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

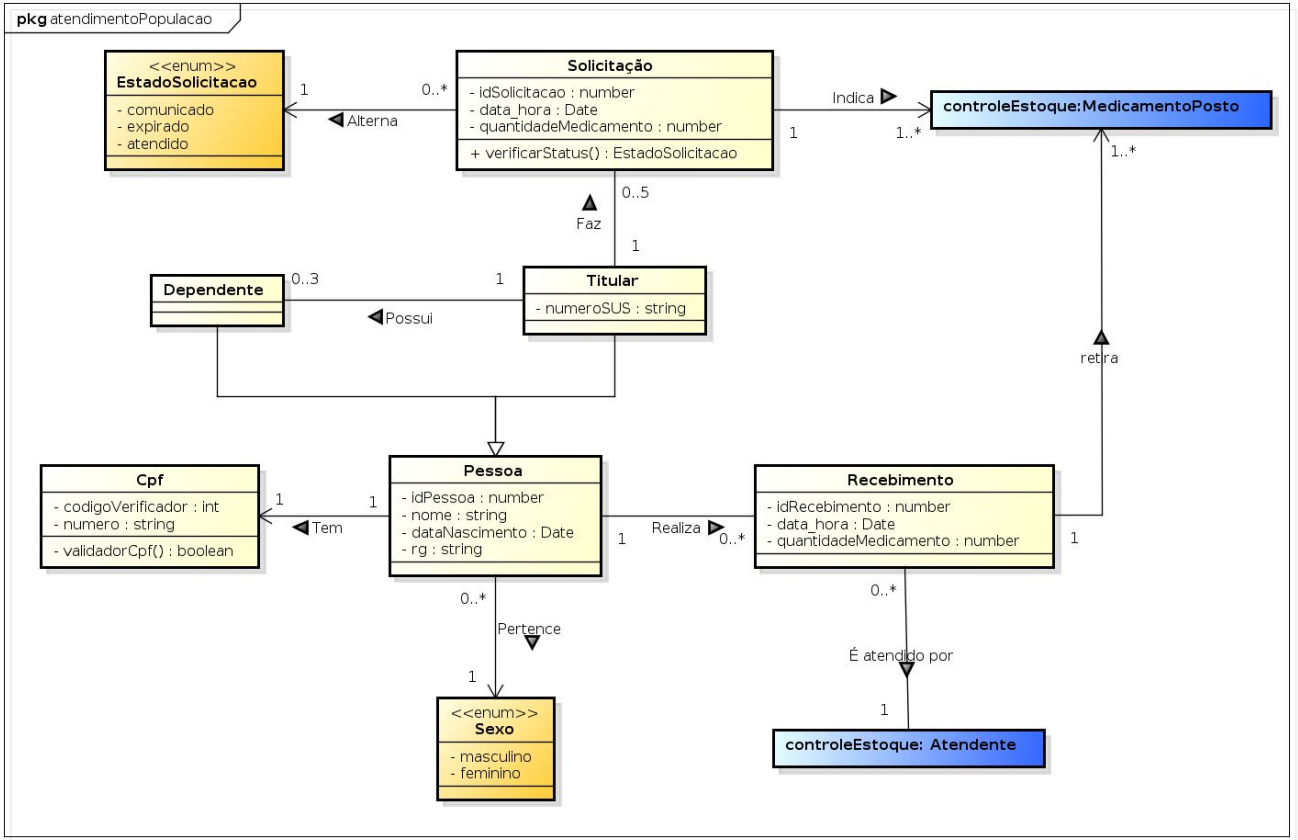
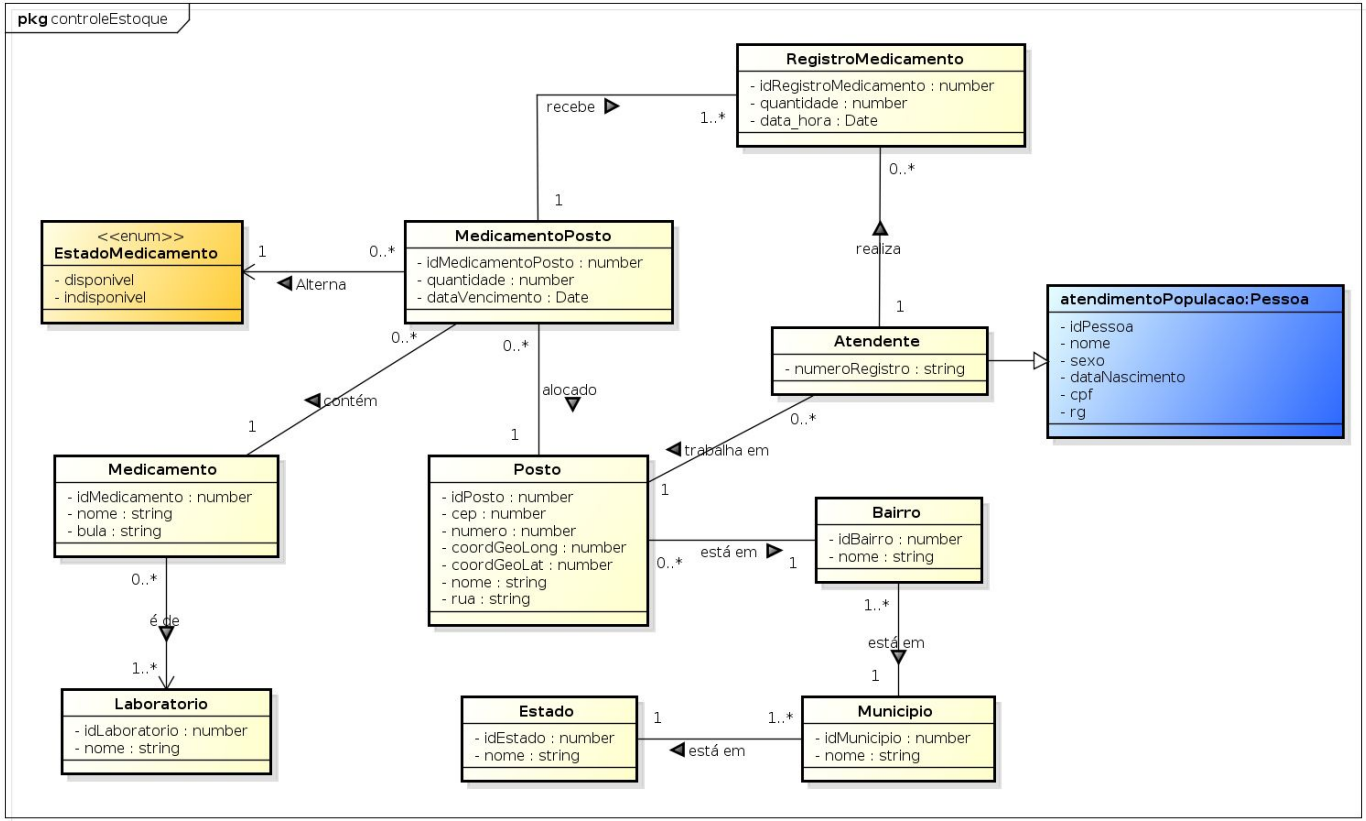
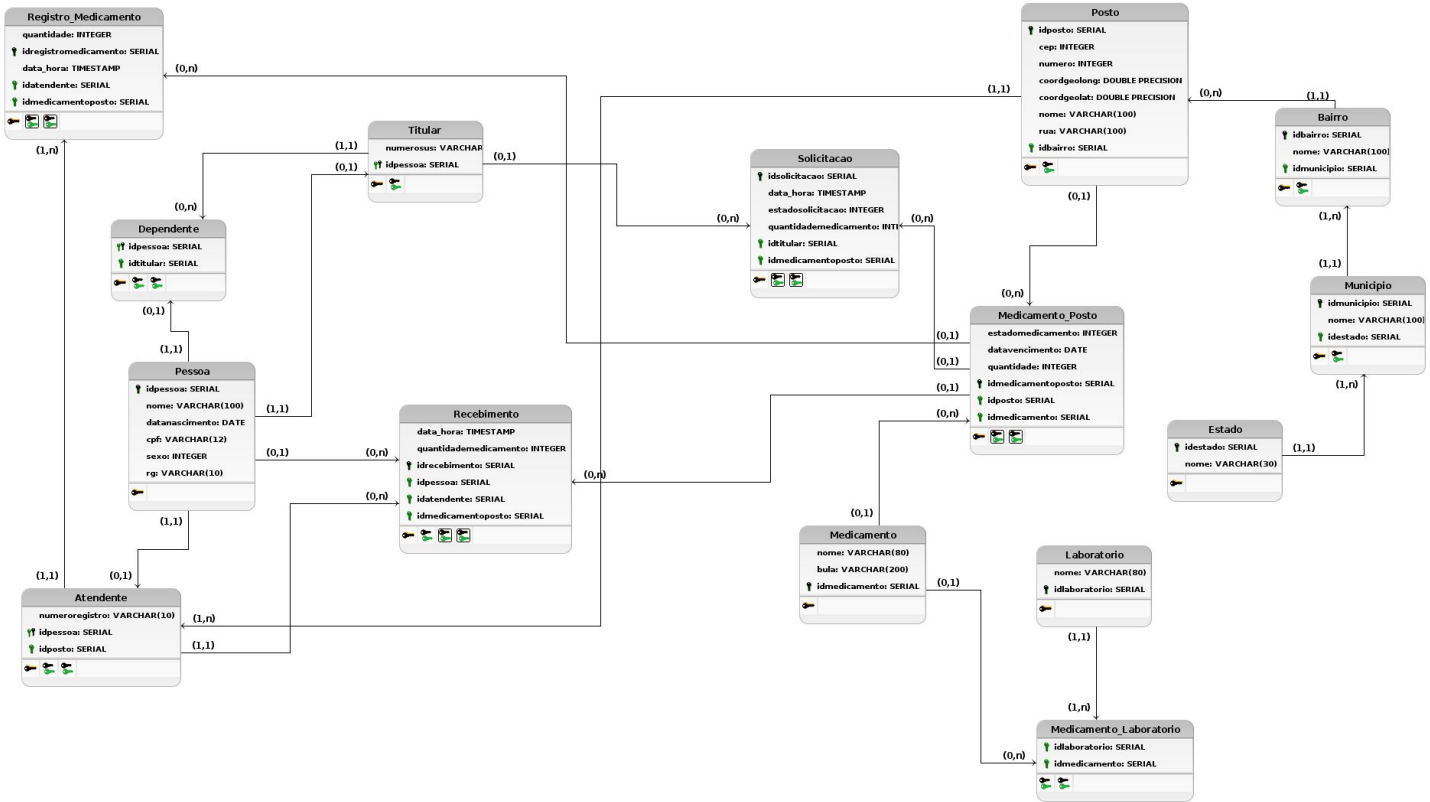


Diagrama de classes - controle de estoque



Modelo lógico do banco de dados

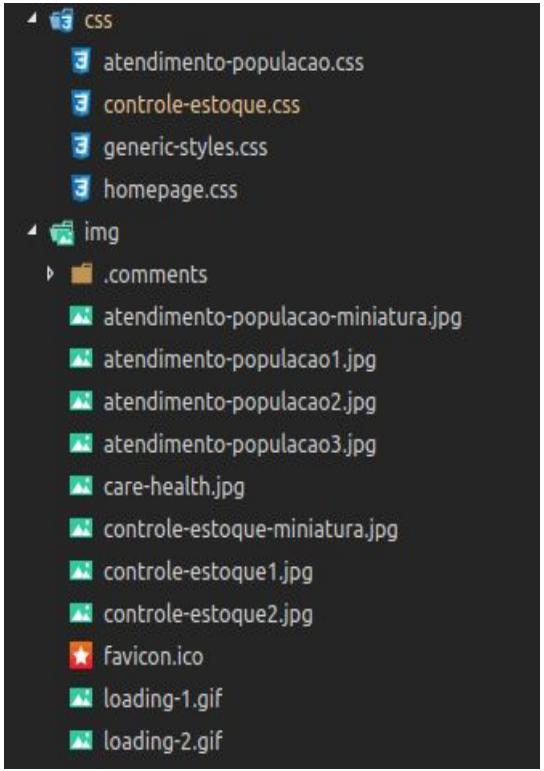


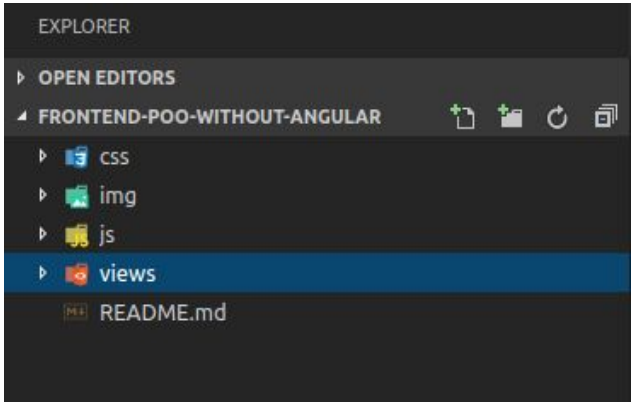
FRONTEND

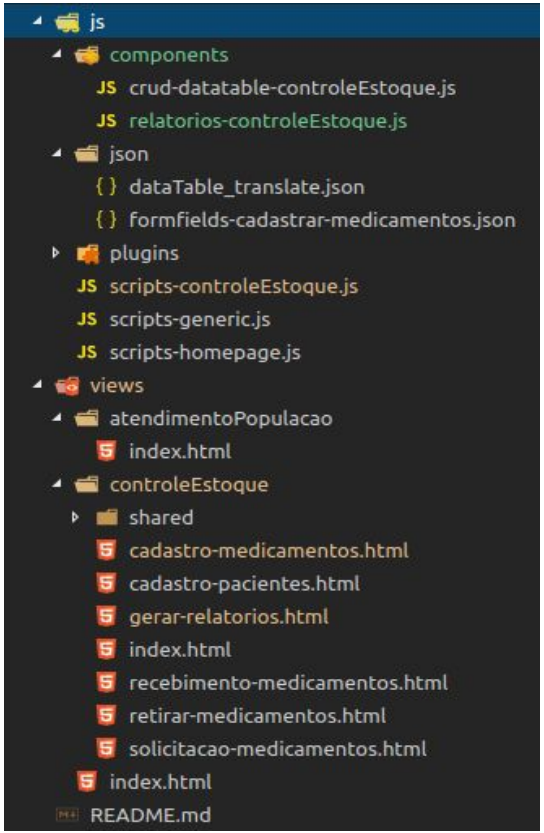
O QUE ESTÁ SENDO E SERÁ UTILIZADO?

- 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

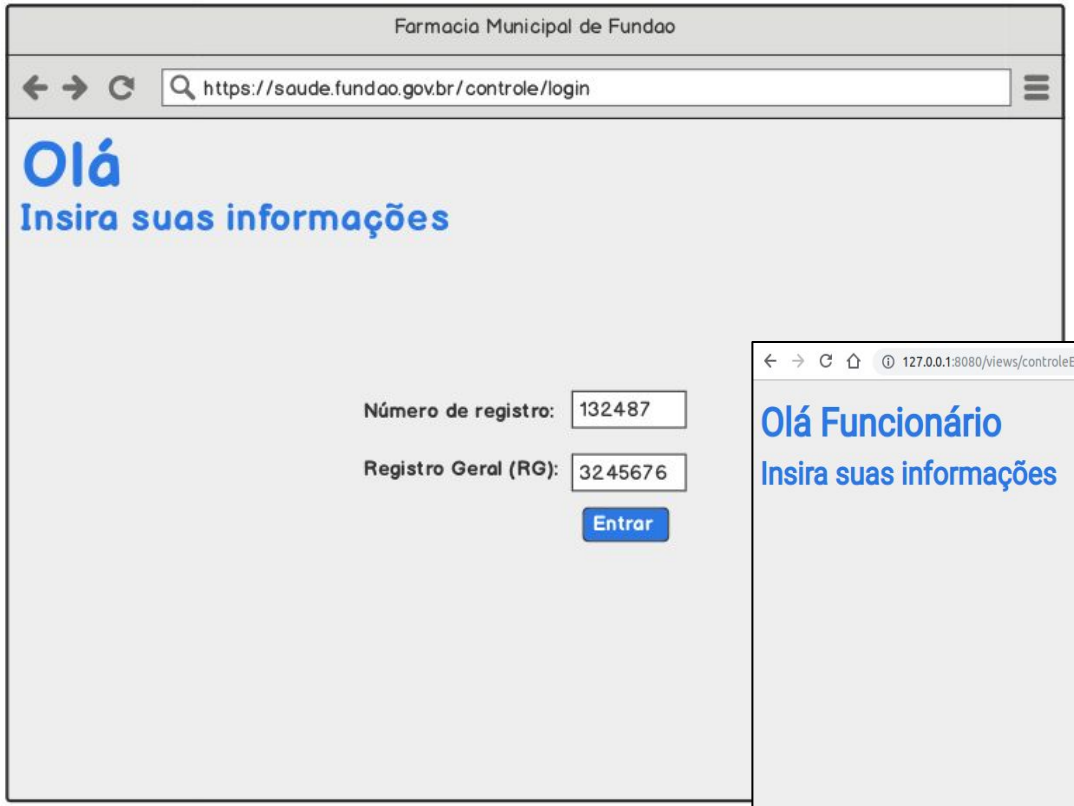
ESCOLHA O SEU DESTINO

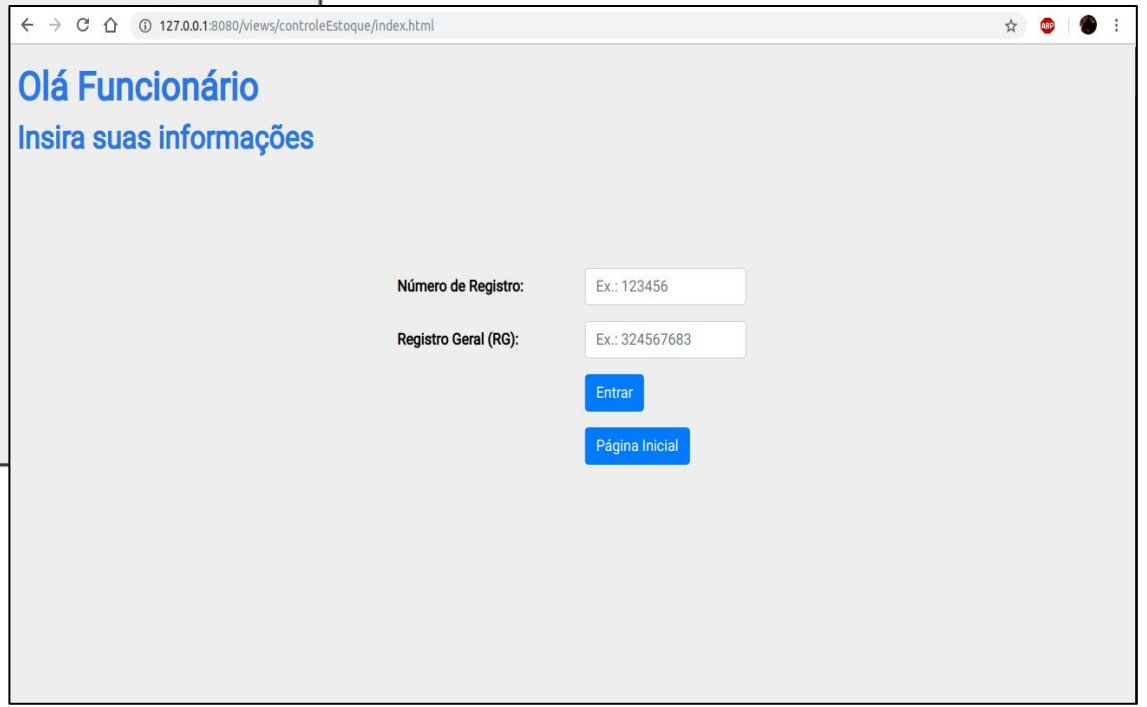
ATENDIMENTO A POPULAÇÃO

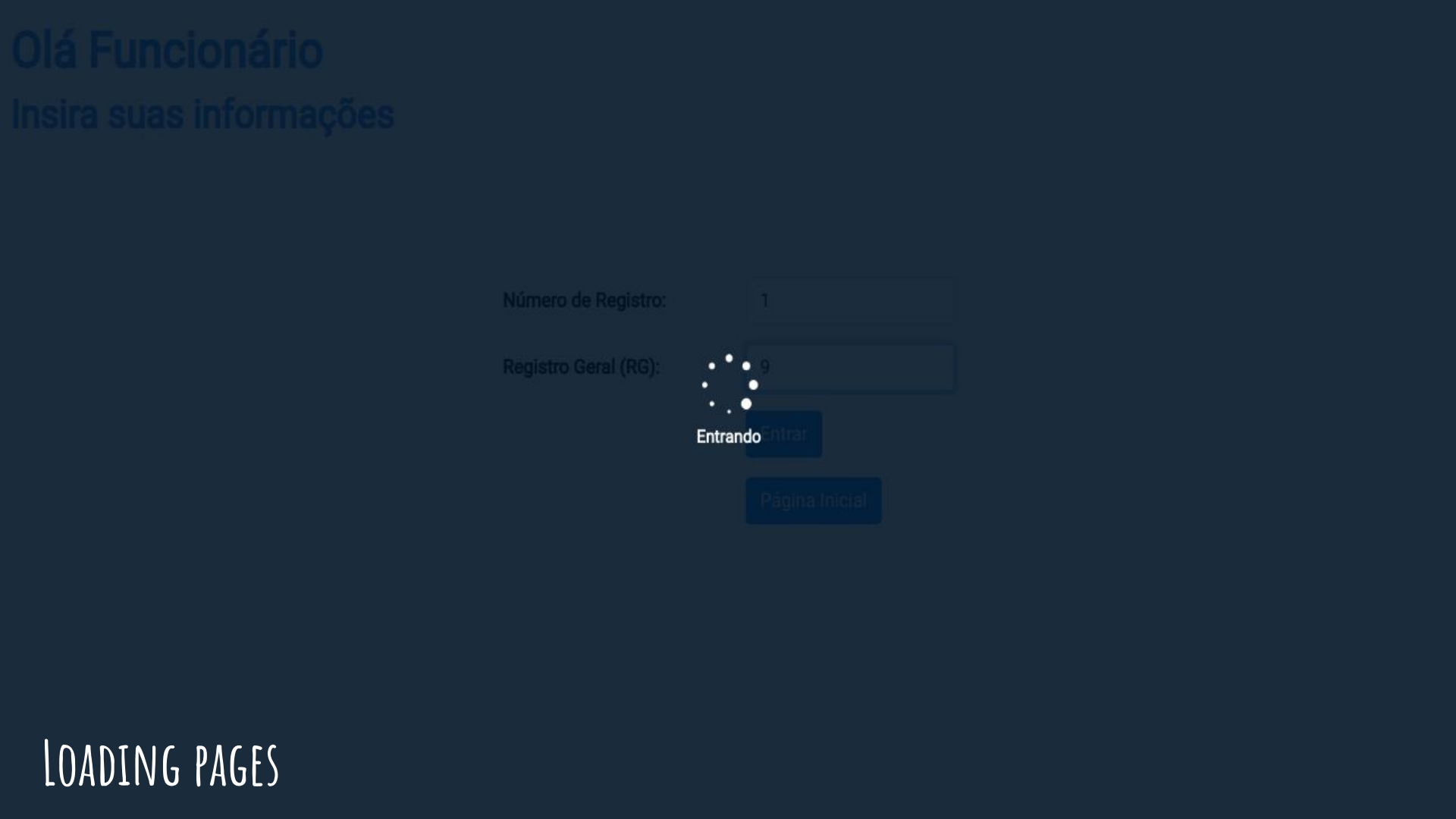
CONTROLE DE MEDICAMENTOS

Voltado para os atendentes dos postos

Protótipo vs aplicação
Login atendente

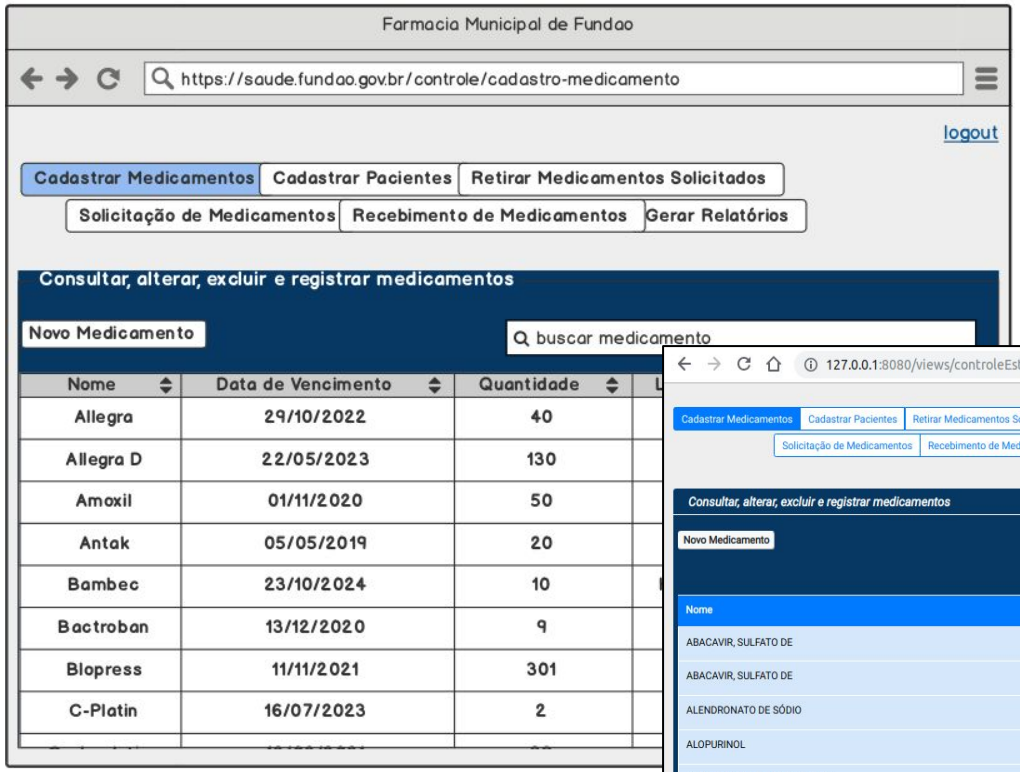


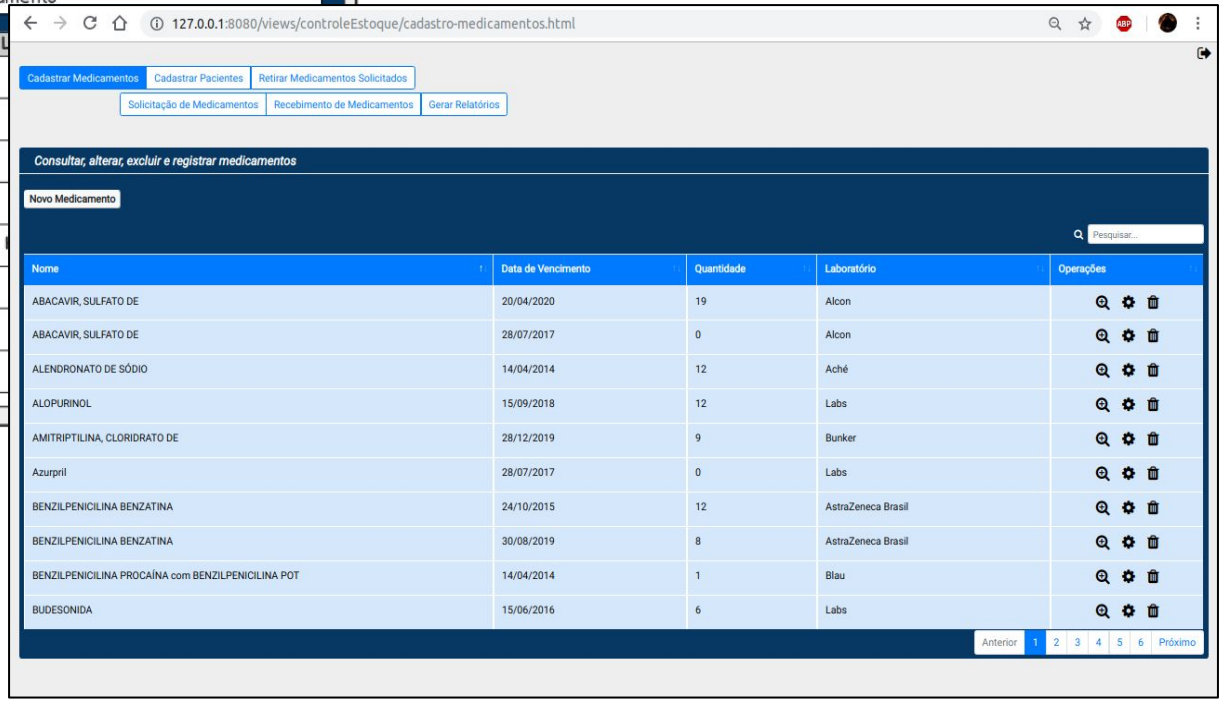




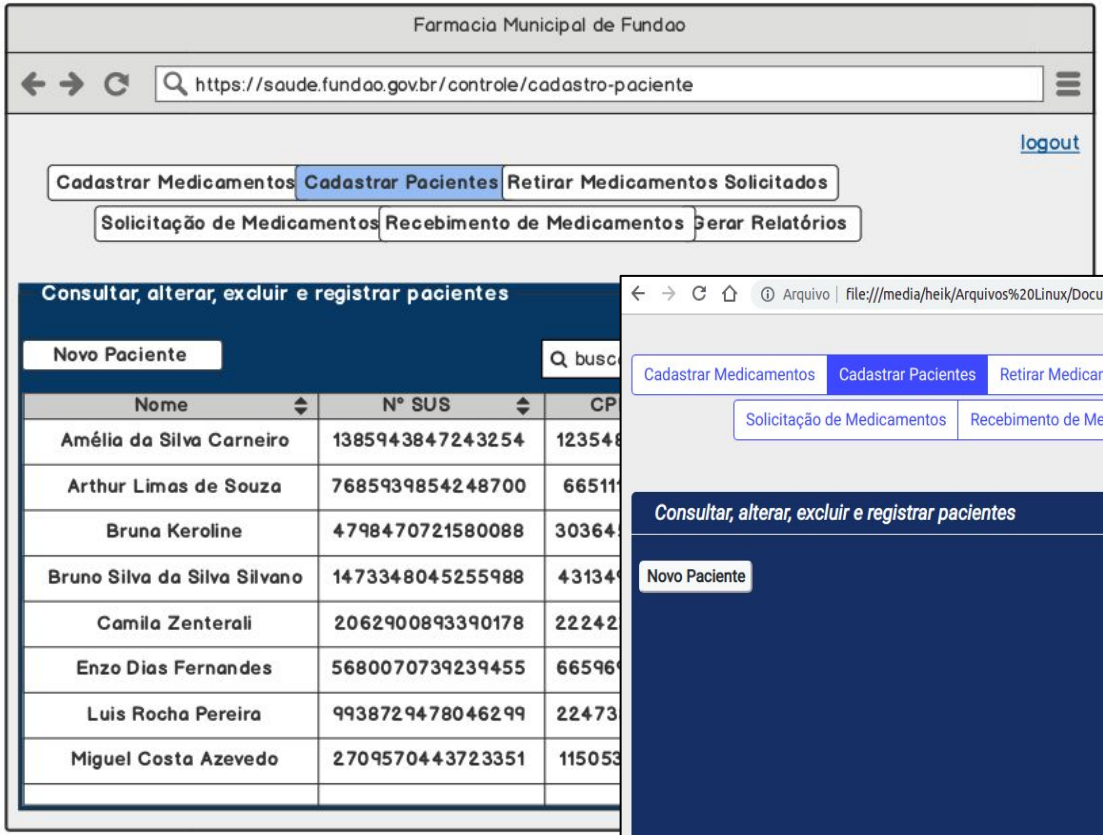
Loading pages

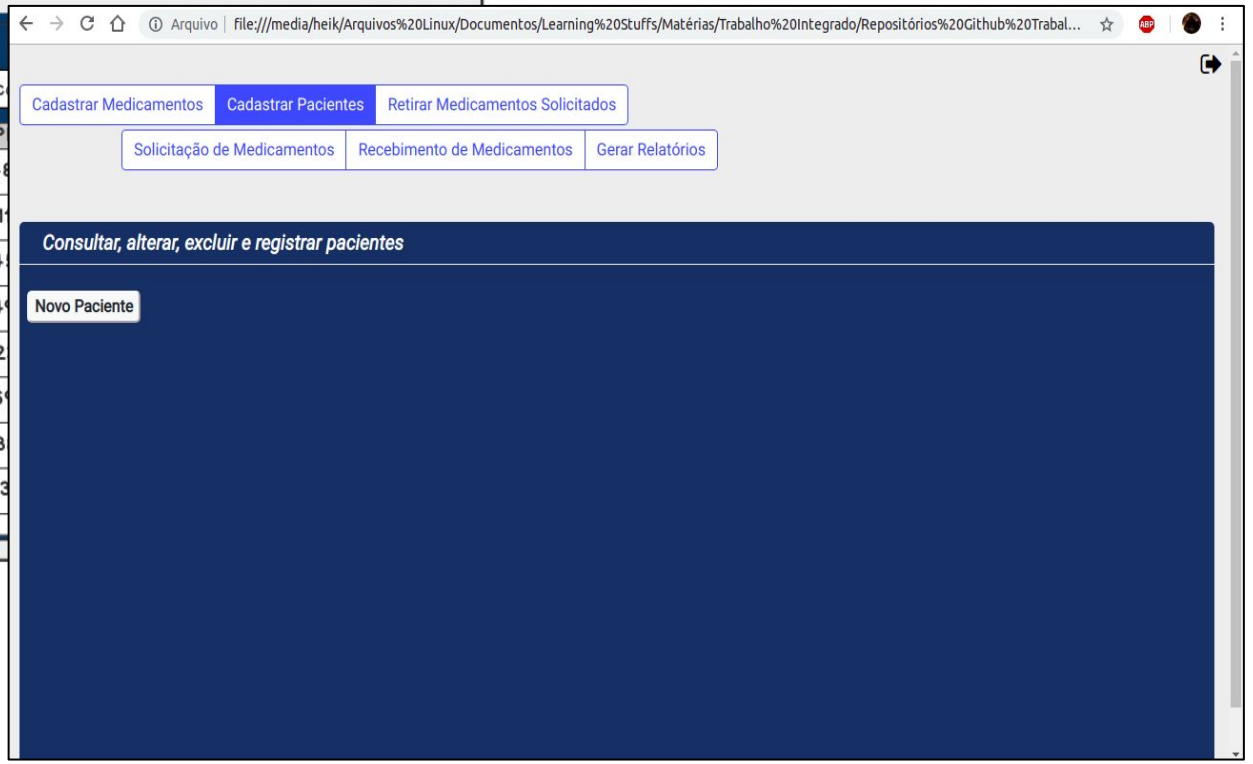
Protótipo vs aplicação
Cadastro de medicamentos



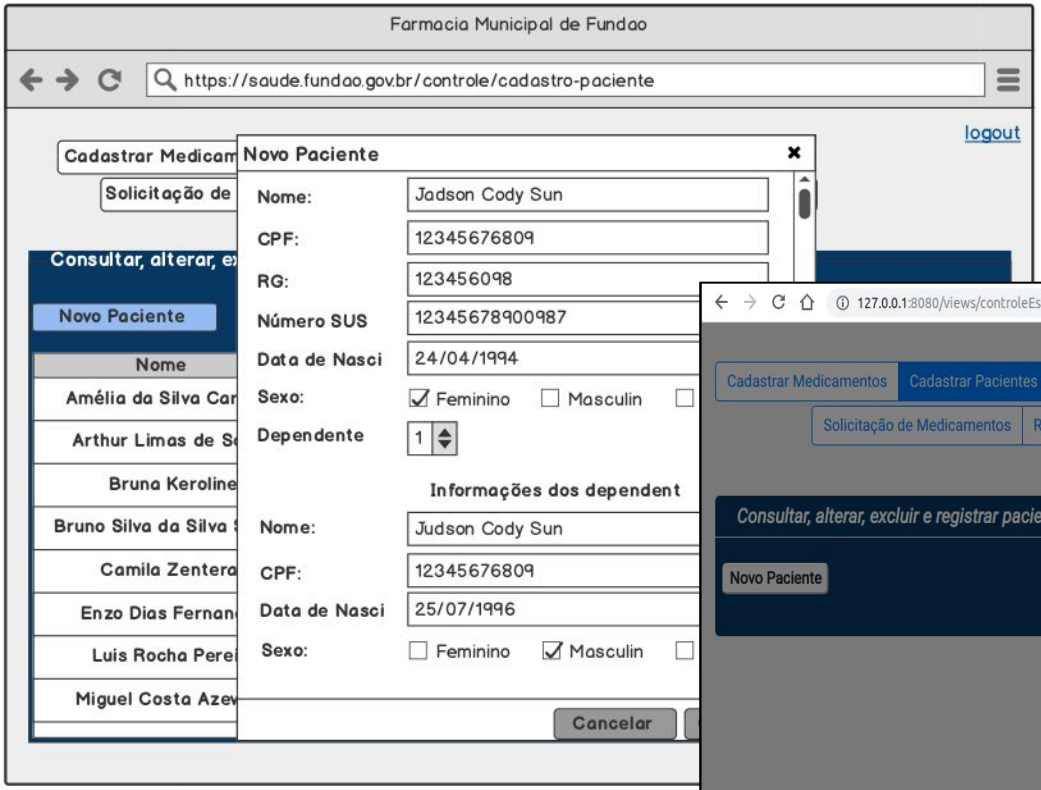


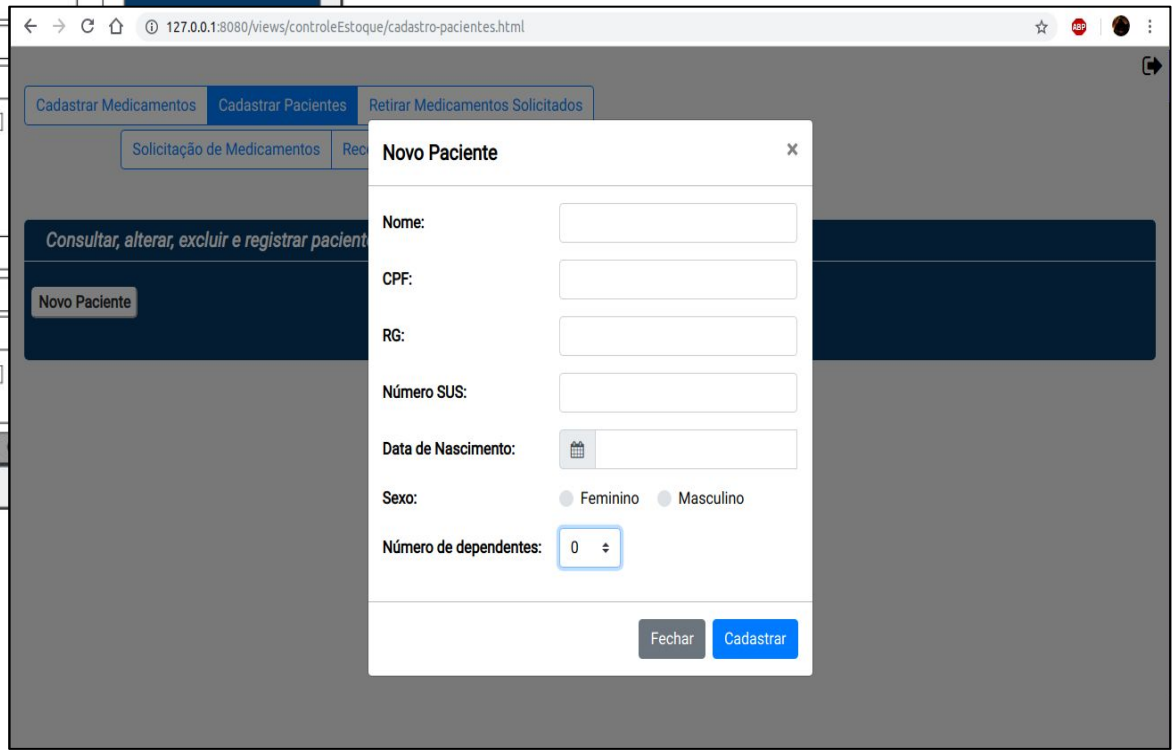
Protótipo vs aplicação
Cadastro de pacientes



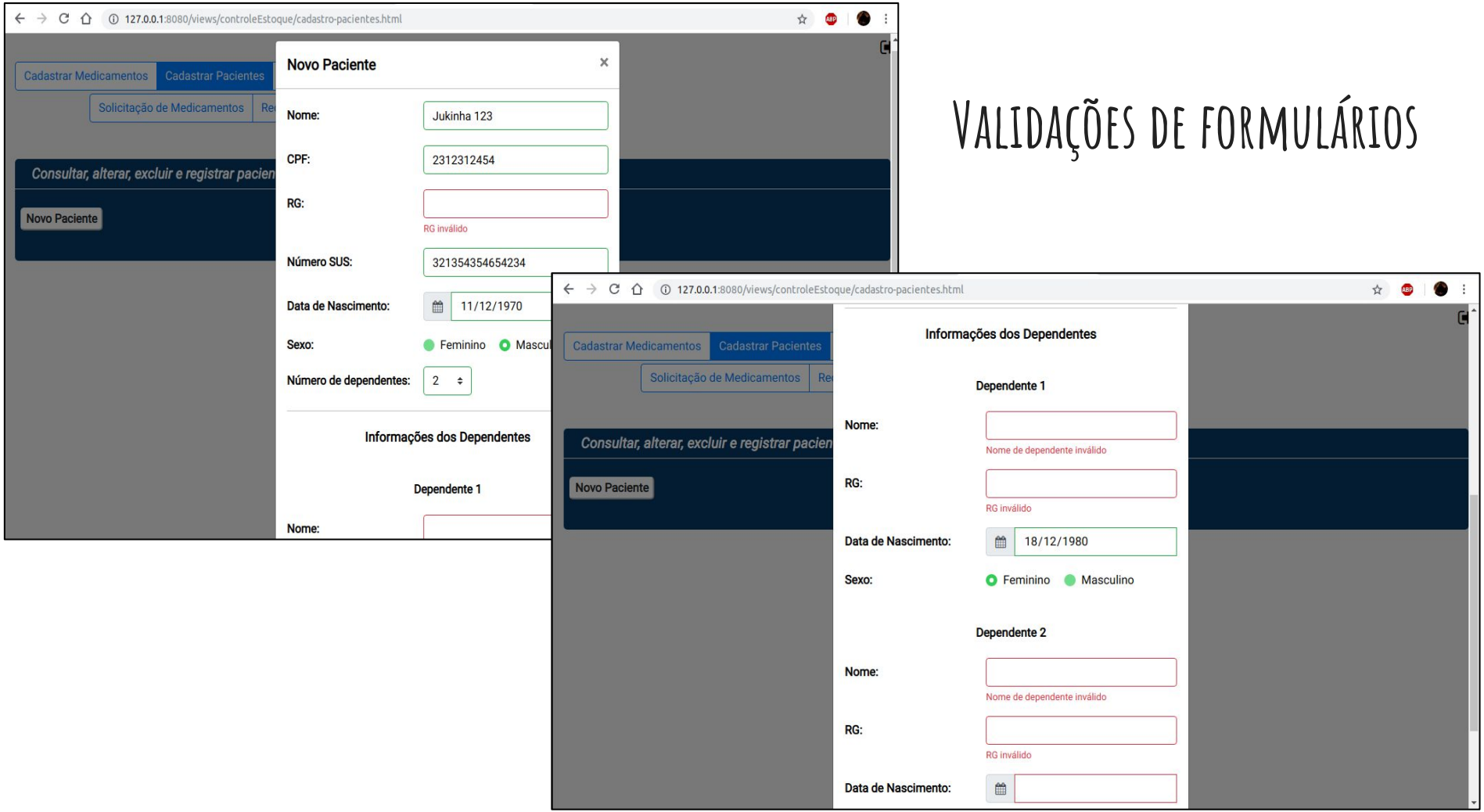


Protótipo vs aplicação
Cadastro de pacientes

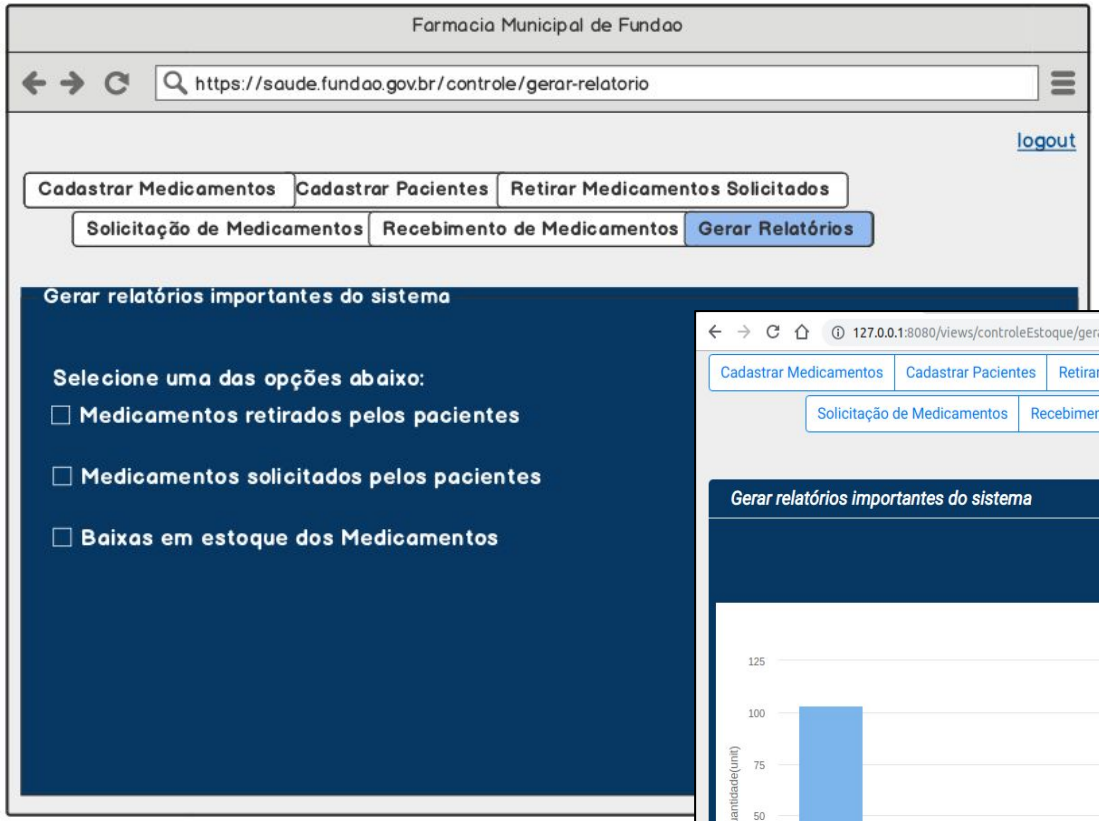


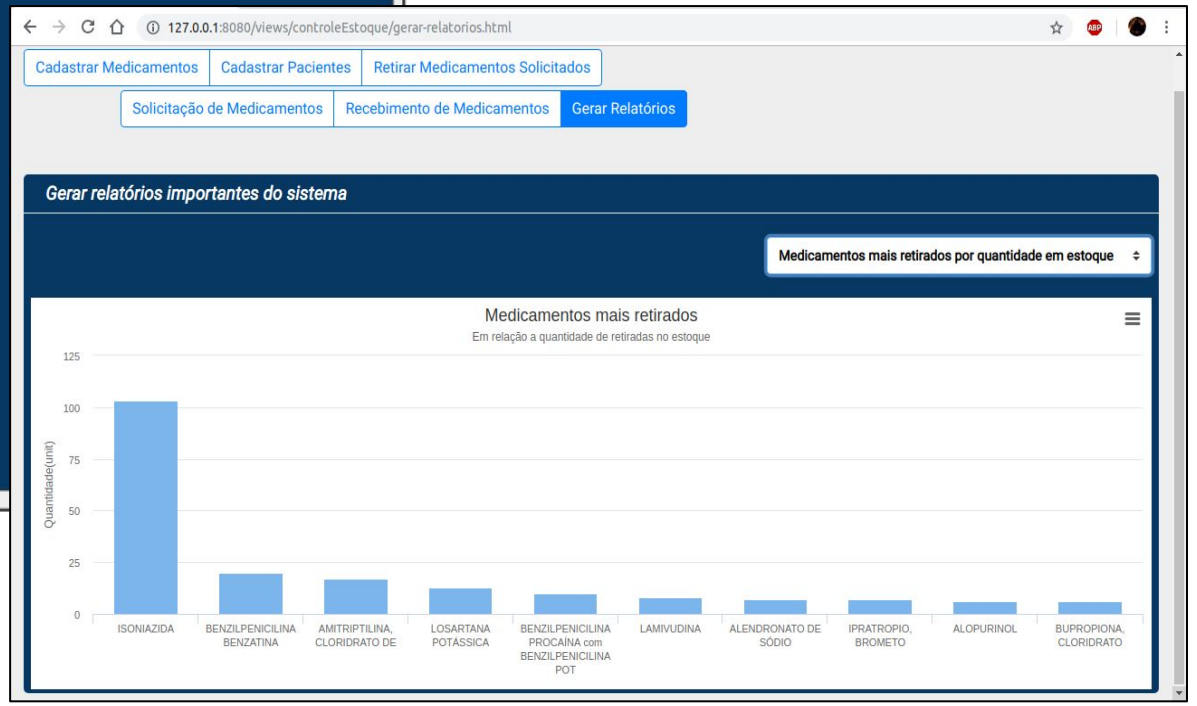


Validações de formulários

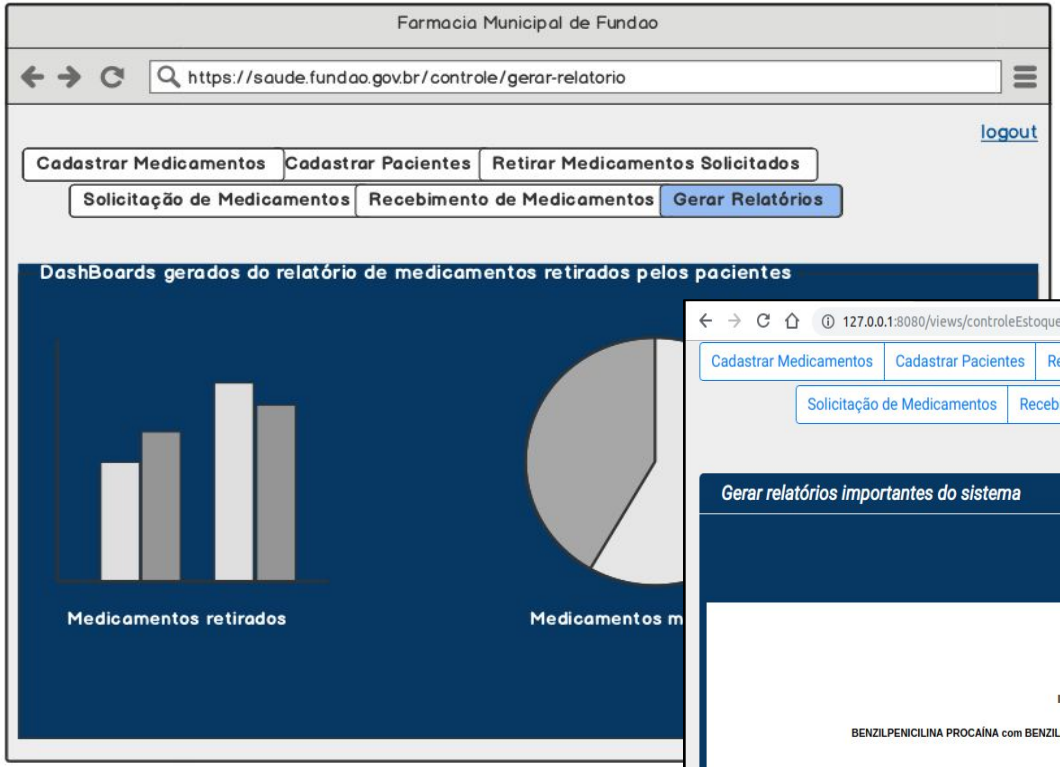


Protótipo vs aplicação
Gerar relatórios

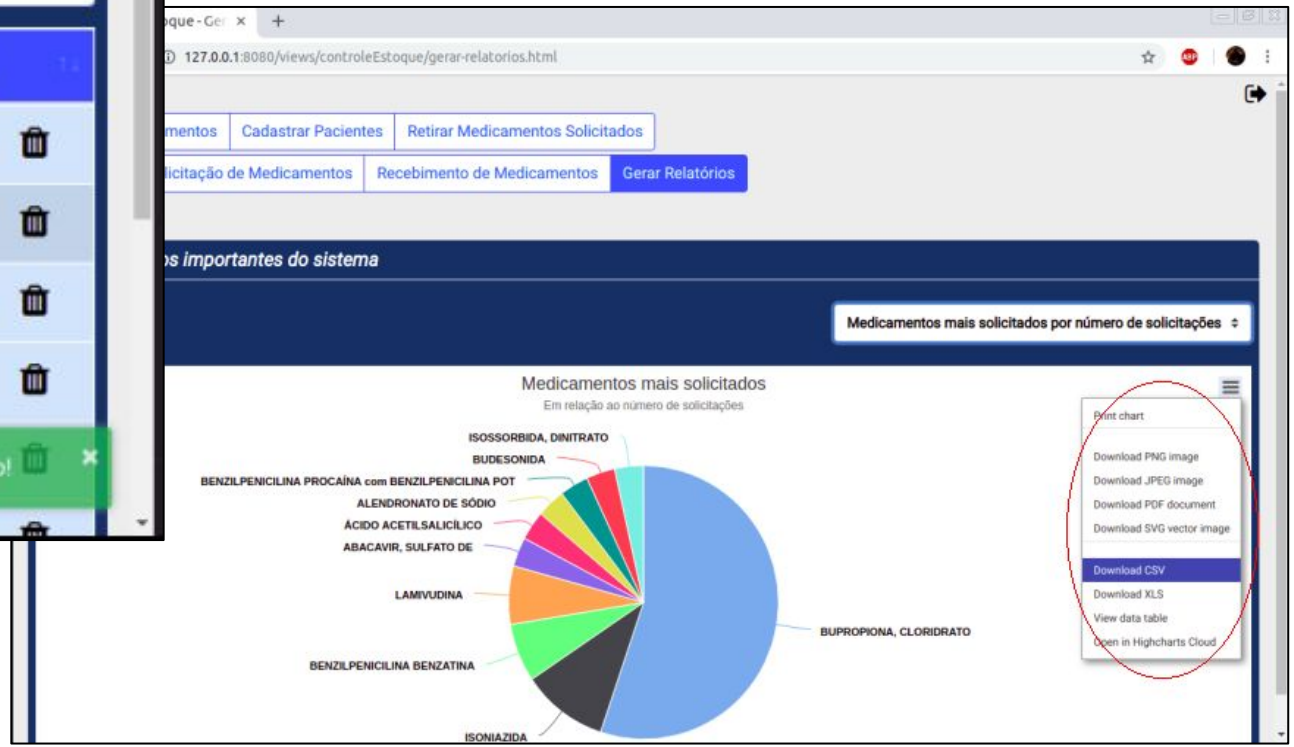




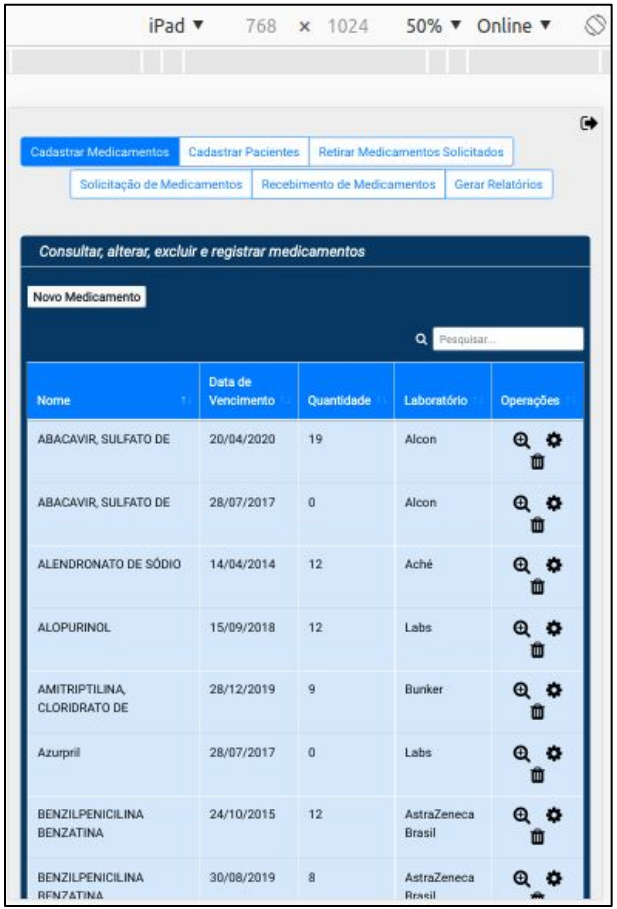
Protótipo vs aplicação
Gerar relatórios



Toasts e exportação de informações do relatórios



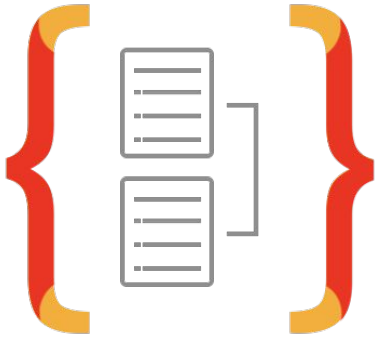
Um pouco de responsividade...



Backend

O que está sendo e será utilizado?

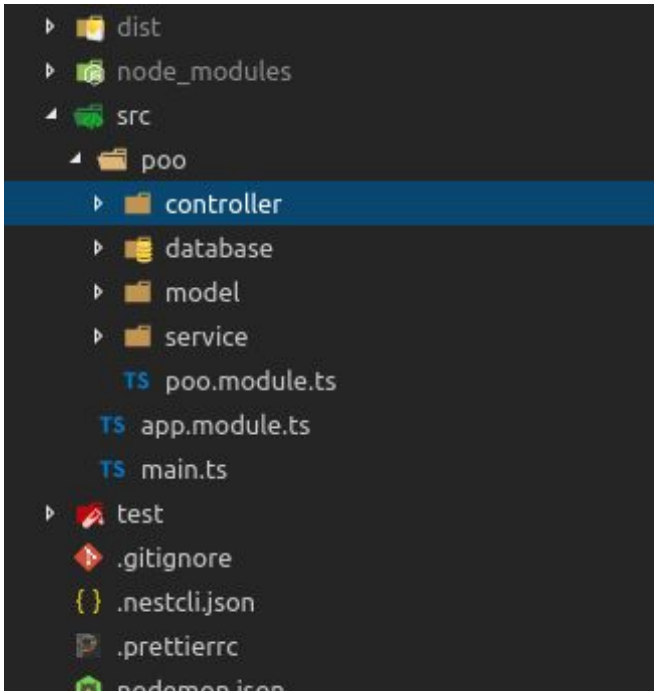
- Linguagem Utilizada
  - Typescript;
- Framework utilizado:
  - Nest JS;
- ORM utilizado:
  - TypeORM;

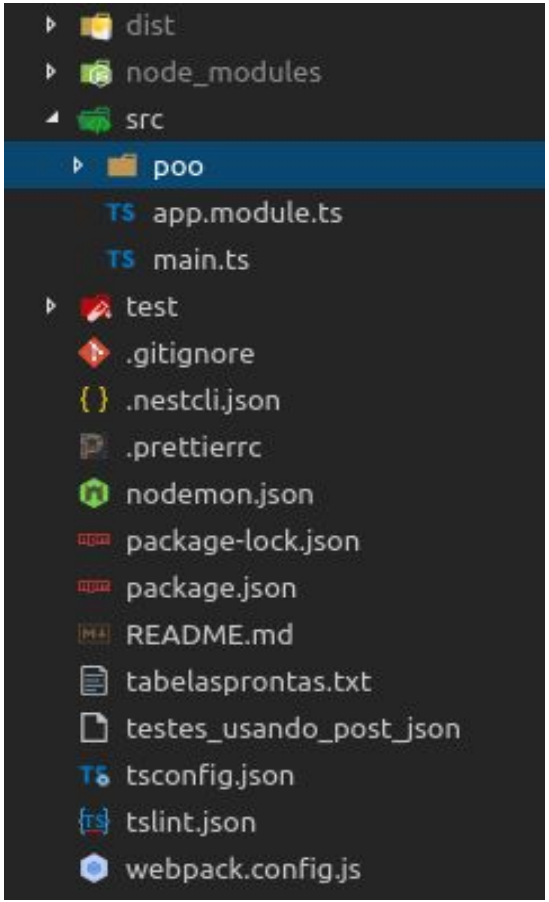


TYPEORM

nest

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');      You, a few seconds ago
    }
  } 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 atedente\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

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

```typescript
  /**
   * 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.
   */
```

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

```typescript
@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;
```

```
        * @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);
```

Padrão cache

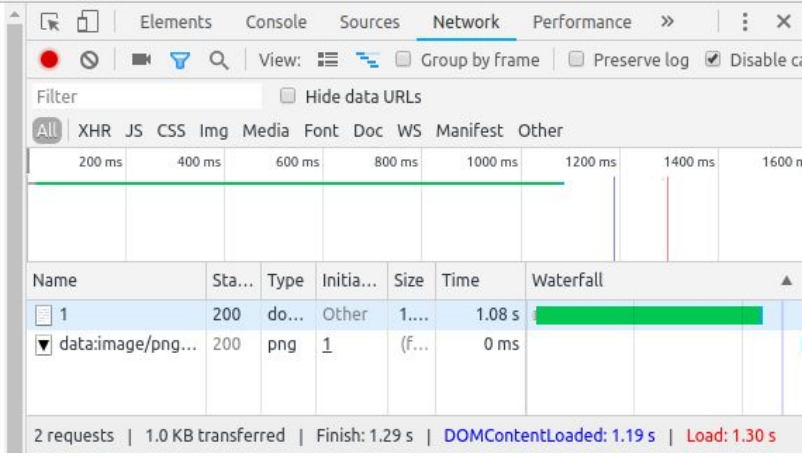
```
@Controller()
@UseInterceptors(CacheInterceptor)
export class SolicitacaoController {
  constructor(private readonly solicitacaoService: SolicitacaoService) {}
  @Get('/solicitacao/:id')      You, a month ago • solicitacao em progress
  async readOne(@Res() res, @Param() id) {
```

```
@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

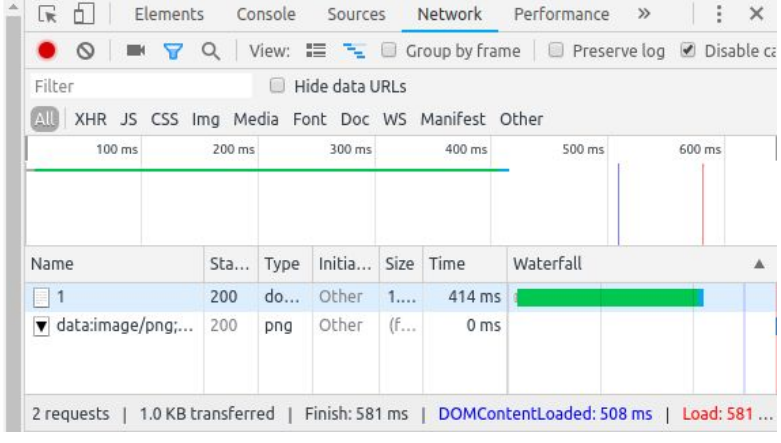

```
{
  "idsolicitacao": 1,
  "data_hora": "2017-07-29T03:00:00.000Z",
  "quantidademedicamento": 1,
  "estadosolicitacao": 1,
  "titular": {
    "idpessoa": 50,
    "numerosus": 10001,
    "depedente": [],
    "pessoa": {
      "idpessoa": 50,
      "nome": "Vilma Gato",
      "datanascimento": "1956-01-31T03:00:00.000Z",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



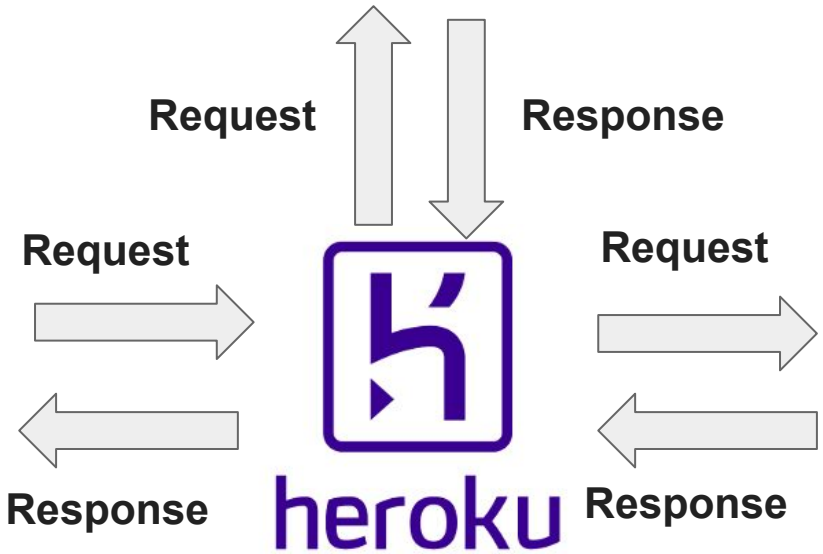
Implantação

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



ElephantSQL

GitHub

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-POO-without-angular
- https://docs.google.com/document/d/1gVortYoQ8UpUQGYBodMUtm81sVFyRQc1K5_Q1l0pu-o/edit