
otterwise

Backend

Backend

- Implementam a regra de negócio de uma aplicação;
- O backend recebe dados para serem armazenados em algum banco de dados;
- Atualmente são implementados em forma de API, podendo ser escritos com qualquer linguagem de programação para backend, pois a comunicação se dá por um padrão que o front-end consegue entender. Por exemplo, o padrão REST.

Fonte: https://www.totvs.com/blog/developers/back-end/

Fastify

Para criar um servidor backend em NodeJS vamos utilizar o Fastify, que é um framework rápido e fácil de implementar.

Fonte: https://www.fastify.io/

Exemplo

- Implementar o exemplo de “hello world” da documentação do fastify
- Criar a primeira rota de GET

Banco de dados

SQL vs NoSQL

- SQL (Structured Query Language)
  - Dados altamente estruturados e podem ser relacionados via tabelas;
  - Os dados são guardados em tabelas;
  - Cumpre as premissas do ACID;
- NoSQL (Not Only SQL)
  - Orientado a documentos;
  - Não há uma estrutura de dados a seguir fielmente;
  - Os dados são mapeados via chave-valor;
  - Maior flexibilidade no formato dos dados;
  - Não cumpre as premissas do ACID.

Fonte: SQL, NoSQL, NewSQL: Qual banco de dados usar?

Chaves primárias e estrangeiras

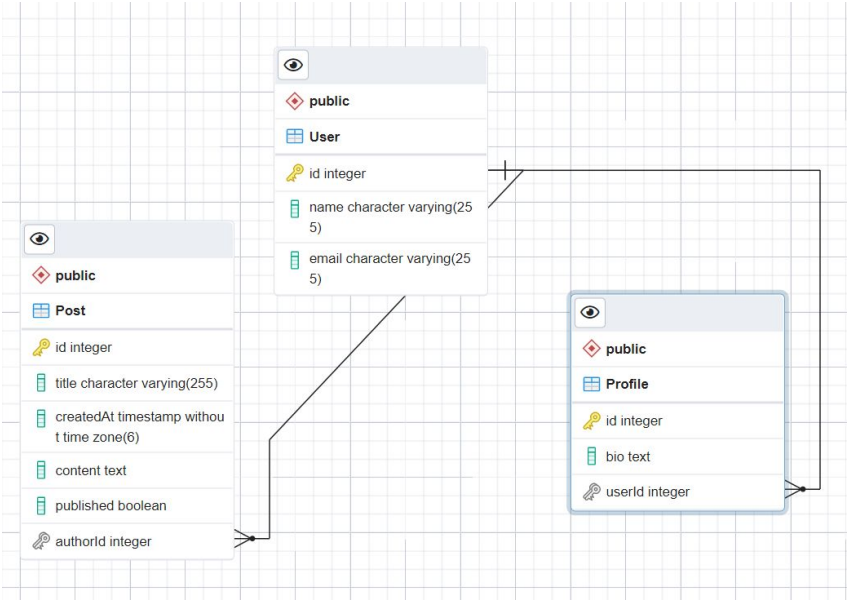
Primárias: chaves identificadores que garantem unicidade de um registro.

Estrangeiras: é chave Primária de uma tabela ‘colocada’ em outra tabela para criar um relacionamento.

Fonte:
Chave Primária e Chave Estrangeira: entenda a diferença entre elas

Relacionamentos

- **1:1:** quando um registro de uma tabela só pode ser associado com UM registro de outra tabela.
    - **Ex.:** um hospital onde um paciente só pode estar em um quarto e um quarto só pode acomodar um paciente
- **1:n:** quando um registro de uma tabela pode ser associado com n registros de outra tabela.
    - **Ex**.: um usuário pode ter 1 ou mais perfis
- **n:n**: quando um registro de uma tabela pode estar relacionado com vários registros dessa outra tabela e vice versa;
    - **Ex:** um ator pode participar de vários filmes assim como um filme pode ter vários atores.



Fontes:
https://pt.wikipedia.org/wiki/Cardinalidade_(modelagem_de_dados)
https://www.devmedia.com.br/modelagem-1-n-ou-n-n/38894

otterwise

Instalação do Postgres

psql: terminal para acessar o postgres por linha de comando
pgadmin: ferramenta visual para o postgres

comandos de terminal para o psql:

- Listar os bancos da máquina: \l
- Listar as tabelas de um banco de dados: \d
- Conectar a um banco da máquina para rodar scripts SQL: \c nome-do-banco

Para baixar o postgres: https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

Criação de um banco de dados

```
CREATE DATABASE otterwise;
```

otterwise

Criação de tabelas

```
CREATE TABLE person (id BIGSERIAL NOT NULL primary key, name varchar(100)
NOT NULL);
```

otterwise

Inserção de valores em uma tabela

```
INSERT INTO person (name) VALUES (‘joao’);
```

otterwise

Atualizando valores de uma tabela

```sql
UPDATE person SET name = 'joao pedro' WHERE id = 1;
```

otterwise

Excluindo valores de uma tabela

```sql
DELETE FROM person WHERE id = 1;
```

Banco de dados relacionais

- ACID
  - Atomicidade: ou tudo acontece ou nada acontece. Toda transação deve ser executada por completo;
  - Consistência: um banco de dados que estava consistente, deve continuar consistente após uma transação;
  - Isolamento: quando duas transações estiverem sendo executadas em paralelo, uma não pode afetar a outra;
  - Durabilidade: um dado deve durar o tempo necessário armazenado em um banco de dados.

Transação: toda ação, seja de escrita ou leitura, feita em um banco de dados relacional.

Fonte: https://pt.wikipedia.org/wiki/ACID

ORM

ORM (Object Relational Mapper)

- É uma técnica de mapeamento objeto relacional que permite fazer uma relação dos objetos com os dados que os mesmos representam.
- Algumas vantagens de utilizar um ORM:
  - Você escreve seu modelo de dados em um só lugar, e é mais fácil de atualizar, manter e reutilizar o código;
  - Você muitas vezes não precisa escrever queries SQL para fazer alguma consulta ou inserção no banco de dados;
  - Abstrai o banco de dados utilizado, já que o ORM que monta as queries;
- Algumas desvantagens:
  - Como o ORM monta as queries, algumas vezes elas podem não ficar otimizadas para o melhor desempenho.

Fonte:
Conceitos Básico em ORM (Object Relational Mapper)
https://pt.stackoverflow.com/questions/138938/quais-s%C3%A3o-as-fun%C3%A7%C3%B5es-de-um-orm

Exemplo

- Criar um projeto com fastify + prisma ORM
- Instalar o prisma e fazer o tutorial com as tabelas User, Profile e Post

Ferramentas

Ferramentas

Para consultar como os dados estão vindo de uma API REST e como é o nome dos recursos utilizados em uma API, utilizamos uma ferramenta para documentar e testar essa API. Uma das ferramentas mais utilizadas é o Postman.

O postman é um aplicativo que pode ser instalado na máquina e funciona como um playground de requisições.

- Postman (https://www.postman.com/)

Exercícios

Exercício

- Crie uma api de CRUD (Create, Read, Update e Delete) de tarefas. Essas tarefas devem ter título obrigatório e descrição opcional

Comandos úteis

Comandos úteis

- **CREATE DATABASE nomedobanco:** criar um banco de dados no postgres (usando o sql shell);
- **npx prisma init:** iniciar as configurações do Prisma no seu backend;
- **npx prisma db push**: serve para atualizar o banco de dados de acordo com o schema.prisma
- **npx prisma studio:** abre o dashboard visual do banco de dados feito pelo prisma;
- **nodemon**: biblioteca que funciona como um live server para o nodejs. Deve ser instalada globalmente ou no projeto. Exemplo de uso: ***nodemon arquivo.js***