Editora Uniesp

DIÁLOGOS CIENTÍFICOS EM SISTEMAS

PRODUÇÕES ACADÊMICAS 2021.1

ORGANIZADORES:
Marcelo Fernandes de Sousa
Hercílio Medeiros Sousa

uniesp
Centro Universitário

ISBN: 978-65-5825-077-7

DIÁLOGOS CIENTÍFICOS EM SISTEMAS: PRODUÇÕES ACADÊMICAS 2021.1

Marcelo Fernandes de Sousa
Hercílio Medeiros Sousa
(Organizadores)

Centro Universitário – UNIESP

Cabedelo - PB
2021

Designer Gráfico:
Mariana Morais de Oliveira Araújo

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

D537 Diálogos científicos em sistemas: produções acadêmicas 2021.1 [recurso eletrônico] / Organizadores: Marcelo Fernandes de Sousa, Hercílio Medeiros Sousa. - Cabedelo, PB: Editora UNIESP, 2021.
116 p.

Tipo de Suporte: E-book
ISBN: 978-65-5825-077-7

1. Produção científica – Sistemas. 2. Sistemas - Computação - Interdisciplinaridade. 3. Diálogos – Conhecimento científico. I. Título. II. Sousa, Marcelo Fernandes de. III. Sousa, Hercílio Medeiros.

CDU: 001.891:004

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

AUTOMATIZAÇÃO DE TESTES DE SOFTWARE DESKTOP UTILIZANDO CUCUMBER, BEHAVIOR DRIVEN DEVELOPMENT E TYPESCRIPT

ASSIS, Victor Augusto Monteiro [1]
SOUSA, Hercilio de Medeiros [2]

1 INTRODUÇÃO

O avanço da tecnologia da informação está cada dia mais evidente na vida do ser humano, a qual é utilizada para diversas práticas, como por exemplo: transações bancárias, consultas de processos, geração de folhas de pagamentos, e-commerce, redes sociais, dentre outras. Com a chegada da pandemia do Novo coronavírus (COVID-19), decretada no mês de março de 2020, o uso de sistemas *Web* potencializou-se de tal forma que o usuário final resolvesse tudo por um *smartphone*, computador ou *tablet*, sem a necessidade de deslocar-se de sua residência.

Com a alta demanda na utilização de sistemas *Web,* torna-se necessário garantir maior qualidade do software disponibilizado, visto que a experiência do usuário (denominada de UX), é primordial para ter uma aplicação de alto nível. Desse modo, os testes que são realizados durante a implementação de uma solução devem ser os mais semelhantes às situações que o usuário final pode tentar realizar.

O teste deve ser planejado desde a primeira fase de um projeto, denominada de “levantamento de requisitos”. Considerando esta logística, o responsável pela qualidade tem como missão encontrar possíveis erros ainda nesta fase, visando mitigar os riscos após a implementação da solução. Nos processos de desenvolvimento de um software, quanto mais precoce for encontrado uma falha, menores os custos gerados para o projeto.

Garantir a qualidade de um software requer planejamento, gestão de riscos e execução. Existem diversos exemplos em que a qualidade de software impediu guerras e conflitos a nível global. Nos dias atuais as empresas têm observado a qualidade de software não só como uma garantia, mas também como uma maneira de economizar, visto que o trabalho para correção do *bug* encontrado pelo testador no tempo de desenvolvimento gera menores custos do que a correção do *bug* após sua publicação da versão final em produção.

[1] https://www.linkedin.com/in/victor-assis-253854195/
[2] http://lattes.cnpq.br/2771260225601245 / https://www.linkedin.com/in/herciliodemedeiros/

Diante do exposto, considerando a importância da fase de testes e da automatização destes nos dias atuais, esta pesquisa versará acerca da utilização de ferramentas para automatização de testes, reduzindo o tempo e mão-de-obra que são necessárias para execução de testes manuais. Para obtenção de sucesso neste estudo, utilizaremos o *framework Cucumber* que promete menos desgaste do profissional e ganho de tempo nos testes de um sistema.

1.1 OBJETIVOS

O objetivo geral do presente trabalho é desenvolver uma automatização de testes de software utilizando o *WebDriver.io*, *Behavior Driven Development* e o *framework Cucumber* com a linguagem Gherkin, na linguagem de programação dos steps sendo *Typescript*.

- Pesquisar sobre métodos e ferramentas para facilitar a escrita de testes automatizados;
- Garantir a qualidade de software utilizando a automação de testes;
- Explicar cada passo dado utilizando a mistura desses componentes.

2 ENGENHARIA DE SOFTWARE

O desenvolvimento de um software é composto por vários modelos abstratos e precisos que permitem ao time de desenvolvimento especificar, projetar, implementar e manter *softwares* desenvolvidos. Antigamente utilizavam o modelo “cascata” para desenvolver um *software,* que vai desde o levantamento de requisitos, ao desenvolvimento do projeto e por fim testes. Com o passar dos tempos foram sendo criados vários modelos distintos para desenvolvimento de *software* (SOMMERVILLE, 2003).

Nos dias atuais, as entregas são constantes, logo, o modelo “cascata” em muitas empresas foi descontinuado, fazendo-se necessário lançar mão de utilizar métodos mais eficientes para as entregas contínuas, que ajudem os times de desenvolvimento a entregar com menos falhas e uma solução mais eficiente.

Vale salientar que um dos métodos mais usados atualmente são as metodologias ágeis, trata-se do *Scrum, Kanban, XP (Extreme Programming),* dentre outras. Essas metodologias permitem ao time de desenvolvimento entregas constantes de pequeno e grande valor. Diante disso, modificações levantadas nas

entregas não geram custos altíssimos para o projeto, pois com entregas de pequenas partes e constantes é possível desenvolver um *software* estável.

Nos modelos mais antigos como “cascata” e RUP, os testes são feitos após toda a implementação da solução, diferentemente das metodologias ágeis, em que a camada de teste é executada durante o desenvolvimento das pequenas partes que são requisitadas pelo dono ou responsável do produto.

2.1.2 QUALIDADE DE SOFTWARE

Garantir a qualidade de um *software* consiste em identificar defeitos em uma aplicação, antes que o sistema seja entregue para o cliente final e seja publicado no ambiente de produção. Assim, testar tem que ser algo rápido e assertivo, pois quanto mais for executado cenários de teste, mais confiável será o sistema (PFLEEGER, 2004).

Os *softwares* precisam seguir fluxos descritos em documentos que são denominados “casos de uso ou histórias", métodos, ferramentas e pessoas. (SPINOLA, 2005). O teste de *software* busca observar qual o comportamento que o sistema segue e se está de acordo com o que foi especificado, seja de uma API, de uma tela ou da comunicação entre API (*back-end*) e o *Front-end.*

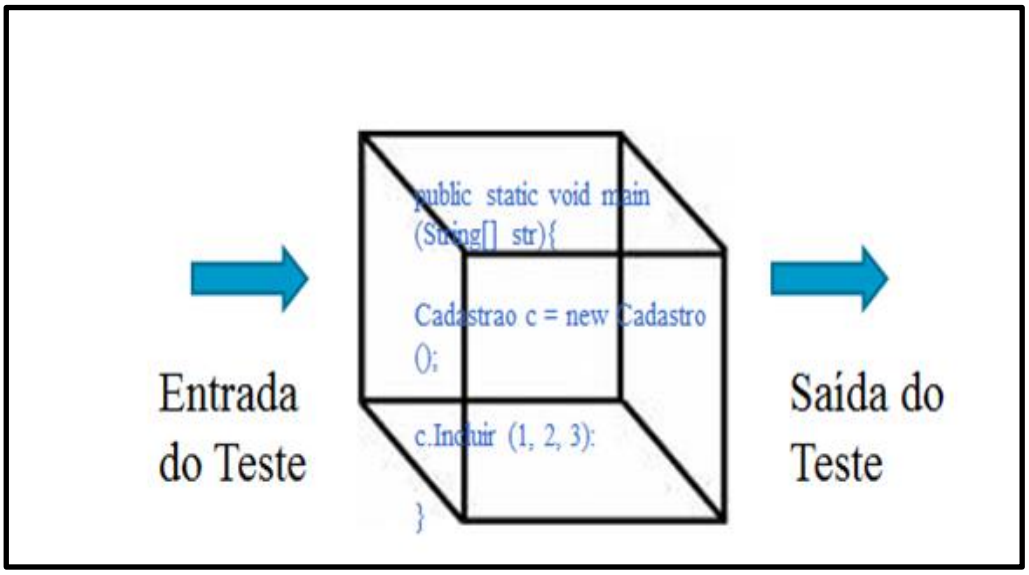
Baixo investimento e um software com um índice de teste baixo pode afetar na qualidade entregue para o usuário final e trazer um custo elevado para a empresa responsável pelo desenvolvimento da solução (INTHURN, 2001).

2.1.3 TIPOS DE TESTE

2.1.3.1 TESTES DE CAIXA BRANCA

Neste teste o profissional deve observar a lógica de um sistema, como se comporta as unidades deste sistema e as partes mais importantes do código. Tais observações devem ser realizadas desde a formatação até a passagem correta dos dados, como demonstrado na imagem a seguir:

Figura 1: Demonstração da prática do teste de caixa branca.



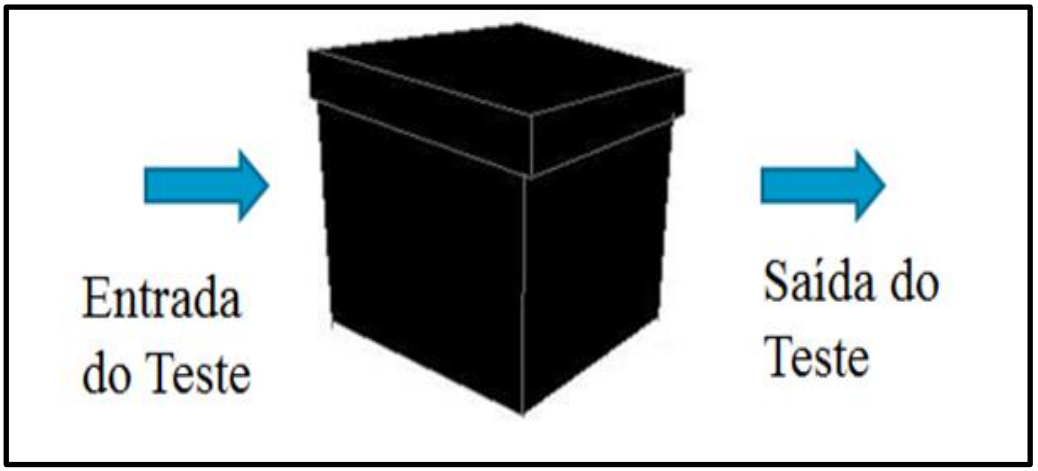
Fonte: Nexxera, 2016.

Vale salientar que podem existir casos que o erro não seja identificável por conta de poucos dados de entrada fornecidos.

2.1.3.2 TESTES DE CAIXA PRETA OU FUNCIONAL

No item anterior foi explanado sobre teste das unidades ou métodos, neste item o profissional deve abordar o teste da funcionalidade por completa tendo em vista que deve simular ao máximo o usuário final baseando-se no documento de requisitos, observar o comportamento do sistema, todas as validações de campos, estilo do *layout*, dentre outros.

Figura 2: Demonstração da prática do teste de caixa preta.



Fonte: Nexxera, 2016.

2.1.3.3 TESTES DE ACEITAÇÃO

O teste de aceitação trata-se de uma validação da versão final de uma solução que está pronta para ser colocada em um ambiente diferente do que foi desenvolvido (COIFMAN, 2020).

2.1.3.4 TESTES DE REGRESSÃO

Os testes de regressão são aqueles executados em aplicações que ficaram por bastante tempo sem manutenção em seu código, ou houve mudança no escopo do projeto que afetou mais de uma funcionalidade. Sendo assim, esse teste é essencial para refazer todos os testes dos componentes de uma determinada aplicação (COIFMAN, 2020).

2.1.3.5 TESTES DE USABILIDADE

O objetivo deste teste é verificar a experiência do usuário durante a utilização do sistema que foi desenvolvido, a organização dos itens, a comunicação dos botões existentes, o tempo de resposta dos botões, o tempo de carregamento da página, a responsividade em vários tamanhos, dentre outros (COIFMAN, 2020).

2.1.3.6 TESTES DE INTEGRAÇÃO

O teste de integração se refere à união de todas as unidades (métodos) se comunicando para formar a aplicação. Tal teste é responsável por validar todas as interfaces (APIs) que serão utilizadas pela implementação do *front-end* (COIFMAN, 2020).

2.1.4 WEBDRIVER.IO

O *webdriver.io* trata-se de uma estrutura de automação de testes que simplifica a interação com o aplicativo e fornece funções que facilitam a criação de testes escaláveis, robustos e instáveis.

Por ser um projeto *Open Source*, a utilização dessa estrutura vem crescendo cada dia mais, facilitando na criação de projetos de automação de testes de *software*. Sendo assim, uma estrutura de código aberto para utilizar o *WebDriverIO* deve instalar a estrutura com o framework desejado, como mostra a imagem a seguir.

Figura 3: Comando de instalação do *WebDriverIO*.

```
npm install @ wdio / cucumber-framework --save-dev
```

Fonte: WebDriverIO, 2021.

2.1.5 TYPESCRIPT

O *Typescript* é uma linguagem de código aberto baseado no *Javascript*, ou seja, um superset do *Javascript.* O Typescript quando transpilado se torna um código JS (NOLETO, 2020).

A principal IDE para o desenvolvimento da linguagem *Typescript* é o Visual Studio e Visual Studio Code que contém a integração com a linguagem, para compilar um código basta utilizar os seguintes comandos: "*tsc –w"* ou *"tsc NomeArquivo.ts".*

Os arquivos com extensão *".ts"* são os arquivos da linguagem *typescript*. Este arquivo é onde o *Quality Assurance* (*QA)* irá desenvolver testes automatizados, onde deve criar seus *steps* que irão conter métodos, que irão realizar os passos descritos nas palavras-chave de cada funcionalidade (*Feature*) da linguagem *Gherkin* demonstrado no *item 2.1.6* deste documento.

Por fim, podemos enxergar o *Typescript* como um grande potencializador do *Javascript* e com uma grande vantagem que é a ferramenta de perceber problemas de compilação e futuras melhorias no código. Além disso, trata-se de uma tipagem muito forte para escrever códigos mais organizados, facilitando o entendimento de uma pessoa leiga no assunto.

2.1.6 GHERKIN

O *Gherkin* é uma forma de padronizar escritas de *features* baseado na regra de negócio (FONSECA, 2021).

Tratando-se de testes automatizados, a linguagem *Gherkin* serve para facilitar a leitura e a escrita para um indivíduo totalmente leigo no assunto. A linguagem utiliza um conjunto de palavras-chave que podem ser traduzidos para vários idiomas facilitando assim a leitura da *feature* de teste implementada.

Figura 4: *Feature* escrita na linguagem *Gherkin*.

```
#encoding: UTF-8
#language: pt

Funcionalidade: Login
    Eu como usuário
    quero logar no sistema com sucesso

Contexto:
    Dado que tenho conta

Cenário: Login com sucesso
    E quero acessar a página de login
    E preencher os campos necessários
    Quando clicar no botão de entrar
    Então deve ser redirecionado para area logada
```

Fonte: Autoria própria, 2021.

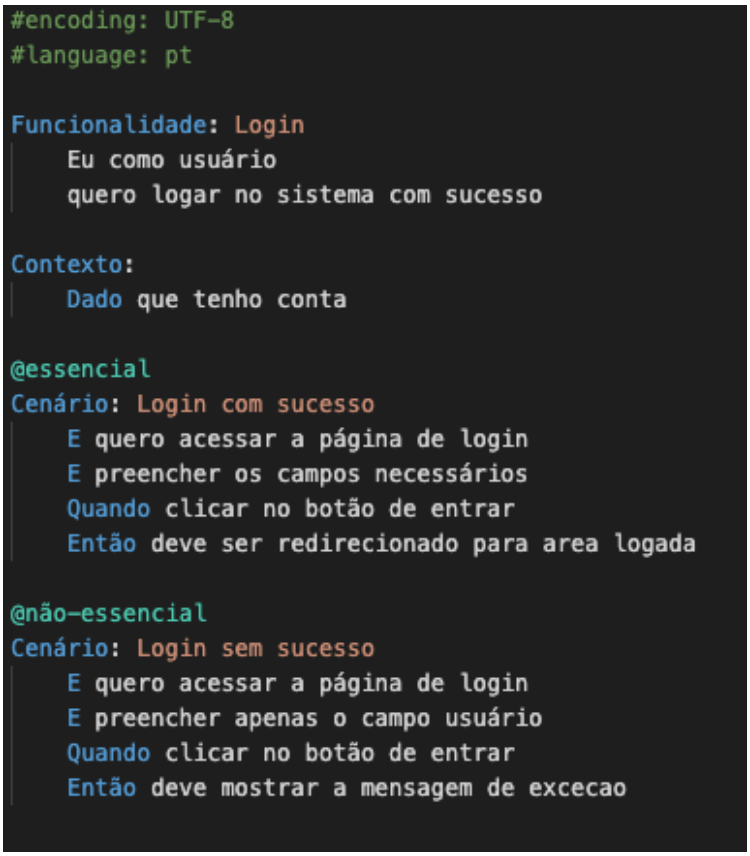
Para auxiliar na escrita dos arquivos "*features"* de automatização, os cenários de testes têm a estrutura como mostra a figura 4. Essa estrutura é formada da seguinte forma:

- *Funcionalidade (Feature)* - onde é colocado o nome da *feature* e as informações do que ela vai fazer;
- *Contexto (Background)* - onde é colocado os passos comuns daquela *feature*, onde são executados sempre antes de cada cenário;
- *Cenário (Scenario)* - onde é colocado o nome para o comportamento do que os passos irão fazer;
- *Dado (Given)* - onde é colocada a pré-condição para executar o teste;
- *Quando (When)* - onde é colocado a realidade do que vai acontecer no teste;
- *Então (Then)* - onde é colocado o que se espera de retorno da aplicação;
- *E (And) e Mas (But)* - onde é colocado para ligar dois ou mais passos em uma *feature*.

Na linguagem *Gherkin* existe uma ferramenta conhecida como "*@tags",* a qual é utilizada para organizar os cenários. O usuário pode definir uma *tag* acima do Funcionalidade *(Feature)* para facilitar na hora de executar o teste, pode ser criado uma *tag* que pule ou execute apenas um teste específico (FONSECA, 2021). Com

isso, as *tags* são utilizadas de acordo com a necessidade da execução do teste, como mostra a figura 5.

Figura 5: *Feature* escrita na linguagem *Gherkin com tags*.



```
#encoding: UTF-8
#language: pt

Funcionalidade: Login
    Eu como usuário
    quero logar no sistema com sucesso

Contexto:
    Dado que tenho conta

@essencial
Cenário: Login com sucesso
    E quero acessar a página de login
    E preencher os campos necessários
    Quando clicar no botão de entrar
    Então deve ser redirecionado para area logada

@não-essencial
Cenário: Login sem sucesso
    E quero acessar a página de login
    E preencher apenas o campo usuário
    Quando clicar no botão de entrar
    Então deve mostrar a mensagem de excecao
```

Fonte: Autoria própria, 2021.

Diante do exposto, após a criação da *feature* na linguagem *Gherkin*, o cenário será executado de acordo com a ordem que foi escrito. Mas para execução desta *feature* é necessário programar o que cada *step* irá fazer, assim como clique nos botões, abertura de páginas, preenchimento de *labels*, validação de exceções, dentre outros.

2.1.7 BEHAVIOR DRIVEN DEVELOPMENT

A definição dos requisitos é uma das partes mais importantes no processo de desenvolvimento de *software*. Mas, as vezes podem se tornar confusos e ricos em detalhes que outros integrantes do time que não entendem tanto de termos técnicos possam vir a não compreender.

O *Behavior Driven Development* (BDD) consiste em uma técnica de desenvolvimento ágil que busca integrar testes com a regra de negócio, com foco direcionado para o comportamento do software. Isto é, o *BDD* é a comunicação e alinhamento de todo o time de desenvolvimento para melhor definição de uma funcionalidade (SMART, 2014).

Ao utilizar o *BDD,* é possível observar que todo o time de desenvolvimento cria uma relação para elaboração dos exemplos para uma funcionalidade. Diante disso, o analista de testes inicia a elaboração de *features* para automatização de todos os cenários de testes possíveis e que também podem ser utilizados como material de apoio para o desenvolvimento.

As *features* contém cenários que seguem uma estrutura para que os testes possam ser executados. Desta forma, agem como um documento de critério de aceitação para o desenvolvedor e ajuda *designers* na construção de uma boa interface funcional. A figura 6 demonstrará a estrutura citada.

Figura 6: Representação de uma *feature* na técnica BDD utilizando a linguagem *Gherkin*.

```
Scenario: <Título do cenário>
    Given passo que representa a pré-condição para um evento
    Then passo que representa o que ocorre no evento
    When passo que representa a saída de um evento
```

Fonte: Adaptado Smart, 2014.

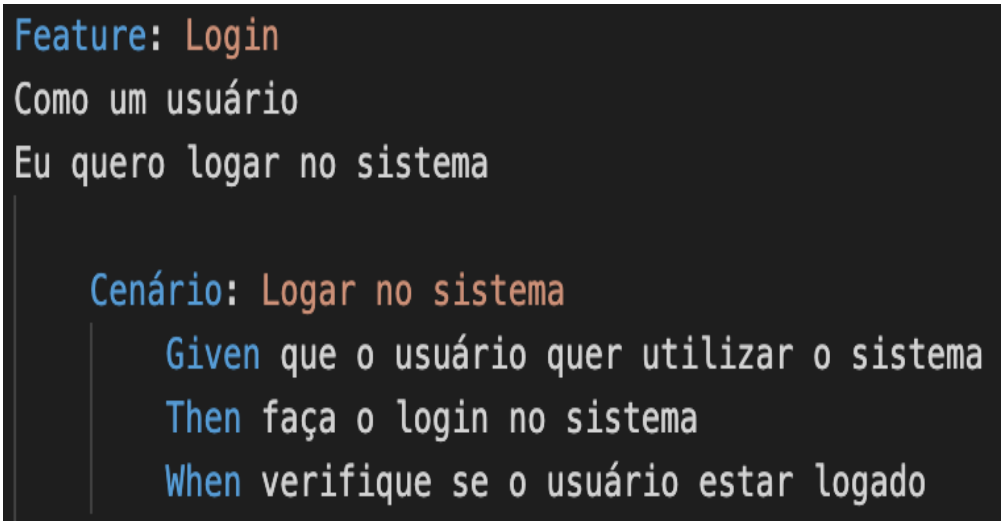
2.1.8 AUTOMATIZAÇÃO DE TESTES COM CUCUMBER

O *Cucumber* trata-se de uma ferramenta que foi construída para facilitar a execução de testes automatizados utilizando a técnica BDD.

Utilizar a ferramenta *cucumber* ajuda a melhorar o entendimento do código da automação de um projeto e a comunicação entre os membros e não-membros de um projeto, ou seja, nas empresas sempre há diferenças de profissionais, pode existir alguns que detenham mais conhecimento na automatização de testes com a ferramenta e outros que tenham mais conhecimento no teste manual dentro de um mesmo projeto (SOARES, 2011).

Com *Cucumber* não importa o nível da habilidade do testador, todos podem participar da automação de testes de software (BUI, 2018). O modelo da *feature* escrita com a ferramenta, segue o modelo como mostra a imagem a seguir:

Figura 7: Representação de uma *feature* utilizando a ferramenta *cucumber* (linguagem *Gherkin*) com a técnica BDD.



Fonte: Autoria própria, 2021

2.2 METODOLOGIA

Um dos principais pontos de uma pesquisa é a forma que serão desenvolvidos os resultados, com isso, neste presente trabalho se fez necessário a divisão entre as seguintes etapas: pesquisa e desenvolvimento. Onde, a pesquisa teve como base um modelo não estruturado, explorando várias referências bibliográficas.

Para a estruturação do código do projeto de automação foi utilizado o *Webdriver.io* na versão 6, com a escrita *Gherkin*, descrevendo os casos de testes: *login.feature* e *logoff.feature*, após ter criado todos os casos com o *Gherkin (Cucumber)* foi implementado os *steps* utilizando a linguagem *Typescript.*

O *Typescript* foi o responsável pela criação do passo a passo da automação de casos de teste. As etapas aconteceram da seguinte forma: implementação do *login.ts* que é a classe que declara todos os métodos utilizados e *loginImpl.ts* que é a classe que implementa de forma explícita a execução do teste (desde a utilização do *click*() até a comparação de *strings* recebidas durante a execução). Feito isso,

sucede-se a parte da implementação da chamada dos métodos na classe *loginSteps.ts,* no qual é feita toda ligação entre o *Gherkin* e o *Typescript.*

Dessa forma, observamos que o desenvolvimento dos testes dessa discussão foi elaborada em quatro partes: 1) utilização do *Webdriver.io* para montar um ambiente de testes escalável; 2) descrição dos casos de testes de uma funcionalidade usando as palavras-chave do *Gherkin*; 3) a implementação dos métodos na linguagem *TS,* no qual o automatizador executa em forma de código o seu teste fazendo validações, cliques, comparações, dentre outros; 4) integração entre escrita *Gherkin (Cucumber)* com o *Typescript,* fazendo assim um *feature* de teste automatizado.

2.3 RESULTADOS E DISCUSSÃO

De início em um projeto de automação de testes, faremos a configuração do *WebDriver.io*, uma vez que essa estrutura tem a finalidade limpar e organizar o código sempre. Para realização desta automação de testes foram utilizadas as seguintes dependências:

Figura 8: Dependências utilizadas no projeto.

```
{
  "dependencies": {
    "@types/faker": "4.1.2",
    "chai-arrays": "2.0.0",
    "chromedriver": "88.0.0",
    "config": "2.0.1",
    "cucumber": "5.0.0",
    "dotenv": "8.2.0",
    "guid-typescript": "1.0.7",
    "lodash": "4.17.11",
    "mocha": "5.2.0",
    "node-fetch": "2.1.1",
    "stubby": "4.0.0",
    "supertest": "6.1.3",
    "webdriverio": "6.4.6",
    "log4js": "6.3.0"
  }
}
```

Fonte: Autoria própria, 2021.

Após a configuração do ambiente, das dependências criadas, pode se dar início a implementação que irá executar um teste automatizado. Em todo teste de *software* são criados casos de teste para validação da funcionalidade implementada, diante disso, os casos de teste são descritos da seguinte forma:

- Título do caso de teste
  - Passo a ser seguido:
    - Ex: Preencha o campo de login.
    - Ex: Preencha no campo da senha.
  - Resultado obtido:
    - Ex: Login feito com sucesso.
    - Ex: Login Recusado.

Dessa forma, a imagem abaixo (Figura 9) demonstra a utilização da linguagem *Gherkin (Cucumber)* para escrita dos casos de testes, a qual também pode servir como uma documentação do projeto, levando em consideração a funcionalidade escrita com *Gherkin* pode ser vista como um caso de uso e ou estória de usuário.

Figura 9: Descrição do que será testado.

```
Feature: Login no sistema

    Scenario: Logar no sistema com sucesso
        Given que o usuário tenha conta
        When ele preencher os campos obrigatórios
        Then acesse a página inicial

    Scenario: Logar no sistema com credenciais inválidas
        Given que o usuário tenha conta
        When ele preencher os campos obrigatórios incorretos
        Then não acesse a página inicial

    Scenario: Deslogar da aplicação
        Given que o usuário queira Deslogar
        When clicar no botão de logout
        Then volte a página inicial
```

Fonte: Autoria própria, 2021.

Neste arquivo com extensão *.feature* (Figura 9) é possível observar a utilização do *Gherkin* na escrita dos casos de teste.

Logo após esta etapa, foi produzido um novo arquivo (Figura 10) com a extensão *.ts*, um arquivo *Typescript* que contém toda a lógica do teste que será implementado, desde cliques, preenchimentos de campos à comparações de saída.

Figura 10: Classe de implementação de comandos.

```
export default class loginPageImpl {
    private campoUser: string = '#userName';
    private campoPassword: string = '#userPassword';
    private buttonLogin: string = '#login';
    private buttonLogout: string = '#submit';
    private titlePage: string = '.pattern-backgound.playgound-header'
    private messageError: string = '#name';

    preencherCampos(user: string, password: string){
        $(this.campoUser).setValue(user);
        $(this.campoPassword).setValue(password);
    }

    sendLogin(){
        $(this.buttonLogin).click();
    }

    sendLogout(){
        $(this.buttonLogout).click();
    }

    validateToTitlePage(): string{
       return $(this.titlePage).getText();
    }

    getMessageError(): string{
        return $(this.messageError).getText();
    }
}
```

Fonte: Autoria própria, 2021.

Tal arquivo (Figura 10) contém seletores em forma de variáveis que recebem *strings*, para tornar o código mais limpo e compreensível, recebendo comandos de cliques e preenchimento de valores dentro dos métodos.

A imagem abaixo (Figura 11) contém o construtor da página de *login*, onde é criada a *loginPage* que faz chamada a página a qual implementa o teste.

Figura 11: Construtor do teste da página de *login*.

```
export default class LoginPage {
    private loginComponent: ILoginComponent;

    constructor(criteria: string) {
        this.loginComponent = new LoginComponentFactory().getLoginComponentImpl(criteria);
    }

    public getLoginComponent(): ILoginComponent {
        return this.loginComponent;
    }
}
```

Fonte: Autoria própria, 2021.

Dessa forma, realizada a implementação dos métodos que formam toda a lógica do teste, será construída a integração entre *Gherkin* e *Typescript*, sendo denominado da seguinte forma: *nome_da_featureSteps.ts* onde são criadas todas as chamadas do *Gherkin* em forma de funções do *Typescript* (Figura 12 e Figura 13).

Figura 12: Desenvolvimento da classe *loginSteps.ts.*

```
import { expect } from 'chai';
import { Given, Then, When } from 'cucumber';
import { loginPage } from '../../../constants';

let url = 'https://demoqa.com/login';
let user = 'victormntr';
let password = 'Victor10#';
let userInvalid = 'paulosgjr';
let passwordInvalid = '1234';

Given('Acesse o site e que o usuário tenha conta', function(){
    browser.url(url);
});

When('ele preencher os campos obrigatórios', function(){
    loginPage.getLoginComponent().preencherCampos(user, password);
    loginPage.getLoginComponent().sendLogin();
});

Then('acesse a página inicial', function(){
    expect(loginPage.getLoginComponent().validateToTitlePage()).to.equal('Profile');
});

When('ele preencher os campos obrigatórios incorretos', function(){
    loginPage.getLoginComponent().preencherCampos(userInvalid, passwordInvalid);
    loginPage.getLoginComponent().sendLogin();
});

Then('não acesse a página inicial', function(){
    expect(loginPage.getLoginComponent().getMessageError()).to.equal('Invalid username or password!');
});
```

Fonte: Autoria própria, 2021.

Figura 13: Continuação do desenvolvimento da classe *loginSteps.ts.*

```
Given('que o usuário queira Deslogar', function(){
    browser.url(url);
    loginPage.getLoginComponent().preencherCampos(user, password);
    loginPage.getLoginComponent().sendLogin();
});

When('clicar no botão de logout', function(){
    loginPage.getLoginComponent().sendLogout();
});

Then('volte a página inicial', function(){
    expect(loginPage.getLoginComponent().validateToTitlePage()).to.equal('Login');
});
```

Fonte: Autoria própria, 2021.

O *browser.url* é uma funcionalidade específica do *WebDriver.io*, que tem a finalidade de abrir uma url específica. Para dar início aos testes, foi utilizado esta função para abrir a página que foi automatizada por esta implementação.

O *expect* também é uma função do *webdribver.io* que tem como objetivo principal comparar valores de entrada e saída durante a execução de um teste.

Visto isso, na integração do *Gherkin* com *Typescript*, é possível observar que cada verbo do *Gherkin: "Given, When e Then*", tornam-se uma função que realiza ligação à classe que implementa os testes da página que será testada.

3 CONSIDERAÇÕES FINAIS

O processo de testes de *software* possui uma grande importância na evolução do desenvolvimento de aplicações, tendo em vista que a execução de testes é a principal prática para a garantia da qualidade de *software* e satisfação do cliente.

Com isso, a automatização de testes de *software* não se refere apenas a utilizar um sistema ver o seu comportamento, suas entradas e saídas de forma mais rápida e menos exaustiva, pode ser considerado como um projeto em paralelo, o que atualmente está sendo muito aplicado nas empresas, engenheiros de teste tem seu próprio repositório, *pipelines* de automação dentro do projeto de desenvolvimento de *software*. Com isso, o processo de testes, sendo praticado da forma mais correta possível, direciona todo time para uma maior detecção de falhas, gerando uma diminuição de manutenção e custos.

Diante do exposto, através deste estudo, foi possível demonstrar a implementação e integração da estrutura WebDriver.io, atrelada ao *Cucumber*

(Gherkin) e ao *Typescript* no processo da criação de uma documentação, uma *feature*, e o ponto principal que é automatização de testes de *software.*

REFERÊNCIAS

BUI, T. **Testes de automação com Cucumber BDD em Times Ágeis**. São Paulo: InfoQ, 2018. Disponível em: https://www.infoq.com/br/articles/cucumber-bdd-automation-testing/. Acesso em: 02 Mar. 2021.

COIFMAN. **Entenda o que é e quais os principais tipos de teste de software.** Rio de Janeiro: CRONNAP, 2020. Disponível em: https://blog.cronapp.io/tipos-de-teste-de-software/. Acesso em: 03 Mar. 2021.

FONSECA, H. **Gherkin: introduzindo seus conceitos e benefícios**. Campinas: OneDayTesting, 2021. Disponível em: https://blog.onedaytesting.com.br/gherkin/. Acesso em: 02 Mar. 2021.

INTHURN, C. **Qualidade de software.** Florianópolis: Visual Books, 2001.

NOLETO, C. **Typescript: o que é, principais conceitos e porquê usar!** Teresina: BeTrype, 2020. Disponível em: https://blog.betrybe.com/desenvolvimento-web/typescript. Acesso em: 02 Mar. 2021.

PFLEEGER, S. L**. Engenharia de Software: Teoria e Prática**. 2. ed. São Paulo: Pearson Prentice Hall, 2004.

SMART, J. F. **BDD in Action: Behaviour-Driven Development for the Whole Software Lifecycle.** 1. ed. Shelter Island: Manning Publications, 2014.

SOARES, I. **Desenvolvimento orientado por comportamento (BDD)**. Rio de Janeiro: DEVMedia, 2011. Disponível em: https://www.devmedia.com.br/desenvolvimento-orientado-por-comportamento-bdd/21127#:~:text=BDD%20%C3%A9%20t%C3%A9cnica%20de%20desenvolvimento,teste%20e%20depois%20o%20c%C3%B3digo. Acesso em: 03 Mar. 2021.

SOMMERVILLE, I. **Engenharia de Software**. 6. ed. São Paulo: Addison Wesley, 2003.

SPINOLA, M. M. **ISO 9001 para software**. Lavras: UFLA/FAEPE, 2005.

AGENDAR ATIVIDADES ATRAVÉS DE UMA APLICAÇÃO WEB COMPARTILHADA.

SILVA, Danilo A Costa da[1]
GOMES FILHO, Carlos Barbosa[2]

INTRODUÇÃO

A evolução constante da internet trouxe consigo o surgimento de aplicações web terceirizadas, a qual fornecem serviços para empresas possibilitando mais flexibilidade e eficiência. Devido a isso surgiu a terceirização web que traz a ideia de contratação de serviço através de um grupo ou empresa especializada em um segmento de mercado, esses serviços terceirizados ganhou força nas empresas por trazer benefícios lucrativos.

Uma pesquisa realizada pela Dieese (Departamento Intersindical de Estatística e Estudos Socioeconômicos) no ano de 2010 concluiu que um empregado pelas normas trabalhista CLT (Consolidação das Leis do Trabalho) recebi 27% a mais e cumpri uma carga horária menor que um funcionário terceirizado. No contexto web não é diferente contratar um serviço online, pois pode apresentar vantagens em algumas atividades.

O presente trabalho resulta em apresentar uma proposta do *Schedule System* um sistema criado para facilitar a organização de tarefas, reuniões, visitas e etc. A ideia é automatizar o agendamento de atividades do dia a dia possibilitando ter um controle sobre elas de forma prática, imagine que o sistema irá fazer o papel de um(a) recepcionista no qual será encarregado de organizar e preparar os convites para uma reunião ou enviar tarefas para funcionários externos via *E-mail* e *Whatsapp*, sem a necessidade de ligações por telefone.

A comunicação através do envio de mensagens por aplicativos (apps) como *Whatsapp* citado acima vem sendo atualmente muito utilizada nos últimos anos. A pesquisa da TIC Domicílios é responsável por mapear acessos a tecnologia de informação e comunicação no Brasil revelou que no ano de 2019 teve um aumento de 28% em relação ao ano de 2015 no consumo de internet através de dispositivos móveis. Atualmente os smartphones vem sendo a ferramenta mais utilizada para

[1] https://www.linkedin.com/in/danilo-silva-3489311b4/
[2] http://lattes.cnpq.br/2017611396853275 / https://www.linkedin.com/in/carlos-barbosa-gomes-filho-b0463542/

navegar na internet apesar das suas limitações o número de conectados só aumenta com o passar dos anos. Tendo isso em mente a forma de mensageira do Schedule System que proporciona uma interação flexível sendo para usuários de dispositivos moveis e desktops (Computadores) uma das vantagens já que é uma sistema web.

Desse modo, o desenvolvimento da aplicação contará com as tecnologias que estão em alta no mercado atualmente, com uma comunidade ativa e atualizações constantes. As linguagens utilizadas serão Java, em nosso caso para web, o framework Spring, React JS uma biblioteca JavaScript junto com BootStrap para estilização e PostgreSQL para armazenamento e gerenciamento de dos dados da aplicação.

2. JAVA

A linguagem de programação Java foi criada no anos 90 pela organização Sun Mycrosystems, atualmente os direitos da tecnologia pertence a Oracle a instituição os comprou no ano de 2010.

Uma das característica da linguagem Java é que ela é totalmente OO (Orientada a Objeto) por isso se torna fácil realizar manutenção e aplicar o reuso no código fonte desenvolvido, outra qualidade desta tecnologia é que um sistema desenvolvido nela não é executado em uma plataforma nativa, ou seja, programas criados com Java são compilados em Bytecode (Bytecode são códigos intermediários entre o código que foi desenvolvido pelo programador e a linguagem de máquina, ou seja, o Bytecode é o código do desenvolvedor em um idioma que o computador compreende.) e executados através da JVM (Java Virtual Machine) a máquina virtual do Java, isso permite que os sistemas desenvolvidos podem ser executados em qualquer plataforma que suporte à tecnologia.

Essa tecnologia é usada para desenvolver aplicações Desktop que seria programas que necessitam de instalação para que o usuário possa utiliza-lo e aplicações Web que são executadas em um Browser (Navegador de Internet). O Java e o JavaScript não são iguais, o JavaScript é uma tecnologia apenas para criar páginas web que só pode ser executada no Browser. Além da criação de sistema o Java possibilita desenvolver jogos, realizar upload de imagens, bate-papos on-line e serviços on-line como por exemplo treinamentos, formulários, transações bancárias e etc.

Segundo França 2020 a linguagem de programação Java está na posição 3º das 20 linguagens mais populares do mundo, é uma tecnologia que está ativa no mercado e que sempre está sofrendo melhorias. A popularidade e os recursos que a tecnologia fornece deram vida a softwares como o EFD ICMS/IPI um sistema desktop criado pelo governo para validar o Sped Fiscal mensal gerados por empresas e o tão conhecido Minecraft um jogo de sobrevivência que possibilita o jogador criar seus próprios cenários e muito mais, o jogo foi desenvolvido pela Mojang Studios.

Para começarmos a desenvolver em qualquer linguagem não apenas em java, precisamos de uma IDE (Integrated Development Environment). A IDE também conhecida como o Ambiente de Desenvolvimento Integrado é aonde iremos escrever nossos códigos, existe diversos tipos de IDE possibilitando um leque muito grande de escolho.

Alguns exemplo de IDE para utilizar com Java: Eclipse, Intellij IDEA, NetBeans, Blue J e etc. São muitas disponíveis.

2.1 SPRING FRAMEWORK

O *Spring* é um *framework open-source (*Código aberto*)* que teve sua primeira versão no ano de 2002, desenvolvido por Pivotal Software e mais tarde licenciado pela Apache License.

Inicialmente o Spring foi desenvolvido para a plataforma Java baseado em *Design Patterns* (Padrões de Projetos) com o intuito de acelerar, melhorar e resolver problemas frequentes que ocorre no desenvolvimento de software. Com o surgimento desta ferramenta problemas como inversão de controle que se diz respeito a delegação de responsabilidade no código do sistema foi resolvido facilitando a manutenibilidade e adição de recursos futuros, e a injeção de dependência que se refere a autonomia de uma determinada classe, ou seja, o quanto essa classe precisa de outra para funcionar, isto está ligado a inversão de controle.

O Spring fornece tecnologias do seu ecossistema que facilita a criação de softwares trazendo o padrão REST (*Representational State Transfer*) e o conceito de API (*Application Programming Interface*). O padrão *REST* tem o objetivo de instruir o desenvolvedor a criar aplicações web seguindo boas práticas durante o processo, e

o conceito de API se refere a interação entre dois sistema sem a necessidade de conhecimento sobre a implementação entre eles. Através desse padrão e do conceito que o Spring trouxe consigo, surgiram as *REST-API* que nada mais é do que a comunicação entre duas aplicações seguindo o padrão arquitetural *REST*, atualmente quando acessamos uma aplicação web muito provavelmente que está aplicação está configurada com os padrões *REST*.

Para exemplificar o como o padrão *REST* funciona citarei um exemplo:

I. Ao acessar uma url (Endereço da página) em seu navegador (Chrome, Firefox, Edge, etc).
II. O navegador inicia uma comunicação com o servidor de destino através da url informada.
III. O servidor recebe e interpreta a sua request (Pedido enviado na url) e processa uma response (Resposta que será enviada pelo servidor).
IV. O servidor pode responder “200 Success” e retorna o que foi solicitado, ou pode responder “404 Not Found” informando que o endereço/recurso que foi solicitado não foi encontrado.

Esse processo ocorre toda vez que acessamos alguma pagina na web.

O Framework Spring que foi criado inicialmente apenas para plataforma Java em versões atuais está disponível também para integração com as tecnologias como *Kotlin* e *Groovy*, para que possamos usar os recursos que o spirng fornece precisamos instalar o *plug-in* na IDE (*Integrated Development Environment*) que está sendo usada ou utilizar a IDE que eles fornecem conhecida como STS (*Spring Tools Suite*) ambos permite utilizar os recursos do ecossistema *Spring*.

2.2 JAVASCRIPT

O *JavaScript* também popularmente conhecido como JS é uma linguagem de programação interpretada, OO (Orientada a Objetos) e de alto nível, criada por Brandan Eich nos anos 90 seguindo o paradigma funcional e de scripts.

O JS mais conhecida como linguagem de *script* da internet obteve essa fama por ser a tecnologia nativa compilada nos *Browsers* (Navegadores de Internet). A tecnologia possibilita customizar uma página web estática transformando-a em dinâmica

permitindo a criação de efeitos animados, mapas e etc. Apesar de ser comumente utilizada no *Front-End* (Interface gráfica da página Web que o usuário está acessando) o JS também pode ser utilizada para criação de regra de negócio, ou seja, a criação do *Back-End* (Camada responsável por tratar as solicitações feitas através do *Front-End*).

O funcionamento da linguagem se base no conceito de client-side, ou seja, a execução provem do lado do cliente mais especificamente o navegador em que o usuário está acessando determinada página. Isto quer dizer que as ações são executadas na máquina do usuário, claro que esse processo ocorre apenas quando o cliente não envia nenhum dado para algum servidor externo, sendo assim apenas o processamento visual da página é mais rápido.

A popularidade do JS veio nos anos 2000, quando o *Facebook* utilizou a tecnologia na construção da rede social que ficou mundialmente famosa. Outras organizações também usufruíram do poder que o Java Script fornece, por exemplo sites que utilizam a tecnologia tanto no seu *Front-end* como no *Back-End*: Ebay e Yahoo.

A evolução da tecnologia abriu o leque para utilização da mesma nas areais de desenvolvimento de jogos, mobile e servidor. O crescimento da linguagem também deu surgimento a framework muito usados, como por exemplo: VueJS, React Native, PhoneGap, Ionic Framework, Flutter e etc.

2.3 REACT JS

O *React JS* é uma biblioteca do *JavaScript* de código aberta mantida pelo *Facebook*, *Instagram* e outras empresas, teve seu surgimento em 2011, quando foi utilizado em tela da rede social Facebook, porém consideram que seu lançamento só ocorreu em 2013.

> O maior erro que cometemos como empresa foi apostar demais em HTML5 em oposição ao nativo (Mark Zuckerberg, 2012, On-line).

Apesar de muitos programadores citarem o *React* como um *framework* ele é uma biblioteca eficiente e flexível para a construção de interfaces web. A tecnologia utiliza o conceito de reuso, onde quando se implementa um componente como por exemplo um input (campo de texto) o mesmo pode ser usado em inúmeras páginas

da aplicação sem a necessidade de criar um novo. Essa flexibilidade que o *React* proporciona é o que faz dele uma biblioteca bastante conhecida e com uma comunidade vida proporcionando a facilidade de tirar dúvidas e buscar soluções online, claro que para começar a desenvolver utilizando os recurso do react é necessário que o indivíduo tenha uma base de *JavaScript, CSS* e *HTML*.

A biblioteca é dividida em duas versões web e mobile, a versão mobile denominada de *React Native* criada pelo *Facebook* com foco para sistemas *Android* e *IOS* de forma nativa, a versão web como já citada *React JS* pode ser usada com *JavaScript* (JS) e *TypeScript* (TS), uma breve explicação sobre TS ele é utilizado para desenvolvimento em grande escala com sua base em JS o *TypeScript* dá ao desenvolvedor uma nova forma de escrever código que no final é executado em JS.

Para mostrar o quão grande é o poder e a popularidade da biblioteca citaremos algumas aplicações *web* que atualmente utilizam *React* no desenvolvimento de suas interfaces. Essas são: Netflix, Whatsapp Web, Feedly, Airbnb, SeatGeek, HelloSign, Walmart e etc.

Para a criação das interfaces deste projeto usaremos a versão web mais precisamente React JS com JavaScript.

2.4 BOOTSTRAP

O *Bootstrap* é uma *framework* de código aberto criando no ano de 2010 e lançado apenas no ano de 2011, desenvolvido por Mark Otto e Jacob Thornton dois funcionários dá *Twitter* uma das maiores redes sociais do mundo.

A criação do *framework* teve inicialmente a ideia de padronizar interfaces gráficas de páginas web. A ferramenta ganhou fama no ano de 2012 sendo uma das mais populares do *Github*.

A ferramenta disponibiliza uma grande gama de estilização pronta em CSS (*Cascading Style Sheets*), tecnologia usada para aplicar estilos em páginas web. Além disso é possível realizar a implementação de componentes animados que contém *JavaScript* ou *JQuery* em sua composição de forma fácil, por exemplo *slideshows e dropdowns são componentes que utilizam css e jquery em sua implementação. A agilidade que o Bootstrap proporciona para a construção de uma página web é o que faz dele uma ferramenta importante para a criação de design de interfaces trazendo os princípios de usabilidade fornecendo páginas com uma*

estética agradável. Citaremos algumas vantagens e desvantagens do bootstrap, vamos inciar com as vantagens:

I. *Disponibiliza uma vasta biblioteca de componentes.*
II. *Permite o reuso de código.*
III. *Trabalha com o conceito de responsividade (Adequação da página ao tamanho da tela).*
IV. *Tem uma documentação rica e de fácil entendimento.*
V. *Possui comunidade ativa.*

Desvantagens:

I. Excesso de padronização: Apesar da padronização ser o foco para qual o Bootstrap foi desenvolvido, isso faz com que possa existir sites com estilização similar.

II. Excesso de Código: Desenvolvedores mais experiente optam por não utilizar muito o framework já que o ele carrega muito código em alguns caso podendo afetar o carregamento da aplicação.

Essas são algumas das vantagens e desvantagens de usar o *Bootstrap* para a criação de páginas *web*. O *framework* se torna uma ótima opção de estilização se o desenvolvedor está buscando praticidade e agilidades.

2.5 Node JS

Node ou *Node.js* não é um linguagem de programação, mas sim um ambiente de execução*.* Desenvolvido pela instituição Node.js em conjunto com a Linux Foundation no ano de 2009 ganhou o título de *server-side* a primeira plataforma criada que permitia a execução de códigos *JavaScript* no *back-end* sem a necessidade de um *browser.*

Apesar de ser considerada recente, empresas como Netflix e LinkedIn já usufrui dos benefícios da plataforma em seus projetos o que a torna mais popular no mercado. Além de instituições de renomes utilizarem o *Node* outras características contribui para sua fama, uma delas é a capacidade de escala e flexibilidade o que

faz dela uma boa escolha para trabalhar em conjunto com o desenvolvimento de Micros-serviços.

Por sua execução ser em *single-thread (*Apenas um Núcleos*)*, ou seja, não demandar muito processamento do servidor seus concorrentes como PHP, C# e outros ficam para trás quando o termo é desempenho já que os mesmos trabalham com *multi-thread* (Vários Núcleos). Apesar de ser ter apenas um núcleo no lado do servidor é possível realizar o tratamento de requisições de forma performática para a aplicação, isto porque o Node gerencia as operações em forma de entrada e saída, ou seja, gerenciamento de forma assíncrona dessa forma o núcleo de processamento não é impactado.

2.6 VENOM

O *venom* é uma biblioteca criada em *JavaScript* para realizar interações com *whatsapp* e *telegram*, existe uma versão chamada de *venom-insta* para integrar aplicações com o *instagram*. O venom é um projeto *open-source*, ou seja, seu código fonte é aperto para a comunidade contribuir com melhorias. Atualmente conta com mais de 15 mil *downloads* por mês na sua versão 3.0.16 e uma comunidade ativa.

A biblioteca disponibiliza funções para atendimento ao cliente, envio de arquivos, reconhecimento de palavras chaves através de inteligência artificial fornecida pelo próprio *venom* e etc. Todos essas funcionalidade via *chat*, podendo ser utilizado em projetos através do *Node,* com uma documentação simples e intuitiva integrar o *venom* em aplicações para envio de mensagens se torna fácil.

2.7 POSTGRESQL

PostgreSQL é uma ferramenta de código aberto utilizada para gerenciamento de banco de dados também chamado de SGBD (Sistema de Gerenciamento de Banco de Dados), teve seu ano de lançamento em 1996 e foi criado pela PostgreSQL Global Development Group fundada no ano de 1986.

Conhecido popularmente também como pgAdmin, a ferramenta interpreta a linguagem SQL (*Standard Query Language*) que é utilizada para realizar operações nos Bancos de Dados. O pgAdmin fornece suporte para várias plataformas de

sistemas operacionais tais como *Windows, Linux, Solaris, Mac OS* e outros, por ser multiplataforma faz com que sua popularidade seja maior no mercado de tecnologia.

O funcionamento padrão do pgAdmin é direto no Browser (Navegador de Internet), ao instalar a ferramenta e executa-la será inicializada no Browser o servidor de banco de dados, como se estive-se acessando uma página web, mas nesse caso será o SGBD que estará disponível para criação e manipulação de dados armazenados.

Ao falarmos de BD e SGBDs devemos nos atentar para os tipos de BD existentes, então citaremos aqui o BD relacional e o não relacional mais conhecido como noSQL. O BD relacional que aqui usado como exemplo o “pgAdmin” tem seu funcionamento através de tabelas que ser relacionam entre si através de chaves denominadas PK(Chave Primária ou *Primary Key*) e FK(Chave estrangeira ou *Foreign Key*), já o BD não relacional que chamaremos a partir daqui de noSQL não utiliza tabelas para armazenamento dos dados, citaremos aqui o “MongoDB” um BD também muito popular, a forma de guardar os dados de um BD noSQL é estruturada em JSON (JavaScript Object Notation), ou seja, o armazenamentos dos dados são em formato de texto que quando acessado por uma sistema é transformado em objeto e dessa forma é possível relacionar as informações.

Apesar de ambos serem ferramentas criadas com o mesmo intuito que seria o armazenamento de informações, os BD relacionais contam com alguns recursos que os noSQL não disponibilizam. Citaremos alguns desses recursos:

I. Herança entre Tabelas
II. Suporta transações com um alto nível de dados
III. Trabalha com ORM (Object-Relacional-Mapping)
IV. Possibilita a criação de procedures (Procedimentos a serem executados no próprio BD).
V. Possibilita a criação de Trigger (Catilhos que são executados após um determinado evento).
VI. Possibilita a criação de Views (View são tabelas Virtuais cujo sua função é apenas exibição dos dados).

Esses são alguns dos recursos que os BDs relacionais carregam e que inclusive o pgAdmin utilizado para o desenvolvimento deste projeto traz embarcado.

3 METODOLOGIA

Para o desenvolvimento de um software é necessário debater sobre os requisitos do sistema.

E o que seriam esses requisitos? Os requisitos de sistema são a definição de regras e recursos que o sistema deve ter embarcado. Para discutir essas definições é importante que tenham uma reunião entre o cliente (gerentes e funcionários que tenha conhecimento técnico sobre o ramo de negócio) e o gerente de projetos e os profissionais técnicos responsáveis pela elaboração do documento sobre o sistema.

O processo citado a cima leva a especificação dos requisitos funcionais e não funcionais.

Requisitos funcionais: São características ou funções que devem atender o proposito para qual o sistema foi desenvolvido, ou seja, ele se diz a respeito de “O sistema deve fazer o que?”.

I. Tabela 1 – Requisito Funcional

IDENTIFICADOR	NOME	DESCRIÇÃO	PRIORIDADE
RF01	Controle de Acesso	O acesso ao sistema será gerenciado através do *login* e senha para que o usuário que está acessando não tenha acesso as tarefas de outros usuários.	Essencial
RF02	Gerenciar Usuários	O sistema deve realizar listagem, cadastros, atualizações e exclusões dos usuários quando necessário.	Essencial
RF03	Gerenciar Tarefas/Reuniões	O sistema deve realizar listagem, cadastros, atualizações e exclusões das tarefas quando necessário.	Essencial
RF04	Gerenciar Permissões	O sistema deverá validar as	Importante

		permissões dos usuários para determina quais recursos dos sistema eles tem permissão para acessar.	
RF05	Gerenciar Histórico de Tarefas	O sistema deverá armazenar todas as tarefas independe dos status em que se encontra	Importante
RF06	Envio de Mensageria aos Usuários	O sistema deve enviar notificações ao usuário sobre as tarefas que são designadas a ele, as mensagens serão enviadas através de *E-mail* e *Whatsapp*.	Importante

Fonte: Próprio Autor

Requisitos não funcionais: É a definição de como o sistema dever fazer para realizar ou executar um requisito funcional. Para melhor entendimento sabemos que os requisitos funcionais são funcionalidades que o sistema terá, e o não funcional será a definição de qual maneira ele irá efetuar a execução.

II. Tabela 2 - Requisitos Não Funcionais

IDENTIFICADOR	NOME	DESCRIÇÃO	PRIORIDADE
RNF01	Plataforma	O sistema será desenvolvido na linguagem Java junto com framework Spring para a criação da aplicação web.	Importante
RNF02	SGBD	O SGBD utilizado será o postgreSQL 12. O postgreSQL tem uma interface amigável e fácil	Importante

		configuração.	
RNF03	Hardware	Para deploy usaremos os servidores da Heroku. Porém o ideal seria: Processador de 16 núcleos e 32 threads, 32 gb de memória ram e 2 TB para armazenamento.	Essencial
RNF04	Backup de Dados	O sistema deve realizar backups diários para garantir a segurança dos dados dos clientes.	Importante
RNF05	Acesso a Rede de dados	Será necessário acesso à internet para utilização de recursos do sistema.	Essencial

Fonte: Próprio Autor

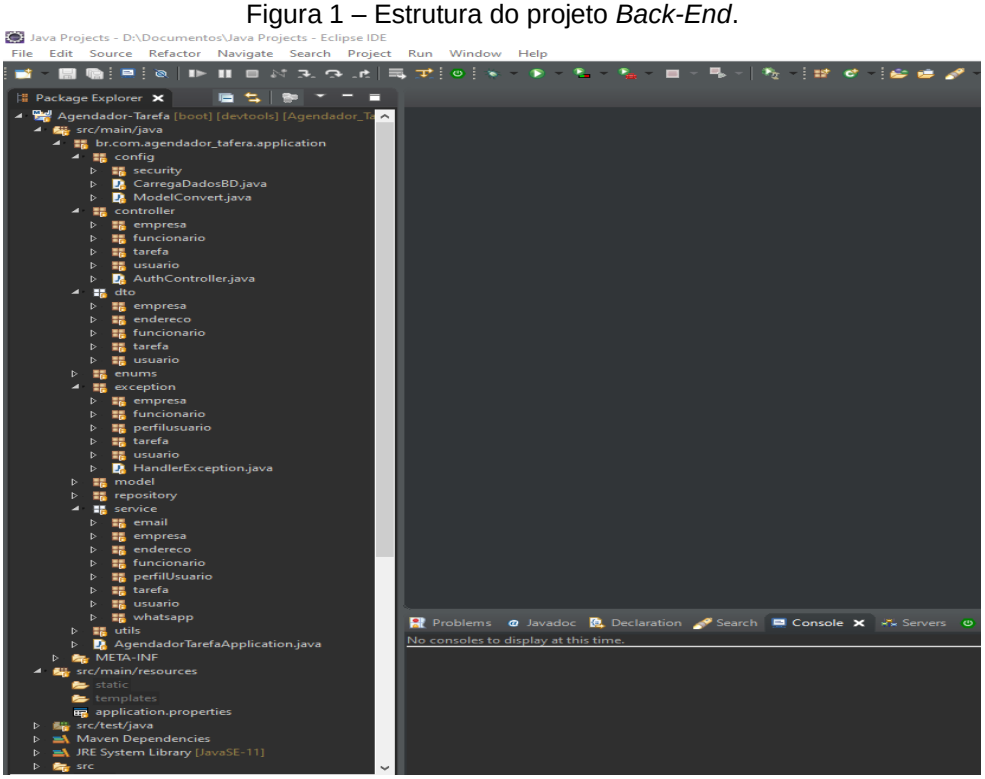
4 Schedule System

Schedule System é uma sistema web de agendamento focado para empresas, mas que possibilita um usuário comum também usufruir das funcionalidades do software. A ideia é que o sistema auxilie no dia-à-dia a marcar atividades para funcionários que trabalham constantemente na rua e agendar reuniões com fornecedores por exemplo.

Apesar do sistema ter um interface gráfica agradável e simples, caso o cliente (empresa) não queria utilizá-la pode ser feita a integração da parte deles com a API do Schedule System, sendo assim o cliente pode consumir os recursos do sistema porém com sua própria interface gráfica.

O diferencial do sistema é a forma de sinalizar os convidados de uma reunião ou de funcionários encarregados de alguma atividade, utilizando um dos meios de comunicação mais popular hoje em dia, o sistema será responsável por enviar uma mensagem no Whatsapp e no Email do funcionário ou do convidado passando as informações necessárias para o mesmo.

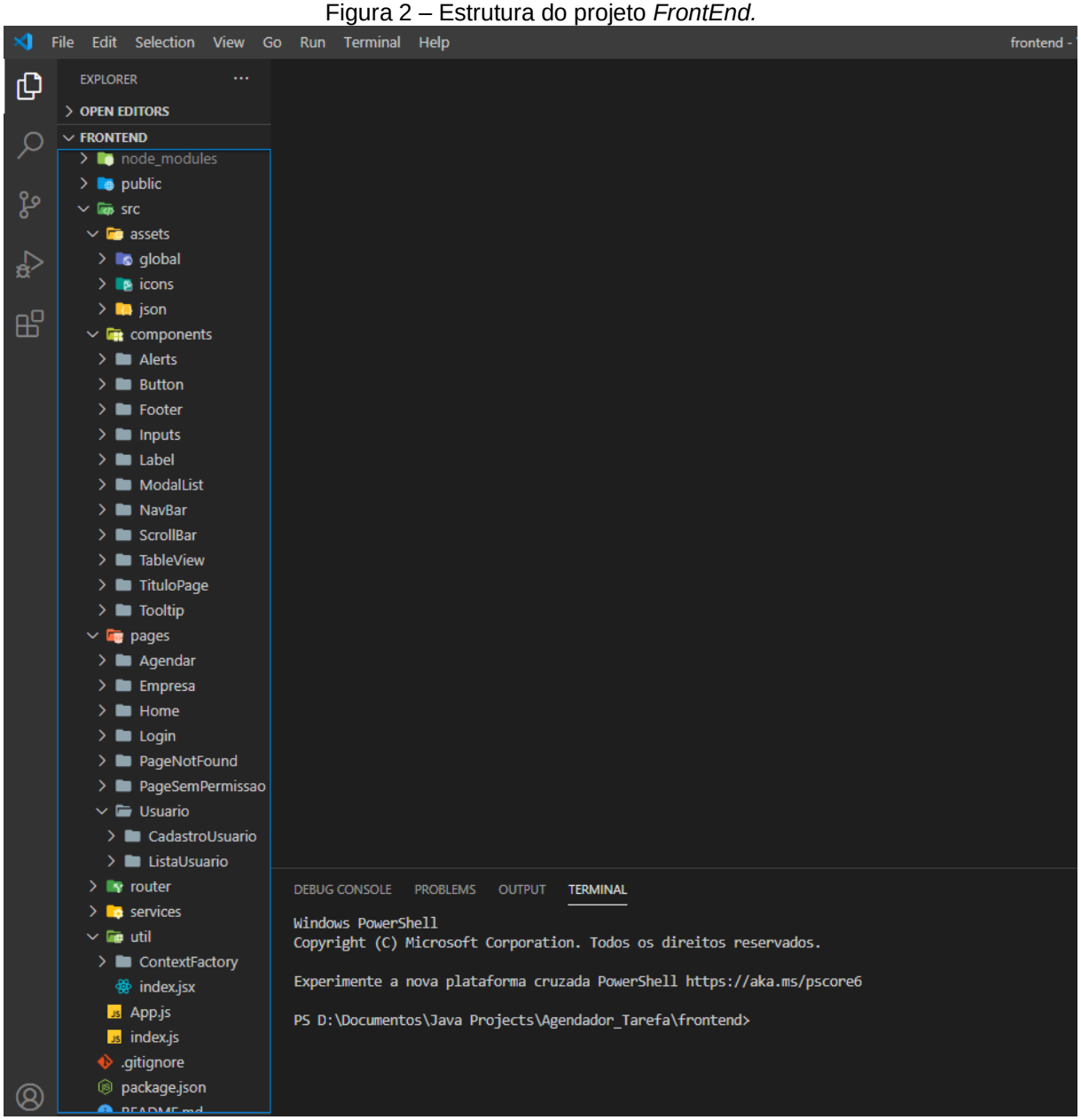
Figura 1 – Estrutura do projeto *Back-End*.



Fonte: Próprio Autor

A figura 1 mostra a estrutura do projeto *back-end* criado em Java na versão 11 junto com Spring Framework na IDE Eclipse.

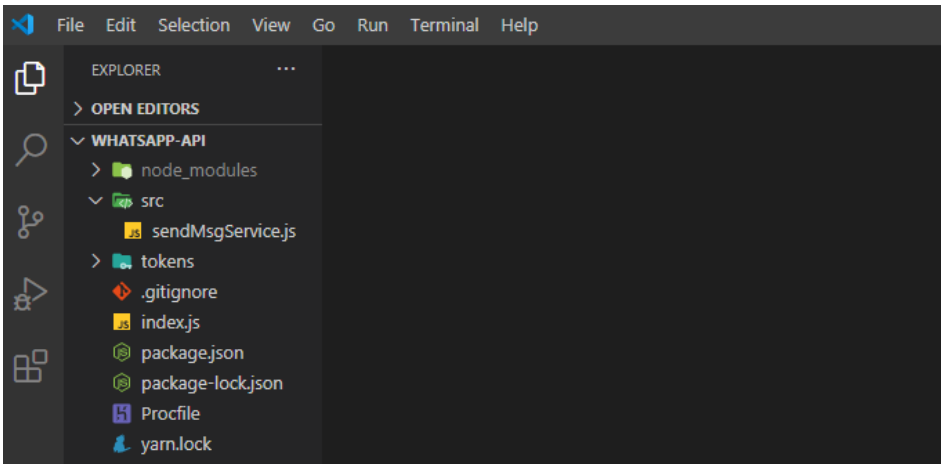
Figura 2 – Estrutura do projeto *FrontEnd.*



Fonte: Próprio Autor

A figura 2 mostra a estrutura do projeto *frontend* criado em React funcional na versão 17 com JavaScript na IDE VSCode.

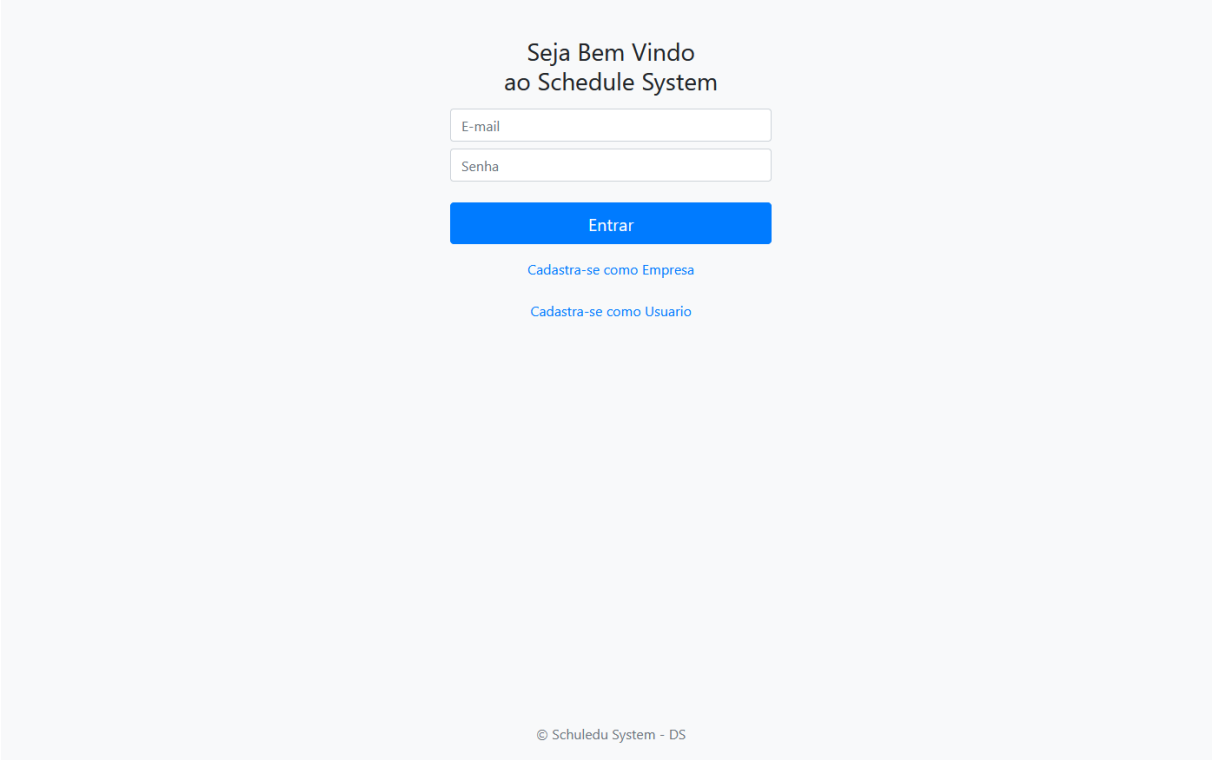
Figura 3 – Estrutura do Serviço de Envio de Mensagem para Whatsapp.



Fonte: Próprio Autor

A figura 3 mostra a estrutura do projeto responsável por receber os telefones dos usuários/convidados a aplicação Java e notifica-los via *Whatsapp*.

Figura 4 – Pagina de *Login* do sistema.



Fonte: Próprio Autor.

A figura 4 mostra a página de *login* com um *design* simples, a forma de autenticação do usuário é através de E-mail e Senha. Após informar os respectivos dados o *frontend* comunica-se com o backend que por sua vez caso as informações sejam validas devolverá um Token JWT (*JSON Web Token)* junto com outros informações segue abaixo uma imagem ilustrativa.

Figura 5 – Retorno do *Backend* após validar usuário.


```
{
    "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJlbXByZXNhMUBnbWFpbC5jb20iLCJpYXQiOjE2MjE3M.2vshfepvauiK-3auvekhsS70L3ZkPoUT5CpapTFcxb_dV2SsjyXrDaCLSHBPd4bRuM1JoTp4K_-_vGm5eao9RQ",
    "type": "Tipo do Token",
    "id": "ID do Usuario",
    "email": "E-mail do Usuario",
    "empresa": "Id da empresa",
    "permissao": ["Lista de Permissões"]
}
```


A imagem acima ilustra a resposta do *backend* após validar o E-mail e a Senha informado pelo usuário. O corpo da resposta é composto por um Token JWT como já mencionado, o tipo do Token que seja usado para novas requisições para o backend, o id que se refere ao código do usuário, o e-mail do usuário, o id da empresa caso usuário que esteja realizando *login* seja vinculado a uma e as permissões que o usuário possui, todos esses dados são salvos no *local storage* do negador.

Figura 6 – Validação de Permissões no *FrontEnd*

AuthRoutes.jsx

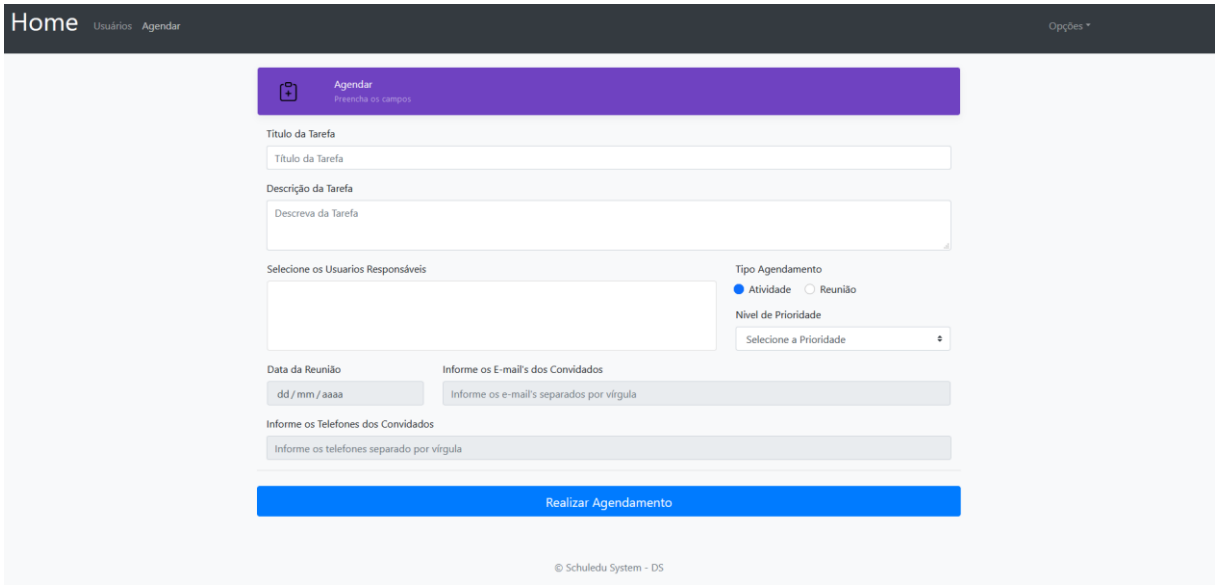
src > router > AuthRoutes.jsx > ...

```jsx
import { Route, Redirect } from "react-router-dom";
import {homePath, semPermissaoPath} from '../util';

const permissao = (path) => {
    let pathsUser = [homePath];
    let permissaoUsuario = [];
    let autorizado = false;
    let permissaoTela;

    permissaoUsuario.push(localStorage.getItem('permissao'));
    permissaoUsuario.forEach(p => {
        let permissao = p.replace(/"/g,"").replace("[","").replace("]","");
        if(permissao === "ROLE_FUNC"){
            permissaoTela = permissao;
        }
    })

    if(permissaoTela === "ROLE_FUNC" && permissaoUsuario.length === 1){
        pathsUser.forEach(e => {
            if(e === path){
                autorizado = true;
            }
        })
        return autorizado;
    }else{
        autorizado = true
        return autorizado;
    }
}

export const PrivateRoutes = ({component : Component, ...rest}) => {
    return <Route {...rest}
        render = {props =>
            permissao(rest.path) ?
            (<Component {...props} />)
            :
            (<Redirect to = {{pathname : semPermissaoPath, state : {from : props.location}}} />)
        }
    />
}
```

Fonte: Próprio Autor

A figura 6 acima apresenta o código responsável por validar se o usuário logado tem permissão para acessar um determinado recurso na tela. A validação é feita através das permissões salva no local storage do navegador, no caso apenas usuários com permissão de funcionário são restritos de algumas ações do sistema.

Figura 7 – Tela de Agendamento



Fonte: Próprio Autor

A figura 7 representa a tela de agendamento em que o usuário pode optar por agendar uma atividade para um funcionário por exemplo ou uma reunião para fornecedores por exemplo. Supondo que o usuário escolha agendar uma atividade par um funcionário, após preencher os campos necessários e selecionar um ou mais responsáveis, os funcionário associado a essa atividades receberam um *E-mail* e uma mensagem no *Whatsapp.*

Figura 8 – Código Responsável por Salvar o agendamento

```
@Transactional
public TarefaResponseDTO salvarAtividade(TarefaRequestDTO tarefaRequest) {

    Empresa empresa = null;
    List<Usuario> usuarios = null;

    if(!ObjectUtils.isEmpty(tarefaRequest.getEmpresa())) {
        empresa = empresaService.setarEmpresa(tarefaRequest.getEmpresa().getCnpj(), tarefaRequest.getEmpresa().getId());

        usuarios = usuarioService.validarUsuariosTarefa(tarefaRequest.getUsuario());

        funcionarioService.validaFuncionarioEmpresa(usuarios, empresa.getId());
    }
    else {
        usuarios = usuarioService.validarUsuariosTarefa(tarefaRequest.getUsuario());
    }

    AgendarTarefa agendarTarefa = AgendarTarefa.builder(empresa, usuarios, tarefaRequest, TipoAgendamento.ATIVIDADE);

    agendaRepository.save(agendarTarefa);

    whatService.enviarMsg(pegarNumerosUsuarios(usuarios) , montaMensagem(agendarTarefa, Utilitarios.CRIACAO));

    enviarEmailTarefa(agendarTarefa, Utilitarios.CRIACAO,TipoAgendamento.ATIVIDADE);

    return tarefaToDto(agendarTarefa);

}
```

Fonte: Próprio Autor

A figura 8 demostra o código responsável por persistir uma atividade no banco de dados, antes de salvar é feita algumas validações, por exemplo se o agendamento tiver associado a alguma empresa será validado se os funcionários associados a atividade tem vínculo com a empresa caso tenha atividade será salva e a rotina envia uma notificação sobre a atividade via *Whatsapp* e no *E-mail*, caso

contrário se o funcionário associado a atividade não tiver vínculo com a empresa em questão não será possível salvar a atividade, essa validação previne que funcionarios de outras empresas sejam associados a atividades de empresas diferentes da que o mesmo faz parte.

Figura 9 – Serviço de Notificação de *E-mail.*
Fonte: Próprio Autor

A figura 9 demostra o código que envia o e-mail de acordo com o tipo de agendamento, será montado o *e-mail* com o título e o corpo da mensagem. O email utilizado como remetente é um e-meio criado próprio para o sistema

```
public List<String> enviarEmail(String Titulo, String corpoMensage, List<?> emailsDesnitario, TipoAgendamento tipo){

    emailNaoEnviado = new ArrayList<>();

    if(tipo.compareTo(TipoAgendamento.ATIVIDADE) == 0) {
        enviarEmailsUsuario(Titulo, corpoMensage, (List<Usuario>) emailsDesnitario);
    }
    else {
        enviarEmailsConvidados(Titulo, corpoMensage, (List<String>) emailsDesnitario);
    }


    return emailNaoEnviado;
}
```

(schedulesystem@gmail.com).

Figura 10 – Serviço de Notificação *Whatsapp*

```
    sendMsg : async (numeros, text) => {
        numeros.forEach(n => {
            client.sendText('55' + n + '@c.us', text).then((resp) => {
                console.log("Sucesso: ", resp);
            }).catch((respError) => {
                console.error("Error: ", respError);
            });
        });
    }
```

Fonte: Próprio Autor

A figura 10 mostra o código utilizado para enviar as mensagens via *whatsapp* para a lista de números recebida da aplicação *Java (Back-end)* informando os funcionários se caso for uma atividade ou convidados se for uma reunião. Com esses dados é feito o mapeamento da lista de telefones e enviado a cada um deles as respectivas informações.

5 CONSIDERAÇÕES FINAIS

O presente artigo tem como objetivo apresentar um sistema para auxiliar no agendamento de atividades diárias com foco em empresas mas que usuários comuns também possa usufruir dos benefícios do sistema. A proposta do sistema é gerenciar atividades e reuniões de forma simples, prática e organizada utilizando *whatsapp* e *e-mail* como canais de comunicação.

As tecnologias escolhidas para a criação do sistema favorecem o desenvolvimento web e mobile que está em alta no mercado de tecnologia. A ideia inicial é de um sistema web responsivo, possibilitando o acesso ao mesmo por dispositivos móveis.

Atualmente o protótipo do sistema já conta com o agendamento e notificação de atividades e reuniões, porém outras funcionalidades como edição, deleção de informações e muitas outras estará disponíveis em versões futuras devido ao tempo de desenvolvimento curto.

6 REFERÊNCIAS

ALFF, Chico. **O que são Requisitos Funcionais e Não Funcionais?** 2018. Disponível em: https://analisederequisitos.com.br/requisitos-funcionais-e-nao-funcionais/. Acesso em: 20 fev. 2021.

BELLAFRONTE, Andre. **Quais as melhores IDE's de Java**. 2018. Disponível em: http://www.elaborata.com.br/blog/2018/08/16/quais-as-melhores-ides-de-java/. Acesso em: 07 mar. 2021.

BRITO, Edivaldo. **Java: Entenda para que serve o software e os problemas da sua ausência**. 2014. Disponível em: https://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/11/java-entenda-para-que-serve-o-software-e-os-problemas-da-sua-ausencia.html. Acesso em: 07 mar. 2021.

BRASIL, Udacity. **React: o que é e como funciona essa ferramenta?** 2018. Disponível em: https://tableless.com.br/react-o-que-e-e-como-funciona-essa-ferramenta/. Acesso em: 12 abr. 2021.

COMUNIDADE, Facebook e. **React Native**. 2020. Disponível em: https://pt.wikipedia.org/wiki/React_Native. Acesso em: 12 abr. 2021.

DIAS, Emílio. **4 Conceitos sobre REST que Qualquer Desenvolvedor Precisa Conhecer**. 2016. Disponível em: https://blog.algaworks.com/4-conceitos-sobre-rest-que-qualquer-desenvolvedor-precisa-conhecer/. Acesso em: 07 abr. 2021.

FRANÇA, Renan. **TOP 20: As linguagens de programação em alta em 2021**. 2020. Disponível em: https://pt-br.classpert.com/blog/linguagens-de-programacao-mais-usadas. Acesso em: 07 mar. 2021.

GEEKHUNTER. **Spring Framework: o que é, seus módulos e exemplos!** 2020. Disponível em: https://blog.geekhunter.com.br/spring-framework/. Acesso em: 07 abr. 2021.

GEEKHUNTER. **Design Patterns: seu código com mais qualidade e elegância**. 2020. Disponível em: https://blog.geekhunter.com.br/design-patterns/. Acesso em: 07 abr. 2021.

GONÇALVES, Leandro da Costa. **Engenharia de Requisitos**. 2011. Disponível em: http://www.semeru.com.br/blog/category/requisitos-de-sistema/. Acesso em: 20 fev. 2021.

LEONE, Leonello de. **Bootstrap: o que é, porque usar e como começar com o framework**. 2018. Disponível em: https://becode.com.br/bootstrap-o-que-e-porque-usar-e-como-comecar/. Acesso em: 09 abr. 2021.

LEONE, Leonello de. **WTF is TypeScript + 6 aplicações extraordinárias da linguagem!** 2018. Disponível em: https://becode.com.br/o-que-e-typescript-e-aplicacoes-da-linguagem/. Acesso em: 13 abr. 2021.

LENON. **Node.js – O que é, como funciona e quais as vantagens**. 05/09/2018. Disponível em: https://www.opus-software.com.br/node-js/. Acesso em: 21 maio 2021.

MULTIEDRO. **PostgreSQL: o que é e como ele melhora a produtividade na empresa?** 2021. Disponível em: https://blog.multiedro.com.br/postgresql/. Acesso em: 09 abr. 2021.

ORACLE. **O Que É um Banco de Dados Relacional**. 2021. Disponível em: https://www.oracle.com/br/database/what-is-a-relational-database/. Acesso em: 11 abr. 2021.

PENA, Rodolfo F. Alves. **Terceirização e trabalho - Brasil Escola**. Disponível em: https://brasilescola.uol.com.br/geografia/terceirizacao-trabalho.htm. Acesso em 11 de abril de 2021.

POSTGRESQL. **O que é o PostgreSQL?** 2021. Disponível em: http://pgdocptbr.sourceforge.net/pg82/intro-whatis.html. Acesso em: 11 abr. 2021.

RICARTE, Ivan Luiz Marques. **Bytecodes**. 2000. DCA/FEEC/UNICAMP. Disponível em: https://www.dca.fee.unicamp.br/cursos/PooJava/javaenv/bytecode.html. Acesso em: 07 mar. 2021.

ROSA, Giovanni Santa. **Maioria das pessoas que usa internet no Brasil se conecta exclusivamente pelo celular**. 2020. Disponível em: https://gizmodo.uol.com.br/pesquisa-tic-domicilios-2019-celular/. Acesso em: 12 abr. 2021.

ROVEDA, Ugo. **JavaScript: o que é, para que serve e como funciona o JS?** 2020. Disponível em: https://kenzie.com.br/blog/javascript/. Acesso em: 09 abr. 2021.

SOUZA, Ivan de. **Entenda o que é Rest API e a importância dele para o site da sua empresa**. 2020. Disponível em: https://rockcontent.com/br/blog/rest-api/. Acesso em: 09 abr. 2021.

WALKE, Jordan. **React (JavaScript)**. 2013. Disponível em: https://pt.wikipedia.org/wiki/React_(JavaScript). Acesso em: 11 abr. 2021.

ZANLUCA, Júlio César. **A Consolidação das Leis do Trabalho - CLT**. 2021. Disponível em: http://www.guiatrabalhista.com.br/tematicas/clt.htm. Acesso em: 12 abr. 2021.

SISTEMA PARA GERENCIAMENTO DE PEDIDOS E PRODUTOS ARTESANAIS

SILVA, PAULO HENRIQUE ROQUE DA
SOUSA, HERCILIO DE MEDEIROS [1]
ROCHA, GLÁUCIO BEZERRA[2]

INTRODUÇÃO

Nós dias atuais com a situação financeira que estamos vivendo no Brasil e com a pandemia do novo Covid-19, os micros/pequenos empreendedores que vendiam seus produtos em “porta em porta” como fonte principal de renda ou renda extra, estão se reinventando com a ajuda da tecnologia.

Hoje muitos com ajuda das redes sociais estão divulgando seus produtos na internet, foi a partir daí que surgiu uma ideia que ajudaria a automatizar e facilitar a gerência de seus produtos artesanais que serão produzidos, elaborando um sistema que apresentará os seus produtos e a partir dele possam armazena os pedidos de seu cliente sem ter que fazer anotações em caderno.

Vai ser utilizada a linguagem de programação *Java* pelo motivo dela está entres as três linguagens mais utilizadas segundo a IEEE *SPECTRUM* (SPECTRUM, 2020), e vem sempre tendo atualizações constantemente através do passar dos anos e também já está consolidada anos no mercado.

Nos próximos capítulos veremos mais a fundo quais são as tecnologias que serão abordada nesse projeto e falaremos brevemente sobre artesanato.

2 LINGUAGEM DE PROGRAMAÇÃO

Se você nunca ouviu falar sobre programação certamente você certamente não ouviu quais ferramentas são utilizadas para desenvolver os produtos ou se sim sabe muito pouco sobre esse assunto, pois bem umas das ferramentas que são utilizadas para desenvolver uma solução são chamadas de linguagem de programação.

Através dessas linguagens que são produzidos sistema de todos os tipos como por exemplos: sistema de contabilidade, redes sociais, controle remoto,

[1] http://lattes.cnpq.br/2771260225601245 / https://www.linkedin.com/in/herciliodemedeiros/
[2] http://lattes.cnpq.br/2885241414019869

inteligência artificial entres outros. Com essas poderosas linguagens vocês consegue fazer infinitas coisas com apenas linhas de código, e umas dessas linguagens que será utilizada o desenvolver desse sistema que iremos abordar nesse trabalho será a linguagem *Java.*

2.1 Java

Java é uma linguagem de programação de alto nível como varias outra linguagem que existe nesse enorme mundo de TI, então esse foi o motivo dela ser escolhida ela, abaixo segue os motivos principais:

1. Linguagem que pode ser uma linguagem multiplataforma, isso que dizer que ela é capaz de rodar em todos os sistemas operacionais graças ao *Javac* e *Jvm*.
2. Linguagem sempre no topo no mercado e com isso vem sempre tendo novas atualizações.
3. Linguagem desenvolvida com intuito de ser utilizada com paradigma de orientação a objetos, um paradigma muito utilizado mesmo sendo criado a alguns anos atrás.

Como foi mencionado no texto anteriormente, o Java tem grandes benefícios apesar de ter alguns contras, porém para quem estar procurando uma tecnologia atual e está desenvolvendo algo e queira que sua aplicação seja robusta, e que sua manutenção seja menos problemática e uma linguagem de programação bem conceituada no mercado, o Java pode ser algo que você procura.

2.1.1 JAVA VIRTUAL MACHINE E JAVA COMPILER

Agora será falado sobre duas coisas importantes que é atrelada a o *Java*, no caso seu compilador e sua “maquina”.

O *JAVAC* ou *Java compiler* é um compilador da linguagem que tem como função transformar nossos códigos que de alto nível que é legível para um código ilegível para humanos, esse código que é após ser passado pelo Javac é chamado de *bytecode*. (ORACLE, 2021).

Já a JVM ou Java *virtual machine* como o próprio nome enuncia, é um computador virtual que trabalha dentro de um computador real, ela tem a função de

pegar o arquivo binário do programa que foi compilado pelo Javac e o executa. (MANZANO, J.A.N.G.; JR., R.A.D.C, 2014).

2.2 Linguagem de marcação de hyper texto e folhas de estilos em cascatas

Após ser falado anteriormente sobre linguagem de programação *Java* que serve para criar toda a parte inteligente do sistema e isso basicamente fica responsável pelo *back end* da aplicação, Será mostrado sobre a parte que é apresentada ao usuário chamada de *front end* que é usado basicamente HTML, css, *Java script* também chamada de *ecma script*. Em desenvolvimento de sistema as tecnologias são separadas de acordo com suas responsabilidades, normalmente ou ela é mais compatível com o *back end* ou *front end*, porém existem exceções como para o *Java script* que pode ser usada tanto para *back* ou *front*.

Começaremos abordando uma tecnologia que é utilizada para criar o "esqueleto" das páginas web que temos hoje disponíveis na web, é a linguagem de marcação de *hyper* texto ou HTML.

O HTML surgiu em 1990 criado pelo Tim Berners-Lee com em uma sede da *NeXTcube* criada por Steves Jobs (WHATWG, 2021).

O Intuito do HTML era se um projeto especificamente como uma linguagem para descrever documentos cientifico. Porem foi visto que ele poderia ser adaptado para ser utilizado até em aplicações, e hoje em dia ainda é utilizada com tecnologia principal na criação de páginas web (SPEC.WHATWG.ORG, 2021).

O HTML tem uma grande gama de extensibilidade que podem ser usadas de maneira segura e adicionar uma semântica sem comprometer a aplicação (SPEC. WHATWG.ORG, 2021).

Após ser mencionado como é criado o esqueleto de uma página web atualmente, agora será citado uma tecnologia que juntamente com o *HTML* é capaz de deixar as páginas mais atrativas para os olhos dos usuários que acessam os sites ou aplicações web, e o nome dessa tecnologia é folhas de estilos em cascatas ou simplesmente *CSS*.

O *CSS* é uma linguagem que serve para implementar efeitos de apresentação em páginas *web*, como *layou*t, cores, fontes das letras como também deixar as páginas responsivas para todos os tipos de dispositivos como por exemplos:

computadores, celulares, *tablets*, impressoras, relógios inteligentes entres outros (W3.ORG, 2021).

O CSS é uma tecnologia alto independente, isso que dizer que ela não precisa especificamente trabalhar juntamente ao HTML, ela pode ser unificada com qualquer outra linguagem que seja baseada em XML. Com isso fica mais fácil de manter sites por essas tecnologias serem independentes vocês podem substituir se vem ao caso (W3.ORG, 2021).

2.3 Java script

O *Java Script* assim como o *Java* também é uma linguagem de programação. Você pode até achar que elas têm parentesco, pois elas têm uma sintaxe bastante parecida, mas elas são duas linguagens completamente diferentes uma da outra, *Java script* diferente do *Java* é uma linguagem utilizada no lado do cliente e também é implementada com uma tipagem dinâmica diferente do *Java*, porém ela pode ser utilizada no lado do servidor também junto a o *node.js* que é outra tecnologia que não será abordada.

O *Java Script* foi criado pela *Netscape* nos primórdios da internet na web, *"Java Script"* na verdade é uma marca registrada pela *Sun Microsystems* que agora passou ser *Oracle*. também chamada de *"ECMA Script"*. porém esse nome ficou marcado para a padronização e versões de lançamento da linguagem atualmente estamos na versão do *ECMASCRIPT 6* (FLANAGAN, David, 2012).

Como foi demonstrada anteriormente a história do *Java script*, a seguir será mostrado um trecho de código bem básico conforme apresentados nas figuras 1 e 2. pois bem, será comparado dois casos em quer o programador lança um evento que mostrar um texto no console porém em linguagem diferentes, as linguagem em comparação serão as que estão sendo debatidas nesse relatório que será *Java script* e *Java.*

Figura 1 - *Hello world* em Java *Script*.

```
//Em Java Script
console.log("Hello World")
```

Fonte: Próprio Autor (2021).

Conforme visto acima está escrito um código em Java *Script* que mostra o quanto é mais fácil de ser lido por ter uma sintaxe relativamente fácil, e também menos trabalhosa ao ser escrito.

Figura 2 - *Hello Word* em Java.

Fonte: Próprio Autor (2021).

Podemos ver que em *Java* houveram bastantes mudanças comparadas ao Java Script. Dar para se notar que ele tem muito mais linhas de código e uma complexidade maior de sintaxe ao ser escrito que dificulta um pouco o entendimento.

2.4 Frameworks

Frameworks são estruturas que possui diversas funcionalidades que podem ser utilizadas pelos desenvolvedores de software.

O principal objetivo dos *frameworks* é resolver problemas recorrentes com uma abordagem genérica. Com isso, o desenvolvedor não precisa ficar reescrevendo *softwares*, podendo focar seus esforços em resolver os problemas em si. (HOSTGATOR, 2020, ONLINE).

Pelos motivos citados acima o desenvolvedor focar em apenas na resolução do problema, pois ele abstrair bastante código, ele também ajuda no desenvolvimento rápido das aplicações além de deixar a curva de aprendizado menor.

2.4.1 ANGULAR

Angular é um *framework Java script* que trabalhar em um estilo spa “*single pages aplication*” que só é carregada uma página que será utilizada em toda a aplicação, baseado em componentes construído em *typescript* que ajuda o desenvolvedor a criar sua aplicação *front end* rapidamente com qualidade e segurança (ANGULAR, 2021).

Os componentes de uma aplicação angular são os blocos de construção que compõem uma aplicação (ANGULAR, 2021, ONLINE).

Um componente de uma aplicação angular tem as seguintes configurações mostradas na tabela 1:

Tabela 1: Configurações de um componente angular.

Uma arquivo *TypeScript*.
Um arquivo HTML.
Um arquivo CSS.

Fonte: Próprio Autor (2021)

Com todas essas ferramentas e possibilidades o Angular é um facilitador fundamental na construção de uma aplicação.

2.5 Artesanato

Artesanato é a arte de criar obras e através delas tentar passar algum sentimento para quem está a admirando, elas podem ser criadas de todos os jeitos possíveis, sendo através de pinturas, molduras ou simplesmente escrevendo algo. Seja isso criado a partir de uma simples ideia que você teve em algum momento em sua imaginação ou sendo criada inspirada em algo que você viu ou sentiu. (CONCEITO.DE, 2020)

Após passar uma de milhares de descrição do que é o artesanato isso ainda é muito atual, ainda é realidade de muitas pessoas na sociedade em que nós estamos inseridos, muitas delas vivem do artesanato para poderem dai tirar a sua renda e dessa forma conseguir viverem suas vidas. E essa cultura muitas vezes foram passadas através de seus próprios antepassados que através dos tempos foram passando essa “herança” de forma que ainda hoje ainda existam pessoas que consegue viver disso.

Muitas pessoas vivem de criar peças através da costura como por exemplos: blusas, *shorts*, e agora mascaras que foi incluído após início essa crise da pandemia que foi aparecer nesses últimos anos, e como o mundo está mudando ao passar dos

anos a tecnologia está sendo enramada em todos os seguimentos seria legal unir esses ramos “tecnologia e o artesanato” buscando facilitar o comercio de produtos feitos por artesãos.

3 Levantamento de requisitos

Nesse capítulo será apresentado o levantamento de requisitos, que será através deles, que abordará os requisitos que o sistema deverá apresentar.

Segundo ROGER, P. e BRUCE, M(2016, p.142) levantamento de requisito é:

> O Levantamento de requisitos(também chamado elicitação de requisitos) combina elementos de solução de problemas, elaboração, regociação e especificação. Para estimular uma abordagem colaborativa e orientada a equipes em relação ao levantamento de requisitos, os envolvidos trabalham juntos para identificar o problema, propor elementos da solução, negociar diferentes abordagens e especificar um conjunto preliminar de requisitos da solução.

E através do levantamento de requisitos será elaborado uma lista de requisitos funcionais e não funcionais para definição das funcionalidades do sistema.

3.1 Requisitos funcionais

Segundo CUNHA (2020) requisitos funcionais são parte da etapa de elicitação, os requisitos são todos os problemas e necessidades que devem ser atendidos e resolvidos pelo *software* por meio de funções ou serviços.

Desta maneira foram elaborados os requisitos funcionais do sistema mostrado na tabela 2:

Tabela 2 - Requisitos funcionais do site de vendas.

Identificador	Nome	Descrição	Prioridade
RF01	Exibir valor total dos Pedidos	O sistema deverá mostrar o total de cada pedido	Essencial
RF02	Cadastrar Pedido	O sistema poderá realizar pedidos	Essencial

RF03	Cadastro Produtos	O sistema terá como cadastrar produtos	Essencial
RF04	Visualizar Pedidos	Administrador poderá visualizar os pedidos.	Importante
RF05	Exibir Produtos	O sistema terá que listar os produtos.	Essencial

Fonte: Próprio Autor (2021).

Considerando a tabela 2 as necessidades iniciais que o sistema precisa está definida, e já está pronta para que seja iniciada o desenvolvimento do sistema.

3.2 Requisitos não funcionais

Segundo CUNHA (2020) os requisitos não funcionais são todos aqueles relacionados à forma como o *software* tornará realidade os que estão sendo planejado. Ou seja, enquanto os requisitos funcionais estão focados no que será feito os nãos funcionais descrevem como serão feitos.

Por consequência foram elaborados os requisitos não funcionais e poderemos observar na tabela 3:

Tabela 3 - Requisitos não funcionais.

Identificador	Nome	Descrição	Prioridade
RNF01	*Web*	Deverá ser um sistema *web*	Essencial
RNF02	*Internet*	poderá ser utilizado sem internet.	Essencial
RNF03	Consumo de memória	Deverá ser performático	Importante
RNF04	SGBD	Banco de dados que será utilizado será o *MYSQL*.	Importante

Fonte: Próprio Autor (2021)

Até esse presente momento inicial da proposta da aplicação que foram definidos na tabela 3 apenas esses quatros requisitos não funcionais podendo ter mudanças no futuro dessa proposta que será desenvolvida.

4 Desenvolvimento

Como vem sendo mencionado anteriormente o intuito do sistema chamado de *Sewing Art* em geral é poder cadastrar os pedidos de produtos que são feitos pelos clientes com intuito de beneficiar o controle dos pedidos.

Figura 4 - Página inicial do *sewing art*.

Fonte: Próprio Autor (2021).

Na figura 4 desse sistema mostra a página inicial bem simples e fácil de ser utilizada com um menu com duas opções que redireciona para as páginas de produtos ou pedidos.

Figura 5 - HTML da página inicial do *sewing art*.

```
<div id="div-principal">
    <div class="row">
        <div class="col">
            <h3 id="titleBemVindo">Seja bem Vindo</h3>
        </div>
    </div>
    <div class="row">
        <div class="col">
            <h5></h5>
            <p id="paragrafoBemVindo">Aqui você poderá administrar seu negocio com eficiência e agilidade.</p>
        </div>
    </div>
    <div class="row">
        <div class="col">
            <img src="../../assets/imagens/welcome-v2.jpg" height="100%" width="100%" alt="">
        </div>
    </div>
</div>
```

Fonte: Próprio Autor (2021).

Na figura 5 está a exibir o HTML da página inicial com algumas *tags HTML*.

Figura 6 - Listagem dos produtos do sewing art.

Fonte: Próprio Autor (2021).

Na figura 6 mostra a tela de listagem dos produtos cadastros no sistema, exibindo uma tabela com os produtos cadastrados do sistema mostrando alguns campos importantes de cada produto e com uma opção de atualizar ou excluir esse produto, mas a direita tem um botão com uma figura de um mais que ao ser clicado ele abre a tela de cadastro de produto.

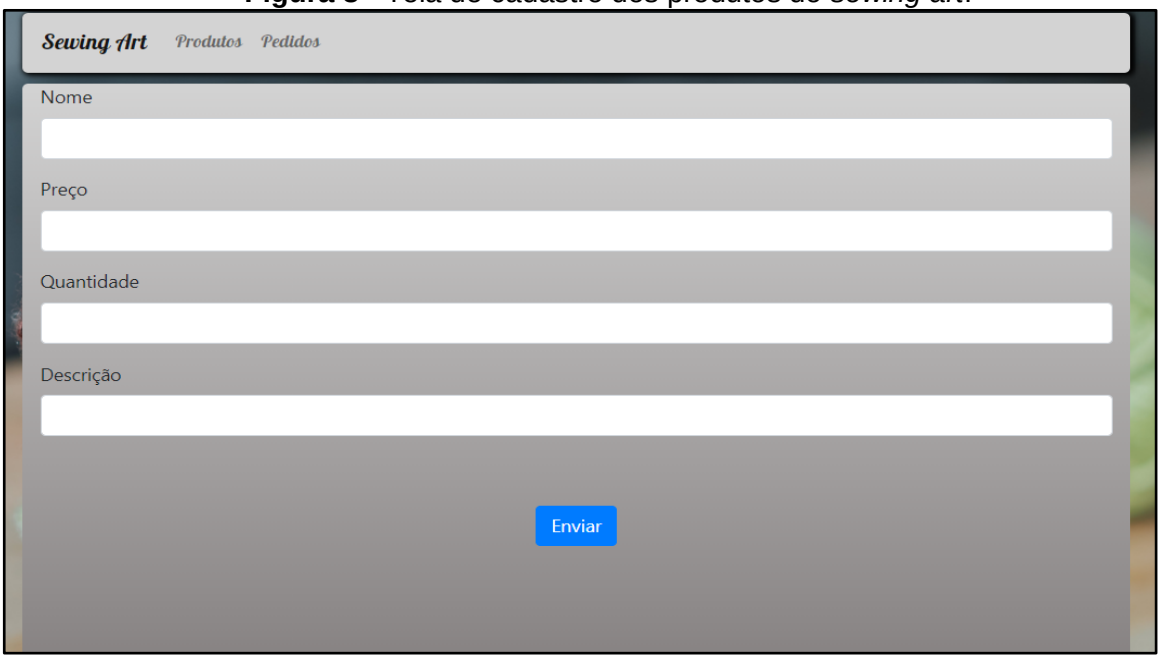
Figura 7 - Arquivo *type script* da tela de listagem de produtos.

```
listar(){
  this.listService.listar().subscribe(
    product =>{
      this.productsArray = product
    }
  );
}

updateProduct(product){
   this.router.navigateByUrl('/updateProduct')
}

removeProduct(id){
  this.productId = id;
}

removerProductConfirm(){
  this.listService.remove(this.productId).subscribe(res => {
    this.listar();
  });
}
```

Fonte: Próprio Autor (2021).

Na figura 7 está exibindo o código do arquivo *type script* da tela de listagem dos produtos com algumas funções que fazem o listar, excluir e atualizar um produto do sistema.

Figura 8 - Tela de cadastro dos produtos do *sewing art*.



Fonte: Próprio Autor (2021).

Na figura 8 mostra a tela de cadastro dos produtos, que vai ter como função cadastrar ou atualizar os produtos que o usuário do sistema submeter. possui quatros campos para serem preenchidos e um botão principal para o envio dos dados do formulário, que após ser enviado vai se redirecionado novamente para a tela de listagem de produtos.

Figura 9 - Arquivo *type script* da tela cadastro de produtos.

```
productCreate = { id: '', nome:  '', preco: '', quantidade: '', descricao: ''
}

constructor( private listProductService : ListProductService, private routerActivate: ActivatedRoute,
  private router: Router) { }
id: any;

ngOnInit(): void {
  this.id = this.routerActivate.snapshot.paramMap.get('id');
  if(this.id !== ''){
    this.listProductService.getById(this.id).subscribe(res => {
      if(res !== null){
        this.productCreate.id = res.id;
        this.productCreate.nome = res.nome;
        this.productCreate.preco = res.preco;
        this.productCreate.quantidade = res.quantidade;
        this.productCreate.descricao = res.descricao;
      }
    })
  }
}

create(){

  this.listProductService.create(this.productCreate).subscribe( res => {
   this.sucesso = true;
    setTimeout(()=> {
      this.router.navigate(['product'])
    }, 2300)
  })
}
```

Fonte: Próprio Autor (2021).

Na figura 9 temos o arquivo *type script* da tela de cadastro de produtos exibindo umas funções que tem como dever submeter o envio dos dados para ser armazenado na base de dados para que o cadastro do produto seja concluído.

Figura 10 - Listagem dos pedidos do *sewing art*.

Fonte: Próprio Autor (2021).

Na figura 10 mostra à tela de listagem dos pedidos cadastrado no sistema, assim como na tela de listagem de produtos, as telas de pedidos também exibiram uma tabela com os pedidos cadastrados no sistema e mostra alguns dados importantes para o usuário do sistema.

Figura 11 - Arquivo type script da tela de listagem dos pedidos.

```
listar(){
  this.pedidosService.listar().subscribe(
    product => this.pedidosArray = product
  );
}
```

Fonte: Próprio Autor (2021)**.**

Na figura 11 está a mostrar a função responsável pela busca dos dados e preenchimento da lista da tabela de pedidos.

Figura 12 - Cadastro dos pedidos do *sewing art*.

Fonte: Próprio Autor (2021).

Na Figura 12 está a demonstrar a tela de cadastro dos pedidos dos clientes, bem intuitiva com dois campos de nome de cliente e um *checkbox* de opções de produtos cadastrado no sistema.

Figura 13 - Classe *Demand Controller* responsável pelos gerenciar de pedidos.

```
public class DemandController {

    @Autowired
    private DemandService demandService;

    @PostMapping
    @ResponseStatus(code = HttpStatus.CREATED)
    public ResponseEntity<?> save(@RequestBody Demand demand){
        return ResponseEntity.ok(demandService.save(demand));
    }

    @PutMapping
    @ResponseStatus(code = HttpStatus.OK)
    public ResponseEntity<?> update(@RequestBody Demand demand)
        return ResponseEntity.ok(demandService.update(demand));
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(code = HttpStatus.OK)
    public void remove(@PathVariable Long id) {
        demandService.remove(id);
    }

    @GetMapping("/{id}")
    @ResponseStatus(code = HttpStatus.OK)
    public ResponseEntity<?> getById(@PathVariable Long id){
        return ResponseEntity.ok(demandService.getById(id));
    }

    @GetMapping
    @ResponseStatus(code = HttpStatus.OK)
    public ResponseEntity<List<Demand>> getAll(){
        return ResponseEntity.ok(demandService.getAll());
    }
```

Fonte: Próprio Autor (2021).

Na figura 13 mostra a classe controladora no *back end* responsável por tratar as requisições dos pedidos.

Figura 14 - Tela de detalhes do pedido.

Fonte: Próprio Autor (2021).

Na figura 14 mostra os detalhes de um pedido com alguns campos como, por exemplo, total do pedido e uma listagem dos produtos pertencente a esse pedido.

5 Considerações finais

Esse sistema inicial foi desenvolvido com a intenção de resolver um problema de administração de pedidos que são feitos ao seu vendedor que anteriormente eram armazenados de forma escrita em cadernetas. Com a tecnologia em alta ajudando vários segmentos foi desenvolvido essa proposta inicial fazendo uso da programação utilizando *Java* com *Spring Boot e Angular*.

O sistema está em fase de desenvolvimento com algumas funcionalidades prontas. para essa primeira versão foram utilizadas duas semanas para conclusão dos requisitos iniciais e já pode ser utilizada. Atualmente está só em formato *web* porém futuramente esse sistema será desenvolvido para plataforma *Android* visando ainda mais praticidade de ser utilizado em qualquer lugar por um *smartphone.*

6.REFERENCIAS BIBLIOGRÁFICAS

ANGULAR. **What is Angular?** 2021. Disponível em: https://angular.io/guide/what-is-angular. Acesso em: 25 abr. 2021.

ANGULAR. **Execute o aplicativo**. 2021. Disponível em: https://angular.io/guide/setup-local. Acesso em: 25 abr. 2021.

CUNHA, Fernando. **Requisitos funcionais e não funcionais: o que são?** 2020. Disponível em: https://mestresdaweb.com.br/fabrica-de-software/requisitos-funcionais-e-nao-funcionais-o-que-sao/. Acesso em: 11 maio 2021.

CONCEITO.DE. **Conceito de artesanato**. 2020. Disponível em: https://conceito.de/artesanato. Acesso em: 31 mar. 2021.

David, F. **JavaScript**. Grupo A, 2014. 9788565837484. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788565837484/. Acesso em: 16 May 2021

WHATWG. **HTML Living Standard**. 2021. Disponível em: https://html.spec.whatwg.org/multipage/introduction.html#is-this-html5?. Acesso em: 23 maio 2021.

HOSTGATOR. **HostGator**. 2020. Disponível em: https://www.hostgator.com.br/blog/frameworks-na-programacao/. Acesso em: 25 abr. 2021.

MANZANO, J.A.N.G.; JR., R.A.D.C. **Programação de Computadores com Java**. Editora Saraiva, 2014. 9788536519494. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788536519494/. Acesso em: 16 May 2021

ORACLE. **Javac - Java programming language compiler**. 2021. Disponível em: https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javac.html. Acesso em: 05 maio 2021.

ROGER, P.; BRUCE, M. **Engenharia de Software**. [Digite o Local da Editora]: Grupo A, 2016. 9788580555349. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788580555349/. Acesso em: 10 May 2021

SPECTRUM, Ieee. **Interactive: The Top Programming Languages**. 2020. Disponível em: https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020. Acesso em: 28 mar. 2021.

WHATWG, W3C &. **HTML**. 2021. Disponível em: https://pt.wikipedia.org/wiki/HTML. Acesso em: 31 mar. 2021.
W3.ORG. **O que é CSS?** 2021?. Disponível em: https://www.w3.org/standards/webdesign/htmlcss.html. Acesso em: 31 mar. 2021.

BOT4WHAT: CHATBOT E SUA IMPORTÂNCIA NO ATENDIMENTO AO CLIENTE

SILVA, Williams José de Aguiar[1]
SOUSA, Hercilio de Medeiros[2]
GOMES FILHO, Carlos Barbosa[3]

INTRODUÇÃO

A atenção ao usuário é um dos componentes que qualquer empresa deve prestar atenção, toda empresa depende da satisfação do cliente, um meio de garantir esta satisfação é através de um bom serviço de atendimento ao cliente, seja no momento da compra de produtos ou serviços, ou até mesmo no pós venda. Por esta razão foram criados metodologias, ferramentas e sistemas para assegurar a informação e ter aspectos de melhora, similar ao sistema de Elogios; Sugestões; Dúvidas ou Reclamações. (GARCÍA, 2018).

Dito isto, atualmente as pessoas estão se comunicando mais ativamente através de tecnologias de mensagens instantâneas, e uma das mais conhecidas hoje é o Whatsapp, onde dos mais idosos até os mais jovens se conectam e trocam mensagens. Com isso em vista, não somente as empresas, como também autônomos aderiram ao Whatsapp como forma de conversar com mais proximidade com seus clientes, trocando informações de todos os tipos, como também prestando ajuda e suporte. Porém de uma forma onde precisaria de uma pessoa para responder prontamente aos clientes. Foi então que o grupo do Facebook criou o Whatsapp Business em janeiro de 2018 (WHATSAPP Business App, 2021). Que permitia criar uma loja virtual dentro do aplicativo de mensagens e responder de forma automática com mensagens previamente cadastradas em textos simples.

O Bot4What não vem pra substituir o WhatsApp Business e sim criar uma extensão de personalização de respostas e automação de tarefas simples como: Criar uma lista dinâmica no app podendo exportá-la em PDF ou Word; Consultar cotação de moedas estrangeiras; Integração com sistemas para envio de mensagens de forma gratuita incluindo imagens, vídeos, áudios e adesivos personalizados; Controle de grupos com personalização de mensagens de boas-

[1] https://www.linkedin.com/in/williams-jsa/
[2] http://lattes.cnpq.br/2771260225601245 / https://www.linkedin.com/in/herciliodemedeiros/
[3] http://lattes.cnpq.br/2017611396853275 / https://www.linkedin.com/in/carlos-barbosa-gomes-filho-b0463542/

vindas; Converter imagens e vídeos em *Stickers* (adesivos) e Gifs animados; entre uma variedade de conexões com outras APIs para consultas.

CHATBOT

Antes de definir o significado de chatbot, é necessário entender que esses assistentes virtuais inteligentes são um resultado de um avanço em várias áreas da ciência no decorrer dos anos, como a inteligência artificial, processamento de linguagem natural, banco de dados e rede de comunicação de dados (CRUZ; ALENCAR; SCHMITZ, 2018).

Para desenvolver um chatbot faz-se necessário a capacidade de entender questões dos seres humanos, estabelecer a heurística, encontrar respostas e consultar banco de dados para compor a resposta de cada pergunta. O estudo da língua e da comunicação também tem se destacado para a evolução dos chatbots, pois a língua é dinâmica e evolui a cada momento, e é importante que as tecnologias estejam atualizadas com o vocabulário atual (CRUZ; ALENCAR; SCHIMTZ, 2018).

TECNOLOGIAS

Nesta seção são apresentados os requisitos principais tecnológicos necessários para o desenvolvimento do Bot4What.

WHATSAPP

O WhatsApp é gratuito e oferece um serviço de mensagens e chamadas simples, seguro e confiável para celulares em todo o mundo. O WhatsApp surgiu como uma alternativa ao sistema de SMS e agora possibilita o envio e recebimento de diversos arquivos de mídia: textos, fotos, vídeos, documentos e localização, além de chamadas de voz. Alguns de seus momentos mais importantes são compartilhados no WhatsApp. Por essa razão, implementamos a criptografia de ponta a ponta no nosso aplicativo. Por trás de cada decisão está o nosso desejo de possibilitar que as pessoas se comuniquem sem barreiras, em qualquer lugar do mundo. WhatsApp (2021)

TYPESCRIPT

Quando se desenvolve para web é necessário utilizar Javascript, HTML e CSS, focando na linguagem em questão, o Javascript é entendido pelos browsers nativamente, porém, após várias atualizações o Javascript passou a ser usado não somente no cliente (*Frontend*), mas também no servidor (*Backend*) e até mesmo em desenvolvimento mobile.

Mas a fraca tipagem do Javascript se tornou um problema para manutenções de aplicativos, e crescimento de sistemas, nesse contexto surge o TypeScript, que é um superset do Javascript, com ele é possível aplicar a OOP (*Object Oriented Programming*) trazendo uma sintaxe com mais clareza no desenvolvimento. Ao utilizar TypeScript, temos a possibilidade de aplicar tipagem estática juntamente com interfaces em um sistema construído unicamente em Javascript.
Algumas de suas vantagens, estão entre:

ENCAPSULAMENTO

Entende-se como o conceito de encapsulamento a forma de estruturação do código, para que determinados blocos de código possuam acesso à pontos específicos num ambiente externo, criando visibilidade e acessibilidade à elementos internos de uma classe. Assim, podemos definir quais atributos de uma classe estará visível à um usuário externo ou expostos em uma interface de sistema pública.

ABSTRAÇÃO

Considera-se abstração características que podemos trazer do mundo real para o código, muito utilizado em OOP (Programação Orientada à Objetos). Tais características são agrupadas em classes, que por sua vez, representam partes de um elemento e seus atributos para solução de determinados problemas.

Com todos estes conceitos, temos uma das maiores vantagens em usar o Typescript ao invés do Javascript, que é descobrir erros durante a implementação de código, mais conhecido como *IntelliSense*, a inteligência da IDE (Ambiente Integrado de desenvolvimento), tais quais apenas descobriríamos no Javascript somente durante a execução do programa. Trazendo a tipagem estática tornamos as aplicações mais seguras e facilitamos sua manutenção, melhorando a produtividade.

EXPRESS

Express é a estrutura da web mais conhecida em execução no ambiente de tempo de execução Node.js. (Contribuidores MDN, 2020).

Desde que o Express foi lançado, tem recebido feedbacks positivos de muitos especialistas, e se tornou uma das escolhas mais populares para desenvolvimento de servidores web. O Express fornece camadas para:

- Criar e manipular solicitações https.
- Escrever modelos de servidor.
- Configurar rotas do servidor web e a porta para ouvir as solicitações de entrada.
- Configurar middlewares para interceptar as solicitações recebidas.

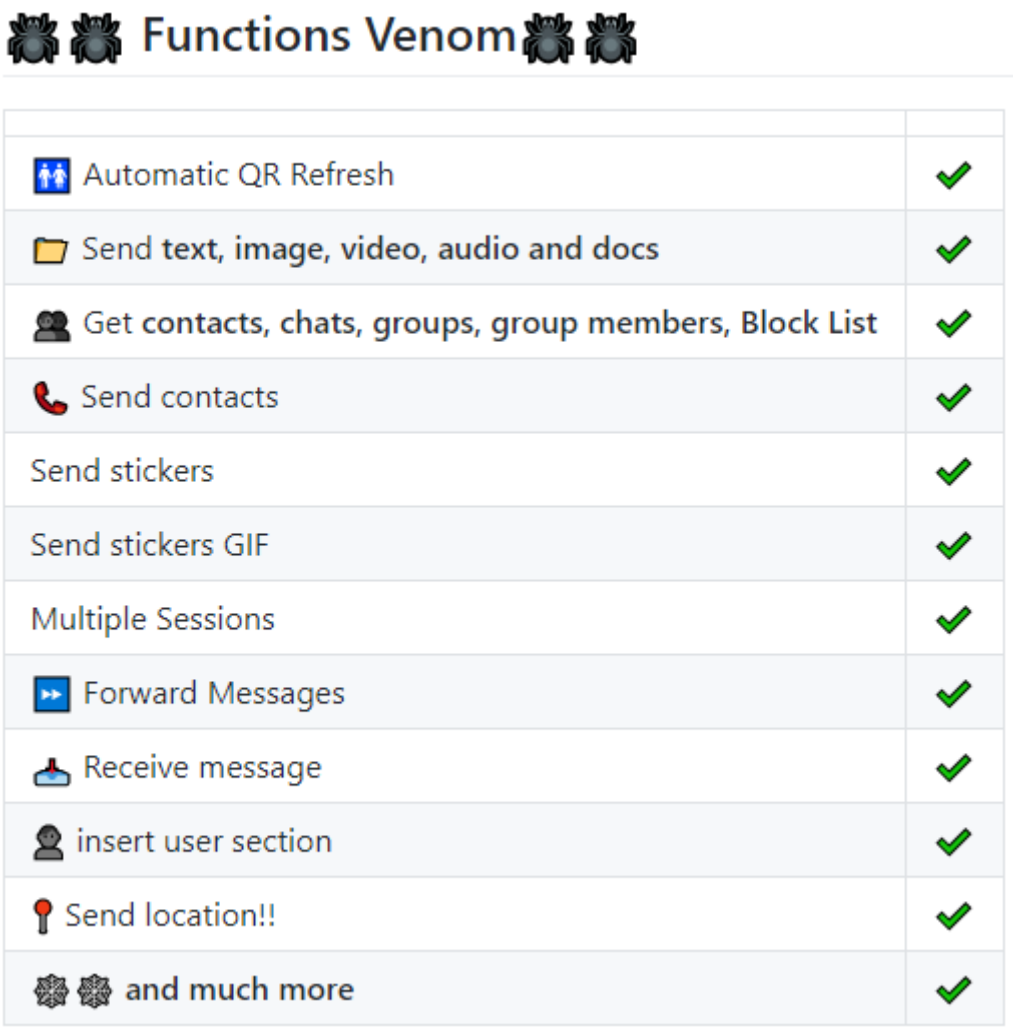
O Express dá aos desenvolvedores a liberdade de escolher a arquitetura do aplicativo (Contribuidores MDN, 2020). Não existindo um jeito certo ou errado de estruturação de arquivos Express, desde que atinja os objetivos técnicos e de negócio.

VENOM

Venom é um sistema de alto desempenho desenvolvido com JavaScript para criar um bot para WhatsApp, suporte para criar qualquer interação, como atendimento ao cliente, envio de mídia, reconhecimento de sentenças baseado em inteligência artificial e todos os tipos de arquitetura de design para WhatsApp. (ORKESTRAL, 2021)

Dos principais recursos oferecidos pelo Venom estão listados:

Figura 1 - Funções do Venom



Fonte: *Orkestral*, 2021.

PUPPETEER

Puppeteer é uma biblioteca Node que fornece uma *API* de alto nível para controlar o Chrome ou Chromium headless (*Headless* é um software capaz de funcionar em dispositivos sem a necessidade de uma interface gráfica) através do protocolo *DevTools*. Ele também pode ser configurado para usar o Chrome completo (non-headless) ou o Chromium. (DEVELOPERS, Google... 2021). O Puppeteer pode fazer a maioria das coisas se pode ser feito manualmente via interface gráfica, como, por exemplo:

- Gerar capturas de tela e PDFs de páginas.

- Criar um SPA (aplicativo de página única) e gerar conteúdo pré-renderizado, ou seja, "SSR" (renderização do lado do servidor).
- Automatizar o envio de formulários, testes de UI (Interface de Usuário), entrada de teclado, etc.
- Criar um ambiente de teste automatizado e atualizado. Executar testes diretamente na versão mais recente do Chrome usando o JavaScript e os recursos do navegador mais recentes.
- Analisar sites para ajudar a diagnosticar problemas de desempenho.
- Testar as extensões do Chrome.

NODE

Desenvolvido por Ryan Dahl, o Node utiliza o mecanismo Javascript V8 disponível nos browsers e mais poderoso da época em 2008, lançado por Lars Bak, Engenheiro de software da Google (PASQUALI, Sandro; FAABORG, Kevin, 2017). E hoje é um dos mais populares ambientes de execução de código Javascript fora dos navegadores. Pela primeira vez na história os desenvolvedores podem criar aplicativos *Backend* utilizando a linguagem Javascript com recursos de alto nível e desempenho (PASQUALI, Sandro; FAABORG, Kevin, 2017). Comparado a outros *frameworks Backend* existentes, o Node.js fornece uma infraestrutura exclusiva devido às características de I/O (Entrada e Saída) orientadas a eventos, *thread* único e sem bloqueio.

SQLITE

SQLite é uma biblioteca em processo que implementa um mecanismo de banco de dados SQL transacional independente, sem servidor e com configuração zero. O código para SQLite é de domínio público e, portanto, é gratuito para uso para qualquer finalidade, comercial ou privada... SQLite é um mecanismo de banco de dados SQL embutido. Ao contrário da maioria dos outros bancos de dados SQL, o SQLite não tem um processo de servidor separado. O SQLite lê e grava diretamente em arquivos de disco comuns. Um banco de dados SQL completo com várias tabelas, índices, gatilhos e visualizações está contido em um único arquivo de disco. O formato do arquivo de banco de dados é multiplataforma - você pode copiar

livremente um banco de dados entre sistemas de 32 e 64 bits ou entre as arquiteturas *big-endian* e *little-endian*. SQLite (2021)

TYPEORM

TypeORM é um ORM que pode ser executado nas plataformas NodeJS, Browser, Cordova, PhoneGap, Ionic, React Native, NativeScript, Expo e Electron e pode ser usado com TypeScript e JavaScript (ES5, ES6, ES7, ES8). Seu objetivo é sempre oferecer suporte aos recursos JavaScript mais recentes e fornecer recursos adicionais que o ajudem a desenvolver qualquer tipo de aplicativo que use bancos de dados - desde pequenos aplicativos com algumas tabelas até aplicativos corporativos de grande escala com vários bancos de dados. Typeorm (2021)

METODOLOGIA

O Sistema vai atender as principais necessidades de empresas (URA) e também pessoas, como Digital Influencers (utilizando do método *MentionAll* para notificar todos os participantes do grupo para um possível lançamento de curso ou início de live) e RH (Ao criar questionários dinâmicos em formato de mensagem, onde todos os participantes podem simplesmente responder com o nome do questionário ou id “identificador único” seguido de sua escolha ex: !vote (id ou nome) (índice do item) = !vote jogos-mais-curtidos 2; e lista de nomes para pessoas que desejam participar de algum evento da empresa.)

Dentre as funcionalidades do sistema, são destacados:

NOME	DESCRIÇÃO	RESTRIÇÃO	Prioridade
MentionAll	Através do comando @everyone em um grupo, todos os membros do grupo serão marcados para notificação	Apenas administradores de grupo	Alta
URA	A capacidade de personalizar perguntas e respostas automáticas para atendimento do cliente, seja em texto ou áudio	-	Alta

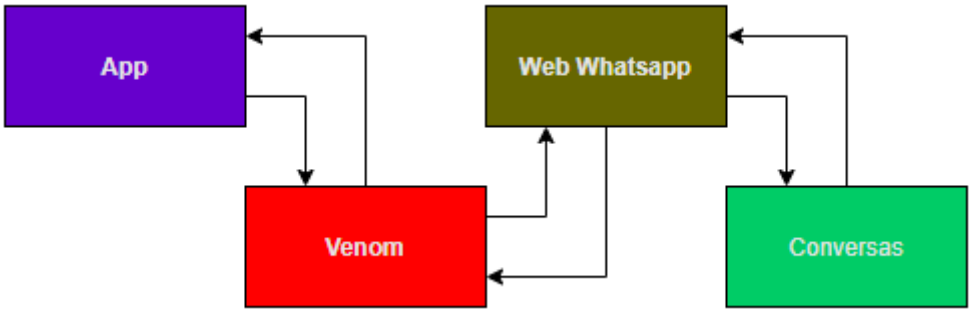
	gravado		
Welcome	Boas-vindas à novos integrantes de um grupo	-	Baixa
SendMessage	Através de exposição de um recurso “https://ip-da-maquina:porta” utilizando um corpo JSON “Javascript Object Notation” numa requisição do tipo POST, é possível enviar uma mensagem para o número do cliente a partir de um sistema terceiro	-	Normal
CreateQuiz	Este comando cria um novo questionário com uma descrição e opções de resposta	Apenas administradores de grupo	Alta
AddAnswer	Cria nova resposta para um questionário	Apenas administradores de grupo	Alta
Vote	Através deste é possível escolher uma resposta de um questionário criado e não fechado	Todos os participantes de grupo	Alta
StopQuiz	Define o status de um questionário como fechado e o torna impossível de ser respondido	Apenas administradores de grupo	Alta
ListQuiz	Lista todos os questionários existentes do grupo	Todos os participantes do grupo	Alta
RemoveAnswer	Remove uma resposta já adicionada de um questionário	Apenas administradores do grupo	Alta
CreateList	Cria nova lista dinâmica	Todos os participantes de	Normal

		grupo	
AddToList	Adiciona um novo item à lista	Todos os participantes de grupo	Normal
StopList	Define o status da lista como fechado e a torna impossível de editar	Apenas o criador da lista	Normal
RemoveList	Remove um item da lista	Apenas o criador da lista	Normal
Show	Mostra os detalhes de uma lista ou questionário	Todos os participantes de grupo	Alta

SISTEMA E FERRAMENTAS

Esta seção dedica-se a mostrar resultados da pesquisa e discussão com alguns trechos de código e ferramentas utilizadas para desenvolvimento do sistema.

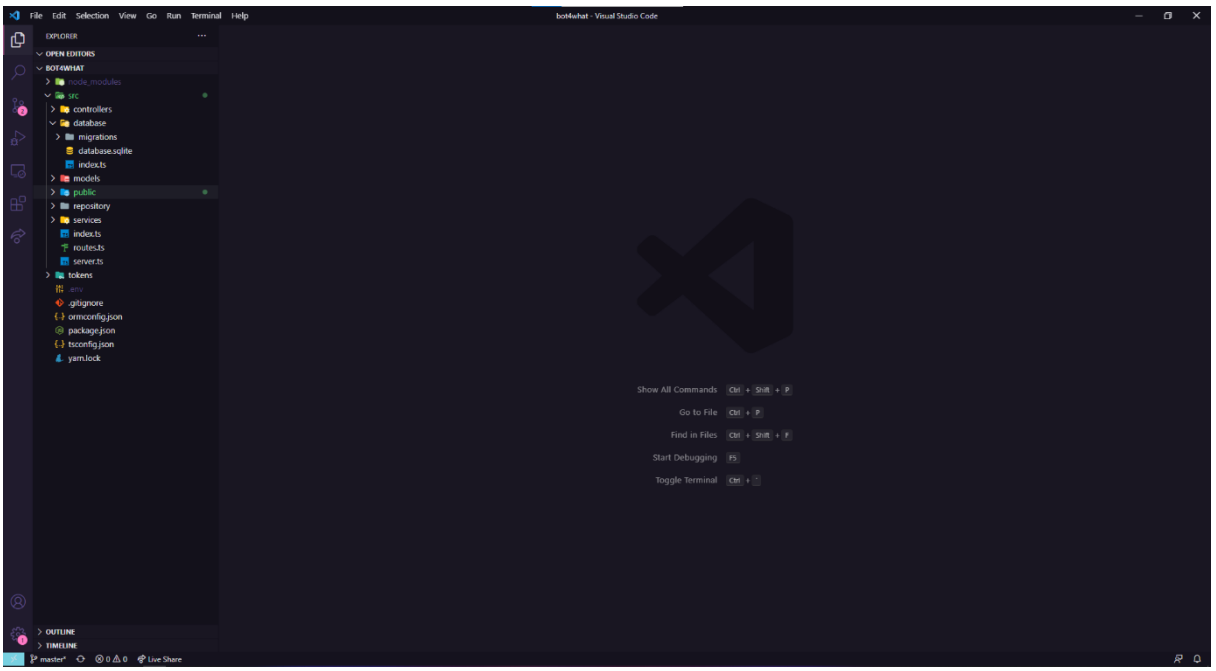
Figura 2 - Diagrama de fluxo



Fonte: Autor, 2021.

Na figura 2 é brevemente apresentado como funciona o fluxo entre as mensagens e a aplicação.

Figura 3 – Estrutura do projeto



Fonte: Autor, 2021.

O VSCode foi escolhido como o ambiente de desenvolvimento integrado devido sua ampla utilização pela comunidade de desenvolvedores Javascript e Typescript, tornando mais fácil resolução de problemas conhecidos.

Figura 4 – index.ts

```
const venom = require('venom-bot');
import { Message, Whatsapp } from 'venom-bot';
import MessagesController from './controllers/MessagesController';
import { startServer } from './server';
import 'reflect-metadata';
import './database';
import createConnection from './database/index';

// conecta ao SQLite
createConnection()
// Inicia o servidor
startServer();
// configura e inicia o venom
venom
  .create(
    'venombot',
    (base64Qr: any) => {
      var matches = base64Qr.match(/^data:([A-Za-z-+\/]+);base64,(.+)$/)
      var response: any = {};

      if (matches.length !== 3) {
        return new Error('Invalid input string');
      }

      response.type = matches[1];
      //@ts-ignore
      response.data = new Buffer.from(matches[2], 'base64');
      var imageBuffer = response;

      require('fs').writeFile(
        './src/public/qrcode.png',
        imageBuffer['data'],
        'binary',
        function (err: any) {
          if (err != null) {
            console.log(err);
          }
        }
      );
    },
    undefined,
    { logQR: true, devtools: true }
  )
  .then((client: Whatsapp) => {
    start(client)
  })
  .catch((erro: JSON) => {
    console.log('There was an error in the bot: ', erro);
  });

async function start(client: Whatsapp) {
  console.log("started")
  client.onAddedToGroup((chat) => {
    console.log(chat)
  })

  client.onMessage((message: Message) => {
    MessagesController.checkMessage(client, message);
  });
}
```

Fonte: Autor, 2021.

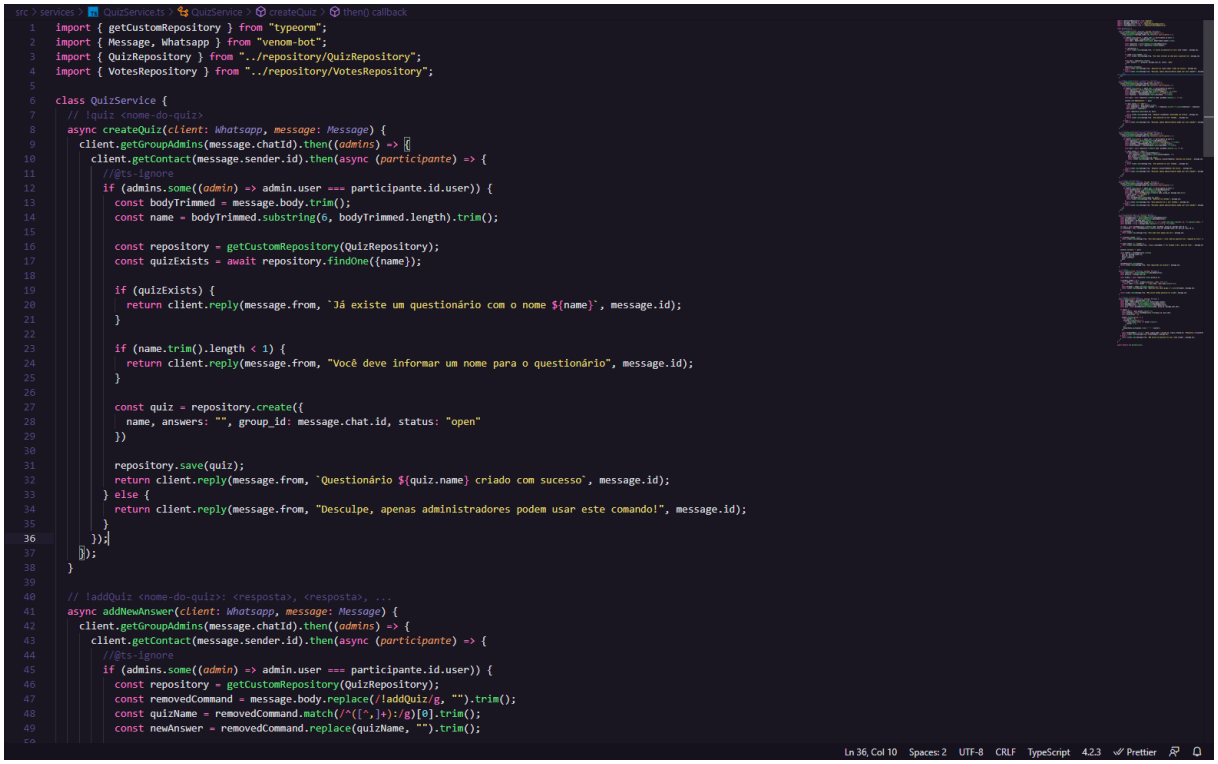
Nesta imagem é mostrado no arquivo principal a configuração inicial do projeto como como conexão com banco de dados, iniciação do servidor, configuração e iniciação da API Venom.

Figura 5 – MessageController.ts

Fonte: Autor, 2021.

Classe controladora de mensagens que passam pelo bot, aqui as mensagens são verificadas e redirecionadas para seus respectivos serviços.

Figura 6 – QuizService.ts



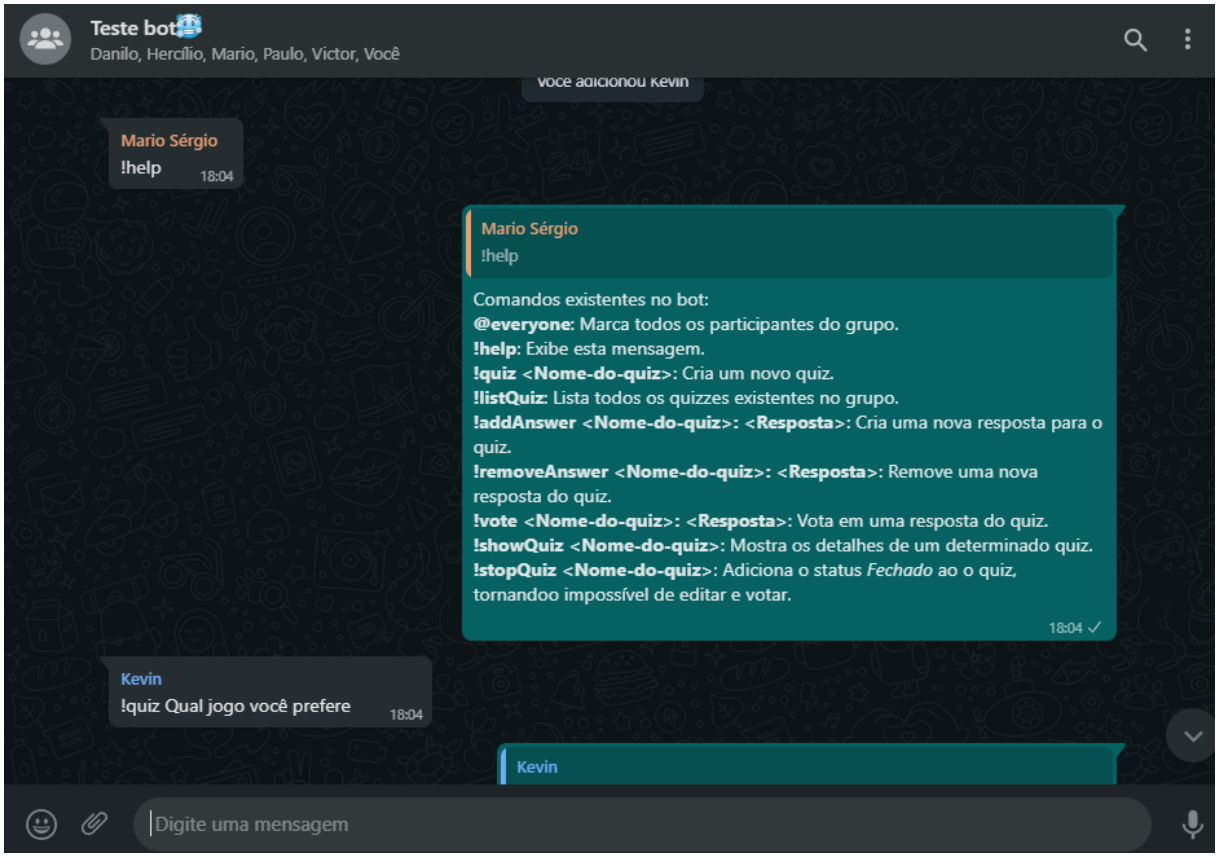
```
src > services > QuizService.ts > QuizService > createQuiz > then() callback
import { getCustomRepository } from "typeorm";
import { Message, Whatsapp } from "venom-bot";
import { QuizRepository } from "../repository/QuizRepository";
import { VotesRepository } from "../repository/VotesRepository";

class QuizService {
  // !quiz <nome-do-quiz>
  async createQuiz(client: Whatsapp, message: Message) {
    client.getGroupAdmins(message.chatId).then((admins) => {
      client.getContact(message.sender.id).then(async (participante) => {
        //@ts-ignore
        if (admins.some((admin) => admin.user === participante.id.user)) {
          const bodyTrimmed = message.body.trim();
          const name = bodyTrimmed.substring(6, bodyTrimmed.length).trim();

          const repository = getCustomRepository(QuizRepository);
          const quizExists = await repository.findOne({name});

          if (quizExists) {
            return client.reply(message.from, `Já existe um questionário com o nome ${name}`, message.id);
          }

          if (name.trim().length < 1) {
            return client.reply(message.from, "Você deve informar um nome para o questionário", message.id);
          }

          const quiz = repository.create({
            name, answers: "", group_id: message.chat.id, status: "open"
          })

          repository.save(quiz);
          return client.reply(message.from, `Questionário ${quiz.name} criado com sucesso`, message.id);
        } else {
          return client.reply(message.from, "Desculpe, apenas administradores podem usar este comando!", message.id);
        }
      });
    });
  }

  // !addQuiz <nome-do-quiz>: <resposta>, <resposta>, ...
  async addNewAnswer(client: Whatsapp, message: Message) {
    client.getGroupAdmins(message.chatId).then((admins) => {
      client.getContact(message.sender.id).then(async (participante) => {
        //@ts-ignore
        if (admins.some((admin) => admin.user === participante.id.user)) {
          const repository = getCustomRepository(QuizRepository);
          const removedCommand = message.body.replace(/!addQuiz/g, "").trim();
          const quizName = removedCommand.match(/^([^,]+):/g)[0].trim();
          const newAnswer = removedCommand.replace(quizName, "").trim();
```

Ln 36, Col 10 Spaces: 2 UTF-8 CRLF TypeScript 4.2.3 Prettier

Fonte: Autor, 2021.

Classe QuizService onde as mensagens são tratadas, processadas e respondidas ao cliente, na figura é mostrado o método para criar um novo questionário a partir do comando !quiz seguido pelo nome do novo questionário.

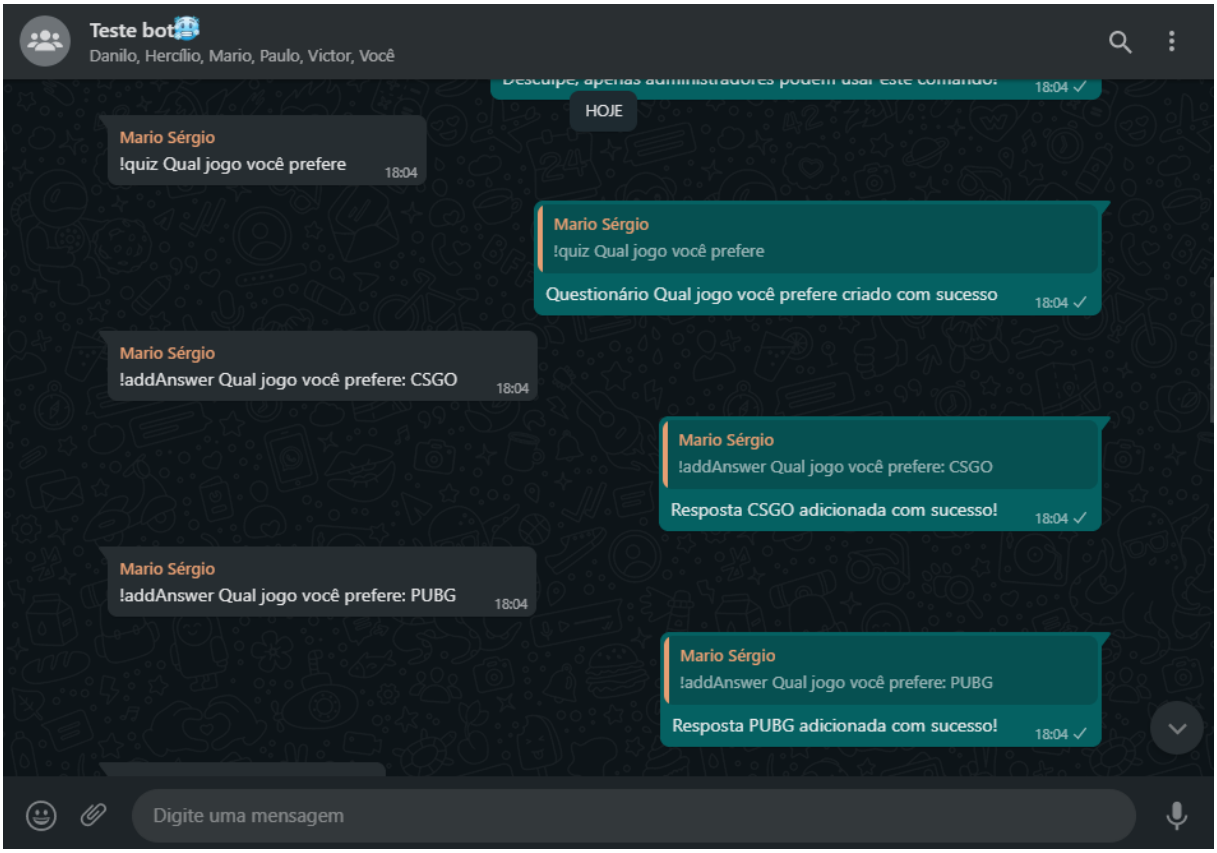
Figura 7 – Executando o comando !help



Fonte: Autor, 2021.

Na figura 7 é mostrado o funcionamento do comando !help, o qual mostra todos os comandos que existem atualmente no sistema.

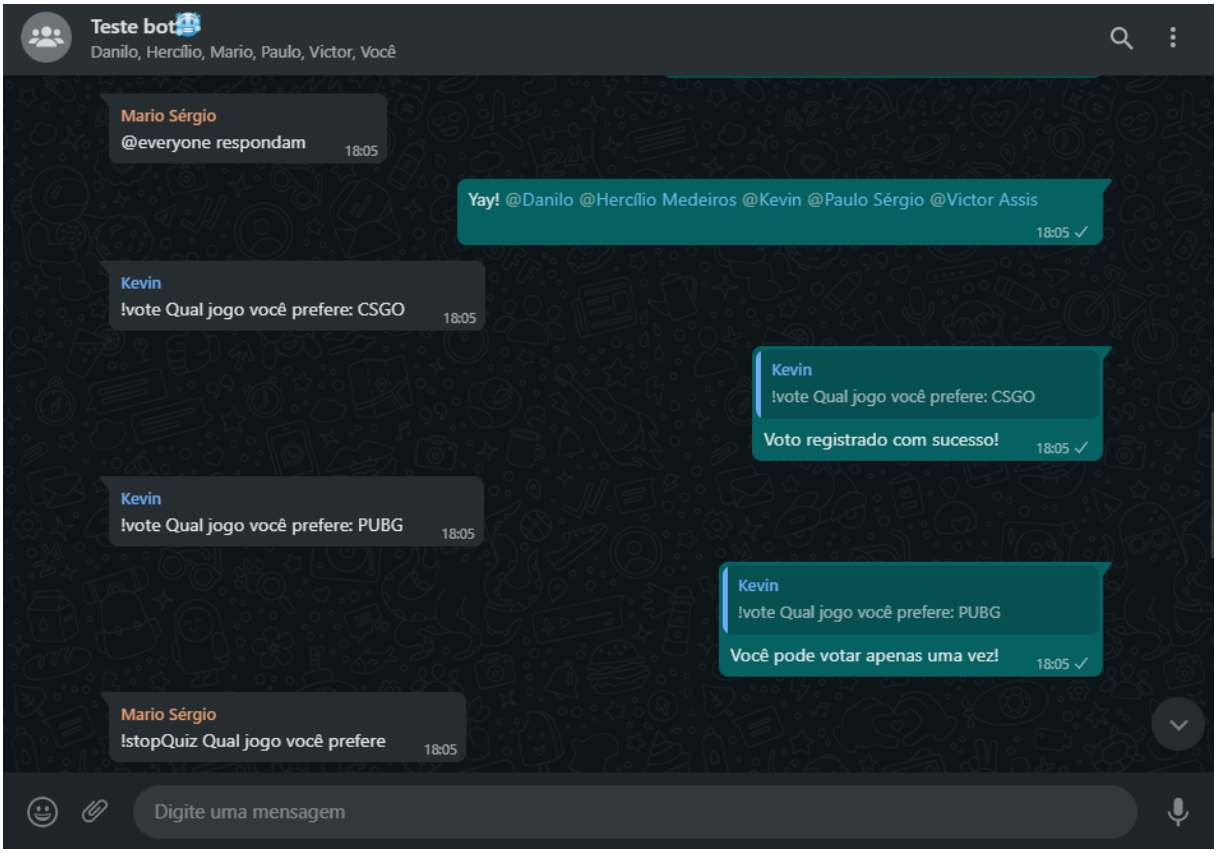
Figura 8 – Criando um quis e suas respostas



Fonte: Autor, 2021.

Na figura 8 é visto a criação de um novo quiz e adicionando duas respostas ao quiz.

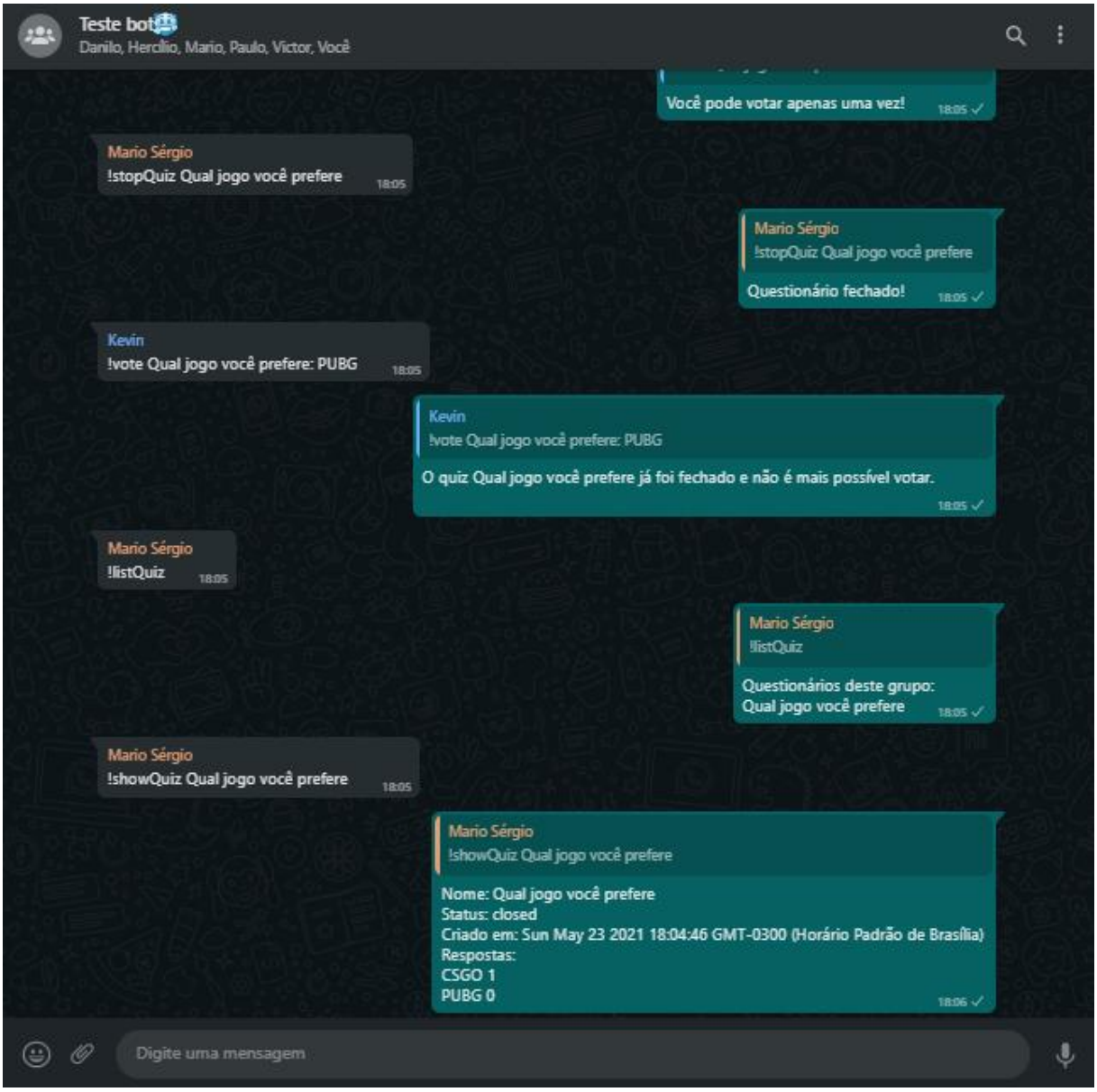
Figura 9 – Marcando e votando



Fonte: Autor, 2021.

Na figura 9 vemos a marcação de todos os participantes do grupo e a votação de um dos integrantes, assim como sua validação para votar apenas uma vez.

Figura 10 – *Quiz* fechado e exibido



Fonte: Autor, 2021.

Na figura 10 é visto o comando para fechar o *quiz*, tornando-o impossível de editar ou votar, logo em seguida a listagem de todos os *quizzes* existentes no grupo, e posteriormente a apresentação de seus detalhes.

CONSIDERAÇÕES FINAIS

O presente trabalho tem como objetivo a praticidade e agilidade de tarefas do dia-a-dia onde temos grupos com até 256 pessoas tornando impossível de marcar todos, e com um simples comando isso agora é possível e imediato.

O Bot4What reduz o tempo de abrir uma nova aplicação e distribuir um link e também alertar todos os participantes a responderem as perguntas em outro

ambiente, no próprio Whatsapp é possível criar uma lista ou questionário e seus participantes ainda no grupo responderem subsequentemente.

A aplicação ainda está em fase inicial e pode evoluir de forma a criar documentos, consultar outras APIs, dentre diversas outras funcionalidades.

O código fonte da aplicação está disponível em: https://github.com/WilliamsJose/bot4what e uma pequena demonstração disponível em: https://youtu.be/7epQ-MsE1d8

REFERÊNCIAS

CRUZ, Leôncio Teixeira; ALENCAR, Antonio Juarez; SCHMITZ, Eber Assis. Assistentes Virtuais Inteligentes e Chatbots: Um guia prático e teórico sobre como criar experiências e recordações encantadoras para os clientes da sua empresa. Rio de Janeiro, Brasil. BRASPORT Livros técnicos e digitais. 2018.

DEVELOPERS, Google. PUPPETEER. 2021. Disponível em: https://developers.google.com/web/tools/puppeteer. Acesso em: 13 mar. 2021.

GARCIA, Reina. Asistente virtual de tipo ChatBot. 2018. Disponível em: https://repository.ucatolica.edu.co/handle/10983/17726. Acesso em: 05 mar. 2021.

MDN, Contribuidores. EXPRESS web framework (Node.js/JavaScript). 2021. Disponível em: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs. Acesso em: 13 mar. 2021.

ORKESTRAL. Venom-bot. 2021. Disponível em: https://orkestral.github.io/venom/index.html. Acesso em: 15 mar. 2021.

PASQUALI, Sandro; FAABORG, Kevin. Mastering Node.js - Second Edition. 2017. Disponível em: https://www.oreilly.com/library/view/mastering-nodejs-/9781785888960/. Acesso em: 13 mar. 2021.

SQLITE. 2021. Disponível em: https://www.sqlite.org/about.html. Acesso em: 31 mar. 2021.

TYPEFORM. 2021. Disponível em: https://typeorm.io. Acesso em: 31 mar. 2021.

TYPESCRIPT. 2021. Disponível em: https://www.typescriptlang.org/docs/. Acesso em: 13 mar. 2021.

VU, Anh. Real-time backend using NodeJS, Express and Google Cloud Platform. 2021. Disponível em: https://www.theseus.fi/handle/10024/429165. Acesso em: 13 mar. 2021.

WHATSAPP Business App. 2021. Disponível em: https://www.whatsapp.com/business/. Acesso em: 26 mar. 2021.

WHATSAPP. 2021. Disponível em: https://www.whatsapp.com/about/. Acesso em: 31 mar. 2021.

UMA APLICAÇÃO MOBILE PARA AUXÍLIO DA QUALIFICAÇÃO DE PRESTADORES DE SERVIÇOS

SANTOS JUNIOR, Paulo Sergio de Oliveira[1]
GOMES FILHO, Carlos Barbosa[2]

INTRODUÇÃO

Com a crescente evolução da internet e mídias sociais, o uso de dispositivos móveis tem proporcionado uma revolução de maior impacto nos últimos tempos. A utilização da internet tornou-se um potencial a ser explorado como forma de elevar a qualidade de vida da população – uma vez que o número de usuários de dispositivos móveis aumenta diariamente. O smartphone, por exemplo, está ganhando um papel cada dia mais importante no auxílio de prestação de serviço pelo mundo todo.

A prestação de serviços por intermédio de um dispositivo móvel é uma prática cada vez mais utilizada em todas as áreas. Nos últimos anos, o assunto vem ganhando força devido à expansão econômica e as oportunidades de crescimento – bem como o aumento das aquisições de smartphones.

Como um aplicativo de prestação de serviços pode melhorar a qualidade dos serviços prestados à população e influenciar no crescimento econômico da Região Nordeste do Brasil?

A motivação principal para o objeto de estudo deste trabalho encontra-se na necessidade e no desejo de melhorar a qualidade na prestação de serviços.

A partir deste estudo surgiram informações que corroboraram na descoberta dos fatores que causaram a diminuição significativa no setor de prestação de serviço em algumas regiões brasileiras. Um dos fatores determinantes foi à qualidade em todo o ciclo do serviço, desde o primeiro contato até o pós-serviço prestado ao cliente.

Este trabalho tem como objetivo observar os aspetos da qualidade da prestação de serviços e criar um aplicativo que gere crescimento vultoso na qualidade da prestação de serviço, bem como facilitar a comunicação entra cliente e a prestador de serviço.

[1] https://www.linkedin.com/in/paulo-sergio-968797161/
[2] http://lattes.cnpq.br/2017611396853275 / https://www.linkedin.com/in/carlos-barbosa-gomes-filho-b0463542/

O objetivo desse trabalho é o desenvolvimento de um aplicativo que facilite o contato entre prestador de serviços e cliente em diversos nichos do mercado ao mesmo tempo que compartilha a qualidade dos serviços prestados, tirando a necessidade de mídia impressa, ligações desnecessárias entre clientes para saber sobre a qualidade do serviço. Atender a necessidade da comodidade de estar em domicilio ou local que melhor for cômodo ao cliente, e assim, oferecer qualidade de vida a ambos os lados.

Parte-se da hipótese de que a melhor forma de um aplicativo de prestação de serviço melhorar a qualidade dos serviços prestados à população e influenciar no crescimento econômico da Região Nordeste do Brasil, é fazer um aplicativo para smartphone que seja: acessível, cômodo, facilite a busca de serviços em todos os nichos do mercado, melhore o processo de comunicação desde o primeiro contado ao fechamento do serviço prestado, e obter o nível de satisfação do cliente, pois grande parte da população tem pouco tempo disponível para pesquisa qual serviço tem total qualidade, já que boa parte está no trabalho, transito, afazeres pessoais, entre outros.

HISTÓRIA DO SMARTPHONE

Para Godoi et al. (2017) a história dos smartphones é atual, e marca décadas de inovação tecnológica, mudança no comportamento e no dia-a-dia das pessoas do mundo todo.

A tecnologia tem um efeito tão intenso e rápido que não sentimos que há menos de uma década não pensávamos em ter tanto apego ao smartphone. O primeiro smartphone foi criado em 1992 pela IBM, com o nome de Simon.

De acordo com Godoi et al. (2017). “Simon foi o primeiro telefone a fundir as funções de um telefone celular e um PDA (*Personal Digital Assistants* – Assistente Pessoal Digital, ou; um computador de dimensões reduzidas) ”.

Além de fazer chamadas, Simon podia: enviar e receber e-mails, faxes, páginas e calcular. Ele também tinha calendário, relógio mundial, agenda de compromissos, e podia abrir aplicativos de outros aparelhos.

Outras empresas como Nokia, Microsoft, Ericsson, e outras, por volta de 1996 lançaram seus aparelhos, que faziam o que o Simon fazia e ainda mais.

Em 2000 foi desenvolvido software Windows Mobile, que só veio chegar ao auge por volta de 2005 e estava rodando em aparelhos com Compaq (Figura 2), HP, HTC, Motorola e Samsung.

Em 2007 o iPhone muda completamente a história do smartphone, com a tela sensível ao dedo. A Apple alavancou a revolução do smartphone (GODOI et al., 2017).

De acordo com Godoi et al. (2017). “Enquanto isso, Andy Rubin estava desenvolvendo sua própria versão de um sistema operacional móvel chamado Android”.

Em 2008 a HTC foi primeira empresa a adotar o Android como sistema operacional após a Google ter comprado o Android de Andy Rubin. Até hoje o Android lidera o mercado de smartphone

Serviços

Serviço é um benefício que uma pessoa de um lado propõe a fazer pela outra, de maneira intangível, ou seja, algo não físico, que não poder ser tocado ou armazenado.

Para Stanton (1980, p 690-691) serviços são “atividades impalpáveis e plenamente identificável que propiciam a satisfação de desejos ou necessidades quando apresentadas aos consumidores e/ou usuários industriais e que não estão, necessariamente, pressas à venda de um produto ou de um outro serviço. ”

Serviços no Brasil

O setor de serviços representa atualmente quase 70%[3] do PIB brasileiro e é considerado o maior empregador do país, segundo o IBGE.

Na tabela 1 é possível observar que, no Brasil, o setor de serviços foi responsável, por 51,5% das ocupações no terceiro trimestre de 2018.

[3] https://www.ibge.gov.br/estatisticas-novoportal/economicas/contas-nacionais/9300-contas-nacionais-trimestrais.html?=&t=resultados

Atividade	%
Agricultura, pecuária, produção florestal, pesca e aquicultura	9,5%
Indústria geral	12,8%
Construção	7,3%
Comércio, reparação de veículos automotores e motocicletas	18,9%
Serviços	51,5%
Total	100,0%

Tabela 1 - Brasil – Participação das atividades econômicas no total de ocupações – 3º trimestre de 2018

Fonte: Elaboração ETENE com dados do IBGE (2018ª).

Biágio (2018) expõe sob uma visão regional, que o Nordeste teve 49% da participação no setor de serviços, considerado abaixo da média nacional, com 54,9%, o Sudeste lidera no percentual do setor.

Biágio (2018) ainda revela que o segmento de serviços profissionais, recuou 1,9% e levando junto a queda do índice em 2018 - foi a quarta taxa negativa seguida, porém, a menos intensa, já que 2017 registrou desfavorável (-7,3%).

IBGE (2018) mostra que, faltando pouco menos de 200 mil para chegar ao seu primeiro milhão de habitantes, e com bom potencial econômico, a cidade de João Pessoa ainda é a dona de um PIB que é totalmente dependente do setor de serviços. O PIB de João Pessoa, aferido em R$ 18,3 bilhões pelo IBGE, é um pouco maior que o dobro da segunda economia paraibana, Campina Grande (R$ 7,9 bilhões).

Aplicativos que prestam serviços

Dino (2018) mostra que muitos aplicativos de serviços vêm crescendo muito no mercado para facilitar a forma de obter e prestar serviços, a fim de que uma vez solicitado, possa ser rapidamente atendido para resolver o problema com qualidade e eficiência. Sabendo que o aplicativo tem segurança e confiança dos prestadores de serviços, uns dos aplicativos está sendo muito utilizado é o Uber, uma empresa

multinacional americana, prestadora de serviços na área do transporte privado urbano, através de um aplicativo de transporte que permite a busca por motoristas baseada na localização.

O serviço está cada vez mais solicitado por aplicativo, facilitando cada vez mais a vida do usuário, sempre querendo mais facilidade, rapidez e comodidade. Por isso a empresa vem criando aplicativos sabendo das necessidades do usuário. Um dos aplicativos que vem crescendo muito, iFood fundada em 2011, atuante no ramo de entrega de comida pela internet, é uma das opções mais acessíveis para pedir comida em casa ou no trabalho. Permite pedir refeições de restaurantes ou lanchonetes próximos, diretamente do celular, sem precisar gastar créditos com ligações.

Qualidade na Prestação de Serviços

A principal função da prestação de serviços é atender o cliente de forma tangível e/ou intangível, com a mesma qualidade. A qualidade no atendimento é um dos fatores mais importantes na hora de escolher um serviço, mas, vem perdendo a atenção do setor nos últimos anos, e é mostrado por Meir (2015) que:

> O setor enfrenta algumas reclamações, que, normalmente, são consequência da forma pela qual muitas empresas são contratadas, em que, muitas vezes, o preço é um fator mais importante do que a qualidade do atendimento. Há ainda a questão da fragilidade dos sistemas tecnológicos e da pouca integração entre os canais de atendimento. O Brasil é o país que menos investe nisso. Esse é um gargalo brutal. (MEIR, 2015, p. 8).

(FITZSIMMONS; FITZSIMMONS, 2014) mostram que, de acordo com estudos, existe três conceitos de sociedade: a pré-industrial onde a condição é de subsistência, a industrial que predomina produção de mercadoria e a pós-industrial, onde, diferente da industrial que se define pela quantidade de bens, a sociedade pós-industrial está mais preocupada com a qualidade dos serviços.

O prestador de serviços tem que ter atenção a eficácia e excelência do atendimento, e ser estratégico ao conduzir esse diferenciador.

Os clientes se tornaram mais críticos em relação à qualidade dos serviços, e passaram a avaliar os aspectos tangíveis e intangíveis do serviço prestado. Atualmente, nota-se que o consumidor está consultando outras pessoas que já tenham utilizado um serviço específico, antes de adquiri-lo.

Parasuraman (1985) mostra que em resposta a este cenário competitivo, muitos prestadores de serviços têm buscado superar as necessidades e expectativas dos clientes – estes prestadores procuram várias formas de superar a concorrência e garantir a fidelidade dos clientes em relação aos seus produtos e serviços. Entretanto, observa-se que a qualidade dos serviços prestados ainda deixa a desejar em muitos aspectos.

Aplicativos e seus meios de mensuração de qualidade

Um exemplo de como os serviços são mensurados, é a forma com que os aplicativos são avaliados. A forma mais conhecida é por classificação, onde o cliente de maneira gráfica dá uma nota de avaliação e comenta sobre o aplicativo. O Android é um exemplo desse tipo de método de avaliação.

O Android usa essa classificação como uma forma de prestação de serviço não obrigatório aos seus clientes, desta forma podem ajudar outros clientes à escolher aplicativos e fazerem a melhor escolha dos aplicativos colocados na Play Store.

Em pesquisa realizada em vários aplicativos do *Play Store* mostram formas de diferentes de avaliação onde, observa-se uma avalição superficial das qualidades do prestador de serviços, simplificado em desempenho do Profissional e custo. Se o profissional fez um bom trabalho, mas foi rude ou se atrasou, não será avaliado porque não existe uma forma específica que o avalie. O Profissional não saberá onde melhorar sem os requisitos de avaliação necessários.

Segundo pesquisa feita por Murphy (2018), 20% dos clientes solicitados a postarem uma avaliação não o fizeram e 58% dos clientes prestam mais atenção à classificação geral de estrelas, ganhando para os 47% de cliente que preferem visualizar os sentimentos dos *reviews*.

Na pesquisa realizada pela Murphy (2018), foi questionado quantas avaliações online os clientes leem antes de confiar em um negócio.

Com os dados obtidos pela Murphy (2018) conclui-se que:

- 90% dos consumidores leem 10 comentários ou menos antes de sentir que podem confiar em um negócio;
- 68% dos consumidores opinam lendo apenas 1-6 comentários;
- Apenas 10% dos consumidores passam tempo pesquisando e lendo mais de 10 comentários (contra 13% em 2015).

Em um segundo momento Murphy (2018) questionou se ao julgar um negócio local através das avaliações dos consumidores, quais fatores são mais importantes.

E conclui-se que:

- 58% dos consumidores prestam mais atenção à classificação geral de estrelas;
- Para 47% o sentimento dos *reviews* é o segundo fator mais importante;
- *Reviews* mais recentes também são importantes, ficando em terceiro com 41% (MURPHY, 2018).

Conclusões geral:

- 7 de cada 10 consumidores irão postar uma avaliação se forem convidados a fazê-lo;
- 50% dos consumidores que foram convidados a postar uma avaliação o fizeram de fato;
- 20% dos consumidores solicitados a postarem uma avaliação NÃO o fizeram (MURPHY, 2018).

A pesquisa aponta que grande parte das pessoas classificam a qualidade de um serviço de maneira gráfica, ou seja, através de estrelas que exibe a classificação do prestador de serviços.

METODOLOGIA

A presente pesquisa enquadra-se quanto aos fins a que se destina como do tipo bibliográfica, descritiva, aplicada. Bibliográfica porque foi baseada em material publicado como: livros, revistas, sites na internet. Como esse método de pesquisa, encontrou-se a evolução dos smartphones (seção 2.1.1) – bem como os aplicativos, – que estão cada vez mais ocupando todos os setores de prestação de serviços

como o iFood, Uber, entre outros (seção 2.1.4). Baseando-se em revistas e sites mais atuais, foi feita uma pesquisa descritiva que pode expor como os serviços estão perdendo qualidade e quantidade no nordeste brasileiro (seção 2.1.3). Com base na problemática foram realizadas pesquisas que indicaram formas de se mensurar a qualidade dos serviços de maneira eficaz (seção 2.1.5).

Foi realizada uma pesquisa mais minuciosa por documentos de órgãos privados para descobrir como os usuários de smartphone estão usando as formas de avaliação de serviços prestados (seção 2.1.6).

Aplicada porque é motivada pela necessidade de resolver problemas reais, portanto, com finalidade prática. A referente pesquisa aplicada teve como base as pesquisas anteriores, descritas no corpo da presente metodologia (seção 2.2), onde, analisou-se a necessidade de realizar uma pesquisa de campo – de forma quantitativa – que se mostrou muito eficaz, pois através de um rápido questionário online de cinco perguntas objetivas feito com oitenta e oito pessoas (seção 2.4), descobriu-se como os smartphones estão sendo usados para avaliar os serviços prestados na cidade de João Pessoa.

RECURSOS TECNOLÓGICOS UTILIZADOS

Android Studio é um ambiente de desenvolvimento integrado (IDE) para desenvolver para a plataforma Android. Baseado no software IntelliJ IDEA de JetBrains, Android Studio foi feito especificamente para o desenvolvimento para Android (ANDROID... 2019).

O Android Studio foi utilizado no desenvolvimento do aplicativo com a linguem Java. Virtual Private Server (VPS) é um plano de hospedagem compartilhada (L, 2018). O hostinger foi utilizado para hospedar o Banco de dados.

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface (MYSQL, 2019). O MySQL foi utilizado para gerenciar o banco de dados.

O *Sublime Text* é um software multiplataforma de edição de texto, no entanto utilizado por muitos desenvolvedores para editar código-fonte, escrito em linguagem *Python* (SUBLIME... 2019).

O *Sublime Text* foi utilizado como IDE na finalidade de manipular os arquivos PHP do servidor em tempo real.

Amaral (2016) explica que “o protocolo FTP (*File Transfer Protocol*) é um dos meios mais populares para fazer upload de arquivos para servidores web. Portanto, é natural que a integração do *Sublime Text* com um cliente de FTP seja uma funcionalidade bastante desejada. Como o próprio nome sugere, o FTPSync serve para sincronizar seu projeto com seus arquivos remotos. A sincronização é feita a cada vez que um arquivo do projeto é salvo”. O FTP Sync foi usado para obter acesso remoto.

Escudelario (2018) explica que “Postman é um aplicativo com a função de testar e desenvolver APIs em uma interface bastante simples e intuitiva. Ele nos permite simular requisições HTTP de forma rápida, armazenando-as para que possamos usá-las posteriormente”.

O Postman foi utilizado para testar os códigos e ver se o PHP está aceitando os formulários e adicionando no banco.

RESULTADOS OBTIDOS

A análise e interpretação dos dados coletados são apresentadas em cinco partes: a primeira parte refere-se ao perfil parte refere-se o perfil dos entrevistados (seção 2.4.1), em seguida as pessoas que solicitam serviços por Smartphone (seção 2.4.2), logo depois, foi perguntado quantas vezes por semana utilizavam o Smartphone para solicitar serviços (seção 2.4.3), procurou-se saber se as pessoas avaliavam os serviços, após serem concluídos (seção 2.4.4), por fim, se a população gostaria de avaliar os serviços prestados de forma mais detalhada, como na Figura 34 (seção 2.4.5).

Perfil dos entrevistados

As faixas etárias predominantes correspondem às idades entre 15 e 20 anos (35,2%) e 21 e 25 anos (28,4%), perfazendo 63,6% das pessoas questionadas, conforme ilustrado na Figura 17.

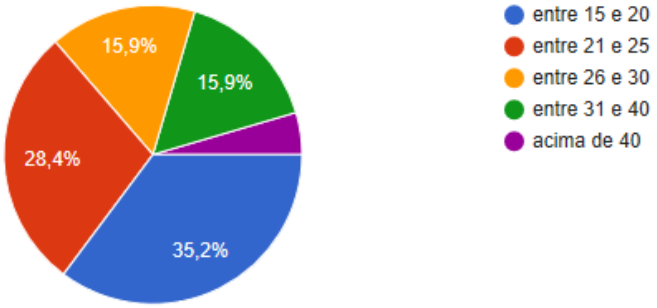


Figura 3 – Perfil dos entrevistados

Fonte: Criado pelo Autor.

Pessoas que solicitam serviços por Smartphone

Na Figura 18, é notável que quase todas as pessoas (96,6%) questionadas utilizam seu Smartphone para solicitar serviços, três pessoas afirmaram não utilizar. Uma das pessoas tinha acima de 40 anos, as outras dois, tinham entre 26 e 30 anos.

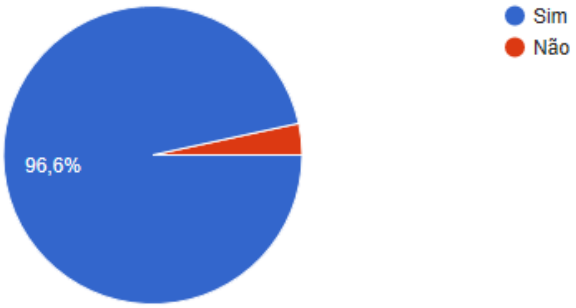


Figura 4 – Pessoas que solicitam serviços por Smartphone

Fonte: Criado pelo Autor.

Pessoas que utilizam semanalmente o Smartphone para solicitar serviços

Aqui (Figura 19) é mostrado que, 21,2% das pessoas usam dez vezes por semana o Smartphone para solicitar serviços, e 12,1% usam sete vezes por semana, juntamente com 8% das pessoas que usam 8 ou 9 vezes na semana, totalizando 41% que utilizam mais de 6 vezes por semana.

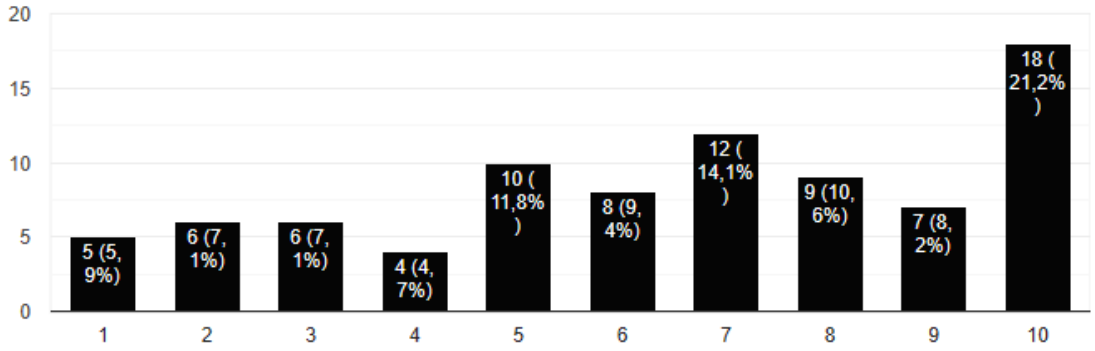


Figura 5 – Pessoas que utilizam semanalmente o Smartphone para solicitar serviços

Fonte: Criado pelo Autor

Pessoas que avaliam o serviço prestado

De acordo com a Figura 8 da seção 2.1.6, 58% dos consumidores prestam mais atenção à classificação geral de estrelas[4].

Aqui (Figura 20) percebesse que 47% das pessoas sempre avaliam os serviços, 33,7% avaliam pelo menos entre 1 a 5 vezes que usam serviços, conclui-se que ainda existe uma queda no número de pessoas que se propõe a avaliar serviços que lhes são prestados.

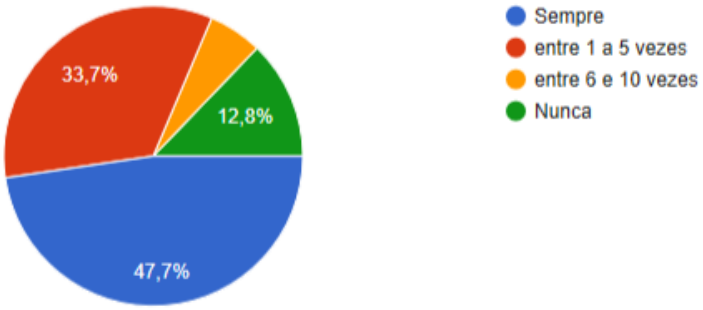


Figura 6 – Pessoas que avaliam o serviço prestado

Fonte: Criado pelo Autor

[4] Forma gráfica de nível que avalia de 0-5.

Pessoas que gastariam de avaliar os serviços através do aplicativo GoodJob

Na Figura 21, pode-se afirmar que a forma de avaliação eficiente sugerida por esse trabalho (Figura 34), foi aceita por 78,4% das pessoas entrevistadas, onde, 12,5% disseram que talvez usassem a forma de avaliação.

Toda via, conclui-se que a forma de avaliação sugerida por esse trabalho indica uma nova visão no setor de serviços e a partir disso, pode-se elevar a quantidade e a qualidade dos serviços na cidade de João Pessoa e progressivamente em toda Região Nordeste.

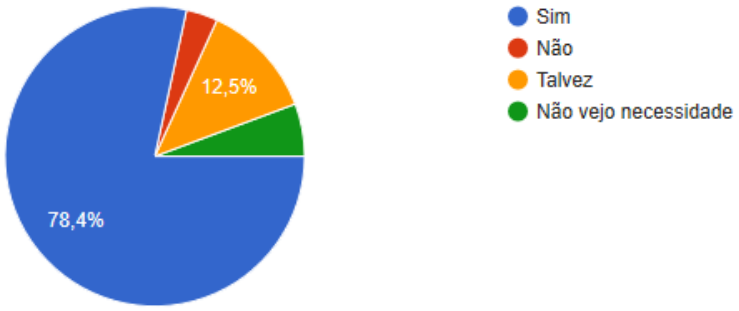


Figura 7 – Pessoas que gastariam de avaliar os serviços através do aplicativo GoodJob

Fonte: Criado pelo Autor

PROJETO

Foi usado a linguagem PHP para criar as seguintes APIs listadas na Figura 22. Esses arquivos foram inseridos no servidor de hospegadem no subdominio conect.vitaldata.com.br permitindo manipular as informações em tempo real do banco de dados online na internet.

As classes são separadas de acordo com o algoritmo planejado para as ações feitas no aplicativo, separadas por pastas e nomes específicos, encapsulando as variáveis e criando objetos de forma organizada e eficiente.

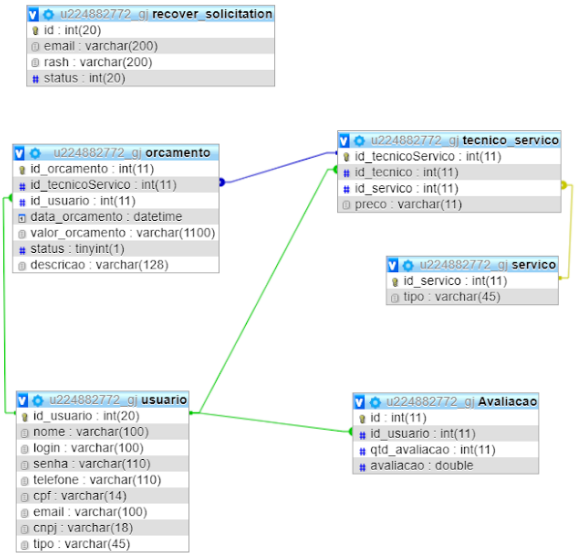


Figura 8 – Banco de dados online (PhpMyAdmin)

Fonte: Criado pelo Autor.

Com interface moderna, agradável e intuitiva, sempre pensando na experiência do Técnico e o Cliente. Na primeira tela chamada “tela de Login” (Figura 26), você insere os dados cadastrados para entrar em seu perfil de forma segura. O aplicativo possui em sua plataforma a criptografia MD5 no PHP, para manter os usuários do aplicativo assegurados que suas senhas não serrão violadas.

Você pode visualizar a senha caso esteja com dúvidas, e se esquecer de sua senha, pode redefinir através da opção “Esqueci minha senha”, onde receberá um e-mail para realizar sua redefinição.

Figura 9 – Tela: Login

Fonte: Criado pelo Autor.

Se você não é cadastrado, basta clicar em “cadastrar”, localizado logo abaixo do botão “Entrar”, vai aparece o seguindo formulário de cadastramento mostrado na Figura 27.

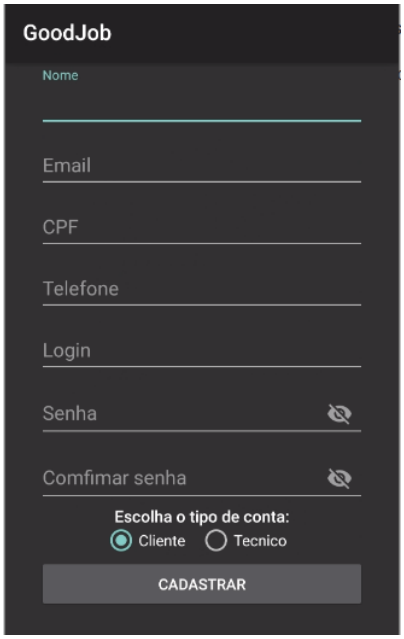


Figura 10 – Tela: Cadastro (para cliente ou técnico)

Fonte: Criada pelo Autor.

No cadastro existem campos que são obrigatórios, não vai conseguir se cadastrar caso não preencha corretamente. As APIs verificam os dados e realiza o cadastro no banco de dados, você pode escolher se quer ser apenas “Cliente”, que pode apenas solicitar serviços, ou ser “Técnico” que também pode contratar serviços, sendo que também vai ter a possibilidade de oferecer serviços.

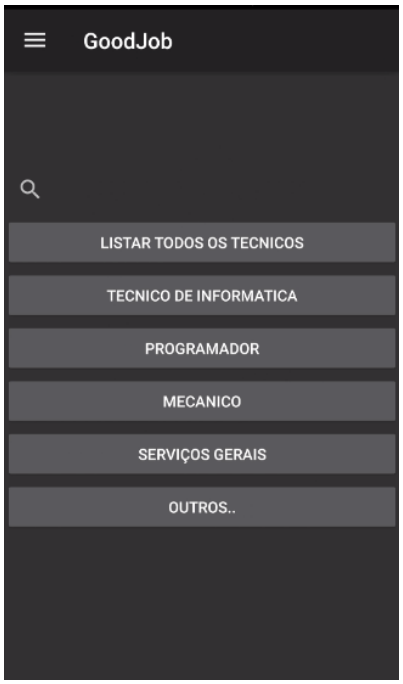


Figura 11 – Tela: Principal

Fonte: Criada pelo Autor.

Nessa tela principal, só é listado apenas os técnicos da sua cidade, você poderá pesquisar de acordo com o que precisa. De acordo com as opções, vai aparecer a lista dos técnicos e seus dados e serviços que ele oferece.

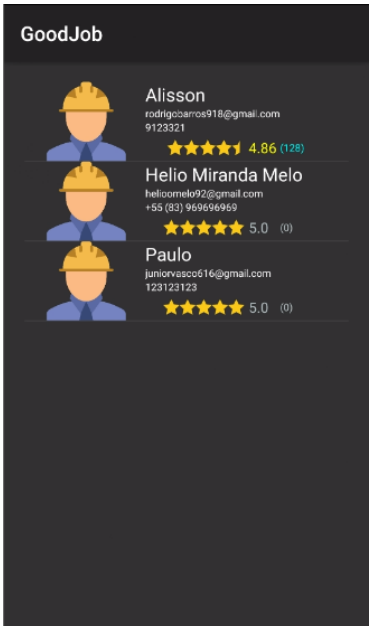


Figura 12 – Tela: Listando os técnicos

Fonte: Criada pelo Autor.

Ao clicar no botão “listar todos os técnicos” da Figura 29, irão aparecer os técnicos com cadastros de serviços ativos. Nela exibe o nome do técnico, e-mail, número para contato e o mais importante que é a média das avaliações descritas por seus clientes, ao lado direito contém o valor numérico da média das avaliações referente ao técnico.

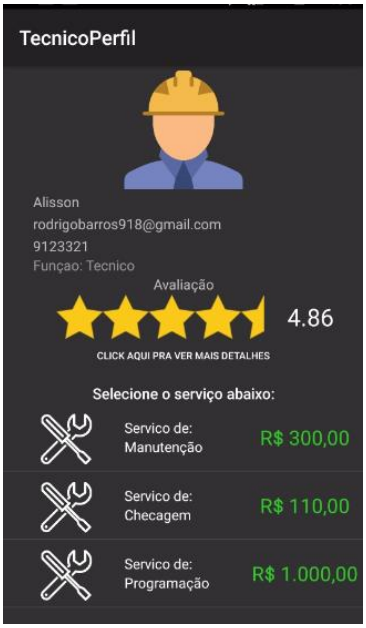


Figura 13 – Tela: Perfil do Técnico

Fonte: Criada pelo Autor.

O Cliente, ao selecionar o técnico, é mostrado a tela de “Perfil do Técnico” (Figura 30), onde é listado todos os serviços que ele oferece.

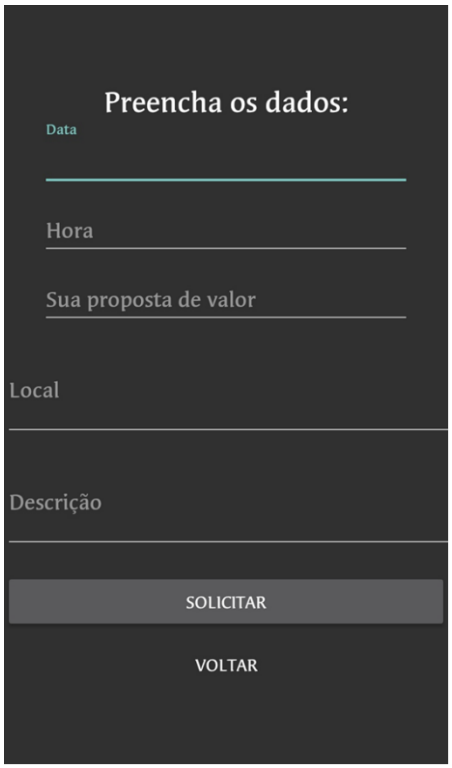


Figura 14 – Tela: agendamento de serviço

Fonte: Criada pelo Autor.

Nesta tela (Figura 31) o cliente irá solicitar um agendamento, propondo a data, o horário e uma contraoferta, o local onde será realizado o serviço. No campo “descrição” o cliente irá descrever de forma breve o sintomas e sentimentos relacionados ao problema.

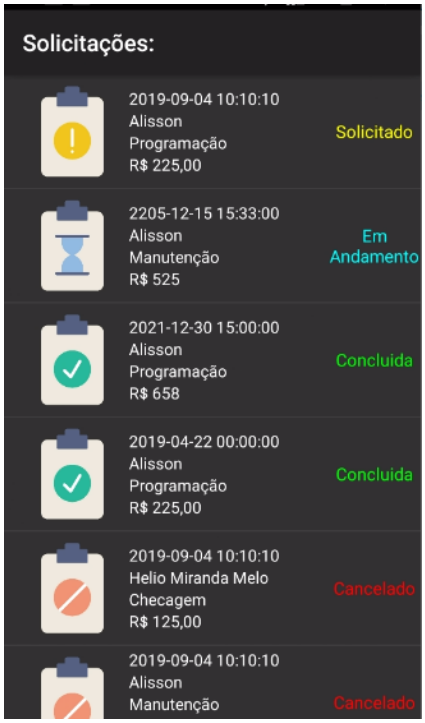


Figura 15 – Tela: Lista de solicitações

Fonte: Criada pelo Autor.

A tela “Lista de solicitações” (Figura 32), exibe todas as solicitações, status de suas situações do serviço, data e hora da solicitação, nome do técnico e o serviço solicitado.

Figura 16 – Tela: Status do serviço

Fonte: Criada pelo Autor.

A tela “Status do serviço” (Figura 33) exibe os detalhes do serviço, como: valor acertado, horário combinado, aqui o cliente poderá cancelar o serviço, também será exibido o status de sua situação em relação a execução do serviço.

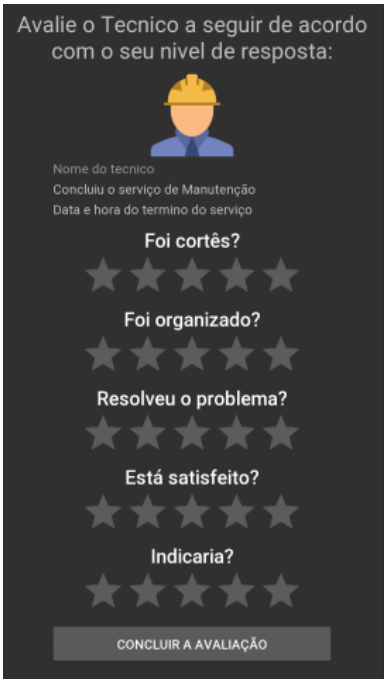


Figura 17 – Tela: Avaliação do serviço

Fonte: Criada pelo Autor

Como uma formulação mesclada dos Determinantes de qualidade de serviço (Quadro 10), a tela de “avaliação do serviço” (Figura 34) mensura a qualidade da prestação de serviço. Essa nova forma de medir a qualidade irá ajudar os prestadores de serviços a identificar as possíveis inadequações mais específicas dos seus serviços e corrigi-las. As experiências que serão avaliadas aqui serviram para mensurar a expectativa de um cliente futuro.

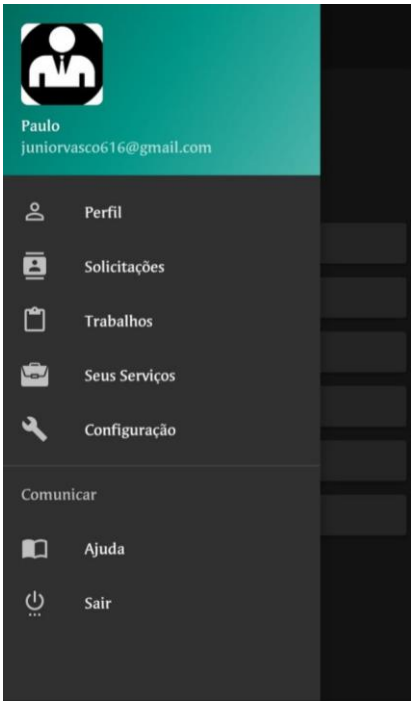


Figura 18 – Tela: Menu de navegação (Técnico)

Fonte: Criada pelo Autor

A tela “menu de navegação (Técnico) ” (Figura 35) contém opções específicas apenas para o técnico e algumas em comum com o menu de navegação do cliente (Figura 31):

- Perfil – Mostra dados do técnico podendo ser editado;
- Solicitações – Exibe as solicitações feitas do cliente para um técnico;
- Trabalhos – O técnico visualiza os trabalhos em andamento;
- Seus serviços – Mostra os serviços que o técnico oferece.

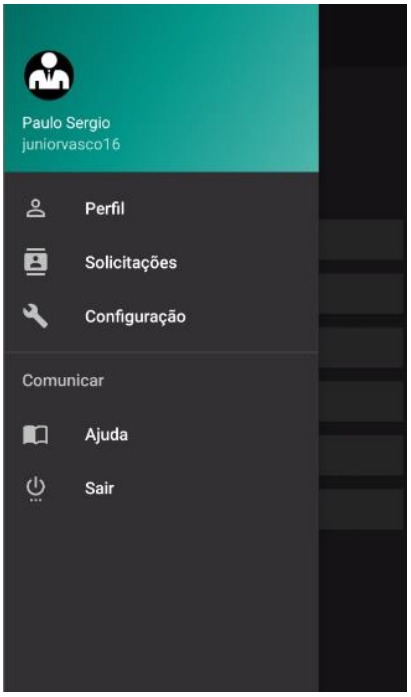


Figura 19 – Tela: Menu de navegação do Cliente

Fonte: Criada pelo Autor

A tela “menu de navegação do cliente” (Figura 36) contém opções para a navegação do cliente:

- Perfil – Mostra dados do técnico podendo ser editado;
- Solicitações – Exibe as solicitações feitas do cliente para um técnico.

CONCLUSÃO

A prestação de serviços por intermédio de um dispositivo móvel é uma prática cada vez mais utilizada em todas as áreas. Os smartphones têm aumentado suas funcionalidades, junto delas vieram os aplicativos de serviços que, no toque de um dedo, chega comida na sua casa, alguém conserta seu encanamento e etc.

O desenvolvimento do presente estudo possibilitou um analise bibliográfica e documental, que corroboraram na descoberta dos fatores que causaram a diminuição significativa no setor de prestação de serviço em algumas regiões

brasileiras. Um dos fatores determinantes foi à qualidade em todo o ciclo do serviço, desde o primeiro contato até o pós-serviço prestado ao cliente.

Em virtude de todos os fatos mencionados, é possível afirmar que a criação deste aplicativo é viável. De acordo com as pesquisas realizadas, 74,4% das pessoas entrevistadas aprovaram a forma de avaliação do GoodJob, entendendo-se que o aplicativo é de grande aceitação pelo público, pois este aplicativo está focado nas principais deficiências do setor de prestação de serviços. O aplicativo GoodJob tem como prioridade atender o cliente de forma tangível e/ou intangível, com a mesma qualidade. As experiências dos clientes anteriores são muito importantes na hora de escolher um serviço. Esse e o diferencial do GoodJob.

Este trabalho mostra à forma mais prática e intuitiva de mensurar a qualidade da prestação de serviço. Essa nova forma de medir a qualidade irá ajudar os prestadores de serviços a identificar as possíveis inadequações mais específicas do processo de atendimento ao cliente, e corrigi-las.

Conclui-se que a forma de avaliação sugerida por esse trabalho indica uma nova visão no setor de serviços e a partir disso, pode-se elevar a quantidade e a qualidade dos serviços na cidade de João Pessoa e progressivamente em toda Região Nordeste.

REFERÊNCIAS

AMARAL, Rodrigo. **Como transferir e sincronizar arquivos via FTP no Sublime Text**. 2016. Disponível em: <http://sublimetextdicas.com.br/como-transferir-e-sincronizar-arquivos-via-ftp-no-sublime-text/>. Acesso em: 26 abr. 2019.

ANDROID Studio. 2019. Disponível em: <https://pt.wikipedia.org/wiki/Android_Studio>. Acesso em: 26 abr. 2019.

BIÁGIO JUNIOR, O.M**. Caderno Setorial – ETENE/BNB:** Perspectivas para o setor de serviços 2018/2019, Ano3, nº 8, Dezembro/2018.

DINO (Brasil) (Ed.). **Tendências para o mercado de apps em 2018.** 2018. Disponível em: <https://exame.abril.com.br/negocios/dino/tendencias-para-o-mercado-de-apps-em-2018>. Acesso em: 30 jan. 2018.

ESCUDELARIO, Bruna de Freitas. **Ganhando eficiência em suas APIs com o Postman.** 2018. Disponível em: <https://imasters.com.br/apis-

microsservicos/ganhando-eficiencia-em-suas-apis-com-o-postman>. Acesso em: 26 abr. 2019.

FITZSIMMONS, James A.; FITZSIMMONS, Mona J.. O papel dos serviços na economia. In: FITZSIMMONS, James A.; FITZSIMMONS, Mona J.. **Administração de Serviços**: Operação, estratégia e tecnologia da informação. 7. ed. Porto Alegre: Amgh Editora Ltda, 2014. Cap. 1. p. 4-16.

GODOI, Susane de et al. **Esta é a história dos smartphones**. 2017. Disponível em: <https://mobizoo.com.br/curiosidades/a-historia-dos-smartphones>. Acesso em: 02 mar. 2019.

IBGE. "**Produto Interno Bruto dos Municípios:** PIB dos municípios". 2016. Disponível em: <**https://www.ibge.gov.br/estatisticas-novoportal/economicas/contas-nacionais/9088-produto-interno-bruto-dos-municipios.html?=&t=resultados>.**
Acesso em: 23/03/2018.
L, Andrei. **Para Que Serve VPS e Quando Usar um Servidor VPS.** 2018. Disponível em: <https://www.hostinger.com.br/tutoriais/para-que-serve-vps>. Acesso em: 26 abr. 2019.

MEIR, Roberto. Bandeira branca para crescer. **Consumidor Moderno**, Pacaembu, v. 199, n. 1, p.7-8, fev. 2015. Mensal.

MURPHY, Rosie. **Local Consumer Review Survey**. 2018. Disponível em: <https://www.brightlocal.com/research/local-consumer-review-survey>. Acesso em: 17 abr. 2019.

MYSQL. 2019. Disponível em: <https://pt.wikipedia.org/wiki/MySQL>. Acesso em: 26 abr. 2019.

PARASURAMAN, A.; ZEITHAML, Valarie A.; BERRY, Leonard L.. A Conceptual Model of Service Quality and Its Implications for Future Research. **Journal Of Marketing**. p. 41-50. 1985.

RUI SILVA (Brasil). **Android: Como classificar uma app no Play Store?** 2017. I-Técnico. Disponível em: <http://www.i-tecnico.pt/Android-como-classificar-uma-app-no-play-store>. Acesso em: 17 abr. 2019.

STANTON, William John. **Fundamentos de marketing**. São Paulo: Pioneira, 1980.

SUBLIME Text. 2019. Disponível em: <https://pt.wikipedia.org/wiki/Sublime_Text>. Acesso em: 26 abr. 2019.

USO DA TECNOLOGIA JAVA EM UM SISTEMA PARA ESCANEAMENTO DE DOCUMENTOS PELO PROGRAMA FARMACIA POPULAR

COUTINHO, Rafael Ewerton F. de Oliveira[1]
ROCHA Gláucio Bezerra [2]

INTRODUÇÃO

Com o objetivo de ampliar a aproximação de medicamentos no Brasil de forma descentralizada foi instituída no ano de 2004, pela lei n° 10.858, e regulamentado pelo Decreto n° 5.090 de maio de 2004, o Programa Farmácia Popular do Brasil (PFPB). Inicialmente a disponibilização de medicamentos era feita pelas suas unidades próprias, que na época tinham parcerias com os governos estaduais, instituições públicas e prefeituras municipais. Essas unidades possuíam um estoque de 112 itens, que eram diversificados em remédios e preservativos masculinos, os quais era dispensados com umaredução de até 90% de mercado. Para a realização de aquisição desses medicamentos, primeiramente tinham que estar disponíveis na unidade e liberados com a apresentação de algum documento com foto, CPF e um receita médica (PINTO, COSTA E CASTO; 2011).

A partir de 2017, foi decretado o fim de repasses para estas unidades próprias, e pactuado que o Ministério da Saúde iria de forma integral repassar as verbas que eram destinadas a manutenção das unidades próprias, para o financiamento da Assistência Farmacêutica Básica, em 100% do território brasileiro, sendo assim um maior investimento para a compra de medicamentos que são essenciais à população.

Em 9 de março de 2006, o Ministério da Saúde expande o programa farmácia popular do Brasil, e recebe o nome de “Aqui tem Farmácia Popular”, na qual adota e aproveita o comercio varejista de produtos farmacêuticos, e são disponibilizados remédios de diabetes e hipertensão com abatimento de até 90%do valor. Em abril de 2010 são incluídas no programa a Insulina Regular e Sinvastatina, que serve para combater o colesterol alto e em outubro, foram incluídos medicamentos para o tratamento de osteoporoses, rinite, asma,

[1] https://www.linkedin.com/in/rafaelewerton/
[2] http://lattes.cnpq.br/2885241414019869

Parkinson e Glaucoma além de incluir a disponibilização de fraldas geriátricas para idosos no tratamento de incontinência urinaria. Atualmente, o PFPB disponibiliza mais de 1000 itens a aquisição desses medicamentos ainda são necessários documento com foto, CPF, receita médica.

Segundo o PFPB, os documentos necessários para dispensação dos medicamentos (cópia dos documentos e receita médica) devem ser arquivados por cinco anos, período mínimo para armazenamento. Ainda segundo o programa, as receitas valem por um mês, porém, devido ao cenário da pandemia esse prazo de validade foi estendido por um ano. Vale salientar, que as farmácias credenciadas no programa precisam repassar ao Ministério da Saúde as vendas realizadas (BRASIL, 2004).

Nesse contexto, o uso da tecnologia pode beneficiar e facilitar o processode armazenamento das documentações necessárias. A tecnologia nesse cenário é de extrema importância, considerando os ganhos com segurança, armazenamento de dados, integridade, acessibilidade, entre outros. Por meio da utilização de softwares é possível solucionar questões como o armazenamento físico de documentos e controle das vendas. Destaca-se que o armazenamento em nuvem, pode trazer mais robustez para farmácia que é credenciada com a farmácia popular.

O objetivo do presente trabalho é o desenvolvimento de um sistema para auxiliar as farmácias que possuem credenciamento com o PFPB, para a realização de digitalização de documentos, que são de caráter obrigatório para a venda, o sistema também irá incluir o controle de cadastros da farmácia, uma aba para digitalização e pesquisa de venda.

Ao final deste trabalho com o sistema web elaborado, os usuários das farmácias poderão realizar configurações do *scanner*, escanear toda documentação do cliente que realizou a compra pelo PFPB, também poderão realizar consultas das vendas, editar documentos e realizar a exclusão, caso necessite.

2 FUNDAMENTAÇÃO TEÓRICA

Considerando a importância do desenvolvimento do Sistema para

escaneamento de documentos citado acima, a linguagem que será utilizada é o Java, e um servidor e gerenciador de banco de dados (SGBD) relacional, que é essencial para aplicações de pequenos e de grandes portes o MySql.

2.1 Java

A *Sun Microsystem*, segundo Deitel e Deitel (2010), anunciou o Java em maio de 1995. Teve grande impacto nas pessoas pois, é uma linguagem voltada para a *Web*. O Java atualmente é utilizado para criação de aplicativos coorporativos de grande escala, melhoramento de funcionalidades para os servidores, dispõe aplicativos para dispositivos entre outros objetivos que o Javaé capaz de realizar.

Eclipse IDE(*Integrated development environment*) – É um ambiente de desenvolvimento Java, que pode também suportar outras linguagens com o auxílio de *plugins, como Kotlin, Python, C/C++,* apesar do auxílio de *plugins* no eclipse eles são pouco utilizados atualmente. O Eclipse é uma ferramenta que possui o modelo *open source*. para programadores escrever, implementar,realizar depurações e compilação de código.

Segundo Horstmann e Cornell (2010), o Java sempre foi uma linguagem com muita robustez. Atualmente existem várias outras linguagens de programação que é comparada ao alcance que o Java obteve, porém poucas se aproximam do potencial do Java. O Java apresenta também uma biblioteca imensa com várias quantidades de códigos disponíveis para serem reutilizados, e é uma plataforma de execução que fornece aos serviços segurança, portabilidade e acessibilidade para diversos tipos sistemas operacionais.

Melo (2010) afirma que por volta do ano de 1970, iniciou-se uma nova metodologia formada por orientação a objeto na qual reforçou ainda mais as mudanças de paradigmas que eram desenvolvidos anteriormente. A partir dela, a programação veio a crescer exponencialmente, na qual deu-se início a análise de orientação a objeto e trouxe a padronização da implementação, análise e projeto.

Deitel e Deitel (2010) explicam que, em 1980, as organizações começaram a utilizar a orientação a objetos para a criação de aplicativos e desenvolveram a OOAD (*Object Oriented Analysis and Design*), na qual muitos estudiosos e metodologistas produziram implementações com processos separados para algumas necessidades presentes, onde cada processo e código possuía sua

própria marca para conduzir os resultados de *design* e análise. No entanto, eles poderiam modelar o software utilizando equivalências informando objetos da realidade e aperfeiçoando o desenvolvimento do modelo.

Segundo Paula Filho (2009), na área de arquitetura de software os procedimentos podem ser destinados para tarefas como desenvolvimento, manutenção, alcance e contratação de um sistema. Em um processo de desenvolvimento de sistemas, inicialmente para a arquitetura este processo é a escolha de modelo de ciclo de vida.

Em meados da década de 1970, criou-se um padrão de projeto para softwares na qual chamava-se MVC, uma arquitetura que significa *Model, View, Controller,* este padrão tinha como base a reutilização dos códigos e o auxílio de separar os acessos de dados e regras de negócio da forma que é exibida ao cliente. O MVC é composto detalhadamente por:

- *Model* (Modelo) – Com ele é apresentado todos os dados e regras de acesso aos dados, pode-se dizer que basicamente é uma forma de aproximaçãodo software com o mundo real.
- *View* (Vizualização) – É a camada que faz a interação com o usuário, serve basicamente para apresentar os dados.
- *Controller* (Controlador) – Totalmente responsável por receber as requisições que o usuário oferece, controla para fazer a utilização de qual modelou view vai ser apresentado ao cliente. A figura abaixo mostra como é o funcionamento, a entrada do usuário, a modelagem e o *feedback* da *view.*

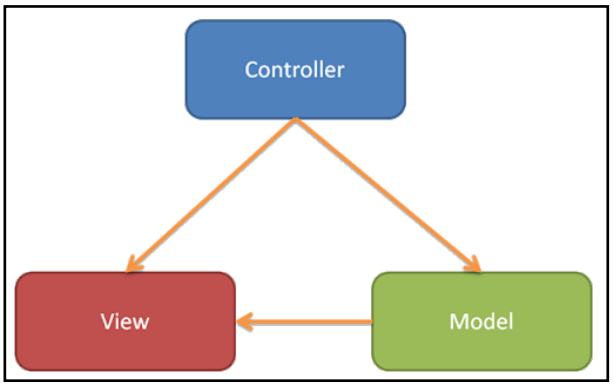


Figura 1 – Modelo MVC (Controller, Model, View)Fonte: MVC, padrões de projeto, Tarcisio (2013)

É demonstrada a entrada do usuário, a modelagem dos dados com o mundo de fora ou externo, a parte de visualização para o usuário e sua interação com o sistema, todos os três são totalmente divididos e gerenciados pelo padrão MVC (*Model, view, controller*).

2.2 MySQL

A utilização de um banco de dados tem como papel essencial em todas as áreas em que há tecnologia envolvida, e com eles teve um grande peso para o acréscimo do uso de computadores.

O MySQL é um *software* interativo que faz permitir a conexão com um servidor. Atualmente é um dos bancos de dados mais populares do mundo, foi desenvolvido e possui total suporte da Oracle Corporation. O MySQL possui um servidor com velocidade e de alta tecnologia de banco de dados SQL, também é multi-usuário e *multi threading* (MySQL Team, 2013).

Em meados dos anos 70 foi desenvolvida pela IBM (*International Business Machines)* uma linguagem de definição e manipulação, o SQL, na qualnão se pode dizer que é uma linguagem totalmente independente, pois para realizar o desenvolvimento com o banco de dados é preciso utilizar uma linguagem de programação e realizar comandos SQL para a manipulação dos dados.

Existe uma ferramenta chamada *phpMyAdmin*, ele é um software totalmente livre e foi desenvolvido em linguagem PHP (*Hypertext Prepocessor*), ele tem como objetivo dar conta de toda a administração do MySQL através via internet e possui várias operações tais como: (tabelas, gerenciamento de banco de dados, permissões, cria relações, entre outras) e todas essas operações podem ser feitas pela interface do próprio usuário. (PHPMYADMIN, 2013).

Linguagem gráfica bastante utilizada entre pessoas de desenvolvimento de *software* chama-se *Unified Modeling Language* (UML), ela é totalmente independente de qualquer linguagem de programação, o que garante para a pessoa que está desenvolvendo atribuir qualquer processo e tem como objetivo realizar a documentação de sistema, especificações, construções, entre outros. É uma linguagem de modelagem e passa ser a melhor forma de explicar cada passo que o sistema deve tomar para seu desenvolvimento.

2.3 Angular

Criada pelos desenvolvedores do google o angular é um *framework open source* que serve para construções de interfaces para aplicações com a utilização de *HTML, CSS* e *JavaScript.*

Pode-se destacar no *framework* muitos componentes, como: *templates,*

módulos, injeções de dependências, e muitas ferramentas para a automatização de certas tarefas além de executar testes unitários de uma aplicação e O

framework Angular possui suporte a Unit e a testes integrados

A primeira versão do angular foi realizada no ano de 2010, e tinha como nome AngularJS, ela foi totalmente reescrita e no ano de 2016 passou a ser chamada de Angular 2, atualmente, a versão mais atual é a 9, na qual foi realizada uma atualização para o funcionamento de TypeScript 3.6 e 3.7.

3 METODOLOGIA

Este formato de trabalho, teve seu começo com um levantamento de requisitos, onde foi analisado e estudado profundamente a necessidade de um software para as farmácias arquivarem todas suas documentações, dentro de um período de cinco anos às vendas relacionadas ao PFPB.

O digital receita permite que os usuários, realizem um cadastro para a utilização de algumas funcionalidades, após o cadastro o cliente necessita realizar o login para realizar suas digitalizações

Com a inserção dos levantamentos de requisitos, tanto funcionais como não funcionais, foi dada a largada para a inicialização do sistema.

3.1 Requisitos funcionais

Requisitos funcionais, são todas as atividades em que o usuário pode realizar e que são permitidas dentro do sistema, também pode ser todos os problemas que necessitam de ser atendidos e de alguma forma resolvido pelo sistema por meio de alguma função ou serviço.

São serviços na qual o sistema pode fornecer, a reação do sistema com devidas entradas, seu comportamento em momentos específicos. E em devidos

casos o requisito funcional pode também mostrar o que não se pode fazer no sistema.

Tabela 1 – Requisitos funcionais

IDENTIFICADOR	NOME	DESCRIÇÃO	PRIORIDADE
RF001	Cadastrar farmácia	O sistema permitirá que a farmácia seja cadastrada com CNPJ, *email,* nome, endereço, senha e telefone	Essencial
RF002	Login da Farmácia	O sistema permite a farmácia utilizar o CNPJ e senha para autenticação e validação	Essencial
RF003	Recuperação de senha	O sistema permite caso a farmácia não tenha mais o acesso, digitar o CNPJ e e-mail para uma nova senha.	Essencial
RF004	Digitalizar	O sistema permite que a farmácia realize as digitalizações e armazenamento de todas as documentações.	Essencial
RF005	Pesquisar	O sistema permite que a farmácia realize a busca daquela determinada documentação através do CPF que foi informado no ato da digitalização.	Essencial
RF006	Salvar digitalização	O sistema permite que a farmácia realize o salvamento de toda documentação.	Essencial

RF007	Procurar pelo computador	O sistema permite que a farmácia realize buscas de documentos que foram escaneados pelo computador.	Essencial
RF008	Recortar	O sistema permite que o cliente recorte para melhor visibilidade da documentação antes de salvar.	Importante

Fonte: Próprio autor (2021)

.

Conforme os requisitos apresentados na tabela 1 do sistema, será necessário que a farmácia realize o seu cadastro na aba de “Cadastrar farmácia”, após a realização de cadastro, a farmácia irá acessar o sistema na aba “*Login*” com o CNPJ cadastrado e senha.

3.2 Requisitos não funcionais

Requisitos não funcionais são todos aqueles que não interferem de forma direta ao desenvolver o sistema, portanto é um requisito que não possui regras de negócios e é necessário informar o que deve ser feito no sistema.

Tabela 2 – Requisitos não funcionais

IDENTIFICADOR	NOME	DESCRIÇÃO	PRIORIDADE
RNF001	Compatibilidade	O sistema será apenas para o sistema Windows	Essencial

RNF002	Acesso	O sistema permitirá que várias farmácias façam as digitalizações simultâneas	Importante
RNF003	Usabilidade	O sistema permite uma simples e fácil entendimento.	Importante
RNF004	Confiabilidade	Sistema simples com poucas taxas de falhas	Desejável
RNF005	Segurança	O sistema permite sigilo total das informações	Essencial
RNF006	Manutenção	O sistema permitirá reparos e evoluções futuras.	Essencial

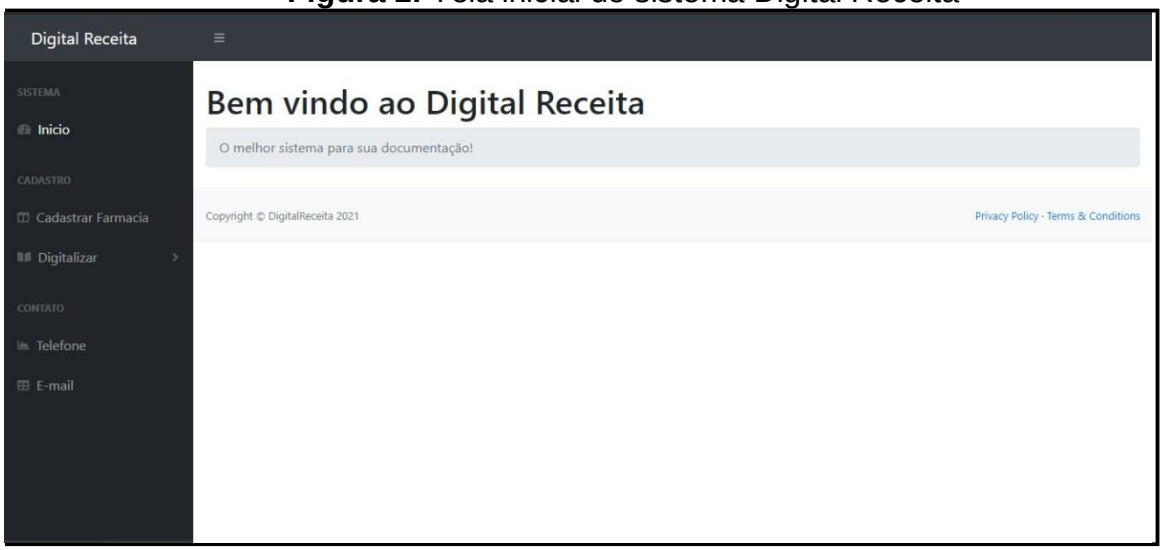
Fonte: Próprio autor (2021)

Conforme ilustrado na tabela 2 de requisitos não funcionais, o sistema irá funcionar apenas em sistemas *Windows,* possuirá sigilo total das informações dos usuários, assim como permitirá atualizações futuras para reparos e correções.

4 DESENVOLVIMENTO

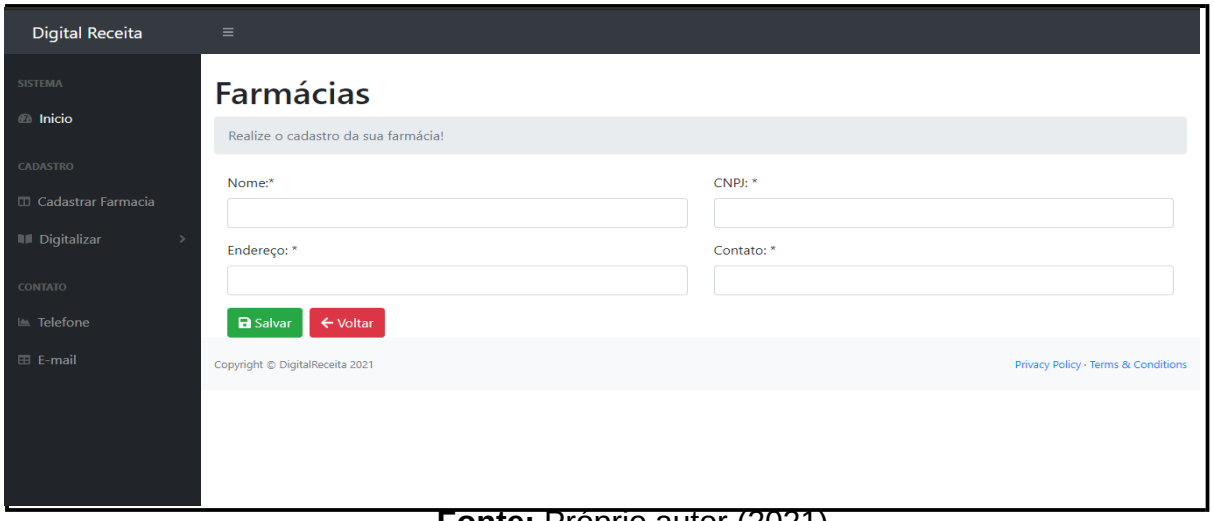
Seguindo com a etapa de desenvolvimento, e com algumas telas do sistema já criadas, será apresentado o procedimento para a fase apresentação do código. Neste momento serão inseridas algumas imagens com códigos e imagens do sistema que foram desenvolvidos em Java e a com o *framework open source Angular.*

Figura 1: Tela inicial do sistema Digital Receita



Fonte: Próprio Autor (2021)

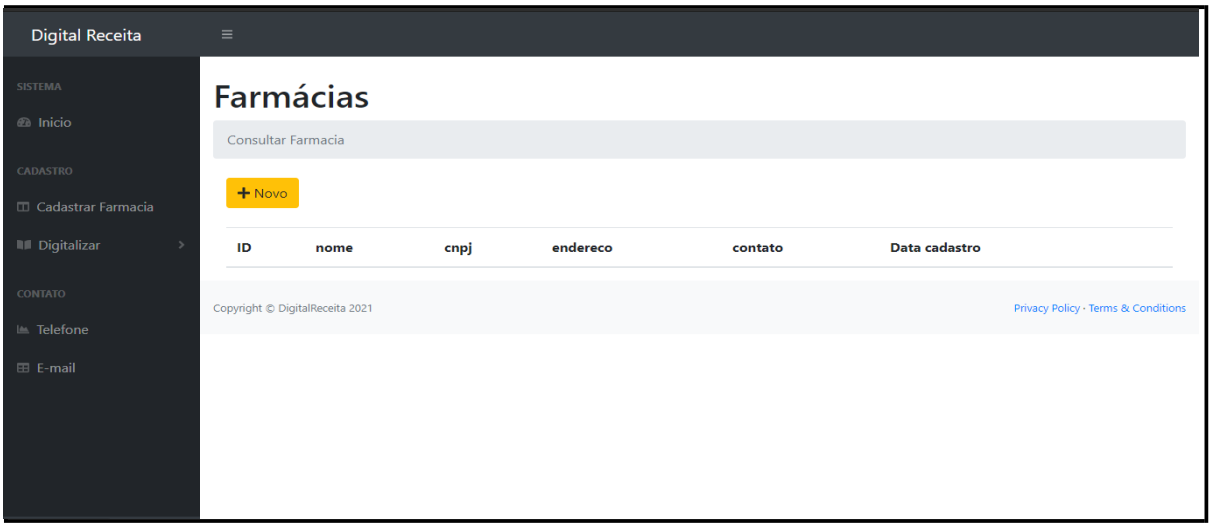
Conforme apresentado nesta primeira imagem, o sistema possui uma barra lateral a esquerda, onde são apresentadas todas as funcionalidades, alémtambém dos meios para contato e no centro da tela temos a saudação de boas- vindas para os usuários.



Fonte: Próprio autor (2021)

Sobre a segunda tela, ilustra uma listagem onde irão aparecer todas as farmácias que forem cadastradas, também haverá como realizar o cadastro de novas farmácias, ao clicar no botão “+ Novo”, onde poderá ser feito todo o preenchimento conforme apresentado no formulário. Na listagem das farmácias cadastrada, também haverá um botão ao lado direito para edição de dados.

Figura 3: Cadastramento de farmácias



Fonte: Próprio autor (2021)

Sobre a terceira tela, ilustra onde vai ser realizado o cadastramento das farmácias onde serão preenchidos todos os campos conforme o solicitado e apóso preenchimento, deverá ser clicado no botão de cor verde “Salvar” para a efetuação de cadastro, e caso queria voltar para a tela principal, deverá ser clicado no botão voltar.

Figura 4: Codificação da classe farmácia

```java
package com.digitalreceita.model;
import java.time.LocalDate;

@Entity
@Table(name= "farmacia")
public class Farmacia {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, name = "nome")
    private String nome;


    @Column(nullable = false, name = "cnpj")
    @NotNull
    private String cnpj;

    @Column(nullable = false, name = "endereco")
    private String endereco;

    @Column(nullable = false, name = "contato")
    private String contato;

    @Column(nullable = false, name = "data_cadastro")
    @JsonFormat(pattern = "dd/mm/yyyy")
    private LocalDate dataCadastro;
```

Fonte: Próprio autor (2021)

Na imagem da figura 4, exibe uma codificação em Java, *Spring boot*, ondefoi inserido o nome para cada tabela e a presença da anotação *nullable* para que nenhum campo apareça como nulo e possibilite que o banco de dados realize o

gerenciamento de todos os dados.

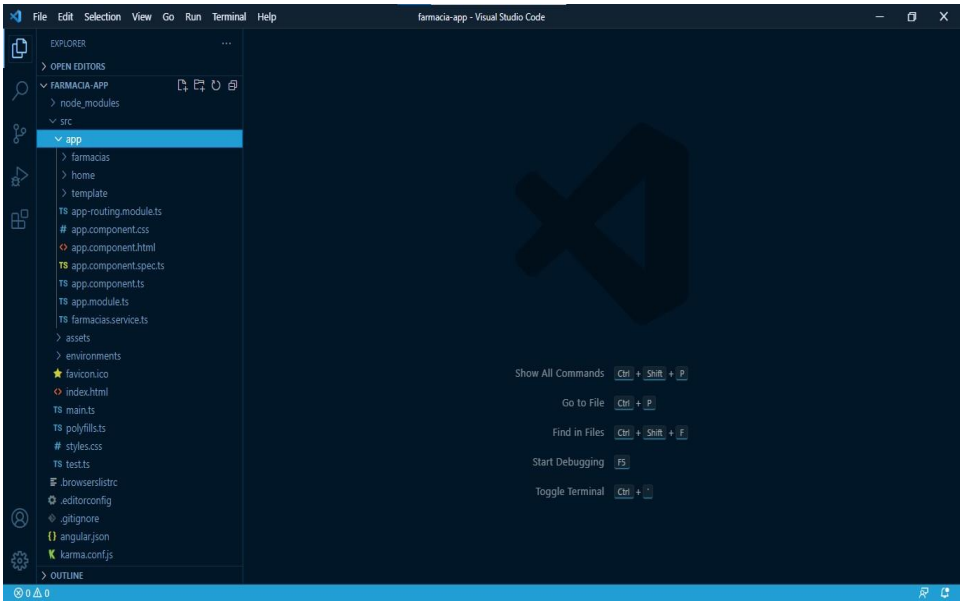
Figura 5: Controller da farmácia

```java
@RestController
@RequestMapping("/api/farmacias")
@CrossOrigin("http://localhost:4200")
public class FarmaciaController {

    @Autowired
    private FarmaciaService farmaciaService;

    @GetMapping()
    public List<Farmacia> farmaciaAll(){
        return farmaciaService.farmaciaListAll();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Farmacia> farmaciaById(@PathVariable Long id){
        return ResponseEntity.ok().body(farmaciaService.farmaciaById(id)
                .orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT_FOUND)));
    }

    @PostMapping()
    @ResponseStatus(HttpStatus.CREATED)
    public Farmacia salvar(@RequestBody @Validated Farmacia farmacia) {
        return farmaciaService.saveFarmacia(farmacia);
    }


    @PutMapping("/{id}")
    public ResponseEntity<Farmacia> updateFarmacia(@RequestBody Farmacia farmacia){
        return ResponseEntity.ok().body(farmaciaService.updateFarmacia(farmacia));
    }

    @DeleteMapping("/{id}")
    public void deleteFarmacia(@PathVariable Long id){
        farmaciaService.deleteFarmacia(id);
    }
```

Fonte: Próprio autor (2021)

Na imagem de figura número 5, apresenta a Classe *controller* do projeto, DigitalReceita, onde esta classe é totalmente responsável pelos processamentosde requisições, e respostas diretamente com o servidor.

Figura 6: Arquitetura e pacotes do sistema



Fonte: Próprio autor (2021)

Na figura 6, ilustra que houve a utilização a ferramenta *VS Code* para o desenvolvimento do *front-end,* aleém de ser uma ferramenta fácil de manuseio ela é bastante utilizada no mercado atualmente.

Figura 7: Formulário de cadastro de farmácia

```
<h1 class="mt-4">Farmácias</h1>
        <ol class="breadcrumb mb-4">
            <li class="breadcrumb-item active">Realize o cadastro da sua farmácia!</li>
        </ol>
        <div class="container">
            <form #farmaciaForm="ngForm" (ngSubmit)="onSubmit()">

                <div class="row">
                    <div class="col-md-12">
                        <div class="alert alert-success" role="alert" *ngIf="success == true">
                            Farmacia salva/atualizada com sucesso!
                        </div>
                    </div>
                </div>
                <div class="row" *ngIf="farmacia.id">
                    <div class="col-md-6">
                        <div class="form-group">
                            <label>ID:</label>
                            <input type="text"
                            [ngModel]="farmacia.id" class="form-control" disabled="true" />
                        </div>
                    </div>
                    <div class="col-md-6">
                        <div class="form-group">
                            <label>Data de cadastro:</label>
                            <input type="text" class="form-control" name="dataCadastro"
                            [ngModel]="farmacia.dataCadastro" disabled="true" />
                        </div>
                    </div>
                </div>
```

Fonte: Autor próprio (2021)

A figura 7 mostra um pouco da codificação do *front-end* com o angular, consta todo o formulário de cadastro de uma farmácia, onde todos os valores preenchidos nos campos, serão armazenados no bando de dados.

4 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo o desenvolvimento de um software, que possibilita que as farmácias credencias a farmácia popular realizem o cadastro e a digitalização de seus documentos.

O sistema proposto está em fase de conclusão, de forma que ainda será implementado novas formas de digitalizações mais detalhadas, novos modelos de cadastros e elaboração de um assistente para digitalização de toda documentação.

O digital receita teve sua primeira versão lançada e toda sua codificação

está armazenada no *github* no seguinte endereço: https://github.com/RafaelEwerton/digitalreceita, e está disponível para acesso do sistema por qualquer pessoa.

REFERÊNCIAS

BRASIL. Decreto nº 5.090, de 20 de maio de 2004. Regulamenta a Lei nº 10.858, de 13 de abril de 2004, e institui o programa "Farmácia Popular do Brasil", e dá outras providências. *Diário Oficial da União* 2004; 21 maio.

DEITEL, Harvey M.; DEITEL, Paul J.; FURMANKIEWICZ, Edson. **Java: como programar**. Pearson educacion, 2008.

HORSTMANN, Cay. **Conceitos de computação com Java**. Bookman Editora, 2009.

MELO, Ana Cristina. **Desenvolvendo aplicações com UML 2.2**. Brasport, 2004.

PAULA FILHO, W. P. (2009) “Engenharia de Software: Fundamentos, métodose padrões." 3. Ed. Rio de Janeiro: LTC.

PHPMYADMIN. (2013) “Documentação PhpMyAdmin.” Disponível em: Acesso em 04 mar. 2021.

SANTOS-PINTO, Cláudia Du Bocage; COSTA, Nilson do Rosário and OSORIO-DE-CASTRO, Claudia Garcia Serpa. **Quem acessa o Programa Farmácia Popular do Brasil? Aspectos do fornecimento público de medicamentos.** *Ciênc. saúde coletiva* [online]. 2011, vol.16, n.6, pp.2963-2973. ISSN 1413-8123. https://doi.org/10.1590/S1413- 81232011000600034.

ISBN 978-65582507 7-7