UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Antonio Moreno Ribeiro Homem

# Concepção e implementação de uma solução de *Business Intelligence* para gestão financeira de clientes em uma plataforma de *e-commerce*

Florianópolis
2021

Antonio Moreno Ribeiro Homem

# Concepção e implementação de uma solução de *Business Intelligence* para gestão financeira de clientes em uma plataforma de *e-commerce*

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Ricardo José Rabelo, Dr.
Supervisor: Daniel Valenti, Eng.

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Ribeiro Homem, Antonio Moreno
Concepção e implementação de uma solução de Business Intelligence para gestão financeira de clientes em uma plataforma de e-commerce / Antonio Moreno Ribeiro Homem ; orientador, Ricardo José Rabelo, coorientador, Daniel Valenti, 2021.
80 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2021.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Business Intelligence. 3. Dados. 4. Desenvolvimento. 5. Dados. I. Rabelo, Ricardo José. II. Valenti, Daniel. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Antonio Moreno Ribeiro Homem

**Concepção e implementação de uma solução de *Business Intelligence* para gestão financeira de clientes em uma plataforma de *e-commerce***

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 27 de setembro de 2021.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

**Banca Examinadora:**

Prof. Ricardo José Rabelo, Dr.
Orientador
UFSC/CTC/DAS

Daniel Valenti, Eng.
Supervisor
BIX Tecnologia

Prof. Carlos Barros Montez, Dr.
Avaliador
UFSC/CTC/DAS

Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Presidente da Banca
UFSC/CTC/DAS

Dedico este trabalho ao meu avô Antonio (in memorian),
que sempre me incentivou a seguir este caminho.

## AGRADECIMENTOS

À minha mãe Ana Lúcia, por todo o apoio incondicional, por confiar em minhas decisões e pelas incontáveis conversas e reflexões que me guiaram, desde o início da minha trajetória, até onde estou hoje. À minha namorada Brenda, pela parceria ao longo desses anos, pelos desafios e conquistas compartilhados, pelo apoio e incentivo durante todo o período da faculdade.

Aos meus amigos da vida, que estiverem sempre ao meu lado em cada nova etapa. Aos amigos da graduação, com os quais divido tantas tantas experiências e dificuldades superadas, tenho certeza que sem eles esse processo não teria sido tão enriquecedor.

À BIX Tecnologia, por me desafiar desde o início, contribuindo para meu desenvolvimento. Ao supervisor deste trabalho, Daniel Valenti, pela oportunidade de fazer parte de uma equipe de profissionais altamente capacitados.

Agradeço ao meu orientador, professor Ricardo José Rabelo, pela atenção desde o início do projeto e pelas valiosas contribuições dadas durante todo o processo.

Por fim, à Universidade Federal de Santa Catarina, que me acolheu durante esses anos, me proporcionou conhecimento e favoreceu ao meu crescimento pessoal e profissional.

## RESUMO

O presente documento diz respeito ao desenvolvimento do Projeto de Fim de Curso (PFC) realizado entre o mês de Maio e Setembro de 2021 na BIX Tecnologia, empresa especializada em consultoria de dados. A BIX foi contratada pela Portobello, maior empresa do setor de revestimentos no Brasil, para desenvolver uma solução de *Business Intelligence* para sua subsidiária nos Estados Unidos, a Portobello America. Com a evolução da tecnologia a geração de dados por sistemas digitais é cada vez maior, produzindo diversas informações, as quais podem ser relevantes ou não para a tomada de decisão de um negócio. Nesse contexto surge o conceito de *Business Intelligence*, conjunto de técnicas e ferramentas utilizadas na transformação de dados brutos em informações significativas e úteis para a análise de negócios. O objetivo deste PFC é apresentar as etapas de concepção e desenvolvimento de uma aplicação de BI, cuja finalidade é ser integrada a uma loja virtual da Portobello America a fim de fornecer de forma estruturada e visual dados pertinentes aos consumidores da empresa. O projeto expõe as fases de construção de uma estrutura de dados na nuvem utilizando a Google Cloud Platform e a implementação do *back-end* e *front-end* da aplicação de BI, utilizando diversas linguagens de programação. Nesse processo foi aplicada uma metodologia ágil para desenvolvimento de software. Os resultados atingidos durantes os testes realizados foram satisfatórios, indicando que a solução desenvolvida cumpriu os requisitos preestabelecidos. Embora o projeto se encontre na versão piloto a avaliação inicial da Portobello foi positiva e a empresa sinalizou interesse em expandir as funcionalidades da aplicação.



**Palavras-chave**: *Business Intelligence*. Dados. Desenvolvimento. Aplicação.

## ABSTRACT

This document concerns the development of the Final Course Project (PFC) executed between May and September of 2021 at BIX Tecnologia, a data consulting company. BIX was hired by Portobello, the largest company in the coating sector in Brazil, to develop a Business Intelligence solution for its subsidiary in the United States, Portobello America. The generation of data by digital systems is increasing with the evolution of technology generating diverse information, which may or may not be relevant for a business decision-making process. In this context, the concept of Business Intelligence emerges, which is a set of techniques and tools used in transforming raw data into meaningful and useful information for business analysis. The purpose of this PFC is to present the stages of conception and development of a BI application that will be integrated into a Portobello America virtual store providing, in a structured and visual way, relevant data to the company's consumers. The project exposes the building phases of a cloud data structure using the Google Cloud Platform and the implementation of the back-end and front-end of the BI application, using several programming languages. In this process, an agile methodology for software development was applied. The results achieved during the tests performed were satisfactory, indicating that the developed solution fulfilled the pre-established requirements. Although the project is in the pilot version, Portobello´s initial assessment was positive and the company signaled interest in expanding the application's functionalities.



**Keywords**: Business Intelligence. Data. Development. Application.

## LISTA DE FIGURAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| API | *Application Programming Interfaces* |
| ASGI | *Asynchronous Server Gateway Interface* |
| AWS | Amazon Web Services |
| BI | *Business Intelligence* |
| CDN | *Content Delivery Network* |
| CPU | *Central Process Unit* |
| CSS | *Cascading Style Sheets* |
| DBMS | *Data Base Management System* |
| DOM | *Document Object Model* |
| DW | *Data Warehouse* |
| ETL | *Extract, Transform, Load* |
| GB | GigaByte |
| GCP | Google Cloud Platform |
| HTML | *HyperText Markup Language* |
| HTTP | *Hypertext Transfer Protocol* |
| I/O | *Input/Output* |
| JS | JavaScript |
| JSON | *JavaScript Object Notation* |
| JWT | *JSON Web Token* |
| KPI | *Key Performance Indicators* |
| ORM | *Object-Relational Mapping* |
| PB | Portobello |
| PBA | Portobello America |
| PDI | Pentaho Data Integration |
| SaaS | *Software as a Service* |
| SGBDR | Sistemas de Gerenciamento de Banco de Dados Relacional |
| SQL | *Structured Query Language* |
| UML | *Unified Modeling Language* |
| XML | *eXtensible Markup Language* |

**SUMÁRIO**

# 1 INTRODUÇÃO

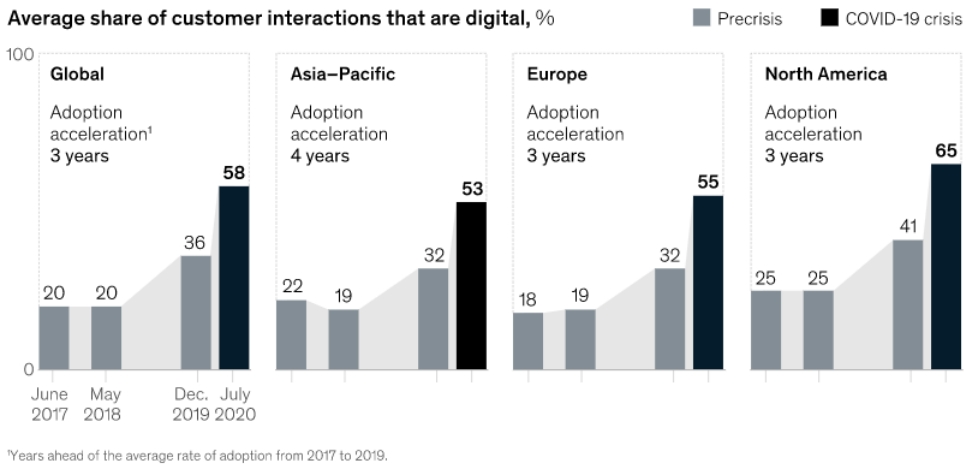
A introdução traz uma abordagem da problemática no contexto atual da tecnologia, assim como os desafios recentes no cenário, propostas de melhoria, objetivos do projeto, gerais e específicos e por fim a estrutura deste documento de PFC. O projeto foi iniciado na última semana de maio de 2021. A principal atribuição do relator do PFC foi desenvolver uma aplicação completa, *front-end* e *back-end*, e participar de diversas reuniões de modelagem, decisão de regras de negócio, arquitetura do projeto, entre outras.

## 1.1 INTRODUÇÃO À PROBLEMÁTICA

A crise causada pela pandemia da COVID-19 trouxe diversas mudanças na sociedade como um todo, entre esses efeitos um dos que mais se destacou foi o impacto na tecnologia. Ocorreu um movimento de digitalização acelerada por parte de empresas dos mais diversos setores, todas procurando um meio de combater os impactos econômicos decorrentes da situação calamitosa enfrentada.

Empresas aceleraram a digitalização da sua interação com o consumidor (Figura 1), da sua cadeia de produção e das suas operações internas em três a quatro anos, e a participação de produtos digitais ou preparados para o ambiente digital em seus portfólios acelerou sete anos (LABERGE *et al.*, 2020). A tecnologia se tornou um ponto crucial para as empresas, não sendo mais apenas uma forma de reduzir custos e aumentar a eficacia de um negocio mas tornando-se uma peça fundamental na estrategia de qualquer organização.

Figura 1 – Média de participação de interações digitais com consumidores.




Fonte – McKinsey & Company.

Esse contexto traz a tona assuntos da tecnologia que anteriormente já ganhavam destaque como Big Data, termo usado para definir dados com maior variedade que chegam em volumes crescentes e com velocidade cada vez maior. Esses conjuntos de dados são tão volumosos que um software tradicional de processamento de dados simplesmente não consegue gerenciá-los (ORACLE, 2021a). No entanto, esses grandes volumes de dados podem ser usados para resolver problemas de negócios que até então não eram possíveis de resolver.

Alinhados a essa crescente de dados estão os sistemas de tomada de decisão, conhecidos como sistemas de *Business Intelligence* (BI), que são as ferramentas utilizadas por empresas no dia a dia para entender a saúde do seu negocio, traçar estratégias e analisar diversas ações a partir das informações pertinentes trazidas por indicadores de negocio e elementos visuais como gráficos que facilitam a compreensão da informação.

Dessa forma, com a evolução do ambiente digital nas empresas é natural que a quantidade de dados gerada pelos diversos sistemas desse meio seja maior, mais complexa e de fontes mais variadas. Essa conjuntura ao mesmo tempo que promove maior entendimento dos problemas enfrentados por uma organização pela disponibilidade de informação também exige que a capacidade analítica seja maior, que os sistemas de BI sejam mais precisos e tragam somente informações pertinentes e confiáveis de forma eficiente.

## 1.2 PROBLEMÁTICA

Atualmente, a empresa Portobello America (PBA) conta com uma ampla gama de parceiros, incluindo desde empresas do setor de construção até escritórios de arquitetura e decoração de interiores, distribuídos no mercado norte americano. Essas lojas realizam a compra dos produtos da companhia através de um sistema de *e-commerce open-source*, que segue o modelo *Software as a Service* (SaaS), chamado Magento, adquirida pela multinacional americana Adobe Inc em 2018. Esta plataforma possibilita ao vendedor configurar sua loja online de acordo com suas necessidades, oferecendo também aos compradores um ambiente voltado ao consumidor.

O software permite criar uma loja virtual e oferece diversas funcionalidades, no entanto, a plataforma não apresenta um painel adequado de BI para acompanhamento de faturas e monitoramento de dados relacionados aos pedidos realizados pelos clientes da Portobello América. Assim a BIX Tecnologia, empresa onde este projeto foi desenvolvido, foi contactada pela PBA com o objetivo de resolver este problema.

Ao implementação do sistema de BI na plataforma de *e-commerce* oferece aos clientes uma visão superior de suas transações, permitindo que estes tenham maior controle do que consomem, das suas vendas e produtos mais comprados, além de apresentar de forma simplificada seu atingimento de metas em um único painel. Essa

implementação também visa melhorar a experiência do consumidor, desenvolvendo assim o relacionamento com o cliente.

Dessa maneira, a BIX se propôs entregar uma solução que atinja as expectativas do cliente e de acordo com os requisitos preestabelecidos, fornecendo um resultado acertado e escalável, dando espaço para melhorias e modificações caso necessário.

## 1.3 OBJETIVOS

O objetivo deste projeto é avaliar a viabilidade e escalabilidade da construção e implantação de uma solução de *Business Intelligence*, levando em consideração previamente o tratamento, processamento e armazenamento de um grande volume de dados.

### 1.3.1 Objetivo Geral

O presente trabalho teve por objetivo realizar a criação de uma solução de BI e sua implantação em uma loja virtual da empresa Portobello America, oferecendo aos clientes da empresa informação sobre seus pedidos e atingimento de metas. A aplicação expõe diversos elementos visuais como tabelas, gráficos e indicadores de medição de desempenho, todos estes agrupados em único painel (*dashboard*). Desta maneira, os consumidores da PBA ganham maior controle e facilidade para gerir suas transações.

### 1.3.2 Objetivos Específicos

- Criar um banco de dados na nuvem para armazenamento dos dados pertinentes dos clientes da PBA;
- Conectar o sistema de integração de dados da PBA com o banco de dados criado e alimentá-lo com as informações recebidas do sistema de integração;
- Criar uma rotina para envio e atualização das informações no banco pelo sistema de integração de dados da PBA;
- Desenvolver uma aplicação web completa. Contando com desenvolvimento *front-end* e *back-end*;
- Criar um *dashboard* seguindo o modelo predefinido pela contratante;
- Implantar a aplicação desenvolvida em um servidor virtual para acesso remoto e operação contínua.
- Conectar a aplicação com o banco de dados criado;

- Implementar um sistema de autenticação dos clientes para que os dados exibidos sejam reduzidos e filtrados de acordo com o usuário, garantido a privacidade;
- Gerar modelos de teste para validação do projeto;
- Gerar um modelo final de acordo com as considerações da PBA;
- Garantir que incorporação do painel na loja virtual na plataforma Magento funcione de forma correta.

## ESTRUTURA DO DOCUMENTO

O presente documento é dividido em 8 capítulos, onde este primeiro é responsável pela introdução ao assunto tratado e contextualização do problema a ser resolvido. O capítulo 2 apresenta as instituições onde o projeto foi realizado e seus respectivos processos.

Em seguida, o terceiro capítulo aborda os principais conceitos empregados na realização das atividades. No quarto capítulo são abordadas tecnologias e ferramentas utilizadas, como linguagens de programação, o funcionamento de bibliotecas ou *frameworks* e ferramentas empregadas na construção da solução proposta.

No capítulos 5 é apresentado o levantamento dos requisitos, casos de uso e a arquitetura do sistema. Logo após, são expostos todos os procedimentos realizados. No capítulo 7 é realizada uma análise dos resultados obtidos. Finalmente, no último capítulo são apresentadas conclusões e propostas de melhorias do presente trabalho.

# 2 AS EMPRESAS

Este capítulo trata das empresas BIX Tecnologia e Portobello, onde desenvolveu-se este projeto e onde foi implementado, respectivamente. Abordando brevemente a história de ambas empresas, seus produtos e serviços, a estrutura organizacional da contratada e por fim é apresentada a metodologia adotada para realizar a gestão de tarefas do projeto.

## 2.1 BIX TECNOLOGIA

A empresa BIX Tecnologia é uma consultoria de dados. A empresa iniciou sua atividade oferecendo soluções na área de *Business Intelligence* porém atualmente também entrega soluções no campo de *data science* e *data engineering*, focando em negócios e na conquista de resultados.

A BIX promove uma cultura de tratar clientes como parceiros, ajudando no desenvolvimento de cada negócio e criando uma relação duradoura com o consumidor. A consultoria atende todo tipo de setor, entre seus principais clientes estão as indústrias Portobello S.A. e Embraco, as varejistas CalCenter e Grupo Soma e, no ramo financeiro, o Grupo Fontes.

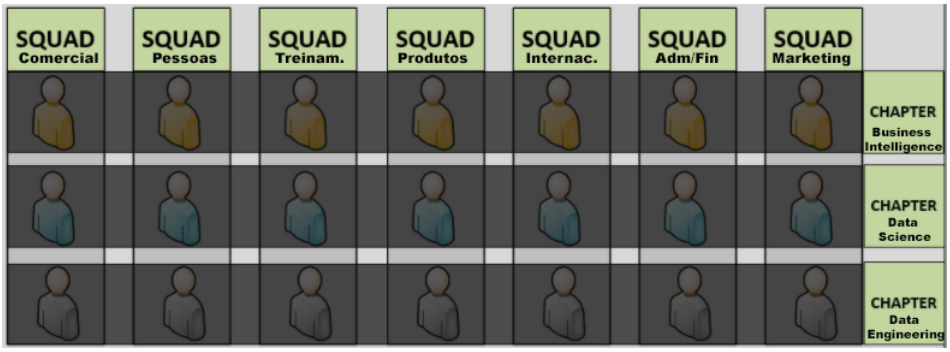
A empresa tem sede em Florianópolis, cidade onde foi fundada no ano de 2014 pelo engenheiro de controle e automação Felipe Santos Eberhardt. Na atualidade, a empresa conta com 40 especialistas nas diversas áreas e tem planos para continuar crescendo para acompanhar a forte demanda do mercado por soluções de tecnologia no âmbito de dados.

O orientador deste trabalho dentro da BIX é Daniel Valenti, sócio e gerente de BI da BIX, formado em Engenharia de Controle e Automação pela UFSC. Daniel atua na área de BI desde que ingressou na organização como estagiário em 2016 e tem experiência atendendo diversos ramos e liderando os consultores do time.

No momento atual, a empresa é dividida em *chapters*, que são times de especialistas de uma determinada área de conhecimento, como BI, *data science* ou *data engineering*. Frequentemente projetos são divididos entre as áreas e consultores de diferentes *chapters* trabalham em conjunto para entregar soluções completas.

Além dessas equipes há também os *squads*, que são times multidisciplinares, como exposto na figura 2. Esta formação pode ser temporária ou não e lida com assuntos específicos, como o *squad* de pessoas que busca implementar ações visando o desenvolvimento e bem estar dos colaboradores da organização.

Figura 2 – Estrutura organizacional da empresa BIX Tecnologia.




Fonte – BIX Tecnologia.

## 2.2 PORTOBELLO

A Portobello S.A., foi fundada em 1979 e está situada no município de Tijucas, em Santa Catarina. A empresa tem 205 mil metros quadrados de área construída, em vista disso, é considerada a maior empresa cerâmica do Brasil na atualidade e o maior parque fabril do ramo na América Latina. A organização tem receita bruta superior a R$ 1 bilhão, o que a torna bastante importante para a região. A produção anual supera os 30 milhões de metros quadrados, são atendidos mais de 65 países, abrangendo os cinco continentes assim como o mercado interno.

O projeto descrito neste documento foi realizado com o objetivo de atender o cliente Portobello America Inc., que é uma subsidiária da Portobello S.A. inaugurada no início de 1990. A PBA realiza a distribuição e vendas dos produtos Portobello (PB) para o mercado norte americano. A subsidiária está situada em Pompano Beach, no estado da Flórida. Atualmente a companhia atende diversas lojas de setores como decoração de interiores, construção e arquitetura.

A Portobello America utiliza no momento atual uma plataforma chamada Magento, a qual permite criar um *e-commerce* e personalizá-lo conforme a necessidade do usuário. Contudo, esta ferramenta carece de funcionalidades que a empresa desejaria ter, como um ambiente de BI voltado ao consumidor. A maioria dos softwares de BI da atualidade requer licenciamento por usuário, por conta disto, torna-se inviável para a PBA contratar uma destas ferramentas, pois, cada cliente implicaria em custos mensais de manutenção do acesso à aplicação.

Em vista disso, a BIX Tecnologia foi contratada com o objetivo de implementar uma solução para o problema em questão. Deste modo, com o desenvolvimento do presente projeto procura-se aprimorar a experiência dos clientes da PBA, fornecendo acesso às suas informações e dados relevantes.

## 2.3 GERENCIAMENTO DE PROJETO

Para realizar a gestão do projeto foi escolhida a metodologia Kanban, pois é um modelo ágil muito conhecido e amplamente utilizado. Dessa forma, o time responsável pelo desenvolvimento do projeto já possuía experiência no método, tornando o planejamento das atividades a serem realizadas mais rápido e simples.

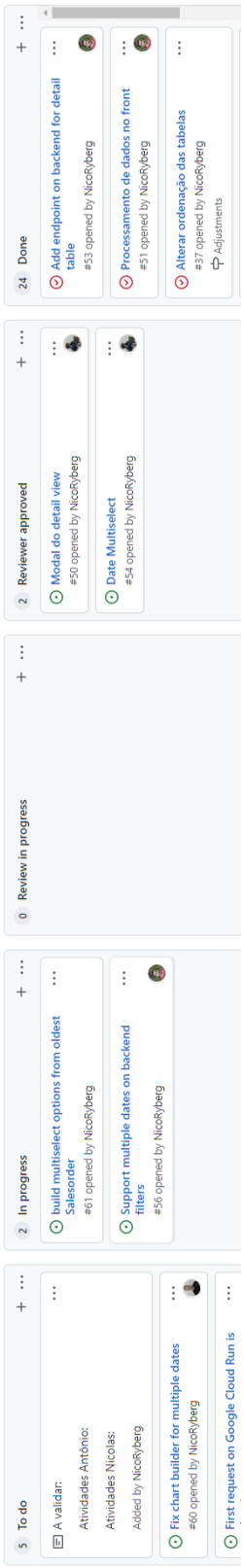
O Kanban surgiu do sistema Toyota de produção, tendo como objetivo controlar o estoque de materiais para não exceder nem faltar produtos, promovendo o equilíbrio entre o estoque e a linha de produção (ESPINHA, 2021). Essa metodologia, passou a ser utilizadas por diversas áreas fora da indústria, onde foi criada. Atualmente, é amplamente utilizada como ferramenta de gestão de tarefas por equipes de marketing, desenvolvimento de software, prestação de serviços, entre muitas outras áreas.

Há três partes importantes na composição do Kanban: cartão, colunas e quadro. O cartão é a menor dessas partes e representa uma ação ou tarefa que deve ser realizada para que o resultado final seja entregue. Colunas são utilizadas para representar o status dos cartões, geralmente um Kanban possui três colunas: a fazer, em execução e realizado. Assim, os cartões são movidos entre as colunas de acordo com seu status atual, indicando o que está pendente e o que já foi concluído. Finalmente, o quadro é a visão do todo, cada quadro representando um Kanban formado por colunas e cartões. Quadros podem ser utilizados simultaneamente por uma equipe, podendo esses ser específicos para um projeto ou gerais, englobando atividades de diferentes projetos (ESPINHA, 2021).

Para o trabalho realizado foi utilizado um Kanban de produção criado na plataforma Github, exposto na figura 3, focado na gestão de tarefas, as quais são conhecidas como *issues*. Esse modelo funciona dividindo as colunas em tarefas não realizadas, tarefas sendo realizadas, tarefas sendo revisadas, aprovadas e concluídas. O time movimenta os cartões entre as colunas à medida que atividades passam a ser executadas ou são finalizadas.

Entre as vantagens da metodologia, destacam-se a autonomia, a priorização de tarefas, o aumento da produtividade, a redução de custos e a colaboração. O Kanban possibilita centralizar todo a gestão de tarefas de uma equipe em uma única ferramenta, permitindo a gestão visual das atividades e simplificando seu uso pelos integrantes, os quais tem autonomia para verificar e movimentar os cartões à medida que vão assumindo e realizando suas atividades. Devido a essas características, a metodologia foi escolhida como opção mais adequada para o desenvolvimento do projeto.

Figura 3 – Quadro de Kanban utilizado no projeto.




Fonte: Arquivo Pessoal.

# 3 CONCEITOS

Neste capítulo é exposta a fundamentação teórica do conceito de *Bussiness Intelligence*. É apresentada uma breve explicação das áreas de desenvolvimento de software. Além disso, são descritos os conceitos de banco de dados, banco de dados relacional e ETL. Os assuntos abordados são fundamentais no entendimento das soluções expostas nos próximos capítulos.

## 3.1 *BUSINESS INTELLIGENCE*

Um dos recursos mais importantes para qualquer empresa da atualidade é a informação. Este recurso é utilizado geralmente para manter registros históricos de uma operação e para realização de análises de negócio (KIMBALL; ROSS, 2013). O conceito de *Business Intelligence* surge com o intuito de contemplar essa última atividade, representando um conjunto de técnicas e ferramentas utilizadas para fornecer através de dados e sistemas tecnológicos acesso rápido a informações relevantes para a tomada de decisão estratégica. Assim, esta definição trata do acesso, a análise e a descoberta de oportunidades por meio dos dados. A aplicação deste conceito é realizada através dos sistemas de BI, aplicações projetadas para esse fim.

Os sistemas de BI são responsáveis por exibir a informação através de painéis, conhecidos como *dashboards*, de forma visual e interativa. Um *dashboard* é composto geralmente por gráficos, tabelas e indicadores de desempenho, contemplando dados relevantes da organização. A informação utilizada pelos sistemas de BI é entregue pelos *Data Warehouse* (DW), responsáveis por armazenar todos os dados em um mesmo ambiente, possibilitando a organização destes e viabilizando as atividades realizadas posteriormente pelo BI. Os DW's também são utilizados para armazenar registros históricos de dados, permitindo trabalhar com informação de diferentes períodos.

Por fim, o acesso aos *dashboards* se dá por meio de páginas web, nas quais o usuário pode interagir com o painel e selecionar as informações desejadas. Assim, é possível restringir o tipo de acesso de cada usuário, definindo quais informações cada usuário consegue acessar, garantindo deste modo a privacidade dos dados exibidos pelo sistema de BI.

### 3.1.1 Evolução dos sistemas de informação de negócios

Apesar do conceito de BI ser amplamente difundido na atualidade, este é antigo e evoluiu com o tempo acompanhando a evolução da tecnologia. O livro "Getting Started with DataWarehouse and Business Intelligence" da empresa IBM, divide os sistemas de informação de negócios em três gerações: *Host-Based Query and Reporting*, *Data Warehousing* e *Business Intelligence*.

#### 3.1.1.1 Primeira geração: *Host-Based Query and Reporting*

Os primeiros sistemas de informação de negócios utilizavam aplicações para fornecer informação aos usuários de negócio. Geralmente, os resultados dessas aplicações continham muitas informações, representadas em um grande volume de folhas de papel. Desse modo, os usuários precisavam buscar as informações que realmente necessitavam.

Com a evolução para os primeiros sistemas computacionais, os usuários passaram a depender de um alto conhecimento da tecnologia utilizada. Assim, geralmente, apenas analistas de dados utilizavam esses programas. Dessa forma, gestores e executivos que necessitavam receber as informações obtidas dependiam dos profissionais encarregados desses sistemas (ALMEIDA *et al.*, 1999).

#### 3.1.1.2 Segunda geração: *Data Warehousing*

A segunda geração contou com os sistemas de *Data Warehousing*, estes possuem algumas vantagens em relação à primeira geração. Esses sistemas são organizados de forma mais clara, facilitando o entendimento dos usuários de negócios. Além disso, armazenam dados históricos e informação sintetizada. Alguns desses sistemas também disponibilizam ferramentas que auxiliam na tomada de decisão, exibindo interfaces voltadas ao entendimento dos usuários. Contudo, tinham como foco principal o desenvolvimento de tecnologia (ALMEIDA *et al.*, 1999).

#### 3.1.1.3 Terceira geração: *Business Intelligence*

Os sistemas de *Business Intelligence* surgiram para atender melhor a necessidade dos usuários de negócio, uma vez que os sistemas de *Data Warehousing* são mais focados na tecnologia do que nos usuários. Assim, o foco dos sistemas de BI é oferecer ferramentas que auxiliem os usuários de negócios em suas tomadas de decisão. A segunda geração teve como objetivo principal o armazenamento dos dados, já a terceira tem como foco facilitar o acesso aos dados (ALMEIDA *et al.*, 1999).

Para isso, investiu-se em desenvolvimento de ferramentas gráficas avançadas e também sistemas de processamento analítico online.

Um dos principais objetivos dos sistemas de BI é facilitar o acesso aos dados, possibilitando dessa forma que os usuários respondam as principais perguntas de negócios que surgem no seu dia a dia.

### 3.1.2 Objetivos dos Sistemas de BI

Sistema de BI podem ser descritos como sistemas de apoio à decisão, porém, para que estes sejam considerados adequados devem atender os seguintes requisitos:

- O sistema de BI deve simplificar o acesso as informações. Os conteúdos do sistema devem ser de fácil entendimento. Os dados devem ser intuitivos e claros para os usuários de negócios. As ferramentas de BI devem ser simples e fáceis de utilizar. Além disso, devem trazer as informações solicitadas pelos usuários no menor tempo possível. Assim, um sistema de BI deve ser simples e rápido;
- O sistema de BI deve apresentar informação de forma consistente. Os dados no sistema devem ser plausíveis, cuidadosamente agregados de diferentes fontes, limpos, ter qualidade assegurada e somente serem entregues aos usuários quando estiverem prontos para o consumo;
- Sistemas de BI devem ser adaptáveis. Necessidades do usuário, condições de negócio, dados e tecnologia estão sujeitos a mudanças. Assim, sistemas de BI devem ser desenvolvidos considerando essa possibilidade. Dessa forma, mudanças não devem invalidar dados existentes ou aplicações;
- O sistema de BI deve apresentar as informações no tempo correto. Considerando que sistemas de BI são utilizados de forma intensiva para decisões operacionais, os dados gerados devem ser convertidos e disponibilizados para o usuário no menor tempo viável;
- Sistemas de BI devem garantir a proteção das informações disponíveis. Dados importantes da organização são guardadas nos DW's. Assim, essas informações confidencias devem ter seu acesso controlado, visando proteger a companhia de intenções maléficas;
- Sistemas de BI devem deve servir como uma base autorizada e confiável para melhorar a tomada de decisão. O DW deve conter as informações necessárias para apoiar este processo. Os resultados mais importantes de um sistema BI são as decisões tomadas com base na evidência analítica apresentada, essas decisões geram impacto nos negócios e atribuem valor ao sistema;
- A comunidade de usuários de negócio deve aceitar o sistema de BI para considerá-lo bem sucedido. Se a comunidade não aceitar e utilizar a solução proposta, o sistema falhou o teste de aceitação. Diferente de ferramentas operacionais, a utilização dos sistemas de BI nem sempre é obrigatória. Usuários de negócio usarão o sistema caso ele seja a fonte mais simples e rápida de informação acionável (KIMBALL; ROSS, 2013);

## 3.2 INDICADORES DE DESEMPENHO

A grande disponibilidade de dados atual e a facilidade de aquisição simplificou o trabalho que organizações tinham anteriormente para realizar a medição e avaliação

de desempenho de processos. No entanto, é necessário identificar de forma eficiente aquilo que se deseja avaliar, buscando tornar a seleção dos dados utilizados para análise mais compreensível. Uma vez definido o processo a ser avaliado, podem ser determinados indicadores para mensuração do desempenho, possibilitando entender de forma objetiva os resultados obtidos.

Dessa forma, indicadores de desempenho são utilizados para medir algo dentro de um certo domínio ou contexto, para um dado usuário, com um certo objetivo relacionado ao processo. Assim, estes são construídos aplicando-se regras de cálculo e unidades de medida para viabilizar a tomada de decisão ao final de análises de causas de problemas e/ou de melhorias, quando possível utilizando métricas de referência (RABELO, 2019). Deste modo, a partir do uso de indicadores de desempenho, organizações conseguem agilizar decisões usando informações confiáveis, verificar o atingimento de resultados e identificar pontos de melhoria nas atividades realizadas, visando atingir os objetivos definidos (GOMES, 2016). É importante ressaltar que indicadores e métricas são diferentes, o primeiro é utilizado como elemento de análise com seu valor observado, enquanto o segundo engloba outro dado que atua como base comparativa para o valor observado.

Ainda, indicadores de desempenho podem ser categorizados como estratégicos ou operacionais, sendo estes qualitativos ou quantitativos. Os indicadores estratégicos, também conhecidos como *Key Performance Indicators* (KPI), são voltados à camada gerencial e seu cálculo é realizado a partir de uma composição de indicadores operacionais. Já os da segunda categoria são voltados à camada de supervisão, ou operacional, tendo cálculo realizado geralmente a partir de uma composição de dados de produção (RABELO, 2019).

## 3.3 ÁREAS DE DESENVOLVIMENTO DE SOFTWARE

O processo de desenvolvimento de software pode ser dividido em diversas etapas e camadas de acordo com o escopo do projeto, simplificando dessa forma o trabalho de desenvolvimento. Para o projeto tratado neste documento segmentou-se esse processo em duas camadas, *front-end* e *back-end*. Esta abordagem, muito utilizada na área de desenvolvimento web, busca simplificar o processo de construção de aplicações, uma vez que caracteriza as tecnologias voltadas a interfaces e serviços. Embora ambas partes sejam fundamentais para a criação de aplicações, elas podem ser desacopladas e desenvolvidas de forma separada, permitindo que profissionais se especializem em uma das áreas, ou até mesmo em ambas. Com essa definição de responsabilidades, o processo de desenvolvimento é otimizado.

*Front-end* é o termo utilizado para caracterizar a camada de apresentação de um *software*, ou seja, sua interface. Esta é responsável pelos componentes da aplicação que lidam com a interação direta do usuário.

Já o *back-end* é responsável por entregar a infraestrutura da aplicação. Além disso, deve fornecer serviços para que o *front-end* apresente e colete os dados utilizados pela aplicação. Entre esses serviços estão a implementação de lógicas de negócio do sistema, envio, armazenamento e tratamento de dados, etc.

## 3.4 BANCO DE DADOS

Podemos considerar que "Um banco de dados é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador" (ORACLE, 2021c). Bancos de dados são controlados geralmente por um *Data Base Management System* (DBMS), ou sistemas de gerenciamento de banco de dados. O conjunto formado pelos dados e o DBMS, juntamente com os aplicativos associados a eles, é conhecido como sistema de banco de dados, habitualmente chamado apenas de banco de dados (ORACLE, 2021c).

Atualmente, a maioria dos bancos de dados em operação organizam dados através de linhas e colunas em uma coleção de tabelas, otimizando assim o processamento e a consulta. A informação pode ser facilmente acessada, gerenciada, modificada, atualizada, controlada e organizada. A linguagem de consulta estruturada (*Structured Query Language* (SQL)) é aceita pela grande maioria dos bancos de dados atuais para escrever e consultar.

Existem diversos tipos de bancos. A escolha do tipo depende de como a organização pretende usar os dados e quais são seus objetivos. Alguns dos tipos disponíveis são (ORACLE, 2021c):

- Bancos de dados relacionais: Este é o tipo mais empregado. Os itens deste banco são organizados como um conjunto de tabelas com colunas e linhas. A tecnologia de banco de dados relacional entrega a forma mais flexível e eficiente para acessar informações estruturadas. Por essas características foi a opção escolhida para este projeto;

- Bancos de dados orientados a objetos: As informações deste tipo de banco são estruturadas na forma de objetos, como acontece na programação orientada a objetos;

- Bancos de dados distribuídos: Um bancos de dados distribuído consiste em dois ou mais arquivos localizados em sites diferentes, podendo ser armazenado em vários computadores, estando no mesmo local físico ou separados em diferentes redes;

- Bancos de dados NoSQL: Conhecido também como banco de dados não relacional, permite o armazenamento e manipulação de dados não estruturados e

semiestruturados. Bancos NoSQL tornaram-se populares com a evolução dos aplicativos web, cada vez mais comuns e complexos;

### 3.4.1 Bancos de dados relacionais

Bancos de dados relacionais são aqueles que armazenam e proporcionam acesso a registros de dados relacionados entre si. Eles são baseados no modelo relacional, uma forma de representar dados em tabelas. Neste tipo de banco, cada linha de uma tabela é um registro que possui uma chave de identificação (ID) exclusiva. As colunas da tabela englobam os atributos dos dados, onde cada registro geralmente tem um valor para cada atributo, facilitando o estabelecimento das relações entre os registros (ORACLE, 2021b).

Existem diversos softwares para armazenar, gerenciar, consultar e recuperar dados armazenados em um banco de dados relacional, estes são conhecidos como Sistemas de Gerenciamento de Banco de Dados Relacional (SGBDR). Estes programas oferecem uma interface entre usuários, aplicativos e o banco de dados, além de incluir funções administrativas para gerenciar armazenamento, acesso e desempenho de dados (ORACLE, 2021b).

Para este projeto foi escolhido um dos cinco SGBDRS mais populares do mercado, o PostgreSQL. Ele é um software gratuito, de código aberto, desenvolvido na década de 80 a partir de um projeto iniciado na Universidade de Berkeley, Califórnia. Por ter mais de 30 anos de atividade a ferramenta dispõe de uma ampla documentação.

## 3.5 ETL

*Extract, Transform, Load* (ETL) representa o processo realizado por sistemas voltados à integração de dados, estes tem como propósito extrair informação de diferentes fontes, transformá-la em formatos adequados segunda a necessidade da empresa e carregá-la em ambientes próprios para armazenamento, como DW's (SILVA, 2021). O processo de ETL é implementado geralmente mediante o uso de softwares desenvolvidos com esta finalidade. Existem diversas ferramentas de ETL disponíveis no mercado, empresas como IBM, Oracle, Microsoft oferecem soluções neste âmbito, contudo, também existem opções *open source* como o PDI, utilizado no desenvolvimento do projeto tratado neste documento.

O processo de ETL é fundamental nas práticas implementadas pelo BI, visto que para realizar análises de dados de negócio é necessário dispor de informação no formato adequado. Dessa forma, é necessário diagnosticar e compreender o problema tratado para identificar qual a transformação de dados mais indicada, utilizando o sistema de ETL para gerar apenas informações pertinentes e completas.

As etapas que compõe o processo de ETL são:

- *Extract*: durante este processo é realizada a extração dos dados de diferentes sistemas ou fontes. Esses dados são encaminhados para outros sistemas, como DW's, onde podem ser trabalhados de forma independente;
- *Transform*: o processo de transformação é composto por várias fases: padronização, limpeza, qualidade. Dados vindos de sistemas diferentes podem apresentar padrões distintos como nomenclatura ou tipo de dado, assim, necessitam ser ajustados de acordo com a necessidade. A operação de qualidade de dados tem o intuito de garantir a confiabilidade da informação que será disponibilizada para análises;
- *Load*: o processo de carregamento é a etapa final. Durante esta etapa os dados são lidos da área de preparação, organizados e carregados de forma consolidada em sistemas como DW's ou *Data Marts* para que possam ser disponibilizados aos usuários finais (O. . . , 2020);

A utilização de ETL's oferece diversas vantagens, entre elas:

- Alta performance através de métodos desenvolvidos para trabalhar com grandes volumes, extraindo, transformando e carregando dados com maior velocidade e menos recursos, como gravações em bloco e operações não logadas;
- Garantia da qualidade dos dados, pois, através do sequenciamento de operações, ferramentas de ETL conseguem solucionar problemas de maior complexidade (O. . . , 2020).
- Quando utilizado com um *data warehouse* corporativo, o ETL fornece o contexto histórico completo à empresa;
- Ao fornecer uma visão consolidada, o ETL facilita o processo de análise e a criação de relatórios sobre dados relevantes às iniciativas dos usuários de negócio;
- ETL's permite melhorar a produtividade de profissionais da área de análise de negócio, uma vez que codifica e reutiliza processos realizados sobre os dados sem exigir dos usuários a capacidade técnica para escrever códigos ou *scripts*;
- Sistemas de ETL evoluíram ao longo do tempo para suportar requisitos emergentes de integração como a funcionalidade de *streaming* de dados (ETL. . . , 2021);

Dessa forma, ETL define o processo executado por sistemas que promovem a consolidação de dados que podem servir como base em tomadas de decisões. Além disso, também contribuem na criação e elaboração de relatórios importantes para planejamentos estratégicos de um negócio, uma vez que trazem de forma consolidada uma visão completa daquilo que foi gerado (SILVA, 2021).

# 4 TECNOLOGIAS

Neste capítulo são revisadas conceitualmente as tecnologias e ferramentas utilizadas na elaboração projeto. As tecnologias descritas compreendem principalmente as *stacks* de *front-end* e *back-end* da aplicação desenvolvida.

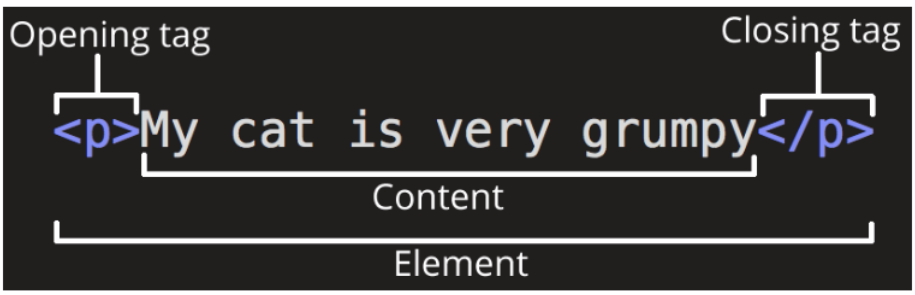
## 4.1 HTML

Linguagem de Marcação de Hipertexto, do inglês *Hypertext Markup Language* (HTML), é um dos componentes fundamentais na construção de uma página web. É importante destacar que o HTML não é uma linguagem de programação mas sim de marcação pois está não possibilita a criação de funções dinâmicas (CURTIS, 2021), contudo está é responsável por indicar ao navegador como deve estruturar a página web visitada.

Os elementos do HTML são conhecidos como *tags* e atributos, uma página é constituída por um conjunto de ambos. Uma *tag* indica ao navegador onde inicia e termina um dado elemento, enquanto o atributo descreve as características do mesmo (CURTIS, 2021).

Esses componentes da linguagem podem ser usados para anexar, envolver ou marcar diferentes partes do conteúdo para que este apareça ou aja de uma certa maneira. O fechamento das *tags* oferece diversas possibilidades como transformar uma parte do conteúdo dentro do elemento em um link que direcione a outra página web, colocar as palavras em itálico ou negrito, e assim por diante (INICIANDO. . . , 2021). Por isso, existem diversos tipos de *tags*, cada uma com a função de criar um determinado elemento como parágrafos, listas, tabelas, formulários, imagens, entre outros. A figura 4 exibe a criação de um parágrafo em HTML.

Figura 4 – Exemplo de estrutura de um elemento HTML.



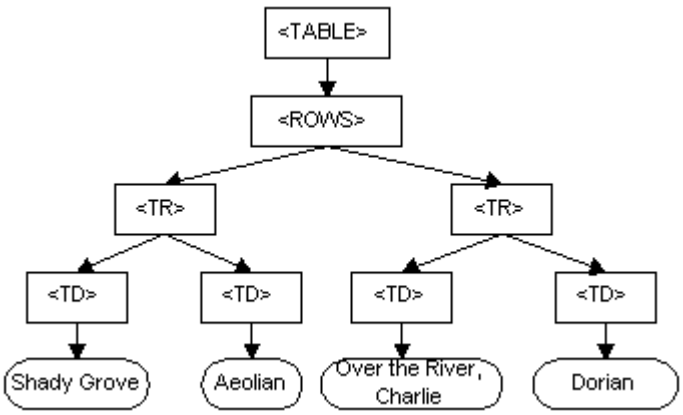

Fonte: Página MDN Web Docs.

### 4.1.1 *Document Object Model*

O *Document Object Model* (DOM) é uma interface que representa como o documento HTML é lido pelo navegador. Uma vez realizada a leitura, ele cria um objeto que representa de forma estrutura, como modelo de árvore, o documento e define meios para acessar essa estrutura (MALDONADO, 2018). O DOM pode então ser acessado e manipulado utilizando JavaScript, que é a forma mais simples e comum.

A figura 5 exibe uma representação da estrutura de elementos de um documento HTML após a leitura do DOM. A estrutura tem como raiz a tabela, *tag <table>*, a qual se divide em linhas na *tag <tr>*, elementos de dados ou células na *tag <td>* e por fim no texto contido neste último componente.

Figura 5 – Exemplo de estrutura HTML no DOM.




Fonte: Página w3

## 4.2 CSS

CSS, sigla para *Cascading Style Sheets* ou Folhas de Estilo em Cascatas, é a linguagem de estilo utilizada para descrever a apresentação de documentos escritos em linguagens de marcação como HTML. O CSS separa o conteúdo da representação visual do site (RÜCKERT, 2021). Através do seu uso é possível modificar a cor de um componente, sua fonte, seu tamanho, sua disposição na página, etc.

A figura 6 mostra um exemplo de website utilizando CSS ao lado da mesma página sem nenhuma estilização, apenas HTML. Apesar do CSS não ser obrigatório na construção de uma página seu uso torna-a mais interessante ao olhar do visitante, podendo muitas vezes torná-la também mais acessível devido ao estilo de organização do layout.

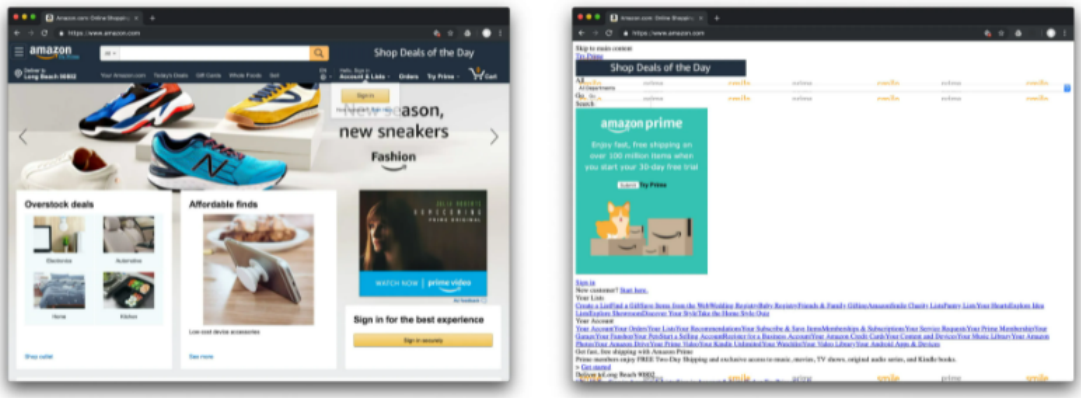
Existem múltiplas formas de alterar os estilos de um elemento via CSS. A primeira é através de estilo interno, desta maneira toda a estilização é realizada no interior

de uma *tag <style>* e carregada cada vez que um site é atualizado, podendo aumentar o tempo de carregamento e não permitindo usar o mesmo estilo CSS em várias páginas.

Há também o método externo. Neste tudo é realizado externamente em um arquivo de terminação ".css" que é incorporado dentro do arquivo HTML, dessa maneira todo o estilo pode ser escrito em um arquivo separado e aplicado a qualquer página desejada, permitindo otimizar o tempo de carregamento.

O último método é o *inline*, este lida com elementos que possuem o atributo *style* nas suas *tags*, portanto, cada componente deve ser estilizado individualmente, por isso não é considerada a melhor ou mais rápida opção para lidar com a estilização. Porém, pode ser útil para alterar um único elemento (RÜCKERT, 2021).

Figura 6 – Exemplo de página web com e sem CSS.




Fonte: Página css-tricks.

### 4.2.1 Bootstrap

O Bootstrap é um *framework* CSS que pode ser empregado na construção do *front-end* de aplicações web. Ele utiliza JavaScript e CSS para estilizar páginas e implementar funcionalidades como menus de navegação, controles de paginação, formulários, janelas modais, etc. (HORN, 2021a).

Entre as principais características do Bootstrap está a responsividade, ou seja, ele possibilita que os elementos da página sejam readaptados para o acesso em telas de diferentes tamanhos, como notebooks, tablets, smartphones.

Para poder utilizar esse *framework* é necessário referenciar seus arquivos na página HTML principal da aplicação. Feito isso, basta aplicar os estilos correspondentes nos elementos da página através de atributos de classe inseridos em cada *tag*. O

Bootstrap é uma ferramenta que pode ser utilizada em aplicações desenvolvidas em diferentes linguagens de programação e *frameworks* (HORN, 2021a).

## 4.3 JAVASCRIPT

O JavaScript (JS) é a linguagem de programação que possibilita a implementação de itens complexos em websites, viabilizando a criação de conteúdo não estático, o qual pode ser atualizado com o tempo, ser interativo, exibir animações, etc. (O. . . , 2021). O JS é o terceiro componente básico na composição do *front-end* de páginas web que seguem os padrões atuais da tecnologia. Entre suas principais funcionalidades estão:

- Armazenar informação através de variáveis;
- Capacidade de realizar diversas operações, tanto com dados numéricos como textuais;
- Executar códigos como resposta a determinados eventos que ocorrem em uma página;

Além dessas aplicações a linguagem permite a criação de API ou Interface de Programação de Aplicativos, que são blocos de construção de código prontos para serem utilizados (O. . . , 2021). Essas interfaces permitem implementar funções de forma simplificada, facilitando exponencialmente o trabalho de desenvolvedores.

API's podem ser divididas em duas categorias, de navegadores e de terceiros. Entre as interfaces de navegadores há a API do DOM, exposta na subseção 4.1.1, que permite manipular o HTML e CSS, aplicando dinamicamente novos estilos em uma página (O. . . , 2021). Na categoria citada também há API's voltadas a geolocalização, áudio, vídeo, etc. Já na categoria de API's de terceiros encontram-se aquelas que não são implementadas diretamente no navegador. Há uma grande variedade destas e para utilizá-las é preciso buscar o código correspondente em páginas da web (O. . . , 2021).

Existem duas maneiras de adicionar códigos da linguagem em um página, semelhante ao CSS. A primeira é de forma externa, na qual o código é escrito em um arquivo de terminação "js." e incorporado no arquivo HTML. Outro meio, é diretamente no corpo do arquivo HTML através das *tags <script>*, todo o conteúdo contido no interior destas é lido como código executável.

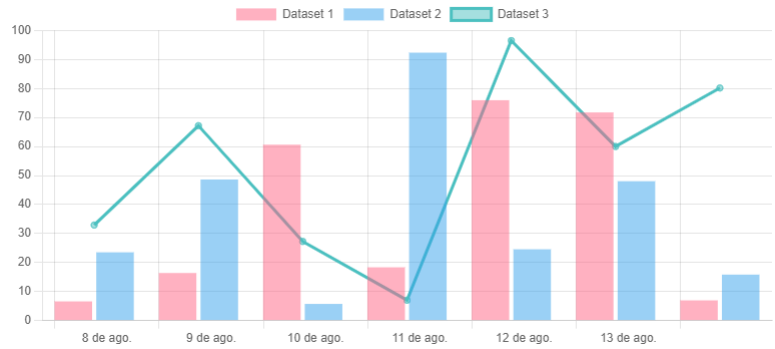
### 4.3.1 Chart.js

Chart.js é uma biblioteca JavaScript voltada a visualização de dados, de código aberto e mantida pelos seus usuários (ROCHA, 2019, p. 57). Ela é baseada no HTML5

Canvas, API do JS que permite criar gráficos vetoriais com duas dimensões (ROCHA, 2019, p. 37).

A biblioteca disponibiliza diversos tipos de gráficos, um exemplo é exibido na figura 7, para utilizá-los basta carregar a biblioteca no corpo do documento HTML, selecionar a visualização desejada na documentação disponível, copiar o código correspondente, colar na aplicação sendo desenvolvida e fornecer os dados que serão representados. Todos os gráficos trazem na sua configuração inicial as características de responsividade e interatividade para adaptação a diferentes telas e possibilidade de interagir com os elementos, respectivamente. Ainda, todas as configurações podem ser modificadas de acordo com a necessidade da aplicação.

Figura 7 – Gráfico de séries temporais da biblioteca Chart.js.




Fonte: Página Chartjs.org.

### 4.3.2 JQuery

JQuery é uma biblioteca de funções do JavaScript que interagem com código HTML. Foi desenvolvida para simplificar os códigos interpretados no navegador do usuário (*client-side*). Em 2013, era utilizada por cerca de 77% dos 10 mil sites mais visitados do mundo, na atualidade é considerada a biblioteca mais popular do JavaScript (TEIXEIRA, 2013). Assim como a biblioteca citada na subseção 4.3.1, está também possui código aberto e somente precisa ser carregada na página HTML principal da aplicação para ser utilizada. Algumas das suas utilidades mais relevantes são:

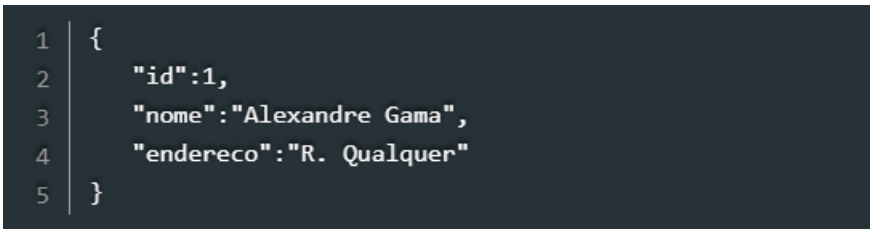
- Adicionar efeitos visuais e animações;
- Acessar e manipular o DOM;
- Carregar componentes Ajax;
- Promover interatividade;

- Realizar alterações de conteúdo;
- Simplificar tarefas JavaScript;

### 4.3.3 JSON

JSON é um formato de dados que se assemelha a objetos do JavaScript, contudo, possui uma série de diretrizes próprias. Tem como características ser compacto e facilmente interpretável, garantindo maior velocidade na execução e transporte de dados. Além disso, está substituindo gradualmente o formato *eXtensible Markup Language* (XML), o mais utilizado anteriormente em serviços web (ROCHA, 2019). A figura 8 apresenta um exemplo do formato de dados JSON.

Figura 8 – Estrutura de dados no formato JSON.




Fonte: Página Devmedia.

JSON é o formato indicado para realizar manipulação de dados com JavaScript, há diversas ferramentas on-line que possibilitam transformar outros formatos em JSON.

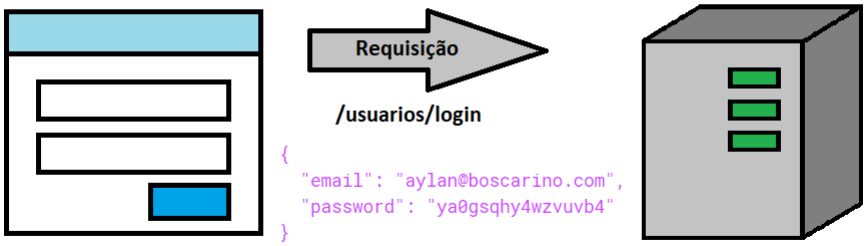
## 4.4 JWT

*JSON Web Token* (JWT) é é um método para realizar autenticação remota e transferência de dados entre serviços por meio de um *token* assinado que autentica uma requisição web. O JWT é amplamente utilizado por API's, estás recebem no seu *endpoint* de autenticação uma requisição, seguindo o protocolo *Hypertext Transfer Protocol* (HTTP), do cliente. A requisição contem no seu corpo dados como endereço de e-mail e senha (BOSCARINO, 2019) , como exposto pela figura 9.

Uma vez realizada a autenticação dos dados enviados pelo cliente no servidor, este cria um *token* JWT assinado com um segredo interno da API e o envia de volta ao cliente (BOSCARINO, 2019), como apresentado na figura 10.

Recebendo o *token* autenticado, o cliente passa a ter acesso aos *endpoints* da aplicação que anteriormente eram restritos. Para isso, o *token* é informado no *header Authorization* de todas as requisições realizadas (BOSCARINO, 2019).

Figura 9 – Requisição do cliente com dados de autenticação.




Fonte: Página Devmedia.

Figura 10 – Envio de *token* pelo servidor.


Fonte: Página Devmedia.

## 4.5 PYTHON

Python é uma das linguagens de programação mais utilizadas da atualidade, ela é versátil e simples de aprender, sendo considerada como uma linguagem de alto nível, cuja sintaxe é voltada para o entendimento humano. Por ser uma linguagem moderna ela foi construída para ser objetiva, assim, desenvolvedores conseguem definir instruções com menos linhas de código (HORN, 2021b). Embora ela seja simples possibilita a tanto criação de pequenos sistemas quanto a criação de aplicações mais complexas como desenvolvimento web, desenvolvimento de jogos ou *Machine Learning*. Além disso, ela é multiplataforma, operando em diversos sistemas operacionais.

O Python é uma linguagem interpretada, deste modo, não gera aquivos executáveis como outras linguagens. Ela possui um interpretador que é responsável por traduzir o código fonte para o formato *byte code* ou código binário e envia-lo para um ambiente virtual conhecido como Python Virtual Machine, onde será executado por

uma máquina virtual Python (HORN, 2021b). Por ser multiparadigma, viabiliza programar seguindo diferentes estilos de programação, como a programação funcional, a procedural e a orientada a objetos (HORN, 2021b).

### 4.5.1 Pandas

Uma das principais propriedades do Python é a modularidade. Dessa forma, há diversas bibliotecas disponíveis na internet que agregam mais capacidades a ela, o que favorece o desenvolvimento de aplicações de diferentes setores, como *Machine Learning* e Ciência de Dados (HORN, 2021b).

Uma das bibliotecas mais conhecidas é o Pandas. Ela é utilizada para realizar operações relacionadas a análise de dados. Suas funções permitem realizar a leitura e manipulação de dados, entregando estruturas e operações para manipular tabelas numéricas e séries temporais.

O Pandas é um pacote gratuito, tem código aberto e pode ser instaladas mediante gerenciadores de pacotes como o pip, que é amplamente utilizado por desenvolvedores que utilizam Python.

### 4.5.2 Fast API

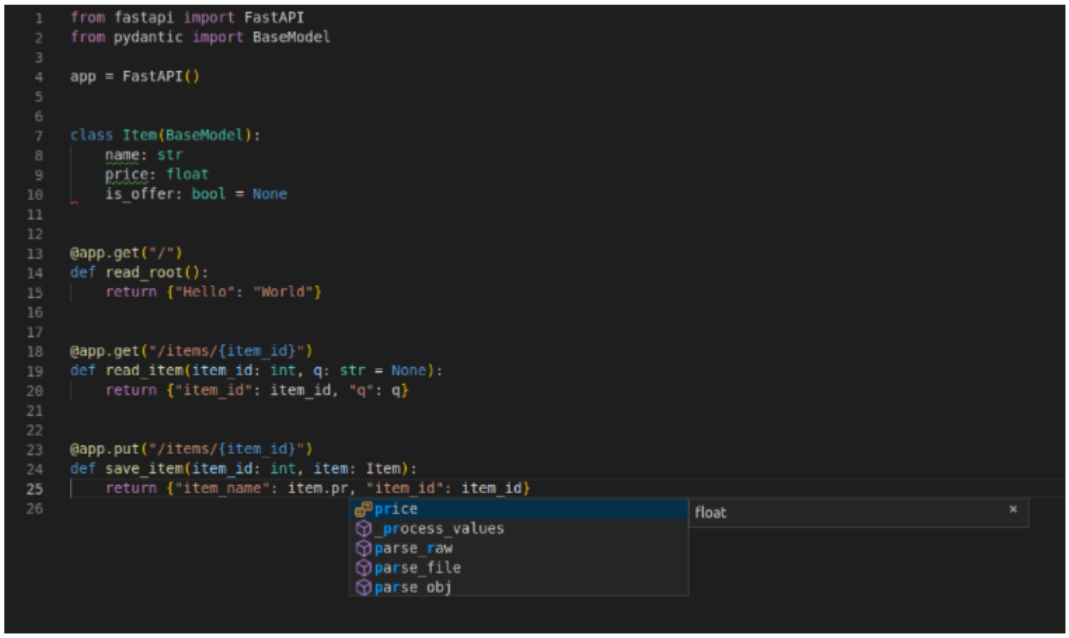
Fast API é um *framework* web moderno, de alta performance voltado ao desenvolvimento de API's com Python 3.6 ou versões mais recentes, ideal para construção do *back-end* de aplicações. Ele é baseado em *type hints*, que servem para indicar a uma função o tipo de variável ela deve receber, restringindo aquelas que não seguem a definição (RAMÍREZ, 2021). As suas principais características são:

- Alta performance: Extremamente eficiente, comparável aos *frameworks* mais conhecidos, como NodeJS e Go;
- Intuitivo: A maioria dos editores de texto atuais oferecem um suporte ideal. Ao ser orientado a *type hints* os ambientes de desenvolvimento conseguem autocompletar perfeitamente o código sendo escrito, como mostrado na figura 11;
- Simples: A API foi desenvolvida com foco em facilidade de uso e aprendizado, reduzindo o tempo de leitura de documentações;
- Poucos *bugs*: Reduz cerca de 40% de erros induzidos por humanos;
- Enxuto: Minimiza a duplicação de código. Cada declaração de parâmetro oferece múltiplos recursos;
- Robustez: Promove depurações mais rápidas, diminuindo o tempo até o envio do código para produção. Disponibiliza documentação visual, automática e interativa

através do Swagger UI, apresentado na figura 12. Está possibilita a interação com os recursos da API mesmo sem ter nenhuma lógica de implementação em execução;

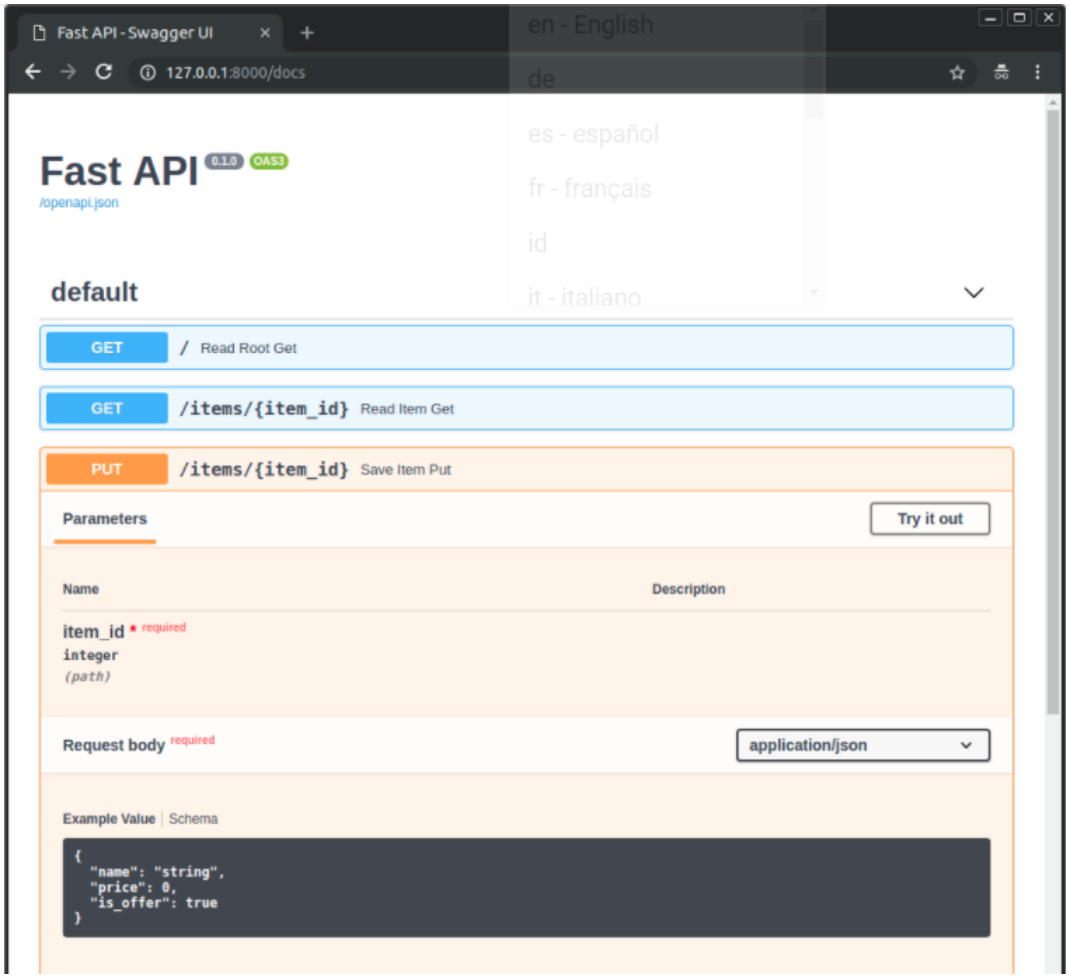
- Baseado em padrões: Baseado e compatível com padrões abertos para API's como OpenAPI e JSON Schema;

Figura 11 – Função de autocompletar em um editor de texto.




Fonte: Página Fastapi.tiangolo.

Figura 12 – Documentação automática gerada via Swagger UI.




Fonte: Página Fastapi.tiangolo.

### 4.5.3 Uvicorn

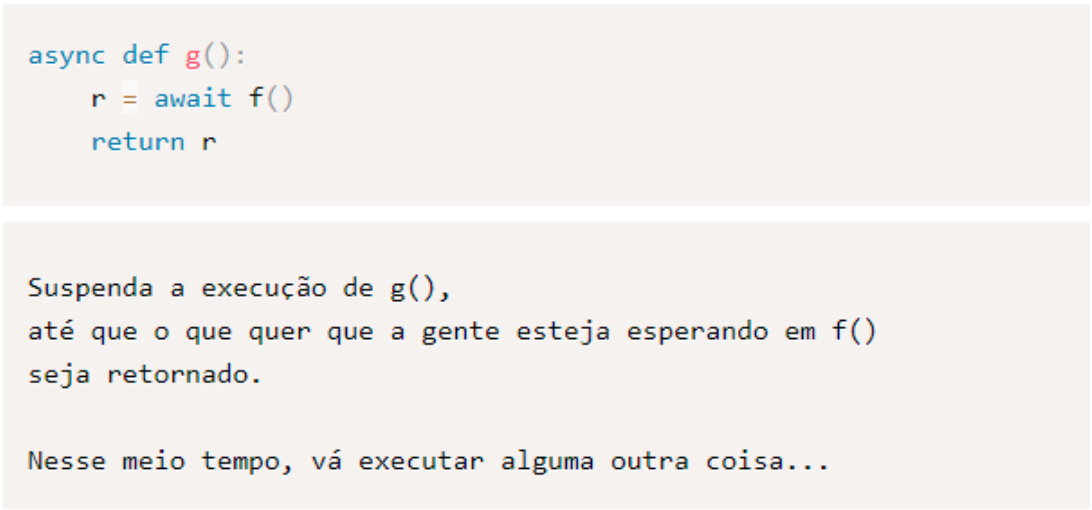
Uvicorn é uma implementação de servidor que segue o modelo *Asynchronous Server Gateway Interface* (ASGI), que proporciona alta performance. Até recentemente, a linguagem Python não tinha uma interface mínima entre servidor e aplicação de baixo nível para estruturas assíncronas.

A especificação ASGI supre essa necessidade permitindo a construção de um conjunto comum de ferramentas aproveitável em todas as estruturas do AsyncIO, biblioteca que usa a sintaxe *async/await* para escrita de código concorrente executado em *coroutines* ou "corrotinas" gerenciadas pelo interpretador Python. É importante destacar que a AsyncIO é uma implementação do Asynchronous I/O, estilo de programação concorrente (LAUBE, 2021).

De forma sintetizada, a concorrência é a capacidade de executar múltiplas tarefas ao mesmo tempo na *Central Process Unit* (CPU), ou unidade central de processamento. A execução pode iniciar, ocorrer e finalizar períodos de tempo sobrepostos. Nos casos onde há somente uma CPU disponível, múltiplas atividades podem ser executados através de *context switching*, mudança de contexto, onde o estado de um processo é armazenado de forma em que este possa ser chamado e executado posteriormente (SHAJI, 2020).

No caso do AsyncIO, uma tarefa pode ser interrompida até o retorno de uma dada resposta. Durante o tempo de espera outras tarefas podem ser resolvidas, tornando o processo completo mais ágil. A figura 13 explica a lógica de uma função assíncrona desta biblioteca.

Figura 13 – Explicação da lógica de programação concorrente do AsyncIO.




Fonte: Página Klauslaube.

A especificação ASGI promove o surgimento de um ecossistema de estruturas

web Python que conseguem concorrer com Node e Go, como a Fast API, em termos de atingir alto rendimento em contextos ligados a operações *Input/Output* (I/O) ou entrada/saída.

### 4.5.4 *Object-Relational Mapping*

*Object-Relational Mapping* (ORM), ou mapeamento de objeto-relacional, é uma técnica de programação que auxilia na conversão de dados entre banco de dados relacionais e linguagens de programação orientadas à objetos (ANDRADE, 2019).

O uso do ORM reduz de forma significativa o trabalho de desenvolvedores na programação de acesso ao banco de dados, aumentando a produtividade. Deste modo, o ORM é considerado um *framework* que promove a redução da impedância, pois realiza todas as conversões necessárias entre o modelo relacional e o modelo orientado a objetos de forma automática. Para isso, cada classe é interpretada com uma tabela e cada linha é tratada como instância do objeto relacionado à tabela em questão (ANDRADE, 2019).

Seu uso também reduz a preocupação dos desenvolvedor com a linguagem SQL, ou linguagem de consulta estruturada, e com as conversões necessárias entre os diferentes tipos de dados, visto que tudo isto é realizado pelo ORM. Assim, torna-se possível utilizar banco de dados relacionais através de objetos escritos em Python, sem necessidade de entender a configuração do banco de dados, apenas mapeando sua estrutura (ANDRADE, 2019).

### 4.5.5 SQLAlchemy

O SQLAlchemy é um *framework* de mapeamento objeto-relacional SQL (ORM) completo, de código aberto, desenvolvido para a linguagem de programação Python. Está biblioteca foi criada a partir da linguagem Python para auxiliar desenvolvedores de aplicativos, incentivando a flexibilização total do SQL. Desta forma, todo acesso ao banco de dados é feito por meio de código Python (ANDRADE, 2019).

Além disso, o SQLAlchemy possui todas as garantias de segurança necessárias para proteger aplicações de ataques ao seus bancos de dados, garantindo assim a criação de aplicativos simples e seguros (ANDRADE, 2019).

## 4.6 GIT

O Git é um sistema de controle de versão de código aberto. A principal função destes sistemas é registrar as mudanças realizadas em um arquivo ou um conjunto de arquivos ao longo do seu desenvolvimento, possibilitando retornar a versões passadas e recuperá-las caso desejado. Além disso, é possível verificar quem submeteu cada mudança, simplificando a resolução de erros na lógica do código.

Uma das principais vantagens dos sistemas de controle de versão é a possibilidade da contribuição simultânea dos membros de um projeto. Para realizar esse trabalho colaborativo são usados serviços web como o GitHub, sendo está página a mais conhecida e utilizada atualmente, hospedando uma quantidade enorme de repositórios Git de desenvolvedores (CHACON; STRAUB, 2014). O GitHub possibilita o compartilhamento de repositórios entre desenvolvedores e permite pedir revisões de código para outros usuários, promovendo o apoio entre a comunidade da página na resolução de problemas e a otimização dos projetos através do trabalho em equipe.

## 4.7 PENTAHO DATA INTEGRATION

O Pentaho Data Integration (PDI) é um software de código aberto que faz parte do pacote Pentaho Open Source Bussiness Intelligence, o qual inclui diversos programas de apoio à tomada de decisão na área de BI. Algumas das soluções inclusas são: ferramentas de análise, criação de processos de ETL ou extração, transformação e carregamento de dados, criação de *dashboards*, entre outras funcionalidades.

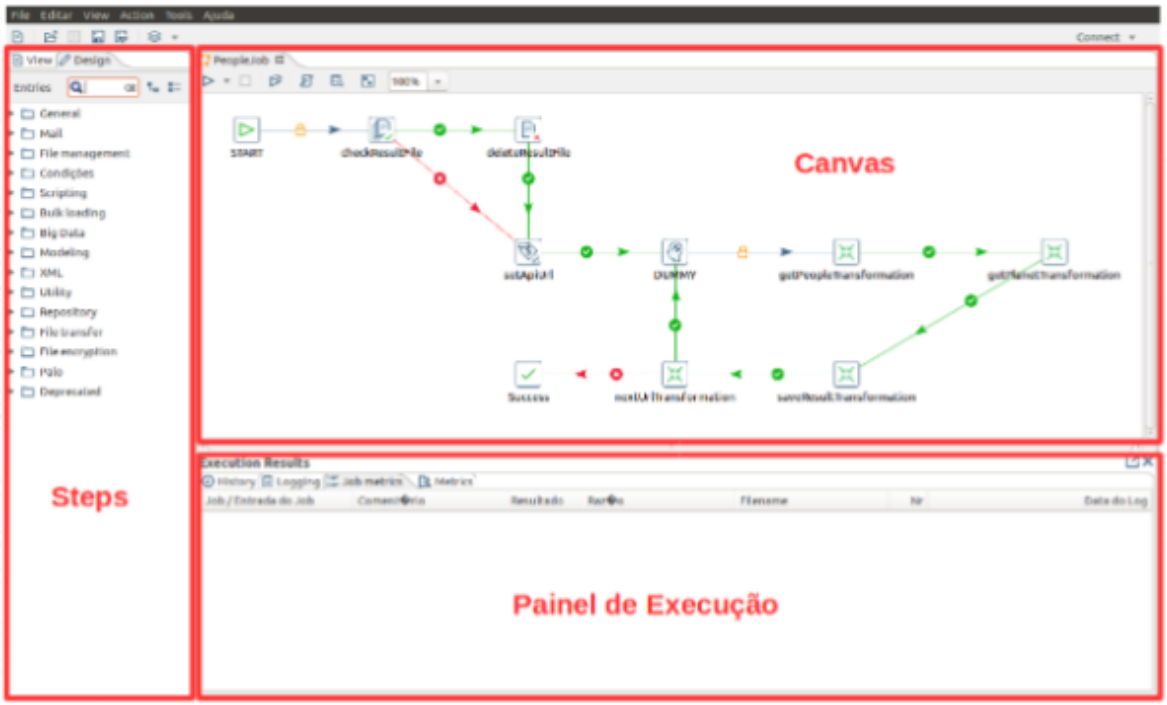
O PDI é conhecido por ser uma ferramenta de fácil domínio. O programa permite realizar inúmeras operações de integração de dados através de uma interface gráfica simples e intuitiva. Algumas dessas operações são:

- Migração de dados;
- Movimentação de grandes volumes de dados;
- Transformação de dados;
- Limpeza de dados;
- Conformidade de dados;

Há algumas ferramentas de desenvolvimento para implementação de processos de ETL no Pentaho, uma destas é o Spoon: interface gráfica utilizada para criar os artefatos do PDI, transformações e *jobs*. A figura 14 exibe esta interface.

As transformações registram os passos de como a extração ou leitura de uma fonte de informação é realizada, ou seja, elas operam sobre os dados. Podem conter: leitura de dados de uma tabela, de um banco de dados, seleção de campos específicos de uma tabela, etc. O aspecto mais importante em uma transformação é que ela opera todas as etapas simultaneamente, não tendo início ou fim, apenas processando todas as linhas recebidas (CESARIO; COSTA, 2017).

Figura 14 – Interface da ferramenta Spoon.




Fonte: Página InfoQ.

*Jobs* são utilizados para sequenciar operações, diferente das transformações eles operam sobre as linhas de dados em paralelo, realizando operações completas, uma por uma. Assim, eles permitem combinar transformações em uma sequencia específica, automatizando uma dada tarefa (CESARIO; COSTA, 2017).

Como o Spoon é só uma interface gráfica para criação dos processos de integração de dados é necessário empregar outras ferramentas para a execução de transformações e *jobs*. Para isso são utilizadas as ferramentas Pan e Kitchen, respectivamente. O conjunto formado pelas três soluções possibilita a criação e execução de artefatos criados para solucionar um problema de extração, transformação e carga de dados em um projeto de ETL com o Pentaho Data Integration (CESARIO; COSTA, 2017).

## 4.8 CLOUD SQL

O Cloud SQL é uma das soluções disponibilizadas pela Google Cloud Platform (GCP), serviço de computação na nuvem oferecido pelo Google. Através da plataforma é possível administrar todos os softwares, armazenamento e serviços utilizados. Entre os serviços oferecidos está o Cloud SQL, que entrega um banco de dados relacional totalmente gerenciado para MySQL, SQL Server e PostgreSQL, SGBDR escolhido

para este projeto. A ferramenta de banco de dados da GCP tem múltiplos recursos e é de baixo custo, o usuário paga de acordo com seu consumo de memória de processamento e armazenamento.

Uma das principais vantagens que motivaram a escolha desse serviço foi o fato dele ser oferecido na nuvem, não precisando de um servidor físico que seria altamente dispendioso. Outras vantagens interessantes são: garantia de segurança no armazenamento através de *firewalls* e criptografia, possibilidade de acesso colaborativo, flexibilidade, uma vez que adaptações na infraestrutura do banco podem ser realizadas facilmente na interface do usuário, entre outras.

# 5 ARQUITETURA

Neste capítulo é apresentada a arquitetura da solução de BI desenvolvida, são expostos os requisitos funcionais e não funcionais assim como os casos de uso.

É importante mencionar que o autor deste trabalho contou com o auxílio e colaboração de colegas de área de sistemas da BIX Tecnologia para tomar decisões de projeto e compreender melhor alguns conceitos utilizados.

## 5.1 CASOS DE USO

Os diagramas de caso de uso em *Unified Modeling Language* (UML), ou Linguagem de Modelagem Unificada, servem para mostrar quem são atores de um sistema e de que forma estes interagem com as funcionalidades definidas dentro do escopo do sistema (DIAGRAMA. . . , 2021).

Dessa maneira, definiu-se que os atores humanos do sistema exercem o papel de “usuário”, sendo este o agente responsável pela utilização da ferramenta e das suas funcionalidades. Ainda, o sistema se comunica com os serviços de banco de dados na nuvem, que são representados pelo ator “GCP”.

No *dashboard* proposto, o usuário é capaz de escolher o período desejado, exibindo apenas dados correspondentes a sua seleção. Com respeito a visualização, o usuário pode acessar a interface do painel tanto utilizando computadores quanto *smartphones*, ou dispositivos com menor resolução, uma vez que este foi projetado para conseguir adaptar a visualização da informação ao tamanho da tela por meio de ajustes automáticos na interface.

Além disso, o usuário tem a possibilidade de interagir com alguns dos componentes apresentados. As tabelas oferecem a opção de modificar a ordenação dos dados de acordo com cada coluna, podendo ser do maior ao menor ou vice-versa para campos numéricos e por ordem alfabética ou vice-versa em campos textuais. Os gráficos de barras apresentam como alternativa a exibição em formato de tabela, as quais expõe a mesma informação. A alteração entre os formatos ocorre por meio de botões próprios para cada formato. Ainda, os gráficos possuem a funcionalidade de exibir *tooltips*, caixas de texto utilizadas para detalhar uma informação presente em uma área do gráfico. Estas surgem quando o usuário posiciona o cursor do mouse sobre a seção desejada.

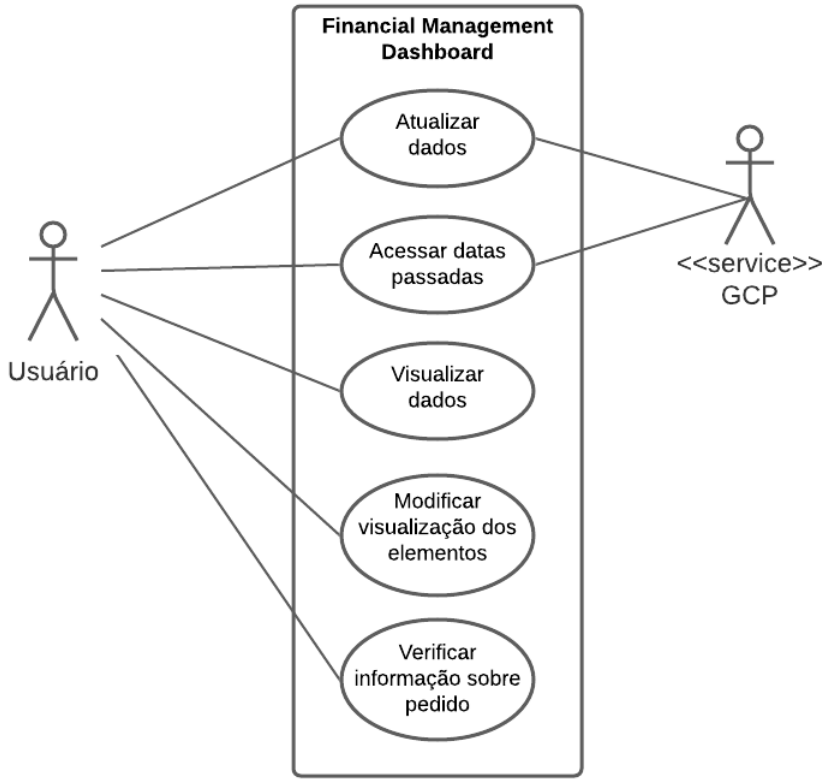
Por fim, o usuário consegue acessar através da tabela de pedidos um registro contendo informações relevantes sobre suas transações. Este exibe o status atual de cada ordem de compra, possibilitando que o usuário consiga acompanhar o andamento do processo desde a aquisição do produto até o recebimento deste.

Quanto ao ator “GCP”, esse está ligado as funcionalidades de acessar diferentes períodos no tempo e atualizar os dados, pois é o intermediário entre a interface da

aplicação e a base de dados. Assim, cada alteração de informação no painel está relacionada a este ator.

Cada uma das funcionalidades citadas acima é apresentada na figura 15 por uma oval que está conectada aos atores do sistema, enquanto o retângulo representa o escopo dos casos de uso do sistema, por fim os bonecos servem de representação aos atores.

Figura 15 – Casos de uso do sistema.




Fonte: Arquivo pessoal.

## 5.2 REQUISITOS DO SISTEMA

Os requisitos podem ser divididos em duas categorias: requisitos funcionais e requisitos não funcionais. Os primeiros têm como objetivo definir o trabalho que o sistema deve realizar a fim de cumprir sua função. Geralmente, eles estão intrinsecamente ligados as funcionalidades do sistema, descrevendo as funções que ele deve ter. Enquanto, os requisitos não funcionais abordam de que forma o sistema realizará as funções propostas pelos requisitos funcionais, não estando relacionados diretamente às funcionalidades, porém indicando necessidades que não podem ser atendidas atra-

vés destas. Assim, os requisitos não funcionais definem as restrições do sistema e as características que ele deve ter (CANGUÇU, 2021).

De acordo com os casos de uso apresentados anteriormente na seção 5.1, definiram-se os seguintes requisitos funcionais:

1. Atualizar interface utilizando os dados disponíveis mais recentes;
2. Separar internamente os dados por usuário;
3. Exibir apenas dados pertinentes ao usuário atual;
4. Apresentar dados através de elementos visuais como tabelas, listas, indicadores ou diversos tipos de gráficos;
5. Dividir internamente datas em períodos de tempo, compreendendo intervalos mensais e trimestrais de um determinado ano;
6. Agrupar dados internamente de acordo com os intervalos de tempo;
7. Exibir dados agrupados por períodos nos diferentes recursos gráficos mostrados;
8. Possibilitar que o usuário selecione períodos específicos, segundo seu interesse, através de elementos visuais como filtros;
9. Permitir que o usuário modifique os elementos visuais, podendo alterar o formato de exibição dos dados em gráficos e a ordenação em tabelas;
10. Integrar com o sistema de login já existente na loja virtual Magento para evitar necessidade de múltiplos logins e garantir acesso somente a usuários que sejam clientes da loja;

Enquanto para os requisitos não funcionais, tem-se o seguinte:

1. Ser compatível com diferentes navegadores, permitindo que qualquer usuário possa utilizar a aplicação e suas funcionalidades;
2. Ser adaptável a diferentes dispositivos, possibilitando que o usuário acesse a aplicação através de telas de diferentes resoluções sem ocorrer erros na interface;
3. Projetar a aplicação com arquitetura modular, tornando-a escalável e simplificando o processo de criação de novas funcionalidades, caso novas expansões sejam solicitadas;
4. Possuir uma interface de usuário seguindo os padrões de design levantados pela empresa contratante, procurando desenvolver um *dashboard* de BI que facilite o entendimento dos usuários e que torne a navegação mais intuitiva;

5. Garantir a disponibilidade dos serviços, visto que um grande volume de usuários deve acessar a aplicação com bastante frequência, assim é necessário que todos os recursos da aplicação funcionem corretamente quando requisitado;

6. Ter bom desempenho, ou seja, o carregamento dos dados deve ocorrer no menor tempo possível para que a renderização do painel finalize o quanto antes, reduzindo o tempo até a interação dos usuários com a aplicação;

7. Garantir a segurança dos dados, uma vez que os dados expostos são sensíveis, pois trazem informações privadas tanto da empresa contratante como de seus clientes, portanto é indispensável proteger o acesso a essas informações;

## 5.3 DEFINIÇÃO DOS INDICADORES

A determinação dos indicadores foi realizada pela PBA visando expor no *dashboard* informação relevante aos seus consumidores, como vendas atingidas, uma vez que os clientes realizam a compra dos itens da Portobello de acordo com suas próprias vendas, ou projetos vendidos. Dessa forma, têm-se o indicador de vendas realizadas em valor monetário, no caso a unidade é o dólar, representado pela soma acumulada do valor obtido a partir das vendas em um determinado período. Além disso, também é exposto o indicador de meta de vendas em valor monetário de acordo com o período selecionado. Este tem o intuito de informar ao usuário o valor a ser atingido, pois são oferecidos descontos nas compras realizadas após o atingimento desta meta.

O último indicador é KPI composto pelos dois indicadores citados anteriormente. Este exibe o percentual de atingimento para um dado período. Os períodos que podem ser filtrados no painel correspondem a trimestres. Dessa forma, o atingimento é sempre calculado em ciclos de três meses ou múltiplos, uma vez que o usuário pode selecionar mais de um ciclo por vez. O cálculo do indicador é dado pela Equação (1):

$$\text{Atingimento em vendas (\%)} = (\text{Vendas realizadas(\$)} / \text{Meta de vendas(\$)}) \times 100. \quad (1)$$

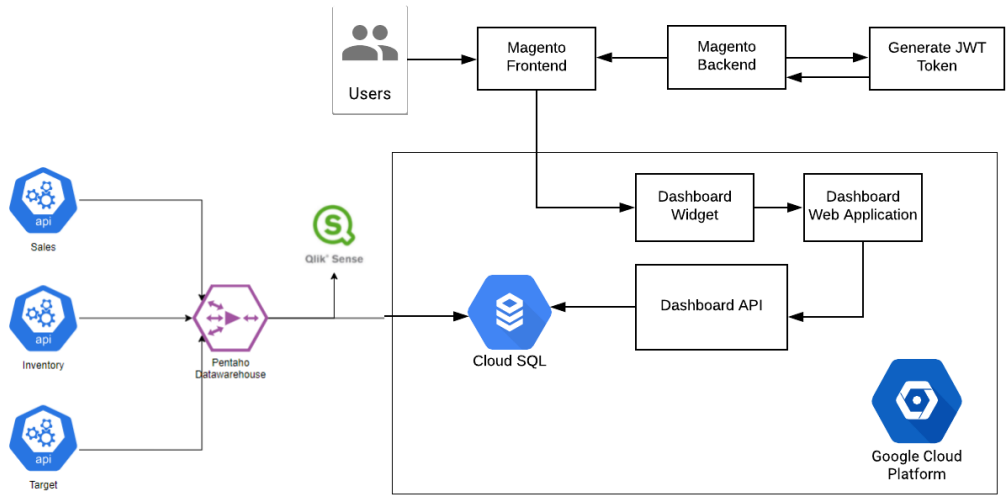
É importante mencionar que o indicador de atingimento percentual também é utilizados pelos gráficos de barras. Assim, foram projetados dois gráficos desse tipo com o propósito de expor o atingimento mensal e trimestral de forma visual. Para o atingimento mensal utiliza-se o valor de vendas realizadas por mês e a meta trimestral disponível é divida em três partes iguais, assim, cada barra indica o percentual atingido num dado mês. Já o gráfico trimestral expõe a mesma informação trazido pelo indicador de atingimento, contudo, caso sejam selecionados mais trimestres no filtro de datas o gráfico exibe uma barra para cada período, enquanto o indicador exibe o valor absoluto. Além disso, foi solicitado pela contratante uma lógica de cores para o indicador de atingimento e para as barras dos gráfico. As barras e o indicador devem ser exibidas

na cor vermelha quando o resultado alcançado estiver abaixo da meta, ou seja, abaixo de 100%. Quando o atingimento for 100% ou maior, a cor dos elementos deve ser azul.

## 5.4 ESTRUTURA DO SISTEMA

Para iniciar o desenvolvimento da aplicação foi necessário separar a estrutura do sistema como um todo em componentes, os quais estão expostos na figura 16 abaixo. A criação de um diagrama dos elementos do sistema proporciona uma visão mais clara sobre cada item do conjunto e a qual domínio cada um desses pertence. Dessa forma, entendeu-se melhor quais eram as atribuições da BIX e quais eram responsabilidades da empresa responsável pela loja virtual na plataforma de *e-commerce* Magento.

Figura 16 – Diagrama da estrutura do sistema.




Fonte: Arquivo pessoal.

Nas subseções a seguir a estrutura completa é desmembrada para que cada componente possa ser explicado com mais detalhes.

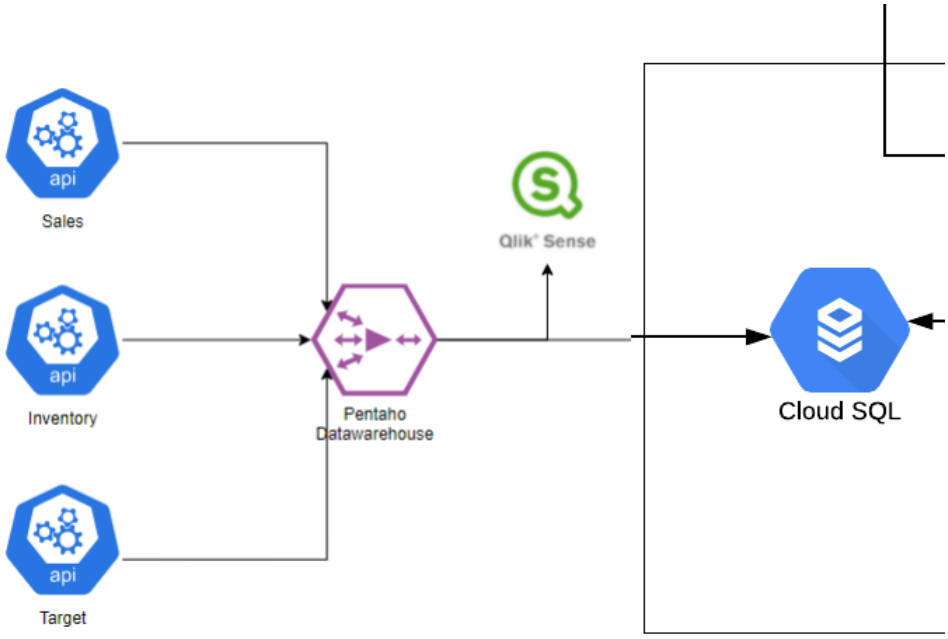
### 5.4.1 Estrutura da integração de dados

A estrutura de integração de dados é composta pelos elementos exibidos na figura 17 que correspondem as API's de coleta de dados utilizadas pela empresa PBA, ao sistema de *datawarehouse* e integração de dados do software Pentaho Data Integration e ao banco de dados na nuvem da Cloud SQL da plataforma GCP.

As API's de coleta são responsáveis por alimentar de forma ininterrupta o PDI como novas informações obtidas de diferentes fontes, como dados do inventário, pedidos, metas, etc. Há diversas API's em operação para suprir a necessidade de todas

as aplicações que utilizam dados como recurso primário. Este é caso dos sistemas de BI, apontados no diagrama pelo ícone do software Qlik Sense utilizado para esta finalidade, assim como da aplicação desenvolvida.

Figura 17 – Diagrama da estrutura de integração de dados.




Fonte: Arquivo pessoal.

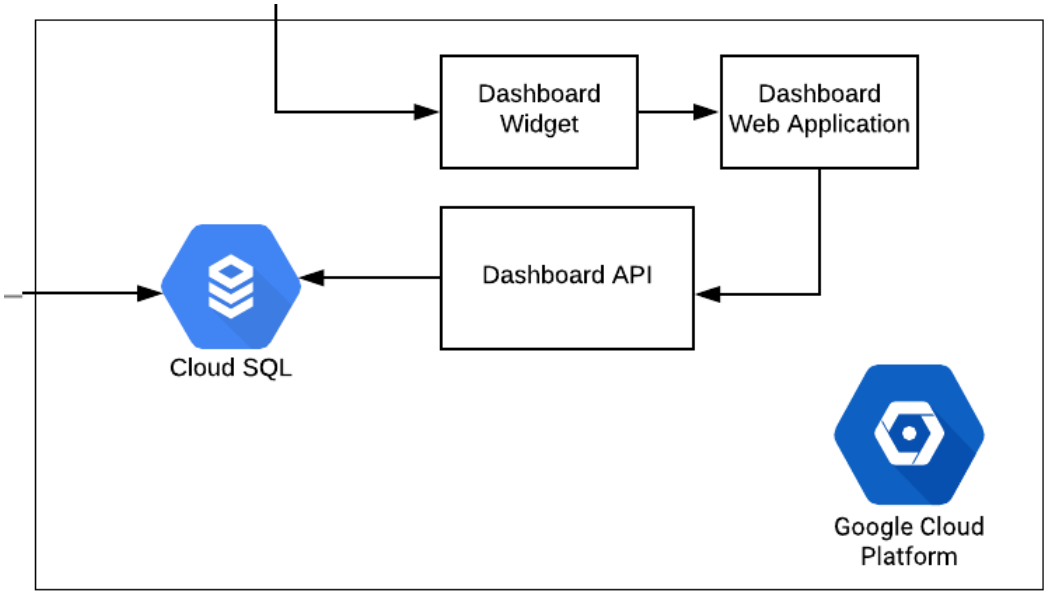
O Pentaho realiza o processo completo de ETL a partir dos dados coletados pelas API's. Ele deve receber toda a informação coletada, realizar todas as modificações e ajustes necessários, como agrupamentos e formatação de dados, para finalmente enviar essa informação tratada a outros sistemas.

O banco de dados PostgreSQL da Cloud SQL recebe todos os dados separados enviados pelo PDI e os armazena separada nas respectivas tabelas de cada fonte, como a tabela de pedidos (*salesorder*) que compreende todos os dados de pedidos realizados ou a tabela de faturas (*invoices*) que contém todas as informações de pedidos já finalizados.

### 5.4.2 Estrutura da aplicação web

A estrutura da aplicação web é composta pelos elementos exibidos na figura 18, os quais se referem ao banco de dados PostgreSQL da Cloud SQL, a API do *dashboard*, a aplicação web do *dashboard* e ao *widget* do *dashboard*. Todos estes elementos funcionando a partir da computação em nuvem graças a plataforma GCP.

Figura 18 – Diagrama da estrutura da aplicação web.




Fonte: Arquivo pessoal.

O banco de dados foi citado na subseção 5.4.1, ele recebe todos os dados em tabelas através dos *jobs* do software Pentaho. Essas tabelas formam a base de dados da aplicação e alimentam todo o sistema. A informação é consultada no banco através das API's da aplicação.

O *dashboard* tem seu *back-end* composto por diversas API's, estas são responsáveis por realizar todas as requisições de dados ao banco. Deste modo, neste módulo do sistema são selecionados todas as informações necessárias para a construção do *front-end* da aplicação, ou seja, são definidos quais campos de quais tabelas devem ser filtrados. Além disso, nesta estrutura também ocorre o tratamento de dados, a fim de adequá-los a formatos mais facilmente interpretados pelo *front-end* e não onerar o usuário (*client-side*) com processamentos que podem ser realizados pelo servidor.

O próximo componente desta estrutura é a aplicação web, considerando o *front-end* que constrói a interface do usuário. Neste elemento são acessados os *endpoints* que realizam a chamada das API's do *back-end* da aplicação. Uma vez que essa informação requisitada pelas API's é acessada torna-se viável armazená-la em variáveis e realizar operações com os dados disponíveis. Assim, por meio de funções contidas nesta camada, os dados são empregados para formar os diversos elementos visuais da interface e entregar assim informação estruturada ao usuário.

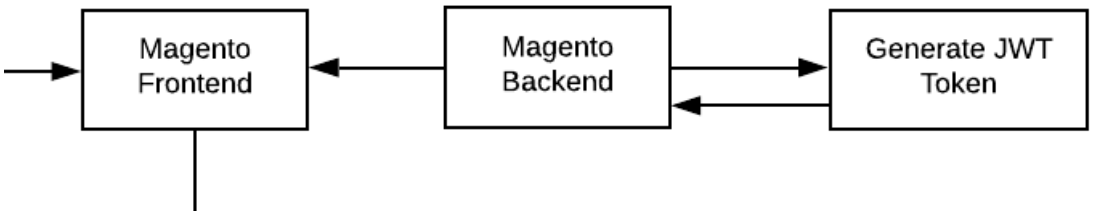
Por fim, o último item desta organização é o *dashboard widget*, que de forma simplificada é uma classe escrita na linguagem JavaScript, a qual serve para integrar o painel a outras aplicações, como páginas web. Essa classe contém uma função

construtora que cria um *frame* HTML, conhecido pela *tag iframe*, no documento original HTML da página na qual se deseja inserir o painel. Dentro desse quadro gerado é inserida a visualização da aplicação. O *widget* simplifica a modularização da aplicação desenvolvida.

### 5.4.3 Estrutura da integração com o Magento

O último segmento da estrutura completa da aplicação é a estrutura de integração com o *e-commerce* da plataforma Magento, exposta na figura 19. Este elemento é de responsabilidade de outra empresa contratada para lidar com a infraestrutura da plataforma.

Figura 19 – Diagrama da estrutura de integração ao Magento *e-commerce*.




Fonte: Arquivo pessoal.

Através do *front-end* do Magento os clientes da PBA acessam diversas funcionalidades, entre elas o painel de BI criado para acompanhamento de pedidos e metas. Para acessar a loja virtual o usuário deve realizar o login com seu cadastro pessoal.

Para evitar a necessidade de um segundo login, que prejudicaria a experiência do usuário, foi necessário inserir uma requisição do tipo “POST” no *back-end* do Magento. Está tem como propósito gerar o *token* JWT e compartilhá-lo com o *back-end* do *dashboard*. O corpo do *token* deverá conter parâmetros, como ID do cliente, que serão utilizados para acessar o banco de dados, filtrando a informação de acordo com o respectivo cliente. Assim, garante-se a privacidade dos dados e mantém-se a aplicação segura.

# 6 DESENVOLVIMENTO

Neste capítulo são apresentadas as fases de desenvolvimento da aplicação criada para suprir a demanda da empresa PBA. É exposto o design da interface disponibilizado pelo cliente, o qual serviu como modelo para a elaboração do projeto. Além disso, são representadas as conexões entre o sistema de integração de dados do PDI, o banco de dados Cloud SQL na nuvem e a aplicação. O autor do documento foi responsável pelo desenvolvimento completo do *front-end* da aplicação, das API's do *back-end*, da estrutura de dados e da integração entre o *front-end* e *back-end*.

É importante mencionar que algumas figuras apresentadas neste capítulo, na subseção 6.1.2, foram parcialmente desfocadas com o propósito de evitar expor informações sigilosas sobre o acesso a base de dados do cliente. Apesar disso, essas informações não são necessárias para a compreensão do desenvolvimento realizado.

## 6.1 ESTRUTURA DE DADOS

Está seção foi escrita com o intuito de detalhar a construção da estrutura de dados, a qual abrange a criação de um banco de dados relacional na nuvem e a conexão deste com o sistema de integração de dados utilizado pela contratante.

### 6.1.1 Banco de dados

A criação do banco de dados envolveu diversas etapas de discussão sobre a plataforma mais indicada para este propósito. Atualmente as opções mais conhecidas no mercado são as soluções oferecidas pelas plataformas: Amazon Web Services (AWS) da empresa Amazon, Azure da empresa Microsoft e GCP do Google. Ainda, cada uma dessas ferramentas oferece diversos bancos diferentes como: PostgreSQL, MySQL, Oracle, SQL Server, entre outros.

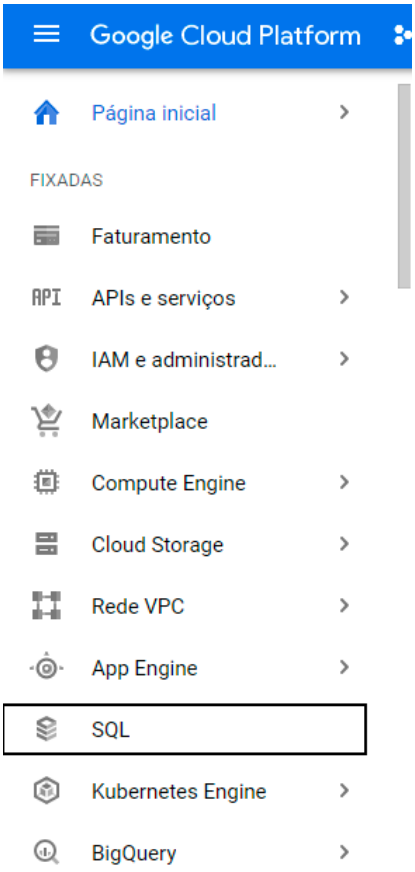
Após discutida qual plataforma era a melhor opção dado o escopo do projeto decidiu-se seguir com a solução do Google, a Cloud SQL da plataforma GCP, pois, outros setores da empresa BIX já utilizam esta ferramenta, facilitando a resolução de eventuais problemas caso houvesse a necessário de suporte técnico. Entre os diversos bancos de dados foi escolhido o PostgreSQL, essa decisão foi motivada pela popularidade do banco e pela disponibilidade de documentação on-line sobre a ferramenta, além disso o autor possui conhecimento desta tecnologia, evitando a necessidade de despender tempo com aprendizado.

O primeiro passo para conseguir criar o banco de dados foi realizar o cadastro na GCP. Uma vez finalizada esse processo foi necessário associar a conta criada à conta da BIX Tecnologia com o objetivo de centralizar as despesas associadas. É importante destacar que toda a infraestrutura de dados depois será migrada pela

contratante para seu próprio domínio, já que este recurso é pago e deve ser mantido para que aplicação possa funcionar.

O passo seguinte foi acessar o site do console da GCP, que contém todas as funcionalidades que podem ser utilizadas na plataforma. A página dispõe de uma barra de navegação lateral, exibida na figura 20, a qual facilita o acesso direto a todas as funções oferecidas. Para criar o banco é necessário utilizar a opção “SQL”, marcada na figura 20 por um retângulo preto.

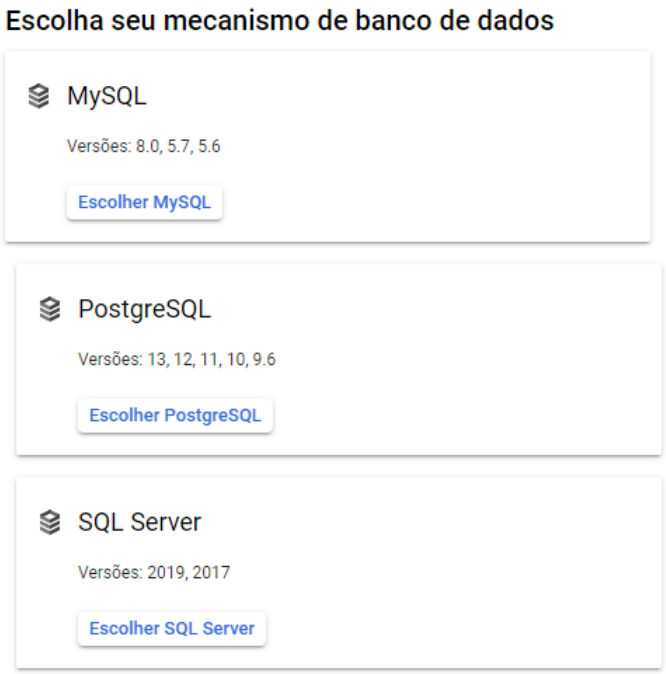
Figura 20 – Barra de navegação do console da plataforma GCP.




Fonte: Arquivo pessoal.

Em seguida, na página seguinte escolheu-se o mecanismo de banco PostgreSQL, entre as três opções disponíveis na ferramenta, apresentadas na figura 21. Após finalizado o processo completo de criação do banco é possível verificar se este foi gerado corretamente através de softwares como o pgAdmin que é uma plataforma de código aberto voltada à administração e desenvolvimento de PostgreSQL e seus sistemas de gerenciamento de banco de dados relacionados.

Figura 21 – Mecanismos de banco de dados oferecidos na plataforma GCP.



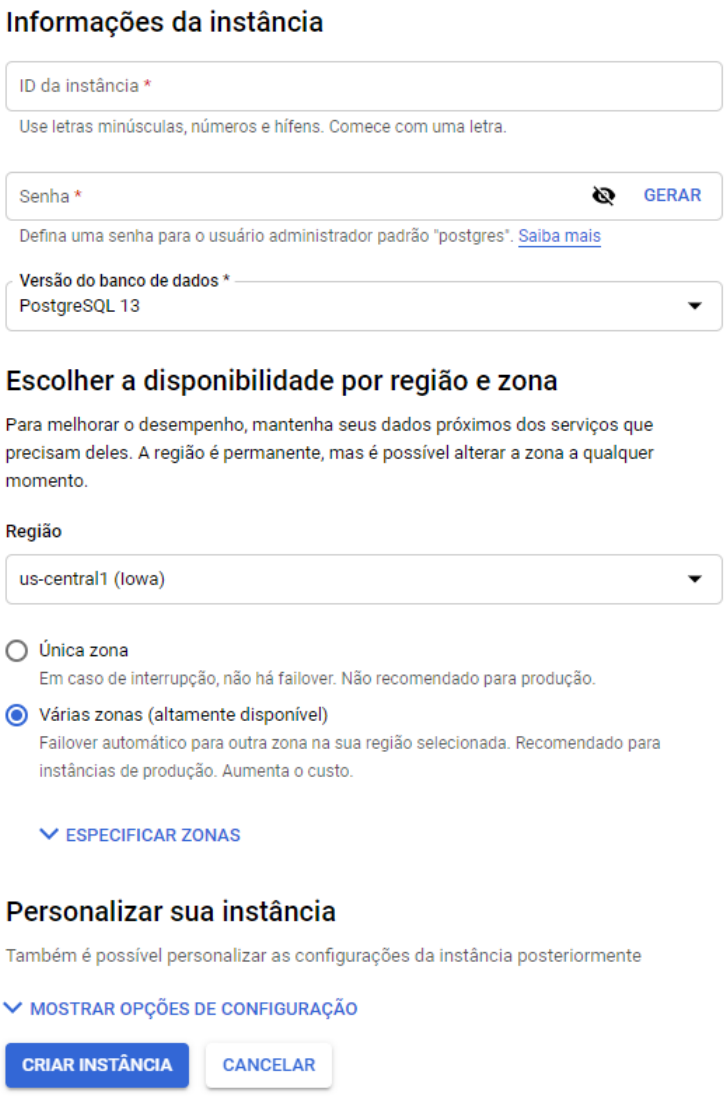

Fonte: Arquivo pessoal.

Logo após, foram preenchidas as informações do banco de dados, utilizando as campos disponíveis exibidos na figura 22. Os principais campos da configuração são o ID da instância e a senha de acesso, ambos são usados posteriormente para realizar consultas e importação de dados. As demais opções permitem escolher a versão do PostgreSQL desejada, para fins de compatibilidade, e também em qual região se encontra a instância do banco, mesmo que este não utilize um servidor físico. Ainda, existe a opção de escolher múltiplas regiões para evitar incidentes em caso de interrupções da operação em uma determinada área, contudo, está escolha aumenta os custos. Em seguida, há diversos campos de personalização das configurações da instância, os principais são:

- Tipo de máquina: permitindo escolher o desempenho esperado a partir da memória utilizada pela vCPU, que é a unidade de processamento virtual de um servidor físico;
- Armazenamento: podendo escolher entre os tipos SSD e HDD, a primeira opção entregando maior desempenho. Além disso, também é possível escolher a capacidade de armazenamento em GigaByte (GB), afetando novamente o desempenho;

- Conexões: existindo duas possibilidades, IP particular ou público. A opção de IP público apresenta uma série de riscos, uma vez que qualquer usuários sem permissão de acesso pode tentar invadir a base de dados;
- Backups: utilizados para proteger-se contra perda de dados. Estes podem ser automáticos, ocorrendo em períodos definidos, ou recuperações pontuais, nas quais se escolhe um momento exato de retorno no tempo;
- Manutenção: onde seleciona-se com que frequência a instância deve ser reinicializada para que atualizações sejam feitas;

Figura 22 – Campos de configuração do PostgreSQL na plataforma GCP.




Fonte: Arquivo pessoal.

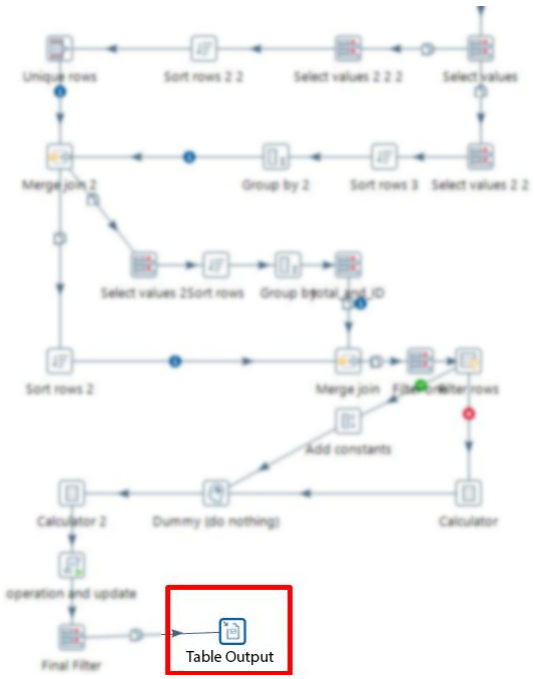
### 6.1.2 Carregamento de dados

Finalizada a criação do banco de dados, chegou-se a etapa de alimentação desse com dados relevantes. Para isto, foi necessário utilizar o sistema de integração de dados da PBA, que é executado por meio de ETLs desenvolvidas pela empresa no software Pentaho Data Integration.

É importante destacar que para este projeto não foi necessário desenvolver uma nova ETL, pois a contrante disponibilizou o seu sistema de integração de dados, com diversas ETL's, para entregar a informação necessária na construção da aplicação. Assim, somente adicionou-se uma lógica de *Load* exclusiva para o envio dos dados ao banco criado.

A figura 23 abaixo apresenta uma seção de ETL utilizada, onde há diversas transformações de dados representadas por cada bloco, as quais fazem parte do *job* que é o conjunto completo de todos esses passos. Na imagem os blocos das diversas transformações foram desfocados para manter a privacidade dos processos da organização, apenas a função *Table Output* em destaque foi mantida sem distorções, visto que está tem papel fundamental no procedimento de carregamento de dados documentado nesta subseção.

Figura 23 – *Job* e transformações do software Pentaho Data Integration.

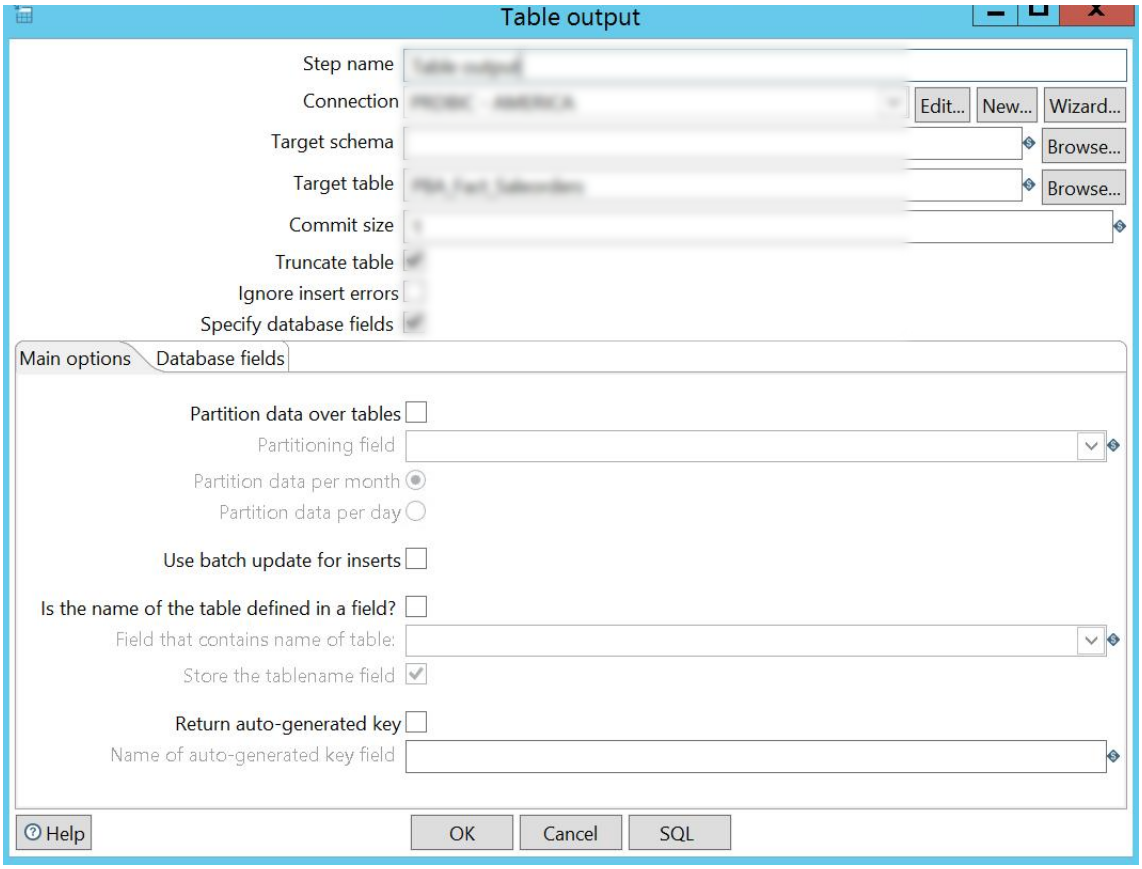



Fonte: Arquivo pessoal.

Ao acessar a função destacada é possível realizar diversas configurações de saída de dados, podendo escolher qual o destino final destes. Neste projeto foi es-

colhida a saída para banco de dados PostgreSQL, no entanto, há diversas opções disponíveis. A figura 24 exibe os campos configuráveis da transformação *Table output*, o campo *connection* permite inserir a saída de dados.

Figura 24 – Transformações *Table output* do software Pentaho Data Integration.




Fonte: Arquivo pessoal.

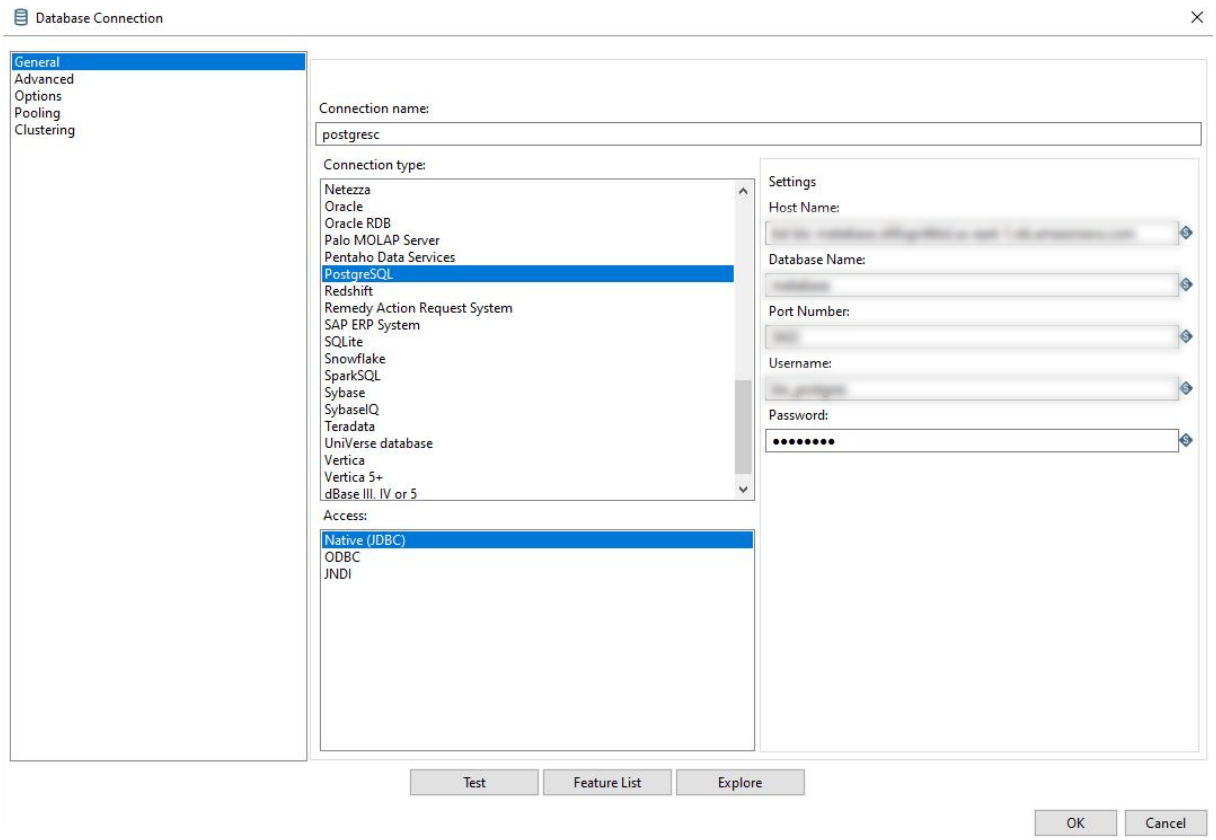
Ao clicar na opção de editar conexão, exibida na figura anterior, abre-se uma nova janela, exibida na figura 25, que contém todos os campos que devem ser preenchidos a fim de realizar a integração entre o PDI e a base de dados.

Em primeiro lugar, adicionou-se um nome para a conexão e selecionou-se o tipo desta, no caso PostgreSQL. Em seguida foram preenchidos os campos *Host Name*, *Database Name*, *Port Number*, *Username* e *Password* com as respectivas informações retiradas do banco de dados criado previamente na GCP. Ao final desse processo foi possível testar a conexão utilizando o botão *Test* exibido na própria janela de configurações da conexão.

Dessa forma, após finalizar a etapa anterior, bastou executar a *job* do PDI e verificar se o banco de dados tinha sido alimentado corretamente. Para isso, utilizou-se o software pgAdmin. Após adicionar todas as credencias necessárias e configurar a

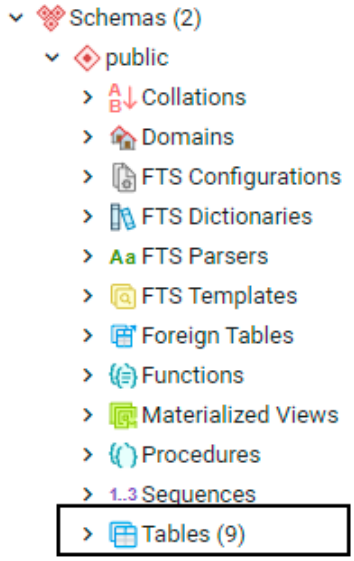
conexão entre o programa e o banco, foi possível verificar que as todas as tabelas de dados tinham sido inseridas no banco corretamente, como exposto na figura 26.

Figura 25 – Configurações da conexão com uma base de dados no software PDI.




Fonte: Arquivo pessoal.

Figura 26 – Conexão com base de dados utilizando o software pgAdmin.




Fonte: Arquivo pessoal.

## 6.2 DESENVOLVIMENTO BACK-END

O objetivo desta seção é descrever a construção do *back-end* da aplicação, expondo a lógica por trás dos códigos de conexão com o banco de dados, das API's com autenticação de usuário e filtragem de datas e do tratamento de dados.

### 6.2.1 Integração entre *back-end* e banco de dados

O primeiro passo para estabelecer a conexão entre o *back-end* da aplicação e o banco de dados foi importar a biblioteca SQLAlchemy da linguagem Python. Assim, importou-se as funções relevantes para criação da conexão (*create_engine*), criação de sessão (*sessionmaker*) e a declaração de base (*declarative_base*) que é usada para declarar uma classe base, a qual serve para indicar a uma função qual o formato da resposta esperada. A função utilizada para conectar ao banco deve receber um parâmetro contendo todas as informações necessárias sobre o banco, na figura 27 a constante "DB_URL" destacada contém todas essas informações requisitadas, armazenando de forma segura o endereço de conexão.

Figura 27 – Conexão com banco de dados utilizando a biblioteca SQLAlchemy.

```
from sqlalchemy.orm import declarative_base
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from server import config as c


engine = create_engine(c.DB_URL, echo=True)

Base = declarative_base()
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
```

Fonte: Arquivo pessoal.

Em seguida declararam-se as classes bases em um arquivo separado, exposto na figura 28. Para isso, foi necessário importar a função *Base*, criada no trecho de código anterior, exibido na figura 27. Além dessa importação, também foi preciso trazer da biblioteca SQLAlchemy os tipos de cada argumento declarado, como o tipo *Integer*, utilizado para indicar quais argumentos de uma classe devem receber números inteiros. Além disso, no interior de cada classe foi necessário declarar, através do argumento "__tablename__", a que tabela pertencem os campos listados, representados pelos demais argumentos. Posteriormente estas bases foram empregadas na construção das API's de dados.

Figura 28 – Declaração de classe base utilizando SQLAlchemy.

```python
from sqlalchemy import String, Column
from sqlalchemy.sql.sqltypes import Date, Float, Integer
from .database import Base

class Invoice(Base):
    __tablename__ = 'invoices'

    id = Column(Integer, primary_key=True)
    customerid = Column(Integer, nullable=False)
    invoicedate = Column(Date)
    quantity = Column(Float, nullable=False)
    cost = Column(Float)
    purchasingunits = Column(String(255))
    itemname = Column(String(255), nullable=False)
```

Fonte: Arquivo pessoal.

### 6.2.2 API's com Fast API

Para iniciar a construção das API's primeiramente importou-se a classe Fast API, utilizada em seguida para definir a variável "app" como instância da mesma (Figura 29). Também foi importada a rota de autenticação criada em um arquivo à parte. Por fim, utilizou-se uma função de inclusão de rotas para adicionar o caminho de autenticação importado na instância Fast API.

Figura 29 – Declaração de instância Fast API e rotas de autenticação.

```python
from fastapi import FastAPI
from server.authentication import routes as authentication_routes


app = FastAPI()
app.include_router(
    authentication_routes.router
)
```

Fonte: Arquivo pessoal.

Logo após, foram desenvolvidas todas as API's, as quais seguem uma estrutura semelhante à exposta na figura 30. Seguindo a sintaxe da Fast API, a primeira etapa

consistiu em criar o decorador na primeira linha anterior a função da API, este método envolve a função e modifica seu comportamento. No caso utilizado, empregou-se a requisição do tipo “GET” para indicar que o retorno esperado da função deve ser do tipo dado. Além disso, passou-se o parâmetro de direção, no caso “/invoices”, o qual determina que a função abaixo do decorador deve lidar com todas as requisições provenientes deste caminho (*path*).

Figura 30 – API desenvolvida com Fast API.

```python
@app.get('/invoices', status_code=200)
def get_all_invoices(
    customer_id: str = authentication.user_dependency,
    year: int = None,
    quarter: int = None
):
    min_date, max_date = year_quarter_filter(year, quarter)

    try:
        db = database.SessionLocal()
        invoices = (
            db.query(Invoice, Producttechinfo)
            .filter(Invoice.customerid == customer_id)
            .filter(and_(Invoice.invoicedate ≥ min_date,
                         Invoice.invoicedate ≤ max_date))
            .join(
                Producttechinfo,
                Invoice.itemname == Producttechinfo.productname
            ).all()
        )
    finally:
        db.close()
    invoices_dict = [···
    ]

    return invoices_dict
```

Fonte: Arquivo pessoal.

A função que segue após o decorador é chamada pelo Fast API toda vez que a direção passada recebe uma requisição. Dessa maneira, ao ser executada esta espera uma série de parâmetros no endereço da requisição, esses são o *customer_id*, utilizado pelo sistema de autenticação e para a filtragem de clientes, *year* e *quarter*, utilizados para realizar a filtragem de datas, sendo apenas o primeiro obrigatório.

O corpo da função traz em primeiro lugar outra função que contém a lógica de filtragem de datas a partir dos parâmetros citados no parágrafo anterior. Logo após, definiu-se “db” como uma sessão no banco de dados e a variável “invoices” que recebe os dados da consulta realizada no banco, identificada na figura 30 pela função *query*.

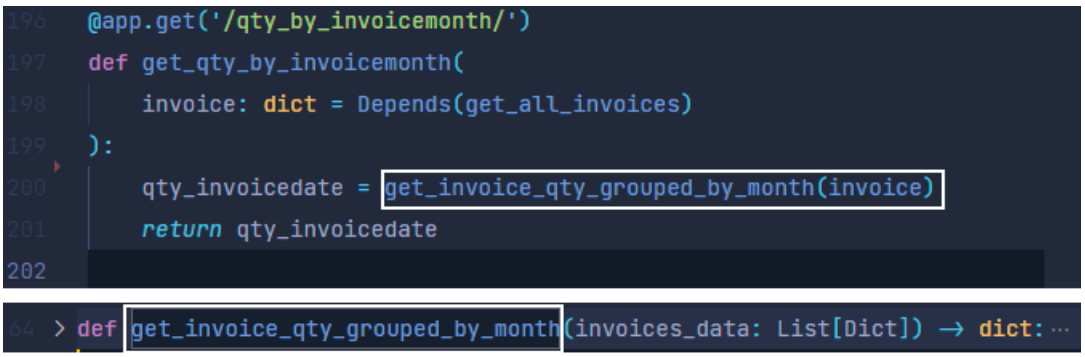
O SQLAlchemy oferece meios para realizar a consulta de forma análoga ao modo tradicional usando SQL, permitindo realizar toda a filtragem de dados necessária. Definiu-se que sessão é finalizada ao término da consulta. Ainda, inseriu-se uma função para modificar o formato dos dados, a fim de que estes retornem como o formato de uma lista de dicionários, estruturas da linguagem Python similares aos objetos do JavaScript, onde cada item da lista é um registro ou linha de uma tabela consultada. É importante destacar também que as classes base definidas previamente, citadas na subseção 6.2.1, são utilizadas no momento da consulta.

### 6.2.3 Tratamento de dados

A etapa de tratamento de dados consistiu no desenvolvimento de um arquivo contendo todas as funções necessárias para este propósito. Para isso criou-se um documento Python no qual importou-se a a biblioteca Pandas, utilizada para realizar diversas operações.

Deste modo, definiu-se uma estrutura de consulta de dados onde foram criadas duas API’s principais para coletar todos os dados brutos, uma delas exposta na figura 30, e outras secundárias (Figura 31), voltadas a realizar o tratamento desses dados coletados e entregá-los quando solicitado. Essas últimas contém no seu escopo as funções de tratamento de dados desenvolvidas, pois, são requisitadas pelo *front-end*, que necessita dos dados em diferentes formatos para gerar as diversas visualizações apresentadas pela interface.

Figura 31 – API utilizando função de tratamento de dados.




Fonte: Arquivo pessoal.

A figura 31 exibe o código de uma API que ao ser chamada deve receber os

dados da requisição da API da figura 30 e tratá-los utilizando a função de tratamento importada, a qual está destacada na imagem pelo retângulo branco. Assim, foram desenvolvidas 16 funções para tratamento de dados, contudo, apenas 8 são empregadas pelas API's, as demais são reutilizadas por essas 8 para executar operações básicas.

## 6.3 DESENVOLVIMENTO FRONT-END

Esta seção descreve a construção do *front-end* da aplicação, expondo como foram construídos os diversos elementos da interface gráfica, como são buscados os dados e como estes são empregados pelas visualizações apresentadas.

### 6.3.1 Layout

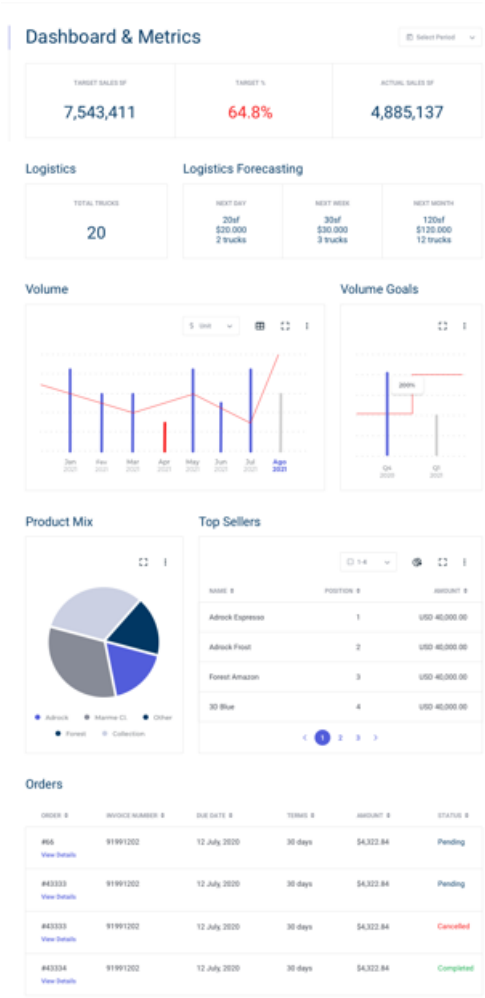
A construção do layout do projeto baseou-se em um design predefinido pela contratante, o qual é apresentado na figura 32. O modelo apresenta as diretrizes da marca, expondo paletas de cores, fontes, formatação da página e distribuição dos elementos. Dessa maneira o design serviu como referência para o desenvolvimento do *front-end* do projeto.

Assim, inicialmente foram estruturados os arquivos HTML, CSS e JavaScript. Criou-se uma pasta cujo objetivo é armazenar todos os *templates* criados no projeto, contudo, esta armazena apenas um único arquivo HTML, o qual é responsável por gerar o esqueleto da página, centralizar todos os elementos criados utilizando JavaScript e atribuir a eles estilos definidos através da linguagem CSS. Esse arquivo HTML é utilizado para renderizar o *dashboard* a cada requisição realizada pelo usuário para visualização da página.

Empregou-se o *framework* de estilização Bootstrap para reproduzir a disposição dos elementos do painel, seguindo o modelo de referência. Dessa forma, utilizando as funcionalidades de código oferecidas pelo *framework* dividiu-se a página em seções. Cada uma dessas divisões é constituída por uma linha principal (*row*), a qual é subdividida em colunas (*column*, seguindo o modelo da figura 33. Por sua vez, as colunas são representadas pelo código "col" e são responsáveis por introduzir um dos diversos elementos visuais.

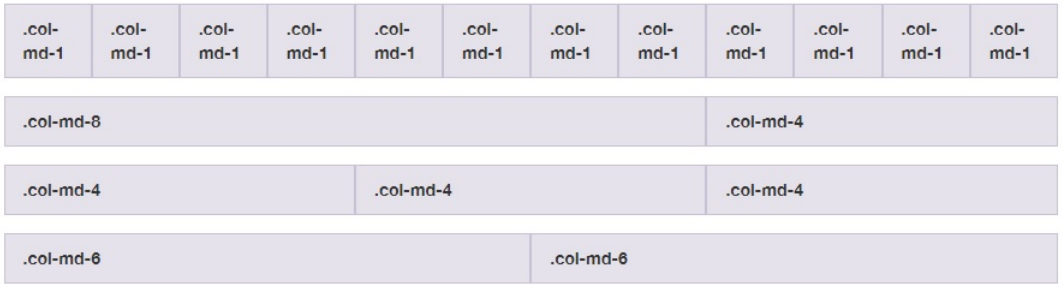
É importante citar que o HTML e o Bootstrap são utilizados apenas para criar divisões de espaço para cada elemento visual e definir a disposição destes na página. Os componentes em si foram desenvolvidos de forma isolada em arquivos JavaScript, porém, estes são executados por meio da *tag <script>* no HTML. Quando ocorre essa execução, cada arquivo JS acionado gera um elemento, definido no seu escopo, o qual é inserido no seu respectivo contêiner visual, criado previamente.

Figura 32 – Referência de layout da interface do *dashboard*.




Fonte: Arquivo pessoal.

Figura 33 – Sistema de *grid* oferecido pelo Bootstrap.




Fonte: Página mdbootsrap.

### 6.3.2 Estilização

Para definir os estilos da interface foi utilizada a linguagem CSS e o *framework* Bootstrap, este último estipula um estilo padrão para o layout que pode ser sobreposto pelas definições do CSS.

O CSS foi escrito em um arquivo isolado do HTML, diferente do Bootstrap, nele são referenciados os elementos do HTML que compõe a página através de suas respectivas classes, *tags* e identificadores (*id*). A figura 34 exibe a referenciação de um elemento do layout e a estilização que lhe foi atribuída. Os atributos do elemento são definidos no interior das chaves, como “font-size” que define o tamanho da fonte em pixels, ou “color’, utilizado para atribuir cores.

Figura 34 – Exemplo de estilização utilizando CSS.

```
156  div#pie-chart-title.type {
157      font-family: 'Roboto', sans-serif;
158      font-size: 30px;
159      margin-bottom: 10%;
160      color: #003763;
161      font-weight: 400;
162      position: relative;
163      top: 35px;
164  }
```

Fonte: Arquivo pessoal.

Esta etapa também envolveu a criação de um design responsivo. A responsividade é uma abordagem de design web que visa adequar a renderização da interface de uma aplicação à resolução da tela ou dispositivo de acesso, assim respondendo ao contexto do usuário. Com isto viabilizou-se o acesso ao *dashboard* através de diferentes dispositivos ou telas.

Por fim, para usufruir dos estilos criados foi necessário empregar no documento HTML a *tag link*, indicando nesta o caminho até o arquivo CSS. Esta *tag* foi declarada no cabeçalho, ou *head* do HTML, onde esta contido todo o conteúdo não exibido pela página, ou seja, apenas metadados do documento.

### 6.3.3 Criação dos elementos visuais

O desenvolvimento dos componentes do *dashboard* foi realizado de forma semelhante à estilização. Cada elemento foi construído de forma isolada e independente dos demais, sendo posteriormente declarados no documento HTML a partir da *tag script*, utilizada para executar arquivos JS.

Foram desenvolvidos os seguintes componentes do painel:

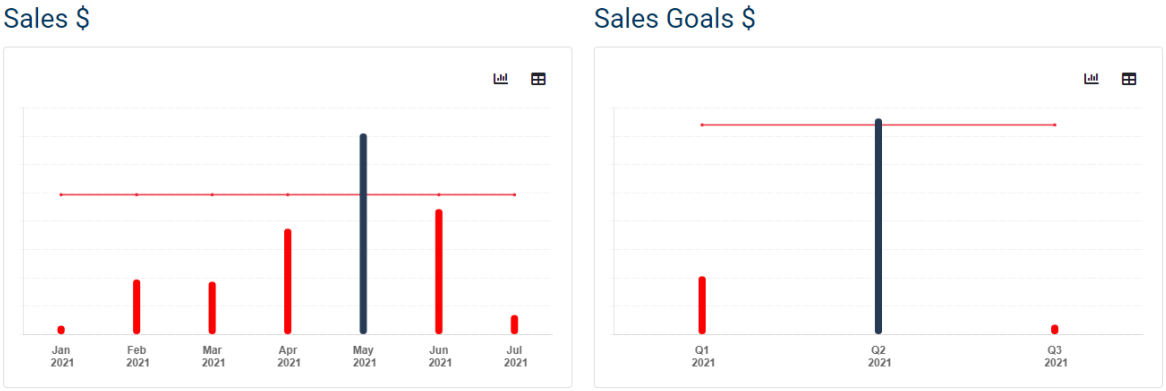
- Indicador de meta trimestral em dólar;
- Indicador de atingimento da meta em porcentagem;
- Indicador de resultado absoluto em dólar, considerando a data selecionada;
- Gráfico de barras de vendas por mês, com opção de visualização em tabela;
- Gráfico de barras de vendas por trimestre, com opção de visualização em tabela;
- Gráfico circular com proporção de vendas por família do produto vendido;
- Tabela de produtos com ranking por valor acumulado de venda;
- Tabela de pedidos, contendo todos os registros e suas respectivas informações relevantes de identificação e status;
- Filtro para seleção de trimestre do ano;

Desses, apenas os indicadores foram criados no mesmo arquivo JavaScript. Os dados utilizados pelos elementos visuais, com exceção do filtro, são obtidos mediante requisições realizadas através dos respectivos *endpoints* das API's, sendo estas realizadas a cada execução ou carregamento da página da aplicação. Foram criadas funções de aquisição para execução das requisições, estas buscam os dados e tem seu retorno armazenado em variáveis, que por sua vez tem seu conteúdo traduzido para o formato JSON. Essa última variável é decomposta em outras variáveis vetoriais, assim, cada dado de um determinado registro é armazenado de forma isolada, contudo, dados de um mesmo registro ocupam a mesma posição nos seus respectivos vetores, permitindo que estes possam ser tratados de forma separada mas viabilizando sua ligação nos elementos mais adiante.

A construção dos gráficos foi realizada mediante o uso da biblioteca Chart.js, esta oferece uma função pronta para criação de gráficos, trazendo diversas opções de personalização. Assim, a partir do uso dessa função desenvolveu-se os gráficos de barras representados na figura 35 e o gráfico de setores da figura 36.

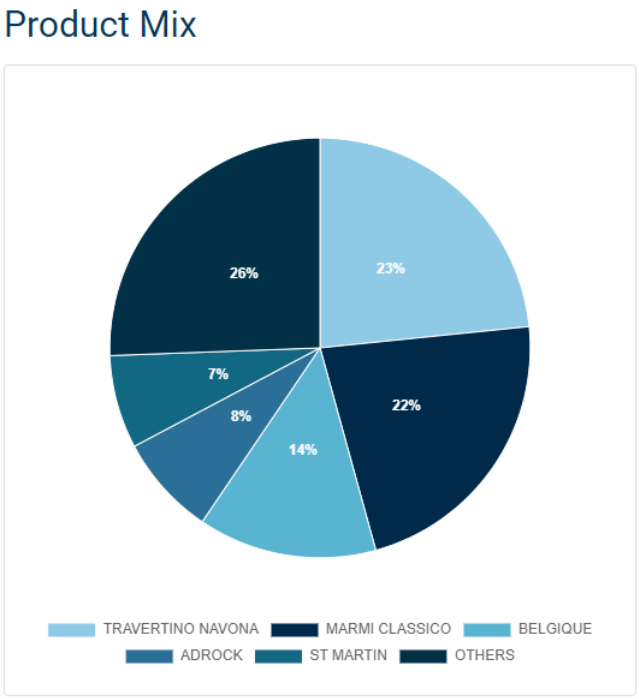
Para viabilizar a atualização dos gráficos de acordo com a mudança do filtro trimestral desenvolveu-se uma lógica de estados onde cada seleção no filtro provoca a limpeza do contêiner HTML que armazena cada dado gráfico, caso este espaço esteja preenchido. Em seguida gera-se uma nova requisição de renderização, a qual contém as informações da data selecionada que devem ser passadas para as requisições realizadas pelos arquivos JS. Assim, estes arquivos são executados recriando os elementos com base nos dados filtrados por data, os quais são gerados através das chamadas realizadas ao banco pelas API's.

Figura 35 – Gráficos de barras desenvolvidos com Chart.js.




Fonte: Arquivo pessoal.

Figura 36 – Gráficos de setores desenvolvido com Chart.js.

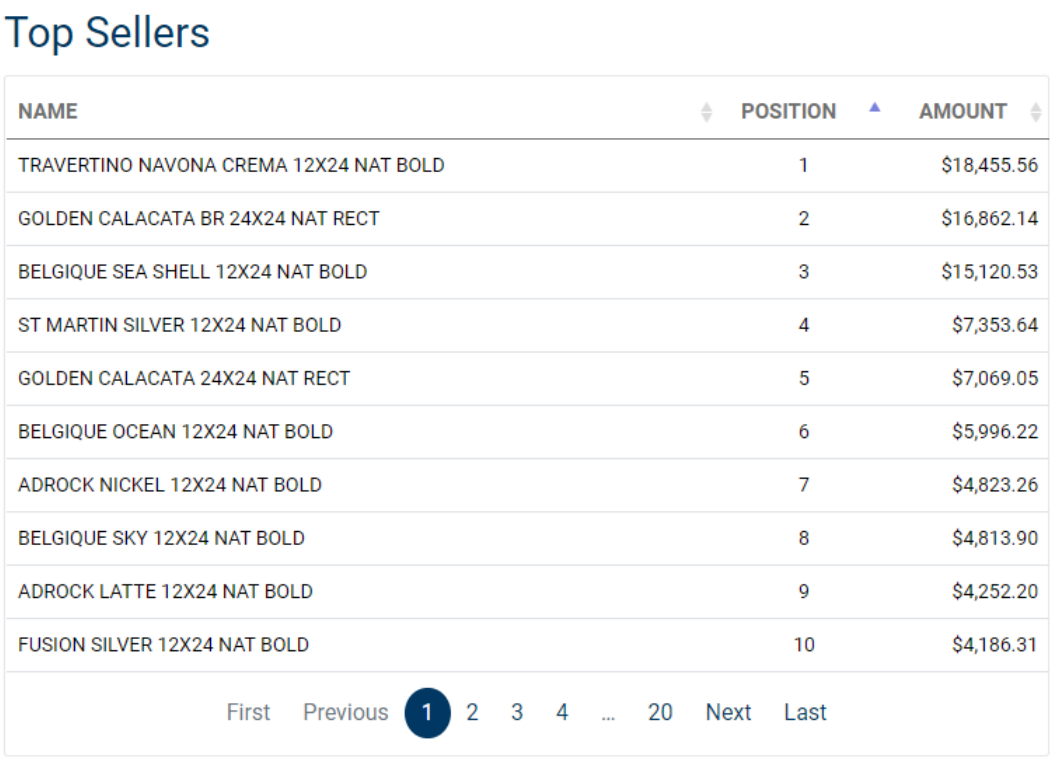



Fonte: Arquivo pessoal.

Para desenvolver as tabelas do painel utilizou-se um *plugin*, ou módulo de extensão, do jQuery chamado DataTables. Este permite organizar e apresentar dados no formato de tabela, oferecendo diversos recursos úteis como pesquisa, paginação, ordenação, filtro, etc. A única dependência deste *plugin* é o próprio jQuery, por isso é necessário instalar a biblioteca no projeto ou declará-la via *Content Delivery Network* (CDN), ou rede de fornecimento de conteúdo, como realizou-se neste projeto. É pre-

ciso ressaltar que as tabelas foram geradas através de funções criadas com JS nas quais definiram-se lógicas para geração dos elementos que compõe a tabela e para a inserção dos dados. Deste modo, o DataTables foi empregado para facilitar o processo desenvolvimento e oferecer ao usuário algumas opções de interação com as tabelas. Assim, criou-se a tabela de produtos mais vendidos e a tabela de pedidos, exibidas nas figuras 37 e 38, respectivamente.

Figura 37 – Tabela de produtos mais vendidos utilizando DataTables.




Fonte: Arquivo pessoal.

A PBA solicitou que os gráficos de barras dispusessem de botões que permitissem transformar o formato da visualização, podendo variar entre o formato de gráfico padrão e o formato de tabela. Para isso, foram desenvolvidos um botão para cada formato. Assim, foi necessário desenvolver uma lógica de manipulação do HTML utilizando JavaScript, a qual foi baseada nos métodos oferecidos pelo DOM, que possibilitam alterar o conteúdo de um contêiner HTML. Essas funções foram programadas para agir quando um dos botões for pressionado. Desta maneira, os botões somente ocasionam modificações quando um modo diferente daquele que está sendo exibido no momento é selecionado. A lógica criada funciona de forma semelhante a atualização provocada pelo filtro, limpando o interior do contêiner e inserindo o novo componente visual. A figura 39 exibe o modo tabela dos gráficos de barras apresentados acima.

Figura 38 – Tabela de pedidos utilizando DataTables.

Orders

| ORDER | INVOICE NUMBER | TERMS | AMOUNT | STATUS |
|---|---|---|---|---|
| 140041 | 16404 | NET 30 DAYS | $764.80 | In Process |
| 140039 | 16403 | NET 30 DAYS | $50.69 | In Process |
| 140037 | 16402 | NET 30 DAYS | $168.30 | In Process |
| 140036 | 16401 | NET 30 DAYS | $183.45 | In Process |
| 140034 | 16400 | NET 30 DAYS | $66.71 | In Process |
| 139935 | 16375 | NET 30 DAYS | $185.85 | In Process |
| 139934 | 16374 | NET 30 DAYS | $480.73 | In Process |
| 139933 | 16373 | NET 30 DAYS | $557.54 | In Process |
| 139932 | 16372 | NET 30 DAYS | $50.69 | In Process |
| 139856 | 16353 | NET 30 DAYS | $101.37 | In Process |

First Previous 1 2 3 4 ... 51 Next Last

Fonte: Arquivo pessoal.

Figura 39 – Modo tabela dos gráficos barras.

Sales $

| Month | Amount | Target | % Results |
|---|---|---|---|
| Apr, 2021 | 37321.54 | 49334.5 | 75.6 % |
| Feb, 2021 | 19374.38 | 49334.5 | 39.3 % |
| Jan, 2021 | 3058.99 | 49334.5 | 6.2 % |
| Jul, 2021 | 6819.19 | 49334.5 | 13.8 % |
| Jun, 2021 | 44254.83 | 49334.5 | 89.7 % |
| Mar, 2021 | 18594.38 | 49334.5 | 37.7 % |
| May, 2021 | 70987.22 | 49334.5 | 143.9 % |

Sales Goals $

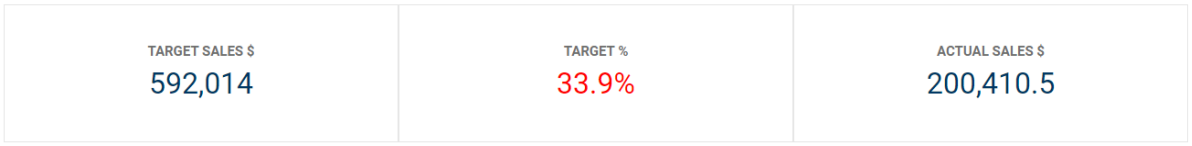
| Quarter | Amount | Target | % Results |
|---|---|---|---|
| Q1, 2021 | 41027.8 | 148003.5 | 27.7 % |
| Q2, 2021 | 152563.6 | 148003.5 | 103.1 % |
| Q3, 2021 | 6819.2 | 148003.5 | 4.6 % |

Fonte: Arquivo pessoal.

Os indicadores, apresentados na figura 40, foram desenvolvidos a partir dos dados recebidos do *back-end*, a requisição realizada para construção destes retorna a informação tratada e pronta para ser apresentada, restando somente adicionar através do HTML e CSS a estilização criada.

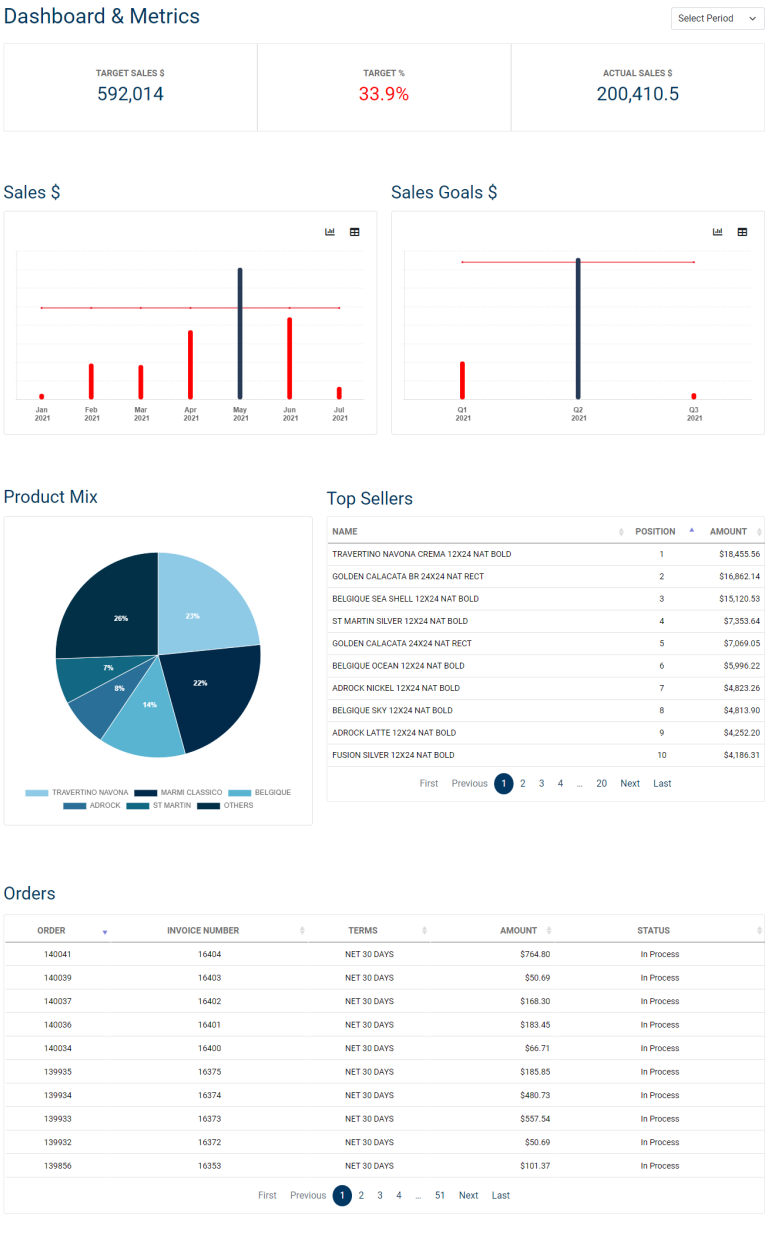
O *dashboard* com todos os elementos é exposto na figura 41, enquanto a mesma versão adaptada para dispositivos com resoluções menores é exposta na figura 42. É importante citar a empresa contratante decidiu retirar do painel os dados relacionados a logística, por isso o modelo final do projeto apresenta algumas diferenças com relação ao layout de referência.

Figura 40 – Indicadores de desempenho desenvolvidos.



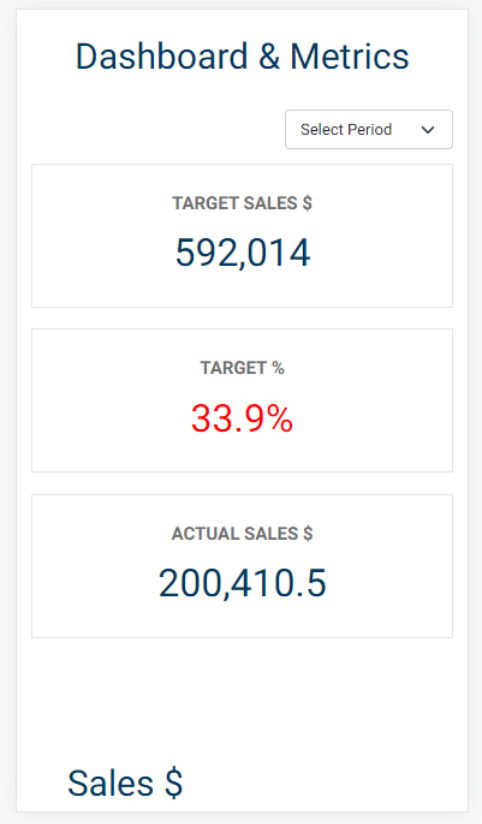

Fonte: Arquivo pessoal.

Figura 41 – Interface do *dashboard* de BI.




Fonte: Arquivo pessoal.

Figura 42 – Interface do *dashboard* de BI em versão *mobile*.




Fonte: Arquivo pessoal.

# 7 RESULTADOS

Neste capítulo abordados os resultados obtidos no projeto desenvolvido. Em primeiro lugar, avalia-se o cumprimento dos requisitos do sistema definidos na seção 4.2. Em seguida, os resultados são analisados, trazendo testes realizados durante o desenvolvimento da aplicação e melhorias implementadas a partir destes. Por último, examina-se a solução levando em consideração as perspectivas de uso da aplicação tanto pela empresa contratante quanto pela empresa BIX Tecnologia, responsável pelo projeto.

## 7.1 CUMPRIMENTO DOS REQUISITOS

Para verificar se os resultados do projeto estão de acordo com as determinações inicias avaliaram-se os requisitos do sistema definidos na fase inicial de desenvolvimento, buscando verificar se estes foram atendidos de forma satisfatória na solução final.

- A atualização da interface ocorre de forma recorrente graças a execução das ETLs no software Pentaho, assim, o banco de dados é atualizado periodicamente, garantindo que a informação acessada é sempre a mais recente;
- O sistema de autenticação implementado promove a redução dos dados no *back-end* da aplicação de acordo com o usuário, uma vez que o ID deste é utilizado como filtro nas extrações de dados, além de ser um parâmetro obrigatório na execução das API's;
- Foram criadas todos os elementos definidos no design da interface proposto pela empresa;
- Realizou-se todo o tratamento de dados no *back-end* da aplicação, separando estes em períodos mensais e trimestrais e agrupado-os segundo seus respectivos intervalos de tempo;
- Realizou-se a manipulação de dados no *front-end* para que estes pudessem ser exibidos nos diversos elementos gráficos;
- Criou-se um filtro trimestral, oferecendo ao usuário a possibilidade de alternar os dados exibidos pelo painel;
- Desenvolveu-se um botão para transformar as visualizações de gráfico de barras em tabelas. Da mesma forma, também foi elaborado um botão para retornar a visualização padrão de gráfico de barras;

- Integrou-se a autenticação do *dashboard* com o login da loja virtual Magento utilizando *tokens* JWT;
- A aplicação foi desenvolvida utilizando tecnologias aceitadas por todos os navegadores atuais e o layout da interface foi adaptado para funcionar em diversas resoluções, não ocasionando problemas de visualização em dispositivos que possuem telas de menor tamanho;
- O desenvolvimento dos componentes da aplicação seguiu um modelo modular, onde cada componente pode ser desacoplado ou substituído facilmente;

## 7.2 ANÁLISE DE RESULTADOS

Ao longo do desenvolvimento da aplicação, realizaram-se diversos testes para verificar a funcionalidade de cada componente desenvolvido. Desta maneira, novos arquivos ou funções somente foram adicionadas ao projeto após a realização de análises do código, as quais foram executadas pelo autor deste documento e pelos colaboradores envolvidos. Assim, a aprovação de novas funções teve como requisito a avaliação de um membro da equipe não envolvido na construção do código tratado. Está prática de revisão é amplamente difundida no setor de desenvolvimento de software, uma vez que promove a redução de falhas e incentiva a escrita de códigos que facilitem a compreensão do leitor.

Assim, com todos os elementos testados individualmente foi possível testar a aplicação como um todo. Primeiramente avaliou-se a responsividade da interface, buscando provocar erros na renderização do painel através da execução deste em diferentes resoluções. Dessa forma, encontraram-se algumas falhas que foram corrigidas mediante ajustes no arquivo de estilos CSS.

Em seguida, testou-se o carregamento da aplicação tanto no ambiente local como em cenários de produção, ou seja, em um ambiente de execução real. Graças a esta análise foram encontrados alguns pontos de melhoria, como a reestruturação das API's de dados, as quais foram modificadas com objetivo de reduzir o volume de consultas realizadas ao banco e otimizar a performance de carregamento.

Por fim, a aplicação foi revisada e testada pelos colabores da equipe de desenvolvimento com o intuito de encontrar falhas não percebidas. Durante esse ciclo de revisão não foram encontrados erros significativos, no entanto, foram feitas diversas sugestões com relação a detalhes da interface. Com isso aprimorou-se a primeira versão do projeto, atingindo um nível satisfatório e adequado para a primeira entrega ao cliente.

## 7.3 AVALIAÇÃO DA SOLUÇÃO

Com relação a solução desenvolvida, acredita-se que todos os requisitos propostos inicialmente foram alcançados e que os casos de uso foram atendidos. Todas as considerações com respeito a integridade do sistema foram positivas, os testes realizados sobre a versão final da interface não expuseram falhas no layout responsivo, indicando que o painel funciona conforme esperado em diferentes resoluções. O tempo de carregamento da página da aplicação também foi examinado com intuito de identificar possíveis gargalos no *back-end*, o resultado dessas análises proporcionou melhorias na estrutura de dados, aprimorando o protótipo final.

Assim, concluiu-se que a primeira versão da aplicação atendeu as expectativas inicias da contratante, contudo, entende-se que a solução entregue deve passar por mais testes no cenário real de uso e receber novas avaliações por parte da PBA antes que o painel possa ser integrado à loja virtual, podendo ocorrer múltiplos ciclos de revisão e implementação de melhorias.

A solução desenvolvida foi amplamente discutida entre gestores e colaboradores envolvidos no projeto. Dessa forma, entendeu-se que a aplicação possui grande potencial para se tornar um *template* para projetos de BI, pois esta viabiliza um alto grau de personalização do *dashboard* e suprime a necessidade de adquirir múltiplas licenças de uso para atender todos os usuários. Com a evolução do projeto, a BIX Tecnologia visa ter um novo recurso a oferecer dentro do seu conjunto de soluções.

# 8 CONCLUSÃO

O objetivo deste trabalho foi o desenvolvimento de uma solução de BI, cuja finalidade é ser integrada a um *e-commerce* para oferecer informação aos clientes sobre seus pedidos e metas relacionadas. Utilizando uma abordagem de desenvolvimento ágil de software o projeto progrediu de forma eficiente desde sua concepção, partindo de um modelo conceitual até alcançar um protótipo funcional testado em condições reais. A aplicação foi desenvolvida utilizando tecnologias atuais, além disso foram implementadas diversas integrações entre sistemas e serviços, buscando entregar ao cliente final uma solução completa e moderna. Optou-se por utilizar algumas das linguagens de programação mais conhecidas e difundidas, como Python e JavaScript. Dessa forma, apesar dos diversos desafios enfrentados durante a construção da aplicação, a disponibilidade de conteúdo on-line sobre as tecnologias utilizadas simplificou a resolução de todos os problemas encontrados.

A versão da aplicação exposta neste documento se encontra na fase piloto. O painel é capaz de exibir dados de cada usuário a partir dos respectivos IDs, caso estes estejam mapeados no banco de dados. A empresa Portobello America verificou o funcionamento da integração entre sua loja virtual na plataforma Magento e a solução desenvolvida, obtendo resultados satisfatórios. A responsividade do *dashboard* não necessitou de ajustes, visto que a loja virtual oferece uma seção própria para o painel, permitindo que este seja renderizado na sua resolução máxima. Além disso, todas as funções foram testadas antes da primeira entrega para garantir a integridade e o bom funcionamento da aplicação.

Atualmente, o painel está em fase de testes, sendo avaliado pela contratante. Assim, são esperadas diversas sugestões de melhorias, principalmente no âmbito da interface. Entende-se que projetos com escopo de negócio, como o tratado neste documento, costumam passar por diversos ciclos de desenvolvimento, pois é comum surgirem novas ideias durante o processo de criação, assim como através de *feedbacks* em reuniões de alinhamento com o cliente. Contudo, a avaliação recebida até o momento foi positiva, validando que o projeto atendeu a necessidade levantada inicialmente. No entanto, a Portobello solicitou que novas funcionalidades sejam adicionadas à aplicação antes de que esta seja disponibilizada na loja virtual da PBA. O objetivo desta proposta em negociação é entregar mais recursos aos clientes da empresa, trazendo mais informações sobre os pedidos realizados. Além disso, a contratante demonstrou interesse na criação de uma versão administrativa do painel para uso próprio, visto que a versão atual do painel teve grande aceitação na área gerencial envolvida. Por isso, esperam-se novas etapas de desenvolvimento após o fim das negociações em andamento entre a BIX e a Portobello.

Um dos maiores desafios para a conclusão deste Projeto de Fim de Curso foi

o trabalho remoto, uma vez que a empresa na qual realizou-se o trabalho adotou medidas restritivas devido à pandemia de Covid-19. Apesar dessa situação, a equipe responsável pelo projeto conseguiu manter a comunicação através de plataformas de colaboração on-line, realizando reuniões semanais e diárias durante o andamento do projeto. Dessa forma, a execução das atividades não foi prejudicada.

Por fim, acredita-se que o projeto foi de suma importância para o desenvolvimento do autor, pois no decorrer do do trabalho foram adquiridas diversas competências e conhecimentos dentro da área de desenvolvimento de sistemas e negócios, como: computação em nuvem, manipulação de dados, desenvolvimento *back-end* e *front-end*, etc. Assim, entende-se que a experiência adquirida enriqueceu a formação do aluno.

## 8.1 TRABALHOS FUTUROS

Os *feedbacks* e as discussões realizadas ao longo do desenvolvimento do projeto foram essenciais para o entendimento do futuro da aplicação. Levando isso em consideração, acredita-se que os seguintes trabalhos sejam relevantes para a evolução da solução de BI apresentada:

- Implementar uma versão administrativa do painel, voltada a atender gestores, oferecendo a possibilidade de filtrar clientes e analisar suas informações;
- Modificar a infraestrutura do *front-end* para um modelo mais escalável e moderno, utilizando bibliotecas como React do JavaScript;
- Adicionar mais elementos ao painel, como um modal para visualização de detalhes dos pedidos listados na tabela relacionada;
- Generalização do *back-end* da aplicação, visando a reutilização das API's em outros projetos, apenas alterando as conexões de dados;
- Implementação de um sistema de login próprio para registrar usuários, definir grupos de acesso e viabilizar a criação de visualizações diferenciadas por usuário;
- Desenvolver um *template* ou *framework* para projetos de BI, contendo *front-end* e *back-end* estruturados;
- Desenvolver um aplicativo mobile próprio para a aplicação, utilizando tecnologias como React Native;

# REFERÊNCIAS

ALMEIDA, Maria Sueli; ISHIKAWA, Missao; REINSCHMIDT, Joerg; ROEBER, Torsten. **Getting Started with Data Warehouse and Business Intelligence**. 1. ed. California: IBM Corporation, ago. 1999.

ANDRADE, Ana Paula de. **O que é o SQLAlchemy?** [*S.l.*: *s.n.*], 2019. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-o-sqlalchemy. Acesso em: 8 ago. 2021.

BOSCARINO, Aylan. **Como o JWT funciona**. [*S.l.*: *s.n.*], 2019. Disponível em: https://www.devmedia.com.br/como-o-jwt-funciona/40265. Acesso em: 9 ago. 2021.

CANGUÇU, Raphael. **O que são Requisitos Funcionais e Requisitos Não Funcionais?** [*S.l.*: *s.n.*], fev. 2021. Disponível em: https://codificar.com.br/requisitos-funcionais-nao-funcionais/. Acesso em: 10 ago. 2021.

CESARIO, Daniel; COSTA, Marcelo. **Pentaho Data Integration - ETL em Software Livre**. [*S.l.*: *s.n.*], nov. 2017. Disponível em: https://www.infoq.com/br/articles/pentaho-pdi/. Acesso em: 8 ago. 2021.

CETAX. **O que é ETL – Extract Transform Load?** [*S.l.*: *s.n.*], set. 2020. Disponível em: https://www.cetax.com.br/blog/etl-extract-transform-load/. Acesso em: 14 set. 2021.

CHACON, Scott; STRAUB, Ben. **Pro Git**. 2. ed. [*S.l.*]: Apress, nov. 2014. cap. 6. Disponível em: https://git-scm.com/book/pt-br/v2. Acesso em: 8 ago. 2021.

CURTIS, Susan. **What Is HTML?** Hypertext Markup Language Basics Explained. [*S.l.*: *s.n.*], jun. 2021. Disponível em: https://www.hostinger.com/tutorials/what-is-html. Acesso em: 3 ago. 2021.

ESPINHA, Roberto G. **Kanban:** O que é e TUDO sobre como gerenciar fluxos de trabalho. [*S.l.*: *s.n.*]. Disponível em: https://artia.com/kanban/. Acesso em: 2 ago. 2021.

GOMES, Pedro César Tebaldi. **Métricas e indicadores de desempenho, utilizando a favor do seu negócio**. [*S.l.*: *s.n.*], ago. 2016. Disponível em: https://www.opservices.com.br/metricas-e-indicadores-de-desempenho/. Acesso em: 7 set. 2021.

HORN, Michelle. **Bootstrap:** o que é, como usar e para que serve esse framework? [*S.l.*: *s.n.*], jan. 2021a. Disponível em: https://blog.betrybe.com/framework-de-programacao/o-que-e-bootstrap/. Acesso em: 4 ago. 2021.

HORN, Michelle. **Python:** guia pra iniciantes na linguagem [básico ao avançado]. [*S.l.*: *s.n.*], 2021b. Disponível em: https://blog.betrybe.com/python/#1. Acesso em: 5 ago. 2021.

KIMBALL, Ralph; ROSS, Margy. **The Data Warehouse Toolkit**: The Definitive Guide to Dimensional Modeling. 3. ed. Indianapolis: Wiley, 2013.

LABERGE, Laura; O'TOOLE, Clayton; SCHNEIDER, Jeremy; SMAJE, Kate. **How COVID-19 has pushed companies over the technology tipping point and transformed business forever**. [*S.l.*: *s.n.*], out. 2020. Disponível em: https://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/how-covid-19-has-pushed-companies-over-the-technology-tipping-point-and-transformed-business-forever#. Acesso em: 31 jul. 2021.

LAUBE, Klaus Peter. **Do WSGI ao ASGI - Parte 1**. [*S.l.*: *s.n.*], mar. 2021. Disponível em: https://klauslaube.com.br/2021/03/01/do-wsgi-ao-asgi-parte-1.html. Acesso em: 6 ago. 2021.

LUCIDCHART. **Diagrama de caso de uso UML:** O que é, como fazer e exemplos. [*S.l.*: *s.n.*]. Disponível em: https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml/#section_0. Acesso em: 10 ago. 2021.

MALDONADO, Leonardo. **Entendendo o DOM (Document Object Model)**. [*S.l.*: *s.n.*], fev. 2018. Disponível em: https://tableless.com.br/entendendo-o-dom-document-object-model/. Acesso em: 3 ago. 2021.

MOZILLA DEVELOPER NETWORK. **Iniciando com HTML**. [*S.l.*: *s.n.*], ago. 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/HTML/Introduction_to_HTML/Getting_started. Acesso em: 3 ago. 2021.

MOZILLA DEVELOPER NETWORK. **O que é JavaScript?** [*S.l.*: *s.n.*], ago. 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Acesso em: 4 ago. 2021.

ORACLE. **O que é Big Data?** [*S.l.*: *s.n.*]. Disponível em: https://www.oracle.com/br/big-data/what-is-big-data/. Acesso em: 31 jul. 2021.

ORACLE. **O Que É um Banco de Dados Relacional?** [*S.l.*: *s.n.*]. Disponível em: https://www.oracle.com/br/database/what-is-a-relational-database/. Acesso em: 9 ago. 2021.

ORACLE. **O Que É um Banco de Dados?** [*S.l.*: *s.n.*]. Disponível em: https://www.oracle.com/br/database/what-is-database/. Acesso em: 9 ago. 2021.

RABELO, Ricardo José. **Indicadores de desempenho**. [*S.l.*: *s.n.*], 2019. 218 slides. Material apresentado para a disciplina de Avaliação de Desempenho de Sistemas de Automação Discretas no curso de Engenharia de Controle e Automação da UFSC. Acesso em: 7 set. 2021.

RAMÍREZ, Sebastián. **FastAPI**. [*S.l.*: *s.n.*]. Disponível em: https://fastapi.tiangolo.com/. Acesso em: 5 ago. 2021.

ROCHA, Helder da. **Learn Chart.js**. 1. ed. Birmingham: Packt Publishing., fev. 2019. Disponível em: https://books.google.com.br/books?id=NHCLDwAAQBAJ&printsec=frontcover&hl=pt-BR&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=true. Acesso em: 7 ago. 2021.

RÜCKERT, Ariane. **O que é CSS?** Guia Básico para Iniciantes. [*S.l.*: *s.n.*], jul. 2021. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css. Acesso em: 3 ago. 2021.

SAS. **ETL**: O que é e qual sua importância? [*S.l.*: *s.n.*]. Disponível em: https://www.sas.com/pt_br/insights/data-management/o-que-e-etl.html. Acesso em: 7 set. 2021.

SHAJI, Amal. **Parallelism, Concurrency, and AsyncIO in Python - by example**. [*S.l.*: *s.n.*], dez. 2020. Disponível em: https://testdriven.io/blog/python-concurrency-parallelism/. Acesso em: 6 ago. 2021.

SILVA, Douglas da. **O que é ETL?** [*S.l.*: *s.n.*], jan. 2021. Disponível em: https://www.zendesk.com.br/blog/o-que-e-etl/. Acesso em: 7 set. 2021.

TEIXEIRA, José Ricardo. **jQuery Tutorial**. [*S.l.*: *s.n.*], 2013. Disponível em: https://www.devmedia.com.br/jquery-tutorial/27299. Acesso em: 7 ago. 2021.