

# Dynamic Dashboard Generation Platform - backend

Relatório Final do Projeto Integrador

Hugo Miguel Lopes da Cruz  
Matheus Negrini Liotti  
Pedro Henrique Pessoa Camargo  
Tomas dos Santos Nunes Loureiro Esteves



Licenciatura em Engenharia Informática e Computação

**Tutor na U.Porto:** Alexandre Miguel Barbosa Valle de Carvalho  
**Orientador na empresa/Proponente:** Fernando Cassola Marques

24/06/2025

## Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
1.1	Enquadramento . . . . .	2
1.2	Objetivos e resultados esperados . . . . .	2
1.3	Estrutura do relatório . . . . .	2
<b>2</b>	<b>Metodologia utilizada e principais atividades desenvolvidas</b>	<b>3</b>
2.1	Metodologia utilizada . . . . .	3
2.2	Intervenientes, papéis e responsabilidades . . . . .	3
2.3	Atividades desenvolvidas . . . . .	4
<b>3</b>	<b>Desenvolvimento da solução</b>	<b>5</b>
3.1	Requisitos . . . . .	5
3.2	Arquitetura e tecnologias . . . . .	6
3.3	Solução desenvolvida . . . . .	8
3.4	Validação . . . . .	13
<b>4</b>	<b>Conclusões</b>	<b>14</b>
4.1	Resultados alcançados . . . . .	14
4.2	Lições aprendidas . . . . .	14
4.3	Trabalho futuro . . . . .	15

# 1 Introdução

## 1.1 Enquadramento

O presente relatório descreve o desenvolvimento de uma Plataforma de Criação de Dashboards Personalizáveis, com foco na vertente backend. O projeto foi idealizado e implementado no âmbito da unidade curricular Projeto Integrador da Universidade do Porto, funcionando como um estágio curricular obrigatório.

A aplicação tem como objetivo fornecer uma plataforma que permita aos utilizadores manipular os seus dados de forma visual e personalizável, possibilitando a criação de dashboards próprios. O trabalho foi concebido como uma proposta idealizada de um modelo otimizado para a manipulação e personalização de qualquer tipo de dados, destinado a utilizadores que pretendem criar dashboards sem necessidade de aprofundar os seus conhecimentos sobre as tecnologias subjacentes à aplicação.

Deste modo, o intuito foi reduzir ao máximo a distância entre essa visão ideal e o projeto real. No entanto, é importante salientar que ainda existem oportunidades de melhoria para que este objetivo possa ser plenamente alcançado no futuro.

## 1.2 Objetivos e resultados esperados

### - Objetivos:

- Desenvolver uma aplicação web intuitiva que permita a personalização de dados de forma visual, incluindo a adição, edição e remoção de conteúdos de forma dinâmica e acessível.
- Implementar funcionalidades de upload de ficheiro CSV com até duas colunas de dados.
- Fornecer métodos de personalização visual ao utilizador durante a criação da dashboard.

### - Resultados:

- Permitir ao utilizador criar uma dashboard do zero a partir de seus dados.
- Permitir que os dados e dashboards do utilizador fiquem associados à sua conta após o processo de autenticação.
- Oferecer ao utilizador a possibilidade de personalizar o seu dashboard conforme as suas necessidades e preferências.

## 1.3 Estrutura do relatório

Este relatório está organizado em quatro partes principais. Na Introdução, é apresentado o enquadramento do projeto, o problema que motivou a sua realização, os objetivos definidos e os resultados esperados. A secção de Metodo-

logia descreve o plano de trabalho adotado, a distribuição de responsabilidades ao longo do desenvolvimento e as principais atividades realizadas.

Na parte dedicada ao Desenvolvimento da Solução, são detalhados os requisitos do sistema, a arquitetura concebida, as tecnologias utilizadas e a explicação da solução implementada. Por fim, na Conclusão, são apresentados os resultados obtidos, as principais aprendizagens decorrentes do projeto e sugestões para evoluções futuras que poderão melhorar a solução desenvolvida.

## 2 Metodologia utilizada e principais atividades desenvolvidas

Nesta secção está descrito a metodologia utilizada durante o projeto, incluindo as etapas de desenvolvimento, ferramentas utilizadas, stakeholders assim como a responsabilidade de cada um dos membros do grupo.

### 2.1 Metodologia utilizada

Durante o desenvolvimento deste projeto foi adotada uma metodologia iterativa, organizada em sprints semanais. Esta abordagem permitiu o planeamento e execução incremental de funcionalidades, com entregas contínuas e validação progressiva do trabalho. Ao longo de cada sprint, foram realizadas reuniões semanais com os orientadores, nas quais se discutiram os avanços realizados, eventuais dificuldades encontradas e os objetivos para a sprint seguinte.

Para suportar o desenvolvimento colaborativo, foi utilizado o GitLab[3] como plataforma de controlo de versões, garantindo o histórico completo das alterações efetuadas, bem como a integração contínua das contribuições dos vários membros da equipa. A estrutura do projeto foi distribuída por três repositórios principais, um para a base de dados, outro para a API e um terceiro para o frontend, complementados por um quarto repositório dedicado a tarefas de DevOps, responsável por coordenar e integrar os restantes componentes da aplicação.

A comunicação e colaboração síncronas da equipa foram asseguradas principalmente através do Discord, plataforma onde se realizaram sessões de trabalho conjuntas e partilha de ficheiros. A marcação de reuniões e a gestão rápida de disponibilidade foram facilitadas através do WhatsApp.

### 2.2 Intervenientes, papéis e responsabilidades

- **Tutor FEUP:** Alexandre Miguel Barbosa Valle de Carvalho
- **Tutores INESC TEC:** Fernando Cassola Marques, André Thiago Netto
- **Cliente:** O cliente foi representado por membros do INESC TEC, que colaboraram no projeto fornecendo orientações, critérios de validação e apoio contínuo ao longo do desenvolvimento.
- **Elementos do grupo:**

- Hugo Cruz (up202205022) – Frontend
- Matheus Liotti (up202102363) – DevOps
- Pedro Camargo (up202102365) – Base de Dados
- Tomás Esteves (up202205045) – API
- Embora todos os membros da equipa tenham participado ativamente em todas as vertentes do projeto, a responsabilidade principal por cada área foi atribuída individualmente, de forma a assegurar uma gestão organizada e uma distribuição equilibrada de tarefas.

## 2.3 Atividades desenvolvidas

1. **Análise preliminar e planeamento** – Incluiu o levantamento de requisitos, definição dos objetivos do projeto, escolha da arquitetura e das tecnologias a utilizar.
2. **Implementação da base de dados** – Envolveu a definição do modelo de dados e a criação da estrutura necessária para suportar as entidades da aplicação, utilizando armazenamento em memória durante o desenvolvimento.
3. **Desenvolvimento da API** – Consistiu na construção de um servidor backend, responsável por gerir utilizadores, dashboards e widgets, garantindo a comunicação com a camada de dados.
4. **Desenvolvimento do frontend** – Abrangeu a criação da interface de utilizador, permitindo a interação dinâmica com os dashboards e widgets de forma intuitiva e responsiva.
5. **Testes e validação** – Incluiu a realização de testes funcionais e exploratórios, identificação de erros, validação de requisitos e melhorias com base no feedback recolhido ao longo do desenvolvimento.

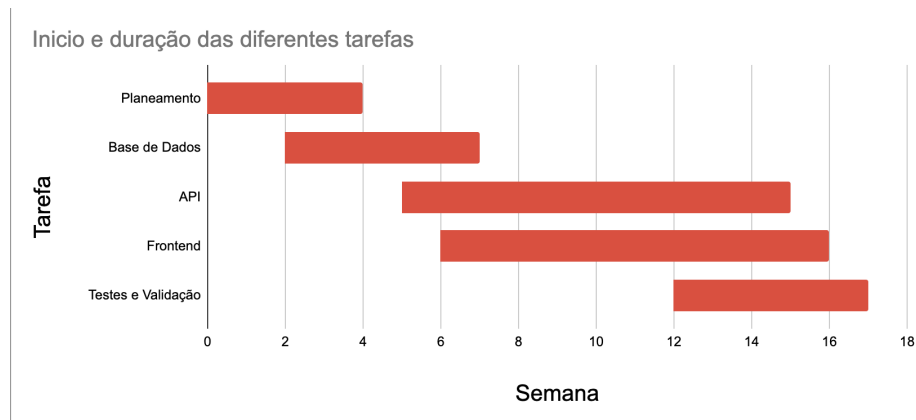


Figura 1: Diagrama de Gantt[6] com a distribuição temporal das atividades do projeto

### 3 Desenvolvimento da solução

Esta secção descreve o processo de conceção e implementação da solução desenvolvida. São apresentados os requisitos identificados, a arquitetura planeada para sustentar a aplicação, as tecnologias seleccionadas e as principais decisões tomadas ao longo do desenvolvimento, com o objetivo de responder de forma eficaz às necessidades do projeto.

#### 3.1 Requisitos

Durante a fase de planeamento, foram recolhidos os requisitos para a aplicação com base no feedback dos tutores do INESC TEC e nas discussões realizadas ao longo do desenvolvimento. Esses requisitos podem ser organizados em três categorias principais: requisitos funcionais, requisitos não funcionais e restrições do sistema. Esta estrutura permite uma compreensão clara das funcionalidades esperadas, das características de qualidade desejadas e das limitações impostas ao projeto.

##### Requisitos Funcionais

- O sistema deve permitir autenticação de utilizadores (registo, login, logout).
- Os utilizadores devem poder criar, gerir e eliminar dashboards personalizados.
- O sistema deve suportar a adição, remoção e reorganização de widgets nos dashboards.
- Os widgets devem apresentar dados atualizados em tempo real, sempre que aplicável.

### Requisitos Não Funcionais

- A aplicação deve ser compatível com browsers modernos e ter uma interface responsiva e intuitiva.
- A plataforma deve ser facilmente escalável.
- A sessão do utilizador deve expirar automaticamente por motivos de segurança.

### Restrições

- O sistema não deve depender de serviços pagos para integração de dados externos.

## 3.2 Arquitetura e tecnologias

A aplicação foi concebida segundo uma arquitetura de três camadas bem definidas, promovendo a separação de responsabilidades e a escalabilidade da plataforma:

- **Frontend:** Implementado em React[5], com React Router para navegação e Apollo Client[1] para integração com a API GraphQL[4].

- **Backend:** Desenvolvido em Node.js com Apollo Server, responsável pela autenticação, gestão de dashboards, widgets e fontes de dados.

- **Base de dados:** A base de dados que suporta a aplicação foi modelada com recurso a UML e implementada numa estrutura relacional (SQLite), concebida em SQL. Esta estrutura foi desenhada para dar suporte à captura de dados em tempo real, integrando entidades como tipos de widgets, fontes de dados, layouts e permissões de utilizador.

Adicionalmente, um servidor Express independente ('uploadServer.js') foi criado para o processamento de ficheiros CSV, permitindo a integração com visualizações gráficas.

### Dificuldades Técnicas e Soluções

- **Integração entre upload CSV (Express) e visualização (React/D3[2]):**

Desafio: A API GraphQL não é adequada para uploads de ficheiros.

Solução: Foi criado um servidor paralelo com Express e multer, que recebe o CSV, converte-o em JSON e retorna os dados para visualização. Estes são posteriormente associados a widgets do dashboard.

- **Autenticação e segurança:**

Desafio: Garantir segurança de acesso aos dashboards e widgets por utilizador.

Solução: Implementação de autenticação JWT com verificação de token no contexto do Apollo Server, protegendo queries e mutations sensíveis.

- **Layout dinâmico de widgets:**

Desafio: Permitir redimensionamento e posicionamento livre dos widgets.

Solução: Uso da biblioteca 'react-grid-layout', integrando com mutations de atualização de layout ('updateWidgetLayout') e sincronização via Apollo.

- **Visualização de dados dinâmicos com D3:**

Desafio: Transformar dados CSV arbitrários em gráficos funcionais.

Solução: Implementação de lógica genérica para interpretar os dois primeiros campos do CSV como 'x' e 'y', com seleção dinâmica do tipo de gráfico.

**Tecnologias Utilizadas e Justificação**

<b>Tecnologia</b>	<b>Função</b>	<b>Justificação</b>
<b>React</b>	Interface do utilizador	Reativo, modular, com grande ecossistema e compatível com bibliotecas como D3.
<b>Apollo Client</b>	Comunicação com GraphQL	Gestão eficiente de cache e integração transparente com React.
<b>Node.js</b>	Backend e upload server	Simples, leve e com grande suporte para APIs REST e GraphQL.
<b>Apollo Server</b>	API GraphQL	Integração nativa com GraphQL, com suporte a contextos e autenticação.
<b>Express</b>	Upload de CSVs	Flexível e adequado para criação rápida de servidores REST.
<b>JWT</b>	Autenticação	Permite autenticação segura e sem estado via tokens.
<b>D3.js</b>	Visualização de gráficos	Biblioteca poderosa para renderização dinâmica e interativa de gráficos SVG.
<b>SQLite</b>	Base de dados relacional	Leve, sem dependência externa, adequada ao contexto de prototipagem local.

Tabela 1: Tecnologias utilizadas e suas justificações



### 3.3 Solução desenvolvida

A solução desenvolvida consiste numa plataforma web que permite a utilizadores autenticados criar, personalizar e visualizar dashboards dinâmicos, utilizando dados próprios ou externos, com representação gráfica interativa.

O sistema foi projetado para utilizadores não técnicos, com foco na simplicidade de navegação, usabilidade intuitiva e flexibilidade visual.

#### - Página de Autenticação:

Ao aceder à aplicação, o utilizador é redirecionado para a página de login, onde pode introduzir o seu email e palavra-passe. Existe também a opção de se registar como novo utilizador com um perfil Editor ou Viewer.

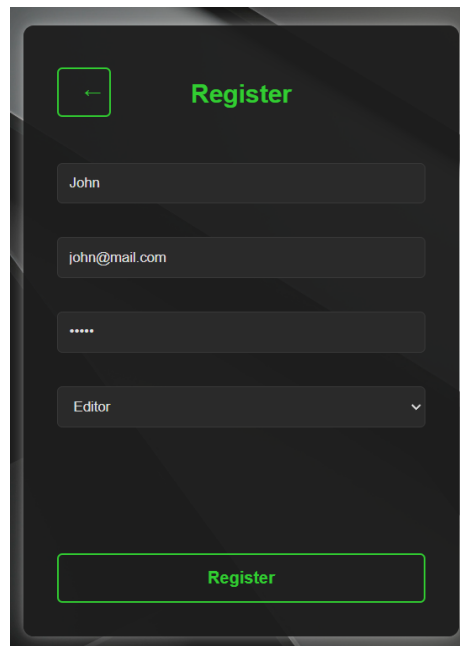
A screenshot of a 'Register' form in a dark-themed application. The form is centered and has a dark background with light gray borders. At the top left, there is a square button with a left-pointing arrow. To its right, the word 'Register' is written in a bright green font. Below this, there are four input fields: the first contains 'John', the second contains 'john@mail.com', the third contains six dots (password field), and the fourth is a dropdown menu currently showing 'Editor'. At the bottom of the form, there is a wide rectangular button with the word 'Register' in green.

Figura 2: Formulário de login com campos preenchidos e botão “Register”.

#### - Página Inicial:

Após autenticação, o utilizador entra na página Home, onde são apresentados todos os dashboards previamente criados. Cada cartão mostra o título do dashboard, um ícone ilustrativo, e um botão de opções para eliminar.

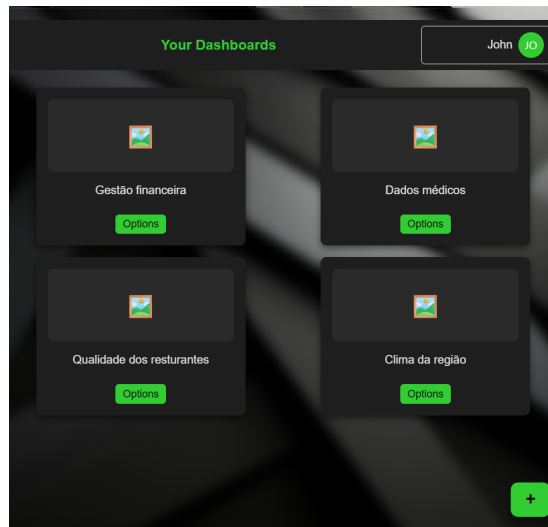


Figura 3: Galeria com múltiplos dashboards criados, botão flutuante de “+”.

Ao clicar no botão “+”, o utilizador pode criar um novo dashboard, introduzindo apenas o título. A criação é imediata e reflete-se na galeria.

Figura 4: Formulário de criação de um novo dashboard.

#### - Dashboard Interativo:

Ao entrar num dashboard, o utilizador é apresentado a uma interface com uma barra de ferramentas lateral com opções de adicionar widget de texto, fazer upload de ficheiro CSV e visualizar fontes de dados.

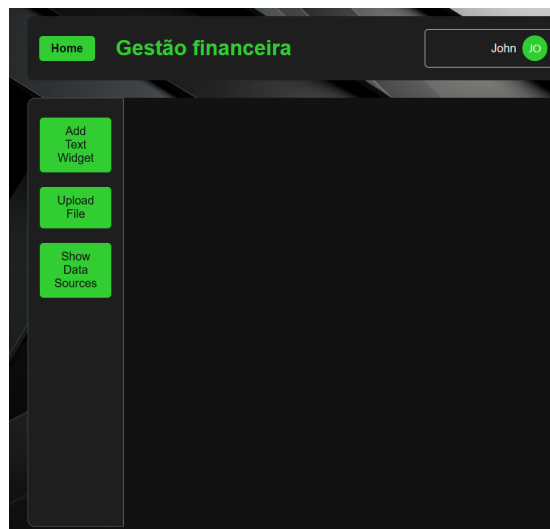


Figura 5: Dashboard vazio com toolbar visível e título no topo

#### - Upload de Ficheiros e Geração de Gráficos:

Ao seleccionar “Upload File”, o utilizador pode submeter um ficheiro ‘.csv’ contendo dados tabulares. Após o upload, é possível escolher entre gráfico de barras ou gráfico circular. O sistema processa automaticamente os dados e insere um widget de gráfico no dashboard, interpretando os dois primeiros campos do ficheiro como ‘x’ e ‘y’.

A screenshot of a modal form titled 'Upload CSV File'. It contains a file selection button labeled 'Escolher arquivo' next to the filename 'produtos.csv'. Below this, there is a label 'Chart Type:' followed by a dropdown menu currently showing 'Bar Chart'. At the bottom of the form, there are two buttons: 'Upload' and 'Cancel'.

Figura 6: Formulário de upload com dropdown “Chart Type”.

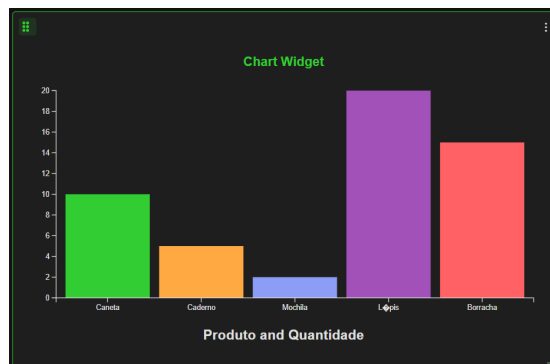


Figura 7: O widget com gráfico gerado a partir do upload.

#### - Widgets de Texto:

Além de gráficos, o utilizador pode criar widgets de texto livre para anotações, títulos ou explicações dentro do dashboard.

Create Text Widget

Title:

Text:

Create Cancel

Figura 8: O widget com gráfico gerado a partir do upload.

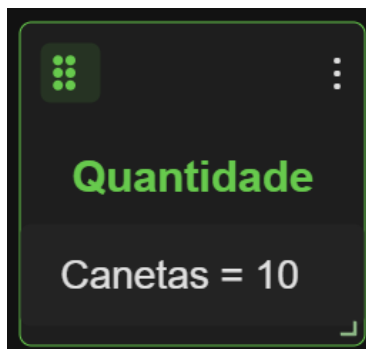


Figura 9: Um widget de texto.

#### - Interação com os Widgets:

Cada widget permite arrastar e reposicionar (via drag handle); redimensionar (com limites mínimos); eliminar, através do menu suspenso “⋮”; visualização de título, conteúdo textual ou gráfico, conforme o tipo.

#### - Fontes de Dados:

O botão “Show Data Sources” permite consultar todos os ficheiros carregados e utilizados nos gráficos, com nome, URL e tipo de origem.

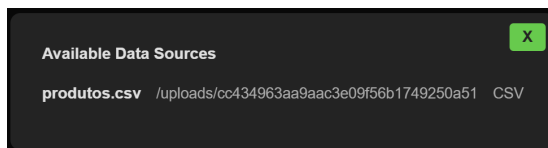


Figura 10: Pop-up com lista de fontes de dados, nomes de ficheiro e URLs.

#### - Logout:

O utilizador pode terminar a sessão através da área do avatar no canto superior direito, onde encontra a opção “Logout”.

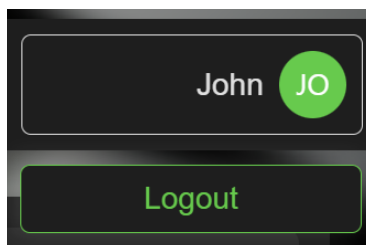
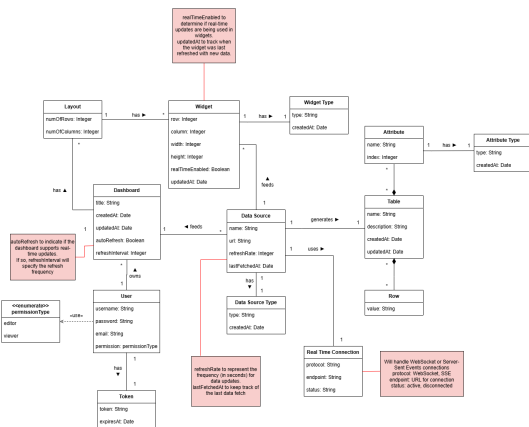


Figura 11: Pop-out de logout expandido com botão de saída.

The screenshot shows a web application interface for financial management. The top navigation bar includes a 'Home' button and a user profile 'John 10'. The sidebar contains links for 'Add Text Widget', 'Upload File', 'Show Data Sources', and 'John 10'. The main content area displays three summary cards: 'Frutas' (Fruits) with 'TOTAL = 50', 'Valores' (Values) with '350€', and 'Gastos' (Expenses) with '17.500'. Below these cards are two charts: a bar chart titled 'Category and Value' showing values for Apples (10), Bananas (20), and Oranges (15); and a pie chart titled 'Category and Value' showing the same distribution.

A imagem que se segue representa o diagrama UML da base de dados desenvolvida para este projeto.



### 3.4 Validação

A validação da solução desenvolvida foi realizada através de um conjunto de testes para avaliar o funcionamento das operações GraphQL, através de uma ferramenta chamada Apollo Sandbox, uma ferramenta integrada do ecossistema Apollo.

### Testes com Apollo Sandbox

Durante o desenvolvimento, todas as queries e mutações expostas pelo backend GraphQL foram testadas. Esta ferramenta permitiu simular chamadas reais ao servidor, validar os tipos de dados retornados e verificar o comportamento da API em diferentes cenários.

Foram testadas operações como:

- Criação de dashboards e widgets
- Associação de layouts e fontes de dados
- Atualização e eliminação de widgets
- Autenticação de utilizadores

### Resultados

Os testes com o Apollo Sandbox permitiram garantir que:

- Os tipos definidos no schema GraphQL estão corretos
- As mensagens de erro emitidas pelo servidor são claras e informativas
- Os dados persistidos na base de dados correspondem exatamente ao esperado

Esta abordagem facilitou também a identificação rápida de problemas na recolha de informação, permitindo corrigi-los com facilidade.

## 4 Conclusões

### 4.1 Resultados alcançados

O site encontra-se funcional. Criámos o protótipo com o objetivo de cumprir os requisitos definidos na proposta apresentada, nomeadamente: permitir a personalização de várias dashboards pelo utilizador, gerir os dados fornecidos para criar as representações gráficas, possibilitar a manipulação da posição dos widgets dentro da grelha da dashboard e permitir a autenticação e o registo de utilizadores, garantindo a exclusividade dos seus dados. Estes resultados foram alcançados através da divisão entre API, Base de Dados, Frontend e DevOps.

Relativamente à participação dos integrantes, todos desempenharam um papel substancial ao longo do projeto. Assim, é seguro afirmar que cada elemento contribuiu com aproximadamente 25% para o desenvolvimento do mesmo.

### 4.2 Lições aprendidas

Durante o desenvolvimento do Projeto Integrador, vivenciámos diversos momentos de aprendizagem que certamente serão valiosos no futuro. Destaca-se,

em particular, a experiência de trabalhar em contacto direto com o cliente, recebendo feedback contínuo e planeando as etapas seguintes com base nas reuniões semanais realizadas ao longo do semestre.

Um aspeto particularmente enriquecedor foi a autonomia de cada membro ao longo do projeto. Embora cada elemento da equipa tivesse uma área de responsabilidade mais focada, não houve uma limitação rígida a essas divisões. Esta flexibilidade permitiu-nos explorar e colaborar em outras vertentes do desenvolvimento, promovendo uma visão mais abrangente da solução e incentivando uma abordagem mais integrada à resolução de problemas.

Outro aspeto de grande relevância foi a escolha das ferramentas, linguagens e estruturas utilizadas no desenvolvimento do projeto. Em alguns casos, já possuíamos um conhecimento superficial sobre determinadas tecnologias, mas, no geral, foi necessário aprender novas ferramentas, o que representou não só um primeiro contacto prático com essas tecnologias, como também uma experiência valiosa para o futuro.

A integração entre o carregamento de ficheiros CSV e a sua visualização nos dashboards revelou-se um desafio significativo. No entanto, aproveitámos essa oportunidade para aprofundar os nossos conhecimentos técnicos e para consolidar boas práticas relacionadas com a gestão de problemas e resolução de imprevistos.

Em suma, reconhecemos que há ainda espaço para evolução, tanto no que diz respeito ao desenvolvimento técnico do projeto como à gestão global do mesmo. Ainda assim, consideramos que esta unidade curricular foi fundamental para o nosso crescimento, proporcionando uma abordagem mais realista dos desafios enfrentados no mercado de trabalho.

### 4.3 Trabalho futuro

Para garantir a escalabilidade e a robustez da aplicação, será essencial implementar persistência real de dados através de uma base de dados como SQLite ou PostgreSQL. Desta forma, dashboards, widgets e utilizadores poderão ser guardados e recuperados entre sessões, assegurando durabilidade e suporte a múltiplos utilizadores em simultâneo.

Adicionalmente, pretende-se expandir os tipos de visualização disponíveis, incluindo gráficos de linhas, área, radar, dispersão e mapas, bem como adicionar interatividade com funcionalidades como zoom e tooltips. Estas melhorias tornarão a plataforma mais versátil para diferentes áreas de aplicação.

Outra evolução importante será a integração com fontes de dados em tempo real, através de APIs externas ou WebSockets, permitindo dashboards dinâmicos e atualizados ao segundo, ideais para contextos como mercados financeiros ou monitorização de sensores IoT.

Será igualmente relevante, numa fase futura, introduzir funcionalidades como a gravação automática das alterações e a gestão de histórico de versões, proporcionando maior segurança, rastreabilidade e flexibilidade ao utilizador.

Por fim, melhorar a interface com um design totalmente responsivo e mobile-first permitirá que a aplicação seja utilizada confortavelmente em qualquer dis-



positivo, ampliando o seu alcance. Entre outras melhorias previstas, destaca-se ainda o suporte a outros formatos de ficheiro para importação de dados, novos tipos de gráficos e a possibilidade de fixar widgets em posições específicas no dashboard.

Estas evoluções contribuirão para tornar a plataforma mais poderosa, adaptável e centrada na experiência do utilizador.

## Referências

- [1] Apollo Client. Introduction to apollo client. <https://www.apollographql.com/docs/react>, 2025.
- [2] D3. Getting started. <https://d3js.org/getting-started>, 2025.
- [3] GitLab. Gitlab. <https://gitlab.com/>, 2025.
- [4] GraphQL. Introduction to graphql. <https://graphql.org/learn/>, 2025.
- [5] React. Quick start. <https://react.dev/learn>, 2025.
- [6] Wikipedia. Gantt chart. [https://en.wikipedia.org/wiki/Gantt\\_chart](https://en.wikipedia.org/wiki/Gantt_chart), 2025.