INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAJAZEIRAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

AUDIF: UM SISTEMA PARA A REALIZAÇÃO DE AUDITORIA FINANCEIRA EM MICRO E PEQUENAS EMPRESAS

JAMES PEREIRA DOS SANTOS AMARANTE

CAJAZEIRAS

2026

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

JAMES PEREIRA DOS SANTOS AMARANTE

AUDIF: UM SISTEMA PARA REALIZAÇÃO DE AUDITORIA FINANCEIRA EM MICRO E PEQUENAS EMPRESAS

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Dr. Fábio Gomes de Andrade

Coorientadora

Profa. Dra. Gabriela Guedes de Souza

Aprovada em: **13 de fevereiro de 2026.**

Prof. Dr. Fábio Gomes de Andrade - Orientador

Profa. Dra. Gabriela Guedes de Souza - Coorientadora

Prof. Me. Francisco Paulo de Freitas Neto - Avaliador

IFPB - Campus Cajazeiras

Prof. Me. Janderson Ferreira Dutra - Avaliador

IFPB - Campus Cajazeiras

Documento assinado eletronicamente por:

- **Fabio Abrantes Diniz**, **COORDENADOR(A) DE CURSOS - FUC1 - UNINFO-CZ**, em 25/02/2026 09:06:24.
- **Fabio Gomes de Andrade**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 25/02/2026 09:10:09.
- **Francisco Paulo de Freitas Neto**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 25/02/2026 09:14:49.
- **Janderson Ferreira Dutra**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 25/02/2026 09:19:14.
- **Gabriela Guedes de Souza**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 25/02/2026 10:17:48.

Este documento foi emitido pelo SUAP em 25/02/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código 840055
Verificador: 657098cd7e
Código de Autenticação:

IFPB / Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Cícero Luciano Félix CRB-15/750

A485a Amarante, James Pereira dos Santos.

Audif : um sistema para a realização de auditoria Financeira em micro e pequenas empresas / James Pereira dos Santos Amarante.– 2026.

107f. : il.

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2026.

Orientador(a): Prof. Dr. Fabio Gomes de Andrade.
Coorientador(a): Profª. Dra. Gabriela Guedes de Souza.

1. Desenvolvimento de sistemas. 2. Sistema de *Business Intelligence*. 3. Auditoria financeira. 4. Pequenas empresas. I. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. II. Título.

IFPB/CZ

CDU: 004.4:336.11(043.2)

JAMES PEREIRA DOS SANTOS AMARANTE

AUDIF: UM SISTEMA PARA A REALIZAÇÃO DE AUDITORIA FINANCEIRA EM MICRO E PEQUENAS EMPRESAS

Trabalho de Conclusão de Curso submetido à coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como requisito final para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Dr. Fabio Gomes de Andrade

Coorientadora: Drª. Gabriela Guedes de Souza

Cajazeiras - PB

2026

Para a minha família, que desempenhou um papel central nesta jornada.

AGRADECIMENTOS

A **Deus**, todo poderoso e misericordioso, por me permitir chegar ao fim deste curso, dando a coragem e força necessárias.

Aos meus orientadores, **Fabio Gomes de Andrade** - Fabio Pai -, e **Gabriela Guedes de Souza**, que foram pacientes, me incentivaram a todo o momento e acreditaram em mim, quando nem mesmo eu acreditava.

Ao Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, *campus Cajazeiras*, por me permitir conhecer pessoas incríveis e proporcionar um ótimo ambiente de estudos.

À minha mãe, *Irene Pereira de Souza* pois, se hoje estou aqui, foi graças ao que a senhora fez, me ensinando que a educação era o meio para conseguir realizar os meus sonhos. Ao meu pai, *José Francisco dos Santos Amarante*, que esteve comigo durante todo este tempo e me incentivava sempre ao *"seu modo"*.

Às minhas irmãs, **Camila Pereira dos Santos**, **Juliana Pereira Amarante** e **Lais Pereira dos Santos Amarante**, que fizeram tudo o que estava ao seu alcance para que eu pudesse estar aqui neste momento. Vocês são as melhores irmãs que alguém poderia ter!

À minha amada tia, **Maria de Souza** (*In Memorian*), a senhora pode não estar presente, mas teve um papel importante para que eu chegasse aqui.

Aos meus colegas de curso, em especial: **Antoniel Damião**, **Diones Gomes**, **Francisco Iarlyson**, **Gustavo Araújo**, **Kelcilene Parnaíba** e **Luz Pereira**, espero que se lembrem das noites não dormidas, das longas horas de conversa, dos momentos de raiva e dos incentivos que me deram. VOCÊS SÃO DEMAIS! Foi uma honra ter estudado e passado parte do meu tempo com vocês.

Aos amigos que fiz no IFPB e espero levar para a vida: **Alan Max**, **Andreza Lopes**, **João Emanuel**, **Suenia Carolino**, **Marinalva Vieira** e **Amanda Sousa**. Agradeço por suportarem minhas loucuras - me trazendo a razão, quando necessário -, e pelas partidas intermináveis de *UNO* na fila do almoço. Não poderia deixar de agradecer a **Edwiges Souza** e **Iriany Martins**, pelas longas conversas noturnas via *Meet*, me ajudando a quebrar a rotina exaustiva de estudos. VOCÊS NÃO EXISTEM.

E, por fim, MUITO OBRIGADO! A todos os meus familiares, amigos, colegas de curso, professores e funcionários da instituição, que tive a oportunidade de ter e conhecer ao longo deste "passeio". Tenham certeza! Tentei ao máximo aproveitar e aprender cada lição, ensinamento e conselho que me foi passado.

“O modo como você reúne, administra e usa a informação determina se vencerá ou perderá.”

William Gates

RESUMO

No mundo corporativo, o desempenho financeiro de uma empresa é tido como um selo de sucesso no mercado. Para que um empreendimento chegue a apresentar bons resultados em seus balanços é necessário um controle minucioso das contas, buscando-se sempre monitorar com o que o dinheiro foi gasto e qual a motivação. Para negócios de médio e grande porte, uma solução existente, e bastante usada é a contratação de empresas externas especializadas na auditoria de contas. Todavia, para as micro e pequenas empresas a obtenção de tais serviços é uma realidade bastante distante da vivida pelas médias e grandes, ao levar em conta o valor cobrado pela prestação destes serviços. Com o intuito de resolver esta situação e possibilitar a auditoria financeira a esses empreendimentos, este trabalho propõe um sistema de *Business Intelligence*, baseado em um *Data Warehouse*, que permita a estas empresas auditar suas contas, possibilitando um acompanhamento constante, bem como a geração de um relatório descritivo.

Palavras-chave: Auditoria Financeira. Micro e Pequenas Empresas. *Business Intelligence*. *Data Warehouse*.

ABSTRACT

In the corporate world, the financial performance of a company is seen as a seal of success in the market. In order for an enterprise to achieve good results on its balance sheets, it is necessary to have meticulous control of the accounts, always seeking to monitor what the money was spent on and what the motivation was. For medium and large businesses, an existing and widely used solution is to hire external companies specialized in auditing accounts. However, for micro and small companies, obtaining such services is a reality quite distant from that experienced by medium and large companies, when taking into account the amount charged for the provision of these services. In order to resolve this situation and enable financial auditing of these enterprises, this work proposes a Business Intelligence system, based on a Data Warehouse, which allows these companies to audit their accounts, enabling constant monitoring, as well as the generation of a descriptive report.

Keywords: Financial Audit. Micro and Small Businesses. Business Intelligence. Data Warehouse.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

ACID	*Atomicity Consistency Isolation Durability*
API	*Application Programming Interface*
BASE	*Basically Available-Soft state-Eventual consistency*
BI	*Business Intelligence*
BSON	*Binary JSON*
DAO	*Data Access Object*
DER	Diagrama Entidade-Relacionamento
DW	*Data Warehouse*
ETL	*Extract-Transform-Load*
GRAPHQL	*Graph Query Language*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
MEI	Micro Empreendedor Individual
MPE	Micro e Pequenas Empresas
NOSQL	*Not Only SQL*
OLTP	*On-Line Transaction Processing*
PDF	*Portable Document Format*
PIB	Produto Interno Bruto
RDBMS	*Relational Database Management System*
REST	*Representational State Transfer*
RF	Requisito Funcional
RNA	Redes Neurais Artificiais
RNF	Requisito Não Funcional
SGBD	Sistema Gerenciador de Banco de Dados
SOAP	*Simple Object Access Protocol*
SOM	*Self-Organizing Maps*
TCC	*Trabalho de Conclusão de Curso*
UC	*Use Case*
UUID	*Universally Unique Identifier*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o Brasil vem registrando um aumento significativo no número de empresas que são abertas todos os dias. De acordo com o Governo Federal (2025), através do Boletim Anual do Mapa de Empresas (disponibilizado pelo Ministério do Empreendedorismo, da Microempresa e da Empresa de Pequeno Porte), só em 2024 foram abertos em torno de 4.254.903 novos negócios no país, representando um crescimento acumulado de 31,47% entre os períodos de 2020 a 2024.

Em contrapartida, o mesmo boletim aponta que 2.436.190 empreendimentos encerraram suas operações de forma definitiva. Vários são os fatores que podem levar uma empresa a encerrar as suas atividades. Dentre muitos dos motivos que podem estar por trás ou até mesmo, que ajudam a explicar esses números, estão a falta de conhecimento do mercado, o negligenciamento do planejamento estratégico, a escassez de investimentos por parte da própria empresa, gastos e custos muitas vezes desnecessários, além da falta de acompanhamento das finanças (EXPERIAN, 2020).

As Micro e Pequenas Empresas (MPEs) representam hoje 99% dos negócios presentes no país, bem como são responsáveis por 30% do Produto Interno Bruto (PIB) produzido e por 55% dos empregos formais (MERCANTIL, 2024) - sendo esses um dos tipos de negócios que mais foram abertos em 2024.

Como forma de manter continuamente as contas e a situação fiscal da empresa em ordem, a fim de evitar alguns dos problemas elencados anteriormente, os donos desses empreendimentos costumam contratar serviços especializados para auxiliá-los, já que, muitas vezes, não possuem conhecimento na área contábil e tributária. Além disso, a contratação de tais serviços faz com que o empreendedor não tenha que lidar diretamente com os assuntos dessa ordem, permitindo que ele volte a sua atenção a outros setores do negócio.

Mesmo que uma empresa tenha as contas e a situação fiscal regularizadas, isso não é uma garantia de que futuramente ela não venha a enfrentar problemas em outras áreas. Assim, por possuírem mais recursos e terem uma abrangência maior no mercado do que as MPEs, as Médias e Grandes Empresas acabam tendo o costume de contratar firmas especializadas para a realização de auditorias financeiras, com a finalidade de encontrar gargalos presentes na companhia, verificar a existência de fraudes, e possibilitar um olhar mais amplo e detalhado das finanças.

> (…) auditoria não é um jogo e, em tese, os objetivos do auditor e do auditado devem ser os mesmos: dar o máximo de **transparência** e **credibilidade** às informações financeiras, permitindo, assim, melhores decisões e alocação mais eficiente e eficaz de recursos. (COSTA; DUTRA, 2014, p. 55)

Como citado por Costa e Dutra (2014), a transparência e a clareza dos dados da empresa é o objetivo central ao se realizar a auditoria financeira do negócio, possibilitando a melhora do empreendimento, além de maximizar os ganhos e alocar os recursos da forma mais eficiente possível.

1.1 Motivação

Ao levar em consideração os custos elevados para a contratação de empresas auditoras, algumas MPEs podem acabar optando por não contratar esse tipo de serviço, já que muitas vezes não dispõem de recursos financeiros para fazer essa aquisição. Tal fator, entretanto, não inviabiliza a análise das contas de uma empresa, já que a documentação necessária para a inspeção encontra-se à sua disposição. Todavia, o inspecionamento de um volume tão grande de dados pode vir a se tornar um desafio para os empreendimentos, devido à necessidade de alocação de pessoas que tenham algum conhecimento prévio sobre as contas da empresa, bem como o tempo que será gasto para a realização de tal tarefa.

Com o surgimento de tecnologias *open source*, que são capazes de efetuar a leitura e interpretação de dados, acaba se tornando possível oferecer uma alternativa viável e acessível para que as Micro e Pequenas Empresas possam realizar o processo de auditoria interna, sem que para isso precisem desembolsar altas quantias em dinheiro.

Como forma de solucionar esse problema, este Trabalho de Conclusão de Curso (TCC) propõe a criação de um sistema de *Business Intelligence* (*BI*) apoiado em um *Data Warehouse* (*DW*) e uma Rede Neural Artificial (RNA) que permita a leitura e análise das informações financeiras de uma empresa. Como estudo de caso, a solução proposta neste trabalho foi utilizada dentro da empresa Indústrias de Churrasqueiras Fortaleza.

1.2 Objetivos

Esta seção descreve os objetivos a serem alcançados a partir do desenvolvimento deste TCC.

1.2.1 Objetivo Geral

Desenvolver um sistema de *Business Intelligence* de baixo custo, que permita às Micro e Pequenas Empresas o acompanhamento e auditoria das suas contas.

1.2.2 Objetivos Específicos

O trabalho proposto tem ainda como objetivos específicos:

- Criar uma base de dados para armazenar os dados transacionais da empresa;
- Projetar e desenvolver um *Data Warehouse* capaz de integrar e centralizar as informações financeiras do negócio, de forma a possibilitar a auditoria de suas contas;
- Desenvolver mecanismos de importação, tratamento e validação dos tipos de dados financeiros da empresa;
- Implementar recursos de *BI* que possibilitem geração de relatórios, indicadores e análises das receitas e despesas do empreendimento.

1.3 Trabalhos Relacionados

O sistema proposto neste TCC tem relação com alguns sistemas já disponíveis no mercado. Um desses *softwares* é o ***ANALYSIS***[1], que consiste em um programa proprietário que permite a análise minuciosa dos balanços de uma empresa, com a explicação do significado de cada indicador avaliado. O sistema também permite ao usuário avaliar os resultados de forma resumida para a obtenção do máximo de informações acerca do desempenho da empresa. Ele também possibilita a comparação dos anos sequenciais e não sequenciais, e analisa o ano corrente de forma parcial. Mesmo fornecendo algumas funcionalidades que permitem a análise dos dados financeiros, os resultados gerados por esse *software* acabam ficando muito restritos ao balanço da empresa. Além disso, ele não realiza a detecção de discrepâncias em seus balancetes.

O ***STRATWs ONE***[2] é um *software* de gestão de performance corporativa que permite a análise dos dados de uma empresa, a fim de torná-los transparentes, encontrar oportunidades de melhoria, e potencializar os resultados do negócio. Ademais, ele possibilita acompanhar a situação atual e as metas a serem alcançadas. Embora o programa realize tais tarefas, ele ainda não apresenta a funcionalidade de detecção de anomalias e discrepâncias nos dados financeiros.

[1] ANALYSIS. https://codevision.com.br/analysis.html. Acesso em: 25 mar. 2023
[2] STRATWs ONE. https://www.siteware.com.br/nossas-solucoes/. Acesso em: 25 mar. 2023

É importante ressaltar que tanto o ***ANALYSIS*** quanto o ***STRATWs ONE*** são ferramentas proprietárias. Isso acaba implicando em custos relacionados a aquisição, licenciamento periódico e a venda de módulos extras ao *software*, o que pode gerar um custo final elevado para as Micro e Pequenas Empresas.

1.4 Metodologia

Com o objetivo de organizar e sistematizar o desenvolvimento do Trabalho de Conclusão de Curso, optou-se por dividi-lo em várias atividades, dispostas da seguinte forma:

- **Estudo sobre o estado da arte (A1):** esta etapa consistiu no estudo mais aprofundado sobre as tecnologias que seriam usadas no trabalho, bem como o levantamento dos trabalhos relacionados já existentes;
- **Modelagem da base de dados *On-Line Transaction Processing* (A2):** esta etapa consistiu na elaboração e criação da base de dados *OLTP* utilizada pela aplicação para a armazenagem dos dados transacionais da empresa;
- **Construção de uma *API On-Line Transaction Processing* (A3):** nesta etapa foi realizada a implementação de uma *API OLTP*, que tem por finalidade o gerenciamento das informações operacionais da empresa, considerando tanto a matriz e quanto às suas filiais;
- **Modelagem do *Data Warehouse* (A4):** esta etapa tratou da elaboração e criação do *data warehouse*, usado para a armazenagem dos dados financeiros do empreendimento;
- **Construção de uma *API RESTful* para o gerenciamento dos dados financeiro (A5):** esta etapa consistiu na criação de uma *API RESTful*, responsável pelo gerenciamento e controle dos dados financeiros com alguns módulos compõem a solução desenvolvida;
- **Construção do módulo de tratamento dos dados (A6):** nesta etapa foi realizado o desenvolvimento do módulo *ETL* (*Extract-Transform-Load*), responsável pela extração, tratamento e carga dos dados que são armazenados no *data warehouse*;
- **Construção do módulo de leitura e escrita de dados (A7):** nesta etapa foi realizado o desenvolvimento dos serviços responsáveis por recuperar e armazenar dados no *DW*;
- **Construção do módulo para detecção de anomalias financeiras (A8):** nesta etapa foi implementado o módulo responsável pela análise dos dados existentes

no *data warehouse*, a fim de localizar discrepâncias e possíveis anomalias nos balanços financeiros existentes;

- **Desenvolvimento da interface *web* (A9):** nesta etapa ocorreu o desenvolvimento de uma interface *web* - que utiliza-se das *APIs* implementadas - para o envio, visualização, análise e gerenciamento dos dados;
- **Elaboração da documentação final do TCC (A10):** nesta etapa foi elaborado o documento final, realizado ao longo de todo o desenvolvimento deste trabalho.

1.5 Organização do Documento

O restante deste documento está organizado da seguinte forma. O Capítulo 2 aborda a fundamentação teórica do projeto, apresentando conceitos e algumas das tecnologias utilizadas para o seu desenvolvimento. O Capítulo 3, por sua vez, oferece uma descrição detalhada da solução proposta, descrevendo os processos de análise e levantamento de requisitos, o projeto arquitetural, as atividades do sistema, os modelos lógicos das bases de dados e a implementação dos módulos do sistema. Por fim, o Capítulo 4 apresenta as considerações finais sobre o trabalho e as próximas etapas.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os principais conceitos e tecnologias usados no desenvolvimento deste trabalho. As suas seções abordam os seguintes temas: *BI*, *On-Line Transaction Processing* (*OLTP*), *DW*, Ferramenta *ETL*, RNAs, *NoSQL*, *RESTful*, *GraphQL* e *React*.

2.1 *Business Intelligence*

O uso do termo *Business Intelligence* (Inteligência de Negócios ou *BI*), faz referência ao conjunto de conceitos e metodologias que têm como principal objetivo auxiliar empresas na tomada de decisões, definindo os rumos a serem seguidos, com base em dados obtidos no decorrer de um determinado período (NEGASH, 2004).

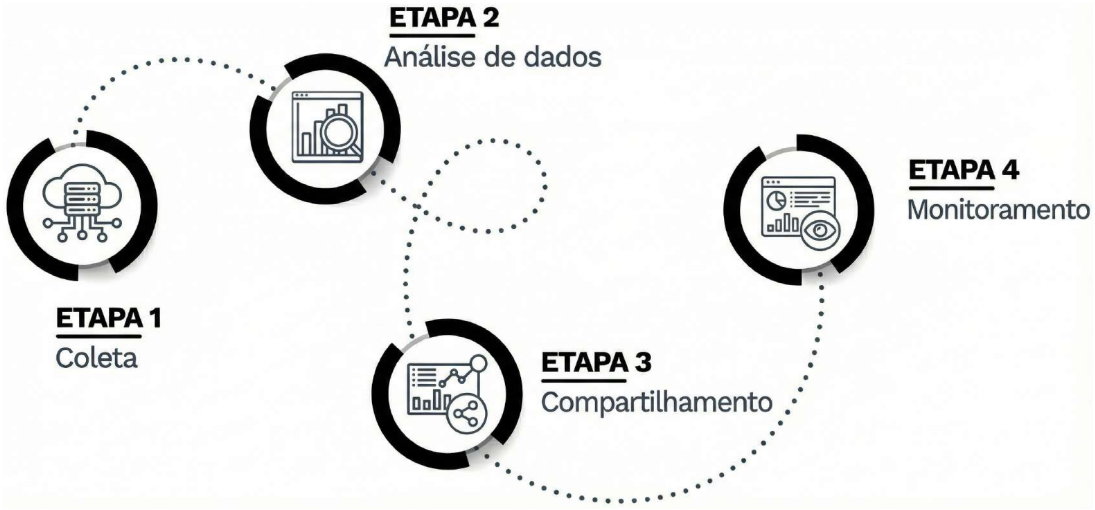
> O conceito de *Business Intelligence* com o entendimento de que é Inteligência de Negócios ou Inteligência Empresarial compõe-se de um conjunto de metodologias de gestão implementadas através de ferramentas de software, cuja função é proporcionar ganhos nos processos decisórios gerenciais e da alta administração nas organizações (ANTONELLI, 2009, apud ANGELONI e REIS, 2006, p. 3)

O conceito de *BI* surgiu na década de 80, através da *Gartner Group*[3]. Porém, para Antonelli (2009), ao olharmos o passado, povos que remontam a antiguidade, utilizavam, ao seu modo, *BI* para auxiliar na tomada decisões, agrupando dados e cruzando informações, que tinham como finalidade decidir desde o melhor momento para o cultivo - tendo como base os períodos de chuva e seca -, ou mesmo para a pesca, levando em conta o comportamento dos mares e das marés.

Com a rápida evolução das tecnologias usadas para o armazenamento e análise dos dados, muitas empresas e corporações passaram a adotar como prática o uso de sistemas baseados em *BI* para a criação dos seus planos de negócios e estratégias, a fim de aumentar a competitividade e eliminar falhas. Segundo a TOTVS (2024), para que o conceito de *BI* possa ser aplicado a um sistema, é necessário seguir quatro etapas. Na Figura 1, é possível observar a representação gráfica dessas etapas juntamente com a descrição de cada uma delas:

[3]https://www.gartner.com/en. Acesso em: 29 jan. 2024

Figura 1 – Etapas de aplicação do *BI*



Fonte: Elaborado pelo autor (2026)

1. **Coleta:** etapa onde é definido que tipo de informação será útil para o negócio e qual o objetivo a ser alcançado;
2. **Análise de dados:** etapa onde se estabelece como os dados ficarão organizados e quais as métricas usadas para a interpretação;
3. **Compartilhamento:** etapa que define a quem cada uma das informações interessa;
4. **Monitoramento:** etapa onde é feita a observação e a busca por novas fontes de dados, com o objetivo de manter as informações sempre atualizadas.

2.2 *OLTP*

A sigla *OLTP* (*On-Line Transaction Processing* ou Processamento de Transações Online), de acordo com Marijan (2021), refere-se a uma categoria de processamento de dados voltada a sistemas que trabalham através de transações realizadas de maneira concorrente por diferentes usuários. Esses sistemas permitem realizar operações de inserção, atualização e exclusão de dados operacionais de uma empresa ou organização.

Caracterizados por lidarem com transações que envolvam pequenas quantidades de dados, gerenciar de maneira eficiente às conexões simultâneas dos usuários, garantir a eficácia e integridade das informações, bem como a indexação destas - com o propósito de diminuir o tempo de resposta -, os sistemas *OLTP* acabam cumprindo um papel central nos empreendimentos, já que além de servirem como principal fonte

de consumo e criação de novos dados, auxiliam na geração de relatórios e no acompanhamento diário do negócio.

2.3 *Data Warehouse*

Data warehouses (*DW*) - ou armazéns de dados, na tradução literal -, são bancos de dados responsáveis por armazenar, de maneira histórica, grandes volumes de dados. Eles são usados para dar suporte a processos de tomada de decisões em empresas. Com o intuito de realizar essa tarefa de forma efetiva devido ao crescimento dos dados existentes nos empreendimentos, segundo Kimball e Ross (2013), o *DW* tem como duas de suas principais características: a responsividade, que possibilita a persistência contínua e ilimitada das informações, e a não limitação a uma única fonte de dados para o seu povoamento, desde que seja possível integrá-la sem a quebra do esquema previamente definido.

A modelagem e criação dos *DWs* se dá em cima da transferência e transformação de informações já existentes dos sistemas utilizados nas operações diárias da empresa para o próprio *DW*. Como descrito por Inmon (2005), o *DW* pode conter algumas características, como: a temporalidade, a orientação por temas, a não volatilidade dos dados, assim como a união de informações. Porém, isso não impede que algumas das características descritas sejam restritivas, podendo assim existir variações de um banco para o outro.

Quanto a sua organização interna, as tabelas presentes na base de dados são agrupadas em dois tipos: *fatos* e *dimensões*. O primeiro tipo de tabela é encarregado de armazenar os fatos de um determinado acontecimento, como ações ou eventos. Já o segundo é encarregado de persistir a descrição de cada fato armazenado no banco de dados.

Para Machado (2013), além dos motivos já citados na utilização desta tecnologia nas empresas - como as ferramentas de tomada de decisões -, existe ainda, como pano de fundo, as constantes alterações das informações nos vários sistemas transacionais do negócio. Isso acaba resultando em dificuldades na recuperação de dados históricos em períodos superiores a um ano e na falta de padronização na integração dos dados espalhados pelos diferentes sistemas.

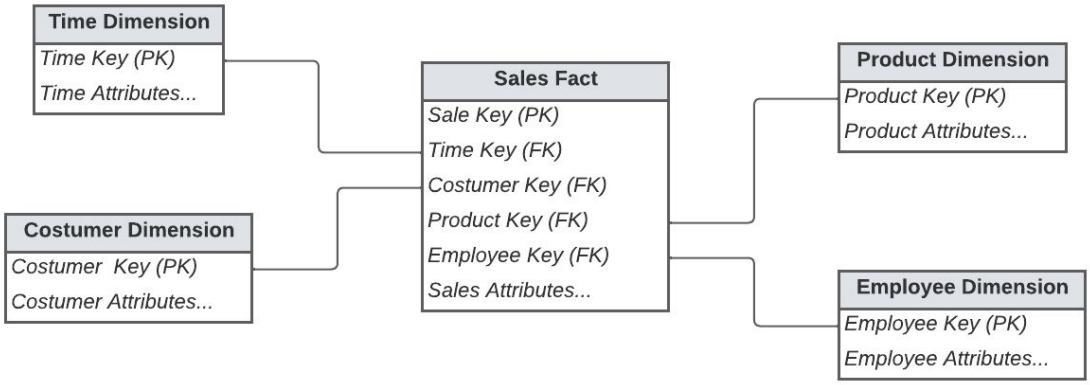
2.3.1 *Star Schema*

Introduzido na década de 90 por Ralph Kimball, o esquema estrela (*Star Schema*) é um dos modelos mais conhecidos e utilizados para a estruturação dos dados em um

DW (INMON, 2005). O nome faz analogia direta ao esquema lógico relacional da base de dados, uma vez que ele assemelha-se bastante ao formato de uma estrela, com as tabelas de *dimensão* irradiando de uma tabela central de *fatos*, responsável por armazenar, entre outras informações, uma referência para cada uma das dimensões existente no banco.

Neste modelo, de acordo com Kimball e Ross (2013), as tabelas do banco são desnormalizadas em relação à 3FN de Codd, com o objetivo de garantir consultas mais eficientes e evitar junções complexas entre várias relações. Tal estratégia, porém, acaba resultando em uma alocação maior de espaço em disco, tendo em vista a redundância de dados presentes na base. A Figura 2 mostra um exemplo de disposição das relações no esquema estrela.

Figura 2 – DER baseado no esquema estrela



Fonte: Adaptado de GeekForGeeks (2023)

Na Figura 2, é possível observar a existência da tabela central de fatos intitulada *Sales Fact*, que contém os dados das vendas realizadas pela empresa. Ela é composta por uma chave primária, *sale key*, um conjunto de chaves estrangeiras (usadas para referenciar as dimensões) e atributos responsáveis por ajudar a mensurar o fato. As tabelas de dimensão, por sua vez, são formadas pelas suas respectivas chaves primárias, bem como por um bloco de atributos que as descrevem. Por exemplo, na relação *Time Dimension*, os seus atributos descrevem as características temporais da venda, como a data, o dia da semana, o período do dia, entre outras.

2.4 Ferramenta *ETL*

Normalmente, um *data warehouse* é povoado com dados coletados a partir de vários bancos de dados locais. Esse processo é chamado de *ETL*, e o seu nome faz referência ao significado das suas letras iniciais: *Extract*, *Transform* e *Load*. O processo de *ETL* é desempenhado por uma ferramenta específica do *DW*, chamada ferramenta

ETL. Segundo Kimball e Ross (2013), tais ferramentas são formadas por estruturas de dados instanciadas, além de um conjunto de processos responsável por executar a operação, fazendo deste componente uma ponte entre os sistemas de origem dos dados e o sistema de *DW/BI*.

Na etapa de extração dos dados, a ferramenta *ETL* identifica e coleta os dados das fontes que são usadas para manter e alimentar continuamente o *DW*. Nessa etapa podem ocorrer algumas dificuldades, já que, a depender da forma como os dados encontram-se dispersos entre os sistemas, pode ser necessária a criação de estratégias diferentes para a recuperação em cada uma das fontes utilizadas.

Passada a fase de extração, tem-se início a etapa de transformação dos dados. Tida muitas vezes como a mais crítica das três etapas - já que precisa lidar com as especificidades de como os dados serão integrados ao esquema, seguindo as regras de negócio pré-estabelecidas -, é nela onde são efetuadas a limpeza, formatação, exclusão de duplicações e muitas vezes a combinação de informações, buscando manter a consistência e fidelidade dos dados originais (TEBALTI, 2022).

Por fim, a última etapa é a de carregamento dos dados. Nela são efetuadas as inserções dos dados transformados para a base de dados de destino, acomodando os novos dados nas suas respectivas tabelas de dimensões e fatos.

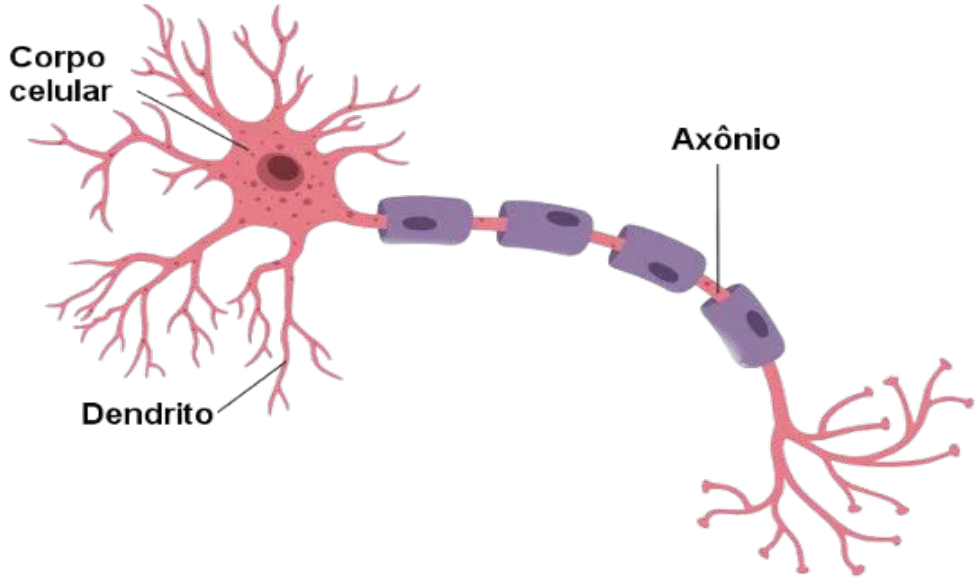
2.5 Redes Neurais Artificiais

Considerado o núcleo de inteligência e aprendizagem do organismo, o cérebro humano é o órgão, segundo Machado e Haertel (2014), responsável por comandar todas as nossas atividades e sentimentos, tais como movimentos corporais, memória, visão, emoções, entre outros. O cérebro é formado por aproximadamente 86 bilhões de neurônios que se conectam uns aos outros através de sinapses[4], para juntos compor o que é denominado Rede Neural. Na Figura 3 é possível observar a estrutura de um neurônio, junto com a descrição das principais partes que o compõem. Essas partes são:

- **Corpo celular:** local do neurônio onde está presente o núcleo, grande parte das organelas celulares e de onde partem os prolongamentos da célula;
- **Dendrito:** prolongamentos do neurônio que garantem a recepção dos estímulos, levando o impulso nervoso em direção ao corpo celular;
- **Axônio:** prolongamento longo, que garante o envio do impulso nervoso.

[4]Região de proximidade entre um neurônio ou outra célula por onde são transmitidos impulsos nervosos.

Figura 3 – Neurônio multipolar

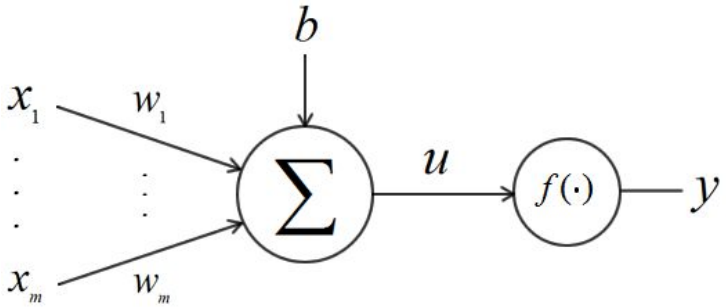


Fonte: Adaptado de Junqueira, Carneiro e Abrahamsohn (2017)

Com o propósito de modelar computacionalmente as conexões do cérebro humano e replicar comportamentos inteligentes nas máquinas, surgiu em 1943 o conceito de neurocomputação que, com o passar dos anos e avanço das tecnologias, ganhou força e abriu caminho para o surgimento das Redes Neurais Artificiais (RNAs). Segundo Grübler (2018), as RNAs são estruturas complexas com capacidade de realizar múltiplas operações em paralelo, a fim de processar dados e representar formas de conhecimento, com o objetivo de solucionar problemas complexos do dia a dia de forma ágil.

Tal como as redes neurais biológicas, as RNAs são constituídas de neurônios (intitulados neurônios artificiais, ou unidades de processamento), que se interconectam com o propósito de armazenar conhecimento experimental e disponibilizá-lo para uso (PACHECO, 2015, apud Haykin, 1998). Segundo descrito por McCulloch e Pitts (1943), a atividade de um neurônio funciona como um processo de “tudo ou nada”, no qual o neurônio recebe múltiplos sinais de entrada e produz uma resposta binária de saída. Como forma de exemplificar, na Figura 4, é apresentado o modelo de neurônio artificial, denominado *perceptron*, criado por Frank Rosenblatt, e baseado nos trabalhos de McCulloch e Pitts (ROSENBLATT, 1958).

Figura 4 – Modelo do neurônio artificial *perceptron*



Fonte: Pacheco (2015)

No modelo observado na Figura 4, as *m* entradas (x_i) do neurônio representam as informações do processo que se deseja mapear. Para cada uma delas, existe um peso (w_i) sináptico ponderado, representando a importância deste ao valor da saída final esperada (*y*). O valor resultante da somatória das entradas multiplicado pelos pesos ($x_i * w_i$), é somado a uma variável de deslocamento linear *(b)*, que resultará no potencial de ativação (*u*), repassado como argumento para a função encarregada de realizar a ativação *f(.)* - a função desempenhará o papel de limitar a amplitude da saída do neurônio dentro de um intervalo fechado -, resultando ao final do processo, na saída desejada.

Este mesmo modelo de neurônio, pode ser expresso matematicamente através da seguinte equação.

Equação 1

$$y = f\left(\sum_{i=1}^{m} x_i w_i + b\right)$$

Ao comparar as estruturas dos neurônios apresentados nas Figuras 3 e 4, é possível notar uma semelhança entre as partes que as compõem e as funções desempenhadas por cada uma delas. O valor das entradas, por exemplo, retratado no modelo perceptron por (x_i), faz referência direta ao papel realizado pelos dendritos celulares da célula biológica. Os pesos, indicados por (w_i), equivalem às sinapses das ligações celulares. Já a soma da multiplicação dos pesos pelas entradas faz uma clara alusão ao corpo celular do neurônio. Finalmente, o resultado final obtido, simbolizado por (*y*), corresponde explícitamente ao axônio celular.

Ainda de acordo com Rauber (2014), às Redes Neurais Artificiais podem ser clas-

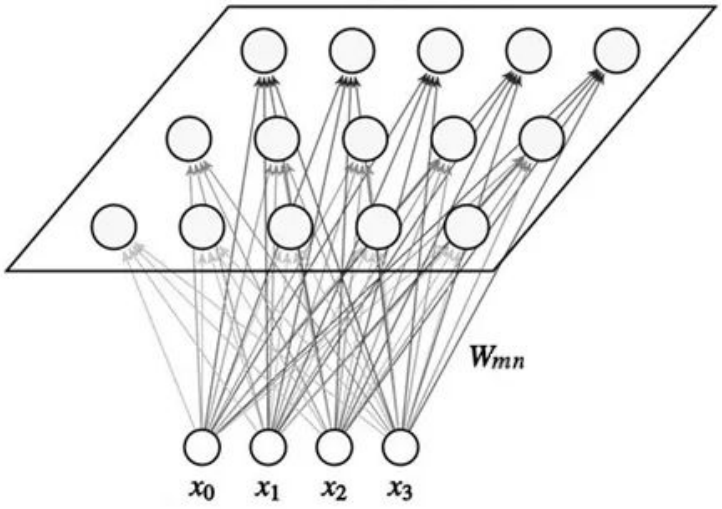
sificadas em dois grandes grupos: as de aprendizagem supervisionada e as de aprendizagem não supervisionada. Nas redes de aprendizagem supervisionada, para cada um dos exemplos utilizados no treinamento da rede, deve ser fornecido o valor desejado como resposta. Já nas redes de aprendizagem não supervisionada, a rede só tem como valores disponíveis os de entrada, o que a obriga a buscar atributos estatísticos relevantes a fim de desenvolver uma representação própria com base nos estímulos recebidos.

2.5.1 *Self-Organizing Maps*

Os *Self-Organizing Maps* (*SOM*), ou Mapas Auto-Organizáveis, é um modelo de rede neural não supervisionada, apresentado por Teuvo Kohonen (1982) e baseado em aprendizagem competitiva. De acordo com Dantas, Matos e Pinho (2020), nesse modelo cada um dos neurônios presentes na rede compete entre si, ajustando os seus pesos de forma contínua, com o objetivo de ficar o mais próximo possível do padrão de entrada fornecido, ao qual cada neurônio está associado.

Segundo Furtado (2019), a ideia por trás do algoritmo é replicar o funcionamento organizacional do cérebro humano, que se divide em áreas distintas, cada uma especializada no processamento de diferentes tipos de informações sensoriais, como a visão, a audição e o tato. No contexto da RNA, é a própria rede que se encarrega de criar os grupos de neurônios especialistas utilizando as entradas fornecidas para representar os dados de forma topológica, empregando estruturas uni, bi ou tridimensionais, com o propósito de auxiliar no agrupamento dos neurônios e no mapeamento de ordem das entradas. Na Figura 5, é apresentada a arquitetura bidimensional da rede *SOM*.

Figura 5 – Arquitetura bidimensional da rede *SOM*



Fonte: Adaptado de Ali (2019)

Como ilustrado na Figura 5, a estrutura da rede *SOM* é disposta em uma matriz bidimensional *m* x *n*, na qual o valor *m* representa a quantidade de linhas e *n* o total de colunas. Para cada neurônio presente na estrutura da rede, o conjunto de pesos passa a ser igual ao tamanho do vetor de entradas fornecido (x_n) - composto pelos valores x_0, x_1, x_2 e x_3 -, o que resulta em uma matriz de pesos (w_{mn}).

Quando comparado a outros modelos de redes neurais, o processo de treinamento da rede *SOM* é considerado relativamente simples. Segundo Pacheco (2017), o treinamento da rede é dividido em duas etapas: competição e cooperação. Na etapa de competição, ao serem fornecidas as entradas à rede (x_n), os pesos sinápticos (w_n) são inicializados com valores aleatórios, para assim ser dado início ao cálculo da distância entre o valor da entrada e os neurônios existentes na rede, tendo como base a utilização de algumas métricas (sendo a mais comum delas a distância Euclidiana, expressa na Equação 2) para que, através dos valores obtidos, e levando em consideração os resultados de menor valor, sejam definidos os neurônios vencedores (*winners*).

Equação 2

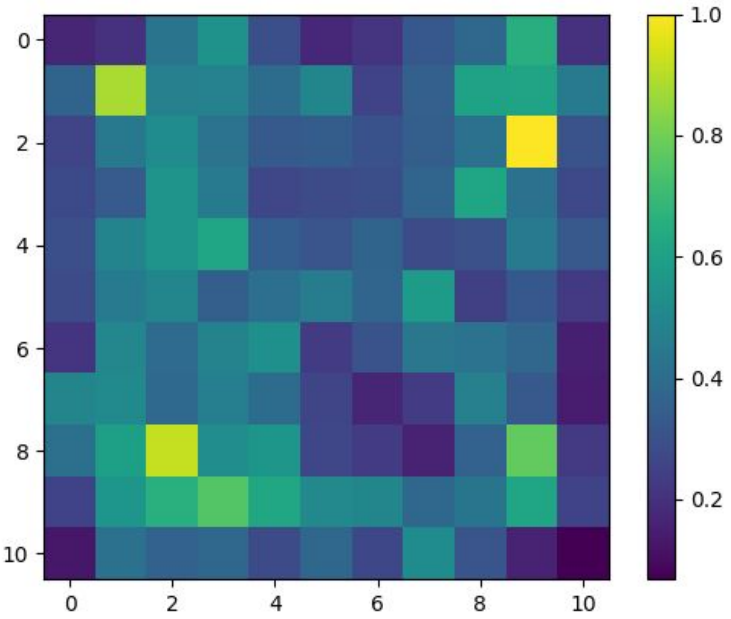
$$T_{j,i}(x) = exp\left(-\frac{S_{j,i}(x)^2}{2\delta^2}\right)$$

Na Equação 2, o valor de *S* representa a distância lateral entre os neurônios *i* e *j*. Já o valor de δ equivale à largura ou escala da zona de influência de cada um dos

neurônios sobre os demais presentes na rede. A função exponencial (com a distância elevada ao quadrado), ao ser aplicada, contribui para a queda do número de treinamentos, fazendo com que os pesos passem a ser atualizados continuamente, forçando-os a se moverem de acordo com as alterações na vizinhança topológica, para que, a cada iteração, o neurônio vencedor puxe os demais para sua zona de influência. O processo termina quando não houver mais nenhuma mudança na matriz.

Ao final do processamento, é gerada uma matriz de resultados com os valores obtidos. Na Figura 6, é mostrado um modelo da matriz de resultados.

Figura 6 – Modelo da matriz de resultados



Fonte: Elaborado pelo autor (2024)

Na Figura 6 é possível visualizar um modelo da matriz de resultados, tendo como base um conjunto de dados fornecidos a uma rede *SOM*. Por padrão, cada uma das posições existentes na matriz é formada pelo agrupamento dos dados analisados. Nesse caso as posições que se encontram mais próximas à escala de 1.0 representam os locais onde estão presentes as informações desejadas.

Dentre as várias aplicações de uso da rede *SOM*, é possível citar o mapeamento e análise de dados geoespaciais, a detecção e busca por fraudes financeiras, bem como a análise de exames clínicos.

2.6 *NoSQL*

Com o surgimento de aplicações que passaram a demandar um alto volume de leitura e escrita de dados, além das constantes mudanças nas regras de negócios e a busca contínua pela redução de custos, relacionados ao armazenamento de dados e escalonamento de servidores, o desenvolvimento de sistemas distribuídos utilizando *RDBMS* (*Relational Database Management System*) tornou-se uma tarefa cada vez mais difícil de ser realizada.

De acordo com Paniz (2016), com o objetivo de atender a essas novas necessidades, um grupo formado por empresas e programadores deu início ao movimento que mais tarde ficaria conhecido como *NoSQL* (abreviação para *Not only SQL* ou Não apenas *SQL*), e que é usado atualmente para se referir aos bancos de dados que não seguem o modelo relacional. Tal movimento teve como principal meta a busca de novas alternativas para modelagem, estruturação e armazenamento dos dados.

Caracterizados por trabalharem com base no paradigma *BASE* (*Basically Available*, *Soft state*, *Eventual consistency*), os bancos *NoSQL* têm como principal foco garantir a alta disponibilidade e a consistência eventual das informações existentes na base, fazendo deles o oposto dos bancos relacionais, que se baseiam no paradigma *ACID* (*Atomicity*, *Consistency*, *Isolation*, *Durability*), que prioriza uma consistência mais rigorosa e uma maior integridade dos dados.

Dentre algumas das características importantes dos bancos *NoSQL*, destacam-se a escalabilidade horizontal da aplicação, que reflete diretamente na queda de custos, a ausência de um esquema fixo para os dados, o suporte nativo à replicação e *APIs* simples de acesso e conexão à base (KOKAY, 2012).

Além disso, devido às diferentes formas de estruturação e armazenamento dos dados, os bancos *NoSQL* são categorizados em quatro principais grupos: chave-valor, família de colunas, orientado a documentos e orientado a grafos.

2.6.1 Bancos de Dados Orientados a Documentos

Neste modelo de banco, as informações são organizadas em coleções de documentos, nas quais os dados são dispostos em estruturas semelhantes àquelas usadas em documentos *JSONs* e *XMLs*, o que permite aos sistemas evoluírem e se adaptarem conforme as necessidades da própria aplicação, uma vez que tais documentos não contam com um esquema fixo de dados.

A organização dos documentos presentes no banco de dados é formada, tal como em um *JSON*, por pares de chave-valor - com cada chave descrevendo o atributo que

compõe o documento, seguido do seu respectivo valor, que pode vir a assumir desde de tipos primitivos de dados ou até mesmo outros documentos. Dentre as características deste modelo de banco de dados, destacam-se o fato de cada documento possuir um identificador único universal (*UUID*), a resolução de consultas por meio de métodos avançados de agrupamento e filtragem (*MapReduce*) e a possibilidade de redundância das informações, uma vez que um mesmo dado pode estar presente em mais de um documento (MEDEIROS, 2014).

Na Figura 7, é possível observar a estrutura de um documento armazenado no servidor de banco de dados *MongoDB*[5]. Nele é descrita uma coleção chamada *Providers*, que é responsável por representar e conter dados essenciais dos fornecedores da empresa, como o *cnpj*, *email*, telefone, nome e endereço.

Figura 7 – Estrutura de documento no *MongoDB*


```
{
   "_id": {
      "$oid": "64232a02dd21b031099f76f5"
   },
   "cnpj": "18.500.685/0001-09",
   "email": "atendimento@bambu.com.br",
   "telephone": "(22) 99766-1187",
   "name": "Grupo Slayder",
   "address": {
      "cep": "28893-076",
      "uf": "RJ",
      "city": "Rio das Ostras",
      "district": "Rio das Ostras",
      "street": "Rodovia Amaral Peixoto",
      "numbering": 4961
   }
}
```


Fonte: Elaborado pelo autor (2024)

2.6.2 Bancos de Dados Chave-Valor

No modelo de banco de dados chave-valor, os dados são dispostos em conjuntos de pares de chaves, bastante similares às estruturas *map* e dicionário das linguagens de programação. Nessas estruturas há um valor distinto associado para cada chave, garantindo que cada informação esteja isolada e dissociada explicitamente de qualquer outro dado presente no banco.

De acordo com Napoleon (2023), os bancos de dados chave-valor são usados amplamente em aplicações para armazenar os dados mais frequentemente acessados

[5]MongoDB versão 4.4.29. https://docs.mongodb.com/v4.4/release-notes/4.4/. Acesso em: 16 mar. 2024

em cache, gerenciar sessões, bem como persistir configurações e metadados. Dentre as principais características desses bancos de dados destacam-se o suporte integrado para recursos de escalabilidade, a alta performance nas operações de leitura e escrita dos dados (uma vez que a base trabalha em cima de uma tabela de dispersão para armazenar as informações), e a facilidade de uso, tendo em vista, não ser necessário o mapeamento dos objetos em tabelas adjacentes (AWS, s.d.b).

No Algoritmo 1, é apresentada a implementação de uma classe que realiza a conexão, e as operações de leitura e escrita de dados no servidor de banco Redis[6], baseado no modelo chave-valor, e que foi utilizado para o desenvolvimento deste trabalho.

Algoritmo 1 – Classe *DAO* do banco *Redis*

```
1  from redis import Redis
2  from dotenv import load_dotenv
3  import os
4  load_dotenv()
5
6  class RedisDao:
7
8      def __init__(self):
9          self.redis = Redis(
10             host=os.environ['HOST_REDIS'],
11             port=os.environ['PORT_REDIS'],
12             db=0
13          )
14
15     def set_value(self, key, value):
16         self.redis.set(key, value, ex=11880)
17
18     def get_value(self, key):
19         return self.redis.get(key)
20
21     def delete_value(self, key):
22         return self.redis.delete(key)
```

Fonte: Elaborado pelo autor (2024)

A escolha do servidor de banco Redis se deu, em grande parte, pela sua característica de armazenar as informações em memória, a fim de garantir maior velocidade na leitura e escrita dos dados. No Algoritmo 1 pode-se ver que no construtor da classe são

[6] https://redis.io/docs. Acesso em: 26 set. 2024

definidos o *host* e a porta onde o banco de dados está localizado. Já na parte de leitura e escrita dos dados, a classe conta com os métodos: *get_value*, que recebe como parâmetro o identificador (chave) do valor a ser recuperado; *delete_value*, usado para a exclusão de um determinado valor da sua chave; e *set_value*, que recebe a chave para a identificação do dado, juntamente com o valor a ser persistido.

2.7 *RESTful*

Proposto nos anos 2000 por Roy Fielding, o padrão *REST - Representational State Transfer* ou Transferência de Estado Representacional, na tradução literal - é um modelo de arquitetura para sistemas baseados em comunicação via rede, que tem como principal objetivo orientar e definir um conjunto de princípios e restrições para o desenvolvimento de sistemas distribuídos. Ao longo dos anos, o modelo acabou ganhando significativa aceitação entre os desenvolvedores, que passaram a considerá-lo uma alternativa viável para a implementação de *Web Services*, em contraponto ao já bastante utilizado *SOAP - Simple Object Access Protocol*[7] (FERREIRA, 2017).

Para que seja considerada *RESTful*, a *API* precisa ter seu escopo voltado para o ambiente Web, bem como seguir os cinco princípios fundamentais definidos por Fielding na arquitetura *REST*, que são:

1. **Cliente-Servidor:** a comunicação deve ocorrer entre os agentes cliente e servidor, na qual o cliente solicita, por meio de requisições, recursos ao servidor;
2. ***Cache*:** alguns dados devem ficar armazenados em cache, para eliminar a necessidade de interações repetitivas entre o cliente e o servidor, a fim de melhorar a performance do sistema;
3. ***Stateless*:** as informações do cliente não devem ficar armazenadas no servidor entre uma requisição e outra;
4. ***Layered*:** a comunicação entre cliente e servidor deve acontecer por meio de camadas, de uma forma que cada camada tenha funcionalidade e responsabilidade específica, permitindo a interação entre elas e uma organização de forma hierárquica;
5. **Interface Uniforme:** a troca de mensagens entre cliente e servidor deve acontecer de forma semelhante, tanto no modo como no formato dos dados.

Ao optar pela criação de um sistema utilizando *REST*, o desenvolvedor acaba adquirindo uma série de benefícios, como facilidade de escalonamento da aplicação,

[7] https://www.w3.org/TR/2000/NOTE-SOAP-20000508/. Acesso em: 25 mai. 2024

maior flexibilidade ao efetuar alterações em nível de banco de dados ou código, independência do uso de tecnologias no *front-end* e *back-end*, otimização do desempenho, controle de acesso dos usuários e a possibilidade de uso em múltiplas plataformas (AWS, s.d.a; CARVALHO, 2022).

2.8 *GraphQL*

Lançado em 2012 pelo então *Facebook Inc.* (atualmente *Meta Platforms*), e tornado *open source* em 2015, o *GraphQL* (*Graph Query Language*) é uma linguagem de consultas e ambiente de execução voltada ao desenvolvimento de *APIs* (ANTONIO et al., 2023). Criada como uma alternativa ao padrão *REST*, ela tem como principal objetivo garantir aos clientes maior flexibilidade na solicitação das informações, uma vez que é possível definir quais campos e relacionamentos deverão ser retornados pelo servidor, evitando a necessidade de múltiplas chamadas a diferentes *endpoints* e o recebimento de dados, muitas vezes, desnecessários.

Diferente da arquitetura *RESTful*, que exige a definição de múltiplas rotas - cada uma associada a determinado método do protocolo escolhido na implementação da *API* - para a execução de tarefas relacionadas à leitura e escrita de dados, o *GraphQL* opera através de um único *endpoint*, o que facilita a evolução da aplicação no médio e longo prazo, já que torna-se fácil a adição de novos campos, tipos e funcionalidades, sem afetar os clientes existentes que venham a não utilizar essas novas funções (DESENVOLVEDOR, 2024).

Em *GraphQL*, as entidades e os métodos de leitura e escrita são representados como tipos. Um arquivo denominado *schema.gql* é criado para a definição e organização desses tipos. Nele também é configurado o funcionamento de toda a aplicação. O Algoritmo 2 apresenta a estrutura parcial de um arquivo *schema.gql*.

Algoritmo 2 – Estrutura parcial do arquivo *schema.gql*

```
type Company {
    _id: ID!
    address: Address
    clients: [Client!]
    closingDate: DateTime
    cnpj: String!
    email: String!
    employees: [Employee!]
    fantasyName: String
    isSubsidiary: Boolean!
    movements: [Movement!]
    name: String!
    openingDate: DateTime!
    products: [Product!]
    providers: [Provider!]
    stateRegister: String!
    telephone: String!
}

type Mutation {
    addCompany(company: CompanyInput!): Company
    deleteCompanyById(id: ObjectId!): Company
    updateCompany(company: CompanyInput!): Company
    {...}
}

type Query {
    listCompanies: [Company!]
    getCompanyById(id: ObjectId!): Company
    {...}
}
```

Fonte: Elaborado pelo autor (2024)

No Algoritmo 2, pode-se observar a existência de três tipos. O primeiro, intitulado *Company*, é responsável por descrever uma empresa. Esse tipo é composto por vários atributos, sendo que alguns desses atributos representam referências a outros tipos definidos previamente no arquivo, como *Address* e *Employee*. O segundo tipo é o *Mutation*, que é responsável por descrever o conjunto de métodos da *API* que ficarão incumbidos de efetuar as operações de escrita no banco de dados(adição, exclusão

e edição), recebendo os valores a serem usados a cada operação e o tipo de retorno esperado ao final da tarefa. Por fim, o tipo *Query* é formado exclusivamente pelos métodos responsáveis por realizar a leitura dos dados, podendo receber ou não parâmetros que venham a ser necessários para cada consulta e o tipo esperado como resposta.

Como forma de demonstrar o funcionamento entre uma *API GraphQL* e *RESTful*, no Algoritmo 3, é possível visualizar a estrutura de uma requisição *HTTP*, por meio de um cliente *web*, para um sistema apoiado no modelo *RESTful*.

Algoritmo 3 – Requisição *HTTP* a *API RESTful* via método *GET*

```
1  api.get(`/financial/bar-chart/${selectedYear}`).then(resp => {
2      setBarChart(resp.data);
3  });
```

Fonte: Elaborado pelo autor (2024)

O Algoritmo 3 mostra a utilização de uma variável denominada *api* que implementa a função *GET*, na qual é informada uma rota baseada no ano, que é definido de forma dinâmica pela variável *selectedYear*. A função retorna um documento no formato *JSON*, que pode ser visto na Figura 8.

Figura 8 – Resposta recebida via requisição *HTTP* no formato *JSON*


```
[
   {
      "month": "Jan.",
      "RECEITA": 143729.22,
      "DESPESA": 115318.98
   },
   {
      "month": "Fev.",
      "RECEITA": 139609.52,
      "DESPESA": 112637.98
   },
   {
      "month": "Mar.",
      "RECEITA": 147856.89,
      "DESPESA": 115631.98
   },
   {
      "month": "Abr.",
      "RECEITA": 130569.96,
      "DESPESA": 110478.98
   }
]
```


Fonte: Elaborado pelo autor (2024)

Como observado na Figura 8, a resposta fornecida pela *API* é formada por um *array* de objetos, definidos previamente no *back-end*, e compostos pelos campos *month*, RECEITA e DESPESA.

Tendo em vista a estrutura da requisição vista na Figura 8, caso fosse necessário obter algum outro dado do sistema ou mesmo acrescentar um novo campo a ser utilizado por algum cliente na aplicação, existiriam dois possíveis caminhos a serem seguidos: alterar o objeto enviado na resposta, o que afetaria diretamente outros clientes que não precisassem da informação, ou implementar uma nova rota, a fim de ser usada unicamente pelo cliente específico, o que acabaria impactando na estruturação e organização do sistema, e que poderia ser facilmente contornado com a utilização do *GraphQL*.

No Algoritmo 4, é apresentada a estrutura da requisição de uma *API GraphQL*.

Algoritmo 4 – Estrutura de requisição a uma *API GraphQL*

```
query {
    listCompany {
        _id
        name
    }
}
```

Fonte: Elaborado pelo autor (2024)

O código do Algoritmo 4 mostra que, diferentemente de uma requisição *RESTful*, as chamadas a *APIs GraphQL* se dão por meio de consultas. Logo, não é preciso especificar qual o método do protocolo definido na implementação (*PUT*, *GET*, *POST* caso seja o *HTTP* ou mesmo *SEND* e *CLOSE*, caso opte-se pelo *WebSocket*) deverá ser chamado, e sim informar na própria consulta qual o tipo de operação, das declaradas no arquivo será realizada (*Query* ou *Mutation*), a função da *API* e os dados que deverão ser retornadas. Tendo como base os tipos definidos anteriormente no arquivo *schema.gql*, foi montada uma consulta do tipo *Query*, para listar todas as empresas cadastradas no sistema, recuperando destas o *_id* e o *name*. O resultado da consulta pode ser visto no Algoritmo 9.

Figura 9 – Resposta da *API GraphQL*


```
{
   "data": {
      "listCompany": [
         {
            "_id": "6423134add21b031099f73d0",
            "name": "Indústria de Churrasqueiras Fortaleza Ltda."
         },
         {
            "_id": "6448f5b19947be01f1c81d8c",
            "name": "Indústria de Churrasqueiras Fortaleza Fl-ce."
         }
      ]
   }
}
```


Fonte: Elaborado pelo autor (2024)

2.9 *React*

Criado em 2011 por Jordan Walke, engenheiro de *software* do *Facebook*, o *React* é uma biblioteca *Javascript* usada para a criação de interfaces de usuário, com foco no desenvolvimento de aplicações para internet e dispositivos móveis (FEDOSEJEV, 2015). Desde 2013, ano em que a tecnologia passou a ser *open source*, a biblioteca popularizou-se rapidamente e passou a ser usada por grandes empresas, como *Netflix Inc.*, *Microsoft Corporation*, *Yahoo-AOL LLC* e *Airbnb Inc.*, bem como por um número expressivo de desenvolvedores independentes. Tal sucesso se deve em parte ao forte apoio da comunidade, assim como à aplicação e ao aprimoramento de conceitos já bastante conhecidos na área da programação. Segundo a JavaScript (2025), o *React* teve uma participação de 67% no mercado, frente a outros *frameworks* e bibliotecas *Javascript*.

Construída com base no conceito de desenvolvimento baseado em componentes, a biblioteca permite aos desenvolvedores modularizar interfaces através da criação de comportamentos e atributos específicos para cada um dos componentes que venham a ser criados, o que possibilita à aplicação adaptar-se à plataforma na qual é renderizada. Dentre outros pontos positivos da tecnologia, destacam-se ainda, a facilidade no reuso de código, a diminuição no tempo de desenvolvimento e a baixa curva de aprendizagem. No Algoritmo 5, é apresentada a criação de um componente.

Algoritmo 5 – Criação do componente *CardTools*

```
1  import React from 'react';
2  import { Card, Title } from './style-card-tools';
3  import { CardToolsI } from '../../utils/interfaces';
4
5  const CardTools: React.FC<CardToolsI> = ({ name, type, ...rest
       }) => {
6
7      return (
8          <div {...rest}>
9              <Card>
10                 {type}
11                 <Title>{name.toUpperCase()}</Title>
12             </Card>
13         </div>
14     );
15 };
16
17 export default CardTools;
```

Fonte: Elaborado pelo autor (2024)

No Algoritmo 6, é possível observar a declaração de duas instâncias do componente *CardTools*, com a passagem de valores distintos para cada um dos atributos existentes.

Algoritmo 6 – Utilização do componente

```
441             return (
442                 <div className="container-utils">
443
444                     <CardTools name="gerentes"
445                         type={<MdOutlineManageAccounts
446                             className="icons-tool" />}
447                         onClick={() =>
448                             setOpenListManager(!openListManager)}
449                     />
450
451                     <CardTools name="mensagem"
452                         type={<MdOutlineMessage
453                             className="icons-tool" />}
454                         onClick={() =>
455                             setOpenListMessage(!openListMessage)}
456                     />
457
458                 </div>
459             );
```

Fonte: Elaborado pelo autor (2024)

Por fim, na Figura 10, é mostrada a renderização da instância dos componentes criados anteriormente.

Figura 10 – Renderização dos componentes

Fonte: Elaborado pelo autor (2024)

3 SOLUÇÃO PROPOSTA

Este capítulo descreve o processo de implementação do sistema proposto por este TCC. Inicialmente, são apresentados os artefatos de análise do projeto, destacando os *stakeholders* e requisitos levantados. Na sequência, são abordados o projeto arquitetural, as atividades do sistema e os esquemas utilizados na criação das bases de dados. Feito isso, o capítulo finaliza com a descrição acerca da implementação do sistema.

3.1 Análise

Esta seção aborda os resultados da análise efetuada para o desenvolvimento do sistema proposto por este trabalho.

3.1.1 *Stakeholders*

Segundo Project Management Institute (2021), os *stakeholders* (partes interessadas) correspondem a todos os envolvidos no desenvolvimento de um sistema. O sistema proposto neste TCC possui dois tipos de atores: o *gerente* e o *diretor-executivo* da empresa, tendo em vista que os mesmos são os usuários finais da aplicação. Os *gerentes* são os responsáveis por gerir as informações da matriz ou de uma das filiais da empresa, mantendo as informações financeiras e operacionais de suas respectivas divisões atualizadas. Já o *diretor-executivo* é incumbido de controlar todo o funcionamento do sistema, adicionando e removendo as filiais, gerentes, consultando informações específicas e gerando o parecer da auditoria financeira.

3.1.2 Requisitos

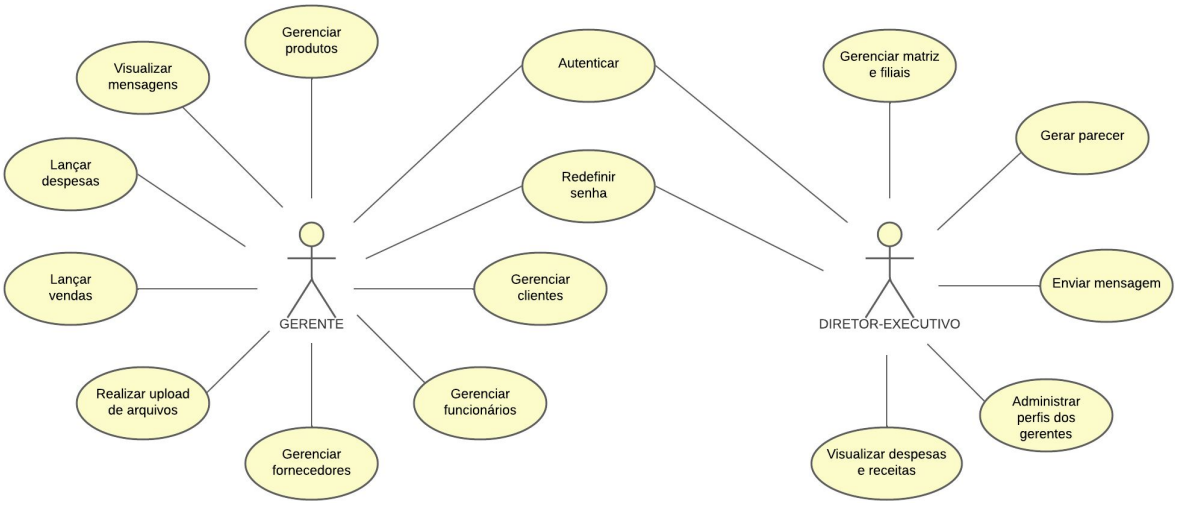
A realização do levantamento de requisitos de um sistema é o ponto fundamental na hora de implementá-lo. Segundo Pressman (2011, p. 56), “a meta do levantamento de requisitos é compreender o que os vários interessados esperam do software a ser desenvolvido”. Com base nessa definição, e após uma série de reuniões com equipe responsável pelo setor administrativo da Industria de Churrasqueiras Fortaleza (descrito de forma detalhada no Apêndice A), foram levantados os seguintes requisitos funcionais:

- **Autenticar usuários (RF_01):** o sistema deve possibilitar a autenticação dos usuários, para que eles tenham acesso apenas às funcionalidades relacionadas ao seu papel;

- **Redefinir senha (RF_02):** o sistema deve permitir aos usuários redefinir as suas respectivas senhas de acesso por meio do *e-mail* vinculado a sua conta;
- **Gerenciar matriz e filiais (RF_03):** o sistema deve permitir ao *diretor-executivo* o controle dos dados da matriz e filiais vinculadas ao negócio, bem como a adição, edição e exclusão de uma filial;
- **Enviar mensagem (RF_04):** o sistema deve permitir ao *diretor-executivo* o envio de mensagens a todos os gerentes;
- **Administrar perfis dos gerentes (RF_05):** o sistema deve permitir ao *diretor-executivo* a administração dos perfis de gerente, possibilitando a ele a adição de novos perfis, edição ou mesmo a exclusão;
- **Visualizar despesas e receitas (RF_06):** o sistema deve permitir ao *diretor-executivo* a visualização das despesas e receitas do empreendimento por períodos e categoria;
- **Gerar parecer (RF_07):** o sistema deve permitir ao *diretor-executivo*, auditar as contas do empreendimento por meio da geração de um parecer;
- **Visualizar mensagens (RF_08):** o sistema deve permitir ao *gerente* a visualização das mensagens cadastradas pelo *diretor-executivo*;
- **Gerenciar produtos (RF_09):** o sistema deve permitir ao *gerente*, o controle dos produtos, possibilitando a adição, edição e exclusão dos dados;
- **Gerenciar clientes (RF_10):** o sistema deve permitir ao *gerente*, o controle das informações dos clientes, possibilitando a adição, edição e exclusão dos dados;
- **Gerenciar funcionários (RF_11):** o sistema deve permitir ao *gerente*, o controle das informações dos funcionários, possibilitando a adição, edição e exclusão dos dados;
- **Gerenciar fornecedores (RF_12):** o sistema deve permitir ao *gerente*, o controle das informações dos fornecedores, possibilitando a adição, edição e exclusão dos dados;
- **Lançar vendas (RF_13):** o sistema deve permitir que o *gerente* efetue o lançamento das vendas realizadas;
- **Lançar despesas (RF_14):** o sistema deve permitir que o *gerente* efetue o lançamento das despesas contraídas;
- **Realizar *upload* de arquivos (RF_15):** o sistema deve permitir ao *gerente*, realizar o *upload* de arquivos com os dados financeiros da filial ao qual é responsável.

Com base nos requisitos funcionais, foi definido o diagrama de casos de uso do sistema, apresentado na Figura 11 e detalhado no Apêndice B.

Figura 11 – Diagrama de casos de uso



Fonte: Elaborado pelo autor (2024)

No diagrama de casos de uso, são apresentados os dois atores que utilizam o sistema: o *gerente* e o *diretor-executivo*. Por ser o responsável pela entidade como um todo, o *diretor-executivo* pode gerenciar os dados da matriz e filiais, enviar mensagens, administrar os perfis dos gerentes, visualizar despesas e vendas, bem como auditar as contas da empresa, gerando o parecer financeiro. Já o ator *gerente* terá acesso apenas às informações da matriz ou filial gerenciada por ele, ficando este encarregado por: gerenciar os produtos, funcionários, fornecedores e clientes; lançar vendas e despesas; visualizar as mensagens, enviadas pelo diretor-executivo, e efetuar o *upload* dos arquivos com os dados financeiros existentes.

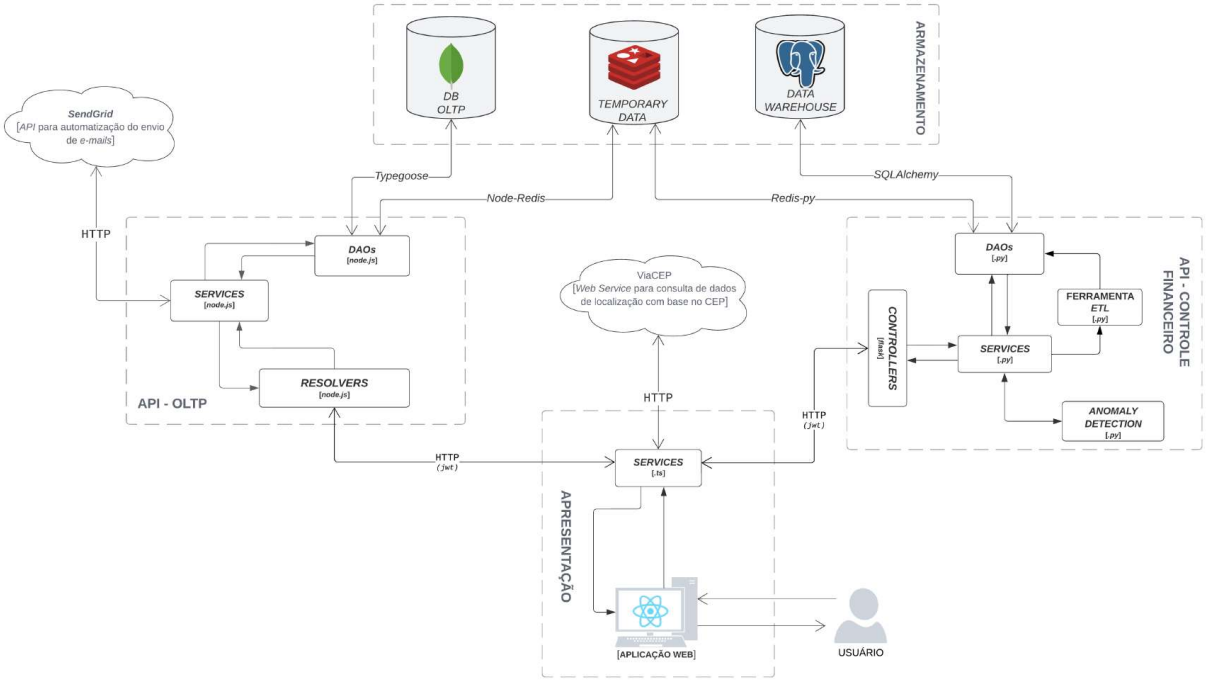
Tendo em vista o fato de o usuário diretor-executivo já vir cadastrado por padrão no sistema e este ser o responsável pelo gerenciamento do perfil gerente, optou-se pela não definição de um requisito funcional para o gerenciamento de usuários. Por questões de legibilidade, no que concerne a criação do diagrama apresentado na Figura 11, os casos de uso descritos com os verbos gerenciar e administrar se desdobram nas funcionalidades de adicionar, editar e excluir o dado ao qual estão sendo indicadas as ações.

Tendo como base, ainda, o levantamento de requisitos já realizado, foi possível identificar um requisito não funcional, no qual o sistema deve garantir que cada usuário tenha acesso apenas às suas funcionalidades (**RNF_1**).

3.2 Projeto Arquitetural

Nesta seção é apresentada uma visão geral do projeto arquitetural do sistema proposto por este TCC. A arquitetura usada na implementação do trabalho é composta de quatro camadas: *Apresentação*, *API de Controle Financeiro*, *API OLTP* e *Armazenamento*, sendo cada uma delas responsável por desempenhar um papel fundamental no funcionamento do sistema, trabalhando conjuntamente a fim de constituir a solução final desenvolvida. Na Figura 12 é possível visualizar o esquema arquitetural do projeto e como se dá a interação entre as camadas.

Figura 12 – Diagrama arquitetural



Fonte: Elaborado pelo autor (2024)

A camada de Apresentação é a responsável pela interação dos usuários finais do sistema com as *APIs* da aplicação. É por meio dela que são consumidos os serviços disponibilizados via rotas *HTTP* e consultas *GraphQL*, para manipulação e exibição das informações. Ela se subdivide em dois módulos.

O primeiro módulo é o de *Aplicação Web*. Construído totalmente em *React*, ele é encarregado de montar os componentes e as páginas finais dos usuários de forma dinâmica, por meio dos dados advindos do módulo *Services*. O módulo *Services*, por sua vez, é responsável por disponibilizar os métodos que realizam a comunicação entre a *Aplicação Web* e as camadas *API de Controle Financeiro*, *API OLTP* e o *Web Service ViaCEP*.

A segunda camada, intitulada *API de Controle Financeiro*, se encarrega do controle e da lógica referente ao tratamento e às consultas aos dados financeiros do sistema. Ela é subdividida em cinco módulos. O primeiro módulo, denominado *Controllers*, tem a função de receber as requisições *HTTP* vindas da camada de *Apresentação* e redirecioná-las para a atividade à qual a rota requisitada está vinculada, com o objetivo de, ao final da tarefa, retornar uma resposta. Já o segundo módulo, *Services*, tem como atribuições receber os dados repassados pelos controladores para realizar o tratamento destes (caso seja necessário), aplicar as regras de negócios referentes à *API* e conectar os demais módulos da camada.

O terceiro módulo, *DAOs*, desempenha o trabalho de comunicação, através de métodos, com o *Data Warehouse* e o *Temporary Data*, para realizar as operações de leitura e escrita nos bancos. Já o quarto módulo, *Ferramenta ETL*, é incumbido de adaptar e carregar as informações disponibilizadas pela camada *API OLTP*, via *Temporary Data*, para o *DW*. Por fim, o módulo *Anomaly Detection* tem a atribuição de executar operações em cima dos dados presentes no *DW* para a detecção de possíveis discrepâncias e anomalias nos balanços financeiros da empresa.

A terceira camada, *API OLTP*, é responsável por gerenciar os dados referentes às transações diárias realizadas pelo empreendimento. Ela é composta por três módulos. O primeiro módulo, intitulado *Resolvers*, compreende o conjunto de *mutations* e *queries* usados para comunicação entre a *API* e a camada de Apresentação do sistema, com o objetivo de realizar as operações disponibilizadas pelo servidor. O segundo módulo, *Services*, tem a função de aplicar as regras de negócio da aplicação, bem como abrir uma comunicação direta com a *API* externa de envio de mensagens *SendGrid* e executar tarefas que venham a ser definidas em segundo plano. O terceiro e último módulo, *DAOs*, disponibiliza os métodos para a leitura e escrita dos dados nos bancos *OLTP* e *Temporary Data*, tendo como base os parâmetros fornecidos pelo módulo *Services*.

Por último, a camada de *Armazenamento* tem a tarefa de guardar as informações geradas pelos usuários do sistema e as obtidas por meio de terceiros. Ela tem como núcleos de armazenagem os SGBDs *MongoDB*, *Redis* e *PostgreSQL*.

3.3 Atividades do Sistema

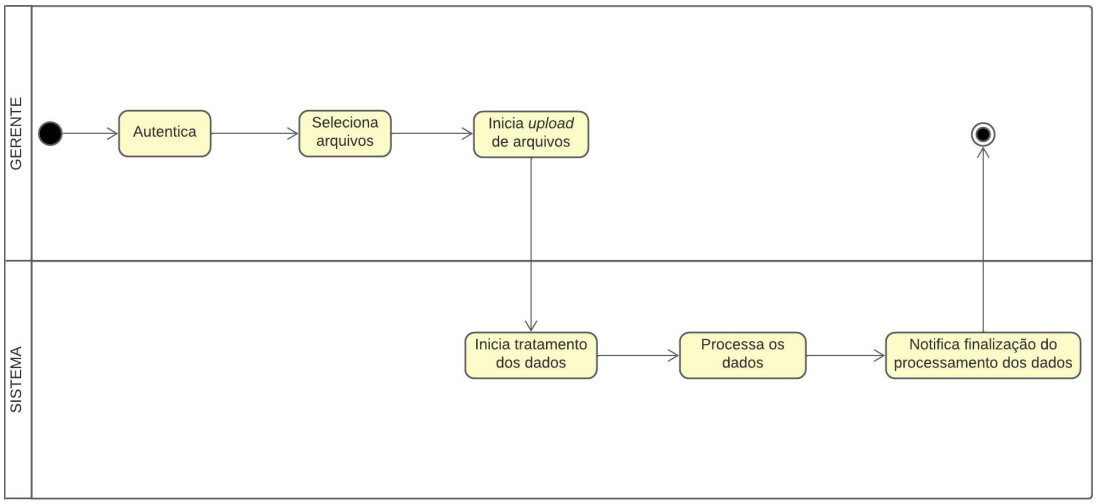
Esta seção apresenta algumas das atividades realizadas pelo sistema.

3.3.1 Importação dos Arquivos Financeiros

Nesta atividade, é mostrado o passo a passo que o usuário *gerente* deve executar para disponibilizar alguns dos dados financeiros da empresa no sistema.

Para dar início ao processo de importação dos dados, o *gerente* deve, antes de mais nada, se *autenticar* no sistema. Na sequência, ele deve *selecionar os arquivos* a serem importados (contendo os dados financeiros da filial ao qual é responsável). Feito isso, é dado início ao *upload dos arquivos*. Ao receber os arquivos, o sistema *inicia o tratamento dos dados* presentes em cada um deles através do processo de transformação das informações: convertendo valores primitivos para os tipos existentes do banco, relacionando coleções, bem como removendo valores nulos. Realizado o tratamento das informações, os dados são processados e o sistema, logo em seguida, *notifica o gerente sobre a finalização do processamento*. Na Figura 13, é apresentado o diagrama com o fluxo de funcionamento da atividade.

Figura 13 – Diagrama de atividades: Importação dos dados financeiros



Fonte: Elaborado pelo autor (2024)

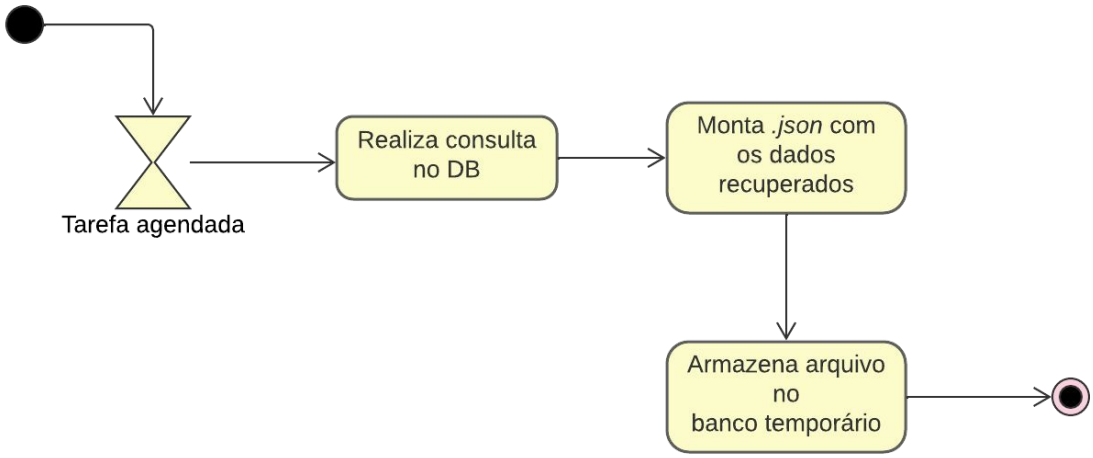
3.3.2 Disponibilização dos Dados Financeiros

Nesta atividade, é apresentado o fluxo de etapas realizado pelo sistema para a disponibilização dos dados financeiros relacionados às receitas e despesas, presentes na *API OLTP* para a *API de Controle Financeiro*.

O processo de disponibilização dos dados financeiros é iniciado ao ser disparada a *tarefa agendada*, que é executada no primeiro dia útil de todo o mês. Com base nas empresas armazenadas no banco de dados, o sistema *realiza uma consulta* para recuperar as informações inerentes às receitas e despesas de cada uma delas, tendo

como parâmetro o mês concluído. Finalizada essa etapa, o sistema *monta um arquivo json* com os dados recuperados, para na sequência *armazená-lo no banco de dados temporário*, a fim de que sejam consultados posteriormente pela *API de Controle Financeiro*. Na Figura 14, é possível visualizar o diagrama com as etapas descritas.

Figura 14 – Diagrama de atividades: Recuperação dos dados financeiros

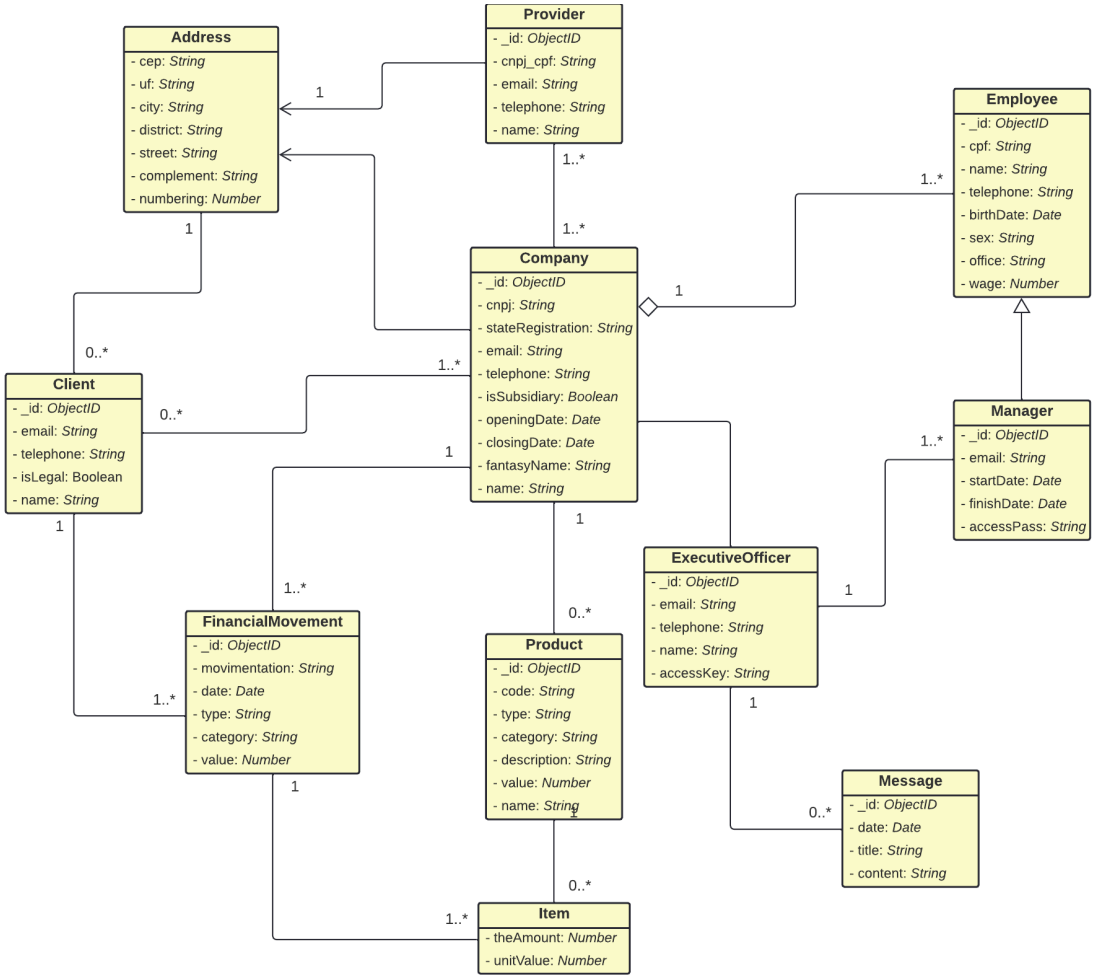


Fonte: Elaborado pelo autor (2024)

3.4 Modelagem do Banco *OLTP*

Esta seção tem como objetivo apresentar e descrever o esquema usado para a implementação do banco de dados *OLTP*. Na Figura 15, é apresentado o diagrama de classes com as entidades que compõem a base e os seus respectivos relacionamentos.

Figura 15 – Modelo lógico do banco *OLTP*



Fonte: Elaborado pelo autor (2024)

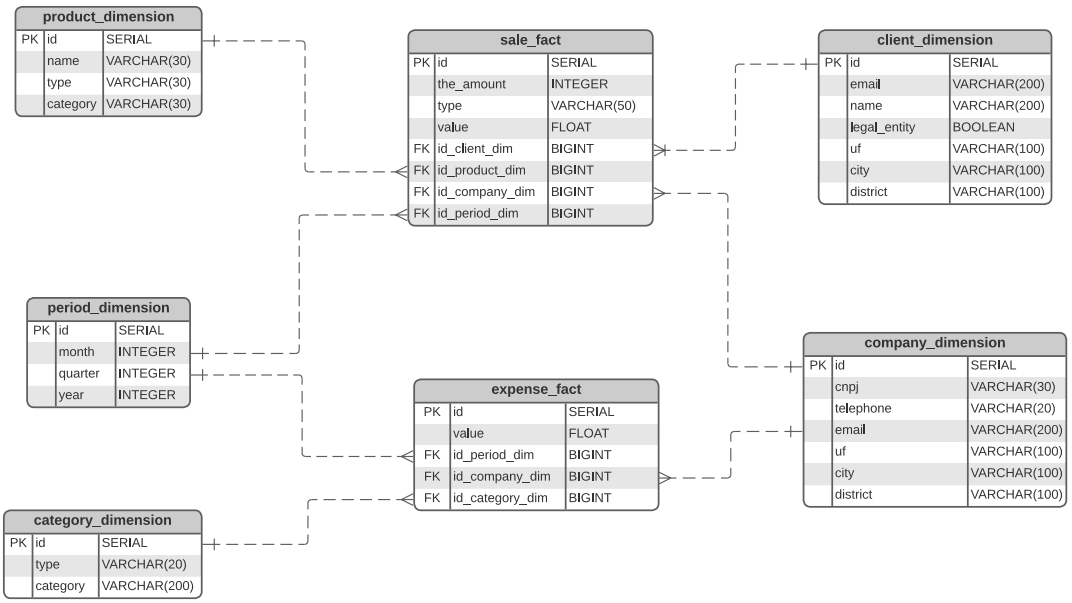
Definidos os atributos de cada uma das classes que compõem o esquema do banco *OLTP*, pode-se observar na Figura 15 a existência do atributo *_id*. Ele representa um atributo padrão gerado pelo próprio SGBD (Sistema de Gerenciamento de Banco de Dados), a fim de realizar o controle interno dos dados presentes na base de dados, tendo em vista que o banco de dados escolhido para a implementação foi o *MongoDB*. A escolha desse banco de dados se deu pela facilidade na manipulação das informações em formato *JSON*, já que o *Mongo* utiliza-se de um formato semelhante para o armazenamento dos dados, o *BSON*, o que permite uma fácil leitura e adaptação de documentos desse tipo.

3.5 Modelagem do *Data Warehouse*

Esta seção apresenta o esquema lógico usado na implementação do *DW*. O esquema, como mostrado na Figura 16 e detalhado no Apêndice C, é baseado no mo-

delo *star schema*. O banco de dados é composto por sete tabelas, sendo duas usadas para armazenar os fatos e as cinco restantes, para as dimensões. Nas tabelas de fato são armazenados os valores mensuráveis das vendas e despesas realizadas por uma determinada empresa, enquanto que nas tabelas de dimensão estão dispostas as características de cada uma delas.

Figura 16 – Modelo lógico do *data warehouse*



Fonte: Elaborado pelo autor (2024)

3.6 Implementação

Esta descreve o processo de implementação do sistema proposto, apresentando uma descrição detalhada dos principais módulos que compõem cada uma das camadas existentes.

3.6.1 Camada *API OLTP*

Esta subseção descreve a implementação da Camada *API OLTP*, construída usando o ambiente de execução *Node.js* com a linguagem de programação *TypeScript* e as bibliotecas *Express* (AZAT, 2014), *GraphQL*, *TypeGraphQL* (LYTEK, 2025) e *Typegoose* (TEAM, 2025).

3.6.1.1 Módulo *Resolvers*

O módulo *Resolvers* tem a tarefa de intermediar a comunicação entre o cliente *web* e o servidor de aplicação *GraphQL*. É por meio dele que são fornecidas as classes

encarregadas de gerenciar as mutações e consultas (vinculadas a cada uma das entidades apresentadas no esquema lógico da Figura 15), processando as solicitações a fim de manipular os dados existentes no banco de dados. No Algoritmo 7, é possível observar a implementação do *resolver CompanyRsl*.

Algoritmo 7 – *Resolver Company*

```
1   import {...}
2
3   @Resolver()
4   export default class CompanyRsl {
5
6       constructor(private companyService: CompanyService){}
7
8       @Mutation(() => Company)
9       public async addCompany(@Arg() company: CompanyInput):
            Promise<Company> {
10              return await this.companyService.add(company);
11      }
12
13      {...}
14  }
```

Fonte: Elaborado pelo autor (2024)

Anotada com o decorador *@Resolver( )*, a classe *CompanyRsl* tem o papel de centralizar todos os métodos relacionados à entidade *Company*, tais como: adição, atualização, exclusão, listagem, etc. Dentro dela, a função membro *addCompany*, por se tratar de um método responsável pela escrita de dados, é anotada com o *@Mutation( )*. Essa mesma função recebe como parâmetro uma variável intitulada *company*, do tipo *CompanyInput*, contendo os dados vindos do *front-end* (por regra, toda e qualquer variável passada como parâmetro deve ser precedida do decorador *@Arg( )* (para que o servidor possa identificá-la como um argumento da requisição e, assim, efetuar a conversão para o seu respectivo tipo). Já no corpo da função é utilizada a variável *companyService*, injetada anteriormente via construtor da classe, com o objetivo de fornecer os métodos responsáveis pelo gerenciamento e aplicação das regras de negócio do sistema.

Tendo em vista que os tipos *Company* e *CompanyInput* não são primitivos da linguagem *GraphQL*, mas sim criados para utilização no sistema, é necessário adaptá-los para que o servidor consiga interpretá-los de forma correta. No Algoritmo 8, é apresentada a estruturação da classe *Company* presente na *API*.

Algoritmo 8 – Classe *Company*

```
import {...}

@ObjectType()
@ModelOptions({ schemaOptions: { versionKey: false } })
export default class Company {

    @Field(() => ID)
    _id: Types.ObjectId;

    @Field()
    @Prop({ unique: true })
    cnpj: string

    @Field()
    @Prop({ unique: true })
    stateRegister: string

    {...}

}
```

Fonte: Elaborado pelo autor (2024)

No código do Algoritmo 8, é possível observar que a classe *Company* apresenta os atributos que constituem a coleção de mesmo nome, com o conjunto de anotações (fornecidas pelas bibliotecas *TypeGraphQL* e *Typegoose*) encarregadas pela conversão da classe em uma coleção do banco de dados, e pela definição dos tipos que deverão ser reconhecidos e aceitos no servidor de aplicação *GraphQL*.

Para a criação dos esquemas no banco de dados *MongoDB* foram utilizados os decoradores *@ModelOptions( )* e *@Prop( )*, ambos disponibilizados pela biblioteca *Typegoose*, que, assim como o JPA, tem o papel de abstrair a criação e gerenciamento das coleções da base de dados via classes. Neles, é possível definir regras inerentes ao esquema e seus atributos. Um exemplo disso é a restrição de unicidade aos atributos *cnpj* e *stateRegister*. Já os decoradores *@ObjectType( )* e *@Field( )*, fornecidos pela biblioteca *TypeGraphQL*, desempenham uma função semelhante às do *Typegoose*, porém voltadas exclusivamente para o mapeamento dos tipos que compõem o

schema.gql.

Por sua vez, no Algoritmo 9 é apresentada a classe *CompanyInput*, que é usada como modelo para o recebimento de dados referentes ao tipo *Company*. Ela segue as mesmas características de configuração da classe apresentada no Algoritmo 8. Porém, como ela se trata do tipo *Input*, no lugar do *@ObjectType( )* é utilizado o *@InputType( )*. Além disso, nela não é necessário o uso das anotações relacionadas à definição das coleções no banco de dados.

Algoritmo 9 – Classe *CompanyInput*

```
1   import {...}
2
3   @InputType()
4   export default class Company {
5
6       @Field()
7       cnpj: string
8
9       @Field()
10      stateRegister: string
11
12      {...}
13
14  }
```

Fonte: Elaborado pelo autor (2024)

3.6.1.2 Módulo *Services*

O módulo *Services* é responsável por definir o conjunto de classes encarregadas de aplicar as regras de negócio do sistema aos dados recebidos ou retornados ao cliente. É nesse módulo também que é realizada a comunicação da aplicação com os serviços externos utilizados pelo próprio sistema. Além disso, ela funciona como um elo entre os módulos *Resolvers* e *DAOs*. No Algoritmo 10, é possível ver a estruturação do método *delete* da classe *CompanyService*.

Algoritmo 10 – Classe *CompanyService*

```
1  import {...}
2
3  export default class CompanyService {
4
5      constructor(
6          private companyDAO: CompanyDAO,
7          private clientsDAO: ClientDAO
8      ){}
9
10     {...}
11
12     public async delete(id: Types.ObjectId): Promise<boolean>
           {
13           const company = await this.companyDAO.readById(id);
14
15           if(!company.isSubsidiary) return false;
16
17           company.clients.forEach(async cl => {
18             await this.clientDAO.remove(cl?._id);
19           });
20
21         {...}
22
23           return (await this.companyDAO.remove(id)) != null;
24     }
25
26     {...}
27
28 }
```

Fonte: Elaborado pelo autor (2024)

Tal como em outros métodos de mesma finalidade, o método *delete* efetua a exclusão de um determinado documento, tendo como base um *id* informado. Entretanto, esse método possui algumas particularidades, dentre as quais está o fato de que, antes mesmo da exclusão do documento ser consumada, é necessário realizar uma consulta com o objetivo de verificar se a empresa é ou não uma subsidiária, já que, pelas regras de negócio, não é permitido remover a empresa matriz.

Por fim, tendo em vista a existência de relacionamentos entre a entidade *Company* e outras coleções do sistema, o método remove os documentos que possuem

relacionamento direto com a empresa especificada.

3.6.1.3 Módulo *DAOs*

O módulo *DAOs* é formado pelas classes que realizam a manipulação (leitura e escrita) dos dados nos servidores de banco de dados *MongoDB* e *Redis*. No Algoritmo 11 é apresentada parcialmente a classe responsável pela manipulação dos dados relacionados à entidade *Company*.

Algoritmo 11 – *DAO Company*

```
1   import {...}
2
3   export default class CompanyDAO {
4
5       public async save(company: Company): Promise<Company> {
6           const companyModel = getModelForClass(Company);
7           await mongoConnection;
8           return companyModel.create(company);
9       }
10
11      {...}
12
13  }
```

Fonte: Elaborado pelo autor (2024)

No Algoritmo 11 é apresentado, de forma parcial, a implementação da classe *CompanyDAO* (responsável pela conexão direta da aplicação com a coleção *Company* do banco de dados), juntamente com o método encarregado por realizar a escrita de informações no banco.

3.6.2 Camada *API* de Controle Financeiro

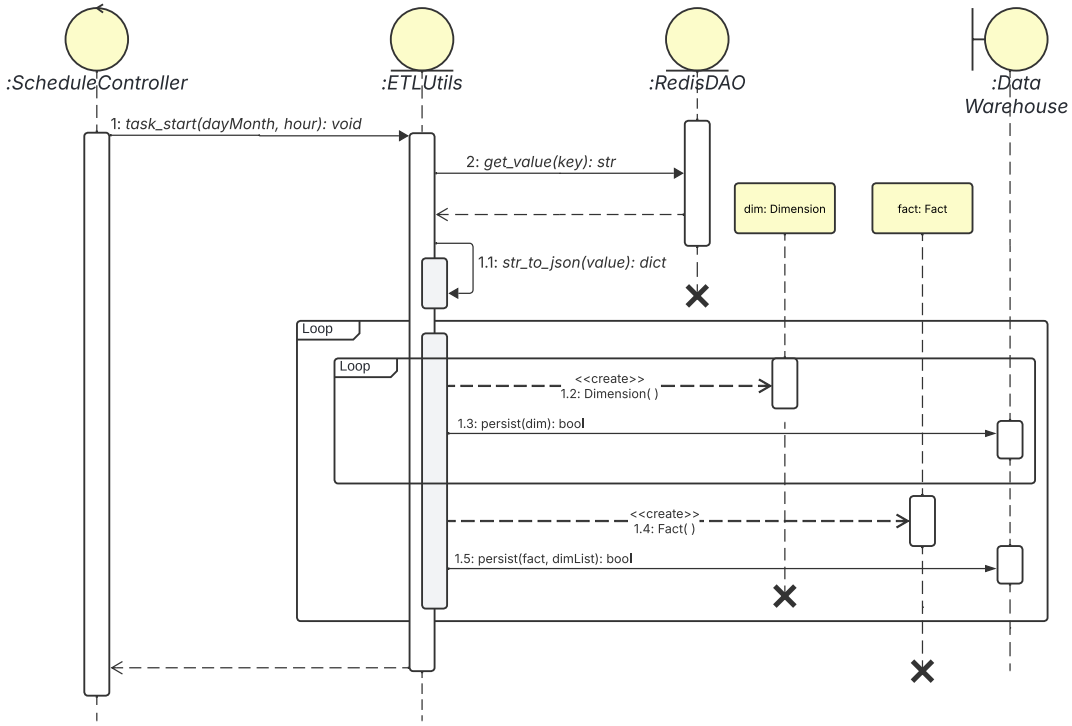
Esta subseção descreve a implementação da Camada *API* de Controle Financeiro, tendo como foco os módulos Ferramenta *ETL* e *Anomaly Detection*. Essa camada foi desenvolvida utilizando-se a linguagem de programação *Python*, juntamente com o *microframework Flask* (GRINBERG, 2018) e as bibliotecas *Scikit-Learn* (FABIAN et al., 2011), *SQLAlchemy* (MYERS; COPELAND, 2015) e *MiniSom* (VETTIGLI, 2018).

3.6.2.1 Módulo Ferramenta *ETL*

Conforme descrito na Seção 2.4, o papel da ferramenta *ETL* é realizar a extração, transformação e carregamento das informações financeiras existentes no banco de dados da camada *API OLTP* para o *Data Warehouse*, tendo como base o esquema apresentado na Figura 16. Ao fim desse processo, os dados recuperados deverão se adequar às relações existentes no banco de dados do *Data Warehouse*, a fim de manter a sua integridade e consistência ao longo de cada uma dessas etapas.

A ferramenta efetua o tratamento das informações por meio de arquivos no formato *.json*, que são gerados e disponibilizados pela *API OLTP*. Nesse processo é aplicada a abordagem *bottom-up* (de baixo para cima), com foco, em um primeiro momento, na inserção das tabelas de dimensões, para, na sequência, efetuar a persistência dos fatos. O diagrama de sequência da Figura 17 apresenta o fluxo das interações por trás da persistência dos dados.

Figura 17 – Diagrama de sequência: Interação dos componentes responsáveis pela persistência de dados no *DW*



Fonte: Elaborado pelo autor (2025)

O diagrama apresentado na Figura 17 mostra a interação entre os diferentes componentes (controladores, entidades, banco, etc) encarregados pelo processo de persistência dos dados financeiros obtidos para o *DW*. O processo tem início quando a tarefa agendada, presente na classe *ScheduleController*, efetua a chamada do método

task_start, da classe *ETLUtils*, que recebe como parâmetros o dia do mês e a hora do acionamento.

Recebidos os valores repassados pela classe *ScheduleController*, o método os utiliza com a finalidade de montar a chave necessária para a recuperação dos dados, que são armazenados previamente pela *API OLTP* no servidor de banco *Redis*. Depois que os dados são recuperados no banco de dados, é dado início à sua conversão para o padrão *JSON* (*dict* em *Python*), tendo em vista que os dados presentes na base de dados encontram-se no formato de *String*.

Finalizada a conversão da etapa anterior, e estando os dados no padrão necessário para manipulá-los, dá-se sequência ao processo de persistência das informações no *Data Warehouse*. Na primeira parte desta tarefa é feita a criação de uma lista contendo todas as dimensões relacionadas ao fato que deverá ser armazenado no banco de dados. Gerada a lista com as dimensões já criadas, o sistema passa para a geração da tupla do fato propriamente dito, passando como parâmetros para o método o fato, juntamente com a sua lista de dimensões.

Na Figura 18, é possível visualizar um dos conjuntos de dados recuperados pelo módulo *ETL* e que é usado no povoamento da base de dados.

Figura 18 – Conjunto de dados recuperado pelo módulo *ETL*


```
{
   "_id": "642ca9dc4108e4dd6e8dde25",
   "identifierType": "VENDA",
   "type": null,
   "category": "À PRAZO",
   "date": "2023-03-01T22:50:52.000Z",
   "value": 2439,
   "client": {
      "email": "lojao.construcao@gmail.com",
      "name": "Lojão da Construção",
      "isLegal": true,
      "address": {
         "uf": "CE",
         "city": "Caucaia",
         "district": "Grilo"
      }
   },
   "product": {
      "name": "Churrasqueira Sol Nascente",
      "type": "CAPITAL",
      "category": "SIMPLES",
      "items": [
         {
            "theAmount": 3,
            "movement": {
               "_id": "642ca9dc4108e4dd6e8dde25"
            }
         },
         {
            "theAmount": 4,
            "movement": {
               "_id": "6447321fc70ccb435e379461"
            }
         },
         {...}
      ]
   }
}
```


Fonte: Elaborado pelo autor (2025)

Na Figura 18 é possível observar algumas das informações que compõem as movimentações financeiras realizadas em uma determinada empresa e disponibilizadas via *API OLTP* no formato *JSON*. Dentre os dados presentes no arquivo está o atributo *date*, que contém a data e hora referentes à movimentação financeira e que, após ser aplicado o processo de transformação da ferramenta, dará origem à dimensão *period_dimension*, conforme apresentado na Figura 16.

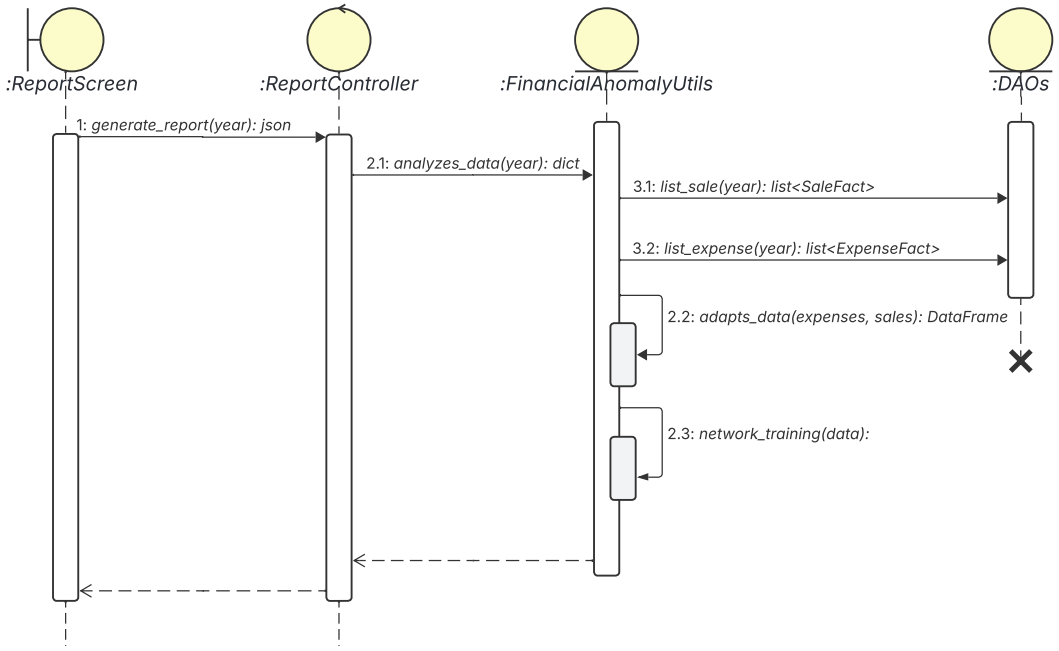
O mesmo processo de transformação, aplicado à dimensão de períodos, ocorre com os demais dados presentes no arquivo, de forma sequencial, sem que haja, assim, a necessidade de alocação de múltiplas *threads* pelo sistema, mas sim apenas o

gerenciamento nativo da fila, já que tal processo ocorre no momento de ociosidade da aplicação.

3.6.2.2 Módulo *Anomaly Detection*

O módulo *Anomaly Detection* tem a tarefa de, com base nos dados presentes no *DW*, realizar a auditoria financeira por meio de análises e buscas, a fim de detectar possíveis discrepâncias nas informações existentes. O módulo é formado pela classe *FinancialAnomalyUtils* e seus respectivos métodos, encarregados da manipulação e análise financeira. Na Figura 19, é possível ver o fluxo e a interação dos componentes responsáveis pelo funcionamento da detecção de anomalias financeiras.

Figura 19 – Diagrama de sequência: Interação dos componentes responsáveis pela análise e detecção de anomalias financeiras



Fonte: Elaborado pelo autor (2025)

Na Figura 19 pode-se ver a interação entre os componentes envolvidos na análise e auditoria dos dados financeiros, sendo eles: *ReportScreen*, *ReportController*, *FinancialAnomalyUtils* e *DAOs*. O primeiro componente, *ReportScreen*, é a tela por onde o usuário do sistema (*diretor-executivo*) realiza a solicitação para a geração do relatório com o parecer e a auditoria das contas. A requisição é recebida pelo componente *ReportController*, que chama o método *analyzes_data*, passando o ano base da análise, para que assim seja iniciado o processo de auditoria dos dados.

É no componente *FinancialAnomalyUtils* que é efetuada a operação de auditoria das contas. Tendo como base o ano recebido, o sistema instancia os componentes

DAOs correspondentes às entidades *ExpenseFact* e *SaleFact* para a listagem dos seus valores e dimensões. Recuperados os valores, a aplicação os adapta para uma estrutura denominada *DataFrame*, com as informações que deverão ser usadas na análise, tendo essas sido definidas previamente. Feito este processo, é iniciado o treinamento da rede neural SOM (com regras específicas e adaptadas exclusivamente ao ambiente da aplicação), utilizando os dados existentes na estrutura criada com o *DataFrame*.

Finalizado o processo de treinamento da rede para auditoria dos dados financeiros, é gerada uma matriz de resultados, como mostrado na Figura 20, com os valores de cada índice variando entre os intervalos de 0 à 1 - sendo que, quanto mais próximo o valor estiver de 1, maior a probabilidade de o índice ter registros com dados financeiros discrepantes. Para cada célula da matriz há uma lista de registros com os dados financeiros.

Figura 20 – Matriz de resultado da análise financeira

	0	1	2	3	4	5	6	7	8	9	10
0	0.16947	0.20057	0.42672	0.54829	0.29536	0.17600	0.20749	0.32624	0.37966	0.65666	0.20243
1	0.36754	0.88112	0.47266	0.48193	0.39767	0.49489	0.25559	0.35490	0.60344	0.61327	0.44829
2	0.26167	0.44424	0.52322	0.42312	0.33105	0.34165	0.30481	0.35061	0.41902	1.00000	0.30540
3	0.27896	0.33484	0.54965	0.45647	0.26616	0.28567	0.29264	0.37090	0.61485	0.41879	0.27504
4	0.29673	0.48955	0.55153	0.61758	0.34559	0.31271	0.36794	0.28161	0.29862	0.45100	0.32848
5	0.28057	0.44822	0.49531	0.35069	0.40984	0.46418	0.37347	0.57936	0.24450	0.32523	0.22810
6	0.21109	0.50107	0.39292	0.48404	0.53740	0.23531	0.30729	0.43924	0.42166	0.37629	0.15055
7	0.49382	0.51316	0.38595	0.47159	0.39716	0.26474	0.16844	0.23052	0.47277	0.32688	0.13932
8	0.41399	0.59838	0.91969	0.52689	0.55998	0.26491	0.23101	0.15849	0.36212	0.77566	0.22558
9	0.25676	0.55831	0.65835	0.74961	0.62531	0.50665	0.49761	0.38036	0.42886	0.61749	0.25624
10	0.12937	0.41514	0.36161	0.37915	0.28443	0.38113	0.27129	0.52267	0.31236	0.16200	0.06835

Fonte: Elaborado pelo autor (2025)

Ao final de todo o processamento, caso sejam detectados registros que venham apresentar inconsistências, o sistema monta um arquivo *JSON* contendo os dados básicos sobre cada caso e a empresa à qual os mesmos estão vinculados, para que possam ser apresentados no formato de PDF ao usuário final.

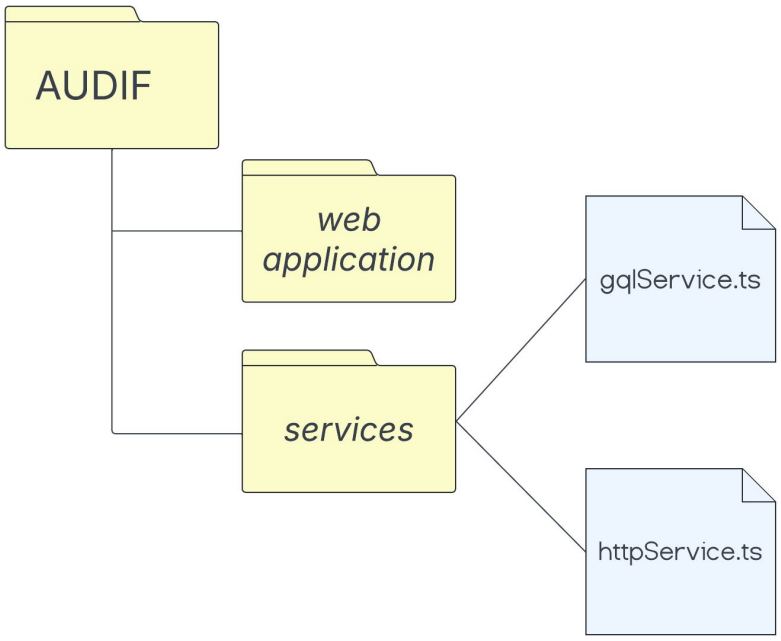
3.6.3 Camada de Apresentação

Esta subseção descreve a implementação da Camada de Apresentação do sistema. A camada foi construída utilizando-se o ambiente *Node.js* e com as bibliotecas *React.js*, *Typescript*, *Apollo Client* (GREBE, 2020), *Recharts* (RECHARTS, s.d.) e *Axios* (KOLCE et al., 2020).

3.6.3.1 Módulo *Services*

O módulo *Services* tem a tarefa de realizar a comunicação entre a camada de Apresentação e as demais camadas do sistema (*API OLTP* e *API de Controle Financeiro*), bem como os serviços que venham a ser utilizados pelo *front-end* da aplicação. A estruturação do módulo se dá por meio de dois arquivos, sendo cada um deles responsável por conectar e gerenciar o tipo de requisição a ser realizado pelo sistema, tendo como base a funcionalidade utilizada. A Figura 21 apresenta a estrutura de organização do *front-end*, com destaque ao módulo *Services*.

Figura 21 – Estrutura de organização do *front-end*



Fonte: Elaborado pelo autor (2025)

Como ilustrado na Figura 21, a camada é organizada em dois pacotes (módulos), sendo o primeiro denominado *Web Application* (onde encontram-se as telas do sistema, componentes, rotas, etc.) e *Services*, onde estão os arquivos responsáveis por servir de ponte entre o *front-end*, as *APIs* do sistema e o serviço *web*. Os arquivos estão estruturados com base no modelo de requisições implementados pelos respectivos *back-ends*, ou seja, *HTTPs* e *GraphQL*.

O primeiro arquivo, denominado de *gqlService.ts*, tem a atribuição de gerenciar as consultas e mutações relativas à manipulação dos dados que utilizam o padrão *GraphQL*. No Algoritmo 12 é apresentada a estrutura de uma das várias consultas existentes no arquivo e empregada pela aplicação.

Algoritmo 12 – *Query* de listagem das empresas em *gqlService.ts*

```
import { gql } from '@apollo/client';

export const COMPANY_LIST = gql`
       query {
              listCompany {
                     _id
                     cnpj
                     fantasyName
                     address {
                            uf
                            city
                     }
              }
       }

{...}
```

Fonte: Elaborado pelo autor (2025)

No Algoritmo 12 é possível visualizar a definição da variável *LIST_COMPANY*, usada na listagem das empresas existentes no sistema, com alguns dos atributos que compõem a entidade e devem ser retornados na resposta. Para a criação dessa variável foi realizada a importação do tipo *gql*, presente na biblioteca *Apollo*, e obrigatório na escrita das consultas que usam *GraphQL*. No Algoritmo 13 pode-se observar a utilização da variável na tela *Company*, do perfil *diretor-executivo*.

Algoritmo 13 – Utilização da variável *LIST_COMPANY*

```
import {...}
import { useQuery, useMutation } from '@apollo/client';

const Company: React.FC = () => {
    const queryListCompanies = useQuery(COMPANY_LIST);
    const [companies, setCompanies] = useState<any[]>([]);
    {...}

    useEffect(() => {
            setCompanies(queryListCompanies.data?.listCompany);
        }, [queryListCompanies.data?.listCompany]);

        {...}

}

export default Company;
```

Fonte: Elaborado pelo autor (2025)

Além da importação da própria variável utilizada para a listagem das empresas, pode-se ver a importação da função *useQuery*, que recebe como parâmetros textos do tipo *gql*. Essa função é responsável por recuperar, no servidor de aplicação, a consulta correspondente a que foi definida no *schema.gql*. A operação de listagem é iniciada no momento de renderização da página, tendo em vista o uso da função dentro do método *useEffect* do *React*, com o valor obtido sendo alterado na variável de estado *companies*.

Já o arquivo *httpService.ts* é encarregado de estabelecer as conexões com o serviço *web ViaCEP*[8] e a *API* de Controle Financeiro, visto que eles são implementados utilizando o padrão *HTTP* puro. O Algoritmo 14 mostra a configuração de conexão para ambos serviços.

[8]Consulta de informações com base no CEP. https://viacep.com.br/. Acesso em: 23 mar. 2025

Algoritmo 14 – Configuração de conexões no arquivo *httpService.ts*

```
1  import axios from 'axios'
2
3  const financialAPI =
       axios.create({baseURL:`${process.env.REACT_APP_URL_FINANCIAL}`});
4
5  const viacepAPI =
       axios.create({baseURL:`${process.env.REACT_APP_URL_CEP}`});
6
7  export { financialAPI, viacepAPI }
```

Fonte: Elaborado pelo autor (2025)

No Algoritmo 14, nota-se a configuração necessária para realizar a conexão do *front-end web* com o serviço *web* e a *API* desenvolvida. Para que fosse possível tal comunicação, o uso da biblioteca *Axios* foi essencial, tendo em vista a abstração fornecida por ela na hora de estabelecer as conexões, por meio de uma *URL* respectiva para cada aplicação.

O Algoritmo 15 mostra a utilização da variável de conexão do serviço *web viaCEP* em uma função que recebe como parâmetro um determinado CEP para, na sequência, buscar os seus dados complementares.

Algoritmo 15 – Utilização da conexão *ViaCEP*

```
1  import {...}
2
3  {...}
4
5  function getDataAddress(cep: string): AddressI {
6      return viacepAPI.get(`/${cep.replace('-', '')}/json/`)
7                          .then(resp => { return resp.data as
                               AddressI });
8  }
9
10 {...}
```

Fonte: Elaborado pelo autor (2025)

3.6.3.2 Módulo *Aplicação Web*

É no módulo de Aplicação *Web* que se encontram as páginas da aplicação responsáveis pela exibição e manipulação dos dados de forma interativa para os usuários

finais do sistema.

O módulo é composto pelo conjunto de páginas correspondentes aos dois perfis existentes no sistema (*diretor-executivo* e *gerente*), com cada um contendo o conjunto de funcionalidades inerentes ao papel desempenhado por ele na empresa. A aplicação conta com um total de sete páginas, que variam entre públicas e privadas, sendo tal número um reflexo direto do emprego do *React.js*, que permitiu a reutilização de muitos dos componentes criados em diferentes telas. A Figura 22 mostra a tela inicial do sistema.

Figura 22 – Tela inicial do sistema

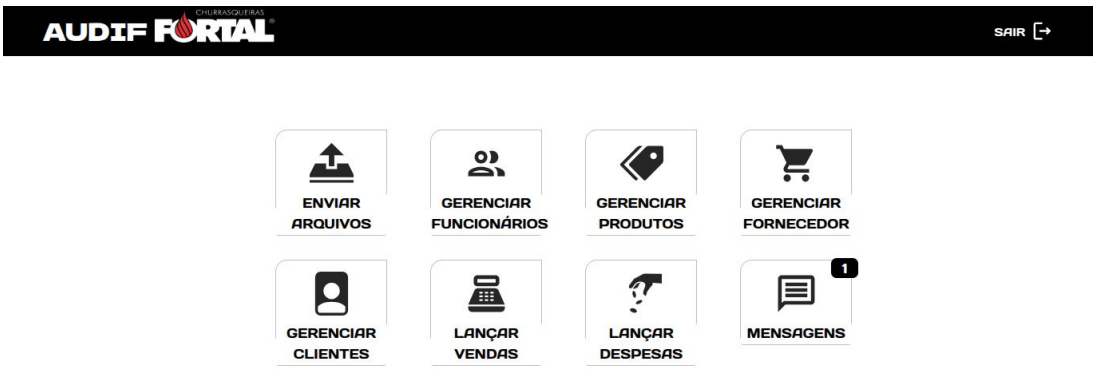
Fonte: Elaborado pelo autor (2025)

Na Figura 22 pode-se ver a tela inicial do sistema, formada pelo formulário que dá acesso aos perfis e duas abas, sendo a primeira nomeada *Sobre*, que fornece uma breve explicação sobre o sistema e uma descrição de algumas funcionalidades que ele disponibiliza, e a segunda *Contato*, que exibe os dados para que o usuário possa entrar em contato com o desenvolvedor ou com o responsável geral pelo sistema.

Depois que o usuário preenche o formulário com suas credenciais e efetua a sua autenticação, com base nas credenciais informadas, o sistema o redireciona para a página correspondente ao seu perfil.

Caso o usuário informado venha a ser do tipo *gerente*, o sistema renderiza e exibe a tela apresentada na Figura 23. Nela, pode-se ver o menu do perfil *gerente* com o seu respectivo conjunto de funcionalidades, referentes ao gerenciamento das informações da empresa matriz ou filial pela qual o usuário é responsável, com tarefas que vão desde o gerenciamento dos dados dos funcionários alocados na empresa até o lançamento de despesas e vendas.

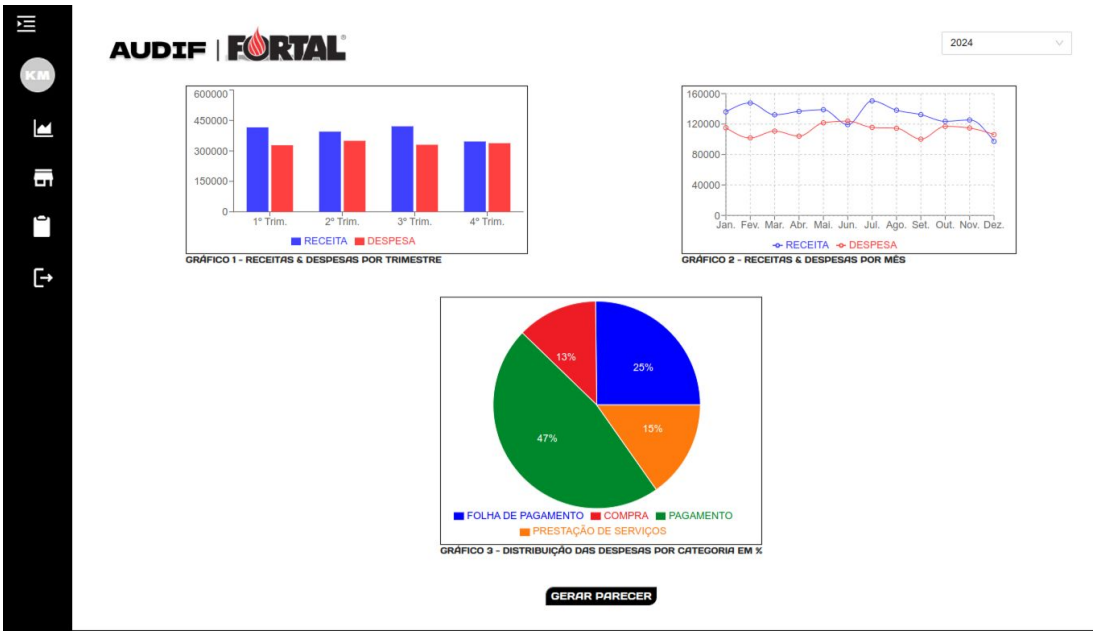
Figura 23 – Tela do perfil *gerente*



Fonte: Elaborado pelo autor (2025)

Por outro lado, caso o usuário seja o *diretor-executivo* da empresa, ele será redirecionado para a tela apresentada na Figura 24. Assim como no caso do perfil de *gerente*, essa tela é composta pelas funcionalidades inerentes às funções desempenhadas pelo usuário *diretor-executivo*, com foco principal no acompanhamento e visualização das despesas e receitas, dispostas através de gráficos renderizados dinamicamente, com base no ano selecionado, por meio do componente *select* presente na parte superior direita da tela, e auxílio da biblioteca *Recharts*.

Figura 24 – Tela do perfil *diretor-executivo*

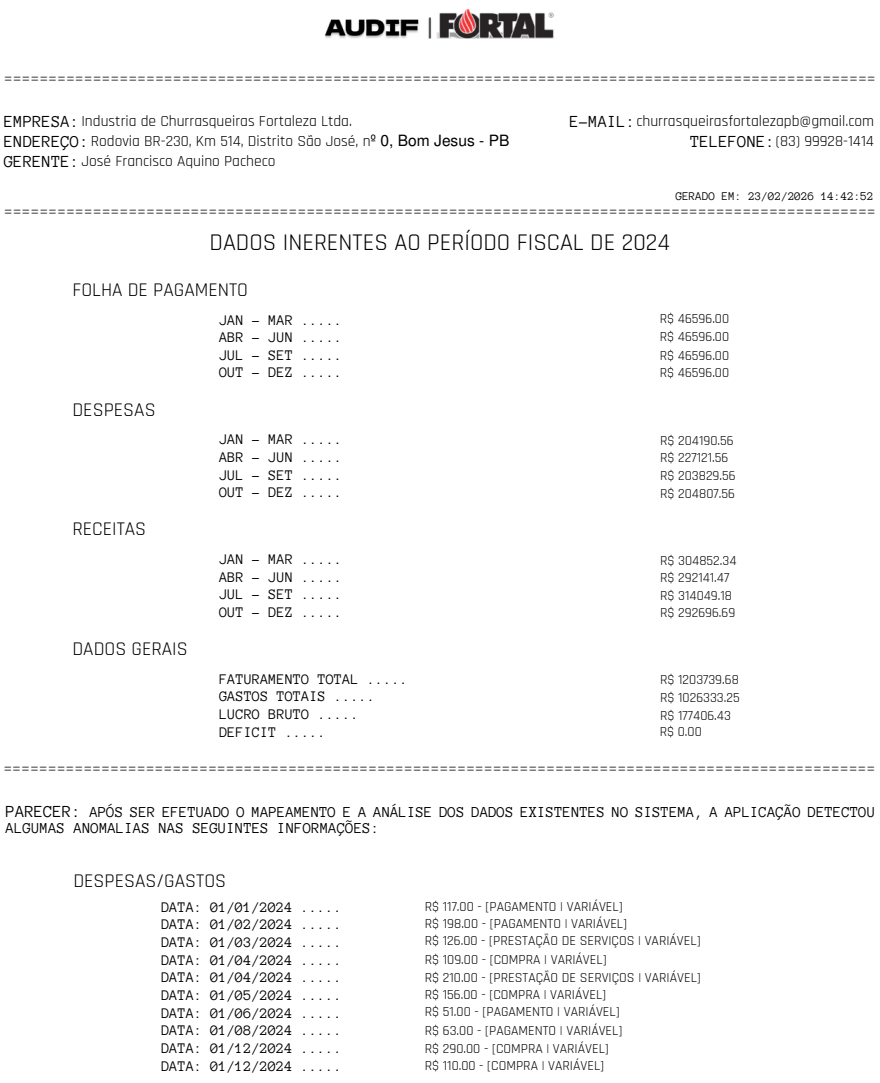


Fonte: Elaborado pelo autor (2025)

Por fim, considerando que o objetivo final do sistema é a auditoria financeira das contas, levando-se em consideração um determinado período, quando o diretor-executivo

seleciona um ano e clica no botão GERAR PARECER, é gerado um relatório - com as despesas e receitas do ano especificado. Caso venham a existir gastos considerados imprecisos ou que estejam em desconformidade com os outros existentes, eles serão informados via relatório. A página correspondente ao relatório gerado ao fim desse processo é mostrada na Figura 25.

Figura 25 – Parecer do ano fiscal de 2024



AUDIF | FORTAL

==========

EMPRESA: Industria de Churrasqueiras Fortaleza Ltda.
ENDEREÇO: Rodovia BR-230, Km 514, Distrito São José, nº 0, Bom Jesus - PB
GERENTE: José Francisco Aquino Pacheco

E-MAIL: churrasqueirasfortalezapb@gmail.com
TELEFONE: (83) 99928-1414

GERADO EM: 23/02/2026 14:42:52

==========

DADOS INERENTES AO PERÍODO FISCAL DE 2024

FOLHA DE PAGAMENTO

JAN – MAR	R$ 46596.00
ABR – JUN	R$ 46596.00
JUL – SET	R$ 46596.00
OUT – DEZ	R$ 46596.00

DESPESAS

JAN – MAR	R$ 204190.56
ABR – JUN	R$ 227121.56
JUL – SET	R$ 203829.56
OUT – DEZ	R$ 204807.56

RECEITAS

JAN – MAR	R$ 304852.34
ABR – JUN	R$ 292141.47
JUL – SET	R$ 314049.18
OUT – DEZ	R$ 292696.69

DADOS GERAIS

FATURAMENTO TOTAL	R$ 1203739.68
GASTOS TOTAIS	R$ 1026333.25
LUCRO BRUTO	R$ 177406.43
DEFICIT	R$ 0.00

==========

PARECER: APÓS SER EFETUADO O MAPEAMENTO E A ANÁLISE DOS DADOS EXISTENTES NO SISTEMA, A APLICAÇÃO DETECTOU ALGUMAS ANOMALIAS NAS SEGUINTES INFORMAÇÕES:

DESPESAS/GASTOS

DATA: 01/01/2024	R$ 117.00 - [PAGAMENTO I VARIÁVEL]
DATA: 01/02/2024	R$ 198.00 - [PAGAMENTO I VARIÁVEL]
DATA: 01/03/2024	R$ 126.00 - [PRESTAÇÃO DE SERVIÇOS I VARIÁVEL]
DATA: 01/04/2024	R$ 109.00 - [COMPRA I VARIÁVEL]
DATA: 01/04/2024	R$ 210.00 - [PRESTAÇÃO DE SERVIÇOS I VARIÁVEL]
DATA: 01/05/2024	R$ 156.00 - [COMPRA I VARIÁVEL]
DATA: 01/06/2024	R$ 51.00 - [PAGAMENTO I VARIÁVEL]
DATA: 01/08/2024	R$ 63.00 - [PAGAMENTO I VARIÁVEL]
DATA: 01/12/2024	R$ 290.00 - [COMPRA I VARIÁVEL]
DATA: 01/12/2024	R$ 110.00 - [COMPRA I VARIÁVEL]

Fonte: Elaborado pelo autor (2025)

Como observado na Figura 25, o relatório final é estruturado em quatro eixos principais: Folha de Pagamento, Despesas, Receitas - sendo estes representados de maneira trimestral - e, por fim, Dados Gerais com as informações consolidadas dos dados existentes até aquele momento. O documento é finalizado com os gastos considerados suspeitos, tendo em vista os padrões e a calibragem definidos para o sistema (indicando a falta de padronização e opacidade destes em relação aos existentes), informando a data de efetivação destes, os valores, o tipo e a categoria.

4 CONSIDERAÇÕES FINAIS

Com a redução da burocracia nos últimos anos, a facilidade em se abrir uma empresa no Brasil fez com que o número de pessoas jurídicas saltasse de 4,5 milhões, em 2014, para 22 milhões em 2024. Desse total de negócios ativos, 93,4% são categorizados como Micro e Pequenas Empresas. Em números absolutos, ao se analisar a totalidade de empresas que fecharam suas portas no país, as MPEs correspondem a uma fatia significativa no cálculo final, se comparada às Médias e Grandes Empresas.

Um dos aspectos mais preocupantes que ajudam a explicar esses números é a má gestão financeira dos recursos da empresa, que acaba refletindo diretamente no fluxo de caixa e comprometendo, por consequência, áreas estratégicas do empreendimento, dificultando-o atingir a meta de *breakeven*[9]. Tal fator, alinhado ao alto custo na contratação de firmas especializadas que ajudem a apontar os erros, auditar as contas e otimizar os lucros e gastos, contribuem para acelerar o processo de encerramento destes negócios.

Com o objetivo de fornecer uma alternativa de baixo custo capaz de auxiliar as MPEs na auditoria de suas contas, a fim de evitar problemas relacionados a má gestão financeira e acompanhar de forma contínua o desempenho do negócio, foi desenvolvido o sistema *AUDIF* que, apoiado em um módulo *OLTP* e um *data warehouse*, faz a leitura e análise de dados financeiros. Algumas das funcionalidades providas por esse sistema são: o gerenciamento de informações do dia a dia da empresa, o acompanhamento e gestão das receitas e despesas, a leitura de arquivos contendo dados financeiros e a geração de um parecer com auditoria das contas da matriz e filiais. Dados os objetivos definidos para o projeto, é possível afirmar que este trabalho conseguiu alcançá-los, tendo, ao final de sua execução, entregue os seguintes artefatos: o Banco e a *API OLTP*, o *Data Warehouse*, a *API RESTful* para o gerenciamento dos dados financeiros, juntamente com o *Módulo de Detecção de Anomalias Financeiras* e a *Interface Web*.

Com a tarefa de aprimorar continuamente o sistema, alguns trabalhos futuros podem ser desenvolvidos como, por exemplo: o suporte ao processamento de novos tipos de arquivos de dados financeiros, como documentos xml e pdf; o desenvolvimento, integração e utilização de um modelo de RNA supervisionado; o desenvolvimento de uma aplicação mobile, para dar maior flexibilidade ao lançamento de receitas e despesas ao usuário gerente; criação de um módulo voltado para MEIs; e a geração de relatórios mais específicos sobre determinados aspectos financeiros.

[9] Ou “ponto de equilíbrio”, é um termo usado no meio empresarial para informar quando os ganhos de uma empresa se igualam aos custos

REFERÊNCIAS

ALI, A. *Self Organizing Maps(SOM) with Practical Implementation*. 2019. Medium. Acesso em: 13 out. 2024. Disponível em: <https://medium.com/machine-learning-researcher/self-organizing-map-som-c296561e2117>.

ANTONELLI, R. Conhecendo o business intelligence (bi): Uma ferramenta de auxílio à tomada de decisão. *Revista TECAP*, v. 3, 01 2009. Disponível em: <http://revistas.utfpr.edu.br/pb/index.php/CAP/article/viewFile/933/544>.

ANTONIO, Q.-M. et al. Graphql: A systematic mapping study. *ACM Computing Surveys*, v. 55, n. 10, p. 1 – 35, 2023. Acesso em: 23 out. 2025. Disponível em: <https://dl.acm.org/doi/full/10.1145/3561818>.

AWS. *O que é API RESTful?* s.d. Acesso em: 26 mai. 2023. Disponível em: <https://aws.amazon.com/pt/what-is/restful-api/>.

AWS. *What is NoSQL?* s.d. Acesso em: 27 set. 2024. Disponível em: <https://aws.amazon.com/nosql/>.

AZAT, M. *Pro Express.js: Master Express.js: The Node.js Framework for Your Web Development*. 1ª. ed. Berkeley, CA: Apress/Springer, 2014.

CARVALHO, T. *API REST: o que é e quais são as vantagens dessa integração?* 2022. Acesso em: 26 mai. 2023. Disponível em: <https://www.iugu.com/blog/api-rest-o-que-e>.

COSTA, G.; DUTRA, T. Auditoria financeira na era do big data: novas possibilidades para avaliação e resposta a riscos em demonstrações financeiras do governo federal. *Revista do TCU*, v. 131, p. 54–61, 01 2014. Disponível em: <https://revista.tcu.gov.br/ojs/index.php/RTCU/article/view/62/70>.

DANTAS, S.; MATOS, N.; PINHO, R. *Self-organizing maps*. 2020. LAMFO. Acesso em: 07 ago. 2024. Disponível em: <https://lamfo-unb.github.io/2020/08/29/Self-organizing-maps/>.

DESENVOLVEDOR, C. do. *API GRAPHQL: O que é e como funciona?* 2024. Acesso em: 18 out. 2024. Disponível em: <https://blog.casadodesenvolvedor.com.br/api-graphql/>.

EXPERIAN, S. *Ir à falência: veja os principais motivos e como evitá-los*. 2020. Serasa. Acesso em: 22 abr. 2021. Disponível em: <https://empresas.serasaexperian.com.br/blog/ir-a-falencia-veja-os-principais-motivos-e-como-evita-los/>.

FABIAN, P. et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Disponível em: <https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>.

FEDERAL, G. *Mapa de Empresas: boletim 3º quadrimestre de 2024*. 2025. Gov.br. Acesso em: 25 jan. 2025. Disponível em: <https://www.gov.br/empresas-e-negocios/pt-br/mapa-de-empresas/boletins/boletim-do-mapa-de-empresas-3o-quad-2024.pdf>.

FEDOSEJEV, A. *React.js Essentials: a fast-paced guide to designing and building scalable and maintainable web apps with React.js*. Birmingham - Mumbai: Packt Publishing, 2015.

FERREIRA, R. *REST: Princípios e boas práticas*. 2017. Alura. Acesso em: 25 mai. 2023. Disponível em: <https://www.alura.com.br/artigos/rest-principios-e-boas-praticas>.

FURTADO, M. *Redes Neurais Artificiais: Uma Abordagem Para Sala de Aula*. 1ª. ed. Paraná: Atena Editora, 2019.

GEEKFORGEEKS. *Star Schema in Data Warehouse modeling*. 2023. Acesso em: 28 ago. 2023. Disponível em: <https://www.geeksforgeeks.org/star-schema-in-data-warehouse-modeling/>.

GREBE, S. Integrating the apollo client. In: *Full-Stack Web Development with GraphQL and React: Taking React from frontend to full-stack with GraphQL and Apollo*. Birmingham: Packt Publishing, 2020. cap. 10.

GRINBERG, M. *Desenvolvimento Web com Flask: Desenvolvendo Aplicações Web com Python*. 1ª. ed. São Paulo, SP: Novatec, 2018.

GRüBLER, M. *Entendendo o funcionamento de uma Rede Neural Artificial*. 2018. Medium. Acesso em: 09 jan. 2025. Disponível em: <https://medium.com/brasil-ai/entendendo-o-funcionamento-de-uma-rede-neural-artificial-4463fcf44dd0>.

INMON, W. H. *Building the Data Warehouse*. 4ª. ed. Indiana: John Wiley & Sons Inc, 2005.

JAVASCRIPT, S. of. *Front-end Frameworks*. 2025. StateofJS. Acesso em: 10 jan. 2025. Disponível em: <https://2024.stateofjs.com/en-US/libraries/front-end-frameworks/>.

JUNQUEIRA, L.; CARNEIRO, J.; ABRAHAMSOHN, P. *Histologia Básica: Texto e Atlas*. 13. ed. Rio de Janeiro: Guanabara Koogan, 2017.

KIMBALL, R.; ROSS, M. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. 3ª. ed. Indiana: John Wiley & Sons Inc, 2013.

KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, Springer, v. 43, n. 1, p. 59–69, 1982. Disponível em: <https://link.springer.com/article/10.1007/BF00337288>.

KOKAY, M. Banco de dados nosql: Um novo paradigma. *Revista SQL Magazine*, v. 102, 2012. Disponível em: <https://www.devmedia.com.br/banco-de-dados-nosql-um-novo-paradigma-revista-sql-magazine-102/25918>.

KOLCE, J. et al. Making http requests. In: *Modern JavaScript Tools & Skills*. Sebastopol, CA: O'Reilly Media, 2020. cap. 4.

LYTEK, M. *TypeGraphQL: Modern framework for GraphQL API in Node.js*. 2025. <https://typegraphql.com/>. Acesso em 29 de Outubro de 2025.

MACHADO, F. *Tecnologia e Projeto de Data Warehouse*. 6ª. ed. São Paulo: Editora Érica, 2013.

MACHADO Ângelo; HAERTEL, L. M. *Neuroanatomia Funcional*. 3. ed. São Paulo: Atheneu, 2014.

MARIJAN, B. *What is an OLTP Database?* 2021. PhoenixNap. Acesso em: 08 jun. 2023. Disponível em: <https://phoenixnap.com/kb/oltp-database>.

MCCULLOCH, W.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, v. 52, p. 99–115, 1943. Disponível em: <https://www.cs.cmu.edu/~./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf>.

MEDEIROS, H. *Introdução ao MongoDB*. 2014. DevMedia. Acesso em: 16 ago. 2024. Disponível em: <https://www.devmedia.com.br/introducao-ao-mongodb/30792>.

MERCANTIL, M. *MPEs são responsáveis por 30% do PIB nacional*. 2024. Acesso em: 20 mai. 2025. Disponível em: <https://monitormercantil.com.br/mpes-sao-responsaveis-por-30-do-pib-nacional/>.

MYERS, J.; COPELAND, R. *Essential SQLAlchemy: Mapping Python to Databases*. 2ª. ed. Sebastopol, CA: O'Reilly Media, 2015.

NAPOLEON. *O que é: Key-Value Database*. 2023. Acesso em: 26 set. 2024. Disponível em: <https://napoleon.com.br/glossario/o-que-e-key-value-database/>.

NEGASH, S. Business intelligence. *Communications of the Association for Information Systems*, v. 13, n. 15, p. 177–195, 2004. Disponível em: <https://files.core.ac.uk/download/pdf/301376512.pdf>.

PACHECO, A. *Introdução a Redes Neurais Artificiais*. 2015. Computação Inteligente. Acesso em: 28 ago. 2023. Disponível em: <https://computacaointeligente.com.br/algoritmos/conceitos/redes-neurais-artificiais/>.

PACHECO, A. *Mapas auto-organizáveis (SOM)*. 2017. Computação Inteligente. Acesso em: 14 out. 2024. Disponível em: <https://computacaointeligente.com.br/algoritmos/mapas-auto-organizaveis/>.

PANIZ, D. *NoSQL: Como armazenar os dados de uma aplicação moderna*. 1ª. ed. São Paulo: Casa do Código, 2016.

PRESSMAN, R. S. *Engenharia de Software: Uma abordagem profissional*. 7ª. ed. São Paulo: AMGH, 2011.

Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. 7. ed. Newtown Square, PA: Project Management Institute, 2021.

RAUBER, T. Redes neurais artificiais. 2014. Disponível em: <https://www.researchgate.net/profile/Thomas-Rauber-2/publication/228686464_Redes_neurais_artificiais/links/02e7e521381602f2bd000000/Redes-neurais-artificiais.pdf>.

RECHARTS. *Recharts: Redefined chart library built with React and D3*. s.d. Acesso em: 29 out. 2025. Disponível em: <https://recharts.org/>.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, American Psychological Association, v. 65, n. 6, p. 386–408, 1958. Disponível em: <https://www.ling.upenn.edu/courses/cogs501/Rosenblatt1958.pdf>.

TEAM, T. *Typegoose: Define Mongoose models using TypeScript classes*. 2025. <https://typegoose.github.io/typegoose/>. Acesso em 29 de Outubro de 2025.

TEBALTI, P. *O que é ETL? Entenda como funciona este processo*. 2022. OpServices. Acesso em: 24 ago. 2023. Disponível em: <https://www.opservices.com.br/etl/>.

TOTVS, E. *Business Intelligence: o que é, vantagens e onde usar*. 2024. TOTVS. Acesso em: 28 jun. 2024. Disponível em: <https://www.totvs.com/blog/negocios/business-intelligence/>.

VETTIGLI, G. *MiniSom: minimalistic and NumPy-based implementation of the Self Organizing Map*. 2018. Acesso em: 29 out. 2025. Disponível em: <https://github.com/JustGlowing/minisom/>.

APÊNDICE A - METODOLOGIA USADA NA ELICITAÇÃO DOS REQUISITOS

Este documento tem como objetivo demonstrar a metodologia utilizada na seleção do cliente e os meios empregados para captação dos requisitos que serviram de base para a implementação do sistema. A seguir, são descritas as etapas criadas para a elicitação dos requisitos.

A. 1 PRIMEIRA ETAPA

Após uma breve pesquisa dos negócios que melhor se encaixavam na proposta do sistema desenvolvido neste trabalho, e levando-se em consideração o contato estabelecido pregressamente com algumas pessoas responsáveis por gerir empresas de Micro e Pequeno Porte, foi realizada uma triagem inicial, com base nos possíveis candidatos a cliente do sistema, para definir quais desses empreendimentos teriam o escopo necessário para atender o maior número de demandas possíveis (demandas estas, obtidas inicialmente por meio de brainstormings formadas pelo desenvolvedor do projeto e pessoas com certo conhecimento na área administrativa). Ao final desse processo, duas empresas, de um total de cinco, foram selecionadas: a ODONTOIMAGEM CLÍNICA DE RADIOLOGIA ODONTOLOGICA LTDA, inscrita sob o CNPJ 05.778.809/0001-62, e a INDUSTRIA DE CHURRASQUEIRAS FORTALEZA LTDA de CNPJ 18.816.547/0001-25.

Selecionado os dois empreendimentos que poderiam vir a assumir o papel de futuro cliente, foi elaborada uma mensagem a ser encaminhada a ambos os negócios, através de seus respectivos contatos comerciais, explicando a proposta do trabalho e o conceito do sistema.

> *Olá, meu nome é James Pereira e sou estudante do curso de Análise e Desenvolvimento de Sistemas do IFPB Campus Cajazeiras. Atualmente, estou envolvido nos estágios iniciais do desenvolvimento de um sistema, que busca com base nas informações financeiras de determinado empreendimento, realizar uma auditoria financeira neste, a fim de encontrar inconsistências que possam vir a gerar gargalos ou distorções contábeis em seu negócio, e que ao mesmo tempo seja capaz de disponibilizar meios visuais para o acompanhamento dos gastos, possibilitando uma*

tomada de decisões antecipada com o intuito de ajudar a evitar problemas futuros. Sendo assim, estou em busca de possíveis empresas que tenham interesse na proposta apresentada. Caso tenha, gostaria de marcar uma data para conversarmos mais a respeito deste assunto, que acredito ser benéfico tanto para mim quanto para a sua empresa.

Desde já, agradeço pela atenção e espero ansiosamente pelo seu retorno.

james.pereira@academico.ifpb.edu.br

Passado algum tempo, as empresas contatadas responderam a mensagem mostrando certo interesse em discutir mais sobre o assunto. No decorrer da conversa combinou-se uma reunião, a ser feita de forma presencial com as pessoas responsáveis por administrar cada um dos negócios, com o propósito de aprofundar melhor o assunto.

A. 2 SEGUNDA ETAPA

Depois de realizar alguns encontros com as pessoas responsáveis pelo gerenciamento de cada uma das empresas, e apresentar a proposta de forma mais detalhada, onde ambas demonstraram interesse, deu-se início a uma nova triagem, com o objetivo de estabelecer qual dentre as duas opções disponíveis seria o cliente ideal na obtenção dos requisitos. Tal decisão teria como base pontos e fatores chave, como: a acessibilidade de deslocamento, a disponibilidade para encontros e conversas, a disposição em compartilhar dados contábeis, entre outras. Ao final dessa etapa, optou-se por trabalhar com a INDUSTRIA DE CHURRASQUEIRAS FORTALEZA LTDA.

Escolhido o cliente, foi dado início a uma série de entrevistas e análise de documentos, bem como o acompanhamento do funcionamento diário da empresa e o papel dos funcionários. Ao final desse estudo, foram definidos os requisitos que iriam ser utilizados para auxiliar na criação do sistema.

Pensando em consolidar ainda mais os dados obtidos via entrevistas, ou mesmo detectar pontos que poderiam ter sido deixados de fora das conversas e dos documentos, elaborou-se um questionário complementar, que pode ser visualizado a seguir.

QUESTIONÁRIO COMPLEMENTAR PARA CONSOLIDAÇÃO DOS REQUISITOS

Este questionário tem o intuito de consolidar as informações obtidas via entrevistas e análise dos documentos disponibilizados pelo empreendimento.

1. Tendo em vista o papel central que o gerente desempenha dentro da organização, na ausência deste - seja por qualquer motivo - algum funcionário, dos já existentes na empresa, pode vir a assumir a função, mesmo que de forma temporária?

2. Ao acompanhar o dia a dia da empresa, notou-se que o controle de algumas despesas fixas e quase a totalidade das despesas variáveis, são anotados em papéis, quando não em planilhas do excel. Esta afirmação está correta?

3. Em uma das primeiras entrevistas realizadas, foi relatado que a entrada e saída de funcionários da empresa é um fato constante. Tal fato, chegou a algum momento bagunçar o planejamento de pagamentos ou qualquer outra questão relacionada?

4. Ao acompanhar mais de perto o funcionamento da empresa, não pude deixar de notar o problema que alguns computadores apresentaram. Esses computadores são usados apenas como ambiente de trabalho? Cos-

tumam armazenar todos os dados financeiros da organização? Se sim, a empresa tem o costume de realizar backups periódicos deles?

5. O pagamento das bonificações e horas extras dos funcionários, realmente entram como despesas variáveis?

6. Em uma escala de 0 à 10 (com 0 sendo improvável e 10 constantemente), qual a probabilidade da empresa ficar sem internet?

0	1	2	3	4	5	6	7	8	9	10

7. Dos sistemas presentes na empresa, excluindo o de lançamento de notas fiscais, tem algum outro que não tenha sido apresentado e que desempenha a função de gerenciar os dados dos funcionários, produtos, serviços ou algo semelhante?

APÊNDICE B - DESCRIÇÃO DOS CASOS DE USO DO SISTEMA

Quadro 1 – Caso de Uso 01: Autenticar

UC: Autenticar
ATOR: Diretor-Executivo | Gerente
PRÉ-CONDIÇÕES: Ter conta ativa no sistema
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário acessa página inicial do sistema; 2 - Usuário informa e-mail e senha de acesso; 3 - Usuário clica no botão *LogIn*; 4 - Sistema redireciona usuário para a página inicial correspondente ao seu perfil.
CENÁRIO(S) SECUNDÁRIO(S)
3 - a) E-mail ou senha incorretos: 1 - Sistema informa: “Verifique seu e-mail ou senha”; 2 - Retorna ao passo 2.

Fonte: Elaborado pelo autor (2025)

Quadro 2 – Caso de Uso 02: Redefinir senha

UC: Redefinir senha
ATOR: Diretor-Executivo | Gerente
PRÉ-CONDIÇÕES: Ter conta ativa no sistema
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário acessa a página inicial do sistema e clica em *Esqueceu sua senha?*; 2 - Sistema redireciona usuário para a página de recuperação da conta; 3 - Usuário informa e-mail vinculado a sua conta; 4 - Usuário clica no botão *Recuperar*; 5 - Sistema redireciona usuário para página de confirmação do código; 6 - Usuário informa código enviado pelo sistema via e-mail; 7 - Usuário clica no botão *Confirmar*; 8 - Sistema redireciona usuário para a página de redefinição da nova senha; 9 - Usuário preenche os campos: *informe nova senha* e *confirme sua nova senha*; 10 - Usuário clica no botão *Salvar*; 11 - Sistema informa: “Sua senha foi redefinida com sucesso”; 12 - Sistema aguarda alguns segundos e redireciona o usuário para a página inicial de login.
CENÁRIO(S) SECUNDÁRIO(S)
4 - a) E-mail incorreto: 1 - Sistema informa: “E-mail informado não existe no sistema”; 2 - Retorna ao passo 3. 7 - a) Código inválido: 1 - Sistema informa: “Desculpe, código informado é inválido”; 2 - Retorna ao passo 6. b) Código informado com letras minúsculas: 1 - Sistema informa: “Desculpe, código informado é inválido”; 2 - Retorna ao passo 6. 10 - a) Senhas informadas diferem: 1 - Sistema informa: “As senhas não batem”; 2 - Retorna ao passo 9.

Fonte: Elaborado pelo autor (2025)

Quadro 3 – Caso de Uso 03: Adicionar matriz ou filial

UC: Adicionar matriz ou filial - [Gerenciar matriz e filiais]
ATOR: Diretor-Executivo
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário vai no menu e seleciona o botão *EMPRESAS*; 2 - Sistema renderiza página de gerenciamento com a lista de empresas cadastradas; 3 - Usuário clica no botão de adicionar nova empresa; 4 - Sistema abre um modal com o formulário para o preenchimento; 5 - Usuário preenche formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema fecha formulário, atualiza lista de empresas e informa: “Empresa criada com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
6 - a) Campo(s) obrigatório(s) não informado(s): 1 - Sistema informa: “Verifique se todos os campos obrigatórios foram preenchidos”; 2 - Retorna ao passo 5. b) Empresa com CNPJ, e-mail ou inscrição estadual já cadastrada no sistema: 1 - Sistema informa: “Empresa com cnpj, e-mail ou inscrição estadual já cadastrada”; 2 - Retorna ao passo 5.

Fonte: Elaborado pelo autor (2025)

Quadro 4 – Caso de Uso 04: Editar matriz ou filial

UC: Editar matriz ou filial - [Gerenciar matriz e filiais]
ATOR: Diretor-Executivo
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário vai no menu e seleciona o botão *EMPRESAS*; 2 - Sistema renderiza página de gerenciamento com a lista de empresas cadastradas; 3 - Usuário escolhe empresa que deseja editar; 4 - Usuário clica no botão com o ícone de edição; 5 - Sistema abre modal com formulário já preenchido; 6 - Usuário altera o(s) dado(s) da empresa no formulário; 7 - Usuário clica no botão *SALVAR*; 8 - Sistema fecha formulário, atualiza lista de empresas e informa: “Empresa atualizada com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
7 - a) Campo(s) obrigatório(s) não informado(s): 1 - Sistema informa: “Verifique se todos os campos obrigatórios foram preenchidos”; 2 - Retorna ao passo 6. b) Empresa com CNPJ, e-mail ou inscrição estadual já cadastrada no sistema: 1 - Sistema informa: “Empresa com cnpj, e-mail ou inscrição estadual já cadastrada”; 2 - Retorna ao passo 6. c) Problema(s) de conexão ou no servidor de aplicação: 1 - Sistema informa: “Erro ao atualizar dados da empresa”; 2 - Retorna ao passo 6.

Fonte: Elaborado pelo autor (2025)

Quadro 5 – Caso de Uso 05: Excluir matriz ou filial

UC: Excluir matriz ou filial - [Gerenciar matriz e filiais]
ATOR: Diretor-Executivo
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário vai no menu e seleciona o botão *EMPRESAS*; 2 - Sistema renderiza página de gerenciamento com a lista de empresas cadastradas; 3 - Usuário escolhe a empresa que deseja excluir; 4 - Usuário clica no botão com o ícone de exclusão; 5 - Sistema abre mensagem de confirmação: “Tem certeza que deseja remover essa empresa?”; 6 - Usuário clica em *SIM*; 7 - Sistema informa: “Ao excluir esta empresa todos os dados vinculados a ela serão excluídos. Deseja continuar?”; 8 - Usuário clica no botão *SIM*; 9 - Sistema exclui empresa e atualiza lista de empresas cadastradas.
CENÁRIO(S) SECUNDÁRIO(S)
8 - a) Usuário tenta excluir empresa matriz: 1 - Sistema informa: “A empresa matriz não pode ser excluída”; 2 - Retorna ao passo 3.

Fonte: Elaborado pelo autor (2025)

Quadro 6 – Caso de Uso 06: Enviar mensagem

UC: Enviar mensagem
ATOR: Diretor-Executivo
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário vai no menu e seleciona o botão *ADMINISTRAÇÃO*; 2 - Sistema renderiza página de administração; 3 - Usuário seleciona o card *MENSAGEM*; 4 - Sistema abre modal com a lista de mensagens já cadastradas; 5 - Usuário clica em adicionar nova mensagem; 6 - Sistema abre um modal com o formulário para criação de uma nova mensagem; 7 - Usuário preenche formulário; 8 - Usuário clica no botão *SALVAR*; 9 - Sistema fecha formulário, atualiza lista de mensagens e informa: “Mensagem criada e enviada com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
8 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Por favor preencha todos os campos”; 2 - Retorna ao passo 7.

Fonte: Elaborado pelo autor (2025)

Quadro 7 – Caso de Uso 07: Adicionar gerente

UC: Adicionar gerente - [Administrar perfis dos gerentes]
ATOR: Diretor-Executivo
PRÉ-CONDIÇÕES: UC - Nº 1 | UC - Nº 3
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário vai no menu e seleciona o botão *ADMINISTRAÇÃO*; 2 - Sistema renderiza página de administração; 3 - Usuário seleciona o card *GERENTES*; 4 - Sistema abre modal com a lista de gerentes já cadastrados; 5 - Usuário clica em adicionar novo gerente; 6 - Sistema abre modal com o formulário para adição de um novo gerente; 7 - Usuário preenche formulário; 8 - Usuário clica no botão *SALVAR*; 9 - Sistema fecha formulário, atualiza lista de gerentes e informa: “Gerente criado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
8 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Por favor preencha todos os campos”; 2 - Retorna ao passo 7. b) Gerente com CPF ou e-mail já cadastrado no sistema: 1 - Sistema informa: “Gerente com cpf ou e-mail já cadastrado”; 2 - Retorna ao passo 7. c) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Gerente não pôde ser criado”; 2 - Retorna ao passo 8.

Fonte: Elaborado pelo autor (2025)

Quadro 8 – Caso de Uso 08: Editar gerente

UC: Editar gerente - [Administrar perfis dos gerentes]
ATOR: Diretor-Executivo
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário vai no menu e seleciona o botão *ADMINISTRAÇÃO*; 2 - Sistema renderiza página de administração; 3 - Usuário seleciona o card *GERENTES*; 4 - Sistema abre modal com a lista de gerentes já cadastrados; 5 - Usuário seleciona o gerente da lista e clica no botão com ícone de editar; 6 - Sistema abre modal com formulário já preenchido; 7 - Usuário altera o(s) dado(s) do gerente no formulário; 8 - Usuário clica no botão *SALVAR*; 9 - Sistema fecha formulário, atualiza lista de gerentes e informa: “Gerente atualizado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
8 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Por favor, preencha todos os campos”; 2 - Retorna ao passo 7. b) Gerente com CPF ou e-mail já cadastrado no sistema: 1 - Sistema informa: “Gerente com cpf ou e-mail já cadastrado”; 2 - Retorna ao passo 7. c) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Gerente não pôde ser atualizado”; 2 - Retorna ao passo 8.

Fonte: Elaborado pelo autor (2025)

Quadro 9 – Caso de Uso 09: Excluir gerente

UC: Excluir gerente - [Administrar perfis dos gerentes]
ATOR: Diretor-Executivo
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário vai no menu e seleciona o botão *ADMINISTRAÇÃO*; 2 - Sistema renderiza página de administração; 3 - Usuário seleciona o card *GERENTES*; 4 - Sistema abre modal com a lista de gerentes já cadastrados; 5 - Usuário seleciona o gerente da lista e clica no botão com ícone de excluir; 6 - Sistema abre mensagem de confirmação: “Deseja realmente excluir este registro?”; 7 - Usuário clica em *SIM*; 8 - Sistema exclui registro e atualiza lista de gerentes.

Fonte: Elaborado pelo autor (2025)

Quadro 10 – Caso de Uso 10: Visualizar despesas e receitas

UC: Visualizar despesas e receitas
ATOR: Diretor-Executivo
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário vai no menu e seleciona o botão *RELATÓRIOS*; 2 - Sistema renderiza página de relatórios; 3 - Usuário seleciona o ano de visualização; 4 - Sistema atualiza os gráficos com as despesas e receitas do ano selecionado.

Fonte: Elaborado pelo autor (2025)

Quadro 11 – Caso de Uso 11: Gerar parecer

UC: Gerar parecer
ATOR: Diretor-Executivo
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário vai no menu e seleciona o botão *RELATÓRIOS*; 2 - Sistema renderiza página de relatórios; 3 - Usuário seleciona o ano de visualização; 4 - Usuário clica no botão *GERAR PARECER*; 5 - Sistema gera o parecer da auditoria financeira e abre uma nova guia contendo um arquivo .pdf, com especificações dos dados financeiros da matriz e filiais.

Fonte: Elaborado pelo autor (2025)

Quadro 12 – Caso de Uso 12: Visualizar mensagens

UC: Visualizar mensagens
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *MENSAGENS*; 2 - Sistema abre modal com a lista de mensagens cadastradas; 3 - Usuário seleciona a mensagem da lista e clica no botão com ícone de visualizar; 4 - Sistema renderiza a mensagem completa.

Fonte: Elaborado pelo autor (2025)

Quadro 13 – Caso de Uso 13: Adicionar produto

UC: Adicionar produto - [Gerenciar produtos]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR PRODUTOS*; 2 - Sistema abre modal com a lista de produtos já cadastrados; 3 - Usuário clica no botão de adicionar; 4 - Sistema abre modal com o formulário para adição de um novo produto; 5 - Usuário preenche formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema fecha formulário, atualiza lista de produtos e informa: “Produto criado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
6 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5. b) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Produto não pôde ser criado”; 2 - Retorna ao passo 6.

Fonte: Elaborado pelo autor (2025)

Quadro 14 – Caso de Uso 14: Editar produto

UC: Editar produto - [Gerenciar produtos]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR PRODUTOS*; 2 - Sistema abre modal com a lista de produtos já cadastrados; 3 - Usuário seleciona o produto da lista e clica no botão com ícone de editar; 4 - Sistema abre modal com formulário já preenchido; 5 - Usuário altera o(s) dado(s) do produto no formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema fecha formulário, atualiza lista de produtos e informa: “Produto atualizado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
6 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5. b) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Produto não pôde ser atualizado”; 2 - Retorna ao passo 6.

Fonte: Elaborado pelo autor (2025)

Quadro 15 – Caso de Uso 15: Excluir produto

UC: Excluir produto - [Gerenciar produtos]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR PRODUTOS*; 2 - Sistema abre modal com a lista de produtos já cadastrados; 3 - Usuário seleciona o produto da lista e clica no botão com ícone de excluir; 4 - Sistema abre mensagem de confirmação: “Deseja realmente excluir este registro?”; 5 - Usuário clica em *SIM*; 6 - Sistema exclui registro e atualiza lista de produtos.

Fonte: Elaborado pelo autor (2025)

Quadro 16 – Caso de Uso 16: Adicionar cliente

UC: Adicionar cliente - [Gerenciar clientes]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona card *GERENCIAR CLIENTES*; 2 - Sistema abre modal com a lista de clientes já cadastrados; 3 - Usuário clica no botão de adicionar; 4 - Sistema abre modal com o formulário para adição de um novo cliente; 5 - Usuário preenche formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema fecha formulário, atualiza lista de clientes e informa: “Cliente criado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
6 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5. b) Cliente com e-mail já cadastrado no sistema: 1 - Sistema informa: “Cliente com e-mail já cadastrado”; 2 - Retorna ao passo 5. c) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Cliente não pôde ser criado”; 2 - Retorna ao passo 6.

Fonte: Elaborado pelo autor (2025)

Quadro 17 – Caso de Uso 17: Editar cliente

UC: Editar cliente - [Gerenciar clientes]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR CLIENTES*; 2 - Sistema abre modal com a lista de clientes já cadastrados; 3 - Usuário seleciona o cliente da lista e clica no botão com ícone de editar; 4 - Sistema abre modal com formulário já preenchido; 5 - Usuário altera o(s) dado(s) do cliente no formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema fecha formulário, atualiza lista de clientes e informa: “Cliente atualizado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
6 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5. b) Cliente com e-mail já cadastrado no sistema: 1 - Sistema informa: “Cliente com e-mail já cadastrado”; 2 - Retorna ao passo 5. c) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Cliente não pôde ser atualizado”; 2 - Retorna ao passo 6.

Fonte: Elaborado pelo autor (2025)

Quadro 18 – Caso de Uso 18: Excluir cliente

UC: Excluir cliente - [Gerenciar clientes]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR CLIENTES*; 2 - Sistema abre modal com a lista de clientes já cadastrados; 3 - Usuário seleciona o cliente da lista e clica no botão com ícone de excluir; 4 - Sistema abre mensagem de confirmação: “Deseja realmente excluir este registro?”; 5 - Usuário clica em *SIM*; 6 - Sistema exclui registro e atualiza lista de clientes.

Fonte: Elaborado pelo autor (2025)

Quadro 19 – Caso de Uso 19: Adicionar funcionário

UC: Adicionar funcionário - [Gerenciar funcionários]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR FUNCIONÁRIOS*; 2 - Sistema abre modal com a lista de funcionários já cadastrados; 3 - Usuário clica no botão de adicionar; 4 - Sistema abre modal com o formulário para adição de um novo funcionário; 5 - Usuário preenche formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema fecha formulário, atualiza lista de funcionários e informa: “Funcionário criado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
6 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5. b) Funcionário com CPF já cadastrado no sistema: 1 - Sistema informa: “Funcionário com cpf já cadastrado”; 2 - Retorna ao passo 5. c) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Funcionário não pôde ser criado”; 2 - Retorna ao passo 6.

Fonte: Elaborado pelo autor (2025)

Quadro 20 – Caso de Uso 20: Editar funcionário

UC: Editar funcionário - [Gerenciar funcionários]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR FUNCIONÁRIOS*; 2 - Sistema abre modal com a lista de funcionários já cadastrados; 3 - Usuário seleciona o funcionário da lista e clica no botão com ícone de editar; 4 - Sistema abre modal com formulário já preenchido; 5 - Usuário altera o(s) dado(s) do funcionário no formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema fecha formulário, atualiza lista de funcionários e informa: “Funcionário atualizado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
6 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5. b) Funcionário com CPF já cadastrado no sistema: 1 - Sistema informa: “Funcionário com cpf já cadastrado”; 2 - Retorna ao passo 5. c) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Funcionário não pôde ser atualizado”; 2 - Retorna ao passo 6.

Fonte: Elaborado pelo autor (2025)

Quadro 21 – Caso de Uso 21: Excluir funcionário

UC: Excluir funcionário - [Gerenciar funcionários]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR FUNCIONÁRIOS*; 2 - Sistema abre modal com a lista de funcionários já cadastrados; 3 - Usuário seleciona o funcionário da lista e clica no botão com ícone de excluir; 4 - Sistema abre mensagem de confirmação: “Deseja realmente excluir este registro?”; 5 - Usuário clica em *SIM*; 6 - Sistema exclui registro e atualiza lista de funcionários.

Fonte: Elaborado pelo autor (2025)

Quadro 22 – Caso de Uso 22: Adicionar fornecedor

UC: Adicionar fornecedor - [Gerenciar fornecedores]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR FORNECEDOR*; 2 - Sistema abre modal com a lista de fornecedores já cadastrados; 3 - Usuário clica no botão de adicionar; 4 - Sistema abre modal com o formulário para adição de um novo fornecedor; 5 - Usuário preenche formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema fecha formulário, atualiza lista de fornecedores e informa: “Fornecedor criado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
6 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5. b) Fornecedor com e-mail ou CPF/CNPJ já cadastrado no sistema: 1 - Sistema informa: “Fornecedor com e-mail ou cpf/cnpj já cadastrado”; 2 - Retorna ao passo 5. c) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Fornecedor não pôde ser criado”; 2 - Retorna ao passo 6.

Fonte: Elaborado pelo autor (2025)

Quadro 23 – Caso de Uso 23: Editar fornecedor

UC: Editar fornecedor - [Gerenciar fornecedores]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR FORNECEDOR*; 2 - Sistema abre modal com a lista de fornecedores já cadastrados; 3 - Usuário seleciona o fornecedor da lista e clica no botão com ícone de editar; 4 - Sistema abre modal com formulário já preenchido; 5 - Usuário altera o(s) dado(s) do fornecedor no formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema fecha formulário, atualiza lista de fornecedores e informa: “Fornecedor atualizado com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
6 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5. b) Fornecedor com e-mail ou CPF/CNPJ já cadastrado no sistema: 1 - Sistema informa: “Fornecedor com e-mail ou cpf/cnpj já cadastrado”; 2 - Retorna ao passo 5. c) Problema(s) no servidor de aplicação: 1 - Sistema informa: “Fornecedor não pôde ser atualizado”; 2 - Retorna ao passo 6.

Fonte: Elaborado pelo autor (2025)

Quadro 24 – Caso de Uso 24: Excluir fornecedor

UC: Excluir fornecedor - [Gerenciar fornecedores]
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *GERENCIAR FORNECEDOR*; 2 - Sistema abre modal com a lista de fornecedores já cadastrados; 3 - Usuário seleciona o fornecedor da lista e clica no botão com ícone de excluir; 4 - Sistema abre mensagem de confirmação: “Deseja realmente excluir este registro?”; 5 - Usuário clica em *SIM*; 6 - Sistema exclui registro e atualiza lista de fornecedores.

Fonte: Elaborado pelo autor (2025)

Quadro 25 – Caso de Uso 25: Lançar vendas

UC: Lançar vendas
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1 | UC - Nº 12 | UC - Nº 15
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *LANÇAR VENDAS*; 2 - Sistema abre modal com a data da última venda registrada e o total de vendas efetuadas; 3 - Usuário clica no botão de adicionar; 4 - Sistema abre modal com o formulário para o lançamento da nova venda; 5 - Usuário preenche formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema abre mensagem de confirmação: “Tem certeza que deseja registrar esta venda?”; 8 - Usuário clica em *CONFIRMAR*; 9 - Sistema lança venda, fecha formulário, atualiza data da última venda, o total registrado e informa: “Venda registrada com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
8 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5.

Fonte: Elaborado pelo autor (2025)

Quadro 26 – Caso de Uso 26: Lançar despesas

UC: Lançar despesas
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *LANÇAR DESPESAS*; 2 - Sistema abre modal com a data e tipo da última despesa registrada e o total de despesas efetuadas; 3 - Usuário clica no botão adicionar; 4 - Sistema abre modal com o formulário para o lançamento da nova despesa; 5 - Usuário preenche formulário; 6 - Usuário clica no botão *SALVAR*; 7 - Sistema abre mensagem de confirmação: “Tem certeza que deseja registrar esta despesa?”; 8 - Usuário clica em *CONFIRMAR*; 9 - Sistema lança despesa, fecha formulário, atualiza data e tipo da última despesa, bem como o total registrado e informa: “Despesa registrada com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
8 - a) Campo(s) obrigatório(s) não preenchido(s): 1 - Sistema informa: “Preencha todos os campos”; 2 - Retorna ao passo 5.

Fonte: Elaborado pelo autor (2025)

Quadro 27 – Caso de Uso 27: Realizar *upload* de arquivos

UC: Realizar *upload* de arquivos
ATOR: Gerente
PRÉ-CONDIÇÕES: UC - Nº 1
CENÁRIO PRINCIPAL - PRIMÁRIO
1 - Usuário seleciona o card *ENVIAR ARQUIVOS*; 2 - Sistema abre modal; 3 - Usuário clica no botão *SELECIONE ARQUIVOS*; 4 - Sistema abre janela do computador para a seleção dos arquivos a serem enviados; 5 - Usuário seleciona arquivos; 6 - Sistema carrega arquivos selecionados; 7 - Usuário clica no botão *INICIAR UPLOAD*; 8 - Sistema efetua upload, limpa lista de arquivos selecionados e informa: “Arquivos enviados com sucesso”.
CENÁRIO(S) SECUNDÁRIO(S)
7 - a) Arquivo(s) selecionado(s) fora do padrão: 1 - Sistema informa: “Verifique se o(s) arquivo(s) segue(m) o padrão definido em regra, e se os dados foram preenchidos corretamente”; 2 - Retorna ao passo 5.

Fonte: Elaborado pelo autor (2025)

APÊNDICE C - DICIONÁRIO LÓGICO DE DADOS

Quadro 28 – Dicionário de Dados 1: Relação *sale_fact*

SALE_FACT: Relação que armazena as informações de venda				
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**DOMÍNIO**	**RESTRIÇÃO**
id	Atributo que identifica a venda	SERIAL	SERIAL	• Chave primária
the_amount	Atributo que representa a quantidade de produtos vendidos	INTEGER	Inteiros positivos	• Não nulo
type	Atributo que representa o tipo de venda efetuada (À VISTA ou A PRAZO)	VARCHAR (50)	Alfabético	• Não nulo • Apenas letras [A-Z]
value	Atributo que identifica o valor da venda	FLOAT	Decimais positivos	• Não nulo
id_client_dim	Atributo que referencia a relação *CLIENT_DIMENSION*	BIGINT	BIGINT (FK)	• Não nulo
id_product_dim	Atributo que referencia a relação *PRODUCT_DIMENSION*	BIGINT	BIGINT (FK)	• Não nulo
id_company_dim	Atributo que referencia a relação *COMPANY_DIMENSION*	BIGINT	BIGINT (FK)	• Não nulo
id_period_dim	Atributo que referencia a relação *PERIOD_DIMENSION*	BIGINT	BIGINT (FK)	• Não nulo

Fonte: Elaborado pelo autor (2026)

Quadro 29 – Dicionário de Dados 2: Relação *product_dimension*

PRODUCT_DIMENSION: Relação que armazena as informações de produto				
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**DOMÍNIO**	**RESTRIÇÃO**
id	Atributo que identifica o produto	SERIAL	SERIAL	• Chave primária
name	Atributo que representa o nome do produto	VARCHAR (30)	Alfanumérico	• Não nulo
type	Atributo que representa o tipo de produto (CAPITAL ou CUSTEIO)	VARCHAR (30)	Alfabético	• Não nulo • Apenas letras [A-Z]
category	Atributo que determina a categoria do produto (COMUM, SIMPLES, COMPLETA ou PREMIUM)	VARCHAR (30)	Alfabético	• Não nulo • Apenas letras [A-Z]

Fonte: Elaborado pelo autor (2026)

Quadro 30 – Dicionário de Dados 3: Relação *client_dimension*

CLIENT_DIMENSION: Relação que armazena as informações de cliente				
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**DOMÍNIO**	**RESTRIÇÃO**
id	Atributo que identifica o cliente	SERIAL	SERIAL	• Chave primária
email	Atributo que representa a e-mail do cliente	VARCHAR (200)	Alfanumérico	• Não nulo
name	Atributo que representa o nome do cliente	VARCHAR (200)	Alfanumérico	• Não nulo
legal_entity	Atributo que identifica se o cliente é pessoa jurídica	BOOLEAN	BOOLEAN	• Não nulo
uf	Atributo que representa o estado onde está o cliente	VARCHAR (100)	Alfabético	• Não nulo
city	Atributo que representa a cidade onde está o cliente	VARCHAR (100)	Alfabético	• Não nulo
district	Atributo que representa o bairro onde está o cliente	VARCHAR (100)	Alfabético	• Não nulo

Fonte: Elaborado pelo autor (2026)

Quadro 31 – Dicionário de Dados 4: Relação *period_dimension*

PERIOD_DIMENSION: Relação que armazena as informações de período				
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**DOMÍNIO**	**RESTRIÇÃO**
id	Atributo que identifica o período	SERIAL	SERIAL	• Chave primária
month	Atributo que representa o mês do período	INTEGER	Inteiros positivos	• Não nulo • Intervalo [1-12]
quarter	Atributo que representa o trimestre do período	INTEGER	Inteiros positivos	• Não nulo • Intervalo [1-4]
year	Atributo que representa o ano do período	INTEGER	Inteiros positivos	• Não nulo

Fonte: Elaborado pelo autor (2026)

Quadro 32 – Dicionário de Dados 5: Relação *company_dimension*

COMPANY_DIMENSION: Relação que armazena as informações de empresa				
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**DOMÍNIO**	**RESTRIÇÃO**
id	Atributo que identifica a empresa	SERIAL	SERIAL	• Chave primária
cnpj	Atributo que representa o CNPJ da empresa	VARCHAR (30)	Alfanumérico	• Não nulo • Apenas números [0-9]
telephone	Atributo que representa o telefone da empresa	VARCHAR (20)	Alfanumérico	• Não nulo • Apenas números [0-9]
email	Atributo que representa a e-mail da empresa	VARCHAR (200)	Alfanumérico	• Não nulo
uf	Atributo que representa o estado de localização da empresa	VARCHAR (100)	Alfabético	• Não nulo
city	Atributo que representa a cidade de localização da empresa	VARCHAR (100)	Alfabético	• Não nulo
district	Atributo que representa o bairro de localização da empresa	VARCHAR (100)	Alfabético	• Não nulo

Fonte: Elaborado pelo autor (2026)

Quadro 33 – Dicionário de Dados 6: Relação *category_dimension*

CATEGORY_DIMENSION: Relação que armazena as informações de categoria				
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**DOMÍNIO**	**RESTRIÇÃO**
id	Atributo que identifica a categoria	SERIAL	SERIAL	• Chave primária
type	Atributo que representa o tipo de gasto (FIXO ou VARIÁVEL)	VARCHAR (20)	Alfabético	• Não nulo • Apenas letras [A-Z]
category	Atributo que representa a categoria do gasto (COMPRA, PAGAMENTO, PRESTAÇÃO DE SERVIÇOS ou FOLHA DE PAGAMENTO)	VARCHAR (200)	Alfabético	• Não nulo • Apenas letras [A-Z]

Fonte: Elaborado pelo autor (2026)

Quadro 34 – Dicionário de Dados 7: Relação *expense_fact*

EXPENSE_FACT: Relação que armazena as informações de gastos				
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**DOMÍNIO**	**RESTRIÇÃO**
id	Atributo que identifica o gasto	SERIAL	SERIAL	• Chave primária
value	Atributo que representa o valor do gasto	FLOAT	Decimais positivos	• Não nulo
id_period_dim	Atributo que referencia a relação *PERIOD_DIMENSION*	BIGINT	BIGINT (FK)	• Não nulo
id_company_dim	Atributo que referencia a relação *COMPANY_DIMENSION*	BIGINT	BIGINT (FK)	• Não nulo
id_category_dim	Atributo que referencia a relação *CATEGORY_DIMENSION*	BIGINT	BIGINT (FK)	• Não nulo

Fonte: Elaborado pelo autor (2026)

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Cajazeiras - Código INEP: 25008978
Rua José Antônio da Silva, 300, Jardim Oásis, CEP 58.900-000, Cajazeiras (PB)
CNPJ: 10.783.898/0005-07 - Telefone: (83) 3532-4100

Documento Digitalizado Restrito

Deposito de TCC

Assunto:	Deposito de TCC
Assinado por:	James Amarante
Tipo do Documento:	Projeto
Situação:	Finalizado
Nível de Acesso:	Restrito
Hipótese Legal:	Direito Autoral (Art. 24, III, da Lei no 9.610/1998)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **James Pereira dos Santos Amarante, ALUNO (201712010033) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 25/02/2026 15:33:00.

Este documento foi armazenado no SUAP em 25/02/2026. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1779034
Código de Autenticação: 054288f4b4