Índice

**CURSO BACHAREL EM**  **SISTEMAS DE INFORMAÇÃO**

**IGOR ALMENARA  
FILIPE SCANDIANI**

**Administração de Bancos de Dados: Entrega do projeto**

**Campos dos Goytacazes/RJ**

**2025**

# Minimundo

A Plataforma de Conteúdo por Assinatura FastDev é um sistema web que permite aos usuários que acessem vídeos exclusivos mediante pagamento.

O acesso pode ser adquirido através de assinatura mensal/anual ou por meio de pagamento único via Pix. Além disso, a plataforma oferecerá um modelo freemium, permitindo o acesso gratuito a determinados vídeos.

Para atrair novos clientes, a plataforma disponibilizará um teste grátis onde o usuário poderá assistir um vídeo específico por até 5 minutos.

Após esse tempo, ele será automaticamente redirecionado para a página de planos para que possa assinar e continuar assistindo o conteúdo completo. O usuário poderá favoritar vídeos para assisti-los posteriormente.

A monetização será gerenciada por um gateway de pagamento, permitindo transações seguras. O usuário poderá cancelar sua assinatura a qualquer momento, perdendo imediatamente o acesso à plataforma.

## 

## 

## 

## 

# Modelo Lógico

# Scripts

## Scripts DDL para criação da estrutura do banco

| CREATE TABLE usuario (  id SERIAL PRIMARY KEY,  nome VARCHAR(100) NOT NULL,  email VARCHAR(100) UNIQUE NOT NULL,  senha VARCHAR(255) NOT NULL,  admin BOOLEAN NOT NULL );  CREATE TABLE assinante (  id INT PRIMARY KEY REFERENCES usuario(id),  data\_cadastro DATE NOT NULL );  CREATE TABLE plano\_tipo (  nome VARCHAR(20) PRIMARY KEY );  INSERT INTO plano\_tipo (nome) VALUES   ('STANDART'),   ('PREMIUM'),   ('FREE');  CREATE TABLE assinatura (  id SERIAL PRIMARY KEY,  plano VARCHAR(20) REFERENCES plano\_tipo(nome),  data\_inicio DATE NOT NULL,  data\_fim DATE NOT NULL,  ativa BOOLEAN NOT NULL,  assinante\_id INT REFERENCES assinante(id) );  CREATE TABLE forma\_pagamento (  nome VARCHAR(30) PRIMARY KEY );  INSERT INTO forma\_pagamento (nome) VALUES   ('PIX'),   ('CARTAO\_CREDITO'),   ('CARTAO\_DEBITO');  CREATE TABLE pagamento (  id SERIAL PRIMARY KEY,  valor NUMERIC(10,2) NOT NULL,  data\_pagamento TIMESTAMP NOT NULL,  forma VARCHAR(30) REFERENCES forma\_pagamento(nome),  assinatura\_id INT UNIQUE REFERENCES assinatura(id) );  CREATE TABLE playlist (  id SERIAL PRIMARY KEY,  titulo VARCHAR(100) NOT NULL );  CREATE TABLE video (  id SERIAL PRIMARY KEY,  titulo VARCHAR(150) NOT NULL,  descricao TEXT,  url TEXT,  gratuito BOOLEAN DEFAULT FALSE,  duracao\_segundos INT );  CREATE TABLE playlist\_videos (  playlist\_id INT REFERENCES playlist(id),  video\_id INT REFERENCES video(id),  PRIMARY KEY (playlist\_id, video\_id) );  CREATE INDEX idx\_email\_usuario ON usuario(email); CREATE INDEX idx\_gratuito\_video ON video(gratuito); CREATE INDEX idx\_data\_inicio\_assinatura ON assinatura(data\_inicio); |
| --- |

## Script DLL para atualização de dados

| INSERT INTO usuario (nome, email, senha, admin) VALUES ('Filipe Scandiani', 'filipe@email.com', 'senha123', false), ('Maria Admin', 'maria@email.com', 'admin123', true);  INSERT INTO assinante (id, data\_cadastro) VALUES (1, CURRENT\_DATE);  INSERT INTO assinatura (plano, data\_inicio, data\_fim, ativa, assinante\_id) VALUES ('MENSAL', CURRENT\_DATE, CURRENT\_DATE + INTERVAL '30 days', true, 1);  INSERT INTO pagamento (valor, data\_pagamento, forma, assinatura\_id) VALUES (49.90, CURRENT\_TIMESTAMP, 'PIX', 1);  INSERT INTO video (titulo, descricao, url, gratuito, duracao\_segundos) VALUES ('Introdução ao Java', 'Aula gratuita introdutória', 'https://youtu.be/Scandianx/java123', true, 600), ('Spring Boot Avançado', 'Curso completo de Spring', 'https://youtu.be/Scandianx/spring', false, 1800);  INSERT INTO playlist (titulo) VALUES ('Playlist de Backend');  INSERT INTO playlist\_videos (playlist\_id, video\_id) VALUES (1, 1), (1, 2); |
| --- |

## Script DCL para controle de acesso

| CREATE USER fastdev\_user WITH PASSWORD 's3nh@.988';  GRANT CONNECT ON DATABASE fastdev TO fastdev\_user; GRANT USAGE ON SCHEMA public TO fastdev\_user;  GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO fastdev\_user; GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO fastdev\_user; |
| --- |

# Previsão de relatórios

## Total de receita mensal agrupada por tipo de plano

| SELECT   DATE\_TRUNC('month', p.data\_pagamento) AS mes,  a.plano,  SUM(p.valor) AS receita FROM pagamento p JOIN assinatura a ON p.assinatura\_id = a.id JOIN plano\_tipo pt ON a.plano = pt.nome GROUP BY mes, a.plano ORDER BY mes, a.plano; |
| --- |

## Classifica assinantes como Ativo, Expirado ou Inativo, com base no status e data

| SELECT   u.nome,  u.email,  CASE   WHEN a.ativa = true AND CURRENT\_DATE <= a.data\_fim THEN 'Ativo'  WHEN a.ativa = false AND CURRENT\_DATE > a.data\_fim THEN 'Expirado'  ELSE 'Inativo'  END AS status\_assinatura FROM usuario u JOIN assinante s ON u.id = s.id LEFT JOIN assinatura a ON s.id = a.assinante\_id; |
| --- |

## Calcula o valor médio pago (ticket médio) no último mês registrado, separado por forma de pagamento

| SELECT   forma,  ROUND(AVG(valor), 2) AS ticket\_medio FROM pagamento WHERE DATE\_TRUNC('month', data\_pagamento) = (  SELECT DATE\_TRUNC('month', MAX(data\_pagamento)) FROM pagamento ) GROUP BY forma; |
| --- |

### 

### 

### 

## Lista os top 5 assinantes que mais pagaram, útil para fidelização ou benefícios VIP

| SELECT   u.nome,  u.email,  SUM(p.valor) AS total\_pago FROM usuario u JOIN assinante s ON u.id = s.id JOIN assinatura a ON s.id = a.assinante\_id JOIN pagamento p ON a.id = p.assinatura\_id GROUP BY u.nome, u.email ORDER BY total\_pago DESC LIMIT 5; |
| --- |

## Percentual de vídeos gratuitos por playlist

| SELECT   pl.titulo AS playlist,  COUNT(\*) AS total\_videos,  SUM(CASE WHEN v.gratuito = true THEN 1 ELSE 0 END) AS total\_gratuitos,  ROUND(  100.0 \* SUM(CASE WHEN v.gratuito = true THEN 1 ELSE 0 END) / COUNT(\*),  2  ) AS percentual\_gratuito FROM playlist pl JOIN playlist\_videos pv ON pl.id = pv.playlist\_id JOIN video v ON pv.video\_id = v.id GROUP BY pl.titulo ORDER BY percentual\_gratuito DESC; |
| --- |

# Procedimentos

## Function:

Essa function retorna o número de dias restantes até o fim de uma assinatura ativa. Ela recebe o id da assinatura como parâmetro, calcula a diferença entre a data final (data\_fim) e o dia atual, e retorna o número de dias restantes. Se a assinatura já expirou ou estiver com data passada, ela retorna 0. Isso permite ao sistema mostrar ao usuário quantos dias ainda tem de acesso ao conteúdo.

| DELIMITER // CREATE FUNCTION dias\_restantes\_assinatura(assinatura\_id INT) RETURNS INT BEGIN  DECLARE dias INT;  SELECT DATEDIFF(data\_fim, CURRENT\_DATE) INTO dias  FROM assinatura  WHERE id = assinatura\_id AND ativa = TRUE;  RETURN IF(dias > 0, dias, 0); END // DELIMITER ; |
| --- |

## Trigger

Essa trigger é ativada automaticamente antes de qualquer atualização na tabela comentario. Ela compara o texto antigo com o novo, se houve alteração, registra o comentário anterior na nova tabela log\_comentario. Isso permite rastrear todas as edições feitas pelos usuários, garantindo histórico na plataforma.

| DELIMITER //  CREATE TRIGGER trg\_log\_edicao\_comentario BEFORE UPDATE ON comentario FOR EACH ROW BEGIN  IF OLD.texto <> NEW.texto THEN  INSERT INTO log\_comentario (comentario\_id, texto\_antigo, data\_alteracao, usuario\_id)  VALUES (OLD.id, OLD.texto, NOW(), OLD.usuario\_id);  END IF; END; //  DELIMITER ; |
| --- |

## Procedure

| DELIMITER //  CREATE PROCEDURE verificar\_e\_inativar\_assinaturas() BEGIN  DECLARE fim\_cursor BOOLEAN DEFAULT FALSE;  DECLARE assinatura\_id\_atual INT;   DECLARE assinaturas\_cursor CURSOR FOR  SELECT id FROM assinatura WHERE ativa = TRUE;   DECLARE CONTINUE HANDLER FOR NOT FOUND SET fim\_cursor = TRUE;   OPEN assinaturas\_cursor;   loop\_assinaturas: LOOP  FETCH assinaturas\_cursor INTO assinatura\_id\_atual;   IF fim\_cursor THEN  LEAVE loop\_assinaturas;  END IF;   IF dias\_restantes\_assinatura(assinatura\_id\_atual) = 0 THEN  UPDATE assinatura  SET ativa = FALSE  WHERE id = assinatura\_id\_atual;  END IF;  END LOOP;   CLOSE assinaturas\_cursor; END; //  DELIMITER ; |
| --- |

# Script de backup

## Script

| NOME\_BANCO="fastdev" USUARIO="fastdev\_user" SENHA="s3nh@.988" DIRETORIO="/home/filipe/backups/fastdev" DATA=$(date +"%Y-%m-%d\_%H-%M-%S") mkdir -p "$DIRETORIO" mysqldump -u"$USUARIO" -p"$SENHA" "$NOME\_BANCO" > "$DIRETORIO/backup\_${NOME\_BANCO}\_$DATA.sql" find "$DIRETORIO" -type f -name "\*.sql" -mtime +7 -exec rm {} \; |
| --- |

## Comandos

| crontab -e 0 2 \* \* \* /home/filipe/backup\_mysql.sh |
| --- |

# Mongo DB

No contexto da aplicação FastDev, armazenar os dados de comentários e favoritos dos vídeos no MongoDB seria interessante porque esse tipo de dado combina perfeitamente com a proposta dos bancos de dados não relacionais. Ações como curtir ou comentar são extremamente frequentes e atualizadas nas plataformas, e o MongoDB lida muito bem com esse tipo de demanda, mesmo em grande escala, sem perder desempenho.

Outro ponto importante é a flexibilidade. Hoje, um comentário pode ter apenas texto e data, mas amanhã pode ser que o usuário também queira adicionar uma reação, responder a outro comentário, ou até receber uma avaliação. Em bancos relacionais, qualquer mudança nesse formato exigiria alterações na estrutura da tabela, o que pode ser trabalhoso e pouco prático.