

Projeto Centro de Distribuição de Medicamentos Distribui Mais

Grupo: Ana Gabriela da Silva Bezerra, Felipe Cassiano Barbosa, Gabriel Gomes Fernandes, Gabriela Araujo de Abreu, Mateus Ferrareze Malgarin

Responsabilidades:

Front-end: Ana Gabriela, Felipe, Mateus

Back-end: Gabriel, Gabriela

Banco de dados: Gabriel, Ana Gabriela

Requisitos: Gabriel, Gabriela

Testes: Mateus

Objetivo:

Desenvolver um aplicativo para gerenciar o estoque de medicamentos e as rotas de entregas deles do centro de distribuição para as farmácias.

Requisitos funcionais:

- [RF001] O sistema deve permitir o cadastro de novos medicamentos no banco de dados
- [RF002] O sistema deve permitir a leitura dos medicamentos cadastrados no banco
- [RF003] O sistema deve permitir a atualização/edição dos medicamentos cadastrados no banco
- [RF004] O sistema deve permitir apagar medicamentos cadastrados no banco
- [RF005] O sistema deve permitir a seleção de um destino para entrega de medicamentos;
- [RF006] O sistema deve permitir a seleção de um medicamento para entrega.

Requisitos não funcionais:

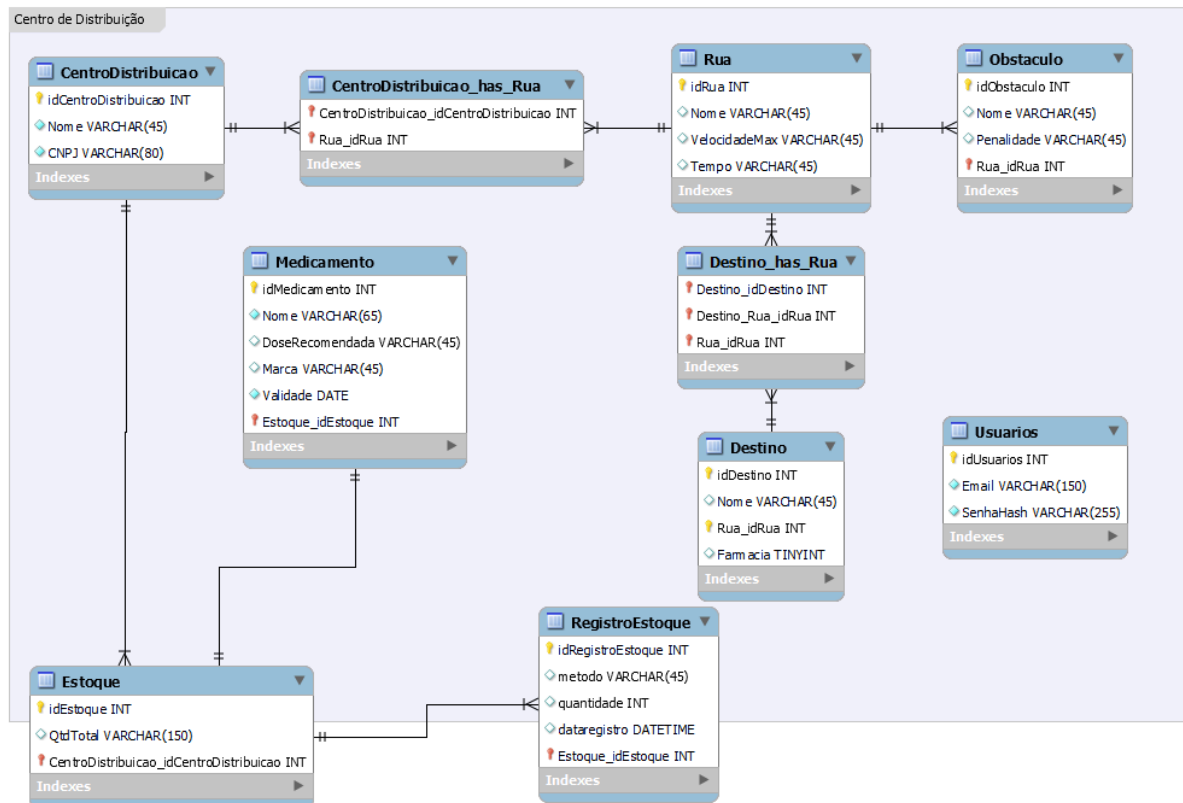
- [RNF001] Deve ser uma plataforma web
- [RNF002] O sistema deve armazenar os dados com integridade
- [RNF003] O sistema deve persistir dados

- [RNF004] O sistema deve armazenar os dados de forma segura
- [RNF005] O sistema deve permitir o acesso aos dados apenas a usuários autorizados

Localização dos requisitos e dos conceitos de programação funcional aplicados no trabalho

- O requisito [RF001], que define que o sistema deve permitir o cadastro de novos medicamentos, foi implementado em `distribuiMais.Client/src/RegisterDrug.jsx` e em `post(self, request)` da classe `MedicamentoView` presente em `distribuiMais.Server/databasegraphproject/projectstructures/views.py`;
- Os requisitos [RF002], [RF003] e [RF004], que define que o sistema deve permitir a leitura, a edição e a deleção de medicamentos, foram implementados em `distribuiMais.Client/src/ManageDrug.jsx` e em `get(self, request, pk = None)`, `put(self, request, pk=None)` e `delete(self, request, pk)` da classe `MedicamentoView` presente em `distribuiMais.Server/databasegraphproject/projectstructures/views.py`;
- Os requisitos [RF005] e [RF006], que definem que o usuário deve ser capaz de selecionar um destino e um medicamento para entrega, respectivamente, foram implementados em `distribuiMais.Client/src/components/Form.jsx`;
- Para contemplar o requisito [RNF001], o trabalho foi desenvolvido utilizando React e Django;
- O requisito [RNF002], que define que o sistema deve ter integridade dos dados, está implementado na função `get(self, request, pk=None)` da classe `MedicamentoView` presente em `distribuiMais.Server/databasegraphproject/projectstructures/views.py`, que permite apresentar na lista de seleção para transporte apenas aqueles medicamentos que não estão com o estoque zerado;
- O requisito [RNF003], que define que o sistema deve ter persistência dos dados, está implementado com o uso de um banco de dados relacional em nuvem. O banco segue o modelo entidade-relacionamento da figura 1 e está definido em `distribuiMais.Server/databasegraphproject/projectstructures/models.py`;

Figura 1: Modelo Entidade-Relacionamento do Centro de Distribuição de Medicamentos Distribui Mais



- O requisito [RNF004], que define que o sistema deve armazenar os dados de forma segura, está implementado na função `post(self, request)` da classe `LoginUserView` presente em `distribuiMais.Server/databasegraphproject/projectstructures/views.py`, que permite encriptar a senha informada pelo usuário no momento da tentativa de login e compará-la com a senha armazenada já encriptada no banco de dados;
- O requisito [RNF005], que define que o sistema deve permitir o acesso aos dados apenas a usuários autorizados, está implementado em `distribuiMais.Client/src/Login.jsx` com a necessidade de realizar login para conseguir acessar o sistema;
- A função lambda recursiva foi implementada em `distribuiMais.Client/src/utills/utills.js` para permitir o acesso à API;
- A função lambda utilizando currying foi implementada na função `put (self, request, pk=None)` da classe `MedicamentoView` em

distribuiMais.Server/databasegraphproject/projectstructures/views.py. Foi utilizada na montagem do serializer para atualizar o registro de um medicamento no banco de dados;

- A list comprehension no escopo de uma função lambda foi implementada na função `get(self, request, pk=None)` da classe `MedicamentoView` presente em `distribuiMais.Server/databasegraphproject/projectstructures/views.py`. É utilizada para adicionar à lista de medicamentos a serem retornados para seleção do usuário apenas aqueles medicamentos que estão no estoque e em quantidade total maior do que zero;
- O dicionário dentro do escopo de uma função lambda foi implementado em `distribuiMais.Client/src/App.jsx` para apresentar o grafo com as rotas disponíveis;
- O functor `map` foi implementado em `distribuiMais.Client/src/components/Select.jsx` para mostrar as opções de destino e de medicamentos ao usuário;
- O monad/função de alta ordem foi implementado na função `cesar` dentro da função `post(self, request)` da classe `LoginUserView` presente em `distribuiMais.Server/databasegraphproject/projectstructures/views.py`. Essa função permite encriptar cada letra da senha apenas se ela tiver sido digitada. Caso o usuário tente fazer login sem digitar uma senha, um erro 400 é retornado;
- A função de alta ordem foi implementada na função `post` em `CentroDistribuicaoView` em `distribuiMais.Server/databasegraphproject/projectstructures/views.py`, permitindo o registro do centro de distribuição na plataforma.