Yasmin Ali El Hage

PLATAFORMA DINÂMICA PARA ARMAZENAMENTO E GESTÃO DE DADOS DE SAÚDE

Plataforma dinâmica para armazenamento de dados médicos.

Dissertação/Trabalho de Projeto/Relatório de Estágio submetida como requisito parcial para obtenção do grau de **Mestre em Engenharia de Software**

Júri

Presidente (Prof. Dr. Claúdio Sapateiro, ESTSetúbal, Instituto Politécnico de Setúbal)

Arguente (Prof. Dr. Osvaldo Rocha Pacheco, DETI, Universidade de Aveiro)

Orientador (Prof. Dr. Miguel Angel Guevara Lopez, ESTSetúbal, Instituto Politécnico de Setúbal)

Data (14 de dezembro de 2023)

POLITECNICO SETUBAL

Dedico minha dissertação de mestrado a minha família. Em especial aos meus pais Ali Hussein El Hage, (em memória), e Valeria Ap. Frisselli El Hage pela dedicação constante para com a minha formação educacional e moral. Se não fosse por eles, eu não estaria aqui.

…

Agradecimentos

Agradeço ao professor orientador Miguel Angel Guevara Lopez, para com minha formação intelectual, pois os professores fazem parte de grande formação educacional, são de grande importância para o desenvolvimento da sociedade. Agradeço ao Hospital da Luz de Setúbal, em especial ao Dr. Nuno Cristino, que proporcionou com grande dedicação a concretização desse projeto.

Resumo

Embora a gestão e preparação de dados constitua uma grande proporção do esforço total envolvido na análise baseada nos registos de saúde eletrónicos, Electronic health record, (EHR), as diretrizes atuais para a adquisição e processamento externo de EHR (fora dos próprios sistemas de informação dos Hospitais e Centros de Saúde) são ainda insuficientes até à data de hoje, e é uma realidade que os médicos e profissionais da saúde são confrontados com o problema (desafio) de poder de extrair, processar e analisar quantidades suficientes de EHR para a realização de investigação clínica conclusiva (com representatividade estatística). Nesse sentido, neste projeto de dissertação de mestrado se propõe desenvolver uma "Plataforma Dinâmica para Armazenamento e Gestão de Dados de Saúde" com foco nos pacientes. Esta plataforma tem como objetivo três temas centrais: (1) simplificar o processo de captura e anonimização dos dados dos pacientes; (2) gestão centralizada dos dados, evitando a formação de silos e lacunas de dados; e o mais importante (3) automatizar o processo de criação de datasets de benchmarking para agilizar, facilitar e melhorar o processamento e a análise de grandes volumes de dados (EHR) nos processos de investigação clínica. Se pretende ainda que esta plataforma seja agnóstica ao tipo de dados, i.e., garantindo que toda a informação está centralizada no registo do paciente e que vincule / enlace todos os tipos de dados (exames) possíveis desde simples dados de texto e valores numéricos até outras estruturas de dados mais complexas.

Palavras-chave: dados de saúde, bases de dados, gerenciamento dinâmico de dados, plataforma dinâmica, EHR

Abstract

Although data management and preparation constitutes a large proportion of the total effort involved in electronic health record (EHR) based analysis, current guidelines for external EHR acquisition and processing (outside of Hospitals and Health Centers' own information systems Health) are still insufficient nowadays, and it is a reality that doctors and health professionals are faced with the problem (challenge) of being able to extract, process and analyze sufficient amounts of EHR to carry out conclusive clinical research (with statistical representativeness). In this sense, this master's dissertation project proposes to develop a “Dynamic Platform for Storage and Management of Health Data” with a focus on patients. This platform aims at three central themes: (1) simplify the process of capturing and anonymizing patient data; (2) centralized data management, avoiding the formation of silos and data gaps; and most importantly (3) automate the process of creating benchmarking datasets to streamline, facilitate and improve processing and analysis of large volumes of data (EHR) in clinical research processes. If you also want this platform to be agnostic to the type of data, i.e., ensuring that all information is centralized in the patient's record and that it links / links all types of data (exams) possible, from simple text data and numerical values to other more complex data structures.

Keywords: health data, databases, dynamic data management, dynamic platform, EHR

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

IA	Inteligência Artificial
EHR	Electronic health record
JSON	JavaScript Object Notation
RGPD	Regulamento Geral sobre a proteção de dados
AJAX	Acrônimo de Asynchronous JavaScript and XML
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
ORM	Mapeamento objeto relacional
CSS	Cascading Style Sheets
CID	Classificação Internacional de doenças
CSV	comma-separated-values
CHITS	Community Health Information and Tracking System
FLOSS	Free/Libre Open-Source Software
BERT	Bidirectional Encoder Representations from Transformers
UMC	Medical Center Utrecht
EMIF	European Medical Information Framework
REGEX	Regular Expression
DQ	Data Quality
PNL	Programação Neurolinguística
REST	Representational State Transfer
DICOM	Digital Imaging and Communications in Medicine

Lista de Símbolos

Capítulo 1

1.1. Antecedentes e Motivação

Na dissertação proposta houve a contribuição do Hospital da Luz de Setúbal, Portugal, onde foram realizadas uma serie de reuniões com a Direção Técnica e investigadores médicos, que ajudaram decisivamente no processo de análise / levantamento de requisitos necessários para a elaboração da solução / plataforma dinâmica que aqui é apresentada.

Nas reuniões técnicas realizadas se verificou a existência de formulários em papel de preenchimento manual em diversas áreas. Os formulários são inquéritos / questões colocadas aos pacientes e são preenchidos com a ajuda de pessoal técnico especializado, (enfermeiros ou outros funcionários), do hospital. Portanto, no momento de iniciar nosso trabalho, o hospital seguia um processo manual complexo em que os especialistas médicos e outros investigadores, para fins de estudos científicos, individualmente criavam ficheiros Excel, (isolados), a partir dos formulários específicos de diversas áreas da saúde respondidos / preenchidos pelos pacientes, criando-se assim lacunas de dados difíceis de interligar, com o problema entre outros de ter informação duplicada dos casos de pacientes. Após análise do problema proposto, identificamos a necessidade de criação de uma plataforma para a inserção dos formulários de forma dinâmica, que possibilite aos médicos gerirem os dados técnicos e os datasets / ficheiros.

A solução técnica proposta foi a geração de formulários dinâmicos para a automatização do processo que atualmente é realizado manualmente. A plataforma possibilitará o cadastro de formulários de diversos assuntos da saúde, assim será possível abandonar o registo em papel, e garantir todo o processo de curadoria dos dados. O objetivo é desenvolver uma plataforma dinâmica e simples, com características de usabilidade e possibilidade de pré processamento, curadoria e extração dos dados (datasets de benchmarking). Nesse sentido, a plataforma será também flexível permitindo a separação por grupos, (perfis), de utilizadores com permissões, para que apenas administradores pudessem cadastrar os campos técnicos e não ocorrer duplicidade e informações incorretas.

A construção da plataforma foi realizada gradualmente, (com recurso ao uso de um protótipo de software evolutivo), e com entregas faseadas das funcionalidades, para que fossem validadas do ponto de vista informático e médico. Realizaram-se reuniões semanais com as presenças do orientador da dissertação e do médico responsável pelo projeto, assim como eventualmente participaram também outros especialistas médicos, convidados para avaliar as diferentes fases de desenvolvimento. Forneceram-nos exemplos reais de formulários e exemplos de ficheiros em Excel, fornecidos pelos médicos, como uma forma de modelo para os testes da plataforma.

Exemplo de formulário ver anexo 1.

1.2. Objetivos

Como principal objetivo, esta dissertação pretende desenvolver uma plataforma dinâmica, em base relacional, que através de um modelo cliente – servidor permita: (1) cadastrar formulários feitos a medida, (custom-made), de questões e repostas com informações de saúde / dados de casos de pacientes; e (2) possibilitar a seleção, integração e extração de conjuntos de dados (datasets de benchmarking), para posteriormente facilitar a realização de tarefas de analise de dados. Esta plataforma servirá como ferramenta para pré processar e curar os dados dos casos de pacientes e garantir a qualidade (veracidade) da informação registada nos mesmos. A plataforma será dinâmica, ou seja, permitirá que formulários de diversas especialidades e campos técnicos sejam cadastrados e também permitirá que os formulários sejam respondidos, assim minimizando a dificuldade de se padronizar a informação que pode ser de tipos de dados variados / diferentes, por exemplo, numéricos ou alfa numéricos.

A unificação / integração das informações (dados) dos casos de pacientes, também foi um dos motivos / requisitos no desenho e desenvolvimento desta plataforma, que visa unificar / associar a informação recolhida para cada caso de paciente particular e ainda ordenada por data de realização dos exames médicos realizados, ou seja, todas as informações preenchidas / recolhidas dos pacientes nos formulários. Assim, será possível, por exemplo, analisar o histórico ao nível do paciente, num período / intervalo de tempo determinado.

Desde o ponto de vista prático, esta dissertação pretende proporcionar uma plataforma dinâmica, onde os investigadores e/ou especialistas médicos, poderão cadastrar formulários com questões e repostas, que posteriormente podem ser extraídas e utilizadas em tarefas de analise de dados. Os dados de pacientes serão cadastrados anonimizados e serão únicos, ou seja, não havendo duplicidade. O código interno do paciente no sistema do Hospital será utilizado como chave e será anonimizado na plataforma.
Os médicos terão a opção de extrair as informações / dados selecionados dos pacientes em ficheiros de formato CSV e/ou Excel. Como manifestado, a implementação será faseada, primeiro em um cenário experimental, e logo validada com dados num cenário real utilizando vários formulários de um conjunto representativo de pacientes, com recurso a um estudo real, onde houve preenchimento dos formulários de um conjunto suficientemente grande de casos de pacientes.

1.3. Contribuições da Tese

Desde o ponto de vista técnico (prático) a principal contribuição desta dissertação é a criação de uma plataforma, que permite automatizar os processos de: adquisição e curadoria de dados de saúde, assim como a criação dinâmica de *datasets* de benchmarking, para uma posterior fase de analise de dados, com recurso a técnicas emergentes de *machine learning*.
Desde o ponto da inovação / científico foi possível: (1) identificar / estudar as principais vantagens e desvantagens dos sistemas atualmente desenvolvidos, para o registo de dados / informação

eletrónica de saúde pública; (2) desenhar / desenvolver novos modelos de dados relacionais, para o registo dinâmico de casos de pacientes; e (3) facilitar o processo de análise de dados, automatizando o processo de criação datasets de benchmarking com dados/informação de saúde de casos de pacientes.

1.4. Estrutura da Tese

Capítulo 1
Introdução das motivações e as necessidades de realizar-se a dissertação e o projeto. Houve uma vasta pesquisa sobre dados de saúde e analise de dados de saúde, EHR, com enfase em limpeza de dados, *data curation* e padronização de informação.

Capítulo 2
Análise do estado da arte, trabalhos realizados na área, com informação sobre os desenvolvimentos prévios que sustentam esta tese, incluindo algumas soluções tecnológicas e de mercado. Houve uma vasta pesquisa sobre o que já foi desenvolvido em relação a extração e limpeza de dados de saúde, com recurso a linguagens de programação e ferramentas (e.g., Python, Django e PostgreSQL), como suporte de soluções para sistemas EHR.

Capítulo 3
Explica-se o que foi desenvolvido tecnicamente. Houve uma preocupação em realizar uma contribuição a sociedade para a melhoria dos processos de curadoria de dados e extração de dados de saúde pública, para facilitar estudos posteriores nas áreas da ciência de dados e a inteligência artificial no que respeita à extração de conhecimentos e predição de doenças.

Capítulo 4
Análise e discussão dos resultados, com base em um cenário experimental.

Capítulo 5
Conclusões sobre o desenvolvimento da tese, resultados obtidos e trabalhos futuros.

Bibliografia
Referencias bibliográficas pesquisadas no decorrer dos estudos, construção da dissertação e do projeto.

Capítulo 2
Estado da Arte
Introdução

Os registos eletrônicos de saúde, Electronic health record, (EHR), referem-se à recolha sistematizada de dados de pacientes e armazenamento de informações de saúde em formato digital (i.e., eletronicamente), permitindo assim que sejam compartilhados entre diferentes cuidados, (serviços), de saúde e configurações. Os registos podem ser compartilhados por meio dos sistemas de informação das entidades envolvidas (i.e., Hospitais, Centros de Saúde, Centros de Investigação e Empresas), conectadas à rede, ou ainda outras redes de informação e intercâmbio. Os EHR podem incluir uma série de dados dos pacientes, incluindo dados demográficos, histórico do paciente, tipos de medicamentos utilizados, tipos de patologias (e.g., alergias, imunização), e imagens médicas, (exames imagiológicos), entre outros. No entanto, é um fato que muitos médicos e outros profissionais de saúde, assim como a comunidade científica que investiga na área da saúde têm insatisfação com os atuais sistemas de registo eletrônico de saúde (EHR), graças em grande parte a processos morosos, recursos limitados, mas principalmente devido a problemas de acesso à informação com base nos novos regulamentos da proteção de dados (RGPD), que estão a dificultar e impedem que a comunidade científica, investigadores, médicos e outros profissionais de saúde consigam desenvolver investigação com conjuntos de dados suficientes, e portanto garantir a representatividade estatística das investigações realizadas.

Os dados, (informações), médicas historicamente foram extraídas dos registos do paciente por especialistas clínicos. Essa abordagem tem limitações claras de escalabilidade e tempo, além de custos[1].

Os dados contidos nos EHR, como relatórios clínicos e observações médicas, podem ser usados para registos de doenças, estudos epidemiológicos, vigilância de segurança de medicamentos, ensaios clínicos, auditorias de saúde e muito mais [2]. Entretanto, torna-se inviável a analise manual desses dados devido ao grande volume de dados não estruturados existente e gerados diariamente [3].

Os EHR são um elemento essencial do aprendizado de sistemas de saúde e oferecem um novo ângulo para examinar os processos e resultados da prestação de cuidados de saúde, dado o facto que contêm dados eletrônicos de uma variedade de fontes diferentes incluindo registos e conjuntos de dados administrativos.

Há uma série de razões para a abundância de problemas de qualidade de dados (DQ) encontrados nos EHR. Primeiro a implementação do EHR ainda é imperfeita na maioria dos ambientes de cuidados de saúde; McGinn et al.[4], relatam uma série de barreiras relacionadas à implementação

do EHR, incluindo problemas de software e hardware, conhecimento limitado do utilizador, intercâmbio de pacientes entre vários ambientes de saúde e carga adicional para o pessoal administrativo e os prestadores – todos os quais afetam a qualidade dos dados contidos nos EHR. Ainda nos estudos de McGinn et al. [4], cobrindo um período de 1999 a 2009, foi realizada uma pesquisa bibliográfica em nove bases de dados eletrônicas. Os estudos foram incluídos se relatassem as barreiras e facilitadores percebidos pelos utilizadores para a implementação compartilhada do EHR, em ambientes de saúde comparáveis ao Canadá. Foram incluídos estudos em todos os idiomas com desenho de estudo empírico. A qualidade e a relevância dos estudos foram avaliadas. Foram direcionados quatro grupos de utilizadores de EHR: médicos, outros profissionais de saúde, gestores e pacientes/público. A análise de conteúdo foi realizada de forma independente por dois autores utilizando uma tabela de extração validada com categorização pré-estabelecida de barreiras e facilitadores para cada grupo de utilizadores de EHR. De acordo com o estudo, questões relacionadas aos aspetos técnicos do EHR foram o fator mencionado com mais frequência, citado por 22 dos 52 estudos incluídos (42,3%). Este fator quase sempre é considerado uma barreira à implementação do EHR. A maioria das barreiras frequentemente mencionadas foram as limitações técnicas relacionadas ao software ou hardware e problemas do sistema, (isto é, velocidade -lenta- do sistema, tempo de inatividade não planejado, e assim por diante). Preocupações que o sistema se tornaria obsoleto também foram mencionados.

O uso de dados contidos em EHR tem o potencial de abrir portas para uma nova onda de pesquisas inovadoras. No entanto, sem o pré-processamento adequado de tais grandes conjuntos de dados o resultado da análise pode ser errônea, o que pode afetar a tomada de decisão clínica e/ou a eficácia dos estudos de pesquisa realizados.

O pré-processamento de dados é uma das etapas mais demoradas na modelagem baseada em dados. Para modelagem da trajetória do paciente, são comuns as etapas de pré-processamento que incluem padronização de formato, redução de dados, imputação de dados ausentes e redução da alta granularidade dos códigos utilizados pelos médicos. A padronização do formato é importante quando os EHR são registados em vários arranjos hospitalares [5].

Os EHR permitem a recolha de um grande volume e uma ampla variedade de dados do paciente e do provedor da informação, (sistema de saúde). Considerando os problemas na qualidade de dados para a analise estatística, a triagem e limpeza adequadas em preparação para análise é crucial. Oliwier Dziadkowiec et al. [6], em seus estudos sobre a utilização de Frameworks de DQ, (data qualitiy), para a limpeza e extração de dados de EHR, identificou uma série de Frameworks na literatura que fornecem informações sobre o seguinte: (1) como avaliar cuidados de saúde em DQ, (data quality), bem como (2) “limpar dados” em preparação para análise estatística. Os Frameworks de DQ, (data quality), foram projetados para examinar dados em base de dados relacionais, mas sem considerar o complexo multidimensional de estrutura de dados armazenados por estas bases, além de não considerar formatos de arquivos simples como o Microsoft Excel (.csv).

Neste cenário, a Inteligência Artificial (IA) tem vindo a ser utilizada de forma crescente na área da análise de EHR / dados de saúde. Em particular, abordagens que combinam técnicas de *Machine*

Learning (ML) à prática clinica, com aplicações para processamento de dados pré-clínicos, assistência a diagnósticos, tomada de decisão para tratamento e alerta precoce como parte da prevenção primária e secundária [7]. Com base na abundância de conjuntos de dados (datasets) de saúde, que estão a ser produzidos, disponíveis (públicos e/ou privados), o ML, (Machine Learning), é uma ferramenta de eleição que permite o reconhecimento de padrões e extrair informações / conhecimentos relevantes dos dados [8].

A linguagem de programação *Python* [9] tornou-se a linguagem de programação de eleição para a análise de dados e ciência de dados em todo o mundo.

Com isso, a principal motivação e desafio desse trabalho de dissertação de mestrado, prende-se com a ideia de desenvolver uma plataforma dinâmica para armazenamento e gestão de dados de saúde de pacientes, que facilite aos médicos e outros profissionais de saúde, assim como à comunidade científica associada a criação / extração de "*datasets* de *benchmarking"* a partir / diretamente dos serviços de saúde, e dessa forma, a exploração maciça de espaços de classificadores de *machine (deep) learning,* com o intuito de melhorar a precisão na análise dos dados de saúde.

Segundo X. Shi et al. [10], os dados recolhidos na prática diária dos serviços de saúde podem incluir diferentes tipos de erros, por exemplo, erros de digitação, erros na identificação / seleção da informação dos pacientes etc. A limpeza de dados de saúde no mundo real é um processo demorado, que requer o domínio de técnicas de processamento de dados. Portanto, uma ferramenta para pessoas não técnicas, que pode fazer a limpeza dos dados automaticamente, pode economizar uma quantidade considerável de tempo e orçamento.

Para limpar e/ou pré processar corretamente os dados EHR, e avaliar a qualidade dos dados, alguns estudos deram suas próprias definições de dimensões de Qualidade de Dados (QoD) para dados EHR. Um exemplo, é a revisão sistemática realizada por Weiskopf et al. em 2013 [11], na qual foram identificadas 5 dimensões de qualidade: integridade, correção, concordância, plausibilidade e atualidade.

Realizar analise de dados com um conjunto de informações de diferentes origens e obtidas de forma manual pode causar problemas de limpeza e recolha de dados, alem de limitar a quantidade de informações que muitas vezes é necessária para se ter uma boa analise dos resultados.

Conforme tem sido identificado em estudos recentes, existe a necessidade de criar métodos automatizados de preparação de dados para diferentes tipos de variáveis relacionadas com o domínio / área dos dados de saúde de pacientes. Estes tipos de métodos / sistemas representam um grande passo para a qualidade de analise destes dados. Além disso, as ferramentas atuais automatizadas de limpeza de dados em outras áreas geralmente usam um cálculo estatístico para processar todas as variáveis uniformemente, o que não ocorre nos dados de saúde que possuem variáveis diferentes e com unidades de medida diferentes. No entanto, na prática clínica, há muitas informações, (variáveis específicas), que precisam ser consideradas. Portanto, o desenvolvimento de métodos automatizados com uma boa relevância clínica [10] são bem-vindos.

Na análise do estado da arte realizada, foi possível verificar que um grande número de investigadores tem a perceção da necessidade de que algum tipo de sistema de gerenciamento de dados, deveria

ser disponibilizado de forma mais concentrada por suas instituições patrocinadoras ou pelas agências financiadoras, o qual se manifesta também ao nível do sistema de saúde em Portugal. Assim com estes serviços de gerenciamento de dados, os investigadores, médicos e outros profissionais de saúde poderiam economizar tempo e os custos associados à implementação de sistemas mais sofisticados, que em muitas situações não respondem à suas necessidades específicas. Em entrevistas, muitos investigadores afirmaram que, em troca dos custos indiretos de suas doações, as instituições forneciam muito menos suporte de dados do que desejavam.

Conforme foi identificado nos estudos de S. H. F. Uller et al. [12] os investigadores também expressaram a opinião de que, se as agências ou instituições de financiamento fornecessem mais apoio, eles seriam capazes de economizar fundos de pesquisa extremamente necessários, evitando a duplicação de esforços e gastos com software analítico e especialização em ciência da computação.

Sobre problemas, necessidades e barreiras no âmbito da pesquisa biomédica em gerenciamento e analise, os investigadores identificaram que um componente importante da pesquisa biomédica é a possibilidade de organizar, armazenar e recuperar dados, e estas ações são mais bem realizadas com um bom gerenciamento. Durante as entrevistas, os investigadores mencionaram a frequência dos formatos de dados complexos e variados com os quais lidam ao longo de suas pesquisas. Os investigadores perceberam que os métodos modernos de troca de informações, (como XML, MAGE), não foram especificamente mencionados, mas os mesmos não solicitaram especificamente essas informações durante as entrevistas.

Ainda de acordo com o estudo de S. H. F. Uller et al. [12], foram identificados dois tipos de erros em relação com dados de tipo numérico:

Os erros de arredondamento de valores, como casas decimais. Como exemplo, temos o campo técnico tensão arterial, os valores errados podem ser 9, que deve ser corrigido para 90, ou 16.000, que deve ser corrigido para 160. Este tipo de erro pode ser facilmente convertido de volta à faixa normal por uma simples transformação, na ordem de grandeza. O segundo tipo de erro são os valores irracionais que não podem ser convertidos de volta ao intervalo normal usando uma transformação simples.

Os investigadores também discutiram questões relacionadas ao número crescente de ficheiros, tamanhos crescentes de ficheiros, formatos múltiplos, indexação e anotação de dados. Um problema comum em torno do gerenciamento e análise de dados é que muitos pesquisadores preferem utilizar seus próprios métodos, (individuais), para organizar os dados, o que atenta contra as possibilidades de estabelecer mecanismos de padronização dos formulários de adquisição de dados. As variadas formas de gerenciamento de dados foram aceitas como funcionais para a maioria das necessidades presentes. Alguns pesquisadores admitiram não ter nenhuma metodologia organizacional, enquanto outros usaram qualquer tipo de design / método que melhor atendesse às suas necessidades individuais.

De acordo com M. J. Diño et al, em “Understanding healthcare providers electronic health record (EHR) interface” [13], nos últimos anos, as exigências para a utilização de competências convencionais e os avanços tecnológicos impactaram substancialmente a entrega de serviços de

saúde em muitos países em desenvolvimento. Embora o uso da tecnologia da informação nos setores da saúde continue a crescer / impor-se, países como, por exemplo, as Filipinas adotaram tecnologias baseadas em computadores de baixo custo no gerenciamento de dados de saúde das comunidades. Previstas para automatizar os processos principais em um centro de saúde, para uma prestação mais eficaz de serviços de cuidados de saúde. A sua criação reflete uma ação responsiva para incutir o valor público da recolha de dados de qualidade e para atender às necessidades locais dos centros de saúde. Neste estudo que representa a implementação de um sistema pioneiro de EHR nas Filipinas, foi mencionado que desde a sua criação em 2004, a "Community Health Information and Tracking System" (CHITS), vem se expandindo nas unidades de saúde rurais em todo o país [13].

Nos esforços tendentes a sistematizar / automatizar os processos de recolha de dados / informação de casos de pacientes, muitos centros de saúde comunitários também utilizam sistemas / plataformas de EHR, classificados como software de código aberto gratuito/livre, um exemplo é o FL/OSS (Free/Libre Open Source Software) [14].

Conforme os princípios de Design de Interface Humano-Computador de Shneiderman et al. [15] onde considera-se que o utilizador e os desenvolvedores de interfaces devem reconhecer a diversidade, considerando as diversas necessidades dos utilizadores do sistema, e que o projeto do sistema deve incorporar elementos para reduzir erros, são identificadas oito regras principais de design de interface, a serem consideradas: (1) buscar consistência, (2) ajudar utilizadores regulares, utilizar atalhos, (3) oferecer feedback informativo, (4) diálogos de composição para gerar encerramento, (5) oferecer prevenção de erros e simples erros de manuseio, (6) permitir a reversão (gerenciáveis) de ações, (7), manter o controle interno e, por último, (8) reduzir a carga de memória de curto prazo.

O autor acreditava que um sistema de computador bem projetado pode produzir sentimentos positivos de sucesso e permite que os utilizadores se concentrem em seus trabalhos e exploração.

Os registos médicos são um dos principais mecanismos para fornecer a continuidade do cuidado seguro e eficaz ao paciente. Tradicionalmente, avaliar a qualidade dos registos médicos em papel com foco na "qualidade dos dados" e questões legais de registos médicos torna-se difícil e passível de erros.

A aplicação da ciência de dados no domínio da saúde teve um crescimento exponencial nos últimos 10 anos, com o inicio da utilização do uso dos sistemas EHR, que agrupam dados em rótulos codificados e estruturados para doenças e utilização de cuidados da saúde [16].

No entanto, as preocupações com a qualidade, privacidade de dados, transparência e comparabilidade destes sistemas restringiram o uso de evidências geradas com dados estruturados de saúde. Essas preocupações também restringiram a aceitação de evidências geradas com dados estruturados de cuidados de saúde por reguladores, reembolso de autoridades e grupos de trabalho de orientação. Apesar da disponibilidade de numerosos padrões de relatórios, não há consenso sobre como implementar os princípios Localizáveis, Acessíveis, Interoperáveis e Reutilizáveis (FAIR) [17].

Uma contribuição importante neste sentido são os estudos realizados por Dipak Kotecha et al. [18], em que uma ampla gama de investigadores / stakeholders participou no desenvolvimento do Framework CODE-EHR, incluindo reguladores, (US Food e Administração de Medicamentos e

Medicamentos Europeus Agência), agências governamentais, (Comissão Europeia, o Instituto Nacional de Excelência em Saúde e Cuidados do Reino Unido, e Iniciativa de Medicamentos Inovadores), revistas médicas, (BMJ, “European Heart Journal, The Lancet e The Lancet Digital Health”), grupos de defesa de pacientes, (Rede Europeia do Coração e Fórum de Pacientes ESC), representantes da indústria farmacêutica, representantes da área da saúde, instituições acadêmicas e profissionais das sociedades. Como resultado, foram identificados quatro aspetos chaves: questões técnicas e administração de processos e dados; segurança e privacidade de dados; as necessidades dos reguladores; autoridades de reembolso; e diretrizes de prática clínica.

De acordo com o estudo de Zhuqi Miao et al. [19], foi criado um Framework com o intuito de orientar e validar os EHR para a análise secundária, (alimentar sistemas de análise de dados). A proposta do projeto foi dividida em quatro grandes fases: extração de dados, desenvolvimento de coorte, compilação de variáveis, e análise. Cada fase envolve um conjunto de dados específicos da fase de operações. Na extração de dados, por exemplo, os dados brutos são consultados e extraídos da base de dados com base no diagnóstico, procedimento, resultados laboratoriais e/ou outras métricas clínicas necessárias para análise. O desenvolvimento da coorte está focado em descobrir e refinar a coorte de pacientes do conjunto de dados brutos extraído na fase anterior. Durante a compilação de variáveis, as variáveis necessárias para análise são identificadas e adicionadas como colunas para a tabela de dados. “Join and codification one-hot” [20] são operações comuns na fase de compilação de variáveis. O *join* pode ser usado para adicionar variáveis numéricas, (por exemplo, valores laboratoriais e sinais vitais), como colunas, enquanto a *codification one-hot* representa diagnósticos, procedimentos e medicamentos como colunas binárias para indicar se esses eventos ocorreram.

Conforme mostrado na Figura 1, o Framework proposto por estes investigadores envolve um fluxo de trabalho que consiste em uma série de etapas e critérios para resolver problemas de qualidade de dados, avaliar o resultado e decidir se deve prosseguir para a próxima fase ou retornar para uma fase anterior [19]. Problemas de qualidade de dados associados a uma determinada etapa de pré processamento, podem aparecer numa fase posterior, o fluxo de trabalho permite ao utilizador que possa retornar a uma fase anterior conforme necessário para resolver os problemas.

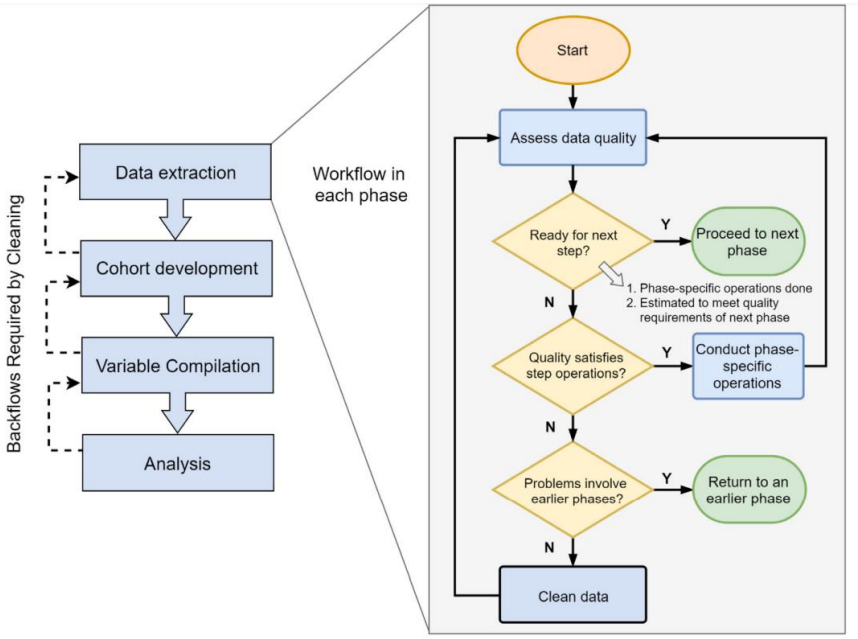


Figura 1 – Fases e workflow propostos para a preparação do Framework para EHR [19].

2.1. Recolha de dados EHR

De acordo com os estudos de D. Morquin et al. [21], implementações com falha de sistemas corporativos são repetidamente documentadas na literatura. Isso geralmente é causado por um desajuste entre a organização e o sistema construído nos conceitos de acessibilidade e atualização de acessibilidade, para propor um método para o diagnóstico e resolução de desajustes entre processos organizacionais e sistemas corporativos. Através de um estudo em um hospital universitário, eles descobriram que a perspetiva de acessibilidade permitiu que os utilizadores entendessem o desajuste, (combinação atual de recursos), e a solução, (uma recombinação apropriada de recursos).

A análise destes autores teve foco nos processos clínicos apoiados pelo sistema de saúde de um hospital universitário localizado em França. Foram recolhidos diversos tipos de dados, incluindo relatórios de reuniões, documentos internos, trocas de e-mails entre o agente de mudança (primeiro autor) e os diversos atores de campo, e os mapas e notas de observação produzidos pelo segundo autor (pesquisador não participante). Além disso, três entrevistas com o proprietário do processo e duas partes interessadas principais forneceram dados para a etapa de avaliação. A conceção dos investigadores acionável do desajuste através da lente das qualidades foi usada como estrutura para análise de dados.

Com a necessidade de pré processar / curar com eficiência os dados EHR, várias propostas para um Framework geral foram feitas. Um estudo em 2016 aplicou o Framework proposto por Kahn et al. [6], no qual, ao longo do estudo, foram definindo indicadores de avaliação de acordo com as 5 dimensões de qualidade, identificadas na tabela 1, (Integralidade, Correção, Concordância, Plausibilidade e

Atualidade) [10].

De acordo com os dados dos resultados obtidos no estudo de Weiskopf et al. [11], conforme a tabela 1, existem 27 termos exclusivos que descrevem dimensões de dados de qualidade. As dimensões são definidas abaixo:

Integralidade: Existe uma verdade sobre um paciente presente no EHR?

Correção: Um elemento que está presente no EHR é verdadeiro?

Concordância: Existe concordância entre os elementos do EHR ou entre o EHR e outra fonte de dados?

Plausibilidade: Um elemento no EHR faz sentido à luz de outro conhecimento sobre o que aquele elemento está medindo?

Atualidade: Um elemento no EHR é uma representação relevante do estado do paciente em um determinado ponto no tempo?

A lista de termos de qualidade de dados e seus mapeamentos para os cinco as dimensões descritas acima são mostradas na tabela 1.

Table 1 Terms used in the literature to describe the five common dimensions of data quality

Completeness	Correctness	Concordance	Plausibility	Currency
Accessibility	Accuracy	Agreement	Accuracy	Recency
Accuracy	Corrections made	Consistency	Believability	Timeliness
Availability	Errors	Reliability	Trustworthiness	
Missingness	Misleading	Variation	Validity	
Omission	Positive predictive value			
Presence	Quality			
Quality	Validity			
Rate of recording				
Sensitivity				
Validity				

Tabela 1- Termos usados na literatura para descrever as cinco dimensões comuns da qualidade dos dados [11].

Integralidade

A Integralidade foi a dimensão mais comumente avaliada da qualidade dos dados e foi área de foco em 61, (64%), dos artigos. De um modo geral, a Integralidade refere-se à existência ou não de uma verdade sobre um paciente que estava presente no EHR.

Correção

A segunda dimensão mais comumente avaliada da qualidade dos dados foi o acerto, que foi incluído em 57, (60%), dos artigos.

Os dados do EHR foram considerados corretos quando as informações que estavam contidas eram verdade. Outros termos comumente usados para descrever este conceito incluíam precisão, erro e qualidade.

Concordância

Dezassete, (17%), dos artigos revisados avaliaram a concordância. Os dados foram considerados concordantes quando houve concordância ou compatibilidade entre elementos de dados. Isto pode significar que dois elementos que registam as mesmas informações para um único paciente têm o mesmo valor, ou que elementos registando diferentes informações têm valores que fazem sentido quando consideradas juntas, (por exemplo, o sexo biológico é registado como feminino e o procedimento é registado como exame ginecológico).

Plausibilidade

Sete, (7%), dos artigos avaliaram a plausibilidade dos dados do EHR. Neste contexto, os dados eram plausíveis se estivessem de acordo com conhecimentos ou informações médicas gerais e foram, portanto, viáveis. Em outras palavras, avaliações de plausibilidade tinham como objetivo determinar se os dados poderiam ou não ser confiáveis ou se eram de qualidade suspeita. Outros termos que foram usados para discutir e descrever a plausibilidade dos dados EHR incluem dados de validade e integridade.

Atualidade

A atualidade dos dados EHR foi avaliada em quatro, (4%), dos 95 artigos. A atualidade era frequentemente referida na literatura como *currency* ou *regency*. Os dados foram considerados atuais se fossem registados no EHR dentro de um período razoável após a medição ou, alternativamente, se fossem representativos do estado do paciente no momento de interesse desejado. Em todos os quatro artigos, a atualidade foi avaliada através da revisão da entrada de dados históricos. Em três dos quatro, os pesquisadores analisaram se desejavam que os dados fossem inseridos no EHR dentro de um limite de tempo definido. E no quarto artigo, os pesquisadores consideraram se cada tipo de elemento de dado foi medido recentemente o suficiente para ser considerado clinicamente relevante.

Muitos artigos avaliaram a integridade dos dados EHR usando outra fonte de dados como padrão ouro. Os padrões ouro usados incluíam registos em papel mantidos simultaneamente, informações fornecidas pelos pacientes, revisão de dados pelos pacientes, encontros clínicos com pacientes, informações apresentadas por pacientes, padrões treinados, informações solicitadas ao médico assistente e fontes de dados alternativas das quais os elementos EHR foram removidos. Uma abordagem semelhante envolveu triangular dados de múltiplas fontes dentro do EHR para criar um padrão ouro [11].

De acordo com o estudo de X. Shi et al. [10] *Intego* é uma base de dados longitudinal de dados EHR recolhidos de pacientes em uma clínica geral em Flandres região da Bélgica. É uma rede de registo computadorizada operacional única. A rede foi fundada em 1994 e 111 clínicos gerais, (GP), participaram em 2015. Os dados são recolhidos anualmente para cada paciente que teve um contato com seu médico naquele ano. No total, cerca de 300.000 pacientes individuais foram registados na base de dados desde o início da rede, correspondendo a uma cobertura anual de cerca de 2% da população flamenga.

Em 2015, foram mais de 3 milhões de medições clínicas, 3 milhões de diagnósticos, 14 milhões de prescrições e 38 milhões de testes de laboratório de cerca de 10.000 variáveis [10]. Como contribuição, estes autores propõem um método que facilita a curadoria dos dados clínicos automaticamente e permite realizar analise dos dados em grande escala. Primeiramente, utilizaram dados de EHR de atenção primária para testar o método. Os resultados demonstraram uma ligeira perda de integralidade, mas a credibilidade, a exatidão e a plausibilidade dos dados melhoraram muito após o pré processamento dos dados.

De acordo com o estudo de X. Shi et al. [10], existem uma série de estudos que comprovam os problemas de qualidade de dados (DQ) encontrados nos EHR. A implementação do EHR ainda contem problemas na maioria das configurações de cuidados de saúde. O design atual de alguns campos EHR, ou seja, os tipos de campos como texto aberto ou resposta livre, acrescentam a probabilidade de erro do codificador; o cotidiano do trabalho na área da saúde que na maioria das vezes é agitado, aumenta ainda mais a vulnerabilidade de erro no EHR. Por fim, às vezes, registos têm datas atrasadas, não estão fechadas, (isto é, os pacientes carecem de datas válidas de alta, procedimento administrativo ou transferência de datas), ou são simplesmente imprecisas. Os dados EHR podem ser estruturados, (dados numéricos que possuem um formato pré-determinado), não estruturados, (campos de texto sem formato pré-determinado), ou semiestruturado, (uma combinação de estruturado e campos numéricos e de texto não estruturados). Cada um desses tipos de dados implica diferenças no seguinte: (1) métodos de extração, (2) padrões de avaliação de qualidade, (3) procedimentos de triagem e (4) limpeza dos procedimentos antes de estar pronto para uso secundário ou análise estatística. Os dados estruturados são os mais fáceis de trabalhar [6].

2.2. Curadoria de dados de EHR

A curadoria dos dados de EHR tem como um dos objetivos a melhora no diagnostico, assim podemos ver no artigo desenvolvido por Lamia Bem et al. [22] que propõe uma nova abordagem para deteção e isolamento on-line de sinais vitais imprecisos. Uma solução que envolve medições em aplicativos móveis de saúde. O principal objetivo deste projeto foi distinguir entre medição imprecisa e degradação da saúde do paciente para reduzir falsos alarmes médicos. A proposta seria de um paciente realizar sua própria medição de sinais vitais (frequência respiratória, pulsação, sangue pressão, SpO2, etc.) usando dispositivos médicos implantáveis e vestíveis com sensores como oxímetro de pulso (conforme mostrado na Figura 2).

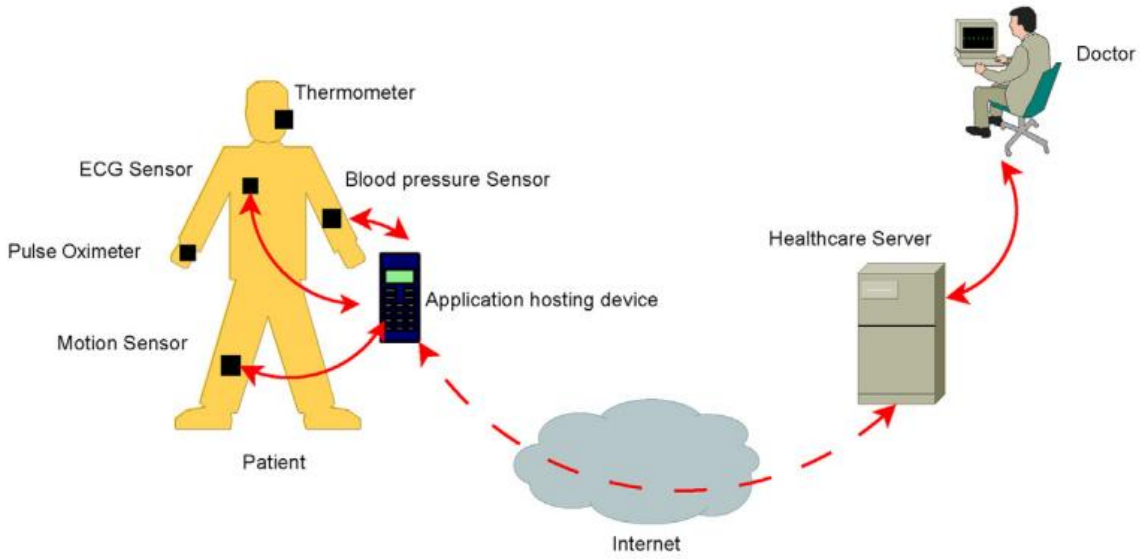


Figura 2. Cenário de saúde mobile [22].

Também podemos encontrar eficácia nos estudos de Antonella Delmestri et al. [23], onde teve o objetivo de criar um "*data accuracy*", contendo co variáveis, por exemplo, dependendo do desenho do estudo observacional, foi verificado que o número de variáveis depende do desenho do estudo e da pesquisa com questões e é definida pelos investigadores. Do ponto de vista do curador, não há limite superior para o número de variáveis. O projeto de nome "Curador", cria tabelas intermediárias contendo apenas os dados relevantes para os pedidos variáveis, que representam uma fração de todo o conjunto de dados.

Uma outra abordagem de curadoria foi utilizada nos estudos de *data mining* de informações de registos eletrônicos de saúde para o status atual de tabagismo por T. Katrien et al. [24], onde foi utilizado um algoritmo de mineração de dados para obter informações capturadas em campos estruturados, incluindo questionários específicos de especialidade e o campo de intoxicações no "University Medical Center Utrecht" (UMC Utrecht), ou texto livre não estruturado, incluindo notas clínicas, correspondência de e para colegas.

2.3. Classificação de EHR

De acordo com os estudos de V. Gupta et al. [25], a classificação de dados estruturados de EHR pode conter dados administrativos e dados auxiliares. Os dados administrativos permanecem inalterados durante todo o período de encontros clínicos, (como dados demográficos), ou continua mudando ao longo do tempo, (como diagnósticos e procedimentos). Auxiliar os dados discretos, (como medições fisiológicas, medicamentos e exames laboratoriais), ou contínuos, (como respiração e tensão arterial).

No âmbito da pesquisa sobre EHR e sua classificação, foi encontrado na literatura académica o artigo O. Trigueros et al. [26] que aborda o problema da classificação multi-rótulo através do

processamento de linguagem natural aplicado a EHR. No método de classificação, os EHR são codificados de acordo com a Classificação Internacional de Doenças, (CID). As abordagens propostas anteriormente na literatura, funcionam como caixas pretas, ou seja, não fornecem muitas informações e não contem dados adicionais. A Inteligência Artificial Explicável, (XAI), ajuda a elucidar o que proporcionou o modelo a fazer as previsões.

Uma área particularmente desafiadora da classificação de textos clínicos relacionado com a codificação CID-10 é a classificação de texto multi-rótulo, pois apresenta o obstáculo de selecionar um subconjunto de rótulos enquanto ainda trabalha com milhares de rótulos possíveis. Trabalhos anteriores mostraram que o problema de classificação do multi-rótulo de EHR codificados com CID-10, pode ser resolvido com uma arquitetura “Bidirectional Encoder Representations from Transformers” (BERT) adaptada [26]. O BERT utiliza procedimentos de pré-treinamento não supervisionados para produzir uma representação significativa da sequência de entrada e fornece resultados de última geração em muitas tarefas importantes de PNL, (Programação Neurolinguística). BERT-XML combina BERT pré-treinamento com atenção multi-rótulo [27].

2.4. Recolha de dados

De acordo com Y. Rohe t al. [28], a recolha de dados é um grande gargalo no aprendizado de *machine learning* e um tópico de pesquisa ativo ao nível internacional. Existem basicamente duas razões pelas quais a recolha de dados se tornou recentemente uma questão crítica. Primeiro, como o aprendizado de *machine learning* está se tornando mais amplamente utilizado, estamos vendo novos aplicativos que não têm necessariamente dados etiquetados/rotulados suficientes. Em segundo lugar, as técnicas de *machine learning* geram recursos automaticamente, o que economiza custos de engenharia de recursos, mas, em troca, podem exigir quantidades maiores de dados rotulados. A nova era de *deep learning, (*um subconjunto dentro do *machine learning*, que é essencialmente o uso de redes neurais que tentam simular artificialmente, o comportamento do cérebro humano), também exige outra necessidade premente de boas ferramentas de gerenciamento de dados e infraestruturas de base de dados que forneçam acesso a *datasets* organizados e facilitam a reutilização para vários aplicativos [12][29].

Conforme K. Maharana et al. [30] identificou, a curadoria de dados é um processo para detetar dados incorretos ou ruidosos e corrigindo-os ou removendo-os do conjunto de dados.

Em geral, atua na identificação e substituição de dados e registos incompletos, imprecisos, irrelevantes ou quaisquer outros dados e registos de ruído. Embora as técnicas utilizadas variem de acordo com a procura do modelo, as etapas básicas seguidas são:

a. Remover dados irrelevantes ou duplicados. Isso geralmente acontece no conjunto de dados. Quando os dados são combinados de uma fonte diferente, extraídos de vários clientes e/ou sistemas diferentes. Há oportunidade para gerar dados duplicados.

b. Correção de erros estruturais, as inconsistências são geradas devido à rotulagem incorreta de

categorias ou classes. Também pode ocorrer devido a convenções de nomenclatura estranhas, erros de digitação ou letras maiúsculas incorretas.

c. Valores ausentes. É comum haver valores ausentes de colunas específicas em um conjunto de dados. O problema pode ser gerado devido a regras de validação de dados ou recolha de dados. Mas é necessário considerar os valores ausentes, porque isso pode impossibilitar a execução / desempenho de um modelo devido aos valores que faltam. Se um número razoável de valores estiver faltando, então métodos simples de interpolação podem resolver esse problema.

O método mais comum usado para lidar com isso é usar valores médios, medianos ou modais relativos aos recursos do modelo.

d. Valores inconsistentes.

Continuando nos estudos de S. H. F. Uller et al. [12], o gerenciamento de dados tornou-se mais complexo devido ao conjunto de dados científicos públicos de alta qualidade. Estes dados são amplamente disponíveis e com acesso mais fácil a tecnologias de alto rendimento em locais com recursos compartilhados. O uso crescente de instalações de serviços centrais dentro das instituições para fornecer conhecimentos especializados, como bioestatística ou ensaios de micro arranjos - bem como tecnologias científicas de produção em massa, como "kits" de laboratório - reduziu muitas barreiras técnicas e permitiu que o investigador possa gerar e recolher dados fora de sua própria disciplina com mais facilidade.

Nos estudos de S. H. F. Uller et al. [12], foi identificado que atualmente, há a necessidade de obtermos métodos aprimorados de gerenciamento de grandes conjuntos de dados. Alguns pesquisadores estão cientes de que seus métodos atuais de gestão, precisam de melhorias. Vários discutiram a necessidade de melhoria de toda a organização de informações de laboratório, passando de uma organização baseada na preferência individual e na necessidade de uma organização de dados estabelecida em todo o laboratório. Um exemplo específico de organização com métodos problemáticos, foi a prática comum de investigadores no mesmo laboratório criarem planilhas personalizadas sem nenhuma estrutura padrão comum. A profusão de planilhas criadas individualmente contendo dados sobrepostos e atualizados de forma inconsistente criou uma grande confusão em alguns laboratórios. Houve pouca consideração para dados futuros requisitos de troca ou apresentação no momento da publicação. Embora os pesquisadores no passado tenham usado planilhas contendo uma apresentação global de dados para sintetizar conceitos e gerar hipóteses, essa abordagem tornou-se menos viável à medida que os dados se tornaram mais complexos.

Segundo K. Maharana et al. [30], um conjunto de dados é uma coleção de objetos de dados referidos como pontos, padrões, eventos, casos, amostras, observações ou entidades. Como resultado, esses itens de dados são frequentemente caracterizados por vários recursos que fornecem as características essenciais de uma entidade, como a massa de um objeto, a hora em que um evento ocorreu e assim por diante. Uma característica pode ser uma qualidade individual medida ou uma parte de um fenômeno que ocorreu, que pode ser amplamente classificado em dois tipos:

1 Categórico: Um recurso categórico é aquele cujos valores são selecionados de uma coleção de

possibilidades predefinidas. Por exemplo, o nome de um mês. Outro exemplo é um conjunto booleano, que contém valores como *True* e *False*.

2 Numérico: Características cujos valores são contínuos ou inteiros. Eles são representados principalmente por números, com os quais compartilham muitas de suas características. Por exemplo, quantos passos você dá todos os dias ou a velocidade com que dirige seu automóvel.

2.5. Problemas com dados

Segundo K. Maharana et al. [30], os dados recolhidos de qualquer fonte persistem dados incompletos, ruidosos e inconsistentes, levando a problemas com a análise de dados. Portanto, é necessário corrigir os problemas de antemão, e eles podem ser classificados em três grupos – muitos dados, poucos dados e dados fraturados. A Figura 3 explica os problemas com dados em formato tabular.

De acordo com D. C. Schoen et al. [31], se os dados disponíveis não incluírem uma quantidade suficiente de dados de todos os tipos, então a confiabilidade do conhecimento adquirido a partir dos dados pode ser incompetente. Atributos ausentes podem prejudicar a precisão do modelo. No caso de indução de árvore de decisão, atributos ausentes podem levar a um comprimento desigual. Ao mesmo tempo, eles estão dividindo o conjunto de dados em conjuntos de treinamento e teste. Isso pode levar a uma distribuição desigual de recursos. Se os dados usados tiverem mais de 20% de dados ausentes, eles devem ser eliminados.

Ainda de acordo com K. Maharana et al. [30], os dados podem estar disponíveis em várias formas: tabelas estruturadas, não estruturadas, tabelas configuradas, imagens, arquivos de áudio, vídeos etc. Uma máquina não pode entender diretamente o texto livre, vídeo ou imagem como ela é; é necessário converter os dados fornecidos em 1s e 0s. Portanto, os dados brutos não podem ser alimentados diretamente no modelo de *machine learning* e esperam que sejam treinados. O pré-processamento de dados é a primeira etapa do *machine learning* em que os dados são transformados/codificados para que possam ser colocados em tal estado que agora a máquina possa passar rapidamente ou analisar esses dados. Em outras palavras, também pode ser interpretado como se o algoritmo do modelo pudesse analisar prontamente os recursos dos dados. O pré-processamento de dados é o mais importante e influente para gerar desempenho de atualização de um Algoritmo de *machine learning* supervisionado. A quantidade de dados de treinamento cresce exponencialmente em relação à dimensão do espaço de entrada. Estima-se que o tempo gasto no pré-processamento pode ocupar de 50% a 80% de todo o processo de classificação, comprovando a importância do pré-processamento na construção de um modelo. Também é essencial melhorar a qualidade dos dados para um melhor desempenho.

A incompatibilidade de dados torna-se um problema significativo se recolhidos de vários grupos ou plataformas diferentes. O objetivo, a profundidade e o padrão para manter e gerenciar os dados podem variar conforme a necessidade. Além disso, o nível de detalhes em que os dados são armazenados na base de dados pode diferir, levando a problemas durante a modelagem.

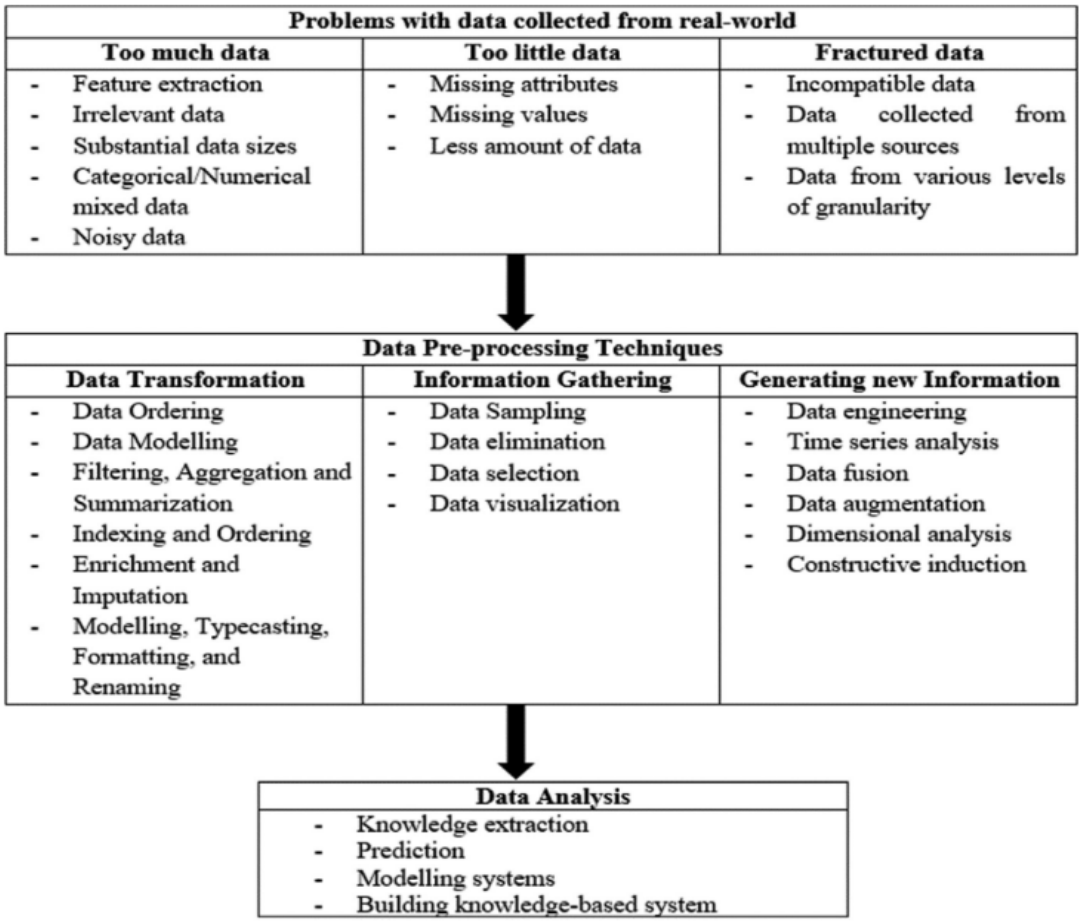


Figura 3. Problemas com dados recolhidos do mundo real [30].

Nos domínios de aplicação da medicina, telecomunicações ou espaço, o volume e a velocidade com que os dados são produzidos é muito grande. O volume desempenha um papel vital no fator limitante na realização de análises com um conjunto de dados em tempo real. Fora isso, dados corrompidos podem enfraquecer a capacidade preditiva do modelo. O pré-processamento dos dados para interpretação adequada é um recurso de formulário que condiciona os dados de entrada para permitir a extração de recursos subsequentes mais fácil e resolução aumentada [30].

2.6. Regulamento Geral sobre a proteção de dados (RGPD)

O Regulamento Geral de Proteção de Dados, (RGPD), é um regulamento europeu, sobre privacidade e proteção de dados, onde os aspetos de segurança da informação são relacionados aos dados pessoais e devem ser aperfeiçoados [32]. Além disso, as transferências de dados pessoais para países terceiros ou organizações internacionais estão sendo consideradas como um dos principais desafios ou temas a serem abordados. Neste contexto, a gestão do consentimento é um aspeto fundamental. O RGPD enfatiza o papel do paciente e o papel do seu consentimento para processamento de seus dados. De acordo com o RGPD, o responsável pelo tratamento dos dados em seus estudos deve conseguir ser capaz de comprovar que os dados do paciente foram utilizados com o seu consentimento.

Nos estudos de M. Javaid et al. [33], a Indústria 4.0 é uma realidade estabelecida que parece atender a vários requisitos no campo da indústria médica, com extensa pesquisa em andamento nesta área. Assim, há necessidade de entender como a Indústria 4.0 pode ser um paradigma útil para atender a vários requisitos da área médica usando diferentes tecnologias.

Conforme foi constatado por X. Larrucea et al. [34], a indústria da saúde 4.0 está cada vez mais sendo um tema importante dentro da Indústria 4.0.

De acordo com o “General data protection regulation” [35], foi demonstrado que as máquinas virtuais dos computadores, podem ser alteradas com código vulnerável e podem dar o controle a potenciais cibercriminosos ou podem causar comportamentos imprevisíveis ou erros fatais. Na verdade, existem diversas vulnerabilidades utilizadas por hackers, como sequestro de serviços, limpeza de dados, manipulação de dados do cliente ou até mesmo criação de máquina virtual maliciosa. Ao mesmo tempo, um sistema federado da indústria de saúde 4.0 envolvendo diferentes países requer a implementação de tecnologias, a superação de problemas de interoperabilidade entre sistemas e uma avaliação de aspetos legais como o Regulamento Geral de Proteção de Dados (RGPD).

As recentes alterações aos regulamentos europeus de proteção de dados pessoais continuam a permitir a utilização de dados de saúde para fins de investigação, mas estabelecem uma avaliação de impacto na proteção de dados como instrumento de reflexão e análise de risco no processo de tratamento de dados. A publicação de um guia facilita a realização desta avaliação de impacto, embora não seja a aplicação direta para projetos de pesquisa. A experiência em um projeto específico é detalhada, e mostra como o contexto do tratamento se torna relevante em relação às características dos dados. A realização de uma avaliação de impacto é uma oportunidade para garantir o cumprimento das princípios de proteção de dados num ambiente cada vez mais complexo e com maiores desafios éticos [36].

2.7. Framework Django

Para o projeto proposto selecionamos o *Framework Django* que utiliza a linguagem *Python* e poderá ter componentes novos em *Pyhton* a serem acoplados a plataforma, como a leitura de imagens de exames médicos utilizando a bibliotecas do Pyhton para tratamento de tipos de dados DICOM, Digital Imaging and Communications in Medicine. Considerando que os estudos de analise de dados referentes são em sua maioria desenvolvidos utilizando a linguagem *Python*, (*machine learning*). Para atender aos requisitos técnicos e funcionais que foram identificados a partir da análise dos desafios atuais na integração de dados EHR, foi verificado na literatura como nos estudos de J. L. Oliveira et al. que produziu a plataforma com o Catálogo EMIF, (European Medical Information Framework) [37] que foi apoiado pelo projeto de L. Bastião Silva et al. MONTRA [38]. Esta estrutura é uma arquitetura ágil para publicação e descoberta de dados, escrita em Python 2.7.6, usando *Django* 1.4.5, uma estrutura que incentiva o desenvolvimento rápido e um design programático limpo. Para complementar isso, uma parte considerável do desenvolvimento foi feita em HTML5, CSS e JavaScript, nomeadamente a interface e a interação com o utilizador final [37]. Depois de ter

identificado a lacuna existente no processo de integração de fontes de dados biomédicos e reunindo requisitos dos stakeholders, foi construída uma arquitetura flexível para centralizar e compartilhar dados biomédicos provenientes de fontes múltiplas, independentes e heterogêneas, bem como uma interface amigável que permite a interação com os dados. As fontes de dados caracterizadas no sistema contêm os dados completos, enquanto o painel disponível através do MONTRA fornece uma visão integrada do esqueleto dessas fontes subjacentes. O MONTRA integra uma API RESTful que fornece um conjunto de *endpoints* programados que podem ser consultados por aplicações de terceiros. A ideia principal por trás da API Web é que outras aplicações possam enviar dados para o MONTRA, no formato de pares chave-valor, contendo metadados extras do registo. É um mecanismo simples para adicionar dinamicamente informações de metadados em cada entrada, (conforme demonstrado na figura 4).

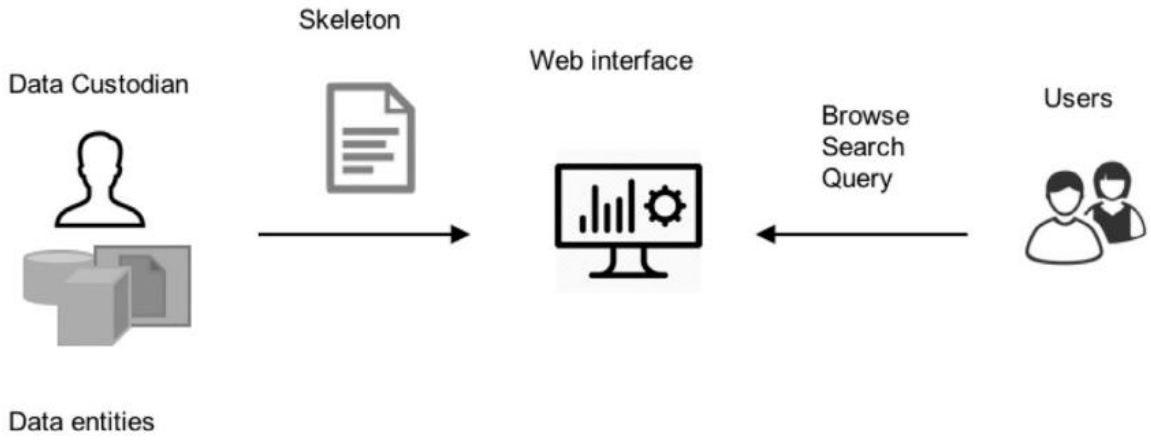


Figura 4. MONTRA general view [37].

Outra referencia de utilização do Framework Django na literatura é o estudo de Zheng et al. [39] onde propuseram o Framework Django para uma estrutura de agendamento de coflow bilateral distribuída. De acordo com os investigadores o número ideal de conexões simultâneas é previsto antecipadamente, o que evita os casos em que o congestionamento acontece resultante de muitas conexões e o desperdício de largura de banda resultante de poucas conexões. Enquanto isso, o *Django* pode combinar as vantagens de abordagens de agendamento orientadas pelo remetente e pelo recetor e alcançar quase o ideal.

Python

Python é uma linguagem que dispõe de mecanismos que facilitam os programadores, por isso, diferentemente das demais linguagens, utiliza poucas palavras, o que implica em menos código. No decorrer da programação em *Python*, nos deparamos com a indentação, que é considerada na interpretação do código de forma lógica, como na declaração de condições, *loop*, (laços), e condições encadeadas, (IF encadeado). Aprendemos que a indentação é primordial e que nesta linguagem temos de ter um código organizado e limpo para funcionar, para assim não obtermos erros de compilação e execução. Esta linguagem faz uso de palavras reservadas curtas da língua inglesa, facilitando assim, o entendimento e o aprendizado, além de deter de um grande poder de

processamento, (flexibilidade e simplicidade), fator que pode ser utilizado tanto por aqueles que estão entrando no mundo da programação, como também para os mais experientes.

Por outro lado, e seguindo a popularidade desta linguagem surgiu o *Django*, um *Framework* para criação de aplicações Web escrito em Python, criado em 2005 por um grupo de programadores do Lawrence *Journal-World* com a intenção de tornar mais rápido o desenvolvimento de aplicações Web. Este *Framework* tornou-se conhecido por fornecer soluções para grande parte dos problemas tradicionais em desenvolvimentos Web, possuindo dezenas de tarefas comuns já prontas para serem reutilizadas, como por exemplo autenticação de usuário, administração de conteúdo, mapas de site, entre outras [40]. Além disso, o *Django* possui integração mais fácil a base de dados relacional ou não relacional e a serialização de modelos de entidades em JSON, facilitando assim, a parte de modelagem de dados do site.

A figura 5 oferece uma visão simplificada dos componentes e camadas de uma aplicação *Django Framework* [41] com um armazenamento de dados utilizando uma base de dados relacional.

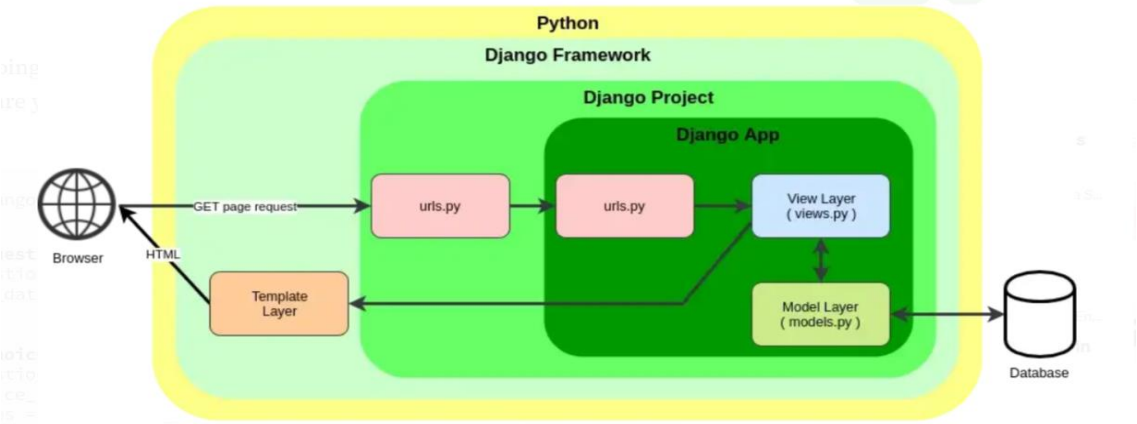


Figura 5. Diagrama do modelo de estrutura do Django Framework. Extraído de (https://medium.com/@cvarelaruiz/why-i-love-python-and-django-26596ce4d82e)

2.7.1. Arquitetura Model Template View

O *Framework Django* utiliza-se do padrão de arquitetura MTV, (Model, Template, View).

Na figura 6 podemos ver como funciona o fluxo entre as camadas da arquitetura *Django*.

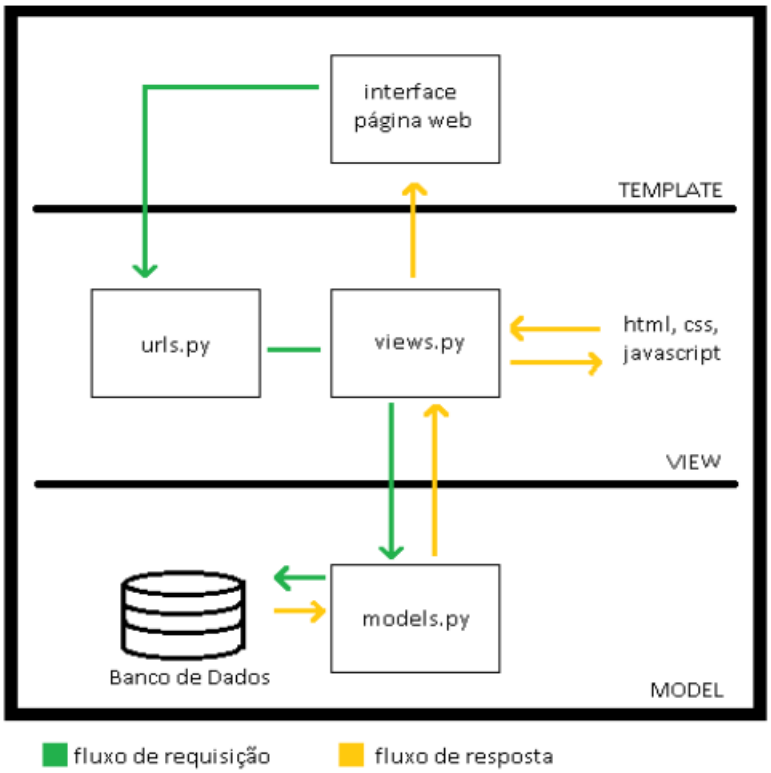


Figura 6. Fluxo MTV Django [14].

- Model: Mapeamento das classes do projeto para a base de dados.
- Template: Páginas para visualização de dados. As páginas possuem código HTML com uma linguagem de tags especificas com a lógica para renderizar os dados dos modelos provenientes das views.
- View: Lógica de negócio. As views contem toda a lógica e regras de negócio. Podem também chamar procedimentos de base de dados, cujo mesmos também possuem regras de negócio. As views recebem a informação e o tipo da requisição (“POST” ou “GET”) do lado do cliente e, em seguida, formatam os dados para que sejam armazenados na base de dados, através dos modelos da camada Model [40].

Toda essa arquitetura é interligada e interagem-se entre si. As camadas dependem umas das outras para realizar determinado serviço e executar a chamada que o utilizador solicitou. Na figura 7 temos uma demonstração de como as camadas interagem-se:

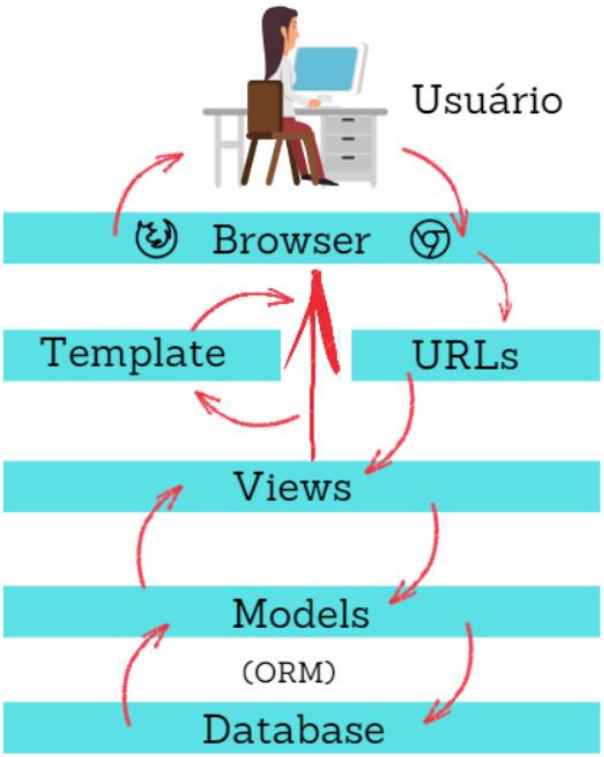


Figura 7. Diagrama do MTV. Extraído de (https://www.treinaweb.com.br/blog/entendendo-o-mtv-do-django)

Model

Um “Model” é como se fosse uma classe onde este mesmo modelo será convertido em uma tabela da base de dados. O Model é um modelo abstrato da entidade de um web site, no caso, temos uma plataforma que contem pacientes, ou seja, a entidade paciente é considerada um model e respetivamente uma tabela de pacientes na base de dados. O model contém os campos e comportamentos essenciais dos dados que devemos armazenar num contexto da entidade. O model é a camada que se relaciona a base de dados.

Esse exemplo de model define um **Patient**, que tem os atributos abaixo:

```
from django.db import models
class Patient(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

First_name e **last_name** são campos do model. Cada campo é especificado como um atributo de classe e cada atributo é mapeado para uma coluna da base de dados.

O model **Patient** acima criaria uma tabela de base de dados como esta:

```
CREATE TABLE myapp_patient (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

View

A view é uma função Python que recebe uma solicitação da Web/utilizador e retorna uma resposta da Web. As views podem conter chamadas a procedimentos de base de dados, tratamento de erros e condições de regras de negócio. A resposta da view pode ser um redirecionamento a um template, a um xml, um erro, ou uma imagem.

As views usam os modelos para aceder os dados da base de dados que podem inserir, atualizar e excluir os mesmos, e de seguida retornam essas informações para os templates.

Templates

Um template contem o código de linguagem estática da saída HTML desejada, bem como alguma linguagem especial que descreve como o conteúdo dinâmico será visualizado.
No template que a renderização dos dados acontece, vindo das views. É como o *Django* visualiza as solicitações para os utilizadores.

Django e a segurança da informação

O *Framework Django* disponibiliza internamente a segurança necessária para as aplicações e facilita a programação aos desenvolvedores, evitando que ocorram erros comuns de segurança no desenvolvimento, como injeção de SQL, script entre sites, falsificação de solicitações entre sites e clickjacking.
O sistema de autenticação do *Django* fornece uma forma segura de gerenciar contas e senhas de utilizadores, bem como grupos de utilizadores.

2.8. Abstração da base de dados

Na programação orientada a objetos, os aplicativos são construídos usando objetos. Esses objetos refletem objetos da vida real e eles contêm dados e comportamento. O modelo relacional que é amplamente utilizado para armazenamento de dados em aplicativos, utiliza tabelas para armazenar dados e linguagens de manipulação de dados para interagir com esses dados. Alguns sistemas de

gerenciamento de base de dados têm recursos orientados a objetos, mas não são totalmente compatíveis. É muito claro que essas duas arquiteturas são amplamente utilizadas e será por muito tempo a partir de agora. Além disso, a programação orientada a objetos e o modelo relacional são utilizados em conjunto, na maioria das vezes para criar aplicativos de todas as escalas. Mas a forma como as duas tecnologias se combinam é longe de ser perfeito. O termo usado para essa incompatibilidade é “incompatibilidade de impedância objeto-relacional” [42][43].

A razão por trás dessa incompatibilidade é que as duas tecnologias dependem de conceitos diferentes. Orientado a Objeto, a programação depende de conceitos de programação comprovados, enquanto os modelos relacionais seguem princípios matemáticos.

Os sistemas de mapeamento objeto-relacional foram desenvolvidos exclusivamente para resolver esse problema.

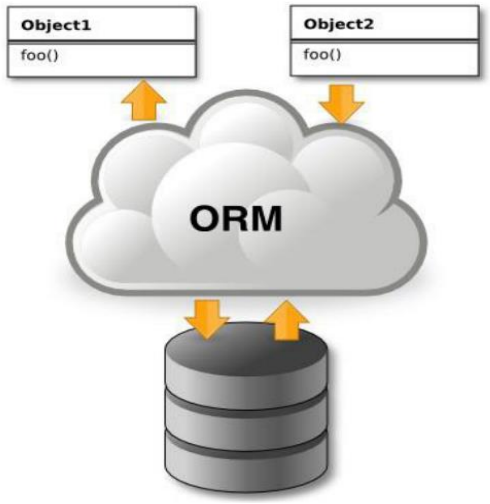


Figura 8. Mapeamento Objeto-Relacional (ORM)[43]

Um sistema de mapeamento objeto-relacional (ORM) é definido como sendo uma ferramenta que fornece uma metodologia e um mecanismo para sistemas orientados a objetos armazenarem dados de forma segura e por um longo período em uma base de dados, tendo controle transacional sobre eles, mas sendo expressos, se necessário, como objetos dentro da aplicação. Um sistema ORM libera o desenvolvedor da preocupação de conhecer a estrutura ou esquema do base de dados. O acesso aos dados é feito usando um modelo conceitual que reflete os objetos de negócios dos desenvolvedores [43].

2.9. Base de dados PostgreSQL

O PostgreSQL é um poderoso sistema de base de dados objeto-relacional de código aberto que usa e utiliza a linguagem SQL combinada com muitos recursos que armazenam e dimensionam com segurança as cargas de trabalho de dados mais complicadas. As origens do PostgreSQL remontam a 1986 como parte do projeto POSTGRES da Universidade da Califórnia em Berkeley e tem mais de 35 anos de desenvolvimento ativo na plataforma principal [44].

Ao longo dos anos esta base de dados conquistou uma forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e dedicação da comunidade de código aberto por trás do software para oferecer consistentemente soluções inovadoras e de alto desempenho. O PostgreSQL é executado em todos os principais sistemas operacionais, é compatível com ACID desde 2001 e possui complementos poderosos, como o popular extensor de base de dados geoespacial PostGIS. O PostgreSQL conquistou a posição de ser considerada a base de dados relacional de software livre preferida por muitos programadores e organizações[44].

O PostgreSQL pode ser integrado ao *Python* usando o módulo psycopg2, sycopg2 é um adaptador de base de dados PostgreSQL para a linguagem de programação Python [45].

2.9.1. A importância da linguagem Structured Query Language (SQL) na análise de dados

Para o projeto da plataforma dinâmica a base de dados foi utilizada fortemente. Com o intuito de agilizar o retorno de dados ao *Framework*, foi utilizado os recursos da base de dados como procedimentos, funções, tipos de dados e o mais importante, a base de dados propicia a utilização da linguagem SQL que auxilia muito nas seleções dos dados de forma a obter-se os dados agrupados em formato JSON ou relacionados, como a busca de colunas em várias tabelas selecionando corretamente a relação entre elas.

A elaboração de relatórios sobre estudos clínicos normalmente requer esforços de um analista de dados para decifrar o esquema da base de dados do estudo. O analista de dados normalmente desenvolve consultas em linguagem de consulta estruturada, (SQL), para encontrar coortes de pacientes que satisfazem critérios clínicos específicos. No entanto muitas vezes é necessário um conhecimento considerável do domínio para implementar o SQL de critérios clínicos e, portanto, o analista de dados é frequentemente emparelhado com um especialista clínico que pode explicar os critérios em termos dos dados disponíveis e seus elementos. Esta colaboração muitas vezes requer semanas ou meses para produzir um resultado com um conjunto de instruções SQL que produzem a análise desejada. No entanto, estas instruções SQL só podem ser modificadas ou solucionadas pelo analista de dados. O especialista clínico não pode modificar as consultas SQL devido ao desconhecimento do SQL, dificultando a validação independente das consultas e da análise resultante [46].

Capítulo 3

Plataforma dinâmica para armazenamento de dados médicos

3.1 Desenho e modelo conceitual da plataforma dinâmica

Se propõe uma plataforma “dinâmica” que utiliza uma abordagem orientada a questões e repostas para registar cada dado e sua relação com experimentos e estudos, desenhado por médicos e/ou investigadores em diferentes áreas de saúde. Os dados genéricos de pacientes serão armazenados utilizando um modelo de dados relacional, (estruturado), com recurso ao *postgreSQL* [45].

Para responder / solucionar o problema da dinamização dos formulários de questões, respostas e campos técnicos, foi desenhada uma plataforma inovativa, suportada nas entidades, (definidas), apresentadas na figura 9, que deverão ser interpretadas / implementadas no *Framework* como modelos, como se descreve a seguir:

A entidade *Answer Field name* guarda todos os tipos de informações técnicas da área médica e pretende-se que sejam dados únicos e controlados por um administrador do sistema.

A entidade *Questions* relaciona-se a entidade *Answer Field name,* que por sua vez contem a lista de possíveis campos específicos que devem ser previamente cadastrados. Este relacionamento é uma das principais contribuições no desenho da plataforma, pois para cada questão, (Question), podemos identificar um campo técnico (*Answer Field name*), relacionado. Os campos técnicos são objetos de analise para *N* estudos envolvendo a área da saúde, e é com base neles que focamos no desenvolvimento da plataforma, que será responsável em manter a integridade e a validação dos mesmos.

A entidade *Formulário* tem como objetivo armazenar os formulários que serão desenhados e registados por cada investigador e/ou especialista médico, (representado na figura 9 como a entidade *Physician/Researcher*). Para cada formulário registado, teremos *N Questions* que poderão ser cadastradas para que os pacientes possam respondê-las.

Os formulários são registados com a informação da especialidade médica e o utilizador que criou o formulário, que no caso será o médico e/ou investigador que está a propor o estudo / projeto de investigação.

Os formulários poderão ser preenchidos repetidamente para um mesmo paciente, mas com datas diferentes, para que os dados apareçam como historial e para comparação e evolução dos estudos para com aquele paciente específico.

Ao determinar o *answer field name* para uma questão, o utilizador que está a cadastrar as questões, deverá indicar também como serão as respostas, (*Answers*).

As possíveis *Answers*, (respostas predefinidas), poderão ser de três tipos diferentes:

- ***Text*** – caracteres alfanuméricos e numéricos.

- ***Choices*** – O utilizador deve cadastrar a lista de opções de resposta que serão visualizadas no momento de responder as questões para o paciente.
- ***Numeric*** – Apenas valores numéricos.

O tipo *numeric* apenas permite números e decimais separados por ponto, (.).

A entidade *Patient* possui todos os registos dos pacientes. Para a anonimização, o hospital selecionou a chave primária de seu sistema interno para a identificação do paciente, o código NHC, assim evitando duplicidades.

Quando o utilizador, (enfermeiro ou técnico de saúde), for preencher o formulário, com as questões e respostas de um paciente, terá de preencher primeiramente o número do NHC, automaticamente ocorrerá uma verificação a base de dados se aquele NHC existe. Caso o NHC não existir, o sistema deverá inserir na base de dados e permitir que o formulário seja preenchido. Caso o NHC existir, automaticamente, a plataforma atribui essa identificação ao formulário como sendo desse paciente encontrado, sempre considerando o "id" (identificador interno) da tabela de pacientes na plataforma dinâmica. A escolha de se utilizar o id interno foi por uma questão de performance, uma vez que estamos a utilizar o índice da chave primária da tabela de pacientes.

Os códigos NHC dos pacientes terão de ser criptografados na base de dados e não poderão ser visualizados no sistema. Como forma de anonimização, um código será gerado a partir do NHC, um external id, que será um código randomizado e gerado a partir do NHC, para que seja utilizado internamente no sistema, mantendo / garantindo assim a anonimização do paciente.

A entidade *FormPatient* guarda os dados preenchidos dos formulários para cada paciente e por data de preenchimento. Com este modelo, podemos então criar um ecrã (visualização), cujo qual o médico / investigador poderão gerar ficheiros CSV(s) com os dados dos casos de pacientes selecionados, filtrando por formulário, data de registo, especialidade e id do paciente, conforme demonstrado nas entidades *dataset Extraction* e *Data Analysis* da figura 9.

A matriz resultante terá como linhas os pacientes por id interno e as colunas serão os *Answer Field name(s)* de cada questão que foi respondida nos formulários.

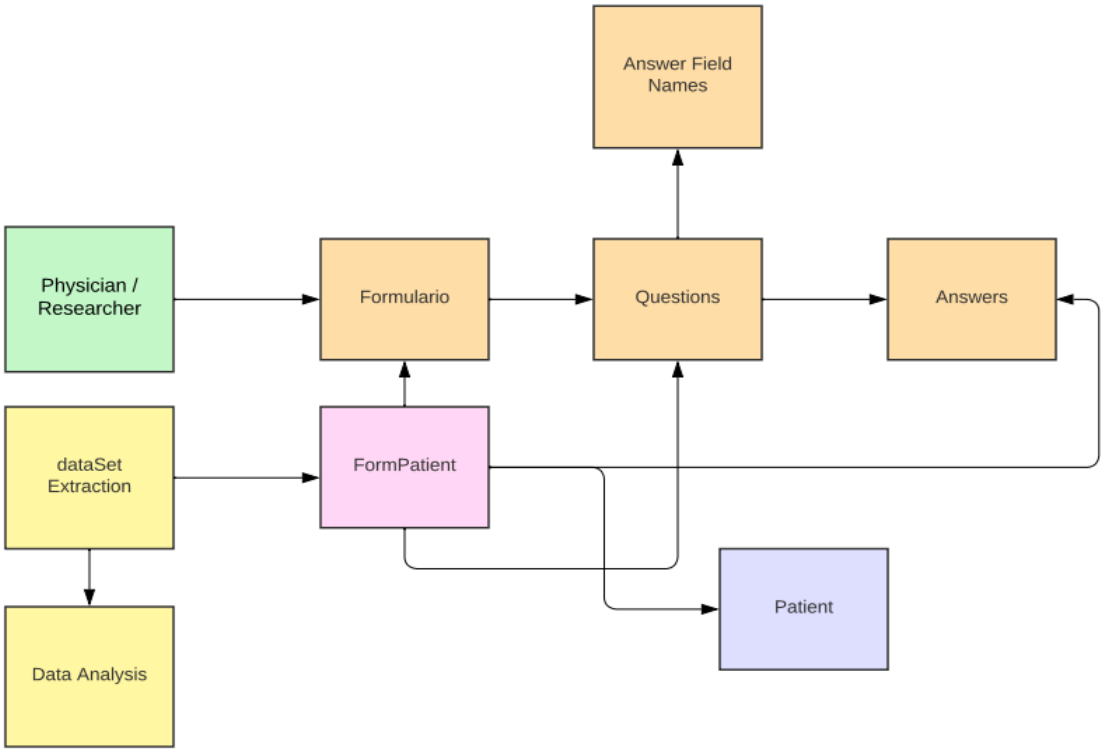


Figura 9. Modelo de entidades do projeto.

Os modelos de entidades desenvolvidos neste trabalho, como são apresentados na figura 9, foram implementados, respeitando os relacionamentos entre os modelos previamente, (conceitualmente), definidos. Na figura 10, se podem observar as tabelas automaticamente geradas, que foram criadas com recurso do Framework *Django*.

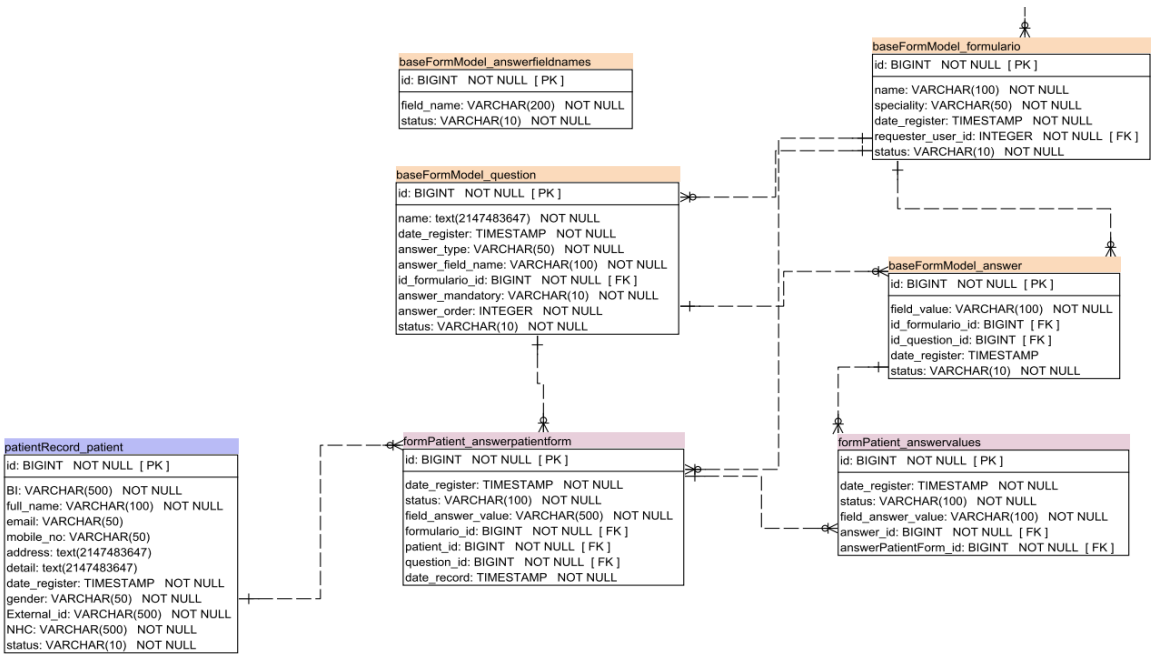


Figura 10. Modelo de entidade e relacionamento Dynamic platform.

3.2 Configurações de utilizadores e permissões da plataforma

Ainda que o *Framework Django* possui seu próprio controle de utilizadores e grupos de utilizadores de acesso, foi necessário adicionar / desenvolver um novo controle de autenticação, para garantir que houvesse três tipos de perfis de usuários diferentes: ***administrador de sistema***, ***investigador*** e um ***utilizador*** (perfil mais básico), só com direito a preenchimento dos formulários. Neste sentido foi então utilizado o import *django.contrib.auth,* que contem os modelos *login, authenticate* e *logout,* conforme demonstrado no código abaixo:

```
django.contrib.auth import login, authenticate, logout
```

Utilizando os modelos acima, houve a necessidade de se criar as tabelas na base de dados, para o armazenamento dos utilizadores, grupos e permissões, conforme figura 11, podemos ver o modelo de dados da parte de login, autenticação e logout da plataforma dinâmica desenvolvido.

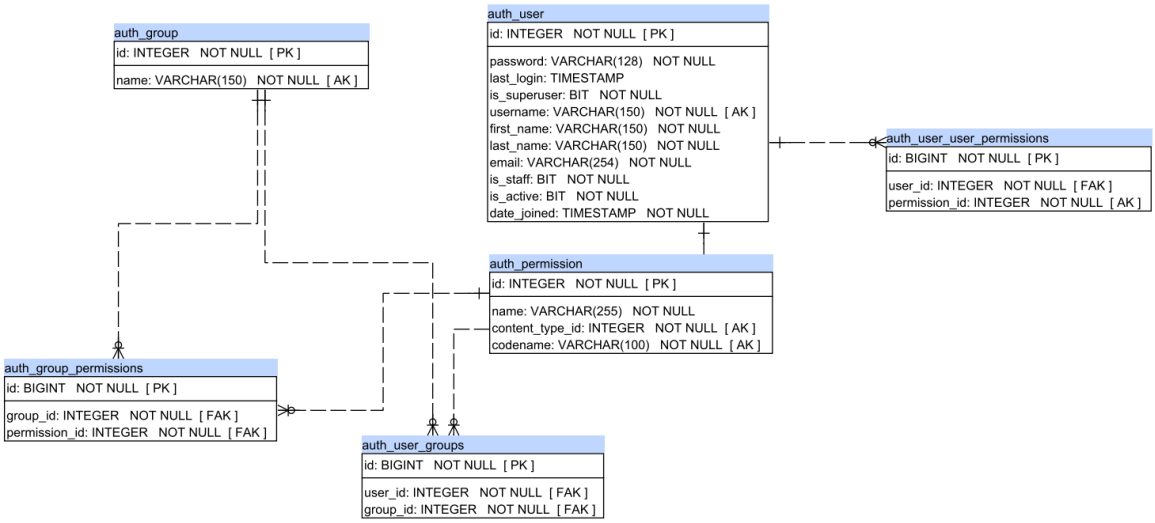


Figura 11. Modelo de entidade e relacionamento django.contrib.auth

O *Django* disponibiliza um site de *BackOffice*, *Django Administration*, que tem o objetivo de controle de dados dos modelos, cadastro e controle de utilizadores e grupos de utilizadores, conforme demonstrado na figura 12.

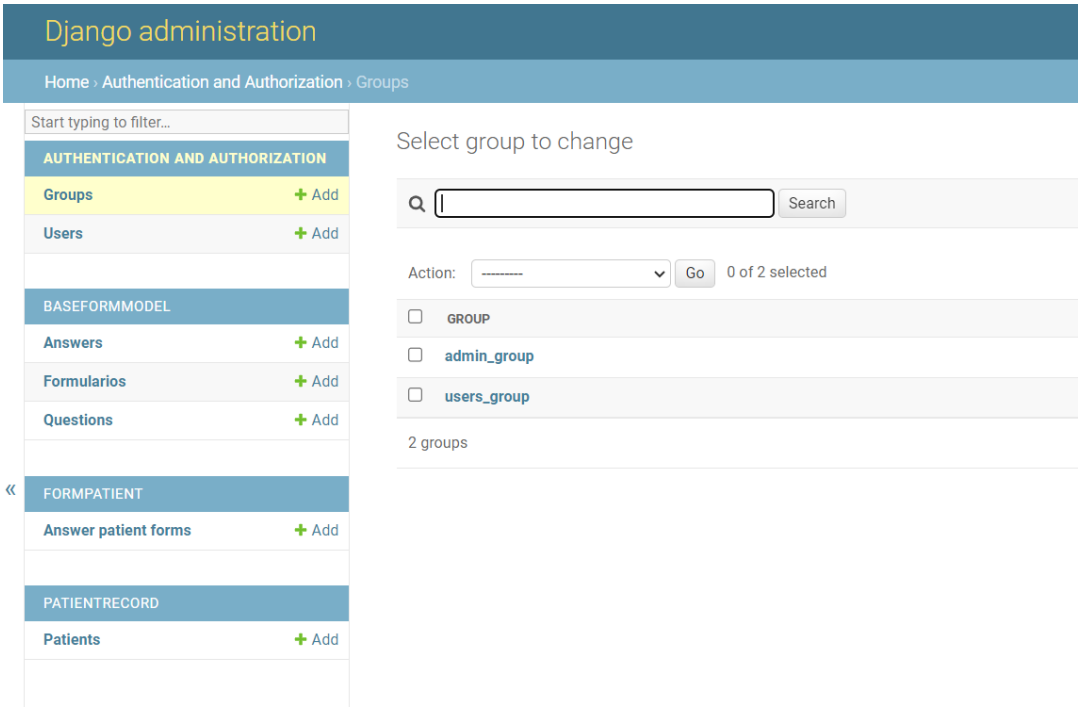


Figura 12. Site de BackOffice Django, administração de dados, modelos, utilizadores e grupos.

3.3 Implementação da plataforma dinâmica

Para a implementação da plataforma dinâmica, dada a versatilidade e complexidade dos formulários "ad-hoc" a serem criados, se decidiu desenvolver um modelo / estrutura cliente – servidor, ou seja, um site contendo as tecnologias definidas para a parte servidor – *back-end* e outro para a parte do cliente – *front-end*. Na figura 13, se descrevem as tecnologias que foram selecionadas, como base para implementar a plataforma dinâmica.

3.3.1 Back-End

Para o "back-end" foi desenvolvido um servidor Web, (hospedeiro do site), uma aplicação para executá-lo e uma base de dados para armazenar as informações. Como o nome sugere, o back-end trabalha na parte de trás da aplicação, escrevendo programas que garantam que o servidor, a aplicação Web e a base de dados funcionem harmonicamente, (em conjunto). O back-end analisa quais são as necessidades de negócio dos stakeholders, (médicos e/ou investigadores em saúde) e fornece soluções de programação eficientes e seguras. Para isso, são utilizadas uma variedade de linguagens do tipo "server-side", como PHP, Ruby, Java ou Python [40]. Nesta tese, todo o desenvolvimento foi feito com recurso à linguagem de programação *Python*.

Foi ainda necessário explorar / utilizar fortemente a linguagem SQL, uma vez que houve necessidade de desenvolver funções e procedimentos complexos para orquestrar o processo de seleção de dados, retornando conjunto de dados com diferentes informações, em diferentes tipos/formatos de dados. Para a criação de procedimentos/funções de base utilizamos a base de dados PostgreSQL,

conforme foi dito anteriormente, este sistema de base de dados objeto-relacional de código aberto com mais de 35 anos de desenvolvimento ativo com uma forte reputação enquanto a características de fiabilidade, robustez e desempenho. Estas funções criadas na base de dados, realizam tarefas mais complexas, onde há relacionamento com várias tabelas e que retornam visualizações para o *front-end*. Para a visualização do formulário de questões e respostas, foi utilizada / ajustada a biblioteca de paginação do *framework Django* [47], com o objetivo de visualizar as questões e respostas por paciente e formulário. No entanto, foi necessário desenvolver uma nova função de acesso à base de dados, para retornar as linhas de questões e respostas que permitem ao site realizar a paginação.

3.3.2 Front-End

O *front-end* tem como objetivo a visualização das páginas/templates do site e permite executar chamadas de funcionalidades do *back-end*, transferindo a informação dos campos do ecrã (views) para os métodos do *back-end*. Nesse sentido, foi necessário criar validações dos campos, como obrigar ao utilizador a preencher algum campo do questionário (dado / informação) ou permitir apenas caracteres numéricos.

Para implementar a parte do *front-end* da plataforma dinâmica, utilizamos / combinamos as linguagens Javascript, Bootstrap e HTML juntamente com a linguagem para templates do Django *Framework*.

Para a gravação dos dados na base de dados, no ecrã de preenchimento do formulário, utilizamos o recurso do AJAX com o JavaScript, conseguindo uma melhor performance.

AJAX, acrônimo de Asynchronous JavaScript and XML, é utilizado como uma técnica que agiliza a interação de aplicações. Uma das principais vantagens é tornar as respostas das páginas Web mais rápidas pela troca de pequenas quantidades de informações com o servidor Web, nos bastidores.

Além disso, evita-se que a página Web inteira tenha de ser recarregada cada vez que alguma nova informação precisa ser consultada no servidor. Em geral, isso significa que páginas Web com recursos AJAX permitem maior interatividade, velocidade de processamento e usabilidade [48].

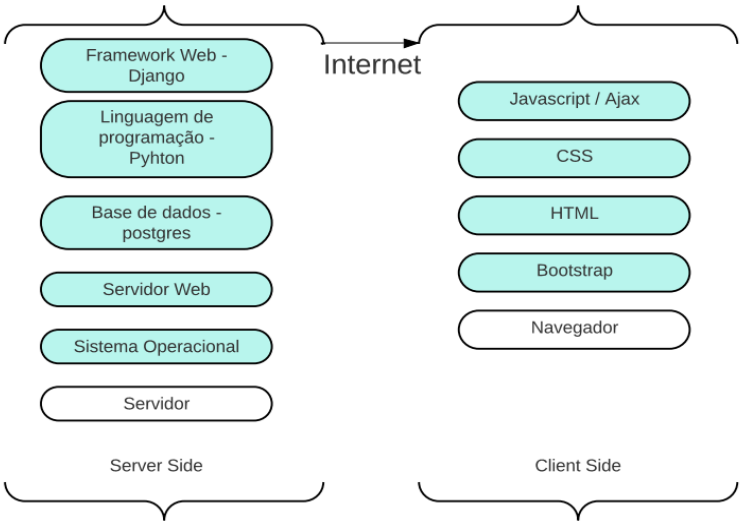


Figura 13. Pilha de tecnologias, adaptada de [40].

3.3.3 Fluxo de atividades da plataforma

Fluxo Inicial

Para um melhor entendimento do fluxo de trabalho da plataforma dinâmica, foi criado o diagrama do fluxo de atividades inicial da plataforma, (figura 14), que corresponde a primeira fase de cadastros necessários.

A figura 14 demonstra que as primeiras ações deverão ser as do Administrador da plataforma, criando os “field names”, (campos técnicos). De seguida, temos o médico / investigador que realizará algum estudo sobre uma determinada especialidade / área de saúde. O investigador terá o papel de definir e cadastrar o formulário, questões e as “Choices”, caso existirem questões do tipo “Choices”, com escolhas definidas, como é demonstrado na figura 14, a condição. O cadastro de uma questão depende do cadastro inicial dos “Field names”, pois uma questão deverá obrigatoriamente conter um “field name” cadastrado.

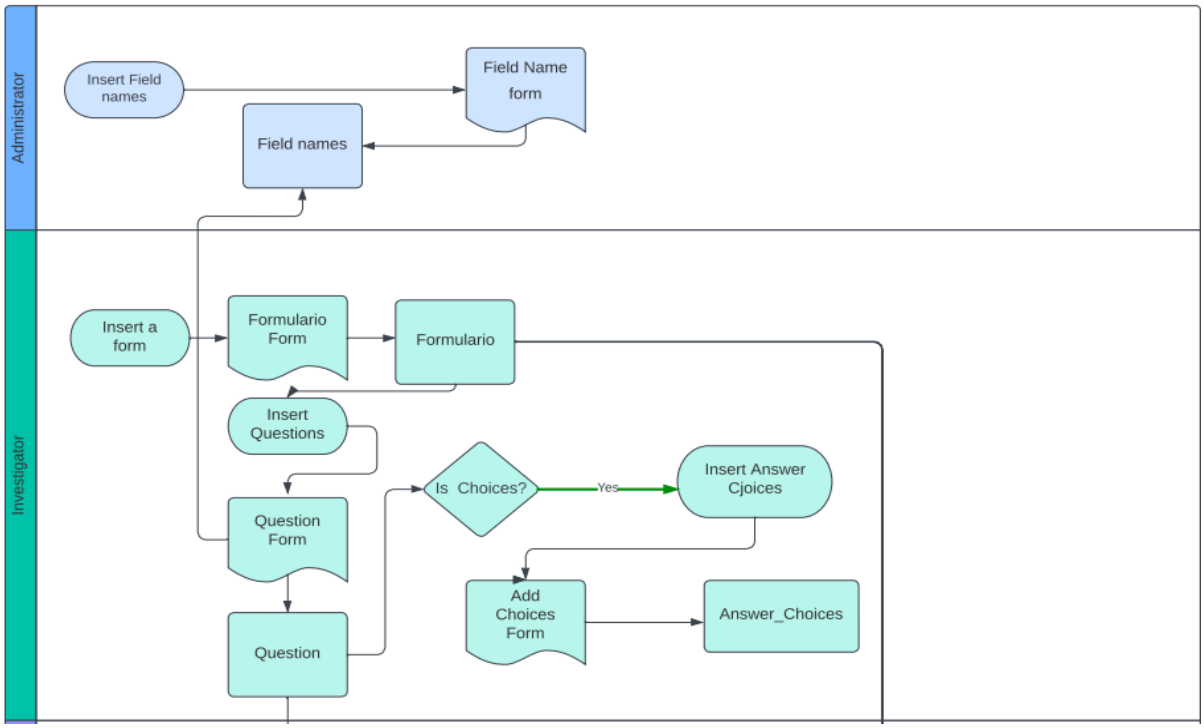


Figura 14. Fluxo inicial, cadastros da plataforma.

Fluxo Completo da Plataforma

Apos a fase de fluxo inicial de cadastros da plataforma, a mesma está preparada para o preenchimento dos formulários. Na figura 15, temos o completo fluxo da plataforma dinâmica e nas raias dos stakeholders “Patient or some employee” e “data curator” podemos ver as atividades de preenchimento dos formulários e a limpeza dos dados / curadoria.

O stakeholder “Patient” poderá ser um enfermeiro ou qualquer outro funcionário da instituição / centro de saúde, que auxiliará o paciente a responder o questionário. No fluxo de preencher o formulário, há a condição de existir ou não o NHC e sua atividade correspondente.

O stakeholder “Data curator” poderá ser o investigador ou qualquer outro analista técnico, ou médico qualificado para a curadoria dos dados. As atividades do data curator serão a de extração dos dados, validação dos mesmos e correção caso houver problemas em respostas dos questionários. Notem que no fluxo de atividades do “Data curator”, existe sempre a possibilidade de se retornar ao formulário para corrigi-lo, e extrair os dados novamente.

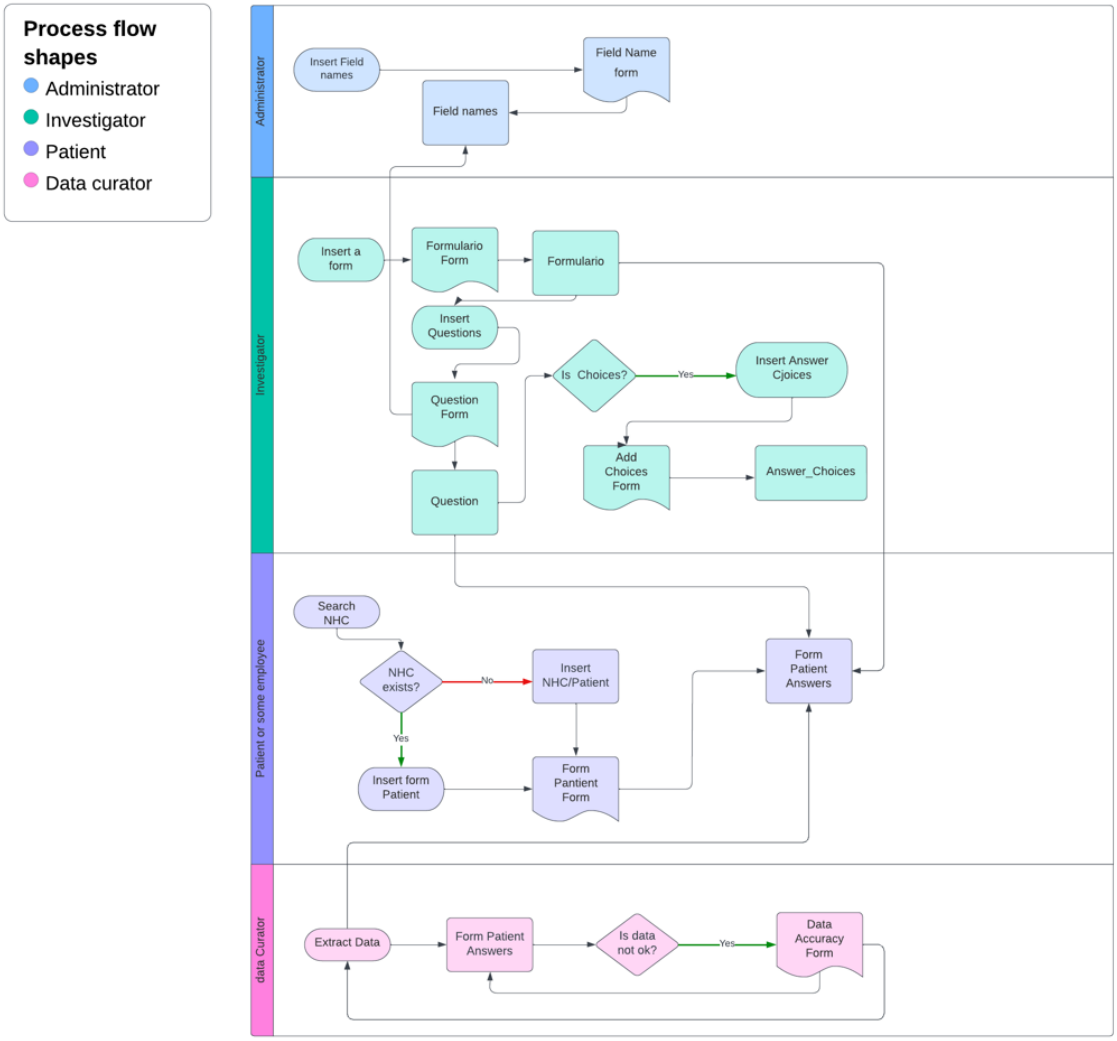


Figura 15. Fluxo final da plataforma.

3.3.4 Menu dinâmico

Para acedermos a todos os ecrãs de cadastros e formulários, foi necessário implementar o menu do site, desenhado / feito à medida, com base na interação com os especialistas médicos.
O menu foi criado considerando dois tipos de visualizações de menu, em relação ao utilizador:

- *Administrator* – cadastros e parte de administração do site, o acesso ao BackOffice.
- *FormPatient* – parte de preenchimento dos formulários, relatório de paciente e extração de dados.

Para a criação do menu, foi necessário a criação de tabelas/ modelos na base de dados e uma view que utiliza o recurso *serializer* do *framework Django* para alcançar uma melhor performance.
Os serializadores permitem que dados complexos, como conjuntos de consultas e instâncias de modelo, sejam convertidos em tipos de dados nativos do *Python* que podem ser facilmente renderizados em JSON, XML ou outros formatos / tipos de conteúdo. Os serializadores também

fornecem opções de desserialização, permitindo que os dados analisados sejam convertidos novamente em tipos complexos, após primeiro validar os dados recebidos.

Os serializadores na estrutura REST, (Representational State Transfer), funcionam de forma muito semelhante às classes Form e ModelForm do *Django*. Os serializadores no *Django* oferecem uma classe *Serializer* que disponibiliza uma maneira poderosa e genérica de controlar a saída de suas respostas, bem como uma classe *ModelSerializer* que fornece um atalho útil para criar serializadores que lidam com instâncias de modelo e conjuntos de consultas [41]. Na figura 16 podemos ver o resultado do menu, com a divisão *Administrator* e *FormPatient*.

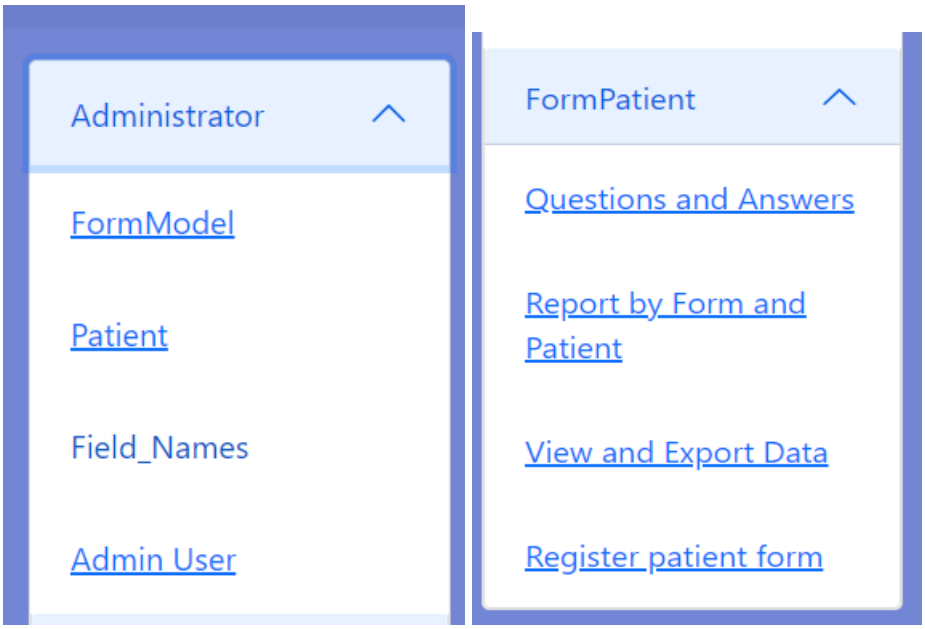


Figura 16 - Menu da Plataforma dinâmica

Para obtermos as permissões de acesso aos menus, foram criados os grupos de utilizadores, *admin_group* e *users_group,* (figura 17). Os grupos foram criados no BackOffice do *Django framework*.

O grupo *admin_group* visualiza todos os menus e submenus e o grupo users_group visualiza apenas os submenus – *Patient*, (cadastro de pacientes), *Questions and Answers*, (preenchimento dos formulários), e *Report by Form and Patient*, (visualização de um relatório do paciente e o formulário que foi respondido).

Para as permissões da plataforma, foi criada uma view de login, que realiza a verificação do grupo do utilizador logado e de seguida retorna as permissões aos itens do menu, conforme o grupo identificado.

As regras de acesso aos menus/submenus poderão futuramente serem redefinidas na plataforma.

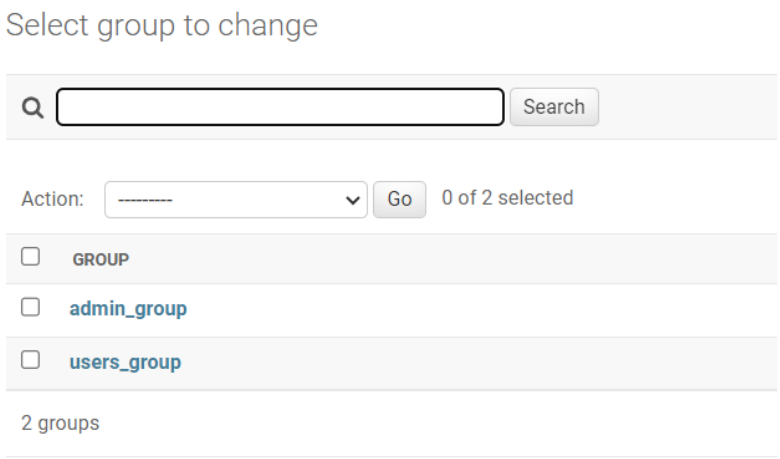


Figura 17. Cadastro de grupos de utilizadores, Django backOffice.

3.3.5 Cadastro do formulário e questões

Para o preenchimento dos formulários dinâmicos, houve a necessidade de se criar um ecrã para cadastrar/atualizar formulários e suas respetivas questões (figura 18).

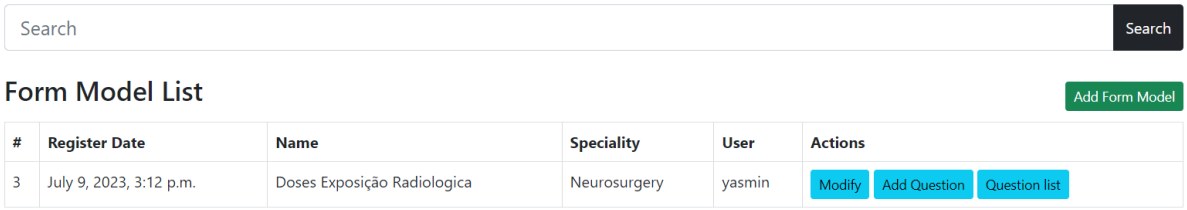


Figura 18. Formulário cadastrado e opções de ações.

Conforme figura 19, inicialmente um dos primeiros ecrãs a serem desenvolvidos foi o de cadastro dos campos técnicos, (*Answer Field Name*). Na figura 19, temos o ecrã que visualiza a lista de campos técnicos cadastrados, (*Field names list*), cuja lista é visualizada no ecrã desenvolvido para o cadastro de questões, (*Questions*), conforme aparece a lista na figura 20, no campo *"Answer field name".*

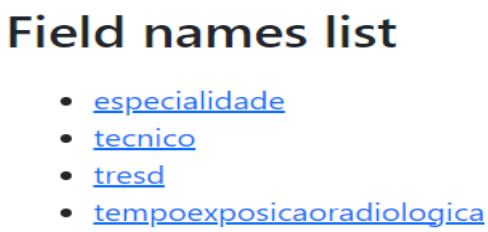


Figura 19. Lista de *field names* cadastrados.

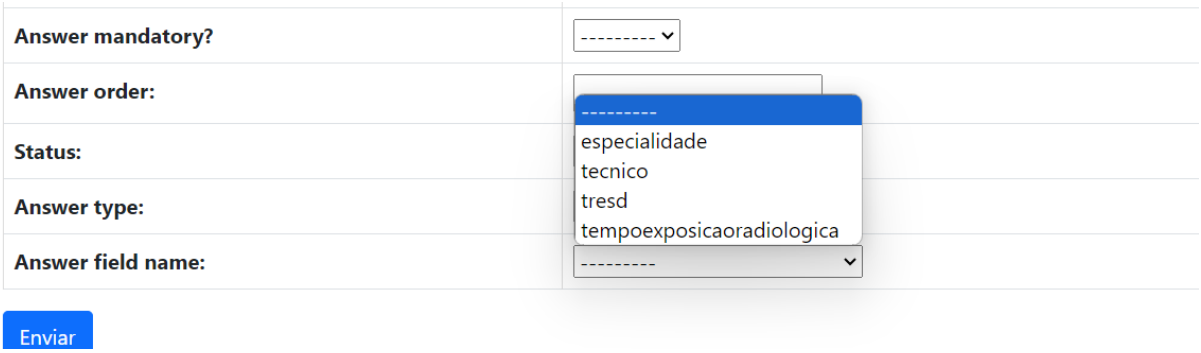


Figura 20. Lista de *field names* cadastrados no ecrã de cadastro da questão.

Para contemplar os tipos de questões, como *Text, choices* e *Number,* o ecrã de cadastro de questões, figura 21, oferece o campo *Answer Type.* Assim, ao selecionar uma questão na base de dados, conseguimos identificar qual é o *Answer type* definido para a mesma.

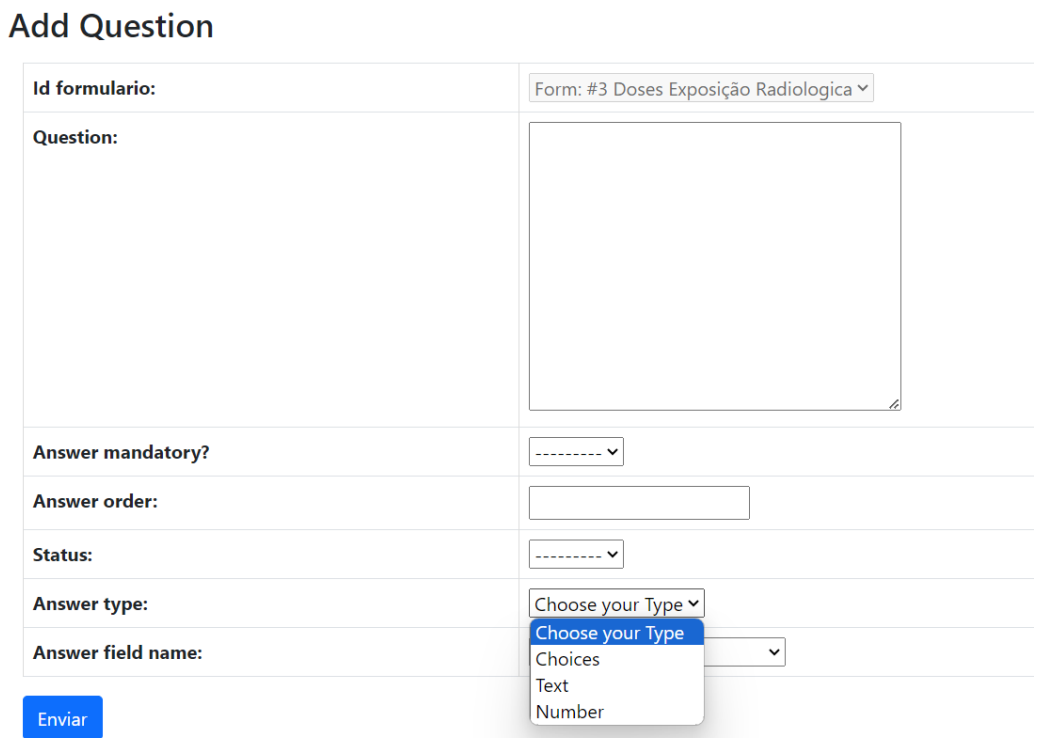


Figura 21. Ecrã de cadastro de questões de um formulário.

No ecrã “question list”, que foi desenvolvido para listar todas as questões de um formulário cadastrado, houve a necessidade de se disponibilizar o botão “Add Choices”, figura 22. Ao clicar no botão “Add Choices”, o aplicativo redireciona a página para o cadastro das escolhas, *(Choices)*, figura 23.

O ecrã da figura 23, de cadastro das opões, *(Choices)*, teve de ser construído para suportar uma tabela em HTML, permitindo assim que o utilizador visualize num só ecrã, todas as opções de respostas cadastradas, e que possa incluí-las, (botão “*Add Answer*”), ou excluí-las utilizando o botão “*delete*”, figura 23.

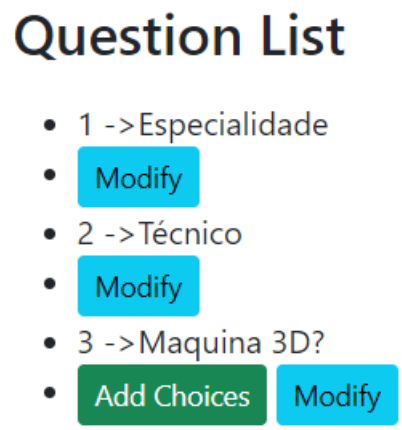


Figura 22. Lista de questões.

Figura 23. Ecrã para cadastro de opções de respostas de uma questão.

3.3.6 Cadastro de pacientes

Para cadastrar o paciente, foi necessário a criação de um ecrã de cadastro de pacientes, um ecrã de lista/pesquisa de pacientes, (*Patient list*), e conforme foi definido anteriormente, no ecrã de preenchimento do formulário, houve a necessidade de se criar uma função que automaticamente, realize a busca do paciente na base de dados e de seguida, realize a inserção na base de dados caso o mesmo não exista.

Juntamente com os especialistas médicos do hospital da luz de Setúbal, foi definido que o número do NHC do paciente é o campo escolhido para a anonimização e identificação do mesmo.

O número NHC é um código interno que identifica o paciente no sistema do hospital da Luz.

Para que o paciente seja anonimizado, foi necessário a criação de uma função que ao receber o NHC digitado pelo utilizador, o mesmo seja randomizado e gravado no campo definido, external_id. A randomização é uma forma de anonimização[49]. Para a randomização, a biblioteca **random** [50] do *Pyhton* foi utilizada, como base, na criação da função **randomize_values**. O código para a

anonimização e encriptação do NHC foi inserido no próprio modelo do paciente. Foi necessária a criação da função ***save*** no modelo *Patient*, para que, no momento de inserção de qualquer paciente, seja gerada essa randomização e também a encriptação do NHC na base de dados.
Os campos Full_name e gender são de preenchimento opcional.

Utility

As funções de randomização e encriptação dos dados, foram criadas nos ficheiros anonymization.py e encryption_util.py dentro de uma pasta denominada ***utility***, assim, as funções criadas puderam ser reutilizadas dentro de outros modelos e views. As funções criadas **encrypt** e **decrypt**, servem para encriptar qualquer informação em base64[51], (uma chave constituída por 64 caracteres nos padrões Ascii85 e Base85).
Dentro do ficheiro anonymization.py foram criadas as funções **split** e **randomize_values** para realizarem a randomização do código NHC. Na figura 24, podemos ver o ecrã de criação de um paciente, onde é chamada a função **save** mencionada anteriormente.

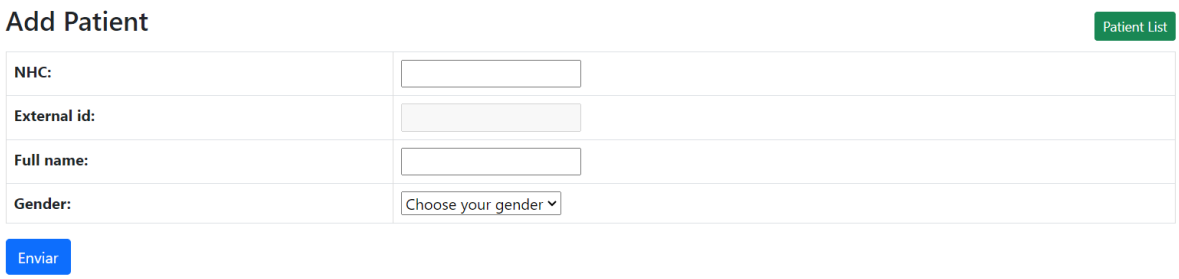


Figura 24. Ecrã add patient.

Search

Nos ecrãs de preenchimento do formulário, houve a necessidade de se criar um campo *Search*, (pesquisa), com o objetivo de retornar o paciente. Para isso, foi construída uma view/função que ao receber o conteúdo preenchido pelo utilizador, no campo *search, a* função realiza uma *lógica de* busca na tabela/modelo de pacientes, pesquisando pelos campos id interno do paciente, campos data de registo do paciente, *external_id*, NHC ou *full_name*. Na figura 25 podemos ver o campo *search* sendo utilizado no ecrã *Patient List*, (lista de pacientes).

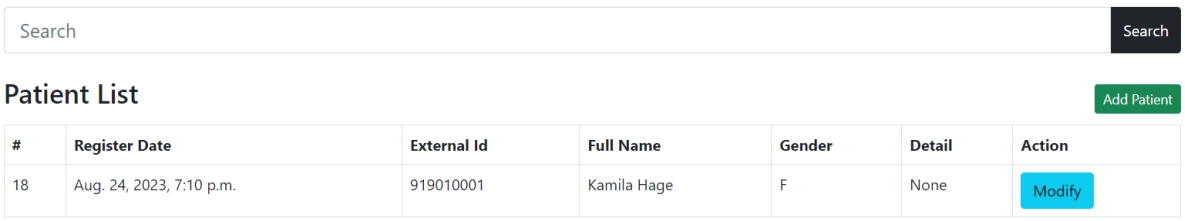


Figura 25. Ecrã patient list.

Update Patient

Para atualização de um paciente, foi criada uma view de “update Patient”. A view permite realizar atualização do paciente, nos campos full name, gender, status e Detail, (figura 26). Existe a opção de remover o paciente alterando o campo status de Active para Inactive, ou seja, ao atualizar o paciente para Inactive, o mesmo não aparecerá mais nos ecrãs, somente na extração dos dados, como histórico.

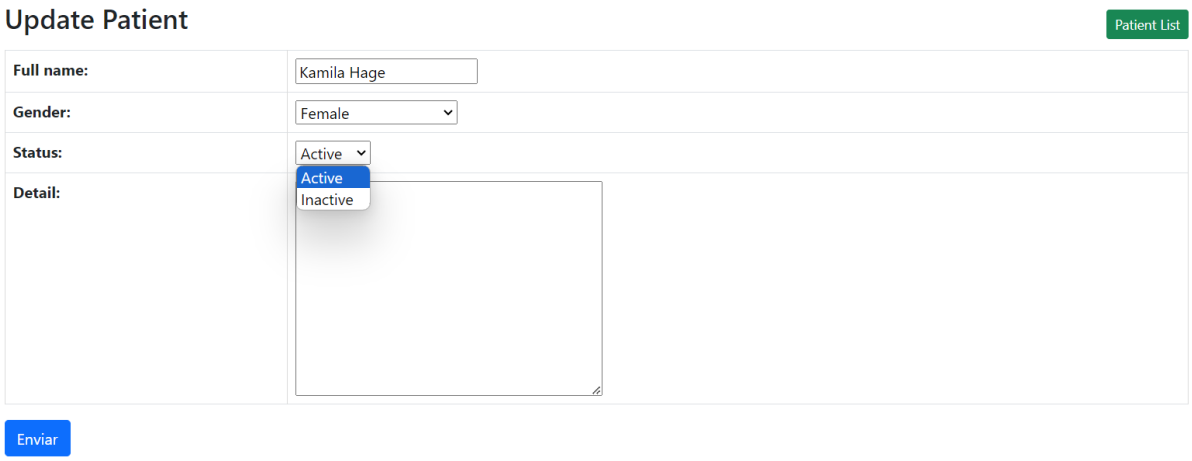


Figura 26. Ecrã de atualização do paciente.

3.3.7 Preenchimento do Formulário, paginação

A solução para a criação do ecrã para o preenchimento do formulário por paciente e data do registo, foi desenvolvida utilizando uma biblioteca de paginação, import paginator[47] e uma função de base de dados *get_form_patient_register* que retorna os dados em formato JSON. A função *get_form_patient_register,* contem uma seleção dos dados entre as tabelas onde são armazenados os formulários, as questões e a tabela de repostas de um paciente, formulário e questão numa determinada data. A lógica interna da função teve de utilizar duas seleções com o operador SQL **UNION**. A primeira seleção contém as questões do formulário e a segunda seleção, contém as questões do formulário com as respostas do paciente na data preenchida, caso existam.

Após a criação da função de base, a view foi criada com o objetivo de executar essa função, obter o seu objeto de retorno e realizar a lógica de visualização do ecrã / template. Para a visualização das questões do tipo “Choices”, “dropdownlist”, (lista de valores), ou do tipo “Number” ou “Text”, (input de dados), foi necessário, com base no retorno da base de dados, criar dois objetos, um de questões, (Questions), e outro objeto de respostas, (Answers). O objeto “answers” é lido dentro do objeto das questões, (Questions). São “loops” encadeados, com quebras por questão e tipo de questão, que assim formam a logica da view. A view, por fim, envia estes objetos já prontos para o template.

Dentro do template, foi necessário colocar uma lógica de visualização do HTML com estes objetos.

Quando o tipo de questão possui o tipo de resposta igual a “Choices”, o template visualiza a lista de valores em formato HTML *select option – dropdownlist*, senão, visualiza um campo INPUT do HTML, para preenchimento do utilizador. Como resultado, temos a figura 27, com 3 páginas / Questões e os botões de navegação fir*st,* para navegar para a primeira página e *previous* para ir a página anterior.
Assim, conclui-se o ecrã de preenchimento dos formulários denominado *Questions and Answers*. Neste ecrã o paciente deverá ser previamente cadastrado para que apareça nas pesquisas, (*search*), e ou na lista de valores *patient*.
O ecrã também disponibiliza a pesquisa de formulários já respondidos e a possibilidade de alteração das respostas de possíveis erros de dados, que possam aparecer posterior a extração dos dados.

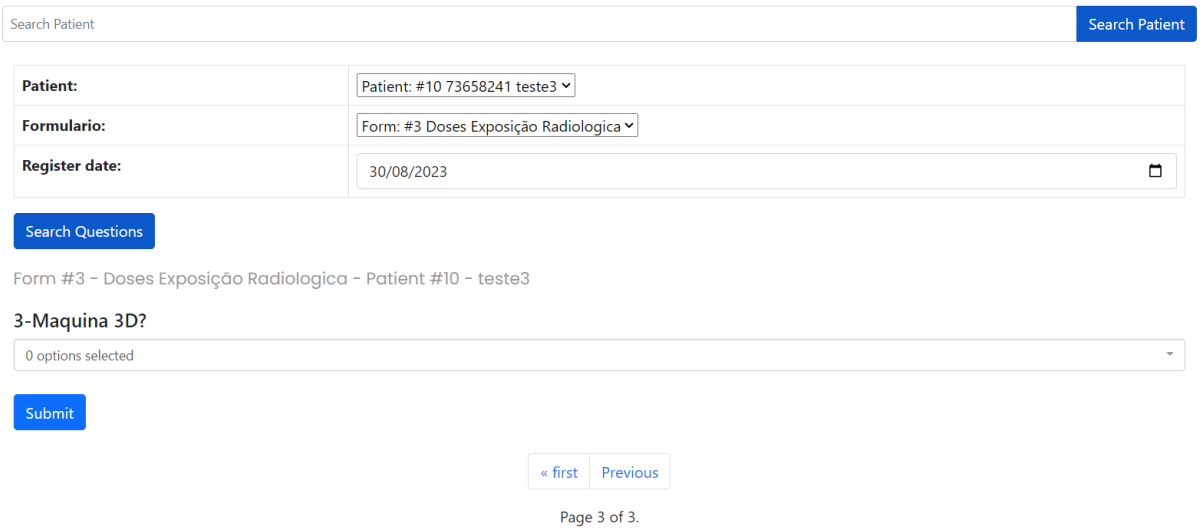


Figura 27. Ecrã de questões e respostas da plataforma dinâmica.

Para a gravação dos dados, foi necessária a criação de uma view que recebe os dados das respostas e grava os dados no respetivo modelo/tabela. A view realiza validações quanto a campos números e realiza a gravação de inserções e ou atualizações na base de dados.

Search Patient

Ainda no ecrã de preenchimento do formulário, para que a lista de valores de pacientes seja sempre uma lista pequena, (prevendo que o cadastro de pacientes tem a tendência de cada vez ser maior ao longo do tempo de utilização da plataforma), foi inserido o campo *search* para que o utilizador pesquise antes o paciente pelos campos NHC, full name, id, (id interno da plataforma dinâmica), external id ou data de registo do paciente. Caso o utilizador não pesquise no campo search, a lista de valores sempre visualiza os pacientes inseridos no mês atual e que estão ativos.

Dropdownlist para Choices

Para visualizar a lista de opções de resposta quando o tipo de questão for igual a *choices*, foi utilizado recurso em JavaScript/HTML combinado com CSS para visualizar a dropdownlist, (lista de valores), com as opções que foram inseridas no cadastro de respostas para a questão, (figura 28).

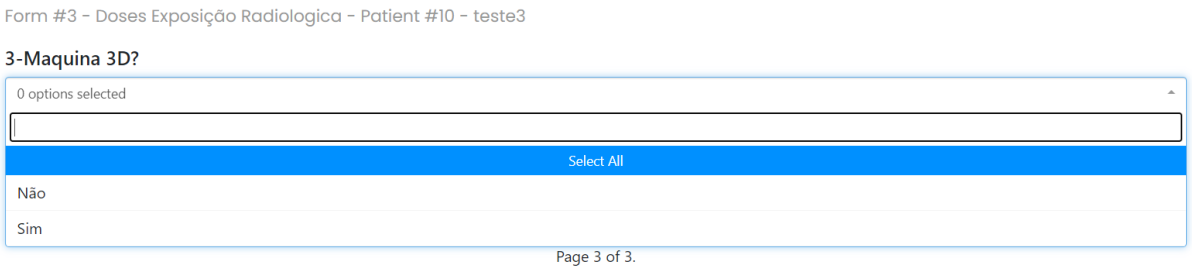


Figura 28. Ecrã do preenchimento de formulários, visualização da lista de valores do tipo de resposta choices.

Respostas do tipo text e number

O template criado para a visualização do formulário também contempla as respostas de texto livre ou numéricas. Caso a questão cadastrada for do tipo Text ou Number, o template disponibiliza um campo em HTML do tipo INPUT que permite ao utilizador o preenchimento de um texto livre quando for do tipo Text. Quanto a questão exigir uma resposta numérica, (Number), a view contem uma lógica de validação, com base no que foi preenchido pelo utilizador. Foi necessário incluir validações de formatos do tipo decimal. O número quando for decimal, necessita ter separador com o caractere ponto (.).

Register patient Form

Também foi nos solicitado, pelo Hospital da luz de Setúbal, que houvesse um ecrã específico para que o utilizador responsável por preencher os formulários tivesse acesso restrito. Foi então criado o ecrã *Register patient form*, que permite o preenchimento do formulário e a possibilidade de criação do paciente internamente, caso o mesmo não exista na plataforma. Se o paciente existir na base de dados, a view deve associar as respostas do formulário ao paciente. Basicamente, o ecrã é parecido com o ecrã mencionado anteriormente, com a diferença de que não terá permissão de pesquisa do paciente, somente o preenchimento do formulário. Na figura 29 temos o campo *Nhc* que deve ser obrigatoriamente preenchido junto com os campos formulário e o *register date*.

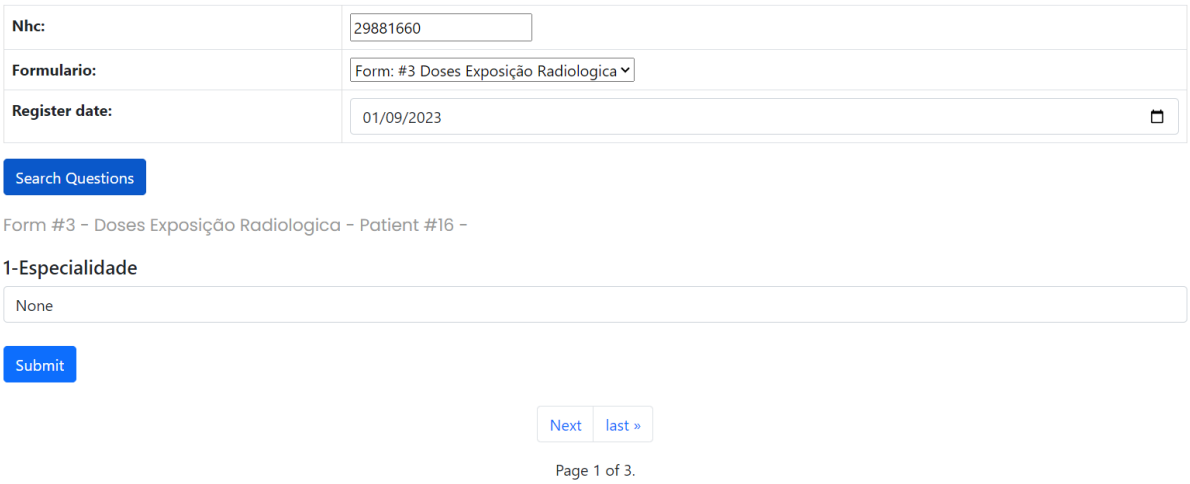


Figura 29. Ecrã Register patient form da Plataforma dinâmica.

Busca do paciente

Para realizar a verificação do paciente na base de dados, considerando que o NHC é gravado encriptado na base de dados, (conforme foi mencionado anteriormente), para verificarmos se o paciente existe, temos de receber o código NHC digitado pelo utilizador e procurar na tabela de pacientes realizando a decriptação do código NHC de todos os pacientes cadastrados.
O intuito é que este ecrã seja apenas para o registo do formulário, e que o utilizador não necessite cadastrar o paciente antes e depois preencher o formulário.

NHC

Para que não existam cadastros erróneos de NHC inexistentes ou erros de utilização deste ecrã, foi definido no ficheiro *form.py* na classe definida para esse ecrã, que o NHC tem um tamanho mínimo de 8 caracteres e um máximo de 20 caracteres.
O ficheiro form.py define campos, validações de input de dados dos ecrãs, um recurso do *framework Django* [52].

Gravar as respostas

Para gravar os dados das repostas, foi criada uma solução que utiliza o recurso *JavaScript/ Ajax* juntamente com o *Django*.
Para gravar as respostas na base de dados, foi criada a função *JavaScript saveQuestionAnswer* que fica gravada dentro da pasta static conforme é definido pelo F*ramework* [53].
A função é acionada pelo *template* conforme requisição do utilizador que ao clicar no botão *submit*, aciona a chamada da função, conforme figura 30.

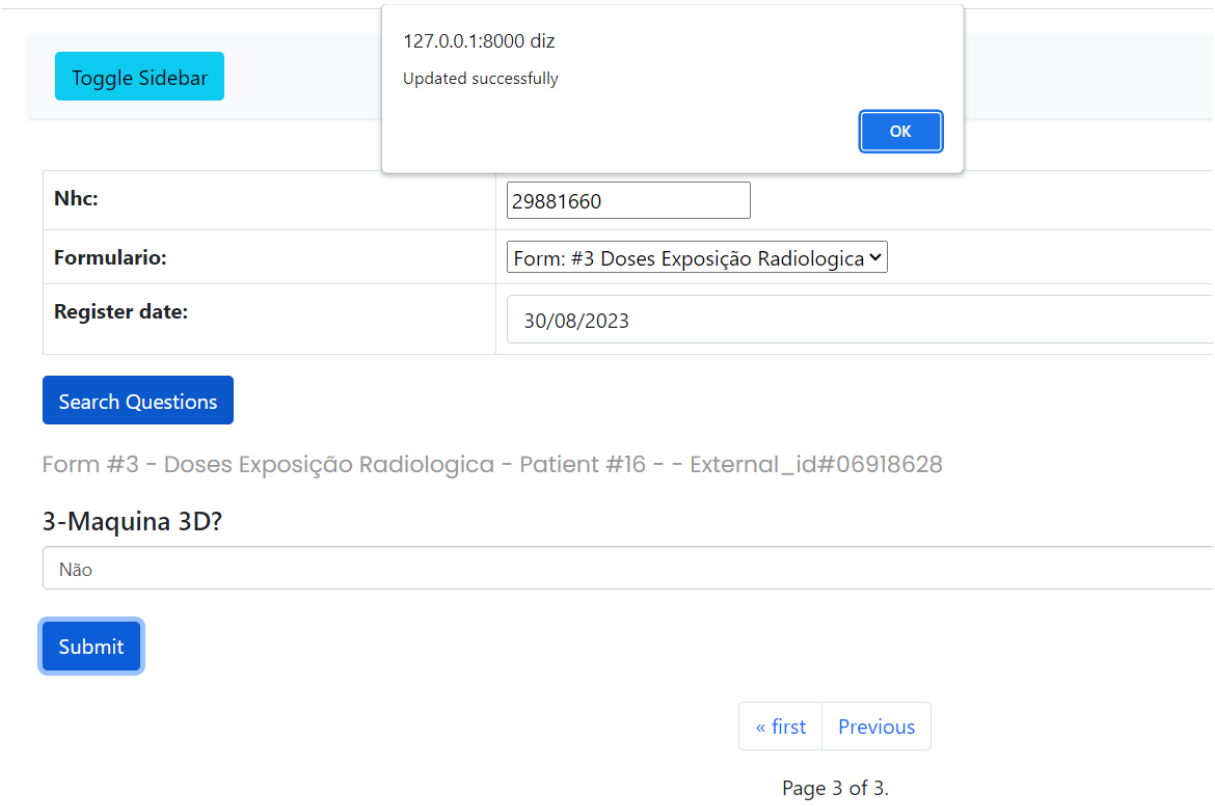


Figura 30. Ecrã com a demonstração da gravação de uma resposta a uma questão do formulário.

A função *JavaScript/ Ajax saveQuestionAnswer* recebe os valores do *template* e depois realiza a chamada da *view updateFormPatient* passando os valores, ver anexo 2.

View edit_answerPatientForm

A view edit_answerPatientForm foi criada com objetivo de validar e gravar os dados na base de dados.
Dentro da logica de validação das repostas do tipo *number*, foi utilizada a biblioteca "**re"** do python que utiliza expressões regulares para validar campos.
Caso o valor inserido pelo utilizador seja incorreto, a view retorna para a função *JavaScript* a mensagem de erro, e a função *JavaScript,* por sua vez, devolve uma mensagem informando ao utilizador o erro.

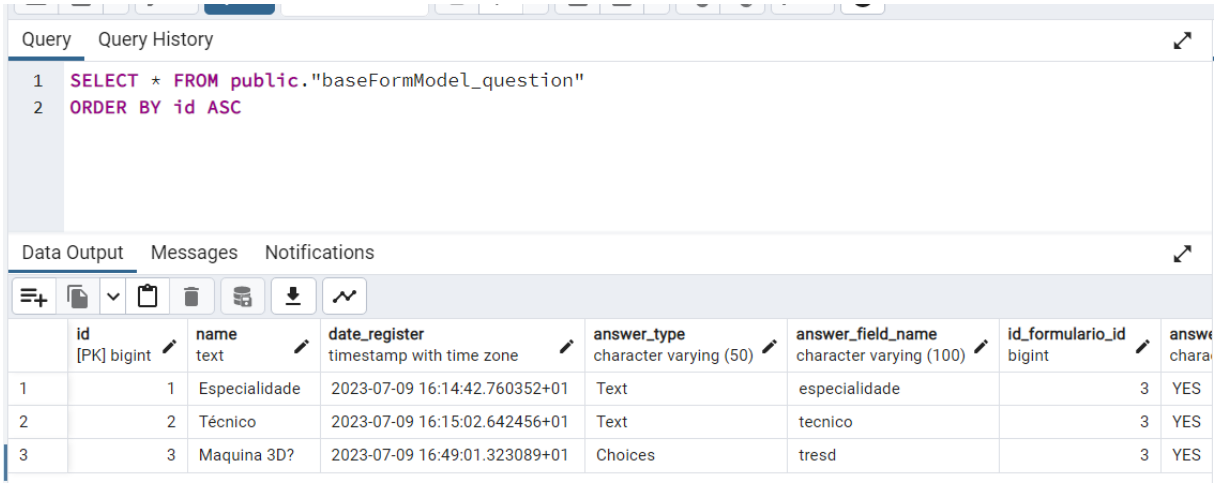
3.3.8 Visualização e extração dos dados

Após a implementação do cadastro dos pacientes, formulários, questões, tipos de respostas e preenchimento do formulário em questão, conforme foi identificado no levantamento de requisitos, houve a necessidade de que a plataforma pudesse disponibilizar um ecrã de visualização e extração dos dados gravados na base de dados. Nas reuniões que tivemos com a diretoria e médicos do hospital da luz de Setúbal obtivemos exemplos de ficheiros em formato Excel, exemplos de datasets

com estudos realizados previamente, onde identificamos que o mesmo é uma matriz de linhas e colunas, onde as linhas são os pacientes e as colunas são os *field names*, ou seja, os *field names* conforme mencionado anteriormente, fazem parte do cadastro das questões, onde o utilizador obrigatoriamente, deve relacionar a questão a um *field name* previamente cadastrado.
Os *field names* são os campos técnicos que definem a que se refere a informação respondida nos formulários.

Transformar linhas em colunas

Para criarmos a matriz em questão houve a necessidade de se transformar linhas da base de dados em colunas. A tabela da figura 31 visualiza os dados da tabela de questões, (baseFormModel_question), em forma de linhas. A tabela do registo das questões de um formulário possui a coluna answer_field_name que foi mencionada na secção de definição do modelo de dados.



	id [PK] bigint	name text	date_register timestamp with time zone	answer_type character varying (50)	answer_field_name character varying (100)	id_formulario_id bigint	answe chara
1	1	Especialidade	2023-07-09 16:14:42.760352+01	Text	especialidade	3	YES
2	2	Técnico	2023-07-09 16:15:02.642456+01	Text	tecnico	3	YES
3	3	Maquina 3D?	2023-07-09 16:49:01.323089+01	Choices	tresd	3	YES

Figura 31. Tabela de questões da Plataforma dinâmica

Tipos Compostos

A solução encontrada para transformar linhas em colunas foi utilizando recurso da base de dados PostgreSQL, utilizando a criação do objeto de base tipo composto.
Um Composite Type, (tipo composto), representa a estrutura de uma linha ou registo; é essencialmente apenas uma lista de nomes de campos e seus tipos de dados. O PostgreSQL permite que tipos compostos sejam utilizados com os mesmos métodos que os tipos simples podem ser utilizados. Por exemplo, uma coluna de uma tabela pode ser declarada como sendo do tipo composto [54].
Para a plataforma dinâmica, o tipo composto tem a necessidade de que sua criação seja dinâmica, ou seja, toda vez que o utilizador solicitar os dados da base, o tipo composto deve ser recriado, assim contemplamos os novos *field names* referentes a novas questões de novos formulários.
A solução para a criação dinâmica foi a criação de um procedimento de base que é chamado toda vez que o utilizador requisitar a geração dos dados, este procedimento recria o tipo composto

obtendo os dados agrupados da tabela de questões.

JSON no PostgreSQL

Para retornamos os dados gravados utilizando o composite type, houve a necessidade de utilização das funções e operadores de manipulação do tipo JSON do postgreSQL.
Estas funções ajudam no retorno dos dados agrupados no formato JSON para que no Django Framework possamos manipulá-los utilizando as bibliotecas *Python* que possuem facilidade de manipulação de dados do tipo JSON.
Os tipos de dados JSON servem para armazenar dados JSON (JavaScript Object Notation), conforme especificado na RFC 7159 [55]. Esses dados também podem ser armazenados como texto, mas os tipos de dados JSON têm a vantagem de garantir que cada valor armazenado seja válido de acordo com as regras JSON. Existem também diversas funções e operadores específicos de JSON disponíveis para dados armazenados nesses tipos de dados; consulte a Seção 9.16 [56].
O PostgreSQL oferece dois tipos de armazenamento de dados JSON: json e jsonb. Para implementar mecanismos de consulta eficientes para esses tipos de dados, o PostgreSQL também fornece o tipo de dados jsonpath descrito na Seção 8.14.7.
Os tipos de dados json e jsonb aceitam conjuntos de valores quase idênticos como entrada. A principal diferença prática é a eficiência. O tipo de dados JSON armazena uma cópia exata do texto de entrada, que as funções de processamento devem analisar novamente em cada execução; enquanto os dados jsonb são armazenados em um formato binário decomposto, o que torna a entrada um pouco mais lenta devido à sobrecarga de conversão adicional, mas o processamento é significativamente mais rápido, já que nenhuma nova análise é necessária. Jsonb também oferece suporte à indexação, o que pode ser uma vantagem significativa.
Como o tipo json armazena uma cópia exata do texto de entrada, ele preservará espaços em branco semanticamente insignificantes entre os *tokens*, bem como a ordem das chaves nos objetos JSON. Além disso, se um objeto JSON dentro do valor contiver a mesma chave mais de uma vez, todos os pares chave/valor serão mantidos. As funções de processamento consideram o último valor como o operacional, por outro lado, jsonb não preserva espaços em branco, não preserva a ordem das chaves de objeto e não mantém chaves de objeto duplicadas. Se chaves duplicadas forem especificadas na entrada, apenas o último valor será mantido.
Em geral, a maioria dos aplicativos deve preferir armazenar dados JSON como jsonb, a menos que haja necessidades bastante especializadas, como suposições herdadas sobre a ordenação de chaves de objetos [56].
Com base nas funções de tratamento do tipo JSON do postgreSQL, foi possível criar uma função de base de dados que retorna um tipo **json**. A função possui uma seleção dos dados que foram armazenados do preenchimento dos formulários. A seleção utiliza a função **json_populate_record** do postgreSQL que expande o objeto json para uma linha com colunas que correspondem ao tipo de registo da base especificado, no caso o **Composite type** mencionado anteriormente. No anexo 3, podemos ver parte do código da função de base de dados criada.

A função **json_agg()**: reúne todos os valores de entrada, incluindo valores nulos, em um array JSON. De acordo com to_json, os valores são transformados em JSON.
Sintaxe: json_agg (qualquer elemento)
Tipo de retorno: json.

A função **json_object_agg()**: todos os pares chave/valor são reunidos em um objeto JSON. Os argumentos de valor são convertidos por to_json enquanto os argumentos principais são convertidos em texto. As chaves não podem ser nulas, mas os valores podem.
Sintaxe: json_object_agg (chave “qualquer”, valor “qualquer”).
Tipo de retorno: json.

Por que armazenar ou consultar dados JSON no PostgreSQL?

A capacidade da base de dados PostgreSQL de armazenar e consultar dados JSON é uma de suas características distintivas. Anteriormente, para processar dados JSON, os analistas e engenheiros de dados precisavam recorrer ao armazenamento de documentos especializados, como o MongoDB[57]. O fascínio das bases de dados relacionais é a capacidade de “gravar dados agora e organizar o esquema depois”. Qualquer estrutura de dados pode ser armazenada como texto simples em base de dados como PostgreSQL e MySQL. O processamento e a velocidade, entretanto, eram problemas, uma vez que a base de dados não tinha conhecimento intrínseco do esquema do documento.
Anteriormente, a base de dados precisava carregar e analisar o blob de texto completo para cada consulta. Além disso, expressões regulares complicadas tiveram que ser usadas ao consultar profundamente o registro JSON [58].
No entanto, o requisito de um armazenamento de documentos externo não é mais necessário devido à robusta funcionalidade JSON incluída no PostgreSQL (após a versão 9.2) [58].

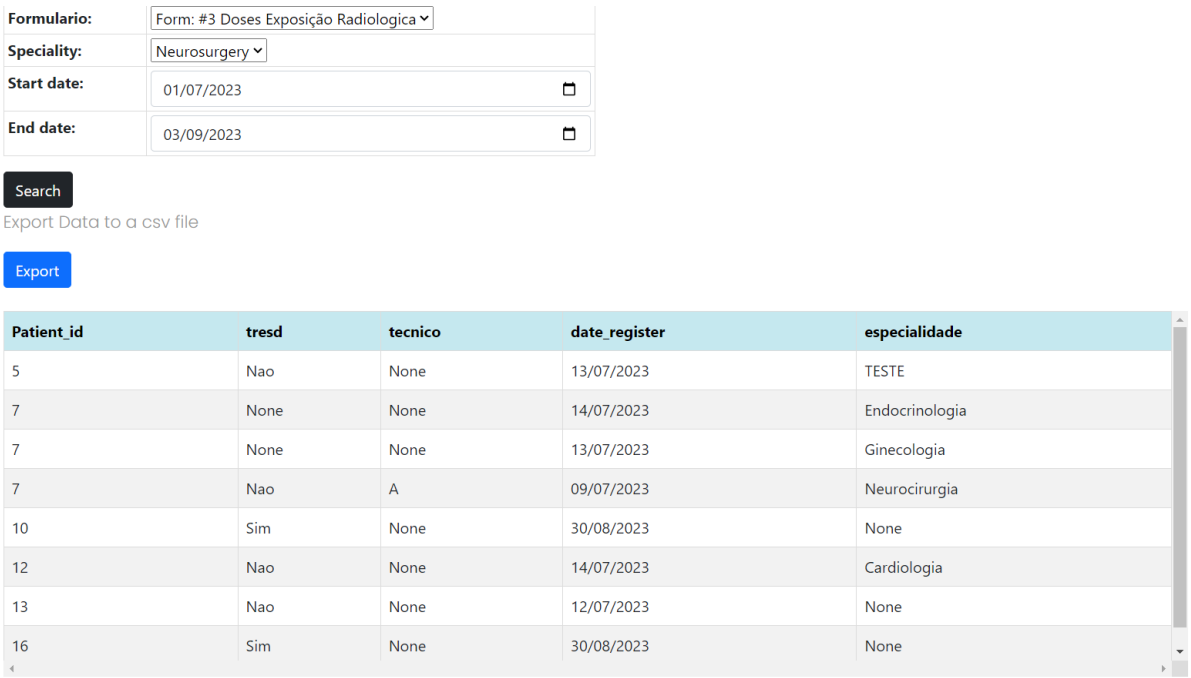
View get_dynamic_transpose_json

Após a criação da função de base de dados **get_dynamic_transpose_json**, a view **get_dynamic_transpose_json** foi criada para visualizar os dados no site via template.
As bibliotecas e funções python pandas, json_normalize e json, foram utilizadas na view para que os dados fossem visualizados no template em forma de uma tabela.
A função de base de dados *get_dynamic_transpose_json* é chamada de dentro da view, em seguida o objeto de retorno possibilita a criação de uma estrutura de dados DataFrame.
A estrutura de dados também contém eixos rotulados, (linhas e colunas). As operações aritméticas se alinham nos rótulos de linha e coluna. Pode ser pensado como um container semelhante a um ditado para uma série de objetos. A estrutura de dados primária do pandas [59].
A biblioteca **json** do Python auxilia com a função loads () na carga da linha retornada pela função da base de dados que retorna uma linha em formato json. A função *json_normalize* da biblioteca *“pandas”*, por fim transporta os dados do JSON para a estrutura criada pelo dataframe. A função

json_normalize, normaliza os dados JSON semiestruturados em uma tabela simples [60].
Para transformar o id do paciente em linha na tabela, foi utilizada a função *set_index()* que define o índice do DataFrame usando colunas existentes [61].

Visualização do template

Para visualizar os dados da view no HTML, foi criado o template que utiliza uma linguagem que mistura o HTML com uma linguagem de programação própria do template.
No caso, a view redireciona para o template o objeto definido de nome **DATA** que tem a estrutura de um DataFrame e contem os objetos *columns e iterrows* que podem ser iterados e serem visualizá-los utilizando o comando **for**, (laço), que percorre o objeto.
Na figura 32 podemos ver o resultado de toda a solução, o ecrã da extração de dados.

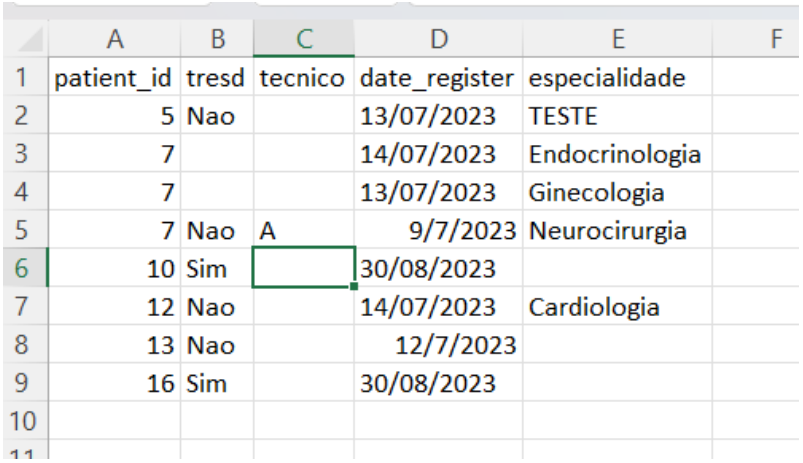


Patient_id	tresd	tecnico	date_register	especialidade
5	Nao	None	13/07/2023	TESTE
7	None	None	14/07/2023	Endocrinologia
7	None	None	13/07/2023	Ginecologia
7	Nao	A	09/07/2023	Neurocirurgia
10	Sim	None	30/08/2023	None
12	Nao	None	14/07/2023	Cardiologia
13	Nao	None	12/07/2023	None
16	Sim	None	30/08/2023	None

Figura 32. Ecrã para visualização e exportação dos dados.

Exportar dados no formato CSV

Para exportar os dados em formato CSV foi utilizada a função *to_csv* da biblioteca *pandas, (Pyhton)*[62].
A função de view de exportação dos dados em formato CSV não executa novamente a geração dos dados, apenas obtém o objeto **DATA** do template e gera o ficheiro CSV a partir desse objeto.



	A	B	C	D	E	F
1	patient_id	tresd	tecnico	date_register	especialidade	
2	5	Nao		13/07/2023	TESTE	
3	7			14/07/2023	Endocrinologia	
4	7			13/07/2023	Ginecologia	
5	7	Nao	A	9/7/2023	Neurocirurgia	
6	10	Sim		30/08/2023		
7	12	Nao		14/07/2023	Cardiologia	
8	13	Nao		12/7/2023		
9	16	Sim		30/08/2023		
10						

Figura 33. Ficheiro exportado do ecrã de visualização e exportação dos dados.

Capítulo 4

Análise e Discussão dos Resultados

Cenário Experimental da plataforma dinâmica

4.1 Testes funcionais

Com o objetivo de obtermos um cenário (experimental) de testes funcionais, o primeiro protótipo de software (evolutivo) da plataforma dinâmica desenvolvida foi instalado localmente na PC de um investigador (médico especialista), que ficou responsável em alimentar, utilizar, e avaliar a plataforma experimentalmente. Ao fim dos testes, foi produzido um documento com as propostas de alterações e correções necessárias para testar a plataforma ao nível do Hospital da Luz de Setúbal.

Inicialmente, o médico responsável pelos testes e mais dois enfermeiros realizaram os cadastrados de formulários da área de neurocirurgia para que fossem posteriormente preenchidos com informações de pacientes. Os testes tiveram o acompanhamento e orientações ao utilizador, tendo foco na usabilidade da plataforma de forma mais eficiente.

A cada reunião, o stakeholder investigador relatava por vídeo conferencia os problemas encontrados e possíveis melhorias. O utilizador realizava as ações na plataforma para que o analista de sistemas percebesse o possível problema. Por fim, era criado um documento com os itens a serem corrigidos / melhorados.

Como resultado do teste ou primeiro protótipo da plataforma resultou um documento contendo um conjunto de propostas correções e melhorias, que se descrevem / identificam a seguir:

1) Alterar o tipo de questão “choices”, para ter a possibilidade de se incluir os campos valor e descrição. O valor seria visualizado na extração dos dados e a descrição no momento de responder o formulário.

   Ao criar as opções de “choices”, o médico deparou-se com o problema na extração dos dados, após o preenchimento do formulário com dados de um paciente. Os dados das questões onde o tipo de resposta eram “choices” vinham com apenas a descrição da escolha, e no caso, deveriam vir com um valor, um código correspondente a descrição, definido pelo ivenstigador. Sem o conjunto de valor e descrição, o Excel gerado após a extração ficou com os dados de descrições das escolhas, quando deveria ter apenas os valores correspondentes aos valores que fossem definidos no cadastro das “choices”, no momento de se cadastrar as questões dos formulários.
2) Alterar o menu “questions and answers” para “Data Accuracy”. Com o objetivo de um melhor entendimento dos médicos e investigadores que vão utilizar a plataforma, foi identificado que este ecrã “questions and answers” deveria ter uma descrição informando melhor sua

utilidade. O ecrã na verdade disponibiliza ao médico uma maneira de realizar uma curadoria dos formulários já respondidos e também um “data clean”. O ecrã disponibiliza uma forma de analisar se os dados estão entendíveis e corretos.

3) No ecrã “register patient form” foi sugerido que ao colocar o NHC, caso não existisse na base de dados, fosse questionado ao utilizador, se o mesmo deseja criar o NHC, caso a resposta fosse sim, a plataforma disponibilizaria um ecrã para o preenchimento dos dados do paciente. Caso a resposta fosse não, não seria permitido o preenchimento do formulário.
4) No cadastro do paciente, foi nos solicitado incluir a data de nascimento e não mais o nome completo. Assim a identificação do paciente pode ser mais anonima.
5) Na criação das questões foi identificado que era possível criar duas questões ou mais com o mesmo “field name”, no mesmo formulário, ocasionando o erro de sobrescrever a resposta de questões com o mesmo “field name”.
6) Outro erro identificado no cadastro de questões, foi a possibilidade de se colocar a mesma ordem para N questões de um mesmo formulário.
7) Criar os “field names” de forma tabular. Para uma melhor usabilidade rapidez, o médico identificou uma melhoria na maneira de se cadastrar os “field names”, de forma tabular e posteriormente por assunto, ou especialidade.
8) A mesma sugestão de criação de “field names” de forma tabular foi sugerida para as *questions*.
9) No ecrã “Report by form and patient” foi nos solicitado a pesquisa pelo NHC, pois este é o código principal que os médicos entendem.
10) A ordem que as opções cadastradas no tipo de resposta “choices” estava a ser aleatória. Há a necessidade de se ordenar estas opções no momento de visualização.
11) Possibilidade de se incluir comentários ao longo das questões. Para um melhor entendimento do formulário, o médico identificou que há a necessidade de se colocar comentários sobre os preenchimentos das questões, intercalando entre um grupo de questões a outro, quando há a mudança de assunto, para que a pessoa que esteja respondendo o formulário entenda.
12) O valor “None”, quando uma resposta não é respondida, também foi um problema para o médico, pois o mesmo tem de remover esta palavra “None” no momento de responder o formulário, quando das questões do tipo “Text”, texto livre. Já na extração dos dados, não ocorre esse problema, a informação vem em branco no excel.
13) Intervalo de formato de números quando for o tipo “Number”. Foi identificada uma melhoria no tipo de resposta “Number”. Para esse tipo, o médico identificou a necessidade de também existir a possibilidade de se determinar limites de intervalo possíveis do número, no momento de preenchimento da resposta, e também de se limitar o formato do número, inteiro ou decimal.
14) No preenchimento do formulário foram pedidas melhorias. Quando for clicado no botão “next” para a próxima página, permitir que a resposta inserida pelo utilizador, seja automaticamente

gravada na base de dados. Atualmente o utilizador deve clicar no botão “submit” para gravar na base de dados a resposta. Também foi sugerido que ao final do preenchimento do formulário, seja disponibilizado um botão de validação. O botão validaria se todas as respostas foram preenchidas e informaria ao utilizar a necessidade de respondê-las caso algumas questões estivessem com a resposta sem preenchimento.

15) Foi solicitado que o ecrã “patient” disponibilize mais filtros de pesquisa, para que o investigador encontre o paciente de uma forma mais fácil.

Ao executar a visualização e extração de dados houve erros quanto a nomenclatura dos “field names”. Ocorreram erros quanto a nomenclatura dos “field names”, pois estes são utilizados na criação do “composite type”. Então, tivemos de colocar uma validação no ecrã de cadastro destes campos, para que não permitissem números e caracteres especiais.

As questões dos formulários, são distintas, mas há pacientes em comum.
Os formulários foram preenchidos pelo médico e constatamos o seguinte:

- Dentre as respostas dos formulários, 3 pacientes responderam os dois formulários;
- Os pacientes que estão nos dois formulários, podem ser pesquisados no ecrã de visualização e extração de dados, somente com o filtro de período de datas e/ou com o id do paciente;
- Ao visualizar as respostas dos pacientes de múltiplos formulários, as linhas de pacientes cujo campos técnicos não pertencem as respostas a questões de um formulário específico, ficam com o valor “None”, significando que aquela coluna não pertence a um determinado formulário. Na figura 34 podemos ver a situação dos pacientes 4 e 9 que possuem 2 linhas e com respostas e campos técnicos, (colunas), com suas respetivas respostas.
- As linhas podem repetir-se por paciente, quando há um filtro somente por data.

Na figura 34, podemos ver a tabela gerada em HTML que é uma prévia do ficheiro CSV.

Patient_id	dor	empe	sexo	andar	idade	pesos	tresd	dormir	sentar	sexual	social	timing	viajar	kapugym	tecnico	airkerma	segmento
4	2	1	F	0	None	2	None	1	1	0	0	3M	0	None	None	None	None
4	None	None	M	None	50	None	Nao	None	None	None	None	None	None	58.07	A	2.8	Lombar
5	None	None	M	None	69	None	Nao	None	None	None	None	None	None	1817.41	B	138.9	Lombar
7	None	None	M	None	75	None	Nao	None	None	None	None	None	None	160.44	V	7.8	Cervical
9	0	0	F	0	None	1	None	1	3	0	0	1 vez	0	None	None	None	None
9	None	None	F	None	72	None	Nao	None	None	None	None	None	None	8.42	A	0.6	Lombar
10	None	None	F	None	42	None	Nao	None	None	None	None	None	None	16.39125	A	13.95	Lombar
11	None	None	F	None	65	None	Nao	None	None	None	None	None	None	10.38	V	0.5	Lombar

Figura 34. Ecrã de visualização e extração de dados.

4.2 Alterações na plataforma após os testes funcionais

Após os testes funcionais, foi possível identificar os principais problemas técnicos e de conceitos para que a plataforma fosse implantada. Abaixo seguem-se as modificações, alterações e/ou melhorias implementadas / realizadas:

1) **Choices**

O ecrã de cadastro de opções de resposta para o tipo de resposta “Choices” foi alterado. Para uma melhor usabilidade, foi criada a opção “modify” um botão que permite alteração das opções na própria tabela HTML disponível no template.

Também foram incluídos os campos “description” e “order” a fim de possibilitar a ordenação das opções e a visualização das descrições das opções, conforme demonstrado na figura 35. Para esse desenvolvimento foi utilizado o recurso ajax e javascript [48].

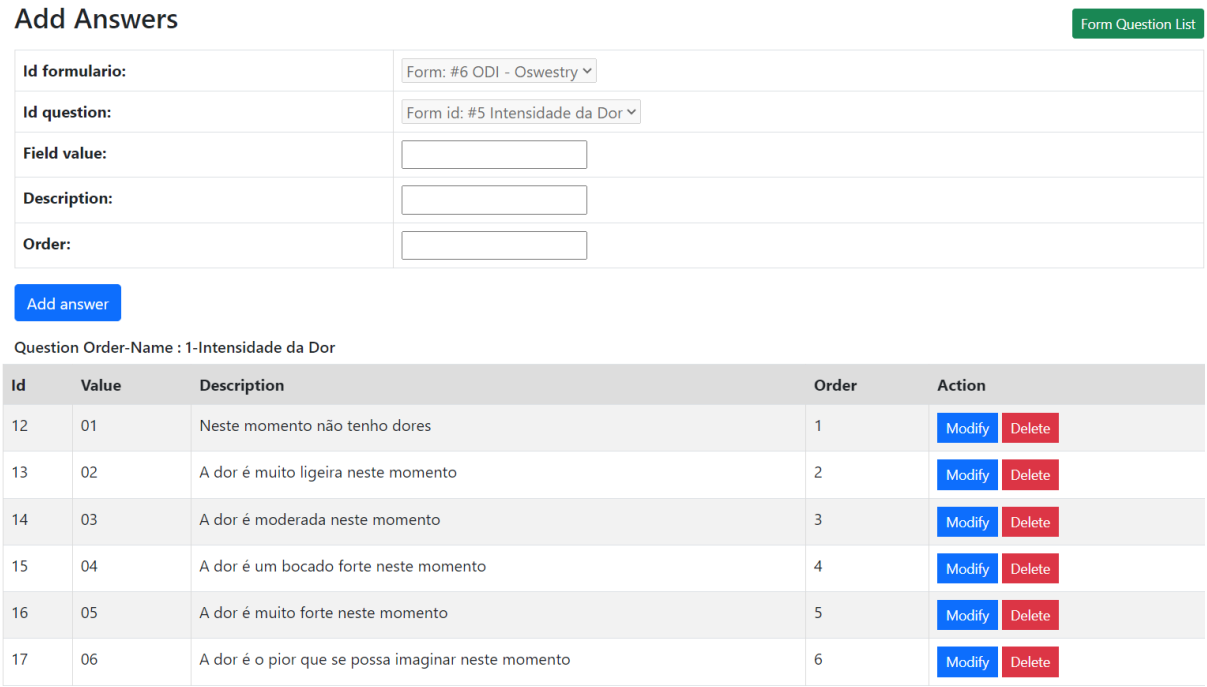


Figura 35. Ecrã cadastro de choices.

O ecrã de extração dos dados, também teve de ser alterado para visualizar o atributo “value” e não mais a descrição, conforme a figura 36, as colunas “cuidadospessoais” e “intensidadedador” possuem os valores cadastrados do campo “value” e não mais as descrições.

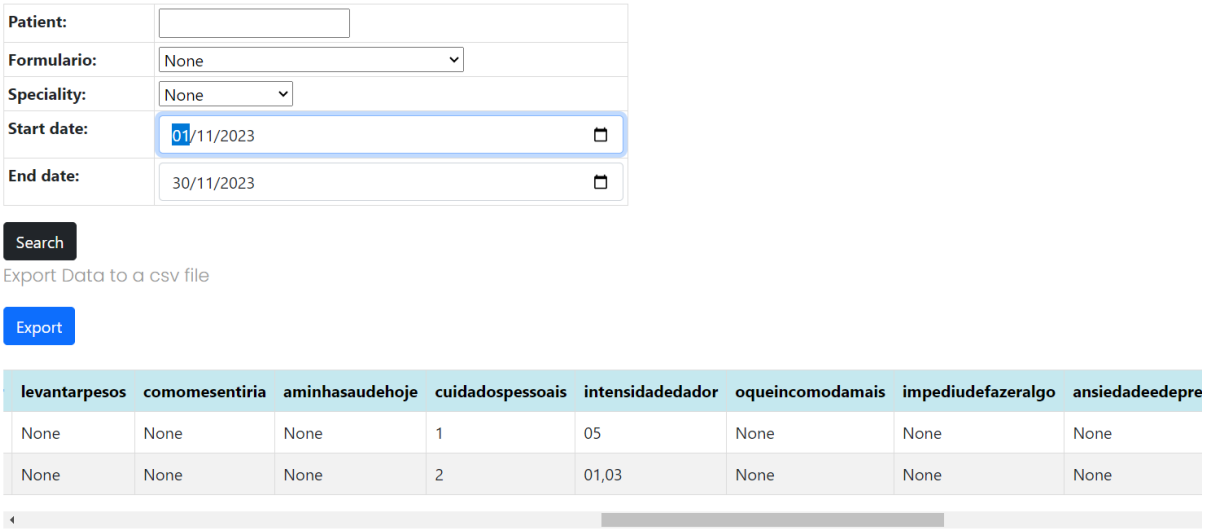


Figura 36. Ecrã de extração dos dados.

Os dois ecrãs de preenchimento de formulário e curadoria dos dados, também tiveram de ser alterados, para visualizar a descrição das “choices” e visualizar as opções seguindo a ordem, (campo “order”), que foi cadastrada, conforme aparece na figura 37.

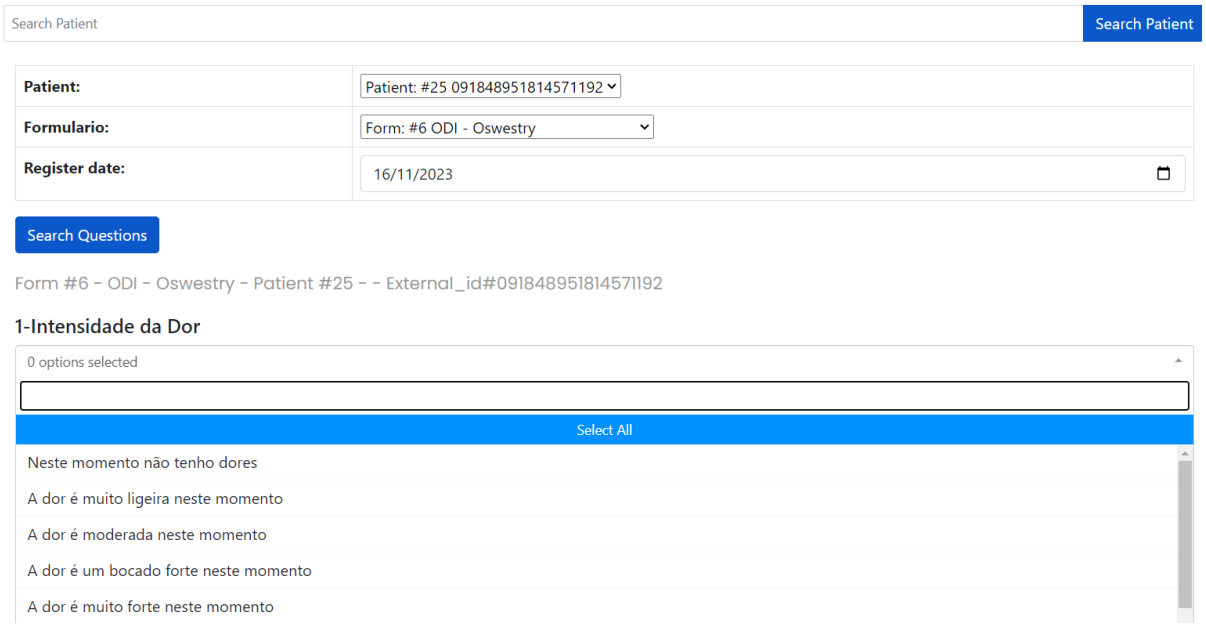


Figura 37. Ecrã de preenchimento de formulário

2) Report by form and patient

Para uma melhor usabilidade e para que seja mais fácil a identificação do paciente, conforme foi identificado no documento de resultado dos testes, houve a necessidade de se colocar o campo NHC no filtro do ecrã “Report by form and patient”. Este ecrã tem como objetivo a consulta das respostas preenchidas de um determinado paciente e formulário. Conforme figura 38, podemos

ver que o utilizador pode utilizar tanto o campo “*patient id”* quanto o campo NHC para a pesquisa.

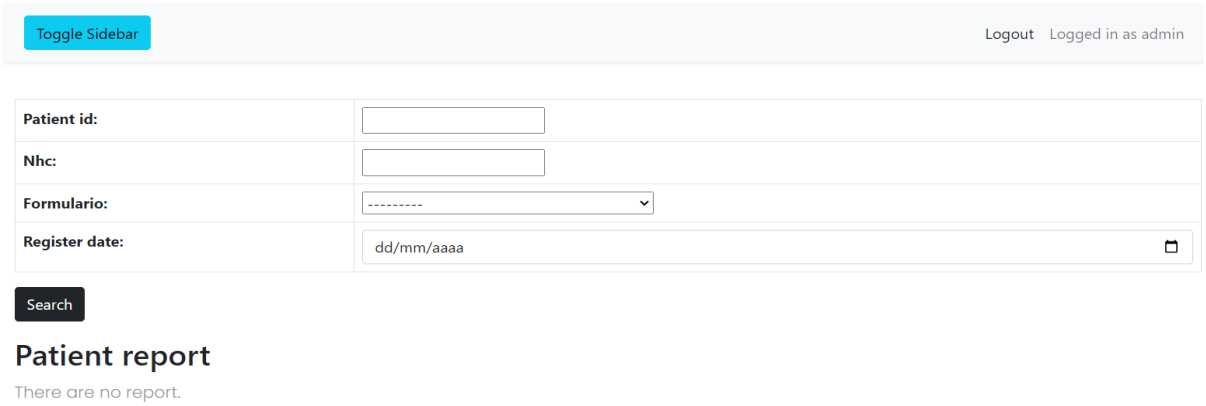


Figura 38. Ecrã de visualização de respostas por paciente e formulário.

3) Menu

O menu sofreu alterações após os testes. Identificamos que apenas dois ecrãs estarão disponíveis para o utilizador mais básico, ou seja, aquele que apenas terá permissões de preenchimento do formulário. Conforme figura 39, o menu “formPatient” contem apenas os dois ecrãs que estarão disponíveis para os utilizadores com permissões básicas e o menu “Administrator” contem todos os ecrãs disponíveis para utilizadores com permissões de administradores da plataforma. Neste caso, conforme foi pedido no documento de testes, o ecrã “question and answers” passa a se chamar “Data Accuracy”.

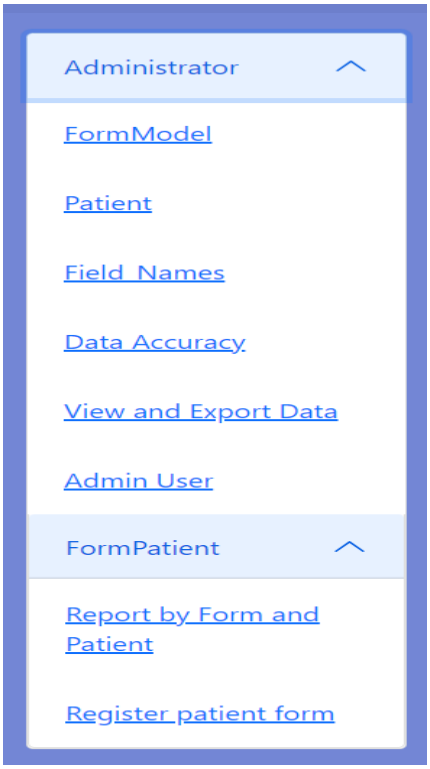


Figura 39. Menu dinâmico

4) Register patient form

O ecrã de registo de respostas do paciente para um formulário cadastrado, é um dos mais importantes da plataforma, pois tanto o enfermeiro, como o paciente, poderão utilizá-lo para responder as questões. Conforme identificado nos testes, existe o caso do NHC não existir na base de dados da plataforma. Neste caso, tivemos de criar uma validação na base e de seguida oferecer ao utilizador a opção de se criar o paciente na base de dados, caso não existir, conforme acontece na figura 40.

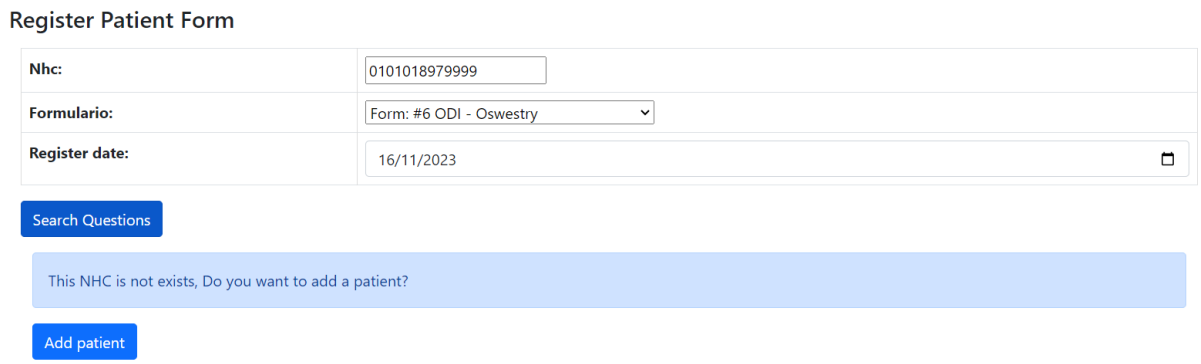


Figura 40. Ecrã de preenchimento do formulário do paciente.

Caso o utilizador clique no botão “Add patient” da figura 41, aparecerá um ecrã menor, onde o utilizador poderá preencher os dados de data de nascimento, (date birth), NHC e por fim incluir o paciente na base de dados da plataforma, conforme figura 41. Após a criação do paciente, o utilizador é direcionado para o ecrã anterior, permitindo assim preencher o formulário com o novo paciente. Para este desenvolvimento foi necessário a utilização de recurso modal do Bootstrap 5 [63].

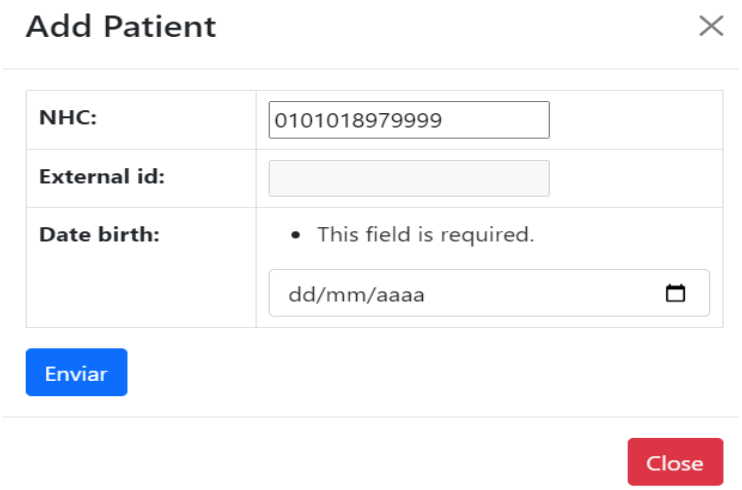


Figura 41. Ecrã de cadastro de pacientes, via preenchimento de formulário.

5) Questions

Conforme foi descrito no documento de testes, o médico responsável pelo teste identificou o problema de cadastrar mais de uma questão com o mesmo “field name” num mesmo formulário e o problema de conseguir cadastrar questões com uma mesma ordenação. Conforme figura 42, agora, temos uma mensagem de erro, que informa ao utilizador que a questão não pode ser adicionada ao formulário pois existe uma outra questão contendo o mesmo “fiel name”, neste mesmo formulário, *“There is another question with this same field name. Question not added.”* A nova validação também ocorre quando colocamos no campo “answer order”, um valor que já existe em outra questão no mesmo formulário.

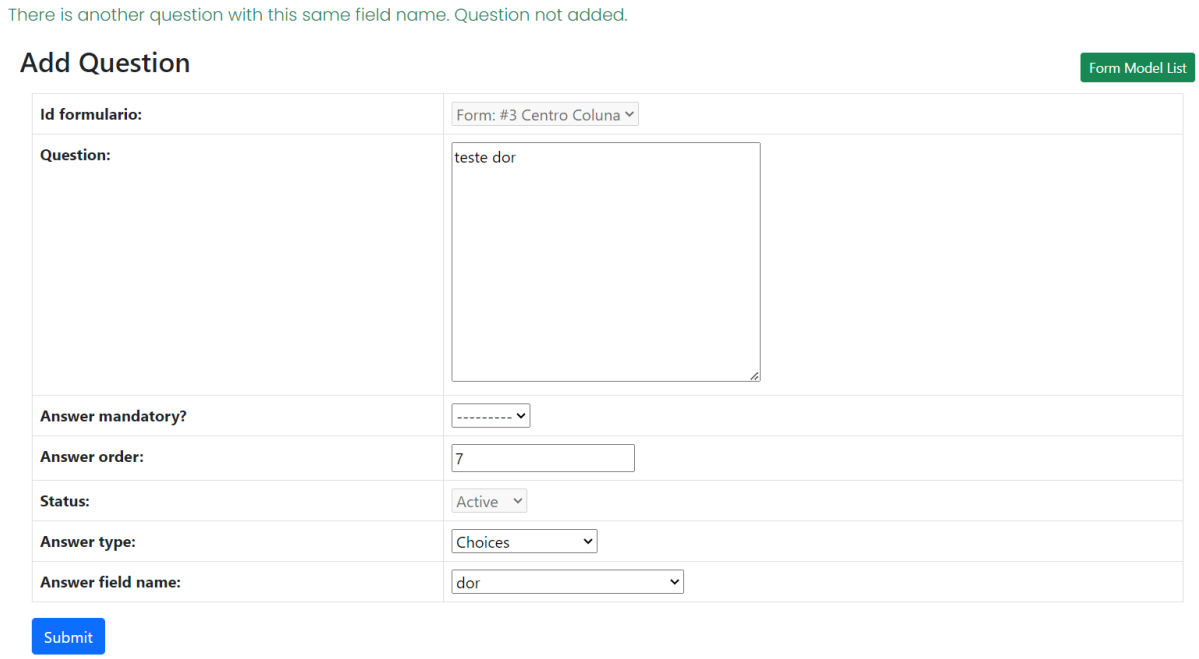


Figura 42. Ecrã de cadastro de questões, validações.

6) Melhoria de performance na busca e decriptação do NHC

O NHC sempre é gravado na base de dados encriptado, assim dificultando a busca do paciente pelo código do NHC, uma vez que o utilizador pode realizar essa busca e que ficou evidenciado nos testes que esse atributo será frequentemente utilizado nas buscas da plataforma pelos investigadores.

A fim de solucionar esse problema identificado nos testes, foi necessária a criação de uma função que busca todos os dados da tabela *patient* e de seguida realiza uma busca para identificar o registo relacionado ao NHC digitado pelo utilizador. A biblioteca **re**[64] do python foi utilizada a fim de se ter maior rapidez na comparação entre o NHC decriptado e o NHC digitado pelo utilizador. Utilizando a função **match** da biblioteca “**re”** do *Python*, foi possível melhorar a performance de busca. A biblioteca “**re”** utiliza o **regex**, (Regular Expression), para realizar a busca na base de dados.

4.3 Resultados e Discussão

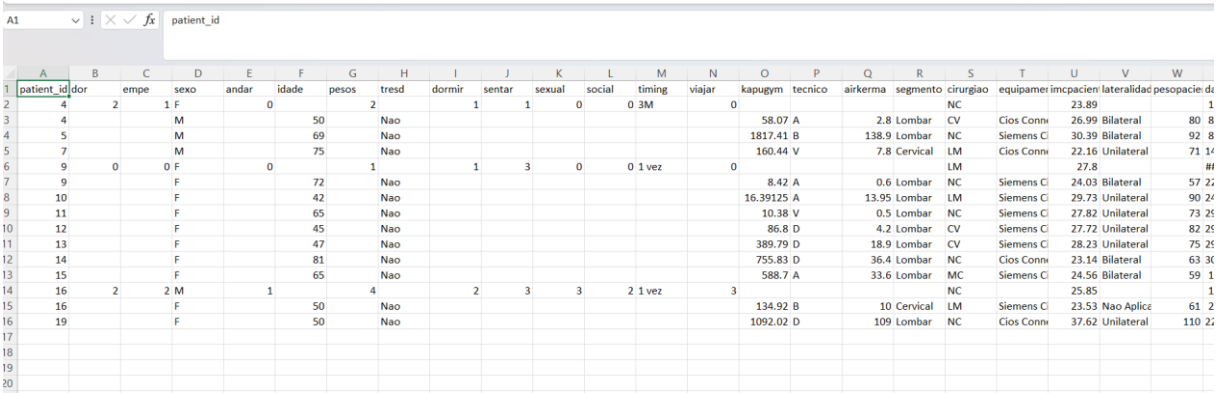
Verificou-se que o primeiro protótipo de software evolutivo desenvolvido (primeiro protótipo da plataforma) já é capaz de garantir os processos de centralização dos dados no paciente e a curadoria dos mesmos. Também permite a realização de consultas e analise do histórico de dados respondidos de um determinado paciente, validando-se a possibilidade de pesquisar por data, formulário ou uma especialidade médica especifica.

Constatou-se que a medida que ocorre o preenchimento dos formulários, posteriormente, o medico tem a possibilidade de analisar os resultados e de corrigir as respostas, quando necessário.

A possibilidade de selecionar de forma dinâmica os atributos de dados, (campos / colunas), é um ponto chave entre os resultados esperados com a implementação desta plataforma, o que permite a partilha / extração massiva de dados dos pacientes pelas diferentes especialidades médicas, melhorando assim as capacidades de analise de dados pelos investigadores de diversas áreas da saúde, que terão a possibilidade de criar “ad-hoc” datasets de benchmarking.

Outro contributo importante é a facilidade em relação a pesquisa de informação / dados dos casos de pacientes por período e especialidade, assim obtendo apenas os atributos específicos relacionados a uma área de saúde / especialidade médica. Um primeiro teste / abordagem satisfatório foi realizado em caso de estudo realizado na especialidade de neurocirurgia.

Por fim, destaca-se o modelo de dados criado, que facilita a criação de formulários dinâmicos uniformizados, que resolve o problema das lacunas de dados que anteriormente eram criadas, pelo facto que cada médico / investigador criava o seu próprio silo, (arquivo Excel), muitas vezes com informação repetida entre os diferentes estudos, que ainda apresentava o problema de dados insuficientes para uma investigação aprofundada / estatisticamente representativa. Um processo que foi automatizado neste projeto, com a vantagem de poder exportar os dados visualizados em formato CSV, (colunas separadas por ponto e virgula), ou Excel, conforme figura 43:



patient_id	dor	empe	sexo	andar	idade	pesos	tresd	dormir	sentar	sexual	social	timing	viajar	kapugym	tecnico	airkerma	segmento	cirurgiao	equipamer	imcpacien	lateralidad	pesopacie	da
4	2	1	F	0		2		1	1	0	0	3M	0					NC		23.89			1
4			M		50		Nao							58.07	A	2.8	Lombar	CV	Cios Conn	26.99	Bilateral	80	8
5			M		69		Nao							1817.41	B	138.9	Lombar	NC	Siemens Ci	30.39	Bilateral	92	8
7			M		75		Nao							160.44	V	7.8	Cervical	LM	Cios Conn	22.16	Unilateral	71	1
9	0	0	F	0		1		1	3	0	0	1 vez	0					LM		27.8			#
9			F		72		Nao							8.42	A	0.6	Lombar	NC	Siemens Ci	24.03	Bilateral	57	2
10			F		42		Nao							16.39125	A	13.95	Lombar	LM	Siemens Ci	29.73	Unilateral	90	2
11			F		65		Nao							10.38	V	0.5	Lombar	NC	Siemens Ci	27.82	Unilateral	73	2
12			F		45		Nao							86.8	D	4.2	Lombar	CV	Siemens Ci	27.72	Unilateral	82	2
13			F		47		Nao							389.79	D	18.9	Lombar	CV	Siemens Ci	28.23	Unilateral	75	2
14			F		81		Nao							755.83	D	36.4	Lombar	NC	Cios Conn	23.14	Bilateral	63	3
15			F		65		Nao							588.7	A	33.6	Lombar	MC	Siemens Ci	24.56	Bilateral	59	1
16	2	2	M	1		4		2	3	3	2	1 vez	3					NC		25.85			1
16			F		50		Nao							134.92	B	10	Cervical	LM	Siemens Ci	23.53	Nao Aplica	61	2
19			F		50		Nao							1092.02	D	109	Lombar	NC	Cios Conn	37.62	Unilateral	110	2

Figura 43. Ficheiro exportado do ecrã de visualização e exportação de dados.

A plataforma dinâmica encontra-se ainda em uma fase de desenvolvimento (protótipo de software evolutivo), pelo que serão necessários novos testes, a partir de agora em um ambiente multiusuário,

pelo que ao longo de sua utilização poderão vir mais customizações para a sua melhoria.

4.4 Hardware e Software utilizados na implantação

Código fonte disponível no GitHub, não público: https://github.com/yashes20/proj_plat

Hardware utilizado na instalação

Mínimo de 16 gb de memória RAM

Mínimo de 500 GB storage

Intel core i5 - 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz

System Type: 64-bit operating system, x64-based processor

Software Requirements

Python 3.11.2

asgiref==3.6.0

cffi==1.15.1

crispy-bootstrap4==2022.1

cryptography==39.0.1

dj-database-url==0.4.2

Django==4.2.5

django-crispy-forms==2.0

django-filter==22.1

djangorestframework==3.14.0

gunicorn==21.2.0

Markdown==3.4.1

numpy==1.24.3

packaging==23.1

pandas==2.0.1

prettytable==3.7.0

psycopg2==2.9.5

pycparser==2.21

python-dateutil==2.8.2

pytz==2022.7.1

six==1.16.0

sqlparse==0.4.3

typing_extensions==4.8.0

tzdata==2022.7

waitress==2.1.2

wcwidth==0.2.6

whitenoise==6.5.0

postgresql-42.6.0

Requisitos de Hardware para implantação no Hospital

Servidor: 16 CPU 64 GB RAM

Mínimo de 500 GB storage

Capítulo 5

Conclusões e Trabalhos Futuros

Para suportar os avanços recentes das técnicas de análise de dados na área da saúde, relacionados com a predição de doenças e padronização dos diagnósticos, cada vez mais há a necessidade de automatizar os processos de captura / aquisição de dados, assim como estruturar o modo de acedê-los de forma lógica e rápida. Nesta dissertação se demonstra, como um primeiro contributo, que a combinação apropriada de *Frameworks* para desenvolvimento de aplicações informáticas, (estrutura-base que contém um conjunto de funções e componentes pré-definidos), e a linguagem SQL com o componente da base de dados relacional, constituem ferramentas de suporte importante para o pré processamento e análise de dados, (ciência de dados). Estas ferramentas facilitam a criação de arquiteturas de software / soluções que permitem a captura e unificação de dados para diversos fins, assim como tarefas relacionadas a curadoria e extração dos dados de saúde, garantindo também uma maior segurança da informação do paciente, o qual é atualmente um desafio. Outra, foi a possibilidade de reutilização de estruturas de código, uma grande vantagem dos *Frameworks* disponíveis.

Foi verificado que a base de dados relacional, assim como o conhecimento da linguagem SQL é de extrema valia nas construções de soluções de tecnologia, promovendo alta performance e facilidade ao trabalhar com diferentes tipos de dados.

Como principal contribuição / resultado desta tese, foi desenvolvida uma plataforma dinâmica com base num modelo de dados relacional, que suporta formulários de saúde dinâmicos, que proporciona aos investigadores e médicos da área da saúde, um Framework automático de recolha, limpeza e curadoria dos dados, resultando em extrações de dados de saúde de benchmarking e prontos para a análise. A plataforma dinâmica engloba em uma só solução web as fases de inserção, limpeza, validações e curadoria dos dados técnicos. Um completo modelo de dados criado com o intuito de dinamizar formulários de saúde e traduzi-los ao entendimento dos investigadores e da ciência de dados. Uma contribuição ao meio científico da saúde que tanto necessita de ferramentas ágeis da tecnologia e que englobem grandes dados para análises.

Com suporte na nova plataforma dinâmica desenvolvida, será possível realizar analise de dados de diferentes vertentes (Bigdata), considerando que a base de dados estará unificada e com os dados de saúde dos pacientes validados, integrados e coerentes com os diferentes tipos de especialidades médicas.

Evolução da plataforma, trabalhos futuros

Com o objetivo de que a plataforma dinâmica consiga evoluir e abranger ainda mais os diferentes tipos de dados da área da saúde, temos em mente de criar as seguintes customizações:

1. Adicionar a plataforma a possibilidade de leitura e armazenamento de imagens de exames da área da saúde, como radiografias, mamografias e ressonância magnética. Assim a plataforma também abrangerá a analise de imagens.

2. Validações de inputs de dados numéricos, por exemplo.
3. Fluxo de avaliação dos diferentes inquéritos produzidos.
4. Melhoria do “front-end” da aplicação, para que seja mais “user friendly”.

Bibliografia

[1] T. A. Koleck, C. Dreisbach, P. E. Bourne, and S. Bakken, “Natural language processing of symptoms documented in free-text narratives of electronic health records: A systematic review,” *J. Am. Med. Informatics Assoc.*, vol. 26, no. 4, pp. 364–379, 2019, doi: 10.1093/jamia/ocy173.

[2] E. Ford, J. A. Carroll, H. E. Smith, D. Scott, and J. A. Cassell, “Extracting information from the text of electronic medical records to improve case detection: A systematic review,” *J. Am. Med. Informatics Assoc.*, vol. 23, no. 5, pp. 1007–1015, 2016, doi: 10.1093/jamia/ocv180.

[3] K. Y. Ngiam and I. W. Khor, “Big data and machine learning algorithms for health-care delivery,” *Lancet Oncol.*, vol. 20, no. 5, pp. e262–e273, 2019, doi: 10.1016/S1470-2045(19)30149-4.

[4] C. A. McGinn *et al.*, “Comparison of user groups’ perspectives of barriers and facilitators to implementing electronic health records: A systematic review,” *BMC Med.*, vol. 9, 2011, doi: 10.1186/1741-7015-9-46.

[5] A. Amirahmadi, M. Ohlsson, and K. Etminani, “Deep learning prediction models based on EHR trajectories: A systematic review,” *J. Biomed. Inform.*, vol. 144, no. October 2022, p. 104430, 2023, doi: 10.1016/j.jbi.2023.104430.

[6] O. Dziadkowiec, T. Callahan, M. Ozkaynak, B. Reeder, and J. Welton, “Using a Data Quality Framework to Clean Data Extracted from the Electronic Health Record: A Case Study.,” *eGEMs (Generating Evid. Methods to Improv. patient outcomes)*, vol. 4, no. 1, p. 11, 2016, doi: 10.13063/2327-9214.1201.

[7] L. Adlung, Y. Cohen, U. Mor, and E. Elinav, “Machine learning in clinical decision making,” *Med*, vol. 2, no. 6, pp. 642–665, 2021, doi: 10.1016/j.medj.2021.04.006.

[8] M. Batta, “Machine Learning Algorithms - A Review,” *Int. J. Sci. Res.*, vol. 18, no. 8, pp. 381–386, 2018, doi: 10.21275/ART20203995.

[9] D. F. Van Rossum G, *Python 3 Reference Manual. Create Space*. 2009. [Online]. Available: https://docs.python.org/3.2/reference/

[10] X. Shi, C. Prins, G. Van Pottelbergh, P. Mamouris, B. Vaes, and B. De Moor, “An automated data cleaning method for Electronic Health Records by incorporating clinical knowledge,” *BMC Med. Inform. Decis. Mak.*, vol. 21, no. 1, pp. 1–10, 2021, doi: 10.1186/s12911-021-01630-7.

[11] N. G. Weiskopf and C. Weng, “Methods and dimensions of electronic health record data quality assessment: Enabling reuse for clinical research,” *J. Am. Med. Informatics Assoc.*, vol. 20, no. 1, pp. 144–151, 2013, doi: 10.1136/amiajnl-2011-000681.

[12] S. H. F. Uller, J. A. B. Rinkley, and P. E. T. A. Ornoch, “Issues in Biomedical Research Data

Management and Analysis : Needs and Barriers," pp. 478–488, doi: 10.1197/jamia.M2114.Introduction.

[13] M. J. Diño *et al.*, "Understanding healthcare providers' electronic health record (EHR) interface preferences via conjoint analysis," *Int. J. Med. Inform.*, vol. 174, no. July 2022, 2023, doi: 10.1016/j.ijmedinf.2023.105060.

[14] R. Subramanyam and M. Xia, "Free/Libre Open Source Software development in developing and developed countries: A conceptual framework with an exploratory study," *Decis. Support Syst.*, vol. 46, no. 1, pp. 173–186, 2008, doi: 10.1016/j.dss.2008.06.006.

[15] B. Shneiderman, "Response Time and Display Rate in Human Performance with Computers," *ACM Comput. Surv.*, vol. 16, no. 3, pp. 265–285, 1984, doi: 10.1145/2514.2517.

[16] R. E. Sherman *et al.*, "Real-World Evidence — What Is It and What Can It Tell Us?," *N. Engl. J. Med.*, vol. 375, no. 23, pp. 2293–2297, 2016, doi: 10.1056/nejmsb1609216.

[17] J. Wise *et al.*, "Implementation and relevance of FAIR data principles in biopharmaceutical R&D," *Drug Discov. Today*, vol. 24, no. 4, pp. 933–938, 2019, doi: 10.1016/j.drudis.2019.01.008.

[18] D. Kotecha *et al.*, "CODE-EHR best-practice framework for the use of structured electronic health-care records in clinical research," *Lancet Digit. Heal.*, vol. 4, no. 10, pp. e757–e764, 2022, doi: 10.1016/S2589-7500(22)00151-0.

[19] Z. Miao, M. D. Sealey, S. Sathyanarayanan, D. Delen, L. Zhu, and S. Shepherd, "A data preparation framework for cleaning electronic health records and assessing cleaning outcomes for secondary analysis," *Inf. Syst.*, vol. 111, p. 102130, 2023, doi: 10.1016/j.is.2022.102130.

[20] S. Tang, P. Davarmanesh, Y. Song, D. Koutra, M. W. Sjoding, and J. Wiens, "Democratizing EHR analyses with FIDDLE: A flexible data-driven preprocessing pipeline for structured clinical data," *J. Am. Med. Informatics Assoc.*, vol. 27, no. 12, pp. 1921–1934, 2020, doi: 10.1093/jamia/ocaa139.

[21] D. Morquin, R. Ologeanu-Taddei, G. Paré, and G. Wagner, "A method for resolving organisation-enterprise system misfits: An action research study in a pluralistic organisation," *Inf. Syst. J.*, no. January 2021, pp. 1–34, 2023, doi: 10.1111/isj.12433.

[22] L. Ben, I. Lahyani, and M. Jmaiel, "Computers in Industry Data accuracy aware mobile healthcare applications," *Comput. Ind.*, vol. 97, pp. 54–66, 2018, doi: 10.1016/j.compind.2018.01.020.

[23] A. Delmestri and D. Prieto-Alhambra, "Curator – A data curation tool for clinical real-world evidence," *Informatics Med. Unlocked*, vol. 40, no. March, 2023, doi: 10.1016/j.imu.2023.101291.

[24] T. K. J. Groenhof *et al.*, "Data mining information from electronic health records produced high yield and accuracy for current smoking status," *J. Clin. Epidemiol.*, vol. 118, pp. 100–106,

2020, doi: 10.1016/j.jclinepi.2019.11.006.

[25] V. Gupta, S. Sachdeva, and S. Bhalla, “A Novel Deep Similarity Learning Approach to Electronic Health Records Data,” *IEEE Access*, vol. 8, pp. 209278–209295, 2020, doi: 10.1109/ACCESS.2020.3037710.

[26] O. Trigueros, A. Blanco, N. Lebeña, A. Casillas, and A. Pérez, “Explainable ICD multi-label classification of EHRs in Spanish with convolutional attention,” *Int. J. Med. Inform.*, vol. 157, no. October 2021, 2022, doi: 10.1016/j.ijmedinf.2021.104615.

[27] Z. Zhang, J. Liu, and N. Razavian, “BERT-XML: Large Scale Automated ICD Coding Using BERT Pretraining,” pp. 24–34, 2020, doi: 10.18653/v1/2020.clinicalnlp-1.3.

[28] Y. Roh, G. Heo, and S. E. Whang, “A Survey on Data Collection for Machine Learning: A Big Data-AI Integration Perspective,” *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1328–1347, 2021, doi: 10.1109/TKDE.2019.2946162.

[29] L. Yin, Z. Cao, K. Wang, J. Tian, X. Yang, and J. Zhang, “A review of the application of machine learning in molecular imaging,” *Ann. Transl. Med.*, vol. 9, no. 9, pp. 825–825, 2021, doi: 10.21037/atm-20-5877.

[30] K. Maharana, S. Mondal, and B. Nemade, “A review: Data pre-processing and data augmentation techniques,” *Glob. Transitions Proc.*, vol. 3, no. 1, pp. 91–99, 2022, doi: 10.1016/j.gltp.2022.04.020.

[31] D. C. Schoen, “Data analysis and interpretation.,” *Orthop. Nurs.*, vol. 8, no. 5, p. 58, 1989, doi: 10.5005/jp/books/12738_11.

[32] “Regulamento Geral sobre a proteção de dados (RGPD).” https://eur-lex.europa.eu/PT/legal-content/summary/general-data-protection-regulation-gdpr.html (accessed Dec. 16, 2023).

[33] M. Javaid and A. Haleem, “Industry 4.0 applications in medical field: A brief review,” *Curr. Med. Res. Pract.*, vol. 9, no. 3, pp. 102–109, 2019, doi: 10.1016/j.cmrp.2019.04.001.

[34] X. Larrucea, M. Moffie, S. Asaf, and I. Santamaria, “Towards a GDPR compliant way to secure European cross border Healthcare Industry 4.0,” *Comput. Stand. Interfaces*, vol. 69, no. December 2019, 2020, doi: 10.1016/j.csi.2019.103408.

[35] D. Water Framework Directive, “60/EC of the European Parliament and of the Council,” *Official Journal of the European Communities L*, 2000. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

[36] F. J. García-León, R. Villegas-Portero, J. A. Goicoechea-Salazar, D. Muñoyerro-Muñiz, and J. Dopazo, “Impact assessment on data protection in research projects,” *Gac. Sanit.*, vol. 34, no. 5, pp. 521–523, 2020, doi: 10.1016/j.gaceta.2019.10.006.

[37] J. L. Oliveira, A. Trifan, and L. A. Bastião Silva, “EMIF Catalogue: A collaborative platform for sharing and reusing biomedical data,” *Int. J. Med. Inform.*, vol. 126, no. February, pp. 35–45, 2019, doi: 10.1016/j.ijmedinf.2019.02.006.

[38] L. Bastião Silva, A. Trifan, and J. Luís Oliveira, “MONTRA: An agile architecture for data publishing and discovery,” *Comput. Methods Programs Biomed.*, vol. 160, pp. 33–42, 2018, doi: 10.1016/j.cmpb.2018.03.024.

[39] J. Zheng *et al.*, “Django: Bilateral coflow scheduling with predictive concurrent connections,” *J. Parallel Distrib. Comput.*, vol. 152, pp. 45–56, 2021, doi: 10.1016/j.jpdc.2021.01.006.

[40] C. Santiago, N. Veras, A. de Aragão, D. Carvalho, and L. Amaral, “Desenvolvimento de sistemas Web orientado a reuso com Python, Django e Bootstrap,” *Minicursos da ERCEMAPI 2020*, pp. 97–120, 2020, doi: 10.5753/sbc.11.5.5.

[41] Django-rest-framework, “django-rest-framework.” https://www.django-rest-framework.org/ (accessed Oct. 10, 2023).

[42] ImpedanceMismatch, “impedanceMismatch,” 2023. http://www.agiledata.org/essays/impedanceMismatch.html (accessed Oct. 10, 2023).

[43] D. P. Pop and A. Altar, “Designing an MVC model for rapid web application development,” *Procedia Eng.*, vol. 69, pp. 1172–1179, 2014, doi: 10.1016/j.proeng.2014.03.106.

[44] “geosemfronteiras.org.” https://geosemfronteiras.org/blog/o-que-e-postgresql/ (accessed Oct. 10, 2023).

[45] Postgresql, “postgresql.” https://www.postgresql.org/ (accessed Oct. 10, 2023).

[46] K. B. Wagholikar *et al.*, “Use of automatic SQL generation interface to enhance transparency and validity of health-data analysis,” *Informatics Med. Unlocked*, vol. 31, no. June, p. 100996, 2022, doi: 10.1016/j.imu.2022.100996.

[47] D. Framework, “https://django-portuguese.readthedocs.io/en/1.0/topics/pagination.html.” https://django-portuguese.readthedocs.io/en/1.0/topics/pagination.html (accessed Oct. 10, 2023).

[48] Ajax, “Ajax,” 2023. https://www.devmedia.com.br/o-que-e-o-ajax/6702 (accessed Oct. 10, 2023).

[49] Anonymization, “Anonymization.” https://mostly.ai/blog/data-anonymization-in-python (accessed Nov. 26, 2023).

[50] P. Random, “Random.” https://docs.python.org/3/library/random.html (accessed Nov. 26, 2023).

[51] Python, “Base64.” https://docs.python.org/3/library/base64.html (accessed Nov. 01, 2023).

[52] Django Software Foundation, “Django Software Foundation Public Records | Django,” *2005*, 2005. https://www.djangoproject.com/foundation/records/

[53] static folder Django, “Django static folder.” https://docs.djangoproject.com/en/4.2/howto/static-files/ (accessed Oct. 10, 2023).

[54] P. Composite Type, “postgresql.” https://www.postgresql.org/docs/current/rowtypes.html

(accessed Oct. 10, 2023).

[55] “RCF 7159 JavaScript Object Notation.” https://datatracker.ietf.org/doc/pdf/rfc7159.pdf (accessed Dec. 16, 2023).

[56] P. JSON TYPES, “JSON TYPES, POSGRESQL.” https://www.postgresql.org/docs/current/datatype-json.html (accessed Oct. 10, 2023).

[57] MongoDB, “MongoDB.” https://www.mongodb.com/docs/ (accessed Oct. 10, 2023).

[58] PostgreSQL Tutorial, “PostgreSQL JSON,” *PostgreSQL Tutorial*. https://www.postgresqltutorial.com/postgresql-json/ (accessed Oct. 10, 2023).

[59] P. Dataframe, “Dataframe, Pandas.” https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html (accessed Oct. 10, 2023).

[60] P. Json_normalize, “Pandas, Python.” https://pandas.pydata.org/docs/reference/api/pandas.json_normalize.html (accessed Oct. 10, 2023).

[61] P. Set_index, “Set_index, Pandas.” https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.set_index.html (accessed Oct. 10, 2023).

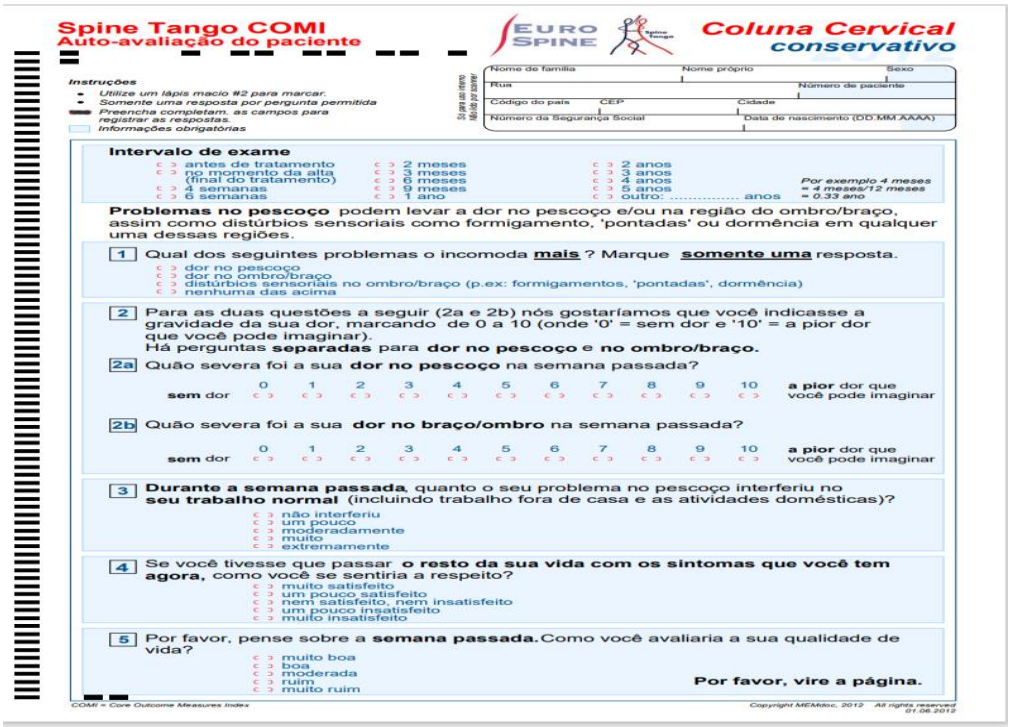
[62] Python, “to_csv Pandas.” https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html (accessed Oct. 10, 2023).

[63] Bootstrap, “Modal · Bootstrap,” 2023. https://getbootstrap.com/docs/4.0/components/modal/ (accessed Oct. 10, 2023).

[64] U. Bhimavarapu and J. D. Hemanth, “Regular Expression,” *Learning Professional Python*, 2023. https://docs.python.org/3/library/re.html (accessed Oct. 10, 2023).

Anexos

1. EXEMPLO DE UM FORMULÁRIO DO HOSPITAL DA LUZ DE SETÚBAL



Spine Tango COMI
Auto-avaliação do paciente

EUROSPINE — Spine Tango

Coluna Cervical
conservativo

Instruções
- *Utilize um lápis macio #2 para marcar.*
- *Somente uma resposta por pergunta permitida*
- *Preencha completam. as campos para registrar as respostas.*
- *Informações obrigatórias*

Só para uso interno não pode ser usado por scanner

Nome de família | Nome próprio | Sexo
Rua | Número de paciente
Código do país | CEP | Cidade
Número da Segurança Social | Data de nascimento (DD.MM.AAAA)

Intervalo de exame

- ○ antes de tratamento
- ○ no momento da alta (final do tratamento)
- ○ 4 semanas
- ○ 6 semanas
- ○ 2 meses
- ○ 3 meses
- ○ 6 meses
- ○ 9 meses
- ○ 1 ano
- ○ 2 anos
- ○ 3 anos
- ○ 4 anos
- ○ 5 anos
- ○ outro: ..... anos

Por exemplo 4 meses = 4 meses/12 meses = 0.33 ano

Problemas no pescoço podem levar a dor no pescoço e/ou na região do ombro/braço, assim como distúrbios sensoriais como formigamento, 'pontadas' ou dormência em qualquer uma dessas regiões.

1 Qual dos seguintes problemas o incomoda **mais** ? Marque **somente uma** resposta.
- ○ dor no pescoço
- ○ dor no ombro/braço
- ○ distúrbios sensoriais no ombro/braço (p.ex: formigamentos, 'pontadas', dormência)
- ○ nenhuma das acima

2 Para as duas questões a seguir (2a e 2b) nós gostaríamos que você indicasse a gravidade da sua dor, marcando de 0 a 10 (onde '0' = sem dor e '10' = a pior dor que você pode imaginar).
Há perguntas **separadas** para **dor no pescoço** e **no ombro/braço.**

2a Quão severa foi a sua **dor no pescoço** na semana passada?

sem dor ○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 **a pior** dor que você pode imaginar

2b Quão severa foi a sua **dor no braço/ombro** na semana passada?

sem dor ○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 **a pior** dor que você pode imaginar

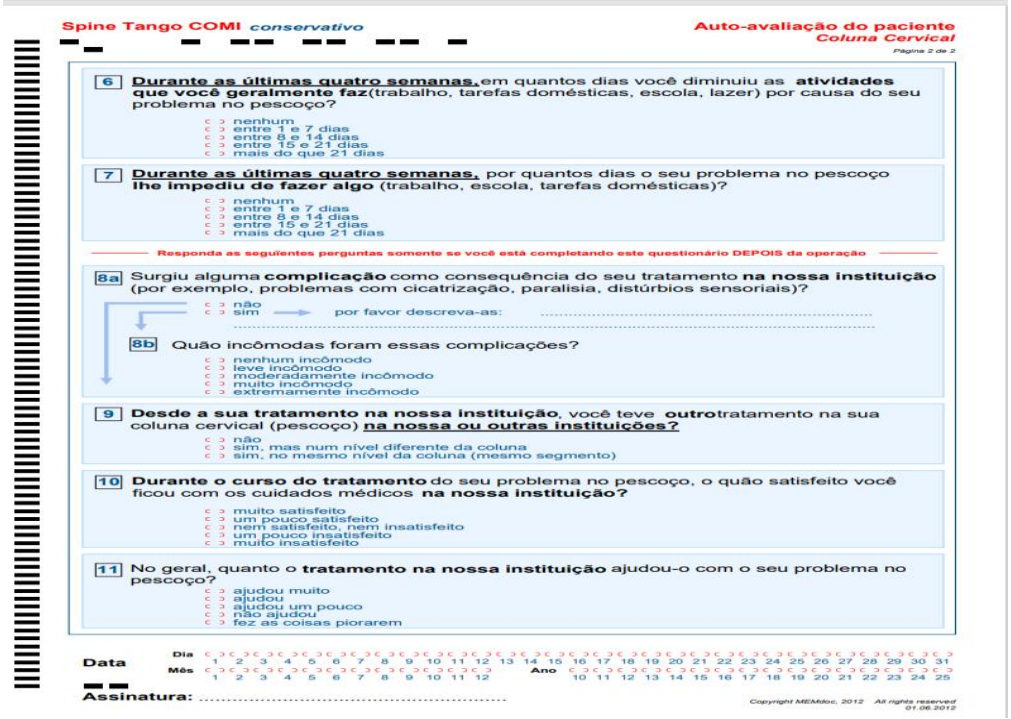
3 **Durante a semana passada**, quanto o seu problema no pescoço interferiu no **seu trabalho normal** (incluindo trabalho fora de casa e as atividades domésticas)?
- ○ não interferiu
- ○ um pouco
- ○ moderadamente
- ○ muito
- ○ extremamente

4 Se você tivesse que passar **o resto da sua vida com os sintomas que você tem agora,** como você se sentiria a respeito?
- ○ muito satisfeito
- ○ um pouco satisfeito
- ○ nem satisfeito, nem insatisfeito
- ○ um pouco insatisfeito
- ○ muito insatisfeito

5 Por favor, pense sobre a **semana passada.** Como você avaliaria a sua qualidade de vida?
- ○ muito boa
- ○ boa
- ○ moderada
- ○ ruim
- ○ muito ruim

Por favor, vire a página.

COMI = Core Outcome Measures Index



Spine Tango COMI ***conservativo***

Auto-avaliação do paciente
Coluna Cervical

6 **Durante as últimas quatro semanas,** em quantos dias você diminuiu as **atividades que você geralmente faz** (trabalho, tarefas domésticas, escola, lazer) por causa do seu problema no pescoço?
- ○ nenhum
- ○ entre 1 e 7 dias
- ○ entre 8 e 14 dias
- ○ entre 15 e 21 dias
- ○ mais do que 21 dias

7 **Durante as últimas quatro semanas,** por quantos dias o seu problema no pescoço **lhe impediu de fazer algo** (trabalho, escola, tarefas domésticas)?
- ○ nenhum
- ○ entre 1 e 7 dias
- ○ entre 8 e 14 dias
- ○ entre 15 e 21 dias
- ○ mais do que 21 dias

Responda as seguintes perguntas somente se você está completando este questionário DEPOIS da operação

8a Surgiu alguma **complicação** como consequência do seu tratamento **na nossa instituição** (por exemplo, problemas com cicatrização, paralisia, distúrbios sensoriais)?
- ○ não
- ○ sim → por favor descreva-as: .....

8b Quão incômodas foram essas complicações?
- ○ nenhum incômodo
- ○ leve incômodo
- ○ moderadamente incômodo
- ○ muito incômodo
- ○ extremamente incômodo

9 **Desde a sua tratamento na nossa instituição**, você teve **outro** tratamento na sua coluna cervical (pescoço) **na nossa ou outras instituições?**
- ○ não
- ○ sim, mas num nível diferente da coluna
- ○ sim, no mesmo nível da coluna (mesmo segmento)

10 **Durante o curso do tratamento** do seu problema no pescoço, o quão satisfeito você ficou com os cuidados médicos **na nossa instituição?**
- ○ muito satisfeito
- ○ um pouco satisfeito
- ○ nem satisfeito, nem insatisfeito
- ○ um pouco insatisfeito
- ○ muito insatisfeito

11 No geral, quanto o **tratamento na nossa instituição** ajudou-o com o seu problema no pescoço?
- ○ ajudou muito
- ○ ajudou
- ○ ajudou um pouco
- ○ não ajudou
- ○ fez as coisas piorarem

Data Dia ○ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Mês ○ 1 2 3 4 5 6 7 8 9 10 11 12 Ano ○ 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Assinatura: .....

2. Código Ajax

```
$.ajax({
                url: "/formPatient/updateFormPatient/",
                type: "POST",
                data: JSON.stringify({data: data,}),
                headers: {"X-CSRFToken": "{{ csrf_token }}"},
                contentType: "application/json; charset=utf-8",
                dataType: "json",
                success: function(response) {
                    alert("Updated successfully ")
                    location.reload()
                },
                error: function(response) {
                    console.log(response);
                    var data=response.responseText;
                    var jsonResponse = JSON.parse(data);
                    alert("An error occurred: " + jsonResponse.messages)
                }
        });
```

3. SQL para o composite type

```
WITH t AS ( select jsonb_agg(ct) as res
            from
    (SELECT t.patient_id, t.date_register, (json_populate_record(null :: composite_type,
json_object_agg(t.label, t.field_answer_value))).*
```