PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

GABRIELA SILVEIRA TRINDADE RODRIGUES

Atividade: Desenvolvimento de uma aplicação backend utilizando NestJS

2024

GABRIELA SILVEIRA TRINDADE RODRIGUES

Atividade: Desenvolvimento de uma aplicação backend utilizando NestJS

Trabalho de Desenvolvimento Backend sobre a utilização do framework Nest JS.

2024

INTRODUÇÃO

A aplicação é um sistema de controle de assinaturas de aplicativos desenvolvida usando o framework NestJS. Ela gerencia três áreas principais:

- **Clientes:** Cadastramento, edição e listagem de clientes.
- **Aplicativos:** Gerenciamento de aplicativos que são oferecidos aos clientes.
- **Assinaturas**: Controle das assinaturas dos clientes aos aplicativos, incluindo a validade e processamento de pagamentos.

A aplicação segue a arquitetura de microserviços, o que garante uma separação clara entre os módulos e facilita sua escalabilidade.

SUMÁRIO

1 HISTÓRIA

O NestJS é um framework Node.js de código aberto, criado por Kamil Myśliwiec em 2017. Ele foi projetado para o desenvolvimento de aplicativos backend escaláveis e modulares, usando TypeScript como linguagem principal. Inspirado por frameworks como Angular, o NestJS adota uma abordagem baseada em decoradores, facilitando a criação de módulos, controladores, e serviços.

Desde sua criação, o NestJS rapidamente ganhou popularidade devido à sua estrutura organizada e escalável. Ele foi projetado para ser altamente extensível, integrando-se facilmente com bibliotecas externas e ferramentas de outras plataformas Node.js, como Express e Fastify.

NestJS é particularmente utilizado em ambientes onde a arquitetura de microserviços é adotada, pois oferece suporte nativo para diversos padrões de comunicação entre serviços, como gRPC, GraphQL, WebSockets, além de eventos assíncronos. Ao longo dos anos, ele se tornou um dos frameworks mais populares para a construção de aplicativos robustos e escaláveis, especialmente no mundo empresarial.o

2 INSTALAÇÃO

A instalação possui um passo-a-passo auto explicativo em seu instalador. As instruções iniciais para instalação do pacote encontra-se no link abaixo:

```
$ npm i -g @nestjs/cli
$ nest new project-name
```

3 MÓDULOS DA APLICAÇÃO

3.1 Módulo ServicoCadastramento

Este módulo gerencia o cadastro e a manipulação de clientes, aplicativos e assinaturas.

Funcionalidades:

- Cadastrar, editar e listar aplicativos.
- Cadastrar, editar e listar clientes.
- Criar e listar assinaturas de clientes.
- Atualizar o valor do custo mensal dos aplicativos.

Endpoints:

- GET /servcad/clientes: Retorna a lista de todos os clientes cadastrados.
- GET /servcad/aplicativos: Retorna a lista de todos os aplicativos cadastrados.
- POST /servcad/assinaturas: Cria uma nova assinatura de um aplicativo por parte de um cliente.
- PATCH /servcad/aplicativos/:id: Atualiza o custo mensal de um aplicativo.
- GET /servcad/assinaturas/:tipo: Retorna assinaturas de acordo com o tipo (ativas, canceladas ou todas).
- GET /servcad/assinaturas/:codcli: Lista todas as assinaturas de um cliente.

3.2 Módulo ServicoPagamentos

Este microserviço é responsável pelo registro e gerenciamento dos pagamentos das assinaturas dos clientes.

Funcionalidades:

- Registrar pagamentos de assinaturas.
- Emitir eventos para informar outros serviços quando um pagamento é realizado.

Endpoints:

- POST /pagamentos/registrarpagamento: Registra um novo pagamento para uma assinatura.

Particularidade:

O módulo ServicoPagamentos utiliza um banco de dados separado, específico para o armazenamento de pagamentos.

3.3 Módulo ServicoAssinaturasValidas

Este microserviço tem como objetivo verificar rapidamente a validade de uma assinatura de cliente. Ele utiliza uma cache interna para otimizar as consultas, e caso a assinatura não esteja na cache, faz uma consulta ao ServicoCadastramento.

Funcionalidades:

- Verificar se uma assinatura é válida ou não.
- Atualizar a cache de validade das assinaturas.
- Limpar a cache quando um pagamento é realizado.

Processo de validação:

1. O serviço consulta a cache interna.
2. Se a assinatura não for encontrada, faz uma consulta ao ServicoCadastramento.
3. Armazena o resultado na cache para consultas futuras.

4 ARQUITETURA

A aplicação é modular e segue a arquitetura de microserviços, facilitando a escalabilidade e manutenção de cada módulo de forma independente.

- **ServicoCadastramento:** Principal módulo que gerencia dados de clientes, aplicativos e assinaturas.
- **ServicoPagamentos:** Microserviço responsável por gerenciar pagamentos e notificar outros serviços.
- **ServicoAssinaturasValidas:** Microserviço de alta performance que valida a assinatura de clientes.

A comunicação entre os microserviços é feita via HTTP e eventos assíncronos, garantindo que a aplicação funcione de maneira eficaz e desacoplada.

5 BANCO DE DADOS

A aplicação utiliza o MySQL para o armazenamento dos dados. A estrutura de bancos de dados está dividida da seguinte forma:

- **Banco de Dados Geral:** Usado pelo módulo ServicoCadastramento para armazenar clientes, aplicativos e assinaturas.
- **Banco de Dados Exclusivo para Pagamentos:** Usado pelo ServicoPagamentos para gerenciar os registros de pagamentos. Cada pagamento realizado é registrado em um banco de dados exclusivo.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 14724*: Informação e documentação – Trabalhos acadêmicos – Apresentação. Rio de Janeiro, 2011.

FAULKNER, William. *Sartoris*. San Diego, California: Harcourt Brace, 1929.

NEST JS. Framework NODEJS: <https://docs.nestjs.com/>. Acesso em 20 de setembro de 2024.

TYPEORM. Managing Queries in TypeORM: <https://typeorm.io/>. Acesso em 20 setembro de 2024.

GLOSSÁRIO

- **NestJS:** Um framework Node.js de código aberto, escrito em TypeScript, projetado para criar aplicativos backend escaláveis e modulares. Ele adota uma arquitetura baseada em módulos e é amplamente utilizado em arquiteturas de microserviços.

- **Microserviços:** Arquitetura de software onde a aplicação é dividida em pequenos serviços independentes, que se comunicam entre si. Cada microserviço é responsável por uma parte específica da aplicação.

- **Cliente:** Entidade que representa o usuário que assina aplicativos. Um cliente pode ter várias assinaturas ativas ou canceladas.

- **Aplicativo:** Entidade que representa o software ou serviço oferecido aos clientes por meio de assinaturas. Cada aplicativo tem um custo mensal e pode ter vários assinantes.

- **Assinatura:** Entidade que representa o vínculo entre o cliente e o aplicativo. A assinatura inclui informações sobre a data de início, fim e status (ativa ou cancelada).

- **Cache:** Mecanismo de armazenamento temporário utilizado para acelerar a resposta de validação de assinaturas. O microserviço ServicoAssinaturasValidas utiliza cache para verificar rapidamente a validade de uma assinatura.

- **Pagamentos:** Operação financeira associada à renovação ou ativação de uma assinatura. O ServicoPagamentos é responsável por registrar os pagamentos realizados pelos clientes.

- **Banco de Dados:** Sistema de armazenamento de dados utilizado para registrar informações sobre clientes, aplicativos, assinaturas e pagamentos. A aplicação utiliza bancos de dados MySQL.

- **Evento Assíncrono:** Mecanismo de comunicação entre microserviços, onde um serviço emite um evento que pode ser processado por outros serviços sem depender diretamente de uma resposta imediata.

- **HTTP:** Protocolo de comunicação utilizado pelos serviços para enviar e receber dados. Os endpoints da aplicação utilizam HTTP para permitir a comunicação entre cliente e servidor.

- **Endpoint:** Ponto de entrada para interações com a aplicação. Um endpoint define uma URL e um método HTTP para acessar ou modificar recursos (como clientes, assinaturas e pagamentos).

- **Módulo:** Unidade organizacional dentro da arquitetura do NestJS. Um módulo pode conter controladores, serviços e repositórios, e representa uma funcionalidade específica do sistema.

- **TypeScript:** Uma linguagem de programação que é um superconjunto de JavaScript, adicionando tipagem estática ao código. O NestJS é desenvolvido em TypeScript para garantir maior robustez e segurança no desenvolvimento de aplicativos.

- **ServicoCadastramento:** Módulo da aplicação responsável por gerenciar o cadastro de clientes, aplicativos e assinaturas.

- **ServicoPagamentos:** Microserviço da aplicação responsável por registrar e gerenciar pagamentos de assinaturas e emitir eventos para notificar outros serviços.

- **ServicoAssinaturasValidas:** Microserviço responsável por validar assinaturas de clientes de forma rápida utilizando cache e consultas ao ServicoCadastramento quando necessário.

- **MySQL:** Sistema de gerenciamento de banco de dados relacional utilizado pela aplicação para armazenar informações de clientes, aplicativos, assinaturas e pagamentos.