

Universidade do Minho
Escola de Engenharia

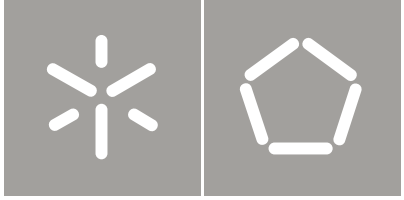
Daniel Cunha Sá

Abordagem Low-Code para Ecossistemas Web

Daniel Cunha Sá Abordagem Low-Code para Ecossistemas Web

UMinho | 2022

fevereiro de 2022



Universidade do Minho
Escola de Engenharia

Daniel Cunha Sá

Abordagem Low-Code para Ecossistemas Web

Dissertação de Mestrado
Mestrado Integrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação do
Professor Doutor Manuel Filipe Vieira Torres dos Santos

fevereiro de 2022

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição-NãoComercial-SemDerivações
CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/4.0/

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

AGRADECIMENTOS

Gostava de agradecer a todos, que de forma direta ou indireta, contribuíram para a realização desta dissertação e me proporcionaram ao longo desta jornada momentos enriquecedores ao nível académico, profissional e pessoal.

Em primeiro lugar, agradecer ao Professor Doutor Manuel Filipe Vieira Torres dos Santos que desde o primeiro momento se mostrou interessado em propor uma dissertação ligada à área que eu pretendia e também pela disponibilidade, confiança e motivação que mostrou e transmitiu ao longo desta jornada. Gostaria também de agradecer ao Professor Tiago Guimarães que me acompanhou na parte mais técnica desta dissertação e se mostrou disponível para responder a questões relacionadas com a plataforma.

Por fim, quero agradecer à minha família, em especial aos meus pais e avós, e à Sandra que sempre me apoiaram e me motivaram em todos os momentos e que sem o apoio deles esta jornada universitária não seria possível. Obrigado.

RESUMO

Abordagem Low-Code para Ecossistemas Web

Low-code é uma abordagem à programação que procura possibilitar a criação de artefactos, tanto por programadores como não programadores. Permite uma rápida criação e disponibilização de aplicações com o mínimo de esforço para escrever numa linguagem de programação assim como para a instalação e configuração de ambientes, formação e implementação. Permitindo, desta forma, que o desenvolvimento acompanhe as necessidades de uma empresa de forma mais rápida e eficiente.

Assim, foi criada uma plataforma web direcionada a empresas da área financeira, designadas por *fintech*, que permita ao utilizador, neste caso em particular, o contabilista, personalizar uma *dashboard* composta principalmente por gráficos e *widgets*, de forma a tirar o máximo partido dos dados das empresas que presta o seu serviço. No entanto, para além desta aplicação em específico, a plataforma poderá ser aplicada a qualquer outra área com o mínimo esforço por parte dos programadores.

A plataforma foi desenvolvida utilizando as tecnologias que mais se adequam ao desenvolvimento de aplicações web. Assim temos o *back-end* que foi desenvolvido utilizando a plataforma Node.js, que utiliza uma das linguagens com maior crescimento no desenvolvimento para a web, o JavaScript. Para a interface foi utilizado o *framework* também em JavaScript, Vue.js. Quanto ao armazenamento de dados, foi utilizado o modelo de dados não relacional e para a gestão e observação do mesmo utilizou-se o software MongoDB.

A abordagem metodológica utilizada para o desenvolvimento desta dissertação foi o *Design Science Research (DSR).*

Palavras-chave: Abordagem Low-Code; Fintech; JavaScript; MongoDB; Vue.js.

ABSTRACT

Low-code aproach for web ecosystem

Low-code is an approach to programming that seeks to enable the creation of artifacts, both by programmers and non-programmers. It allows a quick creation and deployment of applications with a minimum of effort to write in a programming language as well as for the installation and configuration of environments, training and implementation. Allowing, in this way, that the development accompanies the needs of a company in a faster and more efficient way.

A web platform was created aimed at companies in the financial area, called fintech, which allowed the user, in this particular case, the accountant, to customize a dashboard composed mainly of graphics and widgets, in order to take full advantage of the data. of the companies that provide their service. However, in addition to this specific application, the platform can be applied to any other area with minimal effort on the part of programmers.

The platform was developed using the technologies that best suit the development of web applications. So we have the back-end that was developed using the Node.js platform, which uses one of the fastest growing languages in web development, JavaScript. For the interface, the framework also in JavaScript, Vue.js, was used. As for data storage, the non-relational data model was used and the MongoDB software was used for its management and observation.

A methodological approach used for the development of this dissertation was the Design Science Research (DSR).

Keywords: Low-Code Approach; Fintech; JavaScript; MongoDB; Vue.js.

ÍNDICE

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS i

DECLARAÇÃO DE INTEGRIDADE ii

Agradecimentos.....iii

Resumo iv

Abstract v

Lista de abreviaturas/Siglas x

Lista de figuras.....xi

Lista de tabelas xiii

1. Introdução 1

1.1 Enquadramento e Motivação.....1

1.2 Objetivos e Resultados Esperados 1

1.2.1 Objetivos 1

1.2.2 Resultados Esperados.....2

1.3 Organização do Documento.....2

2. Abordagem Metodológica 4

2.1 Design Science Research (DSR) 4

2.2 Aplicação da Metodologia.....5

2.3 Estratégia de Pesquisa Bibliográfica.....6

3. Conceitos Base e Estado da Arte.....7

3.1 Low-Code.....7

3.2 Low-Code vs No-Code 8

3.3 Fintech 8

3.4 Arquitetura de microsserviços 9

3.4.1 Enquadramento....9
3.4.2 Arquitetura Monolítica vs Arquitetura de Microsserviços 10
3.5 Node.JS....11
3.6 Frameworks Interface JavaScript 12
3.6.1 React.js 12
3.6.2 Vue.JS 13
3.6.3 Angular 14
3.7 Base de Dados 14
3.7.1 Modelo de Dados Relacional....14
3.7.2 Modelo de Dados não Relacional....15
3.8 Plataformas de desenvolvimento Low-Code e respetivas Forças e Fraquezas 15
3.8.1 AgilePoint 16
3.8.2 Appian 17
3.8.3 AuraQuantic 18
3.8.4 Betty Blocks....18
3.8.5 Mendix....19
3.8.6 Microsoft 20
3.8.7 OutSystems 21
3.8.8 Salesforce 22
3.8.9 ServiceNow (App Engine)....23
3.8.10 TrackVia 23
4. Abordagem Tecnológica....25
4.1 Ecossistema Web....25
4.2 Ferramentas Utilizadas....25
4.2.1 Programação em Node.js....25

4.2.2 Framework de interface 26
4.2.3 Base de dados não relacional – MongoDB 26
4.2.4 Editor de texto 27
5. Desenvolvimento do Projeto e Resultados Obtidos 28
5.1 Arquitetura 29
5.2 Modelos de Base de Dados 31
5.3 Experiências e resultados obtidos 35
5.3.1 Login e Registo 35
5.3.2 Página Inicial 36
5.3.3 Dashboard - Empresa selecionada na imagem anterior 37
5.3.4 Menu Adicionar Gráfico 39
5.3.5 Adicionar Widget 44
5.3.6 Editar Paleta Cores 47
5.3.7 Partilhar Layout 48
5.3.8 Opções de editar e eliminar para gráficos e widgets 49
5.3.9 Opção de eliminar gráfico ou widget 49
5.3.10 Opção de Update para gráficos 50
5.3.11 Opção de Update para Widgets 53
5.3.12 Painel de utilizador 54
5.3.13 Lista de ficheiros existentes e gestão dos mesmos 55
5.3.14 Opção de upload de ficheiros 56
5.4 Alterações necessárias para aplicação de plataforma com outros dados ou em outras áreas 56
6. Conclusões 60
6.1 Análise Crítica e Limitações 60

6.2 Conclusão 61

7. Referências Bibliográficas 62

LISTA DE ABREVIATURAS/SIGLAS

DSR	Design science research
MVC	Model-View-Controller
IA	Inteligência Artificial
RPA	Robotic Process Automation
BPM	Business Process Management
EUA	Estados Unidos da América
API	Application Programming Interface
iPaaS	Integration platform as a service
SaaS	Software as a Service

LISTA DE FIGURAS

Figura 1 – Diferentes etapas e processo metodológico DSR[1] 4
Figura 2 - Comparação entre os processos de uma arquitetura monolítica e uma arquitetura de microsserviços, respetivamente. 10
Figura 3 - Exemplo de uma arquitetura de microsserviços, onde um dos serviços para e os restantes continuam a funcionar, mantendo assim a aplicação a funcionar parcialmente..... 11
Figura 4 - Interface MongoDB Compass 26
Figura 5 - Interface Visual Studio Code 27
Figura 6 - Arquitetura 29
Figura 7 - Página de login..... 35
Figura 8 - Página de registo 36
Figura 9 - Página Inicial com todas as empresas do utilizador e opção de ver detalhe da empresa 37
Figura 10 - Menu para registar nova empresa 37
Figura 11 - Dashboard personalizada pelo utilizador para a empresa selecionada..... 38
Figura 12 - Botão "Opções Dashboard" e respetivo menu 38
Figura 13 - Modal para adicionar gráfico 39
Figura 14 - Exemplo de gráfico do tipo "Linear Chart"..... 39
Figura 15 - Exemplo de gráfico do tipo "Bar Chart" 40
Figura 16 - Exemplo de gráfico do tipo "Pie Chart"..... 40
Figura 17 - Exemplo de gráfico do tipo "Doughnut Chart" 41
Figura 18 - Exemplo de gráfico do tipo "Polar Area Chart"..... 41
Figura 19 - Exemplo de gráfico do tipo "Radar Chart" 42
Figura 20 - Exemplo do campo que surge quando a seleção do tipo de gráfico é “Linear Chart”, “Bar Chart” ou “Radar Chart” 42
Figura 21 - Exemplo do campo que surge quando a seleção do tipo de gráfico é “Pie Chart”, “Doughnut Chart” ou “Polar Area Chart” 43
Figura 22 - Seleção do tipo de dados a demonstrar no gráfico 43
Figura 23 - Modal para adicionar novo Widget..... 44

Figura 24 - Opção de selecionar o tipo de dados a ver 45
Figura 25 - Exemplo de seleção dos valores para o Ano 1 e para o Ano 2..... 46
Figura 26 - Exemplo de Widget 46
Figura 27 - Modal com as paletas de cores pré-definidas..... 47
Figura 28 - Exemplo de alteração de cor com auxílio ao seletor de cores..... 48
Figura 29 - Modal de Partilhar Layout 48
Figura 30 - Empresas possíveis de partilhar o layout 49
Figura 31 - Botão de opções para gráficos e widgets..... 49
Figura 32 - Confirmação de eliminar gráfico 49
Figura 33 - Confirmação de eliminar widget 50
Figura 34 - Opção de editar gráfico quando o tipo de gráfico é “Pie Chart”, “Doughnut Chart” ou “Polar Area Chart”..... 50
Figura 35 - Opção de editar gráfico quando o tipo de gráfico é “Linear Chart”, “Bar Chart” ou “Radar Chart” 51
Figura 36 - Exemplo de tipos de dados selecionados..... 51
Figura 37 - Exemplo de alteração de cor de um determinado valor com auxílio a um seletor de cores 52
Figura 38 - Exemplo de alteração de cores com quatro tipos de dados selecionados .. 52
Figura 39 - Exemplo de modal para editar widget 53
Figura 40 - Exemplo de visualização dos dados do utilizador 54
Figura 41 - Exemplo de painel de edição dos dados de utilizador 54
Figura 42 - Exemplo de edição apenas de password 55
Figura 43 - Exemplo de modal onde é possível ver os ficheiros inseridos..... 55
Figura 44 - Modal de upload de ficheiros para a plataforma..... 56
Figura 45 - Exemplo de resposta da função GET do ficheiro "layout.controller.js"..... 57
Figura 46 - Exemplo de resposta da função GET do ficheiro “layout.controller.js”..... 58
Figura 47 - Exemplo de dados mostrados na página inicial 59

LISTA DE TABELAS

Tabela 1 - Tabela “colorpalettes” 31
Tabela 2 - Tabela “layouts” 32
Tabela 3 - Tabela “users” 33
Tabela 4 - Tabela “companies” 34
Tabela 5 - Análise Crítica e Limitações 60

1. INTRODUÇÃO

1.1 Enquadramento e Motivação

Low-code é uma abordagem à programação que procura possibilitar a criação de artefactos, tanto por programadores como não programadores. Esta abordagem permite a criação e disponibilização rápida de aplicações com o mínimo de esforço para escrever numa linguagem de programação, assim como para a instalação e configuração de ambientes, formação e implementação. Desta forma, permite que o desenvolvimento acompanhe as necessidades de uma empresa de forma mais rápida e eficiente.

São abordagens com elevado crescimento no mercado, existindo atualmente bastantes alternativas de tecnologias e soluções fechadas assim como abordagens personalizadas, criadas de forma a facilitar o desenvolvimento e integração em soluções mais "tradicionais".

É imperativo conceptualizar e desenvolver soluções que reduzam o custo e o tempo de desenvolvimento assegurando a sua adaptabilidade e dinamismo, assumindo inteiramente uma abordagem *Low-code* recorrendo a tecnologias e soluções já existentes no mercado, ou assumindo uma vertente híbrida, integrando abordagens que garantam um desenvolvimento mais assente em *Low-code* em plataformas de programação "tradicional".

1.2 Objetivos e Resultados Esperados

1.2.1 Objetivos

- Análise das abordagens existentes ao conceito de *Low-Code*, ou seja, será elaborada uma pesquisa bibliográfica e um estado da arte sobre paradigmas *Low-Code*.
- Caraterização de ambientes típicos de aplicação;

- Definição de características para avaliação;
- Pretende-se, se for possível, efetuar um benchmarking com base num conjunto de características definidas anteriormente;
- Selecionar o paradigma mais eficaz para desenvolvimentos Web;
- Com base no paradigma selecionado, desenvolver e implementar artefactos que recorram a abordagens *Low-Code* e que garantam resposta de forma genérica a diferentes soluções e projetos.

1.2.2 Resultados Esperados

Aplicação de uma abordagem *Low-Code* efetuando a prova de conceito em diferentes casos de uso na área de desenvolvimento Web, nomeadamente nas áreas de *Fintech*.

Avaliação da eficiência tecnológica e funcional da abordagem recorrendo a peritos e testes experimentais em ambiente real (ou simulado quando tal não for possível).

1.3 Organização do Documento

O documento apresentado encontra-se dividido em seis capítulos, onde cada um aborda um tópico diferente e imprescindível para o desenvolvimento e apresentação de todos os conteúdos da dissertação.

O primeiro capítulo consiste na introdução ao tema representado na dissertação, onde está inserido um enquadramento e motivação para a realização deste projeto, uma apresentação dos objetivos e dos resultados esperados aquando da realização e entrega do projeto.

O segundo capítulo consiste na Abordagem Metodológica onde se descreve a metodologia de investigação, a forma como esta foi aplicada a este projeto e a estratégia de pesquisa bibliográfica utilizada.

O terceiro capítulo consiste nos Conceitos Base e Estado da Arte, onde são introduzidos os temas e conceitos revelantes para o desenvolvimento da dissertação. Este capítulo inicia-se com um conjunto de conceitos fundamentais para a compreensão

e desenvolvimento desta dissertação, nomeadamente Low-Code, Fintech, Arquitetura de Microsserviços, Node-js, *Frameworks* Interface JavaScript, Base de Dados e um conjunto de Plataformas de desenvolvimento Low-Code existentes e as suas respetivas forças e fraquezas.

O quarto capítulo consiste na abordagem tecnológica, onde é apresentada a arquitetura e uma descrição e demonstração das ferramentas utilizadas no projeto.

O quinto capítulo consiste no desenvolvimento do projeto e nos resultados obtidos, ou seja, na apresentação do trabalho desenvolvido e das atividades da dissertação, bem como os campos da base de dados necessários para o desenvolvimento do projeto acompanhados com uma breve descrição de cada atributo.

O sexto capítulo consiste na realização de uma análise crítica e de algumas limitações encontradas aquando do desenvolvimento desta dissertação e também da conclusão.

O sétimo e último capítulo consiste nas Referências Bibliográficas.

2. ABORDAGEM METODOLÓGICA

2.1 Design Science Research (DSR)

A metodologia de investigação científica utilizada foi o *Design Science Research Methodology for Information Systems,* onde a sua finalidade é investigar possíveis soluções para determinados problemas ou melhorias em determinadas áreas [1]

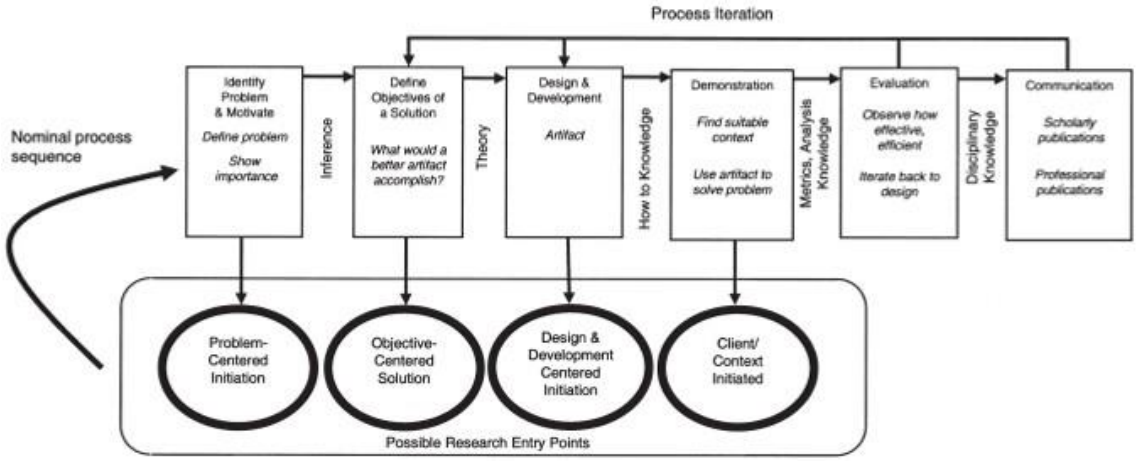


Figura 1 – Diferentes etapas e processo metodológico DSR[1]

Assim, o DSR é composto por seis etapas, onde a primeira, caracterizada por *"Problem identification and motivation",* começa com a perceção do problema e a motivação, é definido o problema a solucionar e uma justificação do valor que a solução do mesmo acarreta. Esta etapa é importante de modo a capturar e explicar a complexidade do problema.

A segunda etapa do ciclo, caracterizada por *"Objectives of a solution",* consiste em definir os objetivos da solução com base na definição do problema, onde estes têm de ser inferidos de forma racional. Os recursos necessários para esta fase incluem o conhecimento do estado dos problemas e soluções existentes nesse momento e a sua eficácia, caso existam.

A terceira etapa do ciclo, caracterizada por *"Design and development",* consiste em definir a funcionalidade desejada para o artefacto e a respetiva arquitetura, para de seguida, criar o artefacto real. Os recursos necessários para passar da fase dois, ou seja,

dos objetivos, para esta fase incluem o conhecimento da teoria que também pode ser usado como uma solução para o problema.

A quarta etapa do ciclo, caracterizada por *"Demonstration",* consiste em demonstrar a eficácia do artefacto para a resolução do problema, podendo ser necessário realizar experiências, simulações, casos de estudo ou outra atividade que se demonstre apropriada para o efeito. Os recursos necessários para esta etapa incluem conhecimento de como utilizar o artefacto na resolução do problema.

A quinta etapa do ciclo, caracterizada por *"Evaluation"*, consiste na observação e aplicação de métricas de modo a entender a capacidade do artefacto em responder ao problema. Envolve a comparação dos objetivos definidos na etapa dois com os resultados reais da solução. Os recursos necessários para esta etapa incluem conhecimento de métricas relevantes e técnicas de análise. É também no final desta etapa que os investigadores devem decidir se é pertinente repetir a etapa três de modo a melhorar a eficácia do artefacto, ou se devem continuar para a etapa seguinte deixando assim as melhorias adicionais para projetos subsequentes.

A sexta e última etapa do ciclo, caracterizada por *"Communication",* consiste na comunicação do problema e a importância do mesmo, a utilidade da solução desenvolvida, o rigor e a sua eficácia em dar resposta ao problema [1].

2.2 Aplicação da Metodologia

O projeto inicia-se com a consciencialização do problema (Etapa 1). Nesta primeira etapa foi identificado o problema, onde este consiste no desenvolvimento de uma aplicação web através de uma abordagem Low-Code que permita à mesma ser aplicada em diferentes casos de uso com o mínimo de alterações.

De seguida avançamos para a definição dos objetivos (Etapa 2), onde foi realizada uma revisão da literatura e caracterização do estado da arte de modo a entender quais as soluções existentes no mercado e de que forma estas podiam ou não ser aplicadas ao problema de forma total ou parcial, ou seja, se era possível utilizar plataformas já existentes ou se existia a necessidade de desenvolver algo mais personalizado.

Após esta análise e revisão da literatura e a caracterização do estado da arte, foi realizada a terceira etapa, ou seja, a etapa de definição das funcionalidades desejadas para a solução do problema bem como a respetiva arquitetura. Nesta fase propôs-se o desenvolvimento de uma plataforma web personalizada e capaz de ser aplicada em diversos cenários com o mínimo de alterações possível, caracterizando assim um ambiente Low-Code completamente funcional e intuitivo para qualquer programador adaptar ao seu problema ou negócio.

De seguida, procedeu-se à etapa quatro, onde a aplicação foi submetida a diversos testes. Ao mesmo tempo realizou-se a etapa cinco, onde foi observado o funcionamento e os resultados obtidos na etapa quatro de modo a compreender a sua eficácia na resolução do problema inicial.

A última etapa, consiste no documento final desta dissertação onde está integrado todo o processo de análise do problema até à demonstração dos resultados obtidos com o desenvolvimento da aplicação Low-Code.

2.3 Estratégia de Pesquisa Bibliográfica

De modo a ser possível elaborar o próximo capítulo, e antes de iniciar a pesquisa e seleção das fontes bibliográficas de suporte a este projeto, foi necessário definir uma estratégia consistente para obter o material relevante, onde as fontes selecionadas incluem artigos científicos, dissertações, artigos de conferências, livros, entre outros.

De modo a utilizar fontes de informação credíveis foram utilizados motores de busca de literatura, tais como, *Google Scholar, Web of Science* e *Science Direct*. Sendo que a foram utilizados conceitos para a pesquisa como *"Low-Code", "Low-Code Development", "Low-Code Software Development", "Low-Code Application Development", "No-code", "Microservice architecture", "node.js", "relational database vs NoSQL" e "Fintech", "DSR", "Design Science Research*".

3. CONCEITOS BASE E ESTADO DA ARTE

Este capítulo consiste numa referência ao estado atual do conhecimento sobre o tema que está em análise ou estudo pelo autor no momento em que inicia a sua dissertação. Assim, esta etapa permite ao autor estabelecer um ponto de partida para o estudo proposto, contextualizando o mesmo sobre os conceitos e o tema em desenvolvimento.

3.1 Low-Code

As plataformas de desenvolvimento Low-Code fornecem mecanismos de tecnologia que permitem facilitar e automatizar o desenvolvimento de aplicações de software, de modo a dar suporte às necessidades atuais das empresas e promover a transformação digital. Possibilitando assim que as empresas desenvolvam aplicações sem que seja necessária uma engenharia complexa e também facilitando a sua configuração alcançando rapidez e agilidade. Além disso, estas plataformas também oferecem uma forma mais econômica de atender aos requisitos internos do mercado e/ou das empresas [2].

Este tipo de desenvolvimento está em ascensão e combina código-fonte com interfaces gráficas interativas, permitindo desenvolver e implementar uma aplicação de forma rápida. De um modo geral as plataformas desenvolvidas em Low-Code são baseadas numa interface gráfica onde o programador apenas necessita de projetar a estética e as funcionalidades da aplicação, não sendo necessário recorrer a técnicas de programação tradicionais, evitando assim o desenvolvimento de sintaxes de código e a resolução de problemas de implementação. As técnicas de desenvolvimento Low-Code são derivadas da junção das linguagens de programação de quarta geração (“4GL – Fourth Generation Language”) com os conceitos de desenvolvimento rápido de aplicações (“RAD – Rapid Application Development”)[3].

3.2 Low-Code vs No-Code

Plataformas e ambientes *Low-Code* requerem alguma perspicácia e experiência de programação a quem os utiliza, pois é necessário projetar a estética e as funcionalidades da aplicação, no entanto uma plataforma *No-Code* permite que os chamados "*Citizen Developers"* desenvolvam aplicações personalizadas com pouca ou nenhuma habilidade de programação.

O desenvolvimento através de plataformas *Low-Code* cada vez é mais autónomo devido à utilização de Inteligência Artificial *(IA), Machine Learning (ML)* e também a Automação Robótica de Processos *(RPA).* Empresas como a *Forrester* e a *Gartner* preveem que o *Low-Code* veio para ficar e o seu crescimento continuará nos próximos anos, sendo que a tendência principal para a criação de código autónomo é que o *No-Code* eventualmente sobreponha o *Low-Code.*

No entanto, embora estas plataformas ofereçam uma maior eficiência no que diz respeito ao desenvolvimento e a facilidade de atingir uma solução completa rapidamente, levantam-se algumas questões de segurança, nomeadamente o facto de a maioria das aplicações desenvolvidas nestas plataformas dependem de redes, dados *upstream* e de outras aplicações. Estas vulnerabilidades podem originar a introdução de dados adulterados ou defeituosos, ou a alteração de processos e originar problemas de larga escala nas organizações. Assim, é fundamental que não ocorram lapsos no controlo e na observação no que diz respeito à cibersegurança dos processos desenvolvidos por *"Citizen Developers"* ou profissionais de TI [4].

3.3 Fintech

Fintech, resultante da junção das palavras *"Financial"* e *"Technology",* representa a tendência e as mudanças que o avanço tecnológico trouxe ao setor financeiro, sendo que estas mudanças facilitaram a gestão financeira por meio da automação, o câmbio de moedas e a globalização das instituições financeiras. O avanço apresentado pelas plataformas *Low-Code* e *No-Code* proporcionou um aumento do potencial do *Fintech* e dos cenários e métodos de pagamento.

Cada vez mais start-ups *Fintech* estão a recorrer a plataformas *Low-Code e No-Code* de modo a personalizar os seus sistemas de pagamento direcionados ao utilizador comum no menor tempo possível e devido a sua facilidade de uso é possível realizar o desenvolvimento com pouca ou nenhuma experiência em código. Estas plataformas estão a tornar o desenvolvimento de soluções *Fintech* mais simples e fácil, ajudando a diminuir os custos e a complexidade [5].

3.4 Arquitetura de microsserviços

3.4.1 Enquadramento

Fowller e Lewis (2014) descrevem a arquitetura de microsserviços como sendo uma abordagem para desenvolver uma única aplicação constituída por um conjunto de pequenos serviços, onde cada um deles executa o seu próprio processo e comunica-se utilizando mecanismo leves.

Desde o princípio da computação foram diversos os problemas e obstáculos solucionados com diferentes tipos de arquiteturas, de modo a acompanhar a evolução e o aparecimento de novas técnicas e métodos que auxiliam o desenvolvimento de aplicações de software.

Com o aparecimento da computação em nuvem, ocorreu um aumento de empresas interessadas nessa tecnologia e consequentemente começaram o processo de migração. No entanto, as equipas de desenvolvimento começaram a demonstrar insatisfação pois necessitavam de aplicações com arquiteturas que suportassem todas as adversidades e desafios existentes.

Após estes acontecimentos, deu-se o aparecimento da arquitetura de microsserviços, onde trata o desenvolvimento de aplicações como um conjunto de serviços independentes e pequenos. Assim, esta migração por parte das empresas ocorreu de forma natural quando entenderam que podiam criar as aplicações de forma rápida ao dividi-la em pequenas partes [6].

3.4.2 Arquitetura Monolítica vs Arquitetura de Microsserviços

Toda a lógica utilizada para lidar com uma determinada solicitação é executada num único processo, permitindo assim utilizar os recursos básicos da linguagem utilizada de modo a dividir a aplicação em classes, funções e *namespaces.* As aplicações monolíticas podem ser escaladas horizontalmente, ou seja, executando muitas instâncias do mesmo código de forma paralela e com auxílio a um balanceador de carga [Fowler 2014]. Á medida que o projeto avança e se torna mais complexo, a dificuldade em manter a estrutura deste tipo de aplicações torna-se extremamente difícil, o que acaba por dificultar a manutenção e consequentemente a produtividade de todos os envolvidos.

Segundo [Newman 2015], os microsserviços são pequenos e totalmente focadas em realizar algo em específico, sendo de alto ou baixo grau de dificuldade. Esta arquitetura também permite às equipas utilizarem diversas e variadas linguagens e serviços numa só aplicação, podendo ser possível utilizar um determinado serviço para solucionar uma situação ou problema específico, o que consequentemente acaba por garantir um melhor desempenho da aplicação como um todo. Uma vantagem desta arquitetura face à monolítica é o escalonamento e a replicação, pois com a arquitetura de microsserviços devido a cada um dos serviços ser independente e autónomo é possível replicar e escalonar apenas um serviço em específico e necessário, ao contrário da monolítica onde é necessário escalonar e replicar toda a aplicação, tal como podemos comparar na Figura 2 [6].

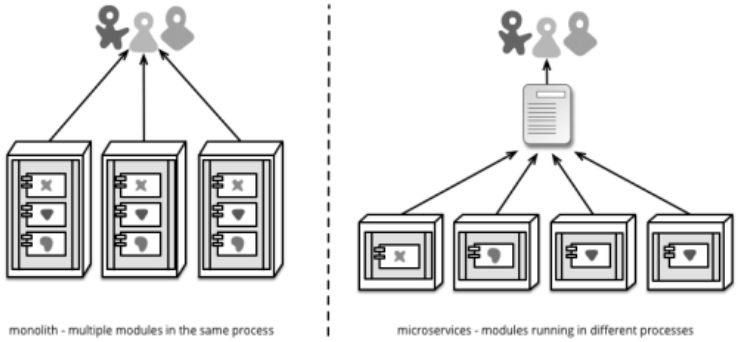


Figura 2 - Comparação entre os processos de uma arquitetura monolítica e uma arquitetura de microsserviços, respetivamente.

A arquitetura de microsserviços também permite à aplicação manter o seu funcionamento parcial caso um dos serviços falhe, ou seja, a aplicação continua a funcionar com todos os restantes serviços, acabando também por facilitar a resolução e identificação desses problemas [6].

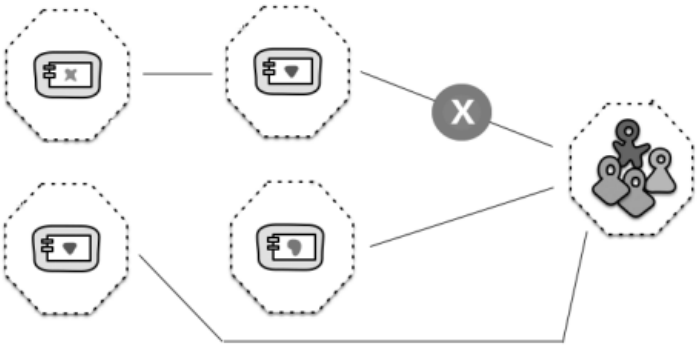


Figura 3 - Exemplo de uma arquitetura de microsserviços, onde um dos serviços para e os restantes continuam a funcionar, mantendo assim a aplicação a funcionar parcialmente.

3.5 Node.JS

O Node.js é uma plataforma de software de código aberto baseado na linguagem de programação JavaScript e é utilizado para o desenvolvimento de aplicações do lado do servidor. De acordo com Tilkov e Vinoski (2010), é baseado no tempo de execução de uma implementação do Google, nomeadamente o motor "V8".

Tem se tornado cada vez mais popular principalmente quando é utilizado em aplicações em tempo real onde é exigida escalabilidade e alto desempenho. Ao contrário de outras tecnologias os processos não dependem de *multithreading,* ou seja, permite que as aplicações sejam executadas apenas numa *thread* de forma assíncrona. Deste modo, torna-se adequado para aplicações com grande número de pedidos e conexões simultâneas, pois enquanto uma solicitação de I/O (Input/Output) está a ser processada o Node.js continua capaz de responder e executar outros pedidos [7].

3.6 Frameworks Interface JavaScript

Na atualidade, a maioria dos websites utiliza Javascript para apresentar páginas mais interativas ou para lidar com alguma funcionalidade específica. As páginas web tradicionais são caracterizadas por serem uma aplicação de várias páginas, onde sempre que o utilizador muda de página é carregado um novo documento HTML. No entanto, cada vez mais estão a ser utilizados modelos de desenvolvimento baseados em *Single-Page Application (SPA),* ou seja, modelos de página única, onde apenas as partes da página alteradas são carregadas através do servidor, reduzindo assim o tempo de espera sempre que é necessário carregar algo novo para a página, o que consequentemente melhora a experiência do utilizador.

Desde os dias em que a sua utilização era opcional até ao recente padrão HTML5 o papel do JavaScript mudou totalmente. Neste momento as aplicações podem ter a lógica de negócio do lado do servidor, o que consequentemente reduz a dependência de JavaScript, ou podem ser completamente desenvolvidas no lado do cliente, onde o servidor apenas é responsável pelo tratamento dos dados. É também possível que seja utilizado um meio termo, pois o JavaScript é uma linguagem bastante flexível e que muitas vezes devido às diferentes maneiras de planear e de desenvolver por parte dos desenvolvedores e respetivas equipas, acabam por surgir diferentes *frameworks* e bibliotecas com soluções distintas para o mesmo problema.

Dos três *frameworks* apresentados temos o ReactJS e o Vue.js que utilizam JavaScript e o Angular que utiliza TypeScript. TypeScript é uma linguagem de programação de código aberto desenvolvida pela Microsoft que consiste num superconjunto de JavaScript, ou seja, adiciona sintaxe ao *JavaScript* tornando-o mais completo e permitindo também converter todo o código TypeScript em JavaScript, podendo deste modo ser executado onde o *JavaScript* também é [8]

3.6.1 React.js

O *framework* React.js é uma biblioteca *JavaScript* que permite criar interfaces de utilizador e teve origem num projeto interno do Facebook, passando a ser *open-source* em maio de 2013. Pode ser utilizada para combinar fragmentos de código curtos e

independentes numa interface de utilizador complexa. O seu principal objetivo é focar-se na performance pois possui um elevado desempenho, logica de código simples e pode resolver facilmente problemas de compatibilidade entre diferentes navegadores Web [8].

Introduziu também a sintaxe JSX, que permite escrever marcações semelhantes a HTML dentro do JavaScript, bem como a reutilização de componentes de forma simples ao mesmo tempo que garante que a estrutura interna dos componentes se mantenha clara [9].

Outra especificação é que utiliza Virtual DOM, ou seja, é mantida uma representação ideal da interface do utilizador na memória antes de ser sincronizada com o DOM real [10].

3.6.2 Vue.JS

O *framework* Vue.js em contraste com outros *frameworks* é caracterizado por um desenvolvimento incremental de baixo para cima (*bottom-up*), foi desenvolvido por Evan You. Foi desenvolvido pois este enquanto trabalhador da Google sentia que estava a repetir muitas vezes o mesmo código HTML, deste modo, começou a procurar ferramentas para otimizar o seu tempo. No entanto não encontrou nenhum *framework* ou biblioteca para satisfazer as suas necessidades. Foi então que começou a desenvolver uma biblioteca que lhe permitisse desenvolver aplicações de forma rápida e também criar e reutilizar facilmente os componentes em toda a aplicação [11].

Tal como o React.js utiliza Virtual DOM, ou seja, é mantida uma representação ideal da interface do utilizador na memória antes de ser sincronizada com o DOM real. Tem foco numa biblioteca central, no entanto é possível ser adicionadas bibliotecas conforme a necessidade do utilizador [10], [12].

Vue.js rastreia automaticamente as mudanças de estado do JavaScript e atualiza eficientemente o DOM aquando das mudanças realizadas, pode-se concluir assim que o Vue tal como o React, ambos são reativos [13].

3.6.3 Angular

AngularJS é um *framework* JavaScript de código aberto utilizada para criar aplicações web dinâmicas. Foi desenvolvido pela Google e permite usar HTML como linguagem base estendendo a sintaxe do HTML de forma a expressar os componentes da aplicação de forma clara e sucinta. A vinculação de dados e a injeção de dependência auxiliam o programador eliminando muito do código que teria de escrever [14].

O AngularJS implementa o modelo MVC com um novo conceito denominado Model-View-Whatever, ou MVW, que significa que o Model e View podem ser sincronizados automaticamente sem a necessidade de um controlador, no entanto, também é possível utilizar um controlador [15].

3.7 Base de Dados

3.7.1 Modelo de Dados Relacional

Foi proposto por Edgar Codd em 1970 e indica que uma visão relacional dos dados permite uma descrição natural dos mesmos, onde não é necessário adicionar novas estruturas para a sua representação, o que consequentemente promove uma maior independência dos mesmos em relação aos programas. A maior vantagem deste modelo face aos seus antecessores é a sua representação simples e a facilidade em expressar consultas complexas.

A linguagem *Structured Query Language (SQL)* desenvolvida originalmente pela IBM, evoluiu e tornou-se a principal e mais usada para a criação, manipulação e consultas em Sistemas de Gestão de Bancos de Dados (SGBD) relacionais.

Este modelo assume que os dados podem ser organizados em tabelas e que as relações entre as mesmas podem ser usadas para consultas mais complexas.

Das várias opções de software do mercado para este modelo de dados, destacam-se o MySQL e PostGres [16].

3.7.2 Modelo de Dados não Relacional

Este modelo conhecido como NoSQL (*Not Only SQL*), surgiu com o objetivo de lidar com a gestão de grandes volumes de dados, semiestruturados ou não estruturados que requerem alta disponibilidade e escalabilidade, pois os modelos relacionais demonstram uma inflexibilidade e ineficiência quando estão em causa o elevado volume de dados de, por exemplo, uma rede social. Este modelo não necessita de um esquema ou então pode conter apenas um esquema flexível pois estes modelos baseiam-se no conceito de chave-valor, permitindo haver uma maior flexibilidade na organização dos dados, em contrapartida não há uma garantia da integridade dos dados como acontece nos modelos relacionais devido a sua estrutura rígida.

Existem diversos softwares NoSQL disponíveis no mercado, dos quais, Hypertable, Cassandra, MongoDB, Redis, CouchDB e Dynamo [17].

3.8 Plataformas de desenvolvimento Low-Code e respetivas Forças e Fraquezas

De modo a clarificar e demonstrar as forças e fraquezas das Plataformas de desenvolvimento Low-Code foi utilizada a Metodologia do "Quadrante Mágico" (*Magic Quadrants*) produzida pela Gartnet Consulting. Este Quadrante Mágico descreve as tendências e a maturidade do mercado bem como os seus participantes. É composto por dois eixos, o eixo da "*Completeness of Vision"* e o "*Ability to Execute",* sendo que estes dois originam os quatro quadrantes dos quais:

- ***"Niche Players"*** (Canto Inferior Esquerdo) – Demonstram ter uma visão insuficiente para serem considerados *"Visionaries"* e uma execução insuficiente para serem considerados *"Challengers".* Embora sejam considerados *"Niche Players"* podem representar uma boa escolha para cenários, ecossistemas ou domínios específicos, e em particular os preços apresentados serão provavelmente mais apelativos para o público.
- ***"Challengers"*** (Canto Superior Esquerdo) – Demonstram uma boa execução, mas limitações na visão. Com extensões e melhorias continuas da visão este grupo podem se tornar *"Leaders"*.

- ***"Visionaries"*** (Canto Inferior Direito) – Demonstram uma visão forte, mas limitações na execução corporativa. Apresentam várias abordagens à inovação, no entanto não atendem aos padrões de execução de um *"Leader"* em termos de níveis ou modelos de negócio.
- ***"Leaders"*** (Canto Superior Direito) – Demonstram uma boa execução, ou seja, um produto tecnicamente bom aliado a um bom desempenho de vendas. Demonstram também visão, ou seja, inovação no produto aliada a uma boa estratégia de marketing. Estão incluídos os principais fornecedores de SaaS na nuvem, dos quais Microsoft, Salesforce, ServiceNow. Especialistas em *Low-Code* dos quais Mendix e OutSystems, e por fim, um fornecedor de uma solução híbrida de automação de processos e de aplicações *Low-Code*, a Appian.

3.8.1 AgilePoint

Esta plataforma é caracterizada por pois é amplamente focada no desenvolvimento de aplicações centradas no processo. Os clientes que utilizam a plataforma tendem a ser grandes e abrangem diversos fatores, sendo que têm maior enfase no setor da manufatura, governo e serviços financeiros.

Forças

Dispõe de um conjunto completo de recursos (*Full Stack*) onde a sua utilização consiste em arrastar e soltar, dispõe também de bibliotecas pré-construídas de modo a ajudar os programadores. Os recursos são constituídos por formulários, processos, modelos de dados, aplicações para dispositivos móveis personalizadas, entre outros. Dispões também de parcerias com empresas como *UiPath* e *Blue Prism* que permitem a automação robóticos de processos (*RPA – Robotic Process Automation*). É integrado em ecossistemas de outros fornecedores como *Salesforce, AppExchange e Microsoft AppSource.*

A plataforma *AgilePoint eForms* oferece uma alternativa multiplataforma sendo que é um substituto para o *Microsoft InfoPath*, pois simplifica a colheita de dados para quem não pretende investir no *Microsoft Power Apps*. Oferecem também fluxos de processo orientados por Inteligência Artificial de modo a fornecer fluxos de trabalho

dinâmicos, sendo que está a inovar com foco em *Low-Code Edge Computing,* suporte a *Digital Twins* e em *Internet of Things (IoT).*

3.8.2 Appian

Esta plataforma é caracterizada por ser "*Leader*", permite a criação de aplicações com *Low-Code,* recursos avançados que permitem várias experiências em simultâneo, organização dos processos de negócio, decisões automatizadas com a utilização de Inteligência Artificial, *Machine Learning* e *RPA.* O principal foco da plataforma são processos complexos, casos *End-to-End,* aplicações que exigem automatização, regras e análises sofisticadas. Diferencia-se tecnologicamente por incluir capacidades de automatização *Full-Stack*, integração *No-Code* pré-desenvolvida que integra vários serviços de *IA* e suporte *End-to-End* para *DevOps.*

Forças

Esta plataforma *Low-Code* contém um robusto desenvolvimento de aplicações orientadas a processos. Diferencia-se pela sua oferta de um conjunto completo de ferramentas *Low-Code* de automatização com capacidade para lidar com fluxos de trabalho complexos, regras de negócio e gestão de casos utilizando *RPA.* Para além disso oferece ferramentas *Low-Code* que permitem criar aplicações multiplataforma que permitem experiência de clientes e funcionários. Caracteriza-se pela sua solução *End-to-End,* permitindo com recurso ao *Low-Code,* criar aplicações de negócio, orquestração de processos complexos e automatização de tarefas repetitivas com *RPA.*

Fraquezas

Fica aquém pela falta de ferramentas que permitam dar suporte a projetos de desenvolvimento web mais complexos e envolvendo várias equipas e pela falta de controlo por parte do utilizador de modo a dimensionar e escalar a plataforma [18].

3.8.3 AuraQuantic

Anteriormente conhecida como *AuraPortal* é caracterizada por ser "*Niche Player*", fornece ferramentas de desenvolvimento *Low-Code* e *BPM* permitindo automatizar as operações de negócios, sendo que estas focam-se em regiões como *EMEA* (Europa, Médio Oriente e África) e *LATAM* (América Latina), onde os clientes tendem a ser organizações de médio porte sendo a maioria nos setores bancários e de seguros.

Esta plataforma fornece uma aplicação centrada no processo com vários editores gráficos separados par diferentes funções. Tem como previsão integrar *IA* como auxilia à tomada de decisão, *RPA* e *IA* para assistência aos programadores.

Forças

Devido à sua experiência com *BPM* oferecem um suporte sofisticado para gestão de processos e decisões, no entanto, e tendo realizado uma transição antecipada para as ferramentas *Low-Code* fornece uma boa plataforma que permite a *"Citizen Developers*" criarem processos extensíveis ou aplicações centradas em dados.

Fraquezas

O suporte para várias "*Personas"* de desenvolvimento é limitado. Apesar da plataforma fornecer um *designer* de modelos de dados e de processos abrangente e fácil de usar, a sua biblioteca possui algumas limitações de componentes de suporte a "*Citizen Developers".* Apenas tem suporte limitado para programadores profissionais fora da área de criação de integrações.

3.8.4 Betty Blocks

É caracterizada por ser "*Visionary*", sendo a sua plataforma *Low-Code* é focada principalmente no "*Citizen Development",* as operações concentram-se na Europa Ocidental e nos EUA, os clientes consistem em empresas de tamanhos variados e visam principalmente casos de uso *B2B (Business-to-Business)*. Distingue-se pela sua inovação tecnológica, pela facilidade de desenvolvimento e reutilização de componentes.

Aprimorou a implementação de *Container,* ambiente de desenvolvimento e mecanismos de extensões.

Vantagens

Identificou e entregou dois dos principais impulsionadores de produtividade baseados em *Low-Code*, as aplicações combináveis de modo a maximizar a reutilização, e a simplicidade do desenvolvimento de aplicações. Estes permitem uma melhor reutilização e velocidade de desenvolvimento, e o mecanismo de "Blocos" incentiva ao desenvolvimento do ecossistema. Contínua inovação em tecnologias de distribuição e implementação baseadas em *Container*, *GraphQL* para integração de aplicações externas e mecanismo de extensões baseadas na nuvem.

3.8.5 Mendix

É uma subsidiária da Siemens e é caracterizada por ser "*Leader"*. Oferece ferramentas de desenvolvimento *No-Code* e *Low-Code* com base numa arquitetura "*Model-Driven" e "Cloud-Native"*, ou seja, orientada a modelos e com base em computação na nuvem. Recentemente introduziu o *Mendix Data Hub* para integração *Low-Code* e dois novos serviços de nuvem, o *Mendix Cloud Dedicated* e o *Mendix for Private Cloud*. Tem uma parceria com a SAP de modo a fornecer uma plataforma *Low-Code* totalmente integrada com o *SAP HANA*.

Forças

Oferece recursos robustos de modo a atender às necessidades de integração, fluxos de trabalho, processamento de eventos e utilização de *IA* como apoio ao desenvolvimento. O *Mendix Assist,* na segunda geração, utiliza *ML* (*Machine Learning*) de modo a integrar e apresentar sugestões assistidas por *IA* e em tempo real. O *Mendix Data Hub* permite aos programadores descobrir, organizar e combinar de forma organizada e simples os dados e recursos de negócio, para tal utiliza as mesmas ferramentas de modelação visual utilizadas para a construção de logicas de negócio e aplicações de modelos de dados. A plataforma *Mendix* é das poucas que contém

recursos avançados de desenvolvimento com suporte para aplicações móveis altamente personalizáveis, *Chatbots* e casos de uso de aplicações de *IoT.*

Fraquezas

As fraquezas demostradas consistem nos seus certificados de segurança na nuvem, na amplitude da lista de parceiros, em alguns dos relatórios avançados e nas características dos processos. Devido a ser uma subsidiaria da Siemens acaba por depender da mesma para soluções especificas [18].

3.8.6 Microsoft

É caracterizado por ser um *"Leader",* a plataforma *Low-Code* consiste no *Microsoft Power Apps* que inclui direitos para o *Power Automate* e para o *Common Data Service*. Juntos fazem parte da *Power Platform* onde é adicionado o *Power BI* para análises de negócio e *Power Virtual Agents* para *Low-Code chatbots*. O *Power Apps* consiste numa aplicação de arrastar e soltar e fornece integrações e lógica de negócios. Incluí suporte para integração com o *Microsoft Teams* para implementar soluções, modelos, habilitar trabalho remoto e permitir expansão para *AI* e *RPA*.

Forças

Recursos avançados de desenvolvimento profissional e *No-Code*, oferece uma das ferramenta mais simples de *design-time* do *Power Apps* onde esta é usada para projetar aplicações através de uma abordagem de arrastar e soltar com uma linguagem de expressão idêntica ao Microsoft Excel, o que o torna adequado para *"Citizen Developers"* e programadores de unidades de negócio. O *Power Apps* e o *Common Data Service* (CDS) contêm um conjunto avançado de *APIs* e *OData endpoints* com o objetivo de executar operações *CRUD (Create, Read, Update, Delete)* de modo a fornecer suporte à integração com *iPaaS* de terceiros. Através da *IA* contém modelos pré-construídos que permitem extrair texto de imagens, extrair elementos específicos de textos, entre outros.

Fraquezas

O *PowerApps* dá suporte a fluxos de trabalho, no entanto a interface para processos de fluxos de negócio é separada e apenas está disponível em aplicações baseadas em modelo. A junção de processos mais complexos pode exigir o uso de produtos separados, como por exemplo o *Microsoft Azure Logic Apps*. A plataforma *Power Automate* só suporta o padrão *BPMN (Business Process Model and Notation)* caso haja uma integração total com o *Microsoft Visio.*

3.8.7 OutSystems

É caracterizada como *"Leader",* a plataforma *Low-Code* suporta soluções na cloud, nos servidores locais *(On-Premises)* e também soluções híbridas. Está a aprimorar recursos de *DevOps* e gestão de dados e a adicionar novos "*Builders"* de fluxos de trabalho e de *UX* de modo a facilitar o desenvolvimento.

Forças

Consiste numa interface gráfica onde tem funções de arrastar e soltar, onde contém diversos modelos e padrões de interface de utilizador. Permite a criação de aplicações personalizadas através de modelos e padrões pré-existentes, mas também permite aos utilizadores criar os seus próprios modelos e padrões de raiz e reutilizá-los.

Suporta ciclos de vida de desenvolvimento de software e a utilização de *"Developer Persona",* permitindo assim o fornecimento de uma arquitetura baseada em padrões segura e escalável, ou seja, garante que todas as aplicações que sejam desenvolvidas e implementadas pelos *Citizen Developers"* sejam robustas e que à medida que a complexidade aumenta possam continuar a ser desenvolvidas por um programador profissional. As aplicações podem trabalhar com diversos serviços e fontes de dados, quer da própria *OutSystems* quer de serviços locais ou baseados na nuvem de terceiros. Fornece também análises de interdependência e análise de padrões entre os diversos componentes e das versões das aplicações.

Fraquezas

O *Workflow Builder* que permite abordar os casos de uso dos *Citizen Developers* ainda não se encontra disponível de forma nativa e geral, pois este recurso ainda não está comprovado, no entanto é uma necessidade de forma a alcançar outro tipo de concorrentes com suporte para estes programadores.

É cada vez mais utilizado para automação de processos de negócio e suporta aceleradores de design de dados, incluindo a importação de ficheiros Excel, bem como modelos relacionais, entre outros. Em contrapartida não é suficientemente competitivo em comparação com outros concorrentes em aspetos relacionados com o desenvolvimento de aplicações centradas em processos, especialmente em modelação de decisões complexas e otimização de processos.

De um modo geral, esta plataforma tenda a ser preferido por programadores profissionais e não tanto por *Citizen Developers*, enquanto a maioria das plataformas Low-Code aqui representadas têm soluções mais mistas. Existem também relatos de falta de apoio por parte do suporte.

3.8.8 Salesforce

É caracterizada por ser *"Leader",* foca-se no desenvolvimento de aplicações de negócio de vários níveis de complexidade, casos de uso e habilidades de programador. Têm o objetivo de expandir o suporte a *Low-Code* para integração, orquestração, composição e unificação.

Forças

Tem em desenvolvimento um ambiente unificado de desenvolvimento e governança que conecta um conjunto de tecnologias da plataforma *"Customer 360",* das quais *Heroku* e *Apex*, permitindo assim oferecer uma experiência de desenvolvimento ampla e *"Multipersona"* o que capacita os clientes com diferentes tipos de habilidades. Contém serviços baseados na nuvem considerados robustos e usados em larga escala [18].

Fraquezas

Dificuldade em evoluir uma plataforma com 15 anos de modo a estar atualizada para a era móvel, pois leva tempo e permite que os concorrentes mais recentes e mais ágeis inovem mais rapidamente. A empresa em vez de criar recursos para a plataforma prefere utilizar recursos de terceiros [18].

3.8.9 ServiceNow (App Engine)

É caracterizado por ser *"Leader"*, a plataforma *Low-Code "App Engine"* foca-se na personalização do *ServiceNow SaaS* e na criação de aplicações empresariais de fluxo de trabalho. Recentemente criou o "*Guided App Creator"* para *Citizen Developers* onde introduziu o novo *"Now Design System"* melhorando assim a experiência de utilizador e os serviços de *Chatbot*. Planeiam adicionar, nas próximas versões, mais suporte a processos, automação e *Citizen Development* e também melhorar a integração com produtos de IA.

Forças

A plataforma *"Now Platform"* oferece suporte a SaaS e ao *App Engine*, onde este é desenvolvido e usado pela própria equipa de desenvolvimento, e conta com o apoio de uma comunidade de cerca de 100.000 programadores [18]. Estão a inserir processamento de dados em tempo real, análises preditivas, *Machine Learning* e aplicações baseadas em eventos.

Fraquezas

O modelo comercial de *Low-Code* é considerado fraco, e oferece aos clientes opções limitadas na implementação das aplicações [18].

3.8.10 TrackVia

É caracterizado por ser *"Niche Player"*, foca-se principalmente em *Citizen Development* de baixo custo e orientado a processos. Foca-se também na simplicidade do desenvolvimento e tenta substituir o Microsoft Excel como caso de uso típico.

Forças

Plataforma de baixo custo, com foco na facilidade de desenvolvimento de modo a facilitar a entrega de aplicações sofisticados nas áreas de Web, dipositivos móveis e processos de negócio. Fornece soluções móveis para operações de campo e *exploits* do *Apache Groovy* para extensões. Para além do desenvolvimento de aplicações utilizando o *Low-Code,* também utiliza abordagens centradas em processos de modo a permitir *Citizen Development,* unidades de negócio e desenvolvimento de tecnologias de informação.

Fraquezas

Utiliza uma experiência de utilizador do tipo *BPM-type* e *portal-style* para todas as aplicações ou processos alocados ao utilizador, o que acaba por gerar uma falta de independência e algumas limitações para alguns casos de uso. Embora esteja a explorar alguns tópicos avançados como o desenvolvimento assistido por IA, é menos avançado no suporte a funções de IA para o armazenamento de dados no sistema ou suporte para *Developer Personas.*

4. ABORDAGEM TECNOLÓGICA

4.1 Ecossistema Web

A arquitetura de uma aplicação web é a estrutura composta pelas interações e relacionamentos entre os elementos dessa aplicação.

Assim, para a criação e estruturação desta aplicação web a arquitetura que mais se enquadra é uma arquitetura em microsserviços que consiste em decompor uma aplicação em pequenas partes independentes e autónomas. Esta arquitetura permite às aplicações funcionar de uma forma mais rápida e eficiente, possibilitando também uma descentralização da administração que permite, por exemplo, usar duas linguagens de programação diferentes entre o *Back-End* e o *Front-End.*

4.2 Ferramentas Utilizadas

Após as análises realizadas anteriormente e as reuniões com os orientadores selecionou-se as ferramentas mais pertinentes e eficazes para o correto desenvolvimento deste projeto.

4.2.1 Programação em Node.js

O *Back-End* consiste na parte do projeto voltada para o funcionamento interno do mesmo, ou seja, tudo o que está por trás da interface de uma aplicação. Deste modo, o desenvolvimento do *Back-End* deste projeto será realizado utilizando o Node.js que é uma plataforma de execução de JavaScript voltada para *Back-End.*

O *Node.js* executa o mecanismo JavaScript V8 baseado no núcleo do Google Chrome, fora do navegador Web. As aplicações correm apenas num processo (s*ingle-thread)* permitindo assim executar o código sem criar novas *threads* para cada pedido e sem bloqueios, ou seja, não precisa de esperar que outras tarefas terminem para executar, o que torna a execução rápida.

Para tal, foi necessário instalar o Node.js, que pode ser obtido através do próprio website, https://nodejs.org.

4.2.2 Framework de interface

Das diversas *frameworks* para realização do front-end em JavaScript, onde todas elas têm como objetivo facilitar o desenvolvimento da interface e torná-la mais dinâmica para o utilizador, foi selecionado o Vue.js. Esta decisão foi tomada após um período de pesquisa onde se concluiu, com a ajuda do professor, que com o Vue.js a curva de aprendizagem seria menor que as restantes e também pelo facto de o Vue.js estar a ser utilizado cada vez mais pela comunidade.

4.2.3 Base de dados não relacional – MongoDB

O modelo de base de dados utilizado foi o não relacional, nomeadamente, o MongoDB, implementada no MongoDB Atlas. De modo a gerir esta base de dados utilizou-se o software online do MongoDB Atlas e também o software local MongoDB Compass, que permite gerir e interagir a base de dados.

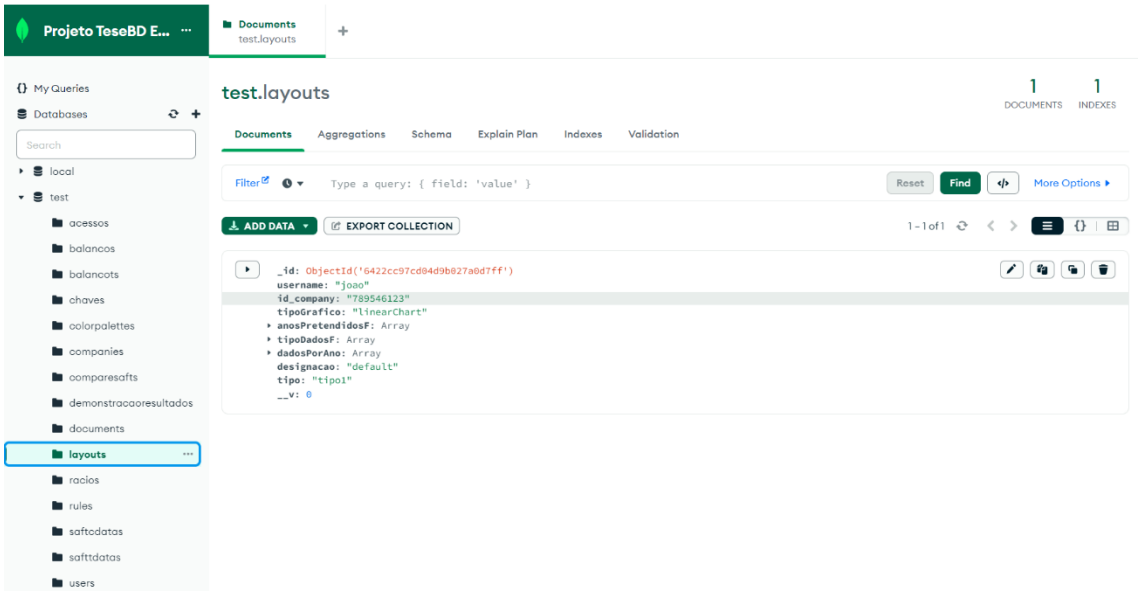


Figura 4 - Interface MongoDB Compass

4.2.4 Editor de texto

Todo o código da aplicação foi desenvolvido utilizando o editor de texto Visual Studio Code.

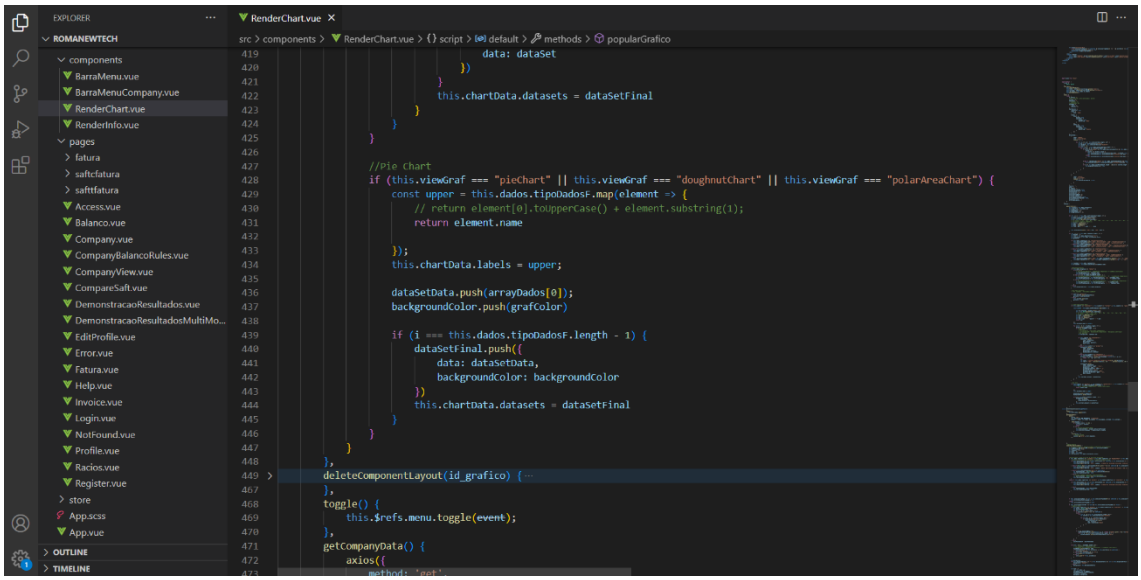


Figura 5 - Interface Visual Studio Code

5. DESENVOLVIMENTO DO PROJETO E RESULTADOS OBTIDOS

Este capítulo tem como objetivo descrever os principais procedimentos realizados para a implementação do projeto, onde serão mencionados os modelos de bases de dados criados e implementados para o correto funcionamento da aplicação e as experiências e resultados obtidos, onde será demonstrada toda a aplicação e o seu funcionamento.

Neste projeto foi proposto uma abordagem *Low-Code* efetuando a prova de conceito em diferentes casos de uso na área de desenvolvimento Web, nomeadamente nas áreas de *Fintech*, para tal foi desenvolvida uma plataforma WEB para aplicação numa *fintech*, onde seja permitido ao utilizador adicionar, remover e editar uma *dashboard* composta por gráficos e cartões (Widgets) de forma simples e intuitiva, permitindo assim criar uma *dashboard* que responda a todas as necessidades do utilizador sem ter de criar ou editar código. Sendo também possível aplicar esta interface em outras áreas sem a necessidade de grandes alterações por parte do programador, alterações essas descritas neste capítulo. (*Low-Code*).

Estes gráficos e cartões são alimentados pelos dados de uma plataforma previamente criada, onde foi reaproveitado parte do código dessa mesma plataforma, nomeadamente, o upload de ficheiros SAFT, a visualização dos valores de balanço, demonstração de resultados e rácios e foi implementado nesta plataforma. Esse código gera dados de balanço, demonstração de resultados e rácios entre balanço e demonstração de resultados de empresas com base nos ficheiros SAFT.

Deste modo, todos os gráficos e cartões personalizados pelo utilizador tem os dados demonstrados com base nesses balanços e rácios.

5.1 Arquitetura

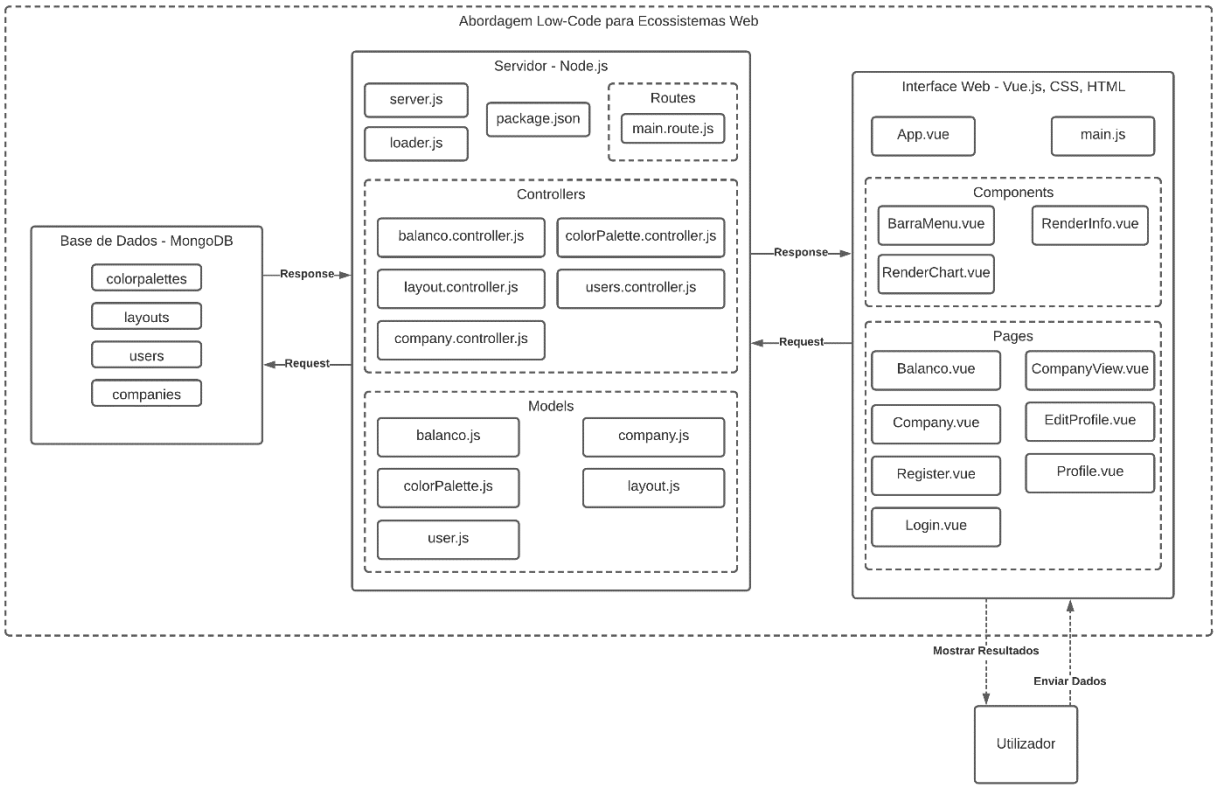


Figura 6 - Arquitetura

De modo a realizar o projeto proposto, foi desenvolvida uma arquitetura para dar resposta ao problema apresentado. Assim, tal como podemos observar na Figura 6 a arquitetura consiste em três grandes categorias, sendo elas a Base de dados, configurada na plataforma MongoDB, o servidor Javascript configurado em Node.js e a interface Web construída em Vue.js, CSS e HTML. Esta arquitetura representa também o processo desde a interação do cliente com a plataforma até aos pedidos na base de dados.

A base de Dados Mongo é constituída por quatro tabelas fundamentais, tabela "colorpalettes" responsável por armazenar as informações relativas aos esquemas de cores predefinidos e também personalizados pelos utilizadores. Tabela "layouts", responsável por armazenar todos os layouts de cada um dos utilizadores, ou seja, é a tabela que contém os valores de cada componente criado na *dashboard*, contendo os gráficos e cartões que um determinado utilizador adiciona na *dashboard* de uma empresa. Tabela "users" responsável por armazenar as informações relativas a cada

utilizador. A tabela “companies” responsável por armazenar a informação de cada empresa. Esta base de dados recebe e responde aos pedidos vindos do servidor.

O servidor, desenvolvido e configurado em Node.js, é responsável por receber e responder a todos os pedidos vindos da interface web e também realiza pedidos e recebe respostas da base de dados MongoDB. Este servidor utiliza os ficheiros demonstrados na Figura 6, onde estes estão divididos por grupos, dos quais temos, o grupo “Routes”, caracterizado apenas por um ficheiro responsável por gerir e encaminhar os pedidos realizados ao servidor para cada uma das respetivas funções, ou seja, as rotas. O grupo “Controllers” onde estão presentes todas as funções que processam e respondem a eventos. O grupo “Models” onde está presente como os dados estão guardados na base de dados. O ficheiro “server.js” é o ficheiro responsável pelo funcionamento e arranque do servidor e também do acesso à base de dados, que por sua vez acede ao ficheiro “loader.js” de modo a carregar todas as dependências e bibliotecas necessárias antes do servidor ficar pronto a responder aos pedidos realizados pela interface web. Por fim, o ficheiro “Package.json” corresponde a todas as dependências utilizadas pelo servidor, bem como a respetiva versão.

A interface web, desenvolvida em Vue.js, CSS e HTML, é responsável pela parte visual e interativa do projeto, sendo a responsável por mostrar ao cliente uma interface interativa, onde esta realiza pedidos e recebe respostas do servidor. Esta interface é composta por dois ficheiros necessários para o arranque da mesma, o “App.vue” e o “main.js”. De seguida, temos os “components” que são instâncias reutilizáveis do Vue.js, ou seja, permitem ao programador reutilizar código sem ter de o repetir, como é o caso dos gráficos, por exemplo, e por fim temos as “pages” que são as páginas mostradas ao utilizador, que por sua vez integram os componentes.

5.2 Modelos de Base de Dados

Para o correto funcionamento deste projeto quer na sua fase final quer no desenvolvimento foram criadas quatro tabelas, nomeadamente, “colorpalettes”, “layouts”,” users” e “companies”. A tabela “colorpalettes” contém os valores referentes aos padrões e temas de cores de cada empresa referente a cada utilizador. A tabela “layouts” contém os valores de cada componente criado na *dashboard*, ou seja, contem os gráficos e cartões que um determinado utilizador adiciona na *dashboard* de uma empresa. A tabela “users” contém os campos e valores referentes a cada perfil de utilizador. A tabela “companies” contém os campos e valores referentes a cada empresa adicionada pelo utilizador.

A tabela *colorpalettes*, representada abaixo na Tabela 1, contém os valores referentes aos padrões e temas de cores de cada empresa referente a cada utilizador.

Tabela 1 - Tabela “colorpalettes”

Atributos	Descrição	Tipo de dados
username	Username do utilizador	String
id_company	ID da empresa	String
color1	Cor 1 do esquema de cores	String
color2	Cor 2 do esquema de cores	String
color3	Cor 3 do esquema de cores	String
color4	Cor 4 do esquema de cores	String
color5	Cor 5 do esquema de cores	String
color6	Cor 6 do esquema de cores	String

A tabela *layouts,* representada abaixo na Tabela 2, contém os valores de cada componente criado na *dashboard*, ou seja, contem os gráficos e cartões que um determinado utilizador adiciona na *dashboard* de uma empresa.

Tabela 2 - Tabela "layouts"

Atributos		Descrição	Tipo de dados
username		Username do utilizador	String
id_company		ID da empresa	String
tipoGrafico		Tipo de gráfico selecionado pelo utilizador	String
anosPretendidosF		Array com os anos que o utilizador pretende observar no gráfico	Array String
tipoDadosF		Array com os dados que o utilizador pretende observar no gráfico	Array de String
dadosPorAno			Array Object
	tipoDadosF	Array com os valores dos anos refentes ao tipo de dados selecionados	Array de int
	Cor	Cor selecionada pelo utilizador, no entanto, caso não selecione nenhuma cor em específico o valor é *"default"* e utiliza o valor da paleta de cores pré-definida.	String
designacao		Designação personalizada para o gráfico ou cartão, caso o utilizador não especifique o	String

		valor é *"default"* e é definida automaticamente.	
Tipo		Distingue se é gráfico ou cartão, podendo apresentar dois valores, *"tipo1"* e *"tipo2",* onde *"tipo1"* representa os gráficos e *"tipo2"* representa os cartões.	String

A tabela *users,* representada abaixo na Tabela 3, contém os valores referentes a cada um dos utilizadores registados na plataforma.

Tabela 3 - Tabela "users"

Atributos	Descrição	Tipo de dados
username	Username	String
email	Email do utilizador utilizado no login	String
password	Password utilizada no login	String
profession	Profissão	String
address	Endereço	String
city	Cidade	String
country	Localidade	String
last_name	Sobrenome	String
name	Nome	String
phone	Número Telemóvel	String
postalCode	Código Postal	String

A tabela *companies,* representada abaixo na Tabela 4, contém os valores referentes a cada uma das empresas adicionadas pelo utilizador. Cada uma das empresas está associada a um determinado utilizador, pelo que apenas o próprio consegue ver as empresas que adiciona.

Tabela 4 - Tabela "companies"

Atributos	Descrição	Tipo de dados
id_company	Id diferenciador de cada empresa, sendo que é igual ao nif	String
user	Utilizador que adicionou a empresa	String
nif	NIF da empresa	String
name	Nome	String
description	Descrição da atividade da empresa	String
adress	Endereço	String
accountSaftc	Valor de contas exportado do Saftc	String
clientSaftc	Clientes exportados do Saftc	String
salesSaftc	Vendas exportadas do Saftc	String
clientSaftt	Clientes exportados do Saft	String
invoiceSaftt	Fatura exportada do Saft	String
productsSaftt	Produtos exportados do Saft	String

5.3 Experiências e resultados obtidos

Neste capítulo são apresentados os principais resultados obtidos com o desenvolvimento do projeto, onde é demonstrado de forma detalhada o funcionamento e as potencialidades da plataforma.

5.3.1 Login e Registo

A aplicação web inicia com a página de login onde são pedidos os campos “email” e “password”, bem como um botão de Registo e de “Forget Password” que permite ao utilizador efetuar o registo na aplicação e recuperar a password, respetivamente, caso se tenha esquecido.



Figura 7 - Página de login

A página de registo conta com os campos “Username” que corresponde ao nome de utilizador pretendido para a aplicação, sendo que este nome não pode ser alterado futuramente, pois é o campo que associa todas as informações ao utilizador. É também necessário preencher o campo “Email” que corresponde ao email do utilizador, sendo que este irá ser utilizado para iniciar sessão na plataforma, como se pode ver na imagem anterior. A “Password” que corresponde à palavra-passe pretendida, “Confirm Password” à reinserção da password e “Profession” à profissão do utilizador. No caso

deste último campo existe uma série de campos predefinidos, podendo sempre ser selecionado a opção “outro”.



Figura 8 - Página de registo

5.3.2 Página Inicial

Após o login demonstrado no ponto anterior, surge a página inicial, onde são apresentadas todas as empresas pertencentes ao utilizador, ou seja, é apresentada uma lista onde mostra todas as empresas adicionadas, previamente, pelo utilizador, bem como alguns valores referentes a cada uma dessas empresas.

Através desta página, utilizando o botão de “Ver detalhe” demonstrado na Figura 9, o utilizador é capaz de aceder a uma página específica com informação detalhada de cada empresa, demonstrado no próximo ponto. É também nesta página que o utilizador pode adicionar uma nova empresa a este menu, através da opção “New Company”, onde surge um novo menu representado na Figura 10. Nesta página e nas seguintes também é possível aceder ao menu do utilizador, posicionado no canto superior direito.

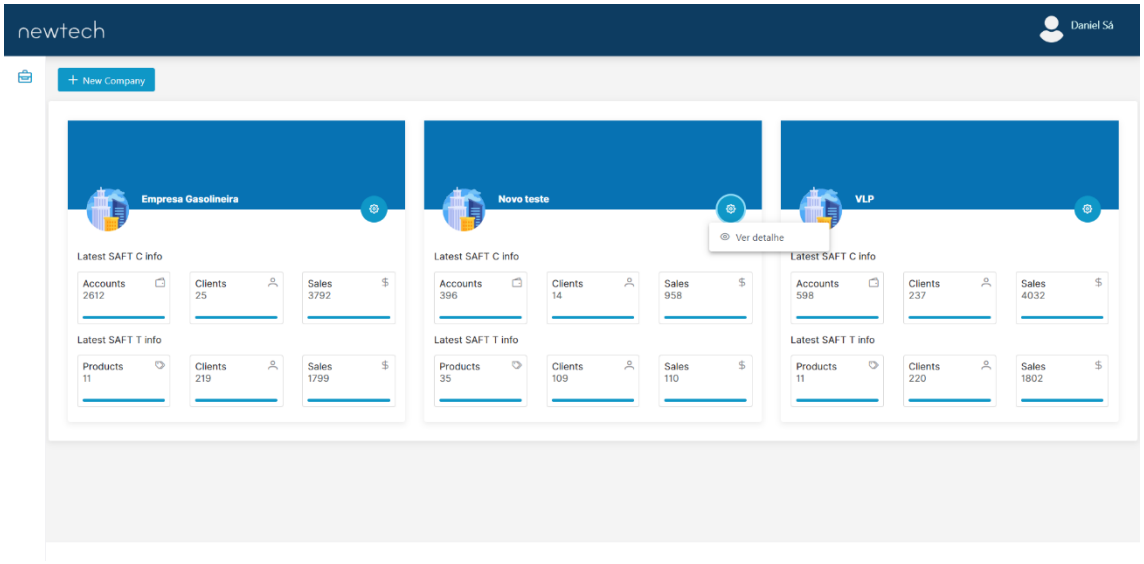


Figura 9 - Página Inicial com todas as empresas do utilizador e opção de ver detalhe da empresa

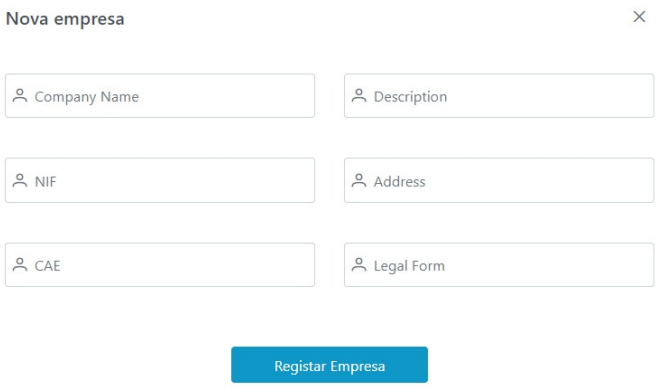


Figura 10 - Menu para registar nova empresa

5.3.3 Dashboard - Empresa selecionada na imagem anterior

Esta página, representa a *dashboard*, personalizada pelo utilizador, da empresa selecionada na página inicial através do botão "Ver detalhe", tal como demonstrado na Figura 9. Nesta página é possível visualizar um conjunto de gráficos e *widgets* personalizados, escolhidos e adicionados pelo utilizador com recurso a poucos cliques. Está página tem como objetivo demonstrar os ponto-chave definidos pelo utilizador para esta empresa em específico.

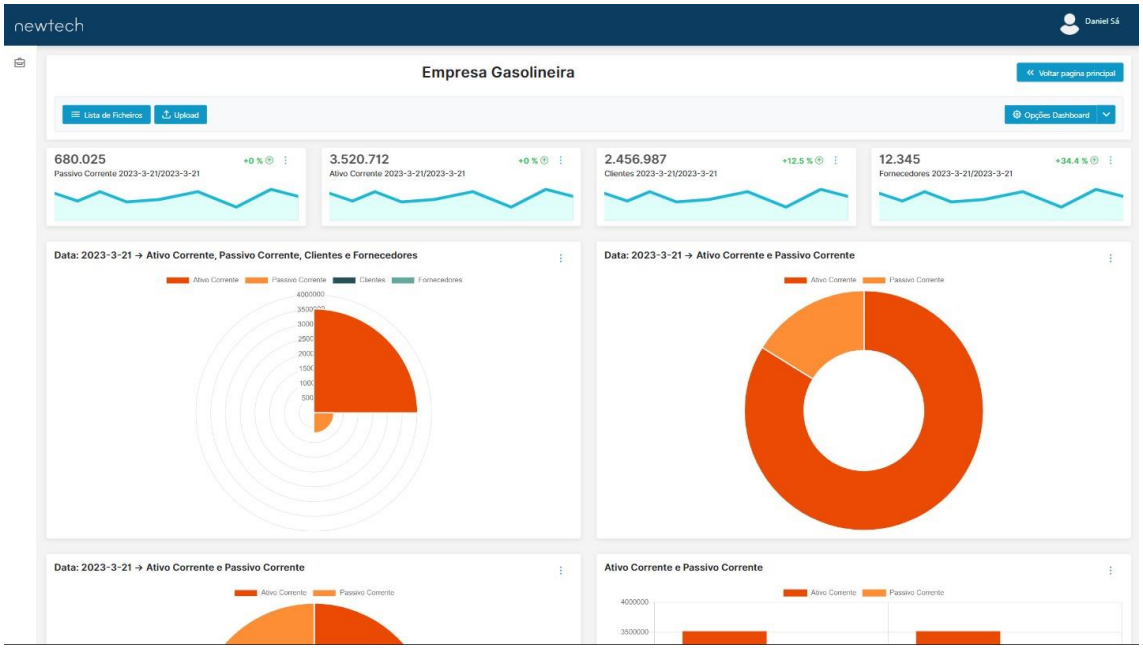


Figura 11 - Dashboard personalizada pelo utilizador para a empresa selecionada

De uma forma simples e intuitiva o utilizador pode adicionar novos gráficos e *widgets* a esta *dashboard*, bem com editar o esquema de cores utilizado e partilhar este layout (caracterizado pelo conjunto de gráficos e *widgets*) com qualquer uma das suas empresas, tudo isto através do botão "Opções Dashboard" localizado na parte superior direita desta *dashboard,* tal como demonstrado na Figura 12.

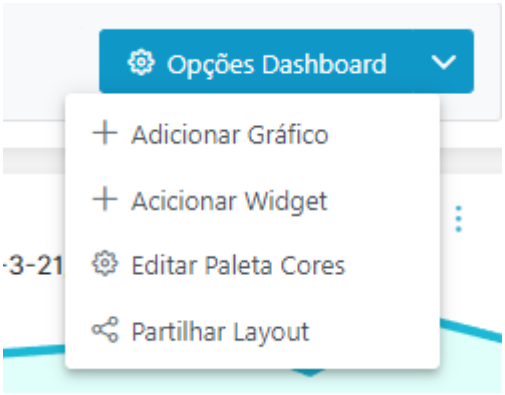


Figura 12 - Botão "Opções Dashboard" e respetivo menu

5.3.4 Menu Adicionar Gráfico

A opção “Adicionar Gráfico”, do menu demonstrado na Figura 12, abre um novo modal com algumas opções. Das quais, na primeira é possível adicionar uma designação específica para esse gráfico, sendo que também é possível deixar esta opção em branco e a designação do mesmo será preenchida de forma automática consoante os dados selecionados para este gráfico e a data dos mesmos.

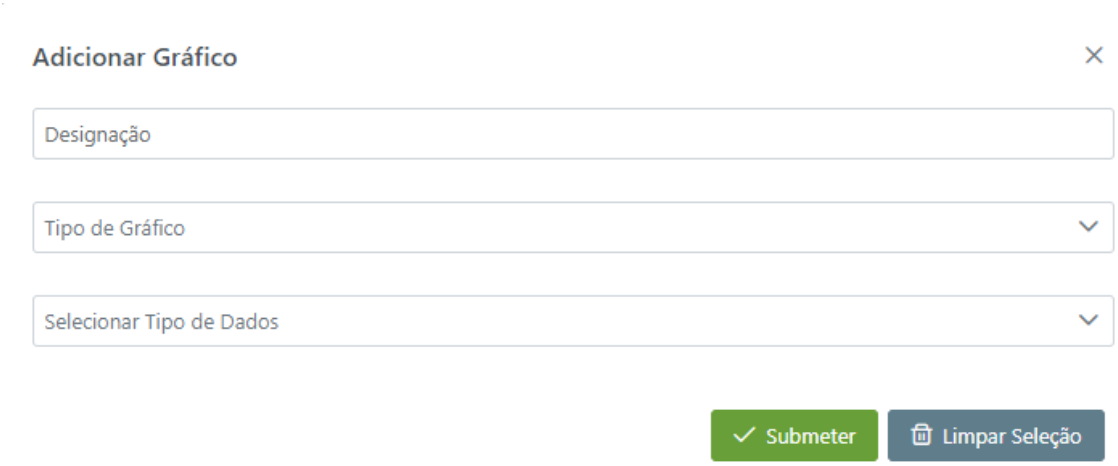


Figura 13 - Modal para adicionar gráfico

A segunda opção permite ao utilizador selecionar o tipo de gráfico que pretende adicionar, onde as opções são as seguintes:

- Linear Chart, caracterizado por ser um gráfico com um conjunto de pontos unidos por linhas.

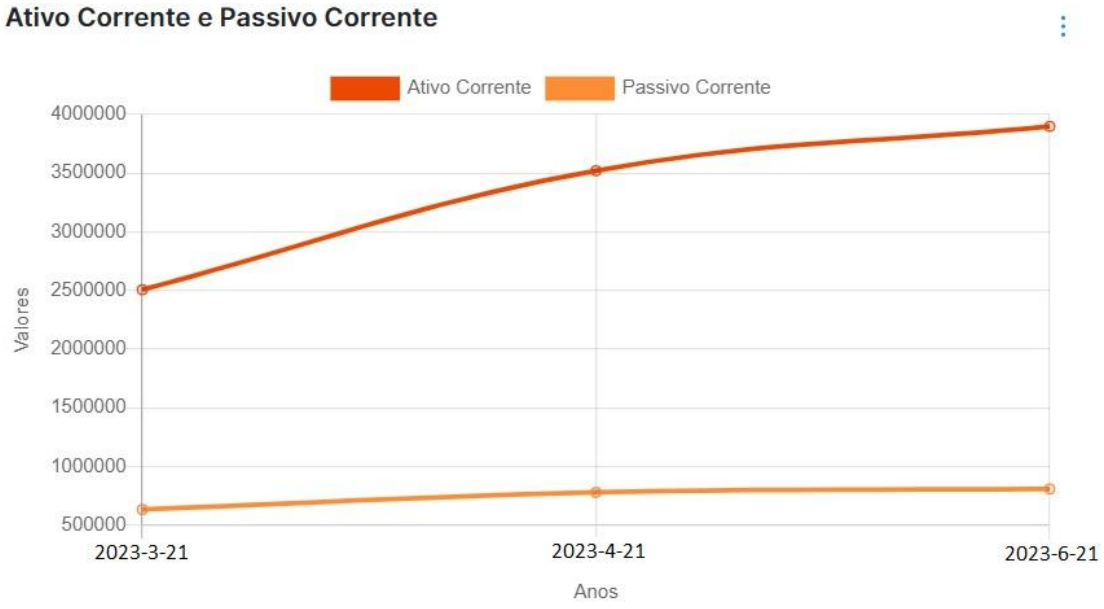


Figura 14 - Exemplo de gráfico do tipo "Linear Chart"

- Bar Chart, caracterizado por ter barras retangulares e comprimento proporcional aos valores que representa.

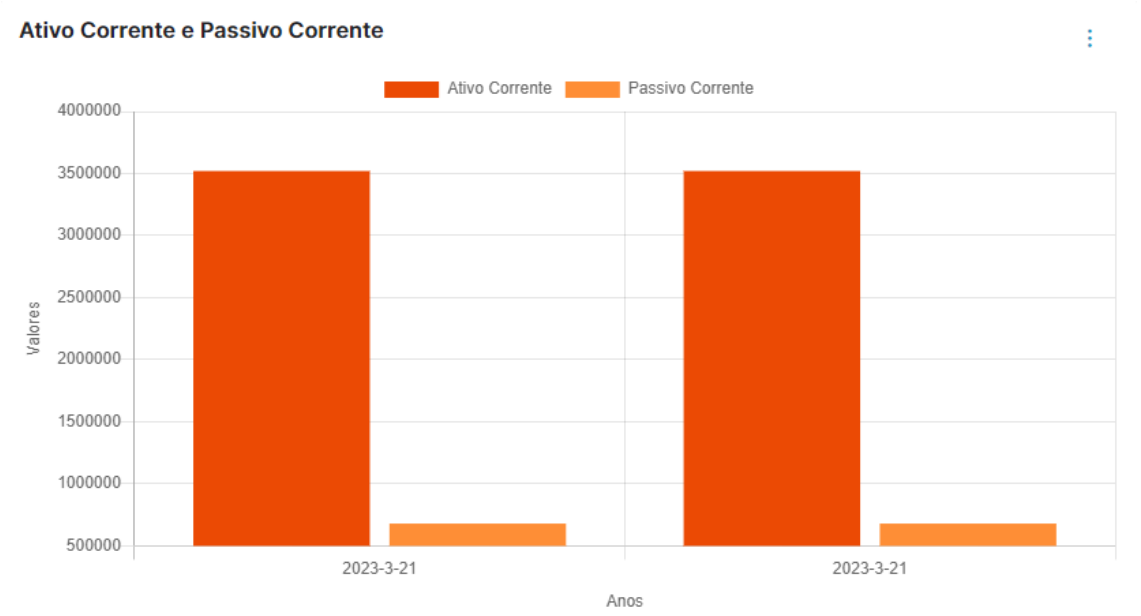


Figura 15 - Exemplo de gráfico do tipo "Bar Chart"

- Pie Chart, caracterizado por ser um gráfico de setores ou gráfico circular, é tradicionalmente chamado gráfico de pizza onde os valores de cada categoria estatística representada são proporcionais às respetivas medidas dos ângulos.

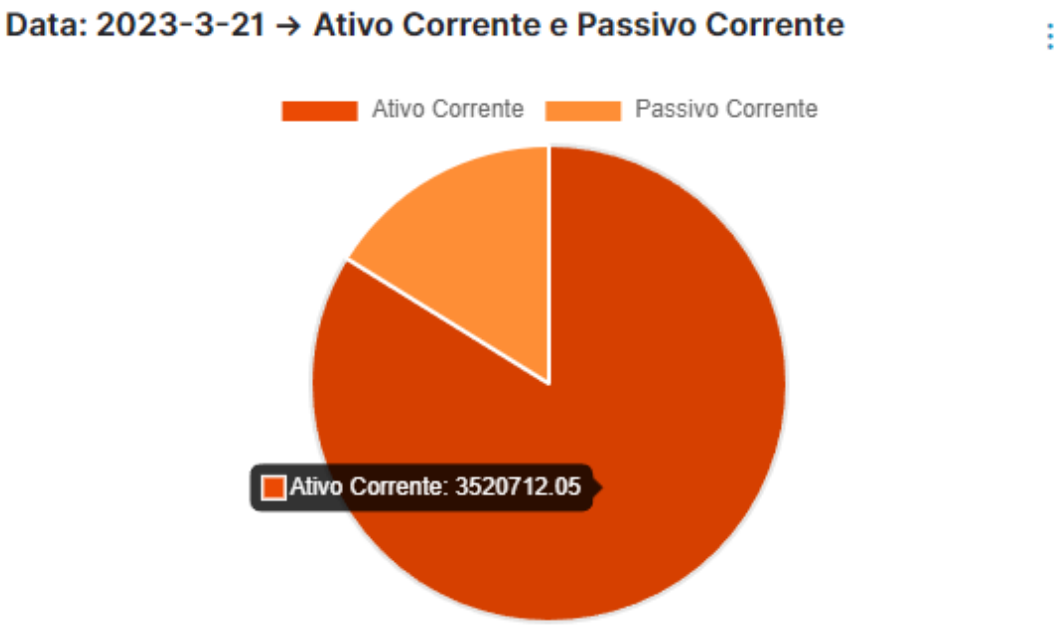


Figura 16 - Exemplo de gráfico do tipo "Pie Chart"

- Doughnut Chart, caracterizado por ser um gráfico de setores e com o formato e funcionalidade igual ao “Pie Chart” apresentado anteriormente, no entanto com o centro cortado, alusivo ao nome “Doughnut”.

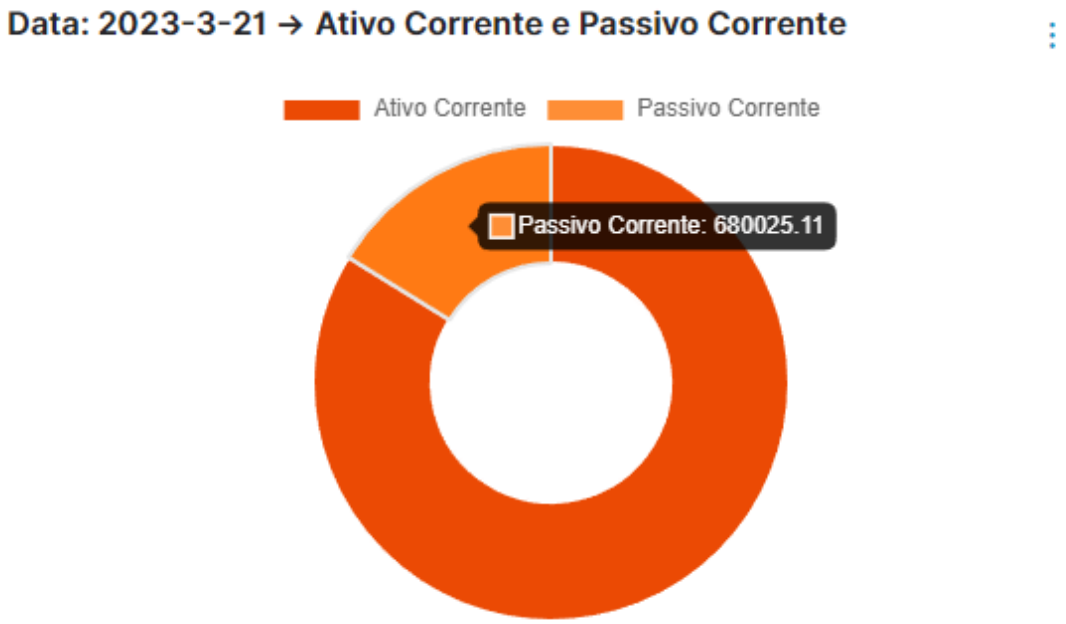


Figura 17 - Exemplo de gráfico do tipo "Doughnut Chart"

- Polar Area Chart, caracterizado por ser idêntico aos dois gráficos anteriores, no entanto, cada um dos segmentos tem o mesmo ângulo, variando o raio do segmento consoante o valor que representa.

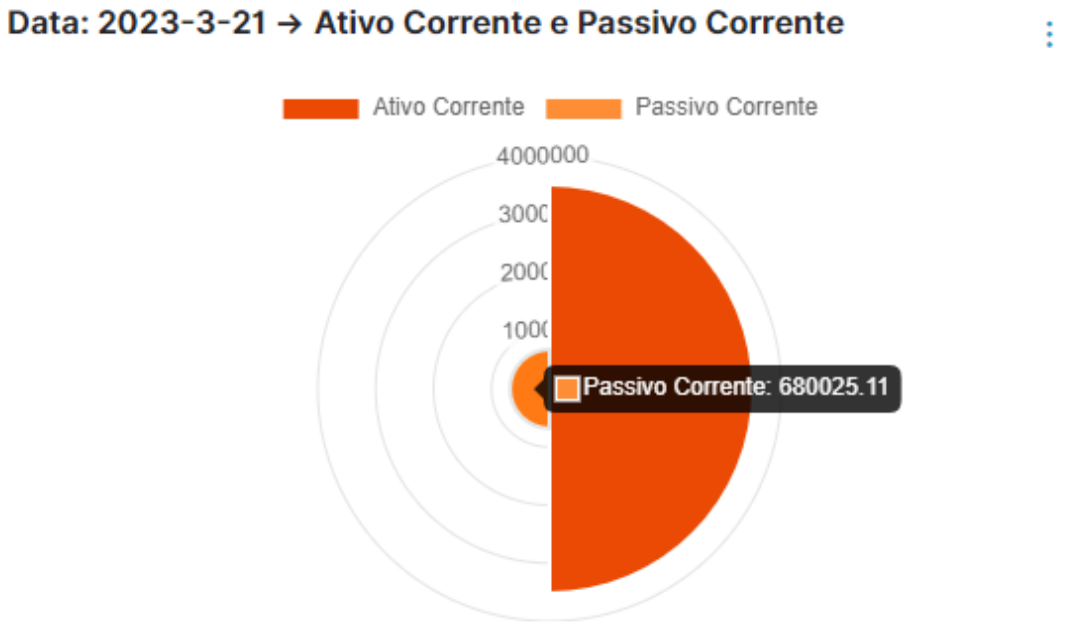


Figura 18 - Exemplo de gráfico do tipo "Polar Area Chart"

- Radar Chart, caracterizado por ser um gráfico de radar e de mostrar vários pontos de dados e as variações entre eles.

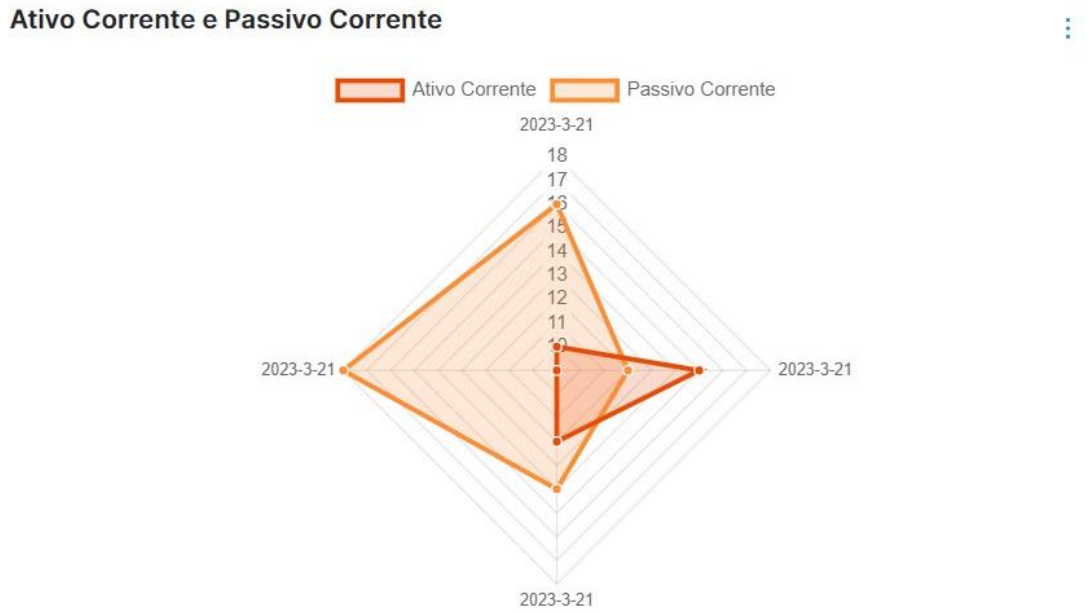


Figura 19 - Exemplo de gráfico do tipo "Radar Chart"

Com base na seleção do tipo de gráfico surge um novo campo, que permite ao utilizador selecionar qual a fonte de dados que quer utilizar. Esta opção surge apenas após a seleção do gráfico pois dos gráficos disponíveis alguns tem a capacidade de mostrar os dados com base em múltiplas fontes de dados, nomeadamente o “Linear Chart”, o “Bar Chart” e o “Radar Chart”, os restantes apenas podem aceitar uma fonte de dados.

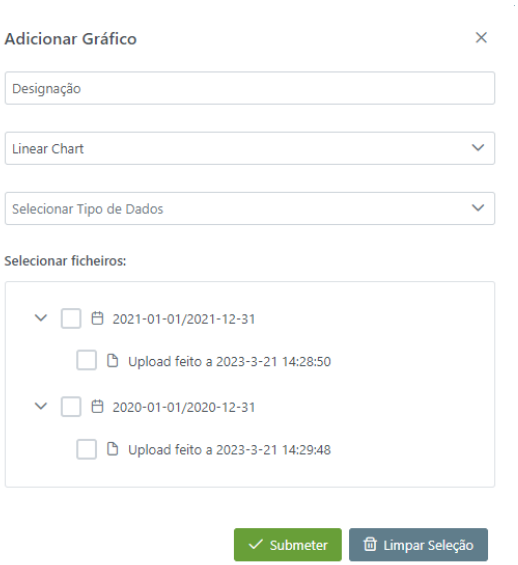


Figura 20 - Exemplo do campo que surge quando a seleção do tipo de gráfico é “Linear Chart”, “Bar Chart” ou “Radar Chart”

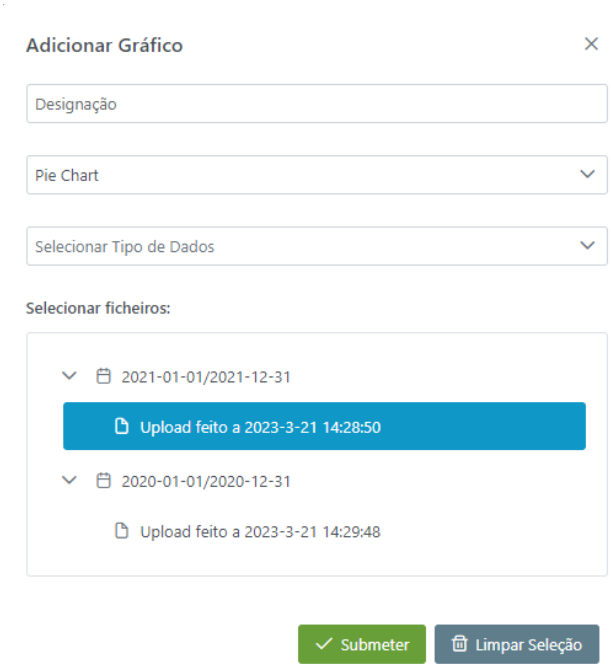


Figura 21 - Exemplo do campo que surge quando a seleção do tipo de gráfico é "Pie Chart", "Doughnut Chart" ou "Polar Area Chart"

Por fim, é possível selecionar ainda o tipo de dados que quer demonstrar. Com base nos dados existentes na base de dados, é criada uma lista com todos os valores disponíveis em cada um dos conjuntos de dados.

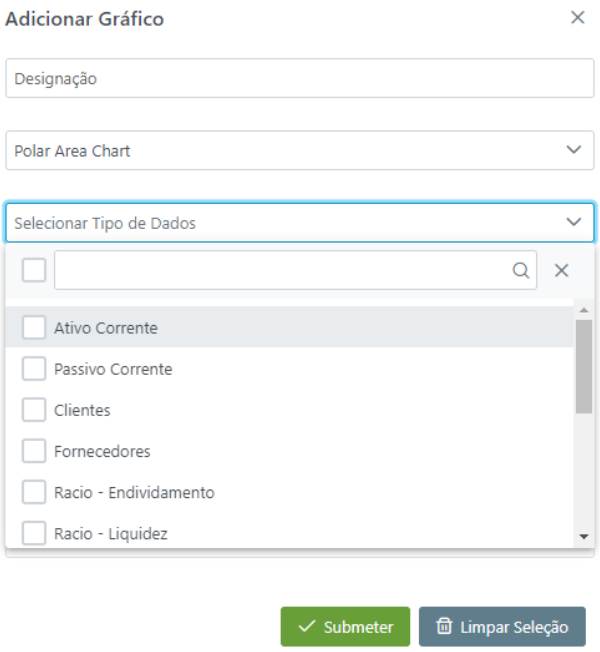


Figura 22 - Seleção do tipo de dados a demonstrar no gráfico

Após a seleção estar terminado e o utilizador submeter o gráfico irá aparecer na dashboard tal como o utilizado o configurou.

5.3.5 Adicionar Widget

Nesta opção do menu “Opções Dashboard” representado na Figura 12, tal como nos gráficos, surge um novo modal. A primeira opção deste modal permite adicionar uma designação especifica para esse widget, ou deixar esta opção em branco e a designação do mesmo é automática consoante os dados selecionados.

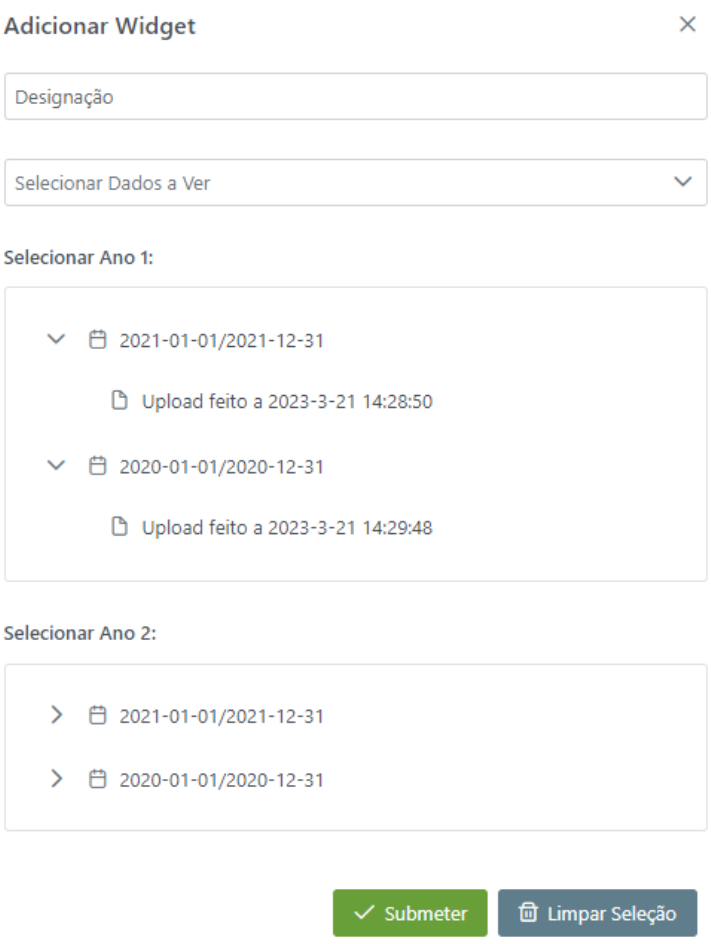


Figura 23 - Modal para adicionar novo Widget

É necessário de seguida selecionar os dados que pretende ver, tal como nos gráficos, estes dados surgem com base nos dados existentes na base de dados, onde é criada uma lista com todos os valores disponíveis em cada um dos conjuntos de dados, e apenas é possível selecionar um para comparação entre duas fontes de dados diferentes.

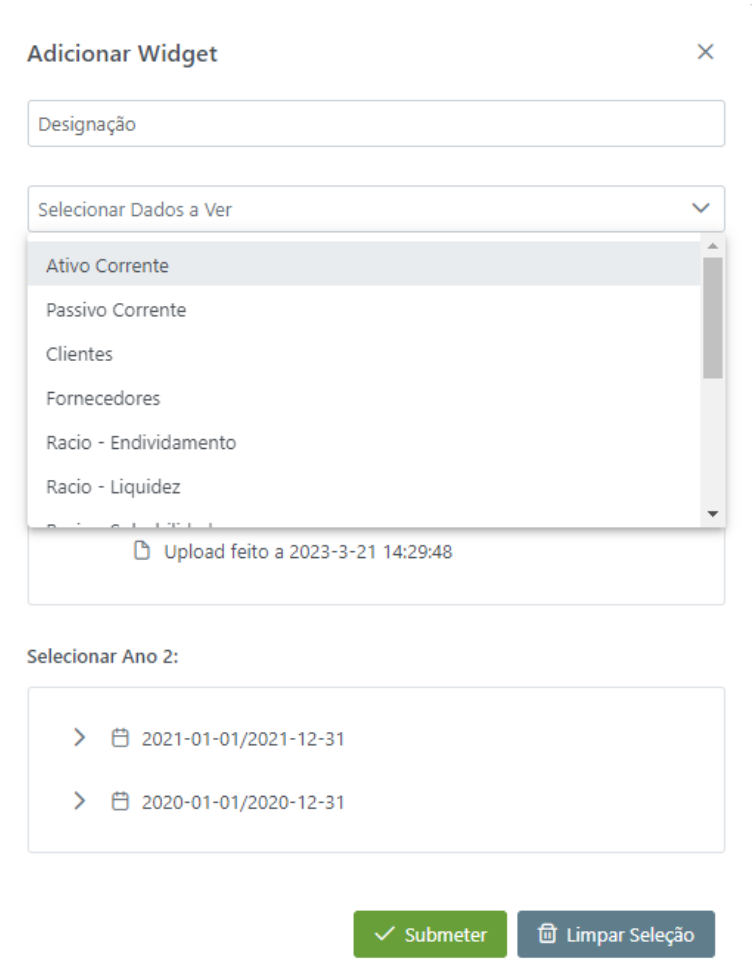


Figura 24 - Opção de selecionar o tipo de dados a ver

Por fim, é necessário selecionar a fonte de dados para o Ano de comparação 1 e 2, ou seja, para o cálculo das estatísticas demonstradas nestes *widgets* é realizada uma subtração do ano dois menos o ano um (2-1) e de seguida o cálculo da percentagem, permitindo ao utilizador saber se o valor aumentou ou diminuiu.

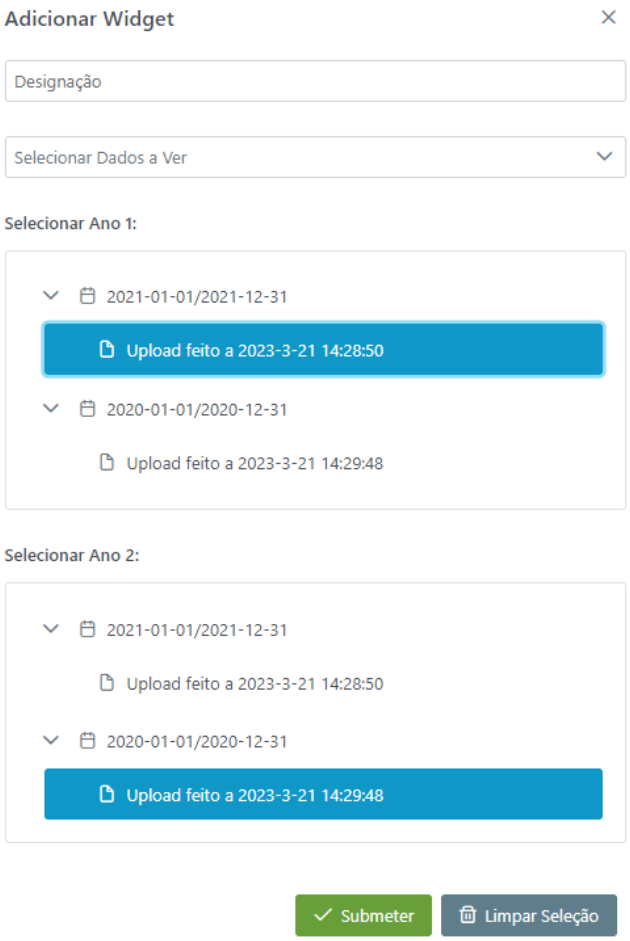


Figura 25 - Exemplo de seleção dos valores para o Ano 1 e para o Ano 2

Após a seleção dos valores pretendidos e de carregar no botão submeter, o modal será fechado e o utilizador irá ver o *widget* na *dashboard*, tal como o configurou. Pode ser observado um exemplo destes *widgets* na Figura 11 e na Figura 26.



Figura 26 - Exemplo de Widget

5.3.6 Editar Paleta Cores

Nesta opção do menu “Opções Dashboard” representado na Figura 12, surge um novo modal com um conjunto de cores pré-definidas para aplicar nos gráficos e widgets da dashboard.



Figura 27 - Modal com as paletas de cores pré-definidas

No entanto, cada um destes conjuntos pode ser editado e selecionado para ser aplicado na *dashboard*, ou seja, caso se pretenda alterar apenas uma cor, por exemplo, no conjunto de cores denominado “Natural”, é necessário apenas selecionar essa cor e editar para a tonalidade pretendida com o auxílio a um seletor de cores, tal como demonstrado na Figura 28. Caso o utilizado pretenda pode editar todas as cores de um destes temas sugeridos.

Após colocar as cores da forma pretendida e submeter as alterações, o esquema de cores selecionado é aplicado em todos os gráficos da empresa em questão, exceto naqueles que o utilizador editou individualmente, como demonstrado mais a frente.

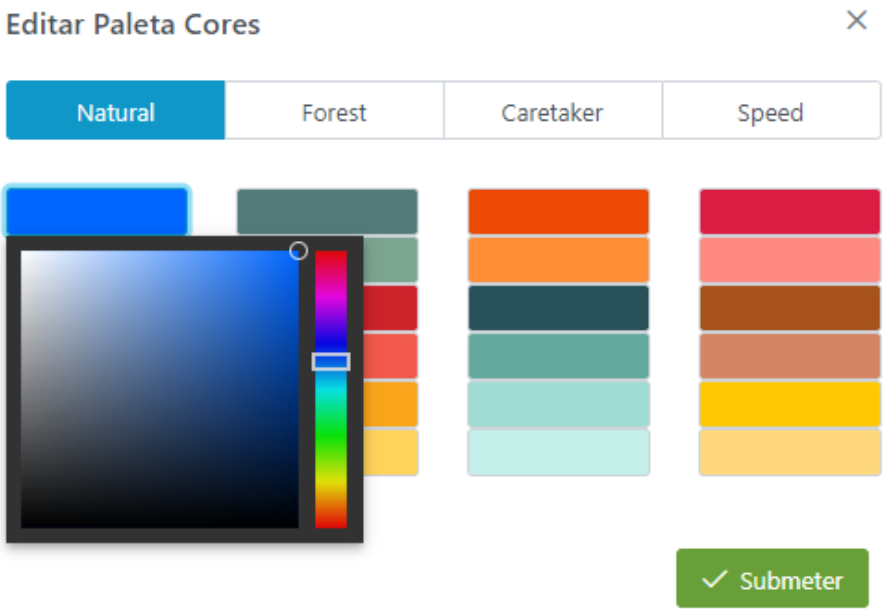


Figura 28 - Exemplo de alteração de cor com auxílio ao seletor de cores

5.3.7 Partilhar Layout

Nesta opção do menu "Opções Dashboard" representado na Figura 12, surge um novo modal com apenas um campo.

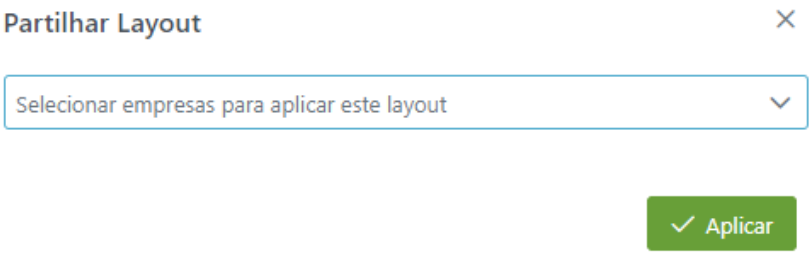


Figura 29 - Modal de Partilhar Layout

Neste campo, é possível selecionar a empresa, ou empresas deste utilizador onde pretende aplicar este layout, ou seja, é eliminado o layout existente na(s) empresa(s) selecionada(s) e aplicado este, com base nos dados dessa mesma empresa.

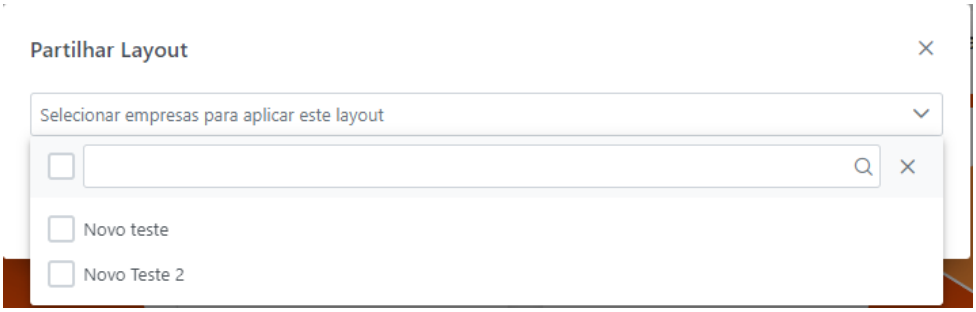


Figura 30 - Empresas possíveis de partilhar o layout

5.3.8 Opções de editar e eliminar para gráficos e widgets

Cada gráfico e *widget*, no canto superior direito, conta com um botão de opções que permite editar e eliminar o mesmo.

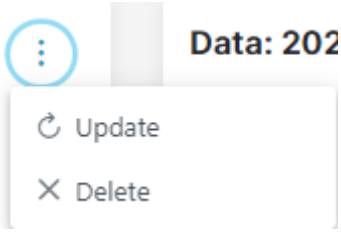


Figura 31 - Botão de opções para gráficos e widgets

5.3.9 Opção de eliminar gráfico ou widget

Aquando da seleção da opção “Delete” presente no menu apresentado na Figura 31, surge um modal que permite eliminar esse gráfico ou widget.

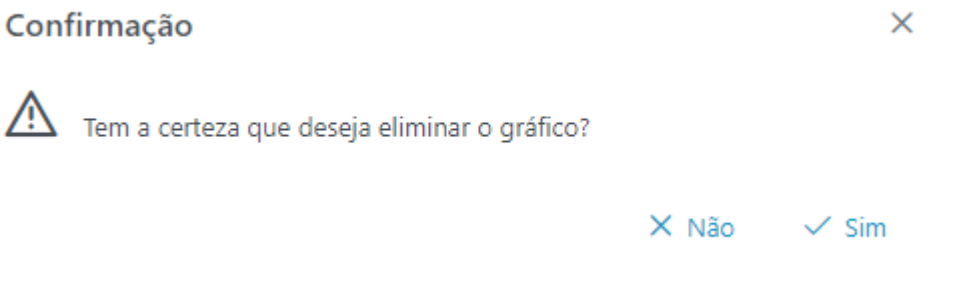


Figura 32 - Confirmação de eliminar gráfico

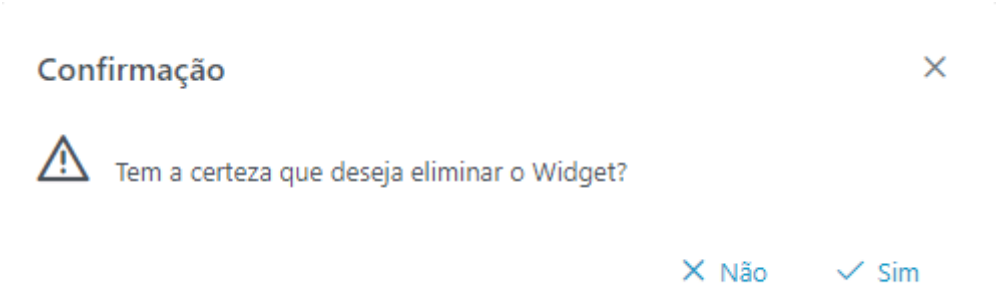


Figura 33 - Confirmação de eliminar widget

Assim, caso o utilizador pretenda eliminar esse componente da sua *dashboard* apenas tem de executar esta ação e selecionar a opção "Sim" no modal de confirmação.

5.3.10 Opção de Update para gráficos

Após selecionar a opção "Update", presente no menu apresentado na Figura 31, para um determinado gráfico surge um novo modal com quatro opções disponíveis, onde a opção de "Selecionar Ficheiros" é diferente consoante o tipo de gráfico que está a ser editado. Tal como na criação dos gráficos, esta opção difere, pois, dos gráficos disponíveis alguns tem a capacidade de mostrar os dados com base em múltiplas fontes de dados, nomeadamente o "Linear Chart", o "Bar Chart" e o "Radar Chart", os restantes apenas podem aceitar uma fonte de dados.

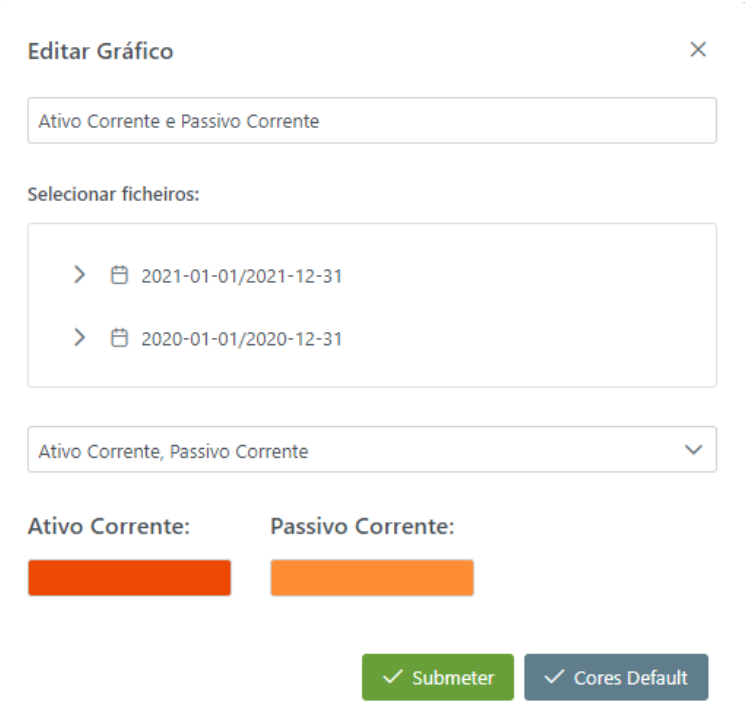


Figura 34 - Opção de editar gráfico quando o tipo de gráfico é "Pie Chart", "Doughnut Chart" ou "Polar Area Chart"

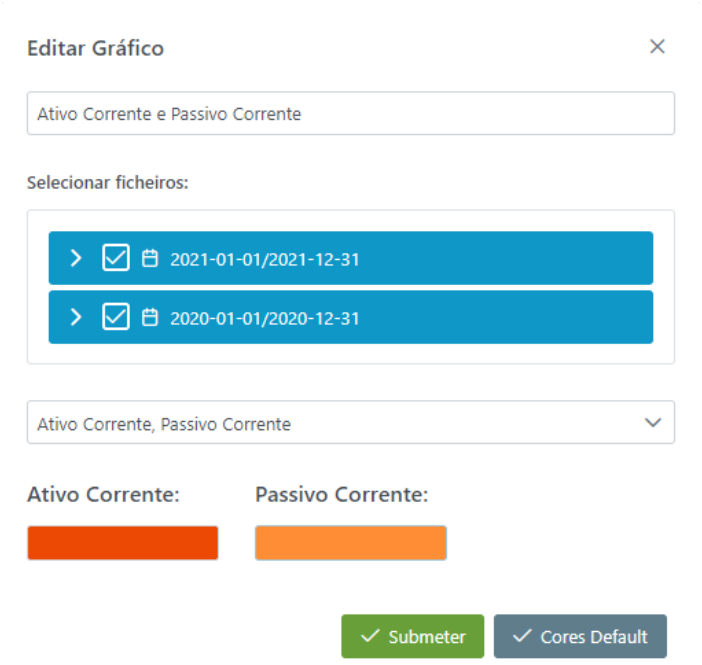


Figura 35 - Opção de editar gráfico quando o tipo de gráfico é "Linear Chart", "Bar Chart" ou "Radar Chart"

Existe também a opção de alteração da designação, demonstrada na Figura 34 e Figura 35 onde esta pode ter sido gerada automaticamente no modal de criação de gráfico e com base nos dados selecionados ou personalizada pelo utilizador, ou predefinida pelo utilizador aquando da criação.

É também possível editar o tipo de dados selecionados, com base nos valores existentes na base de dados.

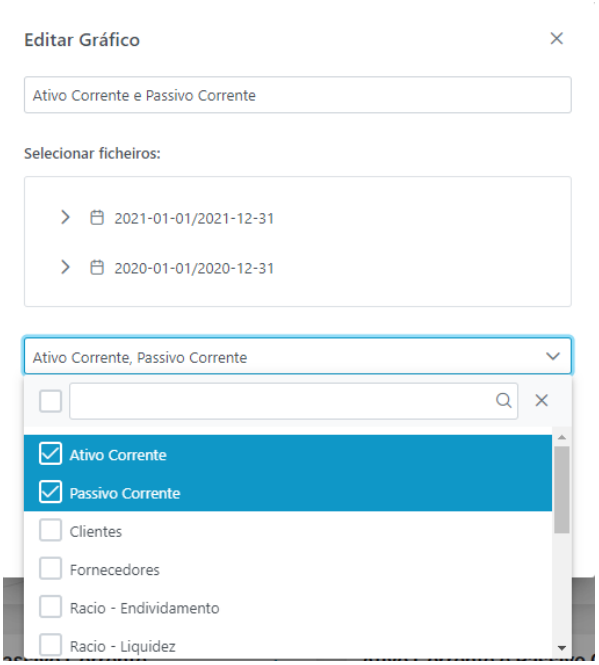


Figura 36 - Exemplo de tipos de dados selecionados

Por fim, o utilizador pode editar as cores de cada um dos dados selecionados de forma individual e totalmente personalizada com auxílio a um seletor de cores. As opções de edição de cores aumentam ou diminuem com base nos dados selecionados, tal como pode ser observado nas Figura 37 e Figura 38.

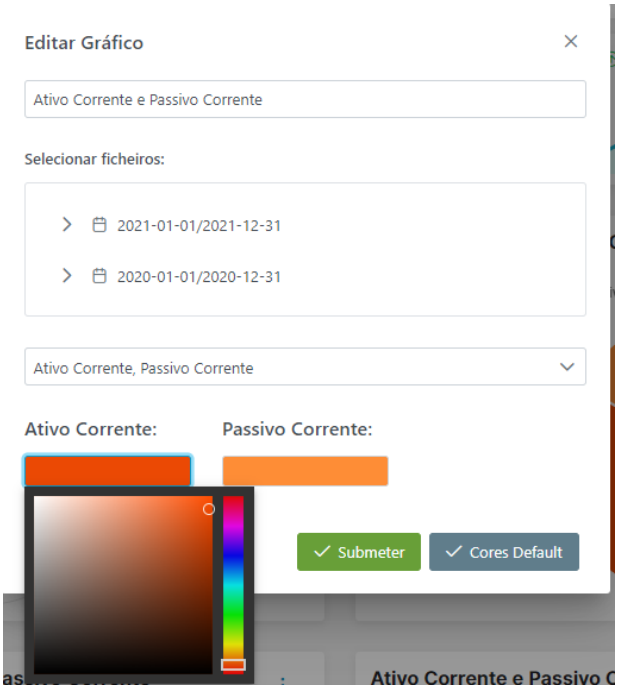


Figura 37 - Exemplo de alteração de cor de um determinado valor com auxílio a um seletor de cores



Figura 38 - Exemplo de alteração de cores com quatro tipos de dados selecionados

5.3.11 Opção de Update para Widgets

Após selecionar a opção “Update”, presente no menu apresentado na Figura 31, para um determinado *widget*, surge um novo modal onde é possível alterar os dados que estão a ser demonstrados, com base nos dados existentes na base de dados. Existe também a opção de alteração da designação, onde esta pode ter sido gerada automaticamente no modal de criação do *widget* e com base nos dados selecionados, ou personalizada pelo utilizador.

Por fim, é possível alterar a fonte de dados para o Ano de comparação 1 e 2, necessários para o cálculo das estatísticas demonstradas nestes *widgets*.

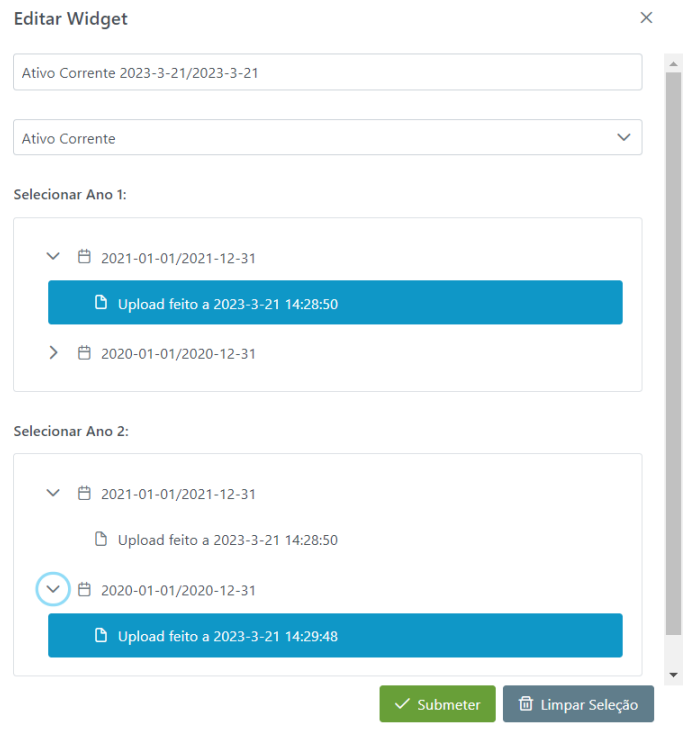


Figura 39 - Exemplo de modal para editar widget

5.3.12 Painel de utilizador

O painel de utilizador, acessível em todas as páginas da plataforma, permite ao mesmo visualizar e editar os seus dados, tal como podemos observar na Figura 40.

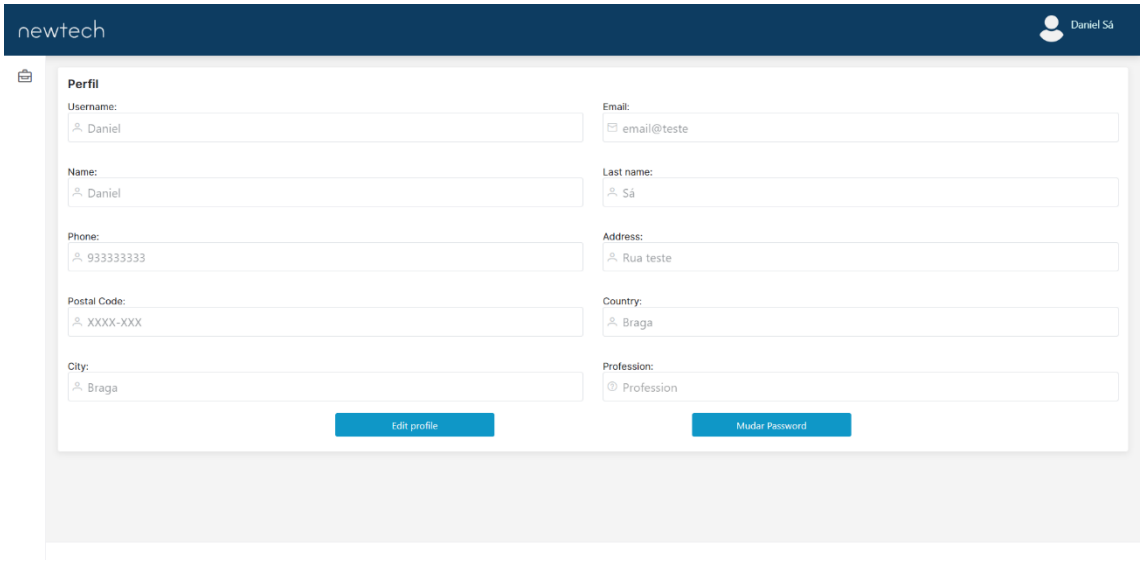


Figura 40 - Exemplo de visualização dos dados do utilizador

De modo a editar esses dados é necessário aceder ao botão "Edit Profile", onde ficam disponíveis as caixas de texto e os mesmos podem ser editados, tal como se pode observar na Figura 41. Para esta alteração ser possível é necessário inserir a palavra-passe atual do utilizador.

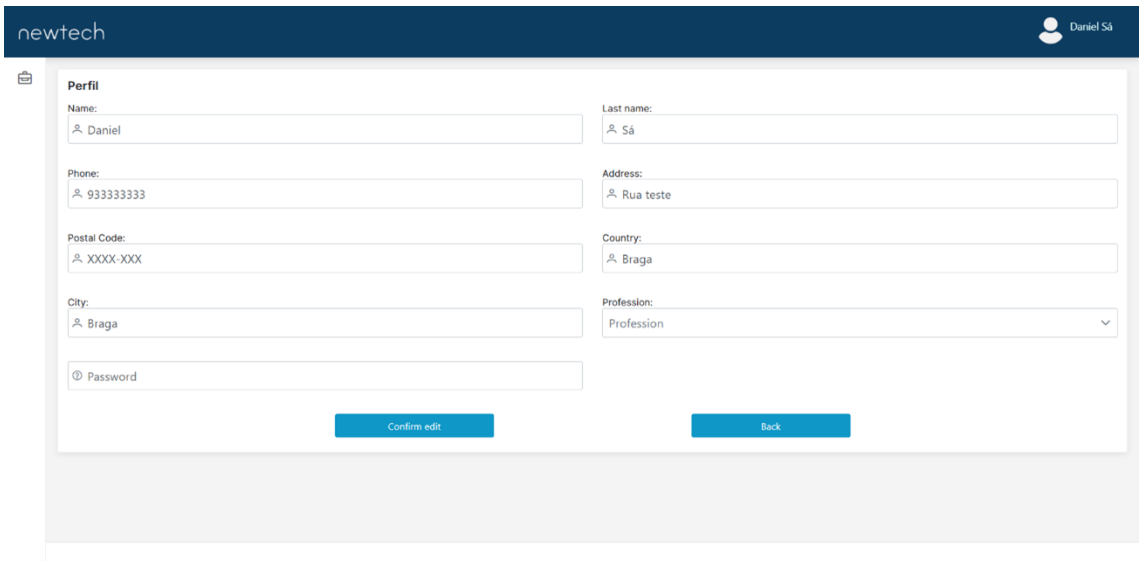


Figura 41 - Exemplo de painel de edição dos dados de utilizador

Por fim, é possível alterar a palavra-passe do utilizador utilizando o botão “Mudar Password”, tal como se pode observar na Figura 42.

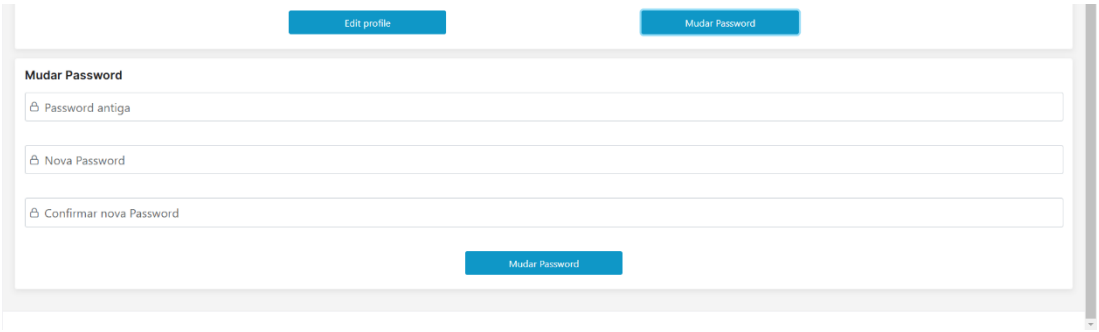


Figura 42 - Exemplo de edição apenas de password

5.3.13 Lista de ficheiros existentes e gestão dos mesmos

Na página inicial da *dashboard*, existe a opção de listar todos os ficheiros inseridos nessa empresa, ou seja, sempre que o utilizador faz upload de um ficheiro esse ficheiro passa a estar disponível nesse modal. Para esse modal ser exibido é necessário aceder ao botão “Lista de Ficheiros” assinalado na Figura 43.

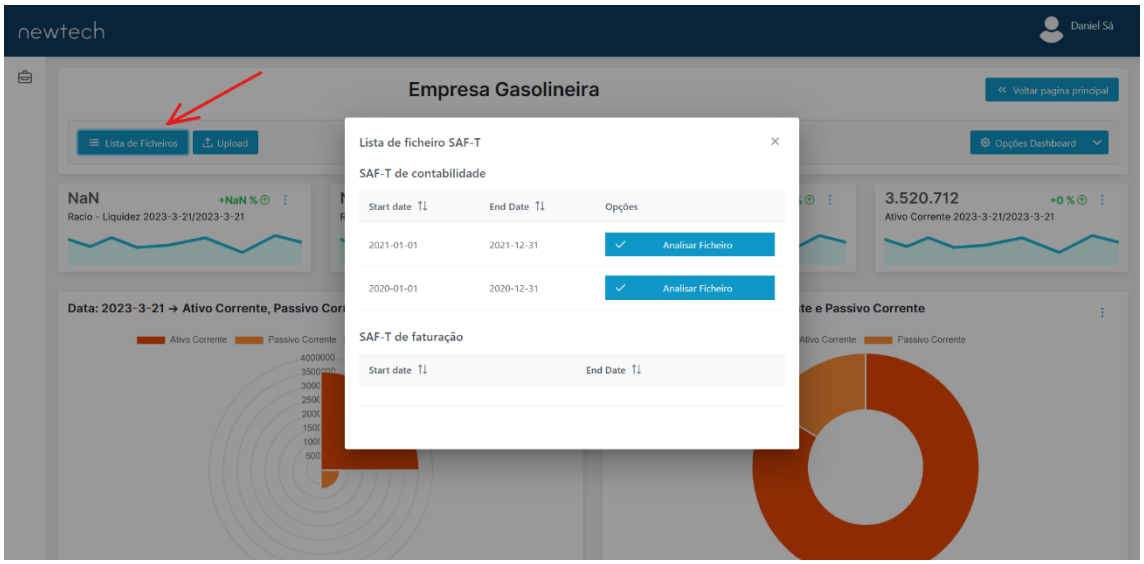


Figura 43 - Exemplo de modal onde é possível ver os ficheiros inseridos

Neste modal é também possível obter uma visualização desse ficheiro através do botão “Analisar Ficheiro”, onde abre uma nova janela com os valores referentes a esse ficheiro. Neste caso, os ficheiros são os SAFT da empresa carregados pelo utilizador.

5.3.14 Opção de upload de ficheiros

Na página inicial da *dashboard*, é possível fazer o upload dos ficheiros que serão utilizados posteriormente para a criação dos gráficos e dos *widgets*. Tal como demonstrado na Figura 44, é necessário aceder ao botão "Upload", assinalado a vermelho, e posteriormente selecionar o ficheiro presente no computador onde a plataforma está a ser utilizada e carregar o mesmo para a plataforma. Após este processo será possível utilizar os dados desse ficheiro nos gráficos e nos *widgets*.

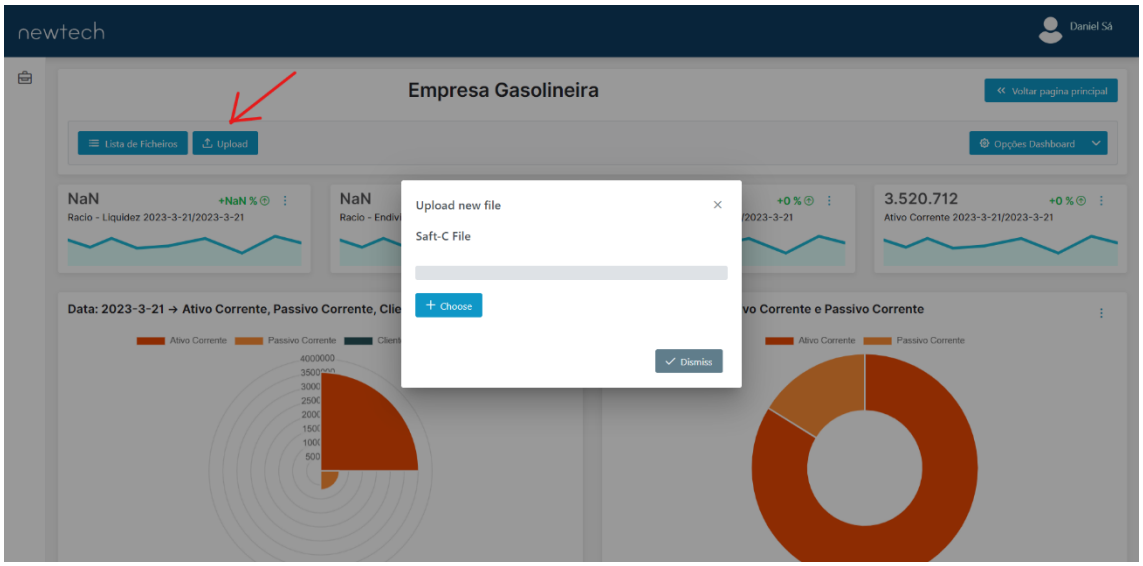


Figura 44 - Modal de upload de ficheiros para a plataforma

5.4 Alterações necessárias para aplicação de plataforma com outros dados ou em outras áreas

Na vertente Low-Code desta plataforma teve-se em conta dois aspetos importantes e diferenciadores, aplicar a vertente Low-Code na perspetiva do utilizador e também na perspetiva do programador. Isto é, para além da facilidade em configurar e personalizar a *dashboard* ao gosto do utilizador, demonstrado no ponto anterior deste capítulo, também existem algumas alterações a serem feitas no caso desta plataforma ser aplicada noutra área ou com outros dados.

Deste modo, e para ser possível utilizar a plataforma em outras aplicações, é necessário, numa primeira fase, alterar a forma como os dados que pretende mostrar

são guardados na base de dados. Após ter os dados guardados e estruturados o programador tem de alterar os seguintes ficheiros.

1. balanco.controller.js

Ficheiro responsável por obter todos os dados referentes a uma empresa de modo a mostrar estes nos menus de criação de gráficos e *widgets* e também responsável por auxiliar outras funções na comparação e obtenção dos dados. Requer que o programador altere o ficheiro de modo que as funções entreguem os dados no formato atual quando a aplicação é chamada.

2. layout.controller.js

Ficheiro responsável pela gestão do layout completo, ou seja, neste ficheiro estão todas a funções relativas à gestão dos gráficos e widgets. Aqui, tal como no ficheiro anterior, é necessário que o programador edite as funções consoante os seus dados, de modo a manter a integridade da resposta de cada uma destas funções, no entanto com base nos seus dados.

Exemplos de resposta dados por esta função, onde é através desta função que surge um gráfico do tipo “LinearChart”. Está representado o “_id” deste Layout na base de dados. O “username” a que pertence este Layout. O “id_company” a que se refere o mesmo. O “tipoGrafico" a que se refere, neste caso “LinearChart”. Os anos que são pretendidos mostrar nesse gráfico “anosPretendidos”. O “tipoDadosF” que representa o tipo de dados a serem mostrados, neste caso, indica que se quer mostrar o Ativo Corrente e o Passivo Corrente. Os “dadosPorAno” é o Array resultante do cruzamento de dados pretendidos com os valores da base de dados, ou seja, indica para cada um dos anos e do tipo de dados o respetivo valor. A designação, “designacao" que



Figura 45 - Exemplo de resposta da função GET do ficheiro "layout.controller.js"

pode assumir duas opções, “default” é gerada posteriormente com base no tipo de dados e nos anos pretendidos, ou, caso o utilizador pretenda uma designação personalizada este campo assumirá o valor dessa designação. Por fim, o “tipo” pode assumir dois valores, “tipo1” e “tipo2”, em que “tipo1” representa os gráficos, e “tipo2” representa os widgets”, em que no caso dos widgets a resposta a esta função é diferente, tal como demonstrado na Figura 46.


```
{
    "_id": "644be4b4ad63402397226b2e",
    "username": "joao",
    "id_company": "852963741",
    "anosPretendidosF": [],
    "tipoDadosF": [],
    "percentagem": "0",
    "tipoDados": "totalAtivoCorrente",
    "ano1": "852963741_joao_1679408930179",
    "ano2": "852963741_joao_1679408988627",
    "valorAno2": "3.520.712",
    "designacao": "default",
    "tipo": "tipo2",
    "__v": 0
},
```


Figura 46 - Exemplo de resposta da função GET do ficheiro “layout.controller.js”

A resposta quando o “tipo” é “tipo2”, ou seja, *widget,* difere do “tipo1” nos campos “anosPretendidosF” e “tiposDadosF” que ficam Array vazio, no “ano1” e “ano2” que representam o valor de dois anos diferentes de modo a comprar um possível aumento ou redução, e o “valorAno2” que representa o valor que surge no *widget.*

3. Ficheiros balanco.js

É necessário também alterar o ficheiro “balanco.js” da pasta “models” de modo a este ficar de acordo com os novos dados.

4. Ficheiro Company.vue

Este ficheiro é responsável, entre outras funções, por demonstrar os campos referentes a cada uma das empresas e visualizados na página inicial após o login, tal como demonstrado na Figura 9 e também na Figura 47.

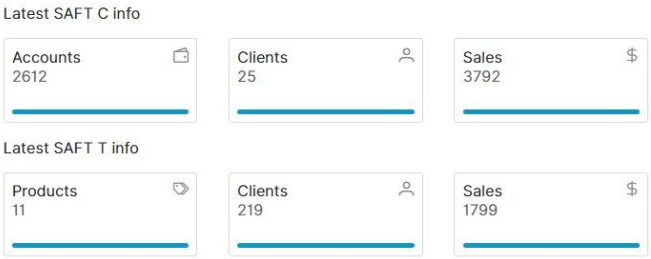


Figura 47 - Exemplo de dados mostrados na página inicial

Estes dados estão armazenados na base de dados juntamente com os dados da empresa, sendo que são gerados aquando do upload do ficheiro SAFT. De modo a alterar esta apresentação, é necessário ajustar neste ficheiro os nomes e parâmetros a receber.

É também necessário alterar o ficheiro de upload dos dados de modo a lidar com os novos dados utilizados. Assim, e após alterar as funções que permitem o upload dos ficheiros, é necessário ter em especial atenção estas variáveis, de modo a aparecerem na página inicial de forma correta.

Após as alterações referidas acima e alguns ajustes necessários por parte dos programadores, a aplicação ficará pronta para ser utilizada da mesma forma que demonstrado no capítulo anterior, mas com outra fonte de dados, podendo ser aplicada noutras vertentes e áreas, de modo a ser possível extrair informação de dados através de gráficos e *widgets* dinâmicos e personalizados pelo utilizador.

6. CONCLUSÕES

Esta capítulo tem como principal objetivo apresentar uma análise crítica e algumas limitações encontradas ao longo do desenvolvimento deste projeto e também uma conclusão sobre o projeto desenvolvido.

6.1 Análise Crítica e Limitações

Nesta análise crítica e limitações teve-se em conta os objetivos previamente estipulados e a forma como foram atingidos e, nos casos em que se aplica, apresenta-se uma referência a determinadas dificuldades encontradas até se atingir esse mesmo objetivo. Deste modo, a Tabela 5 conta com os Objetivos, previamente estipuladas no capítulo 1.2.1 desta dissertação e o respetivo resultado obtido.

Tabela 5 - Análise Crítica e Limitações

Objetivo	**Resultado Obtido**
Análise das abordagens existentes ao conceito de *Low-Code*, ou seja, será elaborada uma pesquisa bibliográfica e um estado da arte sobre paradigmas *Low-Code*.	Concluído com sucesso, no entanto, saliento alguma dificuldade em encontrar as informações necessárias em artigos, teses, livros, entre outros.
Caraterização de ambientes típicos de aplicação.	Concluído com sucesso, sendo que no caso concreto de aplicação deste projeto não se encontraram casos de uso idênticos ou relacionados.
Definição de características para avaliação.	Concluído com sucesso. Sendo que temos por base as plataformas referidas anteriormente.
Pretende-se, se for possível, efetuar um benchmarking com base num conjunto de características definidas anteriormente.	Não foi possível realizar um benchmark devido às exigências de ambientes de teste e as respetivas licenças de uso.

Selecionar o paradigma mais eficaz para desenvolvimentos Web.	Concluído com sucesso. Foi selecionado o paradigma mais eficaz com base na pesquisa efetuada e também com ajuda do orientador.
Com base no paradigma selecionado, desenvolver e implementar artefactos que recorram a abordagens *Low-Code* e que garantam resposta de forma genérica a diferentes soluções e projetos.	Concluído com sucesso. Sendo que, o facto de tornar a plataforma facilmente aplicável noutras valências, demonstrou-se mais moroso e complicado do que o previamente pensado.

6.2 Conclusão

Finalizado o projeto da dissertação podemos concluir que a realização do mesmo foi bem-sucedida na medida em que se atingiu todos os objetivos, exceto um, tal como demonstrado no ponto anterior e devido a exigências de ambientes de teste e às respetivas licenças de uso. Tal como indicado nos resultados esperados a Aplicação de uma abordagem *Low-Code* efetuando a prova de conceito em diferentes casos de uso na área de desenvolvimento Web, nomeadamente nas áreas de *Fintech*, resultou na plataforma descrita e demonstrada no capítulo do desenvolvimento do projeto e resultados obtidos. Essa aplicação permite aos utilizadores, neste caso em concreto, na área de *Fintech*, configurar e personalizar uma plataforma que permite tirar partido dos dados presentes nos ficheiros SAFT através de gráficos e *widgets*, sendo possível selecionar os dados a ver para cada empresa de um determinado utilizador.

É de salientar, e visto que também estava incluído no resultado esperado, a capacidade de adaptar esta plataforma a outras áreas e valências através do mínimo possível de código (Low-Code), tal como demonstrado no capítulo anterior, onde são apresentadas as alterações necessárias a realizar na plataforma para isso ser possível.

No decorrer desta dissertação foi também realizado um artigo onde o tema abordado foi “Low Code Approach for Business Analytics”.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] J. Peffers, «The Design Science Research Process : A Model for Producing and Presenting Information Systems Research», pp. 83–106, 2006, Acedido: 20 de Junho de 2023. [Em linha]. Disponível em: http://rightsstatements.org/page/InC/1.0/?language=en

[2] R. Sanchis, Ó. García-Perales, F. Fraile, e R. Poler, «Low-Code as Enabler of Digital Transformation in Manufacturing Industry», *Applied Sciences 2020, Vol. 10, Page 12*, vol. 10, n. 1, p. 12, Dez. 2019, doi: 10.3390/APP10010012.

[3] R. Waszkowski, «Low-code platform for automating business processes in manufacturing», *IFAC-PapersOnLine*, vol. 52, n. 10, pp. 376–381, Jan. 2019, doi: 10.1016/J.IFACOL.2019.10.060.

[4] «IEEE Xplore Full-Text PDF»: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9655393&tag=1 (acedido 31 de Março de 2023).

[5] V. Dubey e R. S. Walimbe, «Fintech 2022 Trends : The era of F-CUBE "Fast and Furious Fintech"», *Technium: Romanian Journal of Applied Sciences and Technology*, vol. 3, n. 9, pp. 44–50, Nov. 2021, doi: 10.47577/TECHNIUM.V3I9.4976.

[6] M. De e M. Pereira, «Arquitetura baseada em microserviço Arquitetura baseada em microserviço View project», Acedido: 13 de Junho de 2023. [Em linha]. Disponível em: https://www.researchgate.net/publication/329521832

[7] S. Tilkov e S. Vinoski, «Node.js: Using JavaScript to build high-performance network programs», *IEEE Internet Comput*, vol. 14, n. 6, pp. 80–83, Nov. 2010, doi: 10.1109/MIC.2010.145.

[8] W. Xu, «Benchmark Comparison of JavaScript Frameworks React, Vue, Angular and Svelte», Acedido: 15 de Junho de 2023. [Em linha]. Disponível em: http://www.tcd.ie/calendar

[9] «Writing Markup with JSX – React». https://react.dev/learn/writing-markup-with-jsx (acedido 15 de Junho de 2023).

[10] «Virtual DOM and Internals – React». https://legacy.reactjs.org/docs/faq-internals.html (acedido 16 de Junho de 2023).

[11] «Learning Vue.js 2 - Olga Filipova - Google Livros». https://books.google.pt/books?hl=pt-PT&lr=&id=nszcDgAAQBAJ&oi=fnd&pg=PP1&dq=vue.js&ots=9oMhJeTdlQ&sig=nRCyZGe97LwTb0abNfi4cEhtd8E&redir_esc=y#v=onepage&q=vue.js&f=false (acedido 15 de Junho de 2023).

[12] «Vue.js: Construa aplicações incríveis - Caio Incau - Google Livros». https://books.google.pt/books?hl=pt-PT&lr=&id=Ft-8DgAAQBAJ&oi=fnd&pg=PT2&dq=vue.js&ots=_acX7_L03x&sig=VYLZJ-cj9Cuwyu-R4AvI7b783kc&redir_esc=y#v=onepage&q=vue.js&f=false (acedido 16 de Junho de 2023).

[13] «Introduction | Vue.js». https://vuejs.org/guide/introduction.html (acedido 16 de Junho de 2023).

[14] S. Goyal e D. Batra, «ANGULAR JS», *International Research Journal of Modernization in Engineering Technology and Science www.irjmets.com @International Research Journal of Modernization in Engineering*, pp. 2582–5208, 2336, Acedido: 16 de Junho de 2023. [Em linha]. Disponível em: www.irjmets.com

[15] «AngularJS: Uma abordagem prática e objetiva - Michael Henrique R. Pereira - Google Livros». https://books.google.pt/books?hl=pt-PT&lr=&id=MUelBQAAQBAJ&oi=fnd&pg=PA13&dq=angularjs&ots=qIfK-TyZCU&sig=uWNVd7pEsL8EqBcEnJUbEVk0-B0&redir_esc=y#v=onepage&q=angularjs&f=false (acedido 16 de Junho de 2023).

[16] C. Geovana, N. Macário, e S. M. Baldo, «O Modelo Relacional».

[17] B. Farias Lóscio bfl e H. Rodrigues de Oliveira, «NoSQL no desenvolvimento de aplicações Web colaborativas», Acedido: 19 de Junho de 2023. [Em linha]. Disponível em: http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html

[18] J. R. Rymer, «The 13 Providers That Matter Most And How They Stack Up Key takeaways», 2017.