ProjetoMV

Ramicés Moisés do Nascimento
MV

Visão geral

Este documento tem a finalidade de explicar um pouco sobre quais serviços estão disponíveis no front e back-end do projeto, bem como ele foi desenvolvido.

Metodologia

Primeiro, foi definido que os frameworks seriam utilizados no projeto. Os escolhidos foram AngularJS e Bootstrap para o front-end e Play FrameWork para o back-end. Depois, definiu-se um escopo de quantas funcionalidades estariam disponíveis na aplicação, ficando assim:

Front-End: Exibir os dados em forma de lista. Exibir os dados em forma de gráfico.

Back-end: Uma API com um CRUD e pelo menos quatro filtros para os dados.

Com isso, foi necessário buscar algum framework que trabalasse com gráficos da maneira mais simples e clara possível. O escolhido foi o angular-chart, pois ele é desenvolvido para ser usado com o próprio angularJS. Como o angular-char funciona por cima de uma outra biblioteca chamada Chart.js, ela também foi integrada no projeto. Por fim, desenvolveu-se um cronograma de atividades a serem seguidas:

Data	Tarefa
28/03	Planejamento das atividades e início do front-end
29/03	Escrever todo o front-end utilizando dados locais
30/03	Escrever todo o back-end
31/03	Integrar a aplicação
01/03	Revisar o código e escrever toda a documentação

Front-End

O front-end possui três visões diferentes:

Visualizar todos os itens: Todos os itens da base serão visualizados em forma de lista

Visualizar os itens por estado: Podem ser vistos, separados por estados, tanto em forma de lista, como em forma de gráfico.

Visualizar os itens por tipo de gestão: Podem ser vistos, separados por tipos de gestão, tanto em forma de lista, como em forma de gráfico.

Back-end

A API desenvolvida disponibiliza os seguintes serviços:

URL	Descrição	Tipo
/api/getAll	Retorna todos os dados em forma de lista.	GET
/api/add	Recebe um objeto json e o adiciona ao banco. Retorna true, se a operação for concluída com sucesso, e false caso já exista um estabelecimento igual na base	POST
/api/update	Recebe um objeto json e atualiza um objeto na base, caso ele exista. Retorna true se for bem sucedido, e false caso o objeto não exista na base	POST
/api/remove/:CNES	Recebe um código CNES na própria URL e remove o objeto correspondente na base. Se for bem sucedido, retorna true. Se o objeto não existir, retorna false	POST
/api/getCNES/:CNES	Recebe um CNES na própria URL e devolve o estabelecimento correspondente. Se não existir, retorna um objeto vazio	GET
/api/getIBGE/:IBGE	Recebe um código IBGE na própria URL e devolve o estabelecimento correspondente. Se não existir, retorna um objeto vazio	GET

/api/getUF/:UF	Recebe um UF na própria URL e devolve um ArrayList com todos os estabelecimento daquela UF. Se não existir, retorna uma lista vazia	GET
/api/getTipo/:tipo	Recebe um tipo de gestão na própria URL e devolve um ArrayList com todos os estabelecimento daquele tipo. Se não existir, retorna uma lista vazia	GET

Infelizmente, só a primeira foi utilizada com o front-end, pois não houve tempo de integrar os serviços restantes.