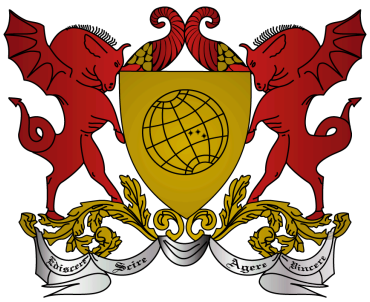


Universidade Federal de Viçosa
Campus Rio Paranaíba

Banco de Dados - Projeto Final

Lucas Moura Vidal da Silva - 6283
Pedro Afonso Lopes Mota - 7602
Felipe Pereira Rodrigues - 7610
Hebert Ribeiro Sampaio - 7633
Gabriel Rodrigues Nunes Alves - 7636

Rio Paranaíba
2023

1. **Descrição do ambiente de negócios**

Ambiente de Negócios - Descrição Detalhada:

O ambiente de negócios é um sistema de gerenciamento de eventos online, onde os usuários podem explorar, se inscrever e avaliar diferentes eventos em diversas categorias. Abaixo, fornecemos uma descrição detalhada das entidades, relacionamentos, atributos e requisitos funcionais e não funcionais.

Entidades:

Usuários (users):
Atributos: id (identificador único), name (nome do usuário), email (endereço de e-mail), password (senha do usuário), user_type (tipo de usuário, como "Regular" ou "Admin").
Relacionamentos: Envolvido em inscrições (registrations) e avaliações (reviews).

Users				
Tabela que armazena os dados dos usuários				
Atributos				
Nome	Tipo	Tipo de chave	Descrição	NULL OR NOT
id	INT	PK	Chave primária	NOT NULL
name	VARCHAR(255)		Nome	NULL
email	VARCHAR(255)		Email	NOT NULL
password	VARCHAR(255)		senha	NOT NULL
user_type	VARCHAR(255)		tipo de usuário	NOT NULL

Categorias (categories):
Atributos: id (identificador único), name (nome da categoria).
Relacionamentos: Usada para categorizar eventos.

Categories				
Tabela que armazena categorias dos usuários				
Atributos				
Nome	Tipo	Tipo de chave	Descrição	NULL OR NOT
id	INT	PK	Chave primária	NOT NULL
name	VARCHAR(255)		Nome	NOT NULL

Eventos (events):
Atributos: id (identificador único), title (título do evento), description (descrição do evento), date (data do evento), time (hora do evento), location (localização do evento), category_id (chave estrangeira referenciando a tabela categories), price (preço do evento), images (caminho para imagens associadas ao evento).

Relacionamentos: Pertence a uma categoria, pode ter inscrições (registrations) e avaliações (reviews).

Eventos				
Tabela que armazena os dados dos eventos				
Atributos				
Nome	Tipo	Tipo de chave	Descrição	NULL OR NOT
id	INT	PK	Chave primária	NOT NULL
title	VARCHAR(255)		Titulo	NULL
description	TEXT		Descrição	NULL
location	VARCHAR(255)		Local	NULL
category_id	INT	FK	Categoria de id	NOT NULL
price	DECIMAL(10,2)		Preço	NULL
images	VARCHAR(255)		Imagens	NULL
date	DATE		Data	NULL
time	TIME		Hora	NULL

Inscrições (registrations):
Atributos: id (identificador único), user_id (chave estrangeira referenciando a tabela users), event_id (chave estrangeira referenciando a tabela events), payment_status (status do pagamento, como "Paid" ou "Pending").
Relacionamentos: Conecta usuários a eventos.

Registros				
Tabela que armazena os dados dos registros				
Atributos				
Nome	Tipo	Tipo de chave	Descrição	NULL OR NOT
id	INT	PK	Chave primária	NOT NULL
user_id	INT	FK	ID do Usuário	NOT NULL
event_id	INT	FK	ID do Evento	NOT NULL
payment_status	VARCHAR(255)		Status do pagamento	NULL

Avaliações (reviews):
Atributos: id (identificador único), user_id (chave estrangeira referenciando a tabela users), event_id (chave estrangeira referenciando a tabela events), score (pontuação da avaliação), comment (comentário da avaliação).
Relacionamentos: Usuários deixam avaliações para eventos.

Revisões				
Tabela que armazena os dados das revisões				
Atributos				
Nome	Tipo	Restrições	Descrição	NULL OR NOT
id	INT	PK	Chave primária	NOT NULL
user_id	INT	FK	ID do Usuário	NOT NULL
event_id	INT	FK	ID do Evento	NOT NULL
score	INT		Pontuação	NULL
comment	TEXT		Comentário	NULL

Requisitos Funcionais:

Exploração de Eventos:
Os usuários podem listar todos os eventos disponíveis.

Contagem de Inscrições por Usuário:
O sistema deve calcular o número de inscrições feitas por cada usuário.

Detecção de Eventos com Inscrições Pendentes:
Identificar eventos com inscrições pendentes de pagamento.

Listagem de Usuários que Deixaram Avaliações:
Gerar uma lista de usuários que deixaram avaliações.

Total de Inscrições por Categoria:
Calcular o número total de inscrições para cada categoria de evento.

Identificação de Eventos sem Avaliações:
Encontrar eventos que ainda não foram avaliados.

Listagem de Usuários Inscritos em um Evento Específico:
Obter uma lista de usuários que se inscreveram em um evento específico.

Seleção de Eventos com Número Mínimo de Inscrições:
Identificar eventos que têm pelo menos uma inscrição.

Cálculo da Média de Avaliações por Usuário:
Calcular a média das avaliações feitas por cada usuário.

Encontrar eventos com a maior quantidade de inscrições pagas:
Apresenta o evento com mais inscrições pagas.

Requisitos Não Funcionais:

Desempenho:
As consultas devem ser otimizadas para garantir uma resposta rápida, mesmo com um grande volume de dados.

Segurança:
As senhas dos usuários devem ser armazenadas de forma segura, utilizando técnicas adequadas de criptografia.

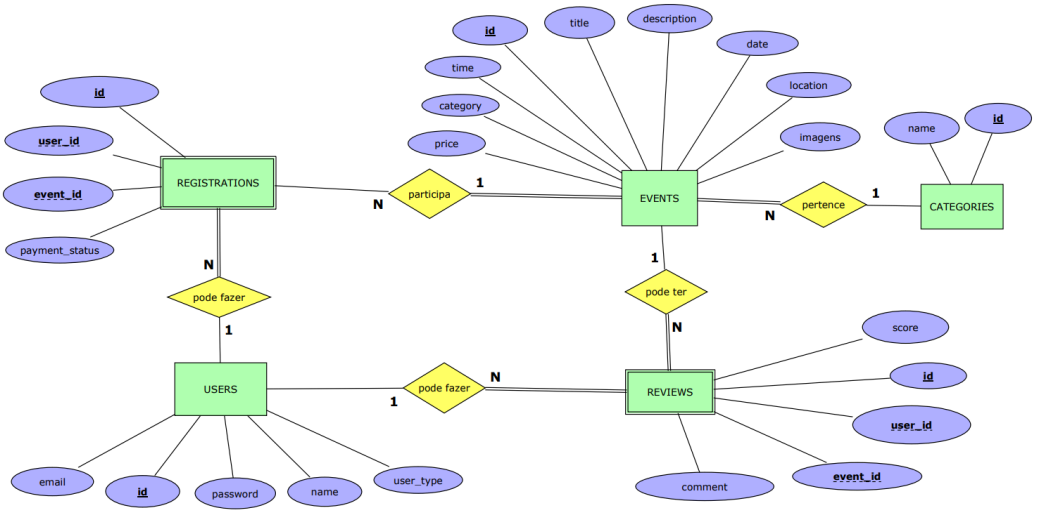
Confiabilidade:
O sistema deve ser robusto e capaz de lidar com falhas de forma a minimizar impactos nos usuários.

Usabilidade:

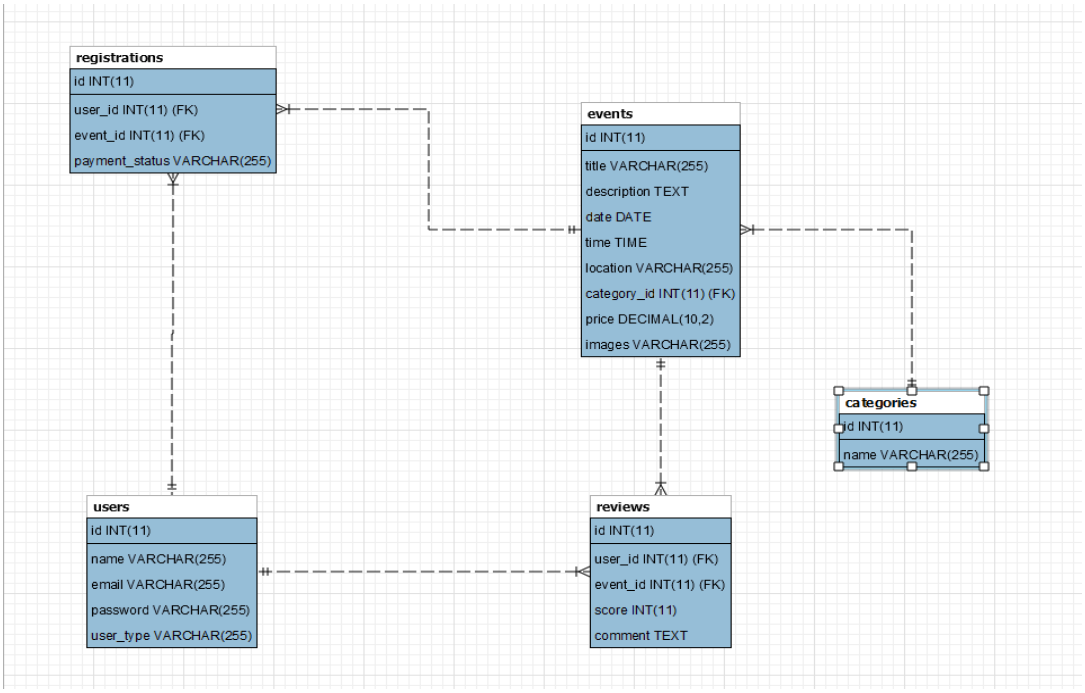
A interface do usuário deve ser intuitiva e fácil de usar, facilitando a navegação pelos eventos.

Escalabilidade:
O sistema deve ser projetado para lidar com um aumento no número de usuários e eventos ao longo do tempo.

2. Modelo conceitual (Diagrama Entidade-Relacionamento)



3. Esquema Relacional (crow's foot)



4. **Modelo Físico na forma de um script SQL de criação do Banco de Dados (DDL)**

```sql
CREATE DATABASE 6283_7602_7610_7633_7636;
USE 6283_7602_7610_7633_7636;

CREATE TABLE users (
  id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
  name VARCHAR(255),
  email VARCHAR(255) NOT NULL,
  password VARCHAR(255) NOT NULL,
  user_type VARCHAR(255) NOT NULL
);

CREATE TABLE categories (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE events (
  id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
  title VARCHAR(255),
  description TEXT,
  date DATE,
  time TIME,
  location VARCHAR(255),
  category_id INT NOT NULL,
  price DECIMAL(10,2),
  images VARCHAR(255)
);

CREATE TABLE registrations (
  id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
  user_id INT NOT NULL,
  event_id INT NOT NULL,
  payment_status VARCHAR(255)
);

CREATE TABLE reviews (
  id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
  user_id INT NOT NULL,
  event_id INT NOT NULL,
  score INT,
  comment TEXT
);
```

```sql
ALTER TABLE events
ADD FOREIGN KEY (category_id) REFERENCES categories(id);

ALTER TABLE registrations
ADD FOREIGN KEY (user_id) REFERENCES users(id),
ADD FOREIGN KEY (event_id) REFERENCES events(id);

ALTER TABLE reviews
ADD FOREIGN KEY (user_id) REFERENCES users(id),
ADD FOREIGN KEY (event_id) REFERENCES events(id);
```

Tabela	Ação	Linhas	Tipo	Colação	Tamanho	Sobrecarga
categories	Visualizar Estrutura Procurar Inserir Limpar Eliminar	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
events	Visualizar Estrutura Procurar Inserir Limpar Eliminar	0	InnoDB	utf8mb4_general_ci	32.0 KB	-
registrations	Visualizar Estrutura Procurar Inserir Limpar Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
reviews	Visualizar Estrutura Procurar Inserir Limpar Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
users	Visualizar Estrutura Procurar Inserir Limpar Eliminar	0	InnoDB	utf8mb4_general_ci	16.0 KB	-

5. Script de Inserção dos dados para as tabelas

DDL.sql | Insert.sql | Consultas.sql

SQL > Insert.sql

```sql
-- Inserir dados na tabela users
INSERT INTO users (name, email, password, user_type) VALUES
  ('User1', 'user1@example.com', 'password1', 'Regular'),
  ('User2', 'user2@example.com', 'password2', 'Admin'),
  ('User3', 'user3@example.com', 'password3', 'Regular'),
  ('User4', 'user4@example.com', 'password4', 'Regular');

-- Inserir dados na tabela categories
INSERT INTO categories (name) VALUES
  ('Concerts'),
  ('Conferences'),
  ('Workshops'),
  ('Exhibitions');

-- Inserir dados na tabela events
INSERT INTO events (title, description, date, time, location, category_id, price, images) VALUES
  ('Concert1', 'Description for Concert1', '2023-01-15', '18:00:00', 'Venue1', 1, 20.00, 'concert1.jpg'
  ('Conference1', 'Description for Conference1', '2023-02-20', '10:00:00', 'Venue2', 2, 0.00, NULL),
  ('Workshop1', 'Description for Workshop1', '2023-03-10', '14:00:00', 'Venue3', 3, 10.00, 'workshop1.j
  ('Exhibition1', 'Description for Exhibition1', '2023-08-25', '11:00:00', 'Venue4', 4, 0.00, 'exhibiti
  ('Concert2', 'Description for Concert2', '2023-04-05', '19:00:00', 'Venue5', 1, 25.00, 'concert2.jpg'
  ('Conference2', 'Description for Conference2', '2023-05-15', '09:30:00', 'Venue6', 2, 0.00, NULL),
  ('Concert3', 'Description for Concert3', '2023-06-20', '20:30:00', 'Venue7', 1, 30.00, 'concert3.jpg'
  ('Workshop2', 'Description for Workshop2', '2023-07-12', '15:30:00', 'Venue8', 3, 15.00, 'workshop2.j
```

```sql
-- Inserir dados na tabela registrations
INSERT INTO registrations (user_id, event_id, payment_status) VALUES
  (1, 1, 'Paid'),
  (2, 1, 'Pending'),
  (3, 1, 'Paid'),
  (4, 1, 'Pending'),
  (1, 2, 'Paid'),
  (2, 2, 'Pending'),
  (3, 2, 'Paid'),
  (4, 2, 'Pending'),
  (1, 3, 'Paid'),
  (2, 3, 'Pending'),
  (3, 3, 'Paid'),
  (4, 3, 'Paid'),
  (1, 4, 'Pending'),
  (2, 4, 'Paid'),
```

```sql
INSERT INTO reviews (user_id, event_id, score, comment) VALUES
  (1, 1, 5, 'Great event!'),
  (2, 1, 4, 'Enjoyed the concert'),
  (3, 1, 3, 'Okay concert'),
  (4, 1, 5, 'Fantastic concert'),
  (1, 2, 4, 'Informative conference'),
  (2, 2, 3, 'Good conference'),
  (3, 2, 5, 'Excellent conference'),
  (4, 2, 2, 'Disappointed with the conference'),
  (1, 3, 4, 'Enjoyed the workshop'),
  (2, 3, 5, 'Well-organized workshop'),
  (3, 3, 3, 'Average workshop'),
  (4, 3, 4, 'Good workshop'),
  (1, 4, 5, 'Amazing exhibition'),
  (2, 4, 4, 'Interesting exhibition'),
  (3, 4, 2, 'Not impressed with the exhibition'),
  (4, 4, 3, 'Average exhibition'),
  (1, 5, 4, 'Nice concert'),
  (2, 5, 3, 'Average concert'),
  (3, 5, 5, 'Fantastic concert'),
  (4, 5, 2, 'Disliked the concert'),
  (1, 6, 3, 'Good workshop'),
```

6. **Consultas em SQL**

Obs: Além dos prints os códigos podem ser encontrados no arquivo(Consultas.sql);

1)

```sql
-- Listar todos os eventos
SELECT *
FROM events;
```

	id	title	description	date	time	location	category_id	price	images
Editar Copiar Remover	1	Concert1	Description for Concert1	2023-01-15	18:00:00	Venue1	1	20.00	concert1.jpg
Editar Copiar Remover	2	Conference1	Description for Conference1	2023-02-20	10:00:00	Venue2	2	0.00	*NULL*
Editar Copiar Remover	3	Workshop1	Description for Workshop1	2023-03-10	14:00:00	Venue3	3	10.00	workshop1.jpg
Editar Copiar Remover	4	Exhibition1	Description for Exhibition1	2023-08-25	11:00:00	Venue4	4	0.00	exhibition1.jpg
Editar Copiar Remover	5	Concert2	Description for Concert2	2023-04-05	19:00:00	Venue5	1	25.00	concert2.jpg
Editar Copiar Remover	6	Conference2	Description for Conference2	2023-05-15	09:30:00	Venue6	2	0.00	*NULL*
Editar Copiar Remover	7	Concert3	Description for Concert3	2023-06-20	20:30:00	Venue7	1	30.00	concert3.jpg
Editar Copiar Remover	8	Workshop2	Description for Workshop2	2023-07-12	15:30:00	Venue8	3	15.00	workshop2.jpg

2)

```sql
-- Contar o número de inscrições por usuário
SELECT users.name, COUNT(registrations.id) AS num_registrations
FROM users
LEFT JOIN registrations ON users.id = registrations.user_id
GROUP BY users.name;
```

name	num_registrations
User1	8
User2	8
User3	8
User4	8

3)

```sql
-- Encontrar eventos com inscrições pendentes
SELECT events.title
FROM events
JOIN registrations ON events.id = registrations.event_id
WHERE registrations.payment_status = 'Pending';
```

title
Concert1
Concert1
Conference1
Conference1
Workshop1
Exhibition1
Exhibition1
Concert2
Conference2
Concert3
Workshop2

4)

```sql
-- Listar usuários que deixaram avaliações
SELECT DISTINCT users.name
FROM users
JOIN reviews ON users.id = reviews.user_id;
```

name
User1
User2
User3
User4

5)

```sql
-- Calcular o total de inscrições por categoria
SELECT categories.name, COUNT(registrations.id) AS num_registrations
FROM categories
LEFT JOIN events ON categories.id = events.category_id
LEFT JOIN registrations ON events.id = registrations.event_id
GROUP BY categories.name;
```

name	num_registrations
Concerts	12
Conferences	8
Exhibitions	4
Workshops	8

6)

```sql
-- Encontrar eventos sem avaliações
SELECT events.title
FROM events
LEFT JOIN reviews ON events.id = reviews.event_id
WHERE reviews.id IS NULL;
```

title
EventSemAvaliacao

7)

```sql
-- Listar usuários que se inscreveram em um evento específico
SELECT users.name
FROM users
JOIN registrations ON users.id = registrations.user_id
WHERE registrations.event_id = 1;
```

name
User1
User2
User3
User4

8)

```sql
-- Encontrar eventos com pelo menos 1 inscrição
SELECT events.title
FROM events
JOIN registrations ON events.id = registrations.event_id
GROUP BY events.title
HAVING COUNT(registrations.id) >= 1;
```

title
Concert1
Concert2
Concert3
Conference1
Conference2
Exhibition1
Workshop1
Workshop2

9)

```sql
-- Calcular a média de avaliações por usuário
SELECT users.name, AVG(reviews.score) AS avg_score
FROM users
LEFT JOIN reviews ON users.id = reviews.user_id
GROUP BY users.name;
```

name	avg_score
User1	4.2500
User2	3.7500
User3	3.8750
User4	2.7500

10)

```sql
-- Encontrar eventos com a maior quantidade de inscrições pagas
SELECT events.title, COUNT(registrations.id) AS num_paid_registrations
FROM events
JOIN registrations ON events.id = registrations.event_id
WHERE registrations.payment_status = 'Paid'
GROUP BY events.title
ORDER BY num_paid_registrations DESC
LIMIT 1;
```

title	num_paid_registrations
Concert2	3