exemplo: “Eu gosto de história” seria dividido da seguinte forma: [“Eu”, ”gosto”, ”de”, ”história”], esta divisão é intuitiva e reflete a estrutura sintática das frases. Esta abordagem é improfícua com palavras “desconhecidas”, sendo por isso vantajoso para idiomas como limites bastante claros de palavras como o inglês.

Tokenização por caracter – o texto é dividido em unidades inferiores: caracteres individuais. A palavra “gosto” seria dividida em: [‘g’,’o’,’s’,’t’,’ó’]. Este tipo de tokenização permite lidar com uma grande variedade de textos e novos léxicos, dado que todos os caracteres são reconhecidos pelo modelo. Esta abordagem é útil em tarefas mais complexas, de estudo granular, por exemplo: correção de textos. No entanto, gera uma enorme quantidade de *tokens* o que pode reduzir a eficiência. Este modelo é útil para sistemas que façam análise de imagens para extração de texto.

Tokenização por subpalavra – o texto é dividido em unidades menores: maiores que um caracter, mas inferiores a uma palavra, a palavra “corrida” seria dividida em [‘cor’,’rer’,’ia’]. O modelo revela-se útil em idiomas onde o significado das palavras é construído a partir de combinações de unidades menores, por exemplo, em alemão: *Kindergarten* significa jardim de infância, onde “Kinder” significa crianças e “garten” jardim. Esta abordagem permite aos modelos lidar com palavras desconhecidas, pois subdividi-las em unidades menores reconhecidas. Estas “subpalavras” fornecem mais informações do que propriamente caracteres individuais. Modelos como o GPT utilizam este tipo de tokenização para analisar os *prompts* introduzidos pelos utilizadores e entender a semântica. Na Tabela 3-4 é feita uma breve comparação entre os três tipos de tokenização [51], [52].

Tabela 3-4 - Comparação dos Tipos de Tokenização

<i>Tipo de Tokenização</i>	<i>Palavra</i>	<i>Caracter</i>	<i>Subpalavra</i>
<i>Granularidade</i>	Alta (unidades complexas)	Baixa (unidades simples)	Média (unidades intermediárias)
<i>Flexibilidade com novos léxicos</i>	Baixa	Alta	Alta
<i>Comprimento da sequência de tokens gerados</i>	Curto	Longo	Moderado

Nesta questão da tokenização é essencial mencionar que uma palavra pode ser particionada em *tokens* completamente diferentes. Conforme apresentado na Figura 3-1.

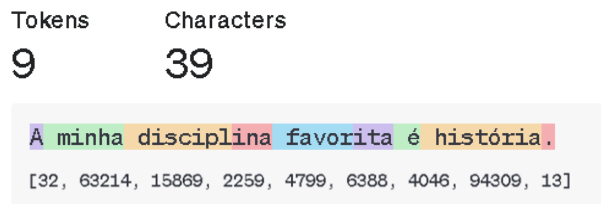


Figura 3-1 - Tokenização de uma frase de exemplo

No exemplo acima, o *token* 94309 corresponde a “história”, na Figura 3-2 observamos que ao digitar a palavra com letra maiúscula, existe alteração do *token* atribuído.

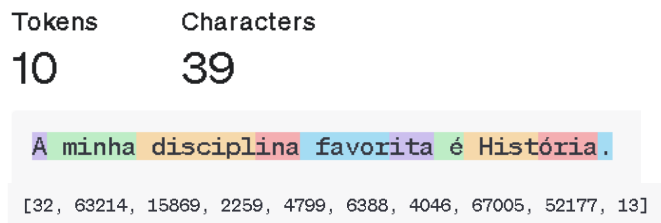


Figura 3-2 - Tokenização de uma frase de exemplo 02

Neste caso a palavra “História” é dividida em 2 *tokens* diferentes. A ocorrência da palavra no início ou no fim também pode levar a que o *token* atribuído seja diferente. Apesar de a palavra “história” aparecer com letra maiúscula, tal como na Figura 3-3, ao aparecer no início da frase faz com que ao conjunto de caracteres “Hist” seja atribuído um *token* diferente.

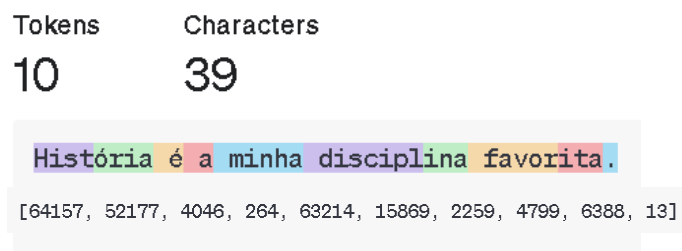


Figura 3-3 - Tokenização de uma frase de exemplo 03

No caso, os caracteres “ória” apresentam o mesmo *token*. Assim podemos comprovar que independentemente da forma e do local na frase onde a palavra ocorra, poderá ser “tokenizada” de formas diferentes e serem-lhe atribuídos *tokens* diferentes [50]. Algo notório nos exemplos acima é que existem *tokens* na ordem das dezenas, centenas e milhares. Após um estudo mais aprofundado foi possível entender que quanto mais frequente um *token* for, menor o *token* que lhe é atribuído, desta forma quanto menos frequente maior o seu número [50].

Após este aperfeiçoamento, o *prompt* é codificado por via de uma representação vetorial, no qual cada *token* é mapeado num vetor numérico utilizando técnicas como Word Embeddings [47] ou One Hot Encoders [48]. Estas representações permitem ao modelo apreender o contexto e a semântica da entrada textual. A resposta é gerada com base numa amostragem probabilística, num espaço de pesquisa onde se encontram todas as possíveis sequências de palavras de que o modelo tem conhecimento [49]. É nesta fase que o modelo gera vários *tokens* de forma sequencial, cada um com os respetivos pesos associados, com fim a dar continuidade ao texto. Os *tokens* com maior peso possuem maior probabilidade de serem escolhidos relativamente a *tokens* com menor peso. O *token* eleito é adicionado à sequência de palavras em construção, e todo o processo de procura pelo próximo elemento é repetido até que o modelo alcance o número máximo de *tokens* permitido na sua janela de contexto ou produza um *token* final que indique o término da geração de conteúdo. Após a geração da resposta, esta pode ainda ser submetida a um pós-processamento, onde existe diversificação de texto, voltar a adicionar caracteres especiais, maiúsculas e minúsculas utilizadas de forma correta e remoção de palavras ofensivas ou impróprias. A partir destas etapas a resposta estará pronta a apresentar ao utilizador: o modelo irá atuar em sentido inverso, isto é, passar a lista de *tokens* previstos para texto possível de ser interpretado pelo ser humano.

Andreia Filipa Almeida Graça

4 ESTADO DA ARTE

Neste capítulo, é apresentada uma revisão do estado da arte relacionado ao tema central do trabalho, com foco nas ferramentas e aplicações existentes no mercado. São abordados conceitos no âmbito de GenAI, descrito um estudo realizado a bases de dados relacionais e não relacionais, essenciais para a concretização do projeto. Apresenta-se a evolução dos modelos GPT, os desafios dos LLMs e soluções emergentes, como as RAG Applications. Assim proporciona-se uma visão abrangente do contexto tecnológico e teórico que sustenta o desenvolvimento do projeto SportGPT.

4.1 Trabalho Relacionado

Antes de iniciar o desenvolvimento da solução foi necessário estudar o mercado para entender que outras ferramentas semelhantes existem, como se posicionam, quais as vantagens e desvantagens, pontos fortes e fracos identificados pelos utilizadores. Com este processo foi possível entender o potencial diferenciador da ideia proposta perante as soluções já existentes: a capacidade de dar respostas atualizadas, precisas e coerentes às questões colocadas pelos utilizadores. No mercado existem várias aplicações e ferramentas que se assemelham à nossa em diferentes vertentes, tendo sido divididas em 2 categorias: Ferramentas de PLN e Aplicações de Futebol. Em B nos anexos, encontra-se disponível uma tabela comparativa entre as aplicações e ferramentas analisadas.

Ferramentas de PLN – ferramentas que fazem uso de IA / GenAI, e possuem a capacidade de conversação em tempo real recorrendo a LLMs. No entanto, diferem ao ter capacidade de responder a perguntas dos mais variados temas, não se limitando exclusivamente ao futebol. Apresentam uma dada incerteza nas respostas fornecidas, estando estas limitadas relativamente à atualidade.

- Bard / Gemini – ferramenta projetada pelo Google em 2023, baseava-se no PaLM2 (modelo de linguagem desenvolvido pelo Google AI, com capacidade para escrever conteúdos criativos, gerar texto, código ou mesmo traduções). A 5 de fevereiro de 2024 foi substituído pelo Gemini, um produto com capacidade de compreensão de diversos tipos de informação: vídeo, texto, áudio, disponível tanto na *web* como para *mobile* [53]. Esta ferramenta foi treinada com milhões de dados, proporcionando uma melhor compreensão do conteúdo introduzido pelo utilizador, em qualquer idioma [54]. Possui um vasto conhecimento em várias áreas, propiciando resposta a um enorme leque de perguntas. Os utilizadores conseguem fazer pesquisas mediante linguagem natural, garantindo facilidade de uso e a sensação de “conversação humana”.

- Chatsonic - ferramenta de *chatbot* desenvolvida pela WriteSonic, lançada em 2022, com cerca de 5 milhões de utilizadores ativos [55]. Encontra-se disponível em *web* e

em *app mobile* (Android e iOS)[56] . Utiliza algoritmos de Inteligência Artificial (IA), *Machine Learning* (ML) e combinação de diálogos (para simular conversação humana) para ter capacidade para responder a questões em tempo real sobre os mais variados temas. Conta com habilidades para fornecer conteúdos criativos, gerar texto, imagens, traduções e *emails*, tudo de forma clara e célere para personalizar a interação com o utilizador.

Aplicações de futebol – estas aplicações têm foco exclusivamente na temática do futebol, não tendo a capacidade de conversação em tempo real. Com exceção da aplicação Football AI que permite diálogo, mas apenas entre utilizadores e não recorrendo a LLMs como proposto no projeto SportGPT. Estas aplicações concentram-se na realização de previsões futebolísticas de acordo com dados futebolísticos históricos.

- Football AI – aplicação futebolística que se concentra na utilização de IA para realização de previsões futebolísticas, posicionando-se como uma potencial adversária às casas de apostas. Encontra-se disponível numa plataforma *web* e para *mobile* contando com mais de 1 Milhão de *downloads* [57]. Descreve-se como um sistema de previsão de futebol para fornecer aos utilizadores previsões acerca dos vários acontecimentos ao longo de um jogo de futebol (golos, cartões amarelos e vermelhos) e estatísticas: posse de bola, substituições de jogadores ou oportunidades de golo. Esta análise é feita através do uso de ML, IA, informações relativas à qualidade e formação das equipas, estatísticas de jogadores, dados em tempo real e ainda um histórico de jogos anteriores (cerca de 1000 *datapoints*). O Football AI apesar de possuir capacidade de conversação em tempo real, difere da ideia proposta no sentido em que apenas a permite entre utilizadores e não entre o utilizador e IA.

- Betty Football - aplicação desenvolvida e lançada em 2023, pela Oddacy, com objetivo de revolucionar o mercado das apostas de futebol [58]. Descreve-se como a 1.ª assistente de futebol com IA a nível mundial já somando cerca de 1000 *downloads* tanto em Android como iOS [59]. Esta aplicação integra o ChatGPT e utiliza aproximadamente 150 algoritmos de IA e ML, treinados com dados das 5 principais ligas europeias, de 1863 a 2023. Assim conta fornecer aos utilizadores, adeptos de futebol, previsões e *insights* em tempo real quanto ao número de golos marcados, qual primeiro jogador a marcar, cartões vermelhos e amarelos atribuídos e resultado dos jogos. Propõe três competências: *Forecaster* prever golos ou quando a equipa vencedora pode mudar, *Analytical* (fornece dados estatísticos adequados ao momento do jogo) e *Watcher* (de decisões do VAR - *Video Assistant Referee*, penáltis e cartões amarelos/ vermelhos)[58].

- Nerdytips, é uma plataforma lançada em 2021 que analisa várias partidas de futebol com fim a realizar previsões futebolísticas [60]. Possui mais de 50 mil *downloads* a nível da *app Mobile* (Android, iOS) [61] e existindo também a versão *web*. Existem três planos de utilização pagos consoante a quantidade de meses que os utilizadores pretendam utilizá-la: 1, 3 ou 6 meses, todos com as mesmas características. Esta aplicação utiliza algoritmos de IA, ML e ainda fórmulas matemáticas (como forma a

umentar a precisão) para prever a probabilidade de: cada uma das equipas ganhar, o n.º de golos marcados num jogo, o n.º de cartões amarelos e vermelhos atribuídos ou mesmo o resultado do jogo. Os algoritmos usados têm em consideração: resultados de jogos e o histórico de *performance* e formação da equipa.

- A BetMines, é uma plataforma, lançada em 2019, de dicas de futebol que realiza previsões de futebol em tempo real. Através da *app mobile* [62], e da plataforma *web* pretende ajudar os adeptos de futebol a ter mais hipótese de ganharem com as suas apostas futebolísticas. Com base em algoritmos de IA e ML, a aplicação consegue processar grandes quantidades de dados e utiliza fórmulas matemáticas para gerar resultados confiáveis. De acordo com estes dados tem a capacidade de prever: a probabilidade de vitória para cada uma das equipas, golos, n.º de cartões vermelhos e amarelos atribuídos e ainda o resultado do jogo [63].

- A Fidelisbet é uma aplicação lançada em 2021, que se destaca perante plataformas futebolísticas ao oferecer uma ampla gama de funcionalidades baseadas em informações futebolísticas para mais de 50 competições e encontra-se disponível na *web* e em *app mobile* com mais de 50 mil transferências [64]. Uma das principais características é a sua capacidade de fornecer alertas em direto de vários eventos de jogo: faltas, cantos, golos, resultado. Para cada jogo apresenta um conjunto de informações históricas antes do jogo: quantos golos a equipa marcou nos últimos jogos, quantas vitórias e derrotas nas competições, a classificação atual da equipa e o onze que entrará em campo. A Fidelisbet utiliza vários algoritmos de IA para prever acontecimentos ao longo dos jogos de futebol: próximo golo, cantos, qual a equipa vencedora, dados deveras úteis para a realização de apostas desportivas. A característica que torna esta aplicação verdadeiramente inovadora e que a coloca bastante bem posicionada no mercado passa pela capacidade dos próprios utilizadores criarem os seus próprios algoritmos recorrendo aos dados que a *app* detém. Caso estes não se sintam confortáveis nessa criação, poderão subscrever algoritmos criados pela comunidade utilizadora da *app*. A eficácia de cada algoritmo é transparente, permitindo aos utilizadores observar a taxa de sucesso de cada um [65].

- A AIStats, uma aplicação lançada em 2023, destaca-se por fornecer previsões futebolísticas em tempo real, baseadas em IA para todas as partidas existentes. Afirma-se como uma aplicação revolucionária com uma precisão 11% superior às casas de apostas, está disponível em aplicação móvel já com mais de 50 mil *downloads* [66]. Os dados de 2 Milhões de jogos são analisados há mais de 3 anos, promovendo uma aprendizagem iterativa dos modelos, e conseqüentemente melhores resultados ao longo dos tempos. Ainda existe um plano “pro” que envia alertas ao utilizador dos novos palpites gerados pelos algoritmos de IA. Nesta aplicação os utilizadores podem inserir as suas apostas numa calculadora e acompanhar os resultados em tempo real, poderão ainda debater com outros utilizadores da aplicação e trocar opiniões acerca de cada partida. Toda a comparação entre as várias aplicações e ferramentas foi sintetizada numa tabela comparativa com objetivo de resumir as

principais diferenças entre elas. Esta tabela encontra-se em apêndice no presente documento.

4.2 Conceitos

Neste subcapítulo será realizada uma linha temporal da evolução do GPT-1 até ao GPT-4, os vários LLMs lançados pela OpenAI. Assim será possível entender os pontos de melhoria e de evolução em cada novo modelo. É importante realçar que o que difere os vários modelos é o número de parâmetros e a base de conhecimento que detêm.

Para entender este subcapítulo será necessário definir alguns conceitos relevantes:

Corpus – conjunto de dados geralmente textuais usados para análise linguística, treino de modelos e PLN [67]. Cada *corpus* contém textos numa única língua, mas um conjunto de *corpus* em diferentes línguas é conhecido como *corpora*. Geralmente os textos estão em diferentes idiomas, géneros e estilos. O *corpus* é essencial para prover os algoritmos de ML e torná-los precisos.

Book Corpus – representa uma extensa coleção de textos extraídos de livros, abrangendo uma grande variedade de géneros. Esta coletânea pode ser utilizada para diversos propósitos, por exemplo: linguísticos ou treino de modelos de IA. Este conjunto comporta uma enorme riqueza de dados e fontes de linguagem natural [68].

Common Crawl – projeto desenvolvido para criar e manter *corpus web* de acesso público. Pretende disponibilizar uma vasta quantidade de dados da *web* de forma gratuita e promover a inovação em várias áreas [69]. Coleta dados e páginas regularmente da *web*. Este projeto pode ser utilizado para uma vasta gama de propósitos: desenvolvimento de modelos de linguagem, pesquisas escolares ou mesmo aplicações de PLN.

Web Crawling – processo automático de navegação e recolha de informações que se encontram disponíveis na Internet. Estas ações são feitas por agentes denominados “*web crawlers*”, cuja função é percorrer a *web* de forma recorrente, seguindo os *links* entre páginas. Os dados recolhidos através deste processo podem ser utilizados para diversos fins. Esta abordagem é muito utilizada por motores de busca como o Google de forma a indexar a *web* e permitir que os utilizadores obtenham de forma rápida aquilo que procuram [70].

Janela de Contexto de um LLM – medida de duração de contexto e memória da conversa. Se um modelo tiver uma janela de contexto com 4 mil *tokens*, o correspondente a 3 mil palavras inglesas [71], [72], tudo o que estiver dentro deste limite será considerado, após este limite alguns *tokens* serão ignorados e as respostas poderão ser menos precisas e possivelmente divergentes relativamente a respostas anteriores [73]. O que poderá ser uma grande desvantagem relativamente a documentos mais longos ou conversas mais longas em *chatbots*.

4.3 Modelos de Bases de Dados

Neste subcapítulo serão apresentados relevantes para o projeto SportGPT relativamente aos modelos de bases de dados existentes e considerados no estudo do projeto.

4.3.1 Base de Dados SQL

Nesta secção será abordado e apresentado o estudo de uma das linguagens utilizadas no projeto SportGPT: o SQL. A escolha desta linguagem foi motivada pela necessidade de lidar com dados fornecidos pela empresa compreendidos num *dump*, com esquema tabular.

O SQL, ou *Structured Query Language* é uma linguagem desenvolvida em 1974 na IBM (*International Business Machines Corporation*), com fim de gerir bases de dados relacionais. Esta linguagem surgiu no contexto do “System R”, um projeto com objetivo de implementar o modelo relacional proposto por Edgar Codd [74]. Mais tarde o SQL foi padronizado pelo *American National Standards Institute* (ANSI) e pela ISO (*International Organization for Standardization*), em 1986 e em 1987, respetivamente. Nesta abordagem os dados encontram-se armazenados em tabelas, onde as linhas representam instâncias e as colunas atributos. O SQL é utilizado em diversos tipos de aplicações, dado que tem a capacidade de integração com linguagens de programação distintas. O SGBD (Sistema de Gestão de Base de Dados Relacional) é um tipo de *software* que faz uso de SQL para gestão de bases de dados relacionais, permitindo a criação, leitura e supressão de dados seja feita de forma eficiente [75]. Existem vários sistemas que fazem uso do SQL para gestão de bases de dados relacionais como o MySQL, PostgreSQL, Oracle e SQL Server.

É fundamental mencionar Transações ACID (*Atomicity, Consistency, Isolation, Durability*) no contexto de SQL e RDBMS (*Relational Database Management System*), pois representam um conjunto de propriedades que garantem a confiabilidade do processamento de transações (sequência de operações que são executadas como uma só unidade lógica, podem incluir vários comandos: CREATE, INSERT, DELETE, UPDATE). Estas transações encontram-se descritas na Tabela 4-1.

Tabela 4-1 - Descrição das Transações ACID

<i>Características</i>	<i>Descrição</i>
<u><i>A</i></u> <i>tomicidade</i>	Garante que a transação é única e indivisível, isto é, ou todas as operações são bem-sucedidas ou então nenhuma delas é efetuada
<u><i>C</i></u> <i>onsistência</i>	A base de dados permanece num estado consistente antes e depois da realização de uma transação

<i>Isolamento</i>	Embora vários utilizadores possam fazer várias transações simultaneamente, uma transação não deve ver os efeitos de outras transações
<i>Durabilidade</i>	Após o término de uma transação esta é guardada com o comando COMMIT. É esperado que estas alterações persistam mesmo em caso de falha do sistema operativo ou hardware

O SQL permite utilização de comandos como: CREATE, DELETE, INSERT, UPDATE e WHERE, permitindo operações de armazenamento, atualização, remoção, pesquisa e recuperação de dados de uma base de dados. A curva de aprendizagem é suave e acessível para iniciantes com conhecimentos básicos de base de dados, pois a sintaxe utilizada é simples, utilizando comandos e palavras-chave em inglês: SELECT, INSERT, UPDATE e DELETE, o que os torna fáceis de aprender.

4.3.1.1 Vantagens do SQL

O SQL é uma das linguagens de consulta mais conhecidas e utilizadas a nível de bases de dados relacionais. Nesta subsecção são apresentadas algumas das vantagens da linguagem [76]:

- **Padronização** – como referido acima, o SQL foi padronizado pela ANSI e pelo ISO, garantindo uniformidade e interoperabilidade. Isto é, os programadores podem escrever código SQL que será funcional em diferentes sistemas de gestão de bases de dados, promovendo a compatibilidade entre plataformas;

- **Documentação** – oferece uma vasta quantidade de documentação proporcionando uma fácil curva de aprendizagem. Através da extensa documentação, como livros e tutoriais, permite que os utilizadores se tornem aptos rapidamente;

- **Alta Utilização** – linguagem padrão muito aplicada na indústria das bases de dados e suportada por muitos sistemas de gestão de bases de dados, como: MySQL, PostgreSQL, Oracle, entre outros;

- **Portabilidade** – pode ser empregue para diferentes aplicações e ambientes, tornando a linguagem bastante versátil;

- **Visualização** – é possível definir visualizações distintas da estrutura da base de dados para diferentes utilizadores, denominadas *views*. Esta possibilidade permite que os diferentes utilizadores acedam e manipulem dados de acordo com a suas necessidades;

- **Fácil adaptação** para tarefas básicas, como consulta, inserção, atualização e exclusão de dados as operações CRUD (*Create, Read, Update, Delete*);

- **Eficiência** – permite a recuperação de enormes quantidades de dados de forma eficaz, graças a técnicas de otimização de consultas e índices;

- **Integridade dos dados** – disponibiliza recursos que garantem a integridade dos dados, principalmente: restrições de *foreign keys* e transações ACID. Desta forma é garantido que os dados permaneçam consistentes, mesmo em situações adversas como falhas;

- **Escalabilidade** – a base de dados consegue expandir à medida que as necessidades de armazenamento avolumam.

4.3.1.2 *Desvantagens SQL*

Embora o SQL seja muito reconhecido como uma ferramenta bastante conhecida e utilizada para a manipulação e gestão de bases de dados relacionais, não está livre de ter desvantagens. Nesta subsecção são apresentadas algumas das desvantagens inerentes à linguagem [77].

- **Dificuldade** com a interface, para utilizadores não acostumados com bases de dados e informática;

- **Custo** – alguns serviços têm custos e licenças operacionais elevados, dificultando o acesso a programadores;

- **Complexidade** – embora seja fácil e simples para operações básicas, a abordagem pode tornar-se complicada para consultas complexas em bases de dados muito elaboradas. Isto é, com imensos dados e tabelas;

- **Limitações** na escalabilidade horizontal: torna-se difícil adicionar mais servidores para lidar com cargas de trabalho. Podem requerer técnicas complexas de particionamento e escalabilidade. As bases de dados NoSQL, nestes casos, são mais flexíveis;

- **Modelo de dados rígido** – o modelo de dados utilizado não é ideal para dados não estruturados ou semiestruturados, comuns nas aplicações atuais. Por exemplo, dados em formatos como JSON (*JavaScript Object Notation*) ou XML (*Extensible Markup Language*) tornam-se difíceis de gerir.

4.3.2 **Bases de Dados Não Relacionais**

As bases de dados não relacionais, contrariamente às bases de dados relacionais como o SQL, não armazenam os dados em linhas e colunas. Essas bases de dados usam diferentes modelos de dados para aceder e gerir os dados, onde os esquemas são otimizados para aplicações que exigem grandes volumes de dados, modelos de dados flexíveis, baixa latência, através do distender de restrições na consistência de dados [78][79]. O esquema de dados permite que estes sejam armazenados sem restrições, contrariamente às bases de dados não relacionais. Existem 4 tipos diferentes de bases de dados: *key-value*, *column*, *document* e *graph* apresentadas e descritas nas secções seguintes.

4.3.2.1 Bases de Dados Key-Value

Tipo de base de dados não relacional, descrita nesta subsecção, em que os dados são armazenados em pares chave-valor (ou *key-value*), onde a chave funciona como um identificador único e os valores correspondem aos atributos, apresentado na Figura 4-1.

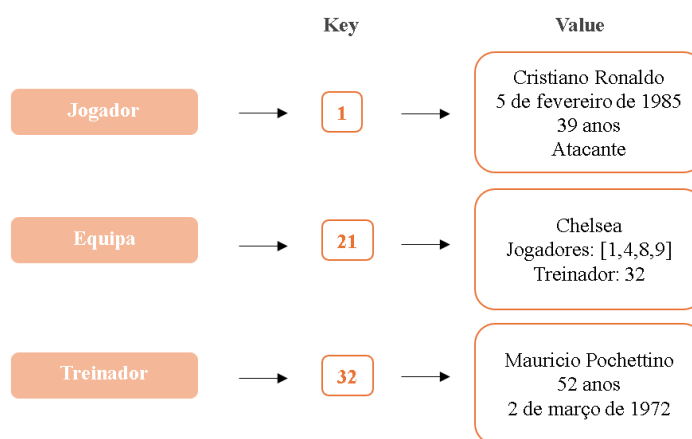


Figura 4-1 - Exemplo de Base de Dados *Key-Value*

Tanto a chave como os valores podem ser de formatos distintos: mais simples como *strings* ou mais complexos como objetos JSON. Estas bases de dados são muito particionáveis e permitem uma escalabilidade horizontal melhor que outros tipos de bases de dados não relacionais [78]. Estas bases de dados seguem um paradigma orientado a objetos, permitindo que os *developers* mapeiem objetos reais com objetos de “*software*”, algo comum em muitas linguagens de programação como Java. As bases de dados *key-value* têm a capacidade de escalonamento horizontal, distribuindo os dados entre os vários servidores para reduzir os pontos críticos (*bottlenecks*) num só servidor. Destaca-se pelo alto desempenho, baixa latência e tempos de resposta mais curtos, assim bem como o constante processamento de operações de leitura e escrita. Contrariamente às bases de dados relacionais não realizam *joins* entre as várias tabelas e ao aplicar menos recursos tornam-se mais rápidas [79]. Os usos mais comuns de bases de dados *key value* são: gestão de sessões de utilizador quando este faz um registo ou *login*, armazenamento de meta dados e cache. Infelizmente, não oferecem suporte para *queries* complexas e ditam limites na quantidade de dados que se podem filtrar. Na Tabela 4-2 são apresentadas as principais características deste tipo de bases de dados.

Tabela 4-2 - Características da base de dados do tipo Chave-Valor

<i>Características da Base de Dados</i>	<i>Par Chave-Valor</i>
<i>Armazenamento</i>	Os dados encontram-se armazenados em pares chave-valor, onde cada valor é associado a uma chave única.
<i>Modelo de Dados</i>	Key-value

<i>Escalabilidade</i>	Vertical e Horizontal
<i>Desempenho</i>	Muito Elevado
<i>Consistência</i>	Eventual
<i>Flexibilidade</i>	Muito Alta
<i>Casos de Uso</i>	Armazenamento de cache, sessões de utilizador
<i>Vantagens</i>	Simplicidade, alta velocidade de leitura e escrita
<i>Desvantagens</i>	Limitado para consultas complexas
<i>Exemplos</i>	Redis, DynamoDB, Riak

4.3.2.2 Bases de Dados Document

Bases de dados do tipo documento (ou *document*), apresentadas nesta subsecção, são utilizadas para armazenar e consultar dados no formato de documentos como, por exemplo, JSON (formato possível de ler tanto por humanos como por máquinas). Este tipo de base de dados permite que os documentos sejam armazenados tendo em conta uma hierarquia e que evoluam consoante as necessidades das aplicações [78]. Consoante esta abordagem é possível criar vários documentos, cada um com os seus próprios campos, podendo diferir entre si na mesma coleção e os dados armazenados podem ser tanto estruturados como não estruturados. Um exemplo é representado na Figura 4-2.

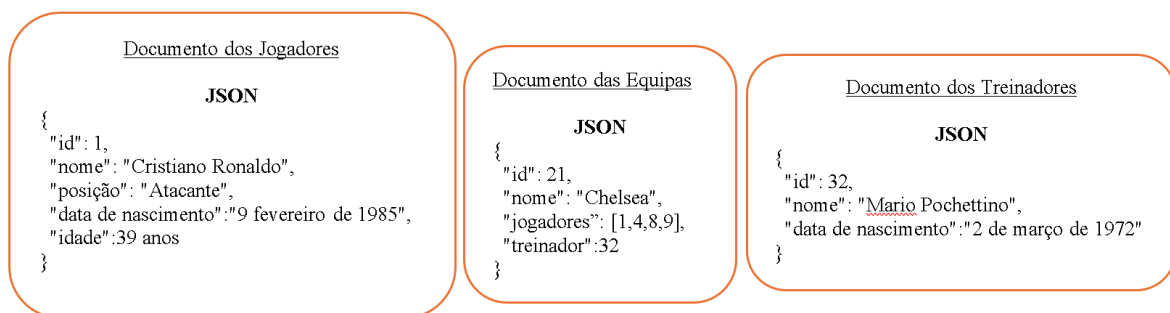


Figura 4-2 - Exemplo de Base de Dados *Document*

A indexação revela ser bastante flexível, os programadores não necessitam de criar modelos de dados com antecedência, tornado o desenvolvimento de aplicações muito mais célere [79]. No entanto, existem bases de dados que executam verificações de esquema e podem apresentar restrições quanto à estrutura. Enquanto nas bases de dados do tipo *key value* tanto as chaves como os valores podem ser de qualquer formato. No caso das bases de dados do tipo *document* só podem ser do tipo *string*, e permitem armazenar PDFs (*Portable Document Format*), imagens ou mesmo documentos de texto de forma direta. Permite escalar os dados de forma horizontal em vários servidores sem que o desempenho seja afetado. Estas bases de dados podem ser utilizadas para gestão de conteúdos como *blogs*, catálogos ou gestão de

dados provenientes de sensores [78]. Na Tabela 4-3 são apresentadas as características deste tipo de bases de dados.

Tabela 4-3 - Características de bases de dados do tipo Documento

<i>Características da Base de Dados</i>	<i>Documento</i>
<i>Armazenamento</i>	Dados armazenados em documentos Cada documento pode ter uma estrutura diferente. Armazenamento é flexível e semiestruturado.
<i>Modelo de Dados</i>	JSON, BSON (<i>Binary JSON</i>) e XML
<i>Escalabilidade</i>	Horizontal
<i>Desempenho</i>	Elevado
<i>Consistência</i>	Eventual
<i>Flexibilidade</i>	Alta
<i>Casos de Uso</i>	Armazenar dados semiestruturados. Com capacidade para representar hierarquias
<i>Vantagens</i>	Esquema flexível e fácil de escalar
<i>Desvantagens</i>	Menos eficiente em pesquisas complexas
<i>Exemplos</i>	MongoDB, CouchBase

4.3.2.3 Bases de Dados de Graph

As bases de dados do tipo grafo (ou *graph*), descritas nesta subsecção, armazenam dados e as relações entre si [78]. Um grafo é uma representação de nós e relacionamentos que se conectam entre si, um nó representa uma entidade ou objeto no grafo, por exemplo: um jogador. Um relacionamento representa a conexão entre 2 entidades, e pode ter ou não associada uma direção e/ou propriedades adicionais. Por exemplo: um jogador pertence a uma equipa como apresentado na Figura 4-3.

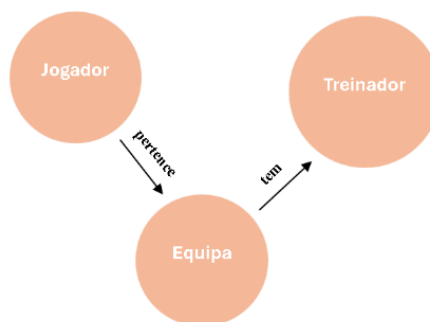


Figura 4-3 - Exemplo de Base de Dados *Graph*

Ao contrário das bases de dados relacionais que armazenam os dados em rígidas tabelas, as bases de dados do tipo grafo armazenam-nos consoante uma rede de entidades / nós e relacionamentos. Estas bases de dados oferecem melhor desempenho e flexibilidade e são as mais adequadas para modelar cenários do mundo real [79]. Podem ser utilizadas para detetar fraudes financeiras através da análise de padrões, mecanismos de recomendação ou mesmo otimização de rotas.

As consultas realizadas, através dos grafos, conseguem analisar diversas situações com muito mais rapidez, pois podem contar e comparar o número de relacionamentos entre 2 nós de forma fácil. Na Tabela 4-4 são apresentadas as características das bases de dados do tipo grafo.

Tabela 4-4 - Características de bases de dados do tipo Grafo

<i>Características da Base de Dados</i>	<i>Grafos</i>
<i>Armazenamento</i>	Os dados encontram-se armazenados em nós (entidades) e arestas (relacionamentos)
<i>Modelo de Dados</i>	Nós e relacionamentos
<i>Escalabilidade</i>	Horizontal
<i>Desempenho</i>	Elevado
<i>Consistência</i>	Forte
<i>Flexibilidade</i>	Alta
<i>Casos de Uso</i>	Redes Sociais e Sistemas de Recomendação
<i>Vantagem</i>	Modelar dados complexos, Capacidade de interligar dados
<i>Desvantagem</i>	Complexo para cenários simples
<i>Exemplos</i>	Neo4J, ArangoDB

4.3.2.4 Bases de Dados de Column

Bases de dados do tipo Coluna (*column* ou *Column Family*) foram desenvolvidas para armazenar e gerir enormes quantidades de dados de forma eficiente. São úteis em casos onde as consultas analíticas necessitam de ser rápidas dado o objetivo de analisar grandes volumes de dados e extrair *insights* através de agregações como somas, médias ou contagens. Cada linha é identificada por uma chave única, e dentro de cada linha existem colunas, onde os dados são armazenados [78]. Cada coluna tem um nome, um valor e um *timestamp* (opcional). Um grupo de colunas pode ser chamado de super coluna, onde cada uma tem uma chave e um conjunto de colunas. Estas bases de dados são excelentes para consultas que envolvem agregações e leituras em grandes quantidades [78]. Para além de serem altamente escaláveis de forma horizontal, permitem que o *overhead* seja reduzido, pois apenas as colunas

necessárias são lidas e processadas. As bases de dados do tipo *Column* são adequadas para *Data Warehousing* e IoT (*Internet of Things*).

No entanto, estas bases de dados, apesar de permitirem otimizar o desempenho, exigem a desnormalização dos dados, o que pode provocar duplicação de informações e conseqüente aumento no uso do armazenamento. As bases de dados não são adequadas para consultas complexas e flexíveis, que envolvam vários *joins* e agregações sofisticadas, e pode demonstrar problemas de desempenho em operações de escrita intensa. Na Tabela 4-5 são apresentadas características deste tipo de bases de dados.

Tabela 4-5 - Características de bases de dados do tipo *Column*

<i>Características da Base de Dados</i>	<i>Column</i>
<i>Armazenamento</i>	Dados armazenados em forma de coluna em vez de linhas.
<i>Modelo de Dados</i>	Tabelas e colunas
<i>Escalabilidade</i>	Horizontal
<i>Desempenho</i>	Elevado
<i>Consistência</i>	Eventual
<i>Flexibilidade</i>	Média
<i>Casos de Uso</i>	Data Warehousing e Análise de dados
<i>Vantagens</i>	Alta compreensão, leitura eficiente de grandes conjuntos de dados
<i>Desvantagens</i>	Inadequado para transações de alta frequência
<i>Exemplos</i>	Cassandra, HBase

4.3.3 Discussão

Bases de Dados Relacionais e Não Relacionais gerem os dados de formas distintas. As bases de dados relacionais ao armazenarem os dados de forma tabular permitem que sejam processadas complexas consultas em dados estruturados mantendo sempre a integridade e consistência dos dados. As bases de dados relacionais ao serem mais flexíveis permitem lidar com situações em que os requisitos são variáveis, podendo armazenar vários tipos de dados, sejam semi-estruturados ou não estruturados: imagens, vídeos e documentos [79]. As bases de dados relacionais seguem propriedades ACID, o que significa que um conjunto de operações será concluído em conjunto. Se uma operação falhar, todo o conjunto de operações falha. Promovendo a precisão dos dados em todos os momentos. As bases de dados não relacionais garantem disponibilidade, mas não consistência imediata. O estado da base de dados pode variar de acordo com o tempo. A performance das bases de dados relacionais prende-se com a subsistência do disco, para melhorar o seu






desempenho podem-se usar SSDs (*Solid-State Drive*). A performance das bases de dados não relacionais prende-se com a latência da rede, do tamanho do *cluster* e do *hardware*. Para melhorar o desempenho pode-se: aumentar o tamanho do *cluster*, minimizar a latência da rede, indexar *cache*. O esquema rígido da base de dados relacional apresenta grandes desafios no que toca à escalabilidade. O mais comum é escalar verticalmente adicionando mais recursos de CPU (*Central Processing Unit*) ou RAM (*Random Access Memory*) ao servidor. Mas também é possível escalar de forma horizontal duplicando os dados entre os servidores, o que necessita de particionamento e fragmentação [80], [81]. As bases de dados NoSQL são altamente escaláveis, os *workloads* podem ser distribuídos por vários nós com facilidade. Esses nós têm a capacidade de processar um grande volume de dados em conjuntos menores ao distribuir os conjuntos em vários nós.

A seleção da base de dados que melhor se adequa a um projeto, quer seja relacional ou não relacional, está inteiramente ligada às necessidades e requisitos em causa. Cada um dos tipos de bases de dados previamente elencados e descritos possui vantagens e desvantagens, o que provoca a necessidade de uma análise exaustiva e criteriosa para determinar aquela que melhor corresponde ao contexto em causa. Assim torna-se imprescindível equilibrar os diferentes aspetos envolvidos e avaliar minuciosamente as características das bases de dados para tomar uma decisão informada.

4.3.3.1 Estudo das Bases de Dados do Tipo Grafo

Um tipo de base de dados essencial no projeto SportGPT é o tipo Grafo, tendo sido realizado um estudo exaustivo de 4 bases de dados desse tipo, apresentado na Tabela 4-6 [81]. A decisão da escolha da base de dados encontra-se descrita na secção 7.3.1.

Tabela 4-6 - Tabela Comparativa entre bases de Dados de Grafos

Base de Dados	 Neo4J [82]	 Memgraph [83]	 Nebula [84]	 Graph	 Janus Graph [85]
Ranking DB Engines	23	98	120		129
Descrição	Base de dados escalável, otimizada para clusters distribuídos e com alto desempenho	Base de dados criada para <i>streaming</i> em tempo real	Base de Dados distribuída de alto desempenho e escalável		Otimizada para clusters distribuídos
1ª release	2007	2017	2019		2017
Versão atual	5.23 (agosto 2024)	2.15	3.8		0.6.3 (fevereiro 2023)

<i>Licença</i>	OpenSource	Comercial	OpenSource	OpenSource
<i>Cloud-based</i>	Não	Não	Não	Não
<i>Linguagem de implementação</i>	Java, Escala	C e C++	C++	Java
<i>SQL</i>	Não	Não	Semelhante	Não
<i>Linguagem de consulta</i>	Cypher	Cypher, GQL(<i>Graph Query Language</i>)	Nebula	Gremlin
<i>Linguagens de Programação Suportadas</i>	.Net, Clojure, Go, Java, Python	.Net, C, C++, Python	.Net, C++, Go, Java, Python	Clojure, Java, Python
<i>Consistência</i>	Imediata	Imediata	Imediata	Eventual e Imediata
<i>Foreign Keys</i>	Sim	Sim	Sim	Sim
<i>Transações</i>	ACID	ACID	ACID	ACID
<i>Simultaneidade</i>	Sim	Sim	Sim	Sim
<i>Durabilidade</i>	Sim	Sim	Sim	Sim
<i>Escalabilidade</i>	Horizontal Mas com maior ênfase na disponibilidade e consistência	Horizontal Com implementação em cluster	Horizontal com alta distribuição	Horizontal com implementação em clusters distribuídos
<i>Ecossistema</i>	Contém bastantes integrações e ferramentas de terceiros	Em crescimento com foco em integrações com Python	Em desenvolvimento com suporte para integração com várias linguagens	Em desenvolvimento com suporte para integração com várias linguagens compatíveis com o ecossistema Apache
<i>Curva de Aprendizagem</i>	Fácil de aprender e documentação abrangente	Fácil de usar e com documentação detalhada	Pode exigir mais esforço para configuração e operação	Mais difícil de aprender, com configurações complexas

A escolha da base de dados mais vantajosa está dependente dos requisitos intrínsecos ao projeto em causa, devendo considerar-se as vantagens e desvantagens da utilização de cada uma.

4.3.3.2 Neo4J

O Neo4J é um sistema de gestão de bases de dados desenvolvido pelo Neo4J, Inc, criado em 2007, tendo lançado a sua primeira versão (1.0) em 2010 [86]. A versão mais recente, 5.23, foi disponibilizada em agosto de 2024, disponível

localmente em versão *Community* e *Enterprise*. O Neo4J é uma base de dados não relacional, do tipo grafo, que contrariamente às bases de dados relacionais não adota o armazenamento convencional dos dados em tabelas, linhas e colunas. Os dados são armazenados em nós e relacionamentos, excluindo a ideia da existência de um esquema fixo, onde os nós representam entidades e os relacionamentos as conexões entre os dados [87]. Desenvolvida em Java e Escala, destaca-se por ser uma base de dados escalável e compatível com transações ACID, promovendo arquitetura distribuída e de alto desempenho com disponibilidade para ser hospedada na *cloud*. Oferece suporte para várias linguagens de programação como .Net, Go, Java, JavaScript, Python, entre outras. O Neo4J ocupa a 21^a posição¹ de entre os 410 sistemas, e ocupa a 1^a posição de entre as bases de dados do tipo grafo, segundo o *ranking* mensal da DB Egres [82], [88]. Este *ranking* é uma plataforma de classificação de sistemas de gestão de bases de dados de acordo com a sua popularidade, tendo em conta critérios como: mecanismos de pesquisa, discussões do Stack Overflow, menções e em redes sociais e Google Trends. Cypher é a linguagem de consulta da base de dados Neo4J, projetada para lidar com grafos e reconhecer padrões entre os dados [88]. Revela ser uma linguagem legível e intuitiva, próxima à linguagem humana, o que facilita a compreensão e construção de *queries*. Esta simplicidade e lógica torna a aprendizagem fácil e acessível para os utilizadores. Através de Cypher, os utilizadores conseguem expressar padrões de relacionamentos entre os diferentes nós e propriedades de forma eficaz. O Cypher possui cláusulas e palavras-chave específicas, que auxiliam a definição e manipulação dos dados e as respetivas relações, tal como apresentado na Tabela 4-7.

Tabela 4-7 -Cláusulas da Linguagem Cypher

<i>Cláusula</i>	<i>Descrição</i>
<i>MATCH</i>	Utilizada para encontrar padrões nos dados. Como o <i>select</i> em SQL
<i>WHERE</i>	Aplica filtros às <i>queries</i> a executar, permitindo especificar condições que os dados devem respeitar para que sejam contabilizados no resultado
<i>RETURN</i>	Elementos encontrados através da execução da <i>query</i> sejam nós, relacionamentos ou propriedades / atributos
<i>CREATE</i>	Utilizado para criar novos nós e relacionamentos
<i>DELETE</i>	Remove nós e relacionamentos da base de dados
<i>MERGE</i>	Combina a cláusula <i>MATCH</i> e <i>CREATE</i> , tenta encontrar um padrão existente, caso não encontra cria um novo nó ou relacionamento

A utilização destes comandos e cláusulas é essencial na manipulação e acesso a dados armazenados em bases de dados de grafos.

4.3.3.3 Vantagens do Neo4J

O Neo4J é a base de dados mais utilizada a nível das bases de dados de grafos [82], e para estar em tal lugar tem de apresentar vantagens [89]:

¹ Dados analisados no dia 20 de setembro de 2024

- **Desempenho elevado** – o Neo4J é otimizado para executar *queries* em grafos, permitindo que operações complexas envolvendo múltiplas relações sejam executadas em milissegundos. Desta forma, as consultas são mais eficientes, sem comprometer o desempenho e a integridade dos dados.

- **Menor necessidade de *Hardware*** – o Neo4J exige 10 vezes menos *hardware* do que outras bases de dados relacionais e não relacionais, graças à sua eficiente arquitetura em termos de otimização de recursos. Ao utilizar grafos como forma de armazenamento, recorre a algoritmos otimizados para armazenar e recuperar os dados. Assim, necessita de menos *hardware* e de um processamento menos intensivo para operações complexas.

- **Flexibilidade** - a base de dados pode ser executada de forma local ou na *cloud*. As empresas / utilizadores podem escolher qual a melhor configuração tendo em conta as necessidades específicas.

- **Linguagem de consulta Cypher** – esta linguagem permite lidar com dados armazenados em grafos e foi construída para ser intuitiva. Com esta linguagem é possível construir *queries* complexas com menos código comparativamente a SQL, assim o desenvolvimento é mais rápido e menos propenso a erros.

- **Esquema Flexível** – o Neo4J detém um esquema de dados flexíveis, permitindo adição de novos dados e relações sem necessidade de redefinir ou reestruturar o esquema já existente. Esta abordagem é deveras útil em ambientes onde os requisitos e os dados se encontram em constante mudança.

- **Forte UI** – o Neo4J oferece ferramentas de interface robustas, como por exemplo, o Neo4J Desktop, que permitem uma fácil visualização e interação com os dados. Assim é mais fácil compreender e extrair *insights* valiosos dos dados de forma rápida.

- **Suporte para Exportar dados para JSON e Excel** – esta capacidade de exportar dados em diferentes formatos aumenta a interoperabilidade entre outras ferramentas e sistemas.

- A base de dados assegura **consistência** (garante que os dados estejam sempre num estado válido), simultaneidade (permite que várias operações sejam executadas em simultâneo) e durabilidade (garante que as alterações nos dados sejam permanentes, mesmo em caso de falha do sistema), promovendo o processamento dos dados de forma confiável e segura.

Atualmente são produzidos enormes volumes de dados, e as relações entre eles são extremamente valiosas e importantes. As bases de dados do tipo grafo permitem representar essas mesmas relações de forma eficaz e eficiente.

4.3.3.4 Desvantagens do Neo4J

Embora a base de dados Neo4J ofereça vantagens significativas, existem algumas limitações a ter em conta [86]. Estes entraves encontram-se descritos nesta subsecção.

- **Não suporta fragmentação** – em cenários de alta escalabilidade, quando os dados crescem exponencialmente, pode levar a *bottlenecks* de desempenho. Este problema pode ser mitigado através de crescimento vertical (adição de memória e CPU);

- Não oferece suporte para objetos JSON e *arrays* complexos, o que pode dificultar o armazenamento de dados mais estruturados dentro da base de dados.

4.4 OpenAI

Empresa de pesquisa com objetivo de promover o desenvolvimento de IA “friendly” criada em 2015, São Francisco por Elon Musk, Salm Altman, Reid Hoffman, entre outros [5]. O objetivo da empresa é o desenvolvimento de ferramentas de IA segura e aberta, onde através de sistemas autónomos com capacidade de ultrapassar o ser humano em vários trabalhos sejam usados para benefício da humanidade e não para erradicá-la. Nos últimos anos a empresa tem sido reconhecida pela contribuição no campo de inovação e IA apresentando ferramentas impressionantes: DALL-E (gerador de imagens com IA) [19] ou, por exemplo, o OpenAI Codex [90] (modelo com capacidade para gerar código e alimenta o CoPilot [91] – a ferramenta de sugestão de código do Github). Mas o que a tornou efetivamente conhecida foi o lançamento do *chatbot* ChatGPT. Esta apresentação impulsionou o desenvolvimento de ferramentas semelhantes como o caso do Bard/Gemini [18], Llama [92], entre outros. O LLM (GPT) perante uma qualquer sequência de palavras, atribui uma probabilidade a toda a sucessão, tal como apreendido no treino, tenta prever a próxima palavra dessa sequência. Ao consistir num enorme conjunto de parâmetros, tem a capacidade de se “assemelhar” em certa parte à capacidade de aprendizagem do ser humano. A OpenAI tem lançado modelos, ao longo dos tempos, mais capazes e poderosos a nível das suas capacidades de entendimento de contexto, tarefas e geração de conteúdo [93], [94].

4.4.1 GPT-1

Em 2017 o Google lançou o *Transformer*, um modelo baseado em redes neuronais com capacidade de apreender o contexto. Como resposta, em julho de 2018, a OpenAI, através da divulgação de um artigo intitulado: “Improving Language Understanding by Generative Pre-Trained Transformer” [95], apresentou o seu primeiro modelo de LLMs: o GPT-1. Este modelo foi treinado com base em 7000 livros, escolhidos estrategicamente por conter longas dependências de contexto e por não terem sido publicados, logo a probabilidade de os encontrar em conjuntos

de dados posteriores era menor [96]. O objetivo do GPT-1 era apreender as nuances existentes na linguagem humana através de pré-treino generativo, para assim identificar padrões e regras nos léxicos e assim gerar frases completas e coerentes. O modelo era composto por cerca de 120 milhões de parâmetros e um *Book Corpus* de 4.5GB. O GPT-1 demonstrou ter capacidade para gerar texto coerente de acordo com as solicitações e algumas tarefas de PLN. No entanto, apresentou limitações ao não conseguir realizar muito mais tarefas para além do que aprendeu nesses livros [97][94]. Tal como um humano, que após ler 4 livros de histórias, não consegue responder a perguntas sobre obras com as quais nunca teve contacto.

4.4.2 GPT-2

A 14 de fevereiro de 2019, era lançado o sucessor do GPT-1. O GPT-2 foi desenhado para superar as limitações do modelo anterior, com 10 vezes mais parâmetros (1,5 bilhões de parâmetros) e dimensão do conjunto de dados também maior. Ambos os modelos tinham como base *Transformers* e foram treinados para gerar texto, prevendo a próxima palavra numa sequência de *tokens*. Apesar de ter sido treinado com cerca de 40GB de texto, cerca de 8 Milhões de documentos e 45 Milhões de páginas mais votadas no Reddit, ainda apresentava alguns problemas: grandes repetições e geração de extensas respostas [94], [97].

A arquitetura do *Transformer* detinha capacidade de processar múltiplas tarefas em simultâneo, onde os modelos poderiam ser treinados com *Corpus* superiores aos modelos de PLN anteriores. Estes utilizavam o Common Crawl, um extenso *corpus* produzido através de *web crawling*. Esta utilização ainda foi ponderada pela OpenAI, no entanto, após uma análise profunda, foi descartada por expor quantidades de conteúdo inteligível. Desta forma, a OpenAI desenvolveu o seu próprio *corpus webText* (informações recolhidas da Internet) em vez de proceder à recolha de conteúdo indiscriminadamente da Internet. Este conjunto de dados foi gerado através da recolha de várias páginas de *posts* de Reddit que receberam pelo menos 3 votos positivos antes de dezembro de 2017. Houve a necessidade de realizar uma limpeza aos dados: os documentos HTML (*HyperText Markup Language*) foram reduzidos em texto mais simples, as páginas que se encontravam duplicadas foram eliminadas e as páginas de Wikipédia foram descartadas, pois a sua presença em outros conjuntos de dados poderia introduzir *overfitting* no modelo. Dado que no pré-treino foram analisadas diferentes conversas do Reddit o modelo adquiriu capacidade de manter conversação de forma natural e coerente semelhante a um humano [94]. Ao contrário do GPT-1, este modelo utiliza modelos de pré-treino não supervisionado para tarefas supervisionadas, mas, em contrapartida, precisa de ajustes para tarefas importantes. Inicialmente o modelo não foi lançado ao público, pois a equipa tinha receio de que fosse utilizado para escrever *emails* fraudulentos ou mesmo construir notícias falsas. Devido à enorme quantidade de dados no seu conjunto de treino, o GPT-2 possuía capacidade de realizar uma maior gama de tarefas para além da simples geração de texto [96].

Algo demonstrativo das incríveis capacidades do modelo é a habilidade de generalização para tradução de textos de francês para inglês, mesmo quando nos seus dados de treino não existiam muitos textos em francês. Partes de textos escritos noutras linguagens, que não o inglês, foram retirados dos exemplos de treino, ficando apenas 10 MB (*Megabyte*) de informações em francês perante as 40.000MB de dados em inglês. Este feito deve-se à capacidade dos modelos de extrair e compreender padrões e relações entre frases e palavras.

4.4.3 GPT-3

O GPT-3 foi lançado a 11 de junho de 2020, através do seu extenso treino em dados disponíveis na Internet, este modelo detém conhecimentos mais amplos e contextualmente convenientes [98]. O modelo tem a capacidade de lidar com várias tarefas associadas a PLN como: resposta a questões, traduções, gerar código, de forma tão eficiente que é árduo distinguir os seus textos daqueles escritos por humanos. Esta característica tanto pode ser bastante vantajosa como também pode acarretar vários riscos: propagação de desinformação, viés e discriminações, sendo necessário ter em conta diretrizes éticas para mitigar possíveis problemas [94], [97]. Os dados de treino utilizados no GPT-3 baseavam-se em cerca de 3 bilhões de dados provindos da Wikipédia, 19 bilhões correspondentes a *webtext*, 410 bilhões de *Common Crawl* e cerca de 67 bilhões de dados de livros, correspondente a aproximadamente 45TB de dados e 175 Bilhões de parâmetros [94], [96]. Este modelo tem a capacidade de aprimorar o desempenho de forma contínua através da análise de novos exemplos e do contexto que lhe é passado. Desta forma é possível que o modelo e a sua aprendizagem se adaptem a vários temas, conseguindo lidar com várias tarefas de forma eficaz.

4.4.4 GPT-3.5 Turbo

O GPT-3.5 Turbo representa uma grande evolução relativamente ao modelo GPT-3, exibindo notáveis melhorias na compreensão e geração de texto assim bem como de retenção de contexto [99]. Estas melhorias devem-se ao facto do modelo ter uma base de dados de treino mais ampla, o que permite ao modelo lidar com tarefas mais complexas de forma eficiente e ser aplicado a uma maior gama de aplicações [100].

4.4.5 GPT-4

O mais recente modelo lançado pela OpenAI é o GPT-4, lançado a 14 de março de 2023 de forma limitada através do ChatGPT Plus [101]. O modelo tem a capacidade de processamento de imagens e não somente texto como os outros modelos. É um modelo muito superior aos outros a nível de complexos *prompts* e consegue identificar 26 idiomas. Mas apesar de todas essas vantagens, o modelo continua a ter alucinações, e não deve ser utilizado sem qualquer verificação e

confirmação das informações em tarefas em que a precisão é importante. Acredita-se que o modelo apresenta mais de 1 Trilião de parâmetros, que facilitam a compreensão de texto, mas ainda não foi confirmado pela OpenAI.

4.4.6 GPT 4-Turbo

Este novo modelo é um aprimoramento do GPT-4, com uma janela de contexto maior e treino com dados mais atualizados [102]. À medida que a OpenAI aprimora vários modelos do GPT-4, observa-se um acréscimo de instruções personalizadas, avançadas análises e novos recursos. As anteriores versões do GPT, nomeadamente o GPT-3.5 e o GPT-4, continham limites temporais de conhecimento até setembro de 2021, não tendo capacidade de resposta a questões de eventos corridos posteriormente, sem ser com a utilização de APIs/ fontes de dados externas [99]. O GPT-4 Turbo por sua vez, expandiu o limite de conhecimento até abril de 2023 (gpt-4-1106-preview), aumentando a confiabilidade e a abrangência de informações.

Na Tabela 4-8, é apresentada uma comparação entre os diferentes modelos de GPT, utilizados no projeto SportGPT [99].

Tabela 4-8 - Tabela Comparativa de GPTs

<i>GPT Model</i>	<i>Input usage</i>	<i>Output Usage</i>	<i>Context Window (tokens)</i>	<i>Treinado até</i>	<i>Data da Lançamento</i>	<i>Tamanho do Modelo</i>	<i>Nº de palavras em memória durante a conversação</i>
<i>GPT-3.5 Turbo</i>	\$3.00 / 1M Tokens	\$6.00 / 1M Tokens	4,096	Setembro, 2021	Novembro, 2020	175B de parâmetros	8,000
<i>GPT-4</i>	\$30.00 / 1M Tokens	\$120.00 / 1M Tokens	8,192	Setembro, 2021	Março, 2023	175B de parâmetros	64,000
<i>GPT-4 Turbo</i>	\$10.00 / 1M Tokens	\$30.00 / 1M Tokens	128,000	Abril, 2023	Novembro, 2023	>175B de parâmetros	32,000

Os modelos lançados pela OpenAI têm evoluído de forma significativa a nível das suas capacidades [93]. O GPT-4 representou um enorme salto relativamente ao seu antecessor: o GPT-3.5 Turbo. O GPT-4 Turbo apesar de mais recente que o GPT-4 demonstra ser uma versão mais *light*, mas com dados mais atualizados. Observamos uma ampliação de conhecimento e raciocínio, otimização e refinação dos conteúdos gerados de forma a auxiliar ainda mais os utilizadores a nível de diferentes tarefas em diferentes temáticas e áreas. A escolha do melhor modelo está dependente das necessidades, do conhecimento requerido ou da adaptabilidade.

4.4.7 GPT-4 vs GPT-4 turbo

Nesta secção será feita uma breve comparação entre os 2 modelos da OpenAI: GPT-4 e GPT-4 Turbo. Ambos são modelos de GenAI direcionados para o PLN, no entanto, apresentam diferenças importantes a nível de desempenho, custo e capacidades. Estes 2 modelos foram utilizados em contexto do projeto SportGPT, em diferentes estágios de desenvolvimento, sendo importante estabelecer as diferenças entre si.

4.4.7.1 GPT-4 [101]

- Trouxe a capacidade de criação de textos mais longos, com maiores conversas e a capacidade de procuras e análises em documentos;
- Data-limite de conhecimento – setembro de 2021, de acordo com esta limitação temporal os resultados podem não ser tão relevantes em contextos de questões de temas recentes;
- Janela de contexto de 8k – aproximadamente 24 páginas – o que significa que é menos provável que se sigam linhas mais longas e complexas.
- Pode não seguir instruções mais complexas, podendo resultar em respostas imprecisas ou pouco detalhadas.

4.4.7.2 GPT-4 Turbo [102]

- Dado que os dados do GPT4-Turbo são mais recentes, é viável obter resultados mais significativos e precisos comparativamente ao GPT4;
- Janela de contexto mais extensa: 128k;
- Melhor a nível de processamento e perceção de textos mais longos, realização de tarefas ou mesmo análise e sumarização de documentos;
- De acordo com informações da OpenAI, este modelo tem um melhor desempenho comparativamente a modelos anteriores para cumprir e desempenhar instruções;
- Existe a possibilidade de aceder a este modelo através do ChatGPT Plus ou Enterprise;
- Preço mais acessível que o GPT-4. Esta descida de preço deve-se ao facto de o modelo ser utilizado por programadores, logo quanto mais baixo o preço maior a utilização.

A principal discrepância entre os 2 modelos é a multimodalidade – isto porque até ao GPT-4 Turbo a entrada permitida pelos modelos era apenas baseada em texto [103]. Com o lançamento deste modelo houve mudanças, pois é possível inserir imagens no GPT-4 Turbo, realizando análise de conteúdo visual, documentos ou mesmo reconhecimento de textos em imagens.

Na Tabela 4-9 é possível comparar estes dois modelos.

Tabela 4-9- Comparação de GPT 4 com o GPT 4-Turbo

	<i>GPT-4</i>	<i>GPT 4-Turbo</i>
<i>Data de Atualização</i>	Setembro de 2021	Abril de 2023
<i>Janela de Contexto</i>	8k (aprox. 24 páginas)	128k (aprox. 300 páginas)
<i>Inputs</i>	Capacidade de lidar com tarefas complexas. Mas não de forma tão bem-sucedida quanto o GPT-4 Turbo	Capacidade de lidar com tarefas complexas de maneira mais eficaz
<i>Precisão</i>	Boa precisão de acordo com informações até 2021	Tem uma precisão maior pois têm dados mais atualizados
<i>Multimodalidade</i>	Suporta texto	Suporta texto, imagens e <i>text-to-speech inputs</i>
<i>Preço</i>	Mais caro (30\$ <i>Input</i> / 60 <i>Output</i>) por 1M <i>tokens</i>	Mais barato (10\$ <i>Input</i> / 30 <i>Output</i>) por 1M <i>tokens</i>
<i>Utilização</i>	Tarefas de GenAI	Tarefas de GenAI, mas também utilizada por programadores.

Através desta comparação entre os 2 modelos da OpenAI, observa-se que o GPT4-Turbo possui um conjunto maior de aspetos mais vantajosos. No entanto, não significa que o modelo GPT-4 seja obsoleto ou insuficiente a nível de potencialidades. A escolha de entre estes 2 modelos depende do tipo de informação, da utilização pretendida e dos requisitos específicos.

Como constatado, o caminho percorrido desde o GPT-1 até ao GPT-4 tem sido incansável e deveras inovador. O primeiro modelo apresentava somente 12 camadas, mecanismos de autoatenção para gerar texto com base na entrada que recebe. Mas também detinha dificuldades em lidar com tarefas mais complexas e generalizar para além do “aprendido” durante o treino. O GPT-2, mais robusto que o anterior, causou bastante agitação devido a preocupações de uso indevido. Assim, inicialmente foi mantido em confidência, mais tarde foi lançado demonstrando capacidade de geração de texto coerente e relevante de acordo com a entrada. O modelo seguinte: o GPT-3 apresentou impressionantes habilidades linguísticas. Por último, o GPT-4, com capacidades de manter muito mais contexto em conversas mais longas, é considerado um poderoso modelo.

4.5 Alucinações de LLM

Os LLMs trouxeram consigo um enorme potencial para realização de tarefas generativas, análise, processamento e geração de texto. No entanto, a sua utilização traz consigo grandes desafios como as alucinações. Alucinações que correspondem

a informações e respostas retornadas pelos LLM que, à primeira aparência parecem reais e plausíveis, mas na realidade são incorretas e sem sentido, mesmo sendo sintaticamente e semanticamente congruentes [104]. As informações prestadas podem ainda divergir das solicitações, de conteúdos anteriormente gerado ou do conhecimento tido como senso comum [105]. Esta adversidade ocorre em vários modelos de LLMs, inclusive no GPT-4, que é considerado um modelo com enorme potencial [106].

Este problema pode ser deveras desvantajoso, uma vez que podem ser difundidos factos irreais, discriminações ou preconceitos. Para além disso, o problema torna-se ainda mais significativo, pois os utilizadores tendem a confiar excessivamente nos resultados obtidos em vez de apurar a sua veracidade. Esta problemática associada aos LLMs, faz com que os programadores se sintam mais receosos e com a responsabilidade de construir mecanismos que controlem os resultados retornados pelos modelos. Em áreas com tarefas críticas, como a medicina, torna-se imprescindível validar as informações provenientes dos modelos, para minimizar ao máximo potenciais riscos para a sociedade.

Existem vários autores que se dedicam ao estudo e mitigação das alucinações com fim a diminuir e alertar para os problemas associados. Os autores referem 3 tipos de alucinações conhecidos [38]:

- **Inconsistência Factual** – quando as respostas provenientes dos LLMs são inconsistentes, factualmente incorretas e discrepantes do conhecimento real. Quando os modelos têm dificuldade de “lembrar” do conhecimento relevante passado na solicitação do utilizador [38]. Estas inconsistências podem ocorrer em diferentes estágios de vida dos LLMs.

Exemplo: afirmar que quem escreveu o Auto da Barca do Inferno foi Luis Vaz de Camões.

- **Inconsistência da Solicitação** – quando as informações retornadas não têm sentido de acordo com a solicitação feita. Ocorre quando o modelo responde de forma incorreta às informações fornecidas, devido a uma má interpretação da intenção do utilizador [38].

Exemplo: Quando o utilizador questiona acerca das causas de uma alergia e o modelo responde com informações acerca de gripes.

- **Desvio Contextual** – quando as respostas vão de encontro com o tópico, mas que não abordam de forma direta a solicitação, fornecendo respostas que não são relevantes [38].

Exemplo: Questionar um LLM acerca da altura do Monte Everest, e este explica como se formou o monte, onde se localiza, mas não aborda diretamente a altura.

Existem outros tipos de problemas associados aos LLMs [105]:

- **Ambiguidade** – quando a resposta retornada pode ser interpretada de várias maneiras, não sendo 100% útil e clara para o utilizador.

- **Incompletude** – quando a resposta gerada está incompleta, com omissão de detalhes ou informações cruciais.

- **Viés** – quando nas respostas obtidas se retrata discriminações e preconceitos, que podem ter origem nos dados de treino. Podem ser vieses de género, sociais ou estereótipos.

- **Subinformação** – dificuldade dos modelos em retornar respostas acerca de várias temáticas. Sendo esperado que tivessem capacidade para o fazer. Por exemplo, um excesso de RLHF (*Reinforcement Learning from Human Feedback*) pode levar a uma otimização excessiva do LLMs [105], [107].

- **Relevância Atribuída às informações em Longos Contextos** – o estudo [108] comprovou que o desempenho dos modelos tende a degradar-se de acordo com o local onde se encontram as informações mais importantes. Propõem que o desempenho se revela mais alto quando os dados importantes ocorrem no início ou no final do contexto de entrada. O desempenho diminui perante informações encontradas no meio de longos contextos.

Os autores também se demonstram focados na procura por técnicas de mitigação das alucinações presentes nos LLMs. Existem estudos que propõem uma análise de sistemas generativos de QA (*Question Answering*) no âmbito da medicina, focando-se na identificação de respostas problemáticas e alucinações através de uma metodologia interativa e de autorreflexão. Os autores pretendem através de um processo de *feedback* melhorar de forma progressiva a consistência das respostas geradas. São propostos 3 *loops* para minimizar os problemas provindos da incoerência dos modelos [106]:

Loop para aquisição de conhecimento factual -> o modelo constrói a resposta de acordo com a solicitação. O modelo usa a sua capacidade de sintetizar informações importantes de acordo com o contexto, treino e a solicitação. O *loop* tem como objetivo garantir que as informações fornecidas sejam factualmente corretas, servindo como base para as etapas seguintes de avaliação e refinamento. Nesta etapa a resposta gerada é avaliada tendo em conta a precisão. Se esta for de encontro com os padrões pré-definidos o processo pode avançar, caso contrário o modelo passará por uma fase de autorreflexão e refinamento. Esta estratégia é repetida iterativamente até que as informações geradas atinjam um nível satisfatório de precisão dos factos, promovendo um progressivo refinar dos factos gerados.

Loop de resposta consistente de acordo com o conhecimento -> processo iterativo que assegura que os LLMs sejam consistentes de acordo com o conhecimento existente. O modelo gera uma resposta inicial com base na solicitação, sendo esta avaliada de acordo com a consistência e relevância no contexto. Se a resposta atingir o nível desejado será continuado o processo, caso contrário o modelo necessitará de refinar progressivamente a resposta alinhando-a com informações tidas como reais, mantendo a integridade.

Loop de resposta com “Entailment- Questions” -> após a ocorrência dos 2 *loops* anteriores, é realizado um *loop* adicional para garantir que a resposta seja logicamente consistente. O *Entailment* corresponde à relação lógica entre 2 afirmações: uma hipótese e uma premissa, isto é, se a premissa implica logicamente a hipótese. São utilizados *embeddings* para verificar e avaliar o *entailment*. A resposta é pontuada, caso essa pontuação seja insatisfatória indica que a resposta não vai de encontro com os conhecimentos tidos como verdadeiros, e será retornado ao primeiro *loop*. Este *loop* tem como objetivo garantir que a resposta dada seja logicamente consistente e proveniente de informações verificadas.

Um estudo de 2023 refere que os problemas vindos dos LLMs advêm de 3 razões [105]:

- **Enormes volumes de dados de treino** – aquando do pré-treino dos modelos são utilizadas grandes quantidades de informações vindas da *web* o que torna difícil separar informações inventadas, desatualizadas ou mesmo tendenciosas. O que aumenta a probabilidade de os modelos gerarem conteúdo alucinado;

- **Versatilidade dos LLMs** – os modelos têm capacidade de atuar em diversas tarefas e domínios linguísticos. Mas esta versatilidade representa um enorme desafio para a avaliação e para mitigação de alucinações nos LLMs. Como é de conhecimento, os LLMs são muito utilizados na geração de texto, mas a falta de referências complica a deteção de alucinações, pois não há um padrão para comparação da precisão da resposta;

- **Impercetibilidade de erros** – como visto, os LLMs podem gerar informações falsas mesmo que pareçam muito verídicas e plausíveis. Assim torna-se bastante difícil para modelos e para humanos detetarem alucinações;

- **Processo de RLHF** – estratégia utilizada para mitigar as alucinações nos LLMs, envolvendo a revisão e o refinar das respostas por parte de humanos especializados antes de serem divulgados. Apesar de ser uma estratégia valiosa para mitigar alucinações nos LLMs também requer bastantes recursos humanos e financeiros, especialmente se a “mão de obra” for especializada. Para além disso, essa revisão está associada a uma enorme subjetividade, preconceitos, erros e perspetivas pessoais que podem enviesar. Todo o processo pode ser demorado, especialmente aquando de enormes volumes de texto, resultado em atrasos na geração e publicação de conteúdos;

- **Limite de Conhecimento Vago** – os LLMs têm um limite de conhecimentos. Embora sejam treinados com grandes quantidades de dados, podem ter dificuldades em lidar com informações ambíguas e mal definidas proporcionando respostas imprecisas e alucinadas;

- **Propriedade black-box dos modelos** – refere-se à falta de transparência dos modelos relativamente ao processo de tomada de decisão. Desta forma, os utilizadores não têm *insights* acerca do caminho percorrido até à resposta fornecida.

De acordo com estas desvantagens / limitações há sempre espaço para melhorias e mitigações.

4.6 Retrieval-Augmentend Generation (RAG) Applications

Existem várias técnicas já estudadas que auxiliam no mitigar das alucinações dos LLMs, como podemos verificar na Figura 4-4.

Desenvolvimento de um protótipo de um Futebol GPT

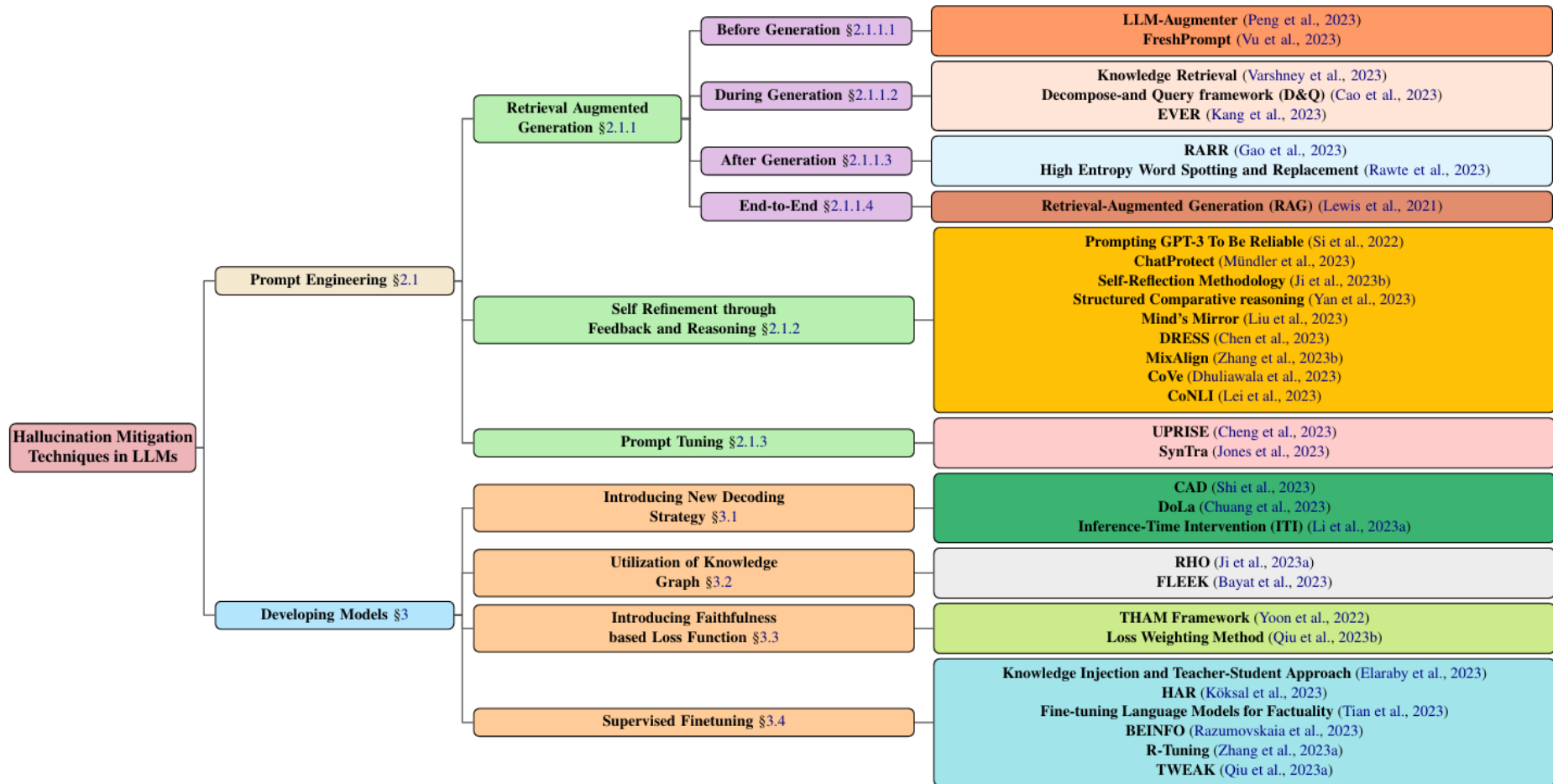


Figura 4-4 - Taxonomia das técnicas de mitigação de Alucinação dos LLMs[106]

Neste subcapítulo será abordada uma técnica de mitigação em específico: *Prompt Engineering*, uma das mais conhecidas e abordada.

Os modelos de GenAI foram desenvolvidos com o propósito de gerar saídas com base nos *prompts* fornecidos. A Engenharia de *Prompts* tem como foco auxiliar esses modelos na compreensão e na geração de respostas adequadas para uma ampla gama de temas [109]. Esta técnica envolve a geração de *prompts* que forneçam o contexto suficiente ao modelo para que este compreenda e responda de forma adequada. O *prompt engineering* compreende várias técnicas como tokenização, ajuste de parâmetros ou mesmo RAG.

A qualidade de um *prompt* influencia a qualidade do conteúdo gerado pela IA: “Bons *prompts* promovem bons resultados”. Os modelos de GenAI dependem bastante do aperfeiçoamento iterativo das diversas técnicas de *Prompt Engineering* para desenvolver capacidades de “aprender” com os dados de entrada e por conseguinte minimizar preconceitos e vieses. Com o ajuste dos *prompts*, é possível otimizar de forma considerável a qualidade e a pertinência dos *outputs* gerados. Desta forma é possível reduzir a revisão manual e a edição pós geração, economizando tempo e recursos. Por exemplo, em modelos de geração de imagens através de texto como o Midjourney ou o DALL-E, é importante que os *prompts* sejam bem entendidos para que a imagem gerada vá totalmente de encontro com a solicitação do utilizador. Os modelos combinam conhecimento técnico com perceção da linguagem natural para que sejam produzidas saídas corretas e pertinentes ao *prompt*. Nesta temática, é importante reforçar que aprimorar *prompts* no modelo GPT-3 ou no GPT-4 é diferente de melhorar *prompts* no Google Bard [110], [111]. Esta diferença reside nas diferenças das arquiteturas dos modelos. Os GPTs utilizam uma arquitetura baseada em *Transformers*, os quais possuem dados de treino estáticos (adquiridos até uma determinada data) e as respostas são geradas com base em probabilidades. O Bard, utiliza uma arquitetura LaMDA, desenvolvida para compreender conversações complexas e contínuas e tem suporte uma base de conhecimentos mais ampla e capacidade de pesquisar na *web* em tempo real. Desta forma tem capacidade de fornecer respostas mais completas e atualizadas.

Existem várias técnicas para melhorar a compreensão e a qualidade da saída do modelo:

Prompting zero-shot – fornece ao modelo uma tarefa para a qual ele não foi treinado [112]. Este *prompting* testa a capacidade do modelo em gerar saídas relevantes sem que tenha de depender de exemplos [113]. O modelo deve inferir a resposta com base apenas na instrução fornecida sem qualquer treino prévio sobre a tarefa em específico.

Por exemplo, ao questionar o modelo “Qual a capital de Portugal?”, ele utilizará o seu conhecimento geral que adquiriu ao longo do treino para inferir a melhor resposta, mesmo que não tenha sido treinado para responder em específico a esta questão.

Esta técnica possibilita que o modelo generalize o conhecimento e desenvolva novas habilidades sem que seja necessário passar-lhe exemplos de cada tarefa.

Prompting few-shot – são fornecidos ao modelo alguns exemplos de saída (denominados *shots*) para auxiliar a apreender aquilo que foi solicitado pelo utilizador [114]. Esses exemplos representam formas de realizar uma dada tarefa, e ajudam a orientar o modelo, dando-lhe uma ideia mais clara do que se espera encontrar na resposta [115].

Por exemplo, utilizar um modelo para realizar traduções de português para inglês. Em vez de simplesmente pedir ao modelo que traduza uma frase sem contexto, poderão ser passados alguns exemplos de traduções:

“Olá” -> “Hello”

“Adeus” - > “goodbye”

“obrigado” -> “Thank You”

E depois pede-se que traduza a frase “How are you” o modelo deverá responder “Como estás?”

Prompting chain-of-thought (CoT) – técnica que oferece um raciocínio passo a passo para o modelo seguir [116]. As tarefas são divididas em etapas intermediárias ou “cadeias” que ajudam o modelo a obter uma melhor compreensão da linguagem e a criar *outputs* mais precisos e confiáveis. Esta abordagem demonstra ser útil para tarefas complexas e que exigem raciocínio e lógica.

Por exemplo: “Se a Rosa tem 3 berlindes, foi à feira e comprou mais 4. No dia seguinte deu 2 à irmã. Quantos berlindes a Rosa tem no total?”

O problema poderia ser resolvido passo a passo:

A Rosa começa com 3 berlindes.

Comprou mais 4 berlindes.

Neste momento tem 7 (3+4).

Entretanto, deu 2 à irmã.

$7-2=5$. A Rosa tem 5 berlindes.

Espera-se que o modelo ao ter um raciocínio semelhante produza na saída: 5 berlindes.

Um conceito bastante relevante no contexto da *prompt engineering* são RAG Applications [106], algo que foi estudado e analisado neste subcapítulo. RAG Applications ou *Retrieval Augmented Generation* são sistemas de Inteligência Artificial que unem métodos baseados em recuperação com modelos baseados em geração, para aprimorar o desempenho e a precisão de tarefas de PLN [117]. Esta abordagem permite melhorar os LLMs com informações externas, tornando as respostas às

questões, mais precisas, relevantes e contextualizadas, de encontro a conhecimentos mais específicos. A título de exemplo: realizar questões específicas ao LLM sobre uma organização acerca de assuntos internos não é útil e não produzirá resultados, pois o modelo não tem informações tão confidenciais. Uma solução ideal para adaptar os LLMs a essas necessidades tão intrínsecas poderá passar pela utilização de Graph RAG Apps. Estas utilizam o conceito de recuperação e geração como as RAG Apps, mas organiza os dados no formato de grafo [118]. Esta abordagem permite reduzir a tendência dos LLMs para alucinações, pois a geração de conteúdos é mais confiável. Este tipo de aplicações pode ser utilizado para tarefas de PLN, extração de informações, análise de sentimentos e sumarização. As RAG Apps são compostas por 2 componentes [119]:

- Componente de Recuperação

O objetivo deste componente é pesquisar documentos ou segmentos de informações relevantes numa enorme base de dados ou corpus externo [120]. Quando é realizada uma solicitação ao componente, este investiga os textos mais pertinentes que se adequem ao pedido ou propiciem mais contexto [121], [122]. Esta vertente pode envolver algoritmos de pesquisa tradicionais (através de uso de índices / “listas de palavras-chave” associados aos documentos que permitem uma busca rápida) ou algoritmos mais avançados, como *dense retrieval*, que faz uso de redes neuronais para assimilar melhor a semântica das solicitações e das informações em registos. Em vez de procurar as palavras-chave, o significado das frases é subentendido, permitindo uma correspondência mais precisa entre o pedido e os dados. Este tipo de algoritmo é útil aquando de buscas que envolvam solicitações complexas ou amplos contextos, pois tem capacidade de encontrar dados semanticamente relacionados, mesmo que as palavras exatas não coincidam. Por exemplo, numa base de dados de artigos científicos acerca de agricultura de precisão, um componente de recuperação com *dense retrieval* poderia analisar a solicitação e entender o interesse em “agricultura”, “drones” ou “mudanças climáticas”. No entanto, não basta procurar pelas palavras exatas, mas sim por documentos que refiram aspetos relacionados, por exemplo: “aquecimento global na agricultura”.

-> Componente de Geração

O objetivo deste componente passa por criar respostas coerentes e contextualmente congruentes com base nos dados recuperados pelo componente de recuperação e o próprio conhecimento do modelo [123]. Este componente utiliza modelos de PLN, como os GPTs, para entender a solicitação e as informações textuais fornecidas. O componente, ao receber a solicitação do utilizador e o *output* do componente de recuperação, processa as informações para garantir que estas se encontram num formato que o modelo entenda. Assim será feita a extração de ruído, normalização de texto e o processo de tokenização. Posteriormente estas informações são codificadas em vetores de alta dimensão, que extraem o significado semântico do texto, através de *embeddings* criados pelo LLM. O modelo usa esses vetores para criar uma sequência de palavras, através da composição de palavras uma a uma, onde cada

léxico tem em consideração os anteriores: formando a resposta, que ainda será refinada a nível de correção gramatical. Associados a este processo, existem considerações para garantir que as respostas sejam coerentes, relevantes e sem ambiguidades tendo em conta a solicitação e as informações recuperadas.

Para utilizar as Graph RAG Apps para âmbitos de pergunta-resposta é importante selecionar as informações a enviar para o LLM, geralmente armazenadas em bases de dados vetoriais [124]. Estas, armazenam os dados em *embeddings* vetoriais: representações numéricas de objetos (palavras, imagens) que auxiliam na procura num conjunto de dados não estruturados ou semiestruturados. Desta forma os dados são organizados através de vetores de alta dimensão, que ao conterem centenas de dimensões permitem representar as diferentes propriedades de um objeto. Os *embeddings* vetoriais são gerados por LLMs ou outros modelos de IA, e permitem extrair o significado semântico de estruturas sintáticas e as relações entre os vários elementos num espaço vetorial. A distância entre cada *embedding* permite que a base de dados estabeleça a similaridade entre vetores, que propicia a perceção de relações e padrões, muito úteis em *Machine Learning*.

As Graph RAG Apps constroem um grafo onde os nós representam os documentos / informações e as arestas representam relações entre eles. Aquando de uma solicitação, o sistema recupera os dados mais relevantes do grafo, e através dessa informação gera uma resposta coerente e informativa.

É importante referir que existem diferentes formas de aprimorar o LLM para além da utilização de Graph RAG Apps, por exemplo: técnicas de *fine-tuning*, onde o modelo é treinado novamente com *corpus* específico com fim de adicionar conhecimento específico, ou através da injeção de conhecimentos (enriquecer o modelo com informações estruturadas).

4.6.1 Comparação entre utilização de Graph RAG e outras melhorias do LLM [125] :

Graph RAG Apps:

- Esta abordagem envolve a arquitetura e manutenção de um grafo de conhecimento e um sistema de recuperação eficaz;
- Embora exista um custo inicial na conceção da base de conhecimento, este dispêndio é menor comparativamente ao treino contínuo do LLM;
- Na construção da resposta, a abordagem requer menos poder de computação, pois a quantidade de informações a processar é relativamente menor;
- A atualização do conhecimento é muito mais fácil e acessível, dado que não existe a necessidade de voltar a treinar o modelo;
- As respostas fornecidas são mais precisas e baseiam-se em dados de domínios individuais, sendo mais contextualizadas, relevantes e atualizadas;

- São mais económicas em âmbitos onde, informações são atualizadas frequentemente e em enorme quantidade.

Fine-Tuning ou Injeção de Conhecimento:

- Ao deter enormes *corpus*, estas abordagens podem revelar-se extremamente caras, demoradas e que necessitem de um elevado poder computacional;
- Aquando da atualização do conhecimento inerente ao modelo é necessário realizar um novo treino, o que é dispendioso e em muitos casos não é escalável;
- Pode ter alta precisão caso o modelo seja treinado com um grande conjunto de dados, mas a precisão pode ser perdida se o conhecimento não for atualizado frequentemente;
- Pode ser desvantajoso em domínios dinâmicos ou em áreas de conhecimento que se encontrem constantemente evolução.

A escolha entre uma das duas abordagens depende de vários fatores, nomeadamente: recursos disponíveis, exigência de atualização de conhecimento e do domínio em específico. No caso do projeto SportGPT, foi utilizada a abordagem de RAG Application dada a necessidade de atualização diária dos dados, pela eficiência computacional necessária e pela existência de dados futebolísticos já em contexto da empresa.

4.6.2 Vantagens das RAG Apps:

- Maior precisão – através da utilização de recuperação, o modelo de GenAI tem uma maior probabilidade de gerar respostas corretas, precisas e relevantes;
- Eficiência – é possível reduzir a necessidade de procurar e agregar todas as informações manualmente (o que iria demorar muito mais tempo);
- Contexto – desta forma é mais fácil existir manutenção do contexto, mesmo em solicitações complexas.

4.6.3 Desvantagens / desafios das RAG Apps:

- Escalabilidade – recuperar documentos importantes e muito grandes, pode ser dispendioso;
- Integração de modelos – integrar outros componentes de recuperação e geração pode ser tecnicamente desafiador;
- Qualidade dos dados – a precisão dos resultados depende veemente da qualidade dos dados no conjunto de recuperação.






5 TECNOLOGIAS







Neste capítulo serão apresentadas as tecnologias e linguagens utilizadas no desenvolvimento do projeto SportGPT. As informações encontram-se descritas na Tabela 5-1.

Tabela 5-1 - Tabela de Ferramentas e Linguagens Utilizadas

<i>Ferramenta</i>	<i>Descrição</i>	<i>Utilização no SportGPT</i>
 <i>Visual Studio Code</i>	Ferramenta de edição de código fonte, desenvolvida pela Microsoft [126]	Ferramenta utilizada para escrita de código associado ao servidor, criação de <i>scripts</i> essenciais para o projeto, código relativo ao estudo das <i>frameworks</i> e desenvolvimento de código para previsão de resultados futebolísticos.
 <i>GitHub</i>	Plataforma de alojamento de código fonte, e controlo de versões com base em Git [127]	Esta plataforma permitiu alojar num repositório, o código correspondente ao servidor e à aplicação <i>mobile</i> do SportGPT, permitindo acompanhar o seu versionamento.
 <i>Postman</i>	Ferramenta de teste de APIs de forma eficiente [128]	Útil para entender o formato de resposta obtido utilizando o LLM (GPT) em contexto de uma aplicação desenvolvida na WIT, mas direcionada para filmes.
 <i>Playground</i>	Plataforma da OpenAI [129]	Testar como as informações da <i>System Message</i> ² conseguem produzir resultados. Análise dos custos associados aos pedidos.
 <i>Neo4J Desktop</i>	Interface gráfica do Neo4J que permite gerir a base de dados de forma direta e fácil de interpretar.[130]	Tornou acessível a criação/ gestão da instância do Neo4J local. Possibilitou uma compreensão detalhada e dinâmica dos dados e uma melhor interatividade e compreensão da estrutura dos nós e relacionamentos. Deste modo, todo o processo foi mais intuitivo e eficiente.

² System Message – diretrizes internas que são fornecidas ao modelo (LLM) para modelar o seu comportamento e respostas. Estas instruções não são visíveis para o utilizador.

 <p>PgAdmin</p>	<p>Interface gráfica para auxílio da gestão, administração e desenvolvimento de bases de dados PostGreSQL [131]</p>	<p>Esta ferramenta foi essencial para criar, analisar as tabelas de dados futebolísticos. Assim, foi mais fácil compreender os atributos e tabelas relevantes e estruturá-las de forma eficiente. Ainda foi possível criar de forma rápida novas colunas para corresponder às necessidades do projeto.</p>
 <p>Jupyter Notebook</p>	<p>Aplicação <i>web</i> para criação de documentos de código executável e visualizações [132]</p>	<p>Utilizada para desenvolvimento e teste de modelos preditivos para previsão de resultados futebolísticos.</p>
 <p>NGrok</p>	<p>Ferramenta que permite criar túneis seguros de forma a expor servidores locais para a Internet. Atribui um URL público temporário [133]</p>	<p>Permitiu partilhar de forma temporária o servidor local com outras pessoas pela Internet, sem ter de configurar <i>firewalls</i> ou DNS (<i>Domain Name System</i>). Útil para questões de demonstrações.</p>
 <p>Discord Teams Rocketchat Zoom Email</p>	<p>Ferramentas de comunicação e colaboração, essenciais para a interação entre os diferentes membros num ambiente de trabalho</p>	<p>Comunicação entre membros da empresa, <i>meetings</i>.</p>
 <p>Google Drive</p>	<p>Serviço de armazenamento na <i>cloud</i> que permite o armazenamento de documentos <i>online</i></p>	<p>Para partilha de documentação do estágio, relatórios, <i>power points</i>, entre outros.</p>

 <i>Excel</i>	<p><i>Software</i> de folhas de cálculo utilizado para a análise de dados, visualizações de dados e gestão de informações</p>	<p>Ferramenta eleita para criar e analisar um conjunto de perguntas relacionadas com o contexto de futebol e avaliar a sua evolução.</p> <p>Colmatar em tabelas os resultados obtidos através da utilização de vários algoritmos de <i>Machine Learning</i> para a realização de <i>prediction</i> de resultados futebolísticos.</p>
 <i>Word</i>	<p><i>Software</i> de processamento de texto utilizado para criar, editar e formatar documentos</p>	<p>Escrita de documentação, por exemplo: relatório de estágio, informações provindas de pesquisa, lista de funcionalidades da <i>app mobile</i>, entre outros.</p>
 <i>PowerPoint</i>	<p><i>Software</i> de apresentação utilizado para criar apresentações</p>	<p>Desenvolvimento de apresentações para a empresa e para o mestrado.</p>
<p>Linguagens</p>		
 <i>Python</i>	<p>Linguagem de programação de alto nível, de sintaxe simples e com uma vasta biblioteca [134]</p>	<p>Utilizada para desenvolvido de código respetivo ao servidor, previsões futebolísticas e <i>scripts</i> essenciais ao projeto.</p>
 <i>Flask</i>	<p><i>Framework</i> de desenvolvimento <i>web</i> em Python, simples e de fácil uso [135]</p>	<p>Foi essencial no desenvolvimento da <i>web app</i> utilizada em questões de demonstrações, permitindo a sua implementação de forma simples, eficaz e ágil.</p>
 <i>Flutter</i>	<p><i>Framework</i> de desenvolvimento de aplicações <i>mobile</i>, desenvolvido pela Google [135]</p>	<p>Utilizada para o desenvolvimento da <i>app mobile</i> de <i>chat</i> de perguntas futebolísticas</p>

6 METODOLOGIA

Ao longo do estágio curricular, adotou-se uma metodologia estruturada para garantir o progresso contínuo e eficaz do projeto. A abordagem colaborativa com várias equipas, realização de reuniões diárias e mensais contribuiu para o sucesso do estágio.

Reuniões Diárias

Para assegurar que o progresso existia, resolver possíveis bloqueios, foram realizadas reuniões diárias. Nestas reuniões era discutido o trabalho realizado, o que estava a ser realizado, o que estava por realizar e ainda impedimentos e desafios encontrados. Esta prática foi importante para manter o estágio alinhado, facilitando a rápida evolução, o alinhamento de ideias e resolução de problemas.

Reuniões Mensais

Mensalmente, a empresa organizava uma reunião mais abrangente, envolvendo várias pessoas da organização. Nestas reuniões eram apresentados os avanços do mês, problemas solucionados e aqueles ainda por resolver. O objetivo era obter *feedback* construtivo e relevante, o que permitia enriquecer o projeto, perceber diferentes pontos de vista e orientar para próximos passos.

Nestas ocasiões eram feitas demonstrações e colhidos *insights* valiosos para melhoria contínua do trabalho.

Acompanhamento

Durante o estágio, para além do acompanhamento realizado pelo tutor e pelo orientador, foram envolvidas pessoas de equipas especializadas, assim que necessário, nomeadamente aquando de bloqueios.

Equipa de GenAI – essencial para lidar com os modelos de linguagem, auxiliando na superação de bloqueios, otimização de tarefas e implementação mais eficiente das soluções baseadas em LLMs. A equipa ajudou a resolver problemas complexos relacionados com esses modelos.

Equipa de Machine Learning – forneceu suporte essencial na previsão de resultados futebolísticos, ajudando a desenvolver os modelos preditivos e fazer ajustes que resultaram em melhores resultados.

Designer – a colaboração com a *designer* foi um componente chave na criação de *mockups* visuais que facilitaram a implementação da aplicação *mobile* do projeto SportGPT.

A metodologia adotada durante o estágio, combinada com uma abordagem colaborativa e reuniões regulares, garantiu o progresso contínuo e eficaz do projeto. O auxílio de equipas especializadas e o suporte constante do tutor e orientador foram fundamentais para superar desafios e otimizar os resultados, culminando num estágio bem-sucedido.

7 SPORTGPT

Neste capítulo será realizada uma descrição do desenvolvimento do projeto desenvolvido no âmbito do estágio curricular: o SportGPT, na empresa WIT Software. Este projeto tem como principal objetivo a construção de um sistema de GenAI com capacidade de “compreender” as questões futebolísticas introduzidas por utilizadores. Utilizando PLN e LLMs será gerada uma *cypher query* [136], tendo como base a descrição do esquema da base de dados interna que armazena dados de futebol. A execução desta *query* retornará resultados atualizados e precisos, de acordo com a questão introduzida. De seguida, recorrendo novamente ao LLM será gerada uma resposta, em linguagem natural, humanizada e coerente, semelhante a uma resposta fornecida por um humano.

7.1 Análise de Requisitos

A aplicação do projeto SportGPT, foi projetada para que os entusiastas de futebol possam realizar as suas questões futebolísticas e obter respostas em tempo real, coerentes consoante a pergunta efetuada. Nesta subsecção serão apresentados os requisitos funcionais e não funcionais da aplicação.

7.1.1 Requisitos Funcionais

- Os utilizadores poderão fazer perguntas relacionadas com futebol através de um campo de texto;
- O sistema utiliza um LLM para entender a questão introduzida e responder de forma adequada em tempo real;
- A resposta fornecida deve ser precisa e contextualmente relevante;
- O sistema deverá ter a capacidade de lidar com diferentes perguntas futebolísticas: jogadores, treinadores, jogos, equipas e competições;
- O sistema deverá aceitar perguntas da atualidade, passado e futuro;
- A interface deve ser intuitiva e simples, tanto a nível de *web apps* construídas para demonstrações como para *app mobile*.

7.1.2 Requisitos Não Funcionais

- O sistema deve ser capaz de lidar com múltiplos utilizadores em simultâneo sem que seja posto em causa o desempenho;
- O utilizador deve ter a capacidade de entender o que está a acontecer à medida que interage com o sistema.

7.2 Fontes de Dados

A ideia principal da aplicação centra-se na funcionalidade de *chat*, onde os utilizadores podem colocar as suas questões futebolísticas e obter uma resposta em tempo real. Compreende-se que cada utilizador detém um conjunto díspar de dúvidas. Assim, a posse de uma vasta gama de dados futebolísticos torna-se essencial, com fim a abranger um amplo espectro de tópicos futebolísticos, de entre o intervalo temporal estipulado inicialmente na proposta de estágio (2016-2022). A procura focou-se essencialmente em APIs gratuitas e *datasets* públicos (nomeadamente do Kaggle [16]) encontrando informações correspondentes a ligas internacionais [17], Mundiais [18] e futebol europeu [19]. A análise efetuada às várias fontes de dados encontra-se em anexo: Q.

7.3 Arquitetura

Neste subcapítulo será introduzida a arquitetura definida para o desenvolvimento do projeto SportGPT conforme ilustrado na Figura 7-1.

Desenvolvimento de um protótipo de um Futebol GPT

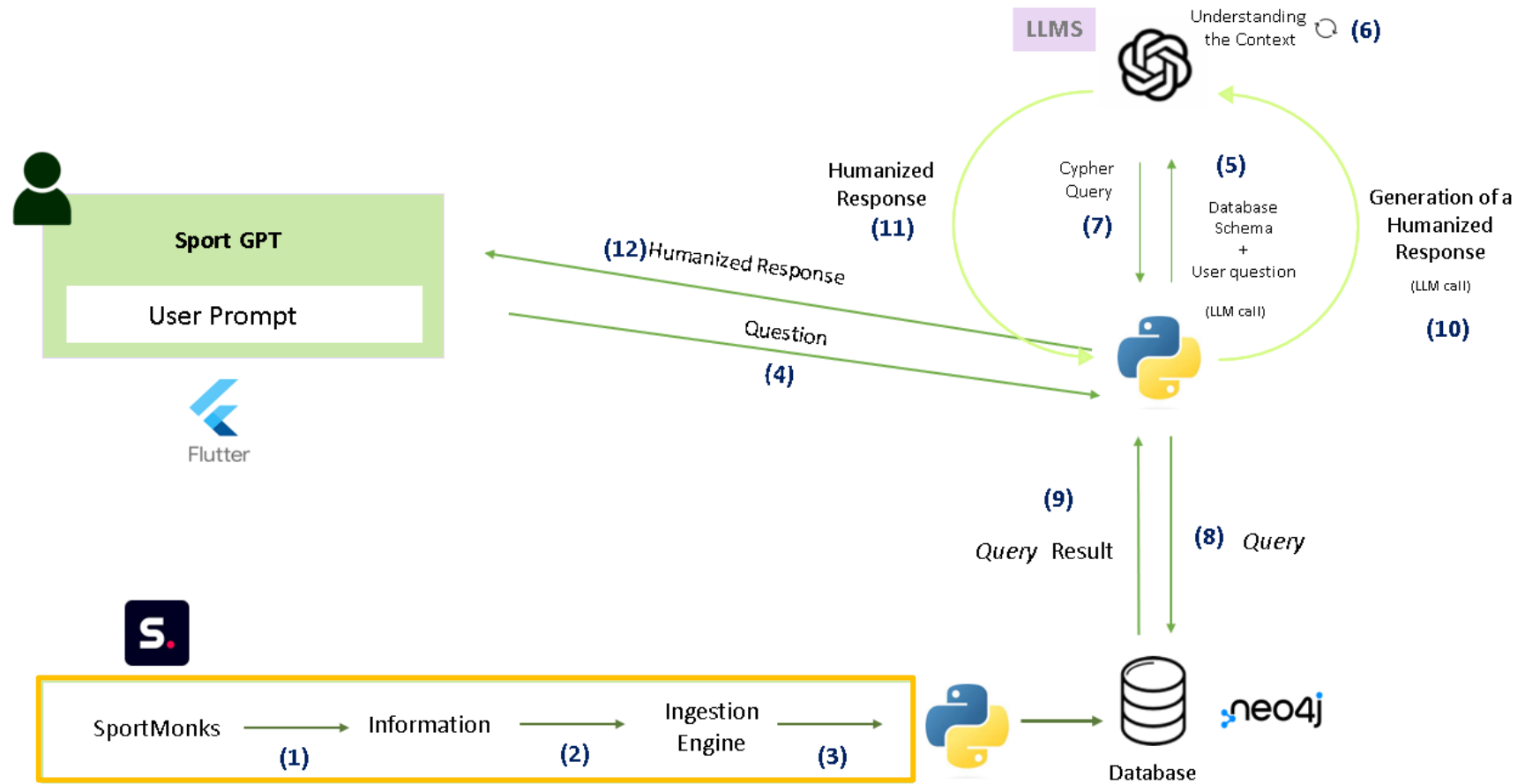


Figura 7-1 - Arquitetura do SportGPT

Esta arquitetura é composta por dois módulos cruciais, onde cada um desempenha um papel fundamental no sistema:

1- Recolha e Processamento de Dados – O primeiro momento demarcado com caixa laranja, representa o fluxo onde as informações são recolhidas a partir de uma API, no caso: a do SportMonks [137], sendo posteriormente processadas e armazenadas na base de dados Neo4J. Este processo é fundamental para garantir a disponibilidade dos dados nas consequentes etapas.

2 – Interação e Conversação – O segundo momento corresponde à interação do utilizador com o *chat*. O utilizador introduz a sua questão futebolística num *user prompt* ou por voz. Esta questão vai ser alvo de uma LLM Call (interação com o modelo de linguagem), para que o contexto da pergunta seja apreendido. O LLM possui informações relativas ao esquema da base de dados: nós, atributos e as respetivas descrições, como é possível analisar nos anexos em C. Com base nestas informações e no contexto da questão será gerada uma *cypher query* de uma de duas formas. De forma direta, com o seu próprio conhecimento ou em caso de incapacidade recorrendo a OpenAI *function calling* (o modelo GPT permite ter um conjunto de funções previamente definidas que permitem gerar saídas personalizadas [138]). A *query* será executada na base de dados, sendo o resultado obtido alvo de uma nova LLM Call, para harmonizar e tornar a resposta mais agradável aos olhos do utilizador. Esta será disponibilizada tanto por texto como por voz (no caso da *web page*).

O modelo GPT-4 utiliza um *endpoint* ChatCompletion [139] onde existe uma combinação de mensagens:

System Message – mensagem escrita pelo programador para especificar ao LLM o seu comportamento. São instruções de como o *bot* deve interpretar a conversação;

User Message – mensagens enviadas pelo utilizador ao LLM;

Assistant Message – mensagens retornadas pelo LLM.

Ao longo do *flow* são realizadas várias LLMs *calls*, sendo necessário distinguir os diferentes *roles* envolvidos como podemos verificar na Tabela 7-1.

Tabela 7-1 - Descrição dos Papéis envolvidos

	<i>Role</i>	<i>Descrição / Conteúdo</i>
<i>S</i>	System	- Informações relativas ao comportamento do <i>chatbot</i> - Esquema da base de dados contendo os nós, entidades e respetivas descrições de forma que o LLM detenha todas as informações relevantes - Exemplos de algumas <i>Cypher Queries</i>
<i>U</i>	User	<i>Role</i> correspondente às <i>prompts</i> enviadas pelo utilizador (as suas questões)
<i>A</i>	Assistant	Contém as respostas retornadas por parte do LLM

F | Function Função utilizada pelo LLM para geração da *query*

Utilização de cada *role* de acordo com a arquitetura proposta:

Passo (5):

Role de System, onde são passadas ao LLM as informações relativas ao seu comportamento, esquema da base de dados, respetivas descrições e exemplos de *Cypher queries* para servir de base para a construção das *queries*;

Role User, onde é enviada a questão realizada pelo Utilizador;

Através da utilização da OpenAI *function calling* [138], é feita uma chamada ao *Assistant*, de forma implícita, para entender se alguma das funções pré-definidas vai de encontro com a questão introduzida pelo utilizador;

Caso alguma das funções se adeque, o modelo assume o *role Function*, gerando uma chamada à função, com os argumentos necessários. Exemplo de interação:

```
{  
  "role": "function",  
  "name": "get_current_team_of_a_player",  
  "arguments": {  
    "player": "Cristiano Ronaldo",  
  }  
}
```

Um exemplo mais completo desta interação, é apresentado no anexo R.

A função é executada com os parâmetros apropriados, retornando a *query* correta a executar na base de dados. No histórico da conversação é armazenada uma entrada com *role "function"* com o *id* da chamada ao LLM, o nome da função invocada, e a *query* a executar. Caso nenhuma das funções definidas vá de encontro à questão do utilizador, será o LLM a gerar a *query* através das informações passadas na *System Message*. Para “simular” que esta *query* foi construída de forma semelhante à utilização de *function calling* foram adicionados os *roles* de forma semelhante ao histórico de conversação anteriormente descrito:

Role Assistant – cujas informações adicionadas ao histórico da conversação são: *id* da chamada, *function_name default* (“bd_function” no caso) e a *query* a executar.

Role Function – as informações guardadas neste *role* são: *id* da chamada, *function name default* e a *query* a executar.

Passo (11):

Para finalizar, é realizada uma nova chamada ao LLM com objetivo de humanizar a resposta proveniente da base de dados. O *role Assistant* é então armazenado na variável de histórico apenas com a resposta já humanizada e o *id* da chamada.

A arquitetura descrita viabiliza uma interação fluída e eficaz entre o utilizador e o sistema / LLM. Como referido anteriormente, dois dos principais problemas enfrentados ao utilizar LLMs é a incerteza associada e a limitação temporal de dados. Ao questionar *chatbots* relativamente a um jogo de futebol que ocorreu no último mês, não obteremos uma resposta. Assim, provamos a relevância da solução proposta, pois ao existir recolha periódica dos dados de uma API conseguimos manter os dados atualizados e responder às perguntas feitas pelos utilizadores de forma correta e precisa.

7.3.1 Escolha do Tipo da Base de Dados

A utilização direta de LLMs para resposta a questões apresenta algumas limitações, nomeadamente a nível de imprecisão de resultados e restrição temporal [140], [141]. Desta forma, ao determos a nossa própria base de dados, propomos uma forma de mitigar estes problemas, obtendo maior veracidade e dados atualizados. No início da implementação do projeto SportGPT, foi fornecido um *dump* de dados futebolísticos, datado de 15 de novembro de 2023, primeiramente inserido numa base de dados PostgreSQL [142], devido à sua natureza relacional. Este conjunto inicial continha 76 tabelas, e após uma análise minuciosa foi possível compreender como as tabelas se interconectavam e como os dados se encontravam distribuídos. As informações mais relevantes foram identificadas, bem como aquelas irrelevantes para o contexto em causa. Com base nesses *insights* foi possível proceder à remoção de variáveis e tabelas desnecessárias (resultando 43 tabelas) e as variáveis JSON foram transformadas em atributos diretos das tabelas, para facilitar o acesso a esses dados.

Considerando os objetivos do projeto SportGPT, foram ponderadas questões relevantes tanto para o trabalho atual como para desenvolvimentos futuros: a necessidade de atualização periódica dos dados (para que as respostas fornecidas sejam o mais atuais possível). Relativamente à futura integração com fontes de dados, era requerida uma base de dados que permitisse armazenar dados em diversos formatos e que suportasse alterações frequentes. Assim, uma base de dados de esquema rígido como as relacionais não era adequada, tendo-se optado por realizar um estudo de bases de dados não relacionais. Após a pesquisa, dos 4 tipos de bases de dados não relacionais, decidimos excluir as bases de dados do tipo *key-value*, pois de acordo com a literatura não ofereciam suporte para *queries* muito complexas, e restringem a quantidade de dados que conseguem filtrar. Estes aspetos não eram vantajosos no caso do projeto SportGPT, dada a necessidade de interligar várias tabelas e filtrar diversos dados, por exemplo, para obter o número de vezes que uma equipa ganhou numa determinada época e em várias competições, são necessárias 7 tabelas.

As bases de dados do tipo *Column* também foram descartadas, pois apesar de permitirem otimização do desempenho, necessitariam de desnormalização dos dados, o que provocaria duplicações e conseqüente aumento do armazenamento.

Este tipo de base de dados também não é adequado para pesquisas complexas, que envolvam vários *joins* e agregações sofisticadas o que poderá ser problemático em operações intensas de escrita. Estes dois cenários são bastante importantes no caso do projeto SportGPT.

Inicialmente, foram escolhidos 2 tipos de bases de dados como possíveis candidatas a utilizar: bases de dados *document* e grafos. O primeiro tipo foi avaliado devido à sugestão de integração com uma API de dados futebolísticos: SportMonks [143], onde os dados eram retornados em formato JSON. Ao utilizar uma base de dados *document* evitava-se a necessidade de conversões complexas ou mapeamentos para armazenar as informações. Os documentos JSON também possibilitariam inserir novos atributos sem haver a imposição de alterar a estrutura da base de dados.

Por outro lado, as bases de dados de grafos permitem que os dados sejam armazenados consoante um grafo, isto é, através de entidades e relacionamentos entre si. Estes aspetos são relevantes no contexto em causa: um jogador pertence a uma equipa, uma equipa tem um estádio, um jogo ocorre entre duas equipas. Após uma reunião com a equipa de GenAI, foi possível entender que já existia um projeto em desenvolvimento em moldes semelhantes: utilizar uma base de dados de grafos de teor cinematográfico e um LLM para geração *queries* de forma a ter resultados mais precisos e pertinentes. Neste projeto a base de dados utilizada era o Neo4J. Durante a reunião ficou claro que a base de dados mais adequada seria uma base de dados de grafos dada a forte inter-relação entre os dados futebolísticos. A base de dados eleita foi o Neo4J, não só por ser uma base de dados já em uso na empresa, mas por ser a mais utilizada, segundo o *ranking* da *dbEngines*. O Neo4J é reconhecido pela sua robustez e confiabilidade, sendo otimizado para pesquisas complexas em grafos e com alto desempenho para procura de dados em grandes conjuntos. A base de dados promove também a utilização de algoritmos que garantem a eficiência tanto nas operações de escrita como de leitura. O *cypher*, a linguagem de consulta desta base de dados, demonstra ser bastante intuitiva, o que facilita a rápida aprendizagem e a modelação de relações complexas entre os dados [86]. A base de dados também oferece suporte para transações ACID, garantindo que a integridade dos dados é mantida, mesmo em aplicações mais críticas. Assim demonstrou ser adequada para o propósito: o projeto SportGPT.

7.4 Integração com Frameworks

Atualmente no mercado existem proeminentes ferramentas desenvolvidas para facilitar o desenvolvimento de aplicações que se baseiem em LLMs. No contexto do estágio foram estudadas duas dessas ferramentas: o Llama Index [8] e o LangChain [9], poderosas opções reconhecidas pelo agilizar e abstrair de todo o processo de construção de RAG (*Retrieval-Augmented Generation*) [144], [145]. Estas auxiliam na otimização das saídas provenientes de LLMs, conferindo-lhes maior versatilidade ao fornecer a capacidade de referência a bases de conhecimento além das fontes de

dados utilizadas no treino. Assim é possível estender o conhecimento do LLM para domínios mais restritos, no caso em estudo (o futebol), não sendo necessário afinar o modelo com dados específicos, tornando-se uma abordagem mais económica. Desta forma a construção das *apps* é mais rápida e iterativa e com baixa dificuldade.

O Llama Index é uma *framework* desenvolvida para criar e simplificar a construção de *apps* que se baseiam em LLMs, tornando o processo de desenvolvimento mais acessível e eficiente [8]. Auxilia no estruturar dos dados ao organizá-los para que sejam legíveis e compreendidos pelos LLMs. Os dados são representados matematicamente, com índices, tornando-se de fácil acesso. Tem a capacidade de recolher dados de APIs, PDFs ou bases de dados.

O LangChain, conhecido anteriormente como GPT Index é uma *framework* desenvolvida para auxiliar no desenvolvimento de aplicações que se baseiem em LLM. Com a utilização de representações vetoriais para frases e palavras é facilitada a compreensão do contexto, sumarização de texto e análise de código. Composto por *chains* (contêm vários componentes de IA promovendo respostas sensíveis ao contexto, úteis para realizar tarefas de nível superior [146]) e componentes (combináveis para trabalhar com LLMs e fáceis de usar). Esta *framework* tem a capacidade de se conectar a diversas fontes de dados como: documentos e APIs. Permite que os LLM lidem com dados específicos para melhorar a sua precisão e personalização, sendo conscientes e sensíveis ao contexto.

Foram realizados testes com o objetivo de verificar a viabilidade da integração destas *frameworks* com o projeto SportGPT. Para tal foram selecionadas algumas questões do conjunto inicialmente construído, cujas *queries* apresentassem níveis diferentes de dificuldade, de forma a analisar o comportamento desta nova abordagem. Na Tabela 7-2 são apresentados os resultados obtidos através do estudo da *framework* Llama Index.

Tabela 7-2 - Resultados Obtidos na *Framework* Llama Index

<i>Pergunta</i>	<i>Query</i>	<i>Resultado</i>	<i>Justificação do Resultado</i>
1- Qual a idade de Cristiano Ronaldo?	MATCH (p:Players {fullname: "Cristiano Ronaldo"}) RETURN p.birth_date	“Cristiano Ronaldo's age is not available in the database.”	Não fez a subtração para saber a idade. E Cristiano Ronaldo é o “name”
2- Qual o nome completo de Cristiano Ronaldo?	MATCH (p:Players {name: "Cristiano Ronaldo"}) RETURN p.fullname	“Cristiano Ronaldo's full name is Cristiano Ronaldo dos Santos Aveiro”	-

Desenvolvimento de um protótipo de um Futebol GPT

<p>3- Quem é o treinador do Benfica?</p>	<pre>MATCH (c:Coach)-[:TEAM_COACH]->(tc:Team_Coach)-[:TEAM_COACHH]->(t:Team) WHERE t.name = 'Benfica' RETURN c.fullname as Coach_Name</pre>	<p>“The coach of Benfica is not available in the provided data.”</p>	<p>Direção do relacionamento TEAM_COACHH incorreto</p>
<p>4- Qual a data do último jogo do Chelsea?</p>	<pre>MATCH (t:Team {name: "Chelsea"})-[:PLAYS]->(m:Matches) WHERE m.date > timestamp() RETURN m.date LIMIT 1;</pre>	<p>“The next game date for Chelsea is currently not available.”</p>	
<p>5- Número de vitórias do Chelsea em casa na Premier League na season 2022/2023?</p>	<pre>MATCH (t:Team)-[:PLAYS]->(m:Matches) WHERE t.name = 'Chelsea' AND m.status = 1 AND m.round_id = 2022 AND m.stage_id = 2023 AND m.home_season_team_id = t.id AND m.home_season_team_id = t.id AND m.home_season_team_id = t.id RETURN COUNT(m) as Numero_de_vitorias</pre>	<p>“Chelsea has not secured any victories at home in the Premier League during the season 2022/2023”</p>	<p>Separou as informações de forma incorreta: m.status = 1 AND m.round_id = 2022 AND m.stage_id = 2023</p>
<p>6- Quem é o guarda-redes do Benfica?</p>	<pre>MATCH (p:Players)-[:PARTICIPATED_IN]->(st:Season_Team)-[:PARTICIPATES_SEASON]->(t:Team) WHERE t.name = 'Benfica' AND st.position = 'Goalkeeper' RETURN p.fullname as Goalkeeper_Name</pre>	<p>“The goalkeeper for Benfica is not available in the current data”</p>	<p>PARTICIPATED_IN, não é para a season team mas sim para season_team_player</p>
<p>7- Capacidade do Estádio do Euskirchen?</p>	<pre>MATCH (t:Team {name: "Euskirchen"})-[:TEAM_VENUE]->(v:Venue) RETURN v.capacity;</pre>	<p>“Final Response: The capacity of the stadium in Euskirchen is 7500.”</p>	<p>-</p>

Os problemas observados durante o teste desta *framework*, no contexto do projeto SportGPT resultaram de vários fatores, como o desrespeito pelos nós e relacionamentos utilizados, a direção desses mesmos relacionamentos e as particularidades dos dados.

Na questão 1, a *query* gerada não retornou resultados devido a dois motivos distintos: “Cristiano Ronaldo” corresponde à variável “*name*” e não a “*fullname*”, e para obter a idade seria necessário subtrair a data de nascimento do jogador à data atual. Informações relativas a estas particularidades: pesquisas pelo “*name*”, e datas de nascimento armazenadas em *string* na base de dados, estão definidas e passadas ao modelo. Na questão número 3, a *query* construída contém um erro crasso: a direção do relacionamento “TEAM_COACHH”, levando a que a sua execução na base de dados não retorne qualquer resultado. Na questão 4 o problema reside essencialmente no relacionamento “PLAYS”, pois este interliga os nós: *Season_Team* e *Matches*, e não os nós *Team* e *Matches*. Para além disso, a cláusula *where* deveria ser diferente: `date(substring(matches.date, 0, 10)) >= date(datetime())`, devido às datas se encontrarem armazenadas no formato de *string*. Não esquecendo que também seria necessário ordenar as datas da mais recente para a menos, para que aquando da filtragem por “LIMIT 1”, ser devolvido o último jogo da equipa. Na questão 5 o modelo não compreendeu o pretendido, pois dividiu incorretamente o valor da *season* em dois nós diferentes com nomes/ funcionalidades completamente distintas: *stage* e *round*. Na questão 6, o modelo utilizou incorretamente o relacionamento PARTICIPATED_IN. Na realidade este interliga os nós *team* e *season_team_player*. Somente a questão número 2 e a questão 6 retornam resultados válidos, possivelmente pelas *queries* terem estruturas mais simples e diretas, onde existe uma correspondência clara entre os nós e relacionamentos. Na Tabela 7-3 são apresentados os resultados obtidos através da integração desta ferramenta no projeto SportGPT.

Tabela 7-3 - Resultados obtidos da *framework* LangChain

Pergunta	Query	Resultado	Justificação do Resultado
1- Qual a idade de Cristiano Ronaldo?	MATCH (p:Players {name: "Cristiano Ronaldo"}) RETURN p.birth_date	[{'p.birth_date': '1985-02-05'}]	Apresenta somente a data de nascimento.
2- Qual o nome completo de Cristiano Ronaldo?	MATCH (p:Players {name: "Cristiano Ronaldo"}) RETURN p.fullname Full Context:	[{'p.fullname': 'Cristiano Ronaldo dos Santos Aveiro'}]	
3- Quem é o treinador do Benfica?	MATCH (c:Coach)-[:TEAM_COACH]->(tc:Team_Coach)-[:TEAM_COACHH]->(t:Team {name: "Benfica"})	[]	Erro na direção do relacionamento TEAM_COACHH

Desenvolvimento de um protótipo de um Futebol GPT

	RETURN c.fullname as Coach_Name		
4- Qual a data do último jogo do Chelsea?	MATCH (t:Team {name: "Chelsea"})-[:PLAYS]->(m:Matches) WHERE m.date > datetime().toString() RETURN m.date ORDER BY m.date LIMIT 1	Erro na <i>query</i> , no toString()	Não é necessário usar o toString() o que leva a uma sintaxe inválida. O relacionamento PLAYS também não é entre aqueles dois nós
5- Número de vitórias do Chelsea em casa na Premier League na season 2022/2023?	MATCH (t:Team)-[:PLAYS]->(m:Matches)-[:STAGE_MATCH]->(s:Stage)-[:OCCURS_IN]->(Season {name: "2022/2023"}) WHERE t.name = "Chelsea" AND m.status = 1 AND (m.home_season_team_id = t.id AND m.second < m.first) OR (m.away_season_team_id = t.id AND m.second > m.first) RETURN COUNT(m) AS Numero_de_vitorias;	[{'Numero_de_vitorias': 0}]	Utilização incorreta do relacionamento PLAYS (este é entre os nós Season_Team e Match). Utilização incorreta do atributo status, e criação de atributos inexistentes (second e first)
6- Quem é o guarda-redes do Benfica?	MATCH (p:Players)-[:PARTICIPATED_IN]->(st:Season_Team) MATCH (t:Team {name: "Benfica"}) WHERE st.team_id = t.id AND p.position = "Goalkeeper" RETURN p.fullname as Goalkeeper_of_Benfica	[]	Utilização do relacionamento PARTICIPATED_IN entre os nós errados.
7- Capacidade do Estádio do Euskirchen?	MATCH (t:Team {name: "Euskirchen"})-[:TEAM_VENUE]->(v:Venue) RETURN v.capacity	[{'v.capacity': 7500}]	-

Os erros cometidos pelo LangChain assemelham-se aos ocorridos na *framework* anterior. Na questão 1, relacionada com a idade de um jogador, o modelo retornou a data de nascimento, algo semelhante ao comportamento inicial do ChatGPT, que ao ser questionado acerca da idade de um jogador retornava somente a data de nascimento. No entanto, esta *framework* utilizou de forma correta a variável “*name*”. De ressaltar que atualmente o *chatbot* já tem capacidade de apresentar a idade. Na questão 3, o modelo atribui uma direção incorreta ao relacionamento TEAM_COACHH, tornando a *query* errónea. Na questão 4, o modelo atribui o relacionamento PLAYS a nós incorretos e utiliza um “`toString()`”, o que resulta numa sintaxe incorreta. Na questão 5 o modelo atribui o relacionamento PLAYS aos nós errados, e utiliza o nó *Matches* da forma equivocada, nomeadamente através da criação de atributos inexistentes. Na questão 6 o modelo utiliza o relacionamento PARTICIPATED_IN com os nós errados, inviabilizando a execução da *query*. À semelhança da *framework* anterior, o LangChain conseguiu responder somente às questões 2 e 7.

Apesar das grandes vantagens apontadas para as *frameworks* em estudo, estas demonstraram não ser a melhor escolha para o projeto em causa. Aquando da sua implementação revelaram não estarem totalmente estabelecidas, não obtendo os resultados tão satisfatórios como esperado. Consoante os resultados obtidos, foi decidido prosseguir com o desenvolvimento em mais baixo nível, para se obter maior destreza e compreensão do comportamento do LLM, ajustando a *System Message* conforme desejado. Com o avançar do tempo, as tecnologias foram aprimoradas, existindo mais documentação e atualizações. Contudo, o servidor já se encontrava em avançado desenvolvimento, não sendo viável retroceder.

7.5 Diagramas

Na Figura 7-2, é apresentado o diagrama de casos de uso do projeto SportGPT.

A descrição dos principais casos de uso encontra-se nos anexos em G.

O diagrama de casos de uso refere os principais casos de uso do projeto SportGPT. Existem cinco atores: o utilizador, a base de dados, o *engine*, o LLM e a API. O utilizador representa quem interage diretamente com o sistema, realizando perguntas futebolísticas. A base de dados é a entidade utilizada para armazenamento de dados de futebol. O LLM é responsável por entender as questões colocadas pelos utilizadores, por gerar a *query* a executar na base de dados e por humanizar o resultado proveniente desta. O *Engine* está encarregue de recolher e processar os dados diversos de diversas fontes como, por exemplo, de APIs.

Desenvolvimento de um protótipo de um Futebol GPT

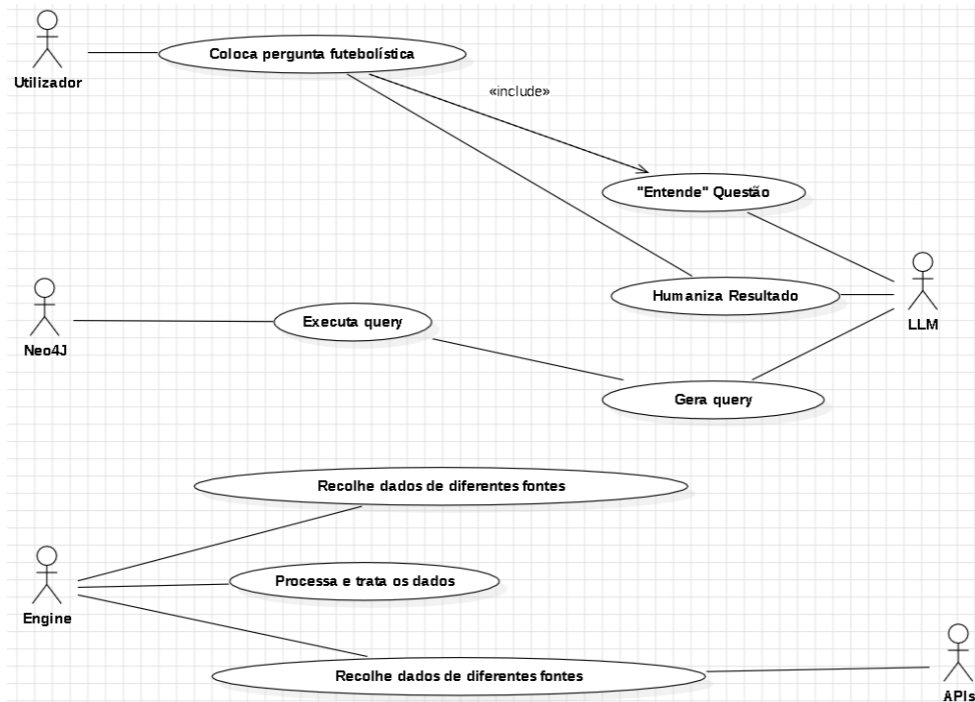


Figura 7-2 - Diagrama de Casos de Uso

Na Figura 7-3 é apresentado o diagrama de sequência do projeto SportGPT.

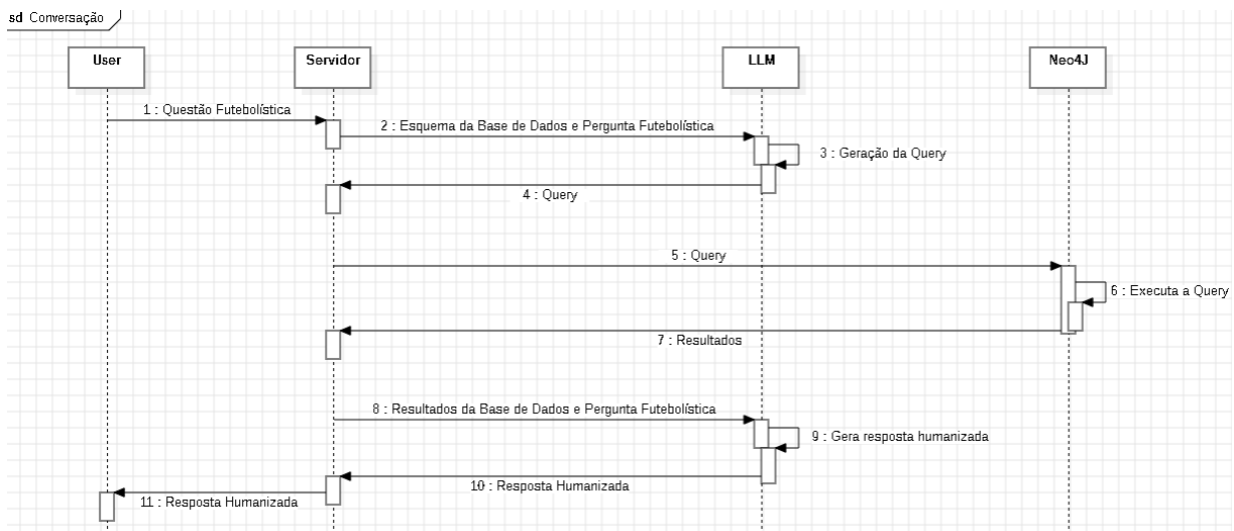


Figura 7-3 - Diagrama de Sequência

Neste diagrama de sequência encontra-se descrito todo o *flow* correspondente ao projeto SportGPT:

- O utilizador faz uma pergunta: O utilizador inicia a interação, enviando uma pergunta futebolística ao servidor;
- O servidor recebe a pergunta: O servidor recebe a pergunta do utilizador e encaminha-a para o LLM (juntamente com o esquema da base de dados);

- O LLM gera a *cypher query*: O LLM processa a pergunta do utilizador e gera uma *query* adequada para pesquisar os dados no Neo4J;
- O servidor envia a *query* para o Neo4J: O servidor encaminha a *query* gerada pelo LLM para o Neo4J;
- O Neo4J executa a *query*: O Neo4J executa a *query* e retorna os resultados obtidos;
- O servidor envia os resultados para o LLM: O servidor encaminha os resultados da *query* ao LLM para que este construa uma frase de resposta ao utilizador;
- O LLM gera a resposta: O LLM processa os resultados da *query* e gera uma resposta humanizada e coerente à pergunta do utilizador;
- O servidor envia a resposta para o utilizador: O servidor envia a resposta gerada pelo LLM para o utilizador.

7.6 Desenvolvimento

No início do desenvolvimento do projeto SportGPT, iniciou-se a implementação do servidor com a utilização do LLM GPT-3.5 Turbo para os dois tipos de chamadas ao modelo: criação das *cypher queries* e pedido de resposta humanizada. Na implementação foi constatado que o modelo ocasionalmente gerava *queries* em SQL, ou fundia essa linguagem com *cypher*. Estes contratemplos foram resolvidos com recurso a especificações detalhadas na *System Message* para compelir ao modelo gerar somente *Cypher queries*.

Além disso, nos dados futebolísticos armazenados existem pequenas nuances necessárias respeitar para obter *queries* e resultados assertivos. Por exemplo: para determinar se um jogador é atualmente membro de uma equipa, é obrigatório que o atributo *is_current* tenha valor “True”, algo que o LLM *a priori* não tem capacidade de reconhecer. Desta forma, foi necessário novamente introduzir informações relativamente a estes pormenores na *System Message*. Verificamos, após vários testes e tentativas de resposta às várias perguntas outrora definidas, que o modelo não tinha capacidade de resposta à maioria das questões, apenas às mais simples. As questões idealizadas encontram-se disponíveis nos anexos em O. Os problemas centravam-se no desrespeito pelas direções e nomenclatura dos relacionamentos e de alguns atributos e confusão da linguagem utilizada.

Para colmatar estes inconvenientes foi realizada uma tentativa de utilização de OpenAI Function Calling. Porém, resultou na criação excessiva de funções para auxiliar o modelo a responder ao máximo de questões possível, levando a que o LLM se desviasse do contexto de geração de *queries* e passasse a tentar adaptar sempre as funções disponíveis. Esta “confusão” poderia levar a sérios contratemplos, tendo sido identificadas duas situações bastante problemáticas:

1. Aquando da chamada ao LLM, com a questão feita pelo utilizador, o modelo pode incorretamente considerar uma função adequada para o propósito. No entanto, os parâmetros dessa função não coincidem com as “entidades” referidas na pergunta realizada. Desta forma, a *query* executada na base de dados não retorna qualquer resultado, sendo obtido como resposta final “Não sei responder a essa questão”.

Exemplo: "Quem é o treinador do Chelsea?". O modelo entra na função correspondente à equipa atual de um jogador, atribuindo Chelsea ao nome do jogador. Como não existe nenhum jogador chamado "Chelsea", a execução na base de dados não retorna resultados.

Nesta abordagem apesar de o modelo errar na execução, não há resultados a serem retornados que possam levar a enganos dos utilizadores.

2. Noutra situação, o modelo poderá escolher uma função que mesmo inadequada, tenha parâmetros coincidentes com as entidades integrantes na questão introduzida. Desta forma, a execução na base de dados retornará resultados, porém inadequados à pergunta realizada.

Exemplo: "Qual a nacionalidade de Cristiano Ronaldo?". O modelo escolhe a função relativa à equipa atual do jogador. Os resultados da base de dados indicam "*Al Nassr*", mas, ao humanizar os resultados, o modelo ajusta a resposta para "A nacionalidade de Cristiano Ronaldo é portuguesa".

Embora a resposta final esteja correta, o problema subjacente pode não ser evidente para o utilizador, podendo resultar em dados desatualizados ou incorretos que podem em certas circunstâncias passar despercebidos.

Perante esta realidade, foi necessário remover a utilização de funções e adotar um modelo mais poderoso: o GPT-4, o qual apresentou resultados significativamente mais promissores e precisos. Contudo, surgiram alguns problemas, associados à construção de *cypher queries*, que foram colmatados novamente com pequenos exemplos de *queries* inseridos na *System Message*. Por exemplo: para obter as estatísticas de uma equipa numa temporada foi necessário adicionar a informação de que o nó *team_stats* permitia atingir esses valores.

Outro problema identificado persistia no fluxo de execução, isto porque aquando da primeira questão futebolística introduzida, o LLM gerava a *query* / entrava numa função de forma acertada. Mas aquando da segunda questão introduzida o modelo tentava sempre entrar numa das *functions* definidas, levando a problemas na *query* e nos resultados obtidos. Por exemplo: se a segunda questão fosse “Qual a data do último jogo do Chelsea”, o modelo entrava na *function* com nome “*get_current_team*” e atribuía o nome da equipa ao nome de um jogador e tentava saber assim qual a equipa atual do “Chelsea”. Esse padrão de comportamento inconsistente era observado nas perguntas subsequentes, enquanto o modelo respondia corretamente apenas às perguntas intercaladas. Para resolver

esta inconformidade foi necessário reformular os *roles* passados para a variável correspondente ao histórico de questões.

Noutra vertente encontrava-se a problemática dos custos elevados associados à utilização deste modelo, sendo solicitada a transição para a versão do GPT-4 Turbo, para que os gastos fossem mais contidos. Contudo, foram imediatamente identificados problemas que não existiam na versão anterior usando o outro modelo mais poderoso: confusão com a direção e nomenclatura dos relacionamentos, invenção de novos atributos e ocasionalmente geração de *cypher queries* conjuntamente com instruções SQL.

Assim foi possível entender que apesar de o GPT-4 ser o modelo mais adequado e que produziu melhores resultados, o elevado custo não era vantajoso face àquilo que se pretendia. Esperava-se que com um modelo mais simples e mais barato como o GPT-4 Turbo se conseguissem resultados semelhantes. Como esse modelo é uma versão *light* do mais robusto acabou, como explicado, por ter algumas nuances e problemas. Desta forma conseguimos entender que existia alguma inviabilidade no projeto no sentido em não se conseguir responder a 100% das questões idealizadas, como é possível verificar no anexo O.

Outro grande problema surge associado ao facto de o *chat* permitir que os utilizadores realizem perguntas futebolísticas de forma livre, o que pode levar a grandes inconvenientes. Pois, tais perguntas podem requerer informações inexistentes na base de dados. Questões ambíguas também podem levar a confusões no entendimento por parte do modelo, aumentando as probabilidades de obter respostas incorretas.

No mês de novembro, o modelo da OpenAI utilizado foi o GPT-3.5 Turbo. O modelo tinha capacidade de responder a questões futebolísticas / construir *cypher queries* simples, tais como: a nacionalidade de um jogador, estádio de uma equipa e a sua capacidade. Nesta fase do desenvolvimento ainda não detínhamos todos os nós e relacionamentos, pois a estratégia era incrementar iterativamente o conhecimento do modelo. Contudo, o modelo apresentava uma dificuldade considerável na compreensão do contexto das perguntas o que prejudicava a construção de *queries* adequadas, resultando numa taxa de resposta assertiva de apenas 6%.

Em dezembro, foram implementadas medidas para colmatar as limitações do modelo: adição da funcionalidade de OpenAI Function Calling, o que melhorou para 27% a capacidade de resposta do modelo. Porém, foram adicionadas várias *functions* à medida que o modelo demonstrava incapacidade de construir *queries* de forma adequada, o que provocou uma extensa lista de *functions* e constantes equívocos do modelo. Perante estes desafios, em fevereiro foi tomado o rumo de migrar para o modelo GPT-4, onde os resultados obtidos de forma correta aumentaram em 20%. No entanto, persistiam ainda alguns problemas associados à excessiva quantidade de *functions*. Testes sem utilização da funcionalidade de funções da OpenAI demonstraram que o modelo possuía dificuldades na resposta a perguntas específicas como: “Quem é o capitão da equipa x?” ou “Quem é o guarda-redes da equipa x?”.

Como solução foram inseridas informações relevantes na *System Message* para auxiliar na construção das *queries*. As *functions* seriam apenas adicionadas em último caso: quando o modelo não tivesse capacidade de elaborar a *query* mesmo com aquele reforço de informações. Com estas alterações, foi possível obter uma taxa de respostas corretas de 54%.

Apesar deste incrível crescimento nas capacidades do *chat*, os custos associados ao GPT-4 eram elevados, levando à proposta de utilização do modelo GPT4-Turbo, uma versão mais económica, mas que se esperava produzir os mesmos resultados. Todavia, algumas das questões respondidas pelo modelo mais poderoso não eram possíveis de responder pelo modelo mais recente, mesmo com detalhes e informações adicionais na *System Message*. Houve uma redução da capacidade de resposta do nosso projeto para 51,28%. Assim foi possível concluir que o uso deste modelo não trazia melhores resultados, e que os resultados do GPT-4, embora melhores, não justificavam os custos elevados.

7.6.1 Lista De Funcionalidades Aplicação Mobile

Nesta secção será apresentada a lista de funcionalidades entregue à *designer* da WIT Software para base da construção dos *mockups* da aplicação *mobile*. Cada funcionalidade encontra-se identificada com a sua prioridade, onde P1 corresponde à mais prioritária e P5 a menos prioritária.

Ideia Principal (P1): *Chat* com capacidade de perguntas-respostas. As perguntas colocadas poderão abranger 3 intervalos temporais:

- Atualidade – Por exemplo: Qual a idade de um jogador
 - Passado – Por exemplo: Resultado do último jogo de uma equipa
 - Futuro – Por exemplo: Qual a data do próximo jogo de uma equipa
-
- O utilizador deverá conseguir colocar a sua dúvida futebolística num *user prompt* e obter a resposta adequada em tempo real (P2);
 - O *chat* deverá permitir que as perguntas sejam feitas através de texto e voz (P1)
 - Por texto – introduzir a pergunta num *user prompt*, selecionando um botão para que esta seja enviada para *backend*;
 - Por voz – deverá existir um botão para que a gravação do áudio seja iniciada.
 - O *chat* deverá permitir que as respostas sejam fornecidas em texto e em voz (P1)
Ao ser recebida a resposta por parte do *backend*, esta deve ser apresentada em texto e por voz, num formato de *chat*;
- Existirá um botão que permita que o utilizador ouça de novo a resposta por voz.

- A aplicação deverá ter uma página de previsão de resultados futebolísticos (P2)

Deverá existir um ecrã na aplicação, onde sejam apresentadas as previsões dos resultados futebolísticos para os jogos a ocorrer no próprio dia e no dia seguinte. Mas estas previsões apenas indicarão se uma equipa ganha, perde ou empata (não o número de golos que poderão marcar).

- Aplicação com Notificações – Oráculo do Futebol (P2)

A exibição de previsões de resultados futebolísticos deverá ser acompanhada do envio de notificações ao utilizador, assim que novas previsões estiverem disponíveis.

- Página Principal da Aplicação

A página inicial deve conter:

- possibilidade de navegação para o *chat* (P1);
- apresentar previsões futebolísticas (P2);
- apresentar notícias futebolísticas (opcional) (P5).

- Página de Perfil de Utilizador (P2)

Nesta página deverá ser possível:

- Alterar a palavra-passe;
- Escolher se quer receber notificações de previsões;
- Alterar a equipa favorita (opcional).

- Autenticação por *email* e *password*

A página de registo deve pedir o nome, *email* e *password*

E a equipa favorita do utilizador (**opcional**)

- Página de *login*

Requerer o *email* e *password*

Funcionalidades opcionais

Foram ainda consideradas algumas funcionalidades adicionais. No entanto, devido ao estágio ter um período definido para término, a viabilidade destas *features* pode ficar comprometida. Perante este cenário, as funcionalidades mencionadas na secção anterior estão sujeitas a restrições de implementação.

- Personalização do *background* do *chat* com o estádio da equipa favorita do utilizador(P4)

A imagem do estádio será gerada com base numa ferramenta de GenAI, que através do conhecimento da equipa favorita do utilizador (escolhida aquando do registo ou na página de utilizador), irá gerar uma imagem do estádio.

- Personalização do *chat* com um avatar (P4)

A resposta por texto será combinada com uma animação realista de um avatar, onde os movimentos labiais serão sincronizados com a resposta sonora correspondente. O avatar será a mascote da equipa favorita do utilizador (definida aquando do registo ou na página de utilizador)

Após a reunião preliminar com a *designer* da WIT Software, foi possível entender que estas funcionalidades adicionais teriam de sofrer alterações:

- Personalização do *background* do *chat* com o estádio da equipa favorita do utilizador

Apesar de na WIT Software ser utilizada uma ferramenta para geração de imagens com GenAI, o uso desta estava alocado apenas a alguns funcionários. Desta forma, a solução sugerida passou por gerar um conjunto limitado de imagens correspondente a estádios de algumas equipas. Seria também gerada uma imagem *default* a apresentar em casos onde o utilizador não escolheu a equipa predileta ou quando seleccionou uma equipa que não tem disponível uma imagem do estádio gerada por IA.

- Personalização do *chat* com um avatar

Dada a complexidade de gerar um avatar dinâmico e a equipa responsável por esta tarefa estar indisponível, foi decidido que seriam gerados avatares estáticos com diferentes expressões. Para além disso, será criado um avatar *default* para situações em que o utilizador não tenha definido a sua equipa favorita ou quando esta não tenha nenhuma mascote associada. Um avatar de exemplo criado pela equipa é apresentado na Figura 7-4.



Figura 7-4 - Exemplo de Avatar

7.6.2 WebApp

Para fases iniciais de demonstrações, antes da construção da aplicação *mobile*, foi construída uma aplicação *web* com recurso à *framework* Flask [135]. Esta aplicação *web* foi o ponto de partida, onde foram implementadas as principais funcionalidades pretendidas: interação por meio de conversação (questões futebolísticas), capacidade de realizar questões por texto e por voz, assim como obter a resposta nesses mesmos moldes. Por ser uma das etapas iniciais, esta *web app* foi mantida com *User Interface* (UI) simples, dada a sua utilização para efeitos somente de demonstração. Esta aplicação consistia apenas num breve texto introdutório acerca da principal funcionalidade da *app*, área para inserção de perguntas futebolísticas, um botão para submissão destas e um botão para início da gravação de áudio a enviar para o *backend*. Para além disso, existia ainda uma área para questões de histórico da conversação, onde as perguntas realizadas pelo utilizador se encontravam à esquerda e as respostas humanizadas, vindas do LLM, à direita.

Na Figura 7-5 é apresentada a página *web* principal assim que é acedida pelo utilizador.



Figura 7-5 - *Web App* SportGPT

Assim que o utilizador introduz uma questão é acionado um *spinner* que tem como função indicar ao utilizador que a sua solicitação está a ser tratada, como apresentado na Figura 7-6.

Desenvolvimento de um protótipo de um Futebol GPT

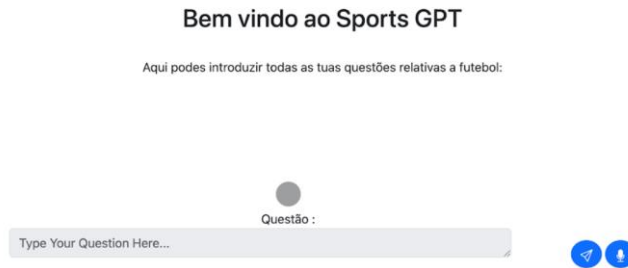


Figura 7-6 - Geração da Resposta

Assim que a resposta é obtida é apresentada ao utilizador à direita, juntamente com a questão introduzida à esquerda, como é observado na Figura 7-7. Sendo possível observar também um histórico de perguntas à medida que o utilizador vai realizando as suas questões.



Figura 7-7 - Apresentação da resposta

Inicialmente apenas era possível realizar perguntas futebolísticas através de texto, mas aquando da introdução da opção por voz, surgiu um desafio relativo à segurança da transmissão dos dados entre *frontend* e *backend*. Esta questão de segurança surgiu devido à natureza sensível dos dados transmitidos, pois o formato áudio é uma forma de dados mais complexa, exigindo medidas para assegurar que a sua transmissão para *backend* ocorra de forma segura. Para contornar este problema, utilizou-se a ferramenta Ngrok [133], uma ferramenta que possibilita a criação de túneis seguros para expor serviços locais na Internet, permitindo que a aplicação executada localmente esteja disponível na *web*. Assim foi possível contornar o problema encontrado na transmissão de dados em formato de áudio. O *ngrok* atuou como uma ponte segura, permitindo que a transferência dos dados fosse feita em segurança sem que a integridade e privacidade das informações fosse comprometida. Esta implementação viabilizou o cumprir da funcionalidade de possibilitar realizar questões por voz, proporcionando uma experiência mais enriquecida e dinâmica ao utilizador.

7.6.3 App Mobile

A aplicação *mobile* foi desenvolvida utilizando a linguagem Flutter tal como havia sido descrito na proposta inicial de estágio. Durante os meses de março e abril foi desenvolvida uma aplicação de exemplo recorrendo a Flutter, para recordar e praticar a linguagem e progredir nessa etapa de estágio. Esta aplicação era bastante elementar, detinha apenas uma página inicial simples com um botão de redirecionamento para um ecrã de *chat*, onde eram realizadas as perguntas futebolísticas e obtidas as respostas. Existia ainda uma página de perfil de utilizador. Esta implementação foi fundamental para a compreensão da conexão entre o *backend* e o *frontend* Flutter, nomeadamente a nível da transmissão dos dados.

Após ser obtido *feedback* por parte da *designer*, foi realizada uma análise dos *mockups* implementados e dos aspetos prioritários a concretizar. Os *mockups* encontram-se em anexo: D, assim bem como as respetivas descrições (anexo E). Houve alguns contratempores tendo em conta inconsistências nos protótipos, tomando como referência a lista de funcionalidades entregue. Estas inexatidões podem ter sido causadas devido à incompletude dos *mockups*, ficando vários aspetos em aberto para discussão futura.

Os problemas encontrados nos *mockups*:

- Incongruência entre páginas de *login*. Num ecrã de *login* inicial é requerido o nome de utilizador e *password*. Mas num cenário de erro (por exemplo: dados incorretos) verifica-se um campo de *email* em vez de nome. Não tendo ficado claro qual a implementação esperada. Assim foi optado por utilizar o campo de *email*;
- A página principal nos *mockups* foi enriquecida com *predictions*. No entanto, os *mockups* ficaram aquém das expectativas, pois apresentavam o número de golos marcados por ambas as equipas num jogo. Quando o esperado e requerido era apresentar somente se a equipa ia ganhar, perder ou empatar. Este problema refletiu-se também na página dedicada à previsão de jogos. A descrição do desenvolvimento das previsões realizadas encontra-se descrita na secção 7.6.5.
- Existência de um ícone de menu no canto superior direito da página principal, para a qual não ficou estabelecida a sua função e conteúdo;
- A página de utilizador não desenhada;
- A experiência do utilizador na página de *chat* ficou um pouco aquém, pois nos cenários de submissão da questão, processamento no *backend* para a geração da *query*, chamada ao LLM e chegada da resposta, existia apenas a alteração de um pequeno texto na *app bar* do ecrã. Algo referido em demonstrações anteriores na empresa foi a importância de destacar e informar o utilizador acerca do que está a ocorrer, por exemplo, a utilização de um indicador de carregamento (*loader*).

A nível do desenvolvimento da *app mobile*, foram priorizados os ecrãs de: *login* e registo, página principal (sem *predictions* e notícias), página de *chat* e de perfil de utilizador. Os restantes aspetos ficaram para posterior realização, pois tal como

referido na lista de funcionalidades entregue, eram opcionais. Assim, seriam implementados apenas no final da conclusão da funcionalidade de *chat*. A página de utilizador foi desenhada com base no *design* dos outros ecrãs projetados, como por exemplo de login e registo. Relativamente à autenticação do utilizador na aplicação *mobile* foi necessário traçar uma estratégia eficaz para armazenar de forma segura os dados dos utilizadores (nome, *email* e palavra-passe) para questões de registo e *login*. A base de dados utilizada é o PostGreSQL, uma vez que é uma base de dados já em uso no estágio. A estratégia encontra-se detalhada no anexo F.

Como foi outrora referido, houve uma mudança no rumo do estágio, resultando na não implementação dos aspetos acima mencionados como opcionais e abertos para discussão, tendo ficado desenvolvidos apenas os aspetos prioritários.

7.6.4 Scripts

No âmbito do projeto de perguntas futebolísticas foram desenvolvidos 4 *scripts* diferentes:

- *Script* de limpeza de tabelas *PostGreSQL* - este *script* é responsável por processar os dados vindos de um *dump* de uma base de dados de teste da empresa. O objetivo passa por eliminar tabelas e atributos indesejados e desnecessários para o projeto SportGPT, e converter atributos em formatos JSON em novos atributos diretos nas tabelas do PostgreSQL.

- *Script* de inserção dos dados do PostgreSQL para o Neo4J – permite de forma automática e dinâmica inserir os dados outrora limpos do PostgreSQL para o Neo4J. Inicialmente os dados eram inseridos de forma individual em cada tabela. Posteriormente, dada a lentidão, foi otimizado para armazenar os dados em *batches* de 10 000 linhas. O processo envolvia a extração individual de cada tabela do PostgreSQL para um arquivo do formato CSV (*Comma-Separed Values*), e posterior inserção em *batch* no Neo4j. Após a conclusão da inserção o arquivo CSV é eliminado, pois torna-se desnecessário.

- *Script* para criação de relacionamentos no Neo4J – *script* com função de criação dos relacionamentos entre os diferentes nós na base de dados futebolística do Neo4J, algo fundamental para a conexão entre os dados.

- *Script* para gestão de dados em formato áudio – considerando que os áudios, provenientes da aplicação *mobile*, relativos às questões são enviados para *backend* para posterior processamento, tornou-se essencial desenvolver um mecanismo que removesse os áudios 10 minutos após a sua criação. O mesmo para os áudios gerados do lado do servidor e enviados para o utilizador com a resposta dada pelo LLM, mas por áudio. A decisão de eliminar os áudios foi tomada para otimizar o uso do espaço de armazenamento ocupado, pois os áudios consomem uma grande quantidade de espaço. Esta questão é deveras relevante num cenário onde a aplicação possua um grande número de utilizadores. Com esta medida, é possível garantir uma gestão eficiente dos recursos, evitando a acumulação desnecessária de dados.

7.6.5 Previsão de Resultados Futebolísticos

Aplicações de casas de apostas viradas para o futebol são muito comuns nos dias que decorrem. Paralelamente, surgiram aplicações de previsões futebolísticas para auxiliar os entusiastas das apostas nas suas decisões. Com objetivo de tornar a aplicação futebolística do projeto SportGPT mais apelativa e com mais funcionalidades, foi estudada a possibilidade de adicionar a funcionalidade de previsão de resultados de jogos futebolísticos. Através da aplicação de algoritmos de *Machine Learning* pretendíamos realizar previsões do número de golos que uma equipa iria marcar nos seus próximos jogos. Posteriormente esta capacidade seria alargada para cartões vermelhos e amarelos atribuídos. Tendo em conta que a variável a prever tratava-se de um valor numérico e não de uma classe, foram estudados quais os algoritmos de regressão possíveis de utilizar [147]: KNN Regressor (*K-nearest Neighbor*) [148], [149], Regressão Linear [150], [151], SVM Regressor (*Support Vector Machine*) [152], [153], Decision Tree Regressor [154], [155], GradientBoosting Regressor [156], [157].

Após um estudo inicial do *dataset* foram tomadas decisões: tratar valores nulos, valores categóricos/ textuais e eliminar variáveis desnecessárias para o contexto. Através de um estudo da correlação entre as variáveis [158], foi entendido que a variável mais *correlacionada* com o target: “*gols_scored*” era “*shots_on_target*”, algo já expectável. Pois esta variável corresponde às tentativas de golo bem-sucedidas. A variável com maior correlação, mas negativa, era “*red_cards*”, indicando que quanto mais cartões vermelhos a equipa detém, menor a probabilidade de ocorrer um golo.

O *dataset* inicial continha os dados provenientes no *dump* futebolístico do projeto SportGPT, sendo realizada uma análise inicial aos atributos: qual o seu significado no contexto e qual o tipo de dado. Este estudo é apresentado na Tabela 7-4 e em P nos anexos.

Tabela 7-4 - Descrição das Variáveis presentes no *dataset*

<i>Variável</i>	<i>Descrição</i>	<i>Tipo de Dado</i>
<i>Team</i>	O nome da equipa que está a jogar	String
<i>Homeoraway</i>	Indica se o jogo foi disputado em casa (HOME) ou como visitante (AWAY).	String
<i>Id</i>	Identificador único da instância	Int
<i>Goals_suffered</i>	Golos sofridos pela equipa	Int
<i>opponent</i>	Nome da equipa adversária	String
<i>match_id</i>	Identificador único do jogo	Int
<i>season_team_id</i>	Identificador único da <i>season_team</i>	Int
<i>goals_scored</i>	Golos marcados pela equipa no jogo	Int
<i>ball_possession</i>	Percentagem de posse de bola no jogo	Float
<i>dangerous_attacks</i>	Percentagem de <i>dangerous attacks</i> realizados pela equipa no jogo	Float
<i>formation</i>	Formação tática da equipa	String

Desenvolvimento de um protótipo de um Futebol GPT

<i>coach_id</i>	Id do treinador	Int
<i>penalty_shootout_goals</i>	Percentagem de golos marcados pela equipa numa disputa de penaltis	Float
<i>half_time_goals_scored</i>	Percentagem de golos marcados até ao intervalo	Float
<i>Pass_accuracy</i>	Precisão dos passes da equipa	Float
<i>Shot_accuracy</i>	Precisão de chutes da equipa	Float
<i>Substitutions</i>	Percentagem de substituições no jogo	Float
<i>Goal Attempts</i>	Percentagem de tentativas de golo feitas pela equipa	Float
<i>Injuries</i>	Percentagem de lesões ocorridas durante o jogo	Float

Perante esta análise foi possível entender que existem variáveis desnecessárias que foram imediatamente descartadas: *match_id*, *season_team_id*, *coach_id* e *formation*. De realçar que existem 2 instâncias por jogo, uma na ótica da equipa visitante e outra na ótica da equipa que jogou em casa.

Foram calculadas as médias das várias estatísticas de desempenho de cada equipa a partir dos dados presentes no *dataset*. Essa abordagem permite uma visão agregada do desempenho de cada equipa ao longo do tempo, auxiliando na identificação de padrões e tendências. Variáveis de texto, como o nome da equipa e do oponente, foram transformadas em variáveis numéricas através de *OneHotEncoder* [159]. Os valores nulos foram inicialmente imputados através da utilização da média, mediana ou moda [160] e das variáveis mais correlacionadas. Posteriormente foram aplicados os diferentes modelos de *Machine Learning* anteriormente descritos e analisadas métricas como: MSE (*Mean Squared Error*), MAE (*Mean Absolute Error*), RMSE (*Root Mean Squared Error*) [161].

Os primeiros testes realizados retornaram resultados aquém do esperado, apresentando taxas de acertos: 34,93%. Estes problemas poderão ter sido causados pela imputação dos valores nulos pela média, mediana e moda, pois a relação entre os dados acaba por se perder [135]. Ou então por estarmos a realizar médias dos vários valores para obter a “performance geral” de cada equipa. Pois o desempenho desta varia ao longo das temporadas e dos jogos devido a lesões, mudanças de estratégia ou de acordo com as equipas com que se defronta. Como forma a contornar a problemática de utilização da média para a imputação de valores nulos, passamos a utilizar um novo método: *KNNImputer* e não foram realizadas médias de performance da equipa. Nomeadamente, os resultados retornados também não foram satisfatórios à semelhança do teste anterior, como observado na Tabela 7-5.

Tabela 7-5 - Resultados do primeiro teste

<i>Modelo</i>		<i>MSE</i>	<i>Valores Acertados</i>	<i>Valores errados</i>	<i>Média de % percentagem de acertos por equipa</i>
<i>Random Forest</i>	Log2	0,2565	318	190	36,141
	sqrt	0,2531	317	191	36,160
<i>KNN</i>	N_neighbors=5	0,4213	270	238	32,800
<i>Regressor</i>	N_neighbors=2	0,4554	282	226	33,097
	N_neighbors=3	0,4641	270	238	31,739
	N_neighbors=4	0,4386	261	247	32,482
	N_neighbors=6	0,4152	272	236	32,977
<i>Linear Regression</i>		0,3184	276	232	32,977
<i>SVM</i>		0,3937	256	252	32,720
<i>Lasso</i>		0,3206	275	233	32,327
<i>Decision Tree</i>		0.0011	505	3	48,183
<i>Gradient Boosting Regression</i>		0.252	320	188	37,557

Os resultados obtidos não foram satisfatórios, nomeadamente em relação à média de percentagem de acertos por equipa, mesmo com a alteração das variáveis mais correlacionadas a considerar nos testes.

Variáveis como data do jogo, temporadas e competições foram adicionadas ao *dataset* de forma a ser possível atribuir diferentes pesos a eventos mais recentes. Posteriormente foram consideradas/ criadas informações como: hora do dia, semana do ano ou estado do campo. Uma das decisões tomadas foi abandonar a tentativa de prever valores numéricos como: golos marcados pelas equipas, cartões vermelhos ou amarelos.

Os resultados insuficientes obtidos até então poderiam dever-se ao facto de o número de golos ser uma variável fortemente variável, sendo influenciável por vários fatores e imprevisibilidades. Para uma previsão rigorosa nesse sentido, seria necessária uma quantidade maior de dados relativos a dados intrínsecos do desempenho das equipas.

Desta forma, foi escolhido um novo caminho: prever se uma equipa irá perder, ganhar ou empatar. Esta tática é estatisticamente mais simples, apresenta uma gama de resultados inferior e permite diminuir a complexidade dos modelos utilizados. Modelos de classificação binária ou ternária, necessitam de menos dados históricos para alcançar resultados satisfatórios. Simplificar a previsão para categorias mais amplas permite otimizar a precisão do modelo, facilitando a obtenção de resultados mais consistentes.

O dataset final continha 90696 instâncias e 37 atributos. De forma mais aprofundada continha 27 competições, 4376 equipas envolvidas e 39 temporadas diferentes. À semelhança das tentativas anteriores foram retiradas variáveis que não demonstravam interesse no contexto e retiradas instâncias cujo *standings* (classificação num campeonato) era nulo, não faria sentido imputar um valor destes, pois iria trazer enviesamento. Variáveis categóricas foram transformadas em numéricas: *team*, *opponent*, *season*, *league* e *pitch*. A partir deste momento o *dataset* encontrava-se limpo, podendo passar-se à fase de testes.

Abaixo são descritos alguns dos testes realizados, e cujas tabelas são apresentadas em anexo: P.

- Datas dos Jogos não ordenadas;
- Datas dos Jogos ordenadas, onde as mais recentes têm um maior peso, pois representam a performance mais recente da equipa;
- Datas Ordenadas e atribuição de pesos às temporadas, onde as temporadas mais recentes têm um peso maior;
- Datas ordenadas e consideração pela hora do dia e pelo dia da semana em que o jogo ocorre;
- Datas Ordenadas e período do dia;
- Datas Ordenadas, semana do ano em que o jogo ocorre e atribuição de pesos às temporadas.

A partir dos resultados obtidos nos testes mais recentes foi possível entender que a utilização de algoritmos de ML permitia a previsão de resultados futebolísticos com métricas satisfatórias. Portanto, nos próximos passos, após o estabilizar do servidor, será possível integrar esta funcionalidade à aplicação, tornando-a mais versátil e com mais funcionalidades que atraiam o utilizador.

7.6.6 Text-To-Speech

Uma das funcionalidades adicionada na *web page*, foi a possibilidade de as respostas retornadas serem apresentadas tanto por texto como por voz. A OpenAI fornece um *endpoint* baseado no modelo TTS (*text-to-speech*), com 6 vozes integradas, com fim de narrar textos, produção de áudio em diferentes idiomas ou fornecer saídas de áudio em tempo real [162]. Desta forma após receber a resposta humanizada do LLM é feita uma chamada ao *endpoint text-to-speech* para transformar o texto em áudio. Como é possível constatar na Figura 7-8, o áudio retornado é guardado no servidor, sendo gerado um identificador único para cada áudio através de UUID (*Universally Unique Identifier*).

```
response1 = openai.audio.speech.create(
    model="tts-1",
    voice="echo",
    input=humanized_response,
)

# make a random UUID, useful to ensure that each audio file has a unique name.
uuid_audio=uuid.uuid4()
relative_path = "static/Sounds/"+str(uuid_audio)+".mp3"

absolute_path = os.path.join(os.getcwd(), relative_path)
response1.stream_to_file(absolute_path)
```

Figura 7-8 - Implementação da Funcionalidade *Text-to-Speech*

Foi definido um caminho relativo onde o arquivo é guardado e utilizado o UUID gerado como nome. Esse, posteriormente foi transformado em caminho absoluto para que o ficheiro de áudio fosse corretamente armazenado na pasta “/static/Sounds”, pois este diretório pode variar dependendo de onde for executado. A utilização do UUID permitiu armazenar, no ambiente, múltiplos áudios gerados dinamicamente, não havendo colisões de nome. Assim não há qualquer risco de um utilizador receber/ ouvir a resposta de outro.

7.6.7 Speech-To-Text

Outra funcionalidade adicionada à página *web* foi a possibilidade de realizar questões futebolísticas por texto e por voz. Para a transcrição de áudio para texto foi utilizado o *endpoint* da OpenAI [163], [164]. O áudio correspondente à questão colocada, vindo de *frontend*, é guardado num caminho relativo e enviado para o *endpoint* de *speech to text* da OpenAI para ser interpretado e convertido em texto.

Na Figura 7-9 é apresentado o código relativo a esta implementação.

```
@app.route('/api/v1/audio', methods=['POST'])
def audio():
    #audio message
    audio_file = request.files.get('audio')
    if audio_file:
        save_path = 'static/Sounds_Received/' # Relative path
        os.makedirs(save_path, exist_ok=True) # Create the directory if it doesn't exist
        audio_filename = "audio.mp3"
        audio_file_path = os.path.join(save_path, audio_filename)
        print("Saving to:", audio_file_path)
        audio_file.save(audio_file_path)

        audio_file_for_openai = open(audio_file_path, "rb")
        transcript = openai.audio.transcriptions.create([
            model="whisper-1",
            file=audio_file_for_openai,
            response_format="text"
        ])

        a=conversation(transcript)
        #return jsonify({"user_question":transcript, "humanized_response":a.humanized_response,"
        return jsonify({
            "user_question": transcript,
            "humanized_response": a.json["humanized_response"],
            "audio_file": a.json["audio_file"]
        })
```

Figura 7-9 - Implementação da Funcionalidade *Speech-to-Text*

Após a transcrição o texto é enviado para a função responsável por realizar a chamada ao LLM para a criação da *cypher query* e posterior geração da resposta a

retornar ao utilizador. De denotar a necessidade de falar pausadamente e articular bem as palavras para que o modelo capte corretamente as palavras e não transforme/entenda o texto em árabe ou com palavras não proferidas.

As funcionalidades acrescentadas permitiram oferecer ao utilizador uma experiência dinâmica e com maior facilidade de interação. Além disso, esta revelou ser o mais natural possível, melhorando a usabilidade e experiência.

7.7 Limitações do Chat

Ao longo de 4 meses de estágio foi desenvolvido o servidor do projeto SportGPT, estando desde o início detalhadas as possíveis limitações:

- O LLM é probabilístico e não determinístico, o que significa que para a mesma pergunta o LLM pode gerar diferentes *queries*. Estas podem levar a resultados corretos, incorretos ou mesmo inexistentes;
- Os utilizadores podem fazer a mesma questão de diferentes maneiras podendo LLM pode interpretá-las de forma diferente;
- Ao longo dos testes à lista de perguntas definida inicialmente, foi possível entender que o LLM detinha alguns problemas a nível da utilização dos relacionamentos. Como forma a contornar este aspeto, foram acrescentadas informações relativamente a essas nuances na *System Message* para que o LLM tivesse informações mais detalhadas e construísse as *queries* de forma correta. Mas no futuro poderá tornar-se um problema, uma vez que existem vários tópicos futebolísticos de interesse e curiosidade dos utilizadores. Estes podem realizar perguntas de futebol que não se encontrem na lista. Assim, é possível verificar a incapacidade de cobrir todas as nuances na *System Message*.
- Aumentar a informação na *System Message* leva a que os custos também sejam mais altos, uma vez que o número de *tokens* passados numa chamada aumenta. Em modelos mais caros como o GPT-4, revelou-se uma problemática.

7.8 Resultados

Os resultados obtidos ao longo dos vários testes e alterações realizadas ao servidor do projeto SportGPT, demonstraram não ser satisfatórios. No anexo O, encontram-se os dados obtidos perante um conjunto de 88 perguntas, bem como os modelos GPT utilizados.

O percentual máximo de questões possíveis de responder foi de 54%, o que se revela inadequado relativamente ao esperado. Em novembro, o mês que iniciamos o desenvolvimento do *chat*, a taxa de satisfação era de apenas 6%, mas correspondia a uma versão bastante preliminar do código. Em dezembro, apesar das alterações implementadas, a percentagem subiu apenas para 27%. No mês de fevereiro,

começamos a utilizar o modelo GPT-4 que, como já mencionado, apresentou os melhores resultados, com uma taxa de acerto de 54%. No entanto, em abril, foi optado pelo modelo GPT-4 Turbo devido a questões de custo, o que resultou na diminuição na capacidade de resposta (51,28%).

Foi estabelecida uma comparação entre a capacidade de resposta do SportGPT e a do ChatGPT, que apenas obteve 7,6%. Os principais problemas do *chatbot* relacionaram-se com perguntas acerca de dados atuais, por exemplo: “Qual a data do último jogo do Chelsea?”. Este entrave deve-se ao facto do modelo ter sido treinado com dados até 2022. A 13 de maio de 2024 foi lançado o modelo da OpenAI: GPT-4o [163]. Este modelo pode ser utilizado, ainda que de forma limitada, no ChatGPT [164]. Conforme os dados apresentados no anexo, o modelo alcançou uma taxa de resposta de 80,68% [163]. Esta elevada performance deve-se à capacidade do modelo de realizar *web browsing* [165].

Muitas questões poderiam ser resolvidas mediante a inclusão de exemplos de *cypher queries* na *system message*, no entanto, essa abordagem poderia tornar-se uma desvantagem no custo e na generalização do modelo. Através dessa solução estaríamos a direcionar de forma explícita o modelo para a *query* correta. Assim o modelo acabaria por não ser capaz de generalizar e criar as *queries* de forma autónoma. Os problemas obtidos poderão ser causados pela grande complexidade do esquema da base de dados apresentada, com bastantes nuances e particularidades, ou mesmo pela limitação do próprio modelo GPT na criação de *cypher queries* mais completas com muita interligação de relacionamentos e nós.

7.9 Conclusão

Após diversas tentativas para instruir o LLM a gerar *queries* corretas consoante o esquema da base de dados fornecido, verificámos que havia persistência de vários problemas. Nomeadamente dificuldade na geração de *queries* correspondentes à lista inicial de perguntas. Para resolver a problemática foram realizadas tentativas como: alteração do modelo e dos *roles* utilizados e adição de informações detalhadas fornecidas ao modelo. Mesmo com modelos mais robustos como o GPT-4 e o GPT-4 Turbo, a existência de problemas e a incapacidade de responder à grande maioria das perguntas persistiu. Desta forma, foi reformulada a proposta inicial do GPT do futebol: para a um projeto de curiosidades futebolísticas. Assim, as ideias tidas como opcionais: apresentação de notícias, utilização de avatares e *background* personalizado ficaram por realizar. O mesmo se aplica à atualização periódica dos dados, à integração de outras fontes de dados, à previsão de resultados futebolísticos e à adição de capacidade de respostas e perguntas por voz na aplicação *mobile*. Estas ideias descartadas poderão ser consideradas em outros projetos ou mesmo em trabalhos futuros caso o desenvolvimento do projeto SportGPT seja retomado.

8 FOOTBALL TRIVIA

O projeto do Football Trivia dá continuidade à temática central do futebol e à utilização de ferramentas de GenAI (LLMs) com fim a reutilizar parte do trabalho já efetuado no projeto anterior. Os dados futebolísticos armazenados no Neo4J formaram a base de conhecimentos necessária para a implementação do projeto. Esta nova abordagem propõe que, aquando do término de um jogo de futebol, seja gerado um conjunto de curiosidades futebolísticas relativas às equipas envolvidas, campeonatos e temporadas. As *cypher queries*, contrariamente ao realizado no projeto SportGPT, foram definidas e construídas manualmente prontas a serem executadas, para mitigar problemas provocados pelos LLMs. Assim, o modelo utilizado: o GPT-3.5 Turbo tem como função apenas gerar o texto associado à curiosidade, segundo a descrição da *query* executada e o resultado obtido. O projeto pretende transmitir aos fãs de futebol informações futebolísticas através de curiosidades apelativas que façam alusão, por exemplo, a *records* dos jogadores, histórico e comparação da performance de uma equipa ao longo das últimas temporadas e competições.

8.1 Análise das várias Contas X

Um estudo exaustivo foi realizado a 12 contas da rede social X, cujas publicações se relacionam com curiosidades futebolísticas, com fim a analisar as informações fornecidas aos utilizadores. A seleção das contas começou com a indicação do orientador, seguida de uma pesquisa abrangente para identificar outras contas similares, isto é, com publicações relacionadas com curiosidades futebolísticas. A análise realizada constou no estudo dos últimos 30 *tweets* publicados por cada conta, relativamente à informação apresentada, temática, *likes*, *retweets*, visualizações e *saves*.

As contas analisadas foram: OptaJoe [166], OptaJoao [167], OptaPaolo [168], OptaJean [169], OptaFranz [170], OptaJose [171], Squawka[172].

Nos anexos, em M, encontra-se disponível a análise exaustiva realizada a essas contas. Foi possível observar os principais temas apresentados e as publicações com maior número de interações, extraindo *insights* acerca dos interesses predominantes dos utilizadores. As principais temáticas representadas nas informações publicadas encontram-se descritas na Tabela 8-1.

Tabela 8-1 - Temáticas das Curiosidades da Rede Social X

<i>Temática</i>	<i>Exemplo de Curiosidade</i>
<i>Performance dos Jogadores</i>	31 ° Golo de Joshua Kimmich marcado nos 4° de final da UEFA (Union of European Football Associations). É o maior número de golos marcados nesta fase de todas as competições da temporada.

<i>Performance das Equipas</i>	O Real Madrid eliminou o atual campeão da UEFA Champions League 6x em eliminatórias. Duas vezes mais que qualquer outra equipa nessa competição.
<i>Performance numa dada competição</i>	Na primeira mão dos quartos de final da UEFA foram marcados 18 golos, o maior nº da temporada como em 2010/11
<i>Performance numa dada temporada</i>	Foram marcados 18 golos nos primeiros quartos de final da UEFA desta época o maior número da temporada

Através das informações recolhidas foi possível idealizar curiosidades utilizando os dados futebolísticos armazenados na base de conhecimentos do projeto SportGPT. Verificando-se a viabilidade da ideia em causa. Com uma lista considerável de ideias de curiosidades em mãos, passámos à fase de construção das *queries* correspondentes à obtenção dessas informações, como forma a corroborar a exequibilidade da solução proposta.

Exemplos das curiosidades e as respetivas *queries*, construídas manualmente, encontram-se descritas em N, nos anexos.

8.2 Análise de Requisitos

Neste subcapítulo serão descritos os requisitos definidos para o projeto do Football Trivia. Esta etapa foi fundamental no processo de desenvolvimento do projeto, pois permitiu identificar todas as funcionalidades necessárias.

8.2.1 Requisitos Funcionais

- As curiosidades apresentadas deverão ter como temáticas principais: equipas, jogadores, temporadas e campeonatos de futebol;
- Deve existir uma página na *web app* para cada temática;
- O utilizador deve poder escolher / filtrar sobre quais equipas, jogadores, temporadas e campeonatos de futebol quer obter factos;
- O LLM deverá gerar curiosidades coerentes e apelativas com base na descrição da *query* e dos resultados obtidos através da execução desta na base de dados.

8.2.2 Requisitos Não Funcionais

- A execução das *queries* na base de dados e as chamadas ao LLM deverão ser realizadas de forma a tentar minimizar ao máximo o tempo de resposta. As curiosidades não deverão demorar mais que 1 minuto a

ser apresentadas ao utilizador (apesar de no futuro o projeto correr em *offline* há sempre espaço para melhorar os tempos de execução);

- A interface deve permitir a fácil navegação entre as várias páginas das diferentes temáticas;
- A arquitetura deve permitir a adição de novas *queries* no futuro de forma fácil;
- As curiosidades geradas devem ser relevantes e interessantes para o utilizador consoante a temática escolhida por este.

8.3 Diagramas

Na Figura 8-1 é apresentado o diagrama de casos de uso, representando os cinco principais casos de uso no projeto Football Trivia.

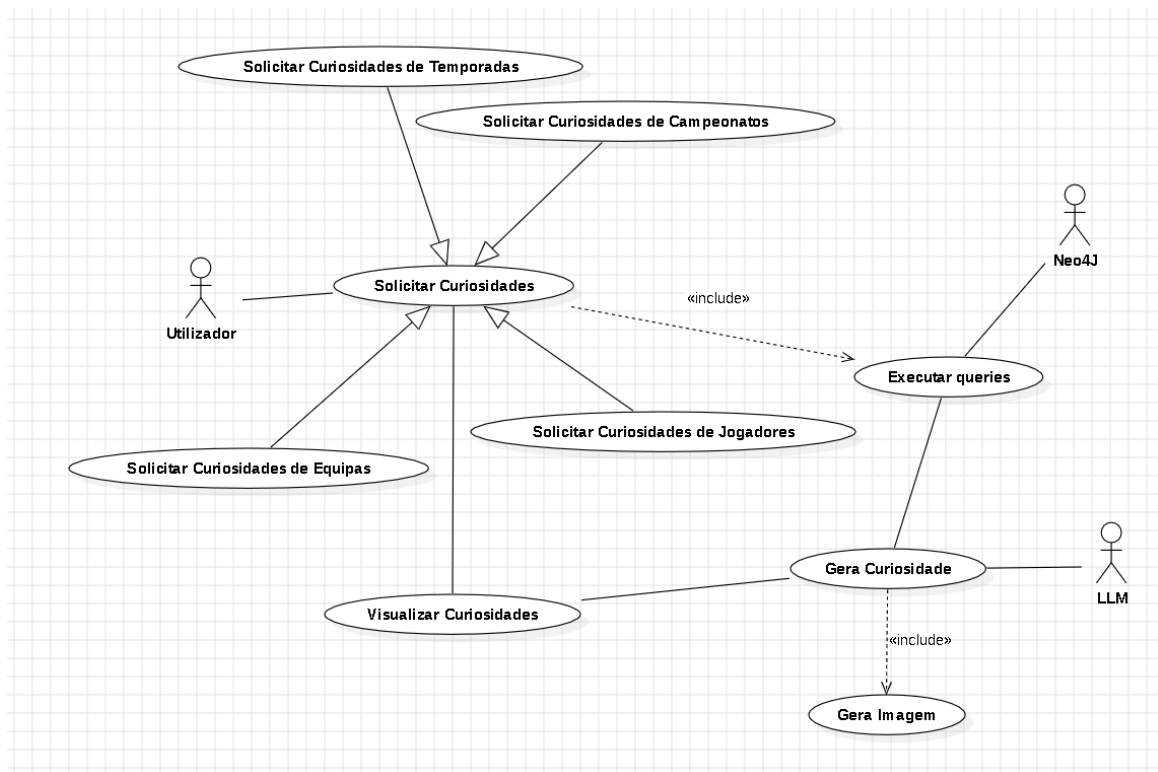


Figura 8-1 - Diagrama de Casos de Uso Football Trivia

A descrição dos casos de uso encontra-se descrita nos anexos em H.

O diagrama de casos de uso refere os principais casos de uso do projeto Football Trivia. Existem 3 atores envolvidos: o utilizador, a base de dados e o LLM. O utilizador é quem interage diretamente com o sistema, solicitando um conjunto de curiosidades acerca de equipas, jogadores, temporadas e campeonatos. O Neo4J é a base de dados utilizada para armazenar as informações sobre futebol. O LLM é

o modelo responsável pela geração de respostas textuais (curiosidades) e imagens futebolísticas.

Na Figura 8-2 é apresentado o diagrama de sequência do projeto Football Trivia.

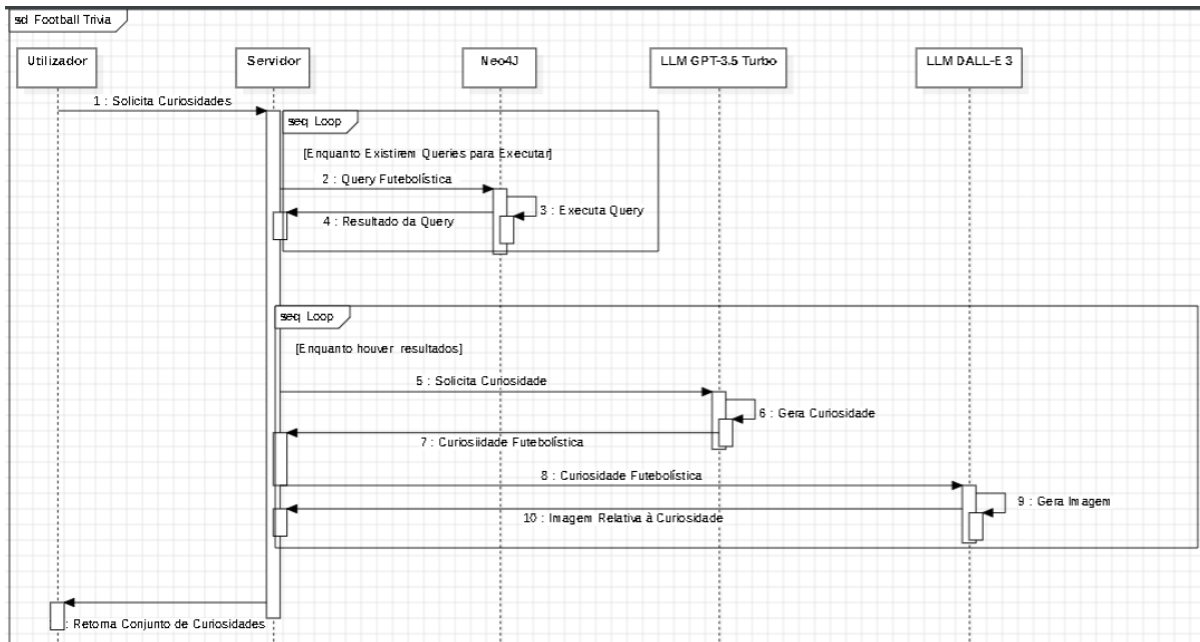


Figura 8-2 - Diagrama de Sequência Football Trivia

Neste diagrama de sequência encontra-se descrito todo o *flow* correspondente ao projeto Football Trivia:

- O utilizador solicita curiosidades futebolísticas: o utilizador inicia a interação com o sistema, solicitando curiosidades relacionadas com o futebol (jogadores, equipas, temporadas ou competições);
- O servidor entra num *loop* de forma a processar múltiplas *queries* a executar na base de dados;
- O Neo4J executa as *queries* de forma a retornar um resultado ou conjunto de resultados;
- O LLM gera uma curiosidade com base na *query* executada e nos resultados obtidos na execução desta no Neo4J;
- O servidor pega na curiosidade gerada e envia-a para o LLM DALL-E 3 para que seja gerada uma imagem relacionada com a curiosidade;
- O servidor retorna o conjunto de curiosidades e as respetivas imagens ao utilizador.

8.4 Arquitetura

Neste subcapítulo será descrita a arquitetura proposta para o Football Trivia. A arquitetura pode ser observada na Figura 8-3.

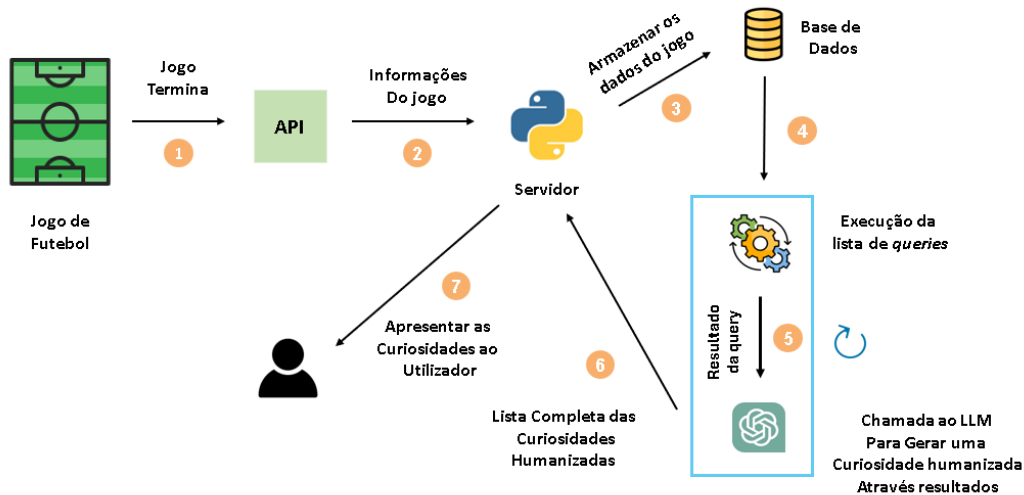


Figura 8-3 - Arquitetura do Projeto Football Trivia

Nos passos 1 e 2: após a ocorrência de um jogo de futebol, os dados e eventos do jogo são disponibilizados, possíveis de aceder através de uma API. No passo 3 os dados são inseridos na base de dados, como forma a enriquecer e atualizar a base de conhecimentos. De ressaltar que este aspeto ficou em aberto para posterior implementação.

No passo 4: será executada uma lista de *queries* (construídas manualmente) correspondente a eventos do jogo, estatísticas, informações de jogadores, temporadas ou do campeonato. A cada *query* executada é realizada uma chamada ao LLM: GPT-3.5 Turbo (passo 5), que mediante a descrição e dos resultados obtidos, após a execução da *query*, cria um texto em formato de curiosidade. É com auxílio do LLM que os dados são apresentados ao utilizador de forma cativante e interessante. As várias curiosidades vão sendo armazenadas numa lista, e assim que todas as *queries* forem executadas, a lista é apresentada ao utilizador (passos 6 e 7), numa *app mobile*, numa *web page* ou mesmo através de publicações numa rede social. As *queries* geradas encontravam-se divididas, inicialmente, de acordo com seis categorias como apresentado na Tabela 8-2.

Tabela 8-2 - Temáticas das *Queries* construídas

<i>Tipo</i>	<i>Descrição</i>	<i>Exemplo:</i>
1	Performance da equipa numa temporada	Quantas vezes uma equipa ganhou nos jogos desta temporada

2	Performance da equipa no jogo do dia	Nomes dos Jogadores que marcaram golos e que fizeram assistências no jogo de hoje
3	Histórico de confronto entre 2 equipas	Datas em que estas 2 equipas se defrontaram
4	Performance de um só jogador numa temporada e numa competição	Quantos golos um jogador já marcou nesta temporada
5	Performance dos jogadores numa temporada e numa determinada competição	Lista de jogadores que mais golos marcaram nesta temporada da competição
6	Estatísticas e informações de uma temporada	Número de clubes que participaram nesta temporada desta competição
7	Estatísticas e informações de uma competição	Jogador que mais golos marcou numa competição

Cada *query*, dentro de cada tema, é identificada por um identificador único como apresentado na Tabela 8-2. Futuramente poderá ser implementado um mecanismo que consoante os acontecimentos e eventos do jogo recém ocorrido determine as *queries* mais adequadas e interessantes aos utilizadores. Para esse método será necessário que cada tema seja também identificado com um *id*. Assim poderá ser feita uma seleção dinâmica da temática e *queries* mais adequadas.

Dentro de cada tema, poderão existir *queries* com o mesmo *id* e de construção semelhante (diferem apenas nos valores dos campos que lhe são passados). Por exemplo: num caso de ser executada uma *query* relativa à data dos últimos 3 jogos de uma equipa, seria interessante retornar os valores para as 2 equipas que se defrontaram.

Assim a 1ª *query* recebe o nome de uma das equipas e a 2ª *query* o nome da outra equipa, Figura 8-4.

```

{#team1
  "id":9,
  "query": f""MATCH (Team_Stats)-[:STATS]-(:Season_Team)-[:PARTICIPATES_SEASON]->(team:Team{{name:
MATCH (Season_Team)-[:Season_Team]-(:season:Season{{name:'{season}'}})-[:EXISTS]-(:competition{{name:'{co
  "description": f"Query que retorna a média de shots on target e shots off target da {team1} nest
  "result":None,
},
{#team2
  "id":9,
  "query": f""MATCH (Team_Stats)-[:STATS]-(:Season_Team)-[:PARTICIPATES_SEASON]->(team:Team{{name:
MATCH (Season_Team)-[:Season_Team]-(:season:Season{{name:'{season}'}})-[:EXISTS]-(:competition{{name:'{co
  "description": f"Query que retorna a média de shots on target e shots off target da {team2} nest
  "result":None,
},

```

Figura 8-4 - *Queries* com Ids iguais

Esta abordagem garante que se uma das *queries* for selecionada como de interesse para o utilizador, ambas as *queries* serão executadas. Tornando a informação prestada mais completa e abrangente.

Desenvolvimento de um protótipo de um Futebol GPT

A nível da primeira demonstração foi desenvolvida uma *web page* utilizando Flask onde foram incluídos cinco campos de *input*: nome das equipas envolvidas no jogo de futebol, temporada, competição, data do jogo ocorrido e ainda um botão de submissão. Foi adicionada uma *dropdown* com uma lista de idiomas de forma que as curiosidades possam ser geradas em várias línguas. A interface da página inicialmente implementada encontra-se na Figura 8-5.

Curiosities about a Match 🌐


Team Name	Team Name	Season	Competition	Game Date	Language ▼	
Team Name	Team Name	(xxxx/xxxx)	Competition	(YYYY-MM-DD)	English	

Figura 8-5 - UI da web page do Fotball Trivia

Após esta demonstração inicial, outras ideias foram consideradas:

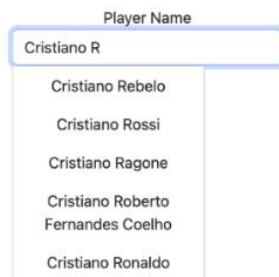
- Adicionar várias páginas à *web app* de forma a permitir uma divisão de acordo com as diferentes temáticas;
- Adicionar uma página direcionada para curiosidades de jogadores, na qual o utilizador possa introduzir o nome do jogador acerca do qual quer obter informações e estatísticas de performance.

As *queries* foram divididas em diferentes listas: *seasons*, *competitions*, *players* e *teams*. Foram implementadas diferentes páginas *web* para cada temática, garantindo que as curiosidades apresentadas fossem separadas consoante os temas.

De seguida, foram realizadas pequenas alterações relativas à UI:

- Adicionar *dropdowns* correspondentes às temporadas, para limitar erros possíveis de serem introduzidos pelos utilizadores;
- Adicionar *dropdowns* associadas à escolha das diferentes *competitions*.

Para melhorar a experiência do utilizador e minimizar erros, foram realizadas alterações na interface da página de *players*, no campo de introdução do nome do jogador. Quando o utilizador digita o nome do jogador para obter as respetivas curiosidades, são apresentadas sugestões de nomes à medida que o texto é inserido. Este processo é facilitado devido à criação de um *endpoint* acionado pelo *frontend* sempre que há uma entrada textual no referido campo. Sempre que o utilizador escreve ou edita o campo, um pedido POST é enviado para o *endpoint* contendo o “nome parcial” introduzido. O *endpoint* ao receber a solicitação procede à execução de uma *cypher query*, no Neo4J, para procurar na base de dados jogadores cujo nome contenha o texto digitado. Da base de dados serão retornados até 10 nomes distintos, enviados para o *frontend* e apresentados no formato de lista ao utilizador, tal como apresentado na Figura 8-6.



Player Name

Cristiano R

- Cristiano Rebelo
- Cristiano Rossi
- Cristiano Ragone
- Cristiano Roberto Fernandes Coelho
- Cristiano Ronaldo

Figura 8-6 - Introdução do nome do Jogador

Todo o mecanismo proporciona uma experiência interativa permitindo que os utilizadores encontrem de forma rápida o nome do jogador que procuram e que tenham contacto com nomes de outros jogadores. Além disso, ajuda a minimizar possíveis erros introduzidos, garantindo que os dados inseridos e as respetivas respostas sejam precisos e relevantes.

À semelhança da implementação descrita no ponto anterior, foi adotada uma abordagem análoga para o preenchimento do campo correspondente ao nome da equipa para a qual o utilizador pretende obter curiosidades. Assim, à medida que o utilizador digita o nome da equipa será feita uma solicitação ao *endpoint* criado no *backend*, onde uma *cypher query* é executada no Neo4J, pesquisando por equipas cujo nome contenha o texto digitado.

Os resultados retornados pela base de dados serão enviados para o *frontend* e apresentados no formato de lista ao utilizador, tal como apresentado na Figura 8-7.



Curiosities about Teams 🇵🇹

Team: Benfic

Region ▼: No regions available

Season ▼: 2018/2019

Language ▼: English

Benfica U21

Abrantes e Benfica

RM Hamm Benfica

Benfica U23

Benfica II

Benfica Castelo Branco

Figura 8-7 - Introdução do nome da equipa

Durante a realização de testes foi descoberta uma pequena incongruência nas curiosidades relativas às equipas, pois não aparentavam ser totalmente corretas. Por exemplo, a nível de obter curiosidades da equipa “Benfica”, eram retornadas informações como: “O treinador do Benfica é Roger Schmitdt e Bernardo Tavares”, quando na realidade o treinador do Benfica é somente Roger Schmidt. Além disso informações acerca de golos, cartões vermelhos e amarelos também não eram consistentes com os dados pesquisados. Após uma análise aprofundada constatamos

que o problema residia na existência de equipas com o mesmo nome armazenadas na base de dados.

Como observamos na Figura 8-8, existem 2 equipas com nome “Benfica”, mas com identificadores únicos diferentes, realçando que apesar de terem o mesmo nome, são equipas distintas.

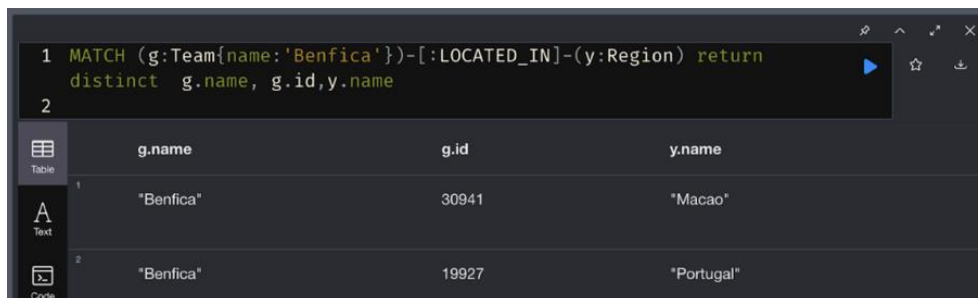


```
1 MATCH p=(c:Coach)←[:SEASON_TEAM_COACH]-(stc:Season_Team_Coach)-[:COACH_SEASON]→(t:Season) MATCH v=(stc)-[:SEASON_COACH_TEAM]→(g:Team) where toUpper(g.name)=toUpper('Benfica') return distinct c.name, c.id, g.name, g.id
```

	c.name	c.id	g.name	g.id
1	"R. Schmidt"	15739	"Benfica"	19927
2	"F. Bernardo Tavares"	21714	"Benfica"	30941

Figura 8-8 - Query relativa a equipas com o mesmo nome

Com um estudo mais detalhado foi perceptível que a diferença entre as equipas constava na região da qual são oriundas, tal como apresentado na Figura 8-9.



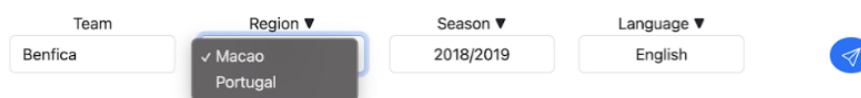
```
1 MATCH (g:Team{name:'Benfica'})-[:LOCATED_IN]-(y:Region) return distinct g.name, g.id,y.name
```

	g.name	g.id	y.name
1	"Benfica"	30941	"Macao"
2	"Benfica"	19927	"Portugal"

Figura 8-9 - Localização das equipas

Para resolver este obstáculo foi necessário alterar e corrigir as queries construídas para que fosse considerada a região a que a equipa pertence. Adicionalmente foi acrescentado um campo no *frontend* que permitisse ao utilizador escolher a região associada à equipa pretendida, tal como observamos na Figura 8-10.

Curiosities about Teams 🏆



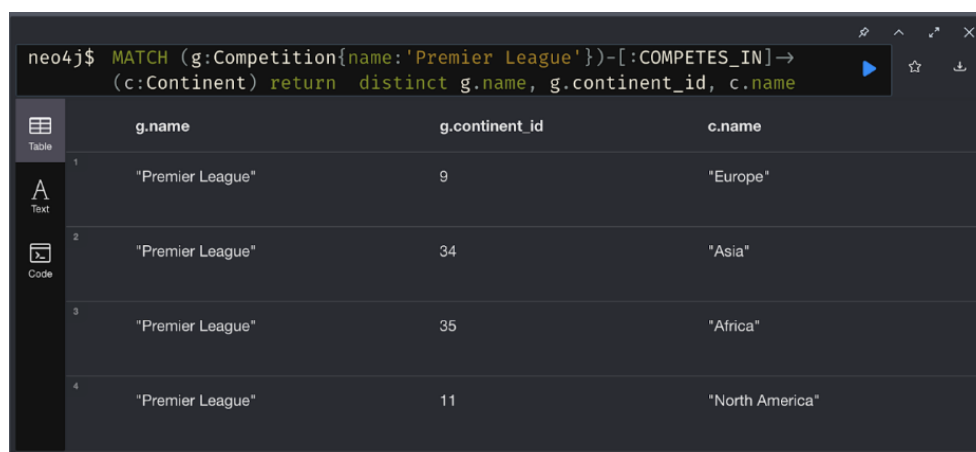
Team: Benfica | Region: **✓ Macao** (Portugal) | Season: 2018/2019 | Language: English

Figura 8-10 - Escolha da região da equipa

Como forma a melhorar a experiência do utilizador e facilitar o conhecimento das regiões de origem das equipas de futebol, sem necessidade de decorar ou de

pesquisa prévia, foi implementado um mecanismo que apresenta uma lista das regiões associadas a uma equipa. No *frontend*, ao digitar o nome da equipa é enviada uma solicitação POST para um *endpoint* no *backend*. Este *endpoint* executa uma *cypher query* na base de dados, que devolve as regiões de origem da equipa com base no seu nome. As regiões são retornadas para o *frontend* e apresentadas ao utilizador na forma de *dropdown*, podendo este seleccionar de entre as opções disponíveis, caso haja mais do que uma. Perante a situação de não existirem regiões associadas a *dropdown* é desabilitada, possivelmente a equipa digitada não existe na base de dados, ou o nome encontra-se mal escrito. Através desta funcionalidade é proporcionado ao utilizador uma forma fácil de escolher a equipa e a sua região de forma mais intuitiva, e as curiosidades apresentadas são mais pertinentes e corretas.

Outra questão observada na fase de testes foi relativamente às competições. Os resultados retornados a nível de vencedores das competições nas finais/semifinais, não era coerente com os dados pesquisados, isto porque uma só competição existe em vários continentes e regiões como apresentado na Figura 8-11.

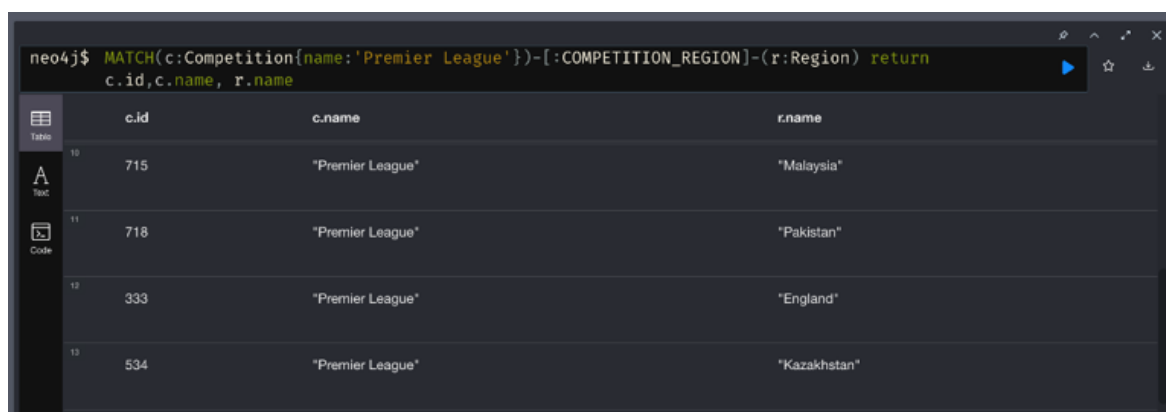


```
neo4j$ MATCH (g:Competition{name:'Premier League'})-[:COMPETES_IN]->
(c:Continent) return distinct g.name, g.continent_id, c.name
```

	g.name	g.continent_id	c.name
1	"Premier League"	9	"Europe"
2	"Premier League"	34	"Asia"
3	"Premier League"	35	"Africa"
4	"Premier League"	11	"North America"

Figura 8-11 - Continentes a que pertencem as competições

Na Figura 8-12, é apresentada a situação de uma competição ocorrer em várias regiões distintas.



```
neo4j$ MATCH(c:Competition{name:'Premier League'})-[:COMPETITION_REGION]-(r:Region) return
c.id,c.name, r.name
```

	c.id	c.name	r.name
10	715	"Premier League"	"Malaysia"
11	718	"Premier League"	"Pakistan"
12	333	"Premier League"	"England"
13	534	"Premier League"	"Kazakhstan"

Figura 8-12 - Regiões onde as Competições ocorrem

Efetivamente, a Premier League mais conhecida é a que ocorre na Europa e em Inglaterra, que corresponde ao *id* 333. Assim no *frontend* foi alterada a *dropdown* das *competitions* para que detenha os ids das principais competições, tal como apresentado na Figura 8-13.

```
<div class="col-md">
  <label for="competition_select">Competition ▼</label>
  <select id="competition_select" class="form-control text-center">
    <option value="726|AFC Champions League">AFC Champions League</option>
    <option value="1074|Belarusian Cup">Belarusian Cup</option>
    <option value="344|Bundesliga">Bundesliga</option>
    <option value="331|Champions League">Champions League</option>
    <option value="610|Chilean Cu">Chilean Cup</option>
    <option value="603|Copa Argentina">Copa Argentina</option>
    <option value="358|Copa do Brasil">Copa do Brasil</option>
    <option value="498|Czech Cup">Czech Cup</option>
    <option value="364|Euro Qualification">Euro Qualification</option>
    <option value="332|Europa League">Europa League</option>
    <option value="333|Premier League">Premier League</option>
    <option value="354|La Liga">La Liga</option>
    <option value="347|Ligue 1">Ligue 1</option>
    <option value="556|Segunda Liga">Segunda Liga</option>
  </select>
</div>
```

Figura 8-13 - Código de frontend relativo à dropdown das competições

Desta forma, no *backend* o valor passado será dividido entre *id* e nome da competição. Ambos os valores são enviados para a função responsável por substituir nas *queries* e nas respetivas descrições os valores corretos. As *queries* foram ajustadas para que passassem a procurar pelo *id* da competição e não pelo nome. O nome foi apenas utilizado para as descrições, uma vez que o LLM necessita de conhecer o nome da competição para gerar a curiosidade e não o id. Após todas as alterações realizadas à UI a experiência do utilizador melhorou consideravelmente, limitando possíveis incongruências.

8.5 Desafios

No Football Trivia houve alguns obstáculos relativamente ao tempo de apresentação das curiosidades. Este tempo é representativo do tempo da execução da *cypher query* juntamente com as diversas chamadas ao LLM. Para entender efetivamente os tempos de execução e poder analisar a evolução destes consoante as alterações posteriormente feitas, foram realizados vários testes. Na Tabela 8-3 encontram-se descritos os tempos médios de execução das *queries* com as múltiplas chamadas ao LLM.

Tabela 8-3 - Tempos de Execução do fluxo

Temática	Tempos médios de Execução, em segundos, todo o fluxo					Média
Seasons	72,28	65,123	69,618	69,809	66,489	68,664
Competitions	23,336	24,204	24,889	23,366	25,809	24,321
Players	90,09	86,959	89,606	83,281	90,034	87,994
Teams	60,138	56,294	55,757	59,413	61,847	58,690

Foram realizados cinco testes, sempre com os mesmos dados de entrada, e calculadas médias finais. Constatamos nos primeiros testes que os tempos de execução não são os melhores. Apesar de não ser um problema crítico e prioritário, dado que o projeto será executado em *offline*, há sempre espaço para otimização e correção destes tempos.

Com fim a reduzir os valores temporais foram realizados testes, realizando somente uma chamada ao LLM. Neste caso, as *queries* são executadas e os resultados retornados e as descrições (das *queries*) são armazenados num dicionário onde a *key* corresponde ao *id* da *query*. O dicionário é enviado para o LLM com informações específicas:

- Com base nas informações apresentadas deverão ser geradas curiosidades futebolísticas;
- Cada índice do dicionário corresponde a uma curiosidade distinta e independente, não devendo existir combinação de contexto;
- As curiosidades geradas devem ser separadas por um separador para facilitar o posterior tratamento.

Os resultados obtidos nesta tentativa encontram-se na Tabela 8-4.

Tabela 8-4 - Tempos de execução do fluxo com apenas 1 chamada ao LLM

<i>Temática</i>	<i>Tempos de Execução, em segundos, todo o fluxo</i>					<i>Média</i>
<i>Seasons</i>	52,034	48,741	43,194	51,099	50,669	49,147
<i>Competitions</i>	13,844	8,900	6,812	7,522	6,742	8,764
<i>Players</i>	93,005	67,777	70,395	60,149	59,062	70,078
<i>Teams</i>	38,420	33,757	33,233	43,589	47,424	39,285

Os resultados retornados eram mais satisfatórios do que os obtidos anteriormente. No entanto, houve um aspeto a ter em conta: a quantidade de informações apresentadas na curiosidade. A realização de uma chamada ao LLM por *query* resultava numa curiosidade mais abrangente, mais rica em informações e claramente mais apelativa. Muitas das *queries* construídas retornam resultados ao longo das últimas temporadas, de várias competições, sendo possível estabelecer um histórico da equipa ou do jogador. Como podemos verificar na Figura 8-14.

Desenvolvimento de um protótipo de um Futebol GPT

Rayan Philippe e Erling Haaland estão empatados no topo da lista dos jogadores que mais golos marcaram na temporada 2022/2023, com 58 golos cada um. Philippe compete na National Division, enquanto Haaland brilha na Premier League. Logo atrás deles, Dominik Stolz, Fenna Kalma e Adam Salley dividem a terceira posição, todos com 52 golos cada. Stolz e Salley jogam na National Division e Championship, respetivamente, enquanto Kalma destaca-se na Eredivisie Women. Esta é uma temporada repleta de talento e golos, com estes jogadores a mostrarem a sua capacidade goleadora em diferentes competições. ID: 1

Surpreendentemente, na temporada 2022/2023, os jogadores que mais assistências fizeram foram Antoine Griezmann, Koen Kostons e Zizo, com 26 assistências cada. Griezmann brilhou na La Liga, enquanto Kostons se destacou na Eerste Divisie e Zizo na Premier League. Além deles, Viktor Gyökeres e A. Lewis também impressionaram, com 24 assistências cada, na Championship e na League Two, respetivamente. Esses atletas certamente foram fundamentais na criação de oportunidades para seus companheiros de equipe e contribuíram significativamente para o sucesso de suas equipes. ID: 2

Curiosidade futebolística: Esta temporada, os jogadores J. David, B. Krasniqi, Abderrazak Hamdallah, Ali Mabkhout e Marko Dugandzic destacaram-se por serem os que mais penalties converteram nas respetivas ligas. J. David lidera a lista com 20 penalties convertidos na Ligue 1, seguido por B. Krasniqi com 18 na Superliga. Abderrazak Hamdallah e Ali Mabkhout estão em terceiro lugar, ambos com 16 penalties convertidos, nas ligas Pro League e Uae League, respetivamente. Já Marko Dugandzic marcou 14 penalties na Liga 1. Esses jogadores mostraram grande eficácia nas grandes penalidades, contribuindo assim com importantes golos para as suas equipas. ID: 3

Figura 8-14 - Ouput das curiosidades das temporadas, num cenário de múltiplas chamadas ao LLM

Ao realizar uma só chamada ao LLM é possível observar resultados bastante simples e diretos. Nos casos referidos anteriormente, em que as *queries* retornam informações de várias temporadas e competições, o modelo apresenta apenas informações relativas a uma só temporada / competição. As curiosidades retornadas são mais pobres em informações, como comprovado pela Figura 8-15.

Sabias que os 5 jogadores que mais golos marcaram na temporada 2022/2023 foram Rayan Philippe (58 golos na National Division), Erling Haaland (58 golos na Premier League), Dominik Stolz (52 golos na National Division), Fenna Kalma (52 golos na Eredivisie Women) e Adam Salley (52 golos no Championship)? ID: 1

Você sabia que os jogadores com mais assistências na temporada 2022/2023 foram Antoine Griezmann, Koen Kostons e Zizo, com 26 assistências cada em diferentes competições? ID: 2

Curiosidade: O jogador com mais penalties marcados na temporada 2022/2023 foi J. David com 20 penalties na Ligue 1. ID: 3

Em jogos da temporada 2022/2023, Akinyoola Samson, Mostafa Fathy, Bruno Petkovic, Lukas Fridrikas, Denis Kolinger e Josh Windass foram os jogadores que marcaram golos em tempo extra. ID: 4

A competição com maior média de cantos por jogo na temporada 2022/2023 foi a A-League, com uma média de 11.13 cantos por jogo. ID: 5

Figura 8-15 - Output das curiosidades das temporadas, num cenário de uma só chamada ao LLM

Curiosidades sobre um tema específico, como o futebol, ganham vida quando são repletas de detalhes adicionais em vez de apresentar somente informações sucintas. Abordagens com textos mais abrangentes poderão envolver e cativar os utilizadores de forma mais eficaz. No caso do futebol, os fãs estão sempre disponíveis para obter conhecimentos adicionais acerca de jogadores, equipas e competições. Ao fornecer *insights* mais completos, os utilizadores podem mergulhar de forma mais profunda no assunto e descobrir curiosos detalhes. Assim, é preferível a nível de conteúdo apresentar informações mais encorpadas do que básicas.

Apesar de os resultados temporais obtidos, com uma única chamada ao LLM, serem significativamente melhores, ainda não eram os ambicionados e os ótimos. Tendo sido feitos testes mais exaustivos para entender o tempo de execução de cada *query* individualmente. Os resultados obtidos são apresentados nas tabelas

seguintes. Na Tabela 8-5 são apresentados os tempos de execução das várias *queries* associadas à temática das competições.

Tabela 8-5 - Tempo de execução de *queries* das Competições

Tempos de Execução das Queries relativas a Competitions em Segundos

<i>Id da Query</i>	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média de tempos
1	0,003	0,003	0,004	0,005	0,006	0,0042
2	0,051	0,047	0,049	0,049	0,240	0,0872
3	0,046	0,042	0,043	0,045	0,043	0,0438
4	0,050	0,047	0,047	0,050	0,047	0,0482
5	0,056	0,044	0,042	0,042	0,042	0,0452
6	0,051	0,048	0,046	0,048	0,048	0,0482
7	0,014	0,012	0,012	0,012	0,030	0,0160
8	0,003	0,003	0,003	0,003	0,003	0,0030
9	0,003	0,002	0,002	0,003	0,003	0,0026

Os tempos de execução das *queries* correspondentes às competições futebolísticas, não representam um problema, pois são realmente baixos comparativamente aos resultados obtidos nas outras tabelas.

Na Tabela 8-6 são apresentados os tempos de execução das *queries* associadas às temporadas.

Tabela 8-6 - Tempo de execução de *queries* das Temporadas

Tempos de Execução das Queries relativas a Temporadas em segundos

<i>Id da Query</i>	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média de tempos
1	8,600	7,781	7,842	7,474	7,574	7,8542
2	1,423	1,333	1,428	1,316	1,278	1,3556
3	1,096	1,061	1,092	1,056	1,037	1,0684
4	0,315	0,325	0,321	0,339	0,32	0,324
5	0,008	0,007	0,006	0,006	0,006	0,0066
6	0,005	0,006	0,006	0,006	0,005	0,0056
7	0,032	0,026	0,024	0,025	0,023	0,026
8	0,182	0,185	0,181	0,176	0,181	0,181
9	0,005	0,006	0,005	0,005	0,005	0,0052
10	6,010	6,272	6,274	5,973	6,143	6,1344
11	0,004	0,004	0,004	0,005	0,006	0,0046
12	0,005	0,006	0,005	0,005	0,005	0,0052
13	0,004	0,005	0,004	0,005	0,005	0,0046
14	0,005	0,026	0,005	0,006	0,006	0,0096
15	0,041	0,064	0,042	0,043	0,044	0,0468
16	0,027	0,029	0,029	0,028	0,03	0,0286

Desenvolvimento de um protótipo de um Futebol GPT

Relativamente à execução das *queries* correspondentes às temporadas observamos enormes discrepâncias de tempos de execução nas *queries* com *id* 1 e *id* 10. Estas *queries* correspondem a: “Qual o nome dos jogadores que marcaram mais golos nesta temporada em 5 competições” e “Qual o jogador que marcou mais golos no total na temporada”, respetivamente. Como forma de reduzir os tempos de execução foram realizadas otimizações nestas *queries* e no caso da primeira limitamos os resultados a apenas 3 competições.

Na Tabela 8-7 são apresentados os tempos de execução das *queries* associadas aos jogadores.

Tabela 8-7 - Tempo de execução de *queries* dos Jogadores

Tempos de Execução das Queries relativas a Players em segundos

<i>Id da Query</i>	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média de tempos
1	1,415	1,201	1,08	1,203	0,937	1,1672
2	0,399	0,364	0,339	0,336	0,343	0,3562
3	3,621	3,767	3,669	3,549	2,969	3,5150
4	1,014	0,848	0,888	0,999	0,818	0,9134
5	0,447	0,464	0,422	0,418	0,442	0,4386
6	0,885	0,656	0,613	0,673	0,645	0,6944
7	8,773	9,196	7,300	7,478	3,360	7,2214
8	0,592	0,348	0,327	0,385	0,342	0,3988
9	5,219	4,465	4,341	4,669	4,384	4,6156
10	5,269	5,269	4,891	5,707	2,709	4,769
11	8,972	8,136	7,384	6,846	3,975	7,0626
12	0,666	0,672	0,804	0,722	0,733	0,7194
13	0,760	0,763	0,838	0,846	0,868	0,8150
14	2,578	2,844	3,063	2,563	2,510	2,7116

As *queries* relativas aos jogadores representam ser as mais demoradas. Pois necessitam que sejam consideradas diferentes temporadas, competições e cruzadas várias informações na *query* executada. Além disso as *queries* que envolvem somatórios de golos ou eventos futebolísticos são as que mais se alongam.

As consultas mais demoradas correspondem aos ids:

7 – Competições em que o jogador mais golos marcou ao longo da sua carreira;

9 – Número de cartões amarelos e vermelhos que o jogador marcou no total pelas equipas das quais já foi integrante;

10 – Equipa atual do jogador, assim bem como a sua idade, data de nascimento, nome completo, entre outros dados pessoais;

11- Número de golos marcados pelo jogador em 5 equipas pelas quais já passou

As *queries* foram otimizadas e foram reduzidos os dados apresentados de 5 para 3 resultados.

Na Tabela 8-8 são apresentados os tempos de execução das *queries* associadas às equipas.

Tabela 8-8 - Tempo de execução de *queries* das Equipas

Tempos de Execução das Queries relativas a Teams

<i>Id da Query</i>	Execução 1	Execução 2	Execução 3	Execução 4	Execução 5	Média de tempos
1	0,018	0,010	0,008	0,007	0,008	0,010
2	0,013	0,007	0,006	0,006	0,006	0,008
3	1,633	2,334	1,513	1,535	1,783	1,760
4	0,004	0,004	0,007	0,024	0,005	0,009
5	23,718	20,803	20,734	19,629	19,545	20,886
6	1,190	1,090	1,177	1,095	1,333	1,177
7	5,527	4,845	4,782	4,523	4,836	4,903
8	0,006	0,005	0,005	0,005	0,005	0,005
9	0,044	0,009	0,008	0,009	0,008	0,016
10	0,032	0,006	0,006	0,006	0,006	0,011
11	0,122	0,006	0,006	0,006	0,006	0,029
12	0,004	0,004	0,004	0,004	0,004	0,004
13	0,006	0,007	0,006	0,006	0,007	0,006

Relativamente às *teams*, o problema de tempos residia maioritariamente na *query* com *id* 5 com descrição: “Informar se a equipa sofreu pelo menos 1 golo nos últimos 3 jogos, sendo o resultado retornado: *True* ou *False*. Se em todos os jogos sofreu pelo menos 1 golo o resultado retornado será *True*, caso contrário, *False*.”

Como forma a corrigir novamente os tempos de geração das curiosidades, foi considerada a possibilidade de realização de chamadas concorrentes ao LLM. Ou seja, as *queries* serão executadas umas a seguir às outras e os resultados armazenados. Após todas as *queries* terem sido executadas, as chamadas ao LLM para geração da curiosidade são feitas de forma concorrente e simultânea. Assim que todas as curiosidades sejam geradas são finalmente retornadas ao utilizador. Na Tabela 8-9 são apresentados os resultados obtidos.

Tabela 8-9 - Tempo de execução do fluxo com chamadas concorrentes ao LLM

<i>Temática</i>	<i>Tempos de Execução, em segundos, todo o fluxo</i>					<i>Média</i>
<i>Seasons</i>	21,346	14,707	14,960	14,574	13,984	15,914
<i>Competitions</i>	4,092	3,041	4,284	3,875	3,759	3,810
<i>Players</i>	41,179	33,965	31,629	30,686	28,388	33,169
<i>Teams</i>	9,967	9,398	9,636	9,151	9,663	9,563

Esta nova versão permite uma interação mais rápida entre o utilizador e o sistema, onde as informações são entregues de forma célere e apelativa, sendo estas

completas e com bastante informação contrariamente à realização de uma única chamada ao LLM.

Por fim, foi realizado um teste final: realizar chamadas concorrentes a nível das *queries*, apresentando a evolução na Tabela 8-10.

Tabela 8-10 - Tempos de Execução com utilização de Concorrência

<i>Temática</i>	<i>Tempos de Execução, em segundos, todo o fluxo</i>					<i>Média</i>
<i>Seasons</i>	9,526	14,819	11,425	10,323	10,591	11,337
<i>Competitions</i>	4,537	3,829	4,092	3,867	3,405	3,946
<i>Players</i>	26,623	13,750	12,813	18,511	13,514	17,042
<i>Teams</i>	11,444	6,717	6,736	6,153	7,658	7,742

Esta última abordagem: realização de chamadas concorrentes ao LLM e execução simultânea de *queries* foi a escolhida para o projeto FootballTrivia, dados os bons tempos obtidos.

8.5.1 Geração de Imagens com GenAI

Como fim de tornar as curiosidades geradas mais apelativas e atrativas, foi adicionado ao projeto a funcionalidade de adição de imagens geradas por GenAI. Foram estudadas diferentes APIs possíveis de auxiliar na implementação da nova funcionalidade como: o DALL-E, DeepAI, HotPot e o Gemini. Todas as possibilidades de geração de imagens têm um custo associado e por vezes a imagem gerada e a sua qualidade não correspondem ao ambicionado. Dada a existência de uma API Key da OpenAI, e após *feedback* da equipa de GenAI da WIT, foi optado por utilizar o modelo DALL-E para a geração das imagens.

Inicialmente foi testada a utilização do modelo DALL-E 2, onde a cada imagem gerada eram gastos 2 cêntimos. Infelizmente as imagens não correspondiam ao esperado relativamente ao retratado na curiosidade e a nível da qualidade como é possível comprovar pelo conjunto de imagens representando na Figura 8-16.



Figura 8-16 - Exemplos de imagens geradas pelo DALL-E 2: a) e b) acerca de Cristiano Ronaldo; c) Jogadores do Chelsea

Desta forma, foi optado por realizar alguns testes utilizando o DALL-E 3. As imagens geradas tinham uma qualidade extraordinária para o propósito pretendido e retratavam bastante bem a curiosidade apresentada, como demonstrado na Figura 8-17.

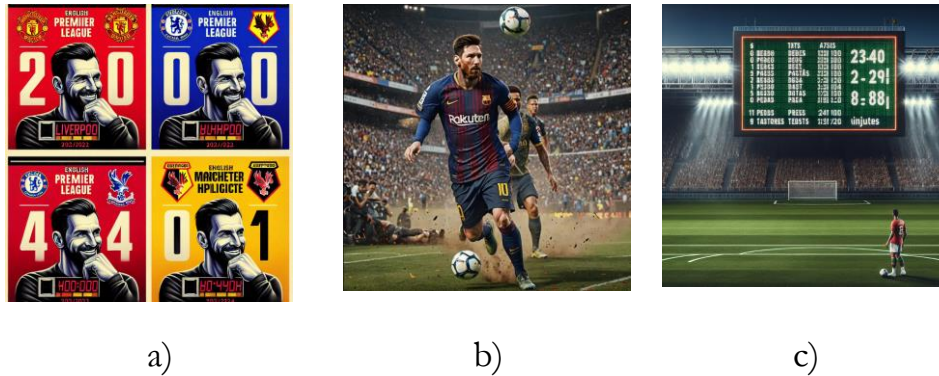


Figura 8-17 - Exemplos de imagens geradas pelo DALL-E 3: a) Resultados dos últimos jogos do Chelsea e b) Últimos golos de Lionel Messi; c) Últimos golos de Cristiano Ronaldo

Apesar de o modelo ter um custo de 4 cêntimos por imagem, foi o eleito para realizar demonstrações.

A nível de implementação, a chamada ao modelo DALL-E teve como base a curiosidade retornada pelo modelo GPT-3.5 Turbo e uma solicitação do tamanho 1024x1024, como apresentado na Figura 8-18.

```
response = openai.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[{"role": "user", "content": f"{request_curiosity}"}]
)

humanized_trivia = response.choices[0].message.content

response = client.images.generate(
    model="dall-e-3",
    prompt=f"{humanized_trivia} ",
    size="1024x1024",
    quality="standard",
    n=1,
)

image_url = response.data[0].url
```

Figura 8-18 - Chamada ao DALL-E para geração de imagens

Durante o processo de decisão sobre qual o curso de ação a seguir, foram utilizadas imagens provenientes do *Provider* existentes no *dump* de dados fornecido, para os jogadores, competições e equipas. A descrição da implementação destas vertentes pode ser encontrada nos anexos J e K.

8.6 Conclusão

O projeto Football Trivia consistiu no desenvolvimento de curiosidades futebolísticas após a ocorrência de um jogo de futebol, onde as informações fornecidas deviam ser atraentes e informativas. O projeto permitiu a reutilização de muito do conhecimento e ferramentas utilizadas no projeto anterior: GenAI, LLMs, Neo4J, dando um novo rumo ao estágio. Foi possível integrar de forma diferente dois LLMs, cada um com a sua função: gerar curiosidades e gerar imagens para tornar a interação dos utilizadores mais interativa e dinâmica. Contrariamente ao projeto SportGPT, observamos que os resultados obtidos neste projeto foram satisfatórios e coerentes com o que foi pedido. Através da construção das *queries* foi possível mitigar a grande problemática sentida no desenvolvimento do projeto SportGPT e consequentemente fornecer resultados mais corretos e precisos. O Football Trivia atingiu assim o objetivo “geral” requerido no início do estágio curricular: unir 2 mundos bastante distintos, o futebol e a GenAI. O projeto não só demonstrou a viabilidade de aplicar tecnologias no meio desportivo como abriu novas possibilidades para trabalhos futuros como extrapolação da ideia para outros campos do futebol (treinadores, estádios, ...) ou para outros desportos.

Andreia Filipa Almeida Graça

9 FOOTBALL QUIZ

O projeto do Football Quiz tem como objetivo desenvolver um jogo no formato de quiz onde os utilizadores possam responder a um conjunto de questões sobre jogadores e testar os seus conhecimentos futebolísticos. O quiz apresenta questões acerca de jogadores de 2 equipas distintas, sendo estas escolhidas pelo utilizador. As perguntas serão construídas com base na utilização de uma base de dados de grafos e um LLM.

A base de dados utilizada é o Neo4J, a mesma utilizada nos projetos anteriores. O LLM, com base no esquema da base de dados que lhe é passado (com descrições dos nós e relacionamentos de jogadores assim bem como exemplos de *cypher queries*) irá gerar um conjunto de perguntas e as respetivas *queries*. Estas serão executadas no Neo4J com intuito de obter um resultado. Caso o resultado retornado seja vazio será feita uma LLM *Call* adicional para solicitar ao modelo que com base no seu conhecimento crie uma questão e as respetivas opções a apresentar ao utilizador. Esta abordagem requer algumas restrições no pedido feito ao LLM: a questão gerada deve ter como base informações adquiridas até ao fim do treino do modelo, minimizando assim a probabilidade de gerar informações falsas. No caso da base de dados retornar resultados, o LLM com suporte nestes e na questão antes construída, irá gerar as respetivas opções de resposta e justificações. Após a construção do quiz, este será enviado para *frontend* para ser apresentado ao utilizador. Ao longo do quiz, o utilizador irá selecionando a cada pergunta a opção que mais lhe parece correta, de entre 4 alternativas. Caso acerte, a resposta será apresentada a verde, caso contrário a resposta selecionada aparecerá a vermelho e a alínea correta a verde. No final, é apresentada ao utilizador a sua pontuação e a possibilidade de iniciar um novo quiz.

Toda a aplicação está personalizada com frases futebolísticas: enquanto o quiz está a ser carregado, e na apresentação da pontuação. Assim é possível cativar o utilizador e envolvê-lo no mundo futebolístico.

9.1 Análise de Requisitos

Neste subcapítulo serão descritos os requisitos definidos para o projeto do Football Quiz. Esta etapa foi fundamental no processo de desenvolvimento do projeto, pois permitiu identificar todas as funcionalidades necessárias.

9.1.1 Requisitos Funcionais

- O utilizador deve poder selecionar duas equipas de futebol distintas através daquelas existentes;
- O utilizador deve poder definir o número de perguntas a que deseja responder no quiz;

- Após a solicitação de quiz, o sistema deve realizar uma *LLM Call* para obter questões futebolísticas de jogadores e as respetivas *queries*;
- Após obter as informações acima elencadas, o sistema deve executar as *queries* na base de dados Neo4J para obter as respostas às questões geradas;
- O LLM deve ser responsável por gerar a estrutura do quiz consoante com os resultados obtidos, nomeadamente: justificação e opções a apresentar;
- O sistema deve apresentar ao utilizador o quiz, com a pergunta e 4 opções de resposta;
- Caso o jogador acerte ou erre deve ser visível que evento ocorreu e a justificação;
- Caso o jogador erre deve ser apresentada a opção correta e a respetiva justificação;
- No final do quiz deve ser apresentada a pontuação que o utilizador obteve;
- A aplicação deve ser adequada e relativa ao mundo do futebol.

9.1.2 Requisitos Não Funcionais

- O sistema deve apresentar o quiz de forma rápida e eficiente;
- A interface deve ser intuitiva e de fácil utilização;
- A interface deve ser representativa dos eventos de jogo

9.2 Diagramas

Na Figura 9-1 é apresentado o Diagrama de Casos de Uso relativo ao projeto Football Quiz.

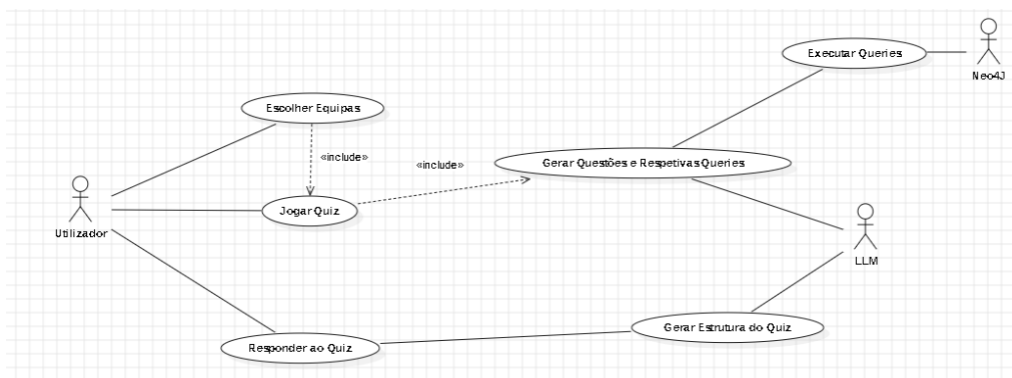


Figura 9-1 - Diagrama de Casos de Uso do Football Quiz

O diagrama de casos de uso refere os principais casos de uso do projeto Football Quiz, conforme apresentado nos anexos em I. Existem 3 atores envolvidos:

o utilizador, a base de dados e o LLM. O utilizador é quem interage diretamente com o sistema, solicitando um conjunto de questões para iniciar o quiz acerca de jogadores. O Neo4J é a base de dados utilizada para armazenar as informações dos jogadores de futebol. O LLM é o modelo responsável pela construção das *queries* e gerar as perguntas, opções e justificações.

Na Figura 9-2 é apresentado o Diagrama de Sequência relativo ao projeto Football Quiz.

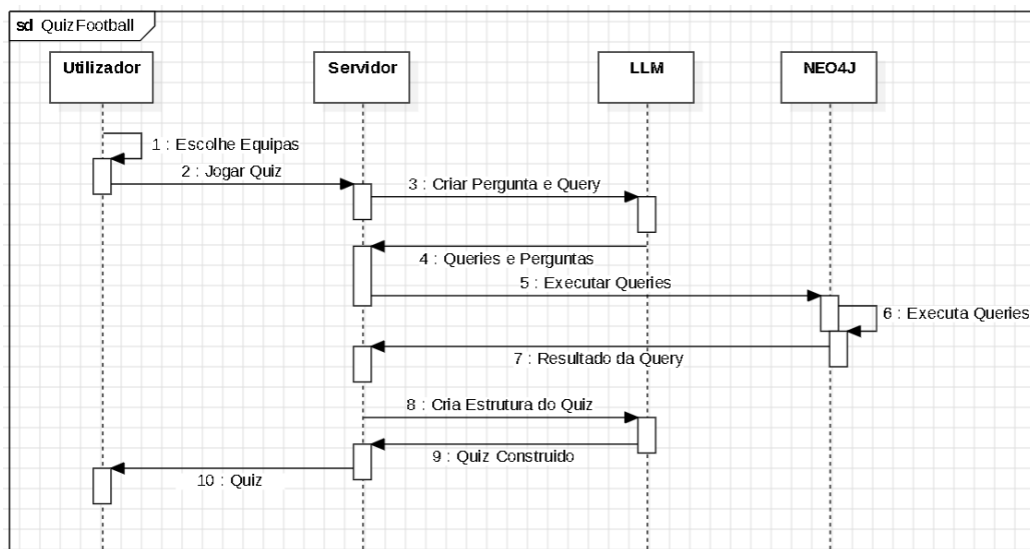


Figura 9-2 - Diagrama de Sequência do Football Quiz

Neste diagrama de sequência encontra-se descrito todo o *flow* correspondente ao projeto Football Quiz:

- O utilizador seleciona as equipas acerca das quais quer responder a um conjunto de questões acerca dos respetivos jogadores;
- O LLM de acordo com o esquema da base de dados gera um conjunto de *queries* a executar na base de dados, assim bem como a respetiva pergunta;
- As *queries* são executadas na base de dados Neo4J e obtidos os respetivos resultados;
- O LLM de acordo com as informações obtidas através da execução das *queries* gera a estrutura do quiz: pergunta, resposta correta, opções e justificações;

Com a estrutura do quiz construída este será apresentado ao utilizador para que este responda às perguntas.

9.3 Arquitetura

Na Figura 9-3 é descrito o primeiro momento da arquitetura do projeto Football Quiz.

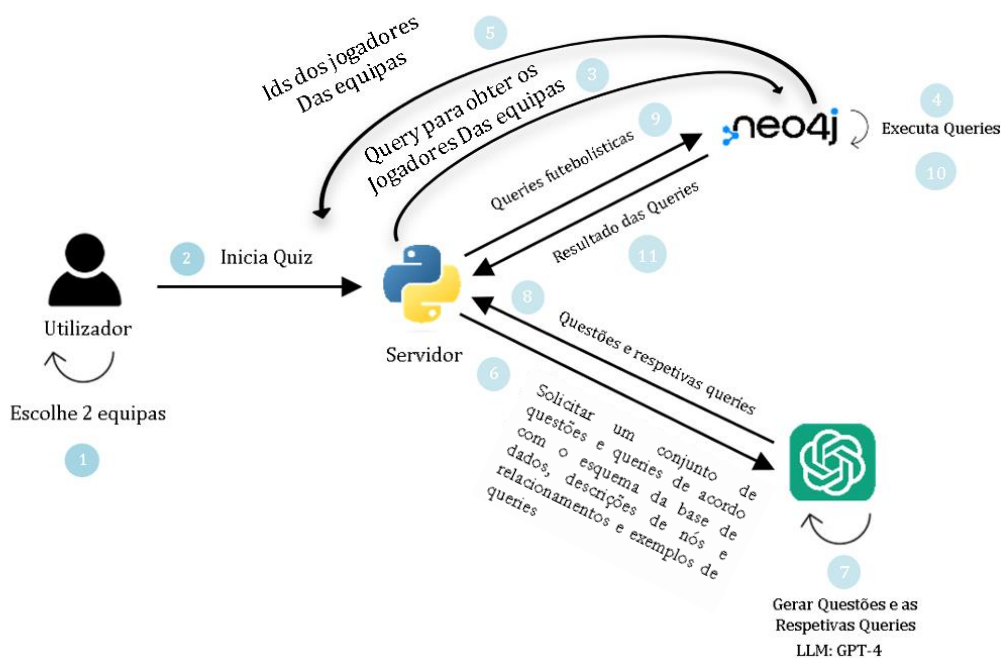


Figura 9-3 - 1º Momento da Arquitetura do Football Quiz

O utilizador, ao aceder à *web page*, começa por escrever nomes de 2 equipas (e as respetivas regiões) e definir o número de questões que pretende que o quiz tenha, com fim a obter curiosidades acerca dos jogadores (passo 1 e 2). Após submeter este formulário as informações inseridas são enviadas para *backend*, onde são obtidos os jogadores atuais das equipas através da realização de uma *query* à base de dados Neo4J (passo 3,4 e 5). De seguida, é feita uma LLM *call* (GPT-4) enviando os *ids* dos jogadores, o esquema da base de dados (com a descrição dos nós e relacionamentos e exemplos de *cypher queries*), representado nos passos 6,7 e 8. Nesta chamada é solicitada a criação de x questões futebolísticas (sendo x o nº de perguntas inserido pelo utilizador) e as respetivas *queries* em *cypher*. É requerido que o formato retornado seja:

```
a=""[{
  "id_player":,
  "question": "",
  "query": ""
}]""
```

Após a obter a resposta do LLM, esta é decomposta nos diferentes elementos, de forma a enviar a *query* para executar na base de dados Neo4J (passo 9).

Neste momento podem ocorrer 2 situações (passos 10 e 11): o resultado da base de dados é válido ou nulo.

Desenvolvimento de um protótipo de um Futebol GPT

Na Figura 9-4 está representado o 2º momento da arquitetura do Projeto Football Quiz.

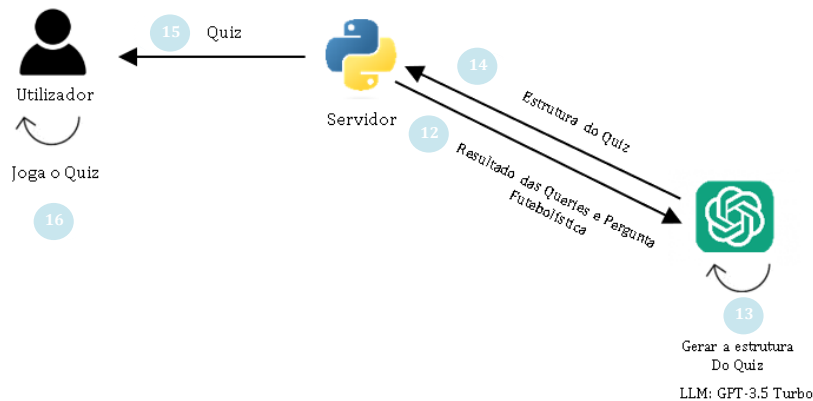


Figura 9-4 - 2º Momento da Arquitetura do Football Quiz

O fluxo das informações varia consoante o acontecimento:

1. O resultado proveniente da base de dados é válido. Com base na informação retornada é feita uma chamada ao LLM (GPT-3.5 Turbo) solicitando que de acordo com a questão e o resultado seja gerada uma estrutura do quiz (passos 12,13 e 14):

```
"question": ,
"options": [" ", "", "", ""],
"answer": 0, index of the correct answer!!
"justification": ""
```

2. O resultado vindo da base de dados é nulo, ou seja, nada é retornado. Como forma a contornar esta situação é realizada uma *LLM call* adicional onde é solicitado ao LLM que crie uma pergunta, as respetivas opções, opção correta e justificação. No entanto a questão gerada deve ser somente acerca de dados até 2022, a última data do treino do LLM, de forma a reduzir ao máximo erros e alucinações do modelo.

Após estarem criadas todas as questões e respetivas respostas, justificações, o quiz está finalizado, pronto a ser enviado para o *frontend* para ser apresentado ao utilizador. Neste momento o jogador poderá começar a responder às questões que lhe são apresentadas.

9.4 Desenvolvimento

No início do desenvolvimento, foi necessário ponderar qual o melhor fluxo, dada a restrição do *scope* apenas a jogadores. As nuances presentes no desenvolvimento do projeto encontram-se descritas neste subcapítulo.

Dado que o utilizador elege 2 equipas, aquando da solicitação do quiz ao *backend* foi optado por primeiro realizar 2 *queries* à base de dados, para obter os *ids* dos jogadores atuais de cada equipa. Esses *ids* são enviados no formato de *array* ao LLM, para que o modelo os utilize na construção das *queries*.

Neste projeto foi necessário implementar uma solução semelhante à do projeto Football Trivia para a apresentação das regiões das equipas ao utilizador no frontend: realização de uma *query* à base de dados.

Para que fosse possível obter os campos essenciais de forma mais eficaz, foi optado por solicitar ao LLM que gerasse a resposta no formato:

```
a=""[{
  "id_player": ,
  "question": "",
  "query": ""
}]""
```

Assim, é possível recuperar rapidamente o *id* do jogador, para executar uma *query* que retorne o nome do jogador. Desta forma, aquando da 2ª chamada ao LLM, no caso em que a base de dados retorna resultados válidos, é possível solicitar ao modelo que substitua o *id* do jogador pelo respetivo nome.

Ao longo do fluxo do projeto são realizadas várias e diferentes chamadas ao LLM como já foi referido na arquitetura acima e encontram-se descritas na Tabela 9-1.

Tabela 9-1 - Utilização dos LLMs

<i>LLM</i>	<i>Utilizado</i>	<i>Informações</i>
<i>GPT-4</i>	Chamada Inicial	<i>Ids</i> dos Jogadores das 2 equipas escolhidas Gerar um número de questões acerca dos jogadores e as respetivas <i>queries</i> . Formato da resposta
<i>GPT-3.5 Turbo</i>	Quando a <i>query</i> fornecida pelo LLM não gera resultados	Gerar uma curta questão de futebol num determinado formato (questão, opções, resposta correta e justificação) acerca das 2 equipas escolhidas pelo utilizador. Informações do comportamento do LLM: a questão gerada deve ter como base informações até à data do seu último treino (2022)
	Caso a <i>query</i> retorne resultados	Tendo em conta a questão gerada e o resultado retornado pela base de dados, deverá gerar uma resposta num determinado formato. Os <i>ids</i> não deverão ser apresentados. A justificação não deve referir que os dados vieram de uma base de dados.

9.5 Resultados

Neste subcapítulo são apresentados os resultados obtidos após a realização de um estudo e validação manual das informações apresentadas no quiz. Estes testes tiveram como alvo: a relação das questões colocadas com as opções apresentadas, a veracidade da resposta correta e a assertividade da justificação. No estudo foram analisados 2 modelos: o GPT-3.5 Turbo e o modelo GPT-4, na Tabela 9-2 são apresentados os resultados obtidos.

Tabela 9-2 - Estudo da Veracidade das Informações Apresentadas no Quiz

	<i>Nº de Perguntas Analisadas</i>	<i>Percentagem de Acertos (%)</i>	<i>Percentagem de Erros (%)</i>
<i>GPT-3.5 Turbo</i>	100	76%	24%
<i>GPT-4</i>	100	90%	10%

A percentagem de erros corresponde a situações em que o modelo não gerou corretamente as *Cypher queries*, sendo necessário pedir ao modelo GPT para criar uma questão de acordo com o seu conhecimento. Em L , nos anexos, encontra-se disponível a análise feita às questões apresentadas no quiz. Para a realização desta análise foram consultados os *sites* de TransferMarket [173], FootyStats [174], Wikipédia [175] e SoccerWay [176].

Da análise efetuada ao modelo da OpenAI: GPT-3.5 Turbo, observou-se que os principais problemas residiam na construção das *queries*: erro na direção dos relacionamentos, utilização de nós / atributos / relacionamentos inexistentes. Constatou-se que a chamada opcional ao LLM, nos casos em que o retorno da base de dados seja nulo, existiam alguns problemas relativos a informações “subentendidas” pelo LLM. Deste modo aquando da utilização do 2º LLM (GPT-4) foi introduzida uma restrição na *System Message* relativa à data das informações apresentadas.

No caso do modelo GPT-4, as dificuldades na geração das *queries* residem em temáticas como por exemplo: “Qual o capitão de uma equipa”, tal como já havia acontecido com o projeto SportGPT.

Adicionalmente, foi feita uma análise relativa à adequação das opções apresentadas de acordo com a pergunta realizada. Na Tabela 9-3 encontram-se os resultados obtidos.

Tabela 9-3 - Análise das Opções apresentadas nas Questões do Quiz

<i>Modelo</i>	<i>Opções de escolha incorretas no caso das perguntas terem como base dados do Neo4J</i>	<i>Opções de escolha incorretas no caso das perguntas terem como base dados do GPT</i>
<i>GPT-3.5 Turbo</i>	10	6
<i>GPT-4</i>	3	1

No caso do modelo GPT-3.5 Turbo, os principais erros ocorreram em perguntas com *queries* mal construídas pelo LLM. Isso resultou na utilização incorreta de atributos, relacionamentos ou nós, o que provocou resultados/opções incorretas. Outro problema identificado, foi a falta da substituição do *id* do jogador pelo seu nome nas questões e justificações, dificultando a identificação do jogador em causa. Para perguntas baseadas no conhecimento do GPT, houve dificuldades devido à falta de informações atualizadas / verídicas. Este problema foi mitigado com a inclusão de informações na *System Message*, de que os dados a apresentar deveriam representar informações disponíveis até à data de treino do modelo. Assim foi possível concluir da análise deste modelo que este apresentou uma taxa de sucesso nos resultados apresentados de 84%.

No caso do modelo GPT-4, as principais incorreções detetadas em perguntas com base nos dados do Neo4J incluíram a geração de perguntas gerais como “Qual o jogador que tem o maior número de presenças?”, sendo que a *query* construída teve como base somente dados de um jogador, ou então dados de somente uma *season*. Em perguntas baseadas em informações do GPT, observamos a apresentação de dados incorretos ou a referência errada a um jogador. No entanto, este modelo apresentou uma taxa de sucesso muito superior nas perguntas apresentadas: 96%.

9.6 Problemas

Ao longo do desenvolvimento, houve alguns pontos desafiadores que necessitaram de maior atenção. À semelhança do projeto SportGPT, a utilização dos modelos GPT-3.5 Turbo e GPT 4 Turbo trouxeram alguns inconvenientes: erros da direção dos relacionamentos, utilização de atributos inexistentes ou em contextos errados. Desta forma optou-se por utilizar o GPT-4, pois produziu melhores resultados.

Inicialmente, o modelo produzia questões como: “Qual o nome do jogador com *id* 1234?”, ou “Qual o jogador responsável pelo evento com *id* 1235?”, o que não era desejável. Este problema foi resolvido com a adição de restrições na *system message* quanto à utilização de *ids* apenas para a construção das *queries*.

Um aspeto a desconsiderar no esquema da base de dados, fornecido ao LLM foi inclusão de variáveis como: “*first_name*” e “*last_name*”. Pois o modelo gerava questões desnecessárias como: “Qual o primeiro nome de Cristiano Ronaldo?” ou “Qual o último nome de António Silva?”, cujas respostas são triviais. Estas perguntas

apenas seriam úteis se o primeiro/último nome do jogador não fosse o nome pelo qual é popularmente conhecido.

Nos casos em que a query gerada pelo LLM apresentava erros na sua construção ou não retornava resultados, o modelo tentava compensar. Nesses casos, o modelo listava opções vazias ou opções como “Team A, Team B, ...”. Para solucionar este problema, foi modificado o fluxo para que perante resultados vazios provenientes da base de dados fossem geradas questões baseadas somente no conhecimento do LLM.

9.7 Conclusão

O projeto Football Quiz teve como base a temática do futebol, a utilização de LLMs e da base de conhecimentos (Neo4J) à semelhança dos dois projetos anteriores. Em analogia ao projeto SportGPT neste projeto também foi requerido ao LLM que gerasse *cypher queries* de acordo com o esquema da base de dados que lhe é fornecida. No entanto, neste projeto, restringimos o tema apenas a jogadores. Ao reduzir a complexidade e a exigência necessária do LLM, facilitamos a geração de *queries*, resultando em respostas mais precisas e satisfatórias. Esta abordagem permitiu contornar os erros cometidos pelo LLM na criação de *queries* ou na falta de informações na base de dados através da solicitação ao modelo de perguntas e dos elementos adicionais com base no seu próprio conhecimento. Esta estratégia demonstrou ser eficaz, ocultando dos utilizadores os erros, e melhorando a precisão dos resultados. Assim a experiência proporcionada é contínua, sem interrupções, aumentando a confiança e satisfação sentida pelos utilizadores.

O sucesso do Football Quiz destaca a importância, em certos cenários, da adaptação e refinamento dos métodos de utilização de tecnologias como os LLMs, para contextos mais específicos. Pois ao focar num *scope* mais restrito foi possível obter resultados mais eficientes.

Andreia Filipa Almeida Graça

10 CONCLUSÕES E TRABALHOS FUTUROS

Ao longo do estágio curricular desenvolvido na WIT Software, comprovamos e atingimos o objetivo inicial: unir o mundo da GenAI com o futebol, criando aplicações distintas a nível de conteúdo apresentado. Os 3 projetos desenvolvidos: SportGPT, Football Trivia e Football Quiz, ao utilizar os mesmos dados futebolísticos armazenados na base de dados Neo4J, e os modelos GPT da OpenAI, demonstraram diferentes formas de gerar conteúdos dinâmicos e interativos.

O primeiro projeto (SportGPT), apesar de não ter alcançado os resultados desejados devido a dificuldades na geração de *Cypher queries*, proporcionou conhecimentos fundamentais para o desenvolvimento de dois projetos subsequentes. Com esse projeto aprendemos as nuances e limitações associadas aos LLMs, o que nos permitiu enfrentá-los e contorná-los de formas diferentes nos projetos seguintes. Através do refinamento e da redução das exigências inicialmente colocadas sobre o LLM, foi possível criar o Football Trivia e o Football Quiz, moldes ligeiramente diferentes, mas que fizeram uso da tecnologia de forma mais eficaz e que produziram melhores resultados. Apesar de, inicialmente, termos sido ambiciosos, aprendemos que os projetos nem sempre correm como esperado. Ao parar e analisar atentamente os problemas e as tentativas de solução, encontramos maneiras de simplificar as abordagens e obter melhores resultados. Ao diminuir a complexidade que desejamos alcançar com o LLM, conseguimos superar as dificuldades iniciais e desenvolver projetos que atenderam aos requisitos, demonstrando que com ajustes e uma visão clara das limitações, é possível obter excelentes resultados.

A experiência com LLMs demonstrou que apesar do grande potencial destes modelos, a geração de *queries* complexas para bases de dados de grafos revelou ser um grande desafio. A ambiguidade inerente à linguagem natural, a necessidade de conhecimento específico e a complexidade da tarefa foram fatores que limitaram a capacidade do modelo utilizado. Foi possível entender que a engenharia de *prompts* detém uma grande importância, pois a forma como as perguntas são formuladas tem um impacto direto na qualidade das respostas. Assim instruções mais claras e específicas tendem a gerar resultados melhores. A redução da exigência requerida do LLM, ao fornecer *queries* pré-estabelecidas, ou limitar a temática somente a jogadores, permitiu obter resultados significativamente melhores. Ao direcionar o modelo de forma mais clara foi possível reduzir a margem para respostas e ações ambíguas.

Os projetos descritos oferecem um panorama abrangente e comprovam a capacidade de juntar 2 mundos bastante distintos, para criação de conteúdos dinâmicos e informativos no contexto do futebol.

Os projetos desenvolvidos forneceram uma base sólida para futuras expansões no campo da GenAI aplicada na temática do futebol e abriram caminho para inovações contínuas. Como trabalhos futuros propomos diversas melhorias e expansões. Primeiramente, será essencial reestruturar a base de dados, de forma a

simplificá-la para utilização do LLM no caso do projeto SportGPT. O projeto SportGPT poderá abranger outros desportos para além do futebol assim bem como o Football Trivia. O Football Quiz poderá ser ampliado para cobrir diferentes temáticas dentro do futebol: treinadores, estádios, equipas, competições. O tempo de geração das perguntas também poderá ser melhorado para que seja mais célere e fluído. Outras alterações bastante relevantes no contexto do futebol são a integração com múltiplas fontes de dados para oferecer informações detalhadas sobre equipas, eventos, datas de jogos. Por fim, poderá ser implementado um sistema de atualização periódica dos dados, assegurando que todos os projetos disponham e apresentem informações atualizadas, corretas e precisas.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] “Football (soccer) | History, Game, Rules, & Significant Players | Britannica.” Accessed: Mar. 15, 2024. [Online]. Available: <https://www.britannica.com/sports/football-soccer>
- [2] “The history of football (soccer).” Accessed: Mar. 15, 2024. [Online]. Available: <https://www.footballhistory.org/>
- [3] “Mundial de futebol 2023 já superou número de adeptos nos estádios em relação a 2019, diz FIFA - Mundial Futebol Feminino - SAPO Desporto.” Accessed: Jun. 04, 2024. [Online]. Available: <https://desporto.sapo.pt/futebol/mundial-feminino/artigos/mundial-de-futebol-2023-ja-superou-numero-de-adeptos-nos-estadios-em-relacao-a-2019-diz-fifa>
- [4] “ChatGPT.” Accessed: Jun. 06, 2024. [Online]. Available: <https://chatgpt.com/auth/login?next=%2F%3Fmodel%3Dauto>
- [5] “OpenAI.” Accessed: Jun. 06, 2024. [Online]. Available: <https://openai.com/>
- [6] J. Daniel and J. H. Martin, “Speech and Language Processing Chatbots & Dialogue Systems,” 2024.
- [7] “What Are Large Language Models (LLMs)? | IBM.” Accessed: Mar. 15, 2024. [Online]. Available: <https://www.ibm.com/topics/large-language-models>
- [8] “LlamaIndex 🦙 v0.10.20.post1.” Accessed: Mar. 16, 2024. [Online]. Available: <https://docs.llamaindex.ai/en/stable/>
- [9] “LangChain.” Accessed: Mar. 16, 2024. [Online]. Available: <https://www.langchain.com/>
- [10] S. Feuerriegel, J. Hartmann, C. Janiesch, and P. Zschech, “Generative AI,” *Business and Information Systems Engineering*, vol. 66, no. 1, pp. 111–126, Feb. 2024, doi: 10.1007/S12599-023-00834-7/TABLES/2.
- [11] J. Daniel and J. H. Martin, “Speech and Language Processing,” 2024.
- [12] “The Limits of Computation | Weizenbaum Journal of the Digital Society.” Accessed: Jun. 04, 2024. [Online]. Available: <https://ojs.weizenbaum-institut.de/index.php/wjds/article/view/106/96>
- [13] “A Gentle Introduction to Generative Adversarial Networks (GANs) - MachineLearningMastery.com.” Accessed: Jun. 04, 2024. [Online]. Available: <https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/>
- [14] “O que é IA Generativa? | Databricks.” Accessed: Jun. 04, 2024. [Online]. Available: <https://www.databricks.com/br/discover/generative-ai>

- [15] “Top 100+ Generative AI Applications / Use Cases in 2024.” Accessed: Jun. 04, 2024. [Online]. Available: <https://research.aimultiple.com/generative-ai-applications/>
- [16] “What Is Generative AI? Definition, Applications, and Impact | Coursera.” Accessed: Jun. 06, 2024. [Online]. Available: <https://www.coursera.org/articles/what-is-generative-ai>
- [17] “Jasper | AI copilot for enterprise marketing teams.” Accessed: Aug. 21, 2024. [Online]. Available: <https://www.jasper.ai/>
- [18] “Gemini - Google DeepMind.” Accessed: Jun. 06, 2024. [Online]. Available: <https://deepmind.google/technologies/gemini/>
- [19] “DALL·E: Creating images from text | OpenAI.” Accessed: Jun. 06, 2024. [Online]. Available: <https://openai.com/index/dall-e/>
- [20] “Stability AI.” Accessed: Aug. 21, 2024. [Online]. Available: <https://stability.ai/>
- [21] “MuseNet | OpenAI.” Accessed: Aug. 21, 2024. [Online]. Available: <https://openai.com/index/musenet/>
- [22] “OpenAI Codex | OpenAI.” Accessed: Aug. 21, 2024. [Online]. Available: <https://openai.com/index/openai-codex/>
- [23] “GitHub Copilot · Your AI pair programmer.” Accessed: Aug. 21, 2024. [Online]. Available: <https://github.com/features/copilot>
- [24] “AI Voice Generator: Human TTS and Voiceovers | Listnr AI.” Accessed: Aug. 21, 2024. [Online]. Available: <https://listnr.ai/>
- [25] “podcast.ai.” Accessed: Aug. 21, 2024. [Online]. Available: <https://podcast.ai/>
- [26] M. Mustak, J. Salminen, M. Mäntymäki, A. Rahman, and Y. K. Dwivedi, “Deepfakes: Deceptions, mitigations, and opportunities,” *J Bus Res*, vol. 154, p. 113368, Jan. 2023, doi: 10.1016/J.JBUSRES.2022.113368.
- [27] “In 2016, Microsoft’s Racist Chatbot Revealed the Dangers of Online Conversation - IEEE Spectrum.” Accessed: Jun. 06, 2024. [Online]. Available: <https://spectrum.ieee.org/in-2016-microsofts-racist-chatbot-revealed-the-dangers-of-online-conversation>
- [28] “Generative AI: What Is It, Tools, Models, Applications and Use Cases.” Accessed: Jun. 06, 2024. [Online]. Available: <https://www.gartner.com/en/topics/generative-ai>
- [29] “How GenAI will impact the labor market | EY - Global.” Accessed: Jun. 06, 2024. [Online]. Available: https://www.ey.com/en_gl/insights/ai/how-gen-ai-will-impact-the-labor-market
- [30] “Inteligência artificial pode substituir 300 milhões de empregos em todo o mundo - CNN Portugal.” Accessed: Jun. 07, 2024. [Online]. Available: <https://cnnportugal.iol.pt/inteligencia-artificial/chatgpt/inteligencia-artificial->

- pode-substituir-300-milhoes-de-empregos-em-todo-o-mundo/20230329/6423f5c30cf2dce741b209bc
- [31] “IA vai gerar um boom de produtividade e afetar 300 milhões de empregos.” Accessed: Jun. 06, 2024. [Online]. Available: <https://pplware.sapo.pt/inteligencia-artificial/ia-vai-gerar-um-boom-de-produtividade-e-afetar-300-milhoes-de-empregos/>
- [32] S. Zhang *et al.*, “DrugAI: a multi-view deep learning model for predicting drug-target activating/inhibiting mechanisms,” *Brief Bioinform*, vol. 24, no. 1, Jan. 2023, doi: 10.1093/BIB/BBAC526.
- [33] “How AI Makes Decisions | Synechron.” Accessed: Aug. 21, 2024. [Online]. Available: <https://www.synechron.com/insight/how-ai-makes-decisions>
- [34] J. Daniel and J. H. Martin, “Speech and Language Processing,” 2024.
- [35] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, “Recent Progress on Generative Adversarial Networks (GANs): A Survey,” *IEEE Access*, vol. 7, pp. 36322–36333, 2019, doi: 10.1109/ACCESS.2019.2905015.
- [36] C. Shen, Y. Yin, X. Wang, X. Li, J. Song, and M. Song, “Training Generative Adversarial Networks in One Stage”, Accessed: Sep. 23, 2024. [Online]. Available: <https://github.com/zju-vipa/OSGAN>
- [37] “GAN Training | Machine Learning | Google for Developers.” Accessed: Sep. 23, 2024. [Online]. Available: <https://developers.google.com/machine-learning/gan/training>
- [38] “The Comprehensive Guide to Understanding Generative AI | by purpleSlate | Medium.” Accessed: Jun. 04, 2024. [Online]. Available: https://medium.com/@social_65128/the-comprehensive-guide-to-understanding-generative-ai-c06bbf259786
- [39] “Understanding Transformers & the Architecture of LLMs.” Accessed: Jun. 06, 2024. [Online]. Available: <https://blog.mlq.ai/llm-transformer-architecture/>
- [40] “What are Large Language Models? - LLM AI Explained - AWS.” Accessed: Jun. 11, 2024. [Online]. Available: https://aws.amazon.com/what-is/large-language-model/?nc1=h_ls
- [41] “LaMDA: our breakthrough conversation technology.” Accessed: Jun. 06, 2024. [Online]. Available: <https://blog.google/technology/ai/lamda/>
- [42] “Google Gemini Vs. Bard — Uncovering The Key Differences Between Gemini And Google Bard | by Jason Stathum | Nerd For Tech | Medium.” Accessed: Jun. 06, 2024. [Online]. Available: <https://medium.com/nerd-for-tech/google-gemini-vs-bard-uncovering-the-key-differences-between-gemini-and-google-bard-9a51d74bc1ed>
- [43] “Google DeepMind.” Accessed: Jul. 24, 2024. [Online]. Available: <https://deepmind.google/>

- [44] “Google Gemini vs Bard: A Comparative Study | by ANIL KUMAR | Multi-Cloud Architect | Medium.” Accessed: Jun. 06, 2024. [Online]. Available: <https://medium.com/@gcp.akp/google-gemini-vs-bard-a-comparative-study-63d9686e9b5e>
- [45] “The Building Blocks of LLMs: Vectors, Tokens and Embeddings - The New Stack.” Accessed: Jun. 06, 2024. [Online]. Available: <https://thenewstack.io/the-building-blocks-of-llms-vectors-tokens-and-embeddings/>
- [46] N. Rajaraman, J. Jiao, and K. Ramchandran, “Toward a Theory of Tokenization in LLMs,” Apr. 2024, Accessed: Jun. 06, 2024. [Online]. Available: <https://arxiv.org/abs/2404.08335v1>
- [47] J. Daniel and J. H. Martin, “Speech and Language Processing,” 2024.
- [48] “OpenAI Platform.” Accessed: Jun. 06, 2024. [Online]. Available: <https://platform.openai.com/tokenizer>
- [49] “O que é tokenização? Tipos, casos de uso e implementação | DataCamp.” Accessed: Jun. 06, 2024. [Online]. Available: <https://www.datacamp.com/pt/blog/what-is-tokenization>
- [50] “What are tokens and how to count them? | OpenAI Help Center.” Accessed: Jun. 06, 2024. [Online]. Available: <https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them>
- [51] “Token e Embedding: conceitos da IA e LLMs - BRAINS.” Accessed: Sep. 23, 2024. [Online]. Available: <https://brains.dev/2024/token-e-embedding-conceitos-da-ia-e-llms/>
- [52] “O que é tokenização? Tipos, casos de uso e implementação | DataCamp.” Accessed: Sep. 23, 2024. [Online]. Available: <https://www.datacamp.com/pt/blog/what-is-tokenization>
- [53] “Google Gemini - Apps on Google Play.” Accessed: Mar. 15, 2024. [Online]. Available: <https://play.google.com/store/apps/details?id=com.google.android.apps.bard>
- [54] “Apresentando o Gemini: nosso maior e mais hábil modelo de IA.” Accessed: Sep. 09, 2024. [Online]. Available: <https://blog.google/intl/pt-br/novidades/tecnologia/apresentando-o-gemini-nosso-maior-e-mais-habil-modelo-de-ia/#performance-estado-da-arte>
- [55] “Chatsonic API With ChatGPT Superpowers.” Accessed: Mar. 15, 2024. [Online]. Available: <https://writersonic.com/chatgpt-api-chatsonic>
- [56] “chatsonic - Android Apps on Google Play.” Accessed: Mar. 15, 2024. [Online]. Available: <https://play.google.com/store/search?q=chatsonic&c=apps>
- [57] “Football AI - Apps on Google Play.” Accessed: Mar. 15, 2024. [Online]. Available: <https://play.google.com/store/apps/details?id=com.footballscore.mobile>



- [58] “betty - The world’s first AI-powered football assistant.” Accessed: Mar. 15, 2024. [Online]. Available: <https://betty.football/>
- [59] “betty.football - Apps on Google Play.” Accessed: Mar. 15, 2024. [Online]. Available: https://play.google.com/store/apps/details?id=com.betty_better
- [60] “NerdyTips | #1 AI Football Tips Platform.” Accessed: Mar. 15, 2024. [Online]. Available: <https://nerdytips.com/>
- [61] “AI Football Tips - NerdyTips - Apps on Google Play.” Accessed: Mar. 15, 2024. [Online]. Available: <https://play.google.com/store/apps/details?id=find.yourbets.NerdyTips>
- [62] “Football Betting tips for today’s matches | BetMines.” Accessed: Mar. 15, 2024. [Online]. Available: <https://betmines.com/>
- [63] “BetMines Betting Predictions - Apps on Google Play.” Accessed: Mar. 15, 2024. [Online]. Available: <https://play.google.com/store/apps/details?id=com.betmines>
- [64] “Fidelisbet: AI Bet Predictions - Apps on Google Play.” Accessed: Mar. 17, 2024. [Online]. Available: <https://play.google.com/store/apps/details?id=com.fidelisbetprd.fidelisbet>
- [65] “FidelisBET - Aim Higher.” Accessed: Mar. 17, 2024. [Online]. Available: <https://www.fidelisbet.com/maintenance>
- [66] “Alstats - Football Predictions - Apps on Google Play.” Accessed: Mar. 17, 2024. [Online]. Available: https://play.google.com/store/apps/details?id=com.aistats.ai_stats
- [67] “What is a ‘corpus’? And why is everyone in AI talking about it?” Accessed: Jun. 06, 2024. [Online]. Available: <https://www.fastcompany.com/90916291/what-is-a-corpus-ai-corpora-chatgpt>
- [68] “Dirty Secrets of BookCorpus, a Key Dataset in Machine Learning | by Jack Bandy | Towards Data Science.” Accessed: Jun. 06, 2024. [Online]. Available: <https://towardsdatascience.com/dirty-secrets-of-bookcorpus-a-key-dataset-in-machine-learning-6ee2927e8650>
- [69] “Common Crawl And Unlocking Web Archives For Research.” Accessed: Jun. 06, 2024. [Online]. Available: <https://www.forbes.com/sites/kalevleetaru/2017/09/28/common-crawl-and-unlocking-web-archives-for-research/>
- [70] “What is a Web Crawler? | A Comprehensive Web Crawling Guide | Elastic.” Accessed: Jun. 06, 2024. [Online]. Available: <https://www.elastic.co/what-is/web-crawler/>
- [71] “O Que é GPT-4? Principais Fatos e Funcionalidades.” Accessed: Sep. 09, 2024. [Online]. Available: <https://pt.semrush.com/blog/o-que-e-gpt-4/>

- [72] “What are tokens and how to count them? | OpenAI Help Center.” Accessed: Sep. 09, 2024. [Online]. Available: <https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them>
- [73] “What is your Context Window?. The context window for a large language... | by Amir Aryani | Medium.” Accessed: Jun. 06, 2024. [Online]. Available: <https://medium.com/@amiraryani/what-is-your-context-window-772b1c65b881>
- [74] “What is SQL? - Structured Query Language (SQL) Explained - AWS.” Accessed: Jun. 06, 2024. [Online]. Available: https://aws.amazon.com/what-is/sql/?nc1=h_ls
- [75] “What is an RDBMS (relational database management system)? | Definition from TechTarget.” Accessed: Jun. 06, 2024. [Online]. Available: <https://www.techtarget.com/searchdatamanagement/definition/RDBMS-relational-database-management-system>
- [76] “What are the Advantages of SQL? Overview.” Accessed: Sep. 23, 2024. [Online]. Available: <https://www.theknowledgeacademy.com/blog/advantages-of-sql/>
- [77] “Difference Between SQL and NoSQL - Naukri Code 360.” Accessed: Sep. 23, 2024. [Online]. Available: <https://www.naukri.com/code360/library/pros-and-cons-of-using-sql-vs-nosql-databases>
- [78] “Dados não relacionais e NoSQL - Azure Architecture Center | Microsoft Learn.” Accessed: Jun. 07, 2024. [Online]. Available: <https://learn.microsoft.com/pt-pt/azure/architecture/data-guide/big-data/non-relational-data>
- [79] “Bancos de dados relacionais e não relacionais — Diferença entre tipos de bancos de dados — AWS.” Accessed: Jun. 07, 2024. [Online]. Available: <https://aws.amazon.com/pt/compare/the-difference-between-relational-and-non-relational-databases/>
- [80] “Sharding vs. partitioning: What’s the difference? — PlanetScale.” Accessed: Sep. 23, 2024. [Online]. Available: <https://planetscale.com/learn/articles/sharding-vs-partitioning-whats-the-difference>
- [81] “DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems.” Accessed: Sep. 23, 2024. [Online]. Available: <https://db-engines.com/en/>
- [82] “Neo4j System Properties.” Accessed: Jun. 07, 2024. [Online]. Available: <https://db-engines.com/en/system/Neo4j>
- [83] “Memgraph System Properties.” Accessed: Jun. 07, 2024. [Online]. Available: <https://db-engines.com/en/system/Memgraph>
- [84] “NebulaGraph System Properties.” Accessed: Jun. 07, 2024. [Online]. Available: <https://db-engines.com/en/system/NebulaGraph>
- [85] “JanusGraph System Properties.” Accessed: Jun. 07, 2024. [Online]. Available: <https://db-engines.com/en/system/JanusGraph>

- [86] D. Fernandes and J. Bernardino, “Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB,” 2018, doi: 10.5220/0006910203730380.
- [87] “What is Neo4j (Graph Database)? Complete Overview of Neo4j.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.graphable.ai/software/what-is-neo4j-graph-database/>
- [88] “Cypher and Neo4j - Cypher Manual.” Accessed: Sep. 23, 2024. [Online]. Available: <https://neo4j.com/docs/cypher-manual/current/introduction/cypher-neo4j/>
- [89] “Neo4j: Features, Install, Advantage & More.” Accessed: Sep. 23, 2024. [Online]. Available: <https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-neo4j-graph-database/>
- [90] “OpenAI Codex | OpenAI.” Accessed: Jun. 07, 2024. [Online]. Available: <https://openai.com/index/openai-codex/>
- [91] “GitHub Copilot · Your AI pair programmer · GitHub.” Accessed: Jun. 07, 2024. [Online]. Available: <https://github.com/features/copilot>
- [92] “Meta Llama.” Accessed: Jun. 07, 2024. [Online]. Available: <https://llama.meta.com/>
- [93] “Models - OpenAI API.” Accessed: Jun. 07, 2024. [Online]. Available: <https://platform.openai.com/docs/models/gpt-4o>
- [94] G. Bansal, V. Chamola, A. Hussain, M. Guizani, and D. Niyato, “Transforming Conversations with AI—A Comprehensive Study of ChatGPT,” *Cognit Comput*, vol. 16, no. 5, pp. 2487–2510, Sep. 2024, doi: 10.1007/S12559-023-10236-2/TABLES/5.
- [95] A. R. Openai, K. N. Openai, T. S. Openai, and I. S. Openai, “Improving Language Understanding by Generative Pre-Training”, Accessed: Jun. 08, 2024. [Online]. Available: <https://gluebenchmark.com/leaderboard>
- [96] “GPT-1, GPT-2 & GPT-3: Learn the Evolution of AI Language Models.” Accessed: Jun. 07, 2024. [Online]. Available: <https://360digitmg.com/blog/types-of-gpt-in-artificial-intelligence>
- [97] “GPT1, GPT2 and GPT3: All probabilistic generative models! | by Ezekiel Zhao | Medium.” Accessed: Jun. 07, 2024. [Online]. Available: <https://medium.com/@2468086464/gpt-1-gpt-2-and-gpt3-what-are-the-differences-bc7d8284a0b8>
- [98] “GPT-3 powers the next generation of apps | OpenAI.” Accessed: Jun. 07, 2024. [Online]. Available: <https://openai.com/index/gpt-3-apps/>
- [99] “Models - OpenAI API.” Accessed: Sep. 23, 2024. [Online]. Available: <https://platform.openai.com/docs/models/gpt-4o>

- [100] “GPT-3.5 Turbo fine-tuning and API updates | OpenAI.” Accessed: Sep. 23, 2024. [Online]. Available: <https://openai.com/index/gpt-3-5-turbo-fine-tuning-and-api-updates/>
- [101] “GPT-4 | OpenAI.” Accessed: Jun. 07, 2024. [Online]. Available: <https://openai.com/index/gpt-4/>
- [102] “GPT-4 Turbo in the OpenAI API | OpenAI Help Center.” Accessed: Jun. 07, 2024. [Online]. Available: <https://help.openai.com/en/articles/8555510-gpt-4-turbo-in-the-openai-api>
- [103] “GPT-4 Turbo vs. GPT-4 - What’s the Difference?” Accessed: Sep. 23, 2024. [Online]. Available: <https://textcortex.com/post/gpt-4-turbo-vs-gpt-4>
- [104] Z. Ji, T. Yu, Y. Xu, N. Lee, E. Ishii, and P. Fung, “Towards Mitigating Hallucination in Large Language Models via Self-Reflection,” pp. 1827–1843, Accessed: Jun. 07, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Noonan_syndrome.
- [105] Y. Zhang *et al.*, “Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models,” Sep. 2023, Accessed: Sep. 23, 2024. [Online]. Available: <https://arxiv.org/abs/2309.01219v2>
- [106] S. Towhidul Islam Tonmoy *et al.*, “A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models”.
- [107] “O que é RLHF? — Explicação sobre aprendizado por reforço com base em feedback humano — AWS.” Accessed: Jun. 08, 2024. [Online]. Available: <https://aws.amazon.com/pt/what-is/reinforcement-learning-from-human-feedback/>
- [108] N. F. Liu *et al.*, “Lost in the Middle: How Language Models Use Long Contexts”.
- [109] “What Is Prompt Engineering? | IBM.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.ibm.com/topics/prompt-engineering>
- [110] “ChatGPT vs Google Bard: A Comparative Guide to AI Chatbots | DataCamp.” Accessed: Sep. 10, 2024. [Online]. Available: <https://www.datacamp.com/blog/bard-vs-chat-gpt>
- [111] “Os prompts do CHATGPT também funcionam para o Bard? - Virtualnauta.” Accessed: Sep. 10, 2024. [Online]. Available: <https://virtualnauta.pt/os-prompts-do-chatgpt-tambem-funcionam-para-o-bard/>
- [112] “What Is Zero-Shot Learning? | IBM.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.ibm.com/topics/zero-shot-learning>
- [113] “Zero-Shot Prompting | Prompt Engineering Guide.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.promptingguide.ai/pt/techniques/zeroshot>
- [114] “Few-Shot Prompting | Prompt Engineering Guide.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.promptingguide.ai/pt/techniques/fewshot>

- [115] “What Is Few-Shot Learning? | IBM.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.ibm.com/topics/few-shot-learning>
- [116] “Chain-of-Thought Prompt | Prompt Engineering Guide.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.promptingguide.ai/pt/techniques/cot>
- [117] “What is RAG? - Retrieval-Augmented Generation Explained - AWS.” Accessed: Jun. 07, 2024. [Online]. Available: https://aws.amazon.com/what-is/retrieval-augmented-generation/?nc1=h_ls
- [118] “RAG vs Graph RAG. A detailed architecture of both RAG and... | by Azharudeen | Medium.” Accessed: Sep. 10, 2024. [Online]. Available: <https://medium.com/@azharudeen2020/rag-vs-graph-rag-edd2f35761dc>
- [119] “What is Retrieval Augmented Generation (RAG)? | A Comprehensive RAG Guide | Elastic.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.elastic.co/what-is/retrieval-augmented-generation/>
- [120] “What is RAG? - Retrieval-Augmented Generation Explained - AWS.” Accessed: Jun. 07, 2024. [Online]. Available: https://aws.amazon.com/what-is/retrieval-augmented-generation/?nc1=h_ls
- [121] “O que é RAG (Retrieval-Augmented Generation)?” Accessed: Jun. 07, 2024. [Online]. Available: <https://triggo.ai/blog/o-que-e-retrieval-augmented-generation/>
- [122] J. Daniel and J. H. Martin, “Speech and Language Processing,” 2024.
- [123] “Geração com Recuperação Aprimorada (RAG) | Prompt Engineering Guide.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.promptingguide.ai/pt/techniques/rag>
- [124] “What is a Vector Database? | A Comprehensive Vector Database Guide | Elastic.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.elastic.co/what-is/vector-database/>
- [125] “RAG Vs Fine Tuning: How To Choose The Right Method.” Accessed: Sep. 23, 2024. [Online]. Available: <https://www.montecarlodata.com/blog-rag-vs-fine-tuning/>
- [126] “Visual Studio Code - Code Editing. Redefined.” Accessed: Jun. 07, 2024. [Online]. Available: <https://code.visualstudio.com/>
- [127] “GitHub: Let’s build from here · GitHub.” Accessed: Jun. 07, 2024. [Online]. Available: <https://github.com/>
- [128] “Postman API Platform | Sign Up for Free.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.postman.com/>
- [129] “OpenAI Platform.” Accessed: Jun. 11, 2024. [Online]. Available: <https://platform.openai.com/usage>
- [130] “Neo4j Desktop - Neo4j Desktop.” Accessed: Jun. 07, 2024. [Online]. Available: <https://neo4j.com/docs/desktop-manual/current/>

- [131] “pgAdmin - PostgreSQL Tools.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.pgadmin.org/>
- [132] “Project Jupyter | Home.” Accessed: Jun. 07, 2024. [Online]. Available: <https://jupyter.org/>
- [133] “ngrok | Unified Application Delivery Platform for Developers.” Accessed: Jun. 07, 2024. [Online]. Available: <https://ngrok.com/>
- [134] “Our Documentation | Python.org.” Accessed: Jun. 07, 2024. [Online]. Available: <https://www.python.org/doc/>
- [135] “Welcome to Flask — Flask Documentation (3.0.x).” Accessed: Jun. 07, 2024. [Online]. Available: <https://flask.palletsprojects.com/en/3.0.x/>
- [136] “Query a Neo4j database using Cypher - Getting Started.” Accessed: Jul. 25, 2024. [Online]. Available: <https://neo4j.com/docs/getting-started/cypher-intro/>
- [137] “Fast and Reliable Sports Data for your web app | Sportmonks.” Accessed: Mar. 23, 2024. [Online]. Available: <https://www.sportmonks.com/>
- [138] “Function calling - OpenAI API.” Accessed: Mar. 16, 2024. [Online]. Available: <https://platform.openai.com/docs/guides/function-calling>
- [139] “Chat Completions API - OpenAI API.” Accessed: Jul. 25, 2024. [Online]. Available: <https://platform.openai.com/docs/guides/chat-completions>
- [140] J. Wallat, A. Jatowt, and A. Anand, “Temporal Blind Spots in Large Language Models,” Jan. 2024, Accessed: Sep. 29, 2024. [Online]. Available: <https://arxiv.org/abs/2401.12078v1>
- [141] “What Are the Limitations of Large Language Models (LLMs)? - PromptDrive.ai.” Accessed: Sep. 29, 2024. [Online]. Available: <https://promptdrive.ai/llm-limitations/>
- [142] “PostgreSQL: The world’s most advanced open source database.” Accessed: Jul. 25, 2024. [Online]. Available: <https://www.postgresql.org/>
- [143] “Dados desportivos rápidos e fiáveis | Sportmonks.” Accessed: Jun. 11, 2024. [Online]. Available: <https://www.sportmonks.com/pt-pt/>
- [144] “How to Finetune the Entire RAG Architecture (including DPR retriever) | by Shamane Siriwardhana | Medium.” Accessed: Mar. 16, 2024. [Online]. Available: <https://shamanesiri.medium.com/how-to-finetune-the-entire-rag-architecture-including-dpr-retriever-4b4385322552>
- [145] “RAG and generative AI - Azure AI Search | Microsoft Learn.” Accessed: Mar. 16, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/search/retrieval-augmented-generation-overview>
- [146] “Streaming |   Langchain.” Accessed: Mar. 23, 2024. [Online]. Available: https://python.langchain.com/docs/expression_language/streaming#chains

- [147] “Regression vs Classification in Machine Learning Explained!” Accessed: Jun. 10, 2024. [Online]. Available: <https://www.analyticsvidhya.com/blog/2023/05/regression-vs-classification/>
- [148] “KNeighborsClassifier — scikit-learn 1.5.0 documentation.” Accessed: Jun. 10, 2024. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
- [149] “What is the k-nearest neighbors algorithm? | IBM.” Accessed: Jun. 10, 2024. [Online]. Available: <https://www.ibm.com/topics/knn>
- [150] “What Is Linear Regression? | IBM.” Accessed: Jun. 10, 2024. [Online]. Available: <https://www.ibm.com/topics/linear-regression>
- [151] “LinearRegression — scikit-learn 1.5.0 documentation.” Accessed: Jun. 10, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- [152] “What Is Support Vector Machine? | IBM.” Accessed: Jun. 10, 2024. [Online]. Available: <https://www.ibm.com/topics/support-vector-machine>
- [153] “1.4. Support Vector Machines — scikit-learn 1.5.0 documentation.” Accessed: Jun. 10, 2024. [Online]. Available: <https://scikit-learn.org/stable/modules/svm.html>
- [154] “What is a Decision Tree? | IBM.” Accessed: Jun. 10, 2024. [Online]. Available: <https://www.ibm.com/topics/decision-trees>
- [155] “1.10. Decision Trees — scikit-learn 1.5.0 documentation.” Accessed: Jun. 10, 2024. [Online]. Available: <https://scikit-learn.org/stable/modules/tree.html>
- [156] “GradientBoostingRegressor — scikit-learn 1.5.0 documentation.” Accessed: Jun. 10, 2024. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html>
- [157] “What Is Boosting? | IBM.” Accessed: Jun. 10, 2024. [Online]. Available: <https://www.ibm.com/topics/boosting>
- [158] “Correlation in machine learning—All you need to know | by Abdallah Ashraf | Medium.” Accessed: Jun. 10, 2024. [Online]. Available: <https://medium.com/@abdallahashraf90x/all-you-need-to-know-about-correlation-for-machine-learning-e249fec292e9>
- [159] “OneHotEncoder — scikit-learn 1.5.0 documentation.” Accessed: Jun. 10, 2024. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder>
- [160] “Null Values Imputation (All Methods) | Kaggle.” Accessed: Jun. 10, 2024. [Online]. Available: <https://www.kaggle.com/discussions/general/248836>

- [161] “Previendo Números: Entendendo as métricas R², MAE, MAPE, MSE e RMSE | by Clébio de Oliveira Júnior | Data Hackers | Dec, 2021 | Medium | Data Hackers.” Accessed: Jun. 10, 2024. [Online]. Available: <https://medium.com/data-hackers/prevendo-n%C3%B0meros-entendendo-m%C3%A9tricas-de-regress%C3%A3o-35545e011e70>
- [162] “Text to speech - OpenAI API.” Accessed: Jun. 08, 2024. [Online]. Available: <https://platform.openai.com/docs/guides/text-to-speech>
- [163] “Speech to text - OpenAI API.” Accessed: Jun. 08, 2024. [Online]. Available: <https://platform.openai.com/docs/guides/speech-to-text/prompting>
- [164] “How to use the OpenAI Text-to-Speech API | DataCamp.” Accessed: Jun. 08, 2024. [Online]. Available: <https://www.datacamp.com/tutorial/how-to-use-the-openai-text-to-speech-api>
- [165] “Hello GPT-4o | OpenAI.” Accessed: Jul. 22, 2024. [Online]. Available: <https://openai.com/index/hello-gpt-4o/>
- [166] “OpenAI lança modelo GPT-4o que vai estar disponível gratuitamente. E ChatGPT ganha nova aplicação para computador – Observador.” Accessed: Jul. 22, 2024. [Online]. Available: <https://observador.pt/2024/05/13/openai-lanca-modelo-gpt-4o-que-vai-estar-disponivel-gratuitamente-e-chatgpt-ganha-nova-versao-para-usar-no-computador/>
- [167] “ChatGPT-4 browsing: o que é e como usar a navegação na web.” Accessed: Jul. 22, 2024. [Online]. Available: <https://olhardigital.com.br/2023/05/18/internet-e-redes-sociais/chatgpt-4-browsing-o-que-e-e-como-funciona-o-chatbot-que-acessa-a-web/>
- [168] “OptaJoe (@OptaJoe) / X.” Accessed: Jul. 22, 2024. [Online]. Available: <https://x.com/optajoe>
- [169] “OptaJoao (@OptaJoao) / X.” Accessed: Jul. 22, 2024. [Online]. Available: <https://x.com/optajoao>
- [170] “OptaPaolo (@OptaPaolo) / X.” Accessed: Jul. 22, 2024. [Online]. Available: <https://x.com/optapaolo>
- [171] “OptaJean (@OptaJean) / X.” Accessed: Jul. 22, 2024. [Online]. Available: <https://x.com/optajejan>
- [172] “OptaFranz (@OptaFranz) / X.” Accessed: Jul. 22, 2024. [Online]. Available: <https://x.com/optafranz>
- [173] “OptaJose (@OptaJose) / X.” Accessed: Jul. 22, 2024. [Online]. Available: <https://x.com/optajose>
- [174] “Squawka (@Squawka) / X.” Accessed: Jul. 22, 2024. [Online]. Available: <https://x.com/squawka>

- [175] “Mercado de transferências, rumores, Valores de mercado e notícias | Transfermarkt.” Accessed: Jul. 22, 2024. [Online]. Available: <https://www.transfermarkt.pt/>
- [176] “Football Stats, Tables & Results | Soccer Stats - FootyStats.” Accessed: Jul. 22, 2024. [Online]. Available: <https://footystats.org/>
- [177] “Wikipédia, a enciclopédia livre.” Accessed: Jul. 22, 2024. [Online]. Available: https://pt.wikipedia.org/wiki/Wikip%C3%A9dia:P%C3%A1gina_principal
- [178] “Resultados ao vivo, calendários, tabelas, estatísticas e notícias - Soccerway.” Accessed: Jul. 22, 2024. [Online]. Available: <https://pt.soccerway.com/>

ANEXOS E/OU APÊNDICES

A. Proposta de Estágio

Ano Letivo de 2023/2024

Título	Desenvolvimento de um protótipo de um Futebol GPT
Aluno	Andreia Graça
Empresa	WIT Software
Supervisor	Nuno Carvalho, nuno.carvalho@wit-software.com
Local de trabalho	Instalações da empresa em Taveiro (Coimbra).

SUMÁRIO

Esta proposta de estágio tem como objetivo implementar um demonstrador do uso de Generative AI para processar dados históricos. Neste caso, vamos usar dados históricos relacionados com jogos de futebol em vários países, várias competições e clubes, desde 2016 até 2022. O demonstrador deverá permitir aos utilizadores uma experiência interativa e personalizada, utilizando técnicas de processamento de linguagem natural e algoritmos de Inteligência Artificial para entender e responder a perguntas dos utilizadores de uma forma inteligente e avançada. Alguns exemplos de perguntas que o software de Generative AI deverá saber responder:

Quantos golos marcou o Morita na Primeira Liga?

Quem marcou mais golos?

Em que posição está o Sporting na Taça da Liga?

Quantas competições o Sporting ganhou nos últimos 10 anos?

Quero saber curiosidades sobre o meu clube.

1. Âmbito

O âmbito deste estágio compreende a implementação de um software do estilo ChatGPT mas focado em dados históricos de futebol. Por esse motivo, vamos chamar o demonstrador de GPT-Futebol. O estágio é justificado pela relevância e

atualidade do tema, bem como pela procura crescente de soluções inovadoras que combinem a tecnologia de AI e desporto. A dimensão do projeto é adequada para um estágio de mestrado, permitindo explorar a complexidade e a novidade das técnicas de inteligência artificial aplicadas ao domínio específico do futebol.

2. Objetivos

O presente estágio pretende atingir os seguintes objetivos genéricos:

Pesquisar e obter dados relevantes sobre futebol, incluindo informações sobre equipas, jogadores, competições, jogos, notícias e curiosidades, de fontes confiáveis e atualizadas.

Desenhar e desenvolver uma arquitetura de software que utilize técnicas avançadas de inteligência artificial para interagir de forma inteligente com os utilizadores.

Implementar o demonstrador, que irá ter dois grandes componentes: um servidor aplicacional e uma simples aplicação *mobile* para Android e iPhone, desenvolvida em Flutter.

Treinar modelos de linguagem a partir dos dados obtidos/guardados, procurando compreender e responder de forma precisa e relevante às perguntas dos utilizadores.

Realizar testes e avaliações para verificar a eficácia das respostas do GPT-Futebol.

Realizar melhorias contínuas, aprimorar as funcionalidades e ajustar as respostas com base nas necessidades e feedback recebido dos utilizadores.

3. Plano de trabalhos

O estágio consistirá nas seguintes atividades e respetivas tarefas:

T1 – Estado da Arte e Tecnologia – Aquisição de conhecimento na área da Inteligência artificial e tecnologias a utilizar no projeto.

T2 – Investigação das fontes de dados na área do futebol – Serão disponibilizados dados de futebol desde 2016 relacionados com competições, jogos, equipas e jogadores. No entanto, pretende-se enriquecer esses dados com informação adicional. Como tal, deverá ser realizado esse levantamento, identificar APIs e definir um plano para o cruzamento desses dados.

T3 – Implementação do Servidor Aplicacional – Esta tarefa inclui o desenho da solução servidor e implementação do mesmo, incluindo uma interface GPT para receber prompts do utilizador. Deverá ainda ter a capacidade de gerar conteúdo automático relacionado com curiosidades do desporto. O conteúdo gerado deve ser em texto. A geração de respostas em imagens/gráficos é um nice-to-have, mas se for conseguido será um bom exemplo de superação.

T4 – Implementação da Aplicação *Mobile* – Esta tarefa inclui a implementação de uma aplicação em Flutter com uma interface de conversação, integrada com o servidor aplicacional, para demonstração das capacidades do GPT-Futebol.

T5 – Optimização e refinamento - Esta tarefa tem como objetivo a melhoria contínua e iterativa do software desenvolvido (servidor e aplicação *mobile*) com base em feedback recebido.

Serão produzidos os seguintes documentos ou materiais:

E1 – Estado da arte: Powerpoint com as conclusões e recomendações tecnológicas para o projeto.

E2 – Arquitetura da solução: Powerpoint com a proposta de implementação do servidor e aplicação *mobile*. Deve incluir vários diagramas de arquitetura e diagramas de sequência.

E3 – MVP (servidor): Demonstração das capacidades do servidor aplicacional.

E4 – MVP (aplicação *mobile*): Demonstração end-to-end com utilização da aplicação *mobile*.

A calendarização prevista para os trabalhos e entregas será a seguinte:

	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul
Tarefas											
T1											
T2											
T3											
T4											
T5											
Entregas											
E1		X									
E2			X								
E3				X	X				X	X	
E4						X	X	X	X	X	

4. Metodologia de Trabalho

Serão utilizadas metodologias ágeis. O acompanhamento do estágio será feito por um orientador e um tutor técnico: o orientador define os requisitos do estágio, as prioridades do Backlog e acompanha os resultados intercalares do projecto. O tutor dará o apoio necessário ao estagiário no dia-a-dia e promove as reuniões semanais.

5. Caracterização do Estágio

O horário será a tempo completo.

Durante o estágio o aluno terá ao seu dispor todos os equipamentos necessários para desempenhar as suas tarefas.

O estágio é remunerado. Se o desempenho do aluno ao longo do mês for positivo, terá direito a receber uma bolsa mensal.

Terá ainda acesso às formações da WIT Academy.


Será feita uma avaliação do desempenho no final do estágio. Se esta avaliação for positiva será feita uma proposta ao aluno para que possa vir a fazer parte da empresa.





Sobre a Empresa:

A WIT tem 20 anos de experiência na área das telecomunicações e é uma empresa francamente exportadora. Prova disso é o facto do software que desenvolve já ter sido exportado para 40 países. A empresa conta com os seguintes clientes: Grupo Vodafone, Deutsche Telekom, NTT DoCoMo, KDDI, Softbank, AT&T, Verizon, Safaricom, Vodacom, Unitel, entre outros. Na WIT somos Groundbreakers e procuramos, todos os dias, desafiar o status quo fazendo mais e diferente, sempre com o objectivo de sermos cada vez melhores. Descobre aqui se também tu podes ser um groundbreaker:


<https://www.youtube.com/watch?v=Mt9sCqvJx9U>

B. Comparação de Ferramentas e Aplicações de GenAI/ Futebol

Aplicação	Propósito	Nº de campeonatos	Possibilidade de conversação	Idiomas	Apenas sobre futebol	API	Desvantagens	Preço	Vantagens
	Fazer previsões de jogos. Auxiliar os adeptos a terem mais chance nas suas apostas	Os mais conhecidos	Sim, Com IA e PLN	40	✗	✓	Incerteza Tendencioso	Não aplicável	Abrange várias temáticas Facilidade de uso
	Assistência na escrita, responder a questões e criar conteúdo de forma criativa e com qualidade	Os mais conhecidos	Sim, Com IA e PLN	+20	✗	✓	Limitado na originalidade Tendencioso Incerteza	Depende das necessidades	Capaz de gerar conteúdo personalizado e original Abrange várias temáticas.
	Auxiliar os adeptos de futebol a fazerem as suas apostas tendo em conta previsões com IA	+20 (40 até ao final de 2023)	Não	8	✓	✗	Algumas ligas e campeonatos Falta de precisão	Não aplicável	Combina <i>insights</i> em tempo real com análises baseadas em dados Fornece visão abrangente dos dados

 Football AI	Fazer previsões de jogos. Auxiliar os adeptos a terem mais chance nas suas apostas	+100	Entre utilizadores	11	✓	✗	Algumas ligas e campeonatos Falta de precisão	Não aplicável	Utiliza várias informações para fazer previsões mais assertivas
 NerdyTips	Fornecer dicas de apostas baseadas em análises estatísticas. Auxiliar os utilizadores a fazer apostas assertivas	+160	Não	12	✓	✗	Algumas ligas e campeonatos Falta de precisão	Para aceder a todos os recursos é preciso subscrição	A app oferece conteúdo exclusivo: análises, estatísticas, previsões
 BetMines	Auxilia utilizadores a tomar decisões informadas acerca das suas apostas	+150	Não	10	✓	✗	Algumas ligas e campeonatos. Falta de precisão	Para aceder a todos os recursos é preciso subscrição	Através de análises estatísticas fornece dicas de apostas
 FidesliBet	Aplicação de previsão de resultados futebolísticos através de algoritmos de IA	+ 50	Não	17	✓	✗	Alguns incerteza associada	Para criação de algoritmos de IA	Permite aos utilizadores criarem os seus próprios algoritmos para previsão de eventos de jogo

Desenvolvimento de um protótipo de um Futebol GPT

 AIStats	App de previsão de resultados futebolísticos através de algoritmos de IA	Principais ligas e campeonatos	Não	-	✓	✗	Alguna incerteza associada	Para aceder a todos os recursos é preciso subscrição	Abrange várias ligas e campeonatos. Aprendizagem contínua
--	--	--------------------------------	-----	---	---	---	----------------------------	--	--

C. Esquema da Base de Dados Neo4J

Entidades e Atributos:

coach: id, name, fullname, nationality_id, birth_date, birth_place, first_name, last_name, public_id

coach_match: coach_id, match_id, total_points, points_details, season_team_id

competition: id, name, type, format, shortname, region_id, rank, is_popular, enable, rank_global, continent_id, public_id

continent: id, name, code, external_id

external_coach: external_id, coach_id

external_competition: external_id, competition_id

external_group: external_id, group_id

external_match: external_id, match_id

external_match_event: external_id, match_event_id

external_player: external_id, player_id

external_referee: external_id, referee_id

external_region: external_id, region_id

external_round: external_id, round_id

external_season: external_id, season_id

external_stage: external_id, stage_id

external_team: external_id, team_id

external_venue: external_id, venue_id

fastest_scorer: id, start_date, end_date, status, creation_date, modification_date

group: id, name, stage_id

group_team: group_id, season_team_id

group_team_stage_standings: id, stage_standings_id, season_team_id, position, games_played, games_won, games_lost, games_drawn, goals_scored, goals_against, points, group_id

match_event: id, match_id, team_id, type, minute, extra_minute, player_id, other_player_id, timestamp, phase, player_name, other_player_name, result

match_info: match_id, season_team_id, goals_scored, ball_possession, dangerous_attacks, attacks, ball_safe, total_shots, shots_on_target, shots_off_target, penalties, corners, free_kicks, offsides, passes,

Desenvolvimento de um protótipo de um Futebol GPT

goalkeeper_saves, shots_blocked, goal_kicks, throw_ins, fouls, yellow_cards, red_cards, color, formation, coach_id, penalty_shootout_goals, half_time_goals_scored, pass_accuracy, shot_accuracy, substitutions, goal_attempts, injuries, tackles

matches: id, round_id, date, venue, home_season_team_id, away_season_team_id, status, first_official_referee_id, first_assistant_referee_id, second_assistant_referee_id, fourth_official_referee_id, stage_id, minute, second, extra_time, extra_minute, group_id, started_at, ended_at, teams_switched, public_id

player_career: id, player_id, team_id, start_date, end_date, transfer_id, season_id

player_matches: player_id, season_team_id, match_id, situation, is_captain, position, field_position, sideline_reason, pos_x, pos_y, number, total_points, team_id, red_cards, yellow_cards, won, total, drawn, committed, scored, assists, conceded, owngoals, shots_total, shots_on_goal, passes, key_passes, total_crosses, accurate_passes, passes_accuracy, crosses_accuracy, dribbles_success, dribbles_attempts, dribbled_past, saves, blocks, pen_won, punches, tackles, offsides, minutes_played, rating

player_season_team_stats: player_id, season_team_id, appearances, goals, assists, yellow_cards, red_cards, position_id, number, captain, minutes, lineups, substitute_in, substitute_out, substitutes_on_bench, injured, owngoals, saves, inside_box_saves, dispossessed, interceptions, tackles, blocks, hit_post, rating, committed, drawn, total, accurate, attempts, success, dribbled_past, duels_total, duels_won, passes_total, passes_accuracy, key_passes, penalties_won, penalties_scores, penalties_missed, penalties_committed, penalties_saves, shots_total, shots_on_target, shots_off_target

players: id, name, fullname, nationality, is_injured, first_name, last_name, birth_date, birth_place, height, weight, position, avg_points, public_id

referee: id, name, fullname, nationality_region_id

region: id, name, code, is_country, iso_code, public_id

round: id, name, start_date, end_date, stage_id

season: id, competition_id, name, is_current, start_date, end_date, public_id

season_stats: season_id, goals, avg_goals_per_match, yellow_cards, avg_yellowcards_per_match, red_cards, avg_redcards_per_match, number_of_clubs, number_of_matches, number_of_matches_played, avg_corners_per_match, goals_conceded_home, goals_conceded_away, goals_scored_home, goals_scored_away, win_percentage_home, win_percentage_away, draw_percentage, avg_homegoals_per_match, avg_awaygoals_per_match

season team: id, season_id, team_id

season team coach: season_id, team_id, coach_id

season team player: season_team_id, player_id, number, is_captain, current, position

stage: id, name, type, season_id, is_current, start_date, end_date

stage standings: id, stage_id

team: id, name, shortname, national_team, region_id, venue, venue_id, public_id

team coach: coach_id, team_id, is_current

team stats: season_team_id, win_home, win_away, draw_home, draw_away, lost_home, lost_away, goals_for_home, goals_for_away, goals_against_home, goals_against_away, clean_sheet_home, clean_sheet_away, attacks, dangerous_attacks, avg_ball_possession, fouls, offsides, red_cards, yellow_cards, shots_blocked, shots_off_target, shots_on_target, failed_to_score_home, failed_to_score_away, avg_goals_per_game_scored_home, avg_goals_per_game_scored_away, avg_corners, total_corners, avg_player_rating, avg_player_rating_per_match, avg_ball_possession_percentage, avg_fouls_per_game, avg_shots_off_target_per_game, avg_shots_on_target_per_game, avg_goals_per_game_conceded_home, avg_goals_per_game_conceded_away, scoring_minute_0_15, scoring_minute_15_30, scoring_minute_30_45, scoring_minute_45_60, scoring_minute_60_75, scoring_minute_75_90, goals_conceded_minute_0_15, goals_conceded_minute_15_30, goals_conceded_minute_30_45, goals_conceded_minute_45_60, goals_conceded_minute_60_75, goals_conceded_minute_75_90, avg_first_goal_scored_home_minutes, avg_first_goal_scored_away_minutes, avg_first_goal_conceded_home_minutes, avg_first_goal_conceded_away_minutes

top players: season_team_id, player_id, position, type, value

venue: id, name, city, capacity, surface, address

Relacionamentos:

COACH_MATCH: Direção: Coach -> Coach_match

COACH_MMATCH: Direção: Coach_match -> Match

COACH_SEASON: Direção: Season_Team_Coach-> Season

COMPETES_IN: Direção: Competition -> Continent

COMPETITION_REGION: Direção: Competition ->Region

Desenvolvimento de um protótipo de um Futebol GPT

EVENT: Direção: Match_Event->Matches
EVENT_TEAM: Direção: Match_Event -> Season_Team
EXISTS: Direção: Competition -> Season
GROUP_STAGE: Direção: Group -> Stage
GROUP_TEAM: Direção: Group_Team -> Group
HAS_PLAYER: Direção: Season_Team_Player <- Season_Team
IS_IN: Direção: Team -> Region
LIVE: Direção: Players -> Region
LIVEC: Direção: Coach -> Region
LOCATED_IN: Direção: Team -> Region
MATCH_INFO: Direção: Match_Info->Matches
MATCH_INFO_COACH: Direção: Match_Info -> Coach
MATCH_INFO_PLAYER: Direção: Player_Matches -> Players
MATCH_INFO_TEAM: Direção: Match_Info -> Season_Team
OCCURS_IN: Direção: Stage -> Season
OTHER_PLAYER: Direção: Match_Event -> Players
PARTICIPATED_IN: Direção: Season_Team_Player <- Players
PARTICIPATES_SEASON: Direção: Team <- Season_Team
PLAYER: Direção: Match_Event->Players
PLAYER_MATCHES: Direção: Player_Matches ->Matches
PLAYER_MATCHES_SEASON_TEAM: Direção: Player_Matches->Season_Team
PLAYER_MATCHES_TEAM: Direção: Player_Matches->Team
PLAYS: Direção: Season_Team -> Matches (para equipas que jogam em casa)
PLAYS2: Direção: Season_Team -> Matches (para equipas que jogam fora)
REFEREE: Direção: Matches -> Referee
SEASON_COACH_TEAM: Direção: Season_Team_Coach->Team
SEASON_TEAM_COACH: Direção: Season_Team_Coach-> Coach
STANDINGS: Direção: Group_Team_Stage_Standings -> Stage_Standings
STANDINGS_GROUP: Direção: Group_Team_Stage_Standings -> Group
Season_Team: Direção: Season_Team -> Season
Season_Team_Coach_Match: Direção: Coach_Match -> Season_Team

TEAM_VENUE: Direção: Team -> Venue

TEAM_COACH: Direção: Coach -> Team_Coach

TEAM_COACHH: Direção: Team -> Team_Coach

PLAYER_ST_STATS: Direção: Players->Player_Season_Team_Stats

SEASON_STATS: Direção: Season->Season_Stats

SEASON_TEAM_STATS: Direção: Season_Team-> Player_Season_Team_Stats

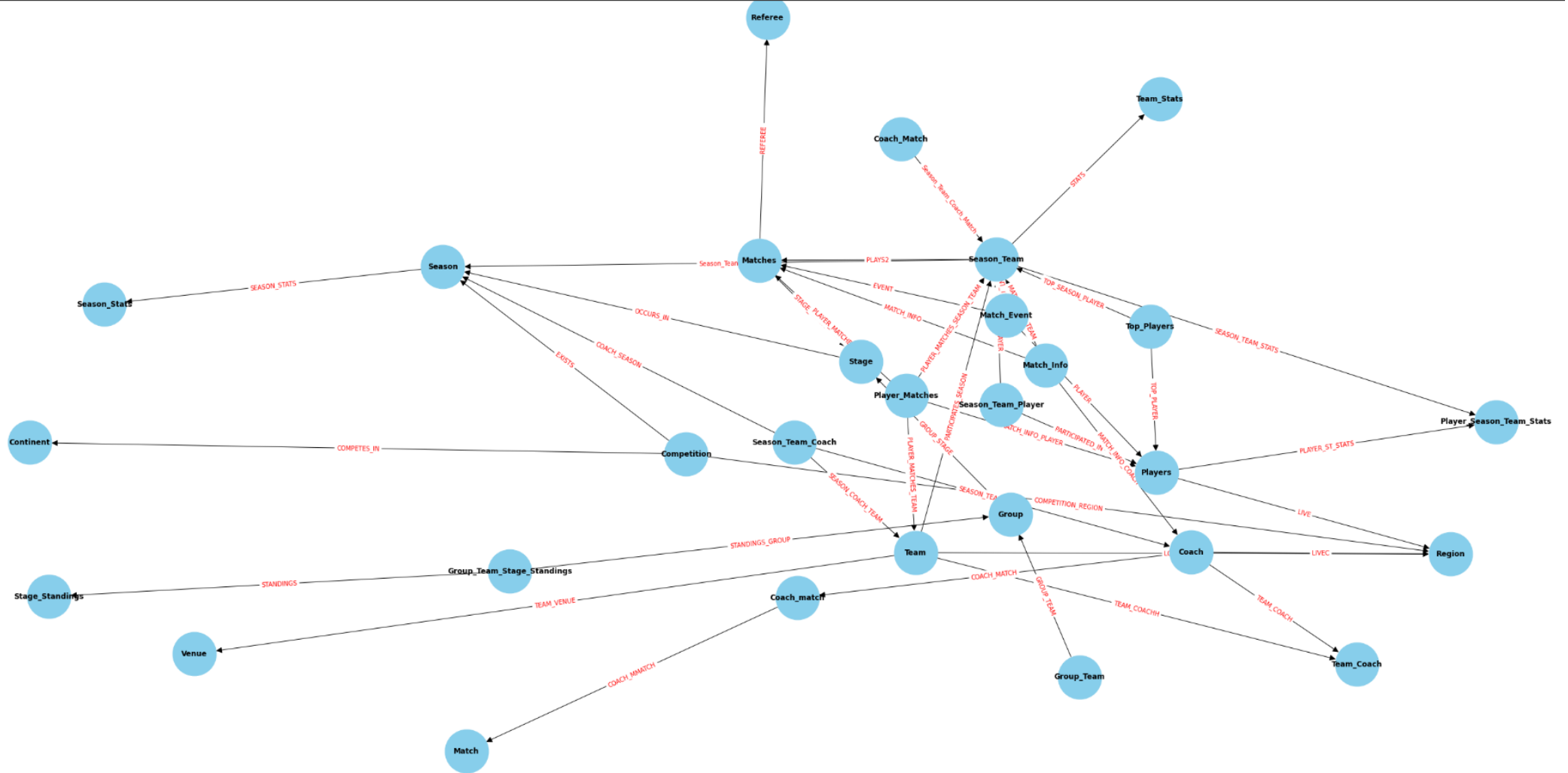
STAGE_MATCH: Direção: Matches->Stage

STATS: Direção: Season_Team->Team_Stats

TOP_PLAYER: Direção: Top_Players->Players

TOP_SEASON_PLAYER: Direção: Top_Players->Season_Team

Desenvolvimento de um protótipo de um Futebol GPT



D. Mockups

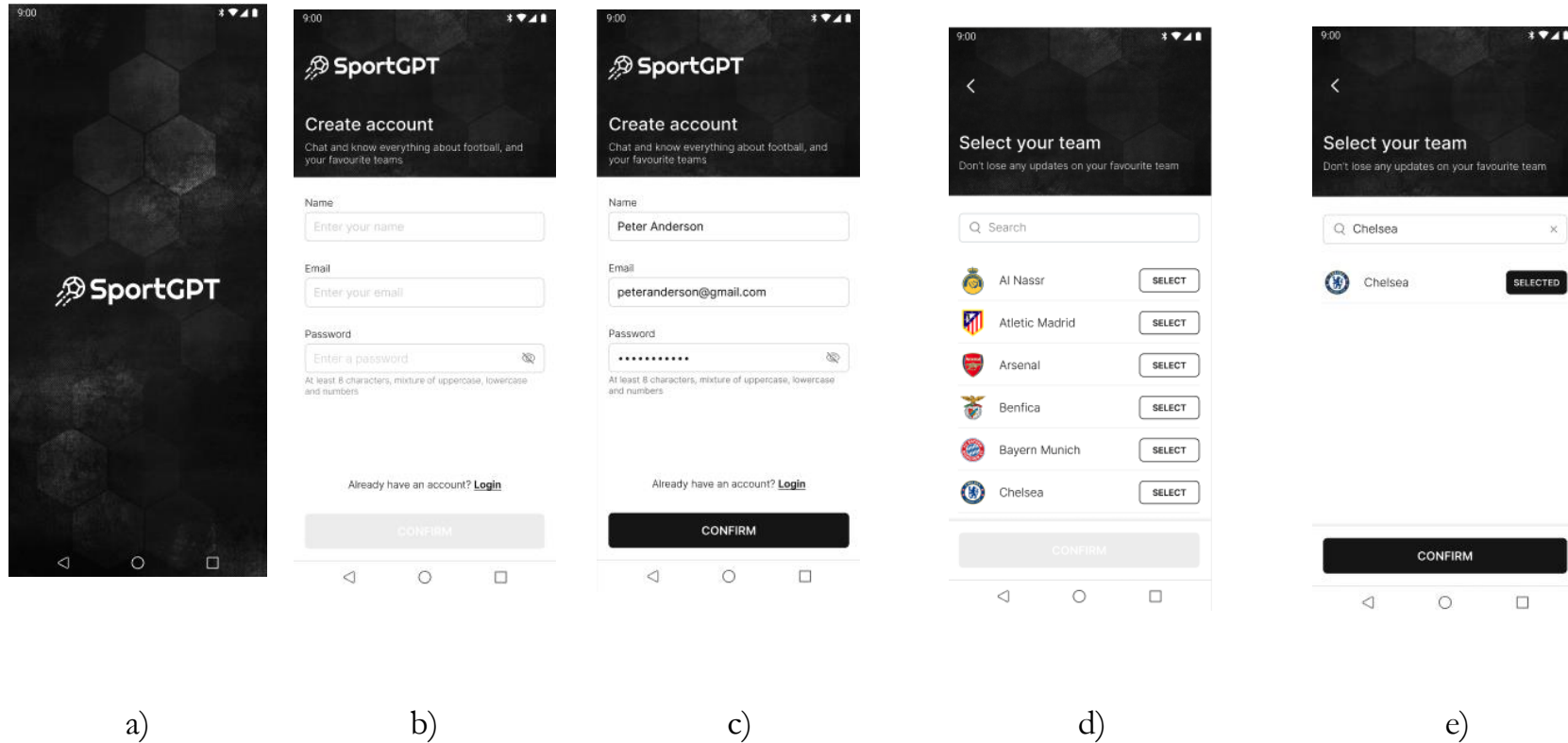


Figura 10-1 - *Mockups* da *App* SportGPT: a) Início da aplicação; b) Ecrã de Registo; c) Ecrã de Registo com dados de exemplo d) Ecrã de Escolha da Equipa favorita do Utilizador e) Ecrã indicativo da Equipa Escolhida pelo Utilizador, com recurso à barra de pesquisa

Desenvolvimento de um protótipo de um Futebol GPT

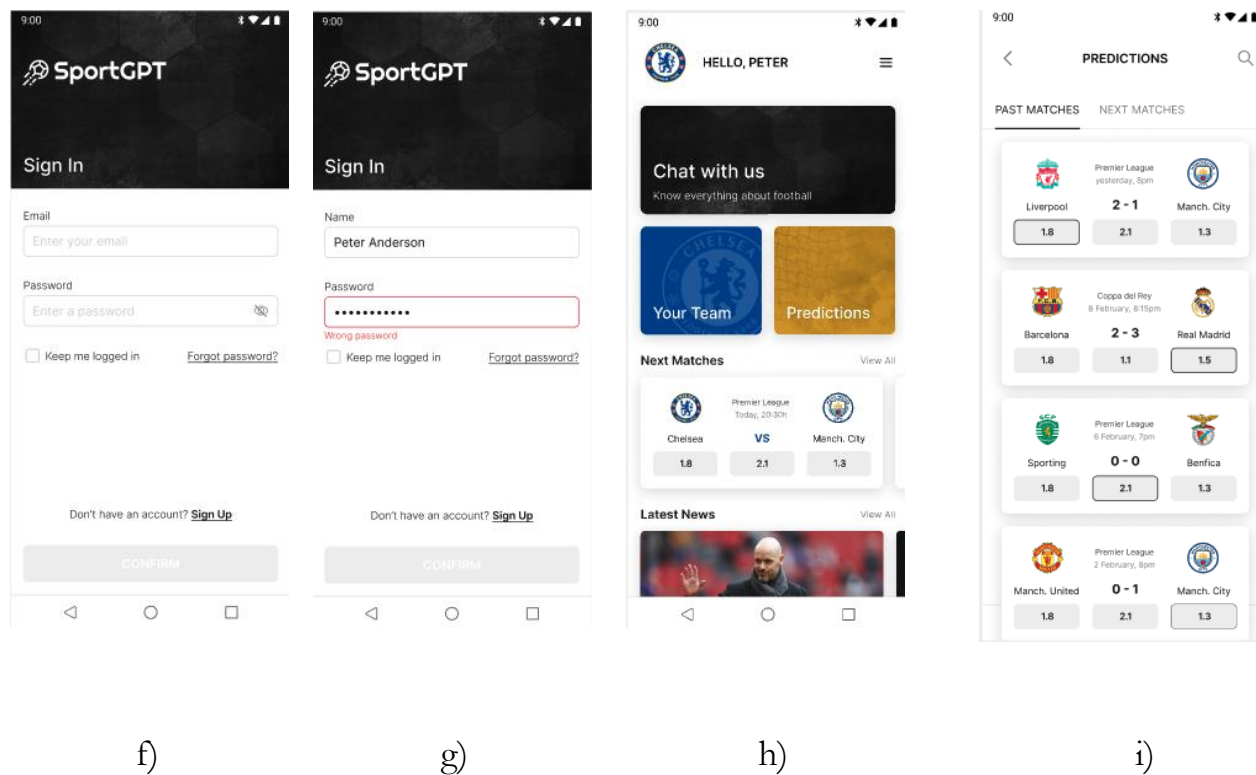
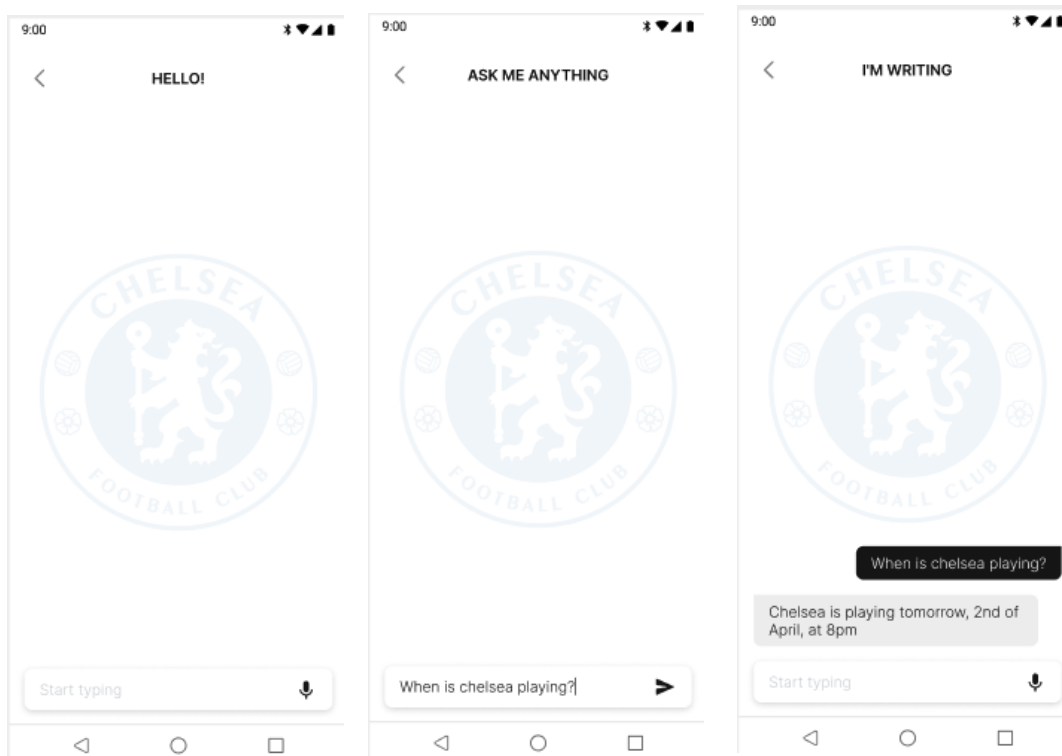


Figura 10-2 - *Mockups* da App SportGPT: f) Ecrã de Login; g) Ecrã de Login cujos dados inseridos demonstram ser incorretos; h) Ecrã Principal da Aplicação após Registo/ Login bem sucedidos; i) Ecrã dedicado às predictions de resultados futebolísticos



j)

k)

h)

Figura 10-3 - *Mockups* da *App* SportGPT: j) Ecrã Inicial do Chat; k) Ecrã do *Chat* após se iniciar a escrita de uma questão futebolística h) Ecrã de *Chat* ao longo da conversação

E. Descrição dos Mockups da Aplicação Mobile do Projeto SportGPT

Ecrã de *loading* da aplicação – ecrã bastante simples com apenas a apresentação do nome da aplicação;

Ecrã de Registo – ecrã enriquecido com campos de preenchimento para dados de utilizador: nome, *email* e *password*. Para a senha da aplicação é requerida uma *password* com no mínimo 8 caracteres: maiúsculas, minúsculas e números. Caso o utilizador já tenha uma conta, conseguirá neste ecrã navegar para o ecrã de *login*. Após a introdução de credenciais, o sistema confirma se o *email* ainda não está em uso e se a *password* está de acordo com as restrições estabelecidas. Caso alguma destas condições não seja cumprida deverá ser prontamente evidenciado ao utilizador que os dados não estão de acordo com os requisitos. De forma a proporcionar uma experiência interativa e clara ao utilizador, os campos serão destacados a vermelho e aparecerá uma mensagem indicadora de erro de autenticação, para que o utilizador saiba que introduziu dados incorretos e possa proceder à correção rapidamente. Sendo possível nesta página navegar para o ecrã de *login* caso já exista uma conta por parte do utilizador. Num cenário de introdução de credenciais de acordo com as exigências, existem 2 cenários possíveis:

- Redirecionado para a página principal;
- O utilizador deverá escolher de entre uma lista, a sua equipa favorita.

A última possibilidade é opcional, conforme descrito na lista de funcionalidades, seria apenas implementado caso se concretizasse a personalização da aplicação *mobile* de acordo com a equipa predileta do utilizador.

Ecrã de Login – a página de *login* seria detentora de apenas preenchimento dos campos de *email* e *password*. Nesta página existia ainda a possibilidade de uma ação de *Keep Me LoggedIn* e ainda *Forgot Password*, que não foram implementados dada a não prioridade destes aspetos para o estágio. Caso o utilizador não possua conta é ainda possível que seja redirecionado para o ecrã de registo.

Página Principal – a página principal da aplicação, conforme projetada nos *mockups*, contém 3 temas principais: *chat*, previsões e notícias futebolísticas. Neste ecrã é possível navegar para o ecrã de *chat* de questões futebolísticas. Para além disso é possível observar outros 2 botões / “*cards*”, cuja funcionalidade não foi implementada, mas pela nomenclatura é possível inferir que um deles seria para redirecionamento para a página de *predictions* de resultados futebolísticos e outro para uma possível página para escolher a equipa favorita. Neste ecrã é ainda possível observar algumas previsões de resultados futebolísticos (embora se encontrem no formato incorreto) e um conjunto de notícias futebolísticas destinadas a informar o utilizador de eventos recentes.

Página de Chat – o ecrã com a funcionalidade de conversação futebolística foi construído de forma simples para oferecer uma experiência direta e intuitiva ao utilizador. Encontramos apenas um campo de entrada para a questão relacionada com futebol e um botão para submissão desta. Para aqueles que preferem realizar as questões por voz existe um botão destinado à gravação do áudio e enviar para o *backend*. Em segundo plano num tom mais transparente existe o logo da equipa preferida, algo a implementar caso fosse tido em conta a personalização da *app* de acordo com informações de preferência dos utilizadores.

Na perspetiva de *user experience*, os *mockups* deixaram a desejar relativamente ao *feedback* fornecido ao utilizador, pois enquanto este aguarda pela resposta apenas é alterado um texto no topo da aplicação: “Hello!”, “Ask Me Anything” e “I’m Writing” como apresentado nos anexos em D. A existência de um *loader* neste caso foi preferível, pois torna-se mais informativo ao cliente de que a resposta está a ser processada. Neste ecrã, à medida que existe a conversação, as mensagens são exibidas de forma organizada e intuitiva, com uma clara divisão: à esquerda encontram-se as mensagens do utilizador, à direita as mensagens do servidor. Para além disso, as cores entre as mensagens de cada entidade também são diferentes de forma a tornar a conversação mais fácil de acompanhar visualmente.

Página de Utilizador – dado que esta página não foi desenhada nos *mockups*, teve de ser construída com base nas outras já existentes. Este ecrã continha apenas um botão para alteração da *password* e sair da conta, sendo redirecionado para a página de *login*.

Página de Predictions – esta página contém informações relativamente à previsão de resultados futebolísticos. A ideia seria apenas prever se uma equipa iria perder, ganhar ou empatar.

F. Autenticação dos utilizadores na aplicação mobile do projeto SportGPT

Para implementação da funcionalidade de autenticação na aplicação foi necessário solicitar dados ao utilizador: nome, *email* e *password* aquando do registo. Estes dados foram armazenados numa base de dados relacional: PostGreSQL, no formato semelhante ao apresentado na Tabela 10-1.

Tabela 10-1 - Armazenamento dos Dados do Utilizador

<i>id</i>	<i>Name</i>	<i>Email</i>	<i>Password</i>
1	Andreia	andreia@gmail.com	TesteWTT.24

Para garantir a integridade e unicidade dos dados aquando do registo do utilizador é realizada uma verificação relativamente ao campo de *email*. Caso o *email* já esteja armazenado na tabela, o sistema impede a criação de um novo registo, solicitando a introdução de um novo *email*. Para esta confirmação foi criado um *endpoint* no servidor que recebe o *email* inserido pelo utilizador e verifica a sua existência na base de dados, e retorna uma mensagem de erro caso exista. Se o *email* introduzido for único, os dados são armazenados na base de dados. No momento do *login*, através da criação de outro *endpoint*, foi possível verificar se as credenciais introduzidas pelo utilizador eram coincidentes através da verificação da correspondência entre o *email* e a *password* com os dados armazenados na base de dados aquando do registo. Caso o *login* seja incorreto é enviada uma mensagem de erro de forma que o utilizador corrija as credenciais.

Em ambos os casos: *login* e registo, se ocorrer sucesso, é chamado um outro *endpoint* que através do *email* permite chegar ao nome com que o utilizador se cadastrou de forma a apresentá-lo na página principal.

G. Descrição dos Casos de Uso do SportGPT

Caso de Uso: Fazer Questão

Nome: Fazer Questão

Atores: Utilizador

Objetivo: O utilizador faz uma questão futebolística, em linguagem natural e aguarda por uma resposta correta e direta.

Pré-Condições: O utilizador tem de estar autenticado na aplicação.

Pós-Condições: A questão feita é enviada para o LLM

Fluxo de eventos:

Cenário principal:

1. O utilizador faz *login* na aplicação e entra no ecrã de *chat*.
2. O utilizador insere a sua questão futebolística.
3. A questão é enviada para o LLM.
4. O utilizador aguarda pela chegada da resposta.

Casos de uso relacionados: Caso de uso Entender Questão

Caso de Uso: Entender Questão

Nome: Entender a questão

Atores: LLM

Objetivo: O LLM “entende” a questão introduzida pelo utilizador e de acordo com o esquema da base de dados gera uma *cypher query*.

Pré-Condições: O utilizador tem de introduzir uma questão futebolística

Pós-Condições: O LLM gera uma curiosidade para posterior execução na base de dados.

Fluxo de eventos:

Cenário principal:

1. O LLM recebe a questão colocada pelo utilizador.
2. O LLM retira da questão introduzida o contexto, nomeadamente as entidades envolvidas e o que é pretendido com a questão.
3. De acordo com o esquema da base de dados e descrições gera uma *cypher query*, ou com utilização de *OpenAI Function Calling*.

Cenário Alternativo:

2. a) a temática e o contexto da questão introduzida não são “entendidos” pelo LLM

1. O LLM entende um contexto diferente:

1.1 O LLM, apesar de entender mal o contexto da questão, verifica que este se adequa aos nós e relacionamentos da base de dados, passando para o passo 3.

1.2 Se o contexto entendido não for de acordo com os nós e relacionamentos da base de dados, o LLM retorna que não sabe responder e o caso de uso termina.

Casos de uso relacionados: Caso de uso Executa Query.

Caso de Uso: Executa *Query*

Nome: Executa *Query*

Atores: Base de Dados

Objetivo: A *query* devolvida pelo LLM é executada na base de dados Neo4J

Pré-Condições: O LLM devolver uma *query*

Pós-Condições: Resultado é devolvido pela base de dados de acordo com a *query* executada.

Fluxo de eventos:

Cenário principal:

1. A *query* gerada pelo LLM será enviada para a base de dados
2. A *query* é executada na base de dados
3. O resultado é retornado ao servidor

Cenário Alternativo:

2. a) a *query* gerada pelo LLM não retorna resultados

1. A base de dados executa a *query*

1.1 Se a *query* não está bem construída, apresentando erros na sua estrutura, é retornado erro. O fluxo passa para o passo 3, onde é retornado *null* para o servidor.

- 1.2 Se a *query*, apesar de errada de acordo com o contexto, é executada na base de dados e produz resultados, passando para o passo 3.

Casos de uso relacionados: Entender Questão e Humanizar Resposta

Caso de Uso: Humanizar Resposta

Nome: Humanizar Resposta

Atores: LLM

Objetivo: Os resultados retornados pela base de dados vão ser alvos de uma *LLM Call* para tornar a resposta mais apelativa para o utilizador.

Pré-Condições: A base de dados ter retornado resultados

Pós-Condições: É gerada uma resposta humanizada.

Fluxo de eventos:

Cenário principal:

1. O LLM recebe a pergunta feita e os resultados retornados da base de dados.
2. De acordo com as informações recebidas gera uma resposta coerente e humanizada a ser apresentada ao utilizador.
3. O resultado é retornado ao servidor.

Casos de uso relacionados: Geração da *Query*

H. Descrição Detalhada dos Casos de Uso do projeto Football Trivia

Caso de Uso: Solicitar Curiosidades

Nome: Solicitar Curiosidades

Atores: Utilizador

Objetivo: O utilizador pretende obter um conjunto de curiosidades acerca de diferentes temáticas: jogador, equipa, campeonato ou temporada futebolística.

Pré-Condições: O utilizador deve preencher todos os campos presentes na página *web*

Pós-Condições: A solicitação de curiosidades é feita.

Fluxo de eventos:

Cenário principal:

1. O utilizador acede à página *web*;
2. Preenche os campos apresentados;
3. Clica no botão de submeter / solicitar as curiosidades;
4. O utilizador aguarda pela chegada das curiosidades.

Cenário Alternativo:

2. a) O utilizador não preenche todos os campos apresentados

1. O botão de solicitar as curiosidades encontra-se desabilitado

1.1 Caso queira solicitar curiosidades o utilizador deve preencher os campos todos: passo 2.

Casos de uso relacionados: Executar *Query*

Caso de Uso: Executar Query

Nome: Executa *Query*

Atores: Base de Dados Neo4J

Objetivo: Executar um conjunto de *queries* na base de dados, de acordo com a temática escolhida pelo utilizador, com fim a obter resultados para a construção das curiosidades.

Pré-Condições: O utilizador solicitar curiosidades acerca de uma temática.

Pós-Condições: São obtidos os resultados para a criação das curiosidades.

Fluxo de eventos:

Cenário principal:

1. Após a solicitação das curiosidades são executadas as *queries* correspondentes à temática escolhida pelo utilizador;
2. O Neo4J retorna os resultados das *queries*.

Cenário Alternativo:

1. a) A execução da *query* falha
 - 1 A base de dados executa a *query* e ocorre uma falha
 - 1.1 O *backend* gera uma mensagem de erro e o caso de uso termina.

Casos de uso relacionados: Solicitar Curiosidade e Gerar Curiosidade

Caso de Uso: Gerar Curiosidade

Nome: Gerar Curiosidade

Atores: LLM (GPT-3.5 Turbo)

Objetivo: O LLM com base na descrição da *query* e dos resultados obtidos gera curiosidades apelativas.

Pré-Condições: A *query* ter sido executada para existirem resultados

Pós-Condições: As curiosidades geradas serão enviadas para que o utilizador possa visualizá-las.

Fluxo de eventos:

Cenário principal:

1. O LLM recebe a descrição da *query* e os resultados retornados pela base de dados;
2. O LLM gera a curiosidade com base nas informações que recebeu e devolve para o *backend*.

Casos de uso relacionados: Executar *Query*, Gerar Imagem e Visualizar Curiosidades

Caso de Uso: Gerar Imagem

Nome: Gerar Imagem

Atores: LLM (DALL-E 3)

Objetivo: O LLM DALL-E tendo como base a curiosidade gerada pelo LLM GPT 3.5-Turbo irá gerar uma imagem sugestiva

Pré-Condições: A curiosidade ser gerada anteriormente pelo GPT-3.5 Turbo

Pós-Condições: A curiosidade será aprimorada e completada com uma imagem

Fluxo de eventos:

Cenário principal:

1. O LLM (GPT-3.5 Turbo) gera a curiosidade futebolística e esta é enviada para o modelo DALL-E 3;
2. O modelo irá gerar uma imagem adequada de acordo com a curiosidade

Casos de uso relacionados: Gerar Curiosidades e Visualizar Curiosidades

Caso de Uso: Visualizar Curiosidades

Nome: Visualizar Curiosidades

Atores: Utilizador

Objetivo: O utilizador visualiza as curiosidades acerca da temática que solicitou.

Pré-Condições: O LLM ter gerado as curiosidades.

Pós-Condições: O utilizador recebe as curiosidades que solicitou.

Fluxo de eventos:

Cenário principal:

1. A *web page* recebe as curiosidades e apresenta-as ao utilizador
2. O utilizador visualiza as curiosidades.

Casos de uso relacionados: Gerar Curiosidades, Gerar Imagens

I. Descrição Detalhada dos Casos de Uso do projeto Football Quiz

Caso de Uso: Escolher Equipas

Nome: Escolher Equipas

Atores: Utilizador

Objetivo: O utilizador escolher 2 equipas distintas (e respetivas regiões) sobre as quais serão geradas perguntas acerca dos jogadores atuais. Neste caso de uso o utilizador ainda pode escolher o número de questões a que pretende responder no quiz.

Pós-Condições: O sistema gerará perguntas relacionadas com os jogadores atuais das equipas selecionadas. O número de perguntas apresentadas será correspondente ao número escolhido pelo utilizador.

Fluxo de eventos:

Cenário principal:

1. O utilizador acede à página *web*;
2. O utilizador preenche os campos relativos aos nomes e regiões das 2 equipas;
3. O utilizador preenche o campo relativo ao número de perguntas do quiz;
4. Perante a situação de todos os campos preenchidos, o utilizador está pronto para solicitar o quiz.

Cenário Alternativo:

2. a) O utilizador não preenche todos os campos apresentados:

1. O botão de solicitar o quiz encontra-se desabilitado

1.1 Caso queira solicitar a geração do quiz o utilizador deve preencher os campos todos: passo 2 e 3.

2. b) O utilizador digita o mesmo nome e região para as 2 equipas:

1. O botão de solicitar o quiz encontra-se desabilitado

1.1 Caso queira solicitar o quiz, o utilizador deverá introduzir nomes diferentes de equipas, ou nomes iguais, mas equipas de regiões diferentes: passos 2 e 3.

Casos de uso relacionados: Responder ao Quiz e Gerar Questões e Respetivas *Queries*.

Caso de Uso: Gerar Questões e Respetivas *Queries*

Nome: Gerar Questões e Respetivas *Queries*

Atores: LLM (GPT-4)

Objetivo: O LLM com base nos ids dos jogadores, esquema da base de dados relativa aos jogadores, descrições e exemplos de *cypher queries* irá gerar um conjunto de questões futebolísticas e as respetivas *queries*.

Pré-Condições: As equipas e o número de questões devem ter sido preenchidos pelo utilizador. E este tem de iniciar o jogo.

Pós-Condições: As perguntas e *queries* necessárias para a construção do quiz são geradas.

Fluxo de eventos:

Cenário principal:

1. O LLM recebe a solicitação para geração de um conjunto de questões e respetivas *queries* acerca dos jogadores;
2. O LLM gera as questões futebolísticas assim bem como as *queries* de acordo com o esquema da base de dados que lhe é passada.
3. O LLM retorna as informações geradas no formato que lhe foi especificado.

Casos de uso relacionados: Executar *Queries*, Escolher Equipas

Caso de Uso: Executar *Queries*

Nome: Gerar Imagem

Atores: Neo4J

Objetivo: A Base de Dados irá executar as *queries* geradas pelo LLM

Pré-Condições: As *cypher queries* deverão ter sido geradas

Pós-Condições: Os resultados das *queries* são obtidos para a construção da resposta das perguntas do quiz

Fluxo de eventos:

Cenário principal:

1. A *cypher query* será enviada para o Neo4J executar
2. A base de dados executa a *query* e retorna o resultado

Casos de uso relacionados: Gerar Estrutura do Quiz, Gerar Questões e Respetivas *Queries*

Caso de Uso: Gerar Estrutura do Quiz

Nome: Gerar Estrutura do Quiz

Atores: LLM (GPT-3.5 Turbo)

Objetivo: O LLM de acordo com a questão anteriormente gerada e o resultado da base de dados o LLM irá gerar a estrutura do quiz, nomeadamente adicionando as várias opções de resposta e a justificação

Pré-Condições: As *queries* têm de ter sido executadas e os respetivos resultados obtidos.

Pós-Condições: A estrutura do quiz será gerada

Fluxo de eventos:

Cenário principal:

1. O LLM recebe a questão anteriormente gerada, o resultado vindo da base de dados e o nome do jogador em causa.
2. O LLM gera a estrutura do quiz: questão, opções, opção correta e justificação.

Casos de uso relacionados: Executar *Queries*, Responder ao Quiz

Caso de Uso: Responder ao Quiz

Nome: Responder ao Quiz

Atores: Utilizador

Objetivo: O utilizador responde às perguntas que lhe forem apresentadas no quiz.

Pré-Condições: O quiz deve estar construído.

Pós-Condições: O utilizador joga o quiz.

Fluxo de eventos:

Cenário principal:

1. O utilizador visualiza a questão que lhe é apresentada
2. O utilizador responde à questão e obtém *feedback* relativamente a se a sua resposta se encontra correta ou não, assim bem como a respetiva justificação.
3. Os passos 1 e 2 são repetidos até que todas as perguntas sejam respondidas.
4. O sistema apresenta a pontuação que o utilizador obteve.

Casos de uso relacionados: Gerar Estrutura do Quiz

J. Flow Relativo à apresentação das imagens dos jogadores

Antes do envio das curiosidades acerca do jogador para o *frontend*, será feita uma *cypher query* que, mediante o nome do jogador, retorna a sua fotografia (vinda do *provider*), representado na Figura 10-4.

```
@app.route('/api/curiosity/player', methods=['POST'])
def curiosity_player():
    data = request.json
    player_name = data.get("player_name").title()
    season = data.get("season")
    language = data.get("language")
    queries = queries_trivia_players(player_name, season)
    trivia = handle_curiosity_queries(queries, language)
    query=f"""MATCH(player:Players)
    |where toUpper(toString(player.name))=toUpper(toString('{player_name}')) | return player.photo"""
    results = graphdb.query(query)
    a=results[0][0]

    return jsonify({"humanized_response": trivia, "image":a})
```

Figura 10-4 - *Endpoint* responsável pelas curiosidades dos Jogadores

Assim juntamente com a curiosidade será enviada a fotografia do jogador para o *frontend*, tal como apresentado na Figura 10-5.

Lionel Messi has been on fire recently, scoring goals in the last five games he played. The most recent goal was scored on 2024-05-05 at the 50th minute mark. Before that, he found the back of the net on 2024-04-28 at the 67th and 32nd minute. On 2024-04-21, Messi scored in the 11th minute, and on 2024-04-14, he scored in the 51st minute. Messi's goal-scoring form has been consistent and impressive, making him a key player to watch in upcoming matches.

ID: 2



Figura 10-5 - Output das curiosidades do jogador juntamente com a imagem

K. Flow relativo à apresentação das imagens das competições e das equipas

As imagens /logótipos relativos às equipas e competições foram fornecidas no *dump* de dados e adicionadas na pasta do projeto na pasta */static*. Os nomes das imagens eram bastante sugestivos, sendo compostos pelo *id* da equipa/ competição facilitando o seu acesso posterior. Antes do envio da lista de curiosidades para o *frontend* é realizada uma *cypher query* que com base no nome da equipa retorna o *id* desta. Através deste, tal como apresentado na Figura 10-6, é recuperada a imagem da pasta *static/images* e enviada para *frontend*.

```
@app.route('/api/curiosity/team', methods=['POST'])
def curiosity_team():
    data = request.json
    team = data.get("team").title()
    region = data.get("region")
    season = data.get("season")
    language = data.get("language")
    queries = queries_trivia_team(team,season,region)
    trivia = handle_curiosity_queries(queries, language)
    query=f"""MATCH(team:Team)
|where toUpper(toString(team.name))=toUpper(toString('{team}')) return team.id"""
    results = graphdb.query(query)
    a=results[0][0]
    link_image=f"static/images/teams/{a}.png"
    return jsonify({"humanized_response": trivia, "logo":link_image})
```

Figura 10-6 - *Endpoint* responsável pelas curiosidades das Equipas

Assim no *frontend* o aspeto será semelhante à Figura 10-7.

Na temporada 2023/2024, a equipa do Chelsea teve como treinadores B. Saltor Grau e M. Pochettino Trossero. Já na temporada seguinte, em 2024, os treinadores foram M. Pochettino Trossero e B. Saltor Grau. Parece que houve uma mudança de treinadores entre as duas temporadas, o que levanta a questão: qual foi o impacto dessa troca no desempenho da equipa ao longo dos campeonatos? Será que os novos treinadores conseguiram melhorar os resultados da equipa, ou será que o sucesso anterior foi mantido? A história por trás das escolhas dos treinadores e o seu impacto no desempenho desportivo do Chelsea está longe de ser simples. Cada nova temporada traz consigo novos desafios e expectativas, tornando o mundo do futebol sempre imprevisível e cheio de surpresas.

ID: 2



Figura 10-7 - *Output* das curiosidades das equipas com logótipo desta

Relativamente à apresentação dos logótipos / imagens das *competitions*, como referido anteriormente, foi necessário alterar o *frontend* para que o *id* da *competition* fosse enviado para o *backend*.

Desta forma torna-se relativamente fácil aceder ao logótipo da competição como é possível verificar na Figura 10-8.

```
@app.route('/api/curiosity/competition', methods=['POST'])
def curiosity_competition():
    data = request.json
    competition_id = data.get("competition").title()
    partes = competition_id.split("|")
    id=int(partes[0])
    competition_name=partes[1]
    language = data.get("language")
    queries = queries_trivia_competition(competition_id)

    trivia = handle_curiosity_queries(queries, language)
    link_image=f"static/images/competitions/{id}.png"
    image="static/images/img-ARYG1FHykPhsmgW0snLP3IIa.png"

    if id<149:
        return jsonify({"humanized_response": trivia, "image_curiosity":image})
    else:
        return jsonify({"humanized_response": trivia, "logo":link_image, "image_curiosity":image})
```

Figura 10-8 - *Endpoint* responsável pelas curiosidades das Competições

No entanto apenas tínhamos logótipos a partir do *id* 149, sendo necessária uma condição para que fosse retornado o logótipo apenas para ids maiores que 149. Nesta página é também enviada uma imagem adicional, apenas para fases de demonstrações, retratando como seriam enviadas as imagens geradas por Inteligência Artificial como apresentado na Figura 10-9.

The 1st game of the Bundesliga in the 2022/2023 season saw FC Union Berlin facing off against Hertha BSC, with Hertha BSC emerging victorious with a score of 3-1.

ID: 3 



Figura 10-9 - *Output* das curiosidades das Competições com imagem

Foi nesta temática que foi implementada a capacidade de geração de imagens com GenAI como foi possível analisar na Figura 10-9.

L. Análise das Informações apresentadas no projeto FootballQuiz

A análise realizada encontra-se num documento excel armazenado na drive, possível de aceder através do link:
<https://docs.google.com/spreadsheets/d/1vhR0O2FdtSSta2WGEZbhT9SD6QxZeaQC/edit?usp=sharing&ouid=107166569985352928400&rtpof=true&sd=true>

M. Análise das Contas da Rede Social X

A análise realizada encontra-se num documento excel armazenado na drive, possível de aceder através do link:

<https://docs.google.com/spreadsheets/d/1qzm0txJ4apBGKXgFLHj3AJSfKZYCGWT6/edit?usp=sharing&ouid=107166569985352928400&rtpof=true&sd=true>

N. Curiosidades e respetivas queries para o Projeto Football Trivia

As curiosidades e as respetivas *queries*, construídas de forma manual, são possíveis de serem observadas através do link:

<https://docs.google.com/spreadsheets/d/1x4G6W40FsWtmsypbKggGQex6xToZV2X1/edit?usp=sharing&ouid=107166569985352928400&rtpof=true&sd=true>

O. Análise da Evolução do SportGPT

No link:

https://docs.google.com/spreadsheets/d/1IHEsAU50iq3EhBZu_VWBMAjIQC02cXJ/edit?usp=sharing&ouid=107166569985352928400&rtpof=true&sd=true, poderá ser consultada a evolução do projeto SportGPT, ao longo da mudança de modelo GPT utilizado. Ainda é possível observar uma comparação entre o nosso projeto e os modelos: SportGPT e GPT4o.

P. Previsão de Resultados Futebolísticos Machine Learning

No link: <https://docs.google.com/spreadsheets/d/1oXx9ObH1o-c1BZfOYm12g9zzt6vkzCNQ/edit?usp=sharing&ouid=107166569985352928400&rtpof=true&sd=true> encontram-se disponíveis os resultados obtidos através da utilização de *Machine Learning* para a previsão de resultados futebolísticos.

Q. Fontes de Dados Analisadas

No link: <https://drive.google.com/file/d/1d78W9JHLG7Q1NM-igtFgorv20y1vtonP/view?usp=sharing> é possível analisar uma análise prévia das fontes de dados.

R. Registo de Interações

No link: <https://drive.google.com/file/d/1cMhIqasCSdAszmDhV29Es0RNEX9i59ew/view?usp=sharing> é possível observar o histórico desde o momento em que o utilizador realiza uma pergunta até à obtenção da resposta.

Andreia Filipa Almeida Graça