



UNIVERSIDADE DE SÃO PAULO  
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO  
Departamento de Ciências de Computação

## **SCC0540 - Bases de Dados**

Profa. Elaine Parros Machado de Sousa

### **Projeto Incentivo ao Esporte Parte 1 - Descrição do Problema e Modelagem (MER): Sistema de Gestão de Instalações Esportivas Públicas**

Jade Bortot de Paiva	11372883
Eduardo Ribeiro Rodrigues	13696679
Bianca Gonçalves Dizio	12732145
Ivan Roberto Wagner Pancheniak Filho	12624224

São Carlos, Dezembro de 2024.

<b>1. Descrição do Problema e dos Requisitos de Dados.....</b>	<b>3</b>
<b>1.1. Principais operações.....</b>	<b>5</b>
1.1.1. Operações do Gestor Esportivo.....	5
1.1.2. Operações do Funcionário de Instalação.....	5
<b>2. Análise de ciclos e inconsistências no MER.....</b>	<b>7</b>
<b>2.1. Ciclos.....</b>	<b>7</b>
• Ciclo: Funcionário de Instalação - Instalação Esportiva - Espaço Esportivo - Reserva Esportiva.....	7
• Ciclo: Gestor Esportivo - Cidade - Instalação Esportiva - Espaço Esportivo - Manutenção - Contrato.....	7
<b>2.2. Inconsistências.....</b>	<b>7</b>
• Range de horário de reserva esportiva e de manutenção.....	7
• Apenas as empresas que prestam serviços de manutenção na cidade da Instalação Esportiva devem ser contratáveis pelo Gestor Esportivo.....	7
<b>3. Projeto Conceitual - Diagrama Entidade-Relacionamento (MER).....</b>	<b>8</b>
<b>4. Mudanças Realizadas na Primeira Parte do Projeto.....</b>	<b>9</b>
4.1. Correções no MER.....	9
4.2. Correções na escrita do projeto.....	9
4.3. Alterações finais no MER.....	9
<b>5. Modelo Relacional.....</b>	<b>10</b>
5.1. Discussões sobre o esquema relacional feito.....	11
<b>6. Mudanças realizadas na primeira e segunda parte do projeto.....</b>	<b>16</b>
<b>7. Implementação.....</b>	<b>17</b>
7.1. Estrutura de diretórios.....	17
7.2. Esquema do banco projetado.....	19
7.3. Consultas realizadas.....	20
7.4. Aplicação.....	22
<b>8. Conclusão.....</b>	<b>34</b>

# 1. Descrição do Problema e dos Requisitos de Dados

O projeto visa desenvolver um sistema de gestão para instalações esportivas públicas, buscando centralizar e automatizar o gerenciamento de locais como ginásios, quadras, piscinas e campos esportivos. O objetivo principal é otimizar o uso dessas instalações, facilitando o agendamento de reservas, o monitoramento de manutenções e a organização de competições locais. Dessa forma, pretende-se garantir que as instalações estejam bem conservadas e sejam utilizadas de forma eficiente, melhorando a experiência dos frequentadores.

O sistema desenvolvido incluirá diversas funcionalidades para atender as necessidades dos diferentes usuários. Primeiramente, haverá uma funcionalidade de agendamento de reservas, que permitirá aos usuários consultar a disponibilidade dos espaços esportivos e realizar reservas para eventos, evitando conflitos de horário e otimizando o uso das instalações. O sistema também oferecerá uma funcionalidade de monitoramento de manutenções, na qual os gestores poderão registrar e acompanhar manutenções preventivas e corretivas, garantindo que as instalações estejam sempre em boas condições de uso. Além disso, será possível gerar consultas e relatórios sobre a utilização das instalações, permitindo aos gestores verificar a disponibilidade dos espaços, consultar históricos de manutenção e acompanhar a frequência de uso por parte dos usuários. Outra funcionalidade será a gestão de competições, na qual os gestores poderão organizar e divulgar eventos esportivos locais, facilitando o acesso dos usuários a essas informações. Também haverá a possibilidade de cadastrar e atualizar informações sobre as instalações, incluindo detalhes sobre cada espaço esportivo, e gerenciar os dados dos funcionários envolvidos na administração e manutenção dos espaços. Todas essas funcionalidades visam proporcionar uma gestão eficiente e otimizada das instalações esportivas públicas.

O sistema é composto por diversos atores que representam os principais elementos envolvidos na gestão das instalações esportivas. A seguir, cada um dos atores será detalhado.

O primeiro ator é a **Instalação Esportiva**, que representa os locais físicos geridos pelo sistema, contendo em cada uma, um conjunto de **Espaços Esportivos**. Cada instalação possui como identificador único o seu CNPJ. Além disso, cada instalação possui nome e endereço, que incluem informações detalhadas de localização (rua, número, bairro, CEP). Vale ressaltar que a instalação possui uma associação com um **Gestor Esportivo** responsável pela sua administração.

O **Espaço Esportivo** representa cada área específica disponível dentro de uma instalação, como uma quadra de tênis ou uma piscina. O identificador de um espaço esportivo é composto pelo identificador da instalação ao qual ele pertence, juntamente com um número sequencial que o diferencia de outros espaços na mesma instalação. Além disso, cada espaço esportivo possui o tipo do espaço (quadra, ginásio, piscina, etc.).

O **Usuário** abrange todas as pessoas físicas ou jurídicas que utilizam as instalações. Cada usuário é identificado por um ID único (User ID), além de informações como nome, endereço, telefone e CPF/CNPJ. Os usuários podem ser pessoas físicas, como atletas e treinadores, ou jurídicas, como escolas e equipes. Os usuários podem realizar uma reserva em um **Espaço Esportivo** de uma determinada **Instalação Esportiva**.

A **Reserva Esportiva** é responsável por gerenciar o agendamento dos espaços esportivos, prevenindo conflitos de horários e mantendo um histórico de utilização. O que diferencia uma reserva é a data, hora de início e o identificador do espaço esportivo reservado. Também deve ser armazenado a hora de término, tipo de reserva (aula, competição, etc.), nome do evento e a quantidade de pessoas envolvidas.

A **Manutenção** representa os serviços de manutenção realizados nas instalações esportivas, assegurando que os espaços esportivos estejam sempre em boas condições de uso. O identificador único da manutenção é composto pela data, hora de início e o identificador do espaço esportivo. Deve-se armazenar também o horário de término, o tipo de manutenção (preventiva, corretiva, limpeza, reparo), a empresa responsável e o status da manutenção.

Os **Gestores Esportivos** representam os responsáveis pela administração das instalações esportivas de cada **Cidade**. Cada gestor é identificado por um ID único e possui atributos como nome, endereço, telefone, documentos e data de contratação. Os gestores são os principais usuários administrativos do sistema (junto com os **Funcionários de Instalação**), sendo responsáveis por negociar e assinar **Contratos** com as **Empresas de Manutenção**.

O **Funcionário de Instalação** é responsável pelas operações do dia a dia nas instalações esportivas. Cada funcionário é identificado por um ID único e possui atributos como nome, endereço, telefones, CPF, documentos e data de contratação. Os funcionários são responsáveis por agendar reservas, monitorar o estado das estruturas e garantir que tudo esteja em ordem para a utilização dos usuários. Eles também interagem com os gestores esportivos e as empresas de manutenção para coordenar reparos e outras atividades necessárias.

O sistema deve armazenar também os **Contratos** feitos entre os gestores esportivos e as empresas de manutenção. Cada contrato possui informações como o orçamento, o período de vigência, a data e os serviços contratados. O contrato é fundamental para formalizar a relação entre o gestor e a empresa de manutenção, garantindo que todas as obrigações e expectativas sejam documentadas e atendidas.

A **Empresa de Manutenção** representa as empresas responsáveis pela realização das manutenções nas instalações esportivas. Cada empresa é identificada pelo seu CNPJ, além de possuir atributos como nome, valor do serviço, telefone(s), tipo de serviço e tipo de empresa (pública ou privada).

É importante que o sistema registre todas as **Cidades**, pois é nelas que estão localizadas as instalações esportivas, e cada Gestor Esportivo é responsável pela administração das instalações de uma única cidade.

## 1.1. Principais operações

- O sistema possui dois principais tipos de usuários que o utilizarão: Gestor Esportivo e Funcionário de Instalações.

### 1.1.1. Operações do Gestor Esportivo

- **Inserções (Cadastros):**
  - Cadastro de Instalações Esportivas: Inserção de novas instalações (quadras, ginásios), com detalhes de endereço e tipo.
  - Cadastro de Usuários: Registro de novos usuários (pessoas físicas ou jurídicas), incluindo informações como nome, CPF/CNPJ, e telefone.
  - Cadastro de Manutenções: Registro de manutenções, especificando o tipo de serviço e a empresa responsável.
  - Cadastro de Funcionários: Registro de novos funcionários responsáveis pela administração das instalações.
- **Atualizações de Dados:**
  - Atualização de Informações das Instalações: Alteração de dados da instalação como nome, tipo, ou endereço.
  - Atualização de Manutenções: Alteração nas datas ou detalhes de manutenção em uma instalação.
  - Atualização de Dados de Funcionários e Usuários: Mudanças de dados pessoais como endereço, telefone, ou nome.
- **Remoções de Dados:**
  - Remoção de Instalações Esportivas: Excluir instalações esportivas quando não estiverem mais ativas, garantindo que não haja reservas ou manutenções associadas.
  - Remoção de Manutenções: Remover registros de manutenções futuras ou passadas, se necessário.
- **Consultas a serem realizadas:**
  - Monitorar Histórico de Manutenção: Listar manutenções realizadas ou agendadas em uma instalação esportiva.
  - Listar Usuários Frequentes: Consultar os usuários com mais reservas.

### 1.1.2. Operações do Funcionário de Instalação

- **Inserções (Cadastros):**
  - Cadastro de Reservas: Inserção de novas reservas de usuários para instalações específicas, com data, horário e evento associado (se houver).

- **Atualizações de Dados:**
  - Atualização de Reservas: Modificação de horários, datas, ou instalação reservada por um usuário.
- **Remoções de Dados:**
  - Cancelamento de Reservas: Excluir ou cancelar reservas quando o usuário desistir da utilização.
- **Consultas a serem realizadas:**
  - Verificar a Disponibilidade de Instalações: Consultar a agenda de uma instalação para verificar horários livres para reservas.
  - Verificar Cronograma de Reservas: Listar as reservas agendadas para uma determinada instalação ou período.

## 2. Análise de ciclos e inconsistências no MER

### 2.1. Ciclos

- **Ciclo: Funcionário de Instalação - Instalação Esportiva - Espaço Esportivo - Reserva Esportiva**

Acontece pela necessidade de associarmos funcionário a instalações esportivas e também indicar que esses mesmos funcionários têm que aprovar as reservas feitas por usuários. Como as reservas são feitas nos espaços esportivos e não nas instalações, o ciclo acaba se tornando necessário e não implica em redundância de dados.

- **Ciclo: Gestor Esportivo - Cidade - Instalação Esportiva - Espaço Esportivo - Manutenção - Contrato**

Esse ciclo acontece por um motivo similar ao ciclo anterior. Temos que associar um gestor esportivo a uma instalação e também esse mesmo gestor é responsável por aprovar as reservas de manutenção que ocorreram nos espaços esportivos de suas respectivas instalações. Da mesma forma que no ciclo anterior, ele é necessário e temos que as informações são independentes entre si. Como não foi possível tratar o ciclo na modelagem ER, deve-se tratar nos níveis superiores.

### 2.2. Inconsistências

- **Range de horário de reserva esportiva e de manutenção**

A aplicação deve garantir que o sistema não permita o cadastro de uma nova reserva ou manutenção em ranges de horários das reservas esportivas e manutenções já cadastradas.

- **Apenas as empresas que prestam serviços de manutenção na cidade da Instalação Esportiva devem ser contratáveis pelo Gestor Esportivo**

A modelagem do Banco de Dados não garante isso, tal restrição deve ser feita em nível de aplicação.

### 3. Projeto Conceitual - Diagrama Entidade-Relacionamento (MER)

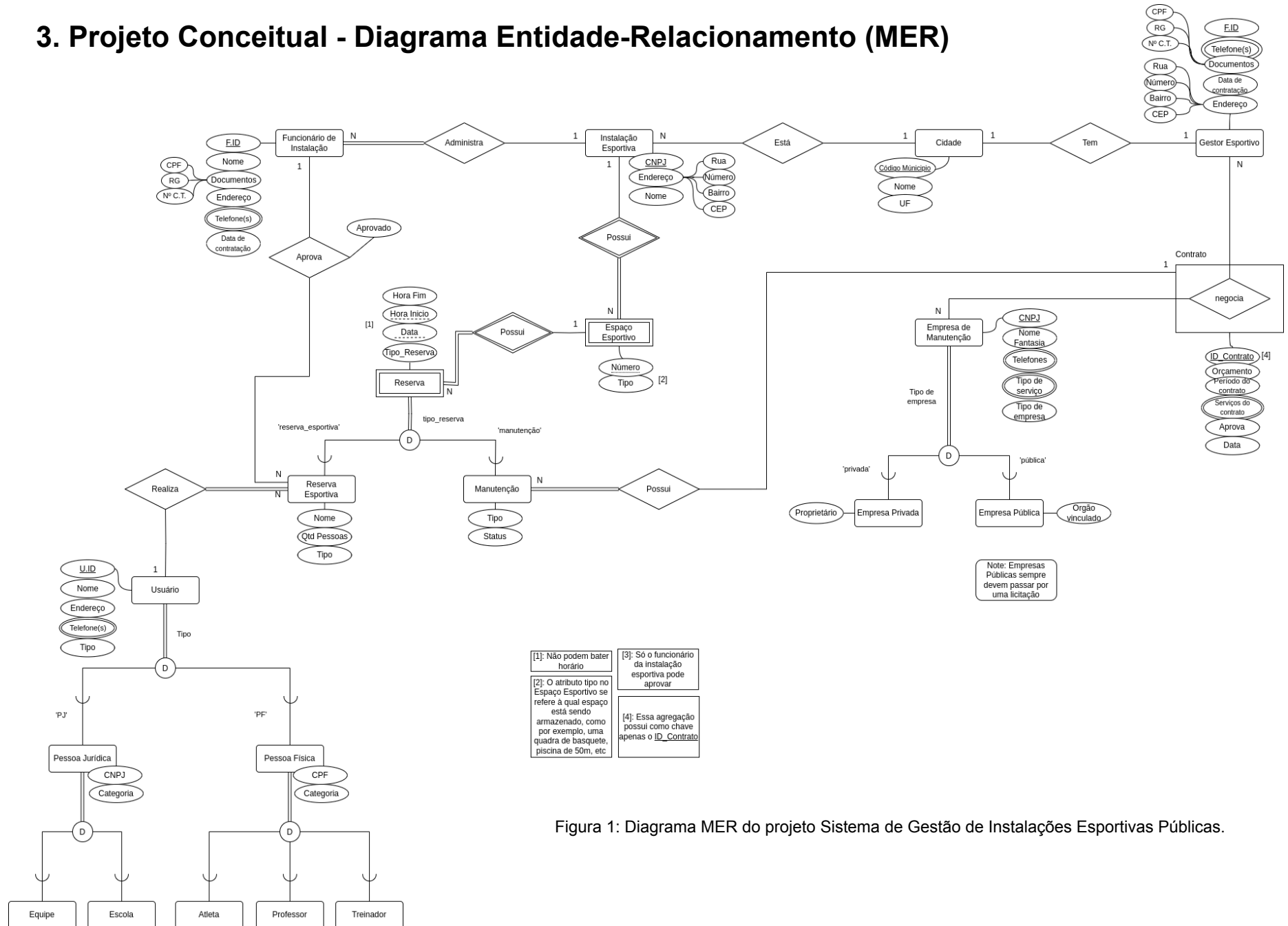


Figura 1: Diagrama MER do projeto Sistema de Gestão de Instalações Esportivas Públicas.



## **4. Mudanças Realizadas na Primeira Parte do Projeto**

### **4.1. Correções no MER**

Realizamos as devidas correções em nosso MER, das quais foram:

- Retirar o relacionamento “Aprova” entre a agregação “Manutenção” e o conjunto de entidade “Gestor Esportivo”;
- Foi colocado um atributo na agregação “Contrato”, chamada “ID\_Contrato”;
- E foi colocado o conjunto de entidade “Espaço Esportivo” relacionado com a agregação “Contrato”, para ser possível saber qual contrato se refere a qual espaço, deixando a modelagem de forma mais consistente.

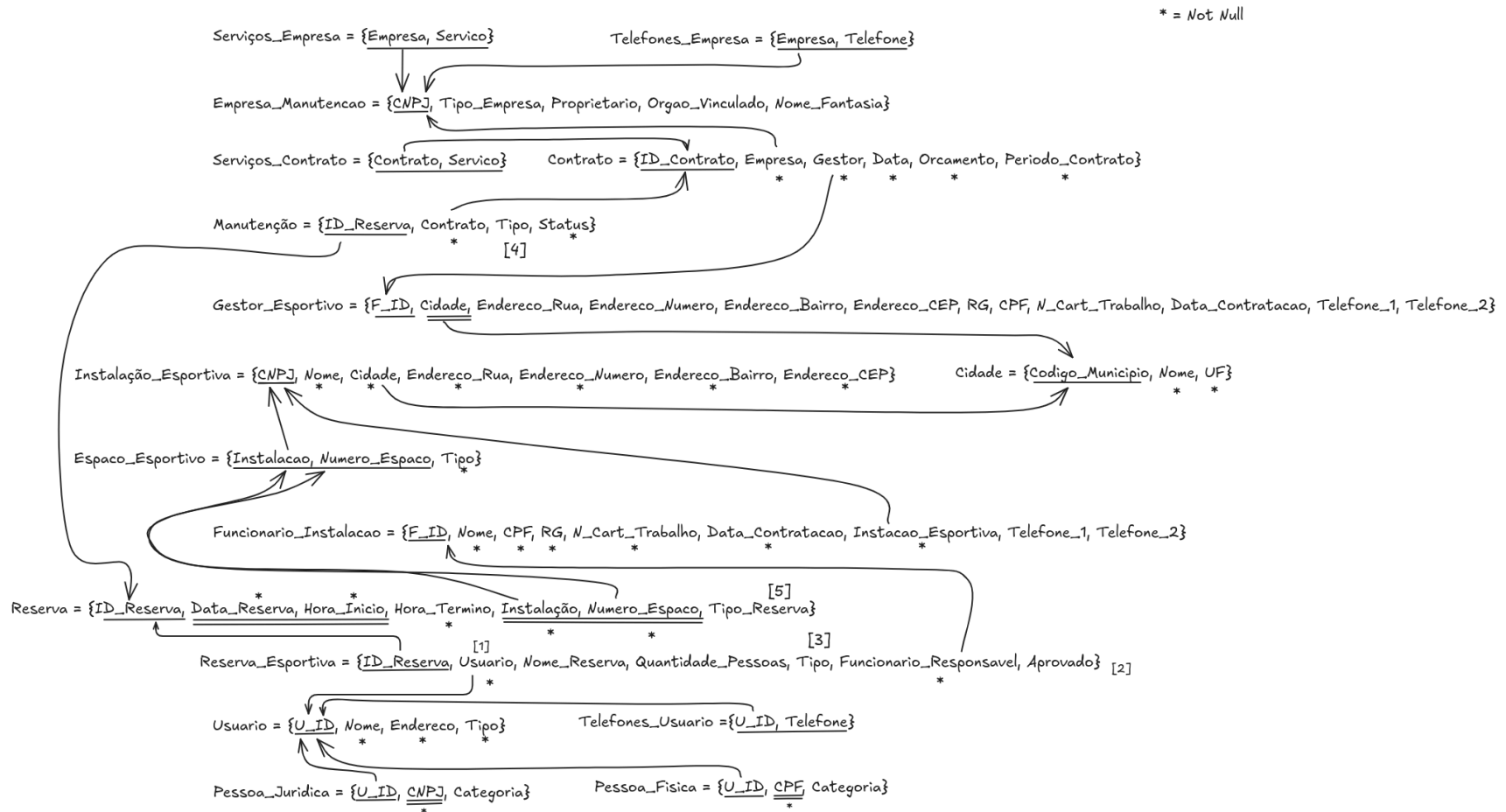
### **4.2. Correções na escrita do projeto**

Em relação à parte escrita do projeto, reescrevemos a descrição do projeto utilizando menos termos técnicos e de uma forma mais detalhada e narrativa, como se essa descrição viesse de um cliente sem muitos, ou até nenhum, conhecimento em base de dados. E também explicamos o que iríamos fazer de uma forma sem tantos termos técnicos, pensando de uma forma que qualquer pessoa possa entender. Em conjunto foi reestruturado também as principais operações para serem mais fáceis de entender por pessoas não técnicas.

### **4.3. Alterações finais no MER**

Foram feitas alterações no MER, do qual se encontra na Figura 2, pois percebemos que o MER feito anteriormente poderia ser alterado de tal forma que o problema em questão que estamos trabalhando possa ser melhor trabalhado e a modelagem possa ser desenvolvida e evoluída de uma forma mais eficiente e também o MER se torna mais explicativo.

## 5. Modelo Relacional



### Notes:

- [1]: O atributo "Usuario" da tabela "Reserva\_Esportiva" é para indicar qual é o responsável pela reserva.
- [2]: O atributo "Aprovado" é do tipo booleano.
- [3]: O atributo "Tipo" da relação Reserva\_Esportiva se refere à ela própria, ou seja, ao tipo de Reserva\_Esportiva (competição, jogo casual, eventos, ...).
- [4]: O atributo "Tipo" da relação Manutenção se refere à ele mesma, ou seja, ao tipo de Manutenção.
- [5]: O atributo "Tipo\_Reserva" é um atributo de condição que só poderá ter os valores "reserva\_esportiva" e "manutenção".

Figura 2: Mapeamento do esquema relacional do modelo MER feito e corrigido

## 5.1. Discussões sobre o esquema relacional feito

- **Relacionamento “Negocia” com agregação “Contrato”**

Para esse mapeamento foi desenvolvida uma solução alternativa, onde o relacionamento “Negocia” entre as entidades “Empresa de Manutenção” e “Gestor Esportivo” formam a agregação “Contrato” de cardinalidade N:N, onde não há outra opção que não seja criar uma tabela para o relacionamento e as tabelas respectivas dos conjuntos de entidades.

Porém, como esse relacionamento não tem atributos, apenas atributos da agregação, não há necessidade de criar tabelas separadas, uma para o relacionamento e uma para a agregação, basta apenas uma tabela que os contempla, resultando nas tabelas “Empresa\_Manutencao”, “Gestor\_Esportivo” e “Contrato”.

- **Vantagens:**

Usando essa solução alternativa evitamos custos de *join* ao criar três tabelas (duas tabelas para os conjuntos de entidades, mais uma tabela para a agregação) e evitamos uso de espaço desnecessário e replicação de dados..

- **Desvantagens:**

Nesta solução não vemos desvantagens.

- **Conjunto de entidade geral fraca “Reserva” e conjuntos de entidades especializadas fracas “Reserva\_Esportiva” e “Manutencao”**

Essa generalização foi mapeada em três tabelas, onde a tabela “Reserva”, referente ao conjunto de entidade fraca, onde serão armazenadas as informações gerais sobre a reserva, das quais são a data, horário de início e término, identificador da instalação, número do espaço e tipo de reserva, dessa forma, essa tabela se torna uma tabela a principal fonte de pesquisa e acesso para verificação de disponibilidade de reserva. Já seus conjuntos de entidades fracas “Reserva\_Esportiva” e “Manutenção” são as tabelas onde armazenamos as informações específicas de cada tipo de reserva que o nosso problema aborda.

- **Vantagens:**

Como temos a tabela “Reserva” que contém as informações básicas para verificação de disponibilidade para efetuar uma reserva, isso polpa a realização de *join* para acessar essas informações, das quais, dada o contexto do problema abordado, essas informações são muito acessadas, então o custo se torna mais baixo.

Outro ponto para ser destacado, que se torna uma vantagem, é o fato de que na tabela “Reserva” temos o atributo “Tipo\_Reserva”, onde, quando necessário obter o acesso às informações, diminuimos a quantidade de *joins* realizados, pois em vez de duas essas operações, será realizado apenas uma.

- **Desvantagens:**

Ainda, para acessar as informações de forma completa, é necessário realizar um *join*. E a especialização total nem a disjunção não são garantidas, além de que há a necessidade de se manter a consistência do atributo “Tipo\_Reserva”, para

não ocorrer um cenário de uma "Manutenção" ser marcada como "Reserva Esportiva".

- **Relacionamento “Possui” entre o conjunto de entidade fraca “Espaco\_Esportivo” e o conjunto de entidade forte “Reserva” com cardinalidade 1:N**

A entidade genérica Reserva é uma entidade fraca da entidade “Espaco\_Esportivo”, que por sua vez é uma entidade fraca da entidade “Instalação\_Esportiva”. “Reserva” é uma entidade fraca pois não possui identificador próprio, dependendo assim da chave das entidades “Espaco\_Esportivo” e “Instalação\_Esportiva”.

- **Relacionamento “Possui” entre o conjunto de entidade fraca “Manutencao” e a agregação “Contrato” com cardinalidade N:1 e o relacionamento “Realiza” entre os conjuntos de entidades “Reserva\_Esportiva” e “Usuario” com cardinalidade N:1**

Escolhemos mapear a entidade “Manutencao” com o atributo “Contrato” sendo uma *FK* para a agregação “Contrato”, pois cada contrato pode englobar diversas manutenções e uma única manutenção só pode estar atrelada a apenas um contrato.

- **Vantagens:**

Não há criação de uma tabela adicional, pois as informações vindas de outras tabelas são armazenadas na tabela “Manutencao”. Não há necessidade de *join* e também temos a não duplicação de dados, assim evitando inconsistências.

- **Desvantagens:**

Não vemos desvantagens nesse caso.

- **Conjunto de entidade “Empresa Manutenção” com os atributos multivalorados “Tipo de serviço” e “Telefones”**

Na modelagem, a entidade “Empresa de Manutenção” contém dois atributos multivalorados, “Tipo de serviço” e “Telefones”, ao mapear essa entidade, foi decidido que a melhor solução é criar uma tabela a parte para cada um desses atributos. Isto foi feito porque a quantidade dessas informações (telefones e tipos de serviços prestados) pode variar muito de empresa para empresa (incluindo os diferentes telefones para as diversas filiais ou setores de uma empresa) e visto que, dentro do problema proposto, é importante não perder essas informações, foi decidido manter esse mapeamento;

- **Vantagens:**

Não haverá perda de informação e nem campos nulos na tabela de empresas de manutenção.

- **Desvantagens:**

Para acessar os telefones e tipos de serviços prestados das empresas será necessário fazer um *join* com as tabelas, o que é muito custoso. Porém, dentro do problema, o acesso para essas informações não deve ser muito frequente.

- **Alternativas:**

Uma alternativa seria criar uma tabela só para os telefones através de uma nova entidade, contudo isso vem no custo de junções adicionais para chegar nesses dados.

- **Conjunto de entidade geral “Empresa de Manutenção” e conjuntos de entidades específicas “Empresa\_Privada” e “Empresa\_Publica”**

Escolhemos mapear a Especialização de Empresas de manutenção em uma única tabela para evitar as junções, deixando as buscas mais rápidas. Além disso, vale notar que as Entidades Especificas só possuem um atributo cada, o que garante que a manutenção de consistência seja mais simples.

- **Vantagens:**

Para acessar as informações, não será necessário um *join*, portanto, não teremos altos custos para acesso dessas informações.

- **Desvantagens:**

Alguns campos nulos na tabela.

- **Alternativas:**

Seria possível ter elas em diferentes tabelas, o que iria otimizar o uso de espaço, em relação a atributos nulos na tabela. Todavia, isso iria requerer mais junções para chegar nesses dados, e por eles serem poucos, o custo de uma junção a mais não parece valer a pena.

- **Relacionamento “Tem” entre as entidades “Gestor Esportivo” e “Cidade” de cardinalidade 1:1**

Nesse caso, fizemos uma tabela para “Gestor\_Esportivo” com o atributo “Cidade” que é *FK* do atributo chave “Codigo\_Município” de uma tabela que fizemos apenas para armazenar cidades, chamada “Cidade”. Na tabela “Gestor\_Esportivo”, o atributo “Cidade” é uma chave secundária e *Not Null*, dessa forma, garantimos que teremos apenas um gestor esportivo cadastrado em uma cidade. E com a tabela a parte de cidade, garantimos também que não teremos cidades cadastradas em estados errados e também não terá repetição de cidade.

- **Vantagens:**

Não haverá registro de cidades duplicadas e mais de um gestor esportivo em uma mesma cidade. E mantém a cardinalidade da relação.

- **Desvantagens:**

Necessidade do *join* para acessar as informações, mas a necessidade dessa junção entre as tabelas para ver as informações de forma completa é muito pouco esperada.

E também a dependência de haver uma cidade específica registrada para se registrar um gestor esportivo que a represente. Caso não haja uma determinada cidade, não será possível registrar um gestor sem antes registrar uma cidade.

- **Alternativas:**

Seria também possível a tabela "Cidade" ter uma *FK* para o "Gestor Esportivo", contudo, além disso não parecer uma maneira natural de organizar essa informação, ela apresentaria a mesma desvantagem apontada, mas no sentido inverso: não seria possível adicionar uma cidade sem antes colocar um gestor.

- **Relacionamento “Está” entre as entidades “Cidade” e “Instalação Esportiva” de cardinalidade 1:N**

Similar ao anterior, porém o atributo “Cidade” não é uma chave secundária pois podemos ter várias instalações esportivas em uma cidade. Mas para não haver instalações esportivas sem cidade, esse atributo é *Not Null*.

- **Vantagens:**

Não haverá registro de cidades duplicadas.

- **Desvantagens:**

Necessidade do *join* para acessar as informações, mas a necessidade dessa junção entre as tabelas para ver as informações de forma completa é muito pouco esperada.

- **Relacionamento “Possui” entre a entidade forte “Instalação Esportiva” e a entidade fraca “Espaço Esportivo” de cardinalidade 1:N**

Dado que isso é um mapeamento clássico de entidade fraca, “Espaço Esportivo” recebe o CNPJ da “Instalação Esportiva” a qual ele pertence e um número de identificação na sua chave primária.

- **Vantagens:**

Evitamos espaços nulos em uma tabela, caso fossemos juntar esses dois conjuntos de entidades em apenas uma tabela.

- **Desvantagens:**

Não é possível garantir que uma Instalação Esportiva tenha um ou mais Espaços Esportivos. Essa restrição não está no MER, mas caso fosse uma necessidade, não haveria como garantir no diagrama.

- **Relacionamento “Administra” entre as entidades “Instalação Esportiva” e “Funcionário de Instalação” de cardinalidade 1:N**

Nesse caso, foi decidido inserir, na tabela de funcionário, qual a instalação esportiva ele administra, tendo uma *FK* chamada “Instalacao\_Esportiva”, que é o atributo “CNPJ” da tabela de “Instalacao\_Esportiva”, como uma *Not Null*, para garantir que todo funcionário cadastrado trabalhe em uma instalação esportiva, como determina a cardinalidade proposta.

- **Vantagens:**

Garante a cardinalidade do relacionamento.

- **Desvantagens:**

Não é possível garantir que uma Instalação Esportiva tenha ao menos um Funcionário de Instalação. Pois a relação Instalação Esportiva é válida mesmo sem possuir relacionamentos com a relação Funcionário de Instalação. Porém, o contrário não se aplica. Um Funcionário de Instalação, precisa estar relacionado obrigatoriamente com uma Instalação Esportiva.

- **Relacionamento “Aprova” entre os conjuntos de entidades “Funcionario\_Instalação” e “Reserva\_Esportiva” de cardinalidade 1:N**

Tal relacionamento também tem cardinalidade 1:N, onde foi optado por mapear da mesma forma que citamos no item anterior, mas podemos perceber que o relacionamento “Aprova” tem o atributo “Aprovado”, que é booleano. Observando ao contexto semântico do nosso problema, temo que o funcionário aprova a reserva esportiva, portanto esse atributo foi inserido na tabela de “Reserva\_Esportiva”, pois colocá-lo na tabela de “Funcionario\_Instalacao” não faria sentido.

- **Vantagens:**  
É fácil e de baixo custo saber se uma determinada reserva esportiva foi aprovada ou não. E garante a cardinalidade do relacionamento.
  - **Desvantagens:**  
É possível que haja reservas, mas não tenha funcionário de instalação, portanto será impossível que as reservas sejam aprovadas.
  
- **Atributo multivalorado “Telefones” na entidade “Funcionario\_Instalacao” e “Gestor\_Esportivo”**  
Esse mapeamento levou em conta principalmente o contexto do problema que estamos trabalhando, nessas duas entidades, temos esse mesmo atributo multivalorado, do qual optamos por, em cada entidade, adicionar dois atributos, que são chamados de “Telefone\_1” e “Telefone\_2”. Pois dentro do contexto, estamos falando de funcionários que, normalmente, são obrigados a sempre estarem presentes no ambiente de trabalho e manter os seus dados atualizados. Caso algum funcionário desapareça, o mais conveniente seria a demissão e substituição do funcionário. Portanto, trabalhar dessa forma, com uma quantidade limitada de espaços para se inserir telefone é mais condizente também com a nossa realidade, da qual normalmente pessoas tem apenas um número de telefone que representa o telefone celular e o segundo telefone é de alguém próximo que em algumas vezes não mora junto.
  - **Vantagens:**  
Evitar alto custo de *join*.
  - **Desvantagens:**  
Temos um espaço limitado para inserir informações, em alguns casos pode haver perda de informações, mas no contexto do problema serão muito poucas.
  
- **Entidade genérica “Usuario” e entidades específicas “Pessoa\_Juridica” e “Pessoa\_Fisica”**  
Nesse mapeamento, optamos por deixar os conjuntos de entidades específicas separados do conjunto de entidade geral, pois as entidades específicas “Pessoa\_Juridica” e “Pessoa\_Fisica” tem outras entidades e relacionamentos específicos, onde se fossemos colocar tudo em apenas uma tabela, teríamos muitos espaços nulos, gastando mais espaço do que desejado. E como, pelo contexto do problema, a tabela que terá mais acessos será a tabela “Usuario”, onde contém as informações gerais do usuário, independente se é uma pessoa jurídica ou uma pessoa física, o custo de realizar *join*, mesmo existindo, não será altamente usado. E para facilitar a busca, a tabela “Usuario” contém o atributo “Tipo”, onde se refere se o registro

é de uma pessoa física ou jurídica, assim facilitando a busca e diminuindo um pouco o custo do *join*.

- **Vantagens:**  
Acesso rápido e mais fácil para os dados gerais dos usuários e a diminuição total dos espaços nulos. Garante a disjunção.
- **Desvantagens:**  
Necessidade de realizar o *join* para obter as informações específicas. Não garante a especialização total.
- **Entidades genéricas “Pessoa\_Juridica” e “Pessoa\_Fisica” e entidades específicas “Equipe”, “Escola”, “Atleta”, “Professor”, “Treinador” - solução alternativa**  
Em vista que os conjuntos de entidades especializadas não têm atributos próprios e essa especialização é do tipo de disjunção, os conjuntos de entidades gerais, “Pessoa\_Fisica” e “Pessoa\_Juridica” e especializadas foram mapeados junto com sua tabela “mãe” respectivamente, onde temos o atributo “Categoria” que sinaliza qual das categorias “filhas” o registro está se referindo. E como a especialização é do tipo de disjunção, então o atributo é um atributo simples e dessa forma garantimos a especialização total.
  - **Vantagens:**  
Não haverá necessidade de realizar o *join*, portanto o custo para acessar as informações será menor.
  - **Desvantagens:**  
Caso, futuramente, seja necessário separar os dados, terá que refazer o código, o que é caso pensando em custos humanos e custo de espaço usado. Mas pelo contexto semântico, essa necessidade será muito pequena.
- **ID sintético**  
O uso de IDs sintéticos para funcionários e usuários está justificado pelo fato de termos a possibilidade de uma pessoa de outro país vir a se tornar um funcionário ou um usuário. Sendo assim, não poderíamos utilizar como chave o CPF. Sobre o ID sintético da entidade Contrato, optamos por utilizá-lo com a justificativa de que não temos um atributo ideal para ser a chave desta entidade.

## 6. Mudanças realizadas na primeira e segunda parte do projeto

Em relação à primeira parte do projeto, foi feita uma atualização do MER para ele melhor representar o problema e o diagrama relacional através do feedback recebido. Dessa atualização, os seguintes pontos foram alterados:

- Foi corrigido na notação a chave composta da entidade fraca *Reserva*
- Foi corrigido a relação Possui entre *Manutenção* e Contrato de uma N:N para N:1



- Tornou-se às entidades *Reserva Esportiva* e *Manutenção* entidades fracas, para fazer sentido com *Reserva* sendo fraca
- *Pessoa Jurídica* e *Pessoa Física* ganharam um atributo de *categoria* para separar o subconjunto no qual eles se referem (*Equipe/Escola* ou *Atleta/Professor/Treinador*)
- Foi corrigido o relacionamento “administra” entre a entidade “Funcionário de Instalação” e “Instalação Esportiva”. Para que o MER represente corretamente o modelo Relacional, tivemos que adicionar uma participação total entre “Funcionário de Instalação” e “Instalação Esportiva”.

Já em relação à segunda parte do projeto, o modelo Relacional teve apenas uma alteração: inserimos o atributo *Data\_Contratacao* na relação *Gestor\_Esportivo* para garantir a consistência com o MER.

## 7. Implementação

A aplicação foi desenvolvida em Python, utilizando os pacotes *CustomTkinter* e *cx\_Oracle*. O *CustomTkinter* foi empregado para a criação de uma interface gráfica moderna e personalizável, facilitando a interação do usuário com as funcionalidades do sistema. Já o *cx\_Oracle* permitiu a integração direta com o banco de dados Oracle, sendo essencial para a execução de comandos SQL, como inserções e consultas de dados.

Para viabilizar a conexão entre a aplicação e o banco de dados, foi necessário configurar o Oracle Instant Client, um conjunto de ferramentas que fornece as bibliotecas e drivers necessários para comunicação eficiente com servidores Oracle. Todas as instruções para configuração do ambiente e execução da aplicação estão detalhadas no arquivo *README.md*, disponível na raiz do projeto ou no repositório do GitHub anexado [aqui](#).

### 7.1. Estrutura de diretórios

Os diretórios do projeto estão organizados de tal forma que tentamos deixar a aplicação e os scripts SQL devidamente separados, para facilitar o manuseio, mas ainda assim, sendo possível mantê-los conectados.

```
1 BANCO_DADOS_2024
2 |— bd-venv
3 |— documentation
4 |— src
5 |   |— Main.py
6 |   |— backend
7 |   |   |— db_connection.py
8 |   |— frontend
9 |   |   |— widgets
10 |   |   |   |— __init__.py
11 |   |   |   |— ConsultationFrame.py
12 |   |   |   |— ContentFrame.py
13 |   |   |   |— MainFrame.py
14 |   |   |   |— RegisterMaintenanceFrame.py
15 |   |   |   |— RegisterReservationFrame.py
16 |   |   |   |— Sidebar.py
17 |   |   |— Settings.py
18 |   |   |— Utils.py
19 |   |— sql
20 |   |   |— consultas.sql
21 |   |   |— dados.sql
22 |   |   |— esquema.sql
23 |— database.db
24 |— README.md
25 |— requirements.txt
```

Figura 3: Esquema dos diretórios do projeto

A estrutura de diretórios do projeto foi organizada de forma a separar claramente as responsabilidades e facilitar o desenvolvimento e a manutenção.

- **bd-venv**: Contém o ambiente virtual Python configurado para o projeto, com todas as dependências necessárias instaladas, incluindo os pacotes **CustomTkinter** e **cx\_Oracle**.
- **documentation**: Armazena os documentos e recursos relacionados à documentação do projeto, como PDFs e imagens requisitados pela professora ou monitor.
- **src**: É o diretório principal que abriga toda a lógica da aplicação, incluindo os códigos SQL e o arquivo **Main.py**, responsável por inicializar a aplicação. Este diretório está organizado em subdiretórios para separar as responsabilidades:

- **frontend**: Contém toda a lógica relacionada à interface gráfica da aplicação, como os frames, dicionários e elementos de design. A subpasta **widgets** inclui arquivos que implementam diferentes partes da interface, como **ConsultationFrame.py**, **ContentFrame.py**, **Sidebar.py**, entre outros.
- **backend**: Inclui o arquivo **db\_connection.py**, que centraliza a comunicação com o banco de dados. Esse arquivo é responsável por estabelecer a conexão, realizar inserções e executar consultas SQL.
- **sql**: Este diretório reúne os scripts SQL organizados em arquivos, como:
  - **consultas.sql**: Para as consultas ao banco de dados.
  - **dados.sql**: Para populações iniciais de dados.
  - **esquema.sql**: Para a criação e definição do esquema do banco de dados.

Além disso, o diretório principal contém o arquivo **requirements.txt**, que lista as dependências necessárias para a execução do projeto.

## 7.2. Esquema do banco projetado

Na construção do esquema, temos apenas alguns pontos pertinentes para serem comentados. Como semanticamente estamos lidando com empresas, pessoas de diferentes cidades e estados, estamos armazenando os CPFs, CNPJs, Código de Município e CEP aproveitando que eles são valores únicos dentro do nosso país, eles varias vezes são ou compões chaves primárias, ou secundárias, seguindo o que está no Modelo Relacional. Para a criação das tabelas, utilizamos o tipo CHAR para representar esses dados, porque em todos os casos eles são NOT NULL e em todo o país eles têm a mesma quantidade de caracteres. Nos atributos destinados para armazenar dados como RG, Número de Carteira de Trabalho, Telefones, por eles, mesmo tendo um padrão, são dados não padronizados no país, foi utilizado o tipo VARCHAR2.

Nesses dois casos citados acima, não utilizamos o tipo NUMBER, pois, nos atributos que irão armazenar o RG e o Número da Carteira de Trabalho em alguns estados pode haver caracteres não numéricos, então não seriam aceitos caso tentássemos inserir e para os outros dados comentados não foram utilizados o tipo NUMBER por conta da semântica, esses dados não são utilizados para realização de contas, mas para identificação, assim optamos em utilizar o tipo NUMBER apenas para os IDs Artificiais e para dados possam ser necessitados para realização de contas, como, por exemplo, o ORCAMENTO na tabela CONTRATO.

### 7.3. Consultas realizadas

```

1 SELECT
2     RE.Nome_Reserva,
3     R.Data_Reserva,
4     U.Nome AS NOME_USUARIO,
5     F.Nome AS NOME_FUNCIONARIO
6 FROM Reserva_Esportiva RE
7     JOIN Reserva R
8         ON R.ID_Reserva = RE.ID_Reserva
9     JOIN Usuario U
10        ON U.U_ID = RE.Usuario
11     JOIN Funcionario_Instalacao F
12        ON F.F_ID = RE.Funcionario_Responsavel
13 WHERE R.Data_Reserva >= TRUNC(SYSDATE + 1);

```

Figura 4: Consulta 1

Na consulta 1 retorna os nomes do funcionário responsável pela aprovação da reserva e do usuário responsável de todas as reservas esportivas que irão acontecer a partir do dia seguinte à consulta.

Nela, o uso dos JOIN internos possibilita, a partir da reserva esportiva, determinar o nome do usuário, do funcionário e a data dela, o que, com o WHERE, permite filtrar as para se devolver apenas as reservas que ainda irão ocorrer.

```

1 SELECT
2     r.ID_Reserva,
3     r.Data_Reserva,
4     r.Hora_Inicio,
5     r.Hora_Termino,
6     r.Instalacao,
7     r.Nro_Espaco,
8     r.Tipo_Reserva,
9     i.Nome AS Nome_Instalacao,
10    c.Nome AS Nome_Cidade,
11    e.Tipo AS Tipo_Espaco
12 FROM Reserva r
13 JOIN Instalacao_Esportiva i ON r.Instalacao = i.CNPJ
14 JOIN Cidade c ON i.Cidade = c.Codigo_Municipio
15 JOIN Espaco_Esportivo e ON r.Instalacao = e.Instalacao AND r.Nro_Espaco = e.Nro_Espaco
16 WHERE EXTRACT(YEAR FROM Data_Reserva) = EXTRACT(YEAR FROM SYSDATE);

```

Figura 5: Consulta 2

Nessa consulta estamos coletando as informações sobre ID da reserva, sua hora de início, de término, a data da reserva, o CNPJ da instalação e seu número, o tipo da reserva, o nome da instalação e da cidade, com o tipo de espaço de todas as reservas feitas no ano corrente.

Como na anterior, o uso dos JOINS internos permitem acessar dados não diretamente disponíveis em RESERVA, e no WHERE com EXTRACT do ano das datas, é possível comparar e retornar apenas as reservas do ano atual.

```

1 SELECT U.Nome, U.U_ID
2 FROM Usuario U
3 WHERE NOT EXISTS (
4   (SELECT Esp.Nro_Espaco FROM Espaco_Esportivo Esp WHERE Esp.Instalacao = '47.267.698/0001-79')
5 MINUS
6 (SELECT R.Nro_Espaco
7  FROM Usuario Ud JOIN Reserva_Esportiva RE ON RE.Usuario = Ud.U_ID
8  JOIN Reserva R ON R.ID_Reserva = RE.ID_Reserva
9  WHERE Ud.U_ID = U.U_ID AND R.Instalacao = '47.267.698/0001-79'
10 )
11 );

```

Figura 6: Consulta 3

Já na consulta 3, está sendo coletado quais usuários fizeram ao menos uma reserva em todos os *espaços esportivos* de uma *instalação esportiva* de CNPJ 47.267.698/0001-79. Esse CNPJ pode ser alterado para se extrair essa mesma informação de outras *instalações esportivas*.

Para fazer essa consulta, essas consultas aninhadas, com o MINUS, e EXISTS tem o papel de uma divisão da álgebra relacional, o que permite a busca desejada.

```

1 SELECT S.Servico, C.Orcamento, C.Empresa, C.ID_Contrato
2 FROM Instalacao_Esportiva I
3 JOIN Reserva R ON R.Instalacao = I.CNPJ
4 JOIN Manutencao M ON M.ID_Reserva = R.ID_Reserva
5 JOIN Contrato C ON C.ID_Contrato = M.Contrato
6 JOIN Servicos_Contrato S ON S.Contrato = C.ID_Contrato
7 WHERE C.Orcamento > 40000 AND I.CNPJ = '98.765.432/0001-98';

```

Figura 7: Consulta 4

Acima, fizemos uma consulta para serem exibidos o serviço, o valor do orçamento, qual o CNPJ da empresa e o ID dos contratos referentes a uma instalação esportiva específica, que tem o seu orçamento maior que determinado valor, na consulta utilizamos a instalação esportiva de CNPJ 98.765.432/0001-98 e o valor de 40.000 reais para filtrar os resultados, essa consulta é muito interessante para ser implementada de forma que esses valores que inserimos sejam alterados pelos usuários.

Nela, há o uso de vários JOINS internos para aquisição de dados em outras tabelas, e no WHERE comparações usando esses dados de tabelas diferentes.

```

1 SELECT F.Nome, F.F_ID, count(R.ID_Reserva) as Reservas
2 FROM Funcionario_Instalacao F LEFT JOIN Pessoa_Fisica P ON P.CPF = F.CPF
3 LEFT JOIN Reserva_Esportiva R ON R.Usuario = P.U_ID
4 GROUP BY F.F_ID, F.Nome;

```

Figura 8: Consulta 5

Para a consulta 5, estamos coletando a quantidade de *reservas* que *funcionários* das *instalações esportivas* fizeram como *usuários*, sendo que *funcionários* que não se cadastraram como *usuários*, ou não fizeram *reserva* ficam com a quantidade 0.

Nela, o uso de JOINS externos se faz necessário para não se perder funcionários que não tem uma conta de usuário e que não tem uma reserva como usuário. Além disso, por se desejar saber a quantidade de reservas, um GROUP BY é feito para agrupar as reservas feitas

pelo mesmo funcionário, com um COUNT sobre os ID\_Reserva das Reservas, para que ele retorne zero para funcionários sem reserva.

```
1 SELECT C.Empresa, M.Status
2 FROM Contrato C
3 JOIN Manutencao M ON C.ID_Contrato = M.Contrato
4 WHERE M.Status IN ('MARCADA', 'EM ANDAMENTO');
```

Figura 9: Consulta 6

E na consulta 6, está sendo devolvido quais *empresas de manutenção* ainda tem *contratos* em aberto, podendo estar com a situação de 'MARCADA' ou 'EM ANDAMENTO'.

Nesta consulta, há o uso do operador IN, o que reduz a necessidade de várias comparações com OR para saber se o Status é 'MARCADA' ou 'EM ANDAMENTO'.

```
1 SELECT C.Nome, I.Endereco_Rua, I.Endereco_Numero, I.Endereco_Bairro, I.Endereco_CEP
2 FROM Cidade C JOIN Instalacao_Esportiva I ON C.Codigo_Municipio = I.Cidade
3 JOIN Espaco_Esportivo E ON I.CNPJ = E.Instalacao
4 WHERE UPPER(E.Tipo) LIKE '%PISCINA%';
```

Figura 10: Consulta 7

Nessa última consulta estamos selecionando as cidades e os endereços das instalações esportivas que contém piscina. O tipo de espaço esportivo é deixado todo em letras maiúsculas pelo operador *UPPER* para não ocorrerem erros pela comparação com o *LIKE*, o qual utiliza de checagem de *substring* com os '%' em volta da palavra PISCINA, permitindo que essa busca ocorra.

## 7.4. Aplicação

O protótipo da aplicação oferece três telas principais: **Tela de Cadastro de Reservas Esportivas**, **Tela de Cadastro de Manutenção** e **Tela de Consulta de Reservas**. Abaixo, detalhamos o funcionamento de cada uma dessas telas. As inserções e as buscas implementadas na aplicação são realizadas no arquivo "db\_connection.py" localizado no diretório "src/backend/" do projeto. O construtor da classe Database é responsável por criar a conexão com o banco de dados e inicializar o cursor.

```

1  import cx_Oracle
2  import platform
3
4  class Database:
5
6      def __init__(self):
7          try:
8              # No Linux, não chame init_oracle_client se o LD_LIBRARY_PATH estiver configurado
9              if platform.system() == "Windows" and not cx_Oracle.clientversion():
10                 cx_Oracle.init_oracle_client(lib_dir=r"C:\\oracle\\instantclient_23_6") # Altere para o d
11          except Exception as e:
12              print("Erro ao inicializar o cliente Oracle:", e)
13
14          # Update these with your Oracle DB credentials
15          self.dsn = cx_Oracle.makedsn(
16              host="orclgrad1.icmc.usp.br",
17              port=1521,
18              service_name="pdb_elaine.icmc.usp.br"
19          )
20          self.connection = cx_Oracle.connect(
21              user="aNUSP",
22              password="senha",
23              dsn=self.dsn
24          )
25          self.cursor = self.connection.cursor()
26

```

Figura 11: Classe Database localizada no arquivo db\_connection.py

## Tela de Cadastro de Reservas Esportivas

Sistema de Reservas Esportivas

**Cadastrar Reserva Esportiva**

**Cadastrar Manutenção**

**Consultar Reservas**

Funcionário logado: 1

**Cadastro de Reserva**

ID Usuário:

Instalação Esportiva:

Espaço Esportivo:

Tipo de Reserva:

Quantidade de Pessoas:

Data da Reserva:

Horário de Início:

Horário de Término:

Figura 12: Tela Cadastro Reserva Esportiva

Nesta tela, o usuário pode registrar reservas esportivas de forma estruturada e interativa. O formulário inclui campos essenciais para garantir o correto registro na base de dados. Alguns destaques desta funcionalidade são:

- **Campo Usuário:**

Ao inserir o ID do usuário, a aplicação realiza automaticamente uma consulta na base de dados para verificar se o usuário existe. Caso o ID não seja encontrado, uma mensagem de aviso será exibida na tela, informando que o usuário não foi localizado.

A imagem mostra a interface de usuário para o 'Cadastro de Reserva'. O formulário contém os seguintes campos:

- ID Usuário:** Campo de texto com o valor '999' inserido.
- Instalação Esportiva:** Combobox com o texto 'Selecione uma instalação'.
- Espaço Esportivo:** Combobox com o texto 'Selecione uma instalação'.
- Tipo de Reserva:** Campo de texto com o texto 'Tipo de reserva (ex: Futebol, Basquete)'.
- Quantidade de Pessoas:** Campo de texto com o texto 'Digite a quantidade de pessoas'.
- Data da Reserva:** Campo de texto com o formato 'DD/MM/AAAA'.
- Horário de Início:** Campo de texto com o formato 'HH:MM'.
- Horário de Término:** Campo de texto com o formato 'HH:MM'.

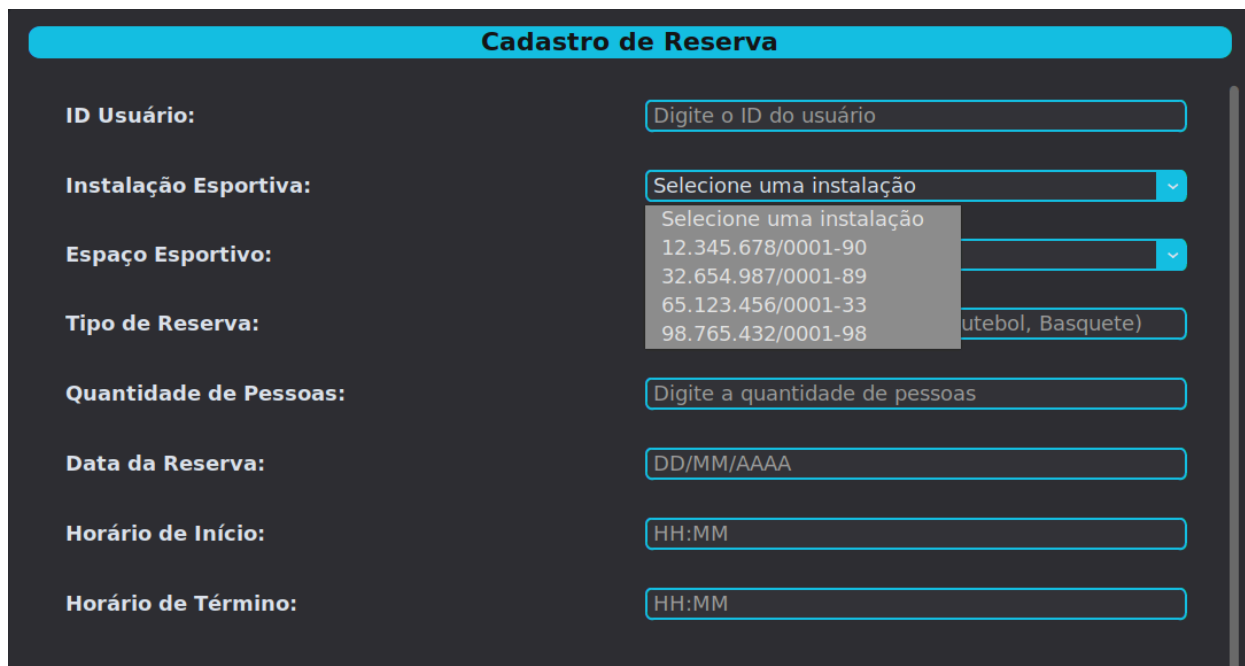
Uma caixa de diálogo de erro de validação está sobreposta ao formulário, com o título 'Erro de Validação' e o conteúdo 'Usuário não encontrado.' e um botão 'OK'.

Figura 13: Usuário não encontrado

- **Campo Instalação Esportiva (Combobox):**

Este campo realiza uma consulta na base de dados para preencher automaticamente os valores disponíveis no combobox, listando todos os CNPJs das instalações esportivas cadastradas.





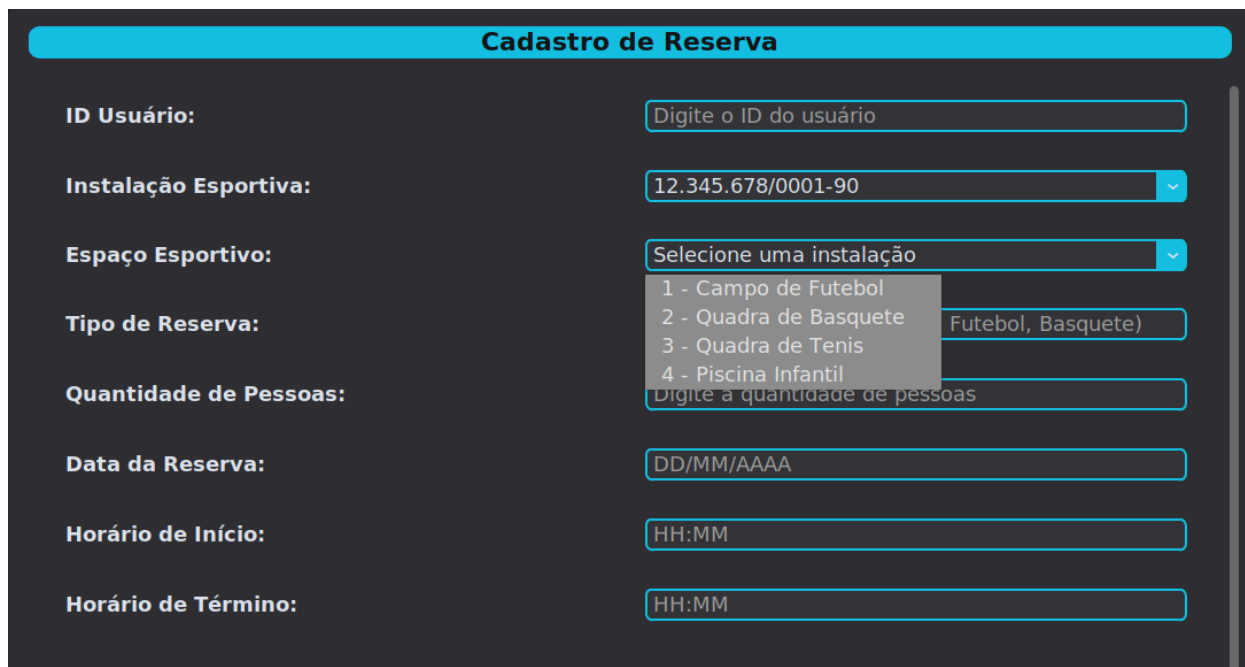
The image shows a web form titled "Cadastro de Reserva" with a dark blue header. The form contains several input fields and dropdown menus. The "Instalação Esportiva" dropdown is open, showing a list of options: "12.345.678/0001-90", "32.654.987/0001-89", "65.123.456/0001-33", and "98.765.432/0001-98". The "Espaço Esportivo" dropdown is also open, showing a list of options: "1 - Campo de Futebol", "2 - Quadra de Basquete", "3 - Quadra de Tenis", and "4 - Piscina Infantil". The "Tipo de Reserva" dropdown is open, showing a list of options: "Futebol", "Basquete", and "Tenis". The "Quantidade de Pessoas" field is empty. The "Data da Reserva" field is empty. The "Horário de Início" field is empty. The "Horário de Término" field is empty.

ID Usuário:	<input type="text" value="Digite o ID do usuário"/>
Instalação Esportiva:	<input type="text" value="Selecione uma instalação"/>
Espaço Esportivo:	<input type="text" value="Selecione uma instalação"/>
Tipo de Reserva:	<input type="text" value="Futebol, Basquete"/>
Quantidade de Pessoas:	<input type="text" value="Digite a quantidade de pessoas"/>
Data da Reserva:	<input type="text" value="DD/MM/AAAA"/>
Horário de Início:	<input type="text" value="HH:MM"/>
Horário de Término:	<input type="text" value="HH:MM"/>

Figura 14: Combobox Instalação Esportiva

- **Campo Espaço Esportivo (Combobox):**

Após a seleção de uma instalação esportiva, a aplicação executa outra consulta para carregar os espaços esportivos vinculados àquela instalação. Os valores do combobox exibem o número do espaço esportivo e seu nome, como por exemplo: "3 - Quadra de Basquete".



The image shows the same "Cadastro de Reserva" form, but now the "Espaço Esportivo" dropdown is open, showing a list of options: "1 - Campo de Futebol", "2 - Quadra de Basquete", "3 - Quadra de Tenis", and "4 - Piscina Infantil". The "Instalação Esportiva" dropdown is now closed and shows the selected value "12.345.678/0001-90". The "Tipo de Reserva" dropdown is open, showing a list of options: "Futebol", "Basquete", and "Tenis". The "Quantidade de Pessoas" field is empty. The "Data da Reserva" field is empty. The "Horário de Início" field is empty. The "Horário de Término" field is empty.

ID Usuário:	<input type="text" value="Digite o ID do usuário"/>
Instalação Esportiva:	<input type="text" value="12.345.678/0001-90"/>
Espaço Esportivo:	<input type="text" value="Selecione uma instalação"/>
Tipo de Reserva:	<input type="text" value="Futebol, Basquete"/>
Quantidade de Pessoas:	<input type="text" value="Digite a quantidade de pessoas"/>
Data da Reserva:	<input type="text" value="DD/MM/AAAA"/>
Horário de Início:	<input type="text" value="HH:MM"/>
Horário de Término:	<input type="text" value="HH:MM"/>

Figura 15: Combobox Espaço Esportivo

```

26
27     def insert_reservation(self, data):
28         """
29         Inserts reservation data into the database.
30         :param data: A dictionary containing reservation details.
31         """
32         # Generate a new ID_Reserva
33         self.cursor.execute("SELECT NVL(MAX(ID_Reserva), 0) + 1 FROM Reserva")
34         new_id_reserva = self.cursor.fetchone()[0]
35
36         # Prepare SQL statements
37         insert_reserva_sql = """
38         INSERT INTO Reserva (
39             ID_Reserva,
40             Data_Reserva,
41             Hora_Inicio,
42             Hora_Termino,
43             Instalacao,
44             Nro_Espaco,
45             Tipo_Reserva
46         ) VALUES (
47             :id_reserva,
48             TO_DATE(:data_reserva, 'DD/MM/YYYY'),
49             TO_TIMESTAMP(:hora_inicio, 'HH24:MI'),
50             TO_TIMESTAMP(:hora_termino, 'HH24:MI'),
51             :instalacao,
52             :nro_espaco,
53             :tipo_reserva
54         )
55         """

```

Figura 16: Início do método que insere reservas esportivas na base de dados. A imagem contempla o código SQL do insert realizado na entidade genérica “Reserva”.

```
56
57     insert_reserva_esportiva_sql = """
58     INSERT INTO Reserva_Esportiva (
59         ID_Reserva,
60         Usuario,
61         Nome_Reserva,
62         Quantidade_Pessoas,
63         Tipo,
64         Funcionario_Responsavel,
65         Aprovado
66     ) VALUES (
67         :id_reserva,
68         :usuario,
69         :nome_reserva,
70         :quantidade_pessoas,
71         :tipo,
72         :funcionario_responsavel,
73         :aprovado
74     )
75     """
76
```

Figura 17: Continuação do método insert\_reservation. A imagem contempla o insert SQL feito na entidade específica "Reserva\_Esportiva".

```

76
77     # Bind parameters
78     reserva_params = {
79         'id_reserva': new_id_reserva,
80         'data_reserva': data['data_reserva'],
81         'hora_inicio': data['hora_inicio'],
82         'hora_termino': data['hora_termino'],
83         'instalacao': data['instalacao'],
84         'nro_espaco': data['nro_espaco'],
85         'tipo_reserva': 'RESERVA ESPORTIVA'
86     }
87
88     reserva_esportiva_params = {
89         'id_reserva': new_id_reserva,
90         'usuario': data['usuario_id'],
91         'nome_reserva': data['nome_reserva'],
92         'quantidade_pessoas': data.get('quantidade_pessoas'),
93         'tipo': data.get('tipo'),
94         'funcionario_responsavel': data['funcionario_responsavel'],
95         'aprovado': 'EM ANALISE'
96     }
97
98     try:
99         # Execute insertion queries
100         self.cursor.execute(insert_reserva_sql, reserva_params)
101         self.cursor.execute(insert_reserva_esportiva_sql, reserva_esportiva_params)
102         self.connection.commit()
103     except cx_Oracle.DatabaseError as e:
104         error, = e.args
105         print("Oracle-Error-Code:", error.code)
106         print("Oracle-Error-Message:", error.message)
107         self.connection.rollback()
108         raise

```

Figura 18: Parte final do método "insert\_reservation".

## Tela de Cadastro de Manutenção

A tela de cadastro de manutenção segue uma lógica semelhante à de cadastro de reservas, mas com algumas diferenças específicas para atender às particularidades do processo de manutenção.

The image shows a web application window titled 'Sistema de Reservas Esportivas'. On the left is a sidebar with three menu items: 'Cadastrar Reserva Esportiva', 'Cadastrar Manutenção' (which is highlighted), and 'Consultar Reservas'. The main area displays a form titled 'Cadastro de Manutenção'. The form contains the following fields: 'Instalação Esportiva:' with a dropdown menu showing 'Selecione uma instalação'; 'Espaço Esportivo:' with a dropdown menu showing 'Selecione uma instalação'; 'Contrato:' with a dropdown menu showing 'Selecione um contrato'; 'Tipo de Manutenção:' with a text input field containing the placeholder 'Digite o tipo de manutenção (ex: Limpeza, Pintura)'; 'Data da Manutenção:' with a text input field containing the placeholder 'DD/MM/AAAA'; 'Horário de Início:' with a text input field containing the placeholder 'HH:MM'; and 'Horário de Término:' with a text input field containing the placeholder 'HH:MM'. At the bottom of the form are two buttons: 'Limpar' and 'Salvar'. In the bottom left corner of the sidebar, it says 'Funcionário logado: 1'.

Figura 19: Tela Cadastro de Manutenção

- **Campos de Instalação Esportiva e Espaço Esportivo:**  
Esses campos funcionam da mesma forma que na tela de cadastro de reservas, realizando consultas na base de dados para preencher os valores disponíveis.
- **Campo Contrato:**  
Um campo adicional e essencial nesta tela é o campo de "Contrato". Ele realiza uma consulta na base de dados e exibe os contratos de manutenção disponíveis para a instalação esportiva selecionada. Isso garante que apenas contratos válidos sejam utilizados durante o registro.

```

110 def insert_maintenance(self, data):
111     """
112     Inserts maintenance data and its associated reservation into the database.
113     :param data: A dictionary containing reservation and maintenance details.
114     """
115     # Generate a new ID_Reserva
116     self.cursor.execute("SELECT NVL(MAX(ID_Reserva), 0) + 1 FROM Reserva")
117     new_id_reserva = self.cursor.fetchone()[0]
118
119     # Prepare SQL for Reserva
120     insert_reserva_sql = """
121     INSERT INTO Reserva (
122         ID_Reserva,
123         Data_Reserva,
124         Hora_Inicio,
125         Hora_Termino,
126         Instalacao,
127         Nro_Espaco,
128         Tipo_Reserva
129     ) VALUES (
130         :id_reserva,
131         TO_DATE(:data_reserva, 'DD/MM/YYYY'),
132         TO_TIMESTAMP(:hora_inicio, 'HH24:MI'),
133         TO_TIMESTAMP(:hora_termino, 'HH24:MI'),
134         :instalacao,
135         :nro_espaco,
136         :tipo_reserva
137     )
138     """
139

```

Figura 20: Início do método “insert\_maintenance”, responsável por inserir uma manutenção na base de dados. A imagem contempla o insert SQL feito para inserir na entidade genérica “Reserva”.

```

139
140     # Prepare SQL for Manutencao
141     insert_manutencao_sql = """
142     INSERT INTO Manutencao (
143         ID_Reserva,
144         Contrato,
145         Tipo,
146         Status
147     ) VALUES (
148         :id_reserva,
149         :contrato,
150         :tipo,
151         :status
152     )
153     """
154
155     # Bind parameters for Reserva
156     reserva_params = {
157         'id_reserva': new_id_reserva,
158         'data_reserva': data['data_reserva'],
159         'hora_inicio': data['hora_inicio'],
160         'hora_termino': data['hora_termino'],
161         'instalacao': data['instalacao'],
162         'nro_espaco': data['nro_espaco'],
163         'tipo_reserva': 'MANUTENÇÃO'
164     }
165
166     # Bind parameters for Manutencao
167     manutencao_params = {
168         'id_reserva': new_id_reserva,
169         'contrato': data['contrato'],
170         'tipo': data.get('tipo', None),
171         'status': data['status'].upper() # Ensure the status is uppercase to match
172     }

```

Figura 21: Parte final do método “insert\_maintenance”. A imagem contempla o insert SQL feito para inserir os dados na entidade específica “Manutenção”.

## Tela de Consulta de Reservas

A tela de consulta de reservas permite ao usuário buscar informações sobre reservas esportivas e manutenções, utilizando filtros para facilitar a pesquisa.

- **Filtros Disponíveis:**

- **Data:** Permite buscar reservas realizadas em uma data específica.
- **Instalação Esportiva e Espaço Esportivo:** Funcionam de forma semelhante às telas de cadastro, carregando os valores diretamente da base de dados.

Para executar uma busca, é obrigatório selecionar pelo menos um dos filtros: **data** ou **instalação esportiva**. Isso garante a eficiência e precisão das consultas realizadas no banco de dados.

Sistema de Reservas Esportivas

**Consulta de Reservas**

ID Reserva: 1  
Data: 15/12/2024  
Horário: 08:00 - 10:00  
Instalação: Estádio Municipal  
CNPJ: 12.345.678/0001-90  
Cidade: São Paulo  
Espaço Esportivo: Campo de Futebol  
Tipo de Reserva: RESERVA ESPORTIVA

ID Reserva: 4  
Data: 16/02/2025  
Horário: 13:00 - 15:00  
Instalação: Estádio Municipal

Data:

Instalação Esportiva:

Espaço Esportivo:

Funcionário logado: 1

Figura 22: Tela Consulta de Reservas



```

190
191 def search_reservations(self, filters):
192     """
193     This method searches for reservations based on the given filters.
194     :param filters: A dictionary containing filters for the query.
195     :return: A list of reservations that match the given filters.
196     """
197
198     query = """
199     SELECT
200         r.ID_Reserva,
201         r.Data_Reserva,
202         r.Hora_Inicio,
203         r.Hora_Termino,
204         r.Instalacao,
205         r.Nro_Espaco,
206         r.Tipo_Reserva,
207         i.Nome AS Nome_Instalacao,
208         c.Nome AS Nome_Cidade,
209         e.Tipo AS Tipo_Espaco
210     FROM Reserva r
211     JOIN Instalacao_Esportiva i ON r.Instalacao = i.CNPJ
212     JOIN Cidade c ON i.Cidade = c.Codigo_Municipio
213     JOIN Espaco_Esportivo e ON r.Instalacao = e.Instalacao AND r.Nro_Espaco = e.Nro_Espaco
214     WHERE 1=1
215     """
216
217     params = {}
218     if filters.get('reservation_date'):
219         query += " AND r.Data_Reserva = TO_DATE(:reservation_date, 'DD/MM/YYYY')"
220         params['reservation_date'] = filters['reservation_date']
221     if filters.get('installation'):
222         query += " AND r.Instalacao = :installation"
223         params['installation'] = filters['installation']
224     if filters.get('space'):
225         query += " AND r.Nro_Espaco = :space"
226         params['space'] = filters['space']

```

Figura 23: Método “search\_reservation” que realiza as buscas de reservas esportivas e manutenções.

## Tratamento de Erros

Um ponto importante na implementação da aplicação foi o cuidado com o **tratamento de erros** durante as inserções de dados. Antes de registrar qualquer informação na base de dados, a aplicação realiza validações detalhadas para garantir que os dados fornecidos pelo usuário sejam válidos e consistentes. Caso algum erro seja detectado, como o preenchimento incorreto de um campo ou a ausência de informações obrigatórias, mensagens de erro claras e intuitivas são exibidas na tela. Esse mecanismo não apenas melhora a experiência do usuário, mas também evita inconsistências ou falhas no banco de dados, garantindo a integridade das informações armazenadas.

## 8. Conclusão

Nesse projeto foi imaginado e concebido um banco de dados e aplicação para modelar e gerir reservas de Instalações Esportivas e seus Espaços Esportivos, dentro do tema proposto de **Incentivo ao Esporte**. Nele teve-se a junção de todos os conhecimentos adquiridos durante o semestre da disciplina de Bases de Dados, desde a modelagem com MER, tradução em tabelas com Diagrama Relacional, até buscas em SQL que usam conceitos de Álgebra Relacional (por exemplo, a divisão).

A seguir temos as opiniões, críticas e sugestões de alguns membros do grupo em relação a disciplina e ao trabalho:

- "Achei a disciplina e o tema do trabalho muito legal. Gostei bastante de como ambos conversavam bem: em um momento estávamos vendo na teoria como as ferramentas funcionam, e no trabalho colocávamos elas em prática. Não tenho críticas grandes nem ao trabalho ou à disciplina, acho que gostaria de ter tido a chance de colocar mais da Álgebra Relacional em prática, mas entendo que esse não é o foco da matéria."
- "Eu gostei bastante da disciplina como um todo, estava com as expectativas altas e sinto que foram atendidas. A forma como foi passado o conteúdo e colocado em prática posteriormente, tanto em aulas de laboratório quanto no projeto, acho que não poderia ter sido melhor. Também gostei bastante dos exercícios de fixação ao final de cada aula, ajudaram imensamente no entendimento dos conteúdos. Só gostaria de pontuar um único detalhe sobre a disciplina, mesmo que não esteja muito no controle da professora ou do monitor. Eu senti que as aulas de terça-feira e quinta-feira acabaram ficando um pouco puxadas no quesito prazo para resolução dos exercícios de fixação. Quando era dado um exercício de fixação na terça-feira para entrega na quinta, acabava sendo corrido para fazer a resolução a tempo. Não vejo como algo crítico, até porque prefiro que tenha sempre um exercício desses até mesmo nas terças, do que não ter. Mas se fosse possível mudar os dias de oferecimento dessa disciplina, uma aula na segunda e outra na sexta seria muito mais interessante. Enfim, gostaria de agradecer à professora Elaine e ao monitor Afonso pelo comprometimento, paciência e dedicação ao longo do curso. A abordagem didática de ambos fez toda a diferença no meu aprendizado, e tenho certeza de que foi um diferencial para toda a turma. Fica aqui meu reconhecimento e gratidão por essa experiência tão enriquecedora."