**Padrões usados no projeto:**

**Modelo Cliente-Servidor:** O código segue o modelo cliente-servidor, onde a aplicação (cliente) faz requisições para o servidor (API) e o servidor processa essas requisições e retorna as respostas apropriadas. O cliente pode ser qualquer aplicação que consome a API, enquanto o servidor é representado pelo código fornecido que expõe as rotas e lida com as requisições HTTP.

**Singleton:** O padrão Singleton é aplicado na classe DatabaseService. Ela implementa a criação de uma única instância da conexão com o banco de dados, garantindo que haja apenas uma conexão ativa durante toda a execução do programa.

**Facade**: O controlador receitasController e o controlador userController atuam como fachadas para as operações relacionadas a receitas e usuários, respectivamente. Eles encapsulam a complexidade das interações com outras classes, como a classe DatabaseService, simplificando a interface para o cliente (no caso, as rotas definidas no router). O uso desses controladores permite que o cliente interaja com as funcionalidades de receitas e usuários sem precisar conhecer os detalhes de implementação de cada uma dessas classes.

**Factory Method (parcialmente):** Embora não haja uma implementação explícita do padrão Factory Method, utilizamos um padrão semelhante em algumas partes do código. Por exemplo, a classe DatabaseService possui métodos como criarUsuario e pesquisarUsuario que encapsulam a lógica de criação e pesquisa de usuários no banco de dados. Esses métodos atuam como métodos de fábrica, fornecendo objetos Usuario ou resultados de pesquisa para outras partes do código.