Microsserviços – Arquitetura de Aplicações

Thiago Rodrigues

Node.js

- O Node.js é um ambiente de execução de código JavaScript do lado do servidor, de código aberto e multiplataforma.

- **Características principais:**
  - **Assíncrono e orientado a eventos:** O Node.js usa um modelo assíncrono e orientado a eventos, o que significa que ele pode lidar com várias solicitações ao mesmo tempo sem bloquear.
  - **JavaScript do lado do servidor:** Com o Node.js, você pode usar JavaScript para escrever código do lado do servidor, como APIs da web, servidores de aplicativos e ferramentas de linha de comando.
  - **De código aberto e multiplataforma:** O Node.js é de código aberto e gratuito, e pode ser executado em uma variedade de sistemas operacionais, incluindo Windows, macOS e Linux.

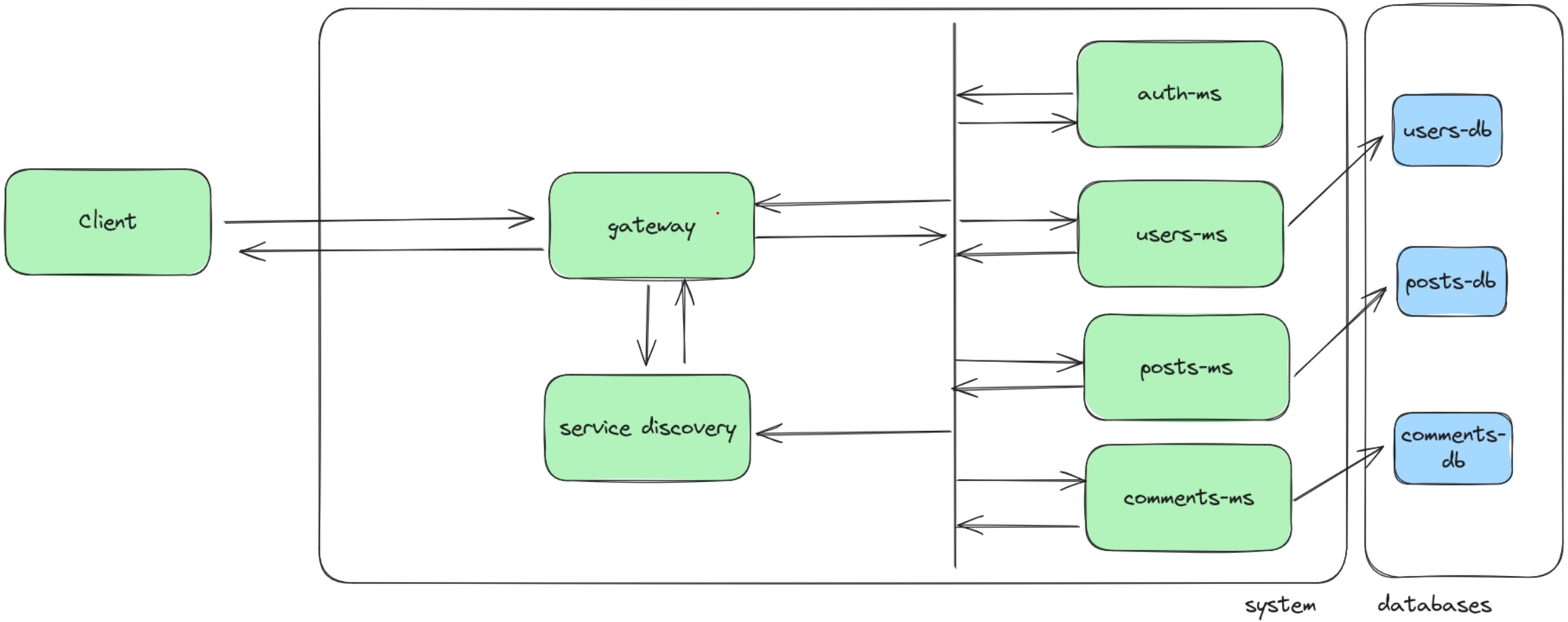
Express

- **Express.js**, também conhecido simplesmente como Express, é uma estrutura popular para a construção de aplicações web em Node.js. É conhecido por ser:
  - **Leve:** O Express fornece um conjunto essencial de recursos para lidar com a funcionalidade do servidor web sem ser intrusivo. Você pode personalizá-lo facilmente com bibliotecas adicionais.
  - **Flexível:** O Express não o força a usar ferramentas ou bibliotecas específicas. Você tem a liberdade de escolher o que funciona melhor para o seu projeto.
  - **Minimalista:** O Express se concentra em fornecer o essencial para a construção de aplicativos da web. Você pode adicionar funcionalidades como modelos ou acesso a banco de dados por meio de middleware.

Express

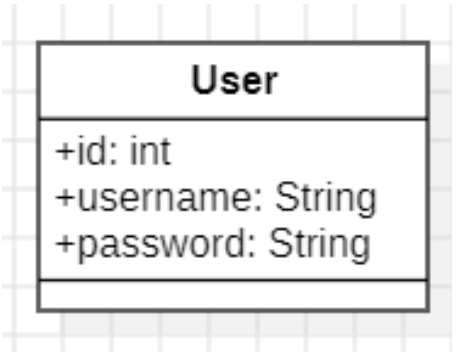
- **Aspecto Chave**
  - **Roteamento:** O Express permite definir rotas para lidar com diferentes solicitações HTTP (GET, POST, etc.) para URLs específicas em seu aplicativo.
  - **Middleware:** As funções de middleware são usadas para processar solicitações e respostas antes que cheguem ao manipulador final. Isso permite tarefas como registro, autenticação e análise de dados.

Arquitetura do Sistema



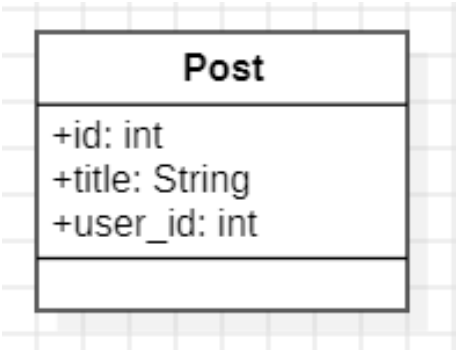
Desenvolvimento da API

- **Funcionalidades Serviço users-ms**
  - **Autenticáveis**
    - findAll – listar todos
    - save - criar
    - findByUserName – exibir pelo username



Desenvolvimento da API

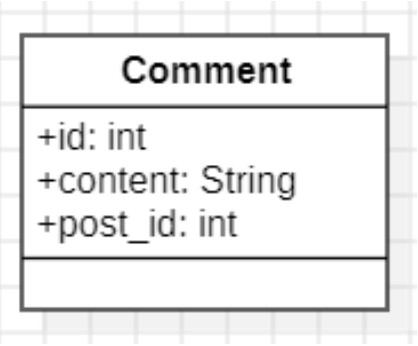
- **Funcionalidades Serviço posts-ms**
  - **Autenticáveis**
    - findAll – listar todos
    - save – criar
    - findByUserId – buscar os posts por um id de usuário



- Passos
  - Inicializar um projeto node.js (npm init –y)
  - Configurar package.json
  - Instalar dependências
    - npm i ...
    - npm i -D ...
  - Começar implementação
  - Banco de dados posts_db
  - Porta 5100

Desenvolvimento da API

- **Funcionalidades Serviço comments-ms**
  - **Autenticáveis**
    - findAll – listar todos
    - save - criar



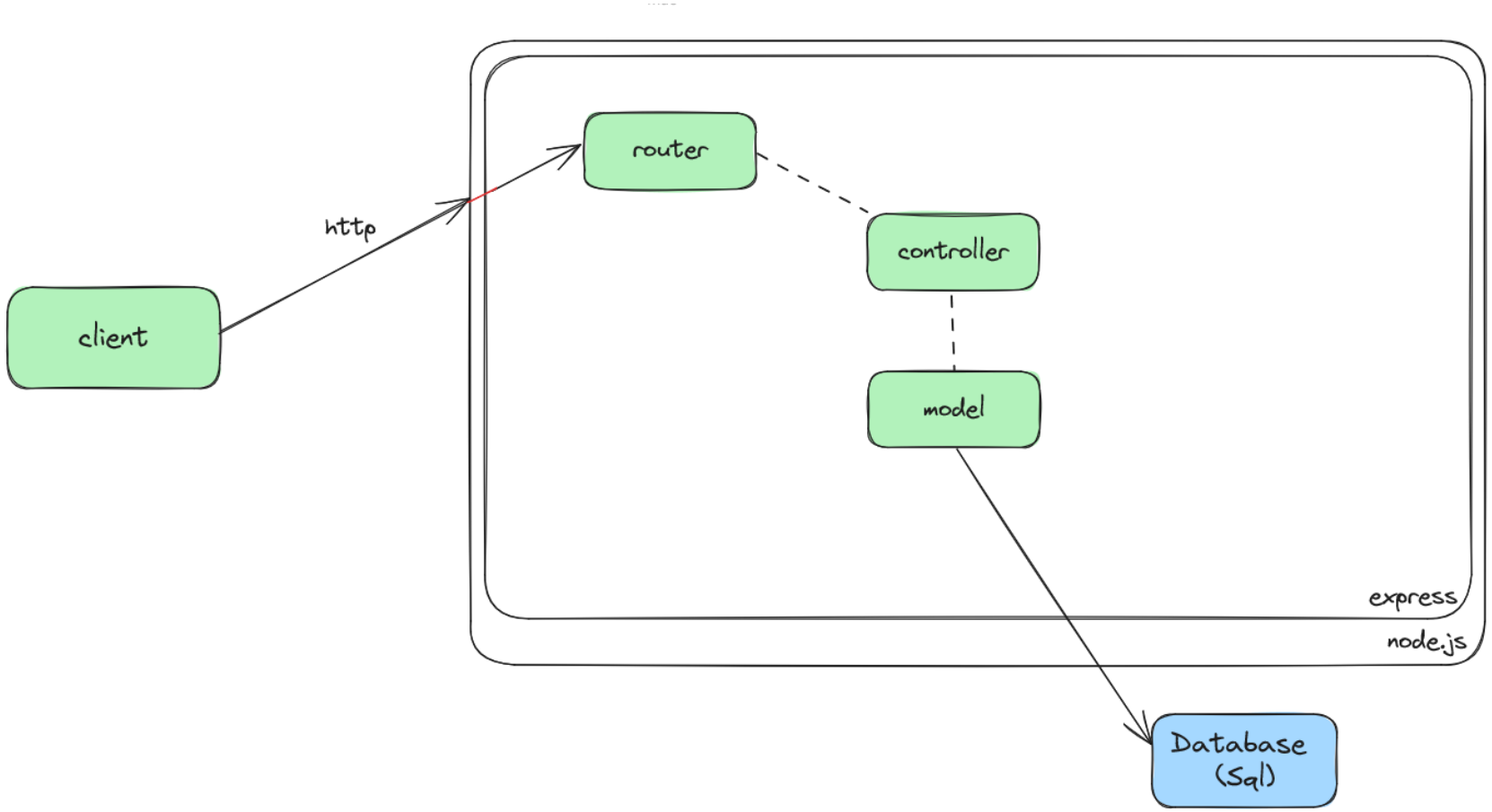
Desenvolvimento da API

- **Funcionalidades Serviço auth-ms**
  - **Autenticáveis**
    - login

Arquitetura do Projeto

- **Arquitetura MVC**
  - Um padrão de design para organizar aplicações em 3 camadas:
    - **Modelo:** Armazena e manipula dados.
    - **Visão:** Exibe dados para o usuário.
    - **Controlador:** Faz a ponte entre modelo e visão.

Arquitetura dos Microsserviços



Gerenciados de dependência

- O **npm**, que significa **Node Package Manager**, é um gerenciador de pacotes crucial para o desenvolvimento com **Node.js**.
- Ele funciona como um repositório gigante de software, onde você encontra diversos módulos e bibliotecas pré-escritos para facilitar seu trabalho.
- **Principais funções do npm**
  - Instalar pacotes
  - Gerenciar dependências
  - Publicar pacotes
  - Gerenciar versões

Dependências

- **Express**
- **Bcrypt**
  - O Bcrypt (Blowfish Crypt) é um algoritmo de hash usado para armazenar senhas de forma segura em aplicações web e desktop. Ele se destaca por sua **alta segurança e robustez contra ataques de força bruta**, tornando-o uma escolha popular para proteger dados sensíveis.

- **Axios**
  - Axios é uma biblioteca JavaScript usada para fazer requisições HTTP a partir do navegador e do Node.js. É bastante popular entre desenvolvedores web por sua simplicidade e facilidade de uso.

Dependências

- **JsonWebToken**
  - É um **pacote popular para o Node.js** que facilita a criação e validação de **tokens JSON Web (JWT)**. Esses tokens são utilizados para implementar **autenticação e autorização em aplicações web e APIs**.
- **Dotenv**
  - O **dotenv** é uma ferramenta popular para gerenciar variáveis de ambiente em aplicações Node.js. Ele permite que você armazene informações de configuração confidenciais, como chaves de API, senhas e credenciais de banco de dados, em um arquivo separado chamado .env e, em seguida, carregue essas variáveis em seu código Node.js.
- **Nodemon (Developer)**
  - O Nodemon é uma **ferramenta de desenvolvimento para Node.js** que **monitora as mudanças em seus arquivos de código e reinicia automaticamente o servidor quando necessário**.

Dependências

- **Mysql2**
  - Mysql2 é uma biblioteca Node.js que fornece uma interface para interagir com bancos de dados MySQL. É uma versão mais rápida e eficiente do popular módulo mysql com suporte a Promises e async/await, o que facilita o desenvolvimento de aplicações assíncronas.

- **Sequelize**
  - O Sequelize é um ORM (Object-Relational Mapping) para Node.js, que facilita a interação com bancos de dados relacionais, como MySQL, PostgreSQL, SQLite, e outros. Ele oferece uma interface baseada em Promises para realizar operações CRUD (Create, Read, Update, Delete) e manipular esquemas de banco de dados usando JavaScript.

Dependências

- http-proxy-middleware
  - É uma biblioteca Node.js que facilita a criação de proxies HTTP para redirecionar ou interceptar requisições HTTP. É frequentemente utilizada em projetos de desenvolvimento web para criar proxies reversos, permitindo que os desenvolvedores redirecionem solicitações de um servidor para outro, manipulem cabeçalhos de requisições, e modifiquem respostas antes que elas cheguem ao cliente.

Roteiro Exercício

- Cria uma aplicação Express e configura o middleware para parsing de JSON.
- Define o endpoint /login que recebe requisições POST.
- Extrai username e password do corpo da requisição.
- Tenta obter os dados do usuário usando o serviço de usuário (definido por USER_SERVICE_URL no ambiente).
- Verifica se o usuário existe e se a senha fornecida corresponde à senha armazenada (usando bcrypt para comparação).
- Se a autenticação falhar, retorna uma resposta 401 (não autorizado) com uma mensagem de erro.
- Se a autenticação for bem-sucedida, gera um token JWT com um payload contendo o ID do usuário e uma expiração de 1 hora. (SECRET_KEY)
- Retorna o token no corpo da resposta.
- Em caso de qualquer erro durante o processo, retorna uma resposta 500 (erro interno do servidor) com uma mensagem de erro.

Roteiro Exercício

- Middleware de autenticação (authMiddleware)
  - Importação de jwt: Importa o módulo jsonwebtoken para verificar o token JWT.
  - Extração do Token: Obtém o cabeçalho de autorização da solicitação e extrai o token, que deve estar no formato Bearer <token>.
  - Verificação de Presença do Token: Se o token não estiver presente, retorna um status 401 (Não Autorizado) com uma mensagem de erro.
  - Verificação do Token: Usa jwt.verify para validar o token com a chave secreta (SECRET_KEY).
  - Se houver um erro durante a verificação (por exemplo, token expirado ou inválido), retorna um status 403 (Proibido) com uma mensagem de erro.
  - Se a verificação for bem-sucedida, anexa o usuário ao objeto req e chama next() para passar o controle para o próximo middleware ou rota.

Roteiro Exercício

- Registrar o serviço de autenticação no service registry
- Habilitar autenticação via token nas rotas de usuários e posts.
  - Não habilitar na rota de /users/:username
- O Middleware de autenticação deve ser criado em cada microserviço.