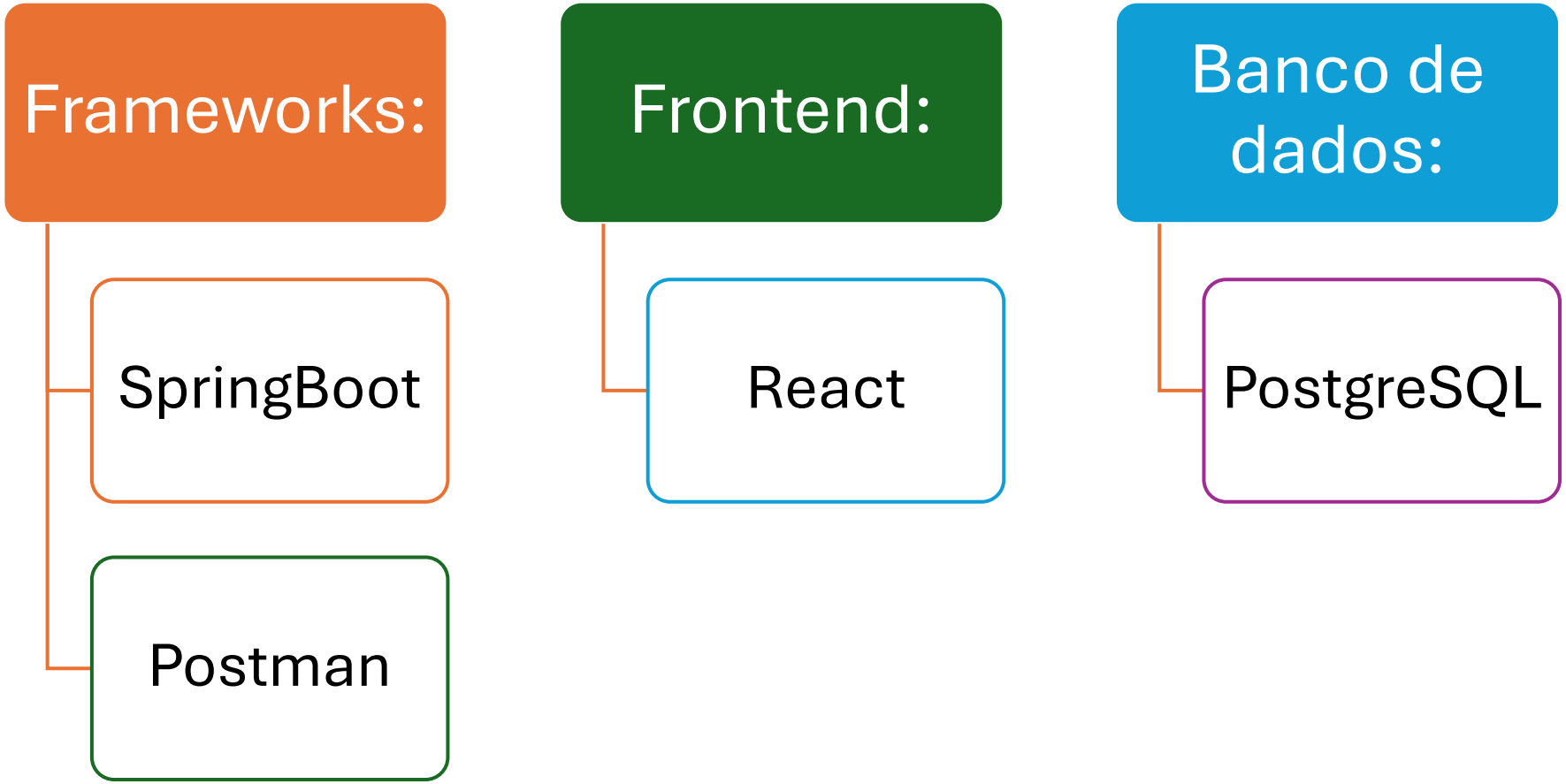
Projeto 4 – Banco de dados

Antonio Castelão

Antonio Vital

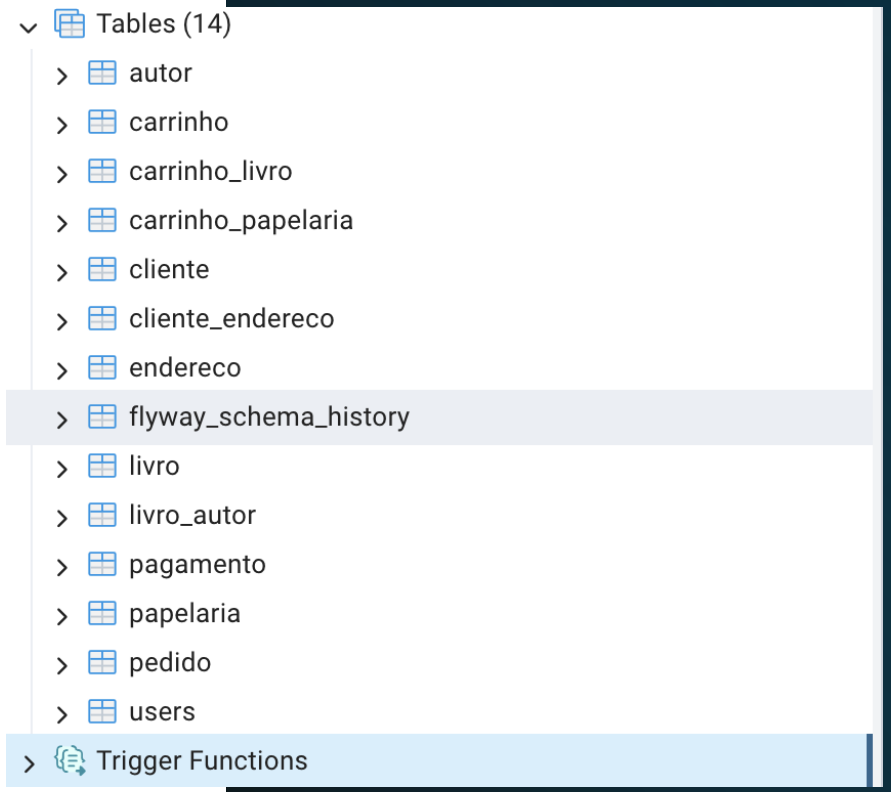
José Carlos de Barros

Tecnologias utilizadas



Tabelas do banco

- User
- Cliente
- Autor
- Carrinho
- Endereço
- Livro
- Pagamento
- Papelaria
- Pedido



Stored Procedures

01

Calcular quantidade total de produto no estoque

02

Calcular quantidade total de livros por autor

03

Calcular quantidade de produtos vendidos

Segurança

Spring Security

Cross Origin (cors)

Autenticação de usuário

Usuário e senha

Normalização

- **1. Primeira Forma Normal (1NF)**
  - A 1NF exige que todas as tabelas do banco de dados:
  - Tenham colunas únicas.
  - Cada entrada na coluna deve ser única (sem duplicatas).
  - Os valores das colunas devem ser atômicos (não divisíveis).

- **2. Segunda Forma Normal (2NF)**
  - A 2NF exige que todas as tabelas já estejam na 1NF e que todos os atributos não-chave sejam totalmente dependentes da chave primária.

- **3. Terceira Forma Normal (3NF)**
  - A 3NF exige que todas as tabelas já estejam na 2NF e que todos os atributos não-chave sejam diretamente dependentes da chave primária, ou seja, não devem haver dependências transitivas.

Trigger – autor_audit

```sql
CREATE TABLE autor_audit (
    id SERIAL PRIMARY KEY,
    autor_id BIGINT NOT NULL,
    nome_autor VARCHAR(255),
    sobrenome_autor VARCHAR(255),
    operacao CHAR(1), -- 'I' para Insert, 'U' para Update, 'D' para Delete
    alterado_em TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```sql
CREATE OR REPLACE FUNCTION log_autor_changes()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO autor_audit (autor_id, nome_autor, sobrenome_autor, operacao)
        VALUES (NEW.id_autor, NEW.nome_autor, NEW.sobrenome_autor, 'I');
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO autor_audit (autor_id, nome_autor, sobrenome_autor, operacao)
        VALUES (NEW.id_autor, NEW.nome_autor, NEW.sobrenome_autor, 'U');
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO autor_audit (autor_id, nome_autor, sobrenome_autor, operacao)
        VALUES (OLD.id_autor, OLD.nome_autor, OLD.sobrenome_autor, 'D');
        RETURN OLD;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

```sql
CREATE TRIGGER autor_changes_trigger
AFTER INSERT OR UPDATE OR DELETE ON autor
FOR EACH ROW EXECUTE FUNCTION log_autor_changes();
```

Trigger – livro_audit

```
CREATE OR REPLACE FUNCTION log_livro_changes() RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'INSERT') THEN
        INSERT INTO livro_audit (livro_id, descricao_livro, estoque_produto, nome_livro, avaliacao, data_publicacao, operacao)
        VALUES (NEW.id_livro, NEW.descricao_livro, NEW.estoque_produto, NEW.nome_livro, NEW.avaliacao, NEW.data_publicacao, 'I');
        RETURN NEW;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO livro_audit (livro_id, descricao_livro, estoque_produto, nome_livro, avaliacao, data_publicacao, operacao)
        VALUES (NEW.id_livro, NEW.descricao_livro, NEW.estoque_produto, NEW.nome_livro, NEW.avaliacao, NEW.data_publicacao, 'U');
        RETURN NEW;
    ELSIF (TG_OP = 'DELETE') THEN
        INSERT INTO livro_audit (livro_id, descricao_livro, estoque_produto, nome_livro, avaliacao, data_publicacao, operacao)
        VALUES (OLD.id_livro, OLD.descricao_livro, OLD.estoque_produto, OLD.nome_livro, OLD.avaliacao, OLD.data_publicacao, 'D');
        RETURN OLD;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER livro_changes_trigger
AFTER INSERT OR UPDATE OR DELETE ON livro
FOR EACH ROW EXECUTE FUNCTION log_livro_changes();
```

```
CREATE TABLE livro_audit (
    id BIGSERIAL PRIMARY KEY,
    livro_id BIGINT NOT NULL,
    descricao_livro VARCHAR(255),
    estoque_produto VARCHAR(255),
    nome_livro VARCHAR(255),
    avaliacao VARCHAR(255),
    data_publicacao TIMESTAMP,
    operacao CHAR(1) NOT NULL,
    alterado_em TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```