Relatório A3 - Sistemas Distribuídos e Mobile

Integrantes:

- Daniela Fernandes Santos de Britto - 1272215981
- Elias Neves Conceição - 12722127604
- Marcelo Victor Copis Sena - 12723110264
- Marlon Barreto Damasceno - 12722114193
- Rafael Almeida Rocha - 1272217730

Apresentando o projeto da matéria de **Sistemas distribuídos e mobile**, referente ao semestre 2023.2, onde seu objetivo é criar uma aplicação que simule a captação de dados de venda de uma rede de lojas.

Para isso nós utilizamos para a aplicação a linguagem **JavaScript** junto com **Node.Js** e para o banco de dados relacional utilizamos o **MySQL**.

Justificativa da escolha da tecnologia:

Optamos por montar nosso sistema utilizando API pois a mesma possui fácil integração e comunicação entre sistemas, além de permitir integração com serviços externos e utilização de tecnologias e/ou frameworks. De modo geral, além de ter bom desempenho e escalabilidade nos possibilitou maior facilidade na manutenção de erros que surgiram ao decorrer do desenvolvimento.

Requerimentos de software necessários para execução da aplicação:

- Node.Js instalado (versão LTS);
- MySQL 8 (Workbench, Server, Router e Shell);

Observações:

1. Para utilizar o banco de dados é necessário importar o arquivo ‘livraria.sql’ que consta na pasta *src* do repositório;
2. Se certificar-se que o serviço ‘MySQL 80’ está em execução;
3. Para fins de compatibilidade e prevenção de erros na conexão do servidor, executar o seguinte comando dentro do MySQL Command Line: "ALTER USER 'root'@'localhost’ IDENTIFIED WITH mysql_native_password BY 'admin';".

Instruções para execução da aplicação

1. Abra o Prompt de Comando dentro da pasta *src* do repositório;
2. Dentro dele execute o comando **npm run dev**;
3. Caso apareçam as mensagens "Servidor rodando!" e "Conectado ao Banco de Dados!" quer dizer que a API já está em funcionamento.

Para utilizar as funções da nossa API utilizaremos o site: https://resttesttest.com.

1. **CRUD Cliente**

   Adicionar cliente (POST): http://localhost:4000/adicionarCliente;

   OBS: necessário adicionar os parâmetros nomeCliente, dataNascimento (8 dígitos), emailCliente e totalPedidos (recomendo iniciar com 0);

   Buscar todos os clientes (GET): http://localhost:4000/todosClientes ;

   Alterar cliente (PUT): http://localhost:4000/alterarCliente/1;

   OBS: o id do cliente que será alterado vai como parâmetro na URL;

   Excluir cliente (DELETE): http://localhost:4000/excluirCliente/1;

   OBS: o id do cliente que será excluído vai como parâmetro na URL;

2. **CRUD Estoque**

   Adicionar cliente (POST): http://localhost:4000/adicionarLivro;

   OBS: necessário adicionar os parâmetros nomeLivro, categoriaLivro, anoLivro, precoLivro, qtdLivro;

   Buscar todos os clientes (GET): http://localhost:4000/todosLivros;

   Alterar cliente (PUT): http://localhost:4000/alterarLivro/1;

   OBS: o id do livro que será alterado vai como parâmetro na URL;

   Excluir cliente (DELETE): http://localhost:4000/excluirLivro/1;

   OBS: o id do livro que será excluído vai como parâmetro na URL;

3. **Pedido de compra**

   Fazer novo pedido (POST): http://localhost:4000/novoPedido;

   OBS: necessário adicionar os parâmetros idCliente(que já esteja cadastrado), idLivro(que já esteja cadastrado), qtdPedido, totalPedido e dataPedido (8 caracteres).

4. **Relatórios estatísticos**

   Produtos mais vendidos (GET): http://localhost:4000/relatorios/maisVendidos;

   OBS: são listados os cinco mais vendidos;

   Produtos por cliente (GET): http://localhost:4000/relatorios/produtoCliente/1;

   OBS: o id do cliente a ser verificado vai como parâmetro na URL;

   Média cons. por cliente (GET): http://localhost:4000/relatorios/mediaCliente/1;

   OBS: o id do cliente a ser verificado vai como parâmetro na URL;

   Prod. com baixo estoque (GET): http://localhost:4000/relatorios/baixoEstoque;

   OBS: são listados os cinco com menor quantidade em estoque.

Apresentação e detalhamento sobre:

- ### Arquitetura:

  Utilizamos na parte do CRUD do cliente e do estoque a ideia da arquitetura de microsserviços, utilizando em arquivos .js separados o banco de dados, as rotas, o controlador e os serviços. Desse modo, toda a API fica mais limpa, fácil de ser otimizada e de caso haja necessidade receber uma manutenção por alguém que nunca tenha tido contato com ela.

  Em toda a nossa API utilizamos também o RESTful, usando operações HTTP para manipular todo o sistema e além disso usamos também um banco de dados relacional (MySQL), garantindo a integridade de todos os dados que são armazenados.

- ### Estratégia:

  Decidimos começar pelo CRUD da nossa API, que era onde demandaria um tempo maior e também maior atenção para que tudo funcionasse de maneira correta. Depois que o mesmo foi finalizado seguimos para a inserção do pedido no banco, parte igualmente importante pois seria através dela que depois seria possível gerar todos os relatórios do sistema.

- **Algoritmos utilizados:**

  Destacam-se em nossa API algoritmos de busca (utilizado sobretudo nos CRUD's) e de filtragem (como os métodos utilizados para gerar os relatórios estatísticos relacionados a venda).

  Vale ressaltar também a utilização dos módulo ***cors*** para otimizar a comunicação com o banco de dados, o ***bodyParser*** utilizado para converter nossas requisições ao formato .json e claro, o ***express*** que é um framework fundamental na construção de API's.