

**Universidade Eduardo Mondlane**

Faculdade de Ciências

Departamento de Matemática e Informática

**Tema:** Sistema de Gestão de Teatro

**Curso:** Licenciatura em Informática

**Nível:** III

**Regime:** Pós-laboral

**Disciplina:** Base de Dados II

**Discentes**:

Edson Da Cruz

**Docente**:

Jordão Uache

Maputo, Outubro de 2024

**Índice**

**[1.](#_Toc91011418)****[Contextualização](#_Toc91011418)** [3](#_Toc91011418)

**[2.](#_Toc91011419)****[Descrição do Sistema](#_Toc91011419)** [4](#_Toc91011419)

[3. Modelo de Base de Dados 5](#_Toc91011420)

[4.](#_Toc91011421) **[Scripts](#_Toc91011421)** [6](#_Toc91011421)

**[FUNÇÕES](#_Toc91011422)** [6](#_Toc91011422)

**[Is\_valid\_login](#_Toc91011423)** [6](#_Toc91011423)

**[Create\_bank\_account](#_Toc91011424)** [6](#_Toc91011424)

**[Get\_commission\_by trans\_type](#_Toc91011425)** [7](#_Toc91011425)

**[PROCEDIMENTOS](#_Toc91011426)** [8](#_Toc91011426)

**[Deposit](#_Toc91011427)** [8](#_Toc91011427)

**[Withdraw](#_Toc91011428)** [10](#_Toc91011428)

**[Get\_account\_statement](#_Toc91011429)** [13](#_Toc91011429)

**[TRIGGERS](#_Toc91011430)** [13](#_Toc91011430)

**[set\_transaction\_commission](#_Toc91011431)** [13](#_Toc91011431)

**[bank\_client\_after\_insert](#_Toc91011432)** [13](#_Toc91011432)

**[Insert\_transaction\_log](#_Toc91011433)** [14](#_Toc91011433)

**[Insert\_commission\_log](#_Toc91011434)** [15](#_Toc91011434)

# **Contextualização**

No cenário atual de entretenimento, a gestão eficiente de teatros é essencial para proporcionar experiências agradáveis ao público. A complexidade das operações, que inclui o agendamento de sessões, a venda de ingressos e a gestão de clientes, exige um sistema integrado que facilite essas interações. Este sistema de gestão de teatro baseado em PL/SQL surge como uma solução para atender a essas demandas, garantindo não apenas a eficiência administrativa, mas também a satisfação dos clientes.

# **Descrição do Sistema**

O sistema de gestão de teatro permite o controle das seguintes funcionalidades:

* **Gestão de Sessões:** Cadastro de salas, sessões e seus estados (aberta, fechada, adiada, etc.).
* **Venda de Ingressos:** Registo de ingressos com detalhes como cliente, assento e status (pendente, arquivado, reembolsado).
* **Reembolsos:** Processo de reembolso com regras baseadas na antecedência da solicitação em relação à data da sessão.
* **Transações:** Registo de todas as transações realizadas, incluindo compras e reembolsos, garantindo um histórico completo.

Este sistema proporciona uma gestão eficiente e organizada, facilitando a interação entre o teatro e os clientes, enquanto garante a conformidade com as políticas de vendas e reembolsos estabelecidas.

# **Modelo de Base de Dados**

O modelo usado foi o Modelo de Entidade e Relacionamento (E-R), que representa a realidade através da entidade, relacionamento entre as várias entidades e atributos que compõem uma entidade.

# **Scripts**

## **CONFIGURAÇÃO DA BASE DE DADOS**

### Tabelas

-- 1. Criação da tabela de Assentos (Seats)

-- Assentos serão fixos para todas as sessões, registrados uma só vez.

CREATE TABLE Seats (

    seat\_id NUMBER PRIMARY KEY, -- Chave primária para o assento

    seat\_category VARCHAR2(50) CHECK (seat\_category IN ('STANDARD', 'PREMIUM', 'VIP')) NOT NULL, -- Premium, VIP, Standard

    seat\_number VARCHAR2(10)  NOT NULL UNIQUE, -- Número único para cada assento

    seat\_row NUMBER,

    room\_id NUMBER NOT NULL,

    created\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updated\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    FOREIGN KEY (room\_id) REFERENCES TheaterRooms(room\_id)

);

-- 2. Criação da tabela de Salas (TheaterRooms)

-- Cada sala de teatro com capacidade definida por assentos.

CREATE TABLE TheaterRooms (

    room\_id NUMBER PRIMARY KEY, -- Chave primária para a sala

    room\_name VARCHAR2(100) UNIQUE NOT NULL, -- Nome da sala

    capacity NUMBER DEFAULT 0, -- Total de assentos na sala

    created\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updated\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- 3. Criação da tabela de Espectáculos/Sessões (Sessions)

-- Sessões marcadas em horários específicos, associadas a uma sala.

CREATE TABLE Sessions (

    session\_id NUMBER PRIMARY KEY, -- Chave primária para a sessão

    session\_name VARCHAR2(100) NOT NULL, -- Nome da sessão

    session\_description VARCHAR2(255), -- Descrição da sessão

    session\_date TIMESTAMP NOT NULL, -- Hora de início da sessão

    duration\_in\_minutes NUMBER NOT NULL, -- Duração da sessão

    room\_id NUMBER NOT NULL, -- ID da sala associada

    session\_state VARCHAR2(50) CHECK (session\_state IN ('aberta', 'fechada', 'cancelada', 'adiada', 'finalizada')), -- Estado da sessão

    FOREIGN KEY (room\_id) REFERENCES TheaterRooms(room\_id), -- Chave estrangeira para a sala

    created\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updated\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- ALTER TABLE SESSIONS RENAME COLUMN session\_date TO session\_date;

-- 4. Criação da tabela de Preços (TicketPrices)

-- Diferentes preços para categorias de assentos por sessão.

CREATE TABLE TicketPrices (

    price\_id NUMBER PRIMARY KEY, -- Chave primária para o preço

    session\_id NUMBER NOT NULL, -- ID da sessão associada

    seat\_category VARCHAR2(50) NOT NULL, -- Categoria do assento

    price NUMBER NOT NULL, -- Preço do ingresso

    FOREIGN KEY (session\_id) REFERENCES Sessions(session\_id), -- Chave estrangeira para a sessão

    created\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updated\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- 5. Criação da tabela de Compradores (Customers)

-- Clientes que compram ingressos.

CREATE TABLE Customers (

    customer\_id NUMBER PRIMARY KEY, -- Chave primária para o cliente

    customer\_name VARCHAR2(100) NOT NULL, -- Nome do cliente

    customer\_email VARCHAR2(100) UNIQUE NOT NULL, -- Email do cliente

    balance DECIMAL(10, 2) DEFAULT 0, --Saldo

    created\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updated\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- ALTER TABLE TICKETS ADD ( balance DECIMAL(10, 2) DEFAULT 0);

-- 6. Criação da tabela de Ingressos (Tickets)

-- Associações de clientes com sessões e assentos, para compra de ingressos.

CREATE TABLE Tickets (

    ticket\_id NUMBER PRIMARY KEY, -- Chave primária para o ingresso

    session\_id NUMBER NOT NULL, -- ID da sessão associada

    customer\_id NUMBER NOT NULL, -- ID do cliente

    seat\_id NUMBER NOT NULL, -- ID do assento

    balance DECIMAL(10, 2) DEFAULT 0, -- Saldo

    ticket\_status VARCHAR2(50) CHECK (ticket\_status IN ('pendente', 'arquivado', 'reembolsado')), -- Status do ticket após validação

    FOREIGN KEY (session\_id) REFERENCES Sessions(session\_id), -- Chave estrangeira para a sessão

    FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id), -- Chave estrangeira para o cliente

    FOREIGN KEY (seat\_id) REFERENCES Seats(seat\_id), -- Chave estrangeira para o assento

    created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- 7. Criação da tabela de Transações (Transactions)

-- Histórico de compras, reembolsos e validações de ingressos.

CREATE TABLE Transactions (

    transaction\_id NUMBER PRIMARY KEY, -- Chave primária para a transação

    customer\_id NUMBER NOT NULL, -- ID do cliente

    ticket\_id NUMBER NOT NULL, -- ID do ingresso

    transaction\_type VARCHAR2(50) CHECK (transaction\_type IN ('compra', 'reembolso', 'validação')), -- Tipo da transação

    transaction\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Data da transação

    transaction\_amount NUMBER NOT NULL,

    description VARCHAR2(255),

    FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id), -- Chave estrangeira para o cliente

    FOREIGN KEY (ticket\_id) REFERENCES Tickets(ticket\_id), -- Chave estrangeira para o ingresso

    created\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updated\_at    TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

### Sequências

-- 8. Criando sequências para cada tabela para gerar IDs automaticamente

CREATE SEQUENCE seq\_seat\_id START WITH 1 INCREMENT BY 1 NOMAXVALUE; -- Sequência para Assentos

CREATE SEQUENCE seq\_room\_id START WITH 1 INCREMENT BY 1 NOMAXVALUE; -- Sequência para Salas

CREATE SEQUENCE seq\_session\_id START WITH 1 INCREMENT BY 1 NOMAXVALUE; -- Sequência para Sessões

CREATE SEQUENCE seq\_price\_id START WITH 1 INCREMENT BY 1 NOMAXVALUE; -- Sequência para Preços

CREATE SEQUENCE seq\_customer\_id START WITH 1 INCREMENT BY 1 NOMAXVALUE; -- Sequência para Compradores

CREATE SEQUENCE seq\_ticket\_id START WITH 1 INCREMENT BY 1 NOMAXVALUE; -- Sequência para Ingressos

CREATE SEQUENCE seq\_transaction\_id START WITH 1 INCREMENT BY 1 NOMAXVALUE; -- Sequência para Transações

### Triggers para personalização de sequências para o formato (ano + xxxx)

-- 9. Trigger para gerar IDs automaticamente

CREATE OR REPLACE TRIGGER trg\_seat\_id

BEFORE INSERT ON Seats

FOR EACH ROW

DECLARE

    v\_year VARCHAR2(4); -- Variável para armazenar o ano atual

    v\_sequence VARCHAR2(4); -- Variável para armazenar a sequência formatada (0001, 0002, etc.)

BEGIN

    IF :NEW.seat\_id IS NULL THEN

        -- Obtenha o ano corrente (ex: 2024)

        v\_year := TO\_CHAR(SYSDATE, 'YYYY');

        -- Obtenha o próximo valor da sequência e formate com 4 dígitos

        SELECT LPAD(seq\_seat\_id.NEXTVAL, 4, '0') INTO v\_sequence FROM dual;

        -- Combine o ano com a sequência para gerar o seat\_id

        :NEW.seat\_id := v\_year || v\_sequence;

    END IF;

END;

/

CREATE OR REPLACE TRIGGER trg\_room\_id

BEFORE INSERT ON TheaterRooms

FOR EACH ROW

DECLARE

    v\_year VARCHAR2(4); -- Variável para armazenar o ano atual

    v\_sequence VARCHAR2(4); -- Variável para armazenar a sequência formatada (0001, 0002, etc.)

BEGIN

    -- Obtenha o ano corrente (ex: 2024)

    v\_year := TO\_CHAR(SYSDATE, 'YYYY');

    -- Obtenha o próximo valor da sequência e formate com 4 dígitos

    SELECT LPAD(seq\_room\_id.NEXTVAL, 4, '0') INTO v\_sequence FROM dual;

    -- Combine o ano com a sequência para gerar o room\_id

    :NEW.room\_id := v\_year || v\_sequence;

END;

/

CREATE OR REPLACE TRIGGER trg\_session\_id

BEFORE INSERT ON Sessions

FOR EACH ROW

DECLARE

    v\_year VARCHAR2(4); -- Variável para armazenar o ano atual

    v\_sequence VARCHAR2(4); -- Variável para armazenar a sequência formatada (0001, 0002, etc.)

BEGIN

    -- Obtenha o ano corrente (ex: 2024)

    v\_year := TO\_CHAR(SYSDATE, 'YYYY');

    -- Obtenha o próximo valor da sequência e formate com 4 dígitos

    SELECT LPAD(seq\_session\_id.NEXTVAL, 4, '0') INTO v\_sequence FROM dual;

    -- Combine o ano com a sequência para gerar o session\_id

    :NEW.session\_id := v\_year || v\_sequence;

END;

/

CREATE OR REPLACE TRIGGER trg\_price\_id

BEFORE INSERT ON TicketPrices

FOR EACH ROW

DECLARE

    v\_year VARCHAR2(4); -- Variável para armazenar o ano atual

    v\_sequence VARCHAR2(4); -- Variável para armazenar a sequência formatada (0001, 0002, etc.)

BEGIN

    -- Obtenha o ano corrente (ex: 2024)

    v\_year := TO\_CHAR(SYSDATE, 'YYYY');

    -- Obtenha o próximo valor da sequência e formate com 4 dígitos

    SELECT LPAD(seq\_price\_id.NEXTVAL, 4, '0') INTO v\_sequence FROM dual;

    -- Combine o ano com a sequência para gerar o price\_id

    :NEW.price\_id := v\_year || v\_sequence;

END;

/

CREATE OR REPLACE TRIGGER trg\_customer\_id

BEFORE INSERT ON Customers

FOR EACH ROW

DECLARE

    v\_year VARCHAR2(4); -- Variável para armazenar o ano atual

    v\_sequence VARCHAR2(4); -- Variável para armazenar a sequência formatada (0001, 0002, etc.)

BEGIN

    -- Obtenha o ano corrente (ex: 2024)

    v\_year := TO\_CHAR(SYSDATE, 'YYYY');

    -- Obtenha o próximo valor da sequência e formate com 4 dígitos

    SELECT LPAD(seq\_customer\_id.NEXTVAL, 4, '0') INTO v\_sequence FROM dual;

    -- Combine o ano com a sequência para gerar o customer\_id

    :NEW.customer\_id := v\_year || v\_sequence;

END;

/

CREATE OR REPLACE TRIGGER trg\_ticket\_id

BEFORE INSERT ON Tickets

FOR EACH ROW

DECLARE

    v\_year VARCHAR2(4); -- Variável para armazenar o ano atual

    v\_sequence VARCHAR2(4); -- Variável para armazenar a sequência formatada (0001, 0002, etc.)

BEGIN

    -- Obtenha o ano corrente (ex: 2024)

    v\_year := TO\_CHAR(SYSDATE, 'YYYY');

    -- Obtenha o próximo valor da sequência e formate com 4 dígitos

    SELECT LPAD(seq\_ticket\_id.NEXTVAL, 4, '0') INTO v\_sequence FROM dual;

    -- Combine o ano com a sequência para gerar o ticket\_id

    :NEW.ticket\_id := v\_year || v\_sequence;

END;

/

CREATE OR REPLACE TRIGGER trg\_transaction\_id

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

    v\_year VARCHAR2(4); -- Variável para armazenar o ano atual

    v\_sequence VARCHAR2(4); -- Variável para armazenar a sequência formatada (0001, 0002, etc.)

BEGIN

    -- Obtenha o ano corrente (ex: 2024)

    v\_year := TO\_CHAR(SYSDATE, 'YYYY');

    -- Obtenha o próximo valor da sequência e formate com 4 dígitos

    SELECT LPAD(seq\_transaction\_id.NEXTVAL, 4, '0') INTO v\_sequence FROM dual;

    -- Combine o ano com a sequência para gerar o transaction\_id

    :NEW.transaction\_id := v\_year || v\_sequence;

END;

/

### Procedure e Job para reiniciar as sequências anualmente

-- 9.1 Procedure para Reiniciar Sequências anualmente

CREATE OR REPLACE PROCEDURE reset\_sequences IS

BEGIN

    -- Reiniciar a sequência de Seats

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_seat\_id RESTART START WITH 1';

    -- Reiniciar a sequência de TheaterRooms

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_room\_id RESTART START WITH 1';

    -- Reiniciar a sequência de Sessions

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_session\_id RESTART START WITH 1';

    -- Reiniciar a sequência de TicketPrices

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_price\_id RESTART START WITH 1';

    -- Reiniciar a sequência de Customers

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_customer\_id RESTART START WITH 1';

    -- Reiniciar a sequência de Tickets

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_ticket\_id RESTART START WITH 1';

    -- Reiniciar a sequência de Transactions

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_transaction\_id RESTART START WITH 1';

END;

/

-- 9.2 Job Para automatizar a Reinicialização das Sequências Automático

BEGIN

    DBMS\_SCHEDULER.create\_job (

        job\_name        => 'reset\_sequences\_job',  -- Nome do job

        job\_type        => 'PLSQL\_BLOCK',          -- Tipo de job (chamando bloco PL/SQL)

        job\_action      => 'BEGIN reset\_sequences; END;',  -- Ação do job (chamar a procedure)

        start\_date      => SYSTIMESTAMP,  -- Primeira execução: agora (pode ajustar para outra data)

        repeat\_interval => 'FREQ=YEARLY; BYMONTH=1; BYMONTHDAY=1',  -- Frequência: anualmente, no 1º dia de janeiro

        enabled         => TRUE                    -- Ativar o job imediatamente

    );

END;

/

## **FUNÇÕES**

## **TRIGGERS**