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

*Coutinho, Rafael Ewerton Feitosa de Oliveira

*Graduando em Sistema para internet

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 uma redução 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, 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 processo de 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 Java é 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 aproximação do 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 model ou 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.*

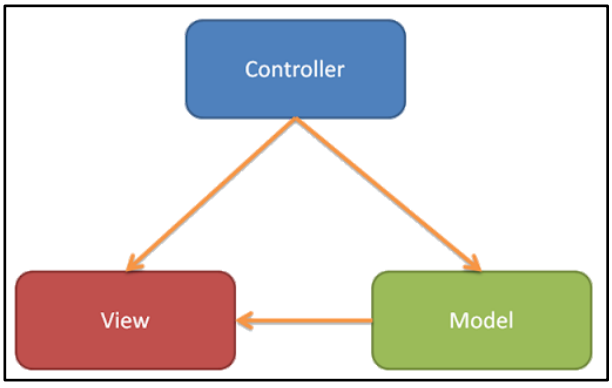


Figura 1 – Modelo MVC (Controller, Model, View)
Fonte: MVC, padrões de projeto, Tarcisio (2013)

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 qual não 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.

2.4 Bootstrap

É um *framework open-source* na qual é utilizado por desenvolvedores na construção de componentes de interfaces e seu *front-end* para todos os tipos de sites e aplicações que sejam web, também faz a utilização HTML*, CSS e JavaScript.*

2.5 Spring

O Spring é um *framework open-source* que foi criado por *Rod Jonhson*, é tem sua principal utilidade no Java e é totalmente baseada em *Servlet* e orientada em HTTP. O *Springboot* é totalmente baseado nos padrões de projeto e injeção de dependências, ele é um utilitário para a realização e configurações para aplicativos com rapidez e oferece toda estrutura pronta para a criação de aplicativos.

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

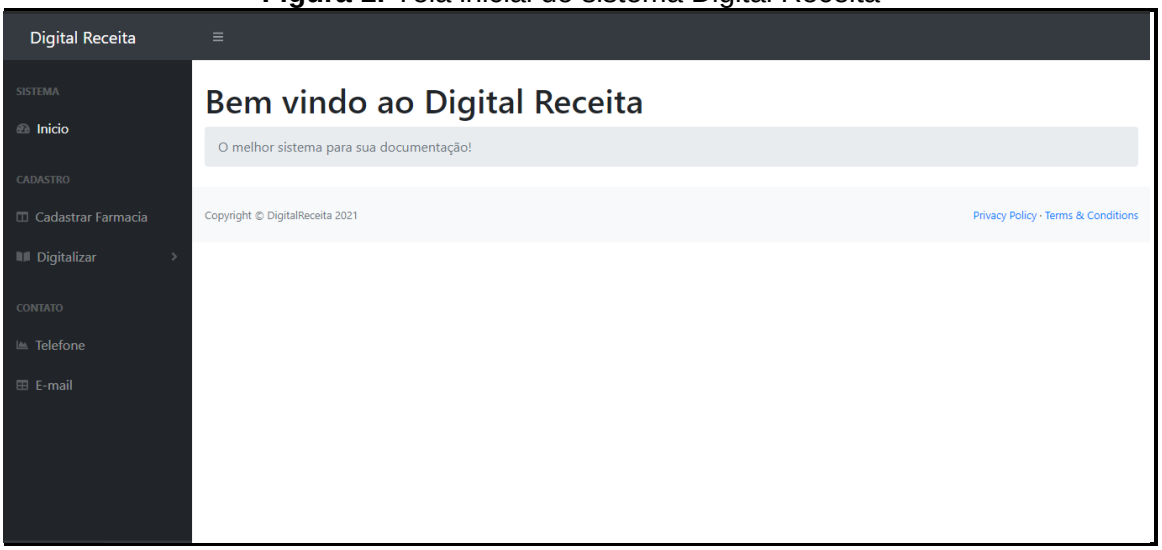
Fonte: Próprio autor (2021)

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.*

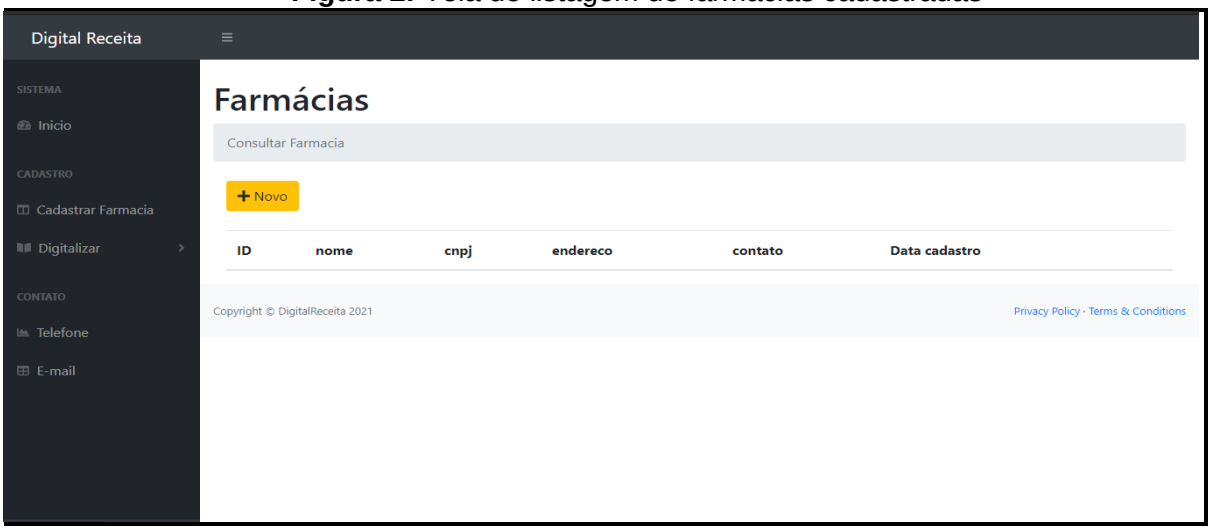
Figura 1: Tela inicial do sistema Digital Receita



Fonte: Próprio Autor (2021)

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

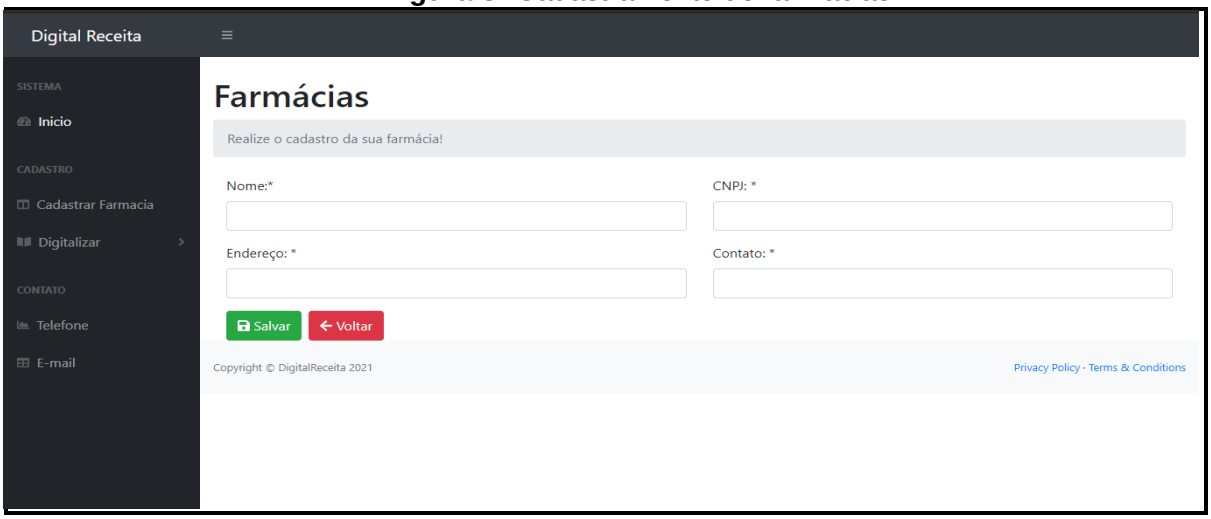
Figura 2: Tela de listagem de farmácias cadastradas



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 após o 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

```
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*, onde foi 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.

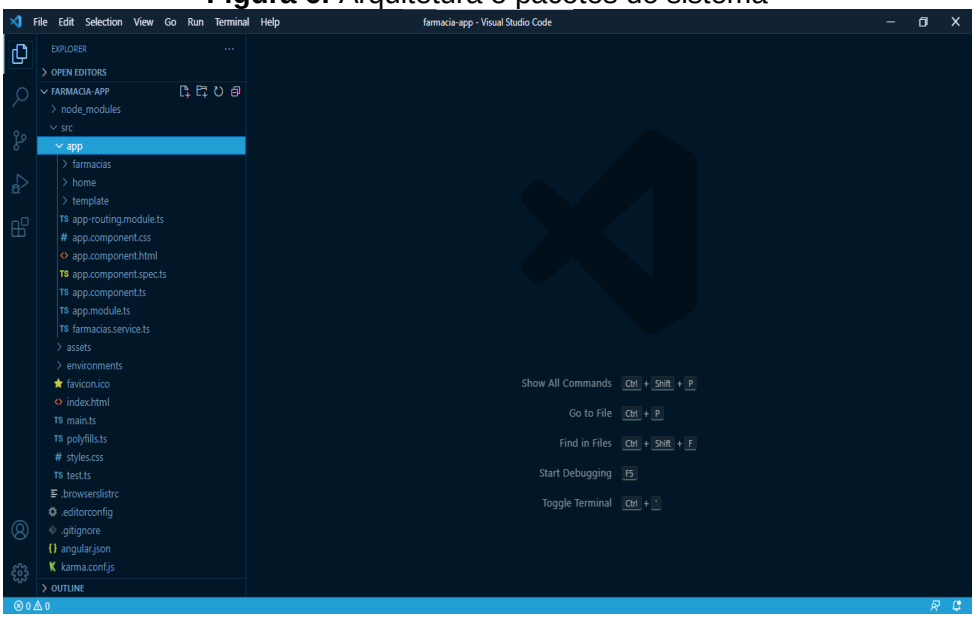
Figura 5: Controller da farmácia

```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 processamentos de 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, métodos e 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.