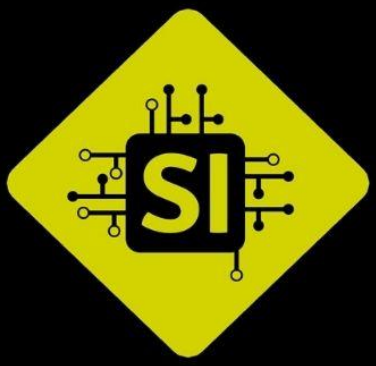
Editora Uniesp

DIÁLOGOS CIENTÍTICOS EM SISTEMAS

PRODUÇÕES CIENTÍFICAS 2022.1



Hercilio de Medeiros Sousa
Marcelo Fernandes de Sousa
Messias Rafael Batista
(Organizadores)

ISBN: 978-65-5825-142-2

DIÁLOGOS CIENTÍFICOS EM SISTEMAS: PRODUÇÕES ACADÊMICAS 2022.1

Hercilio de Medeiros Sousa
Marcelo Fernandes de Sousa
Messias Rafael Batista
(Organizadores)

Centro Universitário – UNIESP

Cabedelo - PB
2022

Diagramação e capa:
Márcia de Albuquerque Alves

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Padre Joaquim Colaço Dourado (UNIESP)

D537 Diálogos científicos em sistemas: produções acadêmicas 2022.1 [recurso eletrônico] / Organizadores, Hercílio de Medeiros Sousa, Marcelo Fernandes de Sousa, Messias Rafael Batista. - Cabedelo, PB: Editora UNIESP, 2022.
187 p.; il : color.

Tipo de Suporte: E-book
ISBN: 978-65-5825-142-2

1. Produção científica – Sistemas. 2. Sistemas – Interdisciplinaridade. 3. Diálogos – Conhecimento científico. I. Título. II. Sousa, Hercílio de Medeiros. III. Sousa, Marcelo Fernandes de. IV. Batista, Messias Rafael.

CDU: 001.891:004.775

Bibliotecária: Elaine Cristina de Brito Moreira – CRB-15/053

Editora UNIESP
Rodovia BR 230, Km 14, s/n,
Bloco Central – 2 andar – COOPERE
Morada Nova – Cabedelo – Paraíba
CEP: 58109-303

SUMÁRIO

ATUAL CENÁRIO DE PUBLICAÇÕES COM USO DE MACHINE LEARNING PARA DIAGNÓSTICOS EM SAÚDE

PEREIRA, Suzana Maria de Freitas
MORAIS, Alana Marques
LEITE, Danilo Rangel Arruda

RESUMO

Machine Learning (ML) possibilita replicar comportamentos, tomar decisões, e utilizar suas técnicas para treinamento. A utilização de ML tem ganhado muito espaço na área da Saúde, apesar de ser uma área bastante complexa. As técnicas de ML podem ajudar a prevenir doenças e na tomada de decisão para pacientes com comorbidades. Este trabalho tem como objetivo entender o atual cenário de publicações de ML na área da Saúde, por meio de um mapeamento sistemático realizado em diferentes bases de dados, no qual foi possível buscar trabalhos em diversas subáreas da Saúde. Os resultados deste estudo apontaram que há um grande volume de publicações nos anos de 2019 e 2020, sendo a China com maior número de publicações, a subárea da Saúde mais estudada foi a de Medicina e a patologia mais explorada foi o Alzheimer. Este trabalho serviu para explorar uma temática muito rica e que muito ainda tem a ser feito e contribuirá para nortear e motivar estudos futuros.

Palavras-chave: Machine Learning; Saúde; Diagnóstico

1 INTRODUÇÃO

Machine Learning (ML) ou Aprendizagem de Máquina é uma área da Inteligência Artificial que, segundo a *International Business Machines Corporation* (IBM), utiliza métodos estatísticos para treinar dados por meio de algoritmos, possibilitando realizar classificações, previsões e insights que geram métricas que contribuirão para a tomada de decisão nas organizações.

Para Schneider (2016) ML faz parte de um subgrupo da ciência da computação, que se preocupa com reconhecimento de padrões e de teorias de aprendizagem de máquina com Inteligência Artificial, caracterizando-se

pelo uso de modelos e algoritmos de aprendizagem automática, que fazem previsões a partir de dados já conhecidos.

Nos últimos tempos, a ciência de dados tem ganhado bastante visibilidade devido à grande quantidade de dados que estamos gerando a todo momento, a exemplo do uso dos diversos tipos de dispositivos eletrônicos que as pessoas têm

utilizado, como *smartphones*, *smartwatch,* eletrodomésticos inteligentes, sistemas de prontuários médicos, e outros MOTTA (2008). Esse fenômeno é denominado Internet das Coisas, onde qualquer dispositivo podeestar conectado a alguma rede, gerando assim a hiperconectividade.

Para Lobo (2018) as evoluções da prática médica com o uso de sistemasde processamento que registram dados nos prontuários eletrônicos por linguagem natural, uso de computadores ou *smatphones* durante a consulta, utilização de tecnologias que fazem reconhecimento de imagens, além de sistemas que ajudam a interação do médico com o paciente, permitindo apoio, confiabilidade e acompanhamento dos quadros de saúde do paciente.

A utilização de ML na área da saúde tem ganhado notoriedade, apesar deser uma área bastante complexa, crítica e que passa por constantes mudanças. As técnicas de ML podem ajudar nos processos de tomada de decisão, seja para diagnosticar, intervir, ou monitorar paciente com comorbidades BATISTA (2019).

Durante a pandemia da Covid-19, a população acompanhou, em tempo real, o crescimento do número de casos, mortes registradas, quantitativo de pessoas vacinadas, graças ao alto poder de processamento de dados, bem como os dados disponibilizados pelo Ministério da Saúde, que podem ser utilizados para diversos estudos, treinar modelos de ML, e assim ter informações relevantes para políticas governamentais e para a ciência comoum todo.

No geral, as técnicas de ML têm sido muito utilizadas para trazer qualidade de vida para o paciente por meio do diagnóstico precoce das doenças de forma precisa, além de reduzir o custo benefício dos tratamentos,contribuindo assim para a redução dos índices de mortalidade da população NURMAINI (2019).

Para entender como está o atual cenário de publicações de ML na área da Saúde, este trabalho irá discutir os resultados de um mapeamento sistemático realizado em diferentes bases de dados científicas: *Science Direct*, IEEE, e CAPES. Foi possível buscar trabalhos em diversas subáreas da Saúde, com estudos que utilizem ML para diagnóstico de doenças e/ou comorbidades. Quais áreas da Saúde estão sendo mais estudadas? Por que elas estão sendo mais discutidas?

Ao final deste trabalho, foi possível verificar que a área da Medicina é a que mais tem publicações e o Alzheimer tem sido mais estudado. O presente trabalho

foi desenvolvido no período de Outubro de 2020 a Outubro de 2021 e está organizado em tópicos: a contextualização da problemática, em seguida a fundamentação teórica, o método usado para os resultados, os quais serão abordados juntamente com alguns insights obtidos, e ao final concluiremos toda a ideia do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Para Tulloch (2020) os constantes avanços na tecnologia que utilizam algoritmos de ML em saúde tem se tornando cada vez mais popular por sua capacidade de diminuir erros e a dependência humana, implicando na redução de custos.

Segundo Rasheed (2020) os avanços tecnológicos têm ajudado aos neurocientistas a utilizar os dados de neuroimagens, e acredita que a utilização de ML e Deep Learning são tecnologias que podem somar ainda mais às comumente usadas para a construção de novas teorias relacionadasao cérebro.

Sun (2020) menciona que na área médica, há uma grande utilização das técnicas ML em diversas especialidades como cardiologia, oftalmologia, nefrologia, radioterapia, neurologia, endocrinologia, oncologia, estomatologia, psicologia e outras. Essas técnicas tem a capacidade de analisar grandes volumes de dados com a finalidade de obter previsão e diagnóstico de doenças, e sua aplicação visa solucionar alguns problemas da prática médica como baixa eficiência e erros de diagnóstico.

É possível encontrar na literatura diversos métodos isolados ou combinados a outros métodos, como Martinez-Murcia (2018) sugere o uso de Redes Neurais Convolucionais para buscar padrões a fim de identificar o surgimento de doenças como o Parkinson. No geral, métodos baseados em aprendizado profundo, ajudarão a descobrir padrões mais complexos e, superando as metodologias estatísticas clássicas para diagnósticos mais precisos.

Graham (2019) afirma que a Inteligência artificial em saúde, utiliza algoritmose software que exercem o papel das funções cognitivas de seres humanos para analisar dados médicos complexos, como imagens de exames ou prontuários médicos. As ferramentas de IA usam esses dados para inferir mudanças patológicas no funcionamento cognitivo e deve ser capaz de identificar aos

primeiros sinais de problemas.

Para Liu (2009) as imagens de exames são evidências muito importantes para o diagnóstico médico, mas sua interpretação depende de análises de seres humanos e com a grande demanda de exames realizados diariamentese torna um desafio enorme para os especialistas, surgindo assim a necessidade de existir outras formas de realizar diagnósticos.

O presente trabalho foi desenvolvido por meio de um mapeamento sistemático que é um tipo de pesquisa que se baseia em evidências da literatura onde é possível argumentar sobre novas hipóteses. Normalmente é usado para investigar situações na área médica, permitindo classificar os resultados dos dados coletados sobre determinado assunto, mostrando insights para pesquisas futuras, evidenciando possíveis lacunas (ROCHA, 2018).

3 METODOLOGIA

Mediante o cenário de inúmeras publicações disponíveis na literatura relacionado ao uso de ML para realização de diagnósticos, foi possível construir este mapeamento sistemático por meio de algumas etapas para filtragem das publicações e um gráfico para facilitar a compreensão (figura1).

- Etapa I: O ponto de partida do trabalho foi definir as bases de dados a serem utilizadas no mapeamento. Foram selecionadas a *Science Direct*, IEEE (*Institute of Electrical and Electronics Engineers*), e a CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), por serem bases renomadas, de grande credibilidade e englobam diferentes áreas. O intuito era explorar bases de diversas áreas e grupos de pesquisa do mundo para ter uma visão mais ampla sobre a problemáticainvestigada.
- Etapa II: As palavras chaves utilizadas na expressão de busca dos artigos foram: “Machine Learning”, “Big Data”, e as subáreas buscadas foram “Neurociência”, “Medicina”, “Mental Health”, “Psychology”, “Physiotherapy”, “Dentistry”, “Speech therapy”, “Occupational therapy”, “Neurocomputing”, estas palavras foram buscadas de forma combinada como por exemplo “Machine learning” OR “Big Data” AND “Psychology”, como também foi definido o período de publicações, que foram filtradas entre 2015 e 2020. Tais termos de busca foram extraídos de acordo com as áreas e temáticas

afins ao tema da revisão sistemática que estava sendo planejado.

- Etapa III: Foi analisado o título, resumo e palavra-chaves de cada artigo mediante alguns critérios de aceitação como: (i) o artigo precisaria falar sobre Data Science e Saúde, (ii) ser uma publicação do tipo artigo (não postagem, notícia, etc.), (iii) o artigo precisa apresentar resultados sólidos, (iv) deve estar escrito em inglês ou português, e (v) as palavras-chaves precisam ser compatíveis com as buscadas.
- Etapa IV: Foi realizada uma leitura mais detalhada do resumo elencando as razões de exclusão dos artigos que não se encaixam na pesquisa, como não ter objetivo claro, não falar sobre diagnóstico, não falar sobre aprendizado de máquina e saúde, não comentar sobre avaliação dos modelos utilizados.
- Na etapa V: Foi realizada uma leitura otimizada mediante introdução, metodologia e conclusão para entender se o corpo do artigo estava conforme o esperado.
- E na etapa VI: Foram feitos gráficos (figura2, figura 3, figura 4) para melhor compreender a quantidade de artigos por área e facilitar a visualização.

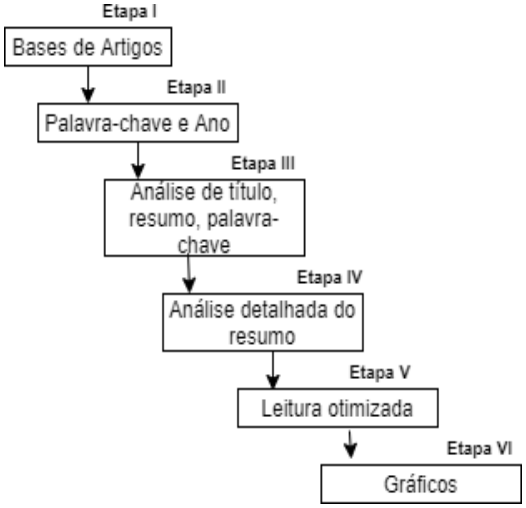


Figura 1. Etapas da metodologia

4 DESENVOLVIMENTO

Em cada etapa foi feita uma filtragem dos artigos, como descrito no tópico

anterior, que resultou nos seguintes quantitativos: Nas etapas I e II, no somatório de todas as áreas, foram encontrados inicialmente 1.205 artigos, na *Science Direct*, 513 na IEEE, e 232 na CAPES. Já na etapa III: serviram apenas 326 artigos na *Science Direct*, 254 na IEEE, e 55 na CAPES. Na etapa IV obtivemos 46 artigos na *Science Direct*, 38 na IEEE, e 7 na CAPES. E na última etapa de filtragem, restaram 28 na Science Direct, 8 na IEEE, e 1 na CAPES.

Para uma visão mais clara da quantidade de estudos disponíveis nas bases de dados usadas neste artigo, é possível entender que as publicações sobre ML e diagnósticos em Saúde se intensificaram no ano 2020 obtendo 0,11% de publicações na *Science Direct* (figura 2), na IEEE o percentual é de 0,05% também em 2020 (figura 3), e na CAPES temos 0,01% no ano de 2019 (figura 4). Percebemos que a base de dados *Science Direct* reúne um volume maior de publicações dentro do tema deste artigo.

De acordo com a presente pesquisa, foi possível perceber que grande parte dos estudos foram publicados nos anos de 2019 e 2020, os países que mais publicaram foram China, Japão, e Brasil, a sub área da Saúde mais estudada foi a Medicina, seguida da Neurociência e Saúde Mental e as patologias mais exploradas foram o Alzheimer, Câncer de Próstata, Câncer de Mama, Esquizofrenia, Parkinson.

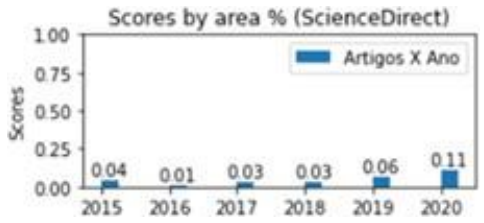


Figura 2. Percentual de artigos (Science Direct)

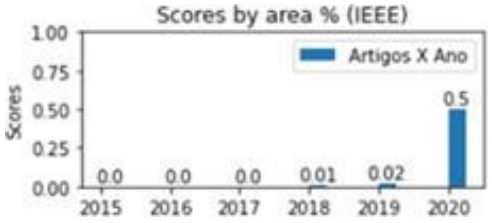


Figura 3. Percentual de artigos (IEEE)

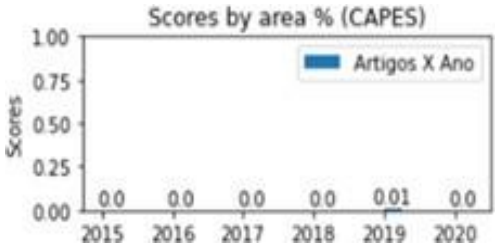


Figura 4. Percentual de artigos (CAPES)

Os gráficos acima foram construídos na ferramenta *Google Colaboratory*,

usando a biblioteca Pandas *Numpy*, na linguagem Python.

A patologia com mais publicações foi o Alzheimer, em vários trabalhosforam usadas metodologias diferentes voltadas para processamento e classificação de imagens para diagnóstico, o que torna bastante interessante, pois mostra que diferentes técnicas estão sendo testadas para melhores resultados e mais avanços.

As técnicas de *Machine Learning* podem ser agrupadas em três diferentes categorias: aprendizagem supervisionada, aprendizagem não supervisionada e aprendizado por reforço. Cada grupo possui característicasprópria a depender da natureza dos dados a serem aprendidos, tarefas e problemas a serem resolvidos.

Normalmente, os modelos de aprendizagem supervisionada são mais usados nas decisões de Saúde, por utilizarem conjuntos de dados etiquetados por especialistas, onde os algoritmos irão prever ou classificar eventos futuros ou para determinar variáveis mais relevantes (LOBO, 2018).

Segundo Schneider (2016) a classificação de dados, é um modelo muito comum e bastante utilizado quando tratamos de ML, que tem como objetivo categorizar novos eventos que irão somar aos existentes em bancos de dados e resultarão em novos comportamentos, novos padrões.

5 CONSIDERAÇÕES FINAIS

A partir deste mapeamento realizado, foi possível entender que a área da Medicina obtém maior número de publicações e outas áreas publicaram menos ou não publicaram dentro do tema deste artigo, podendo está relacionado ao pouco conhecimento sobre a área de ML e as suas contribuições.

Este estudo foi bastante desafiador diante da vasta quantidade de artigos selecionados nas etapas iniciais e posteriormente na filtragem dos mesmos, pois foram revisados por apenas um colaborador. Porém, irá contribuir para nortear e motivar estudos futuros, proporcionando uma melhor visão de como está o atual cenário das pesquisas na área da Saúde que utilizam ML para diagnóstico médico. Outra contribuição deste trabalho é queele está explorando uma temática muito rica e que muito ainda tem a ser feito.

REFERENCIAS

GRAHAM, S. A; et al. **Artificial Intelligence Approaches to Predicting and Detecting Cognitive Decline in Older Adults: A Conceptual Review**. Journal Pre-proof, 58, 2019.

International Business Machines Corporation (IBM). **Machine Leraning**. IBM,2020. Disponível em: (https://www.ibm.com/br-pt/cloud/learn/machine- learning). Acesso em: (24 de Março de 2022).

LIU, X; et al. **A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis**. Science Direct, 1, e271-e297, October, 2019.

LOBO LC. **Inteligência artificial, o Futuro da Medicina e a EducaçãoMédica**. Revista Brasileira de Educação médica, 42 (3), 3-8, Setembro, 2018.

MARTINEZ-MURCIA, F. J; GÓRRIZ J. M; RAMÍREZ, J; ORTIZ A.Convolutional neural networks for neuroimaging in parkinson's disease: Is preprocessing needed?, Internacional Journal of Neural Systems, 0, 0, 1- 18, July, 2018.

MOTTA GS. **A Mobilidade e a Hiperconexão como Tecnologias de Vigilância na Sociedade de Controle**. ENPAG, 1-13, 2008.

NURMAINI S, et al**. An Automated ECG Beat Classification System UsingDeep Neural Networks with an Unsupervised Feature ExtractionTechnique**. Appl. Sci, 9 (14), 2921, 2019.

RASHEED K, et al. **Machine Learning for Predicting Epileptic Seizures Using EEG Signals: A Review**. IEEE, 14, 139 – 155, July, 2020.

ROCHA F. G, NASCIMENTO B. A. R, NASCIMENTO E. F. V. C. **Um modelode mapeamento sistemático para a educação**. Cadernos da Fucamp. 17, 29, 1-6, 2018.

SCHNEIDER, P. H. **Análise preditiva de Churn com ênfase em técnicas de Machine Learning: Uma Revisão**. 2016. 82 - Fundação Getúlio Vargas,Escola de Matemática Aplicada. Rio de Janeiro, 2016.

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Machine Learning aplicadoà Saúde**. São Paulo, 2019. 42 p.

SUN, M-L; et al. **Application of Machine Learning to Stomatology: A Comprehensive Review.** IEEE, 8, 184360 – 184374, October, 2020.

TULLOCH, J; ZAMANI, R; AKRAMI, A.M. **Machine Learning in the Prevention, Diagnosis and Management of Diabetic Foot Ulcers: A Systematic Review**. IEEE, 8, 198977 – 199000, November, 2020.

DESENVOLVIMENTO DE UM SISTEMA DE BUSINESS INTELLIGENCE APLICADO À UM SETOR DE CONTROLADORIA COMERCIAL DE UMA EMPRESA NO RAMO DA PAPELARIA - A “Archivery”

MIRANDA, Letícia de Oliveira
MEDEIROS, Fábio Nicácio de

RESUMO

O Business Intelligence (BI) combina análise de negócios, visualização de dados, ferramentas e infraestrutura de dados. Contemplando práticas que provêm uma maior facilidade para que as organizações possam tomar decisões orientadas por dados. O presente projeto visa oferecer uma solução de BI ponta-a-ponta para uma empresa do ramo de papelaria. Provendo desde mineração à visualização dos dados do negócio, demonstrando como utilizar os dados que estão em uma visão departamentalizada, para uma visão integral. Garantindo na prática uma inteligência de negócios moderna aos gestores, permitindo que obtenham uma visão abrangente dos dados de sua organização e os utilize para impulsionar mudanças, eliminar ineficiências e adaptar-se rapidamente às mudanças de mercado.

Palavras-chave: Negócios; Tecnologia; Dados.

1 INTRODUÇÃO

O uso de Tecnologias da Informação para a administração de processos de gerenciamento já é um fato presente em nossa sociedade. A utilização de softwares de gestão empresarial, sejam físicos ou virtuais, já é uma realidade para a maioria dos empreendimentos de médio e grande porte. A facilidade com que as informações são integradas leva traz mais rapidez e precisão na coleta de resultados, que são de primordial valor para todo tipo de organização. Partindo deste ponto, percebeu-se que um dos principais aliados no desenvolvimento de uma companhia, é a realização de uma gestão eficiente dos dados. Em uma ótica comercial, os dados compreendem à uma gama de informações, como: Clientes, produtos, fornecedores, funcionários, estatísticas de venda, e a utilização eficiente desses dados, aplicadas à uma visão estratégica, trazem um controle maior dos resultados obtidos.

O Business Intelligence é um sistema que manipula diversas informações, agregando-as, filtrando-as e adequando-as às mais diversas necessidades dos administradores das organizações. Com este sistema, a análise de dados se torna muito mais rápida e abrangente, e em consequência disto a tomada de decisões se torna cada vez mais eficiente, alinhando velocidade e satisfatoriedade. Para

desenvolvimento de um sistema de Business Intelligence se mostram necessárias diversas ferramentas que buscam o armazenamento de informações dentro de um mesmo padrão, a captação dos dados em diversos locais, a adequações de tudo que é captado e finalmente a maneira de análise e extração destes dados de forma que a pesquisa se torne relativamente simples ao usuário final, que poderá extrair as informações que desejar em um formato compatível com a sua necessidade. É por toda esta importância que o sistema de Business Intelligence demonstra, e por tudo que existe por trás dele, que uma análise deste tema se mostra muito importante.

2 FUNDAMENTAÇÃO TEÓRICA

As empresas que possuem sistemas avançados que monitoram as atividades de seus concorrentes exibem maior lucratividade. Sistemas de BI permitem que as empresas aproveitem a rentabilidade da racionalização da cadeia de suprimentos para aumentar a competitividade. Além disso, a possibilidade de fechar novos negócios é aumentada quando um sistema integrado facilita as transações. (TEO; CHOO, 2001).

Visualizando essa necessidade, um sistema de Business Intelligence mostra-se como um sinônimo de evolução, principalmente quando se fala de competitividade, e uma companhia que é atualizada perante as tecnologias disponíveis no mercado, se diferencia. Com isso, as organizações têm se mostrado cada vez mais interessadas em ferramentas que gerenciam processos e informações, aliadas com eficiência e velocidade.

2.1 DATA WAREHOUSE

Um Data Warehouse nada é um banco de dados que contém dados extraídos do ambiente de produção da empresa, que foram selecionados e depurados, tendo sido otimizados para processamento de consulta. O Data Warehouse requer a consolidação de outros recursos de dados de origem, e armazenamento em banco de dados relacionais [INMON, 1999].

2.2 DATA MART

Um Data Mart é um subconjunto de um Data Warehouse focado em um determinado departamento ou assunto. Os Data Marts disponibilizam dados específicos para um grupo definido de usuários, o que permite que esses usuários acessem rapidamente insights importantes sem perder tempo pesquisando em um Data Warehouse inteiro. Por exemplo, muitas empresas podem ter um Data Mart alinhado a um departamento específico da empresa, como finanças, vendas ou marketing.

3 METODOLOGIA

O presente trabalho, do ponto de vista da natureza, pode ser classificado como uma pesquisa exploratória aplicada, utilizando-se de informações de uma empresa fictícia. No projeto foram levantados, primeiramente, os conceitos fundamentais Business Intelligence e gestão de dados, presentes na literatura. A partir dessas informações, desenvolveu-se um modelo que agrega as fases de um projeto de BI. Objetivando desenvolver uma solução capaz de coletar dados de uma fonte, organizá-los, analisá-los e compartilhá-los em forma de relatórios descritivos com itens visuais, transformando estes em informações relevantes que sejam capazes de prover apoio na tomada decisões importantes para o futuro da empresa.

As metodologias utilizadas consistem em pontuar, de forma analítica, as fases do projeto de Business Intelligence. Exemplificando cada etapa que define a solução, com objetivo de gerar percepções analíticas acima das visões comerciais de uma empresa do ramo de papelaria. Dessa forma, este trabalho deverá contemplar, de ponta a ponta, um projeto de Business Intelligence, provendo uma implementação que coleta os dados desde a fonte por meio de uma ferramenta Open Source, realiza o tratamento dos mesmos e o envia em para um software destinatário onde serão aplicadas as visualizações acima das informações coletadas. Apontando fatores e trazendo resultados que demonstram os benefícios da aplicação destes projetos em uma empresa. Permitindo desta maneira, que decisões importantes em uma companhia sejam tomadas, e a partir deste ponto, permitindo o ajuizamento da gestão do projeto comercial, fazendo com haja a disponibilidade da utilização dessas ferramentas dentro da corporação, causando

melhoria nos resultados.

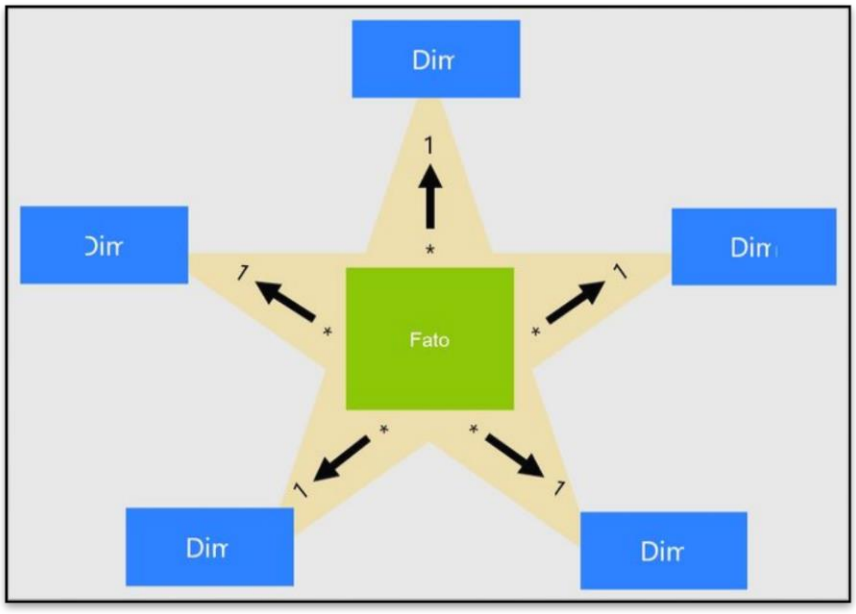
4 DESENVOLVIMENTO

Depois de uma análise profunda das necessidades do setor comercial da empresa “Archivery”, foram tomadas as primeiras medidas para a implementação do projeto de Business Intelligence, onde foram determinadas as etapas do desenvolvimento, iniciando pela definição dos conceitos de arquitetura utilizados na operação de criação do BI.

4.1 ARQUITETURA

A arquitetura escolhida para o desenvolvimento da operação é a de “Star Schema”, que é um modelo de arquitetura de Data Warehouse em que uma tabela de fato faz referência a várias tabelas de dimensões, que, quando vistas como um diagrama, parecem uma estrela, com a tabela de fatos no centro e as tabelas de dimensões irradiando dela. A figura 1 demonstra o conceito de Star Schema:

Figura 1 — Arquitetura Star Schema



Fonte: Microsoft Docs. Disponível em: <https://docs.microsoft.com/en-us/power-bi/guidance/star-schema> Acessado em 18 abr. 2021.

A arquitetura de Star Schema define os conceitos de:

- Tabelas Dim (Dimensão): As tabelas de Dimensão são representadas por

apenas uma tabela com um conjunto de atributos que definem uma dimensão do conceito final desejado. No caso da Archivery, as dimensões serão de Cliente, Produto, Devolução e Venda.

- Tabela Fato: A tabela de fato reunirá todas as tabelas de dimensão em um único produto final, alinhando e conectando as informações de todas as dimensões.

4.2 ETIDADES DO BANCO DE DADO

Uma entidade em um banco de dados é definida como um objeto real. As entidades pertencentes ao banco de dados da Archivery são as de Cliente, Produto, Vendas e Devoluções. As entidades possuem atributos que os identificam e os descrevem. A figura 2 demonstra a estrutura da entidade Cliente:

Figura 2 — Entidade Cliente

Entidade Cliente			
Atributo	**Tipo de dado**	**Tamanho**	**Descrição**
id_cliente	INT (PK)	10	Identificador do cliente
nome_cliente	VARCHAR	50	Nome do cliente
segmento	VARCHAR	50	Segmento do Cliente
cidade	VARCHAR	50	Cidade do Cliente
estado	VARCHAR	50	Estado do Cliente
regiao_br	VARCHAR	50	Região do Cliente
pais	VARCHAR	50	País do Cliente

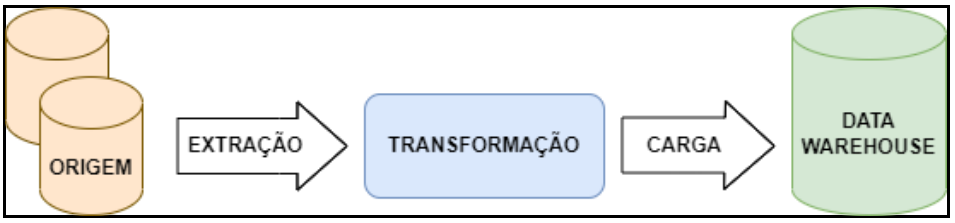
Fonte: Produção do próprio autor.

As entidades possuem atributos que os identificam e os descrevem. A figura 2 demonstra a estrutura da entidade Cliente.

4.3 EXTRAÇÃO, TRANSFORMAÇÃO E CARREGAMENTO

O ETL (Extract, Transform e Load) é um processo de integração de dados que combina dados de várias fontes de dados em um único armazenamento de dados consistente que é carregado em um data warehouse ou outro sistema de destino.

Figura 3 — Entidade Cliente



Fonte: Produção do próprio autor.

Sendo composto pelas regras:

- Extract (extração): Extração dos dados de diversas fontes (arquivos delimitados, banco de dados, aplicações web e etc).
- Transform (transformação): Aplicação das transformações dos dados (regras de negócio, limpeza dos dados, concatenações, agregações, cálculos e etc) direto no destino.
- Load (carregamento): Carga dos dados no destino. A figura 3 destaca o fluxograma de um ETL:

4.3 COMPONENTES DO BI

O Business Intelligence é formado por um grupo de componentes componentes principais para a sua infraestrutura. Eles são o Data Warehouse, o sistema responsáveis pelo conjunto de processos de extração e a análise incorporada por meio de uma ferramenta de visualização.

4.3.1 PENTAHO DATA INTEGRATION

O Pentaho Data Integration é formado por um conjunto recursos para a criação de um BI, que inclui programas para extrair os dados de sistemas de origem em uma empresa, gravá-los em um data warehouse (ou base de dados), limpá-los, prepará-los e entregá-los a outros sistemas de destino ou mesmo a outros componentes da suíte para estudar ou dar acesso aos dados ao usuário final. A ferramenta é formada por duas categorias de artefatos: Jobs e Transformações, e estes são construídos por meio de sua interface gráfica, o Spoon, que facilita na concepção de rotinas e lógica ETL, a descrição destes recursos se baseia em:

- Transformações: Uma transformação registra o passo-a-passo de como a extração ou leitura de uma fonte de informação é realizada.

- Jobs: Um Job sequência operações. Ao contrário de uma transformação, que opera sobre as linhas de dados em paralelo, um Job realiza operações completas, uma por uma. Ele permite, por exemplo, combinar transformações em uma sequência específica e, com isto, automatizar uma determinada tarefa.

4.3.2 MYSQL

MySQL é um sistema de gerenciamento de banco de dados relacional baseado em SQL – Structured Query Language. O sistema pode ser utilizado para uma ampla variedade de propósitos, focando na estruturação e armazenamento dos dados.

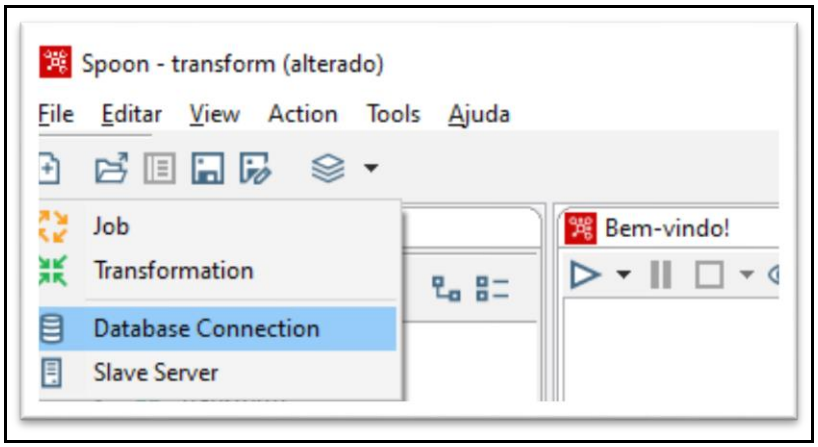
4.3.3 TABLEAU

A Tableau é uma empresa de software que oferece software colaborativo de visualização de dados para organizações que trabalham com análise de informações comerciais. As organizações usam o Tableau para visualizar dados e revelar padrões para análise em Business Intelligence, tornando os dados mais compreensíveis.

4.4 CONFIGURAÇÃO DA CONEXÃO COM O BANCO DE DADOS

Para coletar as informações de origem, é necessário configurar a conexão com o banco de dados, nesta etapa são solicitadas as informações do servidor onde os dados são hospedados, a inclusão é feita a partir da configuração do Database Connection, conforme visualização da imagem 4:

Figura 4 — Conector com banco de dados.



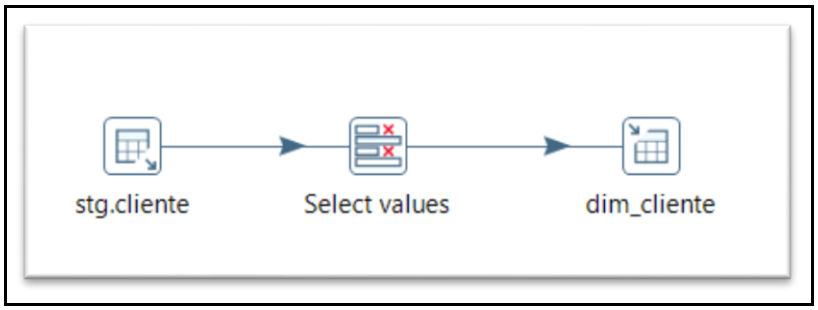
Fonte: Produção do próprio autor.

Inicialmente aplicamos o tipo de conexão utilizada, que descreve o nome do SGBD, no caso do projeto foi utilizado o MYSQL. Após isso, configuramos o Host Name, o nome aplicado ao do banco de dados, a porta utilizada para conexão, o nome do usuário e senha do administrador.

4.5 COLETA DE DADOS

Na transformação da entidade cliente, é definido o fluxograma que faz a carga final da tabela do cliente, este possui três etapas: Coleta de dados, transformação de valores e carga dos dados tratados na tabela final. O fluxo desta operação está destacado na figura 5.

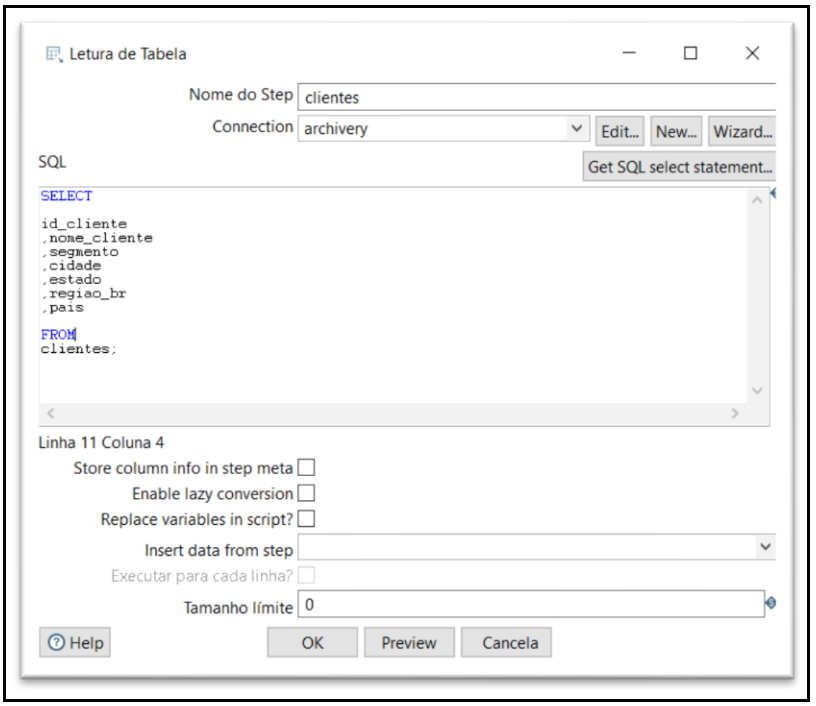
Figura 5 — Fluxograma de carga da tabela



Fonte: Produção do próprio autor.

Na primeira etapa, serão coletadas informações da tabela cliente através da função “Table Input”. Serão selecionados os campos desejados da tabela por meio de uma consulta SQL, demonstrado na figura 6.

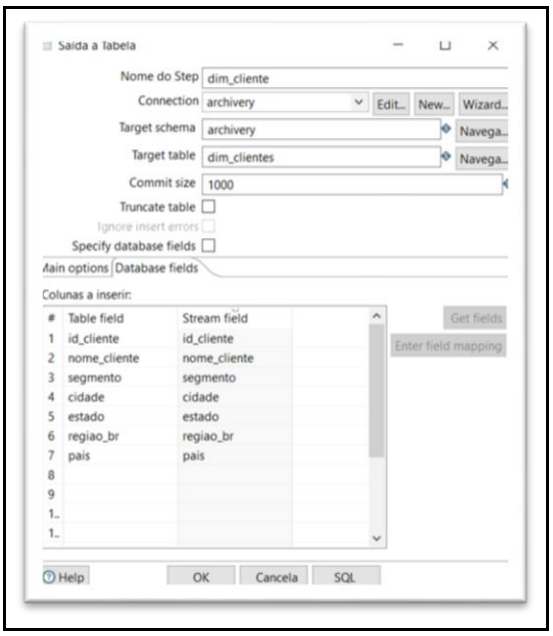
Figura 6 — Etapa de Table Input



Fonte: Produção do próprio autor.

A etapa de select values permite que o usuário altere uma informação específica de um campo da tabela, como por exemplo, alteração de campo, o tipo de dado, etc. Após a escolha destas informações.

Figura 6 — Etapa de Select Values



Fonte: Produção do próprio autor.

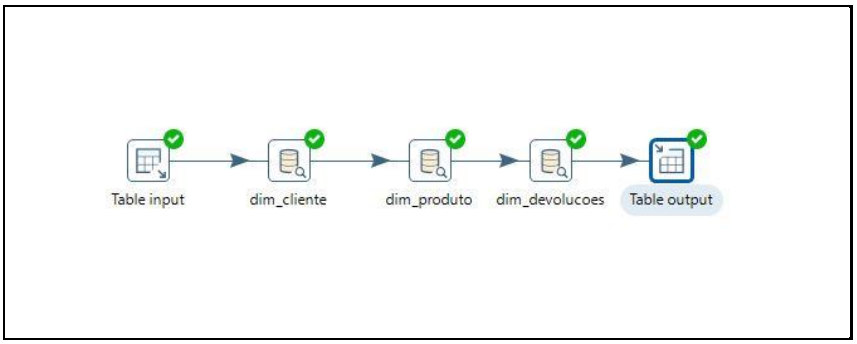
A carga final é feita por meio da função de Table Output, que coleta as

informações da etapa anterior e as carrega em uma tabela final, denominada como *dim_cliente*. Nesta fase, o usuário deve definir as conexões do banco de dados, bem como informar a nomenclatura da tabela final onde os dados serão carregados.

4.6 CARGA DA TABELA DE FATO

A carga da tabela de fato foi feita através da inicialização da tabela de dim_venda, selecionada no primeiro passo do fluxo de criação. Após isso, serão incluídas todas as tabelas de dimensão por meio de funções SQL de Lookup disponíveis na ferramenta do Pentaho. A figura 7 demonstra o fluxograma da carga da tabela de fato.

Figura 7 — Carga da tabela de fato



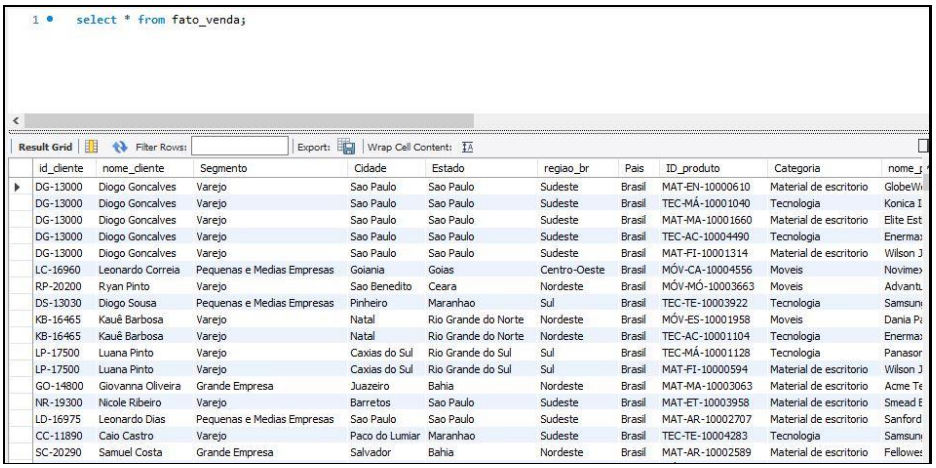
Fonte: Produção do próprio autor.

As funções presentes nesta etapa serão responsáveis por relacionar os campos desejados das tabelas por meio das chaves e carregá-las na tabela de fato, que recebeu a nomenclatura de *fato_venda*.

4.7 CONSULTA SQL DA TABELA DE FATO

A figura 8 demonstra uma consulta SQL aplicada na tabela de fato. Nela são demonstrados os campos selecionados de todas as dimensões, que estão reunidos em uma única fonte de informação, atingindo o objetivo desejado.

Figura 8 — Consulta SQL da tabela fato_venda.



```
1 •  select * from fato_venda;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_cliente	nome_cliente	Segmento	Cidade	Estado	regiao_br	Pais	ID_produto	Categoria	nome_p
DG-13000	Diogo Goncalves	Varejo	Sao Paulo	Sao Paulo	Sudeste	Brasil	MAT-EN-10000610	Material de escritorio	GlobeW
DG-13000	Diogo Goncalves	Varejo	Sao Paulo	Sao Paulo	Sudeste	Brasil	TEC-MÁ-10001040	Tecnologia	Konica I
DG-13000	Diogo Goncalves	Varejo	Sao Paulo	Sao Paulo	Sudeste	Brasil	MAT-MA-10001660	Material de escritorio	Elite Est
DG-13000	Diogo Goncalves	Varejo	Sao Paulo	Sao Paulo	Sudeste	Brasil	TEC-AC-10004490	Tecnologia	Enerma
DG-13000	Diogo Goncalves	Varejo	Sao Paulo	Sao Paulo	Sudeste	Brasil	MAT-FI-10001314	Material de escritorio	Wilson J
LC-16960	Leonardo Correia	Pequenas e Medias Empresas	Goiania	Goias	Centro-Oeste	Brasil	MÓV-CA-10004556	Moveis	Novime
RP-20200	Ryan Pinto	Varejo	Sao Benedito	Ceara	Nordeste	Brasil	MÓV-MÓ-10003663	Moveis	Advant
DS-13030	Diogo Sousa	Pequenas e Medias Empresas	Pinheiro	Maranhao	Sul	Brasil	TEC-TE-10003922	Tecnologia	Samsun
KB-16465	Kauê Barbosa	Varejo	Natal	Rio Grande do Norte	Nordeste	Brasil	MÓV-ES-10001958	Moveis	Dania P
KB-16465	Kauê Barbosa	Varejo	Natal	Rio Grande do Norte	Nordeste	Brasil	TEC-AC-10001104	Tecnologia	Enerma
LP-17500	Luana Pinto	Varejo	Caxias do Sul	Rio Grande do Sul	Sul	Brasil	TEC-MÁ-10001128	Tecnologia	Panasor
LP-17500	Luana Pinto	Varejo	Caxias do Sul	Rio Grande do Sul	Sul	Brasil	MAT-FI-10000594	Material de escritorio	Wilson J
GO-14800	Giovanna Oliveira	Grande Empresa	Juazeiro	Bahia	Nordeste	Brasil	MAT-MA-10003063	Material de escritorio	Acme Te
NR-19300	Nicole Ribeiro	Varejo	Barretos	Sao Paulo	Sudeste	Brasil	MAT-ET-10003958	Material de escritorio	Smead E
LD-16975	Leonardo Dias	Pequenas e Medias Empresas	Sao Paulo	Sao Paulo	Sudeste	Brasil	MAT-AR-10002707	Material de escritorio	Sanford
CC-11890	Caio Castro	Varejo	Paco do Lumiar	Maranhao	Sudeste	Brasil	TEC-TE-10004283	Tecnologia	Samsun
SC-20290	Samuel Costa	Grande Empresa	Salvador	Bahia	Nordeste	Brasil	MAT-AR-10002589	Material de escritorio	Fellowes

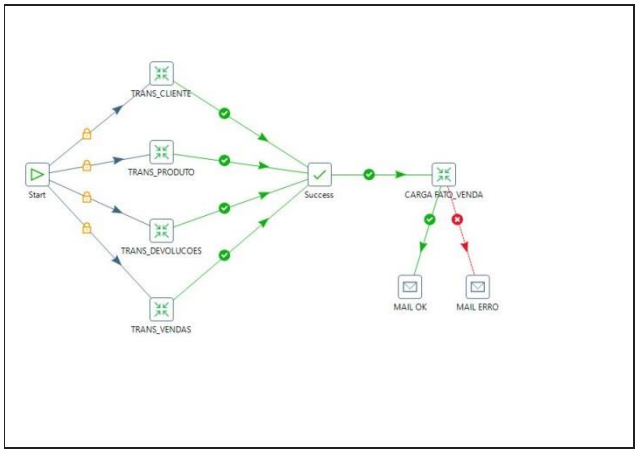
Fonte: Produção do próprio autor.

Esta tabela será objeto de aplicação das visualizações das métricas de venda da empresa.

4.8 AUTOMATIZAÇÃO DE CARGAS

Nesta etapa foi criado um Job no Pentaho que é responsável por automatizar o processo de carga das tabelas. Nele são definidos comandos que seguem o fluxo do processo de automação, o primeiro comando de Start que é responsável por iniciar o script de execução, logo após serão setadas todas as dimensões a serem executadas. Após isso, é aplicado um passo de validação, que identifica se as cargas de todas dimensões foram executadas com sucesso, e caso a informação seja verdadeira, o processo de carga da tabela de fato é feito. A figura 9 destaca o fluxograma do Job:

Figura 9 — Carga da tabela de fato



Fonte: Produção do próprio autor.

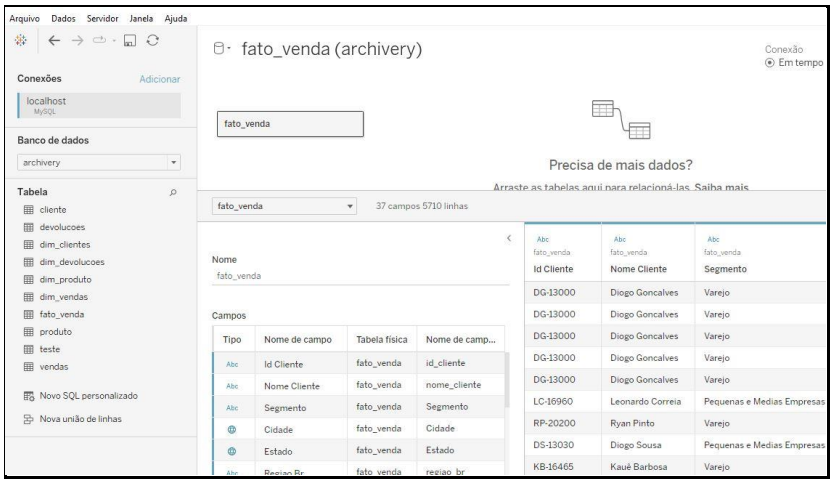
No processo, serão aplicados os steps que permitem o envio de informações da

execução do Job via e-mail. Caso tenha sido feita com sucesso, o passo de “E-Mail Ok” permite definir uma mensagem personalizada para o destinatário, caso contrário, uma mensagem de erro pode ser enviada.

4.9 VISUALIZAÇÕES DE MÉTRICAS

A tabela fato_venda será carregada no Tableau por meio da conexão com o banco de dados “Archivery”. A ferramenta solicita informações de servidor do banco de dados, e após isto, faz a conexão. Com isso, são apresentadas uma amostra dos dados da tabela, concedendo ao usuário permissão de alterações de campo, junção com outras tabelas, inclusão de filtros, etc. A figura 10 expõe esta etapa.

Figura 10 — Carga da tabela de fato



Fonte: Produção do próprio autor.

A partir da inclusão da fonte de dados, com as informações consolidadas no Tableau, o usuário poderá criar diferentes relatórios e visualizações dos dados, que sejam de acordo com as regras de negócio desejadas.

4.10 RELATÓRIO DE VENDAS

O relatório de vendas da Archivery foi desenvolvido baseado no conceito de exibição de dados por período, permitindo ao usuário filtrar a data específica ao qual deseja visualizar as informações. A figura 11 demonstra o resultado da criação.

Figura 11 — Relatório de Vendas

RELATÓRIO DE VENDAS

Data da compra 04/01/2019 30/04/2022

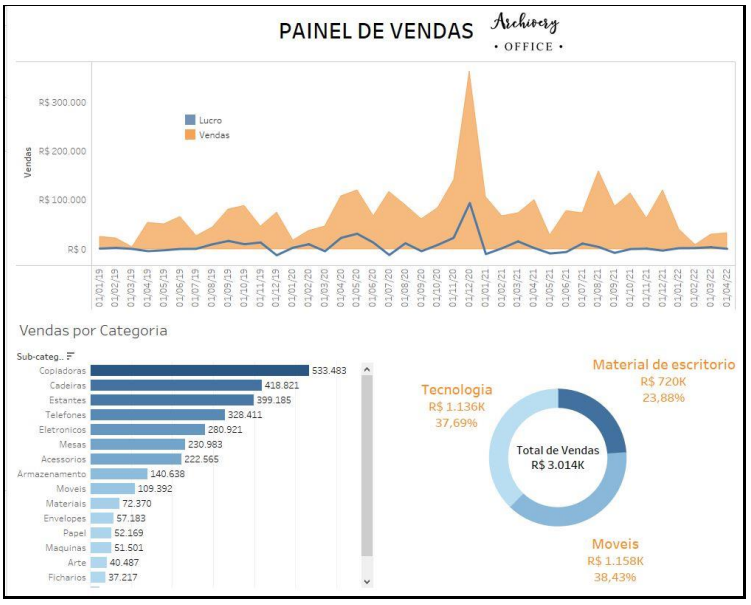
ID do pro..	Nome do produto	ID da compra	Mês de Dat..	Quantida..	ID do cliente	Nome do cliente	Segmento	Cidade
MAT-AC-..	Hamilton Beach Gel..	MX-2019-136476	01/11/2019	3	RF-19915	Rebeca Fernandes	Varejo	Itamaraju
MAT-AC-..	Hoover Geladeira, V..	MX-2021-158106	01/04/2021	4	AL-10210	Alex Lima	Varejo	Colombo
MAT-AC-..	Hamilton Beach Gel..	MX-2021-137568	01/03/2021	2	RD-19900	Rebeca Dias	Varejo	Sumare
MAT-AC-..	KitchenAid Liquidific..	MX-2020-151680	01/05/2020	3	MM-18490	Marisa Martins	Pequenas e Medias ..	Curitiba
MAT-AC-..	Cuisinart Moedor de..	MX-2020-131002	01/01/2020	3	BG-11545	Breno Gomes	Grande Empresa	Aracaju
MAT-AC-..	Breville Forno, Verm..	MX-2019-152037	01/09/2019	2	LF-17320	Livia Ferreira	Pequenas e Medias ..	Bage
MAT-AC-..	Cuisinart Geladeira, ..	MX-2020-154179	01/12/2020	28	ES-14080	Evelyn Sousa	Grande Empresa	Sao Paulo
MAT-AC-..	Breville Liquidificad..	MX-2019-120397	01/09/2019	2	MA-18790	Matilde Alves	Pequenas e Medias ..	Cotia
MAT-AC-..	KitchenAid Forno, Pr..	US-2019-159436	01/04/2019	3	EA-13480	Emily Alves	Pequenas e Medias ..	Limoeiro do Norte
		US-2020-108567	01/09/2020	6	LR-17245	Leticia Rocha	Varejo	Ponte Nova
MAT-AC-..	Breville Torradeira, ..	MX-2020-168130	01/12/2020	2	AC-10390	Aline Correia	Varejo	Sao Paulo
MAT-AC-..	Hamilton Beach Liqu..	MX-2020-140767	01/02/2020	3	RA-19795	Raissa Azevedo	Pequenas e Medias ..	Valinhos
MAT-AC-..	Hamilton Beach Liqu..	US-2021-125675	01/03/2021	1	LP-17500	Luana Pinto	Varejo	Bezerros
MAT-AC-..	Hoover Micro-ondas..	MX-2019-152562	01/09/2019	2	BR-11185	Beatriz Ribeiro	Grande Empresa	Guarulhos
MAT-AC-..	Breville Geladeira, V..	MX-2020-162411	01/07/2020	5	BG-11710	Bruna Goncalves	Varejo	Sao Bernardo do Ca..
		MX-2022-166625	01/03/2022	2	LR-16675	Laura Ribeiro	Grande Empresa	Barueri
MAT-AC-..	KitchenAid Geladeir..	MX-2020-157987	01/08/2020	7	BS-11740	Bruna Silva	Varejo	Bayeux
MAT-AC-..	Hoover Torradeira, ..	MX-2019-145597	01/01/2021	3	VS-21115	Victor Souza	Grande Empresa	Curitiba
		MX-2021-117450	01/05/2021	2	BG-11695	Bruna Gomes	Grande Empresa	Brasilia
MAT-AC-..	Hoover Liquidificado..	US-2019-139150	01/10/2019	1	CS-12220	Carla Silva	Varejo	Presidente Dutra
MAT-AC-..	Breville Forno, Prata	MX-2021-121755	01/02/2021	2	JG-15670	Julia Goncalves	Varejo	Curitiba
MAT-AC-..	Cuisinart Micro-ond..	US-2021-150574	01/06/2021	6	VB-21175	Vinicius Barbosa	Grande Empresa	Sao Miguel dos Cam..
MAT-AC-..	Breville Moedor de c..	MX-2021-108063	01/04/2021	8	LB-16930	Leonardo Barros	Grande Empresa	Osasco
MAT-AC-..	Hamilton Beach Torr..	MX-2019-126648	01/06/2019	3	CC-12280	Carlos Correia	Varejo	Recife
MAT-AC-..	Hoover Moedor de c..	MX-2019-115609	01/06/2019	7	LC-16780	Lavinia Carvalho	Grande Empresa	Sao Goncalo
MAT-AC-..	KitchenAid Geladeir..	US-2021-167850	01/08/2021	9	AM-10330	Alice Martins	Varejo	Indaial
MAT-AC-..	Hoover Geladeira, P..	MX-2021-137855	01/01/2021	3	BC-11140	Beatriz Castro	Varejo	Itajai
MAT-AC-..	Hoover Forno, Branco	MX-2019-102421	01/11/2019	3	DA-12610	Daniel Azevedo	Grande Empresa	Chapeco
MAT-AC-..	Breville Moedor de c..	MX-2020-151680	01/05/2020	3	MM-18490	Marisa Martins	Pequenas e Medias ..	Curitiba
MAT-AC-..	Cuisinart Micro-ond..	MX-2020-169040	01/08/2020	2	BA-11590	Bruna Almeida	Grande Empresa	Cajazeiras
MAT-AC-..	Hamilton Beach For..	US-2020-131793	01/12/2020	2	LS-17620	Lucas Souza	Grande Empresa	Bezerros
MAT-AC-..	Hoover Forno, Preto	MX-2019-120054	01/08/2019	2	SR-20560	Sofia Ribeiro	Pequenas e Medias ..	Taubate
MAT-AC-..	Hoover Micro-ondas..	US-2020-131793	01/12/2020	4	LS-17620	Lucas Souza	Grande Empresa	Bezerros

Fonte: Produção do próprio autor.

4.11 DASHBOARD VENDAS

Um Dashboard (painel) de vendas é uma representação gráfica de fácil leitura dos dados de vendas. Visa permitir que as pessoas tomem melhores decisões em uma companhia. Os painéis podem fornecer aos funcionários uma visão atualizada da saúde de sua organização por meio de métricas de vendas, para que possam fazer previsões de receita informadas, e identificar as principais áreas para otimização.

Figura 12— Painel de Vendas



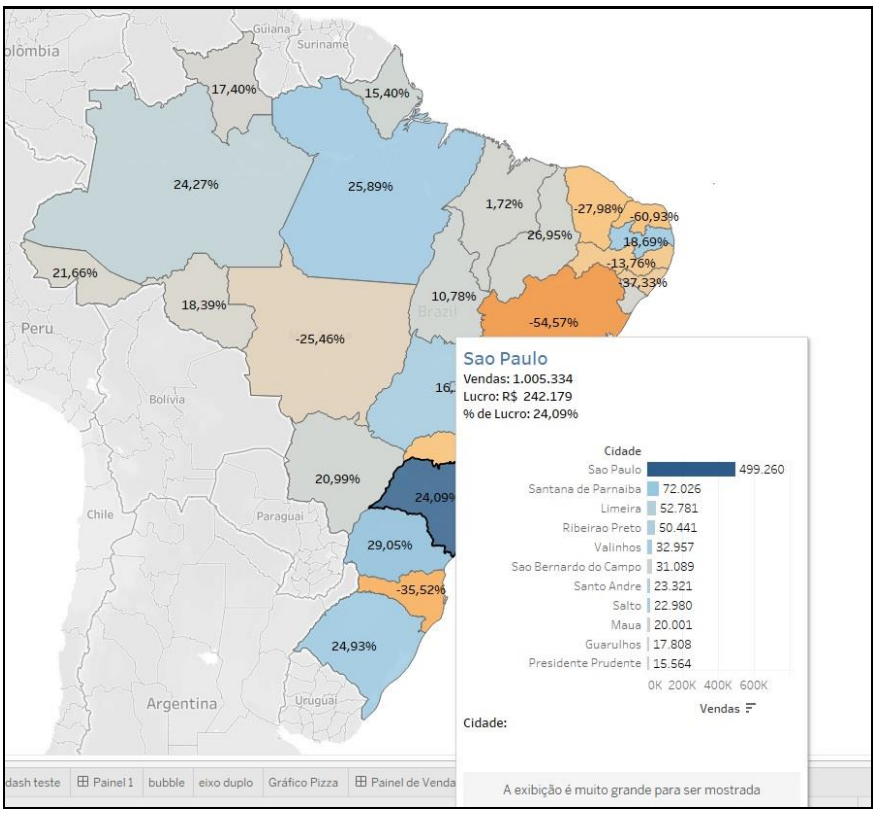
Fonte: Produção do próprio autor.

Para a Archivery foi criado um Dashboards de Vendas, que verifica a análise das métricas de vendas por região, e sob os produtos vendidos. Na imagem 12 podemos visualizar o painel de Venda, demonstrando informações de maneira intuitiva ao usuário, que ao visualizá-lo, terá um resumo geral acerca da receita da empresa. Nesta análise, destacamos as informações de Venda e Lucro, demonstrando por meio de cores que variam que geram uma percepção intuitiva ao usuário.

4.10.1 ANÁLISE GEOGRÁFICA

A figura 13 destaca a análise geográfica dinâmica que permite que ao arrastar o mouse, o usuário possa visualizar detalhes das métricas de vendas para cada estado.

Figura 13— Visão geográfica



Fonte: Produção do próprio autor.

Destacam-se as informações de receita, lucro, e ranking com as cidades que obtiveram maior volume de vendas.

5 CONSIDERAÇÕES FINAIS

O objetivo deste projeto foi realizar um estudo de caso da aplicação de um sistema de Business Intelligence aplicado ao setor comercial de uma empresa. Tendo como objetivo prover uma tecnologia que garanta que a influência nas decisões de negócios de uma organização seja baseada em dados provenientes de recursos de softwares. Esse processo orientado por tecnologia tem o poder de ajudar os usuários finais corporativos a analisar melhor o estado de seus negócios e fornecer informações que contribuam para uma melhor tomada de decisão, oferecendo uma maneira eficaz de aprender sobre as tendências atuais e escolhas de negócios.

O uso da tecnologia na Inteligência de negócios pode ser uma ferramenta inestimável para a equipe de vendas de uma empresa, pois fornece aos funcionários informações atualizadas sobre métricas, tendências atuais, perfil dos clientes, melhorias de produtos e outros conceitos. E o objetivo deste projeto foi trabalhar estes conceitos exemplificando o uso de cada tecnologia presente no desenvolvimento do projeto, desde a coleta dos dados à visualização das métricas.

REFERÊNCIAS

BOUMAN, R.; VAN DONGEN, J. **Pentaho solutions: Business intelligence e data warehousing com pentaho e MySQL**. Wiley: Edição, ago. 2019.

GARTNER. **Glossary. Dashboards**. Disponível em: https://www.gartner.com/en/information-technology/glossary/dashboard Acesso em: 12 mai. 2022.

INMON, W H; WELCH, J D; GLASSEY, K, L . **Gerenciando Data Warehouse.** São Paulo: Editora Makron Books, 1999.

MACHADO, F. N. R. **Tecnologia e projeto de Data Warehouse: uma visão multidimensional**. São Paulo: Érica, 2004.

MYERS, Peter. **Understand star schema and the importance for Power BI**. Microsoft Docs. Disponível em: https://docs.microsoft.com/en-us/power-bi/guidance/star-schema. Acesso em: 18 abr. 2022.

PENTAHO CORPORATION. **Pentaho Data Integration**. Disponível em: <https://www.hitachivantara.com/en-us/products/data-management-analytics>. Acessado em: 20 abril. 2022.

Teo, T.S.H. & Choo, W.Y. (2001) **Assessing the impact of using the Internet for competitive intelligence. Information and Management**, 67-83.

DESENVOLVIMENTO DE UMA API PARA GERENCIAMENTO DE TORNEIOS DE CARD GAME

DONATO, Leonardo Victor Andrade
ROCHA, Gláucio Bezerra

RESUMO

O presente trabalho teve como objetivo solucionar problemas no que diz respeito à organização e ao pareamento dos competidores de forma justa em eventos de *card games* e torneios presenciais, fornecendo uma API para que possam ser organizados e planejar seus torneios, cadastrar jogadores, realizar o ranqueamento de uma região e ter facilmente acesso ao histórico de torneios de determinado lugar. O projeto desenvolvido através da abordagem das definições de API e Banco de Dados, bem como suas aplicabilidades em mais de uma linguagem de programação, a C# e a SQL, as quais foram combinadas para que fosse possível a concepção desse projeto de forma acessível aos usuários como foi proposto.

Palavras-chave: *Cardgame*; API; C#

1 INTRODUÇÃO

Atualmente, existe uma grande variedade de *card games* (jogos de carta), que também são conhecidos como *TCG* (*Trading Card Game*) ou *CCG* (*Collectible Card Game*), dentre eles os mais conhecidos são *Magic: The Gathering, Pokémon Trading Card Game* e *Yu-Gi-Oh!*, onde várias pessoas das mais variadas idades se reúnem para trocar cartas, conversar sobre o jogo e até mesmo competir.

Como esse gênero trata-se majoritariamente de cartas físicas, muitos encontros e eventos são realizados presencialmente para que os jogadores possam interagir com os outros jogadores da comunidade e haja uma competição de forma justa.

No *Yu-Gi-Oh!* e no *Magic: The Gathering,* esses torneios são realizados utilizando o sistema Suíço, ou seja, os jogadores são colocados em pares para que compitam entre si, esse sistema é utilizado oficialmente pois permite que vários participantes estejam incluídos nesses torneios sem a necessidade de muitas rodadas ou eliminação por derrota durante o torneio.

Popularmente conhecidas simplesmente como loja, os torneios geralmente são realizados em lugares referidos oficialmente como *OTS (Official Tournament Store)* e quando são realizados competições oficiais, a própria organização dispõe

de programas e meios para que sejam realizados o pareamento de chaves de forma justa e imparcial, porém para competições amadoras ou jogadores que não estão competindo de forma oficial na *OTS* há uma grande dificuldade em realizar a distribuição de chaves e pareamento de forma a garantir a idoneidade dos organizadores das competições.

Para suprir essa necessidade, será criada uma Interface de Programação de Aplicações, também conhecida como API, com isso é buscado resolver exatamente esse empecilho, fornecendo um histórico de partidas para cada jogador e um histórico de torneios realizados por organização, uma classificação geral de cada jogador em determinada região ou lugar, um perfil de usuário e organização para cadastro, além de outra funcionalidades para que jogadores possam organizar seus próprios torneios sem a necessidade de ter que ir a alguma *OTS*.

Juntamente da API, será utilizado um Sistema de Gerenciamento de Dados Relacional, ou SGBDR, para que todos os dados utilizados na aplicação sejam persistidos e posteriormente acessados por essa aplicação.

Nas próximas seções será discutido sobre o método utilizado para o desenvolvimento da *API*, além de apresentar uma explicação detalhada sobre o funcionamento da mesma, levando em conta critérios de desempate para o sistema suíço, bem como será feito o armazenamento dos torneios, informações dos jogadores e dos organizadores de cada evento.

2 FUNDAMENTAÇÃO TEÓRICA

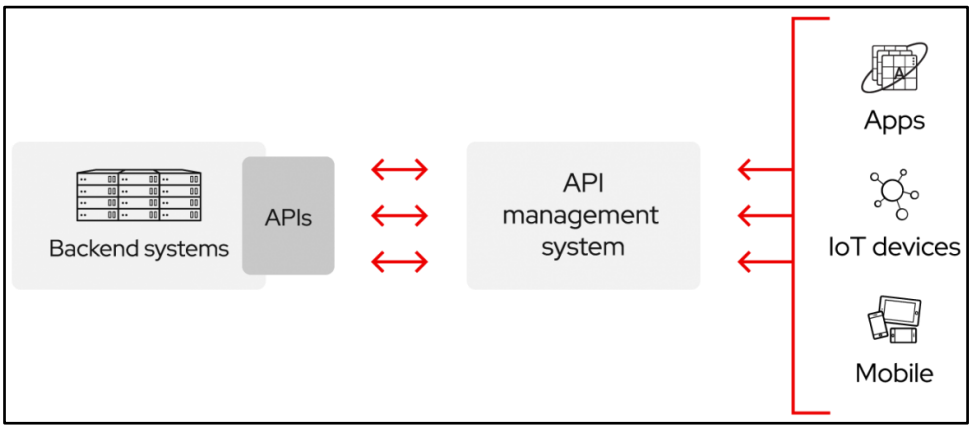
Neste capítulo serão apresentados as técnicas e procedimentos utilizados para o desenvolvimento de uma API, bem como as ferramentas e linguagens utilizadas para este projeto. Com isso, os próximos subcapítulos abordarão cada passo necessário para que todo esse projeto seja possível.

A API foi desenvolvida utilizando a linguagem de programação C#, utilizando a plataforma *Visual Studio Code*, disponibilizada pela *Microsoft* e *frameworks* que apoiam a criação de recursos necessários para que haja a integração da aplicação com um banco de dados *SQL.*

2.1 API

Conhecida como Interface de Programação de Aplicações, a API tem como objetivo facilitar a comunicação entre diversas soluções ou serviços, permitindo que novas implementações sejam criadas sem que haja a necessidade de os desenvolvedores conhecerem como foi implementada nenhuma das outras. Tal configuração pode ser vista na figura abaixo.

Figura 1: Representação do funcionamento de uma API



Fonte: RedHat (2022)

Dessa forma, vários dispositivos podem se comunicar com a mesma API sem que haja a necessidade de uma configuração específica para cada tipo de dispositivo ou uma resposta específica para cada um, tendo apenas a mesma resposta e então será tratada por cada solução individualmente.

2.2 SGBD

Sistemas de Gerenciamento de Banco de Dados Relacional é uma implementação que também é chamada genericamente de SGBD, onde existem várias formas de utilização da mesma, porém os SGBDRs são chamados assim pois é o tipo de banco de dados que modela os dados de forma que sejam percebidos pelo usuário como tabelas.

Um banco de dados relacional utiliza uma forma de relacionar os objetos da tabela entre si através de seus identificadores únicos, podendo assim através de um requerimento de uma aplicação de gerenciamento de banco de dados, acessar

todos os dados relacionados a esse objeto através de consultas em outras tabelas utilizando-se do identificador de cada dado.

2.3 C#

Pensado como uma alternativa para a linguagem de programação *Java*, o C# foi desenvolvido pela *Microsoft* e lançado juntamente com a plataforma também desenvolvida pela mesma, o *Visual Studio .NET 2002,* sendo considerado “como parte de suas metas de design declaradas para ECMA, ela buscava ser uma "linguagem simples, moderna e orientada a objeto de uso geral” (MICROSOFT, 2022).

2.4 Entity Framework Core

Desenvolvido em 2016, “O EF (Entity Framework) Core é uma versão leve, extensível, de software livre e multiplataforma da popular tecnologia de acesso a dados do Entity Framework.” (MICROSOFT, 2022). É a nova versão do *Entity Framework* depois da versão 6.

O EF Core é um *framework* de mapeador relacional de objeto, que fornece aos desenvolvedores mecanismos e ferramentas automatizadas para acessar e armazenar dados em um banco de dados. São suportados dois tipos de abordagem de desenvolvimento:

- ***Code-first***: Com essa abordagem, o EF Core cria um banco de dados e as tabelas usando convenções de migração e configuração fornecidas pelas classes de domínio. Desenvolvedores que seguem os princípios do *Domain Driven Design* (DDD) preferem começar com a criação das classes de domínio primeiro e depois gerar o banco de dados necessário para persistir os dados.
- ***Database-First***: As classes de domínio e contexto são criadas baseando-se em uma base de dados existente utilizando os comandos do EF Core, essa abordagem tem suporte limitado pois não dispõe de recursos da versão em que foi derivado, o *Entity Framework 6.*

2.5 Microsoft Visual Studio

Disponível para o público como um Ambiente de Desenvolvimento Integrado (IDE) com foco na plataforma .NET desde 2002, o *Visual Studio* tem foco nas linguagens *Visual Basic*, C, C++, C# e F#, também é utilizado para desenvolvimento de aplicações *Web*, com a plataforma ASP.NET.

Disponibilizando vários recursos para auxiliar no desenvolvimento, o *Visual Studio* dispõe de:

- *IntelliCode***:** É um sistema de inteligência artificial que oferece sugestões com base em milhares de outros projetos de código aberto.
- Testes feitos enquanto o código é alterado para que o desenvolvedor saiba exatamente o impacto que cada alteração está causando dentro da aplicação
- Integração com serviços em nuvem como o *Microsoft Azure*

2.6 SQL Server

Lançado em 1988, o SQL Server é um sistema de gerenciamento de banco de dados (SGBD) que foi desenvolvido pela *Microsoft* em parceria com a *Sybase* que, a partir do ano 2000, passou a ter mais fama e confiança no mercado.

Desta forma, devido a seu histórico no mercado e sua popularidade, este gerenciador foi escolhido para ser utilizado para o desenvolvimento deste projeto, oferecendo suporte para o C# além da performance oferecida pelo mesmo, para que a aplicação disponha de uma alta performance e possa realizar as pesquisas e acesso aos dados de forma que não haja empecilhos de processamento.

3 FUNDAMENTAÇÃO TEÓRICA

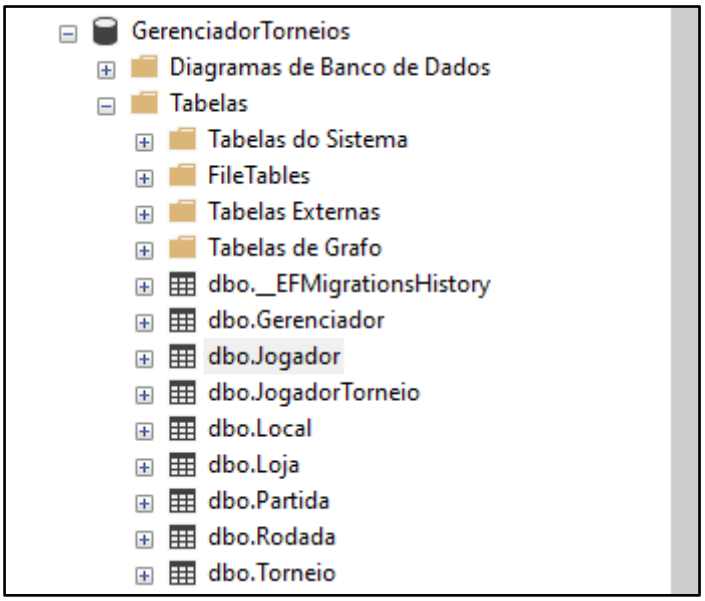
Neste capítulo será apresentado todas as tecnologias utilizadas e discutidas as implementações de cada padrão de projeto, além de uma breve explicação do funcionamento do código de forma que fique claro para o entendimento da API.

3.1 Banco de Dados

Um sistema de banco de dados é responsável pelo armazenamento e gerenciamento de dados, onde estes ficam geralmente guardados em tabelas e colunas e depois resgatados através de um sistema de gerenciamento de banco de dados.

A seguir, na figura 2, veremos como estão dispostas as tabelas do banco de dados da aplicação.

Figura 2: Banco de dados da API



Fonte: do próprio autor

As tabelas do banco de dados foram geradas através da função *Migrations* disponível pelo EF Core, com ela foi possível criar todas as tabelas e definir todas as chaves primárias e os relacionamentos entre cada entidade no banco de dados.

3.2 API

Com a API, alguns padrões de projeto foram implementados, como é o caso do *Unit of Work,* Repositório-Serviço e o *AutoMapper.*

3.2.1 *Unit of Work*

Com a implementação do *Unit of Work,* é necessário apenas a implementação de um repositório genérico, pois esta classe ficará responsável por gerenciar toda a comunicação entre a classe de Serviço e os Repositórios, desacoplando as classes e permitindo que cada Serviço possa implementar este padrão sem a necessidade de conhecer que tipo de banco de dados está sendo utilizado.

Figura 2: Representação do padrão *Unit of Work*

```
namespace GerenciadorTorneios.Infra.UnitOfWork
{

    public class UnitOfWork : IUnitOfWork, IDisposable
    {
        private Dictionary<Type, object> _repositories;
        private readonly DataContext _context;

        public UnitOfWork(DataContext context)
        {
            _context = context;
            _repositories = new Dictionary<Type, object>();

        }

        public void Dispose()
        {
            GC.SuppressFinalize(this);
        }
        5 referências
        public int Save()
        {
            return _context.SaveChanges();
        }

        14 referências
        public IBaseRepository<TEntity> GetRepository<TEntity>() where TEntity : class
        {
            var type = typeof(TEntity);
            if (!_repositories.ContainsKey(type)) _repositories[type] = new BaseRepository<TEntity>(_context);
            return (IBaseRepository<TEntity>)_repositories[type];
        }
    }
}
```

Fonte: do próprio autor (2022)

Desta forma, com todos os serviços implementando este padrão, será possível acessar todos os métodos genéricos disponíveis com o repositório base além de seus próprios métodos específicos de acordo com a necessidade de cada entidade e seus serviços, pois com o método *GetRepository* representado pela figura 2, todos os serviços poderão utilizar os repositórios de suas respectivas entidades.

3.2.2 Repositório-Serviço

Este padrão trata de separar as responsabilidades entre os métodos que interferem com as especificidades de cada classe para em seguida realizar a transação entre o banco de dados.

Figura 3 : Classe *BaseService* que todas as classes de serviço devem implementar

```
namespace GerenciadorTorneios.Services.BaseService
{
    15 referências
    public class BaseService<T> : IBaseService<T> where T : class
    {
        private readonly IUnitOfWork _uow;

        7 referências
        public BaseService(IUnitOfWork uow)
        {
            _uow = uow;
        }

        8 referências
        public T Add(T obj)...

        1 referência
        public IEnumerable<T> GetAll()...
        12 referências
        public IEnumerable<T> GetAll(
            Expression<Func<T, bool>>? predicate = null,
            Func<IQueryable<T>, IOrderedQueryable<T>>? orderBy = null,
            Func<IQueryable<T>, IIncludableQueryable<T, object>>? include = null,
            int index = 0,
            int size = 0,
            bool disableTracking = true
        )...

        18 referências
        public T GetById(int id)...

        9 referências
        public T Update(T obj)...

        1 referência
        public void Delete(int id)...

        8 referências
        public void Delete(T entity)...

    }
```

Fonte: do próprio autor

Este padrão aliado ao *Unit of Work* se torna ainda mais poderoso pois todo o projeto fica desacoplado sem que haja a necessidade de alterações por todo o código caso haja a mudança de alguma funcionalidade

3.2.3 *Automapper*

Devido a natureza sigilosa de alguns dados, é necessário que haja representações da mesma entidade de maneiras diferentes para evitar que algum dado de natureza sensível seja exposto a pessoas não autorizadas.

Figura 4: Configuração das classes do Automapper

```
namespace GerenciadorTorneios.Application.AutoMapper
{
    2 referências
    public class DataMappingProfile : Profile
    {
        0 referências
        public DataMappingProfile()
        {

            CreateMap<Loja, LojaView>().ReverseMap();
            CreateMap<Loja, LojaDTO>().ReverseMap();
            CreateMap<Local, LocalView>().ReverseMap();
            CreateMap<Local, LocalDTO>().ReverseMap();
            CreateMap<Gerenciador, GerenciadorView>()
                .ForMember(dst => dst.TipoGerenciador, map => map.MapFrom(src => (TipoGerenciadorEnum)src.TipoGerenciador));
            CreateMap<Gerenciador, GerenciadorDTO>().ReverseMap();
            CreateMap<Jogador, JogadorView>()
                .ForMember(dst => dst.TipoJogo, map => map.MapFrom(src => (JogoEnum)src.TipoJogo));
            CreateMap<Jogador, JogadorDTO>().ReverseMap();
            CreateMap<Partida, PartidaView>()
                .ForMember(dst => dst.NomeVencedor, map => map.MapFrom(src => src.Vencedor.Nome))
                .ForMember(dst => dst.NomeVencedor, map => map.MapFrom(src => src.Perdedor.Nome));
            CreateMap<Partida, PartidaDTO>().ReverseMap();
            CreateMap<Rodada, RodadaView>().ReverseMap();
            CreateMap<Rodada, RodadaDTO>().ReverseMap();
            CreateMap<Torneio, TorneioDTO>().ReverseMap();
            CreateMap<Torneio, TorneioView>()
                .ForMember(dst => dst.TipoJogoString, map => map.MapFrom(src => (JogoEnum)src.TipoJogo))
                .ForMember(dst => dst.TierTorneioString, map => map.MapFrom(src => (TierTorneioEnum)src.TierTorneio));
        }
    }
}
```

Fonte: do próprio autor

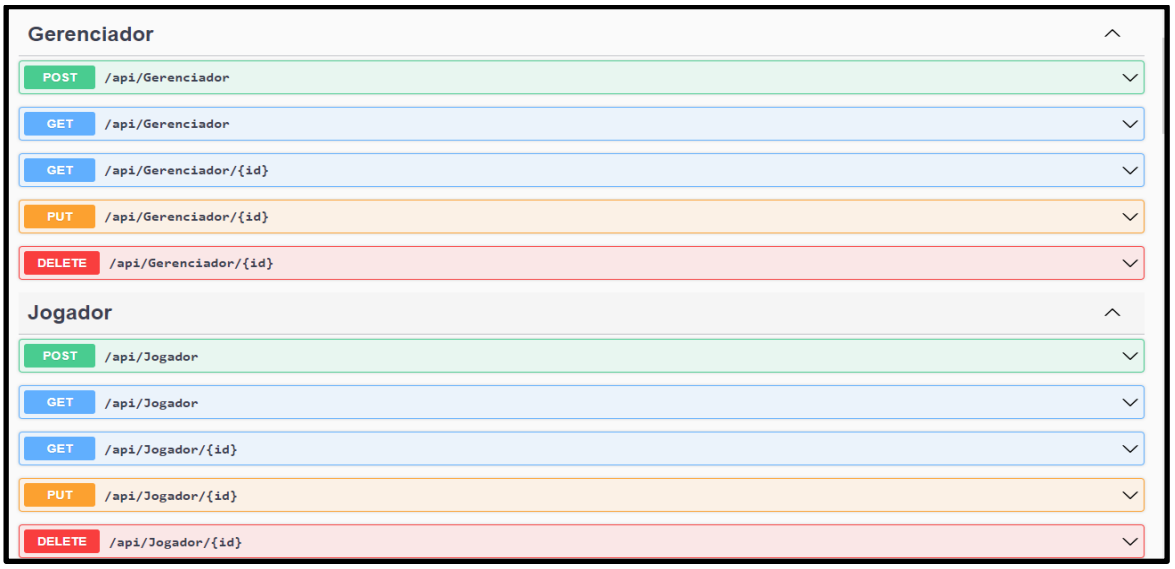
Com a utilização do *Automapper* este trabalho fica por conta de uma classe especializada na conversão destas classes evitando assim que sejam reescritas várias linhas de códigos. Além disso, desacoplando as classes de Entidade, *View* e *Data Access Object* (DTO), não há a necessidade de alterar vários métodos caso haja a adição, remoção ou alteração de algum atributo das classes pois o *Automapper* ficará responsável por essa conversão.

3.3 *Swagger*

Swagger é um *framework* que auxilia na representação, testes e documentação de uma API. Com o *Swagger* é possível ver todos os métodos e verbos que uma aplicação e seus *endpoints* utilizam, bem como o objeto esperado

e o retorno de cada um.

Figura 5: Tela do *Swagger* com os endpoints



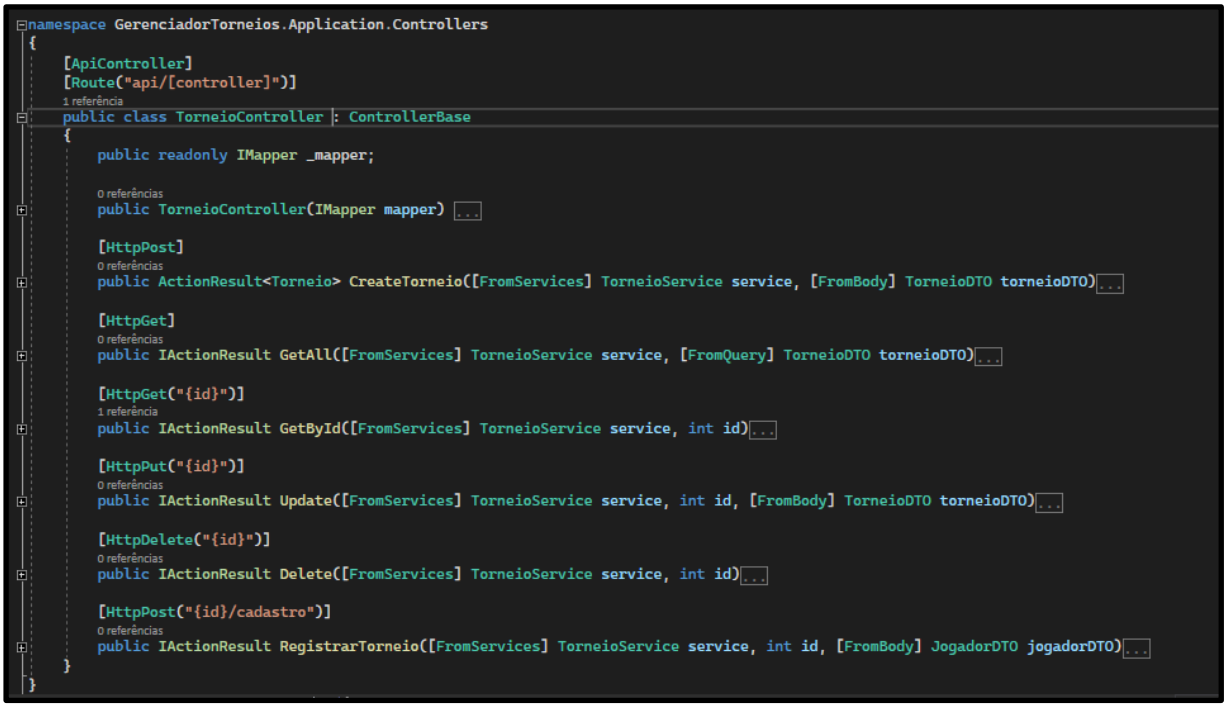
Fonte: do próprio autor

Graças a este *framework*, é possível ver e testar todos os *endpoints* facilmente e vermos a resposta dos mesmos com uma interface amigável e visual.

3.4 *Controllers*

É através dos *controllers*, ou controladores, que as aplicações conseguem se conectar e conversar com outros projetos, com seus métodos e verbos é possível que qualquer outra aplicação possa trocar dados, deletar ou alterar registros e salvar entidades no sistema. Com isso, outros projetos podem implementar e salvar resultados sem a necessidade de conhecer internamente outro projeto em profundidade.

Figura 6: Controlador do Torneio



```
namespace GerenciadorTorneios.Application.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    1 referência
    public class TorneioController : ControllerBase
    {
        public readonly IMapper _mapper;

        0 referências
        public TorneioController(IMapper mapper) ...

        [HttpPost]
        0 referências
        public ActionResult<Torneio> CreateTorneio([FromServices] TorneioService service, [FromBody] TorneioDTO torneioDTO)...

        [HttpGet]
        0 referências
        public IActionResult GetAll([FromServices] TorneioService service, [FromQuery] TorneioDTO torneioDTO)...

        [HttpGet("{id}")]
        1 referência
        public IActionResult GetById([FromServices] TorneioService service, int id)...

        [HttpPut("{id}")]
        0 referências
        public IActionResult Update([FromServices] TorneioService service, int id, [FromBody] TorneioDTO torneioDTO)...

        [HttpDelete("{id}")]
        0 referências
        public IActionResult Delete([FromServices] TorneioService service, int id)...

        [HttpPost("{id}/cadastro")]
        0 referências
        public IActionResult RegistrarTorneio([FromServices] TorneioService service, int id, [FromBody] JogadorDTO jogadorDTO)...
    }
}
```

Fonte: do próprio autor

.Conforme a figura 6, temos um exemplo do controlador para a entidade Torneio, que utiliza o seu respectivo serviço e com isso é possível realizar a implementação dos métodos e a comunicação entre a aplicação e o banco de dados.

4 CONSIDERAÇÕES FINAIS

A primeira parte deste trabalho teve como objetivo o desenvolvimento de uma aplicação *back-end* para auxiliar no gerenciamento de torneios, utilizando a plataforma fornecida pela *Microsoft*, o *Visual Studio*, com o auxílio de ferramentas para auxiliar o desenvolvimento da mesma, como o *Swagger* e o *Sql Server*. Dadas as dificuldades encontradas para pessoas e lojas que queiram organizar torneios mas não dispõe de meios necessários para que tudo possa ocorrer devidamente documentado e certificado da idoneidade em relação à forma que o evento tenha ocorrido.

A produção desta aplicação incluiu diversos desafios que o autor precisou obter mais conhecimento e estudar mais a fundo tecnologias presentes no mercado, como implementação de diversos conceitos ensinados nas disciplinas no

curso de Sistemas para Internet.

5 TRABALHOS FUTUROS

Apesar da aplicação já ser funcional e sua utilização para outras aplicações já estar completa, ainda se faz necessário algumas adições futuras ao trabalho, como a adição de um sistema de autenticação para evitar que usuários não autorizados alterem informações importantes, a inclusão de uma associação entre um usuário e um perfil de jogador e a hospedagem desta aplicação bem como de seu banco de dados em nuvem para que possa ser acessada mais facilmente.

Além dos pontos supracitados, será implementado um aplicativo para android que utilize desta aplicação com fim de fornecer uma interface para o usuário final e que o mesmo possa utilizar de todas as funcionalidades de forma que não seja preciso entender os pormenores da aplicação.

REFERÊNCIAS

CODEMAG. **History of the VS IDE**. 2008. Disponível em: https://www.codemag.com/Article/0710022/History-of-the-VS-IDE. Acesso em: 28 abr. 2022.

JASMINDER, Singh. **Unit of Work in Repository Pattern**. 2018. Disponível em: https://www.c-sharpcorner.com/UploadFile/b1df45/unit-of-work-in-repository-pattern. Acesso em: 28 abr. 2022.

JOBSTRAIBIZER, Flávia. **Guia profissional Microsoft SQL Server 2008**. São Paulo: Digerati Books, 2009.

JONES, Matthew. **The Repository-Service Pattern with DI and ASP.NET 5.0.** 2019. Disponível em: https://exceptionnotfound.net/the-repository-service-pattern-with-dependency-injection-and-asp-net-core/. Acesso em: 28 abr. 2022.

MICROSOFT. **Entity Framework Core**. 2022. Disponível em: https://docs.microsoft.com/en-us/ef/core/. Acesso em: 02 maio de 2022.

MICROSOFT. **The history of C#**. 2022. Disponível em: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history/. Acesso em: 28 abr. 2022.

ORACLE. **WHAT is a relational database(RDBMS)**. 2022. Disponível em: https://www.oracle.com/database/what-is-a-relational-database/. Acesso em: 26 abr. 2022.

REDHAT. **O que é uma API?** 2017. Disponível em: https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces. Acesso em: 25 abr. 2022.

O AVANÇO DA TECNOLOGIA NA FORMA DE PAGAMENTOS E VENDAS NO SETOR DE PACOTES TURISTICOS PÓS PANDEMIA

SILVA, Jéssica Henrique
ROCHA, Gláucio Bezerra

RESUMO

O presente trabalho é um relato de caso sobre o desenvolvimento de um aplicativo mobile para uma agência de viagens virtual. O objetivo foi desenvolver um aplicativo para melhorar as vendas dos pacotes turísticos da empresa, por meio do aplicativo e da modalidade de pagamento via *Pix*. A metodologia abordada foi a pesquisa bibliográfica qualitativa a partir de documentos disponibilizados nas plataformas digitais. Os resultados obtidos foi desenvolvimento satisfatório do aplicativo da Raja Turismo. Porém, o mesmo encontra-se em fase de teste e será lançado ao mercado em 7 de junho de 2022. Aspira-se que as vendas da empresa sejam incrementadas a partir da proposta de inovação do marketing digital, o qual será desenvolvido através do aplicativo.

Palavras-Chaves: Aplicativo Mobile; Tecnologias Digitais; Turismo.

1 INTRODUÇÃO

Para quem viveu esse momento histórico e preocupante que foi a pandemia mundial da Covid-19, sabe o quão difícil foi para todos adaptarse e superar os desafios decorrentes da grande quebra no mercado financeiro, a qual gerou o desemprego, o fechamento de empresas e a necessidade de criar soluções para superar a crise financeira mundial em meio a todo esse caos.

O ano de 2020 iniciou com a crise da pandemia da Covid-19, a qual causou bastante euforia entre toda a população afetando diretamente o setor econômico dos países. Em consequência a essa crise na saúde pública, muitas lojas fecharam as portas devido a falta de consumo das pessoas, obrigando as empresas a demitirem seus funcionários. Consequentemente, sem emprego as pessoas deixaram de consumir bens e serviços formando um ciclo onde as pessoas não gastam devido ao desemprego. Dessa maneira, para as empresas de pequeno porte não fazia sentido permanecer de portas abertas, ocasionando a falência de pequenos negócios.

Em relação as empresas de viagens turísticas, as vendas caíram acentuadamente, e no caso específico da empresa Raja Turismo, a qual antes da

Covid-19 vendia muitos pacotes de viagens, durante a pandemia, viu a necessidade de melhorar sua forma de vender os seus pacotes turísticos.

Da necessidade de alavancar as vendas, a Raja Turismo a partir de indagações do que poderia ser feito para melhorar as vendas dos pacotes de viagens durante a pandemia, já que, essa crise na saúde pública mundial trouxe como medidas preventivas o confinamento, o qual afetou diretamente o setor do turismo de todos os países, causando uma elevada redução e restrição das vendas nesse âmbito.

Durante a pandemia o uso das tecnologias digitais ampliou-se bastante, incluso até teve início da modalidade de trabalho remoto. Entre as inovações tecnológicas surgidas após o desencadeamento da pandemia, destaca-se a nível do Brasil o *pix*, a nova forma de realizar pagamentos e transferências.

O *pix* é uma estratégia inovadora onde é possível realizar pagamentos sem ser preciso se deslocar e de modo quase instantâneo. Essa nova modalidade digital de pagamento foi considerada uma solução positiva para a queda das vendas de empresas como a Raja Turismo. Porém, considerou-se que além de oferecer aos clientes a possibilidade de pagamento via *pix*, idelizou-se o desenvolvimento de um aplicativo para facilitar as vendas dos pacotes de viagens sem a necessidade de uma loja física.

A agência de viagens Raja Turismo foi criada em 2019, a ideia surgiu após uma viagem de um casal para os Lençóis Maranhenses, Barreirinhas – MA. A viagem não foi como esperado por falta de organização e atendimento ao cliente. Pensaram então, na possibilidade de prestar um excelente atendimento aos clientes criando a própria agência de viagens.

A ideia funcionou até o ano seguinte quando começou a pandemia mundial da Covid-19 em 2020, porém com tudo parado devido a determinação de confinamento pelos órgãos competentes, era considerado impossível vender pacotes de turismo em meio a esse cenário pandêmico. Destacando que, as vendas só puderam ser retomadas após a primeira dose da vacina, mas mesmo assim, continuaram em ritmo lento.

A ideia do desenvolvimento de um aplicativo foi a solução definida pela Raja Turismo para resolver o problema da redução das vendas de pacotes de viagens durante a pandemia. O aplicativo está sendo desenvolvido para melhor atender os clientes.

O aplicativo apresentará quatro telas, destacando que, na tela de pagamentos além das opções de crédito, débito e boleto, será implementada também o *pix*, o qual é caracterizado como uma das formas de pagamento em destaque no país desde seu surgimento.

A Raja Turismo é uma empresa online inscrita com o CNPJ 42.366.851/0001-00. A platarforma online utilizada para divulgação da empresa é o Instragram por meio do endereço eletrônico @rajaturismo. O contato com a empresa pode ser realizado através do Instragram, o email (rajaturismo@hotmail.com) e pelo contanto telefônico e WhatsApp Business (+55 98 98819 0821).

Dessa maneira, o presente trabalho refere-se a um relato de caso de abordagem qualitativa, fundamentada no referencial teoríco cujo os objetivos é relatar o desenvolvimento de um aplicativo de vendas online de pacotes turísticos, visando aumentar as vendas da empresa Raja Turismo.

2 FUNDAMENTAÇÃO TEÓRICA

O levantamento bibliográfico foi realizado com o objetivo de elencar os principais pontos importantes sobre a História do Turismo e a sua contribuição para a economia e o desenvolvimento social e cultural de países e regiões.

Desse modo, os capítulos se dividem em uma breve contextualização da História do Turismo; o contexto econômico do Turismo; as vantagens e desvantagens em utilizar aplicativos mobiles para o planejamento de viagens turísticas; a evolução do marketing digital e a nova forma de pagamento via *pix*.

2.1 BREVE CONTEXTUALIZAÇÃO HISTÓRICA DO TURISMO

Indica-se na literatura que o processo de evolução do Turismo surgiu com o desenvolvimento mundial, inicialmente com viagens de famílias pertencentes a elite das sociedades, porém devido a sua evolução, esse setor foi se modificando permitindo que todas as classes sociais pudessem realizar viagens turísticas (MENDES, 2021).

Aponta-se que a revolução do Turismo aconteceu no século XX, entretanto, verifica-se que sua história se remota aos tempos dos impérios. Os primeiros primordios refere-se ao Império Egípcio e Babilônico. Destacando que, as viagens

nessa época era determinadas pelo comércio e posteriormente houve viagens a centros turísticos ou de fins curativos (CUNHA, 2013).

Como menciona Mendes (2021), em meados do século XVII surgiu o hábito entre os jovens aristocratas ingleses pela Europa Ocidental, que permitiu o contato desses jovens com outras culturas, lugares e pessoas. Esse costume ficou conhecimento como o *Grand Tour*, os quais poderiam durar até três anos, o que deu origem a palavra *Tourism*, populamente conhecido como turísta.

No entanto, a expansão do Turismo teve seu início no século XIX a partir do progresso tecnológico e social em consequência da Revolução Industrial. Cabe destacar que, em 1841 foi organizada a primeira viagem da história por Thomas Cook, o pai do Turismo, ele criou nessa mesma época a primeira Agência de Turismo com o nome de Thomas Cook and Son. E já em 1872 ele organizou a primeira viagem ao mundo (RAMOS; COSTA, 2017).

Além disso, Mendes (2021) destaca que o *Voucher* foi criado em 1867 e no século XXI surgiu os *Travel Check* da American Express. Porém, apenas no século XX é que aconteceu a grande revolução no setor do Turismo. Nos anos de 1950 e 1973 identificou-se um crescimento nesse setor, porém no ano de 1970 o setor apresentou uma redução. A partir do século XXI, o Turismo só tem se expandido, apesar das crises de 2001 em decorrêencia do atentando terrorista a cidade de Nova Iorque nos Estados Unidos, e em 2009 em decorrência da crise econômica mundial.

No contexto atual, o Turismo continua sendo considerado um setor estratégico da economia dos países, caracterizado com um dos setores mais impulsionadores na formação de ciddãos e na proteção dos patrimônios ambiental e cultural (MENDES, 2021).

Destacando que, o Turismo é definido como um processo que abarca o contexto social, cultural e econômico de um país, que se articula devido a movimentação de inúmeras pessoas para outros lugares diferentes de sua residência, algumas vezes por motivos de trabalho, outros de lazer, formação, entre outros. As pessoas que viajam para outros lugares fora da sua comunidade são denominadas de turístas, exurcionistas, residentes ou não residentes (MENDES, 2021).

Esse autor aponta ainda que as agências de viagens e turismo são caracterizadas como um dos principais agentes econômicos da atividade turística,

devido a sua especialidade em organizar e planejar os melhores itinerário turístico, independemente de qual será o destino.

Pode-se afirmar que, as agências de viagens e turismo como pessoas singulares ou coletivas apresentam as seguintes funções: operam no desenvolvimento da organização e vendas de pacotes turísticos; representam outras agências; realizam reservas de serviços em locais e empreendimentos turísticos; efetuam vendas de passagens e reservas em diferentes opções de transporte, recepção, transferência e assistência ao turista. Além dessas funções, as agências de viagens e turismo também assumem outras responsabilidades como por exemplo, articulação na tramitação de passaportes, vistos e demais documentos para a viagem (MENDES, 2021).

Porém, após o erupção da pandemia da Covid-19, o setor do Turismo, novamente foi afetado, principalmente em decorrência das determinações dos orgãos mundiais competentes da saúde pública, como a Organização Mundial de Saúde (OMS) e o Ministério de Saúde (MS), do confinamento de todos.

2.2 O CONTEXTO ECONÔMICO DO TURISMO

Com esse cenário pandêmico muitas empresas do ramo do Turismo de pequeno porte fecharam seus negócios, em consequência a isso o desemprego aumento, não apenas nesse setor, mas também em muitos outros.

Mendes (2021) em seu estudo sobre “O Marketing Digital no Turismo em Pandemia: O Caso das Agências de Viagens”, indica que o setor cresceu nas últimas décadas a nível mundial, sendo este o seu interesse em realizar sua pesquisa. Destacando que sua pesquisa foi apresentada ao Instituto Politécnico de Coimbra Instituto Superior de Contabilidade e Administração de Coimbra, Portugal.

De aordo com os estudos desse autor, os setores turísticos benefícia não apenas a economia dos países e regiões territóriais, mas também, favorece o contexto social através da geração de empregos; o ambiental por meio da promoção da conservação dos patrimônios históricos e culturais da humanidade; diminui a pobreza; promove o entendimento entre diferentes povos e contribui para o desenvolvimento do país (MENDES, 2021).

Conforme apontado pela OMT, o turismo é caracterizado como um fenômeno econômico, social e geográfico. Considera-se o Turismo como um setor

estratégico para o desenvolvimento social, político e econômico de países, regiões, cidades e continentes. É um setor que gera rendimentos e receitas consideráveis sendo definido como o setor que mais cresceu e exportou serviços (MENDES, 2021).

Tomé (2020), aponta que o turismo é considerado um setor de suma relevância para a o contexto econômico e social do país, o qual gera empregos tanto a nível superior como também a jovens com nível de escolaridade baixo. Indica-se que 3% do total da geração de emprego no Brasil, são provinientes do setor do turismo.

As estimativas indicam que em 2019, os incentivos na economia brasileira em consequência do turismo no país, foi de quinhetos e cinquenta e um bilhões de reais, equivalente a cento e trinta e nove bilhões de dólares só nesse ano. Esse incremento equivaleu a 7,7% do Produto Interno Bruto (PIB) do Brasil nesse mesmo ano, conforme os dados apresentados pelo Conselho Mundial de Viagens de Turismo (WTTC) (TOMÉ, 2020).

Como aponta Tomé (2020), o turismo no Brasil em 2019 gerou 7,4 milhões de empregos diretos, indiretos e induzidos, caracterizando-se como um setor que promeve uma geração consideravél de recetais cambiais. Destacando que, esse setor gerou nesse ano 5,9 bilhões de dólares através das receitas internacionais de turismo.

Tomé (2020), em sua pesquisa indicou que conforme os dados do Instituto Brasileiro de Geografia e Estatística (IBGE), o setor do Turismo é formado por diferentes cadeias produtivas na geração de receitas, rendimentos e empregos, sendo estas as seguintes: hotéis e pousadas; bares e restaurantes; transporte rodoviário; transporte aéreo; outros tipos de transportes auxiliares; atividades de agências e organização de viagens; aluguel de bens móveis; atividades recreativas, culturais e desportivas.

Essas cadeias produtivas são compostas por diferentes profissionais, como empresários, autonômos, redes hoteleiras, companhias áreas, entre outros de pequeno, médio e grande porte, que após a pandemia sofreram um impacto muito grave (TOMÉ, 2020).

Apesar das dificuldades enfretadas pelo Turismo e todas as suas cadeias produtivas durante a história da sua evolução e mais recentemente com a pandemia da Covid-19, esse setor tem buscando diversificar sua forma de prestar

serviços as pessoas. A partir dessa crise mais do que nunca o uso das tecnologias digitais, foi uma alternativa para avalancar o setor do turismo em muitos países.

As vantagens do uso do aplicativo mobile para incrementar as vendas das agências de viagens e turismo é uma alternativa viável que pode agregar pontos positivos para a prestação de serviço nesse ramo, principalmente após a pandemia onde o mundo inteiro teve que buscar novas alternativas para enfrentar a crise econômica que afetou principalmente ao setor do turismo.

2.3 VANTAGENS DO USO DE APLICATIVOS MOBILES PARA ORGANIZAR PACOTES DE VIAGENS

Identifica-se que o avanço tecnológico, em especial a internet, proporcionou mudanças relevantes na vida das pessoas, das corporações e instituições de modo a globalizar o mundo. O setor do turismo também foi beneficiado e muito com essa inovadora tecnologia, a partir daí o ramo da Tecnologia da Informação e Comunicação (TIC) foi se inovando cada vez mais, até chegar ao contexto atual, onde o uso dos aplicativos mobiles tornaram-se inerente a cultura do século XX e principalmente do século XIX (PATRIOTA, 2021).

Essa revolução da tecnologia afetou diretamente o setor turístico o qual ganhou uma nova ferramenta estratégica para estimular a vendas de pacotes turísticos, principalmente durante e pós pandemia.

Desse modo, pode-se dizer que os aplicativos mobiles são um sistema de *softwares* inseridos em sistemas operacionais os quais efetivam-se em dipositivos *hardwares* mobiles como smartphones e tablets (BIZ; AZZOLIM; NEVES, 2016).

Verifica-se que, o uso de smartphones possibilita aos turistas o acesso as informações sobre viagens, pacotes turísticos, passagem, hospedagem, custos, entre outros de forma rápida, a qualquer hora e lugar, sempre e quando tenha-se a conexão com a internet (PATRIOTA, 2021).

Patriota (2021), destaca ainda em seu estudo que, o uso dos aplicativos mobiles além de facilitar na escolha, planejamento e pagamento aos turistas, também favorece na hora de decidir por um serviço ou outro e por uma agência ou outra, já que, tem-se a vantagem de avaliar os *feedbacks* positivos ou não de outras pessoas que já contrataram o serviço com determinada empresa.

2.4 A EVOLUÇÃO TECNOLÓGICA E O MARKETING DE TURISMO

Pode-se dizer que a definição de marketing foi sendo renovada conforme o processo da evolução tecnológica. A partir do surgimento da internet, houve um processo de globalização que afetou o mundo permitindo a comunicação, antes tão díficil, entre as diferentes civilizações. Para as empresas e fornecedores, essa possibilidade de ampliar e encurtar a acessibilidade entre consumidores e os bens e serviços foi muito benéfica (MENDES, 2021). Que além de favorecer as empresas, também foi muito positiva para o setor econômico, social, cultural e político dos países.

Como aponta Mendes (2021), houve uma transição entre o marketing tradicional e digital. Essa transição aconteceu principalmente em decorrência da evolução tecnológica. Esse autor considera que o Turismo foi um dos setores mais beneficiados em decorrência da evolução tecnológica digital. Em sua pesquisa, ele apontou como principais ferramentas utilizadas no marketing digital as seguintes: Website, o Search Engine Optimization (SEO), o Customer Relationship Management (CRM), as Redes Sociais, as Aplicações Mobille (APPs) e o E-mail marketing.

Porém, nesse estudo destaca-se o aplicativo *Instagram*, o qual já faz parte da cultura da sociedade moderna e que no âmbito do setor do turismo, permite aos clientes obterem imagens e vídeos das viagens que desejam realizar, podendo assim escolher com mais facilidade qual destino visitar. Assim como o *Instagram* outras redes sociais também facilitam a divulgação de pacotes de passeios turísticos.

Desse modo, acredita-se que a possibilidade do setor de turismo utilizar ferrementas digitais, tais como, aplicativo mobile para incrementar seu negócio é uma alternativa tecnológica que já está inserida no contexto atual das sociedades modernas e que como vantagem trazem a facilidade de comunicação entre empresas, fornecedores e consumidores.

2.5 FORMA DE PAGAMENTO *PIX*

O *Pix* é uma forma de pagamento instantâneo que foi criado pelo Banco Central (BC), no qual permite as transferências de recursos a qualquer dia e hora, mesmo em feriados onde os bancos não funcionam (BRASIL, 2022).

Entre as principais vantagens do *Pix* apresentadas pelo BC cita-se as

seguintes: alavancar a competitividade e a eficiência do mercado; baixar o custo, aumentar a segurança e aprimorar a experiência dos clientes; incentivar a eletronização do mercado de pagamentos de varejo; promover a inclusão financeira; e preencher uma série de lacunas existentes na cesta de instrumentos de pagamentos disponíveis atualmente à população (BRASIL, 2022).

Pode-se afirmar que, o *Pix*, lançado em novembro de 2020, é uma ferramenta inovadora destinada a realizar diversas operações bancárias de modo imediato e com baixos custos que tem sido a opção mais utilizada pelos brasileiros.

3 METODOLOGIA

Esse trabalho surgiu da necessidade de melhorar as vendas da Raja Turismo após o início da pandemia. Cabe ressaltar que, mesmo antes dessa crise na saúde pública mundial, aproximadamente depois das férias de julho de 2019, as vendas tiveram uma pequena redução, porém, com a pandemia ficou ainda pior.

Com poucas perspectiva-se de melhorar as vendas após o confinamento, surgiu a indagação de como melhorar as vendas através da opção de pagamento via *Pix*. Para tanto, além de proporcionar facilidades de pagamentos, questionou-se sobre o marketing digital, já que, a empresa é online. Será que o marketing proporciona visibilidade da empresa nas plataformas digitais?

A partir dos questionamentos do que poderia ser melhorado para avalancar as vendas de pacotes de viagens após a pandemia e sobre se o marketing estava adequado para alcançar um público maior, surgiu a hipótesis de que o desenvolvimento de um aplicativo próprio da agência poderia não apenas impulsionar as vendas, mas também seria uma oportunidade de melhorar o marketing digital da empresa.

Dessa maneira, foi realizada um levantamento de dados sobre a criação de aplicativos mobiles e o uso das tecnologias digitais no seguimento do Turismo.

Essa pesquisa é um relato de caso com caráter qualitativo e com abordagem bibliográfica. O levantamento de dados foi realizado na Scientific Electronic Library Online (SCIELO), no Portal de Periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), no Google Acadêmico, na Biblioteca Digital de Teses e Dissertações (BDTD) e no site Science.gov. Os descritores utilizados para a pesquisa bibliográfica foram: Aplicativo Mobile,

Tecnologias Digitais e Turismo.

A coleta de dados conseguiu reunir um total de 27 documentos, porém somente 8 foram selecionados para o referencial teórico do trabalho. Como critério de inclusão, optou pelos documentos com data de postagem inferior a dez anos que apresentou em seus resumos relação direta como o tema proposto. Os critérios de exclusão foram documentos com data de publicação superior a dez anos.

A outra parte do trabalho, foi o planejamento e devolvimento do aplicativo mobile da Raja Turismo. Outro levantamento de dados foi levantado como referência para o desenvolvimento do aplicativo. Nessa nova pesquisa teve com base nas palavras-chaves: Linguagem de Programação e Plataformas de Aplicativos Mobiles. Com base nessa pesquisa optou-se pela execução de um aplicativo com linguagem de programação simples voltado para smartphones e tablets, com ferramentas práticas e funcionais desenvolvidas para facilitar a experiência dos consumidores durante a compra de pacotes de viagens com a Raja Turismo, visando incrementar as vendas dessa empresa.

Dessa forma, decidiu-se pela Linguagem de Programação Java, a plataforma desenvolvedora de aplicativos mobiles Android Studio e a Firebase. O ambiente de desenvolvimento do aplicativo foi em *Home Office*, através do notebook de uso pessoal com as seguintes características de Hardware e Software: Processador: Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz; Memória RAM de 6,00 GB; Disco rígido de 1000GB; Sistema Operacional: Windows 10 Pro e Tipo de Sistema Operacional: 64-bit. Os softwares utilizados no desenvolvimento foram o Java (TM) SE Develop kit 11.0.12 (64-bit) e Android Studio bumblebee 2021.1.1.

O trabalho foi elaborado apenas por uma pessoa e o seu desenvolvimento detalhado vai ser explicado no próximo capítulo. Como resultados obteve-se a criação do aplicativo Raja Turismo, que está sendo testado para poder ser lançado nas plataformas digitais a partir de junho de 2022, aspirando incrementar os rendimentos da agência e melhorar o marketing digital da empresa através do uso desse aplicativo.

4 FUNDAMENTAÇÃO TEÓRICA

De acordo com as pesquisas para viajar e organizar seus passeios os

brasileiros estão entre os que mais usam smartphone. Assim como em várias outras áreas, o uso da tecnologia no turismo aumentou nos últimos anos, principalmente durante a pandemia provocando uma revolução para viajantes, agências e para todo mercado.

Visitas presenciais a agências de turismo eram a melhor opção para fazer o planejamento e fechar as compras de uma viagem, mas esse tempo ficou para trás. Agora, o viajante pode resolver tudo sem precisar sair do seu conforto e do seu lar, enquanto empresas do setor mantém sua relevância ao marcar presença e prestar serviços pelo universo mobile.

Encontrar soluções ao alcance das mãos é algo que o avanço da tecnologia possibilitou à sociedade. Atualmente, considera-se que as empresas que não tem um site responsivo, que roda bem no desktop e no smartphone, está desatualizada, já que, por meio dos aplicativos e das plataformas digitais é possivel conquistar mais oportunidades de mercado, principalmente, quando a empresa desenvolve o seu prórpio aplicativo.

Basta acessar a loja online de aplicativos para verificar que diversos segmentos e nichos atualmente atuam no mercado financeiro por meio do desenvolvimento de aplicativos ou apps que facilitam a comunicação com o consumidor.

O ritmo acelerado do dia a dia faz com que pareça mais apropriado resolver as questões de uma viagem por conta própria, sem precisar se deslocar para falar com um consultor de viagens em uma agência especializada. Ao invés disso, viajantes recorrem a sites e apps que fazem comparativos de preços de passagens, hospedagens e outros *trades* do turismo.

Desse modo, a empresa Raja Turismo desenvolveu o seu próprio aplicativo a partir das tecnologias da informação disponível no mercado.

4.1 TECNOLOGIAS USADAS

Para o desenvolvimento do aplicativo da Raja Turismo, considerou-se relevante utilizar os mais conhecidos e acessíveis programas de linguagem de programação, como por exemplo o Java, e a plataforma desenvolvedora de aplicativos Android Studio.

4.1.1 Java

Nos anos 90 James Gosling comandava uma equipe na empresa Sun Microsystems onde foi desenvolvido a linguagem de programação Java. Em 2008 a empresa Oracle Corporation adquiriu o Java. A linguagem se tornou uma das mais usadas no mundo (SILVA, 2015).

De acordo com Silva (2015), a linguagem de programação Java permite que sejam inscritos comandos para serem executados pelos computadores. Os comandos e instruções desenvolvidas no Java são denominadas de Software. Pode-se dizer que, o Software é a ferramenta que permite o controle do Hardware. Dessa forma, a partir da linguagem de programação Java é possível criar e desenvolver programas de computador, sistemas automotivos, sistema operacionais, automação de residências, jogos, e muito mais.

Destaca-se que o Java foi desenvolvido com os seguintes objetivos: orientação a objetos; independência de plataforma; recursos de rede; segurança e facilidades para criação de programas (SILVA, 2015).

4.1.2 Android Studio

Android Studio é uma plataforma voltada para o desenvolvimento de aplicativos mobiles. Essa plataforma possui um dos sistemas operacionais mais utilizados do mundo, caracterizando-se como a principal ferramenta para a criação de aplicativos para Android. Assim como outras tecnologias no mundo, o Android Studio está sempre passando por atualizações, é necessário ficar atento para nunca perder atualizações sobre novos recursos ou funcionalidades. Hoje está presente em 74,13% dos dispositivos móveis (SILVA, 2015).

O Android Studio é um Ambiente de Desenvolvimento Integrado, uma tradução para Integrated Development Environment (IDE). A plataforma estava em estágio de acesso antecipado lançando a versão 0.1 em maio de 2013, passou por um estágio beta a partir da versão 0.8 que foi lançada em junho de 2014, porém sua primeira compilação estável só foi lançada em dezembro de 2014, iniciando com a versão 1.0. O Android Studio conta com um Emulador para simular o Android e testar o aplicativo em vários aplicativos, sem precisar ter o dispositivo físico, poupando tempo e esforço no teste (SILVA, 2015).

A seguir temos o exemplo de duas telas de smartphone utilizando o sistema operacional Android Marshmallow (5.0).

Imagem 1 **Imagem 2**

Fonte: elaborado pela autora, 2022.

A imagem 1 representa a tela inicial do sistema e a imagem 2 a tela do menu de aplicativos do smartphone. Essas duas telas são o simulador, usado na elaboração do aplicativo de modo que o desenvolvedor tenha acesso à qualidade e funcionamento do sistema do sistema.

A plataforma apresenta os seguintes requisitos de sistema, conforme a tabela 1:

Tabela 1: Requisitos do Sistema para funcionar a plataforma Android Studio

	Windows	OS X/macOS	Linux
Versão do SO	Microsoft Windows 10/8.1/8/7/Vista/2003/XP (32 ou 64 bit)	MacOS X 10.10/MacOS X 10.11(Yosemite/El Capitan) ou superior, até a versão 10.13/10.14 (MacOS High Sierra/MacOS Mojave)	GNOME ou KDE ou Unity no Ubuntu ou Fedora ou GNU/Linux Debian
RAM	4 GB de RAM no mínimo, 8 GB de RAM recomendado		

Espaço em disco	2 GB de espaço em disco
Espaço para Android SDK	No mínimo 4,5 GB para Android SDK, imagens do sistema de emulador e caches
Versão JDK	Java Development Kit (JDK) 6/7/8/9/10/11/12 ou superior
Resolução de tela	1280x800 de resolução de tela no mínimo

Fonte: elaborado pela autora, 2022.

O Android Studio não é o único IDE para criação de aplicativos mobile com sistema operacional do Google, existem outras opções como o Eclipse, por exemplo, e também multiplataformas.

Verifica-se que, entre as principais vantagens de utilizar a plataforma da Android Studio destaca-se a maior variedade de customização; melhor personalização de tudo na ferramenta, podendo usar atalhos do teclado; variedade de recursos e menores problemas de compatibilidade e *bugs* no sistema, já que, indica-se que são menos frequentes no Android Studio do que em outras IDE (SILVA, 2015).

Por outro lado, ressalta-se como desvantagens dessa plataforma, o tempo para executar e carregar um projeto é maior comparando com outras IDE, e que essa plataforma não permite administrar outros projetos na mesma janela (SILVA, 2015).

4.1.3 Firebase

Como aponta Silva (2015), o Firebase é uma plataforma de desenvolvimento e computação em nuvem do google para criar aplicativos móveis e webs. Essa plataforma apresenta uma coleção de ferramentas nas quais os desenvolvedores podem confiar e criar seus aplicativos, os quais podem ser expandidos com base na demanda. No entanto, a plataforma visa resolver três problemas principais para os desenvolvedores, sendo estes: criar um aplicativo rapidamente; liberar e monitorar um aplicativo com confiança.

Destaca-se que, os desenvolvedores que confiam nessa plataforma têm acesso a serviços que precisam desenvolver, permitindo que se concentrem em fornecer uma experiência de aplicativo robusta.

Indica-se que entre os principais recursos em destaque da plataforma

Google Firebase incluem bancos de dados, autenticação, mensagens push, análises, armazenamento de arquivos e muito mais.

Como o serviço é hospedado na nuvem, os desenvolvedores podem executar o dimensionamento sob demanda de forma transparente. Atualmente, o Firebase é uma das principais plataformas de desenvolvimento de aplicativos confiáveis por desenvolvedores de todo o mundo.

De acordo com Silva (2015), a história do Firebase surgiu da Envolve, uma startup anterior fundada em 2011 por Andrew Lee e James Tamplin. A empresa fornece uma API para desenvolvedores para facilitar a integração de bate-papo ao vivo em sites. Os fundadores da Envolve descobriram que seu serviço de bate-papo estava sendo usado para encaminhar mensagens que não eram do bate-papo. Essa funcionabilidade permitia aos desenvolvedores sincronizar dados de aplicativos em tempo real. Desse modo, os seus fundadores resolveram separar a arquitetura em tempo real do sistema de chat o que deu origem a Firebase em 2011, mas somente foi lançada ao público em abril de 2012.

A Firebase lançou seu primeiro produto o Firebase Realtime Database, o qual é uma API que sincroniza dados de aplicativos em dispositivos Android, Web e IOS. Desenvolvedores de aplicativos podem contar com essa plataforma para construir aplicativos colaborativos em tempo real. A partir da contruição de Founder Collective, Greylock Partners, New Enterprise Associates e Flybridge Capital Partners o Firebase teve um acúmulo financeiro inicial de mais de US $ 1 milhão no ano de 2012. Na série A, essa empresa levantou também um financiamento de $5,6 milhões da Flybridge Capital Partners e da Union Square Ventures, em junho de 2013 (SILVA, 2015).

Em 2014 a Firebase lançou a Firebase Authentication e Firebase Hosting, determinando a empresa como um Mobile backend as a servisse (MbaaS). Ainda em 2014 o Firebase se tornou parte do google, a tecnologia então adquiriu a Divshot, uma plataforma de hospedagem na web que se fundiu com o Firebase (SILVA, 2015).

Cabe destacar que, os aplicativos mais famosos que utilizam a plataforma Firebase são: Alibaba, The New York Times, Todoist, Trivago e eBay Motors.

No que se refere aos tipos de aplicativos que podem ser desenvolvidos com o Firebase destaca-se: Android, iOS e Web. Considera-se uma plataforma fácil e ágil que permite maior acessibilidade ao público.

Entre as principais vantagens do Firebase destaca-se: gratuito; API ponta a ponta; armazenamento de arquivos pelo Google Cloud Storage; autenticação via e-mail, google, Facebook e Github; dados em tempo real; hospedagem de arquivos estáticos e segurança. Em contrapartida as desvantagens apontam-se: limitações de infraestrutura; bugs de atualizações; problemas que envolvem UI/UX (SILVA, 2015).

5 RESULTADOS

Os resultados sobre o desenvolvimento do aplicativo da Raja Turismo serão apresentados com figuras que permitem demonstrar sua funcionabilidade. Dessa forma, as imagens seguintes apresentam os resultados obtidos sobre desenvolvimento e funcionamento do aplicativo.

5.1 LEVANTAMENTO DE REQUISITOS

Na tabela 2 são apresentados os requisitos funcionais do aplicativo desenvolvido.

Tabela 2: Requisitos Funcionais

Identificação	Nome	Descrição
RF01	Efetuar login	O sistema permitirá acesso do usuário através de login e senha.
RF02	Cadastrar usuário	O sistema permitirá que usuários sejam cadastrados preenchendo os campos para que a conta seja criada e efetuar o login.
RF03	Selecionar pacotes para compra	O sistema permitirá que o usuário selecione um pacote de passeios
RF04	Escolher quantidade passageiros	O sistema permitirá que o usuário selecione a quantidade de passageiros
RF05	Preencher dados do passageiro	O sistema permitirá que o usuário preencha os dados dos passageiros
RF06	Realizar pagamento	O sistema permitirá que o usuário efetue o pagamento
RF07	Entrar em contato para dúvidas	O sistema permitirá que o usuário se comunique com um chat para tirar dúvidas

Fonte: elaborado pela autora, 2022.

A tabela 3 lista os requisitos não funcionais previstos para o sistema.

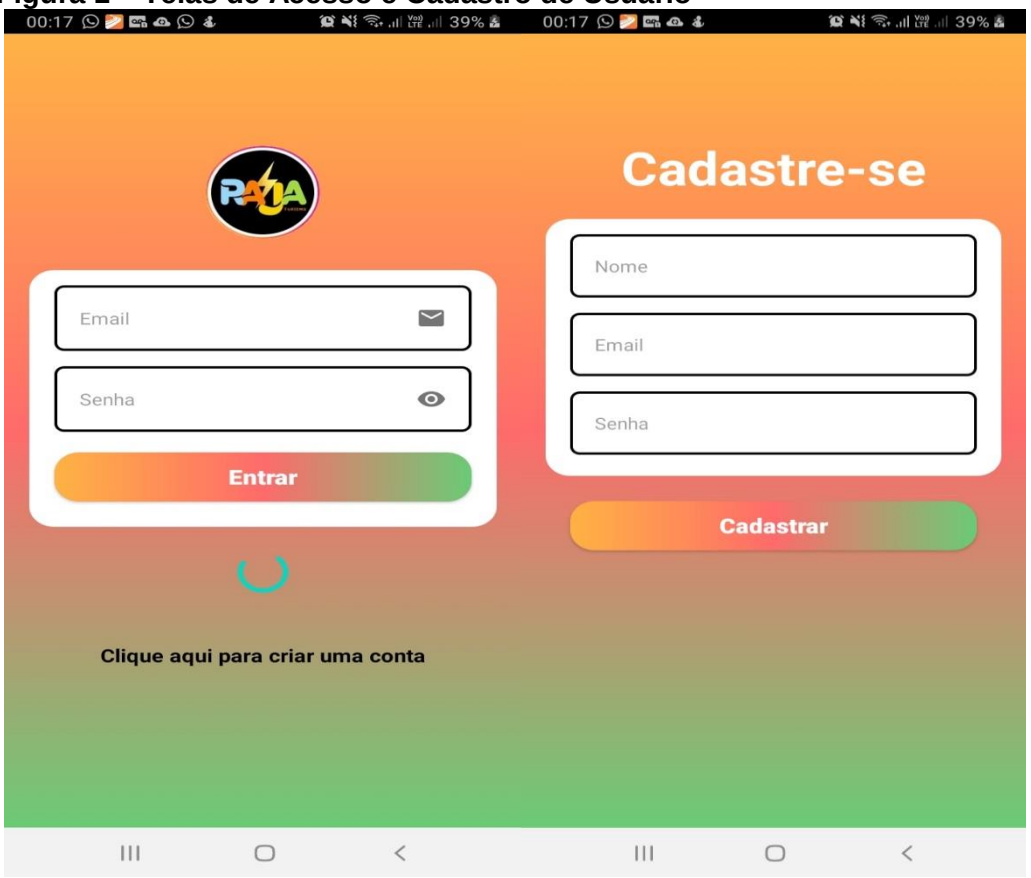
Tabela 3: Requisitos Não Funcionais

Identificação	Descrição
RNF01	O sistema deve ser desenvolvido no ambiente de Android Studio, com linguagem Java.
RNF02	O sistema deve usar Firebase Realtime Database como banco de dados online.
RNF03	A interface deve ser de fácil utilização

Fonte: elaborado pela autora, 2022.

Parte Externa do Aplicativo

Figura 1 – Telas de Acesso e Cadastro de Usuário



(a) Tela de Login **(b) Tela de Cadastro**

Fonte: elaborado pela autora, 2022.

Na figura 1 são ilustradas as telas da parte externa do aplicativo

A tela de login, representada na Figura 1(a), é composta pelos campos onde o usuário deverá informar seus dados para ter acesso ao sistema.

Na Figura 1(b) temos a tela de registro de usuário no sistema e ela é

composta pelos campos necessários para efetuar o cadastro de um usuário.

Ambas as telas, ambas as telas dão acesso à parte interna do sistema, onde estão presentes todas as outras funcionalidades.

Parte Interna do Aplicativo

Figura 2 – Tela Principal com Opções de Pacotes

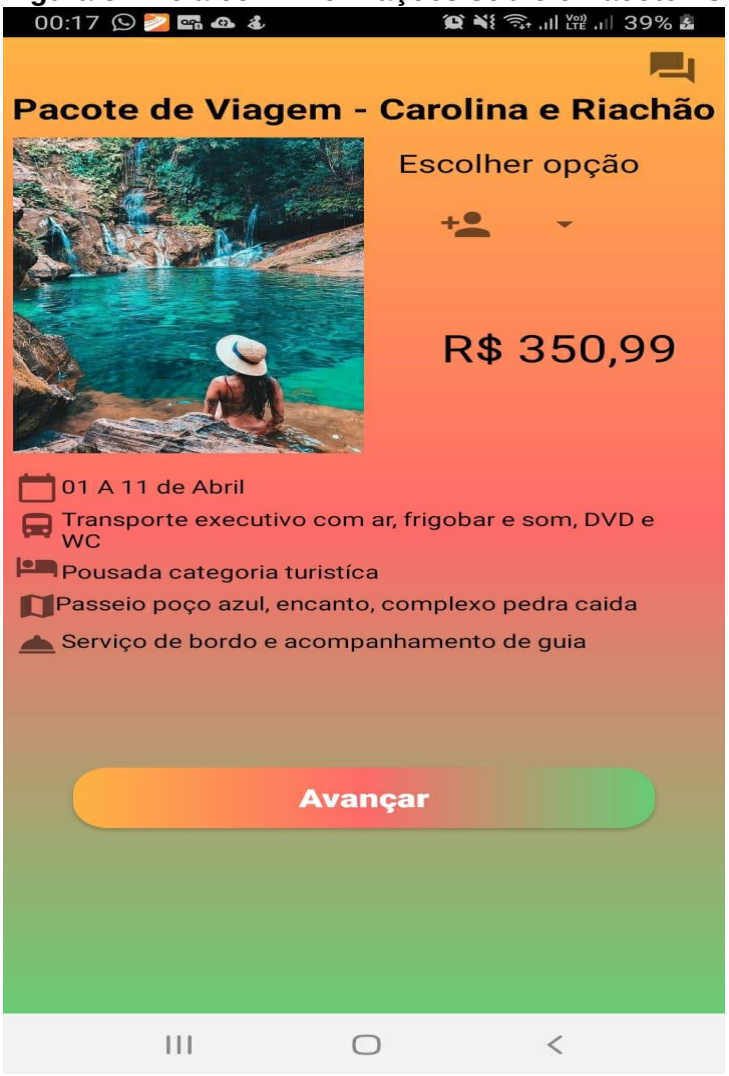
(a) Tela Principal

Fonte: elaborado pela autora, 2022.

Na Figura 2 é exibida a tela principal da parte interna do sistema.

Ao acessar a parte interna, a primeira tela que o usuário vê é representada na Figura 2(a). Sendo exibido pacotes de passeios turísticos no qual o usuário pode escolher.

Figura 3 – Tela com Informações sobre o Pacote Escolhido



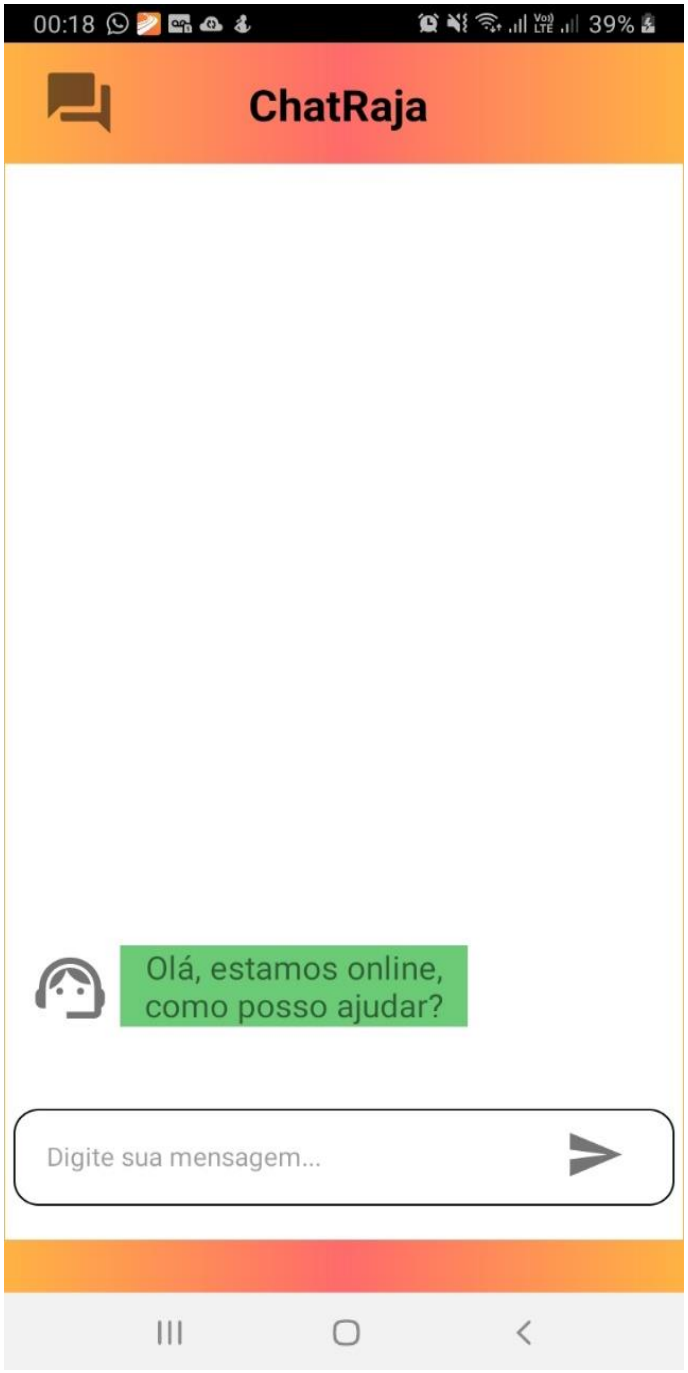
(a) Passeio1

Fonte: elaborado pela autora, 2022.

Na Figura 3(a) mostra a tela de passeio da parte interna do sistema.

Ao escolher um pacote, na tela principal figura 2(a), a tela com as informações sobre o pacote de passeio turístico é exibida como mostra a figura 3(a), contendo escolha para quantidade de passageiros, valor do passeio, informações sobre o passeio, um botão para avançar para próxima etapa, e no canto superior esquerdo consta um botão para dúvidas do usuário.

Figura 4 – Tela de Chat para Tirar Dúvidas



(a) Tela de chat

Fonte: elaborado pela autora, 2022.

Ao clicar na imagem de chat no canto superior esquerdo da figura 3(a), o usuário vai ter acesso a tela de chat pra tirar dúvidas.

Figura 5 – Tela de Pagamentos

(a) Pagamento Crédito/Débito **(b) Pagamento Pix**

Fonte: elaborado pela autora, 2022.

Na figura 5 temos 2 opções de pagamentos para finalizar a compra

Ao clicar em avançar na figura 4(a) o usuário vai estar abrindo a tela de pagamentos apresentada na figura 5 (a) e (b).

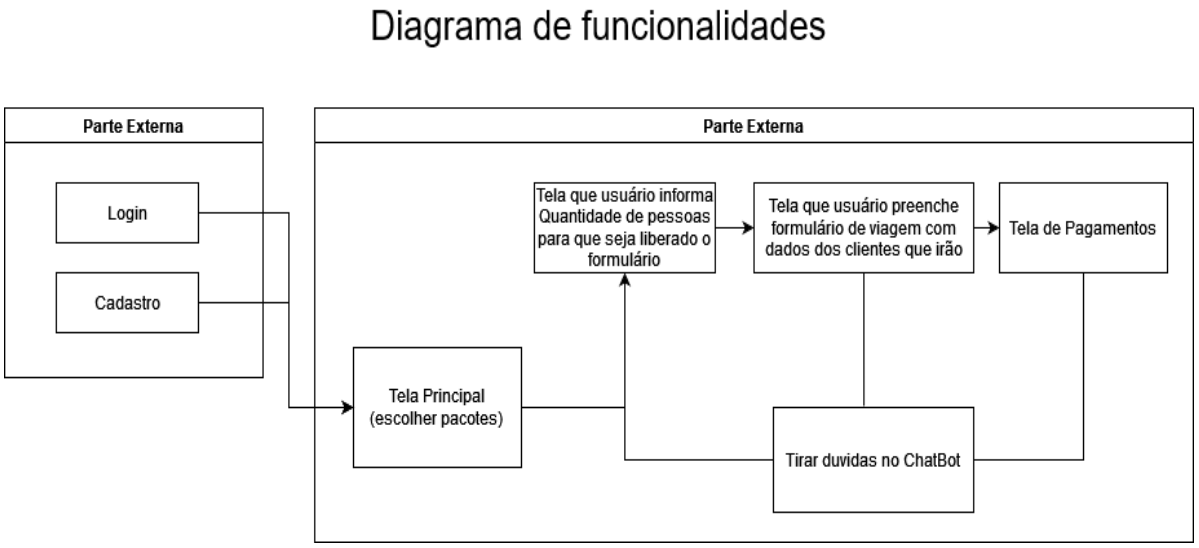
Na figura 5 (a) mostra a opção de pagamento pelo meio de cartão de crédito ou débito, ao selecionar a opção crédito, aparecerá na tela os campos de consumo para adicionar os dados do cartão.

Na figura 5(b) temos a opção meio de pagamento *Pix*, o usuário ao selecionar a forma de pagamento *Pix* aparecer o QR Code para escanear e efetuar o pagamento ou poderá copiar e colar no seu aplicativo do banco.

5.2 Diagrama de funcionalidades

Na figura abaixo representa o diagrama de funções desenvolvidos para criar o aplicativo.

Figura 6: Diagrama das Funções



Fonte: elaborado pela autora, 2022.

Na imagem acima temos o diagrama de funcionalidades do aplicativo, como podemos observar as funcionalidades estão organizadas de forma hierárquica e as setas representam o fluxo de acesso, por exemplo, através da funcionalidade tela de usuário podemos escolher quantas pessoas irão viajar e depois preencher o formulário com dados dos passageiros; a tela de pagamentos apresenta todas as opções diferentes para efetuar o pagamento do pacote; em caso de dúvidas em qualquer tela o usuário pode acionar o chat bot e tirar suas dúvidas.

O sistema é composto por duas partes: parte externa e interna. Na parte externa, temos as funcionalidades de criação e acesso à conta. O usuário consegue ter acesso às funcionalidades internas somente por essas duas alternativas. Na parte interna, todas as telas são acessadas pela tela principal.

A estrutura do sistema é representa pela Figura 7, porém ressalta-se que se trata de uma imagem meramente ilustrativa do sistema Android e Firebase.

Figura 7: Imagem ilustrativa do funcionamento do sistema Android e Firebase

Fonte: elaborado pela autora, 2022.

O sistema foi projetado para ter uma estrutura simples e de fácil manipulação, na figura temos o usuário ao lado com um dispositivo móvel que possui a plataforma Android, o aplicativo foi desenvolvido com a linguagem Java e tem como banco de dados o Firebase.

6 CONSIDERAÇÕES FINAIS

Verificou-se com a realização desse estudo, que o Turismo passou por muitos processos de transições durante a sua história principalmente em decorrência de crises econômicas.

Esse setor é um dos mais relevantes para o desenvolvimento cultural, social e econômico de países, cidades e regiões incrementando rendimentos e receitas significativas.

O Turismo é formado por uma cadeia produtiva de vários segmentos que geram muitos empregos em todo o mundo. Durante a crise da pandemia, houve uma redução considerável nas vendas de pacotes turísticos, principalmente devido ao isolamento social.

A Raja Turismo como uma agência de viagens que atua na plataforma online, como solução para as baixas vendas de pacotes turísticos, optou por inovar sua forma de marketing digital através do desenvolvimento de um aplicativo.

Esse aplicativo foi desenvolvido a partir das plataformas Android Studio e

Firebase. Verificou-se que, o uso do aplicativo é de fácil acesso e permite ao cliente escolher qual destino quer conhecer, quantas pessoas irão viajar e quais serviços querem contratar. Além disso, a forma de pagamento também incluiu o *Pix*, visando facilitar ainda mais o processo de compra online de viagens turísticas.

Conclui-se que, o desenvolvimento desse aplicativo foi satisfatório, o mesmo será lançado ao mercado no início de junho desse ano, devido aos testes que ainda estão sendo desenvolvidos com o intuito de evitar transtornos aos clientes, como por exemplo, os bugs.

REFERÊNCIAS

BRASIL. Banco Central. **O que é o *Pix***. Disponível em: https://www.bcb.gov.br/estabilidadefinanceira/pix. Acesso em: 20 abr. 2022.

BIZ, A; NEVES, A.; BETTONI, A. J. M. O comportamento dos consumidores turísticos no uso da telefonia móvel. **Caderno Virtual de Turismo**, vol. 14, n. 1, p. 34-48.
Universidade Federal do Rio de Janeiro. Río de Janeiro, Brasil. 2014. Disponível em: http://www.redalyc.org/articulo.oa?id=115431119003. Acesso em: 12 dez. 2020.

CUNHA, L. **Economia e Política do Turismo**. Ed. Lidel; 3ª edição. 2013.

MENDES, A. S. do. C. **Marketing Digital no Turismo em Pandemia: O Caso das Agências de Viagens.** Instituto Politécnico de Coimbra. Instituto Superior de Contabilidade e Administração de Coimbra. Coimbra, Portugal, 2021. Disponível em: https://comum.rcaap.pt/handle/10400.26/38997. Acesso em: 3 abr. 2022.

PATRIOTA, V. A. N. **Uso de Aplicativos Móveis como Ferramenta no Planejamento de Viagens em Tempos da Covid-19.** Monografia. Universidade Federal do Rio Grande do Norte. Centro de Ciências Sociais Aplicadas. Departamento de Turismo, Natal – RN; 2021. Disponível em: https://repositorio.ufrn.br/handle/123456789/42543. Acesso em: 10 abr. 2022.

RAMOS, D. M.; COSTA, C. M. Turismo: Tendências De Evolução. PRACS: **Revista Eletrônica de Humanidades Do Curso de Ciências Sociais Da UNIFAP**, v. 10, n. 1, 2017. Disponível em: https://periodicos.unifap.br/index.php/pracs/article/view/2843. Acesso em: 10 abr. 2022.

SILVA, L. A. da. Apostila de Android: Programando Passo a Passo Programação Básica. Versão Android Studio. 2015. Disponível em: https://docplayer.com.br/9920567-Apostila-de-programando-passo-a-passo-3a-edicao-de-luciano-alves-da-silva-lucianopascal-yahoo-com-br.html. Acesso em: 21 abr. 2022.

TOMÉ, L. M. Setor de Turismo: Impactos da Pandemia. Banco do Nordeste. **Caderno Setorial ETENE**. Ano 5, n. 122, ago. 2020. Disponível em: https://www.bnb.gov.br/s482-dspace/bitstream/123456789/300/1/2020_CDS_124.pdf. Acesso em: 21 abr. 2022.

CRYPTO NA MÃO - PROTOTIPAÇÃO DE UM APLICAÇÃO *MOBILE* PARA AUXILIAR NA GESTÃO DE CARTEIRA PESSOAL DE CRIPTOATIVOS

GARCIA, João Henrique Farias
ROCHA, Gláucio Bezerra

RESUMO

O uso de aplicativos como ferramenta cotidiana é fato notório nos dias atuais, a cada dia surgem novas opções de "*apps*" com o objetivo de atender uma necessidade específica, foi nesse contexto que desenvolvemos o projeto “Crypto na Mão”, um sistema que pretende oferecer mais tranquilidade aos investidores de criptomoedas facilitando o monitoramento de sua carteira de investimentos, através do armazenamento das informações desses aportes e notificando-os com mensagens de alertas conforme movimentações do mercado. Dessa forma o investidor pode potencializar seus ganhos e mitigar perdas. No presente projeto, discorreremos sobre várias ferramentas de análise e modelagem de sistemas para embasar a prototipação e implementação final da solução proposta.

Palavras-chave: Investimentos; criptomoedas; mercado financeiro; aplicativo.

1 INTRODUÇÃO

A estabilidade econômica no Brasil, dentre outros benefícios, proporcionou o surgimento de uma nova classe de investidores. Pessoas comuns, sem qualquer entendimento de mercado financeiro, passaram a buscar formas para aplicar seus recursos e lucrar com isso, entretanto o caminho ofertado pelo mercado financeiro tradicional, através das bolsas de valores, ainda hoje se mostra um tanto quanto complicado para o leigo. Paralelo a isso, Vidal (2021) relata que no ano de 2009, um grupo de programadores ou um programador, não se sabe ao certo, fazendo uso do pseudônimo Satoshi Nakamoto lançou a criptomoeda *Bitcoin*, a primeira moeda digital descentralizada do mundo, baseada na tecnologia de *blockchain*, um dinheiro sem amarras de papel, geográfica ou governamental, tal novidade veio a promover uma verdadeira revolução no mercado financeiro. O advento do *Bitcoin* fez surgir várias outras criptomoedas, alguns projetos sólidos como o *Ethereum* e outros nem tanto, todavia proporcionando bons lucros a quem passou a explorá-los.

No Brasil, em decorrência desse súbito interesse, várias corretoras foram criadas para auxiliar os investidores a entender e fazer aplicações neste novo mercado, segundo Oliveira (2022), em matéria publicada pelo Estadão em 14/02/2022, o Brasil registrou “um aumento de 1.266% no número de investidores alocados em fundos *ETFs* (*Exchange Traded Fund*, em inglês) e de *criptoativos* em

2021 ante o ano anterior", atingindo a incrível marca 427 mil players e R$ 2.7 bi aportados.

Mesmo com este crescimento acentuado e uma notória evolução das ferramentas, os clientes ainda não se sentem seguros e encontram muita dificuldade para acompanhar seus investimentos em moedas digitais, pois a maioria das plataformas oferecem gráficos e relatórios complexos e de difícil entendimento, o que poda uma maior atividade desses investidores.

Pensando nisso, o projeto "Crypto na Mão" se propõe a oferecer um aplicativo de fácil manuseio, baseado na plataforma *Android*, onde o usuário poderá lançar seus investimentos, acompanhar cotações de mercado em tempo real e receber notificações, possibilitando ações rápidas e uma tomada de decisão mais precisa.

2 FUNDAMENTAÇÃO TEÓRICA

Projetos que visam dar suporte à gestão de aplicações financeiras já foram desenvolvidos em vários âmbitos, depositar um capital em algo que proporcione lucros é um objetivo almejado por 10 entre 10 investidores, porém nem todos têm a perspicácia necessária para gerir de forma eficiente sua carteira.

Partindo deste cenário, captamos alguns trabalhos acadêmicos como referência. O primeiro, elaborado por Gonçalves(2020) para a UFSC, com o título: "Desenvolvimento de uma plataforma de investimentos em *criptoativos* baseada em *Ethereum*", tem como objetivo democratizar as oportunidades de investimentos neste segmento através de uma plataforma web. O sistema desenvolvido utiliza a *blockchain Ethereum* para registrar as transações na plataforma.

Outra referência vem de Mazzochi (2020) em projeto desenvolvido para a UNIFACVEST, com o título: "Análise de Investimentos na Bolsa de Valores", em seu trabalho o autor discorre com o objetivo de analisar e demonstrar a importância de realizar aportes no mercado financeiro, bem como desenvolver uma aplicação que utilize aprendizado de máquina para análise desses investimentos.

Os trabalhos referenciados, partem de um princípio comum: a necessidade de uma ferramenta simplificada para auxiliar os investidores, dada a grande complexidade e quantidade de informações disponibilizadas pelo mercado financeiro, seja ele o tradicional ou o de *criptoativos*.

Ferramentas de UML (*Unified Modeling Language*, Linguagem de

Modelagem Unificada em tradução livre) se mostraram importantes para compreensão do todo, além de uma boa análise de requisitos. Agregando-se isso, a modelagem de dados conceitual e lógica também foram utilizadas e serão objetos presentes no trabalho, o ponto de ruptura estará na linguagem de programação utilizada, em função da proposta trazida estar focada em desenvolver um protótipo. Por esta razão, a aplicação terá sua implementação pautada no App Inventor, plataforma web disponibilizada pelo MIT - *Massachusetts Institute of Technology*, MA–USA, através do site https://appinventor.mit.edu, o qual possibilita a criação de aplicativos android de forma simples e eficaz.

Em seu artigo Meyer (2020) descreve que o Android é um sistema operacional de código aberto, baseado no núcleo Linux, para dispositivos móveis desenvolvido pela Open Handset Alliance, pool de empresas composto pela Google e outros players de tecnologia. Estima-se que mais de 200.000 dispositivos com este sistema operacional são vendidos todos os dias em todo o mundo, fazendo-o líder no segmento.

3 METODOLOGIA

Uma análise das projeções do mercado financeiro e das ferramentas disponíveis para os clientes que fazem, ou desejam fazer, investimentos neste segmento, associada a pesquisa de alguns trabalhos acadêmicos, foram o ponto de partida para idealização deste projeto, dada a complexidade apresentada para realizar simples controles.

As etapas que se seguem servirão para embasar todo o projeto facilitando seu entendimento e desenvolvimento final. A análise de requisitos estará na vanguarda de tudo, descrevendo de forma clara todas as funcionalidades que o sistema deve oferecer. Na sequência, serão utilizadas ferramentas para modelagem de dados onde, através dos modelos conceitual e lógico, iremos estruturar o fluxo e a forma como os dados serão utilizados no sistema.

Todos esses processos servirão de esteio para a implementação final do aplicativo, pois cada uma dessas etapas estão interligadas consolidando a proposição do trabalho que será representada pelo protótipo desenvolvido.

3.1 ANÁLISE DE REQUISITOS

O processo de levantamento de requisitos é responsável por identificar todas as necessidades e demandas do cliente e do software, que devem ser

descritas especificando suas funcionalidades, comportamentos e características. Sendo esses requisitos classificados como Funcionais ou Não Funcionais (ALFF - 2018).

- Requisitos Funcionais

Podemos entender por requisitos funcionais todas as necessidades ou funcionalidades, desejadas pelos usuários, que devem ser atendidas pelo software.

ID	NOME	DESCRIÇÃO	PRIORIDADE
RF01	Cadastro de Usuários	Os usuários deverão se cadastrar, com o objetivo de garantir a autenticidade e segurança no acesso.	Essencial
RF02	Cadastrar Operações	O cadastro das operações de compra ou venda de ativos deve ser realizado no aplicativo, com o objetivo de manter a carteira atualizada.	Essencial
RF03	Listar todos as operações	A listagem de todas as operações cadastradas deve estar disponível.	Essencial
RF04	Editar operação	O aplicativo deve permitir a edição da operação cadastrada.	Essencial
RF05 RF06 RF07 RF08	Excluir operação Filtrar operação por tipo Filtrar operação por período Filtrar operação por criptomoeda	O aplicativo deve permitir a exclusão da operação cadastrada. O aplicativo deve permitir filtrar as operações cadastradas por tipo (compra/venda). O aplicativo deve permitir filtrar as operações cadastradas por período, com data inicial e final. O aplicativo deve permitir filtrar as operações cadastradas por criptomoeda;	Essencial Essencial Essencial Essencial
RF09	Cadastrar Alertas	O aplicativo deve permitir o cadastro de alertas para que o sistema notifique o usuário no momento que determinada criptomoeda atinja uma cotação específica.	Essencial

RF10	Listar todos os alertas	A listagem de todos os alertas cadastrados deve estar disponível.	Essencial
RF11	Editar alerta	O aplicativo deve permitir a edição do alerta cadastrado.	Essencial
RF12	Excluir alerta	O aplicativo deve permitir a exclusão do alerta cadastrado.	Essencial
RF13	Enviar notificações	O aplicativo deve oferecer as opções para que o usuário seja notificado, por email, sms, ou pelo próprio aplicativo.	Essencial
RF14	Exibir cotações atualizadas	O aplicativo deve exibir uma lista com as cotações das criptomoedas atualizadas.	Essencial
RF15	Exibir a carteira de investimentos	O aplicativo deve oferecer ao usuário uma visualização de sua carteira de investimentos contendo, o nome do ativo, a quantidade em estoque, o valor real em moeda corrente e o resultado médio daquele investimento.	Essencial
RF16	Relatório de operações	O aplicativo deve permitir o download da listagem de operações realizadas.	Essencial
RF17	Relatório de cotação atual	O aplicativo deve permitir o download da listagem das cotações de mercado vigentes.	Essencial
RF18	Relatório da carteira	O aplicativo deve permitir o download da carteira de investimentos.	Essencial

- Requisitos Não Funcionais

Os requisitos não funcionais são tratados como premissas e restrições técnicas do projeto e devem ser mensuráveis.

ID	NOME	DESCRIÇÃO	PRIORIDADE
RNF01	Desempenho	O aplicativo deve garantir desempenho satisfatório, com o intuito de justificar sua utilização em detrimento a outras opções de mercado.	Essencial

RNF02	Fluidez	O aplicativo deve permitir a rápida transição entre as telas, a partir de todos os recursos disponíveis.	Essencial
RNF03	Resiliência	O aplicativo possibilitará ajustes com rapidez de forma periódica e ocasional, evitando perda de disponibilidade.	Importante
RNF04	Tempo de resposta	O aplicativo deve oferecer rápida consulta ao banco de dados e disponibilização de dados.	Desejável
RNF05	Segurança	O aplicativo deve garantir autenticidade, disponibilidade, integridade e restrições de acesso.	Essencial
RNF06	Restrição de acesso	O aplicativo deverá assegurar que cada tipo de usuário possua acesso exclusivo à sua carteira de investimentos.	Essencial
RNF07	Disponibilidade	O aplicativo terá alta capacidade de provimento de serviço contínuo, com baixa probabilidade de indisponibilidade casual, e sincronização dos dados em tempo real.	Importante
RNF08	Simplicidade	O aplicativo apresentará facilidade de uso, com poucos “cliques” para se alcançar qualquer funcionalidade. Interface simples e intuitiva, garantindo usabilidade.	Desejável
RNF09	Compatibilidade	O aplicativo deve ser utilizado no sistema operacional Android .	Essencial

3.2. MODELAGEM DE DADOS

Na modelagem de dados podemos descrever quais tipos de informação serão persistidas e como elas são armazenadas em um sistema de banco de dados. Neste conceito podemos abstrair 02 modelos o conceitual e o lógico.

- Modelo Conceitual

Um modelo conceitual é uma descrição do banco de dados, de forma independente de implementação, em um sistema gerenciador de banco de dados.

O modelo conceitual registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados a nível de SGBD (HEUSER, 1998).

Figura 1 : Modelo conceitual

Fonte: Próprio Autor (2022)

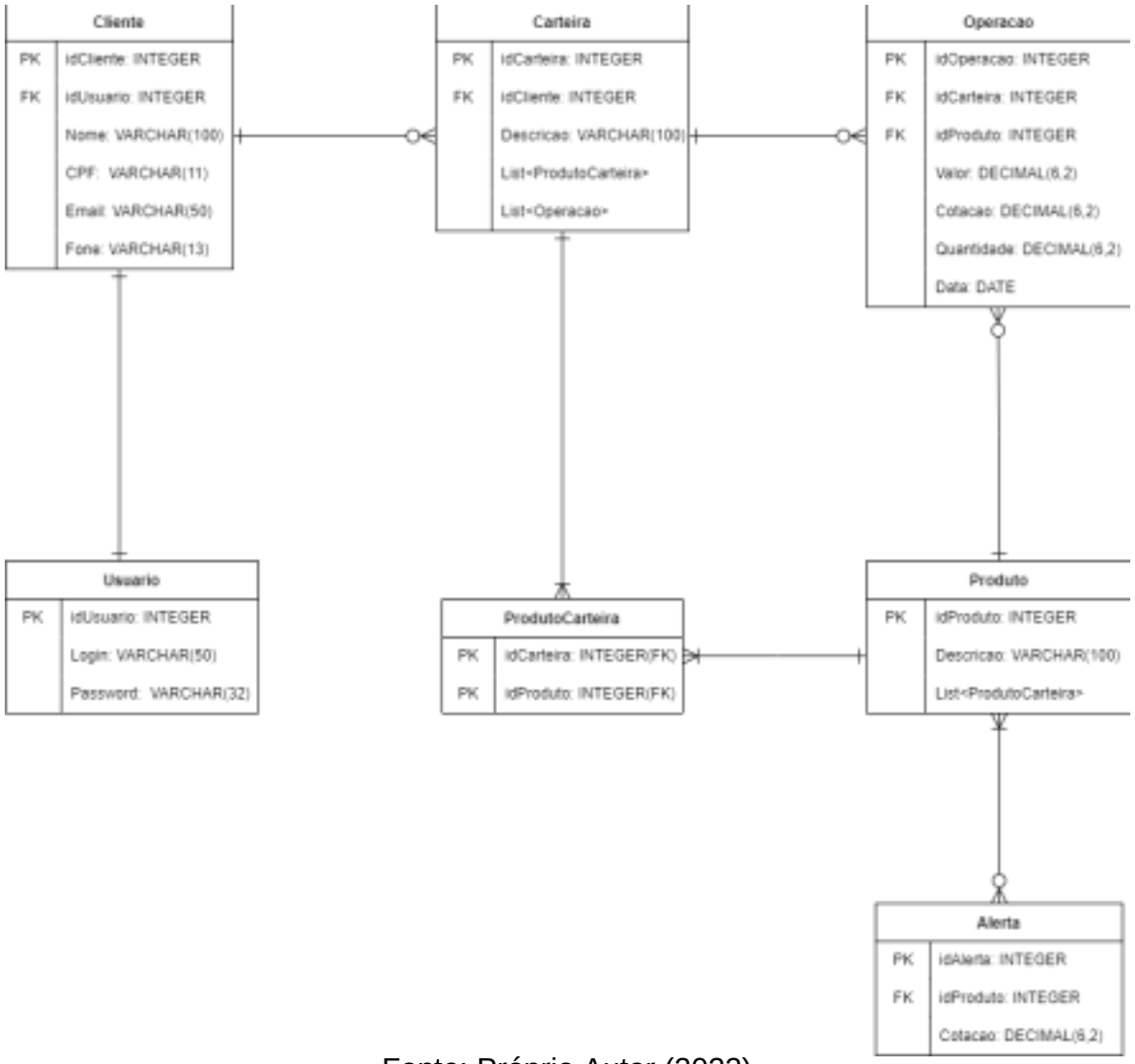
O modelo apresentado na figura 1, demonstra a relação entre as diversas entidades presentes nos requisitos funcionais da aplicação, como já denotado no

seu nome. Analisando o diagrama acima podemos identificar a necessidade da utilização de uma classe intermediária entre as entidades “carteira” e “produto” dada a relação múltipla entre elas.

- Modelo Lógico

Um modelo lógico é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, o modelo lógico é dependente do tipo particular deste SGBD que está sendo usado (HEUSER, 1998).

Figura 2 : Modelo lógico



Fonte: Próprio Autor (2022)

Na elaboração do modelo lógico, representado na figura 2, são preservadas

as cardinalidades presentes no modelo conceitual, porém apresentando as entidades como tabelas, nelas identifica-se as chaves primárias (abreviadas como PKs, do inglês Primary Key) as quais tornam-se estrangeiras (FKs, do inglês Foreign Key) quando vinculadas a outras tabelas. Os campos têm sua tipagem seguindo o padrão SQL.

4 FUNDAMENTAÇÃO TEÓRICA

4.1. IMPLEMENTAÇÃO

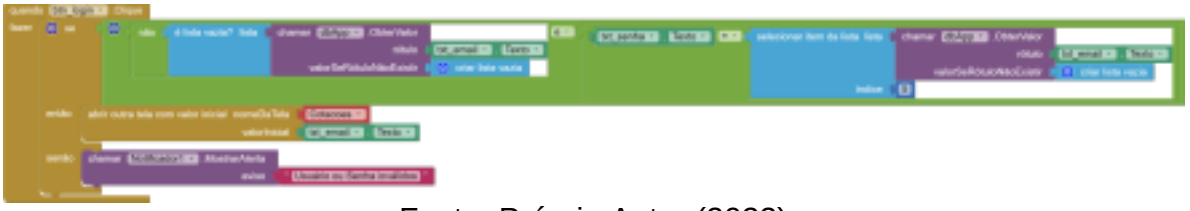
A implementação do MVP, mínimo produto viável, do protótipo foi realizada no MIT App Inventor através da programação em blocos.

Figura 3: Conexão com API CoinGecko

Fonte: Próprio Autor (2022)

Na Figura 3 podemos visualizar a codificação da funcionalidade que realiza a conexão do aplicativo com a API CoinGecko, para porcionar a obtenção dos valores atualizados dos ativos.

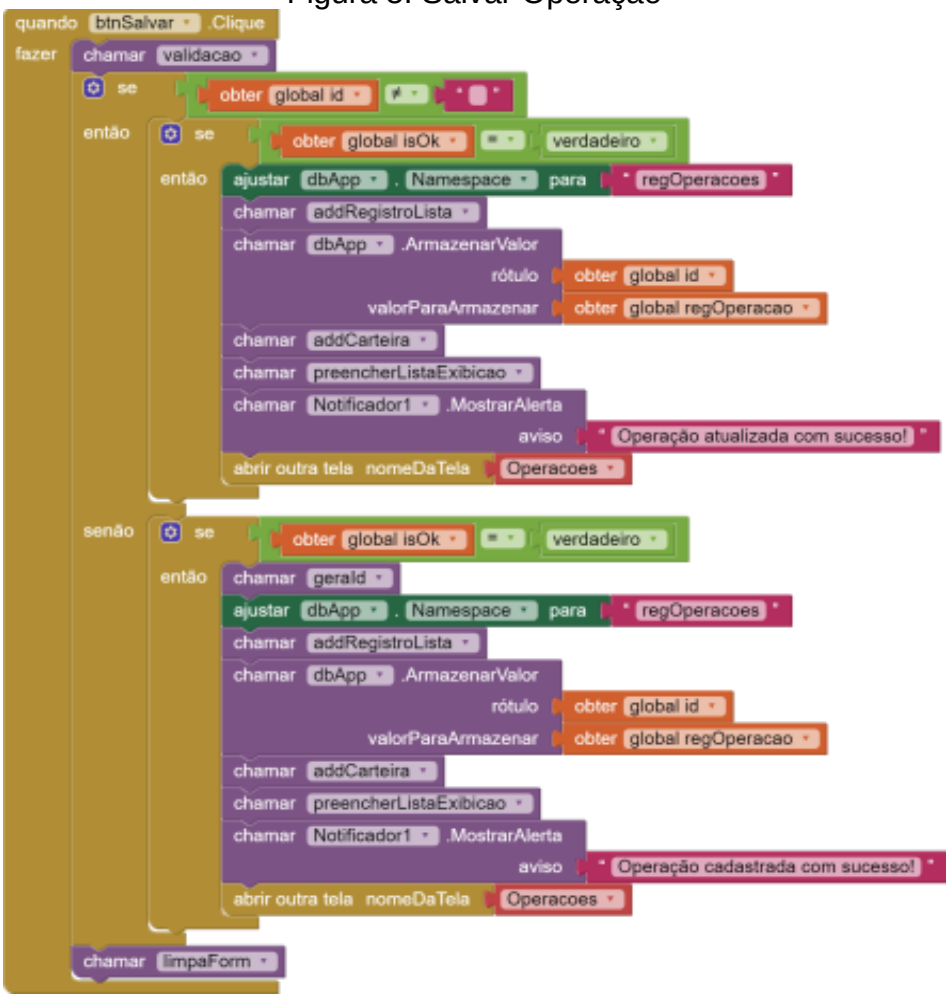
Figura 4: Validação de Login



Fonte: Próprio Autor (2022)

Na Figura 4 podemos visualizar a codificação para validação do login e senha do usuário para acesso ao sistema com consulta ao banco de dados.

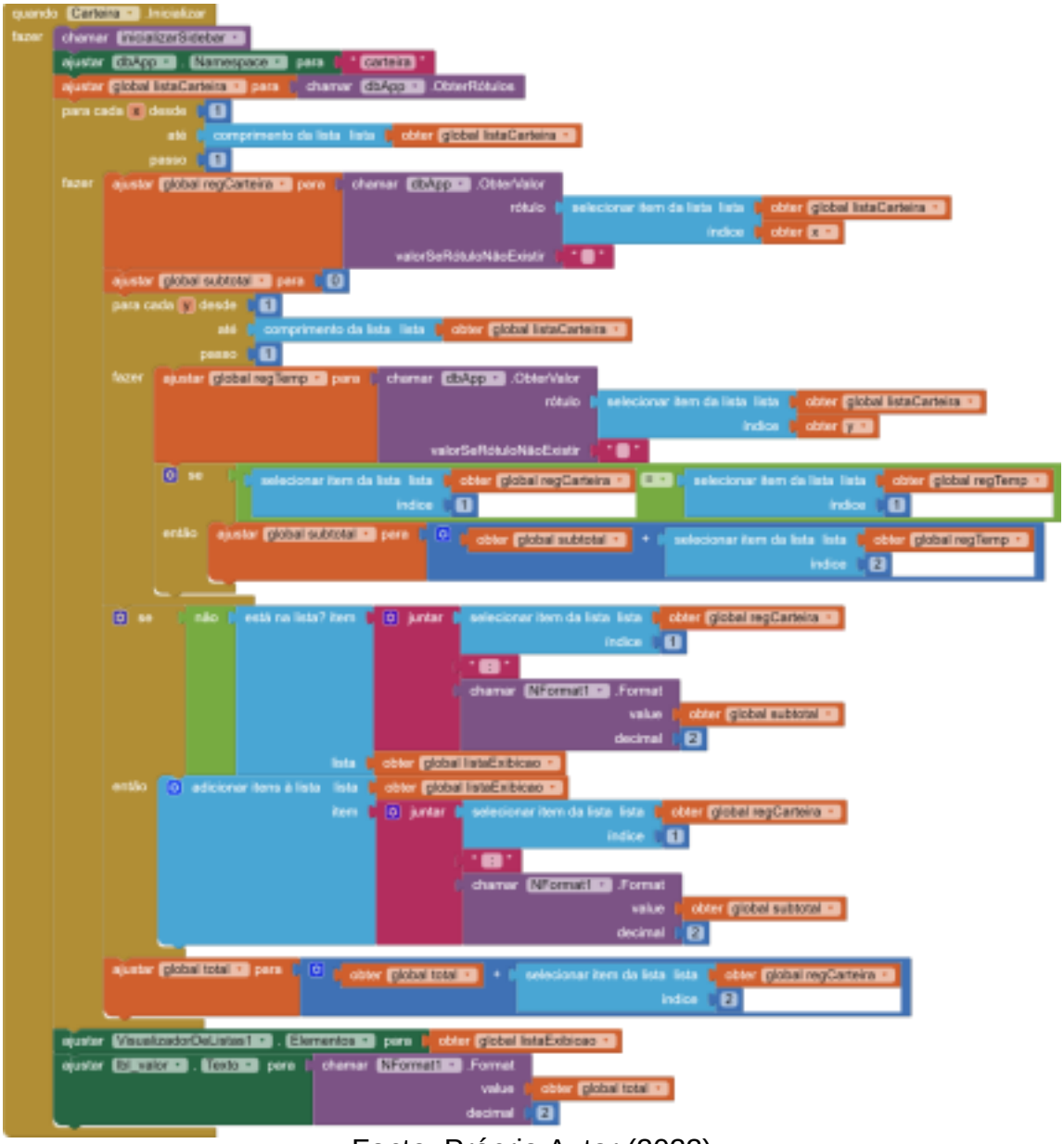
Figura 5: Salvar Operação



Fonte: Próprio Autor (2022)

Na Figura 5 podemos visualizar a codificação para persistência dos dados da operação no banco de dados.

Figura 6: Exibição Carteira



Fonte: Próprio Autor (2022)

Na Figura 6 podemos visualizar a codificação da funcionalidade que realiza a inicialização e exibição da carteira de investimentos.

4.2 DESIGN DE INTERFACE DE USUÁRIO

Figura 7: Tela de Login

Fonte: Próprio Autor (2022)

Figura 8: Tela Cadastro de Usuário

Fonte: Próprio Autor (2022)

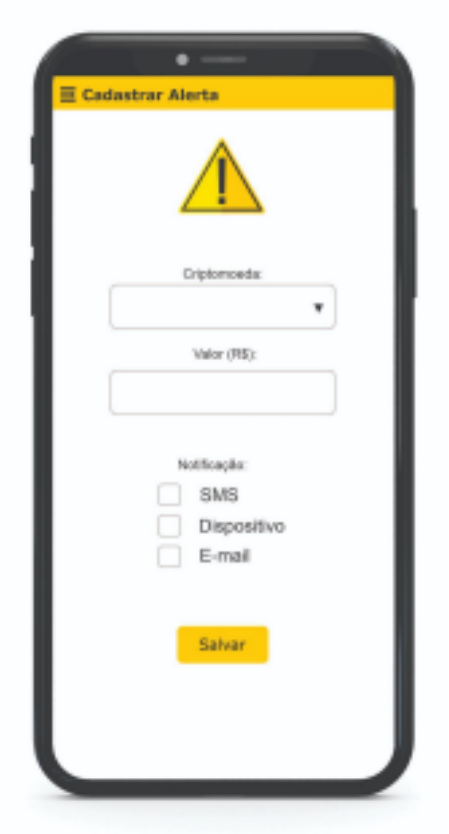
Login

Na figura 7, é possível visualizar a tela onde o usuário irá inserir o e-mail e a senha cadastrados para ter acesso ao aplicativo, caso o mesmo ainda não seja cadastrado, ele poderá escolher a opção “Não tenho cadastro”, assim será redirecionado à página de cadastro de usuário. Caso haja a necessidade de recuperar sua senha também poderá fazê-lo clicando em “Esqueci minha senha”, um e-mail será enviado para o endereço de e-mail cadastrado com a senha vigente.

Cadastro de Usuário

Na figura 8, é possível visualizar a tela onde o usuário irá inserir seus dados para realizar o cadastro no sistema, sendo obrigatório o preenchimento do nome, fone, e-mail e senha, para acesso ao sistema, o CPF é opcional.

Figura 9: Tela Cadastro de Alerta



Fonte: Próprio Autor (2022)

Cadastrar Alerta

Na figura 9, é possível visualizar a tela onde o usuário irá cadastrar alertas de cotação para as criptomoedas desejadas, seja com objetivo de compra ou venda, quando a cotação de mercado atingir o valor desejado, serão enviados alertas com base nas opções de notificação escolhidas. Podem ser cadastrados quantos alertas desejar, para as mais variadas criptos.

Figura 10: Tela adastrar Operação

Fonte: Próprio Autor (2022

Cadastrar Operação

Na figura 10, é possível visualizar a tela onde o usuário irá cadastrar as operações feitas em sua corretora para manter sua carteira no aplicativo atualizada, aqui deverá ser registrado qual a criptomoeda e o valor da operação, seja compra ou venda ao confirmar a carteira será atualizada.

Figura11- Tela Operações Realizadas

Fonte: Próprio Autor (2022)

Operações Realizadas

Na figura 11, é possível visualizar a tela onde serão exibidas todas as operações realizadas, podendo o usuário editar

ou excluir qualquer operação. Nesta tela também haverá um filtro oculto, que será exibido quando o usuário clicar no ícone de filtro, possibilitando ao usuário filtrar o período desejado, o tipo compra ou venda e uma criptomoeda específica.

Figura 12 - Tela Alertas Cadastrados

Fonte: Próprio Autor (2022)

Alertas Cadastrados
Na figura 12, é possível visualizar a tela onde serão mostrados todos os alertas cadastrados no sistema, aqui o usuário poderá realizar ações de edição, exclusão.

Figura 13 - Cotação Atual

Fonte: Próprio Autor (2022)

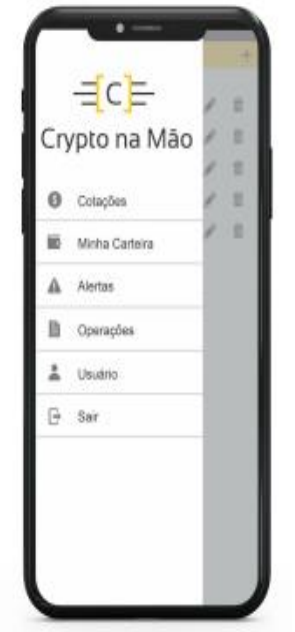
Figura 14 - Carteira

Fonte: Próprio Autor (2022)

Cotação Atual
Na figura 13, é possível visualizar a tela onde o usuário terá acesso a cotação das principais criptomoedas do mercado, tendo a opção de visualizar seu valor de mercado em real, dólar ou euro.

Carteira
Na figura 14, é possível visualizar a tela onde o usuário irá visualizar sua carteira como um todo, com seus valores atualizados, o volume de ativos e o resultado médio de seus investimentos para cada ativo, confrontando valores de compra e cotação de mercado.

Figura 14 - Carteira



Fonte: Próprio Autor (2022)

Menu
Na Figura 14 é possível visualizar o menu da aplicação para facilitar a navegação entre as telas. Opções oferecidas: Cotações, Minha Carteira, Alertas, Operações, Usuário, Sair.

6 CONSIDERAÇÕES FINAIS

Na construção do projeto, podemos perceber que a pesquisa bibliográfica, tendo outras referências acadêmicas e o embasamento teórico fornecidos em livros e publicações, foram essenciais para conceituar o aplicativo como um todo, entretanto para que o mesmo fosse desenvolvido de forma assertiva a utilização das diversas ferramentas foram essenciais.

Inicialmente a análise de requisitos norteou todas as demais etapas, pois nela identificamos as necessidades dos usuários, a partir de então pudemos planejar como atender cada uma delas. A modelagem de dados, nos seus 02 âmbitos, conceitual e lógico, nos mostram de forma clara as entidades envolvidas, seus atributos e como elas se relacionam entre si, fator indispensável para uma boa implementação e eficiência do SGBD. Por fim desenvolvemos o design de interface de usuário, exemplificando em alta fidelidade qual será o resultado da prototipação final em sua versão beta. Concluímos assim que tais processos ratificam a importância de se dedicar na análise e elaboração das etapas de planejamento, pois dessa forma a implementação se torna muito mais produtiva e eficaz.

REFERÊNCIAS

ALFF, Chico, Engenharia de Requisitos , https://analisederequisitos.com.br, 2018.

GONÇALVES, Gabriel José Prá - TCC: Desenvolvimento de uma plataforma de investimentos em criptoativos baseada em Ethereum - Santa Catarina - UFSC: 2020;

HEUSER ,Carlos Alberto.Projeto de banco de dados 6ª edição, Editora Sagra, 1998.

MAZZOCHI, Rafael do Canto (2020) – TCC : Análise de Investimentos na Bolsa de Valores – Centro Universitário UNIFACVEST – Lages – SC: 2020

MEYER, Maximiliano - A história do Android - Oficina da Net: 2020 https://www.oficinadanet.com.br/post/13939-a-historia-do-android.

OLIVEIRA, Isaac de. Matéria: Fundos cripto: número de investidores cresce 1.266% no Brasil em 2021 - Estadão - 2022 - https://einvestidor.estadao.com.br/ criptomoedas/investidores-fundo-criptomoedas-crescem-2021

VIDAL, Vitor. Bitcoin: Descubra sua história e momentos marcantes - Showmetech - 2021 https://www.showmetech.com.br/historia-do-bitcoin-e- momentos- marcantes/

SISTEMA PARA ARTESANATO EM PEÇAS DE CROCHÊ EM JOÃO PESSOA

FERREIRA, Pedro Rhamon Sousa
ROCHA, Gláucio Bezerra

RESUMO

Este estudo tem como objetivo discutir a relevância dos sistemas de captura de *leads* para aumentar as vendas de artesanato. Há uma grande necessidade no setor de artesanato devido ao grande impacto da pandemia, bem como no segmento de vendas de artesanato. Para aumentar as vendas, a pesquisa analisou a importância de sistemas eficazes para entrega do produto ao cliente e coleta de dados do cliente para realizar vendas de peças. Consequentemente, a enquete foi realizada com o objetivo de apresentar a importância de um sistema para este segmento. Assim, foram discutidas as principais tecnologias de informação, tais como: *React, java, SpringBoot, Postgresql* para o desenvolvimento do comércio eletrônico. Consequentemente, pode-se concluir que investir em um sistema eficiente é de extrema importância para alcançar excelentes resultados no mercado de artesanato.

Palavras-chave: Sistemas; *leads*; e-ecomerce; vendas; cliente; *React; Java; SpringBoot; Postgresql.*

1 INTRODUÇÃO

Hoje, os avanços na tecnologia da informação produziram melhorias na captura de dados de clientes para fornecer produtos e serviços. Assim, entende-se que a informatização tem crescido exponencialmente com excelentes resultados. Portanto, o objetivo do sistema é fornecer aos clientes uma plataforma para inserir dados para mostrar suas obras de arte, gerando receita. Ter uma compreensão real de como as peças são feitas.

Nesse contexto, fica claro que um sistema de aquisição de dados de clientes *(LEADS)* é essencial para fortalecer o relacionamento com os clientes para venda de objetos. Dessa forma, o sistema construirá um melhor relacionamento com o cliente.

Para o desenvolvimento do e-commerce, utilizaremos tecnologias legadas como: *Java* e a Biblioteca *React Js*. Para dados de usuários e clientes, usaremos o banco de dados *postgresql*. Portanto, utilizando essas técnicas,
acredita-se que um sistema simples e eficiente possa ser desenvolvido para atingir os objetivos propostos.

Ao mesmo tempo, este estudo visa discutir a relevância dos sistemas de captura de *leads* para a venda de artesanato ao público. Portanto, a realização de pesquisas atuais nessa escala ajudará a refletir sobre a importância da tecnologia da informação na área de captação de *leads* e em que medida a ferramenta pode

otimizar o processo de venda de artesanato e construir melhor relacionamento com clientes e usuários.

Diante disso, acredita-se que essa discussão seja bastante relevante, no segmento do artesanato e a contribuição que estas ferramentas podem proporcionar. Os avanços recentes da tecnologia, o investimento da sociedade no sistema e as vantagens que a informação pode proporcionar ao departamento de vendas.

2 FUNDAMENTAÇÃO TEÓRICA

O aplicativo será baseado no conceito de entrega contínua, visando melhorar continuamente o aplicativo através de versões, iniciaremos com um MVP *(Minimum Viable Product)* básico com funcionalidade de usuário cadastrado, permitindo que os artesãos registrem seu trabalho com uma breve descrição e valor, os usuários podem navegar entre esses títulos e fazer compras.

2.1 RAMO ARTESANATO

De acordo com a empresa DINO (2020), um dos mercados amplamente beneficiados pela ascensão da economia criativa no Brasil foi o de artesanato. Segundo dados IBGE de 2019, o setor movimenta cerca de 50 bilhões por ano no país e é fonte de renda para aproximadamente de 10 milhões. Essa receita é referente, principalmente, a pequenos negócios, sendo a formalização o principal desafio no setor. Ainda de acordo com o IBGE, apenas 40% dos negócios de artesanato possuem CNPJ. No entanto, nos últimos anos, passaram a existir cada vez mais exemplos de empresas que conseguiram aliar industrialização e prática artesanal, tornando-se mais competitivas, sem perder o apelo criativo e emocional de suas criações.

Segundo ASN, Agência Sebrae de Notícias (2022), as oportunidades de vendas e a divulgação de produtos com o apoio de ferramentas digitais se tornou realidade para os pequenos negócios durante a pandemia. Não foi diferente também para os artesãos que são empreendedores. Com as restrições para a realização de feiras e eventos, principal espaço ocupado pela categoria no país, o mercado on-line tem sido a aposta para a recuperação do faturamento e retomada dos negócios.

Dados da 13ª pesquisa de impacto da pandemia nos pequenos negócios, produzida pelo Sebrae em parceria com a Fundação Getúlio Vargas (FGV) (2022),

mostra que 78% dos empreendimentos comandados por empreendedores do setor de artesanato já voltaram a funcionar. A maioria deles, no entanto, precisou se adaptar por causa da crise e ainda sente o peso das mudanças, com perdas de faturamento em torno de 38% se comparado ao período anterior à pandemia.

O levantamento também aponta que 84% dos artesãos empreendedores têm utilizado as redes sociais, aplicativos e internet como ferramenta de vendas, principalmente *Whatsapp Business* (83%), *Instagram*(67%) e *Facebook* (56%).

A eclosão da pandemia da Covid-19 representou uma alteração significativa na rotina e na forma de vida de bilhões de pessoas ao redor do planeta. Os padrões de trabalho que constituíam muitas economias globais não permaneceram intactos, pelo contrário, alterações recentes na dinâmica das tarefas prometem permanecer por um longo período na esfera organizacional, até que outro marco obrigue os humanos a reestruturarem comportamentos.

No caso dos artesãos, após uma trajetória de sucesso crescente na venda das peças manuais nos últimos anos, foi necessário enfrentar a impossibilidade de realização de feiras de artesanato e demais eventos que estabeleciam o contato direto com os consumidores, arranjo responsável pela renda integral de indivíduos. Houve queda expressiva na arrecadação geral e muitos empreendimentos foram encerrados. Rede Artesanato Brasil (2021).

2.2 TECNOLOGIAS E E-ECOMMERCE

Usaremos *Java* como linguagem de programação e seu *frameworkspringBoot* no *backend*, Java é uma linguagem bastante robusta que segue o paradigma *Object Oriented* (O.O.) que permite aos desenvolvedores tornar as aplicações mais seguras e fáceis de entender pois seguem as mesmas Relações Similares Neste mundo, os objetos têm seus usos e podem ser usados para manipular, alterar, transformar ou formar novos objetos com funções maiores e melhores. O Spring Boot é uma ferramenta que visa facilitar o processo de configuração e publicação de aplicações que utilizem o ecossistema Spring.

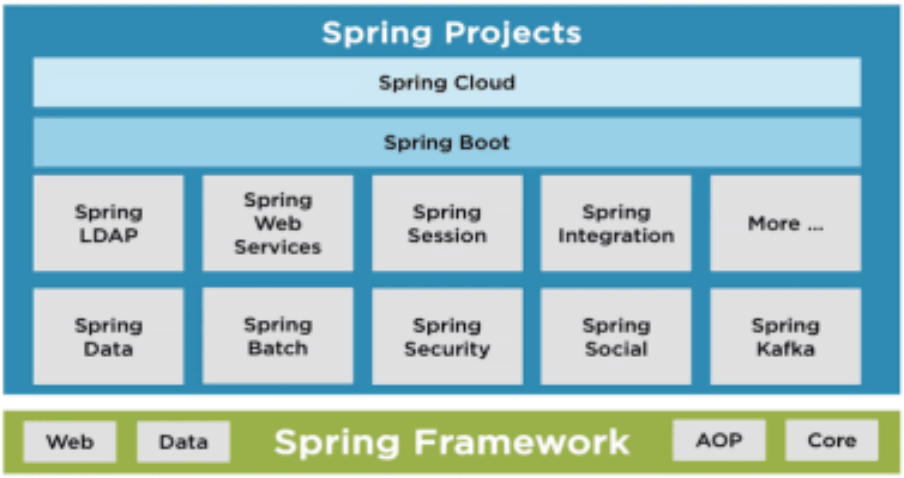
O Spring Boot é um projeto da Spring que veio para facilitar o processo de configuração e publicação de nossas aplicações. A intenção é ter o seu projeto rodando o mais rápido possível e sem complicação.

Ele consegue isso favorecendo a convenção sobre a configuração. Basta que você diga pra ele quais módulos deseja utilizar (*WEB*, *Template*, Persistência,

Segurança, etc.) que ele vai reconhecer e configurar.

Você escolhe os módulos que deseja através dos starters que inclui no pom.xml do seu projeto. Eles, basicamente, são dependências que agrupam outras dependências. Inclusive, como temos esse grupo de dependências representadas pelo starter, nosso pom.xml acaba por ficar mais organizado.

Apesar do Spring Boot, através da convenção, já deixar tudo configurado, nada impede que você crie as suas customizações caso sejam necessárias.



O maior benefício do Spring Boot é que ele nos deixa mais livres para pensarmos nas regras de negócio da nossa aplicação. Alexandre Afonso(2017).

Utilizaremos a biblioteca *ReactJS*, como *frontend*, flexível e facilita a interface com outras bibliotecas e frameworks. O *React* é “uma biblioteca de *JavaScript* para construir interfaces de usuário (UI)”. Comecemos por um importante aspecto: *React* é uma biblioteca, não um *framework*. Isso quer dizer que o *React* contempla um conjunto de funcionalidades relacionadas que podem ser requisitadas pelos desenvolvedores(as) na construção de interfaces do usuário (UI). O que significa que a desenvolvedora ou desenvolvedor é quem determina como são construídas as aplicações usando recursos disponíveis nesta biblioteca.

No caso de um *framework*, há o que se chama de “Inversão de Controle”, fazendo com que quem dite “o que” e “como” será construída a aplicação seja o framework, fazendo a/o dev se adequar a como usar seu código em meio aos elementos de que dispõe o framework. Ou seja, a biblioteca resulta em maior liberdade e flexibilidade, mas acaba sendo menos “completa” em termos de módulos e recursos à disposição. atualmente de acordo com Giovanni Salvador (2018).

Iremos aplicar o sistema de gerenciamento de banco de dados objeto

relacional hoje em dia conhecido por *PostgreSQL* (e por um breve período de tempo chamado *Postgres95*) é derivado do pacote *POSTGRES*escrito na Universidade da Califórnia em Berkeley. Com mais de uma década de desenvolvimento por trás, o *PostgreSQL* é o mais avançado banco de dados de código aberto disponível em qualquer lugar, oferecendo controle de concorrência multi-versão, suportando praticamente todas as construções do *SQL* (incluindo subconsultas, transações, tipos definidos pelo usuário e funções), e dispondo de um amplo conjunto de ligações com linguagens procedurais (incluindo C, C++, Java, Perl, Tcl e Python). *The PostgreSQL Global Development Group(1996-2002).*

Segundo Daniel Sampaio, um e-commerce, ou comércio eletrônico, refere-se aos negócios que estruturam seu processo de compra e venda na
Internet. Assim, todas as transações comerciais são realizadas por meio de ferramentas online.

Dessa forma, fica fácil entender que o conceito de e-commerce envolve muito mais do que apenas a criação de um site. Trata-se de um tipo de empreendimento que se diferencia pela e sua estrutura de funcionamento — altamente relacionada ao digital.

Dessa maneira, ele facilita e agiliza o trabalho de gestão em muitas frentes. Por outro lado, também tem como efeito o maior peso estratégico da questão da logística. Daniel Sampaio(2019).

2.3 FERRAMENTAS DA LOJA

O sistema apresentará os dados das peças escolhidas e os dados do usuário, esta primeira versão será monolítica e desenvolvida para servidor que terá como função mostrar na pratica ao cliente como funcionará a aplicação e suas características de forma em que o cliente possa avaliar melhorias e implanta-las nas versões seguintes como, por exemplo, melhoria de infraestrutura migrando a aplicação para a nuvem e melhoria de código migrando a aplicação para uma funcionalidade que são altamente valorizadas e aprimoradas.

3 METODOLOGIA

Este capítulo descreve como o trabalho foi realizado, incluindo o ambiente e as tecnologias utilizadas (cap 4), como o sistema foi construído, quais telas foram criadas e as etapas de desenvolvimento.

4 DESENVOLVIMENTO

Durante o desenvolvimento deste produto, utilizei o padrão *MVC*no *springboot* para uma estrutura organizada. Cada parte do código é empacotada de acordo com o exemplo do *Clean code Book*. Para facilitar a leitura e compreensão.

O *Springboot* é usado para otimizar seu tempo e aumentar a produtividade, pois simplifica o desenvolvimento de aplicativos. Dito isto, tendo que gastar tempo desenvolvendo o aplicativo do zero, você já tem a maioria das funcionalidades de que precisa.

E dessas vantagens é permitir características responsivas, oferecendo aplicações seguras com elementos completos. Assim, o Spring *MVC*funções administrativas de transações e a garantia da segurança. Entregando soluções seguras para os clientes, preservando formulários e cadastros.

O *React* foi utilizado neste projeto por ser uma grande vantagem, resumido em bibliotecas e ferramentas simples e bem definidas necessárias para a aplicação. Nesse sentido, é bom para o desenvolvimento, o *React* é baseado em componentes. A ideia é criar componentes independentes menores e mais simples e combiná-los para criar *UIs* mais complexas.

Para salvar dados de clientes e usuários, utilizamos o sistema gerenciador de banco de dados relacional PostgreSQL, que é um dos sistemas mais versáteis para os usuários. Principalmente porque é um sistema aberto e gratuito.

Existe um processo servidor que lê e grava os arquivos de banco de dados reais e um conjunto de programas clientes que se comunicam com o servidor.

TELA DE BOAS VINDAS

Figura 2- Tela de Boas Vindas
Fonte: Próprio autor (2022)

AUTENTICAÇÃO DO USUÁRIO / TELA DE LOGIN

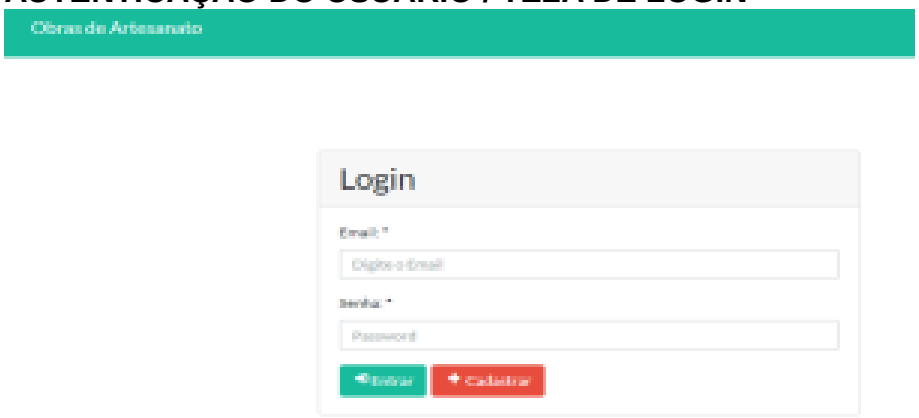


Figura 3- Autenticação do Usuário / Tela de Login
Fonte: Próprio autor (2022)

```
entrar = () => {
    this.service.autenticar({
        email: this.state.email,
        senha: this.state.senha
    }).then( response => {
        this.context.iniciarSessao(response.data)
        this.props.history.push('/home')
    }).catch( erro => {
        mensagemErro(erro.response.data)
    })
}
```

```
@PostMapping("/autenticar")
public ResponseEntity<?> autenticar( @RequestBody UsuarioDTO dto ) {
    try {
        Usuario usuarioAutenticado = service.autenticar(dto.getEmail(), dto.getSenha());
        String token = jwtService.gerarToken(usuarioAutenticado);
        TokenDTO tokenDTO = new TokenDTO( usuarioAutenticado.getNome(), token);
        return ResponseEntity.ok(tokenDTO);
    }catch (ErroAutenticacao e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
```

Estes trechos de códigos são usados para autenticar o usuário na parte do aplicativo, fazendo o login usando e-mail e senha. Codifica a senha inserida como criptografada para segurança do usuário.

TELA DE CADASTRO DE USUÁRIOS

Figura 6- Tela de Cadastro de Usuário
Fonte: Próprio autor (2022)

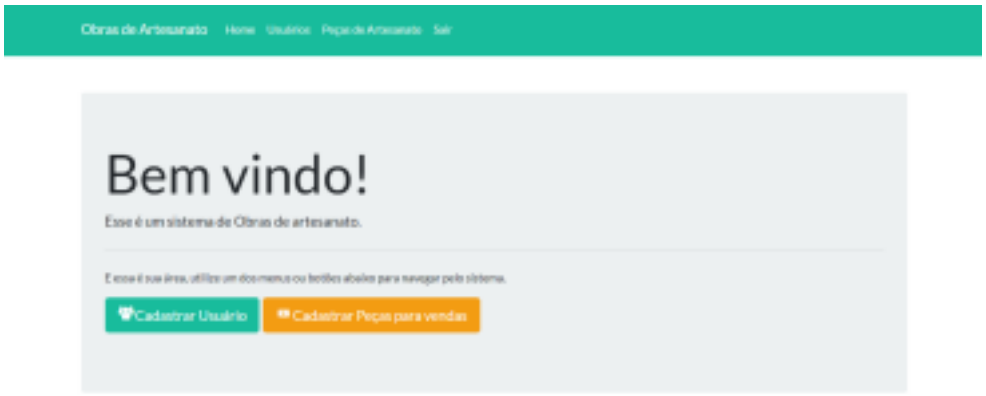
```
cadastrar = () => {

    const {nome, email, cpf, celular, senha, senhaRepeticao } = this.state
    const usuario = {nome, email, cpf, celular, senha, senhaRepeticao }

    try{
        this.service.validar(usuario);
    }catch(erro){
        const msgs = erro.mensagens;
        msgs.forEach(msg => mensagemErro(msg));
        return false;
    }

    this.service.salvar(usuario)
        .then( response => {
            mensagemSucesso('Usuário cadastrado com sucesso! Faça o login para acessar o sistema.')
            this.props.history.push('/login')
        }).catch(error => {
            mensagemErro(error.response.data)
        })
}
```

Figura 7- Tela de Cadastro de Usuário
Fonte: Próprio autor (2022)

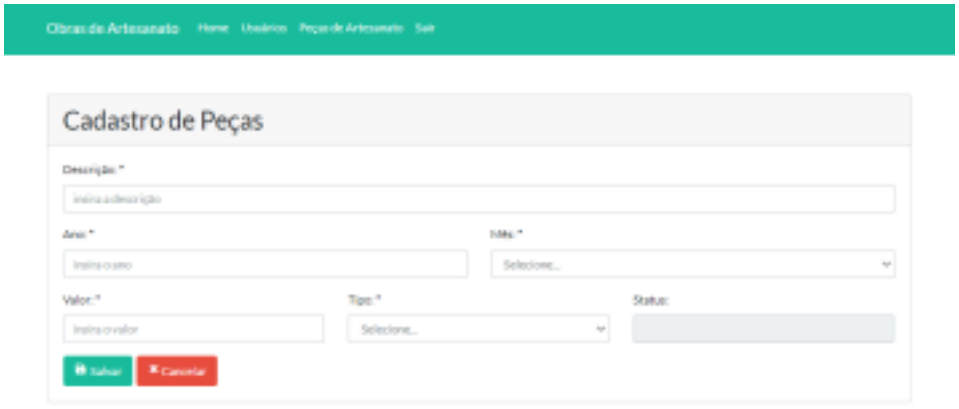
```
@PostMapping()
public ResponseEntity<?>salvar( @RequestBody UsuarioDTO dto ) {

    Usuario usuario = Usuario.builder()
                .nome(dto.getNome())
                .email(dto.getEmail())
                .cpf(dto.getCpf())
                .celular(dto.getCelular())
                .senha(dto.getSenha()).build();

    try {
        Usuario usuarioSalvo = service.salvarUsuario(usuario);
        return new ResponseEntity<>(usuarioSalvo, HttpStatus.CREATED);
    }catch (RegraNegocioException e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
```

O código foi implementado neste trecho para cadastrar o usuário, ter acesso ao aplicativo e completar o login.

TELA DE DENTRO DA APLICAÇÃO DE PEÇAS



TELA DE CADASTRO DE PEÇAS DE ARTESANATO



```java
@PostMapping
public ResponseEntity<?> salvar( @RequestBody PecasDTO dto ) {
    try {
        Pecas peca = converter(dto);
        peca = service.salvar(peca);
        return new ResponseEntity<>(peca, HttpStatus.CREATED);
    }catch (RegraNegocioException e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
```

Nessa parte do código foi desenvolvido onde a peça de artesanato será cadastrada com as especificações fornecidas. Mostrando detalhadamente, os campos que está sendo validado.

TELA ATUALIZAR PEÇAS DE ARTESANATO

Figura 14- Tela de Atualizar Peças
Fonte: Próprio autor (2022)

Figura 15- Tela de Atualizar Peças
Fonte Próprio autor (2022)

A aplicação de peças de artesanato poderá alterando valores, descrição, mês, ano, e estado de pagamento. Nesse trecho de código colocando validações de erros e etc.

5 CONSIDERAÇÕES FINAIS

Este estudo tem como objetivo discutir a relevância do sistema de captação de *LEADS* para a venda de artesanato a clientes e artistas da região. Nesse caso, entende-se que no mercado de artesanato atual, eles precisam ter um sistema de identificação da arte do artista para prestar aos seus clientes um melhor atendimento. Portanto, este estudo constata a importância de um sistema eficaz a fim de coletar dados dos clientes para construir um melhor relacionamento com os clientes para fechar a venda de artesanato.

Nesse interim, no cenário atual foi afetado pela pandemia, demanda de serviços, dessa maneira entende-se que é de fundamental importância o investimento por parte do estado oferecer meios para recuperar as vendas perdidas. E potencializar as vendas através do sistema, fazendo com que aumente o faturamento no exercício.

Assim nosso sistema mostra uma tela de boas-vindas mostrando nosso *login* para acessar a obra, para que os usuários interessados na obra se cadastrem para entrar no sistema e visualizar a tela do artefato. Em troca, nossa empresa adquirirá dados de clientes.

A construção de melhores relacionamentos com os clientes é considerada de extrema importância para o fechamento anual de vendas e
otimização do desempenho, por isso entende-se que o sistema estará bem adequado para fornecer esta solução para vendas de peças. Portanto, conclui-se que nosso sistema é muito importante para otimizar o relacionamento com os clientes, proporcionando trabalhos de crochê de forma clara e objetiva, prestando o melhor

atendimento para que os clientes adquiram e garantindo que seus dados estejam totalmente seguros, por isso é muito importante para otimizar o campo de artesanato em crochê Ótima ferramenta para venda.

REFERÊNCIAS

AFONSO, Alexandre. **O que é Spring Boot?** 2019. Disponível em: https://blog.algaworks.com/spring-boot/. Acesso em: 02 fev. 2017.

BRASIL, Rede Artesanato. **Como a pandemia de Covid-19 afetou o artesanato brasileiro**. 2021. Disponível em: https://redeartesanatobrasil.com.br/2021/07/16/pandemia-da-covid-19/. Acesso em: 16 jun. 2021.

DINO. **Mercado de artesanato movimenta 50 bilhões por ano**. 2020. Disponível em: https://www.mundodomarketing.com.br/noticias corporativas/conteudo/245025/mercado-de-artesanato-movimenta-50-bilhoes-por-ano. Acesso em: 18 set. 2020.

Guia do Usuário do PostgreSQL 7.3.4 1996-2002. Disponível em: http://tecspace.com.br/paginas/aula/postgresql/guia/index.html. Acesso em: 15 maio. 2022.

LAUBE, Klaus Peter. **Engatinhando em Java para a web: Spring Boot**. 2020. Disponível em: https://klauslaube.com.br/2020/05/20/engatinhando-em-java-web spring-boot.html. Acesso em: 20 maio 2020.

MORAIS, Patrick. **A História do React!** 2021. Disponível em: https://medium.com/@ppternunes/a-história-do-react-ba346c416fe1. Acesso em: 17 ago. 2021.

NOTICIAS, Asn Agencia Sebrae de. **Artesanato aposta no mercado on-line para divulgação e venda de produtos**. 2022. Disponível em: https://www.agenciasebrae.com.br/sites/asn/uf/NA/artesanato-aposta-no-mercado-online-para-divulgacao-e-venda-de produtos,4582b281aa89f710VgnVCM100000d701210aRCRD. Acesso em: 17 mar. 2022.

SALVADOR, Giovanni. **Linguagens e tecnologias em alta para criar um produto digital pós MVP**. 2018. Disponível em: https://startupi.com.br/2018/07/linguagens-e tecnologias-em-alta-para-criar-um-produto-digital-pos-mvp/. Acesso em: 24 jul. 2018.

SAMPAIO, Daniel. **O que é E-commerce? Tudo o que você precisa saber para ter uma loja virtual de sucesso!** 2019. Disponível em: https://rockcontent.com/br/blog/ecommerce-guia/. Acesso em: 09 out. 2019.

PROPOSTA DE SOFTWARE DE VENDA DE COSMÉTICOS E PRODUTOS FEMININOS

DE LIMA, UDSON WILLAMS RÊGO
SOUSA, MARCELO FERNANDES DE

RESUMO

O presente trabalho tem o desejo de fazer a representação de um modelo de *ecommerce* para uma loja de venda de cosméticos e demonstrar que esse modelo de negócio chegou para qualquer tipo de negócio, seja ele de tecnologia, móveis, peças automotivas, ou produtos de beleza, podemos concluir que o mercado online vem em uma grande crescente de usuários e que a tendência é que isso não diminui, depois do impacto que a pandemia nos trouxe temos cada vez mais usuários nesse tipo de venda online, foi pensando nisso que esse projeto foi criado e iniciado para desenvolvimento de uma loja que possa atender seus clientes bem, vendendo seus produtos com segurança e rapidez, o projeto foi pensado para ser uma API, recebendo chamadas por meio de requisições os métodos do protocolo *HTTP, utilizamos as* tecnologias que temos mais familiaridade, então escolhemos utilizar a linguagem python e um de seus mais famosos frameworks, o *Django*, utilizando o banco de dados padrão do *Django* o *SQlite.*

Palavras chave: *ecommerce;* cosméticos*; API; python; django,*

1 INTRODUÇÃO

O modelo de *web commerce* no brasil vem crescendo a cada dia, não a nada mais normal do que fazermos compras através da internet hoje em dia, podendo ser qualquer produto, seja produtos domésticos, tecnologia, calçados, basicamente qualquer área pode ter seu produto a venda online, e produtos de beleza não seriam diferentes. Lojas de qualquer segmento tem que se movimentarem a cada dia para se transformarem e oferecerem tanto um serviço presencial como vendas onlines, partindo desse pressuposto, resolvi elaborar um sistema de *e-commerce* de uma loja de produtos de beleza que visa poder oferecer a venda seus produtos online, tentando obter um aumento de pelo menos 25% de lucro num período de 6 meses depois da entrega do projeto. além de poder manter um controle de estoque dos produtos e também conseguir analisar dados de cadastro de clientes mensais e de compras, utilizando a linguagem de programação *python* que é uma das mais utilizadas no mercado hoje, junto ao *framework* django conseguimos criar um sistema robusto baseados em *APIs Rest* que tem como função controlar o sistema por meio de requisições http e que se bem utilizado conseguirá alavancar as vendas da loja.

além de utilizar um banco de dados baseado em *SQL* para salvar os dados da loja.

2 FUNDAMENTAÇÃO TEÓRICA

O comércio de produtos é algo que já existe desde os primórdios do homem, a partir do momento que percebemos que poderíamos trocar mantimentos com outras pessoas ou outros grupos de pessoas o conceito de mercado nasceu, na ocorrência de precisar de algum tipo de recurso e não tê-lo, ô obter por meio de trocas foi oque deu base para o nosso comércio como conhecemos hoje com a troca de dinheiro por recursos, sejam eles comida, roupas, moradias, existindo um comércio para basicamente tudo no mundo hoje em dia.

2.1 O comércio eletrônico no contexto brasileiro

Por mais que o conceito de mercado seja um modelo já bem antigo, o formato de venda online já é algo bem mais novo. Obviamente por conta da questão da criação da internet, computadores pessoais em suma são relativamente novos. Também se entra no mérito das pessoas não terem uma total visão do poder que a internet poderia ter no mundo globalmente, Gigante hoje em dia a Amazon que é uma das líderes do mercado em vendas por meio de e-commerce teve um pensamento diferente das pessoas na época e por isso se destacaram tanto num mundo que a internet se tornou algo essencial.

> Dados da E-bit mostram que o faturamento do setor com vendas de bens de consumo foi de R$35,8 bilhões. O que representa um crescimento de 24% em relação ao ano de 2014, quando se vendeu um total de R$28,8 bilhões. O comércio eletrônico vem conquistando cada vez mais clientes, pelo simples fato das compras serem feitas de qualquer lugar, incluindo sua casa, trabalho, uma comodidade.(ANDRADE, 2017, p. 3)

Por problemas socioeconômicos em geral o Brasil tem uma maior diferença de tempo em como algumas tecnologias chegam até nós, então também tivemos uma certa demora para esse tipo de comércio ser viável em nosso país até por conta de outros fatores como a falta de internet em muitas áreas e até mesmo demograficamente

O Brasil é o quinto maior país do mundo em extensão territorial, o país ocupa uma área de 8.547.403 km² no planeta Terra. (IBGE**,** Acessado em 2022, *online*)

já que somos um país de grande extensão territorial, mas conseguimos chegar

a movimentar cerca de R$ 35 bilhões de reais em 1 ano apenas com o comércio online oque com certeza é algo muito grande, a facilidade que a internet trouxe nos possibilitou fazer diversas coisas, em geral fazer compras sempre foi algo que a humanidade gostou, e poder fazer isso sentado enquanto vê seus programas favoritos na TV sem a necessidade de se deslocar favoreceu muito para o modelo de e-commerce ser um sucesso e movimentar tanto o ramo comercial.

2.2 Comércio eletrônico do país sob a ótica do marketing digital nas redes sociais virtuais

Em nosso país as redes sociais são algo realmente bastante utilizadas, e nada mais normal do que tentar conseguir dinheiro se utilizando dessas plataformas, seja por meio de divulgação, seja por meio de vendas, como é algo que tem muita demanda, hoje em dia as lojas conseguem usar essas plataformas para divulgar seus produtos, concluir vendas, fazer *merchandising,* entre outras coisas. pela alta demanda se torna impossível a venda online utilizando apenas desse meio, daí entra a vantagem do *e commerce*, mantendo a praticidade pro cliente, mas com um toque mais autômato, dando flexibilidade para o vendedor.

“Os resultados do estudo indicam que, diante das atividades exercidas pelas lojas no Facebook, as imagens aparecem como a ferramenta mais utilizada, seguida dos links. Quanto ao conteúdo das postagens, nota-se que estas variam de acordo com o foco das empresas. No que tange ao engajamento do usuário com a loja, observou-se que as taxas de curtidas são superiores às de comentários, o que pode indicar que os usuários buscam atividades que exijam menos esforço de sua parte quando utilizam as redes sociais. Diante disso, conclui-se que a variável “fãs” não representa ser um fator determinante para a criação de engajamento, mas sim, a comunicação e a interação com os usuários detêm relação direta no retorno das fanpages.” (VISENTINI, 2018, P.1)

Essa pesquisa mostra bastante como o engajamento nas redes sociais pode ser usado para demonstrar produtos de forma mais natural para os clientes nas redes sociais oque gera mais vendas, já que o cliente se sente mais próximo do produto, oque influencia nas vendas já que é possível vincular esses produtos a algum link de *e commerce* onde os clientes podem finalizar suas compras e além de ser uma forma mais fácil de divulgação, seja do nome da empresa, seja de produtos ou serviços.

2.3 *Python* - linguagem de programação

Python foi a linguagem escolhida para a criação do projeto, pela simplicidade, rapidez e segurança que a linguagem proporciona. Foi criada na década de 90 pelo Holandes Guido van Rossum. É uma linguagem multiparadigma e que prioriza a legibilidade de código acima de tudo, sendo uma das suas principais características, funcionar a partir de código identado, ou seja a criação de blocos funciona por meio de indentação, não precisando de {}(chaves) ou outras formas para a sua criação

Python é uma linguagem fácil de aprender e poderosa. Ela tem estruturas de dados de alto nível eficientes e uma abordagem simples mas efetiva de programação orientada a objetos. A elegância de sintaxe e a tipagem dinâmica de Python aliadas com sua natureza interpretativa, o fazem a linguagem ideal para programas e desenvolvimento de aplicações rápidas em diversas áreas e na maioria das plataformas.(*Python*, Acessado em 2022, *online*)

Uma das linguagens mais utilizadas no ano de 2021, python com certeza chama a atenção por conta de sua facilidade, além de desenvolver sistemas complexos sem nenhum problema, oque dá mais leveza na hora de trabalhar e escrever codigo.

2.3.1 *Django*

Depois de escolher a linguagem de programação, o *framework* escolhido foi o Django, que é um framework para um rápido desenvolvimento web, ele funciona com um padrão chamado MTV ou *Model, Template, View.* Foi criado para administrar um site jornalístico no Kansas,EUA e a partir daí, se tornou um projeto de código aberto, ou seja, qualquer pessoa pode ir e gerar sua própria versão, a partir de uma já definida, ou apenas ajudar na construção do *framework.*

Django fornece uma maneira segura de gerenciar as contas dos usuários e suas senhas, evitando erros comuns, tais como colocar informações da sessão em cookies, onde ficam vulneráveis (ao invés disso os cookies contém apenas uma chave e os dados são armazenados no banco de dados), ou armazenar as senhas de forma direta, ao invés de gravar um hash para essas senhas.(INTRODUÇÃO AO DJANGO, Acessado em 2022, online)

*P*or ter uma facilidade na hora de criar modelos de banco de dados, facilmente manipular arquivos estáticos com o Django templates e já ter acoplado a sua versão inicial segurança robusta, foi o *framework python* escolhido para a criação do *e commerce.*

2.3 Banco de dados *SQL*

Originalmente o Django utiliza do Banco *SQlite* para salvar suas configurações e salvar dados de seus modelos, sendo possível alterar facilmente o banco de dados utilizado no projeto.

"*SQLite* é uma biblioteca em processo que implementa um mecanismo de banco de dados SQL transacional independente, sem servidor e sem configuração. O código para *SQLite* é de domínio público e, portanto, é gratuito para uso para qualquer finalidade, comercial ou privada. *SQLite* é o banco de dados mais implantado no mundo com mais aplicativos do que podemos contar, incluindo vários projetos de alto perfil."(SQLITE, Acessado em 2022, online)

No projeto pretendo utilizar o *SQlite*, mas pretendo seguir boas práticas de programação para que ele não seja dependente de banco de dados. Com simples comando no arquivo de configuração do Django é possível mudar o banco com facilidade e sem impacto no projeto.

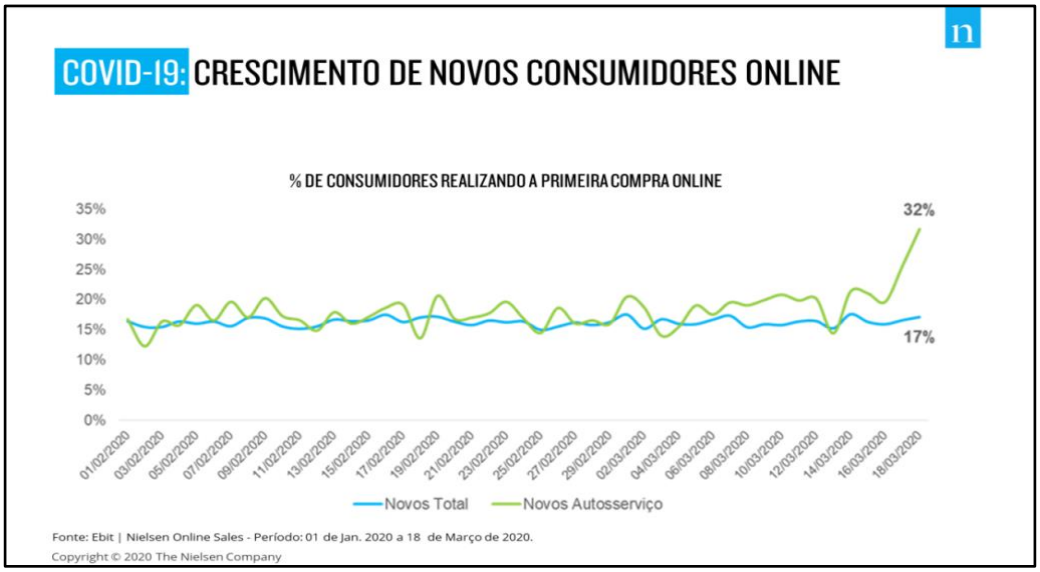
3 METODOLOGIA

O presente trabalho tem o desejo de fazer a representação de um modelo de *ecommerce* para uma loja de venda de cosméticos, produtos de beleza, algo mais focado para um público que consome esse tipo de produto on-line, hoje em dia, logo após a pandemia do novo coronavírus, muitas empresas precisaram se adequar ao formato de vendas online, ou poderia fechar as portas, foi pensando nisso que esse projeto foi criado e iniciado para desenvolvimento de uma pequena loja que possa vender os produtos com segurança e rapidez, garantindo a satisfação dos seus clientes.

3.1 Aumento de novos consumidores online

Mesmo já sendo um mercado em ascensão, após a pandemia de covid-19 no mundo o mercado online teve outro *boom,* sendo o mais seguro não sair de casa, acabou criando uma enorme quantidade de clientes que queriam fazer suas compras mas que não queriam perder sua segurança, é possível ver na (**Figura 1)** um gráfico de porcentagem de aumento de porcentagens o aumento de consumidores realizando a primeira compra online.

Figura 1: Representação por meio de um gráfico da %(percentagem) de primeiras vendas da Nielsen *Online Sales* no período de 01 de janeiro 2020 a 18 de março de 2020



Fonte: Ebit | Nielsen Online Sales(2020)

Com esses dados em mãos, podemos concluir que precisaríamos criar uma opção para o comércio *online* da loja, pois a demanda de utilização seria muito alta.

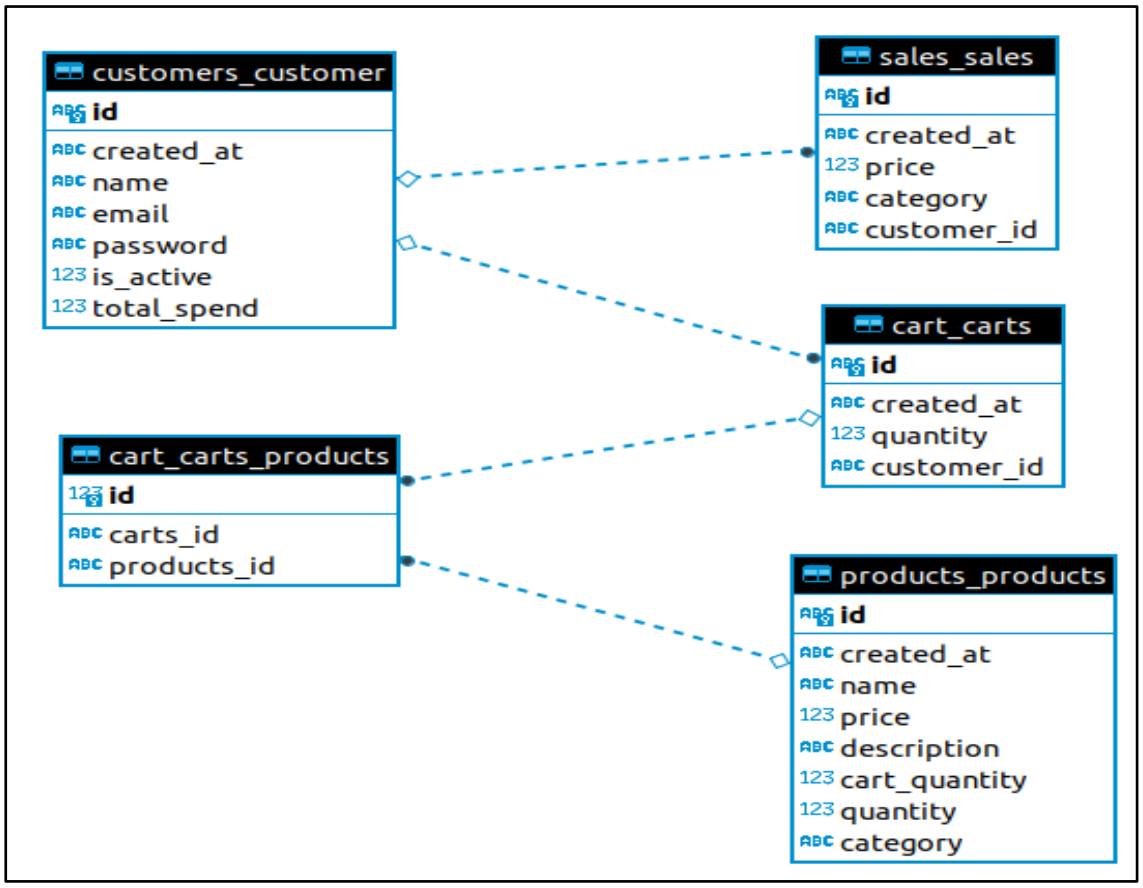
4 FUNDAMENTAÇÃO TEÓRICA

Tentamos desenvolver uma aplicação que seguisse boas práticas, que tivesse segurança e que fosse facilmente manipulada para trazer modificações

4.1 MODELO DE BANCO DE DADOS

Nosso modelo de banco de dados ficou no formato mostrado na **(Figura 2)**, algumas entidades principais como o *Customer/Products* e o *Cart*, e alguns modelos para suprir necessidades de negócio, como estoque, total gasto por um usuário, etc.

Figura 2: Modelagem de banco de dados do *E Commerce*

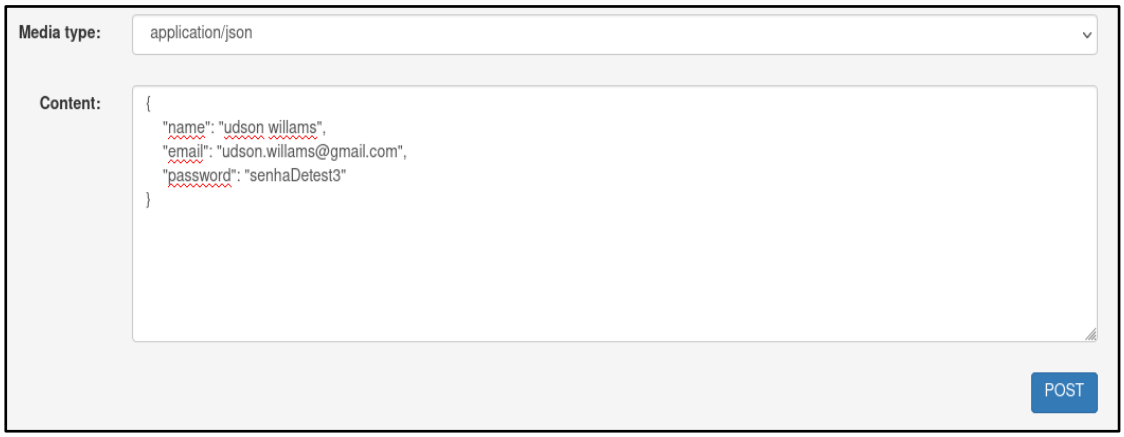


Fonte: Próprio Autor (2022)

4.2 MODELO DE CRIAÇÃO DE USUÁRIO

O modelo mostrado na **(Figura 3)**, demonstra o nosso atual modelo de criação de um usuário da aplicação, a aplicação recebe um *json* com os campos e cria um usuário,e a partir daí ele consegue se logar, e finalizar as suas compras.

Figura 3: Modelo atual de criação de um usuário no nosso banco de dados do *E Commerce*

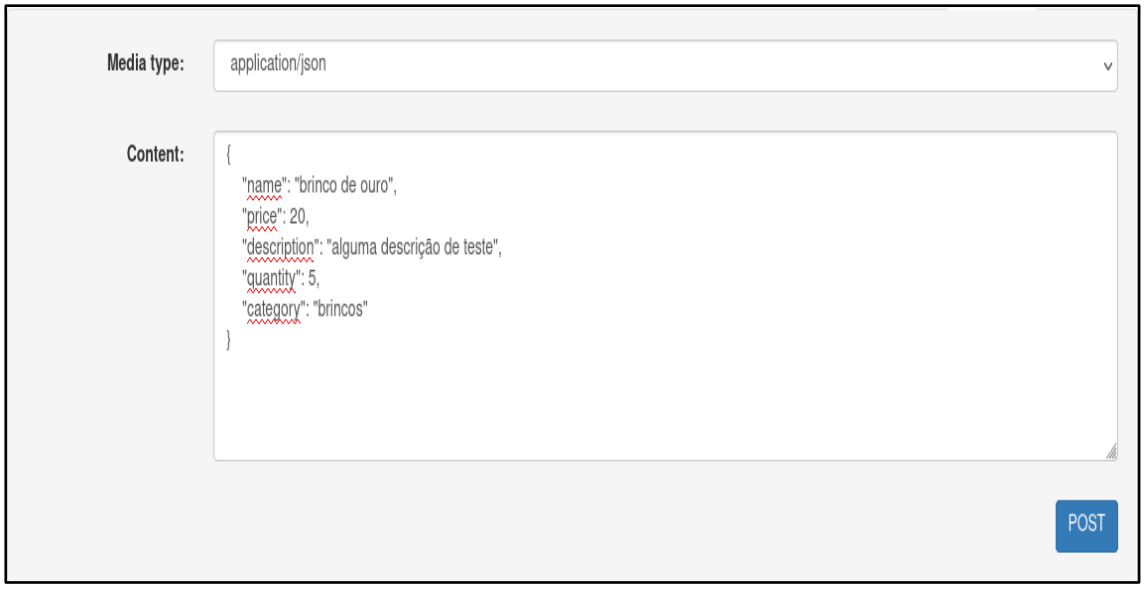


Fonte: Próprio Autor (2022)

4.3 MODELO DE CRIAÇÃO DE PRODUTO

O modelo mostrado na **(Figura 4)** demonstra o nosso atual modelo para a criação de um produto, a aplicação recebe um json com os campos de nome, preço, quantidade, etc e a partir daí ele gera um produto que pode ser adquirido por um usuário

Figura 4: Modelo atual de criação de um produto no nosso banco de dados do *E Commerce*



Fonte: Próprio Autor (2022)

4.4 MODELO DE CRIAÇÃO DE CARRINHO

O modelo mostrado na **(Figura 5)** demonstra o nosso atual modelo para a criação de um carrinho, a aplicação recebe um json com os campos de usuário e produtos, os campos são preenchidos com o id do usuário que está criando o carrinho, e o id dos produtos que ele escolheu pro carrinho.

Figura 5: Modelo atual de criação de um produto no nosso banco de dados do *E Commerce*

Fonte: Próprio Autor (2022)

5 CONSIDERAÇÕES FINAIS

Finalizando o projeto prático percebemos que as ferramentas escolhidas ajudaram muito na facilidade do desenvolvimento prático do produto, O django por exemplo, além de trazer um banco de dados integrado que deixou o desenvolvimento bem fácil, traz um grande ORM que facilita a modificação desses dados, seja por meio do admin, seja por meio das requisições http que foram criadas.

Tivemos alguns desafios para criar os módulos de login de usuários, o django já possui isso nativamente, mas apenas pros usuários django default, oque não é uma boa prática de se utilizar para criação de usuários de um projeto aonde se quer segurança, escalabilidade, etc.

O formato atual do projeto pode ser acessado por meio da plataforma github no link: https://github.com/UdsonWillams/MyMake_tcc_project

REFERÊNCIAS

ANDRADE, M. C.; SILVA, N. G. **O COMÉRCIO ELETRÔNICO (E-COMMERCE): UM ESTUDO COM CONSUMIDORES.** Disponível em: https://periodicos.ufpb.br/ojs2/index.php/pgc/article/view/26895. Acesso em: 23 maio. 2022.

VISENTINI, M. S.; SCHEID, L. L. CHAGAS, F. B. **ANÁLISE DAS PRINCIPAIS LOJAS DE COMÉRCIO ELETRÔNICO DO PAÍS SOB A ÓTICA DO MARKETING DIGITAL NAS REDES SOCIAIS VIRTUAIS,** Disponível em: https://periodicos.ufpb.br/ojs2/index.php/pgc/article/view/32972 Acesso em: 02 maio. 2022.

IBGE. ***O TAMANHO DO BRASIL.*** *2020. Disponivel em:* https://cnae.ibge.gov.br/en/component/content/article/97-7a12/7a12-voce-sabia/curiosidades/1629-o-tamanho-do-brasil.html. acessado em: 16 de abril de 2022

SQLITE. ***ABOUT SQLITE.*** *2009. Disponivel em:* https://www.sqlite.org/about.html. acessado em: 04 de abril de 2022

PYTHON. ***O TUTORIAL DE PYTHON.*** *2010. Disponivel em:* https://docs.python.org/pt-br/3/tutorial/. acessado em: 15 de maio de 2022

INTRODUÇÃO AO DJANGO. ***INTRODUÇÃO AO DJANGO.*** *2010. Disponivel em:* https://developer.mozilla.org/pt-BR/docs/learn/server-side/django/introduction. acessado em: 15 de maio de 2022

PROJETO E DESENVOLVIMENTO DE UMA APIPARA O GERENCIAMENTO DE SERVIÇOS DE UM PROFISSIONAL FREELANCER

PATRÍCIO, Diego Juliano
SOUSA, Marcelo Fernandes de

RESUMO

Em razão das dificuldades enfrentadas por profissionais *freelancers* em gerenciar suas prestações de serviços, este trabalho visa apresentar o projeto de implementação e desenvolvimento de um sistema de gerenciamento de serviços que tem como ideia principal auxiliar as tarefas, pagamentos e comunicação do profissional com o cliente. A API foi desenvolvida utilizando a linguagem de programação Java, Framework Spring, Banco de Dados MySQL e H2, dentre outras tecnologias. No mercado atual existem diversas aplicações para profissionais freelancers. Todavia, estes serviços oferecem uma competitividade muito alta para os profissionais, além de que são cobradas taxas pela prestação de serviço e há um prazo longo para o recebimento do pagamento. A aplicação apresentada neste trabalho permite um ambiente mais focado para os serviços de um único profissional, dando-lhe um melhor controle das tarefas e oferecendo um ambiente de fidelização para o cliente.

Palavras-chave: API; Freelancer; Sistema de Gerenciamento de Serviços.

1 INTRODUÇÃO

O gerenciamento de serviços é um dos problemas mais comuns quando nos deparamos com serviços *freelancer*. Muitos profissionais acabam utilizando os serviços de *e-mails* para realizar o controle das demandas solicitadas, o que ocasiona perda de informação, pois são utilizados tanto no uso pessoal quanto no profissional. Existem no mercado sistemas online que visam cobrir este cenário, porém acabam se tornando uma vitrine com inúmeros profissionais concorrendo entre si. Por conseguinte, a proposta deste projeto é uma solução possível e viável, para suprir a carência dos *freelancers* de ter um ambiente focado apenas nos seus trabalhos, no seu portfólio, permitindo aos mesmos o gerenciamento das prestações de serviços, otimizando assim o atendimento de seus clientes.

A necessidade de aprimorar o cumprimento das tarefas do dia a dia e organizá-las em um ambiente seguro fez surgir o crescimento das aplicações *web*. Esta informatização das tarefas pode ser realizada de maneira *online*, utilizando apenas o acesso à internet e o computador. Segundo Sommerville (2011), antes da web os sistemas de *software* eram executados em computadores locais e acessíveis apenas dentro de uma organização. Com a evolução da internet, mais recursos foram incorporados aos *Browsers, o* que permitiu o acesso do sistema através de um navegador *web*.

O presente projeto visa desenvolver um sistema de gerenciamento de ordem de serviços para *freelancers*. O sistema é uma API desenvolvida na linguagem de programação JAVA, utilizando a arquitetura de micros serviços, padrões de projetos DDD, *Framework Spring*, JPA Hibernate, Maven, H2 e *MySQL* para o banco de dados, que permite o cadastro de clientes, serviços, suas categorias e a emissão da ordem de serviços que serão prestadas, assim com a situação de seu pagamento.

O sistema possibilitará o gerenciamento dos serviços solicitados centralizandoem um único ambiente todos os detalhes necessários, assegurando o armazenamento correto das informações, evitando falhas na comunicação, trazendo uma melhor acessibilidade ao freelancer pois serão acessados via navegador em qualquer lugar.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo deste trabalho é apresentar uma solução que permita suprir as necessidades dos profissionais freelancers em ter um ambiente próprio para apresentar seus serviços e gerir seus clientes e suas tarefas.

No atual cenário existe no mercado diversas aplicações para profissionais freelancers. Todavia, estes serviços como: Workana, 99Freelas, GetNinjas, dentre muitos outros. São plataformas nas quais os profissionais competem uns com os outros, além de que são cobradas taxas pela prestação de serviço e há um prazo longo para o recebimento do pagamento.

A identificação do problema surgiu a partir de conversas informais com os mais diversos freelancers de vários meios de atuação. Para completar, utilizou-se da experiência profissional do autor, que atua como freelancer no ramo de marketing digital e web por cerca de 7 anos e compreende todas essas dificuldades apresentadas, o que

também serviu como motivação para a criação da aplicação.

Para o desenvolvimento deste projeto foram realizados estudos referentes à diversas tecnologias e ferramentas, além de conceitos que envolvem o processo de desenvolvimento de um software.

2.1 Programação Orientada à Objetos

O desenvolvimento deste projeto aplica o paradigma da Programação Orientada a Objetos (POO), uma forma abstrata de representar o mundo real. Um outro tipo de paradigma é a programação estruturada, esta foca nas ações. Já a POO fita seus esforços no objeto, na entidade (DEITEL, 2011).

Neste paradigma são definidos alguns conceitos importantes a quais serão apresentados a seguir:

- Classes: São coleções de objetos semelhantes, determinados por um grupo de atributos e ações (métodos) de natureza correlata (PUGA, RISSETTI, 2004). Como por exemplo, brigadeiro, trufa e brownie ambos podem ser considerados classes herdadas de doces, porém diferenciam-se em alguns atributos como recheio, tamanho, tipo do chocolate, podendo ser classificado como parte de uma classe superior.
- Objeto: São representações, no domínio da solução, de algum conceito abstrato ou concreto que suporta um incidente (evento ou ocorrência) ou uma interação (transação ou contrato) (PUGA, RISSETTI, 2004).
- Herança: Este é um conceito de reutilizar a classe superclasse ou classe pai herdando seus métodos e atributos. Normalmente, uma classe tem derivações quando, além das características (métodos e atributos) da superclasse, a subclasse possui especificações próprias (DEITEL, 2011).
- Polimorfismo: Este possibilita que um objeto admita o comportamento divergente do que foi estabelecido em sua classe, assim dizendo, a habilidade que o objeto tem de adotar várias formas. Este conceito está relacionado ao de herança (PUGA, RISSETTI, 2004).
- Encapsulamento: Também conhecido como ocultamento de informações, corresponde em esconder detalhes internos de uma classe a um objeto

externo. O encapsulamento impede que um programa se torne tão interdependente que uma pequena modificação possa provocar grandes efeitos que se propaguem por todo o sistema (PUGA, RISSETTI, 2004).

- Métodos: São ações que executam tarefas. Eles podem ou não retornar valorese podem ou não receber parâmetros. Regularmente, uma classe possui diversos métodos, que no caso da classe “Atleta” poderiam ser “correr, malhar e descansar" (COAD e YOURDON, 1992).
- Atributos: São as características de um objeto, em outras palavras, a estrutura de dados que vão representar a classe. Os atributos de um objeto representam seu estado, ou seja, o valor de seus atributos em um determinado momento (COAD e YOURDON, 1992).

2.2 Arquitetura de Software

O uso de padrões de arquitetura foi evidenciado através de um livro publicadopor Gamma, em 1995, na qual tinha o propósito de proporcionar padrões de projeto para linguagens orientadas a objetos que estavam recém empregadas no mercado (SOMMERVILLE, 2011). Este trabalho está direcionado no padrão *Domain Driven Design* (DDD) devido a sua estrutura de distribuição de código, conforme imagem 1.

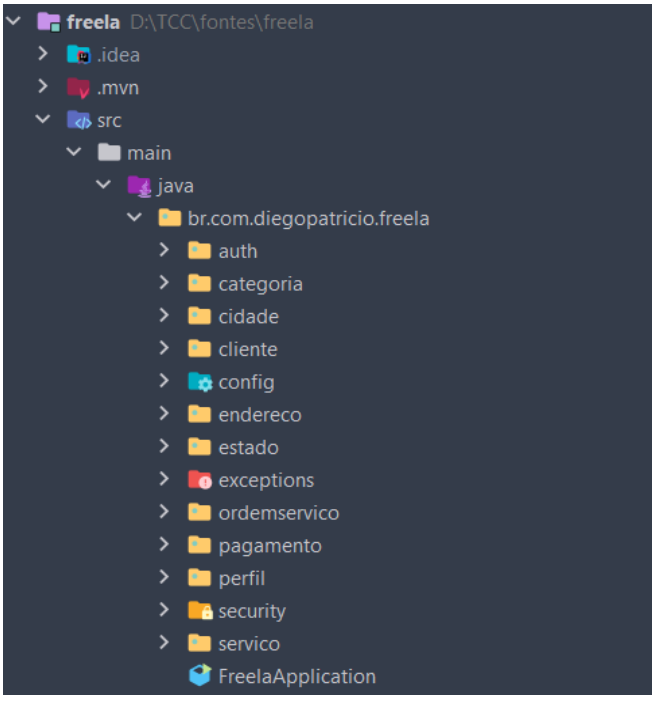


Imagem 1: Estrutura de pacotes do sistemaFonte: Autor (2022).

O padrão DDD é uma técnica de estruturar a aplicação dirigida no que é de fato mais importante, o domínio do negócio (HAYWOOD, 2009). Estimulando os desenvolvedores a estarem presentes no entendimento do negócio para que construam uma linguagem ubíqua (universal) que exprima o conhecimento do time de negócio para um modelo em domínio (para o código).

Também neste projeto foi aplicado o padrão de arquitetura MVC que consiste em desacoplar e facilitar a troca de informações entre a GUI (*Graphical User Interface*) e a manipulação dos dados fazendo com que as respostas sejam mais rápidas e dinâmicas, conforme apresentado na imagem 2.

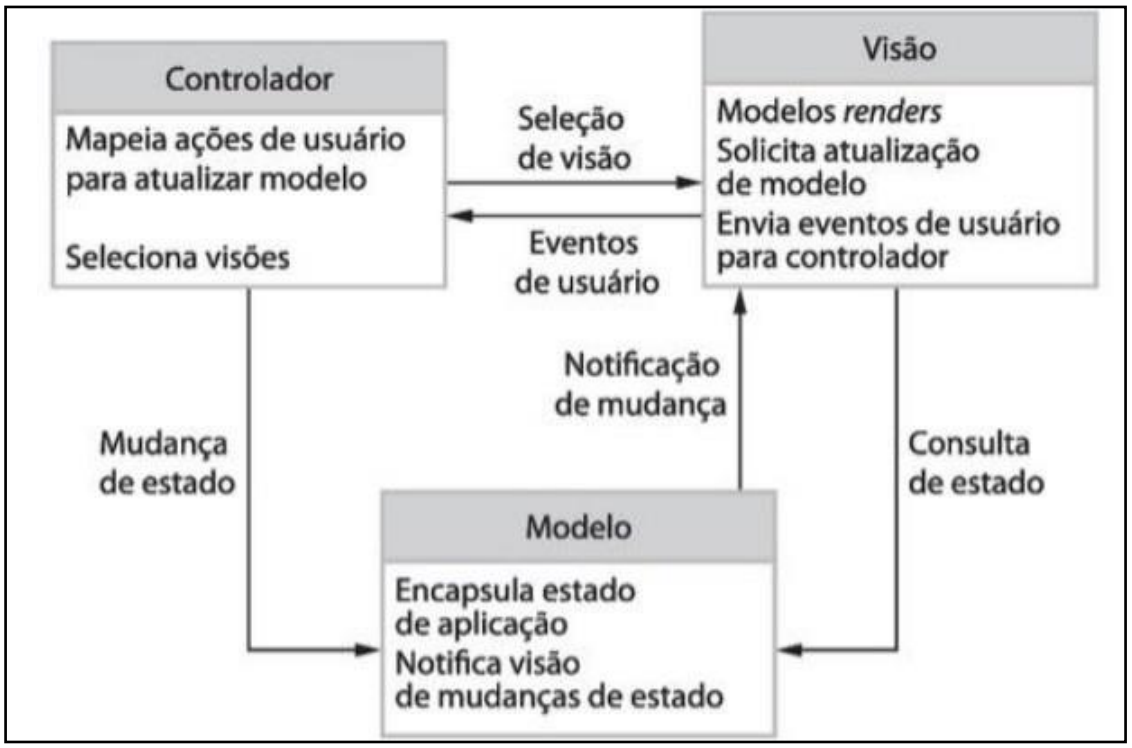


Imagem 2: Diagrama de arquitetura MCVFonte: Sommerville (2011).

Esta divisão é realizada por meio de três componentes com uma atuação direta entre eles: *view, model* e *controller* (SOMMERVILLE, 2011).

2.3 Framework Spring

O Spring proporciona recursos para aplicativos baseados em Java para qualquer tipo de plataforma. Dispõe de diversas tecnologias, que simplificam o desenvolvimento de código, permitindo que os desenvolvedores se concentrem na construção da regra de negócio. O *framework* é organizado em módulos que podem

ser agrupados em seus principais recursos em *Core Container*, *Data Access/Integration, Web, AOP (Aspect Oriented Programming)*, Instrumentação e Teste, conforme apresentado na imagem 3.

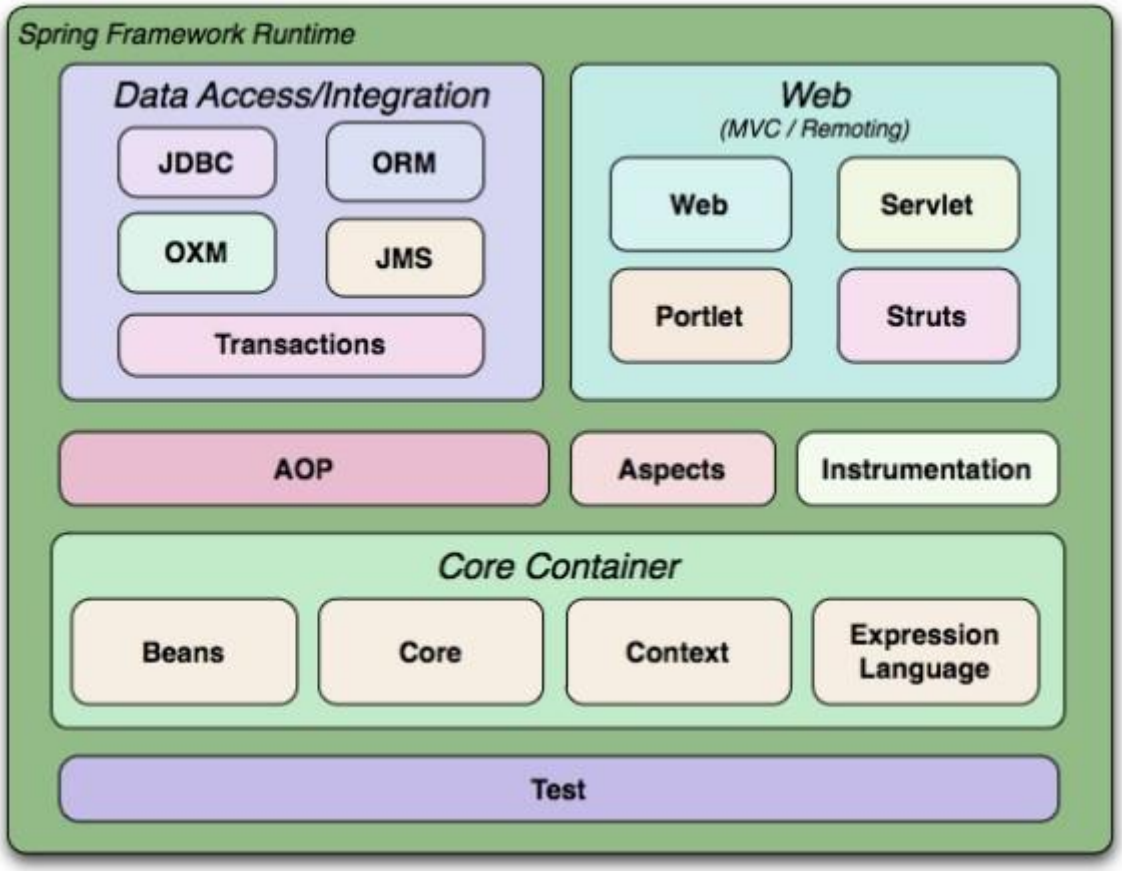


Imagem 3: Diagrama de Organização SpringFonte: Pivotal Software (2018).

O objetivo do projeto apresentado é a construção de uma aplicação *Web* utilizando o módulo *Web MVC* na qual é estabelecido métodos que recebem requisições *HTTP* e retornam respostas em *JSON* através de *API RESTful*. Fazendo uso, também do *Spring Data* para fazer a persistência dos dados no *MySQL*, na qual oferece suporte à *Java Database Connectivity* (JDBC) e *Object Relational Mapping* (ORM) com *Hibernate* e *Java Persistence API* (JPA).

Outra etapa bastante importante do projeto na qual o *spring* irá auxiliar será no desenvolvimento de testes unitários para a verificação das funcionalidades de acordo com os requisitos especificados com o *Spring Test*. O *framework* permite a utilizaçãoda biblioteca *JUnit* e objetos *mocks*, que são implementações de abstrações de contexto e ambiente de execução, inclusive com a injeção de dependência.

Para algumas operações, conforme definido pela regra de negócio, foi requerido que o usuário da requisição apresentasse um selo de autorização, visto quese trataria de dados sensíveis. Para tanto, a aplicação precisaria verificar o usuário erestringir ou não o seu acesso. Como solução foi utilizado o *Spring Security* com o padrão *JSON Web Token* (JWT) que realiza a autenticação e autorização na API.

A JWT é um padrão definido pelo RFC 7519 para transmitir e armazenar de forma compacta e segura objetos JSON entre aplicações. A seguir é apresentado, na imagem 4, a utilização da implementação realizada.

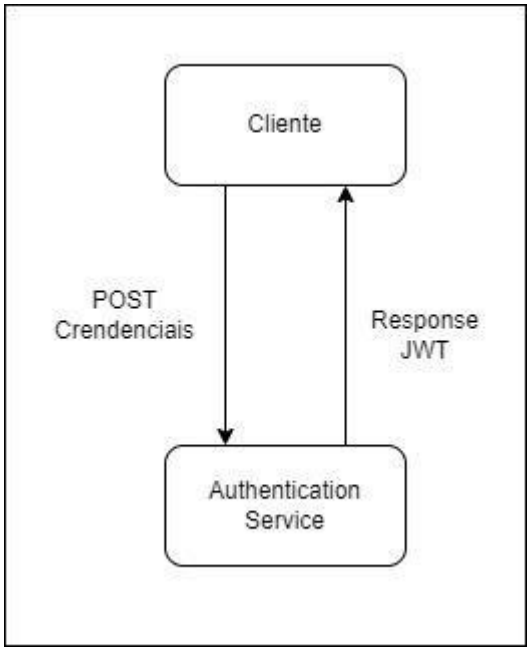


Imagem 4: Diagrama de fluxo JWTFonte: Autor (2022)

Os JSON *Web Tokens* consistem em três camadas separadas por pontos (xxxxx.yyyyy.zzzzz) divididas da seguinte maneira. A primeira parte é o *Header* que consiste no tipo do token e o tipo do algoritmo que está sendo usado, como HMAC SHA256 ou RSA. A segunda parte é o *Payload*, que contém as declarações da entidade tratada. E por último a *Signature* que verifica se a mensagem não foi alteradaao longo do percurso e, no caso de *tokens* assinados com chave privada, também verificar a sua veracidade.

3 METODOLOGIA

A metodologia de desenvolvimento adotada para este trabalho é Análise Orientada a Objeto, por tanto a primeira etapa deste trabalho foi a realização do

diagrama de classe realizada na linguagem UML (Linguagem de Modelagem Unificada), uma linguagem padrão orientada à objetos que auxilia na visualização, construção, especificação e documentação do projeto de software.

3.1 Diagrama de classes

Um diagrama de classes detalha os tipos de objetos da aplicação e os relacionamentos que existem entre eles. As classes representam os objetos centrais em um sistema e é apresentada como um retângulo com 3 compartimentos. O primeiro refere-se ao nome da classe, a do meio os atributos e a inferior o comportamento da classe.

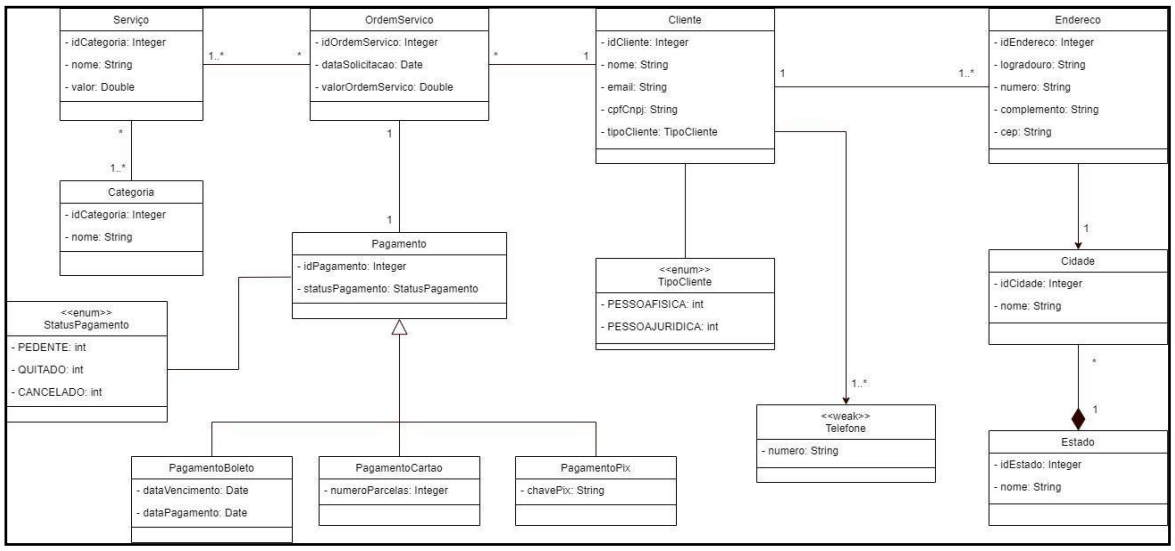


Imagem 5: Diagrama de Classes da API FREELAFonte: Autor (2022)

Na imagem 5, apresentada acima, está representado todo o fluxo de relacionamento que as entidades exercem na aplicação, assim como suas associações.

3.2 Requisitos Funcionais

O Documento de Definição de Requisitos (DDR) tem a finalidade de apresentaro conteúdo dos requisitos funcionais e as regras que serão aplicadas às funcionalidades propostas, assim como seus relacionamentos e dependências. Os requisitos funcionais foram identificados, com base na regra de negócio do sistema de gerenciamento de serviços, para as funcionalidades apresentadas na tabela 1 abaixo.

É considerado como requisito funcional as obrigações da aplicação. Há possibilidade de variações conforme o tipo de produto a ser desenvolvido e quem será a sua audiência. Quando direcionado a usuários, a aplicação deve ter uma abstração maior para que seja possível o entendimento do usuário, e quando escritos para o sistema, devem contemplar todas entradas e saídas possíveis, incluindo suas exceções (SOMMERVILLE, 2011).

ID	Requisito Funcional	Descrição
[RF01]	Manter Serviço	O sistema permite incluir, alterar, listar e excluir informações de serviços como: nome,valor, categoria.
[RF02]	Manter Categoria	O sistema permite incluir, alterar, listar eexcluir categorias.
[RF03]	Manter Cliente	O sistema permite incluir, alterar, listar e excluir informações de clientes como: nome,email, cpf/cnpj, endereço, telefone.
[RF04]	Manter Ordem de Serviço	O sistema permite incluir, atualizar, listar e excluir ordem de pagamento.

Tabela 1: Tabela de requisitos funcionaisFonte: Autor (2022)

3.3 Requisitos Não Funcionais

Os requisitos não-funcionais são aqueles que não se relacionam diretamente com as funções do sistema, no entanto, fazem referências a conceitos como confiabilidade, tempo de resposta, segurança, dentre outras restrições impostas aos serviços ofertados pelo sistema (SOMMERVILLE, 2011).

ID	Requisitos Não Funcionais
[RNF01]	A API deverá ser desenvolvida em JAVA utilizando o framework *Spring Boot*.
[RNF02]	A aplicação deve seguir os princípios de arquitetura de microserviço para garantir eficiências nas manutenções futuras.
[RNF03]	Deve-se utilizar o Spring Security para fornecer proteção no armazenamento de senhas e transferências de dados.

Tabela 2: Tabela de requisitos não funcionaisFonte: Autor (2022)

Conforme apresentado na tabela 2, estão descritos os requisitos não funcionaislevantados para a API FREELA.

3.4 Caso de Uso

O intuito principal de um diagrama de caso de uso é descrever as interações de um usuário com o sistema. A personificação de usuários e funcionalidades são realizadas através de distintos papéis (PRESSMAN, 2016).

Considerando os requisitos descritos anteriormente, é possível modelar as funcionalidades do sistema proposto. Para isso, inicialmente, é apresentado na tabela3 o caso de uso do sistema.

Atores	Clientes
Pré Condições	Cliente Cadastrado
Visão Geral	Este caso de uso consiste no processo de contratação de serviços e fechamento da OS (ordem de serviço) por parte do cliente.
Cenário Principal de Sucesso	
[OUT] O sistema informa os nomes de todas categorias. [IN] O cliente seleciona a categoria e informa o serviço desejado. 3. [OUT] O sistema informa nome e preço dos serviços que se enquadram na pesquisa. [IN] O cliente seleciona um serviço para adicionar a ordem se serviço. 5. [OUT] O sistema exibe a OS com as informações da OS são: nome, serviço contratado, forma de pagamento, status do pagamento e valor total da ordem de serviço. [IN] O cliente informa que deseja finalizar a ordem de serviço.	
Cenários Alternativos	
5. Pagamento Variante 5.1.: Pagamento com boleto [IN] O cliente informa que deseja pagar com boleto. Variante 5.2: Pagamento com cartão [IN] O cliente informa que deseja pagar com cartão e informar a quantidade de parcelas. Variante 5.3: Pagamento com Pix [IN] O cliente informa que deseja pagar com pix e informar a chave para pagamento.	

Tabela 3: Tabela de caso de usoFonte: Autor (2022)

4 FUNDAMENTAÇÃO TEÓRICA

A estrutura da aplicação foi desenhada baseada nos requisitos e casos de uso levantados, como apresentado na documentação em *swagger* (apêndice A), no qual permitiu um maior entendimento do objetivo do negócio e alinhamento entre tecnologias a serem utilizadas. Com um *backend* desacoplado do *frontend* torna possível adicionar qualquer *cliente* sem a necessidade de manutenção ou adição de código, facilitando também a integração entre interfaces.

A aplicação FREELA é uma API *Web RESTful* desenvolvida em Java, com o *framework Spring*, utilizando o paradigma orientado a objetos. O projeto dispõe da ferramenta *Maven* para o gerenciamento de dependências (conforme apresentado no apêndice B) e utiliza para implementar a API *Servlet* de Java o servidor *Tomcat*. As classes controladoras recebem as requisições do *frontend/cliente* que processam a lógica de negócio nas classes de serviços que acessam as classes de repositório, asquais podem manipular o banco de dados, conforme imagem 6.

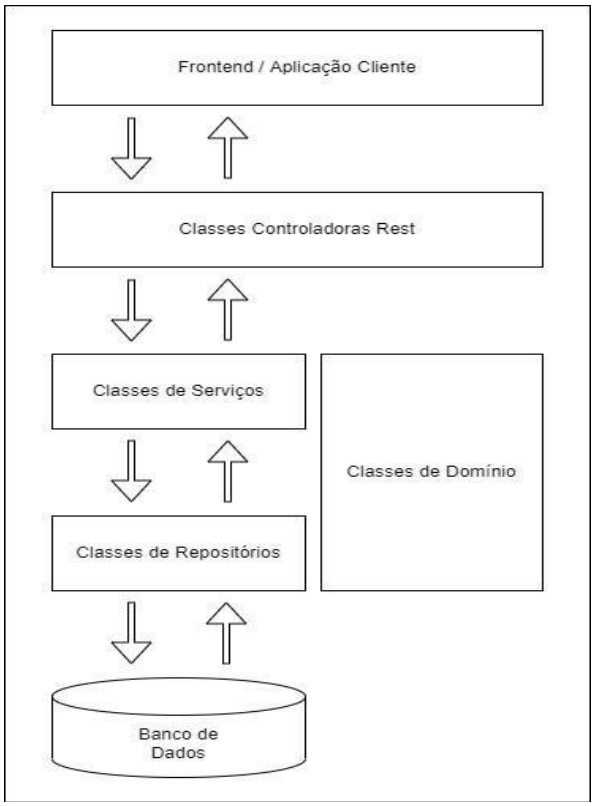


Imagem 6: Diagrama de Aplicação RESTFonte: Autor (2022)

Neste projeto está sendo utilizado o banco de dados MySQL em ambiente de produção e o H2 em ambiente de teste, sendo o acesso a essas bases, gerenciado pelo *Spring Data* e as consultas executadas com o *Hibernate*.

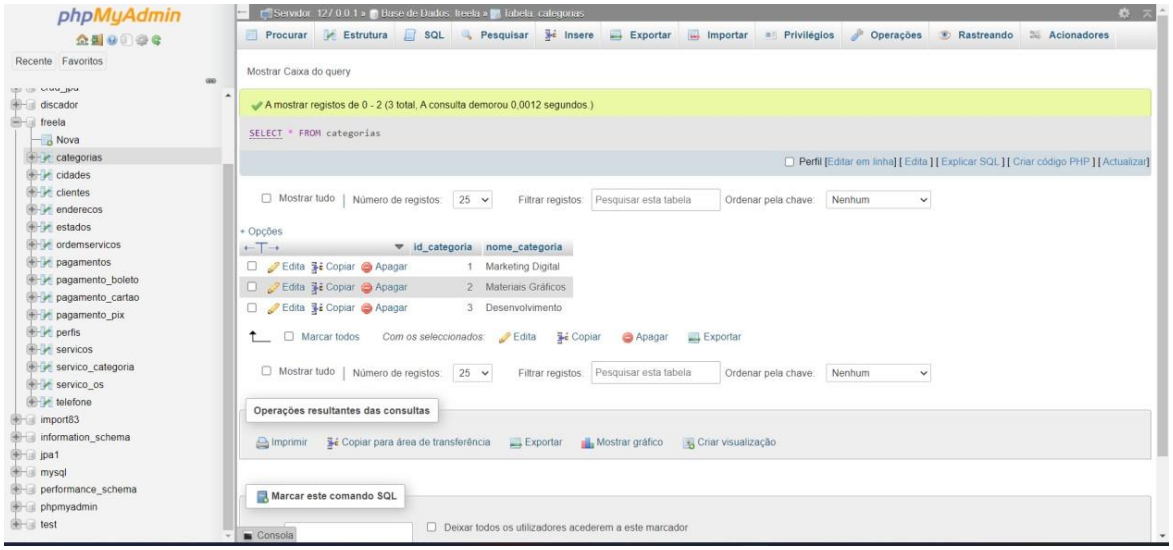


Imagem 7: Banco de Dados MySQLFonte: Autor (2022)

A API é um serviço REST que transfere dados no formato JSON. A comunicação é feita através do protocolo HTTPS através dos métodos GET que busca uma informação na base de dados, POST que insere uma nova informação na base de dados, PUT que faz à atualização em alguma informação já existente na base de dados e DELETE que exclui um registro na base de dados.

Na API foi desenvolvido o CRUD (*Create, Read, Update, Delete*), um acrônimopara as maneiras de se operar em informação armazenada, de categorias, clientes, serviços e ordem de serviço conforme à regra de negócio estabelecida.

Para realizar o armazenamento (*Create*) as requisições são passadas pelo método POST. O *Spring* auxilia neste processo, pois basta apenas utilizar a anotação *@PostMapping* em cima do método desejado e determinar um corpo na requisição através da anotação *@RequestBoy* seguido do tipo do objeto pretendido no parâmetrodo método, conforme na imagem 8.

```
@PostMapping
public ResponseEntity<Void> cadastrarCategoria(@Valid @RequestBody CategoriaDTO categoriaDTO){
    Categoria categoria = service.fromDTO(categoriaDTO);
    categoria = service.cadastrarCategoria(categoria);
    URI uri = ServletUriComponentsBuilder.fromCurrentRequestUri().path("/{id}")
            .buildAndExpand(categoria.getIdCategoria()).toUri();
    return  ResponseEntity.created(uri).build();
}
```

Imagem 8: *Response* do método cadastrar categoriaFonte: Autor (2022)

Para poder resgatar a informação cadastrada é utilizada a anotação *@GetMapping* a qual pode ser atribuída o valor, geralmente a variável id, para buscar um objeto em específico, caso contrário, retornará uma lista com todos os objetos salvos como apresentado na imagem 9.

```
@GetMapping
public ResponseEntity<List<CategoriaDTO>> listarCategorias(){
    List<Categoria> listaCategorias = service.listarCategorias();
    List<CategoriaDTO> listaDTO = listaCategorias.stream().map(CategoriaDTO::new).collect(Collectors.toList());
    return ResponseEntity.ok().body(listaDTO);
}

@GetMapping(value = "/{id}")
public ResponseEntity<Categoria> busarCategoria(@PathVariable Integer id){
    Categoria categoria = service.buscarCategoria(id);
    return ResponseEntity.ok().body(categoria);
}
```

Imagem 9: *Response* do método listar e buscar categoriaFonte: Autor (2022)

Já para poder atualizar o objeto cadastrado é usado o método PUT, *@PutMapping,* que também receberá um valor para identificar o objeto em questão. Neste método pode-se aplicar dois parâmetros: um identificador com a anotação *@PathVariable* e a *@RequestBody* para o conteúdo a qual deseja atualizar, seguindoo modelo da imagem 10.

```
@PutMapping(value = "/{id}")
public ResponseEntity<Void> atualizarCategoria(@Valid @RequestBody CategoriaDTO categoriaDTO, @PathVariable Integer id){
    Categoria categoria = service.fromDTO(categoriaDTO);
    categoria.setIdCategoria(id);
    categoria = service.atualizarCategoria(categoria);
    return ResponseEntity.noContent().build();
}
```

Imagem 10: *Response* do método atualizar categoriaFonte: Autor (2022)

Para o método DELETE, deve-se utilizar a anotação *@DeleteMapping*, atribuindo um valor que se refere ao identificador do objeto e informar este ID com a anotação *@PathVariable* para buscar o recurso alvo, seguindo o modelo da imagem 11.

```
@DeleteMapping(value = "/{id}")
public ResponseEntity<Void> deletarCategoria(@PathVariable Integer id){
    service.deletarCategoria(id);
    return ResponseEntity.noContent().build();
}
```

Imagem 11: *Response* do método deletar categoriaFonte: Autor (2022)

4.1 Executando a aplicação

Conforme regra de negócio estabelecida todo usuário que se cadastrar na API é configurado como cliente, mas é possível alterar seu perfil para administrador caso o profissional freelancer necessite inserir outra pessoa para gerenciar o sistema.

Imagem 12: Cadastrando um clienteFonte: Autor (2022)

Para cadastrar qualquer usuário pode se utilizar do *Frontend* preencher um formulário a qual será consumido pela API FREELA passando as informações no corpo da requisição e será retornado a *URI* com o usuário cadastrado, conforme imagem abaixo.

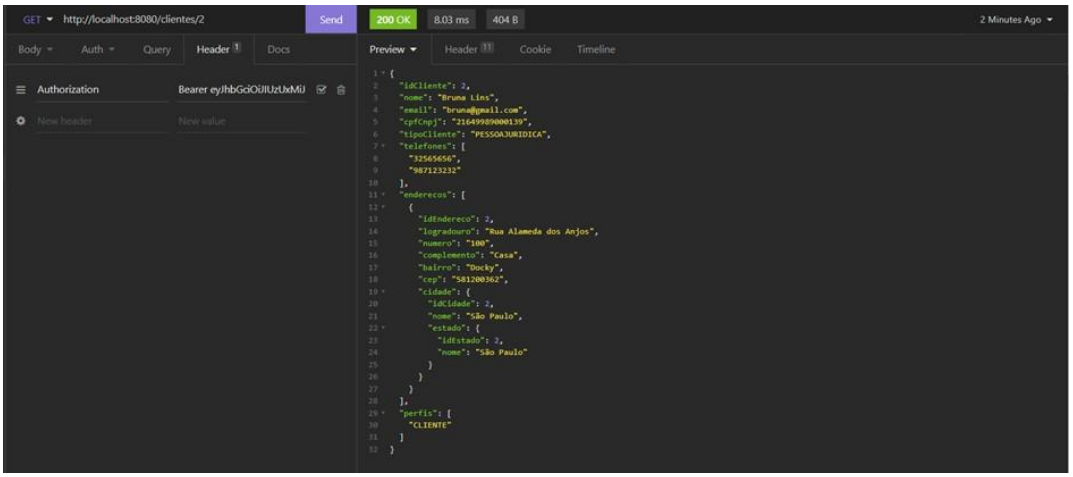


Imagem 13: Resgatando informações de um clienteFonte: Autor (2022)

Para atualizar e buscar um cliente é necessário passar o *ID*. Esta ação será exclusiva do perfil de administrador e do próprio cliente logado no sistema.

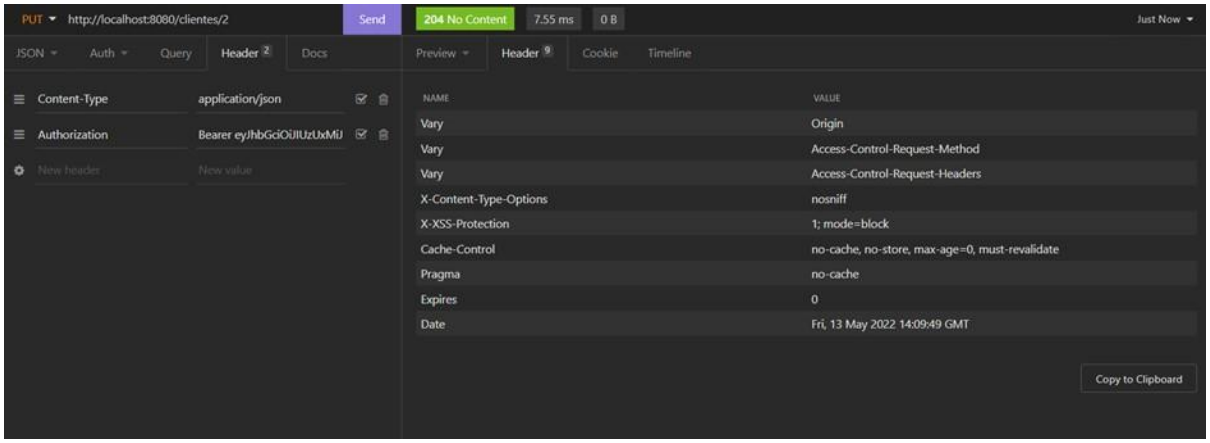


Imagem 14: Atualizando informações de um clienteFonte: Autor (2022)

Já para os métodos de listar e deletar cliente só poderá ser acessado pelo administrador. Para isto deve-se ser efetuado o *login* resgatar o *token* informado e passar como parâmetro no Header das requisições conforme apresentado abaixo.

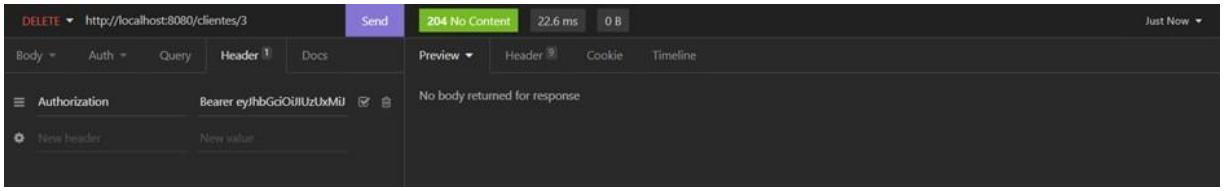


Imagem 15: Deletando um clienteFonte: Autor (2022)

A entidade Categoria e Serviços só poderão ser cadastradas, atualizadas e deletas apenas por um perfil administrador. Mas para os métodos *GET* e listar e buscar é aberta a todos que possuírem a *uri.*

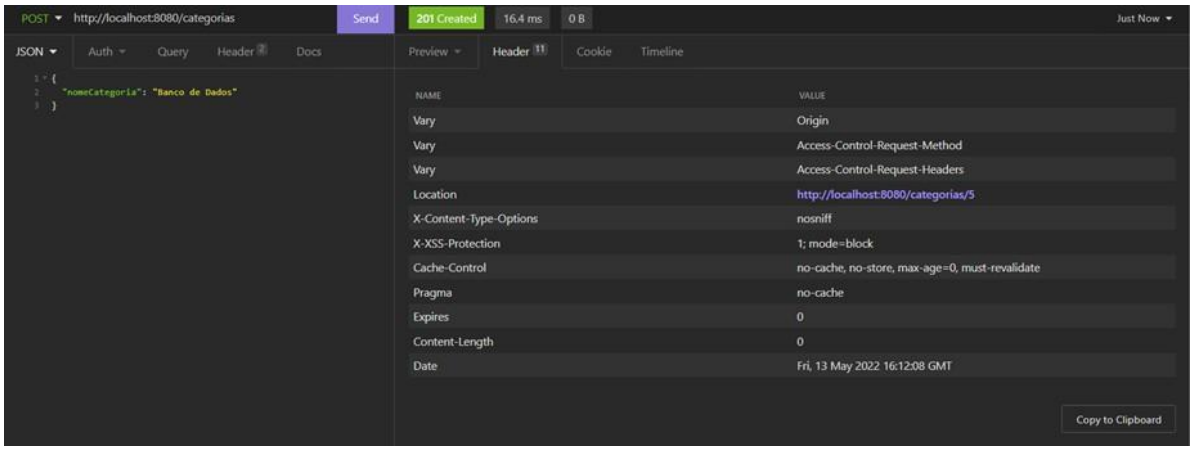


Imagem 16: Cadastrando uma CategoriaFonte: Autor (2022)

Para a ordem de serviço a regra de negócio permite que apenas clientes logados, passando o *token* de acesso, possa registrar uma ordem de serviço, assim também, como resgatá-la. Para os métodos *PUT* que atualiza o status do pagamento e o *DELETE* que exclui uma ordem de serviço está reservado apenas para o administrador, assim como também método *GET* para listar todas as ordens de serviço. O cliente pode visualizar todas as suas OS através do *path ordemservicos/page* que lista uma paginação em ordem decrescente de todas as ordens de serviços por ele contratada.

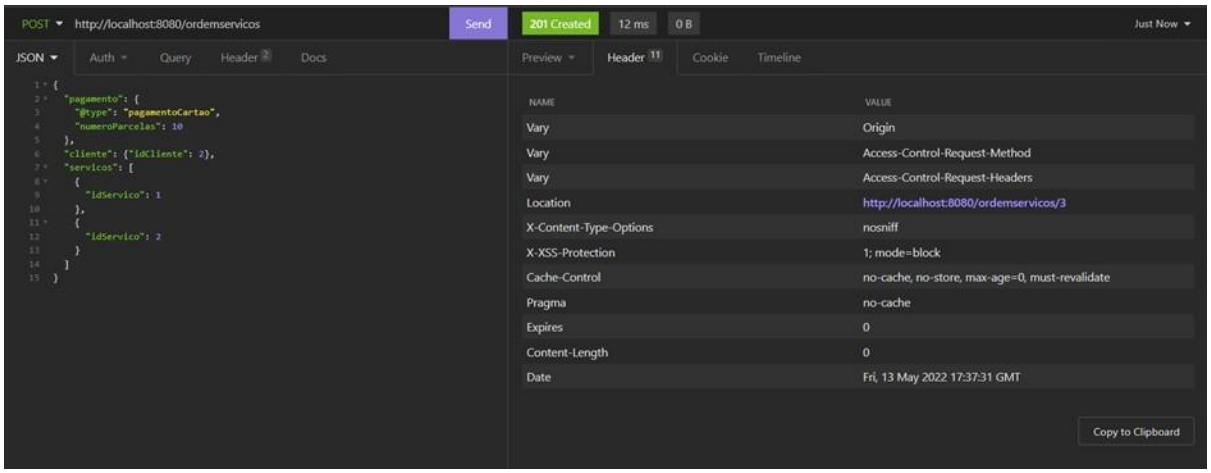


Imagem 17: Cadastrando Ordem de ServiçoFonte: Autor (2022)

Toda a demonstração da execução de cada *path* da aplicação e a explicação das regras de negócios com suas respectivas tratativas de exceção foi realizada em vídeo e hospedada na plataforma de vídeo *YouTube* podendo ser acessada através

do link: *https://www.youtube.com/watch?v=TqnU4Nr2np4*. Assim sendo, é possível verificar a API em funcionamento.

5 CONSIDERAÇÕES FINAIS

Como muitos freelancers não possuem um sistema próprio para gerenciar suas prestações de serviços, este sistema visa atender este mercado carente, uma vez que configura uma necessidade existente. Uma forma de colaborar para que profissionais tenham um maior controle sobre as suas prestações de serviço, otimizando o processo de prestação de serviços e fidelização do cliente.

Foram apresentadas a análise de negócio, o diagrama de classe com suas entidades e seus relacionamentos, caso de uso e a definição de requisitos funcionaise não funcionais que estão inseridos na aplicação.

5.1 TRABALHOS FUTUROS

Este projeto teve como foco o desenvolvimento *Backend* de um sistema de gerenciamento de tarefas para freelancers. Posteriormente a API será hospedada no *Heroku,* uma plataforma em nuvem para fazer *deploy* de aplicações e será desenvolvido o *Frontend*, na qual consumirá a API desenvolvida para executar a regra de negócio e com isso obter o gerenciamento dos serviços.

REFERÊNCIAS

COAD, P., YURDON, E. Análise baseada em objetos. Rio de Janeiro, 1992

DEITEL, Paul; DEITEL, Harvey. Como programar. 6. ed. São Paulo: Pearson Education, 2011.

HAYWOOD, D. Domain-Driven Design Using Naked Objects. [S.l.]: Pragmatic Bookshelf, 2009.

JSON Web Token. Auth0, Inc, 2018. Disponível em: https://jwt.io/introduction/. Acesso em: 01/05/2022.

Módulos Spring Framework. Pivotal Software, Inc, 2018. Disponível em: https://docs.spring.io/spring/docs/3.0.0.M4/reference/html/ch01s02.html. Acesso em: 08/03/2022.

PRESSMAN, Roger; MAXIM, Bruce. Engenharia de Software. 8. ed. Porto Alegre: AMGH, 2016.

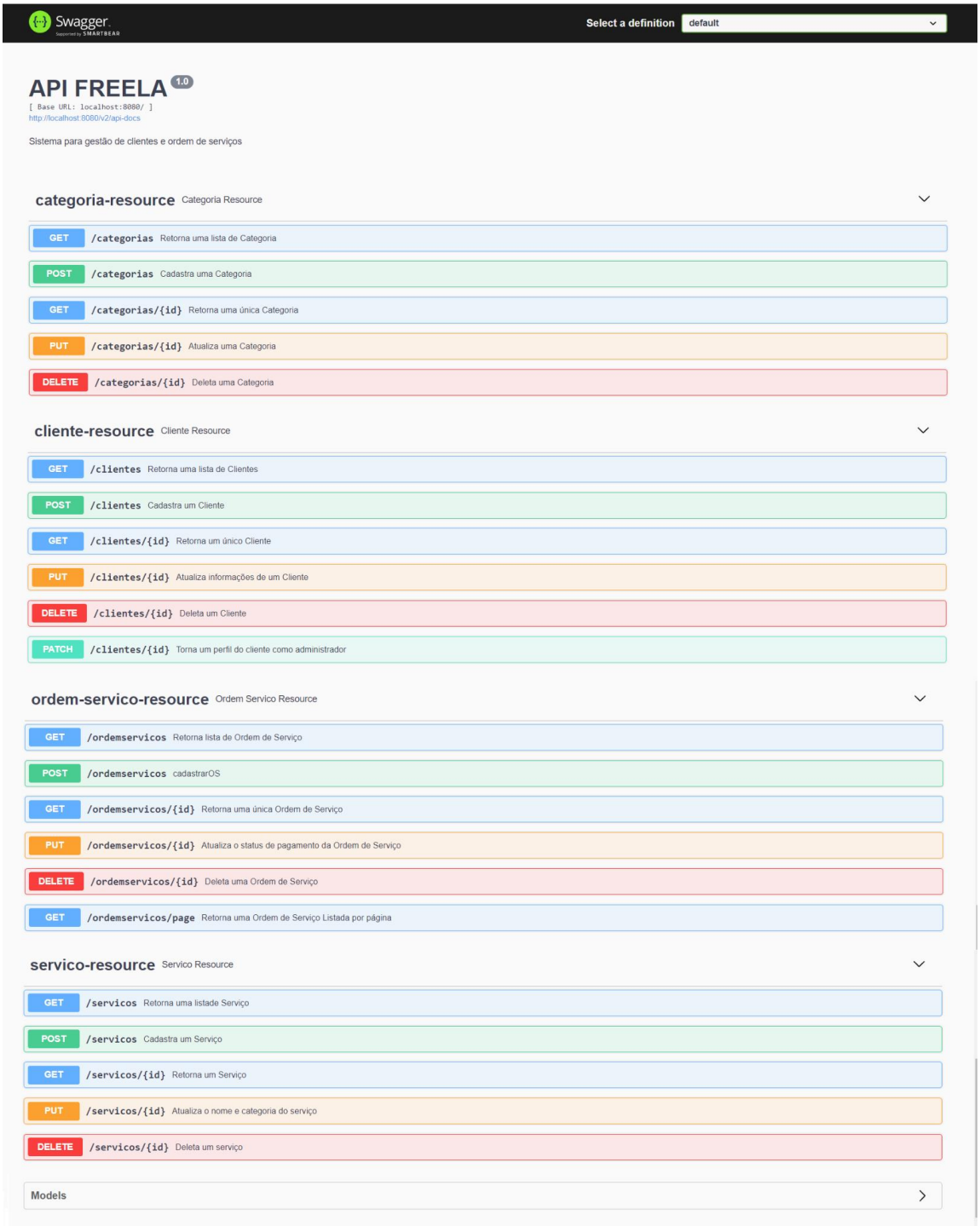
PUGA, Sandra; RISSETTI, Gerson. Lógica de programação e estrutura de dadoscom aplicações em Java. São Paulo: Pearson Education do Brasil, 2004.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo : Pearson Prentice Hall, 2011.

Spring Framework. Pivotal Software, Inc, 2018. Disponível em: https://spring.io/.Acesso em: 08/03/2022.

APÊNDICE A - DOCUMENTAÇÃO DA API

A documentação da API FREELA encontra-se disponível em *swagger* do link: *http://localhost:8080/swagger-ui/index.html.*



APÊNDICE B - DEPENDÊNCIAS

As dependências utilizadas no desenvolvimento da aplicação se encontram detalhadas abaixo. O código apresentado abaixo faz parte do arquivo pom.xml gerenciado pelo maven.

```
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.22</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jdbc</artifactId>
</dependency>
<dependency>
<groupId>com.h2database</groupId>
<artifactId>h2</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
</dependency>
```

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-validation</artifactId>
</dependency>
<dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-swagger2</artifactId>
<version>3.0.0</version>
</dependency>
<dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-swagger-ui</artifactId>
<version>3.0.0</version>
</dependency>
<dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-boot-starter</artifactId>
<version>3.0.0</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
<groupId>io.jsonwebtoken</groupId>
<artifactId>jjwt</artifactId>
<version>0.7.0</version>
</dependency>
</dependencies>
```

DEPRECIAÇÃO DE CÓDIGO E CURSOS ONLINE: ATUALIZAÇÃO DO APLICATIVO *WEATHERAPP*

SOUZA, João Pedro Israel de
BATISTA, Messias Rafael

RESUMO

O presente trabalho consiste na documentação da atualização de aplicação para previsão do tempo ministrada em curso online pela empresa ITPRO.TV em 2017, substituindo as tecnologias ensinadas no curso por soluções mais recentes, que contam com suporte técnico e comunidade de desenvolvedores ativa. Ocorreu a substituição de uma das APIs que alimenta a aplicação, a troca do *framework* principal (de AngularJS para Angular 8), da linguagem de programação principal e de algumas bibliotecas que dão suporte à aplicação. A atualização da aplicação foi um sucesso e conseguiu recriar todas as funcionalidades da aplicação originalmente criada no curso com o benefício de possuir soluções com maior suporte nas comunidades de desenvolvimento atual, facilitando o seu suporte e potencial expansão de seus serviços.

Palavras-chave: Atualização de código; código antigo; aplicações web; API.

1 INTRODUÇÃO

De acordo com o dicionário online Dicio (2022), a tecnologia se define como “[a] teoria ou análise organizada das técnicas, procedimentos, métodos, regras, âmbitos ou campos da ação humana”. Dessa maneira, se pode inferir que a tecnologia, assim como toda teoria, evolui constantemente, trazendo com esse desenvolvimento impactos na ordem da organização social, cultural, trabalhista e ambiental.

Atrelada a essa constante evolução, e com o fortalecimento do mercado de Tecnologia da Informação (popularmente reconhecido pela sigla “TI”), surge uma grande demanda para o desenvolvimento e a manutenção de sistemas já existentes. Devido a alta demanda por profissionais qualificados nas mais diversas tecnologias disponíveis no mercado (e a promessa de carreiras com boa bonificação e estabilidade financeira) , este tem sido um segmento que tem atraindo pessoas dos mais diversos campos e experiência profissional, não necessariamente apenas estudantes da área TI e áreas afins.

Todavia, com o alto custo financeiro, temporal e emocional associado com formações acadêmicas, muitas vezes uma formação na área de TI se torna inviável para uma boa parte dos profissionais. Esse obstáculo, associado com a constante e rápida evolução das tecnologias, faz com que linguagens de programação e

bibliotecas se tornem “obsoletas” em um ritmo cada vez mais veloz, gerando espaço para um mercado paralelo ao mercado de TI: o de cursos online em TI, se destacando plataformas de educação como a *Udemy* e *Codecademy*.

Todavia, tais cursos possuem um tempo de vida limitado à relevância da tecnologia que ensinam, de maneira que o valor profissionalizante de um curso online será por muitas vezes atrelado não apenas ao método de ensino ou qualidade do material, mas a própria tecnologia que ensina.

Tendo isso em vista, este trabalho tem como objetivo utilizar tecnologias modernas para a construção de um aplicativo de previsão do tempo similar ao administrado pela rede de ensino ITPRO.TV, em seu curso *Weather Application with Angular Express* (Aplicação de previsão do tempo com *Angular* e *Express*).

2. FUNDAMENTAÇÃO TEÓRICA

Para poder atualizar a aplicação *WeatherApp* sem perder suas funcionalidades anteriores, se faz necessário compreender os conceitos utilizados em seu desenvolvimento original, assim como as ferramentas que fizeram parte de sua construção.

2.1 *ES5* e *AngularJS*

O curso originalmente utiliza *ECMAScript* 2009 (Também conhecido como ES5) para configurar ambos os front-end e o back-end da aplicação. A linguagem, de acordo com a *Ecma International*:

> [..] foi originalmente projetado para ser uma linguagem de script da Web, fornecendo um mecanismo para animar páginas da Web em navegadores e para executar computação de servidor como parte de uma arquitetura cliente-servidor baseada na Web. (*ECMA INTERNATIONAL,* 2022)

O curso utiliza essa linguagem de programação devido ao uso do *framework* (conjunto de códigos que facilitam o desenvolvimento de um projeto) AngularJS, um projeto *Open Source* (de código aberto, onde qualquer usuário pode contribuir para o seu desenvolvimento) criado e mantido pela Google, cuja proposta é facilitar a criação de aplicações *web*.

Pela compatibilidade do *ES5* com navegadores *web* e suporte a Programação

Orientada a Objetos, ela é a linguagem principal do *framework*, e portanto, a ministrada no curso, onde é utilizada para a criação de módulos, componentes e serviços dentro da aplicação.

2.2 Administração de dependências com NPM

Administradores de pacotes, de acordo com Burrows, Montecelo (2004), são coleções de ferramentas de *software* que automatizam o processo de instalação, *upgrade*, configuração e remoção de programas de computador em uma maneira consistente, sendo amplamente utilizados no desenvolvimento de aplicações web.

Para administrar o uso de dependências do código, o curso utilizou o administrador de pacotes *Node Package Manager* (NPM)[1], que conta com uma vasta biblioteca de softwares de código aberto para o uso de seus usuários, das quais o aplicativo utiliza as seguintes dependências, conforme a figura 1.

Figura 1. Dependências do projeto *WeatherApp* adminstrado pela ITPRO.TV

```
6    "dependencies": {
7      "angular": "^1.6.5",
8      "angular-nvd3": "^1.0.9",
9      "tachyons": "^4.7.4"
10   },
```

Fonte: DENNISON, Justin (2017)

Todas essas bibliotecas possuem funcionalidades específicas: a *Angular-nvD3* é uma implementação da biblioteca *nvD3* para *AngularJS*, que constrói gráficos a partir de dados repassados pela aplicação, já o *Tachyons* é uma biblioteca de estilos que oferece classes para facilitar a criação de interfaces, enquanto a dependência *Angular* é necessária para utilizar o próprio *AngularJS*.

2.3 Obtendo dados com chamadas a APIs

Para adquirir os dados meteorológicos e geográficos necessários para fazer a aplicação proposta funcionar, o curso utilizou-se de duas APIs (*Application Programming Interface*, ou Interface de Programação de Aplicações): a *Dark Sky* e a

[1] Gerenciador de pacotes para Node.js. https://www.npmjs.com/about

Google Geocoding.

Estas APIs, após autenticação do usuário por meio de uma chave gerada ao se cadastrar em suas respectivas plataformas, oferecem seus dados por meio de chamadas do tipo GET[2] no formato de arquivos JSON, conforme a figura 2.

Figura 2. Exemplo de resposta a chamada a API *Google Geocoding*, em JSON

```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "1600",
               "short_name" : "1600",
               "types" : [ "street_number" ]
            },
            {
               "long_name" : "Amphitheatre Parkway",
               "short_name" : "Amphitheatre Pkwy",
               "types" : [ "route" ]
            },
            {
               "long_name" : "Mountain View",
               "short_name" : "Mountain View",
               "types" : [ "locality", "political" ]
            },
            {
               "long_name" : "Santa Clara County",
               "short_name" : "Santa Clara County",
               "types" : [ "administrative_area_level_2", "political" ]
            },
```

Fonte: *Geocoding API* (2022)

Para o seu uso, é necessário criar um registro em ambas as plataformas Google *Developers* e *Dark Sky*, para receber uma chave de acesso às interfaces de programação. Ademais, o uso da api de localização geográfica do Google requer o registro de dados bancários, que oferece um serviço gratuito de até quatro mil acessos por mês, passando a cobrar US$ 4 dólares a cada mil acessos após o limite de requisições ser atingido.

Ambas as APIs trabalham de forma complementar para adquirir a localização pedida pelo usuário, realizar uma pesquisa dessa localização por meio da *Geocoding* API, adquirindo os valores da latitude e longitude aproximadas do local e repassando essas informações para a *Dark Sky* API para encontrar os dados meteorológicos da área.

3 METODOLOGIA

A atualização do aplicativo consistiu no acompanhamento das aulas ministradas pela plataforma online ITPRO.TV, anotando as implementações de

[2] Método do padrão HTTP para requisição de dados de um servidor

funcionalidades no aplicativo e realizando uma pesquisa de mercado para buscar tecnologias mais modernas para a implementação no *WeatherApp,* com funcionalidades similares às propostas no curso.

3.1 Substituição do AngularJS pelo Angular 8

Para a modernização do aplicativo, foi substituído o uso do *framework* AngularJS para a sua versão mais moderna em Typescript: Angular. Essa escolha foi tomada devido a depreciação do AngularJS, que parou de receber suporte do Google desde Janeiro de 2022 (AngularJS, 2022). Outro fator de grande importância para a troca foi o suporte que o Angular possui para desenvolvimento *mobile*, sua maior velocidade de performance e por possuir uma CLI (*Command Line Interface*, ou Interface de Linha de Comando) integrada, facilitando a implementação de novas funcionalidades.

3.2 Utilizando outra API para requisição de dados meteorológicos

A API utilizada para a requisição de dados meteorológicos ministrada no curso é a *Dark Sky* API, que descontinuou seu suporte para novos usuários em 2021, com o suporte da API previsto para se encerrar no fim de 2022, tornando-a inviável para a aplicação.

Para substituir a *Dark Sky*, foi utilizada a *OpenWeatherMap* API, que oferece um serviço similar de aquisição de dados meteorológicos de forma gratuita para fins não-comerciais, necessitando apenas de uma chave de acesso obtida via cadastro em sua plataforma, conforme apresenta a figura 3.

Figura 3. Chamada a *OpenWeatherMap* API

```
https://api.openweathermap.org/data/2.5/onecall?lat=
{lat}&lon={lon}&exclude={part}&appid={API key}
```

Fonte: *OpenWeather* (2022)

A *OpenWeatherMap* API oferece, na sua chamada *onecall* (única chamada) uma alternativa viável a *Dark Sky*, trazendo dados similares a ela como o horário do pôr-do-sol, amanhecer, temperatura local, humidade, pressão atmosférica e a

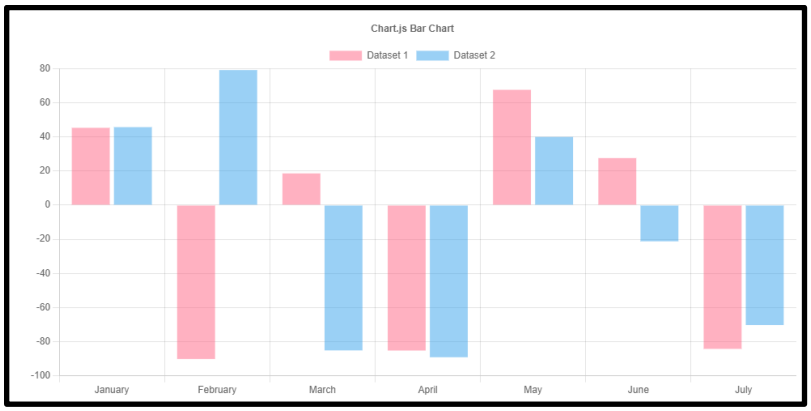
previsão para os próximos dias da semana, horas e minutos e ícones representando o tempo atual.

3.3 Criando gráficos informativos com a biblioteca ng2-charts

O curso utiliza a biblioteca *Angular-nvd3* para a criação de gráficos com a previsão de tempo em minutos, sendo essa dependência uma implementação da biblioteca *nvd3* para Angular, utilizando as funcionalidades dela em módulos que podem ser utilizados no programa sem configurações adicionais. Todavia, esta biblioteca não recebe atualizações a 6 anos, se optando pela sua substituição pela biblioteca *ng2-charts*.

O *ng2-charts*, assim como o *nvd3*, é uma implementação para Angular de outra biblioteca, o *charts.js*, que auxilia na criação de gráficos informativos nos formatos barras, linhas e pizza, conforme a figura 4.

Figura 4. Exemplo de gráfico de barras construído com *charts.js*



Fonte: *Chart.js* (2022)

Devido a suas boas avaliações em sua página no NPM[3], frequentes atualizações e funcionalidade similar a biblioteca que propõe substituir, a *ng2-charts* foi escolhida para sanar a necessidade de representação de dados gráficos na aplicação final.

3.3 Mantendo o uso do NPM e *Tachyons*

[3] https://www.npmjs.com/package/ng2-charts

Apesar de existirem alternativas mais modernas para administradores de pacotes (como o Yarn), foi decidido manter o uso do NPM pela sua grande popularidade no mercado, com quase o dobro de downloads do Yarn no último ano (NPM TRENDS, 2022).

Também foi optado pela continuação da utilização da biblioteca de estilos *Tachyons* para o aplicativo, uma vez que suas classes possuem design responsivo e carregamento veloz.

4 FUNDAMENTAÇÃO TEÓRICA

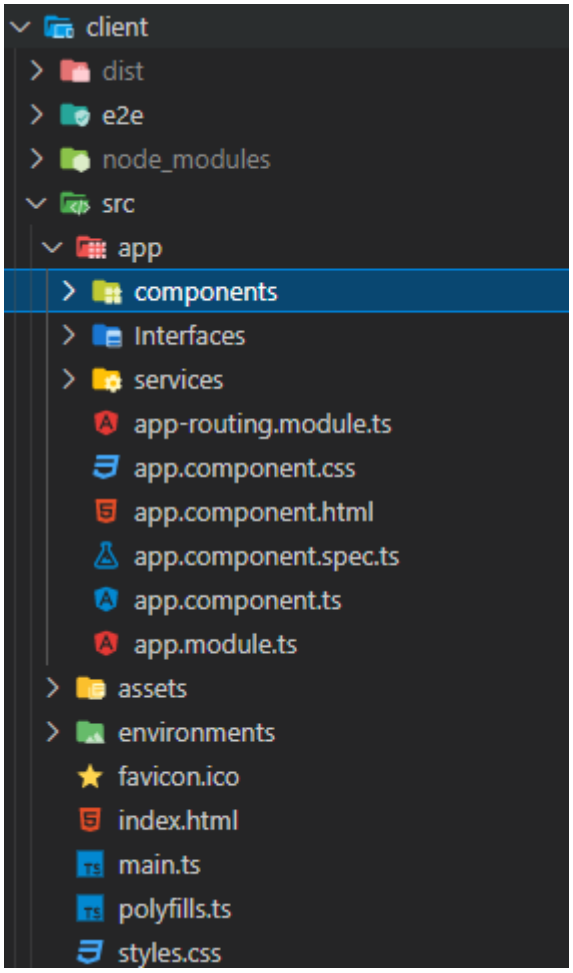
O desenvolvimento do aplicativo acompanhou as aulas do curso, utilizando como principais recursos as documentações oficiais do Angular 8, Google *Developers*, *OpenWeatherMap* API e *Charts.js*.

4.1 O ambiente de desenvolvimento Angular

Para facilitar o desenvolvimento da aplicação foi utilizado a interface de linha de comando (CLI) Angular, que permite rapidamente criar componentes e serviços por meio da linha de comando, além de configurar serviços de teste de software, roteamento e a atualização automática do projeto conforme mudanças são implementadas no código.

Para criar um projeto com a Angular CLI, basta abrir uma instância do terminal, navegar a pasta desejada e inserir o comando *ng new <nome do projeto>*, criando uma nova pasta com o nome do projeto e contendo a estrutura apresentada na figura 5, notoriamente na raiz do projeto contendo a pasta *dist* (de *distribution,* ou distribuição), onde versões compiladas para web são exportadas e a pasta *src* (de *source*, ou fonte) onde o desenvolvedor cria arquivos os componentes, serviços e testes da aplicação.

Figura 5. Principal estrutura do projeto client, parcialmente gerada pelo Angular CLI



Fonte: Elaboração própria (2022)

Fora as pastas de *services* e *components*, foi criada uma terceira pasta em */app*, chamada *interfaces*, cujo objetivo é a padronização de respostas JSON em objetos, essa decisão foi tomada para facilitar a classe de serviços *geolocation.service*, que será elaborada no próximo tópico.

4.2 Comunicação com as APIs por meio de classes de serviço

Com o objetivo de apresentar dados meteorológicos para o usuário, a aplicação é alimentada por duas APIs públicas, a *OpenWeatherMap* e a *Google Geocoding* API, ambas gratuitas para uso não comercial.

Visando manter a segurança do aplicativo, foi criada a classe *credentials*, que cria um objeto JSON contendo as chaves api de ambos os serviços.

O aplicativo interage com as interfaces por meio de duas classes de serviços: a *weather.service* e a *geolocation.service*. Os métodos implementados nessas classes diferem com os administrados pelo curso, visando respeitar o princípio de

responsabilidade única nas classes. No curso, a classe *geolocation.service* tem apenas um método chamado *getPosition()*, que retorna a localização do usuário no Browser, se esta for permitida, conforme apresentado na figura 6.

Figura 6. Método *getPosition()* do curso

```
7     getPosition(){
8       return this.$q((resolve, reject) => {
9           const {navigator} = this.$window;
10          if(!navigator){
11            reject('Geolocation Services are not available');
12          } else {
13            const success = position => {
14              const {latitude, longitude} = position.coords;
15              resolve({
16                latitude,
17                longitude
18              });
19            };
```

Fonte: DENNISON, Justin (2017)

A versão atualizada utiliza dois métodos na classe, um similar ao *GetPosition()* original do curso, renomeado como *GetWindowLocation()* e um novo método chamado *GetLocation*(), que realiza uma chamada a *Geocoding* API, conforme figura 7.

Figura 7. Métodos da classe *GeolocationService* atualizada

```
  public getLocation(location: String) {
    const url = `https://maps.googleapis.com/maps/api/geocode/json?address=${location}?
    return this.http.get(url)
  }

  //Pedir localização no browser, recolher o dado e enviar como latitude e longitude
  public getWindowLocation(){      You, há 7 dias • Versão funcional, sem CSS … ···
  }
}
```

Fonte: Elaboração própria (2022)

O método *getLocation()* recebe uma *String* com o nome da localização que se deseja buscar, repassa esse dado para uma constante que monta a url de requisição e retorna a chamada API por meio da classe *HttpClient*, que faz parte do Angular 8, retornando um objeto JSON contendo diversas informações geográficas, ao qual interessa a aplicação a latitude e longitude aproximada da cidade, como apresentado na figura 8.

Figura 8. Trecho da Resposta JSON da Google Geocoding API


```
"formatted_address": "João Pessoa, PB, Brasil",
"geometry": {
  "bounds": {
    "northeast": {
      "lat": -7.058043000000001,
      "lng": -34.7931903
    },
    "southwest": {
      "lat": -7.2351747,
      "lng": -34.9700047
```


Fonte: Elaboração própria (2022)

Já a função *getWindowLocation()* foi alterada em funcionamento, primeiramente recebendo uma promessa de resposta tipada, a interface *Icoordinates*, facilitando a leitura dos dados de latitude e longitude esperados. Por fim, a classe também faz o tratamento de possíveis erros, como incompatibilidade com navegadores que não suportam serviços de localização e se o usuário não permitir que a página tenha acesso aos dados geográficos, apresentados na figura 9.

Figura 9. Trecho da Resposta JSON da Google Geocoding API

```
public getWindowLocation(){
  return new Promise<Icoordinates>((resolve, reject) => {
    const { navigator } = this.window
    if (!navigator) {
      reject('Esse browser não é compatível com os serviços de geolocalização')
    } else {
      const success = position => {
        const {latitude, longitude} = position.coords
        resolve({
          latitude,       You, semana passada • updating geolocation service, tryi
          longitude
        })
      }
      const error = err => {
        reject(err)
      }
      navigator.geolocation.getCurrentPosition(success, error)
    }
  })
```

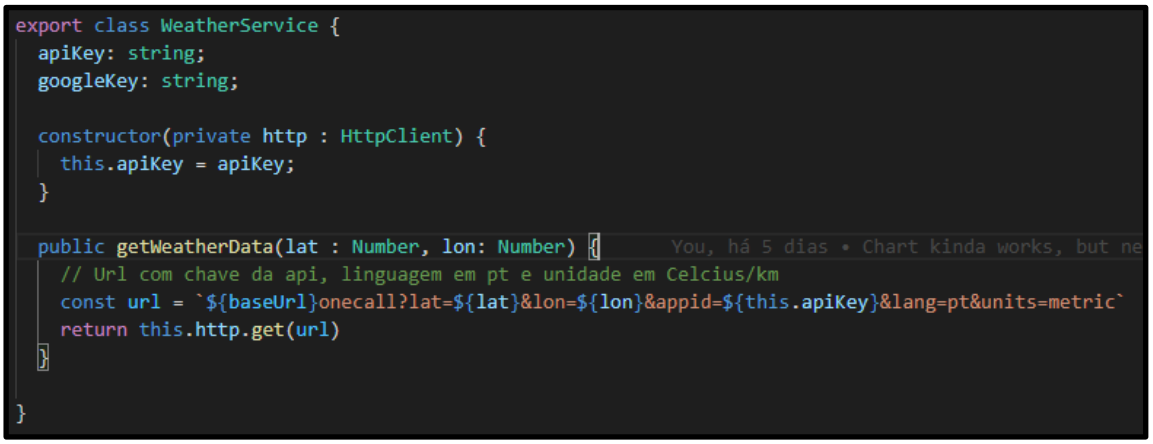
Fonte: Elaboração própria (2022)

Já a classe *WeatherService* possui apenas a função de realizar a chamada API utilizando os dados de latitude e longitude, por meio do método *getWeatherData()*. No curso, essa classe possui duas funções, uma que realizava a chamada buscando pela

latitude e longitude e outra buscando por localização, todavia, a busca por latitude e longitude é uma função exclusiva da classe de geolocalização, ferindo o princípio da responsabilidade única, tornando esse trabalho exclusivo a classe *GeolocationService*.

O método *getWeatherData()* funciona de forma similar a sua implementação no curso, recebendo como argumentos dois valores numéricos referentes a latitude (lat) e longitude (lon), montando a url de requisição, adicionando opções de linguagem preferida dos dados e unidade de medida, e realizando a chamada, conforme mostrado na figura 10.

Figura 10. Classe *weatherService*, com seus principais métodos



Fonte: Elaboração própria (2022)

Após montar a url, ela é chamada por meio do método HttpClient.get(), retornando um objeto JSON com diversos dados meteorológicos, incluindo temperatura, sensação térmica, temperaturas mínimas e máximas e umidade local.

4.3 Integração dos dados com o componente App-controller

Para manusear os dados recebidos, a classe *App-controller* funciona como o componente principal da aplicação, armazenando e administrando os dados adquiridos pelas APIs, garantindo sua obtenção e repassando-os para outros componentes da aplicação.

Figura 11. Classe *weatherService*, com seus principais atributos

```
export class AppControllerComponent implements OnInit {
  weatherData: any;
  locationData: any;

  weatherLoaded: Promise<Boolean>;
  locationLoaded: Promise<Boolean>;

  lat: Number;
  lon: Number;
  location: string;

  constructor(
    private ws: WeatherService,
    private gs: GeolocationService
  ) {}
```

Fonte: Elaboração própria (2022)

A classe é, assim como todos os componentes Angular, estruturada como um objeto, possuindo seus próprios atributos, construtor e métodos, conforme apresentado na figura 11. Notavelmente, o construtor recebe injetado instâncias das classes *weather.service* e *location.service*, permitindo que o componente possa invocar o seus métodos.

Os atributos *weatherData* e *locationData* recebem os valores obtidos através dos métodos da classes *weather.service* e *geolocation.service*, permitindo que estes sejam repassados para outros componentes. As booleanas *weatherLoaded* e *locationLoaded* garantem que o método *onSearch()* só executará seus próximos passos se os atributos terminados em *data* possuírem dados, conforme a figura 12.

Figura 12. Método *onSearch()*, chamando dados meteorológicos

```
onSearch(location: string) {
  this.gs.getLocation(location).subscribe(
    res => {      You, há 2 semanas • Adicionando geocoding no google
      this.locationData = res

      //Garantir que dados foram pegos
      this.locationLoaded = Promise.resolve(true)
      if (this.locationLoaded) {
        this.lat = this.locationData.results[0].geometry.location.lat
        this.lon = this.locationData.results[0].geometry.location.lng
        this.ws.getWeatherData(this.lat, this.lon)
        .subscribe(
          res => {
            this.weatherData = res
            //Garantir que o dado foi carregado antes de executar
            this.weatherLoaded = Promise.resolve(true)
          })
      }
    })
```

Fonte: Elaboração própria (2022)

A função *onSearch()* tem como única funcionalidade preencher as variáveis da classe *app-controller*, chamando as classes de serviço da aplicação, repassando seus dados para as variáveis terminando em *Data*, garantir que elas estão preenchidas e tornar esses dados disponíveis por meio das classes terminando em *Loaded* e tornando esses dados disponíveis para outros componentes que queiram utilizar-los.

Os dados recebidos pelo *app-component* são repassados para outros componentes por meio de *binds* no HTML, que correspondem a atributos e métodos presentes na classe-pai, conforme figura 13.

Figura 13. *Binds* do *app-component*

```
<search
    (updatedLocation)="onSearch($event)"
>
</search>
<div *ngIf="weatherLoaded">
    <current-weather
        [weatherData]="weatherData"
    >
    </current-weather>

    <hourly-weather
        [weatherData]="weatherData"
    >
    </hourly-weather>

    <data-chart
        [weatherData]="weatherData"
    >
    </data-chart>
</div>
```

Fonte: Elaboração própria (2022)

A condicional **ngIf* faz com que o bloco HTML apenas apareça se a variável *weatherLoaded* estiver com o valor *True*, garantindo que essa parte do código só será apresentada ao usuário se existirem dados para serem apresentados ao usuário.

As *Tags current-weather*, *hourly-weather* e *data-chart* correspondem aos outros componentes da aplicação, todos possuindo *binds* (os blocos HTML envoltos por colchetes “[]”) repassando a variável *weatherData* do componente pai para os filhos.

Os componentes foram separados de maneira que todo o controle de dados seja administrado pela classe app-controller, com os outros componentes tendo a

função de apenas repassar dados da interface para alimentar os métodos do *app-controller*, que por sua vez utiliza as classes de serviços.

4.4 Divisão de responsabilidades com componentes

A aplicação possui cinco componentes, cada uma com uma única função, visando facilitar a sua manutenção. O componente *app-controller* tem como responsabilidade emitir os seus dados adquiridos por meio das classes de serviços aos componentes *current-weather*, *data-chart* e *hourly-weather* possuem a responsabilidade de receber uma cópia dos dados *weatherData* adquiridos pela classe *app-component* e apresentá-la de maneiras diferentes, por meio do uso de *binds*, cuja funcionalidade foi explicada na seção 4.3.

Os dados, quando recebidos pelos componentes filhos e alimentando a sua cópia da variável *weatherData*, são utilizados no código HTML (por meio do uso de chaves “{{}}”) para mostrar aos usuários as informações que o componente se propõe a trazer, como por exemplo, o componente *current-weather*, que traz ao usuário um ícone representando o tempo, a descrição do céu, a temperatura atual e a sensação térmica, conforme figura 14.

Figura 14. HTML da classe *current-weather*

```
<div class="mw5 center bg-white br3 pa3 pa4-ns mv3 ba b--black-10">
    <div class="tc pa4">
        <img
            src="http://openweathermap.org/img/wn/{{ weatherData.current.weather[0].icon }}@2x.png"
            class="br-100 pa1 ba b--black-10 h3 w3" alt="{{ weatherData.current.weather[0].description }}"
        >
    </div>      You, há 5 dias • Adicionando Tachyons …
    <p>{{ weatherData.current.weather[0].description }}</p>
    <p>A temperatura é atualmente: {{ weatherData.current.temp }}ºC</p>
    <p>A sensação térmica é de: {{ weatherData.current.feels_like }}ºC</p>
</div>
```

Fonte: Elaboração própria (2022)

Por fim, o componente *search* é o único que emite um evento e não recebe dados da classe *app-controller,* tendo como responsabilidade montar o formulário de busca de localização e enviar essa *String* por meio do evento *updatedLocation* para *app-controller*, que utiliza essa localização para interagir com as APIs.

4.5 Apresentando variação de temperatura com *ng2-charts*

O componente *data-chart* utiliza a biblioteca *ng2-charts* para criar um gráfico de barras com a variação da temperatura ao longo do dia, oferecendo informações das próximas 24 horas.

Para isso, a dependência oferece o seu próprio componente adaptado ao Angular 8, que aceita receber variáveis informando o tipo de gráfico desejado, os dados a serem apresentados, suas legendas e um objeto com opções extras, informações que são providenciadas quando a classe *data-chart* é iniciada e recebe os dados em *weatherData*, conforme a figura 15.

Figura 15. Informando os dados para a construção do gráfico de barras

```
ngOnInit() {

  if (this.weatherData.length != 0) {
    let barChartUpdatedData = []
    let barChartUpdatedLabels = []

    this.barChartOptions = {
      scaleShowVerticalLines: false,
      responsive: true
    }
    this.barChartType = 'bar';
    this.barChartLegend = false;

    this.weatherData.hourly.forEach(hour => {
      barChartUpdatedData.push(hour.temp)
      barChartUpdatedLabels.push(new Date(hour.dt * 1000).getHours() )
    });

    this.barChartLabels = barChartUpdatedLabels;
    this.barChartData =
    [
      {data: barChartUpdatedData},
    ];      You, há 10 horas • Versão final V1.0.0 …
  }
}
```

Fonte: Elaboração própria (2022)

Ao iniciar a classe, ela é alimentada pelos dados da variável *weatherData*, informando a variação da temperatura pelas próximas 24 horas, recebendo também esse horário como as legendas para o gráfico.

4.6 Resultados

A aplicação apresentou no final de seu desenvolvimento uma aparência similar ao *weatherApp* criado no curso, conforme apresentado na figura 16.

Figura 16. Componentes *search* e *current-weather* apresentados na aplicação

Fonte: Elaboração própria (2022)

A utilização do *Tachyons* na aplicação se limitou a manter a aparência dela similar a versão ministrada no curso, utilizando as mesmas classes presentes para estilizar os componentes, uma vez que o escopo do trabalho estava limitado na atualização das tecnologias utilizadas no desenvolvimento da aplicação, não realizando incremento em suas funcionalidades.

5 CONSIDERAÇÕES FINAIS

Ao final do processo, a recriação da aplicação *WeatherApp* utilizando novas tecnologias foi um sucesso, sendo capaz de recriar todas as principais funções da aplicação desenvolvida no curso. O aplicativo em Angular 8, assim como o desenvolvido em AngularJS, é capaz de obter a localização do usuário por meio de seu navegador, pesquisar a previsão do tempo de um local utilizando APIs e

apresentar esses dados em uma interface para o usuário final.

A substituição da API *Dark Sky* pela *OpenWeatherMap* não afetou o recebimento de dados pela aplicação, nem os gráficos produzidos pelo *chart.js* apresentaram muitas diferenças aos produzidos em *nvD3*, todavia, o *WeatherApp* desenvolvido com essas novas tecnologias trouxe o benefício de maior facilidade no desenvolvimento, manutenção, adaptação para dispositivos móveis e performance atrelados com o uso do *framework* Angular 8.

Não é o objetivo deste trabalho desmerecer sistemas que utilizem as tecnologias ministradas no curso ministrado pela ITPRO.TV, mas sim servir como uma prova de conceito para a modernização de uma aplicação antiga com o uso de tecnologias com melhor suporte e atualizações no mercado tecnológico.

Por fim, a aplicação em si é bem simples e focada em uma singular funcionalidade, podendo ser expandida de diversas formas, como a implementação de um sistema de recomendação de roupas baseado no tempo ao longo do dia, ou transformando seu serviço em uma API que integre as funcionalidades de busca geográfica com dados meteorológicos. Ademais, este trabalho pode se expandir em uma discussão do custo tempo-benefício da atualização de aplicações, estabelecendo uma métrica entre os benefícios adquiridos por um *upgrade* na aplicação contra o tempo (e consequentemente capital) investido para tal.

REFERÊNCIAS

ANGULARJS. Disponível em: <https://angularjs.org/>. Acesso em 8 de Março de 2022.

BURROWS, Daniel; MONTECELO, Manuel. ***What is a package manager?*** Disponível em: <https://www.debian.org/doc/manuals/aptitude/pr01s02.en.html>. Acesso em 9 de Maio de 2022.

CHART.JS: ***Vertical Bar Chart***. 12 de fevereiro de 2022. Disponível em: <https://www.chartjs.org/docs/latest/samples/bar/vertical.html>. Acesso em: 17 maio 2022.

DENNISON, Justin. **Angcaster-client**. 1 de agosto de 2017. Disponível em: <https://github.com/justindevpro/angcaster-client>. Acesso em: 17 maio 2022.

DICIO. **Significado de Tecnologia**. Disponível em: <https://www.dicio.com.br/tecnologia/>. Acesso em 8 de Março de 2022.

ECMA INTERNATIONAL. ***ECMAScript® Language Specification*****.** Disponível em:

<https://262.ecma-international.org/5.1/>. Acesso em 9 de Maio de 2022.

GEOCODING API: ***Get Started***. 13 de maio de 2022. Disponível em: <https://developers.google.com/maps/documentation/geocoding/start>. Acesso em: 17 maio 2022.

NPM. ***About npm***. Disponível em: <https://www.npmjs.com/about>. Acesso em: 17 maio 2022.

OPENWEATHER. Disponível em: <https://openweathermap.org/>. Acesso em: 17 maio 2022.

POTTER, JOHN. ***Npm Trends***. Disponível em: <https://www.npmtrends.com/npm-vs-yarn>. Acesso em 8 de Março de 2022.

DESENVOLVIMENTO DE UM APLICATIVO EM *ANDROID* PARA O SINDICATO DOS CONDUTORES EM TRANSPORTE PÚBLICO ALTERNATIVO DE PEDRAS DE FOGO - PB

ARAÚJO, Bruno Rodrigues de
BATISTA, Messias Rafael

RESUMO

A expansão dos smartphones nos tem possibilitado diversas transformações ao longo dos anos e em um espaço curto de tempo, têm surgido vários aplicativos que nos possibilitam fazer determinadas tarefas. Nesse contexto, tem surgido ferramentas que nos auxiliam na construção de aplicativos, como por exemplo, o Android Studio que nos possibilita criar aplicativos reais para o sistema operacional Android. É através dele que iremos criar a nossa aplicação. O trabalho tem como objetivo desenvolver um aplicativo para auxiliar os motoristas de transporte alternativo, visto que ainda é utilizado de uma forma manual, a realização do check-in, foi necessário a criação desse aplicativo em Android, utilizando a linguagem *Kotlin,* se comunicando com o Google Maps API, e através da localização do motorista e dentro da área determinada, poderá realizar o check-in.

Palavras chaves: Google Maps; Android; Firebase.

1 INTRODUÇÃO

Com o grande crescimento de *smartphones*, cada vez mais empresas têm se voltado para o mundo *mobile*, criando seus próprios aplicativos e investindo cada vez mais nessa área. Uma pesquisa sobre o uso de *smartphone* realizada pela empresa *Strategy Analytics* informou uma estimativa de que 3,85 bilhões de pessoas possuem um celular em 2021. "Nós estimamos que a base de usuários de *smartphones* cresceu dramaticamente, de 30 mil usuários em 1994 para 1 bilhão em 2012, e agora bateu o recorde com 3,85 bilhões em 2021", afirmou *Yiwen Wu*, analista sênior da empresa. "Com uma estimativa de 7,9 bilhões de pessoas no planeta em junho de 2021, isso significa que 50% de todo o mundo possui *smartphones*", acrescentou (TECMUNDO, 2021).

Alinhado com o avanço da tecnologia, as empresas e desenvolvedores têm criado diversos aplicativos com diferentes soluções para várias áreas, mas ainda assim, sentimos falta de um aplicativo em um contexto específico. Sendo assim, foi colocada em prática a ideia de um aplicativo para o transporte alternativo de Pedras de Fogo - PB.

O transporte alternativo nada mais é que um transporte de locomoção de uma cidade para outra em carros comuns. Nas cidades do interior da Paraíba existem

esses transportes que levam as pessoas para a capital (João Pessoa - PB), que possui um local ou "ponto" na cidade, onde as pessoas que precisam se locomover para a capital pagam uma taxa para ir. Existem dois pontos onde ficam os carros, na cidade local (Pedras de Fogo - PB) e na cidade de destino (João Pessoa - PB). São dezenas de carros e quando o carro chega na capital, é necessário que uma pessoa anote o nome do motorista em um caderno, para que ele saiba qual a vez dele, já que são vários carros e geralmente só saem quando tem pessoas suficientes para a viagem.

Pensando nisso, foi criado um aplicativo *Alternative Check-In* para substituir essa forma manual de anotar o nome do motorista. O aplicativo feito em *Android*, funciona da seguinte forma, o motorista assim que chegar a cidade de destino, poderá fazer o *check-in* pelo aplicativo, o *check-in* verifica se o usuário está realmente no local, através da *API* do *Google Maps* e depois envia as informações do motorista para uma lista utilizando o *Realtime Database* do *Firebase*, onde irá mostrar a sua posição em tempo real.

2 ANDROID

O Android é o sistema operacional móvel mais utilizado do mundo, é projetado principalmente para dispositivos móveis com tela sensível ao toque como smartphones e tablets, além de ter o código aberto (*open source*), isso facilita para desenvolvedores contribuir para a plataforma. Para trabalhar com desenvolvimento *Android* é necessário ter o *Android SDK* em sua máquina.

O *Android SDK* é o software utilizado para desenvolver aplicações no *Android*, que tem um emulador para simular o dispositivo, ferramentas utilitárias e uma *API* completa para a linguagem *Java*, com todas as classes necessárias para desenvolver as aplicações, (LECHETA, 2015). Utilizaremos o *Android Studio* que já vem com o *Android SDK* integrado com a linguagem de programação *Kotlin* para criarmos nossa aplicação.

O *Android Studio* é o ambiente de desenvolvimento integrado (*IDE*, na sigla em inglês) oficial para o desenvolvimento de aplicativos *Android*. O *Android* Studio possui diversos recursos para aumentar sua produtividade na criação de aplicativos *Android*, dentre eles um sistema de compilação flexível baseado em *Gradle*, um ambiente unificado que possibilita o desenvolvimento para todos os dispositivos *Android*,

Frameworks e ferramentas de teste cheios de possibilidades *(ANDROID DEVELOPERS, 2021).*

KOTLIN

A linguagem de programação *Kotlin* foi desenvolvida pela *JetBrains*, a mesma criadora do *Android Studio*, e veio como uma alternativa ao *Java* para desenvolvimento de aplicativos *Android.* Em 2017, durante o evento *Google I/O,* foi feito o anúncio que *Kotlin* é oficialmente a linguagem para desenvolvimento de aplicativos *Android*. Podemos definir Kotlin como uma linguagem de programação pragmática que combina os paradigmas de Orientação a Objetos e Programação Funcional (RESENDE, 2018).

GOOGLE MAPS API

O *Google Maps API* (Interface de Programação de Aplicações) nos permite integrar a funcionalidade de mapas geográficos no desenvolvimento de um aplicativo *Android*. Os métodos da *API* fornecem localização através do *GPS*, recuperando latitude e longitude que serão usados para recuperar a localização atual do motorista, além de permitir a interação do motorista com o mapa.

FIREBASE

O *Firebase* é uma plataforma de desenvolvimento de aplicativos que ajuda você a criar e desenvolver aplicativos e jogos que os usuários adoram. Apoiado pelo *Google* e confiável por milhões de empresas em todo o mundo. (FIREBASE, 2022).

Segundo *Moroney* (2017), o objetivo é fornecer as ferramentas e a infraestrutura de que você precisa para criar ótimos aplicativos. É um aprimoramento, oferecendo serviços comuns que você pode precisar, como um *back-end* de banco de dados, autenticação segura, mensagens e muito mais. Isso evita a necessidade de criá-los por conta própria, permitindo que você se concentre no que torna seu aplicativo distinto. Além disso, possui tecnologias que você pode colocar em seu aplicativo e site que o ajudarão a expandir seus negócios por meio de referências, links e muito mais.

3 METODOLOGIA

A pesquisa sobre o determinado problema realizado nos leva a colher dados para chegarmos a uma solução. O objetivo da pesquisa foi desenvolver um aplicativo para fazer o *check-in* do motorista no local do destino através da sua localização.

REQUISITOS FUNCIONAIS

ID	NOME	DESCRIÇÃO	PRIORIDADE
RF01	Mostrar o mapa	O aplicativo deverá mostrar um mapa ao usuário	Essencial
RF02	Mostrar localização	O aplicativo deverá mostrar a localização do usuário	Essencial
RF03	Mostrar posição	O aplicativo deverá mostrar posição do usuário ao realizar check-in	Essencial
RF04	Mostrar posições de terceiros	O aplicativo deverá mostrar as posições de outros usuários que realizaram check-in	Essencial
RF05	Check-in	O sistema deve permitir realizar check-in do usuário	Essencial
RF06	Check-out	O sistema deve permitir realizar check-out do usuário	Essencial
RF07	Administrador	O sistema deve ter um usuário administrador	Essencial
RF08	Administrador cadastrar/editar/deletar	O sistema deve permitir que somente o administrador cadastre, edite e delete os motoristas	Essencial

REQUISITOS NÃO FUNCIONAIS

ID	NOME	DESCRIÇÃO	PRIORIDADE
RNF01	Desempenho	O sistema deve garantir desempenho satisfatório, com o intuito de justificar sua utilização em detrimento ao atendimento a outras plataformas do mercado..	Essencial
RNF02	Fluidez	O sistema irá garantir a rápida transição entre as telas, a partir de todos os recursos disponíveis.	Essencial
RNF03	Resiliência	O sistema possibilitará ajustes com rapidez de forma periódica e ocasional, evitando perda de disponibilidade.	Importante
RNF04	Tempo de resposta	O sistema garantirá rápida consulta ao banco de dados e disponibilização de dados após autenticação, além de boa comunicação com servidores.	Desejável
RNF05	Segurança	O sistema deverá garantir autenticidade, disponibilidade, integridade e restrições de acesso.	Essencial
RNF06	Restrição de acesso	O sistema assegurará que cada tipo de usuário (administrador, motorista) possua acesso exclusivo às suas respectivas áreas.	Essencial
RNF07	Autenticidade	O sistema garantirá acesso exclusivo dos usuários aos seus respectivos perfis, impedindo ao máximo acesso falsos ou não autorizados.	Importante
RNF08	Internet	O sistema deverá funcionar apenas com acesso a internet.	Essencial
RNF09	Sistema operacional	O sistema irá rodar no sistema operacional Android, a partir da versão 7.0	Essencial
RNF10	Localização	O sistema deverá funcionar apenas com a permissão da localização do usuário.	Essencial

4 FUNDAMENTAÇÃO TEÓRICA

Inicialmente foram desenvolvidas as telas ou interface do usuário, para uma visão melhor do que está sendo aplicado.

FIGURA 1: Gradle - Dependência do Google Maps

```
//Maps
implementation 'com.google.android.gms:play-services-location:19.0.1'
implementation 'com.google.android.gms:play-services-maps:18.0.2'
```

Fonte: Próprio autor (2022)

Começando pela tela do mapa, foi necessário a inclusão da dependência do *Google Maps API* no gerenciador de dependências conhecido como *Gradle* no

Android Studio, assim como mostra a imagem acima.

FIGURA 2: Gradle - Dependência do Google Maps

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyAE9Hcl2UEcj7AOZZZzoPOIi_Gk1RnzSY0" />
```

Fonte: Próprio autor (2022)

Sendo necessário a criação do projeto através do console *developers* do google, onde é gerado uma chave, e através da chave é que temos acesso ao *Google Maps API*, essa chave deve ser integrada ao projeto do Android no arquivo Manifest (arquivo onde possuem os requisitos do aplicativo).

Desta forma, como vemos na imagem agora podemos acessar o *Google Maps* e definir regra através do *Android*.

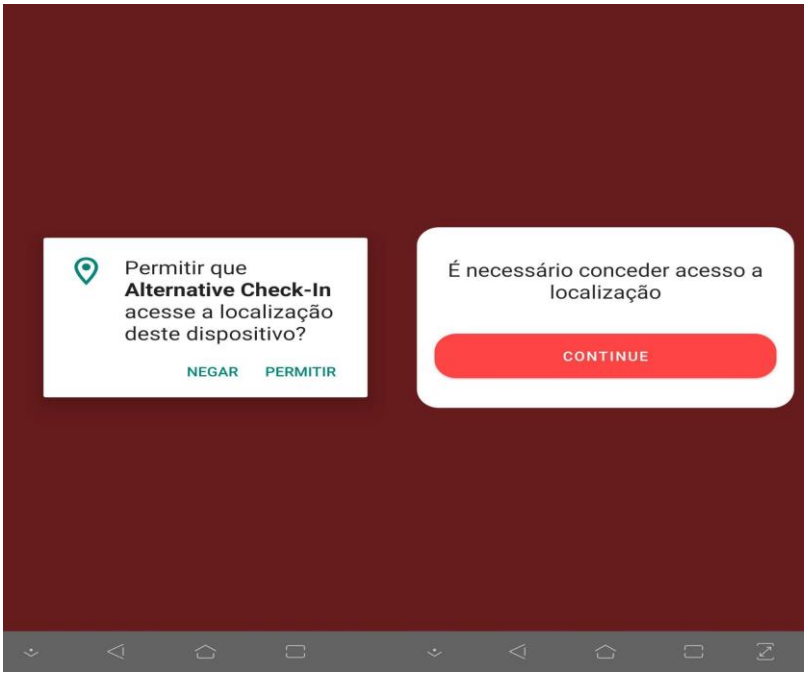
FIGURA 3: Arquivo Manifest - Permissões para obter localização

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Fonte: Próprio autor (2022)

Para obter a localização primeiramente precisamos obter a permissão do usuário para ter acesso a sua localização, também à permissão para acesso a internet, visto que o aplicativo precisará ter acesso à internet.

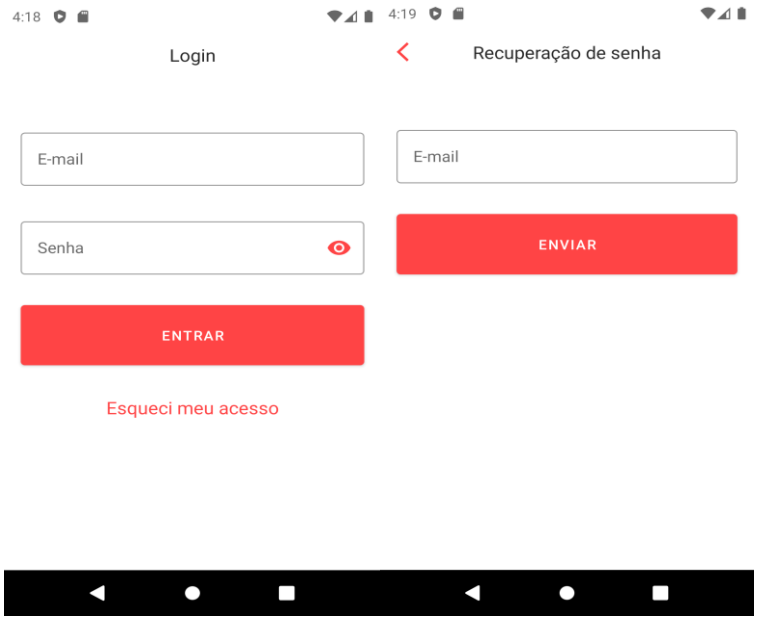
FIGURA 4: Dialog- Dialog para aceitar permissão



Fonte: Próprio autor (2022)

Na tela inicial solicitamos a permissão da localização, assim que o usuário permite, ele é direcionado para a tela de login, caso contrário é informado um *dialog* onde explica ao usuário que se faz necessário o acesso a sua localização.

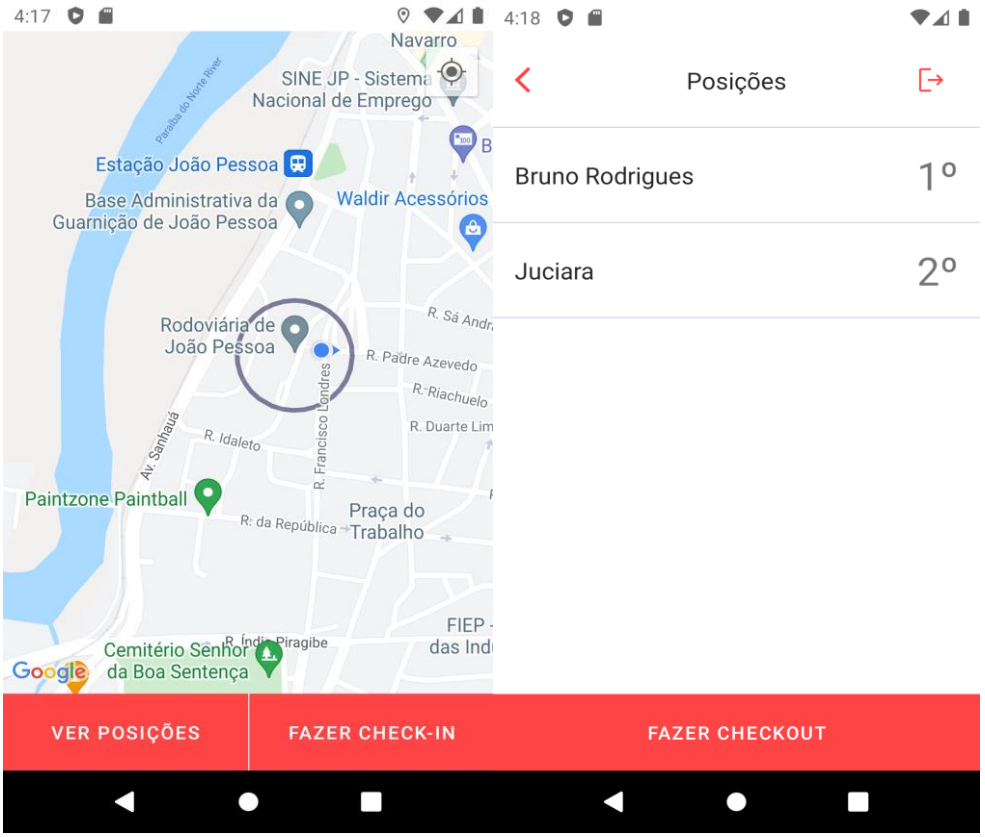
FIGURA 5: Telas - Tela de login e recuperação de senha



Fonte: Próprio autor (2022)

A tela de login do usuário é autenticado via Firebase Authentication, se caso o usuário não esteja cadastrado, é retornando uma mensagem avisando-o, senão o usuário é direcionado para tela de mapa.

FIGURA 6: Telas - Tela do Google Maps e Posições



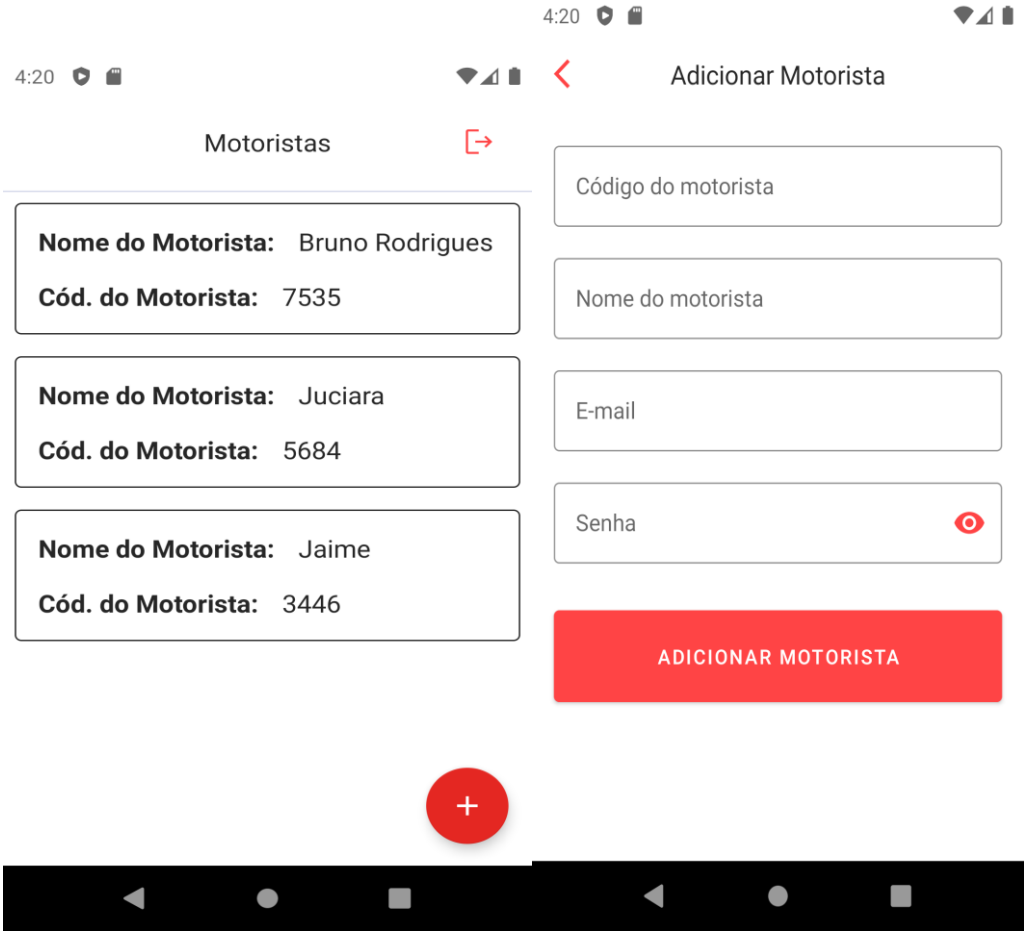
Fonte: Próprio autor (2022)

Na tela principal do motorista fica o mapa, onde mostra sua localização atual, e a opção de ver as posições de outros motoristas, o motorista só poderá fazer check-in na área determinada dentro do mapa, se ele não estiver naquela área, é exibido uma *dialog* informando, assim que ele faz o check-in, o motorista é direcionado para a tela de Posições, onde vai mostrar sua posição. Na tela de posição o usuário tem a opção de fazer o check-out, assim ele é retirado da lista.

Assim que o motorista faz o check-in, é enviado os dados para o Realtime Database do Firebase, uma ferramenta do Firebase onde mostra os dados em tempo real. Na tela de posições é onde pega esses dados enviados e lista cada motorista que fez o check-in. Quando o motorista faz o check-out, seu nome é excluído do database, sendo assim, a tela é atualizada e aquele motorista já não está mais na

lista.

FIGURA 7: Telas - Tela para adicionar um novo motorista

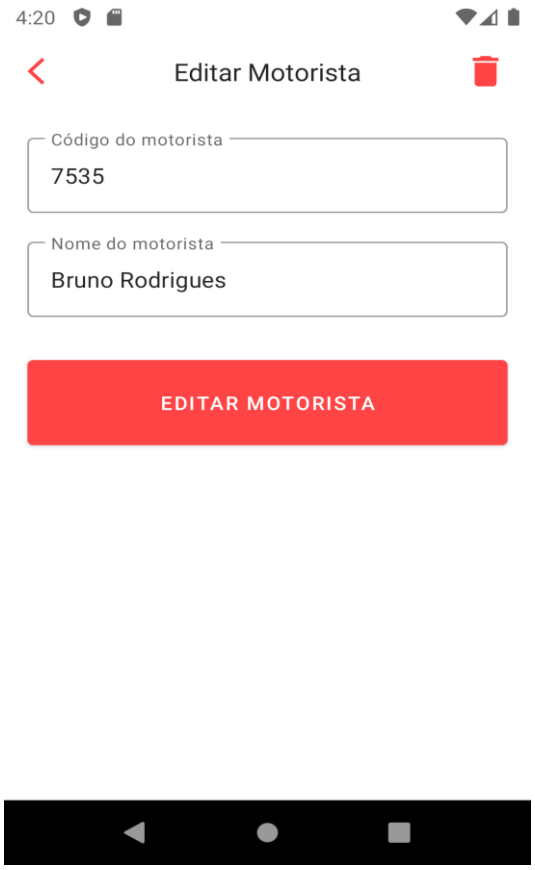


Fonte: Próprio autor (2022)

Por último temos a tela do administrador, onde tem a opção de cadastrar, editar e deletar um motorista. Foi um grande desafio já que tive que aprender como trabalhar com essas ferramentas do Firebase e fazer as integrações corretamente dos motoristas cadastrados. Na tela inicial, fica listado os motoristas que estão cadastrados,

No cadastramento, o usuário é integrado ao Realtime Database e Firebase Authentication, são feitas validações para os campos, o campo do código e nome, as validações são feitas no código, já o e-mail e a senha as validações são feitas através do Authentication, o mesmo retorna os possíveis erros e apenas listamos via código e transmitirmos para o administrador

FIGURA 8: Tela - Tela de editar motorista



Fonte: Próprio autor (2022)

Ao clicarmos em um motorista, é direcionado para a tela de edição, onde pode editá-lo como também deletá-lo.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo a criação de um aplicativo para transporte alternativo, utilizando o Android, Kotlin, Google Maps API e Firebase. Visto que, a forma como é feito o processo de incluir e verificar a posição do motorista, é de forma manual e que não garante uma forma eficaz e eficiente, foi necessário à criação do aplicativo, que garante que o motorista esteja no local, e garante a sua posição correta na lista, sem o devido esquecimento que poderia acontecer de forma manual.

O trabalho também fez com que o autor pudesse obter mais experiência na área, como também conhecimentos adquiridos ao decorrer do projeto, visto que foi necessário aprender algumas ferramentas, para que o aplicativo tivesse eficiência.

Sendo assim, foi possível aplicar os conhecimentos teóricos e tecnológicos adquiridos nas disciplinas do curso de sistemas para internet de uma forma integrada

para poder desenvolver o aplicativo proposto.

6 TRABALHOS FUTUROS

O aplicativo até então desenvolvido já consegue atender aos usuários, mas necessita de algumas mudanças futuras, como interface, novas implementações, separação em dois aplicativos, um para o administrador e outro para o motorista, relatórios de check-in, quantidades de motoristas, suporte, avaliações, expansão para outras cidades, já que a maioria possui o mesmo ponto na capital. Entre outras funcionalidades também que podem surgir através de pesquisa ao decorrer do tempo.

REFERÊNCIAS

ANDROID DEVELOPERS. 2021. Disponível em: <https://developer.android.com/studio/intro?hl=pt-br>. Acesso em 17 de Abril de 2022.

FIREBASE, Firebase. 2022. <https://firebase.google.com/?hl=pt>. Acesso em 30 de Abril de 2022.

LECHETA, Ricardo R. **Google Android**: Aprenda a criar aplicações para dispositivos móveis com o Android SDK. 5. ed. São Paulo: Novatec Editora Ltda, 2015.

MORONEY, Laurence. **The Definitive Guide to Firebase**: Build Android Apps on Google's Mobile Platform. Apress; 1st ed. edição, 2017.

RESENDE, Kassiano. **Kotlin com Android**: Crie aplicativos de maneira fácil e divertida. Brasil: Casa do Código, 2018.

TECMUNDO, 2021. **Pesquisa estima que metade da população mundial tem smartphones.** Disponível em: <https://www.tecmundo.com.br/mercado/220009-pesquisa-estima-metade-populacao-mundial-tem-smartphones.htm#:~:text=A%20empresa%20de%20pesquisa%20e,cerca%20de%20R%242%20 mil.> Acesso em 4 de Março de 2022.

SISTEMA DE CAPTAÇÃO DE LEADS PARA VENDAS DE CURSOS DE CAPACITAÇÃO CONTÁBIL EM ESCRITÓRIOS DE CONTABILIDADE.

FILHO, MARCOS ANTONIO ARAUJO MARQUES
BATISTA, MESSIAS RAFAEL

RESUMO

A presente pesquisa tem como objetivo discutir a relevância de um sistema de captação de *leads* para potencializar as vendas de cursos de formação em contabilidade. Nota-se uma grande necessidade no setor capacitação contábil devido ao grande número de pessoas que entram no mercado sem conhecimento específico em contabilidade, assim nota-se um potencial enorme no segmento de vendas de cursos. Então, para potencializar as vendas foi analisado na pesquisa a importância de sistemas eficazes a fim de oferecer o produto ao cliente e fazer a captação dos dados dos clientes para realizar as vendas dos cursos. Por isso, a pesquisa foi realizada com o objetivo de apresentar a importância de um sistema para esse segmento. Assim, foi discutido as principais tecnologias de informação como: *HTML, CSS e JavaScript* para o desenvolvimento do site. Portanto, conclui-se que é de extrema importância o investimento em um sistema eficaz para alcançar resultados de excelência no mercado de venda de cursos.

Palavras-chave: Sistemas; *leads*; cursos; vendas; cliente; *HTML; CSS; JavaScript.*

1 INTRODUÇÃO

Hoje em dia, o avanço da tecnologia de informação tem gerado melhorias em relação a captação de dados de clientes a fim de oferecer vendas de produtos e serviços. Dessa forma, entende-se que a informatização se tem desenvolvido de forma exponencial e isso tem gerado excelentes resultados para os empreendedores nos mais diversos ramos. Assim, o objetivo desse sistema é oferecer uma plataforma para o cliente inserir seus dados afim de comprar o curso de capacitação contábil para capacitação e treinamento dos funcionários dos escritórios de contabilidade.

Nesse contexto, percebe-se que o sistema de captação de dados dos clientes (*LEADS*) será primordial para estreitar o relacionamento com o cliente a fim de realizar a venda do curso. Dessa forma, o sistema proporcionará um melhor relacionamento com o cliente. Para desenvolvimento do site, utilizaremos tecnologias modernas, como: *Javascript* e *node JS*. Para salvar os dados dos clientes interessados na compra do curso, utilizaremos o banco de dados *postgress. De*ssa forma, utilizando essas tecnologias, acredita-se ser capaz de desenvolver um sistema simples e eficiente, para alcançar o objetivo proposto.

Neste interim, a presente pesquisa visa discutir a relevância de um sistema de

captação de *leads* para realizar vendas de cursos nos escritórios de contabilidade.

Para o desenvolvimento desta pesquisa, foram realizadas leituras sobre os conceitos de tecnologia de informação e sistemas de captação de dados visando criar a fundamentação teórica para a pesquisa a ser realizada sobre sistemas de captação de *leads* para realizar vendas dos cursos.

Portanto, realizar a presente pesquisa com essa magnitude possibilitará uma reflexão a respeito da importância das tecnologias de informação no segmento de captação de leads e o quanto esta ferramenta pode otimizar o processo de vendas dos cursos, assim como irá proporcionar um melhor relacionamento com o cliente.

Dessa forma, acredita-se que é bastante relevante essa discussão, tendo em vista o crescimento da tecnologia nos últimos anos, o quanto a sociedade tem investido em sistemas e as vantagens que as tecnologias de informação podem proporcionar para os segmentos de vendas.

2 METODOLOGIA

A presente pesquisa caracteriza-se como bibliográfica, de natureza explicativa, a fim de obter informações suficientes para responder a problemática elaborada, visando uma abordagem objetiva e clara para os leitores.

De acordo com Santos e Filho (2012) independente do campo a ser pesquisado, é necessária uma pesquisa bibliográfica, pois proporciona um conhecimento prévio do assunto aos leitores.

Ainda segundo Santos e Filho (2012) a pesquisa bibliográfica é baseada em informações já escritas em livros, jornais, revistas e entre outros.

A presente pesquisa requer demanda aos pesquisadores de informações sobre os sujeitos da pesquisa, percebe-se que a pesquisa bibliográfica proporciona aos leitores um crescimento considerável do conhecimento de determinado assunto a ser discutido na pesquisa, onde se utiliza-se de livros e revistas para elaboração das informações presentes na mesma.

Segundo SANTOS e FILHO (2012) pesquisar fatos da atualidade que ainda não foram publicados em forma de livros, revistas, jornais entre outros e de extrema importância para o pesquisador.

Portanto, entende-se que esta pesquisa necessita de informações atuais do mercado e que a tecnologia de informação tem gerado resultados relevantes para o

segmento de venda de cursos, assim como um ótimo relacionamento com os clientes.

3 FUNDAMENTAÇÃO TEÓRICA

Nesse espaço, iremos tratar da importância da tecnologia da informação, do software e as tecnologias *HTML*, *CSS* e *JavaScript* utilizadas para o desenvolvimento do sistema. Nesse contexto, serão apresentadas algumas telas de funcionamento do sistema, assim como telas de desenvolvimento do código e suas respectivas funcionalidades.

3.1 TECNOLOGIA DE INFORMAÇÃO

De acordo Ramos (2010) a tecnologia da informação é uma ferramenta capaz de proporcionar aos usuários, sistemas de informações afim de atender as suas necessidades no sentido de auxiliar na gestão através solicitações de demandas e respostas de relatórios e serviços que podem auxiliar no processo de tomada de decisão de determinada execução de um problema.

Nesse sentido, entende-se que as tecnologias têm grande relevância a fim de ajudar os usuários conhecerem o produto e se cadastrarem para fins de aquisição dos cursos de capacitação contábil.

O sistema proporciona ao usuário maior velocidade no atendimento e processamento dos dados de acordo com a demanda do cliente. Assim, entende-se que se faz necessário os sistemas para captação dos dados dos clientes a fim de oferecer o produto que ele tem interesse em adquirir.

Dessa maneira, percebe-se que os sistemas são fundamentais para a venda dos cursos de contabilidade no sentido de captar os contatos dos clientes e oferecer os produtos, fazendo com que otimizem as vendas ao fim do exercício.

Acredita-se também que para obter sucesso com as tecnologias de informação através de sistemas de software, a qualidade técnica do sistema é condição necessária para alcançar resultados positivos.

Nesse sentido, vai depender de fatores como: o desenvolvimento do sistema, o preenchimento de dados corretos pelos usuários, bem como a gestão das informações geradas através dos usuários no sistema e o relacionamento com o cliente de forma excepcional para realização da venda dos cursos de contabilidade.

Assim, entende-se que é de extrema importância o investimento no desenvolvimento do sistema para criar um sistema eficiente a fim de obter resultados positivos e aumentar as vendas de cursos no decorrer do exercício através da ferramenta de captação de dados dos clientes.

3.2 SOFTWARE

De acordo com Roger e Brune (2016) Software de computador é o produto que profissionais da tecnologia de informação desenvolvem e dão suporte. Assim, entende-se que é necessário a manutenção do sistema a longo prazo para dar continuidade ao funcionamento do mesmo e evitar bugs e erros de operacionalização do software. Os sistemas englobam programas executáveis em um computador para solucionar um problema especifico ou genérico.

Segundo Hirama (2011) software desempenha um papel fundamental em várias áreas de negócios seja para controle e segurança de informações, controle de processos, apoio a decisões, entretenimento e entre outras necessidades dos clientes.

Nesse interim, há 40 anos atrás, o software já começava a se tornar complexo. Problemas de qualidade e atendimento de prazos e custos já eram apontados. Assim, nasceu a engenharia de Software tratando o software como um produto de engenharia para fazer frente aos novos desafios e proporcionar aos usuários soluções inovadoras para os problemas complexos de seus sistemas (Hirama, 2011).

As atividades de desenvolvimento de software abrangem essencialmente atividades técnicas de Engenharia de Sistemas, análise, projeto, codificação e testes.

Leciona Hirama (2011) que a atividade de Engenharia de Sistemas tem por objetivo entender as necessidades de negócio do cliente e especificar os requisitos do sistema computacional, incluindo-se os requisitos do software em alto nível.

A atividade de Análise busca entender os requisitos do sistema e detalhar os requisitos do software. A atividade de Projeto tem por objetivo estabelecer uma arquitetura do software que seja executável. A atividade de Codificação pretende traduzir as especificações de software em códigos de programa que sejam processados por um sistema computacional. A atividade de Testes tem por objetivo descobrir defeitos no software, considerando aspectos estruturais e funcionais do software.

Portanto, percebe-se que o software é uma ferramenta de extrema importância

no contexto de venda de cursos de contabilidade como um dispositivo de otimização nos processos de venda, no cadastramento do cliente e captação de dados dos clientes para realizar as vendas dos cursos de capacitação contábil.

3.3 HTML

De acordo com Joel (2019) HTML significa *hypertext Markup language*. Essa tecnologia é uma linguagem de marcação na estrutura de uma página web. Tem característica de linguagem simples, baseada em *tags* que representam elementos, como por exemplo imagens, links e entre outros.

Segue abaixo a estrutura básica de um HTML com as principais tags de marcação da página web.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8"/>
5  <title>Document</title>
6  </head>
7  <body>
8  <!-- Conteúdo -->
9  </body>
10 </html>
```

Figura 1 – Estrutura do *HTML*
Fonte: Próprio autor (2022)

Segundo Joel (2019) DOCTYPE sempre aparece primeiro na estrutura do *HTML*, ele é responsável por indicar qual linguagem será utilizada. A *tag html* irá delimitar o documento geral, ou seja, todas as *tags* da página irão estar dentro dela. O *head* irá definir o cabeçalho do documento, porém esse conteúdo não será visível no browser. A *tag* meta define qual é o conjunto de caracteres (*character* set ou *charset*) será utilizado para renderizar a página web. A *tag title* define qual será o título da página. Na *tag body* marca todo o espaço onde será contido os códigos para apresentarem o conteúdo visual da página.

Nesse contexto, para desenvolver uma página em *HTML* pode-se utilizar aplicativos de edição de texto como: bloco de notas no Windows, *sublime text* e *visual*

studio code que são bastante utilizados no mercado do desenvolvimento de sistemas.

3.4 CSS

Ricardo (2016) diz que *CSS* (*cascating style sheet)* significa folha de estilo em cascata. Essa tecnologia é responsável por estilizar uma página *HTML,* ou seja, ela trabalha a aparência de uma página web. Cabe ressaltar que CSS não é uma linguagem de programação, porém ela é aplicada ao *HTML* para dar estilo a página web.

O CSS será responsável por fazer estilizações como por exemplo: colocar palavras em negrito, centralizar textos, inserir cores de fundos na página, alterar fontes e diversas outras estilizações.

Segundo Ricardo (2016) existem maneiras distintas de inserir um código css na sua página web. Sendo elas: CSS INLINE, CSS na mesma página e CSS externo.

CSS INLINE é raramente utilizado em paginas web porque ele deixa o código com muita informação, causando poluição no código. Sua utilização é bastante simples, uma vez que é só inserir sua o *style* na *tag* que irá inserir o estilo, como veremos na imagem abaixo:

```
<h1 style="font-style: italic;">Sou a tag h1</h1>
```

Figura 2: Demonstração de aplicação do *CSS INLINE*
Fonte: Próprio autor (2022)

De acordo com Ricardo (2016) *CSS* na mesma página ela tem uma semelhança da *CSS INLINE* pois ambas são executadas na mesma página. Contudo, a diferença é que a *CSS* na mesma página é aplicada de forma global, dessa forma evita-se que o *style* seja aplicado dentro da *tag*, como veremos na imagem abaixo:

```
<html>
<head>
  <title>Código CSS - Entendendo a folha de estilos</title>
<style type="text/css">
h1{
  font-style: italic;
}
p{
  color: red;
}
</style>
</head>
<body>
<h1>Sou a tag h1</h1>
```

Figura 3 – Demonstração de aplicação do CSS na mesma página
Fonte: Próprio autor (2022)

O *CSS* externo é considerado o tipo mais utilizado para estilização de paginas web, pois ela consiste em inserir todo o código CSS em um arquivo externo .css e no HTML iremos chamar o arquivo para que o código seja estilizado. Para fazer essa chamada devemos utilizar o código abaixo:

```
<link rel="stylesheet" type="text/css" href="aqui entra o
link do seu arquivo externo">
```

Figura 4 – Demonstração do CSS externo
Fonte: Próprio autor (2022)

Após inserir o código de chamada do *css* apresentada acima, cria-se um arquivo com o nome de estilo.css para dentro desse arquivo estilizar as *tags* da forma que lhe for conveniente, como veremos na imagem abaixo:

```
h1{
  font-style: italic;
}
#fundo{
  background-color: #a3a9fa;
}
.paragrafo{
  color: red;
}
.paragrafo2{
  color: #0018ff;
}
```

Figura 5 – Demonstração do CSS externo
Fonte: Próprio autor (2022)

Nesse contexto, entende-se que através do *CSS* externo é possível ter uma melhor organização do código e maior facilidade de manutenção da página web, por esses motivos ela é mais utilizada no mercado do que os outros tipos citados anteriormente.

3.5 *JAVASCRIPT*

De acordo com Jailson (2012) é importante destacar que a linguagem de programação *javascript* nada tem a ver com a linguagem de programação *java*, pois existe uma grande discussão pela comunidade de programadores iniciantes e confundem bastante sobre esse quesito.

Segundo Jailson (2012) a definição mais importante do *javascript* é que ela é uma linguagem de programação *client side*, ou seja, ela é executada no computador do usuário. Assim, entende-se que o fato de um computador receber um código e interpretar, isso quer dizer que é uma linguagem do tipo *client side*.

Nesse contexto, quando o usuário acessa o browser, a partir daí haverá uma comunicação entre o servidor e o usuário solicitando para que seja enviado os arquivos solicitados. Com isso, o servidor interpreta a solicitação e retorna a página solicitada em formato de texto, contendo código *HTML* e *javaScript* incorporados na página. Ao acessar uma página web, existe um fluxo de eventos que ocorrem entre o servidor que está página e o nosso computador. O qual será ilustrado na figura abaixo.

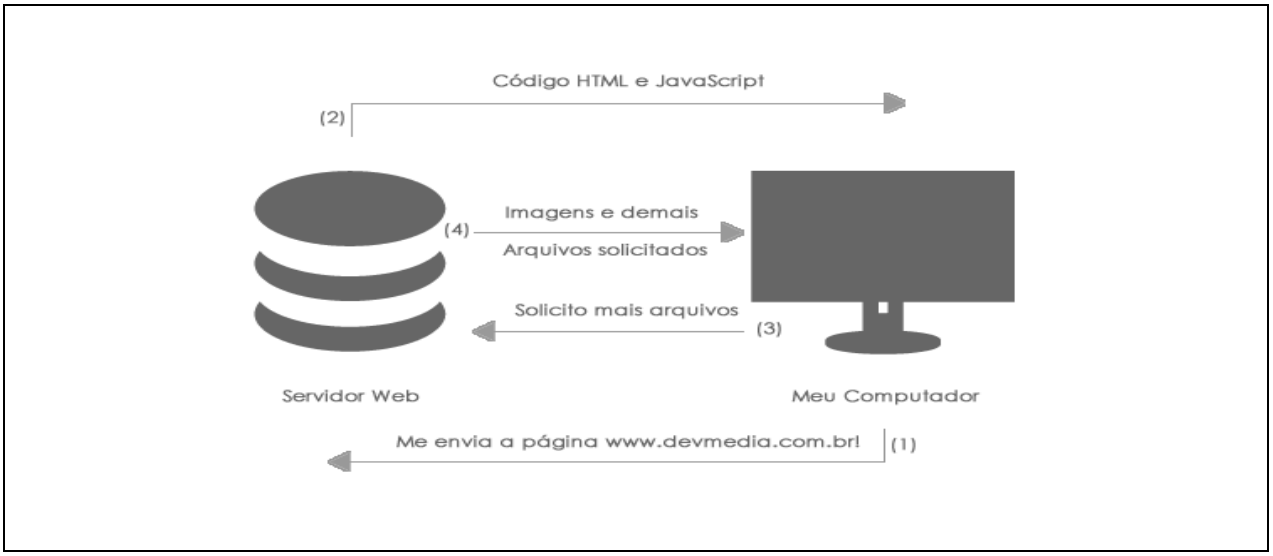


Figura 6 – Demonstração de fluxos de eventos utilizando *Javascript.*
Fonte: Jailson (2012)

Portanto, conclui-se que a linguagem *Javascript* é uma linguagem do tipo *client* porque é interpretada pelo browser ou navegador e gera os resultados pelo próprio computador do usuário.

Jailson (2012) diz que a linguem Javascript consegue se relacionar com praticamente todo o *HTML*, para trabalhar com variáveis, gerar resultados, alterar a imagem de elementos na página sem ficar recarregando a página. Existem aplicativos desenvolvidos totalmente com *Javascript* e que essa linguagem é cada vez mais comum no mundo do desenvolvimento e vem evoluindo cada vez mais com o passar do tempo.

Existem ferramentas específicas para desenvolver com a linguagem *Javascript*, mas iremos mostrar como funciona para criar um *script* simples na figura abaixo:

```
<html>
  <head>
    <title>Artigo 01 - Olá World!</title>
    <script language="javascript" type="text/javascript">
      alert('Olá World!');
    </script>
  </head>
  <body>
  </body>
</html>
```

Figura 7 – Demonstração de Script
Fonte: Próprio autor (2022)

Portanto, o script é necessário para desenvolvermos com a tecnologia Javascript, porque o HTML só irá entender o comando com a *tag* de script.

3.6 DESENVOLVIMENTO

O sistema consiste em uma página web, com o objetivo de proporcionar ao nosso cliente a apresentação da nossa empresa e os cursos que serão ofertados. O site irá dispor de informações da empresa como: contatos, quem somos e os cursos ofertados. Acredita-se que através do site os usuários terão maior facilidade para conhecer a empresa e os nossos cursos, com isso irá potencializar a credibilidade da empresa para com os clientes e, consequentemente, iremos otimizar as vendas dos cursos.

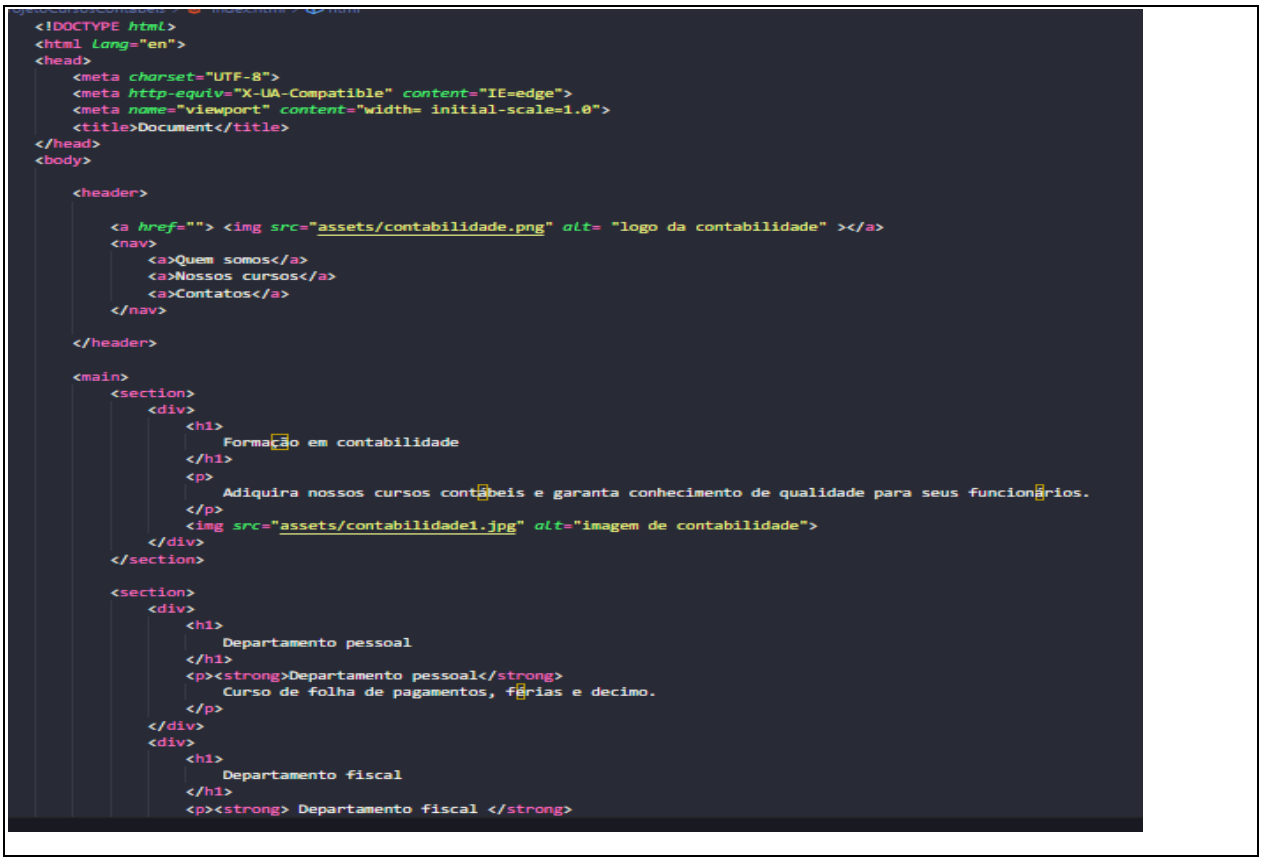
No site os clientes poderão ver informações sobre o que cada cursos irá

trabalhar, assim caso o cliente tenha interesse em comprar algum curso, ele prontamente poderá fazer um cadastro simples e rápido. Assim, após concluir o seu cadastro, automaticamente os seus dados irão para um banco de dados que serão repassados para a equipe de vendedores entrarem em contato com os clientes com maior agilidade e eficiência para efetuar a venda dos cursos.

Acreditamos que através do sistema a nossa empresa irá alcançar resultados excelentes de vendas no período. Portanto, o desenvolvimento desse sistema se faz de extrema importância.

Para o desenvolvimento da página web foram utilizadas as tecnologias *HTML, CSS, JavaScript, Node Js* e o banco de dados *postgress*.

Segue abaixo as imagens do HTML onde fizemos toda a marcação da página.



```
<!DOCTYPE html>
<html Lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width= initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <header>

        <a href=""> <img src="assets/contabilidade.png" alt= "logo da contabilidade" ></a>
        <nav>
            <a>Quem somos</a>
            <a>Nossos cursos</a>
            <a>Contatos</a>
        </nav>

    </header>

    <main>
        <section>
            <div>
                <h1>
                    Formação em contabilidade
                </h1>
                <p>
                    Adiquira nossos cursos contábeis e garanta conhecimento de qualidade para seus funcionários.
                </p>
                <img src="assets/contabilidade1.jpg" alt="imagem de contabilidade">
            </div>
        </section>

        <section>
            <div>
                <h1>
                    Departamento pessoal
                </h1>
                <p><strong>Departamento pessoal</strong>
                    Curso de folha de pagamentos, férias e decimo.
                </p>
            </div>
            <div>
                <h1>
                    Departamento fiscal
                </h1>
                <p><strong> Departamento fiscal </strong>
```

Figura 8 – Demonstração do HTML
Fonte: próprio autor (2022)

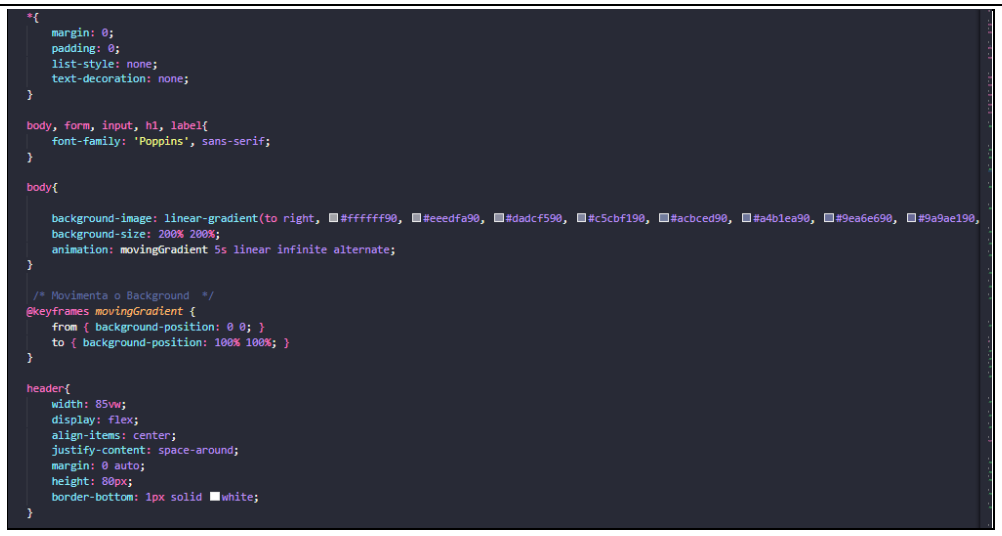
Na primeira parte do código em HTML, foi feita a marcação do início da página onde contém duas imagens e também a proposta da nossa empresa.

```
                <h1>
                    Departamento pessoal
                </h1>
                <p><strong>Departamento pessoal</strong>
                    Curso de folha de pagamentos, férias e decimo.
                </p>
            </div>
            <div>
                <h1>
                    Departamento fiscal
                </h1>
                <p><strong> Departamento fiscal </strong>
                    Curso de tributação fiscal.
                </p>
            </div>
            <div>
                <h1>
                    Departamento contábil
                </h1>
                <p><strong>Departamento contábil</strong>
                Curso de contabilidade responsável por elaborar balancetes, razonetes, balancos e demonstrações de resultados.

                </p>
            </div>
        </section>
    </main>

    <footer>
        <div>
            <h2>Quem somos</h2>
            <p>
                Nossa empresa de capacitação contábil...
            </p>
        </div>

        <div>
            <h2>Contatos</h2>
            <ul>
                <li>
                    (83)987521152
                </li>
                <li>
                    (83) 3034-6364
                </li>
                <li>
                    Email:marcoscursosontabeis@gmail.com
                </li>

            </ul>
        </div>

    </footer>
        <div>

        </div>

</body>
</html>
```

Figura 9 – Demonstração do HTML
Fonte: Próprio autor (2022)

Na segunda parte do código foi feito o restante da marcação da página, onde colocamos informações relevantes dos cursos ofertados no site, assim como contatos e quem somos, a fim de proporcionar informações relevantes sobre nossa empresa e a proposta em vender cursos de qualidade para nosso cliente.

Segue abaixo a imagem que demonstra a parte da estilização da página com a tecnologia CSS.



```
*{
    margin: 0;
    padding: 0;
    list-style: none;
    text-decoration: none;
}

body, form, input, h1, label{
    font-family: 'Poppins', sans-serif;
}

body{

    background-image: linear-gradient(to right, #ffffff90, #eeedfa90, #dadcf590, #c5cbf190, #acbced90, #a4b1ea90, #9ea6e690, #9a9ae190,
    background-size: 200% 200%;
    animation: movingGradient 5s linear infinite alternate;
}

 /* Movimenta o Background  */
@keyframes movingGradient {
    from { background-position: 0 0; }
    to { background-position: 100% 100%; }
}

header{
    width: 85vw;
    display: flex;
    align-items: center;
    justify-content: space-around;
    margin: 0 auto;
    height: 80px;
    border-bottom: 1px solid white;
}
```

Figura 10 – Demonstração do CSS
Fonte: Próprio autor (2022)

Na primeira parte da estilização foi trabalhado as fontes da página, assim como

o *background* do *body* com as cores. Foi utilizado uma propriedade muito bacana chamada *movinggradient,* onde o *background* fica se movimentando e variando as cores.

```css
form {
  flex-direction: column;
  display: flex;
  flex-wrap: wrap;
}

label {
  font-weight: bold;
  font-size: .8rem;
}

input {
  width: 95%;
  border-bottom: 2px solid #323232;
  padding: 10px;
  font-size: .9rem;
  margin-bottom: 15px;
  display: flex;
  justify-content: flex-end;
}

input:focus {
  border-color: #1700e6;
}

input[type="submit"] {
  justify-content: center;
  display: flex;
  background-color: #1700e6;
  color: #FFF;
  border: none;
  border-radius: 20px;
```

Figura 11 – Demonstração do CSS
Fonte: Próprio autor (2022)

Nessa segunda parte do código foi criado um formulário para credenciamento dos usuários interessados em comprar os cursos no nosso site, assim foi estilizado as fontes, bordas, tamanhos, cores, margem e etc.

Para finalizar o desenvolvimento da página será utilizada a linguagem *JavaScript onde será* feita integração da página web com o servidor e o banco de dados para armazenar os dados dos clientes. Para integrar o *HTML e CSS* foi utilizado a biblioteca *REACT.*

4 CONSIDERAÇÕES FINAIS

A presente pesquisa teve como objetivo discutir a relevância de um sistema de captação de *LEADS* para realizar vendas de cursos de contabilidade para escritórios de contabilidade.

Nesse contexto, entende-se que no mercado atual os escritórios contábeis necessitam cada vez mais de capacitar e treinar seus funcionários para prestarem um melhor serviço para seus clientes. Assim, a presente pesquisa constatou a importância de um sistema eficaz afim de coletar os dados dos clientes para ter um melhor relacionamento com o cliente e consequentemente fechamento de venda dos cursos.

Nesse interim, no cenário atual os escritórios de contabilidade têm muita demanda de serviços, e cada vez mais está inserindo novos profissionais no mercado sem experiencia, dessa maneira entende-se que é de fundamental importância o investimento por parte dos escritórios em treinamento e capacitação desses funcionários.

Nesse contexto, cabe ressaltar os resultados positivos dos escritórios contábeis que investem em capacitação dos seus funcionários e a satisfação dos seus clientes na prestação dos seus serviços.

Nota-se que o fator de vendas de cursos na web tem crescido bastante no cenário atual e que é de extrema importância para os empresários se atualizarem nesse sentido.

Portanto, nosso sistema apresenta uma tela de boas-vindas, apresentando ao nosso cliente a proposta e os cursos da nossa empresa, assim os usuários interessados nos cursos fazem seu cadastro com a finalidade de comprar os cursos de contabilidade. Em contrapartida a nossa empresa vai captar os dados dos clientes e prontamente irá entrar em contato com os mesmos através de *WhatsApp*, ligação ou e-mail.

Acredita-se que um melhor relacionamento com o cliente é de extrema importância para fechamento de vendas e otimização dos resultados no exercício, por esse motivo entende-se que o sistema será excelente para oferecer essa solução para o empresário.

Portanto, conclui-se que o nosso sistema é de extrema importância para otimizar o relacionamento com o cliente, para ofertar o curso de forma clara e objetiva, proporcionar o melhor serviço para o cliente comprar e total segurança dos dados de quem compra, sendo assim uma excelente ferramenta para otimização das vendas no segmento de cursos de capacitação contábil.

REFERENCIAS

Farah, Osvaldo Elias, et al. *Empreendedorismo estratégico: criação e gestão de pequenas empresas*. Cengage Learning Brasil, 2017.

Hirama, K. **Engenharia de Software**. São Paulo: Grupo GEN, 2011. 9788595155404. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595155404/.

Jailson, DEV MEDIA, INTRODUÇÃO AO JAVASCRIPT. Acesso em: 28 de abril de 2022, Disponível em: JavaScript: Uma introdução ao completa ao JavaScript (devmedia.com.br) 2012.

Joel, DEV MEDIA, HTML básico – códigos HTML. Acesso em: 29 de abril de 2022, Disponível em: HTML básico: Códigos HTML para Iniciantes (devmedia.com.br) 2019.

Masiero, Gilmar. *ADMINISTRAÇÃO DE EMPRESAS*. Editora Saraiva, 2012.

Ramos, M. C. *Gestão de Tecnologia da Informação - Governança de TI: Arquitetura e Alinhamento entre Sistemas de Informação e o Negócio.* Rio de Janeiro*;* Grupo GEN*, 2010.* 978-85-216-1972-7. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/978-85-216-1972-7/. Acesso em: 08 de Outubro de 2020.

Ricardo, DEV MEDIA, Código CSS: entendendo a folha de estilos. Acesso em: 30 de abril, Disponível em: Código CSS: entendendo a folha de estilos (devmedia.com.br) , 2016.

ROGER, P.; BRUCE, M. **Engenharia de Software**. Porto Alegre: Grupo A, 2016. 9788580555349. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788580555349/.

SANTOS, João Almeida; FILHO, Domingos Parra. Metodologia CIENTIFICA 2ª edição. CENGAGE Learning, 2012.

API PARA GERENCIAMENTO DE *FOOD SERVICES* UTILIZANDO JAVA ESPRING BOOT

SENA, Eduardo
BATISTA, Messias Rafael

RESUMO

A utilização de um sistema de gerenciamento para um estabelecimento de serviços alimentícios vem cada vez mais ganhando espaço no mercado como um meio de alavancar as vendas e otimizar os processos. Com isso em mente este artigo trata da produção de uma *API* que atuará como aplicação *backend* base para a construção de sistemas *Food Services* completos. Os processos de implementação foram feitos utilizando um modelo de domínio da aplicação e análise dos requisitos, a etapa de desenvolvimento se deu por meio da construção da *API* utilizando a linguagem Java e o framework *SpringBoot* e a documentação foi feita com o auxilio da biblioteca do *Swagger.*

Palavras-chave: Food Service; Spring Boot; API Rest; Java

1 INTRODUÇÃO

Os *food services*, como restaurantes, lanchonetes e bares são utilizados a muito tempo em nossa sociedade, poder fugir da rotina de preparar a refeição em casae ter a possibilidade de comer algo diferente em um local diferente do habitual é algoque todo mundo experimenta vez ou outra, o Instituto Food Service Brasil (IFB) constatou que no ano de 2018, 37% da população brasileira realiza suas refeições fora de casa (BUYCO, 2020).

O início da pandemia do coronavírus trouxe uma crise muito grande para diversos setores e entre os afetados estão os serviços de comida, que até então necessitavam do público presente para realizar o consumo dos seus produtos, e como isolamento social é um dos meios de combate para evitar a proliferação da infecção muitos desses estabelecimentos fecharam as portas, felizmente esse setor começa a mostrar sinais de recuperação apresentando expectativas positivas para o mercado, segundo a Associação Brasileira de Bares e Restaurantes (ABRASEL) o faturamento do ano de 2021 para esse mercado é estimado em R$ 215 bilhões enquanto o de 2020 foi de apenas R$ 175 bilhões (OIMENU, 2022).

Só que a pandemia não trouxe apenas desastre para essa área de negócio, para se manterem muitos empresários optaram por adotar o sistema de *delivery*, de acordo com a Associação Brasileira de Indústria de Alimentos (ABIA) em 2020 o setor de restaurantes *delivery* teve um crescimento de 150%, e para isso dar certo se fez necessário o auxílio da tecnologia (OIMENU, 2022)

Neste artigo iremos apresentar a construção de uma API que auxilie na gestãode um *food service* que servirá para facilitar o acesso às informações dos produtos oferecidos, proporcionando agilidade e organização no negócio, será construída uma aplicação *web backend* com as tecnologias Java e *Spring Boot* e utilizaremos o bancode dados relacional MySQL para armazenamentos dos dados.

2 FUNDAMENTAÇÃO TEÓRICA

Neste tópico iremos discutir os aspectos gerais da linguagem de programação Java e do framework *SpringBoot*, que são as tecnologias utilizadas para a construçãoda *API*.

JAVA

O Java é uma linguagem de programação orientada a objetos que começou a ser criada em 1991 pela *Sun Microsystems*, inicialmente recebeu o nome de *Oak* inspirada em uma arvore que tinha próxima de onde trabalhavam, e tinha como objetivo fazer com que os computadores e eletrônicos trabalhassem em convergência (LUCKOW e DE MELO, 2010).

Figura 1 – Aparelho StarSeven
Fonte: FECCHIO (2006)

A primeira implementação do projeto *Oak* foi um controle remoto nomeado *StarSeven* (figura 1), que possuía uma interface *touchscreen* onde ensinava aos

usuários como utilizar o controle, mas a tecnologia da época não avançou tanto comoos idealizadores haviam previsto (PACIEVITCH, 2022).

No entanto, nesta época, a internet vinha mostrando avanços e sua popularidade estava crescendo, e foi aí que os desenvolvedores da *Sun Microsystems* começaram a pensar em aplicações do *Oak* voltada para a internet, e em 1995 o Java foi lançado como uma adaptação do *Oak* para internet (LUCKOW eDE MELO, 2010).

O Java rapidamente se tornou uma das principais linguagens de programação e a grande diferença entre o Java e as outras linguagens da época é que por ser executado sobre uma máquina virtual (JVM ou *Java Virtual Machine*), qualquer dispositivo que pudesse executar uma *virtual machine conseguiria também rodar o java (*LUCKOW e DE MELO, 2010).

Java Virtual Machine

A *java virtual machine* ou JVM é capaz de realizar a execução de aplicações java, é graças a ela que os sistemas em java podem funcionar em qualquer plataformaque tem uma JVM instalada *(*LUCKOW e DE MELO, 2010).

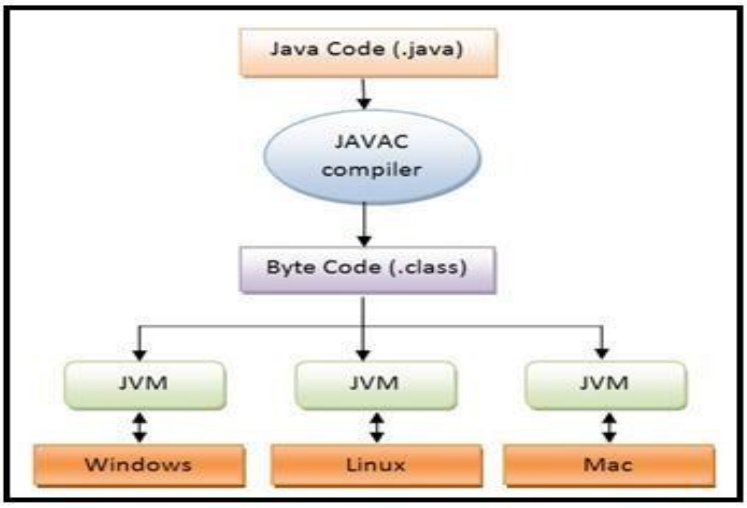


Figura 2 - Esquema de funcionamento de uma JVM
Fonte: ROMANATO (2022)

A *JVM* não entende o código java em si, ela interpreta um código em Bytecodeque é gerado pelo compilador java, e é a partir disso que o código é traduzido para amáquina de destino conforme indicado na figura 2 *(ROMANATO, 2013).*

Características

Hoje pode-se dizer que o java é uma das principais linguagens de programação existentes no mundo da tecnologia, sendo executado em bilhões de dispositivos emtodo o mundo, sendo eles mais de 850 milhões de computadores (OLIVEIRA, 2012).

O java apresenta algumas características bem atraentes que fazem com que ela seja diferenciada de outras plataformas, a primeira é que o java atende todas as características de uma linguagem orientada a objetos, segundo é que o java tem uma independência de plataformas, por rodas em uma máquina virtual (JVM) que pega o código em *bytecode* do java e interpreta ele para plataforma que será executado a aplicação, por fim temos que por ser capaz de converter os bytecode em códigos nativos no sistema *Just in Time* o java possui um desempenho semelhante aos programas obtidos a partir de códigos nativos (OLIVEIRA, 2012).

SPRING BOOT

O *Spring Boot* é um framework que nasceu a partir do Spring e veio para facilitar o processo de construção de aplicações *backend*, facilitando a configuração inicial de um projeto, ele utiliza vários padrões e recursos que reduzem bastante a complexidade e o tempo de desenvolvimento (SILVA, 2019).

O *Spring Boot* possui três componentes principais que trabalham por baixo dos panos, são eles: *Spring Boot Starter* que possui inicializadores que unificam diversas dependências e configurações dentro de uma única dependência*, Spring Boot Autoconfigurator que é responsável por dar as coordenadas de configuração para a aplicação, é ele que une também as configurações padrão e as desenvolvidas na aplicação e por fim o Spring Boot Actuator que monitora a saúde da aplicação, fornecendo métricas para que possa ser determinado como anda o funcionamento doprojeto (ARAUJO, 2021)*

O Spring fornece suporte para a criação de aplicações *REST (RepresentationalState Transfer)* também, facilitando a criação de *Web Services REST,* utilizando anotações embutidas nele que atribuem a classes e métodos alguns comportamentosdesejados, como por exemplo a *annotation @RequestMapping* que são utilizadas nosmétodos que tratam as requisições dos clientes (BIANCHI, 2015).

3 METODOLOGIA

O presente trabalho trata-se da construção de uma API voltada para o gerenciamento de um estabelecimento de *food service*. A aplicação será um *MinimumViable Product (MVP)* desenvolvida utilizando o *framework SpringBoot* e seus dados serão armazenados em um banco de dados relacional MySql, a parte de documentação da API será apresentado na ferramenta do Swagger API.

UNIFIED MODELING LANGUAGE (UML)

UML foi criado para estabelecer uma linguagem de modelagem comum para a arquitetura, design e implementação de sistemas de software, ela implementa uma modelagem com uma visão orientada a objetos, no qual definem-se as classes, atributos, métodos e seus relacionamentos (DEVMEDIA, 2022).

Modelo de domínio

O modelo de domínio usa a notação UML para um conjunto de diagrama de classes nos quais não são definidas as operações, ele mostra os objetos do domínio, os atributos das classes e as associações entre as classes (BARANAUSKAS, 2017).

REQUISITOS

A análise de requisitos são um conjunto de técnicas usadas para detalhar e documentar todos os aspectos que compõem a criação de um software, esta etapa é fundamental para que fique claro quais são as necessidades do sistema, o escopo dessa análise é dividido em requisitos funcionais, e requisitos não funcionais (CAMPOS E AZEVEDO, 2008).

Requisitos funcionais

Os requisitos funcionais descrevem explicitamente as funcionalidades do sistema, ele documenta como o sistema deve reagir a entradas especificas, como se comportar em determinadas situações e o que não pode fazer (FIGUEIREDO, 2011).

Requisitos não funcionais

Os requisitos não funcionais definem as propriedades e restrições do sistema em si e desempenham um papel critico durante o desenvolvimento do software, documentando os requisitos gerais como custo, performance, confiabilidade, portabilidade entre outros (CYSNEIROS E LEITE, 2001).

ARQUITETURA EM CAMADAS LÓGICAS

A arquitetura em camadas tem por objetivo promover a separação das lógicas de apresentação, lógica de negócio e lógica de acesso a dados por meio de camadas,tornando assim os sistemas mais flexíveis (MIGNON, 2022).

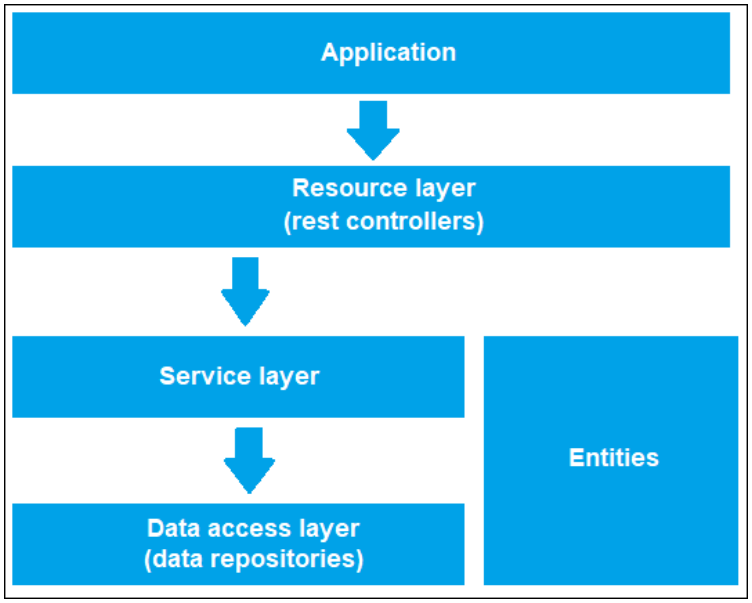


Figura 3 - Esquema arquitetura em camadas
Fonte: Autor (2022)

A figura 3 demonstra bem a organização da arquitetura em camadas, que traz o beneficio do isolamento, onde qualquer modificação em um componente não afete as outras camadas (MIGNON, 2022).

Camada de negócio

Esta camada que também é denominada de lógica empresarial ou regras de negócio é responsável por todas funções e regras de todo o negócio, nela estão presentes as entidades que compõem a estrutura do projeto e também é responsável pelo acesso e manipulação dos dados (GUEDES, 2021).

Camada de controle

Essa camada organiza os eventos ocorridos na interface do usuário e por suavez os direciona para a camada de modelo (GUEDES, 2021).

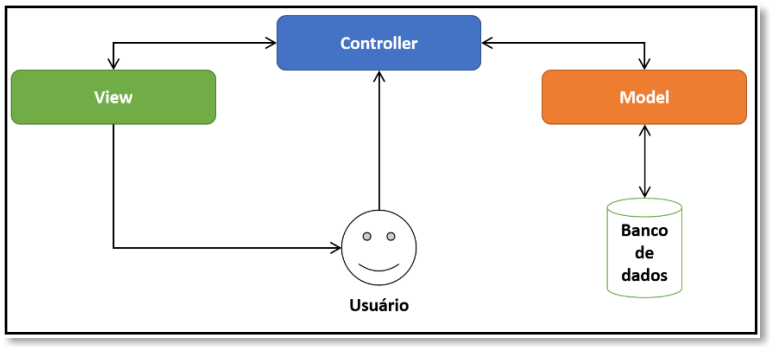


Figura 4 - Arquitetura MVC
Fonte: Guedes (2021)

A função dessa camada é servir como intermediador (Figura 4) entre a camada de apresentação e a camada de negócios (GUEDES, 2021).

Camada de apresentação

É nesta camada que o usuário na aplicação *frontend* irá se comunicas com a aplicação *backend*, esta seria a interface do usuário e sua principal finalidade é exibir e coletar informações do usuário (IBM, 2022).

DESENVOLVIMENTO

Neste capitulo mostraremos as etapas da construção da *API* de *Food Service*, demonstrando todos os passos desde o planejamento à implementação.

Modelo De Domínio

O modelo de domínio da aplicação, foi produzido utilizando a ferramenta *StarUml* da empresa *MKLab*.

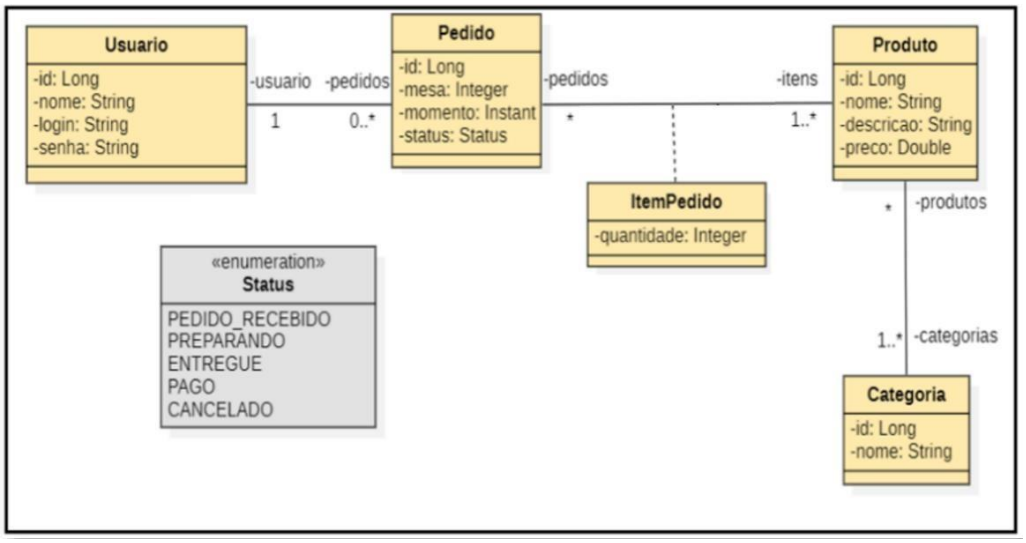


Figura 5 – Modelo de domínio da API de *food service*
Fonte: Autor (2022)

Como podemos verificar na Figura 5 esse modelo apresenta todas as entidades envolvidas para a construção da *API* bem como também retrata os relacionamentos entre as entidades.

Requisitos funcionais

Para os requisitos funcionais foi levado em consideração as funcionalidades esperadas para a aplicação.

Tabela 1 – Requisitos funcionais da API de *food service*

ID	Nome	Descrição	Prioridade
RF01	Gerenciamento de usuários	O sistema deve permitir incluir, excluir, atualizar e pesquisar usuários.	Essencial
RF02	Gerenciamento de categorias	O sistema deve permitir incluir, excluir, atualizar e pesquisar categorias.	Essencial
RF03	Gerenciamento de produtos	O sistema deve permitir incluir, excluir, atualizar e pesquisar produtos.	Essencial
RF04	Cadastro de pedidos	O sistema deve permitir um cadastro de pedidos.	Essencial
RF05	Atualizar status dos pedidos	O sistema deve permitir atualizar status do pedido.	Essencial
RF06	Atualizar produtos dos pedidos	O sistema deve permitir atualizar os produtos do pedido.	Importante
RF07	Buscar pedidos	O sistema deve permitir uma pesquisa dos pedidos.	Essencial

Fonte: Autor (2022)

A tabela 1 mostra uma descrição dos requisitos funcionais bem como sua prioridade.

Requisitos não funcionais

Para os requisitos não funcionais foi levado em consideração as propriedades esperadas para a aplicação.

Tabela 2 – Requisitos não funcionais da API de *food service*

ID	Nome	Descrição	Prioridade
RNF01	Gerenciamento de dados	rá utilizar um banco de dados relacional	Essencial
RNF02	Documentação API	O sistema deve ser documentado pelo Swagger API	Essencial

Fonte: Autor (2022)

A tabela 2 mostra uma descrição dos requisitos funcionais bem como sua

prioridade.

Entidades e relacionamentos

Neste tópico apresentaremos as entidades utilizadas para a confecção da *API Food Service* bem como seus relacionamentos, todas as classes foram anotadas com as *annotations @Getter @Setter @NoArgsConstructor e @AllArgsConstructor* do pacote *lombok* que é uma biblioteca java que visa a produtividade e redução de código, essas anotações são responsáveis pela criação dos construtores vazio e com argumentos e também dos *getters* e *setters* dos atributos e também as *annotations @Entity e @Table* do pacote *javax.persistence*, essas duas ultimas foram utilizadas para informar que a classe é uma entidade e a partir disso o *Java Persistence API(JPA)* estabelece a ligação entre a entidade e uma tabela na base de dados com o nome da classe ou o nome selecionado na anotação *@Table*. Os *Ids* das entidades possuem uma estratégia de autoincremento que foi possibilitado pela adição das *annotations @Id* e *@GeneratedValue* também do pacote *javax.persistence.*

```java
@Entity
@Table(name = "tb_usuario")
@Getter @Setter @NoArgsConstructor @AllArgsConstructor
public class Usuario implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotEmpty(message = "O campo nome é obrigatório")
    private String nome;
    @NotEmpty(message = "O campo login é obrigatório")
    private String login;
    @NotEmpty(message = "O campo senha é obrigatório")
    private String senha;

    @JsonIgnore
    @OneToMany(mappedBy = "usuario")
    private List<Pedido> pedidos = new ArrayList<>();
```

(A)

```java
@Entity
@Table(name = "tb_categoria")
@Getter @Setter @NoArgsConstructor @AllArgsConstructor
public class Categoria implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotEmpty(message = "O campo nome é obrigatório")
    private String nome;

    @JsonIgnore
    @OneToMany(mappedBy = "categoria")
    private Set<Produto> produtos = new HashSet<>();
```

(B)

```java
@Entity
@Table(name = "tb_produto")
@Getter @Setter @NoArgsConstructor @AllArgsConstructor
public class Produto implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotEmpty(message = "O campo nome é obrigatório")
    private String nome;
    @NotEmpty(message = "O campo descrição é obrigatório")
    private String descricao;
    @NotNull(message = "O campo preço é obrigatório")
    private Double preco;

    @ManyToOne
    @JoinColumn(name = "categoria_id")
    private Categoria categoria;
```

(C)

```java
public enum StatusPedido {

    PEDIDO_RECEBIDO( codido: 1),
    PREPARANDO( codido: 2),
    ENTREGUE( codido: 3),
    PAGO( codido: 4),
    CANCELADO( codido: 5);

    private int codido;
    private StatusPedido(int codido) {
        this.codido = codido;
    }
    public int getCodido() {
        return codido;
    }
    public static StatusPedido valueOf(int codido) {
        for (StatusPedido value: StatusPedido.values()) {
            if (value.getCodido() == codido) {
                return value;
            }
        }
        throw new IllegalArgumentException("Código PedidoStatus Inválido");
    }
}
```

(D)

Figura 6 – Entidades Usuário (A), Categoria (B), Produto (C) e Status do Pedido (D)

Fonte: Autor (2022)

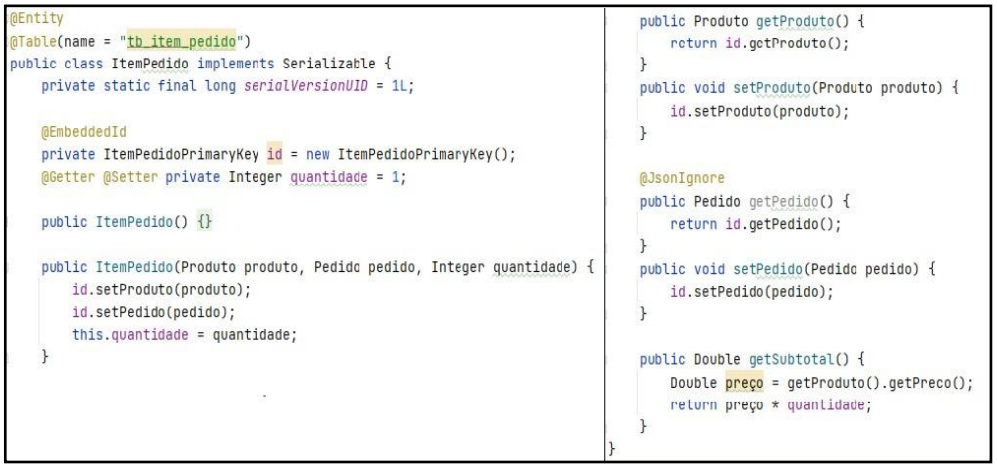
A figura 6a mostra a entidade que é composta pela regra de negócio dos usuários, além dos atributos nome, login e senha, essa classe possui um relacionamento com a classe de pedidos, a *annotation @OneToMany* do pacote *javax.persistence* informa que os usuários tem uma relação de um para muitos com os pedidos, ou seja, um usuário pode realizar vários pedidos. A figura 6b trata das categorias dos produtos, essa entidade possui como atributo o nome e um também possui um relacionamento de um para muitos com os produtos. Na figura 6c temos aimplementação da entidade de produtos, essa classe possui um relacionamento muitos para um com a entidade de categorias, e possui um registro adicional em suatabela descrito pela *annotation @JoinColumn,* para referenciar qual o id da categoriafoi associada ao produto. Na figura 6d temos uma entidade enumerada com as informações sobre os status dos pedidos, nessa classe o método *valueOf* foi implementado para converter o código do status dos pedidos de valor numérico paraо seu tipo enumerado em questão.

```java
@Entity
@Table(name = "tb_pedido")
public class Pedido implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Getter private Long id;
    @NotNull(message = "Informe o número da mesa")
    @Getter @Setter private Integer mesa;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy'T'HH:mm:ss'Z'", timezone = "GMT")
    @Getter @Setter private Instant momento;

    @ManyToOne
    @JoinColumn(name = "usuario_id")
    @Getter @Setter private Usuario usuario;

    private Integer status;

    @OneToMany(mappedBy = "id.pedido")
    @Getter @Setter private Set<ItemPedido> itens = new HashSet<>();

    public Pedido() {}

    public Pedido(Long id, Integer mesa, Instant momento, Usuario usuario, StatusPedido status) {
        this.id = id;
        this.mesa = mesa;
        this.momento = momento;
        this.usuario = usuario;
        setStatus(status);
    }

    public StatusPedido getStatus() {
        return StatusPedido.valueOf(status);
    }
    public void setStatus(StatusPedido status) {
        if (status != null) {
            this.status = status.getCodido();
        }
    }

    public Double getTotal(){
        double total = 0.0;
        for (ItemPedido item: itens) {
            total += item.getSubtotal();
        }
        return total;
    }
}
```

Figura 7 – Entidades Pedidos

Fonte: Autor (2022)

A figura 7 representa a classe responsável pelos pedidos, pra essa entidade eseus relacionamentos funcionarem como desejado, ou seja, conseguir fazer uma ligação com a entidade de produtos, precisamos implementar uma classe auxiliar, queé a Item de pedido.



```java
@Entity
@Table(name = "tb_item_pedido")
public class ItemPedido implements Serializable {
    private static final long serialVersionUID = 1L;

    @EmbeddedId
    private ItemPedidoPrimaryKey id = new ItemPedidoPrimaryKey();
    @Getter @Setter private Integer quantidade = 1;

    public ItemPedido() {}

    public ItemPedido(Produto produto, Pedido pedido, Integer quantidade) {
        id.setProduto(produto);
        id.setPedido(pedido);
        this.quantidade = quantidade;
    }

    public Produto getProduto() {
        return id.getProduto();
    }
    public void setProduto(Produto produto) {
        id.setProduto(produto);
    }

    @JsonIgnore
    public Pedido getPedido() {
        return id.getPedido();
    }
    public void setPedido(Pedido pedido) {
        id.setPedido(pedido);
    }

    public Double getSubtotal() {
        Double preço = getProduto().getPreco();
        return preço * quantidade;
    }
}
```

Figura 8 – Classe intermediaria entre pedidos e produtos

Fonte: Autor (2022)

A classe Item de pedido (Figura 8) é responsável por instanciar os dados do pedido, os dados do produtos associados ao pedido e também a quantidade desse item de produto adicionado do pedido, essa classe também tem um método capaz derealizar o cálculo do valor subtotal, que servirá para facilitar o cálculo total que é feitopela entidade pedido.

```java
@Embeddable
@Getter @Setter
public class ItemPedidoPrimaryKey implements Serializable {
    private static final long serialVersionUID = 1L;

    @ManyToOne
    @JoinColumn(name = "produto_id")
    private Produto produto;

    @ManyToOne
    @JoinColumn(name = "pedido_id")
    private Pedido pedido;
}
```

Figura 9 – Classe auxiliar dos itens de pedido

Fonte: Autor (2022)

A classe Item de Pedido possui uma particularidade que é não possuir um id próprio gerado pelo banco de dados, o id que representa essa entidade vem da instanciação de uma segunda classe auxiliar que é a responsável por registrar o pedido e o produto que foi nomeada como *ItemPedidoPrimaryKey (*figura 9), para que essa classe seja visualizada como uma classe auxiliar de chave primaria composta foi necessário colocar a *annotation @Embeddable* do pacote *javax.persistence* e também é necessário anotar o atributo id responsável pela instanciação dessa classe presente na entidade Item Pedido com a *annotation @EmbeddedId.*

Services e RestControllers

Os *services* e os *controllers* foram implementados como classes intermediarias, onde os services pegam os dados do repositório e tratam da regra de negócio da aplicação e o *controller* se comunica com a camada de apresentação, fazendo uso dos métodos implementados no service.

Figura 10 – Classes de serviços de Usuário (A), Categoria (B), Produto (C) e Pedido (D)
Fonte: Autor (2022)

Na figura 10 temos as classes de serviço implementadas para o sistema, nelas realizamos uma injeção de dependência por meio da *annotation @Autowired* para podermos acessar os dados presentes na classe de repositório que fará a ligação dos dados presentes na base de dados e regras de negócio implementadas.

Para as classes de serviço produto (figura 10c) e pedido (figura 10d), além da injeção de dependências com o *repository* também precisamos acessar outras classes de serviço como a CategoriaService para os produtos e UsuarioService e ProdutoService para os pedidos, essa ligação foi necessária para poder trabalhar na regras de negócio dos métodos salvar e atualizar conforme vemos nas imagens abaixo.

Figura 11 – Classes de controle de Usuário (A), Categoria (B), Produto (C) e Pedido (D)
Fonte: Autor (2022)

Já na figura 11, temos nossas classes de controle, nelas realizamos a injeção de dependência com as classes de serviço, para tratar das requisições que vão para a base de dados e as respostas que vão para a camada de apresentação. Nessas

classes de controle também definimos o *endpoint* da nossa *API* a partir da *annotation @RequestMapping* e criamos descrições para a documentação do *Swagger API* a partir das *annotations @Tag e @Operation.*

Swagger

O *Swagger* em nossa aplicação além de ser a documentação da *API,* exerce a função da camada de apresentação.

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.3</version>
</dependency>
```

Figura 12 – Dependência adicionada ao pom.xml para utilização do *Swagger API*
Fonte: Autor (2022)

Para utilizar o *Swagger API* adicionamos no arquivo *pom.xml* uma dependência da *org.springdoc* (figura 12) e as descrições de cada serviço foram inseridas nas *annotations @Tag e @Operation* de cada *controller* (figura 11).

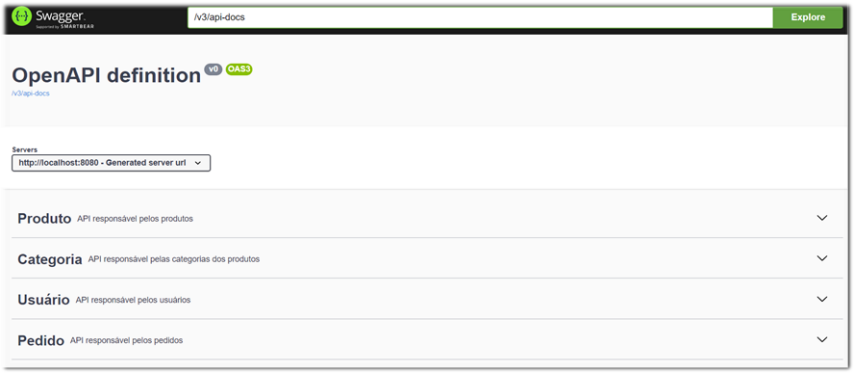


Figura 13 – Interface geral da API no Swagger
Fonte: Autor (2022)

Acima mostraremos como se comporta visualmente o *Swagger API* e as operações registradas em cada *endpoint* da aplicação, na interface geral (figura 13) podemos verificar o resultado do *name* e *description* inseridas na anotação *@Tag*.

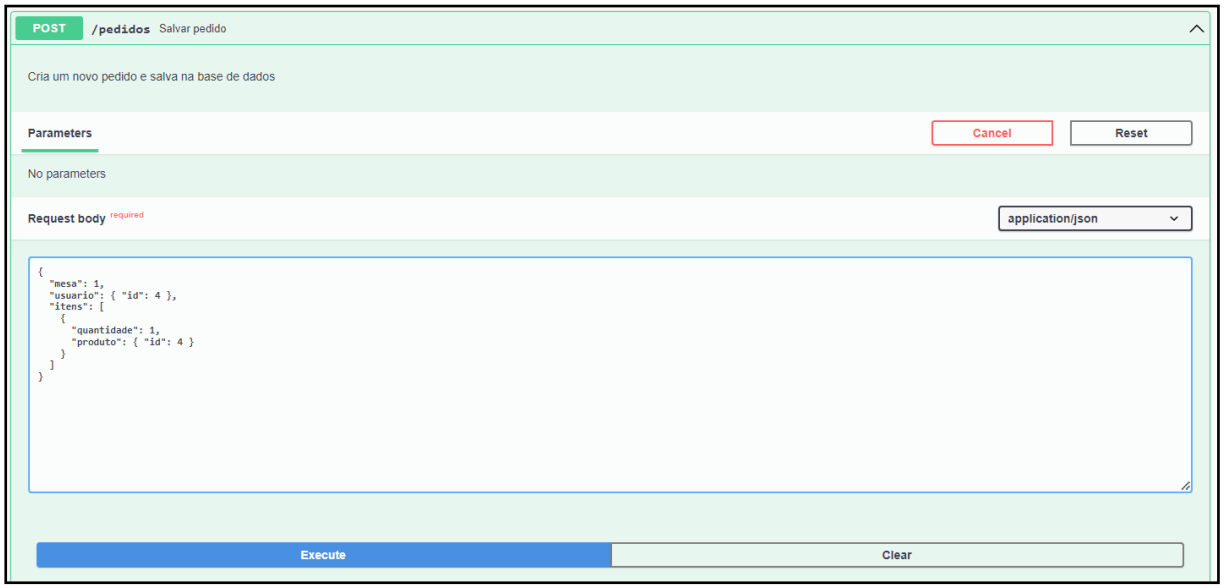


Figura 14 – Exemplo de requisição para salvar um pedido

Fonte: Autor (2022)

Para ilustrar o comportamento e testar nossa aplicação a figura 14 representa um exemplo no qual estamos enviando uma requisição através do verbo *http POST* para salvar um pedido.

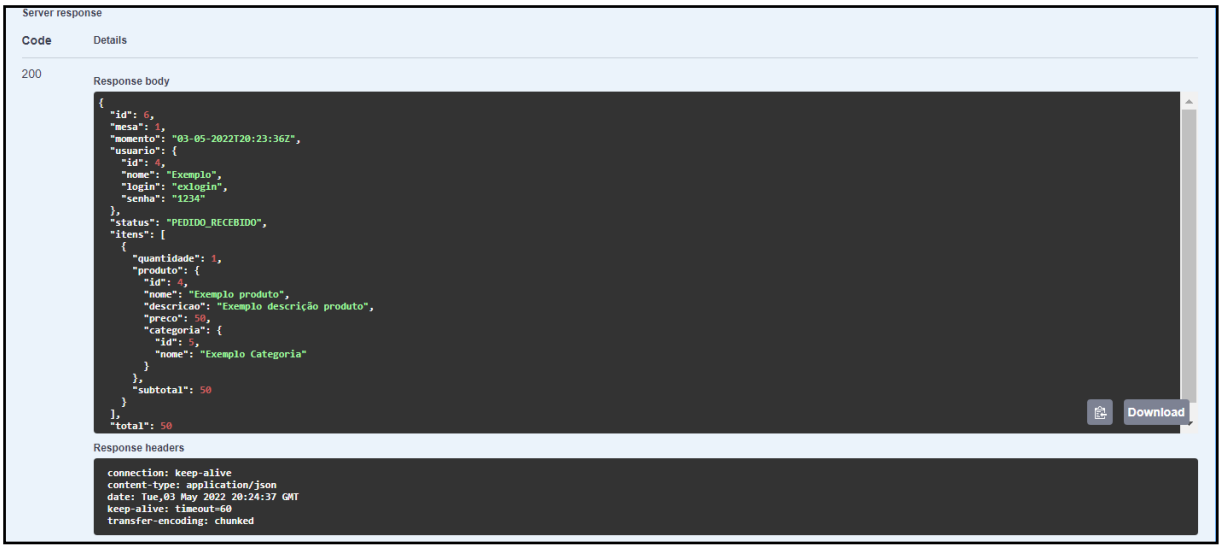


Figura 15 – Exemplo de resposta para listar um pedido

Fonte: Autor (2022)

E na figura 15 temos um exemplo da solicitação para listar os pedidos através do verbo *http GET*.

4 CONSIDERAÇÕES FINAIS

A etapa de planejamento do projeto teve grande relevância na construção da aplicação, visto que foi mais rápido e prático implementar os modelos previamente estudados bem como realizar de correta as etapas de desenvolvimento seguindo os requisitos.

Quanto a etapa de desenvolvimento, foi seguido um modelo de arquitetura em camadas que fez com que cada classe desempenhasse seu papel de forma organizada e exercendo suas respectivas responsabilidades, mantendo assim o código mais limpo. A documentação da *API* se deu pela utilização da ferramenta *Swagger* que serviu também como uma camada de apresentação para que fossem realizados os testes de requisições e respostas.

REFERÊNCIAS

ARAUJO, Daniela. **Spring Boot: o que é e como usar: O guia inicial.** Disponível em: <https://blog.betrybe.com/framework-de-programacao/spring-boot-tudo- sobre/#1>. Acesso em 03 abr. 2022.

AZEVEDO JUNIOR, Delmir Peixoto de; CAMPOS, Renato de. **Definição de requisitos de software baseada numa arquitetura de modelagem de negócios. Production**, v. 18, n. 1, p. 26-46, 2008.

BARANAUSKAS, M. C. C. **Modelo de domínio: Visualizando conceitos.** Disponívelem: <https://www.ic.unicamp.br/~ariadne/mc436/1s2017/Lar10ModDom.pdf>. Acesso em 26 de abr. 2022.

BIANCHI, Evaldo Augusto. **Sistema de envio e recebimento de mensagens para plataforma android utilizando Spring Boot e Google Cloud Messaging**. 2015.

BUYCO. **Mercado de restaurantes: um setor em crescimento.** Disponível em: <https://buyco.com.br/mercado-de-restaurantes>. Acesso em 02 mar. 2022.

CYSNEIROS, Luiz Marcio; LEITE, J. C. S. P. **Requisitos não funcionais: da elicitação ao modelo conceitual. PhDTese, PUC-RJ**, 2001.

DE OLIVEIRA, ROBERTO GUIMARÃES DUTRA. **Utilização De Recursos Java NaPlataforma Android Para Facilitar O Processo De Compra**. 2012.

DEVMEDIA. **Introdução a UML.** Disponível em: <https://www.devmedia.com.br/introducao-a-uml/6928>. Acesso em 27 de abr. 2022.

FIGUEIREDO, Eduardo. **Requisitos Funcionais e Requisitos NãoFunciona
FECCHIO, Alessander. **Java EE 5: Desenvolvendo aplicações corporativas.** 2006.

GUEDES, Marylene. **O que é MVC?** Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-mvc>. Acesso em 28 de abr. 2022.

IBM. **Arquitetura de três camadas.** Disponível em: <https://www.ibm.com/br-pt/cloud/learn/three-tier-architecture>. Acesso em 28 de abr. 2022.

PACIEVITCH, Yuri. **História do Java.** Disponível em: <https://www.infoescola.com/informatica/historia-do-java/>. Acesso em 21 de abr. 2022.

MIGNON, Alexandre. **Arquitetura em 3 camadas.** Disponível em: <https://sites.google.com/site/amignon/poo2/arquitetura-em-3-camadas>. Acesso em27 de abr. 2022.

ROMANATO, Alan. **Introdução ao Java Virtual Machine (JVM).** Disponível em: <https://www.devmedia.com.br/introducao-ao-java-virtual-machine-jvm/27624>. Acesso em 03 abr. 2022.

SILVA, Alice Fernandes et al. Gerador de código para uma API REST com base noframework Spring Boot. 2019.

LUCKOW, Décio Heinzelmann; DE MELO, Alexandre Altair. **Programação Java paraa WEB**. Novatec Editora, 2010.

OIMENU. **Setor de restaurantes no Brasil: o futuro pós-pandemia.** Disponível em: <https://www.oimenu.com.br/blog/administracao/setor-restaurantes-brasil>. Acesso em 02 mar. 2022.


uniesp
Centro Universitário
COOPERE


ISBN 978-655825142-2
9 786558 251422