FILA ZERO

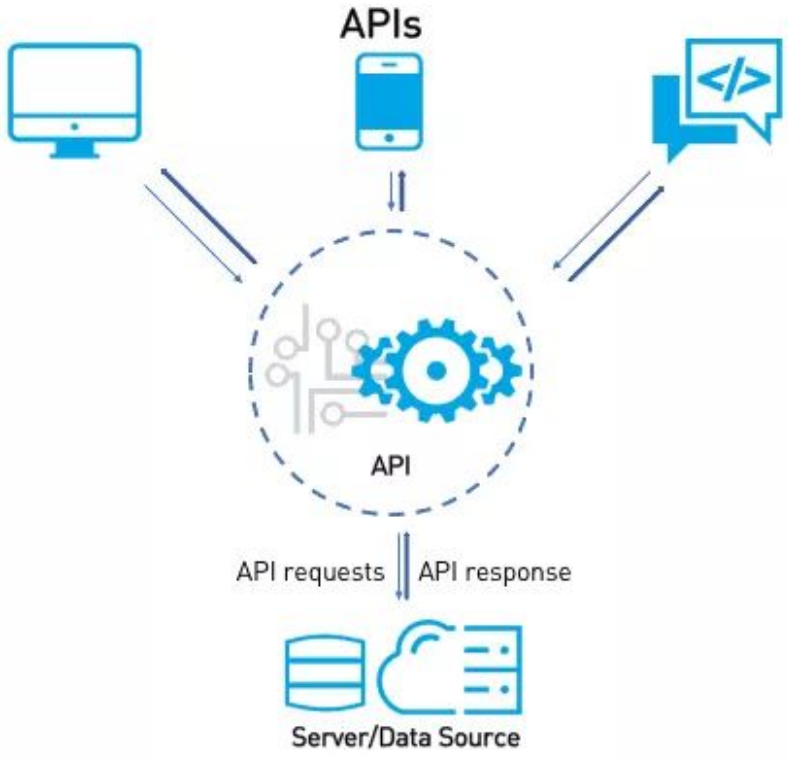
Web services + dashboards

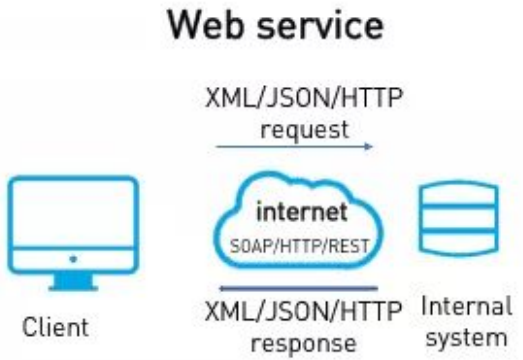
- **Harã Heique**
- **Lucas Gomes**

Introdução

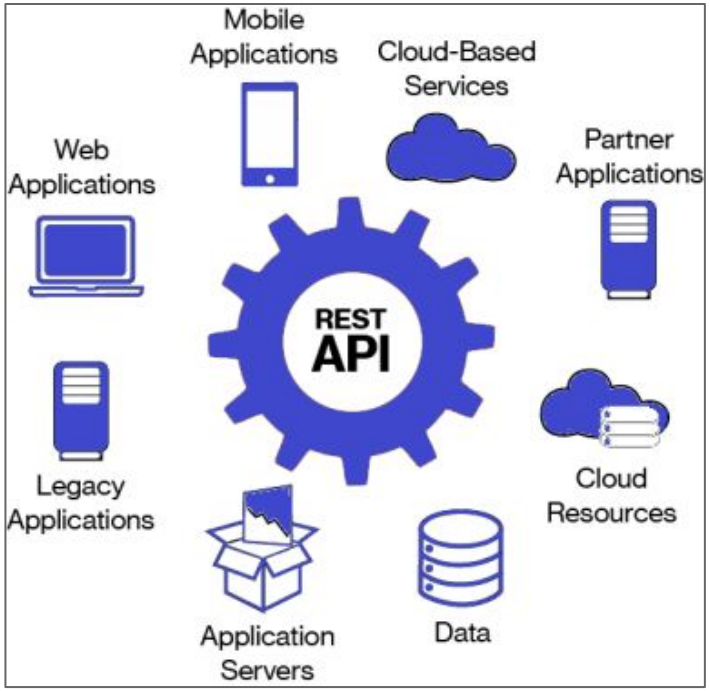
- Web Service;
- Transferência de Estado Representacional(REST);
- Dashboards/Gráficos utilizando Highcharts;

Web Service


APIs
API
API requests
API response
Server/Data Source


Web service
XML/JSON/HTTP
request
internet
SOAP/HTTP/REST
XML/JSON/HTTP
response
Client
Internal
system

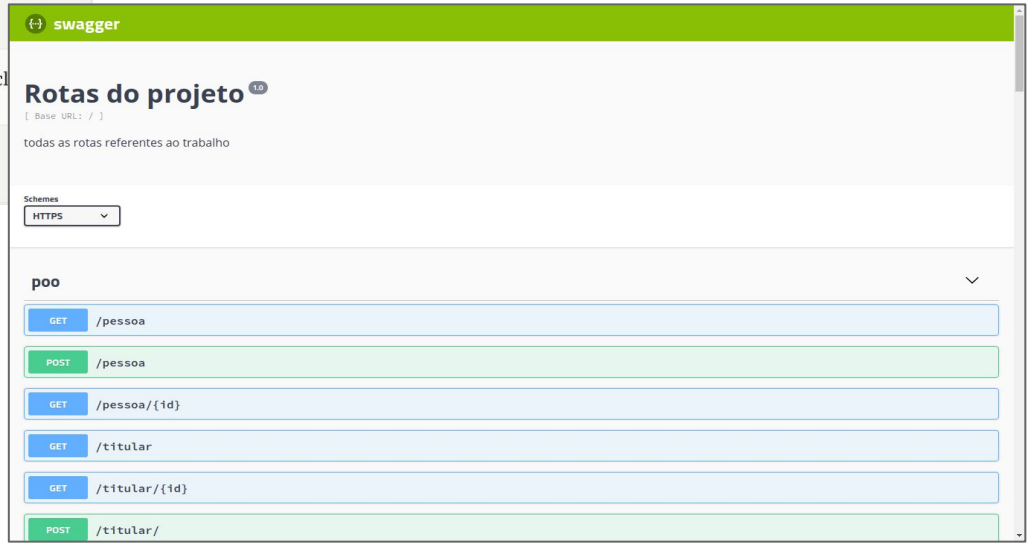
Transferência de Estado Representacional(REST)



Método	URI	Utilização
GET	/clientes	Recuperar os dados de todos os clientes.
GET	/clientes/id	Recuperar os dados de um determinado cliente.
POST	/clientes	Criar um novo cliente.
PUT	/clientes/id	Atualizar os dados de um determinado cliente.
DELETE	/clientes/id	Excluir um determinado cliente.

Transferência de Estado Representacional(REST)

Método	URI	Utilização
GET	/clientes	Recuperar os dados de todos os clientes.
GET	/clientes/id	Recuperar os dados de um determinado cliente.
POST	/clientes	Criar um novo cliente.
PUT	/clientes/id	Atualizar os dados de um determinado cl
DELETE	/clientes/id	Excluir um determinado cliente.

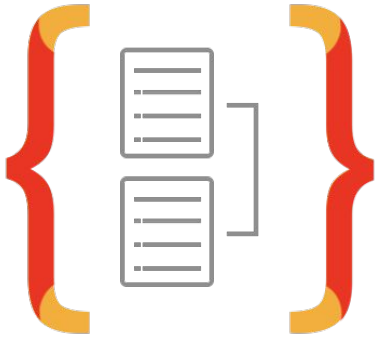


O que está sendo e será utilizado no Front end

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

O que está sendo e será utilizado no Back end

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



TYPEORM

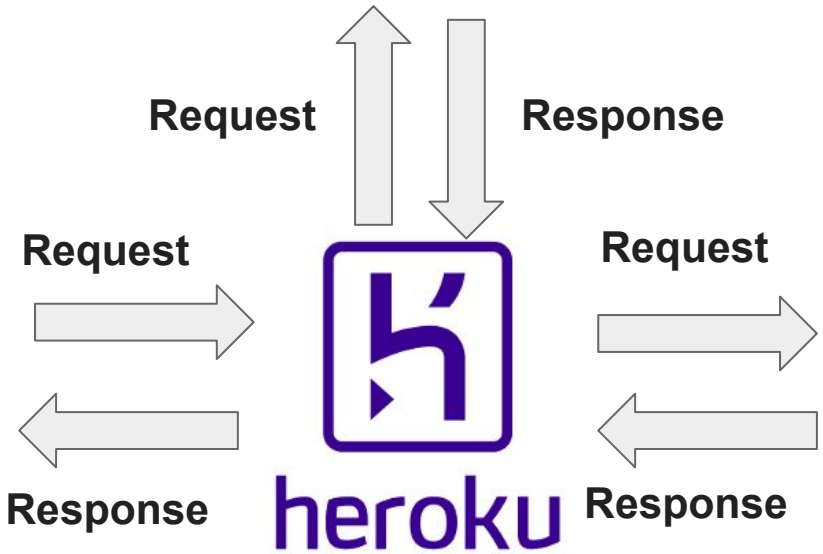
Implantação

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"
    }
  },
```

RELATÓRIOS

Relatórios importantes do sistema

Através dos dados recebidos via JSON do backend foram realizados gráficos utilizando a biblioteca do highcharts no frontend.

Esses gráficos são as views que representam alguns dos relatórios do sistema, onde o atendente pode visualizar informações acerca das solicitações e retiradas de medicamentos realizado pelos clientes, além de ter a capacidade de exportação dos relatórios em planilhas, imagens e afins.

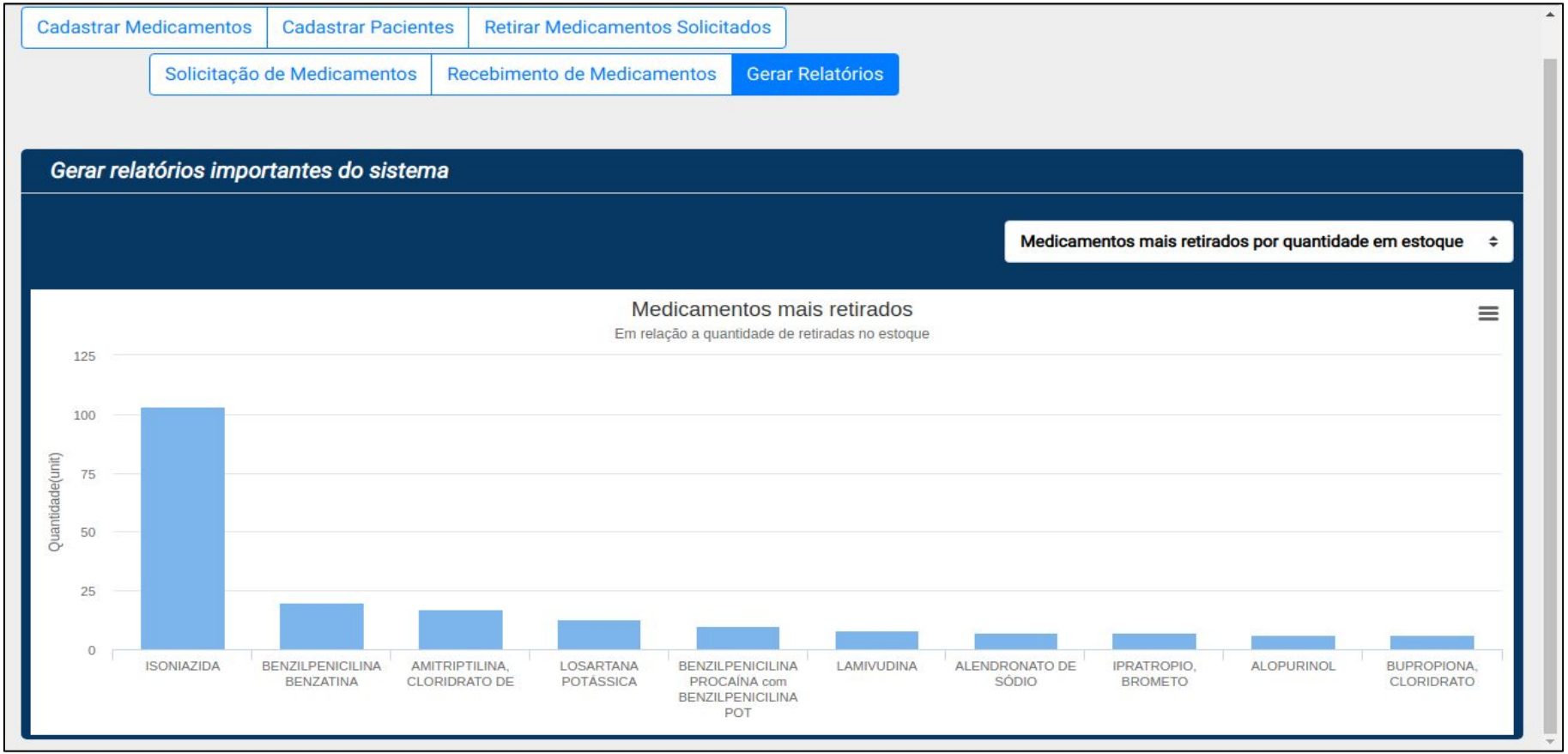
View - Medicamentos mais retirados por quantidade

```sql
-- Para relatório de medicamento mais retirados pelos pacientes em relação a quantidade retirada no estoque
CREATE VIEW view_medicamentos_mais_retirados_por_quantidade AS
SELECT me.nome, SUM(re.quantidademedicamentos) AS "Quantidade" FROM medicamento AS me
INNER JOIN medicamento_posto AS mp ON (mp.idmedicamento = me.idmedicamento)
INNER JOIN recebimento AS re ON (re.idmedicamentoposto = mp.idmedicamentoposto)
GROUP BY me.idmedicamento
ORDER BY "Quantidade" DESC;
```

```typescript
async SumMedicamentosEstoque(): Promise<Medicamento | any> {
  try {
  return Medicamento.createQueryBuilder("medicamento")
  .select("medicamento.nome")
  .addSelect("SUM(recebimento.quantidademedicamentos) as quantidade")
  .innerJoin("medicamento.medicamentoPosto", "medicamentoPosto")
  .innerJoin("medicamentoPosto.recebimento", "recebimento")
  .groupBy("medicamento.nome")
  .groupBy("medicamento.idmedicamento")
  .orderBy("medicamento.nome")
  .limit(10).getRawMany()
  } catch (err) {
    throw new Error(
      `Erro ao verificar o relatorio \n Erro: ${err.name}\n Mensagem: ${
        err.message
      }\n Os parametros estao certos?`,
    );
  }
}
```

Query/código no backend

Resultado do relatório no front-end



View - Medicamentos mais retirados por número de retiradas

```sql
-- Para relatório de medicamento mais retirados pelos pacientes em relação por quantidade de retiradas
CREATE VIEW view_medicamentos_mais_retirados_por_numero_retiradas AS
SELECT me.nome, COUNT(me.idmedicamento) AS "Número de Retiradas" FROM medicamento AS me
INNER JOIN medicamento_posto AS mp ON (mp.idmedicamento = me.idmedicamento)
INNER JOIN recebimento AS re ON (re.idmedicamentoposto = mp.idmedicamentoposto)
GROUP BY me.idmedicamento
ORDER BY "Número de Retiradas" DESC;
```

```typescript
async CountMedicamentosEstoque(): Promise<Medicamento | any> {
  try {
  return Medicamento.createQueryBuilder("medicamento")
  .select("medicamento.nome")
  .addSelect("COUNT(recebimento.quantidademedicamentos) as numero_de_retiradas")
  .innerJoin("medicamento.medicamentoPosto", "medicamentoPosto")
  .innerJoin("medicamentoPosto.recebimento", "recebimento")
  .groupBy("medicamento.nome")
  .groupBy("medicamento.idmedicamento")
  .orderBy("medicamento.nome")
  .limit(10).getRawMany()
  } catch (err) {
    throw new Error(
      `Erro ao verificar o relatorio \n Erro: ${err.name}\n Mensagem: ${
        err.message
      }\n Os parametros estao certos?`,
    );
  }
}
```

Query/código no backend

Resultado do relatório no front-end

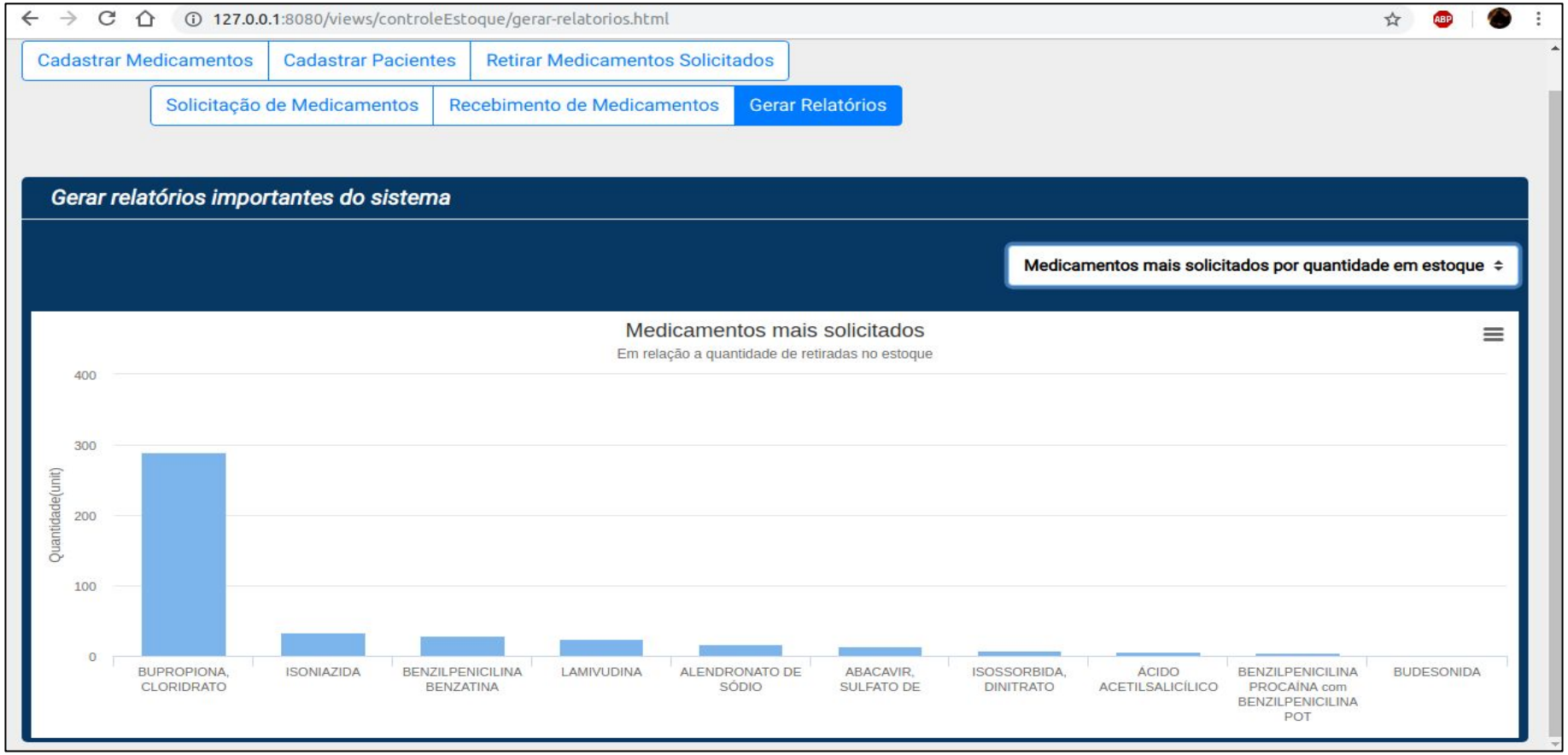
View - Medicamentos mais solicitados por quantidade

```sql
-- Para relatório de medicamento mais solicitados pelos pacientes em relação a quantidade solicitada do estoque
CREATE VIEW view_medicamentos_mais_solicitados_por_quantidade AS
SELECT me.nome, SUM(me.idmedicamento) AS "Quantidade" FROM medicamento AS me
INNER JOIN medicamento_posto AS mp ON (mp.idmedicamento = me.idmedicamento)
INNER JOIN solicitacao AS so ON (so.idmedicamentoposto = mp.idmedicamentoposto)
GROUP BY me.idmedicamento
ORDER BY "Quantidade" DESC;
```

```typescript
async SumQuanntidadeSolicitado(): Promise<Medicamento | any> {
  try {
  return Medicamento.createQueryBuilder("medicamento")
  .select("medicamento.nome")
  .addSelect("SUM(solicitacao.quantidademedicamento) as quantidade")
  .innerJoin("medicamento.medicamentoPosto", "medicamentoPosto")
  .innerJoin("medicamentoPosto.solicitacao", "solicitacao")
  .groupBy("medicamento.nome")
  .groupBy("medicamento.idmedicamento")
  .orderBy("medicamento.nome")
  .limit(10).getRawMany()
  } catch (err) {
    throw new Error(
      `Erro ao verificar o relatorio \n Erro: ${err.name}\n Mensagem: ${
        err.message
      }\n Os parametros estao certos?`,
    );
  }
}
```

Query/código no backend

Resultado do relatório no front-end



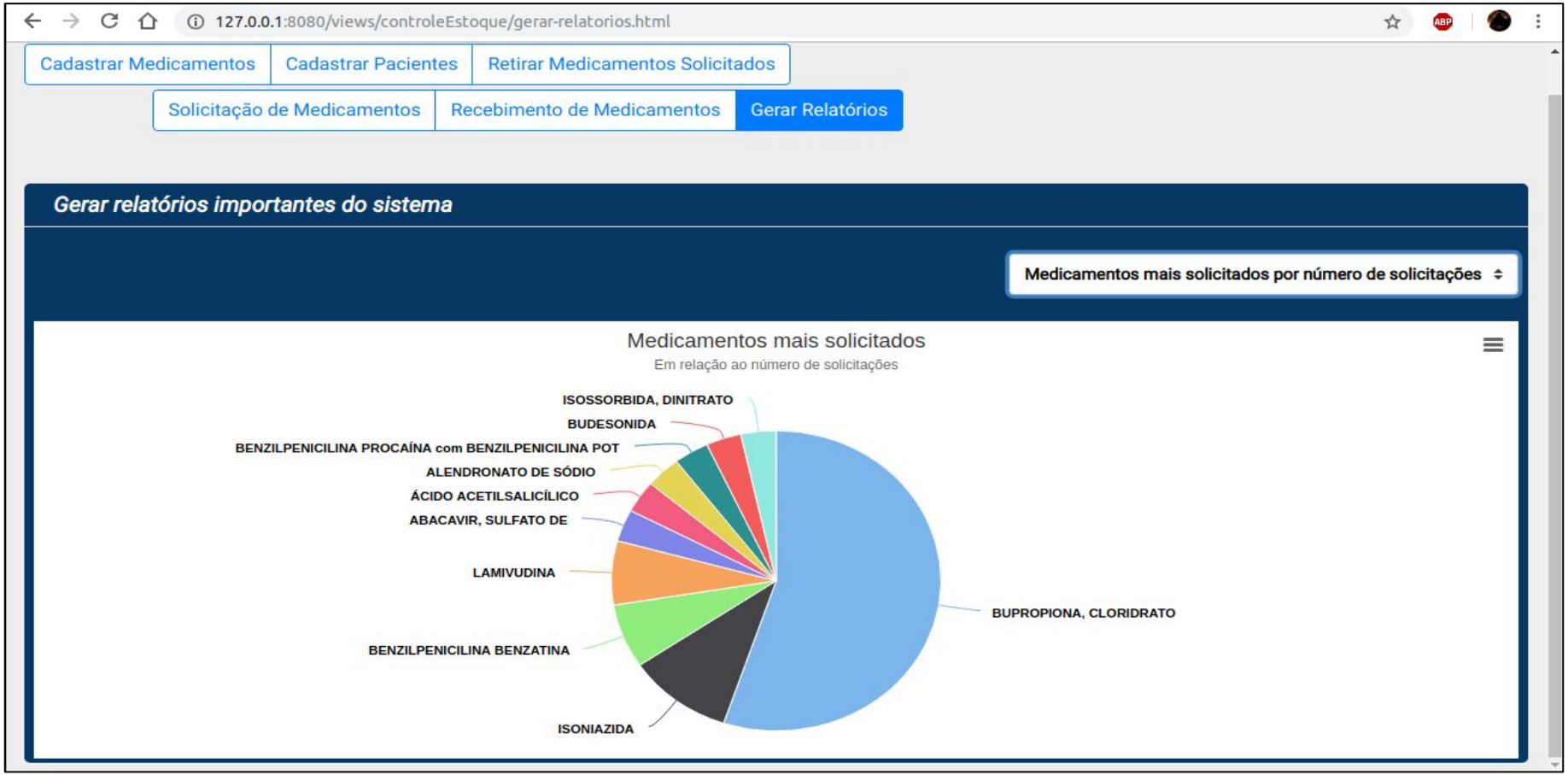
View - Medicamentos mais solicitados por número de solicitações

```sql
-- Para relatório de medicamento mais solicitados pelos pacientes em relação por quantidade de solicatações
CREATE VIEW view_medicamentos_mais_solicitados_por_numero_solicitacoes AS
SELECT me.nome, COUNT(me.idmedicamento) AS "Número de Solicitações" FROM medicamento AS me
INNER JOIN medicamento_posto AS mp ON (mp.idmedicamento = me.idmedicamento)
INNER JOIN solicitacao AS so ON (so.idmedicamentoposto = mp.idmedicamentoposto)
GROUP BY me.idmedicamento
ORDER BY "Número de Solicitações" DESC;
```

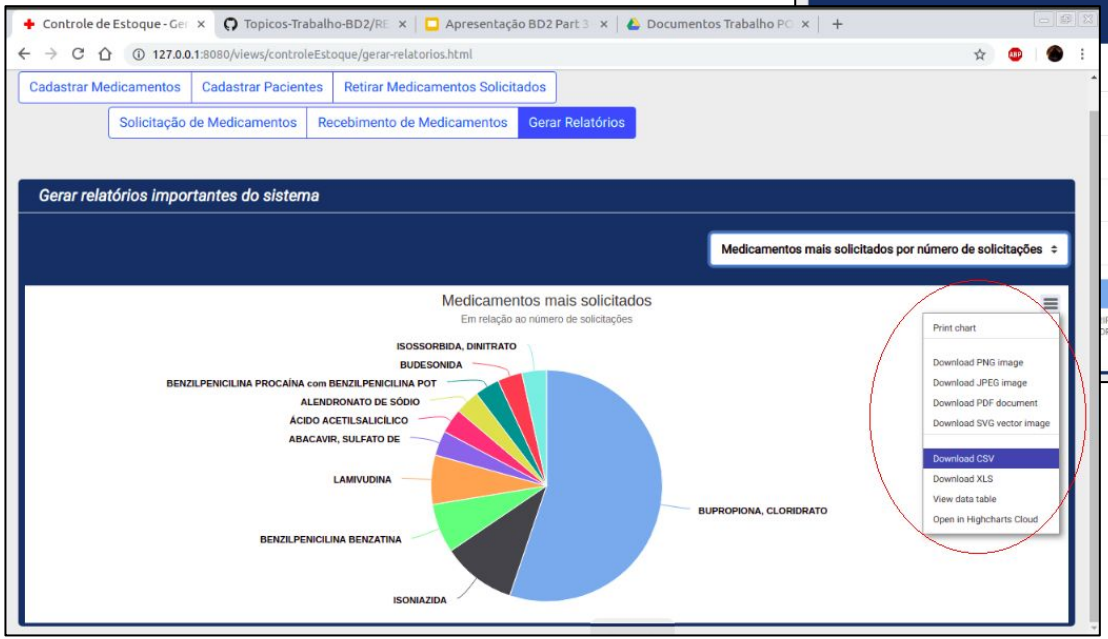
```typescript
async CountQuanntidadeSolicitado(): Promise<Medicamento | any> {
  try {
  return Medicamento.createQueryBuilder("medicamento")
  .select("medicamento.nome")
  .addSelect("COUNT(solicitacao.quantidademedicamento) as numero_de_solicitacoes")
  .innerJoin("medicamento.medicamentoPosto", "medicamentoPosto")
  .innerJoin("medicamentoPosto.solicitacao", "solicitacao")
  .groupBy("medicamento.nome")
  .groupBy("medicamento.idmedicamento")
  .orderBy("medicamento.nome")
  .limit(10).getRawMany()
  } catch (err) {
    throw new Error(
      `Erro ao verificar o relatorio \n Erro: ${err.name}\n Mensagem: ${
        err.message
      }\n Os parametros estao certos?`,
    );
  }
}
```

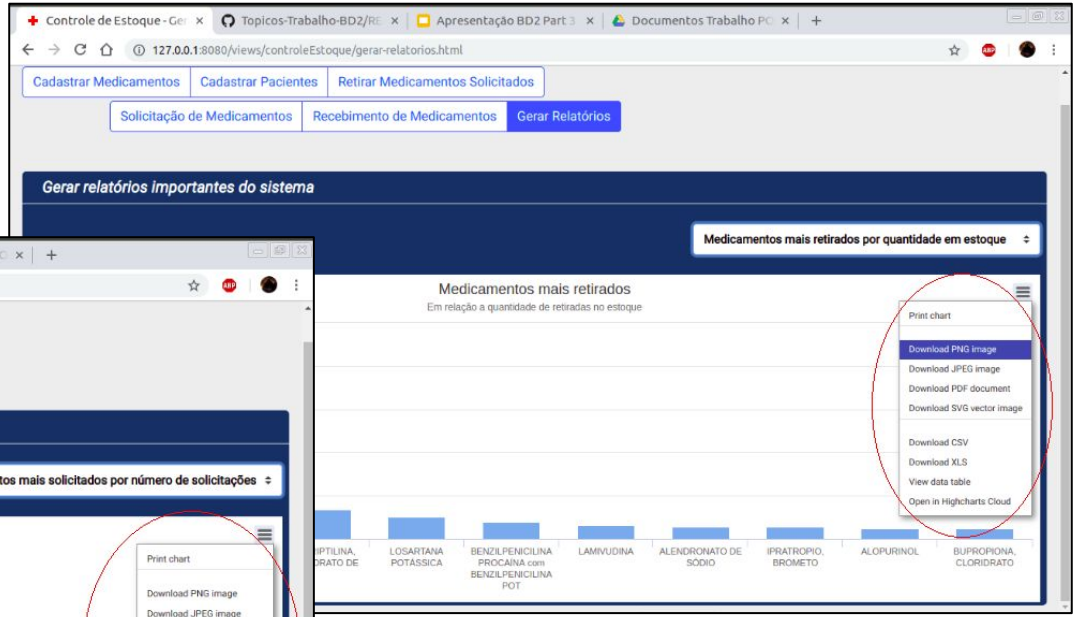
Query/código no backend

Resultado do relatório no front-end



Exportação de informações dos relatórios





Github

- https://github.com/lukasg18/Topicos-Trabalho-BD2
- https://github.com/lukasg18/poo2-backend
- https://github.com/HaraHeique/frontend-POO-without-angular

Referências

- http://blog.caelum.com.br/rest-principios-e-boas-praticas/
- https://becode.com.br/o-que-e-api-rest-e-restful/