# GERENCIAMENTO DE SISTEMAS DE INFORMAÇÃO DE UM CLUBE SOCIAL: DESENVOLVIMENTO DE UMA APLICAÇÃO EM PYTHON

Ana Júlia Barros Ferreira (UFCG) julia.barros@estudante.ufcg.edu.br
Cayllane Nathalia Silva Simão (UFCG) cayllane.nathalia@estudante.ufcg.edu.br
Cecir Barbosa de Almeida Farias (UFCG) cecir.barbosa@professor.ufcg.edu.br
Iza Maria da Silva Nunes (UFCG) iza.maria@estudante.ufcg.edu.br
Nathália da Lima Odon (UFCG) nathaliaodon1@gmail.com

**Resumo**

O presente artigo possui como objetivo apresentar o desenvolvimento de um banco de dados por meio da aplicação de ferramentas de modelagem de sistemas como diagrama use case, diagrama de entidade e relacionamento e modelo relacional, concentrando-se especificamente na utilização do SQL – *Structured Query Language* (Linguagem de Consulta Estruturada), no software *DB browser* para montagem do banco de dados e também a linguagem de programação *Python* para a interface e interação dos códigos. O objeto da pesquisa é a realização da análise e desenvolvimento de um Sistema de Banco de Dados para um clube social de acordo com a necessidade de armazenar dados e obter informações cada vez mais ágeis, de modo a aumentar a produtividade e eficiência do sistema, e auxiliar gestores do clube no controle e tomada de decisões. Concluiu-se que a utilização de um sistema de banco de dados permite que a organização obtenha acesso rápido ao realizar consultas acerca da lista de associados, bem como atividades e eventos proporcionados pelo clube, assim como a possibilidade de gerar alterações e futuras inserções ou exclusão de dados.

**Palavras-Chaves:** Banco de dados, *SQLite*, *DB browser*, ferramenta organizacional

## 1. Introdução

Em um cenário mundial em constante evolução, a relação com os dados e as informações é crescente em todos os âmbitos, de modo que é imprescindível para qualquer organização que

deseje se manter competitiva e relevante no mercado, uma gestão eficiente dessas informações por meio de sistemas de informações.

Por conseguinte, com a explosão de dados e sua rápida dissipação, os membros de um sistema estão sempre em busca de experiências ágeis e personalizadas, assim como de uma garantia a segurança de dados e a privacidade, logo um sistema de banco de dados bem projetado é crucial para proteger as informações e tornar a experiência no sistema mais agradável. Desta maneira, a montagem de um sistema de banco de dados, bem como as ferramentas utilizadas anteriormente e durante seu gerenciamento alavancam a eficácia e eficiência do sistema gerado, assim como do negócio ao qual se está aplicado.

Neste contexto, este artigo visa apresentar um sistema de banco de dados para um clube social, levando em conta que clubes possuem diversas atividades e eventos, assim como existe necessidade de se manter registros e atualizações a respeito dos associados, o que torna o gerenciamento de informações complexo e propenso a erros. Um sistema de banco de dados eficaz permite que um clube colete e analise dados relacionados a frequência, preferências e necessidades dos associados, fornecendo uma base sólida para tomadas de decisões estratégicas e oferecendo oportunidades para oferecer experiências mais relevantes e envolventes.

Sendo assim, este artigo visa explorar a importância do sistema, não apenas para simplificar e aprimorar a administração do clube, mas também para manter a organização em sintonia com a realidade tecnológica do mundo atual. Ao longo do artigo, são discutidos os componentes essenciais do sistema e como ele pode impactar positivamente a eficiência operacional, envolvimento dos associados e competitividade de um clube.

## 2. Revisão bibliográfica

### 2.1 Sistemas de informação

"Um sistema pode ser definido simplesmente como um grupo de elementos inter-relacionados ou em interação que formam um todo unificado. Um sistema dessa ordem (às vezes chamado sistema dinâmico) possui três componentes ou funções básicas em interação" (O'Brien, 2004).

Um sistema de informação caracteriza-se como sendo "toda ferramenta que manipula dados, transformando-os em informações, utilizando ou não meios tecnológicos para isso. A entrada de dados é geralmente feita manualmente, no decorrer do processo se verifica o tratamento

desses dados, onde eles são processados e transformados por meio de tecnologias" (Gonçalves, 2006).

Os sistemas de informação atuam para que "os dados sejam tratados e para que seja eficiente é necessário que todas as informações sejam inseridas adequadamente, só assim, poderá obter um controle e tomada de decisões eficiente" (Oliveira, 2000).

Conforme Laudon (2004), "um SI consegue ser definido como um conjunto de componentes inter-relacionados, que realizam coletas, armazenamentos, processamentos e distribuem informações destinadas para auxiliar nas tomadas de decisões, coordenação e controle de uma organização. Além disto, esses sistemas auxiliam gerentes e trabalhadores a analisarem problemas, visualizar assuntos complexos e criar produtos".

## 2.2 Diagrama de *use case*

Segundo Schneider (1998), "o diagrama de Use Case formalizado em UML (*Unified Modelling Language)* possui foco na descrição do uso de um sistema utilizando atores, onde um ator representa os elementos externos que interagem com o sistema". Desta maneira, o diagrama descreve uma sequência a ser seguida por um usuário quando ele está em interação com o sistema buscando realizar uma tarefa.

Desta forma, o diagrama apresenta a possibilidade de gerar vários cenários, surgindo como uma instância de um *use case*, cada diagrama envolve um cenário de uso por um ator e durante esse uso, várias sequências podem ser seguidas, dependendo do contexto do sistema.

## 2.3 Diagrama de entidade e relacionamento (DER)

Conforme Ribeiro (1992), o "Diagrama de Entidade e Relacionamento (DER) trata de um modelo popular para modelagem conceitual de dados, baseado em três conceitos básicos: entidades, atributos e relacionamentos, onde as entidades representam um elemento do mundo real, o atributo pode ser simples ou atômico, e se refere as características ou propriedades que descrevem as entidades do banco de dados e os relacionamentos descrevem a forma como uma entidade se relaciona com a outra e os seus atributos".

"Para um banco de dados toda entidade possui um ou mais atributos chave, cujos valores são distintos para cada entidade do conjunto de entidades. Portanto, os valores desses atributos podem identificar uma única entidade" (Santos, 2017).

"Todo tipo de relacionamento contém um grau, que indica o número de tipos de entidade que participam desse relacionamento. Para os tipos de relacionamento de grau dois, ou binários, tem-se dois tipos de restrições: cardinalidade e participação, a cardinalidade especifica o número máximo de instâncias do relacionamento em que uma entidade pode participar e a participação envolve o relacionamento de entidades em uma associação" (Muslah; Ghoul, 2019).

## 2.4 Modelo relacional (MR)

O Modelo Relacional (MR) trata de um modelo de dados representativo proposto por Ted Codd em 1970 fundamentando-se basicamente em conceitos matemáticos. Desta maneira, os primeiros sistemas comerciais baseados no MR foram disponibilizados em 1980 e após isso, ele tem sido implementado em vários sistemas.

"O MR trata de três aspectos principais dos dados: estrutura de dados, integridade e manipulação. Sendo assim, neste modelo o BD é representado como um conjunto de relações, uma relação se comporta de modo similar à uma tabela de valores e segundo a terminologia do MR diz-se que as linhas se denominam *tuplas*; as colunas, *atributos*; e a tabela em si, *relação*" (Navathe, 2011).

## 2.5 Sistemas de banco de dados

Segundo Date (2014), "Um banco de dados se trata de uma coleção persistente de dados usada pelos sistemas de aplicação de uma empresa, é um local onde informações necessárias para as operações de uma organização são armazenadas".

Além disso, um banco de dados oferece várias vantagens, incluindo:

Controle Centralizado de Dados: Os dados estão concentrados em um único local, o que proporciona um maior controle.

Controle de Redundância, Economia de Espaço e Compartilhamento de Dados: Em processamento de arquivos convencional, a mesma informação é frequentemente duplicada em vários arquivos, levando a desperdício de espaço de armazenamento. Em um Banco de Dados, os dados são armazenados apenas uma vez e podem ser compartilhados por vários usuários.

Eliminação de Inconsistências e Garantia de Integridade: Em métodos tradicionais baseados em arquivos, informações inconsistentes podem surgir devido a repetição de dados armazenados. No entanto, em Bancos de Dados, a consistência e a integridade dos dados podem ser mantidas.

Estabelecimento de Padrões e Facilidade de Acesso aos Dados: Devido à centralização dos dados em um Banco de Dados, é mais fácil estabelecer padrões de nomenclatura e documentação. Isso facilita a recuperação eficiente de informações, ao contrário do armazenamento convencional onde os dados estão em vários formatos e linguagens de programação diferentes são usadas.

Independência de Dados: Em sistemas de arquivos, a estrutura de armazenamento e o método de acesso aos dados estão incorporados ao código das aplicações, tornando-as dependentes dos dados. Por outro lado, Bancos de Dados permitem a independência de dados, permitindo a abstração de dados.

## 2.6 Linguagem SQL

Segundo Alura (2023), "os anos de 1970 começaram a viabilizar a implementação de um banco de dados baseado em um modelo teórico denominado Banco de Dados Relacional, tal Banco de Dados possibilita realização de pesquisas mais avançadas, aproveitando a ligação dos dados que se relacionam de uma tabela para outra".

Deste modo, conforme Cardoso (2017), "a linguagem SQL - *STRUCTURED QUERY LANGUAGE* (Linguagem de consulta estruturada) surgiu na década de 70 com base nos modelos de dados relacionais, quando a empresa IBM, desenvolveu a linguagem SEQUEL - *SEQUENCE QUERY LANGUAGE* (Linguagem de consulta sequencial), cujo objetivo era implementar o modelo relacional de dados, e assim posteriormente o termo evoluiu e se reduziu para SQL".

A linguagem SQL busca armazenar e processar informações em um Banco de Dados Relacional. Ela permite armazenar informações em formato tabular, com linhas e colunas representando diferentes atributos de dados e as várias relações entre os valores dos dados, podendo utilizar instruções SQL para armazenar, atualizar, remover, pesquisar e recuperar informações do banco de dados, também pode-se usar SQL para otimizar a performance do banco de dados. Segundo Cardoso (2017) a SQL é formada por várias partes com propósitos específicos:

- "DLL (Linguagem de definição de dados): Trata-se de uma parte que permite determinar o esquema do banco de dados, assim como realizar alteração e excluí-lo;
- DML (Linguagem de Manipulação de dados): Permite determinar a manipulação dos dados, a inclusão, alteração ou exclusão de dados;
- DCL (Linguagem de Controle de Dados): Permite controlar a licença e autorização de acesso dos usuários aos dados;
- DTL (Linguagem de Transação de dados): Oferece comandos para trabalhar com transações;
- DQL (Linguagem de Consulta de Dados): Proporciona a consulta de dados".

## 2.7 Linguagem de programação *Python*

De acordo com Borges (2010), "a linguagem de programação Python foi criada em 1990 por Guido Van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda (CWI). Além disso, a linguagem Python foi criada a partir da linguagem ABC que já existia na época".

Atualmente, *Python* tem grande destaque e movimenta diversos usuários por ser uma linguagem poderosa e de fácil compreensão. Com ela é possível desenvolver diversos tipos de trabalhos e projetos, como: aplicativos, automações, jogos, análise de dados, inteligência artificial, entre outros.

*"Python* é interessante por sua simplicidade e clareza, ainda assim é uma linguagem poderosa, podendo ser usada para administrar sistemas e desenvolver grandes projetos. É uma linguagem clara e objetiva, que vai direto ao ponto" (Menezes, 2014).

"A linguagem inclui diversas estruturas de alto nível, como listas e dicionários, e coleções de módulos prontos para uso, além de frameworks de terceiros que podem ser adicionados. A linguagem é interpretada através de *bytecode* pela máquina virtual *Python*, tornando o código portável. Com isso, é possível compilar aplicações em uma plataforma e rodar em outros sistemas ou executar o direto do código da fonte" (Borges, 2010).

## 3. Metodologia

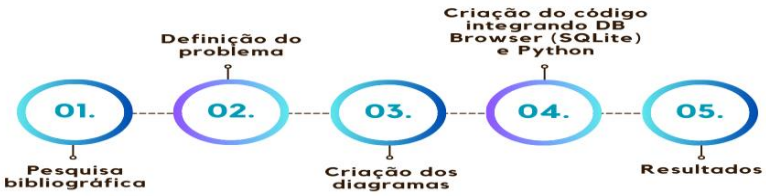
A metodologia utilizada neste trabalho é classificada como: aplicada, qualitativa, descritiva e bibliográfica.

No que se refere à sua natureza, a pesquisa pode ser classificada como aplicada, pois o principal objetivo é "gerar conhecimentos para aplicações práticas visando a resolução de um problema específico" (Silva, 2005). Quanto aos objetivos, a pesquisa é classificada como "descritiva, pois tem a finalidade de descrever as características de um determinado caso, estabelecendo relações entre as variáveis por meio de uso de técnicas padronizadas de coleta de dados" (Gil, 2002).

Pode-se classificar a abordagem do problema de acordo com a pesquisa qualitativa, a qual "faz uma interpretação dos fenômenos e a atribuição de significados através de uma análise indutiva do pesquisador, dispensando assim o uso de métodos estatísticos" (Silva, 2005). Do ponto de vista dos procedimentos técnicos, é classificada como "pesquisa bibliográfica, pois foi realizado um estudo geral sobre os principais trabalhos referentes ao tema do estudo por meio de livros, artigos científicos e sites" (Lakatos, 2003).

Na Figura 1 é possível observar as etapas metodológicas para a formulação deste trabalho.

**Figura 1** - Fluxograma Metodológico




**Fonte:** Autoria Própria (2023)

A princípio foi realizado um estudo acerca de Sistemas de Informação, Modelo Relacional, Banco de Dados e Linguagem SQL, com a finalidade de reunir informações e conhecimentos pertinentes para a resolução do problema proposto. Paralelo a isso, na primeira etapa do projeto, foram descritos objetivos e funcionalidades do sistema, bem como diagramas referentes aos *Use Case* e Entidades e Relacionamentos, e por fim, foi descrito o Modelo Relacional o após o Diagrama de Entidade e Relacionamento.

Posteriormente, para a segunda etapa do projeto, com o objetivo de alcançar algumas exigências do problema em questão, foi utilizado o software *DB Browser (SQLite)* para a criação e manutenção do Banco de Dados referente ao armazenamento de informações necessárias, além da linguagem *Python* no ambiente de desenvolvimento *VS Code.* Ambas as ferramentas foram

utilizadas de forma integrada na criação de códigos com a finalidade de tornar automático o sistema de cadastramento e consulta de um clube social.

## 4. Resultados

### 4.1 Etapa 1 - análise do sistema de informação do clube

Inicialmente foi realizada uma análise do sistema de informação para sócios de um clube, com isso, se determinou que o Banco de Dados em questão tem por objetivo registrar e manter informações sobre os sócios de um clube, bem como a interação deles com atividades e eventos. Desta forma, foram elencados os principais objetivos e as funcionalidades deste Sistema de Banco de Dados.

Sendo assim, o objetivo principal consiste em gerenciar os sócios, disponibilizando um cadastro que pode ser consultado, atualizado ou excluído, contendo informações pertinentes sobre cada sócio, como número de matrícula, nome, endereço, telefone, status, entre outros. Além disso, o sistema também contém o registro de atividades, que são criadas e mantidas por código e nome e são oferecidas pelo clube. No que se refere às atividades, o sistema registra a participação de cada sócio nas atividades escolhidas por eles.

Outra funcionalidade é o controle de eventos, onde o sistema irá cadastrar quais são os eventos realizados pelo clube identificando qual o seu código, descrição e data. No que se refere aos eventos, o sistema rastreia o tipo de evento que o sócio participa, se é público ou privado, e permite que apenas os sócios de classe "A" se inscrevam e participem de eventos privados.

Com o sistema do banco de dados se torna possível consultar informações pessoais dos sócios e quais/quantas atividades e eventos eles participaram. É possível consultar as datas que o evento ocorreu, bem como seu tipo. No que se refere aos registros em atividades ou eventos, o sistema libera ou rejeita a inscrição dos sócios dependendo das restrições do clube.

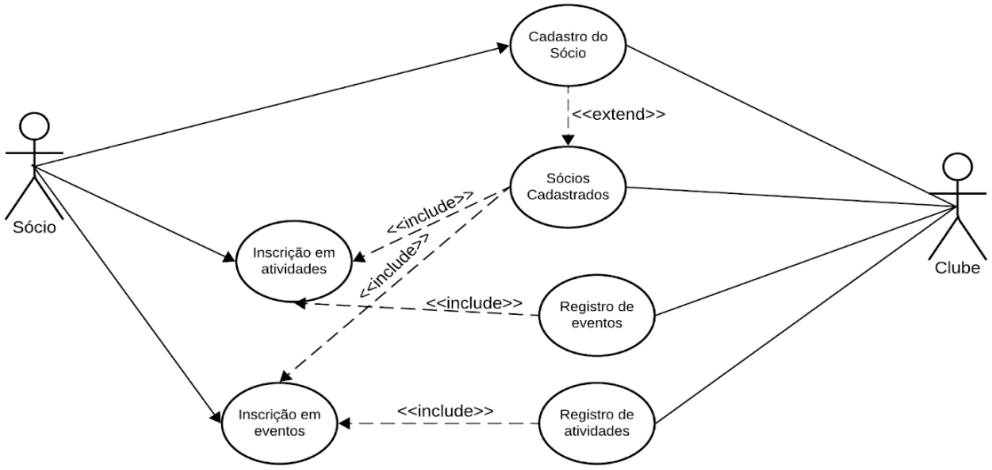
#### 4.1.1 Diagrama *use case*

Foram analisadas as relações presentes entre os atores do sistema do clube, onde os atores existentes são "sócio" e "clube", que representam os elementos externos que interagem com o sistema, bem como através do diagrama foram elencadas as relações de extensão e inclusão presentes para melhor demonstrar o fluxo de eventos.

O objetivo do Diagrama de *Use Case* é capturar a funcionalidade do sistema em questão, a partir da perspectiva dos atores e dos casos de uso. Deste modo, o diagrama gerado para o sistema do clube social, é apresentado na Figura 2.

Por meio do diagrama, nota-se que o caso de uso de cadastro do sócio e de sócios cadastrados possuem uma relação de extensão, devido ao contexto condicional entre os casos de uso. Para os demais casos de uso foi percebida uma relação de inclusão onde existe um conjunto de passos comuns.

**Figura 2** - Diagrama de Use Case




**Fonte:** Autoria Própria (2023)

### 4.1.2 Diagrama de entidades e relacionamentos (DER)

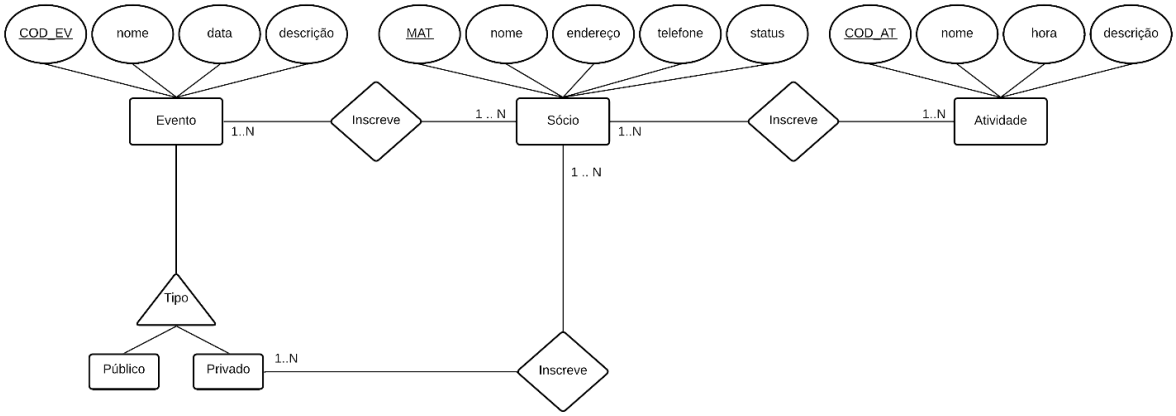
Nesta etapa foram formuladas as entidades presentes no sistema, os atributos e os relacionamentos existentes entre eles. Com entidades tem-se o sócio, a atividade e o evento. Para cada entidade em questão são estabelecidos atributos que caracterizam as entidades, bem como as chaves que são elementos cruciais para a identificação exclusiva de registros dentro de uma tabela ou entidade. Estas chaves são divididas em primárias e secundárias, para primárias temos as usadas para identificar registros exclusivos em uma tabela, para as secundárias as usadas para estabelecer relacionamentos entre entidades em um DER. Desta maneira, para este sistema temos para as entidades:

- Sócio: (MAT, nome, endereço, telefone);
- Atividades: (COD_AT, nome, hr e descrição);
- Eventos: (COD_EV, nome, data e descrição).

Por conseguinte, para a entidade sócio existe uma classificação quanto à sua classe podendo ser A, B ou C. Da mesma forma, os eventos podem ainda ser classificados em tipos privado ou público.

Uma das regras do clube é que apenas os sócios classe A se inscrevam em eventos privados, diante disso a Figura 3 representa as relações existentes entre as entidades e os relacionamentos (DER), juntamente com suas cardinalidades.

**Figura 3** - Diagrama de Entidade e Relacionamento dos Sócios de um Clube




**Fonte:** Autoria Própria (2023)

### 4.1.3 Modelo relacional (MR)

Nesta etapa, foi realizada a transformação do Diagrama de Entidade e Relacionamentos (DER) para o Modelo Relacional (MR). Essa transformação envolve a conversão das entidades, atributos, relacionamentos e restrições representados no DER em tabelas, colunas e chaves primárias e estrangeiras no Modelo Relacional que é a estrutura que será implementada no sistema de banco de dados. Com isso para o sistema em questão o MR se apresenta do seguinte modo:

Sócios (MAT, nome, end, tel, status);
Atividades (COD_AT, nome, hr, descrição_at);
Eventos (COD_EV, nome, data, descrição_ev, tipo);
Ev_publico (COD_EV, colaborador);
Ev_privado (COD_EV, empresa_res);
Inscrição_atividades (ID_INS_AT, MAT, COD_AT);
Inscrição_eventos (ID_INS_EV, MAT, COD_EV).

## 4.2 Etapa 2 - desenvolvimento e testes do sistema de banco de dados no software db browser sqlite

### 4.2.1 Criação de tabelas

Iniciando a segunda etapa, para organizar as informações foram desenvolvidas inicialmente, foi utilizado o software *DB Browser SQLite* para a criação de todas as tabelas do sistema, referentes aos cadastros dos sócios, as atividades e eventos que o clube dispõe, bem como a inscrições dos sócios nas atividades e eventos contendo as chaves necessárias para descrição de cada entidade.

Todas as tabelas do sistema foram criadas com os comandos CREATE TABLE. Na Figura 4 está representado o código de criação para a tabela de Sócios do sistema, informando o nome da tabela e seus atributos. Na Figura 5 está representado o código de criação para a tabela de Eventos do sistema e na Figura 6 as Atividades. Todas as chaves-primárias são do tipo INTEGER (inteiro) para facilitar a consulta.

**Figura 4** - Código em SQL para Tabela Sócios

Criar tabela

```
1  CREATE TABLE socios (MAT INTEGER PRIMARY KEY,
2  nome TEXT, end_socio TEXT, tel_socio NUMERIC, status TEXT)
```

**Fonte:** Autoria Própria (2023)

**Figura 5** - Código em SQL para Tabela Eventos

Criar tabela

```
1  CREATE TABLE eventos (COD_EV INTEGER PRIMARY KEY,
2  evento TEXT, data TEXT, descricao_ev TEXT, tipo TEXT)
```

**Fonte:** Autoria Própria (2023)

**Figura 6** - Código em SQL para Tabela Atividades

Criar tabela

```
1  CREATE TABLE atividades (COD_AT INTEGER PRIMARY KEY,
2  atividade TEXT, hora TEXT, descricao_atvd TEXT)
```

**Fonte:** Autoria Própria (2023)

Após criar as tabelas do sistema, pôde-se adicionar informações em cada coluna, realizar consultas, realizar edições ou até mesmo deletar algum atributo ou atribuição.

### 4.2.2 Alteração de um tipo de dado de um atributo da tabela

Utilizando o comando ALTER TABLE foi possível realizar a atribuição de mais um tipo de dado na tabela de sócios, como ilustra a Figura 7. O atributo adicionado é referente ao e-mail, para que será solicitado ao realizar o cadastro de um sócio no clube, e foram definidos os seus tipos (TEXT, INTEGER, NUMERIC, dentre outros).

**Figura 7** - Código em SQL para Alteração de um Dado da Tabela

ALTERAÇÃO DE DADO

```
1   ALTER TABLE socios ADD email TEXT
```

**Fonte:** Autoria Própria (2023)

### 4.2.3 Inserção de dados nas tabelas

Por meio dos comandos INSERT INTO e VALUES é possível adicionar dados nas tabelas criadas, informando quais serão as informações adicionadas, como ilustra o código SQL da Figura 8 a seguir e resultado na Figura 9.

**Figura 8** - Código em SQL para Inserção de Dados na Tabela Sócios

Inserir dados

```
1   INSERT INTO socios VALUES (01,'Maria José Pereira', 'Avenida Brasil', 123456789, 'A', 'maria@gmail.com')
```

**Fonte:** Autoria Própria (2023)

**Figura 9** - Resultado da Inserção de Dados na Tabela Sócios

| | MAT | nome | end_socio | tel_socio | status | email |
|---|---|---|---|---|---|---|
| | Filtro | Filtro | Filtro | Filtro | Filtro | Filtro |
| 1 | 1 | Maria José Pereira | Avenida Brasil | 123456789 | A | maria@gmail.... |
| 2 | 2 | José Emanoel Silva | Avenida Brasil | 456789123 | B | jose@gmail.c... |

**Fonte:** Autoria Própria (2023)

Uma vez executada a inserção de novos dados, automaticamente eles são atribuídos nas suas respectivas tabelas, e quanto maior for o número de dados adicionados, maior será a estrutura do Banco de Dados.

### 4.2.4 Alteração de dados cadastrados

Utilizando os comandos UPDATE, SET e WHERE é possível realizar alterações nas tabelas criadas e nos dados inseridos. A Figura 10 ilustra a alteração de status B para status A do sócio cadastrado de matrícula 02. A Figura 11 mostra o resultado após o dado ser alterado e o resultado na Figura 14.

**Figura 10** - Código em SQL para Alteração de Dados na Tabela Sócios

Alteração de dados

```
1   UPDATE socios SET status = 'A' WHERE MAT = 02
```

**Fonte:** Autoria Própria (2023)

**Figura 11** - Resultado da Alteração de Dados na Tabela Sócios

| | MAT | nome | end_socio | tel_socio | status | email |
|---|---|---|---|---|---|---|
| | Filtro | Filtro | Filtro | Filtro | Filtro | Filtro |
| 1 | 1 | Maria José Pereira | Avenida Brasil | 123456789 | A | maria@gmail.... |
| 2 | 2 | José Emanoel Silva | Bairro Nassau | 456789123 | A | jose@gmail.c... |

**Fonte:** Autoria Própria (2023)

### 4.2.5 Consulta de um dado na tabela

Utilizando os comandos SELECT e FROM é possível consultar os dados de uma determinada tabela. Aqui, foi consultado os dados dos nomes dos eventos que estão armazenados na tabela eventos, o código utilizado está na Figura 12  resultado na Figura 13.

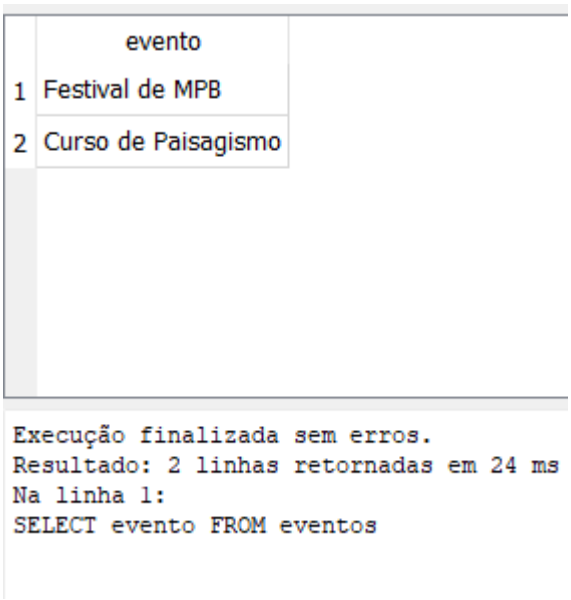
**Figura 12** - Código em SQL para Consulta de Dados na Tabela Eventos

Consulta de dados

```
1   SELECT evento FROM eventos
```

**Fonte:** Autoria Própria (2023)

**Figura 13** - Resultado da Consulta de Dados na Tabela Eventos



| | evento |
|---|---|
| 1 | Festival de MPB |
| 2 | Curso de Paisagismo |

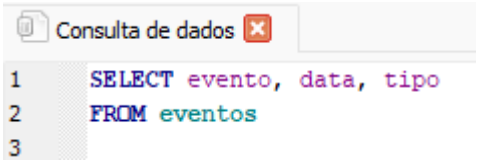
```
Execução finalizada sem erros.
Resultado: 2 linhas retornadas em 24 ms
Na linha 1:
SELECT evento FROM eventos
```

**Fonte:** Autoria Própria (2023)

### 4.2.6 Consulta envolvendo diversos dados

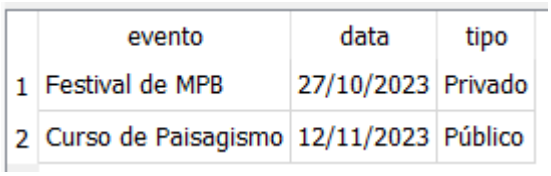
A Figura 14 mostra o código utilizado para consultar três tipos de dados de uma tabela, fazendo uso dos comandos SELET e FROM. A Figura 15 mostra o resultado desta consulta.

**Figura 14** - Código em SQL para Consulta de Vários Dados




Fonte: Autoria Própria (2023)

**Figura 15** - Resultado da Consulta de Dados com Duas Tabelas



| | evento | data | tipo |
|---|---|---|---|
| 1 | Festival de MPB | 27/10/2023 | Privado |
| 2 | Curso de Paisagismo | 12/11/2023 | Público |

**Fonte:** Autoria Própria (2023)

### 4.2.7 Apagando um registro de uma tabela

Para deletar um dado ou registro de uma tabela utiliza-se o comando DELETE, a Figura 16 mostra o código utilizado para deletar uma inscrição realizada na tabela de inscrição de eventos. Com o comando WHERE informa alguma especificação para deletar, neste caso, a

especificação escolhida foi a matrícula do sócio, então a inscrição no evento do sócio de matrícula '02' é automaticamente excluída utilizando este comando, e o resultado é mostrado na Figura 17.

**Figura 16** - Código em SQL para Deletar Dados de uma Tabela

Deletar um dado

```
1    DELETE FROM inscricaoeventos WHERE MAT = 02
```

**Fonte:** Autoria Própria (2023)

**Figura 17** - Resultado de Deletar um Dado da Tabela Inscrição de Eventos

Tabela: inscricaoeventos

| | ID_INS_EV | MAT | COD_EV |
|---|---|---|---|
| | Filtro | Filtro | Filtro |
| 1 | 202 | 1 | 12 |

**Fonte:** Autoria Própria (2023)

### 4.2.8 Desenvolvimento da interface para o sistema utilizando a integração da linguagem *Python* e SQL

Para montar a interface do sistema utilizou-se a integração do *DB Browser (SQLite)* e da linguagem de programação *Python* para gerar o código desta automação. Para isso, foi necessário realizar duas importações e conectar o banco de dados com o ambiente de programação, conforme a Figura 18.

**Figura 18** - Importações no *Python*

```
import sqlite3 as sq
from os import path
#C:\Users\ncayl\OneDrive\Área de Trabalho\BD
banco = sq.connect(path.join("C:/","Users","ncayl","OneDrive","Área de Trabalho","BD", "Projeto Clube.db"))
cursor = banco.cursor()
lista = []
```

**Fonte:** Autoria Própria (2023)

Para uma melhor visualização das funcionalidades do sistema, foi estruturado um menu por meio de um bloco WHILE TRUE, como ilustra a Figura 19, com sete opções para que o usuário informe a função que deseja realizar. O bloco inicia o sistema e só irá parar de funcionar uma vez que escolhida a opção correspondente ao break. A Figura 20 mostra a estruturação do menu uma vez que executado o código.

**Figura 19** - Código do Menu das Funcionalidades do Sistema

```
while True :
    print(" [ 1 ] Adicionar Sócio ","\n", "[ 2 ] Alterar Status", "\n","[ 3 ] Consultar Data",
    operacao = int(input("Escolha uma opção: "))
```

**Fonte:** Autoria Própria (2023)

**Figura 20** - Menu do Sistema

```
PS C:\Users\ncay1> & C:/Users/ncay1/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/ncay1/OneDrive/Área de Trabalho/BD/projeto_gi.py"
 [ 1 ] Adicionar Sócio
 [ 2 ] Alterar Status
 [ 3 ] Consultar Data
 [ 4 ] Sócios em Ordem Alfabética
 [ 5 ] Cadastrar Sócios em Eventos
 [ 6 ] Deletar Sócio
 [ 7 ] Sair
Escolha uma opção:
```

**Fonte:** Autoria Própria (2023)

A opção 1 do código da Figura 21 está relacionada com o cadastro de um sócio ao clube, onde o usuário deve informar inicialmente a matrícula que será a chave primária deste Banco de Dados. É importante mencionar que o número da matrícula deve ser único para cada cliente, caso contrário o programa não será executado corretamente pois irá ocorrer um erro. Além disso, demais dados como: nome, telefone, endereço e status são inseridos. A Figura 22 mostra o código estruturado para inserir um novo cliente ao clube.

**Figura 21** - Código no *Python* para Cadastro de um Sócio

```
if operacao == 1 :
    cursor.execute("""INSERT INTO socios (MAT, nome, tel_socio, end_socio, status) VALUES (0,"1",1,"1", 1)""")
    adict_mat = int(input('Digite o novo número da matrícula: '))
    cursor.execute("""UPDATE socios SET MAT = (?) WHERE MAT = 0 """, [adict_mat])
    banco.commit()
    adict_nome = str(input("Digite o nome do sócio: "))
    cursor.execute("""UPDATE socios SET nome = (?) WHERE nome = '1' """, [adict_nome])
    banco.commit()
    adict_tel_socio = int(input("Digite o número do sócio: "))
    cursor.execute("""UPDATE socios SET tel_socio = (?) WHERE tel_socio = 1 """, [adict_tel_socio])
    banco.commit()
    adict_end_socio = str(input("Digite o endereço do sócio: "))
    cursor.execute("""UPDATE socios SET end_socio = (?) WHERE end_socio = '1' """, [adict_end_socio])
    banco.commit()
    print(" [ 1 ] Classe A", "\n", "[ 2 ] Classe B", "\n", "[ 3 ] Classe C", "\n", "Informe o status do sócio: ")
    adict_status = int(input())
    cursor.execute("""UPDATE socios SET status = (?) WHERE status = 1 """, [adict_status])
    banco.commit()
    print("Operação concluída!")
```

**Fonte:** Autoria Própria (2023)

O sistema também permite alterações do status dos sócios já cadastrados, e esta função é referente à opção 2 na interface, e seu código conforme a Figura 22.

**Figura 22** - Código no Python para Alterar Status de um Sócio

```
if operacao == 2 :
    print( " [ 1 ] Classe A", "\n", "[ 2 ] Classe B", "\n", "[ 3 ] Classe C", "\n", "Informe o status do sócio: ")
    update_status = int(input())
    cursor.execute("""UPDATE socios SET status WHERE = (?) """, [update_status])
    banco.commit()
    print("Operação concluída!")
```

**Fonte:** Autoria Própria (2023)

O usuário também consegue visualizar quais são as datas dos eventos cadastrados pelo clube quando escolhida a opção 3, e seu código conforme a Figura 23.

**Figura 23** - Código no Python para Consultar Eventos do Clube

```
if operacao == 3 :
    cursor.execute(""" SELECT evento, data FROM eventos """)
    print(cursor.fetchall())
```

**Fonte:** Autoria Própria (2023)

O nome dos sócios é mostrado em ordem alfabética quando a opção 4 é escolhida, e seu código consta na Figura 24.

**Figura 24** - Código no Python para Consultar Nomes dos Sócios em Ordem Alfabética

```
if operacao == 4 :
    cursor.execute("""SELECT * FROM socios ORDER BY nome ASC""")
    print(cursor.fetchall())
```

**Fonte:** Autoria Própria (2023)

A opção 5 é referente à inscrição dos sócios em eventos. Vale ressaltar que apenas sócios Classe A podem se inscrever em eventos privados. Então, para um melhor funcionamento da operação, inicialmente o sistema pergunta qual o tipo de evento que o usuário deseja participar, inserindo 1 para Evento Público ou 2 para Evento Privado, bem como o código do Evento, que é a chave primária na tabela eventos e serve para identificá-lo. Posteriormente, o sócio insere a matrícula e com isso, se o sistema identificar que o cliente deseja se inscrever para um evento privado, mas não é Classe A, a operação não é concluída. Tais operações são ilustradas nas Figuras 25 e 26.

**Figura 25** - Código no Python para Inscrição em Eventos Públicos

```
if operacao == 5 :
    print("Qual o tipo de evento que você deseja de inscrever?")
    print( " [ 1 ] Público", "\n", "[ 2 ] Privado", "\n")
    tipo_evento = int(input())

    if tipo_evento == 1:
        cod_ev_cliente = int(input("Digite o código do evento que você deseja se inscrever: "))
        cursor.execute("""INSERT INTO inscricaoeventos (ID_INS_EV, MAT, COD_EV) VALUES (000, 000, 000) """)
        cursor.execute("""UPDATE inscricaoeventos SET COD_EV = (?) WHERE COD_EV = 000 """, [cod_ev_cliente])
        banco.commit()
        INS_MAT = int(input("Digite a matrícula do sócio: "))
        cursor.execute("""UPDATE inscricaoeventos SET MAT = (?) WHERE MAT = 000 """, [INS_MAT])
        banco.commit()
        cursor.execute("""SELECT MAX (COD_EV) FROM eventos """)
        COD_EV_MAX = cursor.fetchone()
        for line in COD_EV_MAX:
            COD_EV_MAX = line
            COD_EV_MAX += 1
        cursor.execute("""UPDATE eventos SET COD_EV = (?) WHERE COD_EV = 000 """, [COD_EV_MAX])
        banco.commit()
        print("Operação concluída!")
```

**Fonte:** Autoria Própria (2023)

**Figura 26** - Código no Python para Inscrição em Eventos Privados

```
if tipo_evento == 2:
    MAT_SOCIO = int(input('Digite a matrícula do sócio: '))
    cursor.execute("""SELECT status FROM socios WHERE status = (?) """, [MAT_SOCIO])
    NUM_MAT_SOCIO = cursor.fetchone()
    if MAT_SOCIO != 1:
        print("Apenas clientes Classe 'A' podem se inscrever neste tipo de evento!")
    else:
        cod_ev_cliente = int(input("Digite o código do evento que você deseja se inscrever: "))
        cursor.execute("""INSERT INTO inscricaoeventos (ID_INS_EV, MAT, COD_EV) VALUES (000, 000, 000) """)
        cursor.execute("""UPDATE inscricaoeventos SET COD_EV = (?) WHERE COD_EV = 000 """, [cod_ev_cliente])
        banco.commit()
        INS_MAT = int(input("Digite a matrícula do sócio: "))
        cursor.execute("""UPDATE inscricaoeventos SET MAT = (?) WHERE MAT = 000 """, [INS_MAT])
        banco.commit()
        cursor.execute("""SELECT MAX (COD_EV) FROM eventos """)
        COD_EV_MAX = cursor.fetchone()
        for line in COD_EV_MAX:
            COD_EV_MAX = line
            COD_EV_MAX += 1
        cursor.execute("""UPDATE eventos SET COD_EV = (?) WHERE COD_EV = 000 """, [COD_EV_MAX])
        banco.commit()
        print("Operação concluída!")
```

**Fonte:** Autoria Própria (2023)

Além destas funções, o sistema também permite deletar o cadastro dos sócios na opção 6, a qual pede apenas a informação da matrícula que será excluída, conforme o código da Figura 27. É importante informar o código da matrícula pois ela é a chave referência do sócio na tabela.

**Figura 27** - Código no Python para Inscrição em Eventos Privados

```
if operacao == 6 :
    del_socio = int(input('Informe a matrícula do sócio que será deletado: '))
    cursor.execute(""" DELETE FROM socios WHERE MAT = (?)""", [del_socio])
    banco.commit()
    print("Operação concluída!")
```

**Fonte:** Autoria Própria (2023)

Para encerrar o sistema, o usuário deve selecionar a opção 7- "fechar" da Figura 28.

**Figura 28** - Código no Python para Encerrar o Sistema

```
if operacao == 7 :
    banco.close
    break
```

**Fonte:** Autoria Própria (2023)

Ademais, é importante salientar que para evitar erros durante a execução do programa, bem como os dados sejam armazenados corretamente dentro do *DB Browser (SQLite),* é necessário as operações sejam realizadas com o Banco de Dados fechado, selecionando a opção correspondente, e caso contrário os dados inseridos e modificados não serão armazenados.

## 5. Considerações finais

O objetivo inicial de desenvolver um Sistema de Banco de Dados para um clube foi alcançado durante o desenvolvimento deste trabalho. As ferramentas utilizadas Diagrama de Entidade e Relacionamento, Diagrama de Use Case e Modelo Relacional, além do sistema Banco de Dados *DB BROWSER SQLITE* e linguagem de Programação *Python* no ambiente de desenvolvimento *VS Code*. possibilitarem uma análise completa do sistema da organização e as suas conexões, assim como as funções desenvolvidas com SQL - *Structured Query Language* para inserir dados no sistema de informação criado, consultar dados para cada tabela do sistema, alterar dados específicos, realizar inserções de dados, pesquisas e remover dados do banco de dados, quando necessário. Todas as funcionalidades se mostram eficazes para proporcionar maior agilidade e conforto para o gestor responsável pelo clube social. O desenvolvimento da interface do sistema na linguagem de programação *Python* permite uma apresentação com informações que possibilitam um maior controle e visibilidade quanto a execução do sistema e sua estruturação.

## REFERÊNCIAS

ALURA**. Introdução ao SQL Oracle: Manipule e Consulte Dados.** Disponível em: <https://www.alura.com.br/conteudo/introducao-sql-oracle-manipule-consulte-dados>. Acesso em: 05 out. 2023.

BORGES, L. E. **Python para Desenvolvedores**, 2010. Rio de Janeiro. ed.2ª. Edição do autor. Disponível em: <https://ark4n.files.wordpress.com/2010/01/python_para_desenvolvedores_2ed.pdf_>. Acesso em: 06 out 2023.

CARDOSO, GISELLE CRISTINA; CARDOSO, VIRGÍNIA M. **LINGUAGEM SQL**. Saraiva Educação SA, 2017.

CODD, E. F. ***A Relational Model of Data for Large Shared Data Banks***. Communications of the ACM, 13(6), 377-387, 1970.

DATE, Christopher J. **Introdução a Sistemas de Banco de Dados**. Editora Campus. 1ª Edição, 2014.

ELMASRI, Ramez e NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. Pearson Addison Wesley. 6ª Edição, 2011.

GIL, Antonio Carlos, **Como elaborar projetas de pesquisa** - 4. ed. - São Paulo: Atlas, 2002.

GONÇALVES, L. S. **Sistema de informações gerenciais**. IESDE Brasil S.A., Curitiba, 2006.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de informação gerenciais: Administrando a empresa digital.** 5ª ed. São Paulo: Pearson Prentice Hall, 2004.

LAKATOS, E.M; MARCONI, M.A**. Fundamentos de metodologia científica.** ed.5ª. São Paulo: Atlas 2003.

MENEZES, N. N. C. **Introdução à Programação com Python: Algoritmos e lógica de programação para iniciantes**, 2014. São Paulo. ed.2ª. Novatec Editora Ltda.

MUSLAH, E. & GHOUL, S. ***Requirements variability specification for data intensive software.*** *International Journal of Software Engineering & Applications*, 10(2). 2019.

O´BRIEN, J. A. **Sistemas de informação e as decisões gerenciais na era da Internet.** 2ª ed. São Paulo: Saraiva, 2004.

OLIVEIRA, Jair Figueiredo. **Sistema de Informação: um enfoque gerencial inserido no contexto empresarial e tecnológico**. Ed. – São Paulo: Érica, 2001.

RIBEIRO, C. D. **Um gerador de transações voltado para o modelo entidade-relacionamento.** 119f. Tese (Doutorado em Ciências em Engenharia de Sistemas e Computação) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, 1992.

SANTOS, P. H. **Diagrama ER é uma ferramenta voltada ao ensino de modelagem de dados.** 44p. Dissertação (Mestrado em Ciências da Computação) – Universidade Federal de Minas Gerais, Belo Horizonte, MG. 2017.

SANTANDER, V. F. A.; CASTRO, J. F. B. **Desenvolvendo *Use Cases* a Partir da Modelagem Organizacional.** III Workshop de Engenharia de Requisitos, Rio de Janeiro - RJ, Brasil. 2000. Disponível em: Acesso em: 05 out. 2023.

SCHNEIDER, G., WINTER, J. P. ***Applying Use Cases: a Practical Guide,*** Addison Wesley, (1998). Yu, Eric, Modelling Strategic Relationships for Process Reengineering, Phd Thesis, University of Toronto, 1998.

SILVA, E.L.; MENEZES, E.M. **A pesquisa e suas classificações. Metodologia da pesquisa e elaboração de dissertação**. 4. ed. rev. atual. Florianópolis: UFSC, 2005.