Manual de Instalação e Uso

Sistema de Clínica Veterinária(VetFofinhos)

João Pedro Amorim / 2320307
João Victor Lopes de Carvalho / 2320335

Junho de 2025

Sumário

1 Visão Geral do Projeto

Este manual tem como objetivo guiar você, passo a passo, na instalação e configuração dos ambientes de desenvolvimento para os nossos backends. Ele foi feito para quem está começando, explicando tudo do zero para que você consiga rodar as aplicações localmente.

Nossos sistemas são compostos por diversos **microserviços** (pequenos backends independentes que se comunicam entre si) que gerenciam diferentes partes da lógica de negócios.

1.1 Tecnologias Utilizadas

Para você ter uma ideia geral, vamos trabalhar com:

- **Backends Java:** Usamos o framework **Spring Boot** para construir a maioria dos nossos serviços.
- **Backend Node.js:** Um dos nossos serviços é feito com **Node.js** e o framework **Express**, usando **TypeScript** para um código mais organizado.
- **Banco de Dados:** Todos os nossos backends interagem com bancos de dados **MySQL**.
- **Ferramenta de Edição de Código:** Usaremos o **VS Code**, um editor super popular e fácil de usar.
- **Gerenciamento de Banco de Dados:** O **MySQL Workbench** nos ajuda a visualizar e interagir com o banco de dados.
- **Descoberta de Serviços:** Utilizamos o **Netflix Eureka Client** (nos Java) e **eureka-js-client** (no Node.js) para que nossos microserviços se encontrem na rede.
- **Balanceamento de Carga:** O **Spring Cloud LoadBalancer** (nos Java) ajuda a distribuir as requisições entre as instâncias dos serviços.

—

2 Instalação das Ferramentas Essenciais

Antes de mexer no código, precisamos instalar algumas ferramentas básicas no seu computador.

2.1 Instalando o VS Code (Visual Studio Code)

O VS Code é o "caderno"onde vamos escrever e organizar todo o nosso código.

1. **Acesse o Site:** Abra seu navegador e vá para o site oficial do VS Code: `https://code.visualstu`
2. **Baixe o Instalador:** Clique no botão de **Download** (geralmente azul, bem visível). Ele vai detectar seu sistema operacional (Windows, macOS, Linux) e te dar a versão correta.
3. **Execute o Instalador:**

- No Windows, execute o arquivo `.exe` que você baixou.
- Siga os passos do assistente de instalação. É importante que, na tela de **"Tarefas Adicionais"** (ou similar), você **marque a opção para adicionar o VS Code ao "PATH"**. Isso facilita muito a vida depois.
- Continue clicando em "Avançar"ou "Instalar"até finalizar.

Pronto! Agora você tem o VS Code instalado.

2.2 Instalando o Java Development Kit (JDK) - Para os Backends Java

O Java é a linguagem que usamos em 3 dos nossos 4 backends. Para que o código Java funcione, precisamos do **JDK** (Java Development Kit), que é como um "pacote de ferramentas"para desenvolver em Java.

1. **Acesse o Site:** Vá para a página de download do OpenJDK (uma versão gratuita e muito usada do Java): `https://jdk.java.net/`
2. **Baixe a Versão LTS:** Procure pela versão **LTS (Long-Term Support)** mais recente. No momento, a versão **JDK 17 LTS** ou **JDK 21 LTS** são boas opções. Baixe a versão compatível com seu sistema operacional (Windows x64 Installer, macOS x64 DMG, etc.).
3. **Execute o Instalador:**
   - Siga os passos do assistente. Geralmente, as opções padrão são suficientes.
   - **Importante:** O instalador do JDK geralmente configura as variáveis de ambiente automaticamente. Se por acaso ele perguntar sobre "PATH"ou "JAVA_HOME", deixe as opções padrão marcadas para que ele faça isso.
4. **Verifique a Instalação:**
   - Abra o **Terminal** (no macOS/Linux) ou o **Prompt de Comando/PowerShell** (no Windows).
   - Digite o seguinte comando e pressione Enter:

     ```
     java --version

     ```

   - Você deverá ver a versão do Java instalada. Se aparecer algo como "command not found"ou um erro, pode ser que as variáveis de ambiente não foram configuradas corretamente. Nesse caso, peça ajuda!

2.3 Instalando o Node.js e npm - Para o Backend Node.js

O Node.js é o ambiente onde nosso backend em Node.js roda. O **npm** é o gerenciador de pacotes do Node.js, que usamos para baixar todas as "bibliotecas"e "dependências"que o projeto precisa.

1. **Acesse o Site:** Vá para a página oficial do Node.js: `https://nodejs.org/`

2. **Baixe a Versão LTS:** Recomenda-se baixar a versão **LTS (Long-Term Support)**, que é a mais estável e recomendada para a maioria dos projetos. Clique no botão correspondente.
3. **Execute o Instalador:**
   - Siga as instruções do assistente de instalação.
   - Mantenha as opções padrão, elas já vêm configuradas para instalar o Node.js e o npm, e adicionar ao "PATH"do sistema.
4. **Verifique a Instalação:**
   - Abra um **novo** Terminal (ou Prompt de Comando/PowerShell) – é importante que seja um novo para que ele reconheça as mudanças nas variáveis de ambiente.
   - Execute os seguintes comandos e pressione Enter após cada um:

```
node -v
npm -v

```

   - Você deverá ver as versões do Node.js e do npm impressas no terminal.

2.4 Instalando o MySQL Workbench

O MySQL Workbench é uma ferramenta visual que nos ajuda a "enxergar"e interagir com o banco de dados MySQL de forma mais fácil.

1. **Acesse o Site:** Vá para a página oficial de downloads do MySQL Workbench: https://dev.mysql.com/downloads/workbench/
2. **Baixe e Instale:** Siga as instruções do site para baixar a versão correta para seu sistema operacional e execute o instalador. As opções padrão geralmente funcionam bem.

—

3 Configurando o Ambiente do Projeto

Agora que temos as ferramentas, vamos preparar seu computador para rodar os projetos.

3.1 Obtendo o Código do Projeto

Você deve ter recebido os códigos-fonte dos backends, provavelmente em um arquivo `.zip` ou através de um repositório Git.

- **Se for um `.zip`:** Extraia o conteúdo do arquivo para uma pasta em seu computador onde você queira guardar seus projetos (ex: `C:\Projetos\meu-trabalho` ou `~/Projetos/meu-trabalho`).
- **Se for Git:** Você precisará de um cliente Git instalado (o VS Code tem um bom suporte). Se você não tem, é um bom momento para instalar o Git: https://git-scm.com/downloac Depois, use o comando `git clone [URL_DO_REPOSITORIO]` no terminal para baixar o código.

3.2 Abrindo o Projeto no VS Code

1. Abra o **VS Code**.
2. Vá em `File` (Arquivo) no menu superior.
3. Selecione `Open Folder...` (Abrir Pasta...).
4. Navegue até a pasta onde você extraiu ou clonou os projetos (ex: `C:\Projetos\meu-trabalho`) e selecione-a.
5. O VS Code pode perguntar se você "confia nos autores"da pasta. Clique em **"Yes, I trust the authors"** (Sim, confio nos autores) para que ele habilite todas as funcionalidades.

Agora você verá os arquivos do projeto no painel lateral esquerdo do VS Code. Você deve ver algo como:

```
backend-java-1/
backend-java-2/
backend-java-3/
backend-node/
```

3.3 Instalação de Extensões Essenciais no VS Code

As extensões adicionam superpoderes ao VS Code! Elas nos ajudam a escrever código, debugar e entender melhor o que está acontecendo.

1. No VS Code, clique no ícone de **Extensões** na barra lateral esquerda (parece com quatro quadradinhos, um deles separado, ou pressione `Ctrl+Shift+X`).
2. Na barra de busca, digite o nome das extensões abaixo e clique em **"Install"** (Instalar) em cada uma:
   - **Para os Backends Java:**
     - **Extension Pack for Java:** Este pacote instala várias extensões úteis de uma vez só para Java, incluindo suporte a Spring Boot.
     - **Spring Boot Tools:** Ferramentas específicas para o Spring Boot que ajudam a autocompletar e navegar no código. **Observação:** É provável que o ‘Extension Pack for Java‘ já instale o suporte a Maven e Debugger, mas se algo não funcionar, verifique se elas estão separadas.
     - **Maven for Java:** (Se os projetos Java usarem Maven) Ajuda a gerenciar as dependências do projeto.
     - **Debugger for Java:** Essencial para "debugar"(encontrar e corrigir erros) seu código Java.
   - **Para o Backend Node.js / TypeScript:**
     - **Prettier - Code formatter:** Ajuda a manter seu código Node.js/TypeScript sempre formatado de forma padrão e bonita.
     - **ESLint:** Ajuda a identificar problemas de código e seguir boas práticas no JavaScript/TypeScript.
     - **Postman:** Uma extensão super útil para testar as requisições para a sua API. (Veremos mais sobre ela depois)

3.4 Conectando-se ao Banco de Dados MySQL com o Workbench

Nosso banco de dados principal está em um servidor na nuvem. Você não precisa criá-lo, apenas se conectar a ele para poder ver os dados e entender como as coisas funcionam.

Informações de Conexão do Banco de Dados:

- **URL de Conexão:** `mysql://root:hiqMIcmcicZQPCPGRsAMMzIqLqFjYVeY@tramway.proxy.rlwy`
- **Hostname:** `tramway.proxy.rlwy.net`
- **Porta:** `28655`
- **Usuário:** `root`
- **Senha:** `hiqMIcmcicZQPCPGRsAMMzIqLqFjYVeY`
- **Schema (Banco de Dados Padrão):** `railway`

Passos para Conectar no MySQL Workbench:

1. Abra o **MySQL Workbench**.
2. Na tela inicial, clique no ícone + ao lado de "MySQL Connections"para criar uma nova conexão.
3. Preencha os campos da nova conexão:

   Connection Name: Dê um nome fácil de lembrar, por exemplo, `Banco Railway do Projeto`

   Hostname: `tramway.proxy.rlwy.net`

   Port: `28655`

   Username: `root`

   Password: Clique no botão "Store in Vault..."(Armazenar no Cofre...), insira a senha `hiqMIcmcicZQPCPGRsAMMzIqLqFjYVeY` e clique em "OK".

   Default Schema: `railway`
4. Clique em **"Test Connection"** (Testar Conexão). Se tudo estiver certo, você verá uma mensagem de sucesso. Se não, revise os dados que você digitou.
5. Clique em **"OK"** para salvar a conexão.

Agora você pode clicar na conexão que acabou de criar na tela inicial do MySQL Workbench para abrir o banco de dados e explorar suas tabelas.

—

4 Configurando e Rodando os Backends

Agora, vamos configurar e iniciar cada tipo de backend. Lembre-se que cada backend (Java e Node.js) vai precisar do seu próprio terminal no VS Code.

4.1 Configurando e Rodando os Backends Java (Spring Boot)

Temos 3 backends em Java. Eles compartilham a mesma forma de configuração e execução.

4.1.1 Dependências Comuns do Spring Boot

Todos os nossos projetos Spring Boot utilizam as seguintes dependências em seus arquivos `pom.xml` (que é o arquivo que gerencia as dependências Java):

- `spring-boot-starter-web`: Para criar APIs web.
- `spring-boot-starter-data-jpa`: Para interagir com o banco de dados usando JPA.
- `mysql-connector-j`: O "driver"para conectar ao MySQL.
- `lombok`: Uma biblioteca que reduz a quantidade de código que precisamos escrever.
- `spring-boot-starter-test`: Para escrever testes (algo que você fará no futuro!).
- `spring-boot-starter-validation`: Para validar os dados que chegam na nossa API.
- `spring-cloud-starter-netflix-eureka-client`: Para que os serviços se "descobrem"entre si.
- `spring-boot-starter-webflux`: Para construir aplicações reativas (alguns dos nossos serviços podem usar isso).
- `spring-cloud-starter-loadbalancer`: Para balancear as requisições entre os serviços.
- `spring-boot-starter-actuator`: Para monitorar e gerenciar a aplicação.

Você não precisa instalar essas dependências manualmente; o Maven (ferramenta de build do Java) fará isso automaticamente quando você pedir para "construir"o projeto.

4.1.2 Configuração de Ambiente (`application.properties` ou `application.yml`)

Cada backend Java terá um arquivo de configuração, geralmente chamado `application.properties` ou `application.yml`, dentro da pasta `src/main/resources`. Nesses arquivos, precisamos configurar a conexão com o banco de dados e outras informações:

1. Para **cada um** dos seus backends Java (ex: `backend-java-1`, `backend-java-2`, `backend-java-3`), navegue até a pasta `src/main/resources`.
2. Localize o arquivo `application.properties` ou `application.yml`.
3. **Copie e cole** o seguinte conteúdo, ajustando o nome do banco de dados (database name) se for diferente para cada microserviço (se todos usarem `railway`, pode manter):

```
# Configura  es do Banco de Dados MySQL
spring.datasource.url=jdbc:mysql://tramway.proxy.rlwy.net
   :28655/railway?createDatabaseIfNotExist=true&useSSL=false
   &allowPublicKeyRetrieval=true
spring.datasource.username=root
spring.datasource.password=hiqMIcmcicZQPCPGRsAMMzIqLqFjYVeY
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Configura  es do JPA (para o Hibernate)
spring.jpa.hibernate.ddl-auto=update # Cuidado! Em
   produ  o , geralmente    'none' ou 'validate'
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

# Configura  es do Eureka Client (para descoberta de
   servi os)
eureka.client.serviceUrl.defaultZone=http://localhost:8761/
   eureka/ # Se tiver um servidor Eureka rodando localmente
eureka.client.fetchRegistry=true
eureka.client.registerWithEureka=true

# Porta em que o servi o vai rodar (ex: 8080, 8081, 8082 -
   cada servi o em uma porta diferente!)
server.port=8080 # Troque esta porta para cada backend Java
   para evitar conflito!
```

Atenção: Mude a linha `server.port=` para que cada backend Java rode em uma porta **diferente**. Por exemplo:

- `backend-java-1`: `server.port=8080`
- `backend-java-2`: `server.port=8081`
- `backend-java-3`: `server.port=8082`

4.1.3 Como Rodar os Backends Java

Para cada backend Java que você quiser iniciar:

1. Abra um **novo Terminal** no VS Code (`Ctrl+‘^‘` ou `View > Terminal`).
2. Navegue até a pasta do backend Java que você quer rodar. Por exemplo:

```
cd backend-java-1

```

3. **Construa e Execute o Projeto:**
   - Se o projeto usa **Maven** (o que é comum com Spring Boot), execute:

```
./mvnw spring-boot:run

```

   (No Windows, pode ser `.\mvnw.cmd spring-boot:run`)

- Ou, se você tem o Maven instalado globalmente:

```
mvn spring-boot:run

```

- Se preferir apenas compilar e depois rodar o arquivo JAR:

```
./mvnw clean install
java -jar target/[nome-do-seu-arquivo].jar # Troque pelo
    nome real do JAR

```

4. Aguarde até ver a mensagem de que o Spring Boot iniciou, geralmente indicando a porta em que o serviço está rodando, por exemplo: `Started ApplicationNameApplication in X.X seconds (JVM running for Y.Y)` e a porta `Tomcat started on port(s): 8080 (http)`.

Repita esses passos para cada um dos 3 backends Java, cada um em um terminal diferente.

4.2 Configurando e Rodando o Backend Node.js (Express com TypeScript)

Nosso backend Node.js é o `microservicevenda`.

4.2.1 Dependências do Node.js

O projeto Node.js tem um arquivo `package.json` que lista todas as dependências que ele precisa. As principais são:

- `express`: O framework web para criar a API.
- `typescript`: Para escrever o código Node.js com TypeScript.
- `sequelize`: Um "ORM"para interagir com o banco de dados MySQL de forma mais fácil.
- `mysql2`: O driver para conectar ao MySQL.
- `dotenv`: Para gerenciar as variáveis de ambiente.
- `bcrypt`, `cors`, `helmet`, `jsonwebtoken`: Para segurança e funcionalidades web.
- `eureka-js-client`: Para descoberta de serviços, similar ao Eureka do Java.
- `pdfkit`: Para geração de PDFs.

4.2.2 Configuração de Ambiente (`.env`)

O backend Node.js usa um arquivo chamado `.env` (dotenv) para suas configurações sensíveis, como a conexão com o banco de dados.

1. Dentro da pasta do backend Node.js (ex: `backend-node`), crie um novo arquivo chamado `.env`.
2. **Copie e cole** o seguinte conteúdo para dentro desse arquivo `.env`:

```
# server/.env
DB_HOST=tramway.proxy.rlwy.net
DB_USER=root
DB_PASS=hiqMIcmcicZQPCPGRsAMMzIqLqFjYVeY
DB_NAME=railway
DB_PORT=28655
PORT=3000 # Porta em que este servi o Node.js vai rodar
JWT_SECRET=slaide # Chave secreta para tokens JWT
```

4.2.3 Instalando as Dependências do Node.js

Antes de rodar o backend Node.js, precisamos instalar todas as dependências listadas no `package.json`.

1. Abra um **novo Terminal** no VS Code (`Ctrl+‘^‘` ou `View > Terminal`).
2. Navegue até a pasta do backend Node.js:

```
cd backend-node

```

3. Execute o comando para instalar as dependências:

```
npm install

```

   Isso pode levar alguns segundos. Ao final, você verá uma mensagem de sucesso ou um resumo da instalação.

4.2.4 Como Rodar o Backend Node.js

Depois de instalar as dependências, você pode iniciar o serviço Node.js.

1. Certifique-se de que você ainda está no terminal na pasta `backend-node`.
2. Execute o comando para iniciar o servidor:

```
npm start

```

3. Aguarde até ver a mensagem de que o servidor está rodando, algo como: `Servidor rodando em http://localhost:3000` ou similar.

—

5 Testando a API com a Extensão Postman do VS Code

Para ter certeza de que seus backends estão funcionando e respondendo corretamente, vamos usar a extensão **Postman** dentro do próprio VS Code.

5.1 Instalar e Fazer Login na Extensão Postman

1. Se você ainda não instalou, vá para a aba de **Extensões** no VS Code (`Ctrl+Shift+X`).
2. Procure por "Postman"e clique em **"Install"**.
3. Após a instalação, um novo ícone do Postman aparecerá na barra lateral esquerda do VS Code (parece um "P"com um raio). Clique nele.
4. Você será solicitado a fazer login ou criar uma conta. É grátis e simples; faça login para sincronizar suas requisições.

5.2 Criar uma Nova Requisição HTTP

1. Na aba do Postman no VS Code, clique em **"New HTTP Request"** (Nova Requisição HTTP).

5.3 Testando o Login (Exemplo com Backend Node.js)

Vamos usar o backend Node.js (`http://localhost:3000`) para simular um login e obter um token.

1. **Configure a Requisição:**
   - Selecione o método `POST`.
   - No campo de URL, digite: `http://localhost:3000/login` (ou a URL de login específica do seu backend Node.js).
2. **Corpo da Requisição (Body):**
   - Clique na aba **"Body"**.
   - Selecione o tipo `raw` e, no dropdown ao lado, escolha `JSON`.
   - No campo de texto, insira um JSON com as credenciais de login (ex: usuário e senha). Pergunte ao seu colega ou líder quais credenciais de teste usar, ou se o sistema permite registro. Um exemplo seria:

```
{
    "username": "seu_usuario_teste",
    "password": "sua_senha_teste"
}
```

3. **Cabeçalhos da Requisição (Headers):**
   - Adicione um novo header:

- **Key:** `Content-Type`
- **Value:** `application/json`

4. Clique no botão **"Send"** (Enviar).
5. **Verifique a Resposta:** Na parte inferior da tela, você verá a resposta da API. Se o login for bem-sucedido, você provavelmente receberá um **token JWT** (uma sequência longa de caracteres) no corpo da resposta. **Copie esse token!**

5.4 Testando uma Requisição Autenticada (Exemplo com Backend Node.js)

Agora, vamos usar o token que obtivemos para acessar uma rota que exige autenticação.

1. Crie uma **nova requisição HTTP** no Postman.
2. **Configure a Requisição:**
   - Selecione o método `GET`.
   - No campo de URL, digite: `http://localhost:3000/usuarios` (ou uma URL de alguma rota que liste usuários no seu backend Node.js que exija autenticação).
3. **Autorização (Authorization):**
   - Clique na aba **"Authorization"**.
   - No dropdown "Type", selecione `Bearer Token`.
   - No campo "Token", **cole o token JWT** que você copiou do passo anterior.
4. Clique no botão **"Send"**.
5. **Verifique a Resposta:** Você deverá ver a lista de usuários ou os dados que a API retornar para essa rota, indicando que a autenticação funcionou.

—

6 Diagramas do Sistema

Para te ajudar a entender melhor como os dados se relacionam e como as classes são organizadas, incluímos alguns diagramas.

6.1 Diagrama Entidade-Relacionamento (DER)

Este diagrama mostra as tabelas do banco de dados e como elas se conectam entre si. Cada "caixa"é uma tabela, e as linhas indicam as relações.

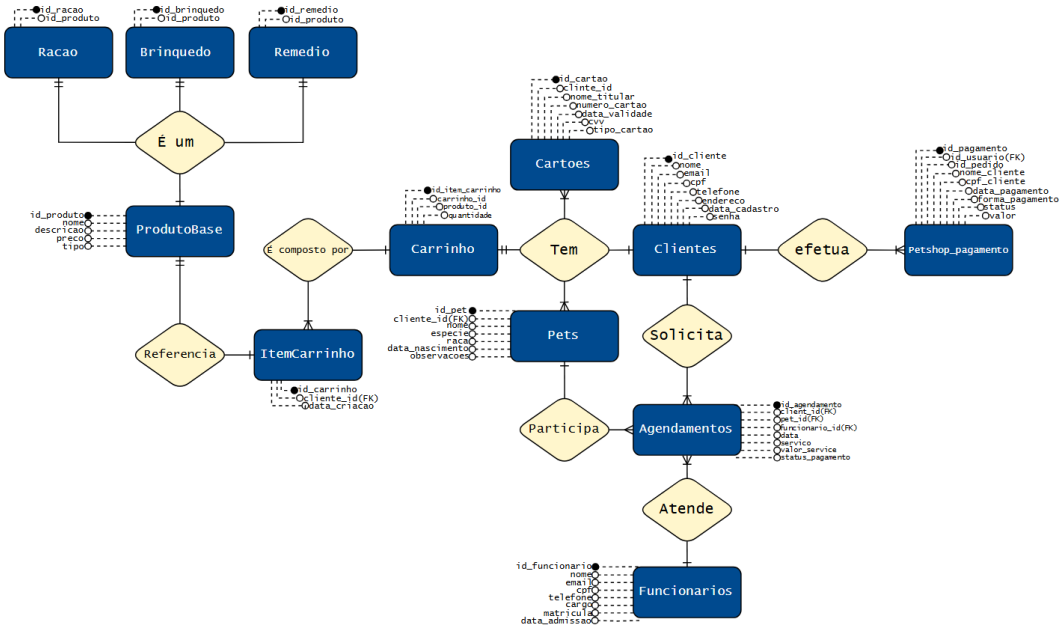

Figura 1: Diagrama Entidade-Relacionamento dos Backends.

6.2 Diagrama de Classes UML

Este diagrama mostra as principais classes (pedaços do código) e como elas se relacionam, dando uma visão mais "abstrata"da estrutura do sistema.

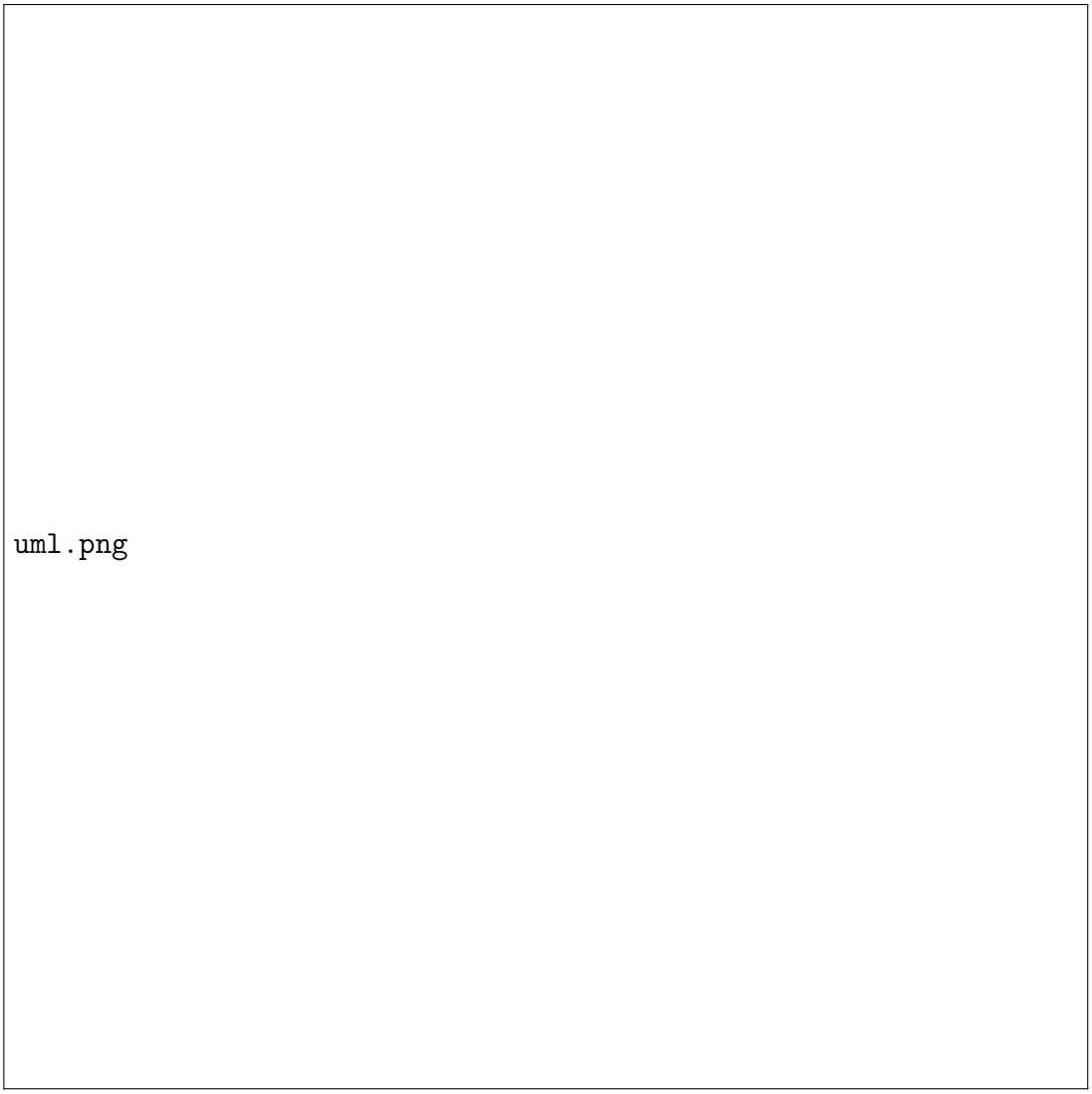

Figura 2: Diagrama de Classes UML (simplificado).

—

7 Considerações Finais

Parabéns! Você chegou ao final do guia de instalação. Se seguiu todos os passos, seus backends Java e Node.js devem estar rodando e prontos para serem testados.

Lembre-se:

- Se tiver qualquer problema, **não hesite em pedir ajuda**. Seu colega ou líder de equipe está aqui para te auxiliar.
- Comece explorando o código, entenda as pastas principais e como os arquivos se organizam.

Qualquer dúvida, é só chamar! Estamos aqui para te dar todo o suporte.