

# FILA ZERO

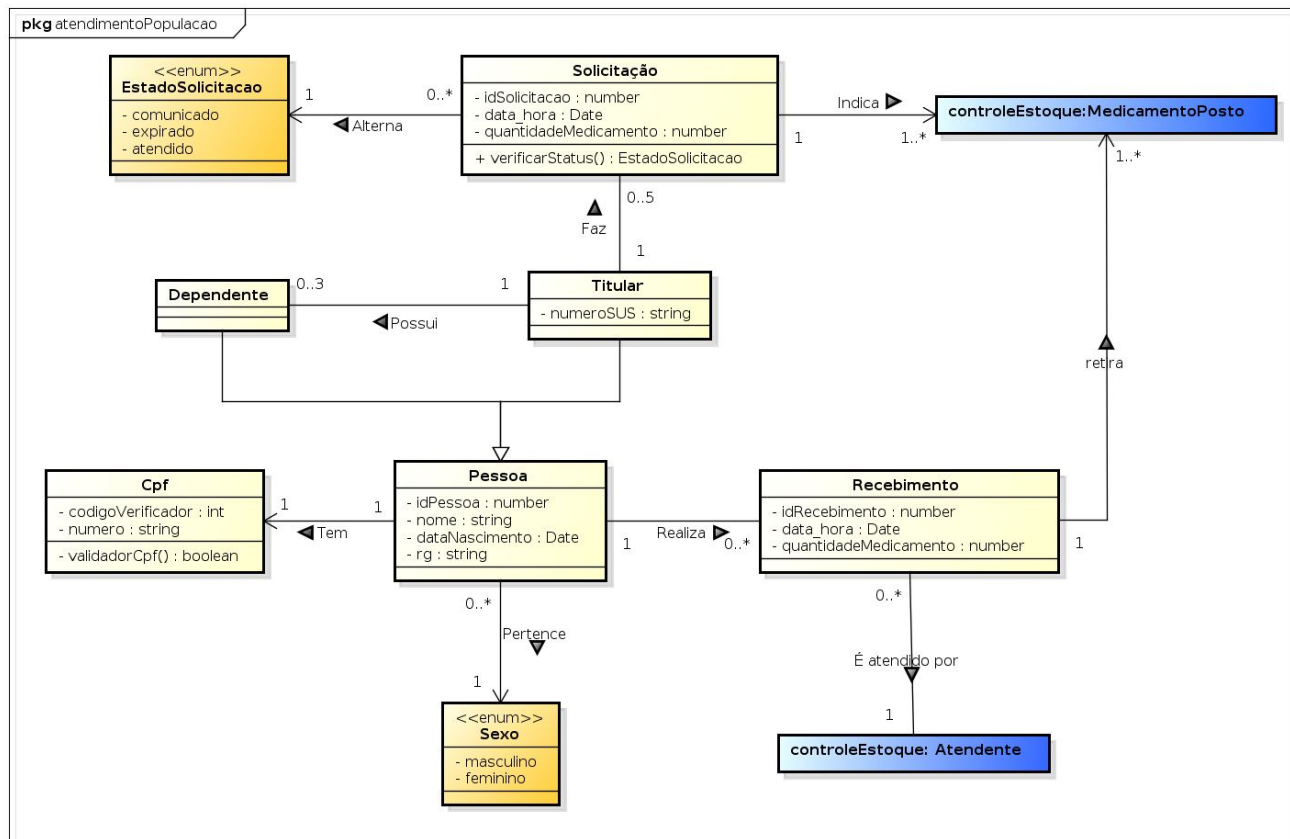
- **Harã Heique**
- **Jennifer Amaral**
- **Lucas Gomes**
- **Luiz Henrique**

# INTRODUÇÃO

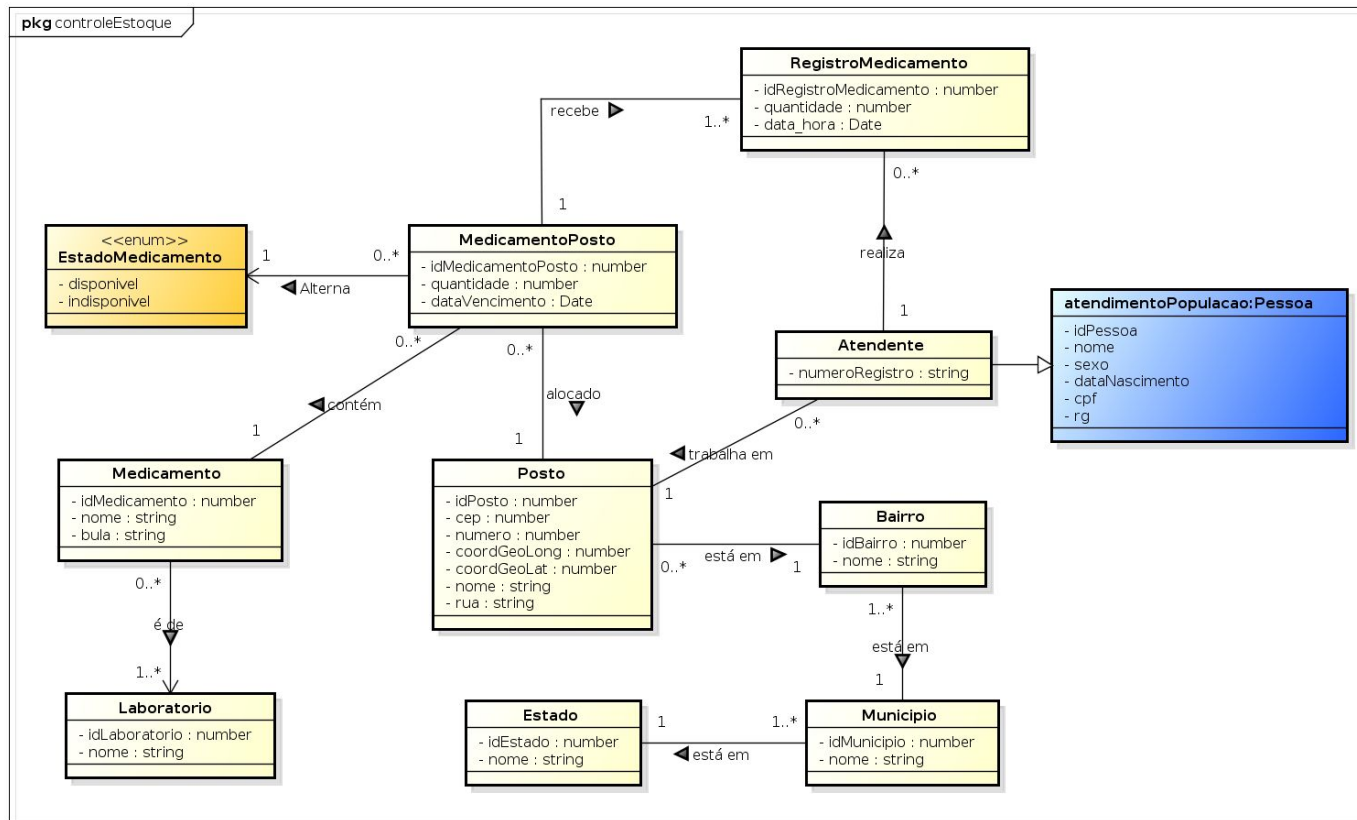
Os motivos da escolha do sistema proposto são da população não ter a necessidade de ir até o posto sem ter a certeza da obtenção do medicamento, o que consequentemente evitaria filas enormes para a solicitação.

# MODELAGEM DO SISTEMA

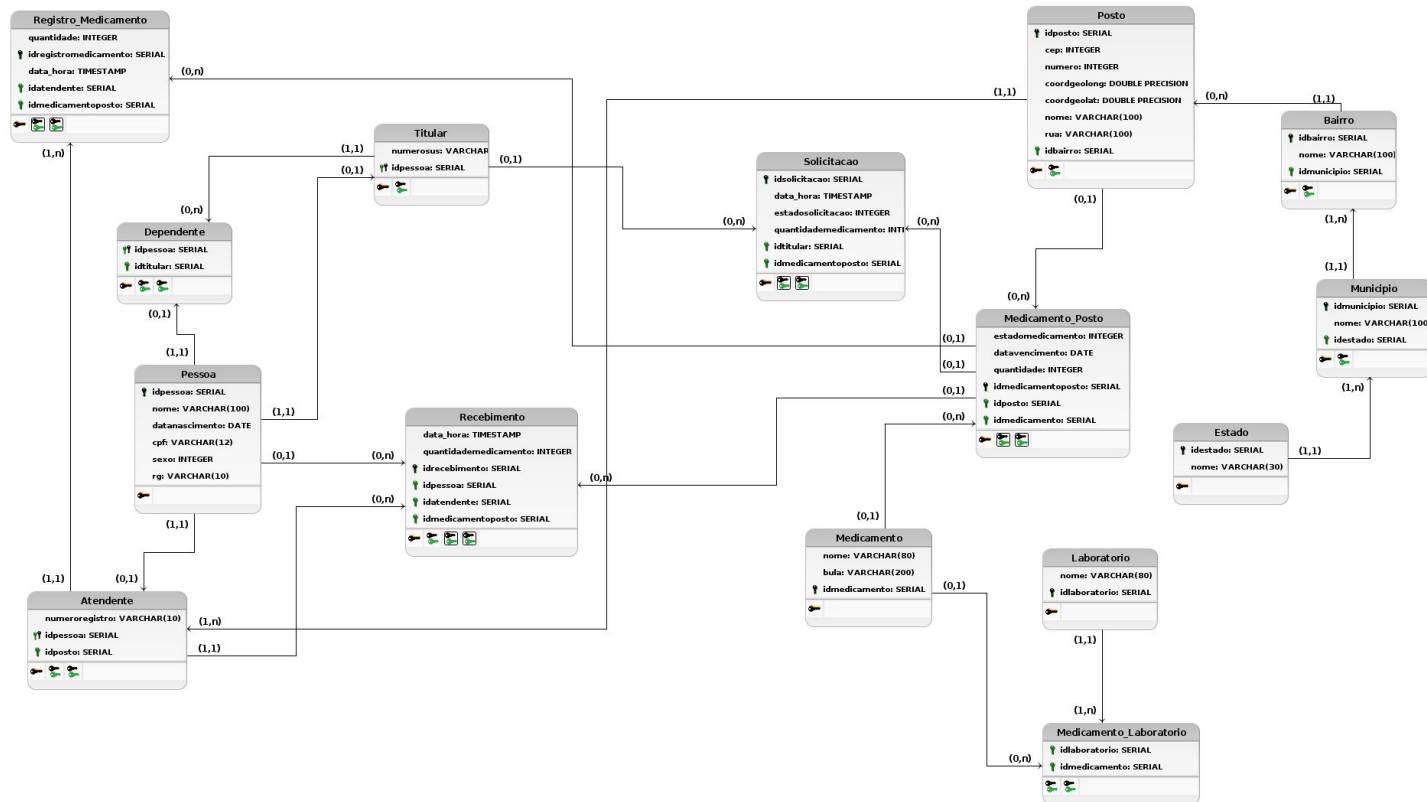
# DIAGRAMA DE CLASSES - ATENDIMENTO POPULAÇÃO



# DIAGRAMA DE CLASSES - CONTROLE DE ESTOQUE



# MODELO LÓGICO DO BANCO DE DADOS



FRONTEND

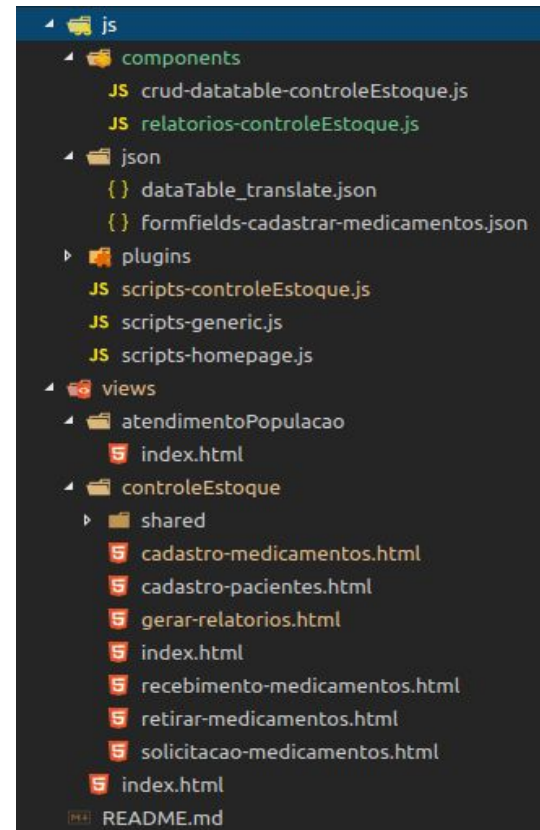
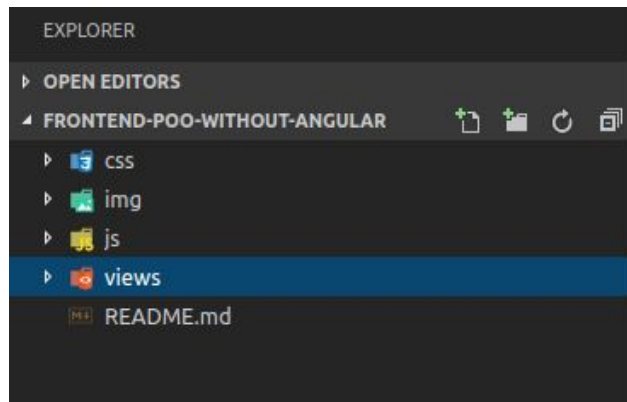
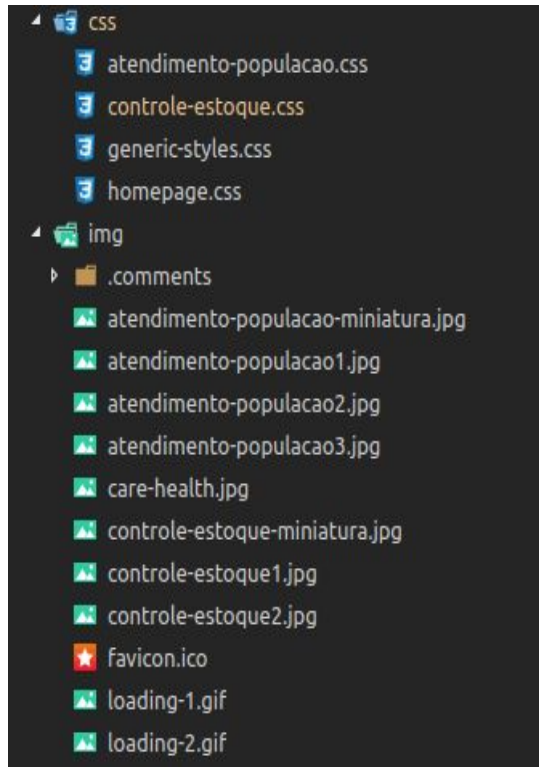
# O QUE ESTÁ SENDO E SERÁ UTILIZADO?

- Pilares do front-end:
  - HTML;
  - CSS;
  - JS;
- Bibliotecas, frameworks e plugins:
  - JQuery;
  - Bootstrap;
  - DataTables JS (plugin for jquery);
  - Highcharts;
  - FontAwesome web Application;
  - Toastr
  - Afins...





# ESTRUTURA DO CÓDIGO



# PÁGINAS DO PROTÓTIPO VS PÁGINAS DA APLICAÇÃO

Serão apresentadas comparações entre algumas das páginas realizadas na prototipação e as realizadas durante o desenvolvimento das views da aplicação.

Obs.: O subsistema de atendimento à população não foi criado devido ao tempo que não foi o suficiente para o desenvolvimento.

# ESCOLHA O SEU DESTINO



HOMEPAGE: REDIRECIONAMENTO PARA OS SUBSISTEMAS

## ESCOLHA O SEU DESTINO



**ATENDIMENTO A POPULAÇÃO**  
Voltado para população

**CONTROLE DE MEDICAMENTOS**





## ESCOLHA O SEU DESTINO



# PROTÓTIPO VS APLICAÇÃO

## LOGIN ATENDENTE

Farmacia Municipal de Fundao

← → ↻ 🔍 https://saude.fundao.gov.br/controle/login ☰

**Olá**  
Insira suas informações

Número de registro:

Registro Geral (RG):

[Entrar](#)

← → ↻ 🏠 ⓘ 127.0.0.1:8080/views/controleEstoque/index.html ☆ 🔴 🖱 ⋮

**Olá Funcionário**  
Insira suas informações

Número de Registro:

Registro Geral (RG):

[Entrar](#)

[Página Inicial](#)

# Olá Funcionário

## Insira suas informações

Número de Registro:

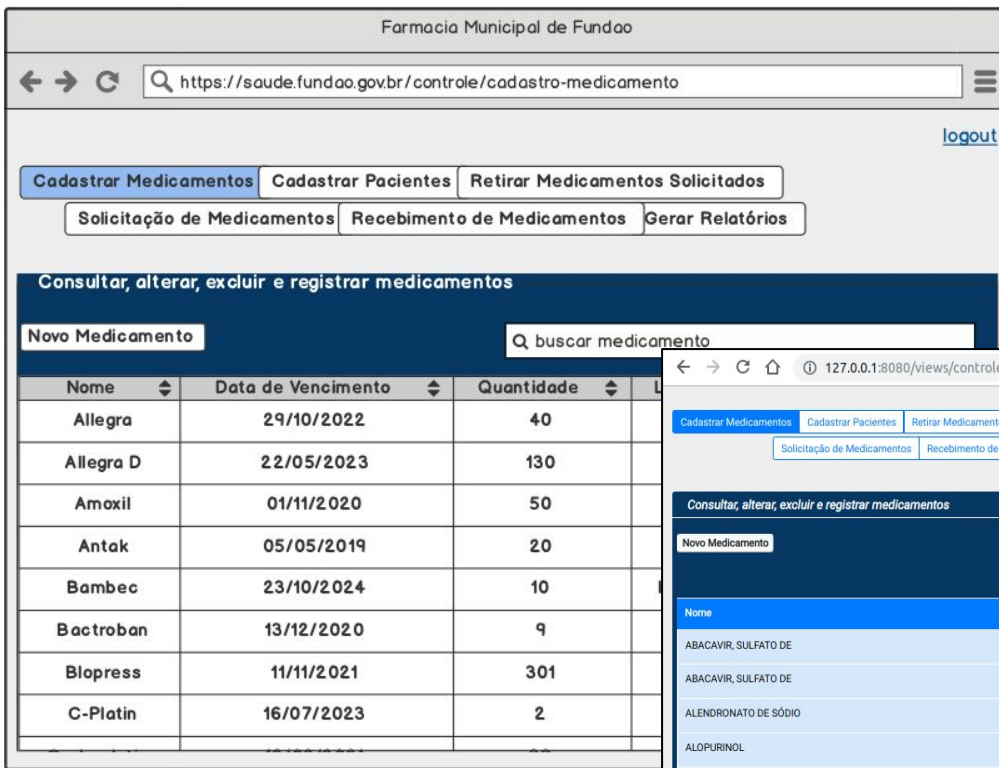
Registro Geral (RG):

Entrando

Entrar

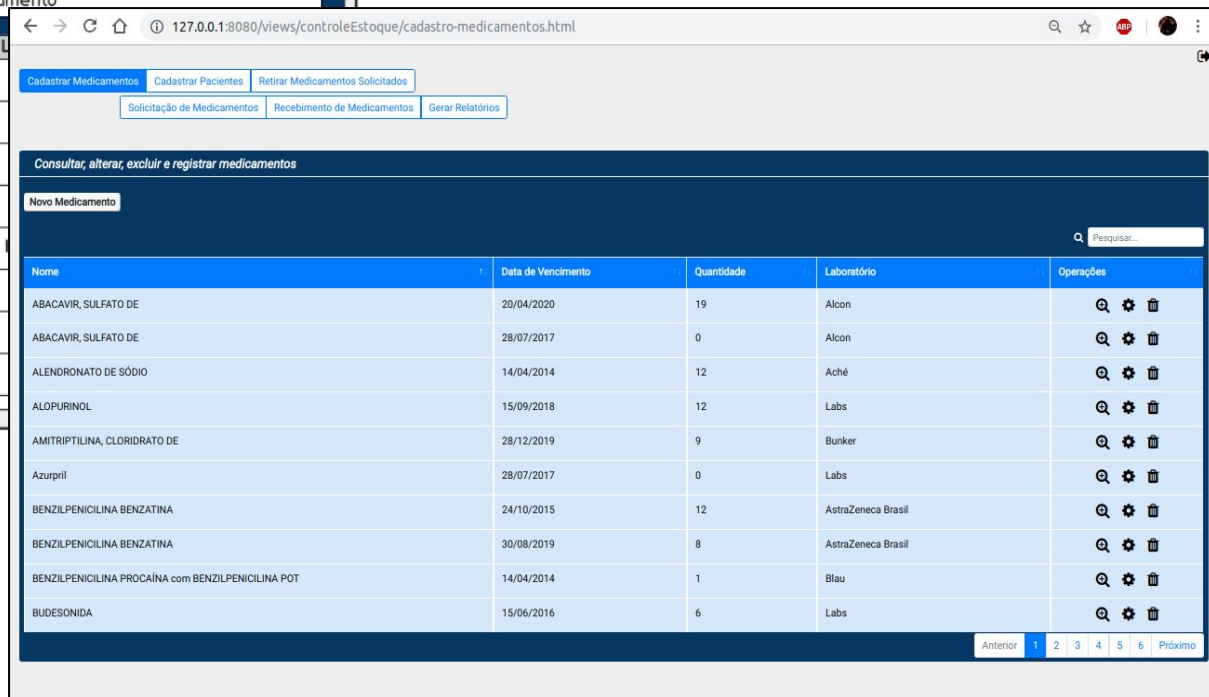
Página Inicial

LOADING PAGES

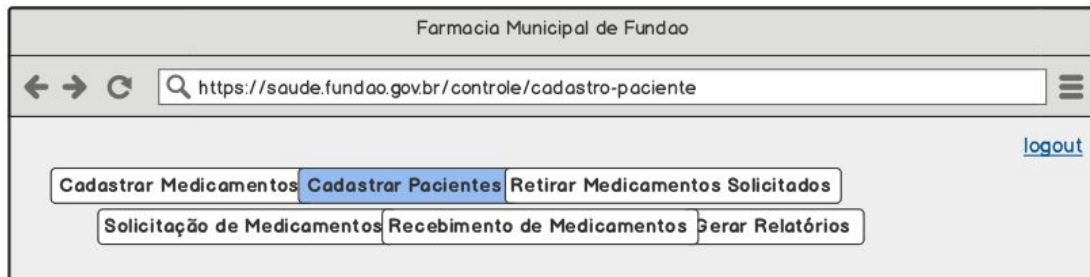


# PROTÓTIPO VS APLICAÇÃO

## CADASTRO DE MEDICAMENTOS







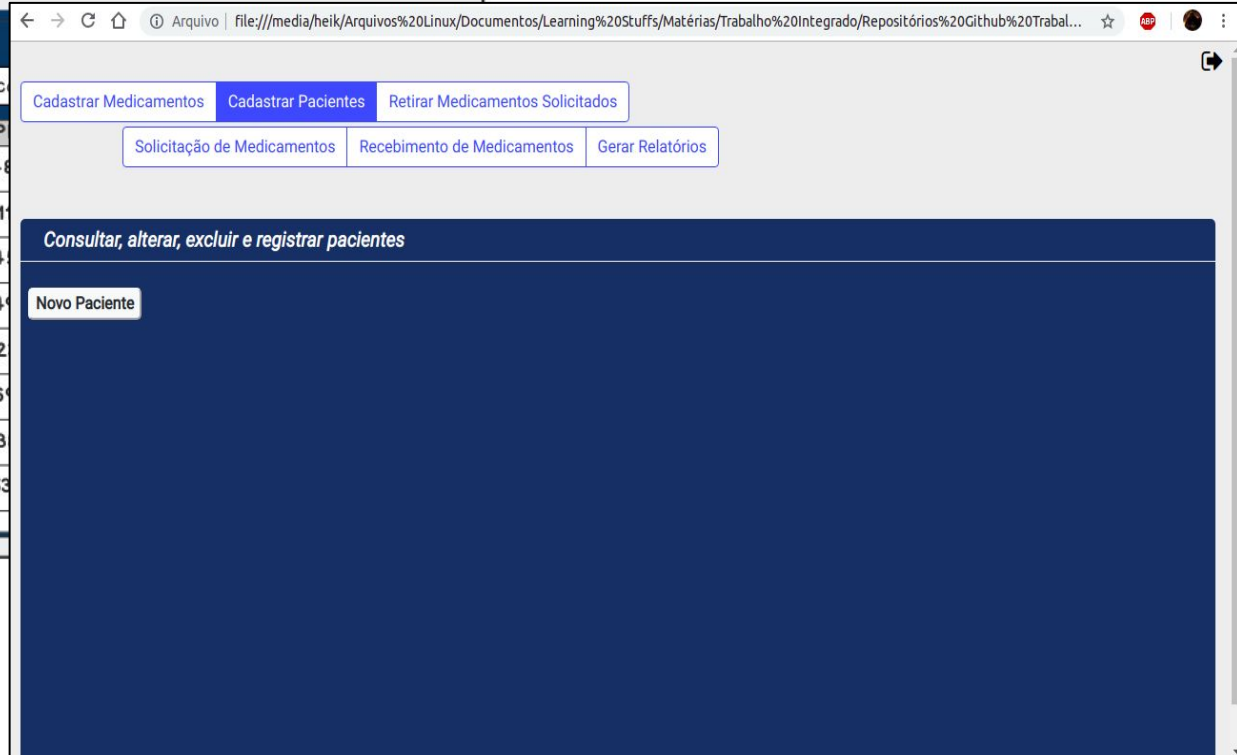
# PROTÓTIPO VS APLICAÇÃO

## CADASTRO DE PACIENTES

Consultar, alterar, excluir e registrar pacientes

Novo Paciente 🔍 buscar

Nome	N° SUS	CP
Amélia da Silva Carneiro	1385943847243254	123548
Arthur Limas de Souza	7685939854248700	66511
Bruna Keroline	4798470721580088	30364
Bruno Silva da Silva Silvano	1473348045255988	43134
Camila Zenterali	2062900893390178	22242
Enzo Dias Fernandes	5680070739239455	66596
Luis Rocha Pereira	9938729478046299	22473
Miguel Costa Azevedo	2709570443723351	115053



Farmacia Municipal de Fundao

← → ↻ 🔍 https://saude.fundao.gov.br/controle/cadastro-paciente ☰

Cadastrar Medicamentos Solicitação de

**Novo Paciente** ✕

Nome: Jadson Cody Sun

CPF: 12345676809

RG: 123456098

Número SUS: 12345678900987

Data de Nasci: 24/04/1994

Sexo: ☒ Feminino ☐ Masculino ☐

Dependente: 1 ▾

**Informações dos dependent**

Nome: Judson Cody Sun

CPF: 12345676809

Data de Nasci: 25/07/1996

Sexo: ☐ Feminino ☒ Masculino ☐

Cancelar

**Consultar, alterar, e**

Nome
Amélia da Silva Car
Arthur Limas de S
Bruna Keroline
Bruno Silva da Silva
Camila Zentera
Enzo Dias Fernan
Luis Rocha Pere
Miguel Costa Azev

**Novo Paciente**

# PROTÓTIPO VS APLICAÇÃO

## CADASTRO DE PACIENTES

← → ↻ 🏠 ⓘ 127.0.0.1:8080/views/controleEstoque/cadastro-pacientes.html ☆ 🔴 🔴 🔴 ⋮

Cadastrar Medicamentos Cadastrar Pacientes Retirar Medicamentos Solicitados

Solicitação de Medicamentos Rec

**Novo Paciente** ✕

Nome:

CPF:

RG:

Número SUS:

Data de Nascimento: 📅

Sexo: ☐ Feminino ☐ Masculino

Número de dependentes: 0 ▾

Fechar Cadastrar

*Consultar, alterar, excluir e registrar pacien*

**Novo Paciente**

# VALIDAÇÕES DE FORMULÁRIOS

127.0.0.1:8080/views/controleEstoque/cadastro-pacientes.html

Cadastrar Medicamentos Cadastrar Pacientes

Solicitação de Medicamentos Re

Consultar, alterar, excluir e registrar pacien

Novo Paciente

### Novo Paciente

Nome: Jukinha 123

CPF: 2312312454

RG:   
RG inválido

Número SUS: 321354354654234

Data de Nascimento: 11/12/1970

Sexo: ☒ Feminino ☒ Masculino

Número de dependentes: 2

#### Informações dos Dependentes

Dependente 1

Nome:   
Nome de dependente inválido

RG:   
RG inválido

Data de Nascimento: 18/12/1980

Sexo: ☒ Feminino ☒ Masculino

Dependente 2

Nome:   
Nome de dependente inválido

RG:   
RG inválido

Data de Nascimento:   
RG inválido

127.0.0.1:8080/views/controleEstoque/cadastro-pacientes.html

Cadastrar Medicamentos Cadastrar Pacientes

Solicitação de Medicamentos Re

Consultar, alterar, excluir e registrar pacien

Novo Paciente

### Informações dos Dependentes

Dependente 1

Nome:   
Nome de dependente inválido

RG:   
RG inválido

Data de Nascimento: 18/12/1980

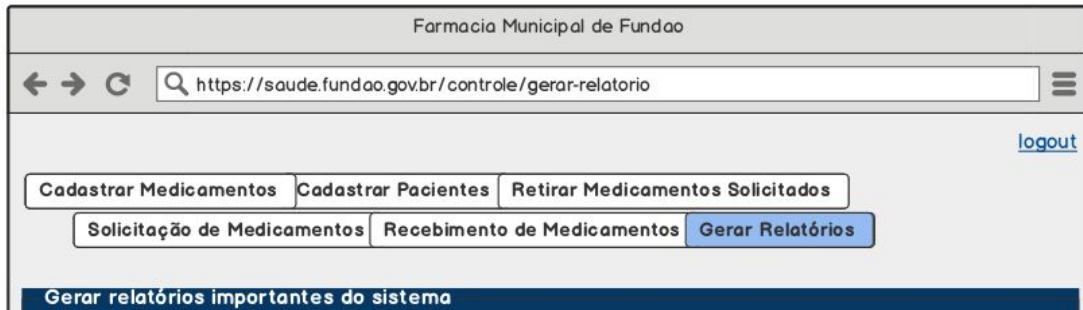
Sexo: ☒ Feminino ☒ Masculino

Dependente 2

Nome:   
Nome de dependente inválido

RG:   
RG inválido

Data de Nascimento:   
RG inválido

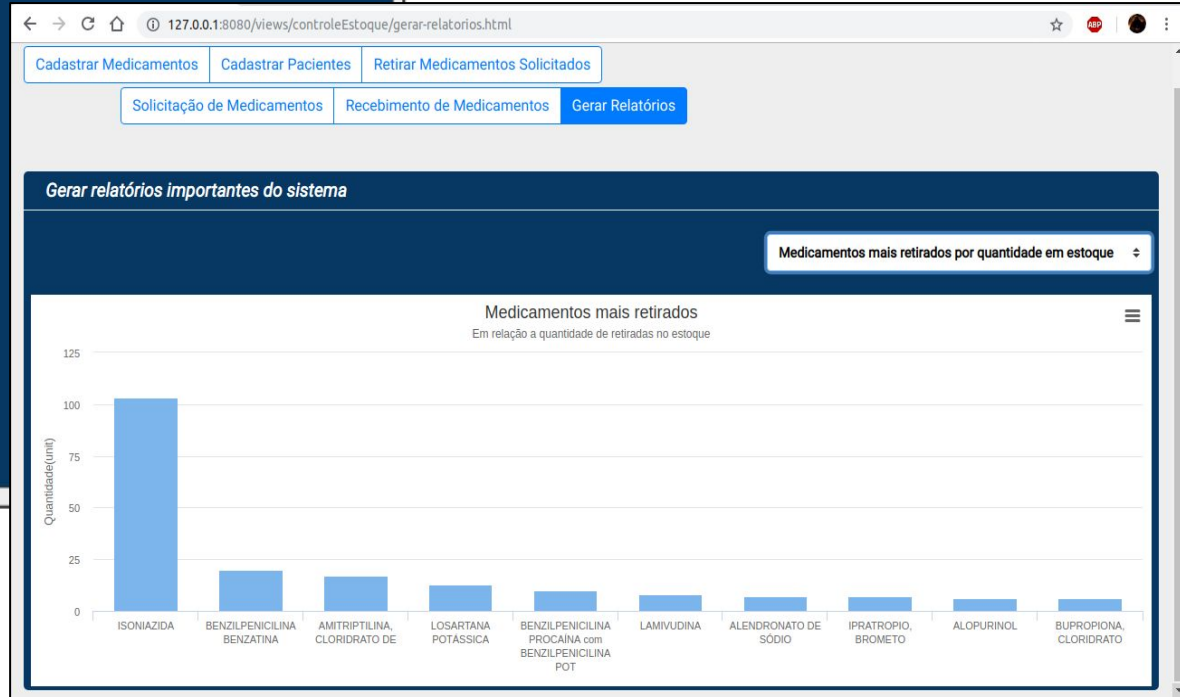


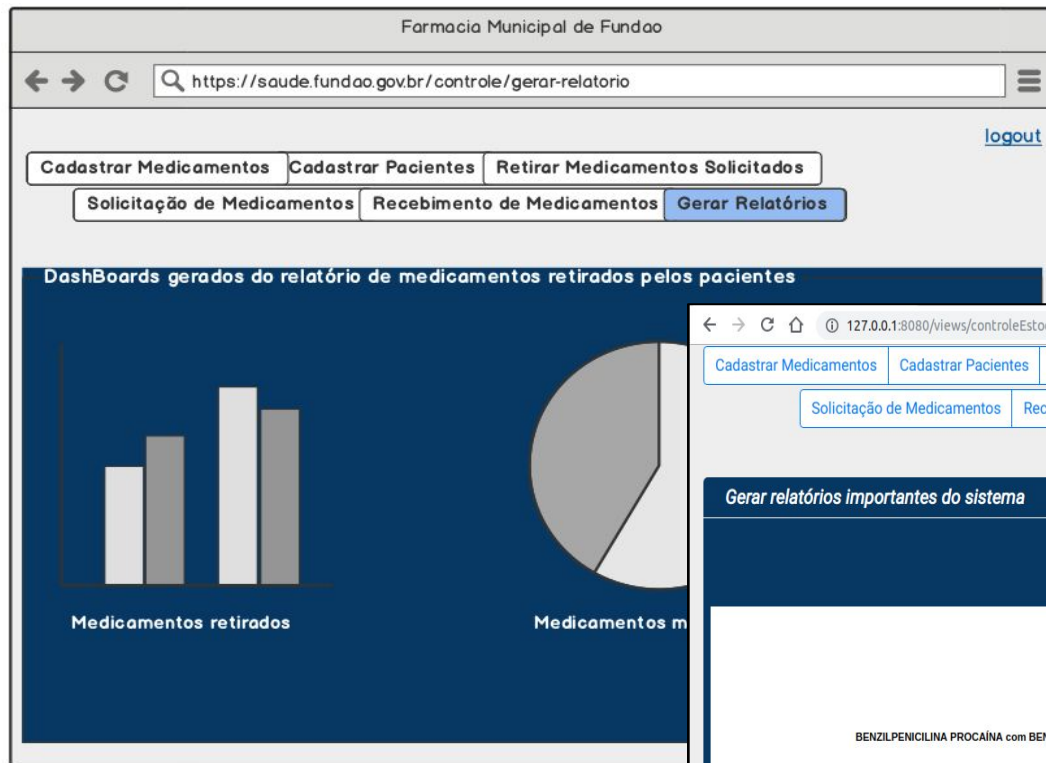
Selecione uma das opções abaixo:

- ☐ Medicamentos retirados pelos pacientes
- ☐ Medicamentos solicitados pelos pacientes
- ☐ Baixas em estoque dos Medicamentos

# PROTÓTIPO VS APLICAÇÃO

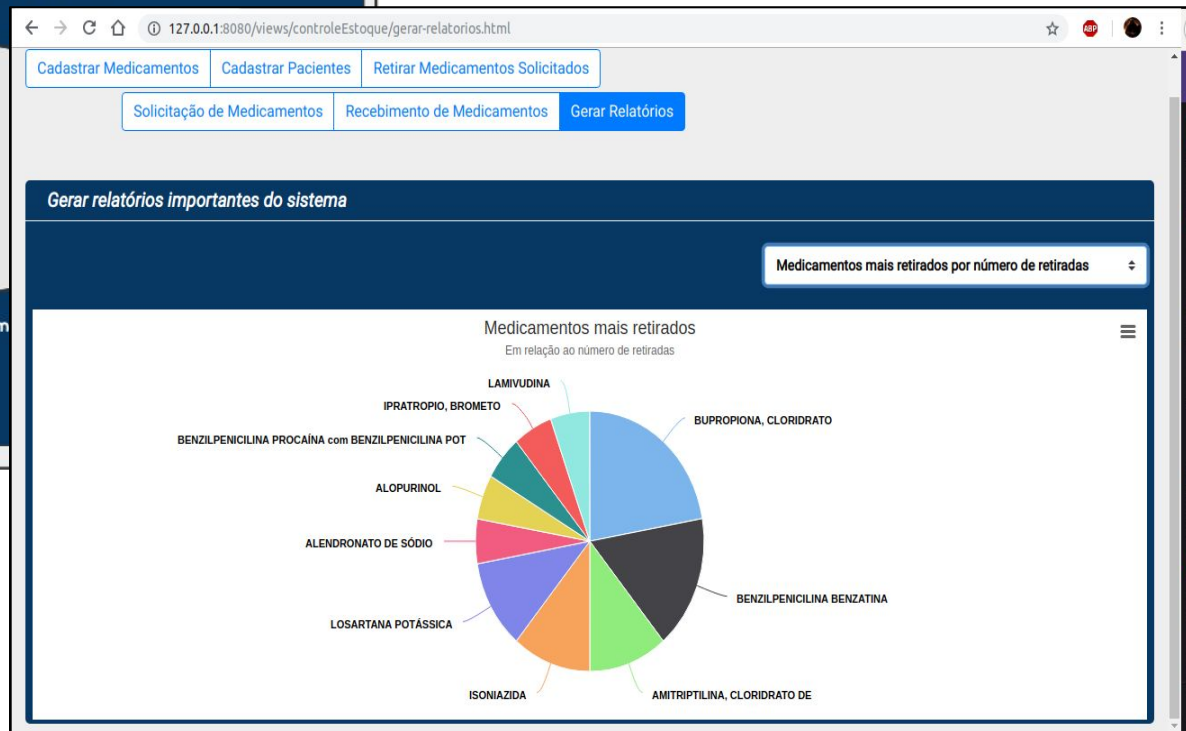
## GERAR RELATÓRIOS





# PROTÓTIPO VS APLICAÇÃO

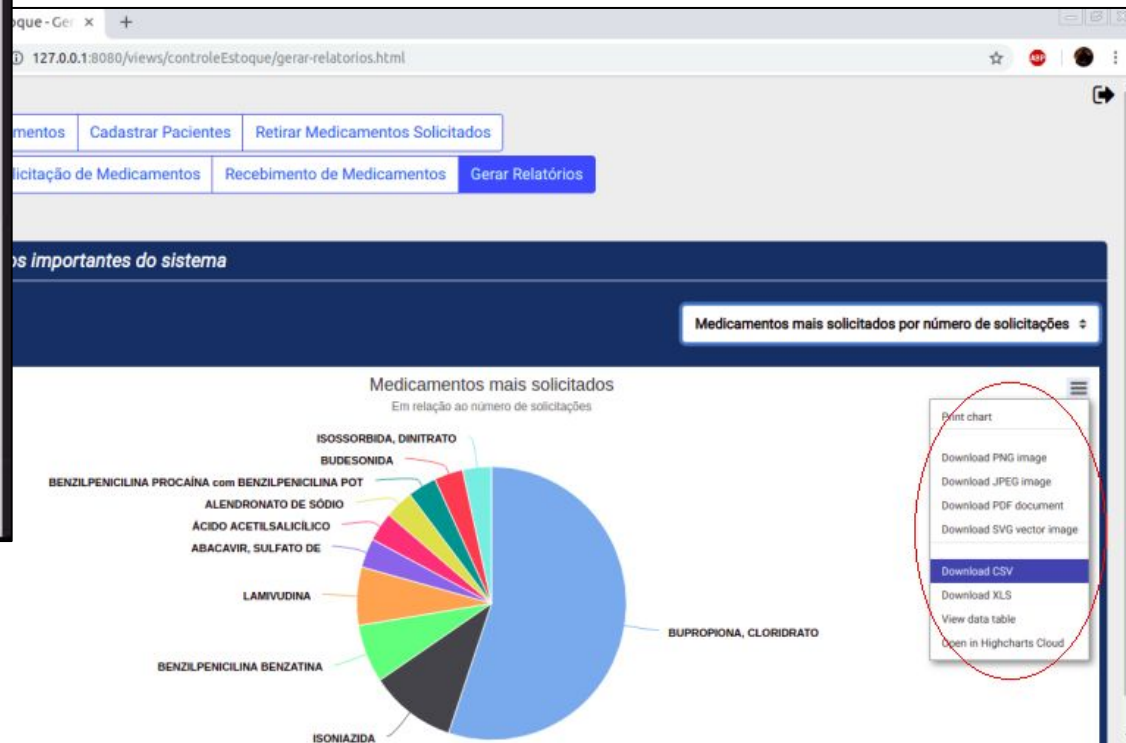
## GERAR RELATÓRIOS



# TOASTS E EXPORTAÇÃO DE INFORMAÇÕES DO RELATÓRIOS

Pesquisar...	
Laboratório	Operações
Alcon	  
Alcon	  
Aché	  
Labs	  
Bunker	  

✓ Cadastrado com sucesso!



# UM POUCO DE RESPONSIVIDADE...

iPad 768 x 1024 50% Online

Cadastrar Medicamentos Cadastrar Pacientes Retirar Medicamentos Solicitados

Solicitação de Medicamentos Recebimento de Medicamentos Gerar Relatórios

Consultar, alterar, excluir e registrar medicamentos

Novo Medicamento

Pesquisar...

Nome	Data de Vencimento	Quantidade	Laboratório	Operações
ABACAVIR, SULFATO DE	20/04/2020	19	Alcon	
ABACAVIR, SULFATO DE	28/07/2017	0	Alcon	
ALENDRONATO DE SÓDIO	14/04/2014	12	Aché	
ALOPURINOL	15/09/2018	12	Labs	
AMITRIPTILINA, CLORIDRATO DE	28/12/2019	9	Bunker	
Azurpril	28/07/2017	0	Labs	
BENZILPENICILINA BENZATINA	24/10/2015	12	AstraZeneca Brasil	
BENZILPENICILINA BENZATINA	30/08/2019	8	AstraZeneca Brasil	

Pixel 2 411 x 731 79% Online

Cadastrar Medicamentos

Cadastrar Pacientes

Retirar Medicamentos Solicitados

Solicitação de Medicamentos

Recebimento de Medicamentos

Gerar Relatórios

Consultar, alterar, excluir e registrar medicamentos

Novo Medicamento

Pesquisar...

Nome	Data de Vencimento	Quantidade
ABACAVIR, SULFATO DE	20/04/2020	19
ABACAVIR, SULFATO DE	28/07/2017	0

Pixel 2 411 x 731 79% Online

ALENDRONATO DE SÓDIO	14/04/2014	12
ALOPURINOL	15/09/2018	12
AMITRIPTILINA, CLORIDRATO DE	28/12/2019	9
Azurpril	28/07/2017	0
BENZILPENICILINA BENZATINA	24/10/2015	12
BENZILPENICILINA BENZATINA	30/08/2019	8
BENZILPENICILINA PROCAÍNA com BENZILPENICILINA POT	14/04/2014	1
BUDESONIDA	15/06/2016	6

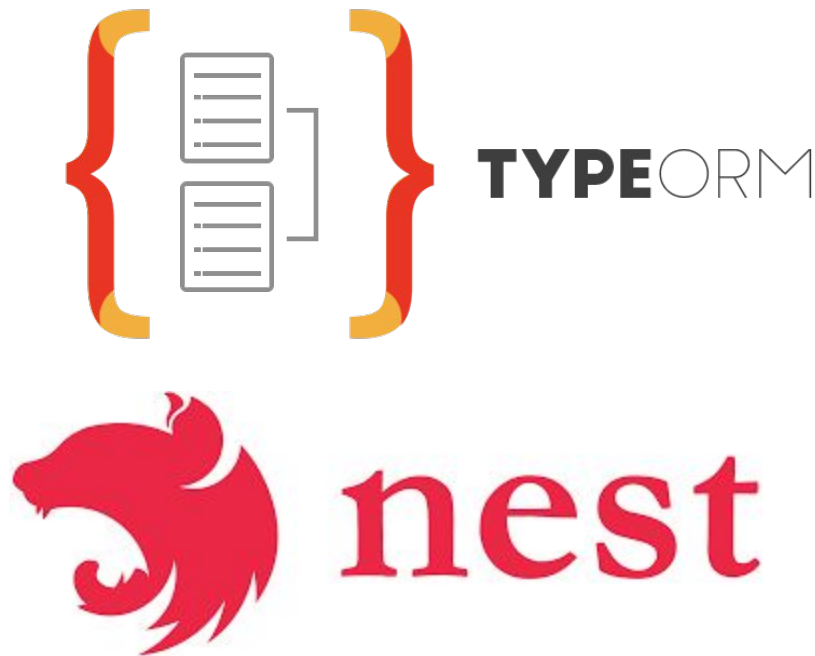
Anterior 1 2 3 4 5 6 Próximo

BACKEND

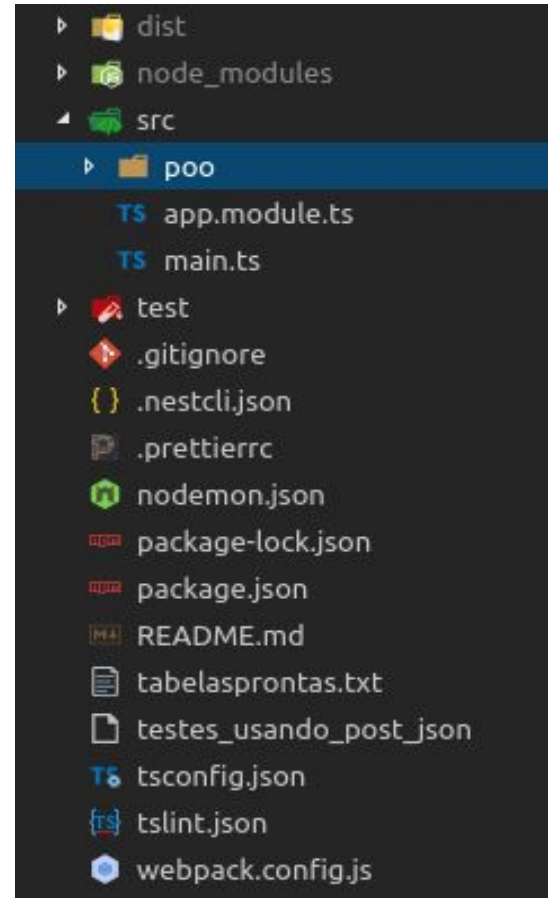
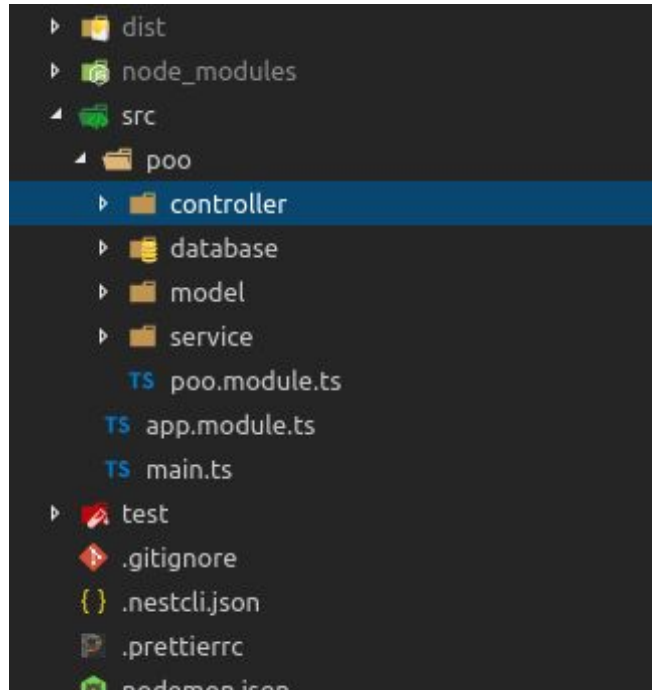


# O QUE ESTÁ SENDO E SERÁ UTILIZADO?

- Linguagem Utilizada
  - Typescript;
- Framework utilizado:
  - Nest JS;
- ORM utilizado:
  - TypeORM;



# ESTRUTURA DO CÓDIGO



# ENTIDADE - ATENDENTE

```
@Entity() You, 2 months ago • criando tabelas do BD
export class Atendente extends BaseEntity {
  @PrimaryColumn()
  idpessoa: number;

  @Column({ nullable: false, length: 10, unique: true })
  numeroregistro: string;

  //#####
  //##### RELAÇÕES #####
  //#####

  @OneToMany(type => Recebimento, recebimento => recebimento.atendente)
  recebimento: Recebimento[];

  @ManyToOne(type => Posto, posto => posto.atendente, { cascade: true, onDelete: "CASCADE" })
  @JoinColumn({ name: 'idposto' })
  posto: Posto;

  @OneToMany(
    type => RegistroMedicamento,
    registroMedicamento => registroMedicamento.atendente,
  )
  registroMedicamento: RegistroMedicamento[];

  @ManyToOne(type => Pessoa, pessoa => pessoa.atendente, {
    eager: true, cascade: true, onDelete: "CASCADE"
  })
  @JoinColumn({ name: 'idpessoa' })
  pessoa: Pessoa;
```

# CONTROLLER - ATENDENTE

```
@Get('/atendente')
async readAll(@Res() res) {
  try {
    let atendente: Atendente[] = await this.atendenteService.readAll();
    if (atendente !== undefined) {
      res.status(HttpStatus.OK).send(atendente);
    } else {
      res
        .status(HttpStatus.NOT_FOUND)
        .send('Nenhum atendente encontrado na busca');
    }
  } catch (err) {
    res.status(HttpStatus.BAD_GATEWAY).send(err.message);
  }
}

@Post('/atendente/create')
async Create(@Res() res, @Body() body) {
  try {
    let atendente = await this.atendenteService.Create(body);
    if (atendente !== undefined) {
      res.status(HttpStatus.OK).send(atendente);
    } else {
      res
        .status(HttpStatus.NOT_FOUND)
        .send('Erro ao salvar o atendente');
    }
  } catch (err) {
    res.status(HttpStatus.BAD_GATEWAY).send(err.message);
  }
}
```

# SERVICE - ATENDENTE

```
async readAll(): Promise<Atendente[]> {  
  return Atendente.find();  
}  
  
async readOne(id: number): Promise<Atendente> {  
  return Atendente.findOne({ idpessoa: id });  
}  
  
async Create(body: any): Promise<Atendente> {  
  let atendente = new Atendente();  
  try {  
    atendente.numeroregistro = body.numeroregistro;  
    atendente.idpessoa = body.idpessoa;  
    atendente.posto = body.idposto;  
    return await Atendente.save(atendente);  
  } catch (err) {  
    throw new Error(  
      `Erro ao salvar atendente\n Erro: ${err.name}\n Mensagem: ${  
        err.message  
      } \n Os parametros estao certos?`,  
    );  
  }  
}
```

# CONEXÃO COM O BANCO DE DADOS

```
export const databaseProviders = [  
  {  
    provide: 'DbConnectionToken',  
    useFactory: async () => await createConnection({  
      type: 'postgres',  
      host: 'elmer.db.elephantsql.com',  
      port: 5432,  
      username: 'aiexkamd',  
      password: 'QpfBdkd4AT2chCRjdjGbPPoSXdctwU9y',  
      database: 'aiexkamd',  
      entities: [  
        {  
          __dirname + '/../**/*.entity{.ts,.js}',  
        },  
      ],  
      synchronize: true,  
    }),  
  },  
];
```

You, 24 days ago • fazendo crud

# PADRÕES DE PROJETO



```
@Entity()
export class Atendente extends BaseEntity {
  @PrimaryColumn()
  idpessoa: number;
```

```
static getConnection(connection?: Connection): Connection {
  /**
   * Gets current entity's Repository.
   */
  static getRepository<T extends BaseEntity>(this: ObjectType<T>): Repository<T>;
  /**
   * Returns object that is managed by this repository.
   * If this repository manages entity from schema,
   * then it returns a name of that schema instead.
   */
```

```
/**
 * Finds entities that match given options.
 */
static find<T extends BaseEntity>(this: ObjectType<T>, options?: FindManyOptions<T>): Promise<T[]>;
```

```
@Injectable()
export class AtendenteService {

  async buscaTodosAtendentes(): Promise<Atendente[]> {
    return await Atendente.find();
  }
}
```

# PADRÃO ACTIVE RECORD



# PADRÃO MÉTODO FÁBRICA

```
async function bootstrap() {  
  const app = await NestFactory.create(AppModule);  
  
  await app.listen(parseInt(process.env.PORT) || 3001);  
}  
bootstrap();
```

```
export declare class NestFactoryStatic {  
  private readonly logger;  
  /**  
   * Creates an instance of the NestApplication  
   * @returns {Promise}  
   */  
  create(module: any): Promise<INestApplication & INestExpressApplication>;  
}
```

```
@Module({  
  providers: [...modelProvider, ...modelService],  
  controllers: [...modelController],  
})  
export class PooModule {}
```

# PADRÃO INJEÇÃO DE DEPENDÊNCIA

```
@Injectable()
export class AtendenteService {
  You, 23 days ago • criando tabelas do BD
  async buscaTodosAtendentes(): Promise<Atendente[]> {
    return await Atendente.find();
  }
}
```

```
/**
 * Defines the injectable class. This class can inject dependencies through constructor.
 * Those dependencies have to belong to the same module.
 */
export declare function Injectable(): ClassDecorator;
```

```
declare type ClassDecorator = <TFunction extends Function>(target: TFunction) => TFunction | void;
```

```
    Creates a new function.
    * @param args A list of arguments the function accepts.
    */
    new(...args: string[]): Function;
    (...args: string[]): Function;
    readonly prototype: Function;
}
```

```
declare const Function: FunctionConstructor;
```

# PADRÃO PAGINAÇÃO

```
@Injectable()
export class SolicitacaoService {
  async readAll(pag: number): Promise<Solicitacao[] | any> {
    return Solicitacao.createQueryBuilder('solicitacao')
      .select(
        'solicitacao.idsolicitacao, solicitacao.data_hora, solicitacao.quantidademedicamento,
      )
      .innerJoin('solicitacao.titular', 'titular')
      .innerJoin('solicitacao.medicamentoPosto', 'medicamentoPosto')
      .innerJoin('titular.pessoa', 'pessoa')
      .innerJoin('titular.depedente', 'depedente')
      .innerJoin('medicamentoPosto.posto', 'posto')
      .innerJoin('medicamentoPosto.medicamento', 'medicamento')
      .innerJoin('medicamento.laboratorio', 'laboratorio')
      .offset(pag * 10)
      .limit(10)
      .getRawMany();
  }
}
```

```
@Get('/solicitacao/page/:id')
async readAll(@Res() res, @Param() id) {
  try {
    let solicitacao: Solicitacao[] = await this.solicitacaoService.readAll(id.id);
  } catch (err) {
    // ...
  }
}
```

# PADRÃO CACHE

```
@Controller()
@UseInterceptors(CacheInterceptor)
export class SolicitacaoController {
  constructor(private readonly solicitacaoService: SolicitacaoService) {}

  @Get(['/solicitacao/:id']) You, a month ago • solicitacao em progresso
  async readOne(@Res() res, @Param() id) {
```

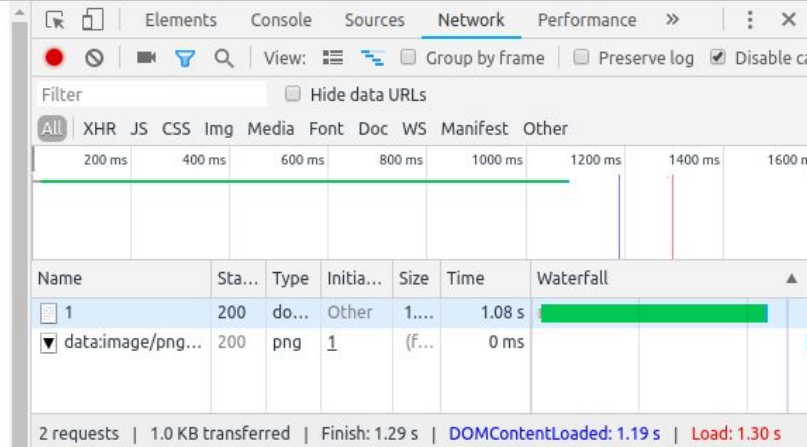
```
@Module({
  imports: [CacheModule.register({
    ttl: 10, You, a day ago • colocando cache
    max: 10,
  })],
  providers: [...modelProvider, ...modelService],
  controllers: [...modelController],
})
export class PooModule {}
```

# COMPARAÇÃO DE RESULTADO EM CACHE

```
{
  "idsolicitacao": 1,
  "data_hora": "2017-07-29T03:00:00.000Z",
  "quantidademedicamento": 1,
  "estadosolicitacao": 1,
  "titular": {
    "idpessoa": 50,
    "numerosus": 10001,
    "depedente": [],
    "pessoa": {
      "idpessoa": 50,
      "nome": "Vilma Gato",
      "datanascimento": "1956-01-31T03:00:00.000Z",
      "cpf": "50",
      "sexo": 0,
      "rg": "50"
    }
  }
}
```

Raw

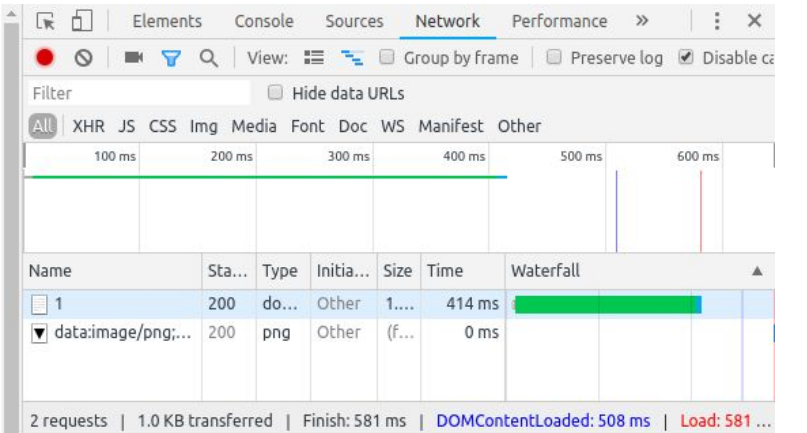
Parsed



```
{
  "idsolicitacao": 1,
  "data_hora": "2017-07-29T03:00:00.000Z",
  "quantidademedicamento": 1,
  "estadosolicitacao": 1,
  "titular": {
    "idpessoa": 50,
    "numerosus": 10001,
    "depedente": [],
    "pessoa": {
      "idpessoa": 50,
      "nome": "Vilma Gato",
      "datanascimento": "1956-01-31T03:00:00.000Z",
      "cpf": "50",
      "sexo": 0,
      "rg": "50"
    }
  }
}
```

Raw

Parsed



IMPLANTAÇÃO

# IMPLANTAÇÃO

A API se encontra atualmente no heroku.

O Heroku é uma plataforma de cloud que oferece "Platform as a Service", ou seja, ele permite que você hospede suas aplicações em um ambiente facilmente escalável e com suporte a várias tecnologias. Ele tem um plano free, que é indicado para testes, e opções pagas com mais funcionalidades e suporte.

# ARQUITETURA



Request

Response

Request

Request

Response

heroku

Response



ElephantSQL



# RESPOSTA DO HEROKU

<https://poo2.herokuapp.com/atendente>

```
[
  {
    "idpessoa": 963,
    "numeroregistro": "4",
    "pessoa": {
      "idpessoa": 963,
      "nome": "Diva Pinheiro Carneiro",
      "datanascimento": "1988-06-28T00:00:00.000Z",
      "cpf": "963",
      "sexo": 0,
      "rg": "963"
    }
  },
  {
    "idpessoa": 66,
    "numeroregistro": "5",
    "pessoa": {
      "idpessoa": 66,
      "nome": "Elisa Cerejeira",
      "datanascimento": "1971-01-08T00:00:00.000Z",
      "cpf": "66",
      "sexo": 0,
      "rg": "66"
    }
  },
]
```

# GITHUB'S E DIÁRIO DE BORDO

- <https://github.com/lukasg18/Topicos-Trabalho-BD2>
- <https://github.com/lukasg18/poo2-backend>
- <https://github.com/HaraHeique/frontend-P00-without-angular>
- [https://docs.google.com/document/d/1gVortYoQ8UpUQGYBodMUtm81sVFyRQc1K5\\_01l0pu-o/edit](https://docs.google.com/document/d/1gVortYoQ8UpUQGYBodMUtm81sVFyRQc1K5_01l0pu-o/edit)