Ficha-Resumo

Atividade 1 da A3 - Sistemas Distribuídos e Mobile

Aluno: Guilherme Amaral Motta

Dentre os padrões de projeto mais utilizados em API's REST NodeJs, pode-se listar **Model View Controller (MVC), Repository Pattern, Service Layer, Factory Pattern, Middleware Pattern, Router Pattern, Data Transfer Object (DTO) e Singleton Pattern.**

O padrão de projeto escolhido para a Lojinha foi o **Model-View-Controller (MVC).** A motivação para a escolha foi principalmente a sua boa separação de responsabilidades, o que facilita a organização, manutenção e escalabilidade do código. O MVC consiste da divisão do código em três camadas interconectadas:

- **Model** consiste na parte lógica da aplicação que gerencia o comportamento dos dados através de regras de negócios, lógica e funções. Aguarda a chamada das funções, que permite o acesso aos dados.
- **View** é a representação dos dados, como uma tabela ou um diagrama. É onde os dados solicitados do Model são exibidos. No caso da API, os dados são representados em forma de JSON.
- **Controller** faz a mediação da entrada e saída, comandando o View e o Model para serem alterados de forma apropriada conforme o usuário solicitou.

Para além do **Model**, **View** e **Controller**, o código também possui um componente **DB** dedicado à conexão ao banco de dados *./db/connection.js* e outro chamado **Routes** dedicado à definição dos endpoints *./routes/routes.js*.

O código se conecta ao seguinte banco de dados MySQL:

```
CREATE DATABASE `loja`

CREATE TABLE `produto` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `nome` varchar(100) NOT NULL,
 `preço` decimal(10,2) NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8
COLLATE=utf8_bin;

CREATE TABLE `cliente` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `nome` varchar(100) NOT NULL,
 `vip` tinyint(1) NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

1. Server.js

Parte do código responsável por criar o servidor. Ele é executado pelo comando:

node server.js

2. Connection.js

Parte do código responsável por realizar a conexão com o banco de dados.

3. Routes.js

Parte do código responsável por definir os endpoints da API e direcionar as requisições para os controladores.

4. Model

Inclui os seguintes arquivos:

- **clienteModel.js**
- **produtoModel.js**

Define os métodos:

- **getAll**: busca todos os dados do banco de dados
- **getById**: busca dados do banco de dados baseando-se no Id
- **create**: adiciona dados no banco de dados
- **update**: atualiza dados no banco de dados
- **remove**: remove dados no banco de dados

5. View

Inclui os seguintes arquivos:

- **clienteView.js**
- **produtoView.js**

Define as funções:

- **view**: formata uma saída
- **viewAll**: formata todas as saídas

6. Controller

Inclui os seguintes arquivos:

- **clienteController.js**
- **produtoController.js**

Utiliza os métodos do Model e retorna uma saída formatada pelo View.