
**João António Mangerico Ribeiro**

# Projeto IPSystems

Revolução da programação no mercado

Dissertação submetida como requisito parcial para obtenção do grau de **Mestre em Engenharia de Software**

**Júri**

*Presidente* (Professor Doutor, Cláudio Sapateiro, IPS)

*Orientador* (Professor Doutor, João Ventura, IPS)

*Arguente* (Professor Doutor, Bruno Silva, IPS)

# Agradecimentos

É com um enorme gosto que expresso a maior gratidão a todas as pessoas que me apoiaram ao longo deste percurso no mestrado, como na tese.

Começar por salientar a ajuda do professor orientador João Ventura por todos os conselhos e apoio prestado em todos os níveis, não só a nível do desenvolvimento, mas do trabalho de pesquisa e a forma correta de competir no mercado com este projeto.

Os membros do júri, neste caso, o professor Cláudio Sapateiro, foi sempre fundamental até à data com todas as suas sugestões e comentários de apoio e o professor arguente Bruno Silva pela a análise atenta e pelas contribuições valiosas que enriqueceram significativamente este trabalho.

Os colegas de pesquisa, com destaque para o colega Guilherme Malhado que partilhou comigo ideias e decisões para o trabalho, assim como algum apoio técnico e opinião prestadas sobre o projeto em si e a apresentação.

A minha família foi sempre um pilar e uma fonte de motivação para o meu sucesso, com destaque para o meu pai que faz todos os esforços para o meu sucesso profissional e a minha namorada que sempre me motivou para alcançar todos os meus objetivos.

# Resumo

Existe uma grande dinâmica no mercado de desenvolvimento de software, onde cada vez mais é necessário otimizar processos. Desta forma, as ferramentas *low-code* apareceram no mercado por serem otimizadas para o desenvolvimento de software e entregas mais rápidas. Existem funcionalidades que com apenas uma função de arrastar, permite ao utilizador visualizar de forma prática a construção da aplicação a partir de blocos de código que são gerados por trás que permitem definir as funcionalidades desejadas.

Este protótipo surge para colmatar essa necessidade, permitindo de uma forma amigável aceder a uma versão leve e replicar aplicações simples e criativas. O intuito é ter uma aplicação competitiva no mercado eficaz, gratuita e de desenvolvimento rápido.

Com o IPSystems, o utilizador consegue arrastar alguns ícones visuais e perceber a interação do ecrã, podendo adicionar classes de CSS à sua escolha e defini-las. Pode em seguida publicar essa página e gravá-la e visualizar via browser como utilizador. Desta forma, o utilizador com menor conhecimento de código, consegue desenvolver uma página funcional e robusta.

Comprometi-me a adotar a metodologia ágil Kanban, organizando o trabalho por meio de tópicos representados como tarefas em diferentes etapas, "*To Do*" (a fazer), "*In Progress*" (em progresso) e "*Done*" (concluído). Essa abordagem permitiu uma visualização clara do fluxo de atividades, facilitando o acompanhamento e a gestão eficiente do progresso do projeto.

Durante o desenvolvimento, as tarefas foram organizadas e realizadas com base na sua prioridade, iniciando pelas funcionalidades essenciais, como a criação de recursos fundamentais para o desenvolvimento de projetos e a geração de páginas.

A aplicação permite ao utilizador criar uma conta, iniciar sessão no sistema, desenvolver um projeto e publicá-lo na página dedicada ao seu desenvolvimento. Além disso, existe uma página funcional de cursos disponíveis na plataforma. Essas funcionalidades foram priorizadas como as mais importantes para garantir uma experiência interativa e eficiente para o utilizador.

O IPSystems é uma plataforma voltada para o desenvolvimento *low-code*, permitindo a criação de projetos e a geração de páginas web. Além disso, destaca-se pelo bom desempenho no acesso às suas páginas, proporcionando ao utilizador uma experiência intuitiva e agradável.

**Palavras-chave:** Plataforma *low-code*, versão leve, desenvolvimento ágil, simplificação de código, entrega rápida, eficácia.

# Abstract

There is a great dynamic in the software development market, where it is increasingly necessary to optimize processes and automate them. In this way, low-code tools appeared on the market because they are optimized for software development and faster deliveries. There are features that, with just a drag function, allow the user to practically visualize the construction of the application from blocks of code that are generated behind that allow defining the desired features.

This prototype appears to fill this need, allowing users to access a lightweight version and replicate simple and creative applications in a user-friendly way. The aim is to have a competitive application on the market that is effective, free and rapidly developed.

With IPSystems, the user can drag some visual icons and understand the screen interaction, being able to add CSS classes of their choice and define them. You can then publish that page and save it and view it via browser as a user. This way, users with less code knowledge can develop a functional and robust page.

I committed to adopting the agile Kanban methodology, organizing work through topics represented as tasks in different stages, “To Do”, “In Progress” and “Done” (completed). This approach allowed a clear visualization of the flow of activities, facilitating the monitoring and efficient management of project progress.

During development, tasks were organized and carried out based on their priority, starting with essential functionalities, such as creating fundamental resources for project development and page generation.

The application allows the user to create an account, log in to the system, develop a project and publish it on the page dedicated to its development. In addition, there is a functional page of courses available on the platform. These features were prioritized as the most important to ensure an interactive and efficient experience for the user.

IPSystems is a platform aimed at low-code development, allowing the creation of projects and the generation of web pages. Furthermore, it stands out for its good performance when accessing it’s pages, providing the user with an intuitive and pleasant experience.

**Keywords**: Low-code platform, lightweight version, agile development, code simplification, fast delivery, effectiveness.

# Índice

# Lista de Figuras

# Lista de tabelas



# Lista de Anexos

# Lista de Siglas e Acrónimos

| | |
|---|---|
| *ARIA* | *Accessible Rich Internet Applications* |
| *ATAG* | *Authoring Tool Accessibility Guidelines* |
| *BD* | Base de Dados |
| *CORS* | *Cross-Origin Resource Sharing* |
| *CRUD* | *Create, Read, Update, Delete* |
| *CSS* | *Cascading Style Sheets* |
| *IT* | *Information Technology* |
| *JS* | *JavaScript* |
| *LC* | *Low Code* |
| *NC* | *No Code* |
| *ORM* | *Object-Relational Mapping* |
| *PDF* | *Portable Document Format* |
| *QR Code* | *Quick Response Code* |
| RDIS | Rede Digital com Integração de Serviços |
| *SPA* | *Single Page Application* |
| *WebAIM* | *Web Accessibility In Mind* |
| *WiFi* | *Wireless Fidelity* |

# Capítulo 1 - Introdução

Em seguida será apresentado um enquadramento teórico do IPSystems e de como nasceu esta ideia de apostar no desenvolvimento de um protótipo de aplicação *low-code* com potencial para se tornar competitiva no mercado digital atual. O nome IPSystems, surge a partir de uma combinação que reflete a inspiração a partir de 2 elementos principais. Primeiramente, faz referência ao Instituto Politécnico de Setúbal (IPS), destacando a ligação institucional e o contexto académico em que o projeto foi desenvolvido, em seguida, o nome incorpora o conceito de OutSystems que é a plataforma *low-code* com maior relevância no mercado.

Esta aplicação auxilia o desenvolvimento simples e completo de aplicações web, de forma a participar na transição das necessidades do mercado, mudança essa que tem sido vivenciada por todos os que trabalham na área da tecnologia.

Para além disso, será abordada a motivação que leva o aluno a escolher esta ideia e seguir com a mesma, os objetivos deste projeto e a estrutura do trabalho.

## 1.1. Estado da arte

O desenvolvimento de aplicações mobile e web é cada vez mais comum por parte das empresas, exigindo soluções que agilizem o processo e democratizem o acesso à criação de software. As plataformas de baixo código (*low-code*) e sem código (*no-code*) surgem como resposta a essa procura, permitindo que mesmo utilizadores sem conhecimentos aprofundados de programação criem aplicações funcionais e eficientes. Existem atualmente várias plataformas de *low-code* como por exemplo:

- **OutSystems:** Uma plataforma líder no mercado, com grande comunidade e variedade de recursos. Permite a criação de aplicações omnichannel, com integração a APIs e bases de dados externas, além de extensões em .Net e C# (Outsystems, 2024).
- **Appian:** Focada na computação em cloud e automação de processos de negócio com inteligência artificial. Oferece escalabilidade e flexibilidade, mas não é open source (Appian, 2024).

- **Mendix:** Plataforma de alto desempenho e controlo, com ambiente de desenvolvimento flexível e rápido. Permite desenvolver *frontend* para qualquer *backend*, mas exige conhecimentos de programação específicos (Mendix, 2024).
- **Cronapp:** Plataforma open-source que permite desenvolvimento visual ou codificado, com suporte a BPM e publicação em nuvem ou on-premise (CronApp, 2024).

As plataformas de baixo código e sem código oferecem diversas vantagens, como agilidade no desenvolvimento, democratização da criação de software e redução de custos. No entanto, é importante ter em mente que plataformas mais simples podem ser limitadas para aplicações complexas, sendo o desenvolvimento tradicional com código ainda fundamental para aplicações mais complexas ou que exigem alto nível de personalização.

É importante nos dias de hoje, existir um desenvolvimento rápido, isto porque o mercado e as necessidades de negócio o exigem. Desta forma, o IPSystems é estruturado de maneira a apoiar o utilizador a agilizar algumas dessas tarefas e apresentar resultados mais rápidos, aumentar a mão de obra e assim satisfazer as necessidades do cliente e do mercado. Também é importante notar, que o utilizador, com menor esforço e conhecimento, consegue desenvolver algumas aplicações interessantes, embora a necessidade de saber programar continuará sempre a ser uma mais valia e indispensável para o desenvolvimento.

No entanto, existem também desvantagens, por exemplo, o OutSystems é um produto caro no mercado, com o valor das licenças elevado, podendo o preço variar de acordo com o número de colaboradores a utilizarem a plataforma e de acordo com as funcionalidades em uso. Existem também demasiados recursos usados pela plataforma que dificultam o processo de customização, pois existe já implementações de JavaScript criadas pela plataforma que não podem ser modificadas e obrigam o utilizador a ter ações mais limitadas na fase de implementação, ou seja, o que poderia ser um acelerador, em casos específicos, torna-se uma barreira, pelo facto de exigir que o utilizador invista mais tempo em aplicar uma solução que numa ferramenta com maior necessidade de código seria mais rápida e eficaz essa solução.

No caso da Appian, embora exista uma grande simplificação no processo de desenvolvimento, possui uma grande barreira no UI por não existir um padrão entre as widgets e existe muita complexidade na sua utilização, para além de que grandes conjuntos de dados afetam imenso o desempenho das páginas, não estando otimizada para lidar com grandes modelos de dados.

A plataforma Mendix, embora também seja bastante completa, possui alguns problemas como lentidão quando muitas informações estão a ser tratadas em simultâneo, possui também uma comunidade muito pequena sem tantas informações. O desenvolvimento de UI também é complexo de implementar e não é a melhor opção para se desenvolver projetos que exigem um alto nível de customização visual.

Por fim, o Cronapp tem uma comunidade menor relativamente às outras plataformas no mercado, para além de que tem uma curva de aprendizagem da própria plataforma associada à mesma, o que se pode tornar uma barreira para alguém com menos experiência na plataforma.

### 1.1.1. Análise SWOT

Após uma análise cuidada dos principais concorrentes de mercado e das suas características, é importante fazer um levantamento dos principais pontos fortes e fracos, assim como oportunidades e ameaças, com o objetivo de determinar a melhor forma de estruturar e desenvolver a aplicação. Em seguida, é feita uma análise mais detalhada:

- **Pontos Fortes:** O desenvolvimento rápido é uma característica comum das plataformas *low-code* no mercado, o desenvolvedor consegue de forma rápida, desenvolver um site sem necessitar de instalar bibliotecas e fazer configurações, possuindo componentes otimizados e prontos para arrastar.

  É possível fazer integrações com serviços REST ou SOAP e utilizar as funcionalidades na aplicação de forma mais simples.

  Com a sua fácil implementação, é possível desenvolver um protótipo em pouco tempo sem a necessidade de custos adicionais e sem muito esforço.

- **Pontos Fracos:** O facto de existir muitos componentes preparados para arrastar, já com blocos de código por trás que gerem as funcionalidades, torna-se mais complexo em alguns casos personalizar os componentes. Desta forma, o utilizador pode optar por desenvolver o seu próprio componente com o estilo e funcionalidades desejadas o que pode ser um pouco mais dispendioso a nível de tempo.

  O facto de existirem componentes mais complexos com funcionalidades específicas e que podem ser adaptadas para o utilizador, ao definir as suas propriedades, como por exemplo, um calendário customizado que pode ter a integração dos dados que o utilizador quiser utilizar e disponibilizar a informação de uma forma específica, pode levar a alguns problemas, se essa versão for descontinuada e se for necessário atualizar a mesma e até adicionar novas funcionalidades, será necessário fazer o *upgrade* dos componentes.

- **Oportunidades:** A integração das plataformas *low-code* com inteligência artificial ainda está numa fase inicial, o que poderá ser uma oportunidade em aberto para o futuro.

  As pequenas empresas e não só, podem aproveitar uma oportunidade de usufruir de licenças gratuitas, visto que, o valor das licenças é elevado.

- **Ameaças:** A concorrência tem aumentado ao longo dos últimos anos e as funcionalidades das plataformas têm sido cada vez mais otimizadas e oferecem cada vez mais, melhores soluções.

  Caso exista uma grande curva de aprendizagem da plataforma, pode influenciar a empresa a não optar pela plataforma.

Com base nesta análise, o IPSystems adota uma posição acessível e eficiente, sendo projetada para os utilizadores que procuram soluções práticas e leves, mas sem abrir mão de funcionalidades robustas. A possibilidade de existirem licenças gratuitas, possibilita que o IPSystems esteja mais acessível a todo o tipo de utilizadores e também de empresas. Qualquer utilizador que não possua grandes conhecimentos na área de programação, tem facilidade em produzir um protótipo rápido para servir de mockup para outros projetos mais robustos que podem ser criados em seguida.

## 1.2. Motivação

Durante o percurso no mestrado de Engenharia de Software, encontrei uma oportunidade única para explorar a implementação de código em diversas linguagens de programação, essa oportunidade surgiu em paralelo com a sua atividade profissional, que envolve o desenvolvimento de aplicações utilizando uma ferramenta de desenvolvimento low-code, especificamente o OutSystems.

Inicialmente, deparei-me com algumas dificuldades ao utilizar novas *frameworks*, o que despertou a sua curiosidade e desejo de compreender mais profundamente essas tecnologias. Essas experiências desafiadoras motivaram o embarque num projeto ambicioso: criar uma solução capaz de replicar os bastidores de uma ferramenta de desenvolvimento low-code, que gera automaticamente uma página padrão.

Essa ideia representa não apenas um desafio técnico, mas também uma oportunidade para explorar os fundamentos subjacentes ao desenvolvimento de software. Ao desenvolver uma

solução que simula o funcionamento de uma ferramenta low-code, existe a chance de aprofundar o conhecimento sobre os princípios de desenvolvimento de software, arquitetura de sistemas e automação de processos.

Além disso, essa iniciativa oferece uma oportunidade única para aplicar conceitos teóricos aprendidos durante o mestrado num contexto prático e relevante para a minha carreira profissional, ao enfrentar desafios reais e procurar soluções inovadoras, é possível não apenas a expandir o meu conjunto de habilidades e conhecimento, mas também contribuir para o avanço e a evolução do campo da Engenharia de Software.

## 1.3. Objetivos

Este trabalho foi desenvolvido para automatizar processos na programação e compilar código, o protótipo destaca-se pela capacidade de gerar páginas HTML personalizadas e definir estilos de CSS de forma dinâmica, desta forma, alguém que pretenda criar uma página de destino ou *landing page* e tenha como objetivo falar de um produto digital ou físico, ou até mesmo de ambos, pode realizar essas tarefas de forma prática e eficiente, atendendo às necessidades dos utilizadores menos experientes até aos que possuem mais experiência.

A nível empresarial, poderá ser útil para uma organização, desenvolver produtos de forma ágil e prática mediante as suas necessidades. O IPSystems no futuro tende a tornar-se uma ferramenta mais robusta que oferece ao utilizador uma forma mais prática de desenvolver, fácil de manter e capaz de interagir com qualquer tipo de serviço.

A aposta é feita para uma versão mais leve, diferente dos concorrentes no mercado que oferecem executáveis mais pesados e com várias dependências na construção de projetos, uma melhor experiência de utilizador, ao ser mais prática e simples de utilizar, com uma curva de aprendizagem ainda mais rápida, para que o utilizador se foque mais em desenvolver páginas e menos em aprender a plataforma.

Por fim, o licenciamento gratuito, torna a plataforma acessível para todos os utilizadores, para que qualquer pessoa consiga utilizar a plataforma e poder aprender com a sua utilização, para além de ser uma oportunidade para as empresas, porque o custo das ferramentas é muitas vezes uma barreira significativa para a adoção de novas tecnologias por influenciar no orçamento.

## 1.4. Áreas Estratégicas

A ideia de criar o produto IPSystems surge da tendência do mercado ao adotar metodologias de trabalho ágeis na fase de desenvolvimento de novos produtos, tendo em conta que as necessidades do negócio mudam constantemente. A adoção de ferramentas *low-code* virou moda no mercado e está a ser bastante adotada pelas empresas, sendo que, quanto mais personalizada for a aplicação, maior será a sua necessidade de código. Por isso, de forma estratégica são utilizadas plataformas *low-code* para agilizar todo o processo de desenvolvimento dos produtos.

Podemos ter uma melhor noção desta realidade com base numa notícia do jornal de negócios que garante com base num estudo feito pela KPMG, uma consultora portuguesa, que 64% das empresas inquiridas consideram o *low-code* fundamental para tornar o desenvolvimento tecnológico mais eficiente e confiável e 43% dessas empresas já fazem o uso do *low-code* para o desenvolvimento de aplicações de negócio. (Rui Gonçalves, 2024).

Tendo em conta esta realidade, o IPSystems foi pensado para o desenvolvimento de plataformas mais leves e produção de soluções rápidas, de forma a garantir que a longo prazo, não existam quaisquer restrições no consumo de serviços, no uso de bases de dados e que o utilizador, de forma mais facilitada, consiga desenvolver.

Este estudo serve de expectativa para uma nova realidade virtual, com processos de negócio mais rápido, fluxos mais otimizados, melhoria contínua dos serviços, maior satisfação do negócio ou dos clientes e melhores transações no mercado a longo prazo.

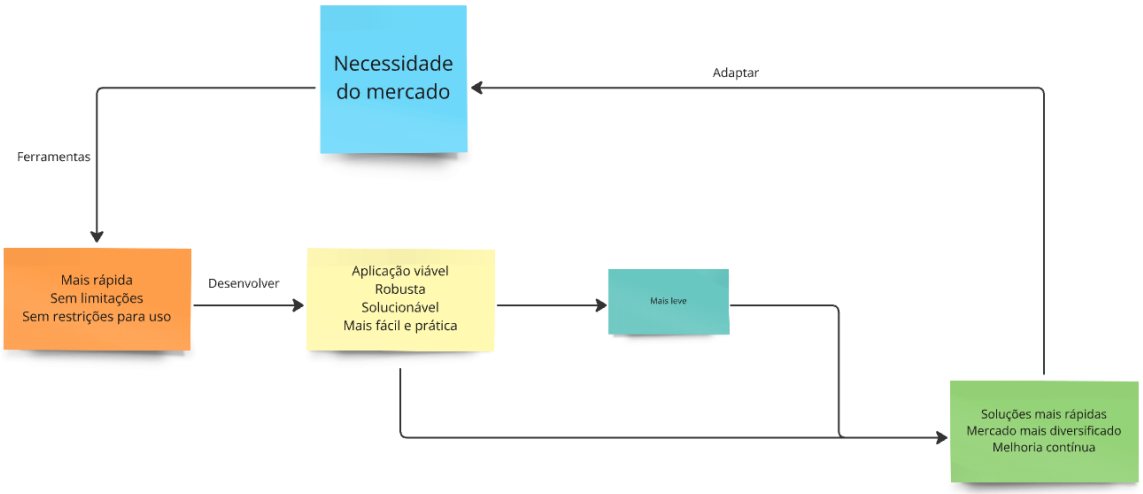



*Figura 1- Esquema da motivação para o projeto e necessidade identificada*

Como é observável no diagrama da figura 1, existe um fluxo de desenvolvimento iterativo que parte da necessidade de suprir lacunas ou de responder às necessidades do mercado, com o objetivo de oferecer soluções relevantes, práticas e inovadoras. A partir desta necessidade, o foco passa pelo desenvolvimento de ferramentas que proporcionam um processo mais rápido, eficiente e acessível, garantindo que o desenvolvimento de aplicações seja mais intuitivo e solucionável.

Estas ferramentas têm o potencial de democratizar o acesso à área de desenvolvimento, atraindo todo o tipo de clientes e utilizadores, incluindo aqueles que não possuem tanto conhecimento na área de programação ou que não se identificam como programadores tradicionais e que prefiram atuar mais numa ótica de automação de tarefas e simplificação de processos.

Por meio destas ferramentas produzidas, é possível dar origem a uma aplicação viável, caracterizada por ser robusta, leve e prática, enquanto soluciona problemas identificados no mercado. Esta abordagem para além de colmatar lacunas já existentes, também contribui para a melhoria contínua no setor, possibilitando a criação de soluções rápidas e escaláveis, adaptadas às procuras de um mercado mais diversificado e em constante evolução.

## 1.5. Estrutura do Relatório

No capítulo 2, é feita uma revisão sobre o que são as plataformas low-code e o que estas oferecem atualmente ao mercado, a acessibilidade e usabilidade nas plataformas, metodologias de trabalho adotadas, com foco na metodologia Ágil e em cascata que são duas das principais metodologias adotadas no mercado e duas abordagens que podem ser projetadas concentradas no uso do IPSystems como ferramenta de trabalho. Ainda no capítulo, são abordadas todas as linguagens que dão suporte à ferramenta IPSystems, desde a linguagem de estrutura HTML e linguagem de estilos CSS, às linguagens de programação de Python e JavaScript. Assim como, as *frameworks* e bibliotecas usadas no desenvolvimento do frontend e do backend da ferramenta, no frontend o React, uma biblioteca de JavaScript de código aberto, criada pelo Facebook em 2011 e a *framework* Django para o desenvolvimento rápido de páginas web ou de serviços de APIs REST.

O capítulo 3 aborda a o planeamento do projeto, a sua estrutura de páginas e os métodos usados desde o início do desenvolvimento do IPSystems, como a metodologia de trabalho mais

ágil, o planeamento base dos primeiros desenhos no figma sobre o projeto e o respetivo esboço, o processo de mudança de algumas funcionalidades, os testes que têm sido aplicados e a análise dos resultados.

É ainda abordado o desenvolvimento do projeto, que começou a ser desenvolvido em Django, algumas páginas e algumas APIs REST e foi depois desenvolvido o frontend em React, que posteriormente integrou as APIs disponibilizadas em Django, por isso, é abordado todo esse processo, assim como a arquitetura adotada em React e no Django.

Para terminar o capítulo 3, são tratadas todas as limitações sentidas no projeto, funcionalidades implementadas que exigiram maior esforço e resiliência no processo de desenvolvimento que foram colmatadas. São ainda abordadas ideias futuras para o IPSystems, que serão uma mais-valia para a plataforma numa fase mais avançada e que deixará a mesma mais robusta.

Por fim, no quarto capítulo são abordados todos os pontos do IPSystems e as suas limitações, assim como, as melhorias futuras que serão adotadas no projeto. Já no capítulo 5, estão as conclusões gerais sobre o projeto e o desenvolvimento do mesmo.

# Capítulo 2 - Revisão de Literatura

## 2.1. Introdução

Neste capítulo, será feita uma abordagem sobre o que são as plataformas *low-code* e em que consistem, as metodologias de desenvolvimento e quais as vantagens e desvantagens de cada tipo, as linguagens de programação e sintaxe que foram utilizadas para o desenvolvimento do projeto IPSystems, assim como as *frameworks* e bibliotecas.

## 2.2. Plataformas *Low-Code*

Como o próprio nome sugere, as plataformas *low-code,* são plataformas que dispensam o uso em massa de código, ou seja, definem-se como programas de desenvolvimento com funcionalidades de arrastar e soltar e que permitem gerar código de forma mais simplificada e rápida, reduzindo o esforço no desenvolvimento desde a implementação do modelo de dados ou criação de classes, à representação gráfica.

Se pensarmos num exemplo muito simples, que é o desenvolvimento de um site ou de uma aplicação, onde é necessário desenvolver uma página estática com linguagem HTML, é necessário realizar algumas configurações para correr localmente no computador uma solução que possua um código de sintaxe HTML para disponibilizar algum conteúdo.

A tarefa do *low-code,* concentra-se em fazer esse processo de forma mais rápida e visual, ao possuir alguns componentes visuais, como o componente de texto que permite ao utilizador arrastar e escrever o título que deseja, tendo uma noção melhor do que está a desenvolver e não precisa de ter um conhecimento muito aprofundado sobre uma linguagem de programação nem de saber a tag de HTML que está a ser utilizada para implementar uma solução deste tipo.

As plataformas *low-code* são compostas por diversos componentes que facilitam o desenvolvimento de software (Alves & Alcalá, 2022):

- **Área de Desenvolvimento Visual:** Permite arrastar e soltar componentes para construir o ecrã da aplicação e definir as interações entre os elementos com facilidade.
- **Módulos de Lógica de Negócios:** Fornecem funcionalidades pré-construídas para processos de negócio, fluxos de trabalho, validações de dados e muito mais, agilizando o desenvolvimento.

- **Gerador de Código:** Permite que o utilizador utilize os elementos necessárias para o desenvolvimento do projeto de forma automática sem necessitar de executar no terminal as configurações, automatizando e simplificando o processo.
- **Integração com Base de Dados:** Permite estabelecer a conexão entre a base de dados e a aplicação que está a ser desenvolvida.
- **Ferramentas de Gestão de Ciclo de Vida de Aplicações:** Auxiliam na gestão completa da aplicação, desde o desenvolvimento até à implementação e manutenção, simplificando o processo.

A principal vantagem das plataformas *low-code* reside na sua **capacidade de acelerar drasticamente o desenvolvimento de software**, com base na interface visual intuitiva e os módulos pré-construídos, é possível desenvolver de forma mais rápida e eficaz as aplicações.

Desta forma, é possível com menos esforço implementar todos os requisitos necessários para o projeto e dedicar mais tempo à lógica de negócio e experiência do utilizador.

A irrelevância da preocupação do utilizador com a sintaxe do código, juntamente com a disponibilidade de configurações pré-estabelecidas para ambientes, bases de dados e elementos de interação visual, como o Bootstrap e estilos de CSS pré-definidos, reduz significativamente o esforço e o tempo na fase de implementação. No backend, a criação de um CRUD (*Create, Read, Update, Delete*) e o mapeamento de dados podem ser realizados de forma muito mais simples e personalizada, atendendo às necessidades do utilizador e aumentando a produtividade ao economizar tempo e recursos.

As plataformas *low-code* facilitam o desenvolvimento rápido de aplicações através de uma arquitetura por camadas. Os principais componentes dessa arquitetura são (Marques, 2023):

- **Camada de Aplicação:** Ambiente gráfico para construção de interfaces gráficas e construção de modelos para especificar o comportamento da aplicação.
- **Camada de Integração de Serviços:** Facilita a comunicação entre diferentes serviços, utilizando APIs e mecanismos de autenticação, integrando vários serviços.
- **Camada de Integração de Dados:** Manipulação e integração de dados a partir de diversas fontes.
- **Camada de Deployment:** Garante o deploy da aplicação automaticamente, implementando-a em cloud ou infraestruturas locais, ao associar todo o código

necessário em contentores com configurações, bibliotecas, dependências e orquestração de aplicações (configuração, gestão e coordenação automatizada de serviços, facilitando a gestão dos fluxos de trabalho).

Contudo, é importante notar que apesar das semelhanças entre as plataformas *low-code* e *NC* (*no-code)*, existem vários detalhes e recursos que diferenciam as plataformas *low-code* das soluções *NC*. Isto, porque, o *low-code* é útil para o desenvolvimento de aplicações e portais independentes tanto no desenvolvimento web, como mobile, e que provavelmente exigirão integrações com outros sistemas e BD's.

As ferramentas *NC*, por outro lado, só devem ser usadas para casos de uso de *frontend*, como o desenvolvimento de uma *SPA (single page application)*, por exemplo, uma página de marketing sobre um produto digital ou físico, que não exigem uma grande complexidade a nível de lógica nem de muitos recursos.

Se o objetivo for desenvolver uma aplicação mais robusta, o *low-code* será sempre uma melhor aposta por permitir definir as BDs e o respetivo modelo de dados e uma lógica de negócio mais complexa.

O IPSystems surge como uma aplicação *low-code* que permite ao utilizador desenvolver uma página com os estilos CSS que deseja, existindo assim por parte do utilizador, a necessidade de compreender algum código.

Portanto, apenas no caso de serem desenvolvidas aplicações mais simples e menos exigentes, é provável que o *low-code* seja a melhor opção. Outra grande diferença é que o *low-code* permite a implementação de algumas linhas de código para ajudar no desenvolvimento da aplicação, contrariamente ao *no-code* (Matos, 2022).

| *Tipo* | *No-Code* | *Low-Code* |
|---|---|---|
| *Complexidade* | Baixa complexidade | Moderada complexidade dependendo do tipo de solução. |
| *Customização* | Limitada | Alta |
| *Tipo de Utilizadores* | Sem conhecimento de programação | Algum conhecimento lógico de programação para desenvolvimento e customizações. |

| | | |
|---|---|---|
| *Flexibilidade* | Baixa | Alta |
| *Curva de aprendizagem* | Extremamente baixa | Moderada |
| *Uso* | Baixa aposta no mercado | Grande aposta no mercado |

*Tabela 1- Tabela de comparação entre NC e LC*

A Tabela 1 apresenta uma comparação entre as abordagens *NC* e *LC*, destacando as principais características de cada uma em diversos aspetos importantes no desenvolvimento de soluções digitais. No que diz respeito à complexidade, o *NC* caracteriza-se por ser uma abordagem de baixa complexidade, ideal para utilizadores sem experiência técnica, enquanto o *LC* apresenta uma complexidade moderada, dependendo do tipo de solução desenvolvida, exigindo algum conhecimento lógico de programação para customizações mais avançadas.

Em termos de customização, o *NC* oferece opções limitadas, pois os utilizadores dependem de funcionalidades pré-definidas pela plataforma. Por outro lado, o *LC* permite um nível elevado de personalização, possibilitando o desenvolvimento de soluções mais específicas.

Quanto ao tipo de utilizadores, o *NC* é restrito a utilizadores sem conhecimento em programação, enquanto o *LC* é destinado a utilizadores que possuem algum nível de conhecimento técnico.

A tabela também destaca a flexibilidade das abordagens, onde o *LC* se mostra superior ao *NC*, oferecendo maior liberdade de criação. No entanto, a curva de aprendizagem do *NC* é extremamente baixa, tornando-o mais acessível a iniciantes, enquanto o *LC* apresenta uma curva de aprendizagem moderada.

Finalmente, no que diz respeito à sua utilização no mercado, o *NC* é associado a uma baixa aposta no mercado, sendo mais adequado para soluções simples e rápidas, orientadas para o cliente e não para o servidor. Já o *LC* é amplamente adotado, representando uma grande aposta no mercado devido à sua combinação de rapidez no desenvolvimento com flexibilidade e personalização.

## 2.3. Metodologias de Desenvolvimento

A metodologia de desenvolvimento estipula um conjunto de princípios, práticas e ferramentas que direcionam o processo de desenvolvimento de software. O seu uso, contribui para uma estrutura melhor das atividades elaboradas, promovendo a interação entre utilizadores, engenheiros de sistemas e a tecnologia. A utilização de metodologias é crucial para garantir a qualidade, eficiência e previsibilidade do processo de desenvolvimento.

As atividades bem definidas numa metodologia são uma mais-valia na produção de software de alta qualidade, atendendo a todas as necessidades e requisitos do negócio, para além de facilitar a gestão dos projetos, pois permite ter um controlo de custos, prazos e melhor colaboração entre os elementos da equipa. Nesta secção iremos abordar metodologias tradicionais, ágeis e híbridas, tendo maior foco nas metodologias ágeis (Semedo, 2012).

É importante considerar a natureza do projeto, ou seja se estável pode ser favorecedor a adoção de uma metodologia tradicional, enquanto que em projetos dinâmicos, metodologias ágeis poderão ser mais eficientes, a complexidade do projeto (se complexos podem exigir metodologias mais estruturadas, enquanto projetos simples podem beneficiar da flexibilidade ágil), o tamanho da equipa (grandes equipas podem encontrar vantagens em metodologias tradicionais, enquanto equipas pequenas podem beneficiar da agilidade), os recursos, tempo e orçamento disponíveis (Semedo, 2012).

### 2.3.1. Desenvolvimento Ágil

O desenvolvimento ágil de software surgiu como uma alternativa inovadora aos métodos tradicionais de desenvolvimento, como o modelo em cascata. Para contrariar a rigidez e a documentação excessiva das metodologias clássicas, o desenvolvimento ágil propôs um conjunto de princípios e práticas que priorizam a entrega incremental e iterativa de valor ao cliente, a colaboração entre equipas e a adaptação a mudanças. O desenvolvimento ágil baseia-se em quatro pilares fundamentais (Ribeiro, 2018):

1. **Valorização dos indivíduos e interações:** Reconhece a importância da comunicação e do trabalho em equipa, priorizando a interação entre os membros para alcançar os objetivos do projeto.
2. **Software funcional em vez de documentação abrangente:** O mais importante é a entrega contínua do software funcional, priorizando a criação de valor tangível para o cliente em detrimento de extensa documentação.

3. **Colaboração com o cliente:** Incentiva a participação ativa do cliente ao longo do projeto, garantindo *feedback* contínuo para ajustar e melhorar o produto final.
4. **Adaptação a mudanças:** Contando com as mudanças nos requisitos que são inevitáveis, o foco é adaptar-se rapidamente a essas alterações, mantendo a flexibilidade durante todo o ciclo de vida do projeto.

Em virtude da adoção deste tipo de metodologia, o cliente pode receber novas funcionalidades constantemente e ter um papel ativo na fase de desenvolvimento ao fornecer um *feedback* regular e pedidos constantes, o que faz com que o produto final corresponda às suas necessidades e seja o mais próximo possível das expectativas, resultando numa maior satisfação. Desta forma, o projeto é menos resistente à mudança, tem menor probabilidade de se atrasar e de existirem custos adicionais, contribuindo assim para o aumento da produtividade.

Apesar destes benefícios, existem alguns contratempos na adoção deste tipo de metodologia quando se trata de projetos de grande escala, por norma, existe resistência por parte das equipas e dos gestores, dado que já possuem um enorme conhecimento do negócio e já têm o hábito de produzir algumas tarefas comuns no dia-a-dia, por isso, a importância da comunicação e sensibilização sobressalta nestas ocasiões para a possibilidade de adotar novas práticas. Os aspetos culturais, nomeadamente a cultura e suporte da chefia para a adaptação à nova metodologia e restrições financeiras, falta de expertise específica na equipa e falhas na comunicação entre as partes envolvidas, aumentam a dificuldade na fase de adoção do desenvolvimento ágil (Ribeiro, 2018).

De forma a ultrapassar todos os obstáculos que surgem quando se está a trabalhar com um desenvolvimento ágil, devem ser adotadas boas práticas, como os testes de metodologia através de um projeto piloto/mais pequeno, para que se possa ter uma melhor noção dos pontos que são precisos ajustar. Por outro lado, é fundamental capacitar os profissionais, por meio de formações, para que possam ter um conhecimento mais apurado dos princípios e práticas ágeis. O cliente e todas as partes interessadas, devem ser envolvidos ao longo do ciclo de vida do projeto, para garantir o alinhamento com as expectativas, assim como reuniões de *feedback* constantes para medir o esforço que foi feito, o que foi alcançado e o que pode melhorar, para além do *feedback* de que se é alvo constantemente por parte do cliente ou da área de negócio, de forma a garantir a entrega de valor e criar um ambiente de trabalho positivo e colaborativo (Ribeiro, 2018).

O IPSystems, nasceu com a ideologia atual na aposta das metodologias ágeis, para dar resposta às rápidas necessidades do mercado e às divergências que surgem. O facto de não existir dependência de código e de configurações, leva o IPSystems a apresentar soluções mais rápidas à área de negócio em questão e trazer maior evolução para o cliente, ao conseguir uma rápida resposta com menor esforço, garantindo todas as funcionalidades da aplicação.

Ao desenvolvermos um projeto no IPSystems, é possível, com pouco esforço, gerar rapidamente um código padrão de HTML e CSS, facilitando a criação de páginas funcionais num novo separador. Isso garante um deploy automático, realizado em aproximadamente 1 segundo, o que reduz significativamente o tempo investido em etapas manuais repetitivas. Essa agilidade torna-se especialmente vantajosa durante a fase de prototipagem, permitindo testar rapidamente novas ideias, validar funcionalidades com os utilizadores e fazer ajustes contínuos sem grande esforço. Desta forma, a plataforma otimiza o processo de desenvolvimento, garantindo eficiência, produtividade e uma entrega incremental mais alinhada às necessidades do mercado.

Além disso, existe uma colaboração mais eficiente entre quem está a desenvolver e os elementos das equipas não técnicas, tendo por base uma interface mais intuitiva com funcionalidades simplificadas, que permitem ajustar o projeto de forma incremental, respondendo rapidamente a mudanças de requisitos e feedbacks dos utilizadores. Desta forma, esta abordagem iterativa permite que as entregas sejam uma constante e mais controladas e assim, garantir o alinhamento entre o produto final e as expectativas do cliente.

Para além disso, a reutilização de código com o uso dos componentes padronizados e gestão das versões, economiza tempo e reduz os erros humanos. Assim, todos os esforços são aplicados em tarefas mais complexas e criativas, como a personalização de funcionalidades avançadas ou a integração com outras ferramentas, melhorando a qualidade do código e agilizando o desenvolvimento, para além de promover uma evolução contínua.

### 2.3.2. Metodologia em Cascata

A metodologia tradicional do mercado conhecida como metodologia waterfall ou em cascata, tem por base a definição de um conjunto processos que são projetados e segmentados para a equipa de desenvolvimento dar sequência às tarefas. A regra é desenvolver de forma faseada, cada uma das etapas escolhidas e não passar a uma próxima tarefa sem a anterior estar concluída (Adobe Communications Team, 2022).

Cada processo é dividido em 5 fases distintas, existe a fase de análise e requisitos, que é crucial para se ganhar conhecimento do negócio e fazer o levantamento de todos os requisitos necessários, tanto funcionais como os requisitos não funcionais, a fase de desenho que é a fase em que se faz o esboço daquilo que será o projeto que irá nascer e os respetivos diagramas e fluxos de atividades. Depois temos a fase de implementação, onde os desenvolvedores produzem de forma sequencial o seu trabalho, mas caso se justifique, os mesmos podem voltar novamente à fase de desenho para concluírem novos esboços das funcionalidades. Após os desenvolvimentos, temos a fase de verificação ou teste, para garantir a uniformidade dos requisitos e garantir a integridade da aplicação, neste caso, se não possui qualquer tipo de erros, e assegurar uma boa interface e experiência de utilizador, para além de serem praticados casos de teste, que podem ser de forma manual ou automática de acordo com o objetivo dos testers do projeto. Por fim, a fase de deployment e manutenção, que surge quando o projeto vai para produção e existe uma equipa de testes focada em assegurar que todas as funcionalidades cumprem os requisitos e o que mudou na plataforma (Adobe Communications Team, 2022).

Esta metodologia exige maior esforço na fase de definição dos requisitos iniciais e das fases posteriores e o objetivo é não existir uma mudança repentina nos requisitos, até porque as mudanças são difíceis de integrar no meio do desenvolvimento do projeto, por isso é fundamental que cada fase seja bem compreendida e implementada (Adobe Communications Team, 2022).

As mais valias desta metodologia são a existência de requisitos do projeto, o que leva os desenvolvedores a poderem antecipar possíveis erros ao fazer uma análise cuidadosa e desta forma, planear otimizações a serem feitas e alguns detalhes a ter em conta quando a fase de implementação começar. O orçamento do projeto na área tecnológica pesa sempre imenso, pois se não houver margem de lucro, o desenvolvimento do projeto jamais será uma opção viável e terão de ser repensadas as condições para o mesmo e redefinir tudo novamente (Adobe Communications Team, 2022).

O acompanhamento de um projeto bem delineado com todos os requisitos necessários, será sempre mais fácil de acompanhar por parte das equipas funcionais e de gestão, para além de os desenvolvedores terem sempre um guia durante a fase de implementação, que quando muito bem definida e detalhada, é uma enorme ajuda na fase de desenvolvimento e como não existem requisitos constantes por parte da área de negócio ou do cliente, o desenvolvimento é sempre feito de forma contínua (Adobe Communications Team, 2022).

Mas também existem contratempos como o facto de existir cronograma para as entregas, pois pode haver a possibilidade de uma entrega ter mais detalhes que não são fáceis de prever, imprevistos e até complexidade misturada que faz o seu desenvolvimento ser mais longo do que o que foi estimado antes e assim uma tarefa levar mais tempo do que o estimado anteriormente (Adobe Communications Team, 2022).

O cliente não tem uma ideia concreta do design do projeto, dado que uma fase inicial é feita através do levantamento de requisitos, planeamento do negócio e as respetivas necessidades. Então, o cliente será sempre surpreendido com o resultado e o risco de ter uma ideia definida e no final um resultado diferente é bastante elevado. O cliente do projeto não tem atividade durante a fase de desenvolvimento, então será mais complicado satisfazer todas as suas pretensões. Basta uma etapa do processo delimitado para o desenvolvimento do projeto se atrasar, para todas as outras etapas ficarem atrasadas (Adobe Communications Team, 2022).

O IPSystems, embora seja feito mais a pensar na metodologia Agile, também pode dar suporte à metodologia em cascata, na medida em que consegue garantir entregas mais rápidas e facilmente pode ser adaptado a alterações visto que permite ao utilizador com menos esforço, construir um projeto sólido e que atenda a todas as necessidades de negócio. Para além disso, o IPSystems, permite agilizar o processo de manutenção, por permitir que rapidamente seja desenvolvido um projeto e alterado, essa ideia pode ser um extra para a metodologia em cascata, pois mesmo que o resultado inicial não seja o desejado, rapidamente pode sofrer alterações e ser melhorado.

## 2.4. Linguagens de programação ou sintaxe

Nesta secção, serão abordadas todas as linguagens utilizadas no processo de desenvolvimento do IPSystems, desde a linguagem de estrutura, o HTML, ou a linguagem de estilos CSS às linguagens de programação, JavaScript e Python.

### 2.4.1. HTML

O HTML (Hyper Text Markup Language) é uma linguagem de marcação que serve para criar a estrutura básica de páginas web. Esta linguagem funciona através de elementos e tags que estruturam o conteúdo da página e formatam a sua aparência, cada página é composta por elementos, que são containers que definem seções da página. As tags, por sua vez, são instruções inseridas nos elementos para indicar como o conteúdo deve ser apresentado. Os

principais elementos e tags consistem em elementos de cabeçalho, utilizados para títulos (<h1>, <h2>, <h3>, etc.), elementos de parágrafo (<p>), elementos de imagem (<img>), elementos de link que criam hiperlinks para outras páginas ou recursos (<a>), elementos de lista não ordenadas e ordenadas (<ul>, <ol>, <li>), elementos de tabela (<table>, <tr>, <td>) e elementos de formulários interativos (<input>, <textarea>, <button>) (Caldeira, 2015).

Adicionalmente, os atributos são utilizados para adicionar informações e funcionalidades aos elementos. Por exemplo, no elemento <img>, os atributos src, height e width podem ser usados para especificar a origem, altura e largura da imagem. Os atributos são adicionados dentro das tags de abertura dos elementos, e eles modificam o comportamento ou a apresentação do elemento. (Caldeira, 2015).

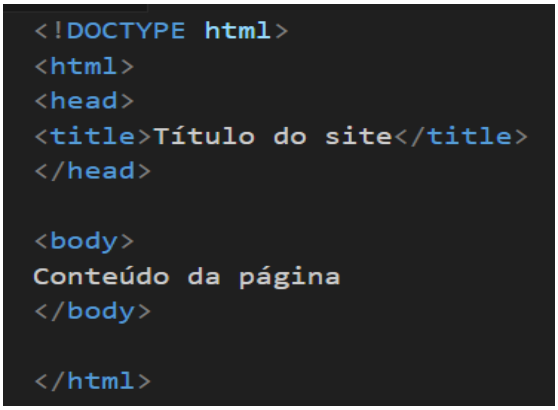



*Figura 2 - Estrutura básica de HTML*

O HTML é uma linguagem fundamental na criação de páginas web como podemos observar na figura 2, porque proporciona uma estrutura organizada e a capacidade de incorporar diversos tipos de conteúdo, como texto, imagens, vídeos e formulários, a partir das suas tags e elementos. Os atributos adicionam flexibilidade e funcionalidades adicionais aos elementos, tornando as páginas mais dinâmicas e interativas (Caldeira, 2015).

### 2.4.2. CSS

O CSS (Cascading Style Sheets) é uma linguagem de folha de estilos utilizada para personalizar a aparência das páginas web. Com esta linguagem, conseguimos definir propriedades como cor (do texto), fontes (define a família e tamanho da fonte), tamanho (definindo largura e altura dos elementos), margem e preenchimento (adapta o espaçamento em torno dos elementos), posicionamento de elementos e fundo, nomeadamente a cor e imagens de fundo e possui também vários tipos de seletores, designadamente (Scheidt, 2015):

- **Seletores por tipo de elemento** (por exemplo, h1, p, img).

- **Seletores por classe:** Identificam elementos com uma classe específica.
- **Seletores por ID:** Identificam elementos com um ID específico (por exemplo, #id).
- **Seletores por descendência:** Direcionam elementos que são descendentes de outro elemento especificado (por exemplo, h1 p).
- **Seletores por pseudo-classe:** Selecionam elementos com base no seu estado ou posição (por exemplo, :hover, :active).
- **Seletor filho direto:** seleciona qualquer elemento que corresponda ao segundo elemento que seja filho direto de um elemento que corresponda ao primeiro elemento (*Seletores Avançados Em CSS*, 2022).

Para além das características e propriedades já mencionadas, o CSS apresenta também inúmeros modelos de Layout, como os seguintes (Scheidt, 2015):

- **Layout em Bloco:** Elementos são empilhados uns sobre os outros, ocupando toda a largura disponível.
- **Layout em Linha:** Elementos fluem horizontalmente, permitindo que outros elementos fiquem ao lado deles.
- **Layout Flutuante:** Coloca elementos lado a lado, permitindo que o texto e outro conteúdo sejam moldados em torno deles.

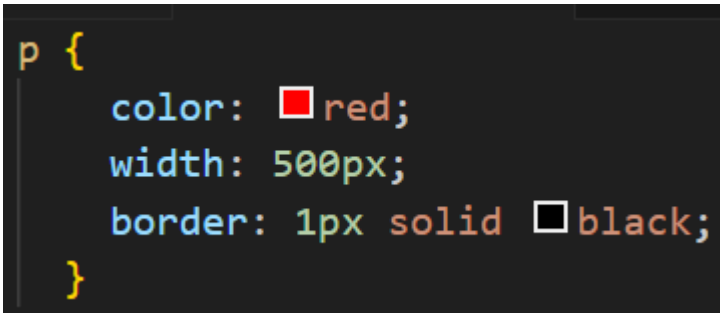



*Figura 3 - Exemplo de classe CSS*

Adicionalmente, nas classes de CSS que definimos, como podemos observar o exemplo da figura 3, permitem a utilização do flexbox css que é uma tecnologia eficiente para os layouts, oferecendo um controlo detalhado sobre a disposição dos elementos em uma dimensão de cada vez. Com propriedades como flex-direction, flex-wrap e flex-grow, é possível adaptar layouts responsivos e controlar o crescimento e o encolhimento dos elementos. As propriedades align-items e justify-content são cruciais para alinhar e justificar os elementos dentro do layout,

proporcionando precisão no posicionamento (*Conceitos Básicos De Flexbox - CSS*, n.d.). Também o CSS Grid Layout oferece flexibilidade bidimensional para organizar elementos numa página, permitindo posicionamento preciso em ambas as direções.

Com dimensões fixas ou flexíveis, os desenvolvedores podem criar layouts responsivos que se adaptam a diferentes dispositivos. Além disso, é possível posicionar itens com precisão, criar grids adicionais e alinhar elementos dentro do grid, proporcionando um controlo refinado sobre o layout da página e a sobreposição de conteúdos (*Conceitos Básicos De Flexbox - CSS*, n.d.).

Para adaptar a aparência das páginas web a diferentes dispositivos e resoluções de ecrã, além das propriedades, características e funcionalidades, as media queries são usadas para definir as páginas em smartphones, tablets e desktops. Já as animações CSS podem ser usadas para adicionar efeitos dinâmicos e interativos às páginas web, incluindo transições, transformações e animações de quadros-chave para criar experiências de utilizador envolventes.

Por fim, os pré-processadores como Sass (Syntactically Awesome Stylesheets) e Less (Leaner CSS) oferecem recursos como variáveis, mixins e funções para simplificar o desenvolvimento de estilos complexos, permitindo folhas de estilos mais organizadas e fáceis de manter. O CSS pode ser integrado com JavaScript para adicionar interatividade e comportamento dinâmico às páginas web. Isto inclui a manipulação de estilos, resposta a ações do utilizador e criação de efeitos animados (Scheidt, 2015).

### 2.4.3. JavaScript

O JavaScript é uma linguagem de programação inicialmente desenvolvida por Brendan Eich na Netscape, lançada em setembro de 1995 com o nome oficial de "livescript", que mais tarde foi alterado para JavaScript por razões de marketing. Possui uma sintaxe baseada em variáveis (‘var’, ‘let’ ou ‘const’), tipos de dados, os quais incluem strings, números, booleanos, arrays, objetos, entre outros, operadores aritméticos (‘+’, ‘-’, ‘*’, ‘/’), lógicos (‘&&’, ‘||’, ‘!’) e relacionais (‘==’, ‘===’, ‘!=’, ‘!==’, ‘>’, ‘<’, ‘>=’, ‘<=’), estruturas de controle (‘if’, ‘else’, ‘for’, ‘while’, ‘switch’) e funções, declaração utilizando `function` e chamada com parâmetros (Duarte, 2015).

O DOM (Document Object Model) é uma interface de programação que representa documentos HTML e XML como uma árvore de nós. Os principais métodos e propriedades destas interface são o ‘getElementById()’, ‘querySelector()’, ‘addEventListener()’,

‘innerHTML’ e ‘appendChild()’. Esta interface é então o que permite que o JavaScript interaja com os elementos HTML de uma página web, possibilitando manipulação dinâmica do conteúdo, estrutura e estilo da página. Para além do DOM, a aplicação dos princípios da Programação Orientada a Objetos (POO) ao JavaScript permite criar código mais modular, reutilizável e organizado.

### 2.4.4. Python

O Python é uma linguagem de programação de alto nível, o que significa que ela se aproxima da linguagem humana e possui uma sintaxe mais simples e intuitiva em comparação com linguagens de baixo nível, como Assembly, que é uma linguagem de programação interpretada, o que significa que o código fonte é executado linha por linha por um interpretador em tempo de execução, em vez de ser compilado em código de máquina antes da execução, proporcionando maior facilidade de desenvolvimento e portabilidade do código entre diferentes plataformas multiparadigma, o que permite aos desenvolvedores escolher o estilo de programação mais adequado para resolver um determinado problema. Para além disso, é uma linguagem de uso geral, o que significa que pode ser usada para uma ampla variedade de aplicações, desde desenvolvimento web até análise de dados, automação de tarefas, inteligência artificial e muito mais (Soubhia et al., 2019).

O Python é uma das linguagens mais populares e versáteis de programação, a sua grande utilização, deve-se sobretudo ao facto de ter uma sintaxe simples e legível, que facilita a escrita e compreensão do código, à grande variedade de bibliotecas e *frameworks* disponíveis. O facto de também possuir uma comunidade ativa e acolhedora, que contribui para o desenvolvimento de novas bibliotecas, oferece suporte e compartilha conhecimento, e ser multiplataforma, o que significa que os programas escritos em Python podem ser executados em diferentes sistemas operacionais sem a necessidade de modificação. A sintaxe básica do Python inclui (Soubhia et al., 2019):

- **Variáveis:** são declaradas atribuindo-lhes um valor e não é necessário declarar o tipo de uma variável explicitamente, pois o Python é uma linguagem de tipagem dinâmica.
- **Operadores.**
- **Funções:** são definidas usando a palavra-chave `*def*` e podem ter parâmetros e retornar valores.

• **Tipos de dados:** possui diversos tipos de dados, incluindo string, número e booleano, que são inferidos automaticamente pelo interpretador.

É importante notar que o Python utiliza indentação para delimitar blocos de código, e que é possível escrever blocos de código em uma linha se tiverem apenas um comando. Comentários podem ser adicionados ao código utilizando hashtags para comentários de uma linha e aspas simples ou duplas para comentários de várias linhas.

As principais estruturas de dados podem ser utilizadas para armazenar e organizar dados em programas Python de diversas maneiras: as listas podem ser usadas para armazenar coleções ordenadas de itens; as tuplas podem ser usadas para representar coleções imutáveis de itens, como coordenadas geográficas; os dicionários são úteis para armazenar dados estruturados em pares chave-valor; os conjuntos são úteis para eliminar duplicados de uma coleção de itens; e as strings são utilizadas para representar texto e são imutáveis, o que significa que não podem ser modificadas após a criação (Soubhia et al., 2019).

Realizar testes de unidade e integração é fundamental para garantir a qualidade e confiabilidade do código Python. Os testes de unidade verificam se unidades individuais de código, como funções e métodos, funcionam conforme o esperado, enquanto os testes de integração verificam se essas unidades funcionam corretamente quando combinadas umas com as outras. Isso ajuda a identificar e corrigir erros de forma proativa, reduzindo o risco de falhas no software em produção.

As principais ferramentas e *frameworks* utilizados para realizar testes em Python incluem, unittest, módulo de teste integrado à biblioteca padrão de Python, que fornece uma estrutura para escrever testes de unidade de forma organizada e eficiente e pytest, uma *framework* de testes de terceiros, que oferece recursos avançados para escrever e executar testes de forma mais simples e expressiva (Soubhia et al., 2019).

## 2.5. Frameworks e bibliotecas

Uma *framework* é um conjunto de ferramentas, bibliotecas e componentes que fornecem uma estrutura pré-definida para o desenvolvimento de software. Esses componentes, podem ser entendidos como blocos de código que podem ser reutilizados dentro da *framework* para

agilizar o processo de desenvolvimento. Neste capítulo, vamos abordar o React e o Django que são duas ferramentas utilizadas no desenvolvimento do IPSystems.

### 2.5.1. React

O React é uma biblioteca de JavaScript de código aberto desenvolvida pelo Facebook em 2013, com o objetivo de simplificar o desenvolvimento de páginas web interativas e dinâmicas. Destaca-se por ter uma abordagem centrada na criação de componentes reutilizáveis que representam partes da interface do utilizador. Esses componentes podem ser compostos de forma hierárquica para construir interfaces mais complexas e cada componente pode conter a sua própria lógica, estado e propriedades. Mais especificamente, os principais conceitos do React são (De Jesus Gomes, 2019):

- **Componentes**: unidades independentes e reutilizáveis de UI, que encapsulam o seu próprio estado e comportamento.
- **Estado (State)**: representa os dados mutáveis de um componente. Quando o estado de um componente muda, o mesmo pode renderizar novamente para refletir essas alterações.
- **Propriedades (Props)**: parâmetros passados para componentes, permitindo a personalização e a comunicação entre os mesmos.
- **JSX (JavaScript eXtension)**: trata-se de uma extensão de sintaxe que permite escrever HTML dentro de JavaScript, facilitando a criação de componentes.
- **Ciclo de Vida dos Componentes**: refere-se aos diferentes estágios pelos quais um componente passa, desde o seu início até à sua remoção da DOM.
- **Eventos:** permitem que os componentes respondam a interações do utilizador, como cliques e digitações.
- **Roteamento:** Gestão de diferentes "páginas" ou estados da aplicação, permitindo a navegação entre ambos de forma dinâmica.

O React tem o poder de se integrar com inúmeras outras tecnologias, como o Redux, que é uma biblioteca de gestão de estado que pode ser utilizada em conjunto com o React para gerir o estado de forma mais eficiente em aplicações grandes. O GraphQL, uma linguagem de consulta para APIs que pode ser integrada com o React para obter e manipular dados de forma eficiente.

Existe ainda a *Webpack*, que é uma ferramenta de empacotamento de módulos que pode ser usada para organizar e otimizar o código do React. A interconexão de conceitos do React, como componentes, estado e roteamento, permite a criação de interfaces web complexas. As integrações com APIs, bibliotecas de UI e outras ferramentas ampliam as funcionalidades do React e permitem a criação de aplicações web completas. Para além dos benefícios da interconexão e da possibilidade de integrações com outras tecnologias o React apresenta as seguintes vantagens (De Jesus Gomes, 2019):

- **Reutilização de Componentes:** Permite a construção de interfaces modulares e reutilizáveis.
- **Performance**: Utiliza um algoritmo de reconciliação eficiente que otimiza a renderização do DOM.
- **Comunidade Ativa:** Possui uma grande comunidade de desenvolvedores e uma vasta quantidade de recursos disponíveis.

Desta forma, o React foi escolhido para o desenvolvimento do *frontend* do IPSystems pela sua versatilidade e por permitir o desenvolvimento de aplicações robustas. Existe ainda, a possibilidade de se desenvolver uma aplicação mobile com React Native, o que é uma mais-valia para o futuro, por permitir a reutilização de código e um conhecimento mais sólido do mesmo.

### 2.5.2. Django

O Django é uma *framework* de código aberto que permite desenvolver aplicações sólidas, complexas e escaláveis com base na linguagem de Python e fornecer serviços como *API's REST*, podendo fornecer lógica de negócio, CRUDS que são ações de criar, atualizar, obter e eliminar, podendo também armazenar os dados respetivos em bases de dados. Esta *framework* apresenta uma arquitetura, com modelos que representam a estrutura de dados da aplicação, vistas que são responsáveis por processar requisições e gerar respostas, *templates* que definem a apresentação das páginas HTML, mapeamento de *URLs* para *views* específicas e *middleware* que interceta requisições e respostas, permitindo adicionar funcionalidades como autenticação e cache.

O *Django* aparece também como um facilitador do desenvolvimento web, o qual vem das seguintes funcionalidades (Kumar & Jaiswal, 2015):

1. **Roteamento de URLs:** permite mapear URLs para views específicos, facilitando a organização do código e a criação de URLs amigáveis para o SEO.

2. **Sistema de Templates:** permitem separar a lógica da apresentação, tornando o código mais limpo e fácil de manter.
3. **ORM (Object-Relational Mapping):** abstrai a interação com a base de dados, permitindo que os desenvolvedores trabalhem com objetos Python em vez de SQL, tornando o desenvolvimento de aplicações web mais rápido e eficiente.
4. **Autenticação e Autorização:** sistema que permite gerir o acesso dos utilizadores à sua aplicação, economizando tempo e esforço no desenvolvimento de recursos de segurança.

De forma a entender melhor o funcionamento do Django e das componentes e funcionalidades previamente mencionadas, é importante perceber como se processa a interação entre os vários componentes desta *framework*. O Django recebe a requisição do cliente, que contém informações como a URL, o método HTTP e os dados da requisição. Depois, o sistema de roteamento de URLs analisa a URL da requisição e mapeia-a para a view específica que deve processá-la.

A *view* é responsável por processar a lógica da requisição, podendo aceder à base de dados, realizar cálculos, interagir com outros sistemas e gerar a resposta. Pode também, renderizar um *template* para gerar a resposta HTML. O *template* define a estrutura e o layout da página, e pode ser personalizado com variáveis e *tags*. Após o sucedido, a resposta passa pelo *middleware*, que pode adicionar funcionalidades como autenticação, cache e compressão. Por fim, a resposta final é enviada para o cliente (Kumar & Jaiswal, 2015).

O Django é muitas vezes integrado com outras tecnologias. Estas podem ser uma API *Rest*, *WebSockets* ou Bibliotecas de *Machine Learning*. O Django pode ser facilmente integrado com *frameworks* de API REST para fornecer acesso aos dados da aplicação via HTTP, permitindo criar APIs RESTful que podem ser consumidas por outras aplicações web, dispositivos móveis e *frontends* de JavaScript.

Pode ser integrado a bibliotecas de *WebSockets* como Django Channels para permitir comunicação bidirecional em tempo real entre o cliente e o servidor, permitindo criar funcionalidades como chat em tempo real, notificações e atualizações ao vivo. E pode também ser integrado a bibliotecas de *machine learning* como *TensorFlow* e *scikit-learn* para adicionar funcionalidades inteligentes, permitindo criar sistemas de recomendação, classificação de conteúdo, deteção de fraudes e outras aplicações de *machine learning*.

As integrações com outras tecnologias podem ampliar as funcionalidades do Django e criar aplicações web completas, como por exemplo, aplicações de chat em tempo real, sistemas de recomendação, análise de dados, *dashboards* interativos, redes sociais e jogos online. A escolha das tecnologias a serem integradas dependerá das necessidades específicas do projeto (Kumar & Jaiswal, 2015).

As várias possibilidades referidas nos parágrafos acima resultam em várias vantagens do Django como *framework*, sendo algumas delas a rapidez de desenvolvimento, ser amigável para iniciantes, com uma curva de aprendizagem suave, ser capaz de lidar com grandes volumes de tráfego, inclui recursos de segurança integrados que protegem a aplicação de contra-ataques e possui uma comunidade grande e ativa que oferece suporte e ajuda aos desenvolvedores (Kumar & Jaiswal, 2015).

O Django é uma *framework* poderosa e versátil que pode ser usada para criar uma grande variedade de aplicações web. No entanto, é importante ter em mente que a integração com outras tecnologias pode aumentar a complexidade do projeto e exigir conhecimento adicional das tecnologias escolhidas. Ao escolher tecnologias para integrar com o Django, é importante considerar as necessidades do projeto, a experiência e conhecimento existentes e a existência de uma comunidade ativa e boa documentação (Kumar & Jaiswal, 2015).

A *framework* Django foi a escolhida para o desenvolvimento backend do IPSystems por oferecer robustez e a pensar no futuro, com o aumento da complexidade do IPSystems, é importante existir como base um software poderoso que dê suporte a todas as funcionalidades da aplicação.

## 2.6. Conclusão do Capítulo

Com base na análise das plataformas *low-code*, as metodologias de desenvolvimento de software, linguagens de programação e as *frameworks*, podemos concluir que as tecnologias *low-code* oferecem uma abordagem inovadora para acelerar o desenvolvimento de software. Além disso, a integração de linguagens de programação como JavaScript e Python com *frameworks* e bibliotecas poderosas, como React e Django, amplia as possibilidades de desenvolvimento de aplicações web sofisticadas e escaláveis.

As metodologias tradicionais e ágeis apresentam diferenças significativas nas suas abordagens e valores. Enquanto as metodologias tradicionais geralmente enfatizam a documentação detalhada e a definição rígida de requisitos, as metodologias ágeis priorizam a

adaptação a mudanças, a comunicação eficaz e iterações curtas. São destaques, as principais diferenças em cinco aspetos-chave (Semedo, 2012):

**1. Definição de Requisitos:**

- Metodologia Tradicional (Cascata): É feita uma definição pormenorizada dos requisitos essenciais e respetiva documentação no início do projeto.
- Metodologia Ágil (Scrum): Os requisitos são definidos empiricamente de acordo com as necessidades dos clientes, com ênfase nos objetivos dos utilizadores e tarefas, permitindo mudanças ao longo do desenvolvimento.

**2. Natureza dos Requisitos:**

- Metodologia Tradicional (Cascata): Abordagem preditiva, com requisitos aprovados pelos clientes antes da concepção do projeto.
- Metodologia Ágil (Scrum): Abordagem empírica, com necessidades dos clientes trabalhadas de acordo com o valor do projeto, permitindo mudanças ao longo da construção do projeto.

**3. Estrutura da Equipa:**

- Metodologia Tradicional (Cascata): Existe uma hierarquia com um responsável pelo projeto e um grupo focado em vários projetos simultaneamente.
- Metodologia Ágil (Scrum): Proximidade da equipa de projeto, desde o início até a entrega final, com responsabilidade compartilhada e sem hierarquia, onde é possível deixar considerações constantes sobre o trabalho produzido.

**4. Implementação do Projeto:**

- Metodologia Tradicional (Cascata): As diferentes fases do projeto são implementadas de forma sequencial, só se inicia uma fase quando a anterior estiver concluída e todas dependem umas das outras e entregue quando o cronograma estiver concluído.
- Metodologia Ágil (Scrum): É implementado de maneira iterativa, dividindo o cronograma em iterações curtas, onde a equipa e o cliente decidem os recursos a serem implementados em cada interação.

**5. Atitude em Relação a Mudanças:**

- Metodologia Tradicional (Cascata): Dificuldade e custo associados à introdução de mudanças durante o projeto.

- Metodologia Ágil (Scrum): Incentiva a mudança em todas as fases do projeto, permitindo alterações sem aumento do orçamento.

Estas diferenças evidenciam as abordagens distintas das metodologias tradicionais e ágeis, oferecendo insights valiosos para a escolha da abordagem mais adequada às necessidades do desenvolvimento de software (Semedo, 2012).

Neste capítulo, explorámos a variedade de metodologias de desenvolvimento de software, desde as tradicionais até as ágeis e híbridas. A importância das metodologias para garantir a qualidade, eficiência e previsibilidade do processo de desenvolvimento foi destacada, juntamente com os tipos de metodologias de desenvolvimento e suas características, vantagens e desvantagens.

A discussão sobre metodologias híbridas e os fatores importantes na escolha da metodologia apropriada para um projeto contribuiu para a compreensão da complexidade envolvida na seleção da abordagem de desenvolvimento mais adequada. É evidente que a escolha da metodologia de desenvolvimento de software tem um impacto significativo no resultado do projeto, e ponderar os diversos fatores, como a natureza do projeto, complexidade, tamanho da equipe, recursos, tempo e orçamento, é crucial para o sucesso do empreendimento. Essa análise proporciona uma base sólida para futuras pesquisas e práticas no campo do desenvolvimento de software.

# Capítulo 3 - Projeto IPSystems

## 3.1.1. Introdução e Planeamento

A plataforma IPSystems surge como alternativa às plataformas *low-code* existentes apostando no desenvolvimento rápido de soluções web para poderem ser desenhadas algumas aplicações ou protótipos de forma mais rápida e eficaz. Apresenta como vantagens ser open source, bem como a fácil utilização *light* (web e mobile). Por exemplo, sendo possível a edição de um widget de forma rápida e imediata, em qualquer lado a partir da aplicação. No entanto, tal como todas as plataformas, apresenta as suas limitações, nomeadamente não ser muito focado no servidor e estar dependente de conexão à internet.

Tendo em mente a crescente procura por soluções web de fácil utilização e a integração de novas tecnologias para manter a plataforma inovadora e alinhada com as tendências do mercado, esta plataforma tem como cliente potencial o utilizador que não domina tanto a

programação e pretende fazer uma landing page, podendo assim fazer uma página única usando classes de CSS e não precisando de ter tanto trabalho na produção de estrutura com o HTML. Por fim, relativamente à disposição de pagamento, é apresentada a um preço acessível e alinhado às necessidades de utilizadores menos técnicos.

Antes de definir a estrutura do projeto foi feito o planeamento do mesmo, tal como é possível observar na figura 4.

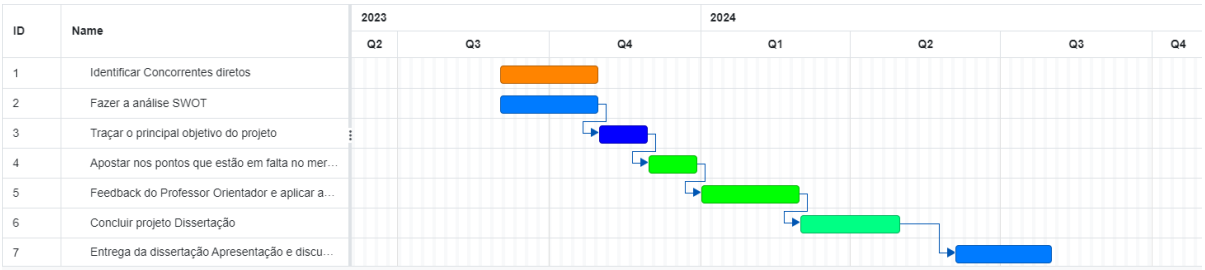



*Figura 4- Esquema do plano da tese*

A figura 4, representa um gráfico de Gantt, que é uma ferramenta amplamente utilizada na gestão de projetos para acompanhar tarefas e cronogramas. Neste gráfico, podemos observar as principais etapas na fase de desenvolvimento do IPSystems com as suas durações e dependências. Num primeiro momento, foram identificados os concorrentes diretos já existentes no mercado, de forma a perceber as soluções já existentes. Em paralelo, foi feita a análise SWOT para identificar os pontos fortes, fracos, oportunidades e ameaças, para alinhar a estratégia do IPSystems e a sua posição no mercado.

Ainda no ano de 2023, foi traçado o principal objetivo do projeto, tendo por base os insights da análise SWOT, sendo posteriormente feita a aposta nos pontos em falta no mercado, ou seja, a partir das necessidades e lacunas identificadas, foram priorizadas as funcionalidades que diferenciam a plataforma das que já existem, agregando valor para o público-alvo.

Em 2024, foram feitas sessões de refinamento técnico com o professor orientador para perceber o estado do projeto e os respetivos ajustes para melhorar alguns pontos em falta.

Seguindo-se assim, mais 2 fases, a primeira de conclusão do projeto e da dissertação, consolidando tudo o que foi desenvolvido no projeto e o relatório escrito e por fim, a entrega da dissertação e respetiva discussão.

## 3.1.2. Estrutura do Projeto

O IPSystems, é uma plataforma que pode ser acedida online e é constituída por diferentes tipos de páginas, depois de o utilizador se registar e iniciar sessão com a sua conta. A partir desse ponto, é possível usufruir de cursos sobre a plataforma, consultar documentação e desenvolver aplicações a partir da página de desenvolvimento, portanto a plataforma está distribuída da seguinte forma (figura 5):

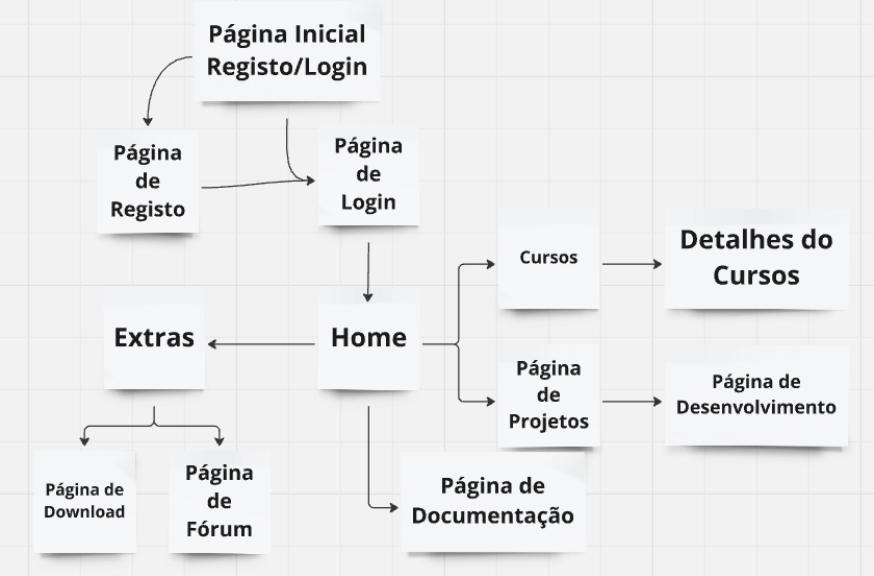



*Figura 5- Diagrama de Arquitetura geral do IPSystems*

### 3.1.2.1. Página de Registo

**Página de criação de utilizador:** Permite que um novo utilizador ou outro utilizador já existente, crie uma conta, para isso precisa de escolher um nome, email e password e efetuar o registo.

### 3.1.2.2. Login

**Página de Acesso:** Página de autenticação dos utilizadores, onde deve ser colocado o respetivo email e palavra-passe, caso esses dois campos estejam devidamente preenchidos e existam na base de dados por parte do servidor, o utilizador conseguirá aceder às próximas páginas, sendo reencaminhado para a página inicial.

#### 3.1.2.3. Home

**Dashboard:** Área constituída pela barra de navegação para se redirecionar nas diferentes páginas existentes e com algum contexto sobre a plataforma.

#### 3.1.2.4. Cursos

**Página de Cursos:** Esta página apresenta os vários cursos IPSystems criados para o utilizador poder enriquecer o seu conhecimento na plataforma e dar os seus primeiros passos dentro da mesma.

#### 3.1.2.5. Detalhes do Curso

**Detalhes de um Curso:** Esta página apresenta um curso IPSystems específico com o respetivo título, descrição e um vídeo para ajudar o utilizador a estudar e aprender mais sobre a plataforma.

#### 3.1.2.6. Página de Projetos

**Página de Projetos:** Página onde se encontram todos os projetos já criados pelo utilizador que esteja com sessão iniciada, no fundo trata-se da sua área pessoal de projetos onde pode retomar atividades dos mesmos que já foram iniciadas anteriormente.

#### 3.1.2.7. Página de Desenvolvimento

**Página de desenvolvimento:** Área onde o utilizador trabalha um novo projeto ou um projeto já existente, escolhendo o nome do mesmo, arrastando as diferentes widgets disponíveis e aplicando algumas classes de css ou JavaScript à sua escolha.

#### 3.1.2.8. Documentação

**Página da Documentação:** Esta página mostra algumas boas práticas da plataforma para que o utilizador consiga utilizar as melhores práticas ao longo do seu processo de desenvolvimento.

#### 3.1.2.9. Extras: (Download e Fórum)

**Página de Download:** Esta página permite fazer a instalação de todos os executáveis IPSystems que estiverem disponíveis no portal.

**Página do Fórum:** Permite que os utilizadores e desenvolvedores de IPSystems comuniquem ativamente com dúvidas que podem ser respondidas por qualquer elemento registado.

### 3.1.3. Métodos

Para desenvolver este projeto, foi adotada uma metodologia ágil, dividindo o desenvolvimento em iterações para garantir flexibilidade e ajustes contínuos. Numa primeira fase, foi desenvolvido algum trabalho de pesquisa sobre vários tipos de aplicações, tanto de *low-code*, como aplicações sem código, de forma a avaliar qual a melhor alternativa de aposta neste projeto. A maior diferença encontrada, foi que o desenvolvimento de aplicações consideradas low-code, acabam por ter maior ênfase a nível de mercado, dado que o desenvolvimento sem necessidade de código, acaba por limitar o utilizador a um determinado nível de aplicação, sendo mais complicado definir certos padrões e principalmente, ampliar a complexidade do projeto.

Portanto, embora o IPSystems, ainda tenha um foco de desenvolvimento bastante prático a nível de construção de frontend na íntegra do utilizador, é focado de uma forma a permitir a utilização de classes e é pensado de forma a adotar mais tarde, a possibilidade de lógica a nível de JS e alguma facilidade na integração de BDs, embora numa primeira fase não tenha todas essas íntegras devido ao faseamento e prioridade no projeto, tendo em conta que se trata em primeira instância de um protótipo.

Foi adotada a metodologia Kanban para gerir e priorizar as tarefas ao longo do desenvolvimento do projeto. O Kanban é uma abordagem ágil que utiliza um quadro visual para acompanhar o fluxo de trabalho, ajudando a identificar gargalos e a manter o foco nas entregas mais importantes, desta forma, as tarefas foram categorizadas em diferentes estados, como *a fazer (To Do), em progresso (In Progress)* e *concluído (Done),* permitindo uma visualização clara do estado de cada atividade.

Além disso, o Kanban possibilitou a atribuição de novas tarefas prioritárias de forma dinâmica, sempre que surgiram novas necessidades ou ajustes necessários para adotar no projeto. Essa flexibilidade garantiu que o trabalho se mantivesse alinhado às necessidades imediatas e organizar as entregas de maneira eficiente, sem comprometer os prazos e a qualidade do resultado final.

Ainda na primeira fase, de pesquisa, pedida pelo Professor Orientador, foi feita uma análise de todos os tipos de concorrentes dentro deste mercado de desenvolvimento sem código ou de *low-code*, assim como uma análise SWOT mais à frente, com o intuito de saber os pontos fortes e fracos de cada uma das ferramentas e todo o tipo de oportunidades e ameaças que se enfrentam neste mercado.

Em seguida, foi delineado o objetivo principal do projeto, estabelecendo claramente o que se pretende alcançar. Foi cuidadosamente considerado o produto e o nicho de mercado no qual se pretende atuar, bem como as estratégias para se diferenciar e se tornar um produto de destaque e sucesso. Além disso, foram identificados todos os pontos em falta no mercado, permitindo uma compreensão mais precisa das lacunas a serem preenchidas e das oportunidades de melhoria futura.

A definição do objetivo principal proporcionou uma visão clara e direcionada do propósito do projeto, servindo como guia para todas as etapas de desenvolvimento e implementação. Ao delinear o produto e o mercado-alvo, foi possível concentrar os esforços na criação de soluções que atendam às necessidades específicas dos utilizadores e se destaquem num cenário competitivo.

A estratégia de diferenciação foi cuidadosamente elaborada, considerando os pontos fortes do produto, as tendências do mercado e as expectativas dos consumidores, envolvendo a identificação de recursos exclusivos, aprimoramento da experiência do utilizador e estabelecimento de uma marca distinta e reconhecível.

Ao analisar os pontos em falta no mercado, foram identificadas oportunidades de inovação e aprimoramento contínuo do produto. Essas lacunas serviram como base para futuras melhorias e desenvolvimento, garantindo que o produto permaneça relevante e competitivo ao longo do tempo.

Portanto, esta fase de planeamento estratégico foi fundamental para estabelecer uma base sólida para o projeto, definindo metas claras, estratégias de diferenciação e áreas de foco para melhoria contínua, uma abordagem proativa e orientada para o mercado é essencial para o sucesso a longo prazo do produto.

Para o desenvolvimento do projeto foram escolhidas as *frameworks* de React no frontend e Django no backend, dado que foram ambas abordadas neste curso e dado o interesse do aluno em ambas para além da robustez e combinação, pois são ambas compatíveis e integráveis, pois

os *frameworks* foram projetados de forma a trabalharem bem em conjunto por permitirem uma integração prática e eficaz entre ambas.

O React é uma plataforma de frontend bastante interessante por ter uma arquitetura simples e prática, facilmente é criado um design interessante e dinâmico, o uso de rotas e a utilização de componentes permitem poupar muitas linhas de código e replicar bastante trabalho para algumas páginas, para além de fornecer interfaces dinâmicas e modernas, puxando os dados de forma assíncrona no carregamento de páginas.

O Django prevalece por ser um *framework* backend robusto e com um conjunto de ferramentas, que podem acelerar o desenvolvimento devido às funcionalidades implementadas na sua íntegra. Também a facilidade em aceder às bases de dados, é particularmente útil para simplificar a manipulação de dados e reduzir a quantidade de código e a complexidade para interagir com as BDs.

Ambas as *frameworks* de React e Django têm uma comunidade ativa e uma ampla documentação, boa flexibilidade e escalabilidade. Em suma, o conhecimento do professor orientador, acaba por ser um ponto de influência nas escolhas pois tem a capacidade de dar apoio técnico e ajudar em algumas tomadas de decisão, face à sua experiência e domínio.

Foram planeadas as principais funcionalidades da aplicação que são consideradas prioritárias para este protótipo e pensaram-se algumas funcionalidades que não são prioridades dada esta fase inicial em que se pretende criar um protótipo com uma pequena solução. Todos estes detalhes são abordados de forma mais pormenorizada nos próximos tópicos.

#### 3.1.3.1. Planeamento base de cada página do projeto

Antes de se iniciar o processo de desenvolvimento do projeto, foram feitos alguns esquemas gerais sobre a aplicação, tendo em conta a estrutura pré-definida sobre o mesmo. Portanto foi pensado primeiramente uma página de acesso ao site, uma página inicial de boas-vindas do IPSystems e uma página de desenvolvimento, para definir alguma lógica de trabalho e perceber a eficácia da ideia do projeto. Então foram pensadas estas primeiras páginas numa fase inicial tendo em conta que iriam ser necessárias mais páginas ao longo do tempo para o projeto. Por isso, a primeira fase de implementação, constituiu três páginas de esboço com as ideias principais do projeto para servir de guia no desenvolvimento.

### 3.1.3.2. Esboço dos mockups do projeto

Foram desenvolvidos alguns mockups do projeto IPSystems no Figma com o intuito de fornecer uma representação visual inicial do sistema, facilitando a definição das páginas e do layout do projeto. Estes mockups foram elaborados de forma geral e prática, visando estabelecer uma base visual e sólida sem se aprofundar em detalhes excessivos no início do processo.

A principal razão para essa abordagem foi evitar discrepâncias visuais no decorrer do desenvolvimento. Sem uma representação clara do design, pode haver divergências e pode resultar num produto final visualmente inconsistente, para além de que com uma representação torna-se mais fácil a fase de desenvolvimento, porque existe uma base já estabelecida.

A primeira página que foi pensada, foi a página inicial do utilizador do IPSystems (figura 6), onde encontra dois botões, o primeiro botão "registar", caso não tenha conta no IPSystems e pretenda criar, onde é redirecionado para uma página com três campos obrigatórios de registo: o nome, email e senha. Na situação de já ser um utilizador IPSystems, existe a possibilidade de escrever as suas credenciais e caso estejam corretas, efetuar a entrada no site, caso contrário é negado o redireccionamento para outra página do site, que será a página de início do portal.

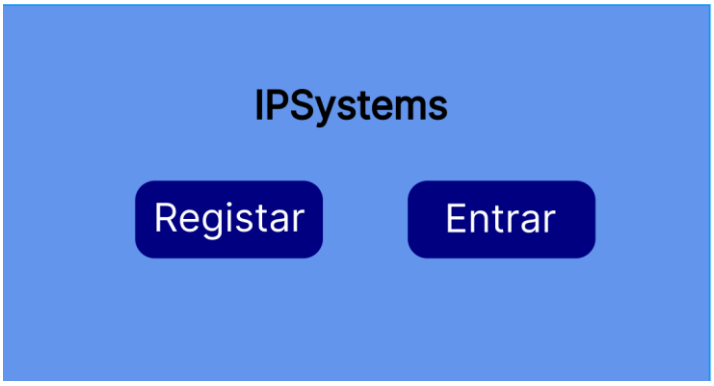



*Figura 6- Página inicial do IPSystems*

As páginas de registo e de login com credenciais não foram detalhadamente desenhadas durante a fase de criação dos mockups, uma vez que se assumiu que seguiriam um padrão visual semelhante ao das primeiras representações gráficas.

Em contrapartida, foi desenvolvida a próxima página do fluxo, a **página de boas-vindas do IPSystems** (Figura 7). Esta página foi desenhada com o objetivo de rececionar o utilizador

após a autenticação bem-sucedida, estabelecendo o ponto de partida para a navegação no sistema.

A prioridade que foi dada no design à página de boas-vindas, reflete a importância de criar uma experiência agradável e acolhedora, mantendo a consistência visual do projeto e garantindo que o utilizador se sinta orientado ao prosseguir com a utilização do IPSystems.


*Figura 7- Representação da página de boas-vindas do IPSystems*

A página de boas-vindas do IPSystems é mais informativa que a representada e tem alguns estilos diferentes, mas a representação não deixa de ser idêntica e deixa uma mensagem informativa para todos os utilizadores que entram no portal, principalmente os que entram pela primeira vez e não têm uma noção tão clara do que se trata o IPSystems.


*Figura 8-Página de boas-vindas do IPSystems*

A página de boas-vindas do IPSystems (figura 8) está disposta como mostra a figura anterior e com algum texto em baixo informativo, sobre os valores da marca. O retângulo observável que diz “IPSystems” está representado como uma animação, tendo ficado na imagem anterior um pouco deslocado.

Para dar suporte ao portal, foi planeada a construção do Backoffice desenvolvido em Django, e também foram feitos alguns mockups para projetar as interfaces da aplicação. A página inicial foi pensada para ter um redireccionamento para a página de Documentação e a página de cursos (figura 9).

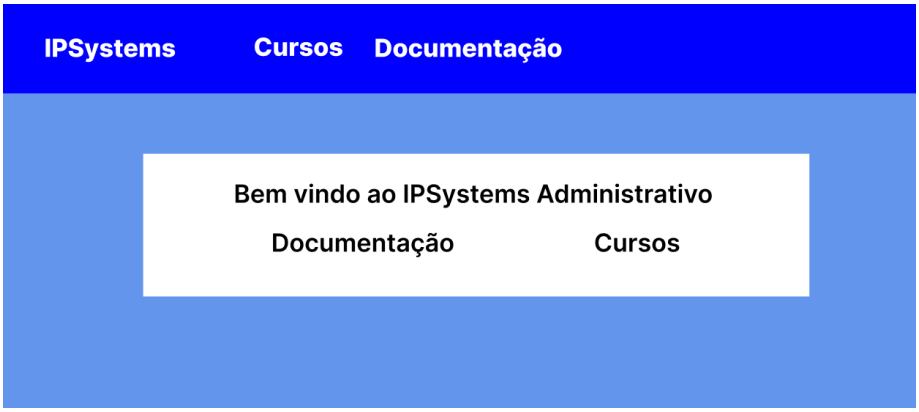



*Figura 9- Página inicial do Backoffice*

Em seguida, foram pensadas mais duas páginas principais, a página de listagem que é viável para a documentação e para os cursos e também foi pensada uma página para se criar itens relativo a ambos, visualizar esses dados, alterá-los ou eliminá-los.

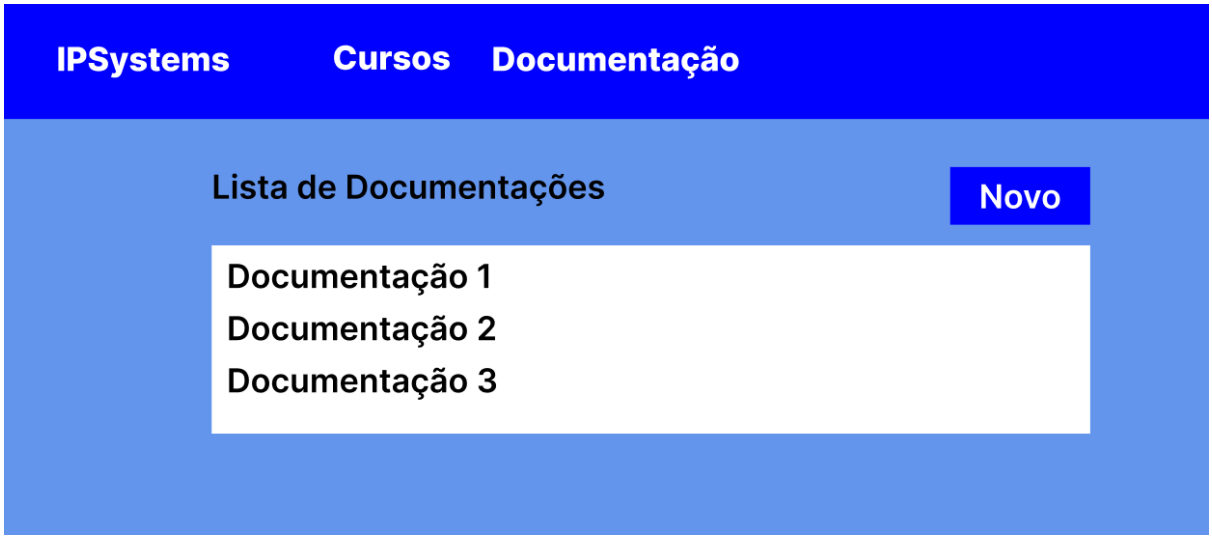



*Figura 10- Página de listagem das documentações*

A figura 10, mostra a representação da lista de documentações que foi pensada para o projeto, assim como para os cursos é utilizada o mesmo tipo de lista seguindo o mesmo padrão, com um botão "novo" para criar um conteúdo novo e o texto do conteúdo da lista apresenta-se sempre como clicável para se poder editar ou eliminar o conteúdo. Ao carregar para criar um conteúdo será apresentado um formulário para se adicionar um novo conteúdo.

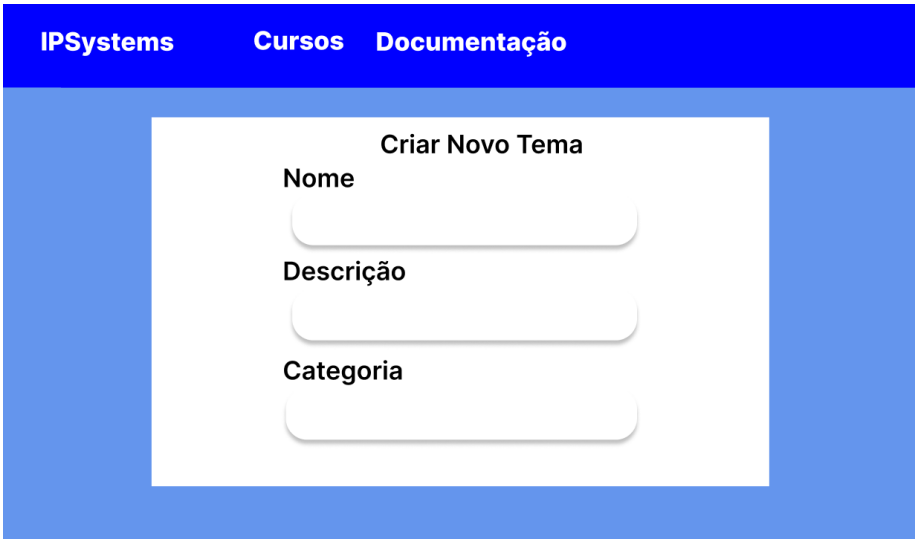



*Figura 11- Criação de novo documento*

Como é observável na imagem 11, existem 3 campos que permitem definir o nome da documentação, a descrição e a sua categoria, da mesma forma, a criação de cursos possui alguns campos para preenchimento para se efetuar a criação. Esta página de criação, também é utilizada, quando na tabela da listagem da documentação se clica em algum nome de um

documento específico, mas a diferença é que os campos aparecem já preenchidos com os dados do documento já criado.

### 3.1.3.3. Alterações feitas nos Mockups

Para colmatar algumas dificuldades atravessadas pelo utilizador na página de desenvolvimento, foram feitas alterações na disposição dos ícones e na estrutura dos botões. Numa primeira fase, foi estruturada a página de forma que os botões de "CSS", "Publicar" e o input do título do site ficassem alinhados do lado esquerdo do ecrã, mas chegou-se à conclusão que não seria boa solução dado que ficava mais complicado no processo de arrastar e soltar por parte do utilizador, como podemos perceber na figura 12.

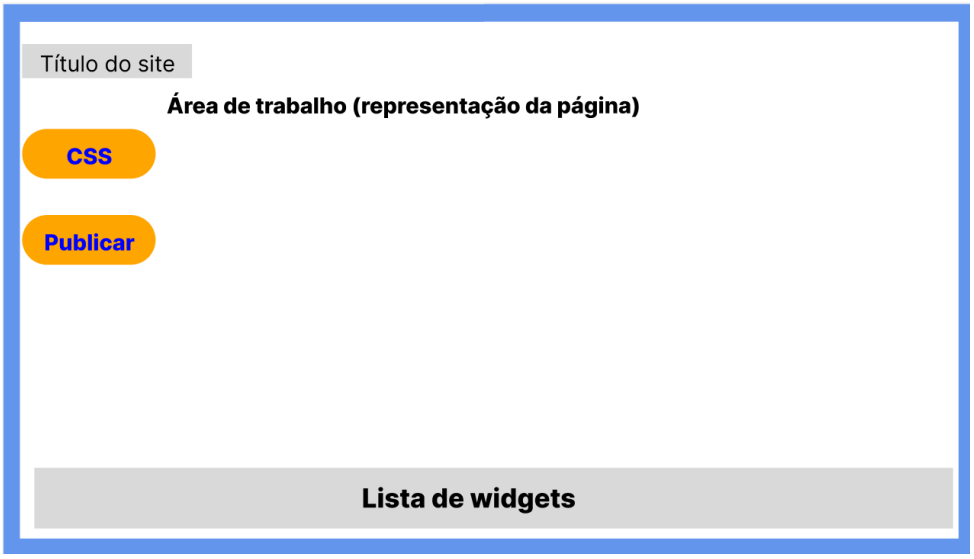



*Figura 12- Primeiro Esboço da Página de Desenvolvimento*

Depois de concluir que o primeiro esboço não é a melhor solução para o projeto, elaborou-se outro esboço mais bem preparado e pensado para o utilizador na fase de implementação. A caixa de ferramentas encontra-se ainda localizada na parte inferior, assim como o input do título do site e os botões de "CSS" e "Publicar". Nota-se também, a adição de dois novos botões, um de "Esconder opções", que ao ser carregado, oculta o input do título do projeto e os botões dispostos ao lado do css e de publicar. Para além da existência do botão "Esconder widgets" que permite esconder a caixa de ferramentas para o utilizador ter um espaço mais limpo da sua área de trabalho e poder arrastar melhor os elementos que deseja, tal como é visível na figura 13.


*Figura 13 - Segundo Esboço da Página de Desenvolvimento*

Desta forma, acaba por ser mais interativo para o utilizador, a fase de desenvolvimento, pois tem a possibilidade de arrastar e soltar os componentes que pretende de forma mais facilitada e também tem a opção de esconder as widgets ou as opções em cima para aproveitar melhor o espaço.

### 3.1.3.4. Processo de mudança de algumas funcionalidades

Depois de implementadas algumas páginas do projeto, foi necessário repensar algumas funcionalidades, uma das páginas foi a **Página de Desenvolvimento**, que inicialmente estava definida com widgets no lado esquerdo da página e com botões fixos então a área arrastável também era mais reduzida, o que tornava a fase de implementação mais limitada e complexa. Assim, foi necessário pensar numa melhor forma de aproveitar a página e permitir ao utilizador ter uma melhor experiência a desenvolver, aproveitando mais espaço da página e ter a possibilidade de esconder ou mostrar alguns botões.

A **Página de Desenvolvimento**, também possui uma área com um botão "CSS" que ao ser clicado, permite abrir um popup que é uma folha de estilos para o utilizador definir as classes que pretende adotar às widgets arrastadas na área de desenvolvimento. Um botão "publicar" que permite abrir um novo separador com a representação trabalhada na área de desenvolvimento da página, gerando um deploy automático no link do site.

Ainda existe nessa área uma pequena caixa de texto para digitar o título do site que está a ser desenvolvido. Ambos esses dois botões "CSS" e "publicar" e a caixa de texto do título do projeto podem ser escondidos, caso o utilizador na fase de implementação pretenda, de modo

a aproveitar uma maior área de desenvolvimento. Numa primeira fase, existia uma disposição diferente, estando também alinhados no canto superior esquerdo da página.

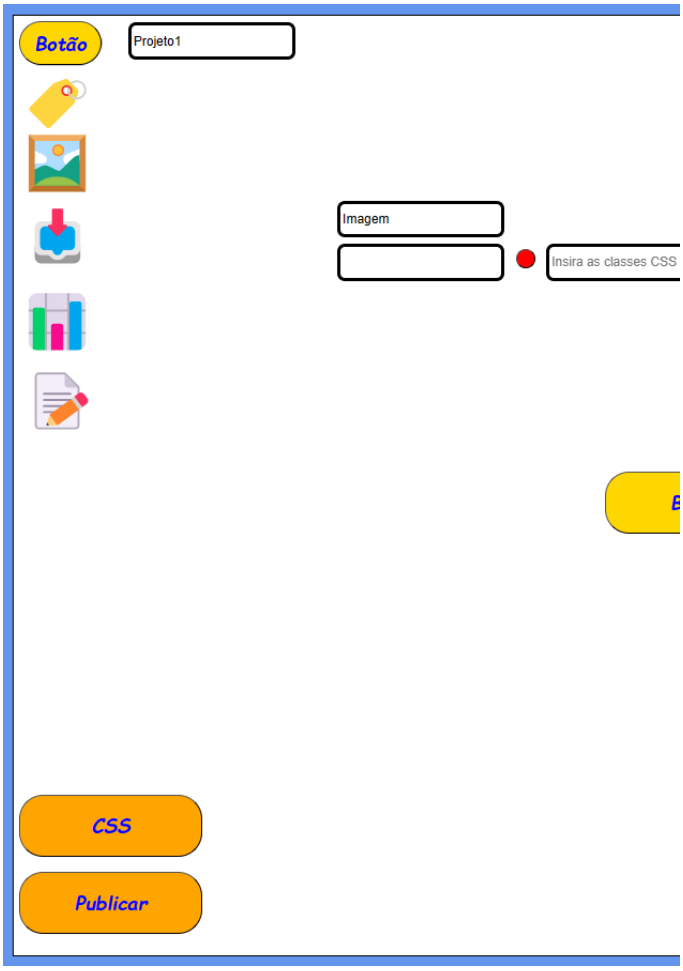



*Figura 14- Disposição dos botões das Widgets, de publicação, estilos e caixa de entrada do título*

Como é observável na figura 14, a disposição dos botões relativo aos estilos e publicação do projeto, assim como a caixa de texto de entrada do nome do projeto, onde aparece por defeito "Projeto1" e os restantes botões relativos às ferramentas que podem ser arrastadas para a área de desenvolvimento, encontravam-se numa disposição do lado esquerdo e tornava-se um pouco confuso na ótica do utilizador trabalhar desta forma, porque para além de ocupar algum espaço da área arrastável, também era confuso para o utilizador, distinguir as ferramentas que estavam arrastadas ou não. Principalmente a caixa de texto do título, ficando ao lado de outras caixas de texto, existe uma tendência para se pensar que esta caixa já foi arrastada e pertence ao ecrã do projeto que está a ser desenvolvido.

Outra característica interessante, é a cor de fundo da caixa de ferramentas, torna mais fácil de distinguir as ferramentas e destaca a área de desenvolvimento com uma separação tornando o ecrã mais limpo (figura 15).


*Figura 15-Estado atual dos botões de publicação, estilos, ferramentas e caixa de texto do título do projeto*

Com a nova mudança na estrutura da página de desenvolvimento, é possível esconder os botões e as ferramentas quando não estão a ser necessários para uso ou sempre que o utilizador pretenda, carregando no botão "Esconder opções", para esconder a caixa de texto do título do projeto e os botões "CSS" e "Publicar". Também é possível carregar no botão "Esconder widgets" para esconder as ferramentas que podem ser arrastáveis para a área de trabalho do utilizador.


*Figura 16- Botões para mostrar opções ou widgets*

Como é observável na figura 16, também é possível voltar a ver as opções como anteriormente carregando no botão "Mostrar opções" ou mostrar as ferramentas arrastáveis ao carregar no botão "Mostrar widgets".

Quando o botão "CSS" era carregado, o popup era aberto numa área fixa, o que dificultava a fase de implementação, pois ficava disposto ao nível das outras widgets arrastadas e tornava-se complicado para o utilizador trabalhar, quando já existem muitas widgets na área arrastadas. Então, a nova solução passou a ser, projetar o popup de forma centralizada na página bloqueando toda a área da página, exceto o popup e os dois botões "Guardar" e "Cancelar" (figura 17).

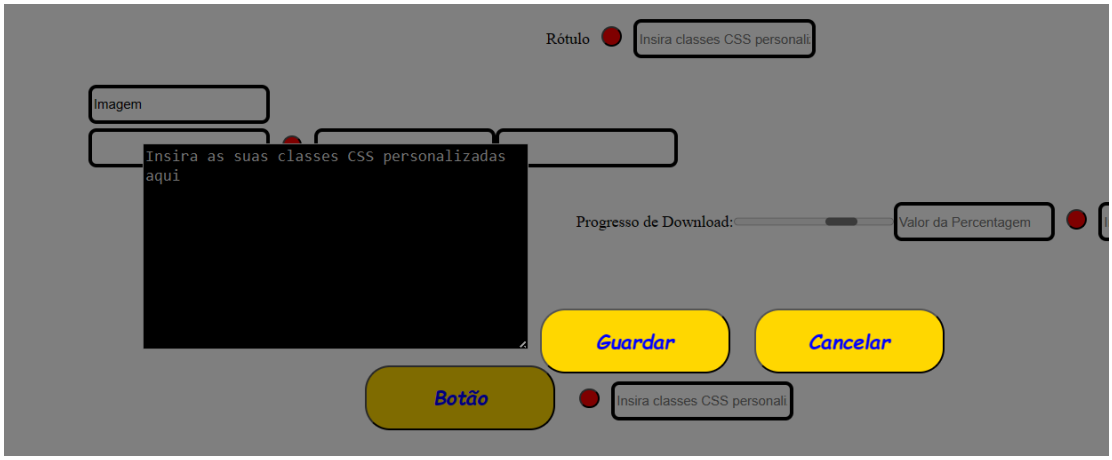



*Figura 17- Popup da folha de estilos "CSS"*

Com estas mudanças, o utilizador registou uma melhor experiência na utilização, sem qualquer tipo de limitações e uma interface mais agradável.

### 3.1.4. Teste da Aplicação

O IPSystems contou com uma equipa de 3 elementos que não são da área de *IT*, mas deram uma colaboração ao testar a aplicação e contribuíram com um *feedback* contínuo sobre a mesma, apontando bugs, melhorias e sugestões futuras. Os bugs quando apontados, são tratados de imediato e são prioritários no projeto, as sugestões de melhoria ou de novas ideias são apontadas com menor prioridade, embora todos os pontos sejam reconhecidos como um acréscimo de valor para a aplicação e tidos em conta para serem postos em prática em breve.

A aplicação de testes tem sido importante ao longo da fase de desenvolvimento do projeto, pois permite que ajustes sejam feitos, em alguns detalhes visuais ou de funcionalidades.

Um recurso importante utilizado durante esse processo foi a extensão *wave*, uma ferramenta amplamente reconhecida para análise de acessibilidade em páginas web. Essa extensão, desenvolvida pela *WebAIM* (*Web Accessibility In Mind*), permite que os desenvolvedores avaliem a conformidade do código com as boas práticas de acessibilidade estabelecidas por normas como as *Diretrizes de Acessibilidade para Conteúdo Web* (*WCAG*).

A extensão *wave* tem sido utilizada ao longo do desenvolvimento para analisar se a estrutura de código respeita todas as boas práticas e se a aplicação cumpre as boas práticas de acessibilidade, de modo a garantir que qualquer pessoa, mesmo que tenha alguma limitação física ou deficiência, consiga aceder de forma cómoda ao site sem nenhuma restrição.

### 3.1.5. Análise de Resultados

Foi definido desde o início do projeto um conjunto de testes funcionais e não funcionais para garantir que o sistema corresponde ao esperado. Os testes funcionais realizados foram o teste de *CRUD* para verificar se as operações básicas de criação, leitura, atualização e exclusão funcionam de forma correta. Também foram feitos testes de integração de APIs para validar que a comunicação entre o *backend* (Django) e o *frontend* ocorre sem erros e que os dados são enviados e recebidos de forma correta. Para além disso, foram feitos alguns testes de validação de dados para garantir que o sistema aceita entradas válidas, como por exemplo, o preenchimento de campos obrigatórios e por fim, foi feito o teste de navegação para garantir que o fluxo de redireccionamento das páginas a partir de *links* ou botões é feito de forma correta.

Os testes não funcionais realizados tiveram por base, testar o desempenho de páginas, quando feito um registo ou início de sessão, ou até mesmo quando se realiza a publicação de um projeto e é gerado o respetivo *deploy*. O teste de compatibilidade para perceber se o IPSystems é suportado em diferentes tipos de browsers, teste de usabilidade, tendo em conta o feedback de outros utilizadores que usaram o IPSystems e por fim, confirmar que o sistema permanece disponível quando existe uma falha no servidor.

A nível de testes funcionais, os testes são feitos no âmbito de averiguar se todas as funcionalidades estão de acordo com o esperado, isto inclui botões a redirecionar ou a completar uma ação específica, assim como as widgets que são arrastadas, ou o vídeo de um curso que é acedido. Foi tido em conta, conjuntamente a estas necessidades, uma aplicação com um padrão específico que seja consistente em todas as páginas do projeto e também uma preocupação constante com a experiência do utilizador no todo da aplicação, mas com maior destaque para a página IPSystems, a página de desenvolvimento.

Tendo em conta a necessidade de produzir uma boa experiência para o utilizador final, foi feito um plano com um grupo de 3 elementos para testar a aplicação de forma autónoma e dar *feedback* sobre a mesma de modo a confirmar se esta corresponde aos requisitos essenciais que foram planeados para o projeto e também com o intuito de receber um *feedback* contínuo deste grupo de pessoas que dão a opinião sobre a plataforma e deixam ideias de melhoria para concretizar.

### 3.1.5.1. Métricas Quantitativas

Para avaliar de forma mais objetiva a usabilidade do IPSystems para os utilizadores e também testar a consistência da plataforma, foram feitos alguns testes por parte de 3 elementos, que realizaram algumas tarefas na aplicação de forma autónoma. Ambos os elementos que realizaram os testes não são da área de IT e não possuem conhecimentos de programação.

Em seguida, podemos ver uma tabela de análise do desempenho dos 3 elementos que acederam à mesma, com uma avaliação das tarefas numa escala de 1 a 3, em que 1 significa que sentiu bastante dificuldade, o 2 significa que sentiu alguma dificuldade a realizar a tarefa, podendo demorar mais tempo, mas que conseguiu realizar autonomamente e 3 significa que realizou a tarefa rapidamente sem grandes problemas.

| *Métricas* | *Cátia* | *António* | *Bruno* |
|---|---|---|---|
| *Registo na aplicação* | *3* | *3* | *3* |
| *Iniciar Sessão* | *3* | *3* | *3* |
| *Aceder às páginas* | *3* | *3* | *3* |
| *Visualizar Cursos* | *2* | *3* | *2* |
| *Criar um projeto* | *2* | *2* | *2* |
| *Criar classes para os componentes* | *1* | *1* | *1* |

*Tabela 2- Análise de resultados dos utilizadores*

Com base na tabela 2, podemos concluir que os 3 utilizadores conseguiram aceder à aplicação de forma relativamente simples e desenvolver um projeto de forma autónoma. Inicialmente, na página da lista de cursos, não era tão intuitivo para os utilizadores o acesso a um curso específico, visto que a lista estava disposta de forma diferente da atual, que está organizada com uns cartões destacados e quando se passa o cursor do rato em cima aparece uma mão para indicar que é clicável (Figura 18).


*Figura 18 - Página Inicial de Cursos do IPSystems*

Na discussão de reporte de resultados, os utilizadores justificaram a avaliação mais baixa na categoria de criação de projeto e de classes para os componentes com as seguintes retóricas *"é muito intuitivo", "a página de cursos parece aquele tipo de páginas de leitura", "tive dificuldade a interpretar a função dos ícones de arrastar, acho que seria importante legendar as imagens dos ícones para clarificar a função de cada um"* e *"necessitei de ir pesquisar regras de código para iniciar um primeiro momento de implementação de texto"*. Esta última frase, confirma a dificuldade do utilizador em criar classes de CSS no projeto.

# 3.2. Desenvolvimento da aplicação

## 3.2.1. Desenvolvimento do Backend (Django)

Nesta seção, apresentamos os principais aspetos relacionados ao desenvolvimento do backend no projeto IPSystems, desde a configuração do ambiente até à implementação do modelo de dados e a lógica para as APIs, cada elemento desempenha um papel crucial no funcionamento e na eficácia do sistema.

### 3.2.1.1. Configuração do Ambiente

A configuração do ambiente de desenvolvimento foi a primeira fase a ser trabalhada no desenvolvimento do backend. O ambiente de desenvolvimento foi configurado utilizando o Python como linguagem principal por ter uma sintaxe clara e concisa e por ter sido escolhida como *framework* backend. O Django fornece a base de dados PostgreSQL para guardar os

dados necessários que são gravados no projeto. No início do desenvolvimento, foi criado um ambiente virtual denominado por venv, de modo a seguir todas as boas práticas de programação no Django, evitando problemas de versões dos pacotes e bibliotecas necessários ao longo do processo de desenvolvimento.

#### 3.2.1.2. Modelação de Dados

Os modelos de dados do projeto, estão projetados de forma a refletir as entidades principais do sistema, na criação de cursos, documentação, assim como o html e css gerados pelo utilizador para depois replicar na página novamente quando o projeto for continuado. Foram planeadas para cada uma das bases de dados, os atributos essenciais para serem projetados nos ecrãs e foi garantida a integridade referencial das tabelas, respeitando todas as boas práticas necessárias, garantindo assim não sobrecarregar as BDs com excesso de dados desnecessários e garantir melhor performance ao obter dados e em todo o tipo de ações de CRUD.

#### 3.2.1.3. Desenvolvimento de APIs REST

No desenvolvimento de API's, são detalhadas as visualizações, os serializadores para as respostas de JSON, que são dadas para o frontend, assim como os endpoints que são expostos para a utilização das ações no React. É importante também configurar o CORS (*Cross-Origin Resource Sharing*) de forma a permitir as solicitações entre domínios diferentes, neste caso, o domínio do frontend para conseguir ter acesso aos dados, tendo a responsabilidade de manter a segurança e integridade da aplicação.

#### 3.2.1.4. Alterações no Backend

No servidor foram acrescentadas algumas ações mediante as decisões tomadas na aplicação, principalmente no frontend, foi criado um backoffice que passou a ter uma página com uma lista de cursos e outra para editar, excluir ou criar um curso. Assim como, a criação de uma tabela da documentação para ser exibida no site do IPSystems e a descrição da mesma. Para isso, foi necessário separar as API's utilizadas para o frontend em React e as utilizadas no backoffice do Django, para estruturar melhor o código e ser mais percetível na fase de implementação e desenvolvimento.

### 3.2.2. Integração com o Frontend

Para integrar os endpoints do backend com o frontend, houve um trabalho inicial no arquivo **next.config.js** para configurar algumas permissões do **Next.js** de forma a existir a permissão para fazer as solicitações ao backend Django, como mostrado na imagem que se segue.

```
  headers: [
      { key: 'Access-Control-Allow-Credentials', value: 'true' },
      { key: 'Access-Control-Allow-Origin', value: '*' },
                              { key: 'Access-Control-Allow-Methods', value:
'GET,OPTIONS,PATCH,DELETE,POST,PUT' },
       { key: 'Access-Control-Allow-Headers', value: 'X-CSRF-Token, X-Requested-With,
Accept, Accept-Version, Content-Length, Content-MD5, Content-Type, Date, X-Api-
Version, Authorization' },
      ],
    },
```

*Figura 19- Configurações de acesso*

Na primeira linha, 'Access-Control-Allow-Credentials' definida como 'true', indica que as credenciais podem ser incluídas. Já na segunda linha, 'Access-Control-Allow-Origin', com o valor '*', indica que existe permissão de qualquer domínio, o que no ambiente de desenvolvimento não é tão impactante. Agora na terceira linha, 'Access-Control-Allow-Methods', lista os métodos HTTP que estão permitidos a fazer solicitações cross-origin, neste caso, (GET, OPTIONS, PATCH, DELETE, POST, PUT) que abrangem as operações CRUD necessárias no frontend do projeto.

Por fim, a quarta linha, 'Access-Control-Allow-Headers', inclui uma lista referente aos cabeçalhos HTTP que podem ser incluídos nas solicitações.

### 3.2.3. Alterações no Frontend

O projeto foi influenciado automaticamente, pelas alterações visuais a nível do esboço dos mockups, na medida que foi reconstruída a página de desenvolvimento do utilizador, assim como acabou por ser reformulada a página de projetos que antecede esta página de desenvolvimento. O objetivo é o utilizador ter acesso à página de projetos e proceder às alterações ou continuação do desenvolvimento da mesma de forma a retomar o trabalho já iniciado.

Também foi pensada posteriormente, a página de cursos para mostrar ao utilizador vídeos de aulas sobre a plataforma IPSystems e explorarem mais sobre a mesma. A ideia, é dar a conhecer ao utilizador que programa com IPSystems os padrões para programar da melhor forma possível em IPSystems e adotarem as melhores decisões na fase de desenvolvimento de um projeto, a nível de arquitetura do projeto, boas práticas de frontend, boas práticas de backend, noções básicas, intermédias e avançadas da plataforma, entr outro tipo de conteúdos.

### 3.2.4. Arquitetura e estrutura

O desenvolvimento deste projeto é feito com base nas *frameworks* de React para o frontend e Django para o backend. O desenvolvimento no frontend é feito tendo por base uma abordagem simples conhecida com arquitetura baseada em componentes, tendo em conta que as páginas e folhas de estilos são desenvolvidas de raiz, o que torna mais direto o processo de desenvolvimento, face ao tamanho atual do IPSystems e considerando ser uma versão leve que permite ao utilizador trabalhar diretamente no site, sem ter de fazer o download de um programa executável.

A estrutura geral da aplicação no frontend está organizada mais ou menos da seguinte forma:

- **Componentes**: São utilizados alguns componentes, como algumas páginas de popup que servem para ser reutilizadas em outras páginas, por exemplo, na página de desenvolvimento do projeto do IPSystems, chamada IPSystems, é utilizado um popup que serve para adicionar algumas classes CSS e foi feito com o intuito de separar o grande volume de código, para além de ser uma das boas práticas no React, pois permite usar umas das suas principais capacidades que é a reutilização.
- **Páginas**: São utilizadas as páginas necessárias padrão e em alguns casos é feita a integração com alguns componentes, como a barra de navegação do topo do site que serve para ser reutilizada em outras páginas. Neste caso, acaba por estar presente em quase todos os ecrãs.
- **Estilos**: O IPSystems está padronizado com base nas classes definidas no index.css, que é a folha principal dos estilos neste projeto desenvolvidos com React, seguindo assim, as boas práticas de estilização e padronização do mesmo. Existem ainda outras páginas de CSS com menos classes, que são a App.css, bar.css que definem algumas

classes específicas da barra de navegação e Begin.css que define algumas animações, especialmente da homepage, embora também possua outras classes específicas.

- **Estado local**: São utilizados alguns hooks na página IPSystems com o objetivo de gerir o estado local dos componentes utilizados, realizar operações assíncronas, assim como efeitos colaterais e criar referências a elementos do DOM.
    - O ‘useState’ é utilizado na página para declarar variáveis de estado no componente, cada vez que é feita uma chamada a este hook, é retornado um par de valores, o estado atual e uma função para atualizá-lo, mais especificamente na página, está a ser usado para gerir os estados do ‘publishedTitles’, ‘userDefinedTitle’, ‘draggingElement’, ‘elements’, entre outros.
    - Já o ‘useEffect’ permite realizar efeitos colaterais nos componentes funcionais que estão a ser usados, como a busca dos dados da API, quando o componente é montado, o ‘projectIdParam’ muda. Este hook também é responsável por atualizar o estado ‘projectHtml’ quando o projeto é encontrado.
    - Por fim, o ‘useRef’ serve para criar uma referência mutável que pode ser associada a elementos do DOM ou valores que precisam persistir entre renderizações sem acionar uma rerenderização, como no caso do ‘fileInputRef’, que é uma referência associada a um elemento de input de arquivo ‘<input type=”file” />. Desta forma, a aplicação consegue ter uma melhor performance e o código mais bem estruturado.

Já no desenvolvimento de backend na *framework* Django, está a ser utilizada uma arquitetura de padrão Model-View-Controller (MVC). A arquitetura no backend encontra-se distribuída da seguinte forma:

- **Model (Modelo):** No arquivo models.py estão definidos alguns modelos que representam objetos das bases de dados, como ‘Categoria’, ‘Documentacao’, ‘Curso’, entre outros.
- **View (Visualização):** O código relacionado com as visualizações do projeto está situado no arquivo ‘views.py’ como exemplo, tem-se ‘get_html’, ‘cursos’, ‘get_pages’, entre outros, cuja sua responsabilidade é processar solicitações HTTP, interagir com os modelos conforme necessário e retornar respostas HTTP, para obter dados da API REST, ou nos templates de frontend no próprio Django, relativos ao backoffice da aplicação.

- **Serializer (Serializador):** A *framework* está a lidar com serializers para traduzir objetos complexos, como modelos Django, em representações que podem ser facilmente convertidas em JSON, como é o caso do 'get_pages'. Os serializers têm a função de serializar/deserializar os dados ao interagir com a API REST.
- **URLs (URLs):** A configuração de URLs está a ser tratada no arquivo 'urls.py' que mapeia os padrões de URL para as visualizações correspondentes.
- **Rotas (Routes):** O Código relacionado à configuração das rotas para as APIs está definidos no arquivo 'urls.py', definidosos endpoints das APIs e a respetiva associação às visualizações relevantes.
- **Settings (Configurações):** No arquivo 'settings.py' estão todas as configurações necessárias para o IPSystems relacionadas a esquemas de bases de dados, autenticação, aplicações instaladas e outras opções de configuração.
- **Migrações:** As migrações são usadas para gerir qualquer alteração que seja feita ao nível das BDs, de forma a manter as BDs sincronizadas com a estrutura do modelo.

### 3.2.4.1. Diagrama de Workflow

O IPSystems foi estruturado, de forma a garantir uma boa comunicação entre o frontend (React), o backend (Django) e as respetivas bases de dados integradas em SQLite, como podemos observar na figura 20.

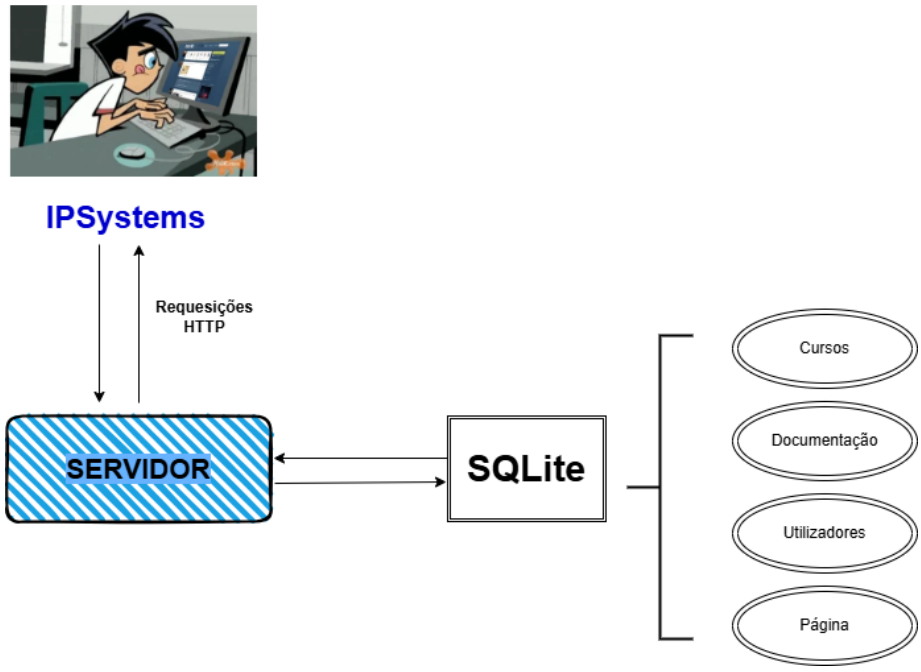



*Figura 20 - Diagrama de Workflow do IPSystems*

O diagrama de workflow, tem em conta o cliente que é o utilizador da plataforma e interage diretamente com o sistema. O utilizador consegue realizar ações como a criação de páginas dinâmicas, consulta de documentação e manipulação de dados a partir da interface.

Ao ser efetuada uma ação por parte do utilizador, o frontend que está construído com React, comunica com o servidor de Django tendo por base requisições HTTP, que podem incluir operações de criar, atualizar, consultar ou até eliminar dados. O servidor, processa as requisições e atua como intermediário entre o IPSystems e a base de dados SQLite, que contém tabelas diferentes relacionadas, como cursos, documentação, utilizadores e páginas.

Este diagrama demonstra uma arquitetura simples e intuitiva, garantindo a escalabilidade de aplicações robustas, apesar de inicialmente apostar numa base de dados menos robusta. E também é feita uma forte aposta na facilidade de uso, com um fluxo simples e direto de comunicação entre o cliente e o sistema.

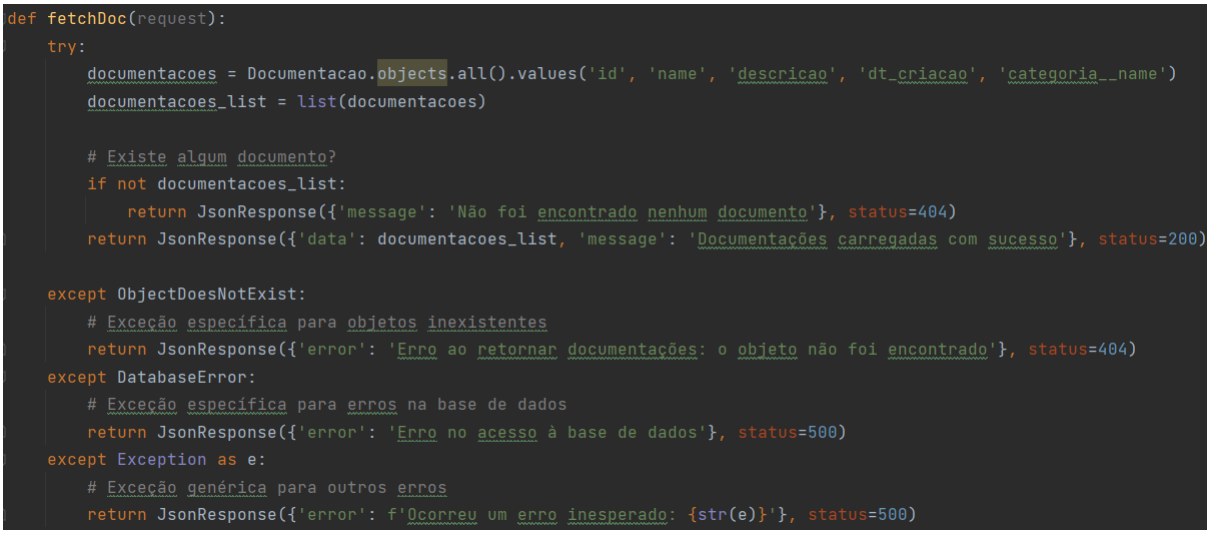
### 3.2.5. Detalhes técnicos de implementação

Ao longo do desenvolvimento do projeto, foi necessário garantir boas práticas de programação e estabelecer um padrão consistente tanto no *frontend* quanto no *backend*. Esta abordagem, ajudou a garantir a organização do código, a sua legibilidade e manutenção, auxiliando à diminuição de erros na aplicação.

#### 3.2.5.1. Detalhes técnicos do *backend*

Durante o desenvolvimento do *backend* em *Django*, os detalhes técnicos, foram cuidadosamente planeados para garantir um desempenho eficiente, segurança e escalabilidade, assegurando um código limpo e percetível.

Na construção do CRUD, foram implementadas boas práticas, como mostra a figura abaixo numa operação de *Read* (leitura), que permite a obtenção dos dados relativamente à documentação existente.



```python
def fetchDoc(request):
    try:
        documentacoes = Documentacao.objects.all().values('id', 'name', 'descricao', 'dt_criacao', 'categoria__name')
        documentacoes_list = list(documentacoes)

        # Existe algum documento?
        if not documentacoes_list:
            return JsonResponse({'message': 'Não foi encontrado nenhum documento'}, status=404)
        return JsonResponse({'data': documentacoes_list, 'message': 'Documentações carregadas com sucesso'}, status=200)

    except ObjectDoesNotExist:
        # Exceção específica para objetos inexistentes
        return JsonResponse({'error': 'Erro ao retornar documentações: o objeto não foi encontrado'}, status=404)
    except DatabaseError:
        # Exceção específica para erros na base de dados
        return JsonResponse({'error': 'Erro no acesso à base de dados'}, status=500)
    except Exception as e:
        # Exceção genérica para outros erros
        return JsonResponse({'error': f'Ocorreu um erro inesperado: {str(e)}'}, status=500)
```

*Figura 21 - Operação de leitura dos documentos existentes*

Na figura indicada, podemos perceber que foi utilizado um bloco *try-except* que permite obter *feedback* da operação, neste caso, se foi ou não bem-sucedida e permite a qualquer programador no mundo perceber o resultado da mesma, isto porque os códigos do status HTTP, seguem um padrão global definido pela RFC 9110 para uso de todos os programadores a nível global. Caso, o resultado seja 404, por exemplo, indica que o recurso indicado não foi encontrado, mas se retornar um erro 200, significa que o requisito foi bem-sucedido.

Dentro das exceções, ainda é possível adicionar mais detalhe, a partir de erros comuns, como *ObjectDoesNotExist* e *DatabaseError* que são tratados separadamente, de modo a facilitar a captura de erros e facilita a resolução de problemas. Nos casos em que existe outro tipo de exceção que não seja prevista, é apresentada de forma genérica ao utilizador a informar que ocorreu um erro inesperado.

Para obter os dados relativamente à documentação, é utilizado o método *values*() na função para retornar apenas os campos necessários da base de dados, desta forma, é possível otimizar a consulta e evitar a sobrecarga de dados desnecessários. A conversão do *QuerySet* para uma lista baseado no método *list*(), permite manipular os dados de forma eficiente antes de retorná-los como um JSON.

Apesar desta operação ser de leitura, é importante prevenir erros, fazendo uma validação explícita dos resultados e evitar o envio de respostas incompletas ou inesperadas, evitando que a API retorne dados mal estruturados, como campos vazios ou mal formatados e evitar falha na aplicação cliente que consome esta operação e previne falhas de execução e problemas no layout. Ao converter o *QuerySet* para uma lista, é possível realizar operações adicionais para manipular e validar os dados antes de retornar o resultado, com base nesta conversão os dados

são enviados para o cliente no formato correto e evita inconsistências, como valores vazios ou inesperados, e evita a injeção de dados maliciosos como o *SQL Injection* ou *Cross/Site Scripting (XSS)*.

Todas estas operações foram tidas em conta de acordo com *OWASP* (*Open Web Application Security Project*) *Top 10* que se trata de um relatório atualizado regularmente e que descreve questões de segurança que devem ser tidas em conta no desenvolvimento de aplicações web, com foco nos 10 riscos mais críticos.

Com o código organizado e documentado, torna-se mais fácil perceber o fluxo e a lógica da aplicação que tende a ser mais complexa de interpretar de acordo com o aumento da complexidade do projeto ao longo do tempo.

Durante a fase de criação de uma página HTML no projeto, quando o utilizador seleciona a opção de publicar a página que está a ser construída, o sistema gera automaticamente um identificador único (UUID) para essa página. O UUID (gerado através da função uuid4()) assegura que cada página publicada tenha um identificador exclusivo, eliminando possíveis conflitos de identificação, mesmo em casos de múltiplas publicações ou ambientes distribuídos.

```
uuid = uuid4()
page = Pages.objects.create(
    uuid=str(uuid),
    type=type,
    text=text,
    left=left,
    top=top,
    custom_class=custom_class,
    custom_css=custom_css
)

return JsonResponse({'success': 'http://localhost:8000/ipsystems/?id=' + str(uuid)})
```

*Figura 22 - Geração de um uuid e respetivo URL da acesso à página*

Conforme ilustrado na Figura 22, o UUID gerado é então associado aos detalhes da página na base de dados, a partir da criação de registo na tabela **Pages**, que armazena informações como o tipo da página (type), o texto (text), as coordenadas de layout (left e top), além das classes personalizadas (custom_class) e estilos CSS adicionais (custom_css).

Após a criação bem-sucedida na base de dados, o sistema retorna uma resposta no formato JSON contendo o link para a nova página publicada. O URL gerado inclui o UUID como parâmetro de consulta, garantindo que a página seja acedida de forma única.

### 3.2.5.2. Detalhes técnicos do *frontend*

Para o desenvolvimento de *frontend* no *React*, foram adotados alguns padrões e boas práticas, de modo a organizar melhor o código e torná-lo mais percetível com um padrão existente e garantir a sua eficiência na aplicação.

Uma das práticas adotadas é a utilização do *hook useEffect* que lida com efeitos colaterais nos comportamentos funcionais em situações como, chamadas de uma API, interação com o DOM ou execução de eventos de forma assíncrona.

```
useEffect(() => {
  const searchParams = new URLSearchParams(window.location.search);
  const projectIdParam = searchParams.get('uuid');
  console.log(projectIdParam);
  setProjectIdParam(projectIdParam);
```

*Figura 23- Uso do useEffect*

Como é possível observar na figura anterior, o código dentro do *useEffect* é executado sempre que se regista uma mudança no parâmetro *projectIdParam*, o que evita requisições desnecessárias e melhora o desempenho da aplicação.

Neste exemplo de código, o parâmetro *uuid* é extraído da URL usando o *URLSearchParams*, para aceder à informação passada. Esta prática, evita a manipulação manual da *string* da URL e evita a geração de erros.

```
const fetchProjectHtml = async (projectId) => {
  try {
    const response = await fetch(`http://localhost:8000/api/pages/?id=${projectId}`);
    if (response.ok) {
      const responseData = await response.json();
      console.log('Response Data:', responseData);
```

*Figura 24- Tratamento de Exceções, chamadas assíncronas e tratamento de erros*

Outro dos cuidados e boas práticas tidas em conta no IPSystems foram, o tratamento de exceções *try-except* semelhante ao utilizado no *backend* em *Django*, para garantir que todas as

exceções são tratadas de forma correta. Para além de, fazer uma chamada assíncrona a partir do fetch para obter os dados da página do projeto HTML com base no parâmetro projectId e verifica se a resposta foi bem-sucedida, a partir de response.ok, desta forma o utilizador no frontend, consegue perceber se a operação foi realizada com sucesso ou não e obter esse *feedback*.

```
    } else {
      console.error('Failed to fetch HTML. Status:', response.status);
      const responseData = await response.text();
      console.error('Response data:', responseData);
    }
  } catch (error) {
    console.error('Erro durante a solicitação HTTP:', error);
  }
```

*Figura 25 - Tratamento dos erros de validação e exceção*

Em caso de erro na validação, será possível observar o estado do erro para melhor *feedback* do utilizador, de modo a perceber se o erro estará relacionado por exemplo, com a base de dados, com o objeto ou se é um erro genérico. Caso haja, um erro de exceção, também será possível obter *feedback* do mesmo.

### 3.2.6. Conclusão do Capítulo

Este capítulo proporcionou uma análise do projeto na sua totalidade, desde a avaliação do mercado e dos concorrentes até às decisões estratégicas tomadas em relação à plataforma, tecnologias adotadas e metodologias escolhidas, todas com o objetivo de impulsionar o sucesso do projeto. A importância desta análise reside na compreensão global da aplicação e no entendimento do planeamento realizado, permitindo avaliar como as decisões podem impactar positivamente ou não o mercado.

Ao examinar o mercado e os concorrentes, foi possível identificar oportunidades e desafios que influenciaram as diretrizes adotadas no desenvolvimento do projeto. Isso incluiu uma análise detalhada das necessidades dos utilizadores, lacunas no mercado e estratégias competitivas.

As decisões relacionadas à plataforma e tecnologias foram fundamentais para definir a arquitetura do sistema, garantindo escalabilidade, desempenho e segurança adequados. Além disso, a escolha das metodologias usadas no desenvolvimento deste projeto, teve como objetivo

otimizar a colaboração entre o professor e o aluno e garantir melhor qualidade no processo de desenvolvimento.

Esta análise geral proporciona uma visão panorâmica do projeto, destacando a sua posição no mercado, os seus pontos fortes relativamente aos competidores e os desafios enfrentados. Compreender estes aspetos é essencial para orientar o desenvolvimento futuro, adaptando-se às procuras e necessidades do mercado e maximizando as chances de sucesso e de colmatar possíveis carências.

# Capítulo 4 - Discussão

O IPSystems, consegue garantir que o utilizador que esteja registado na aplicação, consiga aceder à mesma e desenvolver uma página web com sucesso, estruturando a mesma e estilizando da forma que desejar, garantindo a sua integridade e robustez. Desta forma, é possível ficar com a página gravada na lista de projetos e mais tarde continuar a desenvolver, assim como aceder à página de cursos da aplicação e trabalhar sobre os mesmos.

O utilizador consegue de forma independente, construir um projeto e com menos esforço em comparação à utilização de uma *framework* e garantindo a qualidade da mesma, isso é uma mais-valia para acelerar o processo de desenvolvimento com menor esforço e conhecimento de linguagens de sintaxe ou programação.

Apesar de todas estas mais valias, existem algumas limitações e sugestões futuras que vamos abordar neste capítulo, para implementar no projeto de forma a tornar-se mais complexo e responder de forma mais eficiente e robusta ao utilizador.

## 4.1. Principais desafios

Neste capítulo são abordados todos os tópicos que foram mais difíceis de ultrapassar ao longo do processo de desenvolvimento do projeto, na medida em que exigiram maior pesquisa e uma grande força de vontade para se conseguir ultrapassar, algo que não estava funcional numa primeira instância.

### 4.1.1. Comunicação das APIs

No início do projeto, houve alguns problemas na comunicação de APIs e respetivo consumo, exigindo uma investigação aprofundada para perceber a melhor abordagem de trabalhar as configurações necessárias tanto do lado do backend, como do frontend. A necessidade de estabelecer uma comunicação eficiente entre esses componentes impulsionou um esforço substancial de pesquisa para otimizar as configurações necessárias.

### 4.1.2. Deploys automáticos

A publicação de uma página trouxe consigo a necessidade crucial de preservar o HTML gerado e o respetivo CSS. Além disso, a geração automática de links ao abrir uma nova janela

emergiu como uma consideração essencial. Essa complexidade representou uma das principais preocupações do projeto, abordada através de um árduo trabalho de pesquisa e esforço, resultando na implementação bem-sucedida de deploys automáticos.

## 4.2. Segurança e escalabilidade

Ao longo do processo de desenvolvimento, é fundamental não apenas garantir o funcionamento adequado das funcionalidades, mas também assegurar que a mesma seja segura. Sobretudo, se a mesma for destinada a uso público e com interação dos utilizadores. Para isso, foi importante assegurar na fase de publicação de cada projeto, medidas de segurança robustas para proteger dados sensíveis, assim sendo, quando é gerada uma nova página, é gerado um link encriptado com um identificador único, de forma a tornar mais complexo, o acesso à página, como podemos ver na figura abaixo.


*Figura 26 - Identificador único de geração da página*

Além disso, é necessário estabelecer formas para garantir a privacidade da conta de cada utilizador com um sistema de autenticação forte que exige que o utilizador introduza o email e password, de forma a bloquear acesso a quem introduza os dados de email ou password, ou até mesmo ambos de forma incorreta. Assim, é possível garantir um acesso restrito para os utilizadores que estão registados na plataforma e garantir a sua privacidade.

## 4.3. Sugestões Futuras

É importante analisar de forma concisa o protótipo IPSystems e pensar nas melhorias que podem ser adotadas de forma a colocar este produto no mercado e fazer do mesmo, uma marca de sucesso.

### 4.3.1. Implementação de Recursos Adicionais

Para melhorar a experiência de utilizador e fornecer um maior leque e flexibilidade na fase de implementação do layout das páginas, deve ser feita a expansão de mais widgets no futuro, introduzir novos tipos de elementos como gráficos interativos, controladores de formulários avançados e outro tipo de elementos que enriqueçam a ferramenta. Esta expansão deverá ser realizada na mesma dinâmica de implementação já existente, sem o recurso de bibliotecas ou componentes já existentes. Desta forma, será possível cumprir mais facilmente

o objetivo de padrão único do IPSystems por defeito e com a própria lógica toda implementada, não dispensando a possibilidade de alterações por parte do utilizador ao estilizar da sua forma todos os seus componentes e implementar a sua própria lógica.

Uma das características também pensadas para breve, é melhorar a customização e as funcionalidades da aplicação, oferecendo aos utilizadores, a capacidade de criar a sua própria lógica de backend de uma forma mais facilitada, por exemplo, adicionar validações e atribuição de valores a variáveis da página que permitam gerir o comportamento padrão de um botão ou outro widget.

#### 4.3.1.1. Maior número de páginas de desenvolvimento

A adição de páginas permitirá ao utilizador expandir a quantidade de páginas disponíveis para desenvolvimento dentro do projeto. Este ponto específico destaca a importância de fornecer um ambiente flexível e expansível para os desenvolvedores trabalharem, permitindo-lhes criar e integrar uma variedade de páginas com diferentes funcionalidades e conteúdos.

Ampliar o número de páginas de desenvolvimento oferece uma série de vantagens significativas, incluindo:

- **Diversificação de Funcionalidades:** Ao disponibilizar mais páginas, os desenvolvedores têm a liberdade de implementar uma variedade maior de funcionalidades e recursos no projeto, atendendo a diferentes necessidades e requisitos dos utilizadores.
- **Personalização e Adaptação**: Uma maior variedade de páginas permite uma maior personalização e adaptação do projeto às preferências e procuras específicas dos utilizadores, oferecendo uma experiência mais rica e personalizada.
- **Escalabilidade:** A capacidade de adicionar mais páginas de desenvolvimento promove a escalabilidade do projeto, permitindo que o mesmo cresça e se adapte conforme necessário para lidar com novos requisitos e cenários de uso.
- **Exploração Criativa:** Com mais páginas disponíveis, os desenvolvedores têm mais espaço para explorar a sua criatividade e inovação, experimentando novas ideias e conceitos no projeto.

### 4.3.2. Melhorar a Documentação

A documentação é um dos elementos mais importantes em qualquer plataforma, pois auxilia os utilizadores a ter um melhor conhecimento da mesma e a dar *feedback* para o

desenvolvimento contínuo e melhoria do IPSystems. Para além disso, existem outros benefícios que são, promover as boas práticas de desenvolvimento, aumento da produtividade e da satisfação do utilizador.

A documentação deverá contemplar as seguintes alternativas:

1. **Guia do desenvolvedor:** O Guia do Desenvolvedor representa uma compilação meticulosa de informações essenciais, organizadas para fornecer uma compreensão abrangente e profunda da arquitetura subjacente do projeto. Este compêndio delineia não apenas a estrutura hierárquica do projeto, mas também delineia as suas dependências intricadas, inculcando assim uma visão unificada e coerente para os colaboradores. Além disso, este guia não se limita meramente à disposição estrutural do código, mas transcende para abordar normas de codificação que são fundamentais para a consistência e legibilidade do código-fonte, bem como para a sustentabilidade do projeto a longo prazo.
2. **Exemplos de uso:** Os Exemplos de Uso representam uma ferramenta pedagógica de inestimável valor, oferecendo um contexto prático e tangível para os desenvolvedores. Estas demonstrações perspicazes são habilmente concebidas para encapsular cenários de utilização variados, abarcando desde os casos de uso mais elementares até as aplicações mais sofisticadas, proporcionando assim uma compreensão profunda do projeto.
3. **FAQs e Resolução de problemas:** A seção dedicada a FAQs e Resolução de Problemas serve como um repositório de conhecimento dinâmico, destinado a abordar as consultas mais prementes e as dificuldades técnicas mais proeminentes enfrentadas pelos desenvolvedores ao longo do ciclo de desenvolvimento e implementação. Neste compêndio, são oferecidas soluções meticulosamente articuladas para desafios comuns, juntamente com explicações concisas e esclarecedoras, destinadas a capacitar os desenvolvedores a superar obstáculos de forma independente e eficaz.
4. **Atualizações e Notas de Versão:** As Atualizações e Notas de Versão representam um registo cronológico meticuloso de todas as modificações iterativas, refinamentos incrementais e melhorias evolutivas implementadas em cada iteração do projeto. Este arquivo detalhado não apenas documenta alterações tangíveis no código-fonte, mas também contextualiza essas alterações dentro do escopo mais

amplo do projeto, fornecendo uma narrativa coesa e contínua da evolução do mesmo ao longo do tempo.

### 4.3.3. Fórum Ativo

A intenção de ter um fórum virtual vibrante e dinâmico, concebido como um espaço interativo destinado à comunidade de desenvolvedores, onde ideias são trocadas, discussões são fomentadas e colaborações são incubadas. Este fórum surge como um centro pulsante de atividade intelectual, onde os membros da comunidade se reúnem para partilhar conhecimento, explorar conceitos emergentes e resolver desafios técnicos de forma colaborativa.

Sob a égide do Fórum Ativo, os desenvolvedores são incentivados a iniciar e participar de discussões sobre uma ampla gama de tópicos relacionados ao projeto, desde questões de implementação técnica até estratégias de otimização de desempenho. Esta plataforma interativa não apenas promove a disseminação de informações essenciais, mas também cultiva um ambiente de aprendizagem contínua e troca de ideias, onde os membros da comunidade são encorajados a contribuir com perspetivas únicas e experiências pessoais.

Além disso, o Fórum Ativo serve como um recurso inestimável para a resolução colaborativa de problemas, permitindo que os desenvolvedores partilhem soluções para desafios técnicos comuns, forneçam *feedback* construtivo sobre questões específicas do código e colaborem na identificação e resolução de bugs. Esta abordagem colaborativa para a resolução de problemas não apenas acelera o processo de desenvolvimento, mas também fortalece os laços dentro da comunidade de desenvolvedores, promovendo um espírito de camaradagem e colaboração.

Em última análise, o Fórum Ativo representa não apenas uma ferramenta prática para a troca de conhecimento e a resolução de problemas, mas também um reflexo do ethos colaborativo e inclusivo que permeia o projeto como um todo. Ao fornecer um espaço dedicado para a interação entre os membros da comunidade, o Fórum Ativo desempenha um papel crucial na promoção de um ambiente de desenvolvimento saudável e sustentável, onde a inovação floresce e o progresso é impulsionado pela colaboração coletiva.

### 4.3.4. Simplificar as traduções linguísticas

Esta proposta de simplificação das traduções linguísticas, trata-se uma abordagem mais acessível e intuitiva para lidar com a localização do projeto em diferentes idiomas. Ao invés de depender de métodos convencionais, como o uso de arquivos JSON para armazenar traduções, a ideia é implementar um sistema semelhante ao encontrado na plataforma OutSystems.

Neste contexto, os utilizadores seriam agraciados com um recurso de seleção de idiomas incorporado, apresentado de forma elegante e intuitiva por meio de um popup cuidadosamente elaborado. Este componente ofereceria aos utilizadores uma ampla gama de opções de idiomas, permitindo-lhes selecionar a linguagem que queriam implementar para além da linguagem padrão. Por exemplo, o inglês e fazer a tradução das palavras desejadas que tinham implementadas nas suas páginas, como exemplo, o português e criar de maneira rápida e eficiente, sem a necessidade de manipular arquivos de configuração complexos ou passar por processos manuais de tradução.

### 4.3.5. Consumir serviços

A capacidade de Consumo de Serviços REST, SOAP, SAP, entre outros, será uma necessidade com extrema importância no IPSystems, para permitir integrar diversos serviços externos no âmbito do projeto. Isso implica a capacidade de consumir APIs RESTful, serviços SOAP e interagir com sistemas corporativos como SAP, entre outras integrações necessárias para o funcionamento e aprimoramento do projeto.

Neste contexto, a implementação de mecanismos robustos de integração é essencial para permitir a comunicação eficiente e confiável com uma variedade de serviços e sistemas externos. Isso envolve:

- **Consumo de APIs RESTful**: As APIs RESTful são amplamente utilizadas para troca de dados entre sistemas heterogéneos e são uma abordagem popular para desenvolver e integrar sistemas distribuídos na web, e utilizam os métodos HTTP padrão (GET, POST, PUT, DELETE) para realizar operações CRUD em recursos identificados pelas URLs. Ao integrar o projeto com serviços que disponibilizam APIs RESTful, é necessário desenvolver componentes de software conhecidos como clientes REST. Esses clientes são responsáveis por interagir com os endpoints fornecidos pela API, enviando solicitações HTTP apropriadas e processando as respostas retornadas pelo serviço.

- **Interação com Serviços SOAP**: Interagir com os serviços SOAP, ou Simple Object Access Protocol, é uma necessidade em determinados cenários de integração de sistemas. Esse protocolo, baseado em XML, é bastante utilizado para troca de mensagens entre sistemas distribuídos na web. Para viabilizar essa interação no projeto, é preciso desenvolver componentes de software conhecidos como clientes SOAP.
    - O processo de interação com serviços SOAP segue uma sequência de etapas. Em primeiro lugar, é necessário criar um cliente SOAP, que será responsável por gerar e enviar solicitações SOAP para os serviços externos. Esse cliente é geralmente criado utilizando ferramentas ou bibliotecas específicas da linguagem de programação utilizada, e é baseado na descrição do serviço (WSDL - Web Services Description Language).
- **Integração com Sistemas Corporativos como SAP**: Em ambientes corporativos, é comum integrar o projeto com sistemas empresariais, como o SAP (Systems, Applications, and Products in Data Processing). Isso pode envolver a troca de dados com módulos SAP, como o SAP ERP (Enterprise Resource Planning), SAP CRM (Customer Relationship Management) e outros, para realizar operações como consulta de dados, criação de pedidos, atualização de registos, entre outros.
- **Gestão de Autenticação e Autorização**: Ao interagir com serviços externos, é crucial implementar mecanismos de autenticação e autorização adequados para garantir a segurança e a privacidade dos dados transmitidos entre o projeto e os sistemas externos. Temos como exemplos:
    - Autenticação com Token JWT (JSON Web Token): Quando um utilizador faz login no projeto, o backend gera um token JWT contendo informações de autenticação, como o seu ID e o tempo de expiração. Esse token é então enviado para o frontend e incluído em todas as solicitações para os serviços externos como parte do cabeçalho HTTP Authorization. Os serviços externos validam o token JWT recebido, verificando a sua assinatura e o tempo de expiração, para autenticar o utilizador.
    - Autorização Baseada em Funções: Cada utilizador tem uma permissão específica, como "administrador", "editor" ou "utilizador padrão". Os serviços externos consultam o backend do projeto para validar essas permissões, e verificar se o utilizador está autorizado a aceder aos recursos solicitados.

- Autenticação OAuth para Integração com APIs de Terceiros: O projeto permite a autenticação através de provedores de identidade externos, como Google, Facebook ou GitHub, por meio do protocolo OAuth. Após a autenticação bem-sucedida, o projeto recebe um token de acesso OAuth, que é usado para aceder aos recursos protegidos do provedor de identidade. Esse token OAuth é incluído nas solicitações aos serviços externos como parte do cabeçalho de autorização, permitindo o acesso aos dados do utilizador fornecidos pelo provedor de identidade.

O consumo de serviços REST, SOAP, SAP e outros será uma mais-valia para o IPSystems, de modo a poder estender as suas funcionalidades e capacidades, permitindo integrações com mais sistemas e serviços externos. Isso aumentará a capacidade de atender às necessidades específicas do negócio e ter o IPSystems como principal solução.

### 4.3.6. Integração com Bases de Dados

A integração com bases de dados é fundamental e indispensável para o projeto para além de ser uma característica crucial na área de desenvolvimento, deve ser garantida a persistência e manipulação adequada dos dados. Uma das funcionalidades a implementar deverá ser a possibilidade de o utilizador criar de forma prática e manual, sem recursos a comandos específicos e sem a necessidade de fazer integrações, a implementação de bases de dados com alguns atributos à sua escolha, onde poderá definir o tipo de dados, valor de defeito e até mesmo o tamanho de caracteres suportados.

Será também importante existir uma secção onde poderá ser feito o ORM que é uma técnica poderosa para mapear as relações entre as bases de dados e assim, tornar mais flexível a fase de planeamento do projeto e consequentemente do seu desenvolvimento.

### 4.3.7. Componentes de fábrica

Os componentes de fábrica são componentes avançados e complexos na maioria dos casos, mas também podem existir alguns componentes mais simples, para facultar aos desenvolvedores algum auxílio na fase de implementação dos projetos. Por exemplo, um componente que permita gerar PDFs automáticos, um login com o recurso ao QR Code ou a partir da conta Google ou até mesmo uma área de assinaturas, são algumas ideias que podem ser desenvolvidas muitas vezes com recursos de JavaScript e adaptadas por utilizadores que

não dominam tão bem o código, implementando com menor esforço e padronizando à sua maneira, este tipo de recursos.

### 4.3.8. Templates de páginas pré-feitos

No desenvolvimento de aplicações de *low-code*, é demasiado importante o uso de páginas pré-definidas, porque ajudam em alguns casos a desenvolver algum tipo de páginas específicas, como por exemplo, uma página com uma tabela, filtros, botões de exportar, paginação, entre outros. Se necessário, o utilizador pode pegar em grande parte dessa página e personalizar da forma que quiser, isso ajuda no processo de aceleração do código e permite uma entrega mais rápida garantindo ao mesmo tempo a eficiência desse tipo de páginas e a sua integridade e facilitando trabalhos futuros do mesmo nível, podendo também o utilizador fazer de forma mais prática a reutilização de código e garantir a padronização da aplicação.

### 4.3.9. Visualizar e definir ecrãs em mobile e tablet

Uma característica viável do IPSystems será poder definir as propriedades do ecrã que está a ser desenvolvido por parte do utilizador, com recurso a mecanismos que simulem a visão da página em diferentes tamanhos de ecrã, como por exemplo, na ótica do mobile onde a largura e a altura são bastante reduzidas, obrigando a um planeamento diferente da interface para continuar com uma visão agradável e não piorar a experiência de utilizador.

Widgets para comportamento responsivo serão um recurso importante para que o utilizador consiga implementar soluções adaptadas ao tipo de ecrã com menor esforço e poupando assim a prática exaustiva de classes de css, podendo então acelerar o desenvolvimento de forma eficaz.

### 4.3.10. Folha de estilos e personalização de layouts pré-definida

É uma mais valia para o utilizador, ter algum auxílio quanto à parte visual e interativa do site, portanto será necessário algum trabalho complementar de JavaScript e CSS e também a construção de um bootstrap que dê apoio na fase de implementação, para as widgets já terem mais vida, assim como o próprio site, embora seja depois necessário o utilizador à sua escolha em detrimento do seu objetivo, adicionar as suas classes e modificar valores de variáveis nas classes e alguma lógica de programação. Esta funcionalidade será apenas um acelerador e não algo que seja tido como regra na implementação de uma aplicação IPSystems, para que o

utilizador enquanto apresenta uma solução rápida, consiga garantir a sua própria personalidade e robustez.

# Capítulo 5 - Conclusões

Ao longo dos dois últimos semestres do mestrado de Engenharia de software, foi desenvolvido o protótipo IPSystems, contando com muitos desafios ao longo deste percurso, embora o principal objetivo fosse conseguir criar algo sólido e que permita ao utilizador ter um desenvolvimento rápido e gerar uma página dinâmica mesmo sem possuir grandes conhecimentos de linguagem de programação, o projeto também visou garantir uma interface intuitiva, um desempenho eficiente e um ambiente confiável para o utilizador. Além de possibilitar o desenvolvimento rápido e dinâmico, desenvolveu-se uma solução acessível para atender tanto a utilizadores iniciantes quanto a profissionais experientes, assegurando que , mesmo sem conhecimento técnico, o utilizador possa criar projetos personalizados e funcionais de forma ágil.

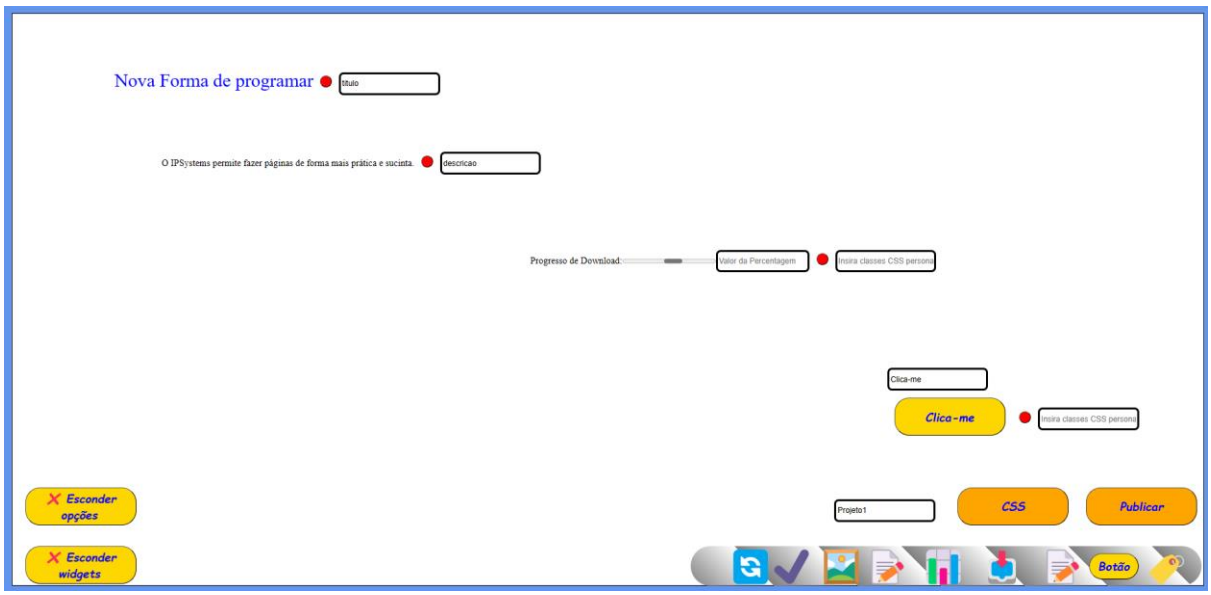
As principais dificuldades sentidas ao longo do desenvolvimento do IPSystems foram a comunicação entre as APIs e o frontend no início da implementação assim como algumas configurações de permissão para consumir um serviço e a geração de deploys automáticos que são feitos ao publicar o projeto que está a ser implementado, de modo a aparecer num novo separador com um link adaptado com novos caracteres únicos que são a chave do projeto específico do utilizador.

Apesar de todas as advertências, o projeto ficou estável e visualmente agradável e percetível na ótica do utilizador, permitindo ao mesmo ter uma interação agradável com o projeto e explorar a plataforma.

O trabalho desenvolvido nesta tese, contribuiu significativamente para o meu crescimento profissional, proporcionando uma experiência abrangente em todas as fases no desenvolvimento de software, desde o planeamento até à implementação e respetiva fase de testes. Desta forma, tive a possibilidade de ter contacto com ferramentas de desenvolvimento de software das quais ainda não tinha tido experiência como React e Django.

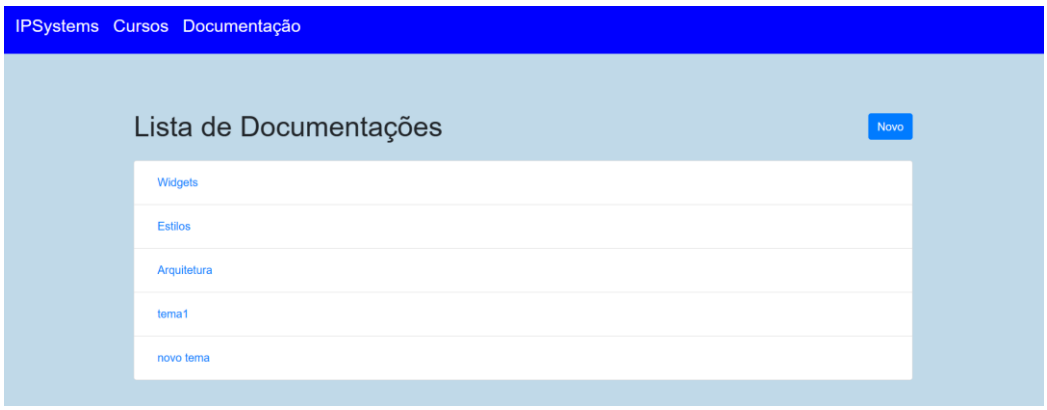
No futuro, pretendo aplicar estes conhecimentos adquiridos em projetos reais no desenvolvimento de aplicações, de forma eficiente e alinhada às melhores práticas de desenvolvimento, garantindo segurança, escalabilidade e qualidade nos projetos. Além disso, pretendo aprimorar continuamente as minhas capacidades em todas as fases do desenvolvimento, com o objetivo de entregar soluções robustas e confiáveis.

# Anexos




*Anexo A - Representação de uma página desenvolvida na ótica do utilizador*

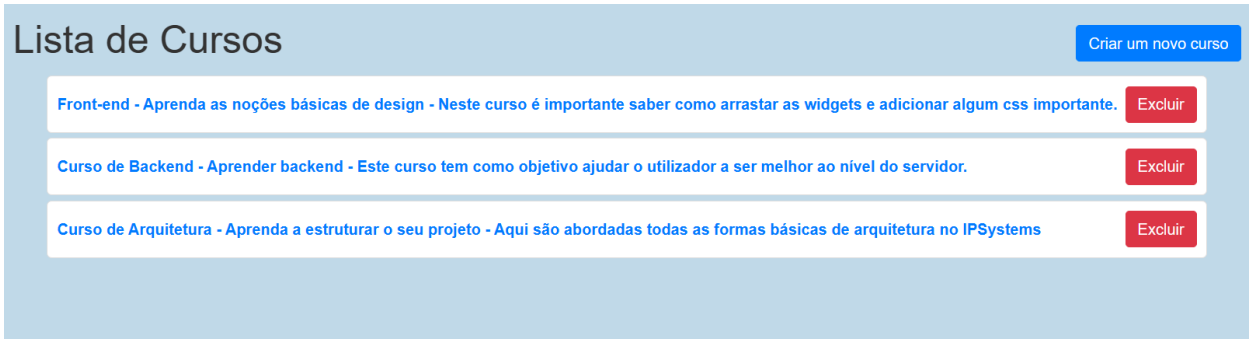

*Anexo B - Página inicial do backoffice*




*Anexo C - Lista de Documentação do backoffice*


*Anexo D - Página de criação de novo tema no backoffice*


*Anexo E - Edição ou exclusão de um tema já criado no backoffice*




*Anexo F - Lista de Cursos no backoffice*

# Bibliografia


*Adobe Communications Team. (2022, March). Waterfall Methodology: A Complete guide. Adobe Experience Cloud Blog.* *https://business.adobe.com/blog/basics/waterfall*

Appian. (2024). *Appian*. https://appian.com/fr.html

*Low-code application development Platform | Mendix*. (2024, January 31). Mendix. https://www.mendix.com/

OutSystems. (2024). *High-Performance Low-Code for app development*. https://www.outsystems.com/

*Plataforma Low-Code+ | CronApp*. (2024). https://www.cronapp.io/

Alves, F. R., & Alcalá, S. G. S. (2022). Análise da abordagem LOW-CODE como facilitador da transformação digital em indústrias. *E-tech*, *15*(1). https://doi.org/10.18624/etech.v15i1.1186

Caldeira, C. P. (2015). *INTRODUÇÃO AO HTML (HyperText Markup Language)* [Artigo Académico, Departamento de Informática da Universidade de Évora]. https://dspace.uevora.pt/rdpc/bitstream/10174/13240/1/Introdu%C3%A7%C3%A3o%20ao%20HTML.pdf

*Capterra. (2021). Mendix [Online forum post]. Capterra.* *https://www.capterra.pt/software/126575/mendix-app-platform*

*Capterra. (2022). Appian [Online forum post]. Capterra.* *https://www.capterra.pt/software/149795/appian*

*Conceitos básicos de Flexbox - CSS | MDN*. (n.d.). MDN Web Docs. Retrieved February 17, 2024, from https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox#os_eixos_do_flexbox

*Conceitos básicos de grid layout - CSS*. (n.d.). MDN Web Docs. Retrieved February 17, 2024, from https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_grid_layout/Basic_concepts_of_grid_layout

De Jesus Gomes, A. (2019, July 22). *JavaScript para aplicações móveis*. (Doctoral dissertation). http://hdl.handle.net/10400.26/36451

Duarte, N. F. B. (2015). *Frameworks e bibliotecas JavaScript* [Dissertação de Mestrado, Instituto Superior de Engenharia do Porto]. https://recipp.ipp.pt/bitstream/10400.22/8223/1/DM_NunoDuarte_2015_MEI.pdf

Fielding, R. T., Nottingham, M., & Reschke, J. (2022, June 1). *RFC 9110: HTTP Semantics*. https://www.rfc-editor.org/rfc/rfc9110.html

Kumar, R., & Jaiswal, S. (2015). *Learning Django Web Development*.

Maldonado, J. C., Braga, R. T. V., Germano, F. S. R., & Masiero, P. C. (2002). Padrões e frameworks de software. *Notas Didáticas, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, ICMC/USP, São Paulo, SP, Brasil*, 28. https://sites.icmc.usp.br/rtvb/apostila.pdf

Marques, M. G. (2023). *Comparação de plataformas low-code* [Dissertação de Mestrado, Escola de Engenharia da Universidade do Minho]. https://repositorium.sdum.uminho.pt/bitstream/1822/86203/1/Mariana%20Goncalves%20Marques.pdf

Matos, A. A. R. (2022). *Desenvolvimento de aplicações web em plataforma de Low-Code* [Relatório Final de Estágio do Mestrado]. Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança. https://bibliotecadigital.ipb.pt/bitstream/10198/26419/1/Andr%C3%A9%20Matos.pdf

Outsystems. (2020, June 20). *What are the disadvantages of Outsystems? | OutSystems*. https://www.outsystems.com/forums/discussion/62038/what-are-the-disadvantages-of-outsystems/

*OWASP Top 10:2021*. (n.d.). https://owasp.org/Top10/

Passerino, L. M., & Montardo, S. P. (2007). Inclusão digital e acessibilidade digital: interfaces e aproximações conceituais. *Encontro da Associação Nacional dos Programas de Pós-Graduação em Comunicação, 16*, 1-17. ID-acess_compos_2007_versão final.PDF (pbworks.com)

Pinho, M. (2020, July 31). *Outsystems: Será o futuro da programação? - Dellent*. Dellent. https://dellentconsulting.com/blog/outsystems-futuro-programacao

Ribeiro, A. F. P. (2018). *Aplicação de uma metodologia ágil de desenvolvimento de software numa organização do sector público* [Dissertação de Mestrado, Instituto Universitário de Lisboa]. https://repositorio.iscte-iul.pt/bitstream/10071/18784/4/master_afonso_portela_ribeiro.pdf

Ribeiro, H. (2012). *USABILIDADE ACESSÍVEL: Metodologias para a avaliação qualitativa da usabilidade no design para a web* [Dissertação de Mestrado]. Faculdade de Belas Artes da Universidade do Porto.

Scheidt, F. A. (2015). *Fundamentos de CSS: criando design para sistemas web*. Outbox Livros Digitais. Fundamentos de CSS: criando design para sistemas web - Felippe Alex Scheidt - Google Livros

*Seletores avançados em CSS*. (2022). Acervo Lima. Retrieved February 17, 2024, from https://acervolima.com/seletores-avancados-em-css/

Semedo, M. J. M. (2012). *Ganhos de produtividade e de sucesso de metodologias ágeis VS metodologias em cascata no desenvolvimento de projectos de software* [Dissertação de Mestrado, Universidade Lusófona de Humanidades e Tecnologias].

https://recil.ensinolusofona.pt/jspui/bitstream/10437/6174/1/Disserta%C3%A7%C3%A3o-Maria%20Semedo%5Bentrega%5D.pdf

Soubhia, A. L., Da Costa, E. T., Freitas, F. L. M., Menezes, L. B., Dos Santos, M. A., & Maran, V. (2019). *Python 101* (1.0). Universidade Federal de Santa Maria Campus Cachoeira do Sul. Retrieved February 17, 2024, from https://lumac.ufsm.br/wp-content/uploads/2022/04/Apostila_Python_v_1.pdf

*Tecnologia Low-Code | Transformação digital*. (n.d.). Jornal De Negócios. https://transformacaodigital.jornaldenegocios.pt/tecnologia-low-code.php