Universidade Católica Do Salvador
Análise e Desenvolvimento de Sistemas

Caroline de Jesus Santana

Karilana Muniz dos Santos

Lucas Santana e Silva

Milena Santos Matos

Yuri de Jesus Goes

Documentação para projeto de Biblioteca

Salvador, 2021

Sumário

1 ESTRUTURA

Nome do Projeto:

- *Biblioteca.*

Pacotes do projeto:

- *Model*;
- *Connection*;
- *Controller*;
- *Methods*;
- *View*.

Classes do projeto:

- *Model*: Livro, Autor, Categoria e Editora.
- *Connection*: Conexao;
- *Controller*: Validar , Buscar, Listar, Adicionar, Editar e Excluir;
- *Methods*: Metodos;
- *View*: Janela.

2 MODEL

Tabelas do banco de dados. Todos os atributos serão criados com a propriedade private.

2.1 LIVRO

Modelo abstrato da entidade Livro.

Atributos:

- int codigo;
- int ISBN;
- String nome;
- int categoria;
- int editora;

- int ano;

Métodos:

Criar *construtor* e *get* e *set* para todos os atributos.

2.2 AUTOR

Modelo abstrato da entidade Autor.

Atributos:

- int codigo;
- String nome;

Métodos:

Criar *construtor* e *get* e *set* para todos os atributos.

2.3 CATEGORIA

Modelo abstrato da entidade Categoria.

Atributos:

- int codigo;
- String descricao;

Métodos:

Criar *construtor* e *get* e *set* para todos os atributos.

2.4 EDITORA

Modelo abstrato da entidade Editora.

Atributos:

- int codigo;
- String nome;
- String cnpj;

Métodos:

Criar *construtor* e *get* e *set* para todos os atributos.

3 CONNECTION

Cuidará da conexão com o banco de dados.

3.1 CONEXAO

Essa classe tem o objetivo de iniciar conexão com o banco de dados. Para isso deve ser feito o download do Driver do banco de dados (https://jdbc.postgresql.org/download.html) e colocado na biblioteca do projeto.

Atributos:

- String url;
- String usuario;
- String senha;
- Connection conexao;

Métodos:

Construtor não recebe parâmetros.

```
    private String url;
    private String usuario;
    private String senha;
    private Connection conexao;

    public Conexao(){
        url="jdbc:postgresql://localhost:5432/BIBLIOTECA";
//BIBLIOTECA é o nome do banco de dados
        usuario ="postgres";
        senha="postgres";

        try {
            Class.forName("org.postgresql.Driver");
            conexao = DriverManager.getConnection(url,usuario,
senha);

            System.out.println("Conexão establecida!");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

4 CONTROLLER

Cuidará da manipulação das funções do banco de dados. Com exceção da classe *Validar*, todas as classes desse pacote receberam como parâmetro o objeto **Connection** gerado na classe *Janela* do pacote *View*.

4.1 VALIDAR

Nessa classe terão os métodos que receberão as entradas dos dados informados pelo usuário e checarão se esses dados estão aptos a serem adicionados ao banco de dados.

4.1.1 Validar Nome

Parâmetros:

- String nome;

Objetivo:

1. Verifica se o nome está vazio.
2. Retornar true ou false.

Exceptions:

Não há.

Adicionais:

Não há.

4.1.2 Validar Código

Parâmetros:

- int código;

Objetivo:

1. Verifica se o código é maior que 0.
2. Retornar true ou false.

Exceptions:

Não há.

Adicionais:

Não há.

4.1.3 Validar Ano

Parâmetros:

- int ano;

Objetivo:

1. Verificar se o ano informado é menor ou igual ao ano atual.
2. Retornar true ou false.

Exceptions:

Não há.

Adicionais:

Não há.

4.2 BUSCAR

Está classe irá buscar no banco de dados o(s) objeto(s) que corresponde ao parâmetro informado pelo usuário.

4.2.1 Buscar Livro

Parâmetros:

- int código / int ISBN / int editora / int categoria / int ano / String nome;

Objetivo:

1. Pesquisar na tabela LIVRO_AUTOR o valor de atributo digitado.
2. Retornar um array com o(s) resultado(s) obtido(s).

Exceptions:

1. Nenhum resultado encontrado.
   a. **Prevenção:** Criar em *Listagem* condicional que verifica o código do objeto ou os itens do array, caso seja nulo, informar que a pesquisa não retornou resultados.

Adicionais:

1. Criar um método para pesquisar cada atributo do livro.

4.2.2 Buscar Autor

Parâmetros:

- int código / String nome;

Objetivo:

1. Pesquisar na tabela AUTOR o código ou nome digitado.

2. Retornar um array ou objeto com o(s) resultado(s) obtido(s).

Exceptions:

1. Nenhum resultado encontrado.
    a. **Prevenção:** Criar em *Listagem* condicional que verifica o código do objeto ou os itens do array, caso seja nulo, informar que a pesquisa não retornou resultados.

Adicionais:

1. Deve ser usado o retorno em objeto para quando a pesquisa for por código e em array quando a pesquisa for pelo nome.

4.2.3 Buscar Categoria

Parâmetros:

- int código / String nome;

Objetivo:

1. Pesquisar na tabela CATEGORIA o código ou nome digitado.
2. Retornar um array ou objeto com o(s) resultado(s) obtido(s).

Exceptions:

1. Nenhum resultado encontrado.
    a. **Prevenção:** Criar em *Listagem* condicional que verifica o código do objeto ou os itens do array, caso seja nulo, informar que a pesquisa não retornou resultados.

Adicionais:

1. Deve ser usado o retorno em objeto para quando a pesquisa for por código e em array quando a pesquisa for pelo nome.

4.2.4 Buscar Editora

Parâmetros:

- int código / String nome / String CNPJ;

Objetivo:

1. Pesquisar na tabela LIVRO o código ou nome digitado.
2. Retornar um array ou objeto com o(s) resultado(s) obtido(s).

Exceptions:

1. Nenhum resultado encontrado.
    a. **Prevenção:** Criar em *Listagem* condicional que verifica o código do objeto ou os itens do array, caso seja nulo, informar que a pesquisa não retornou resultados.

Adicionais:

1. Deve ser usado o retorno em objeto para quando a pesquisa for por código ou CNPJ e em array quando a pesquisa for pelo nome.

4.3 LISTAR

4.3.1 Listar Livro

Parâmetros:

- Livro livro / ArrayList<Livro> livro / não recebe parâmetro.

Objetivo:

1. Imprimir os dados de um ou mais livros existentes no banco de dados.

Exceptions:

1. Array ou objetos nulos.
    a. **Prevenção:** Antes de fazer a impressão verificar se o objeto ou o array são nulos. Não imprimir. Sair do bloco de instrução.

Adicionais:

1. Quando o método não receber parâmetro isso indica que deve ser impresso a lista de todos os dados da tabela LIVRO_AUTOR, com o código, categoria, ano e ISBN do livro.

4.3.2 Listar Autor

Parâmetros:

- Autor autor / ArrayList<Autor> autor / não recebe parâmetro.

Objetivo:

1. Imprimir os dados de um ou mais autores existentes no banco de dados.

Exceptions:

1. Array ou objetos nulos.

a. **Prevenção:** Antes de fazer a impressão verificar se o objeto ou o array são nulos. Não imprimir. Sair do bloco de instrução.

Adicionais:

1. Quando o método não receber parâmetro isso indica que deve ser impresso a lista de todos os dados da tabela AUTOR.

4.3.3 Listar Categoria

Parâmetros:

- Categoria categoria / ArrayList<Categoria> categoria / não recebe parâmetro.

Objetivo:

1. Imprimir os dados de uma ou mais categorias existentes no banco de dados.

Exceptions:

1. Array ou objetos nulos.
   a. **Prevenção:** Antes de fazer a impressão verificar se o objeto ou o array são nulos. Não imprimir. Sair do bloco de instrução.

Adicionais:

1. Quando o método não receber parâmetro isso indica que deve ser impresso a lista de todos os dados da tabela CATEGORIA.

4.3.4 Listar Editora

Parâmetros:

- Editora editora / ArrayList<Editora> editora / não recebe parâmetro.

Objetivo:

1. Imprimir os dados de uma ou mais editoras existentes no banco de dados.

Exceptions:

1. Array ou objetos nulos.

a. **Prevenção:** Antes de fazer a impressão verificar se o objeto ou o array são nulos. Não imprimir. Sair do bloco de instrução.

Adicionais:

1. Quando o método não receber parâmetro isso indica que deve ser impresso a lista de todos os dados da tabela EDITORA.

4.4 ADICIONAR

4.4.1 Adicionar Autor

Parâmetros:

- Não recebe.

Objetivo:

1. Capturar código do autor.
2. Capturar nome do autor.
3. Inserir no banco de dados.

Exceptions:

1. Código informado já existe no banco de dados.
   a. **Prevenção:** Utilizar o *Buscador* para verificar se o código informado já existe. Imprimir mensagem informando que o código já existe. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.
2. Código informado inválido.
   a. **Prevenção:** Utilizar o *Validador* para verificar se o código informado é maior que 0. Imprimir mensagem informando que o código é inválido. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.
3. Nome vazio.
   a. **Prevenção**: Utilizar o *Validador* para verificar se o nome está vazio. Imprimir mensagem informando que o nome não pode ficar vazio. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.

Adicionais:

1. Antes de passar para próxima etapa, verificar se a etapa atual é válida.

4.4.2 Adicionar Categoria

Parâmetros:

- Não recebe.

Objetivo:

1. Capturar código da categoria.
2. Capturar nome da categoria.
3. Inserir no banco de dados.

Exceptions:

1. Código informado já existe no banco de dados.
    a. **Prevenção:** Utilizar o *Buscador* para verificar se o código informado já existe. Imprimir mensagem informando que o código já existe. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.
2. Código informado inválido.
    a. **Prevenção:** Utilizar o *Validador* para verificar se o código informado é maior que 0. Imprimir mensagem informando que o código é inválido. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.
3. Nome vazio.
    a. **Prevenção**: Utilizar o *Validador* para verificar se o nome está vazio. Imprimir mensagem informando que o nome não pode ficar vazio. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.

Adicionais:

1. Antes de passar para próxima etapa, verificar se a etapa atual é válida.

4.4.3 Adicionar Editora

Parâmetros:

- Não recebe.

Objetivo:

1. Capturar código da editora.
2. Capturar nome da editora.
3. Capturar CNPJ da editora.
4. Inserir no banco de dados.

Exceptions:

1. Código informado já existe no banco de dados.
    a. **Prevenção:** Utilizar o *Buscador* para verificar se o código informado já existe. Imprimir mensagem informando que o código já existe. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.
2. Código informado inválido
    a. **Prevenção:** Utilizar o *Validador* para verificar se o código informado é maior que 0. Imprimir mensagem informando que o código é inválido. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.

3. Nome vazio.
    a. **Prevenção**: Utilizar o *Validador* para verificar se o nome está vazio. Imprimir mensagem informando que o nome não pode ficar vazio. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.
4. CNPJ repetido.
    a. **Prevenção:** Utilizar o *Buscador* para verificar se o CNPJ digitado já existe no banco de dados. Imprimir mensagem de que já existe uma editora com esse CNPJ. Perguntar ao usuário se ele deseja inserir um novo dado ou sair.

Adicionais:

1. Antes de passar para próxima etapa, verificar se a etapa atual é válida.
2. O CNPJ pode ficar em branco.
3. Para a **Exception 4**, só deverá ser feita validação se o CNPJ não estiver vazio.

4.4.4 Adicionar Livro

Parâmetros:

- Não recebe.

Objetivo:

1. Capturar o código do livro.
2. Capturar o ISBN do livro.
3. Capturar o nome do livro.
4. Capturar o ano do livro.
5. Capturar o código da editora.
6. Capturar o código da categoria.
7. Capturar a quantidade de autores do livro.
8. Capturar o código do(s) autor(es) do livro.
9. Adicionar informações no banco de dados.

Exceptions:

1. Código informado já existe no banco de dados.
    a. **Prevenção:** Utilizar o *Buscador* para verificar se o código informado já existe. Imprimir mensagem informando que o código já existe. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.
2. Código informado inválido.
    a. **Prevenção:** Utilizar o *Validador* para verificar se o código informado é maior que 0. Imprimir mensagem informando que o código é inválido. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.
3. Nome vazio.
    a. **Prevenção**: Utilizar o *Validador* para verificar se o nome está vazio. Imprimir mensagem informando que o nome não pode ficar vazio. Perguntar ao usuário se ele deseja inserir novamente o dado ou sair.
4. Ano informado é maior que o ano atual.
    a. **Prevenção:** Utilizar o *Validador* para verificar se o ano informado é válido.
5. Código da editora não cadastrado.

a. **Prevenção**: Utilizar o *Buscador* para verificar se o código informado existe. Imprimir mensagem informando que a editora não foi cadastrada. Oferecer a opção para o usuário de cadastrar editora, visualizar lista de editoras ou sair.

6. Código da categoria não cadastrado.

a. **Prevenção**: Utilizar o *Buscador* para verificar se o código informado existe. Imprimir mensagem informando que a categoria não foi cadastrada. Oferecer a opção para o usuário de cadastrar categoria, visualizar lista de categorias ou sair.

7. Código do autor não cadastrado.

a. **Prevenção**: Utilizar o *Buscador* para verificar se o código informado existe. Imprimir mensagem informando que o autor não foi cadastrado. Oferecer a opção para o usuário de cadastrar autor, visualizar lista de autores ou sair.

8. Livro com mais de um autor.

a. **Prevenção**: Perguntar ao usuário quantos autores o livro tem e capturar o autor a quantidade de vezes digitada.

9. Duplicação de ISBN.

a. **Prevenção:** Utilizar o *Buscador* para verificar se o ISBN informado existe. Informar que o ISBN já existe. Perguntar se o usuário deseja inserir a informação novamente ou sair.

Adicionais:

1. Antes de passar para próxima etapa, verificar se a etapa atual é válida.

2. O código do livro e do autor devem ser adicionados na tabela LIVRO_AUTOR. Se o livro possuir mais de um autor, os dados devem ser adicionados de forma que o código do livro seja estático e o código do autor percorra um array temporário criado para armazenar o código dos autores.

3. Não deve ser adicionado o código do autor na tabela LIVRO.

4.5 EDITAR

Essa classe cuidará de realizar modificações nos dados cadastrados.

4.5.1 Editar Livro

Parâmetros:

- int codigo.

Objetivos:

1. Recber código do livro.
2. Perguntar qual atributo deseja editar.
3. Editar objeto.

Exceptions:

1. Código já pertence a outro objeto.
    a. **Prevenção:** Utilizar o *Buscador* para conferir se o código foi cadastrado no banco de dados. Informar que código já existe. Perguntar ao usuário se ele deseja inserir novo código ou sair.
2. Exclusão de autor em livro de autor único.
    a. **Prevenção:** Utilizar o *Buscador* para verificar se o livro possui mais de um autor. Imprimir mensagem de impossibilidade de exclusão de autor em livro de autor único. Perguntar ao usuário se ele deseja fazer outra alteração ou sair.
3. Duplicação de ISBN.
    a. **Prevenção:** Utilizar o *Buscador* para verificar se o ISBN já existe. Informar que o ISBN já está sendo usado. Perguntar ao usuário se ele deseja fazer outra alteração ou sair.
4. Código de Editora / Categoria / Autor não existem.
    a. **Prevenção:** Utilizar o *Buscador* para verificar se o código do atributo escolhido existe no banco de dados. Informar que o código não existe. Perguntar ao usuário se ele deseja inserir um novo código ou sair.

Adicionais:

1. O usuário pode alterar os seguintes atributos:
    a. Código;
    b. Nome;
    c. ISBN;
    d. Autor;
    e. Categoria;

f. Editora;

g. Ano.

2. Para a edição do autor deve ser exibida a opção de “Excluir” ou “Adicionar”. Sendo que a exclusão só poderá estar disponível para livros com mais de um autor.

4.**5.2 Editar Autor**

Parâmetros:

- int codigo.

Objetivos:

1. Receber código do autor.
2. Perguntar qual atributo deseja editar.
3. Editar objeto.
4. Perguntar ao usuário se as mudanças foram concluídas.

Exceptions:

2. Código já pertence a outro objeto.

a. **Prevenção:** Utilizar o *Buscador* para conferir se o código foi cadastrado no banco de dados. Informar que código já existe. Perguntar ao usuário se ele deseja inserir novo código ou sair.

Adicionais:

1. O usuário pode editar os seguintes atributos:

a. Código;

b. Nome.

4.5.3 Editar Categoria

Parâmetros:

- int codigo.

Objetivos:

1. Receber código da categoria.
2. Perguntar qual atributo deseja editar.
3. Editar objeto.

Exceptions:

1. Código já pertence a outro objeto.
    a. **Prevenção:** Utilizar o *Buscador* para conferir se o código foi cadastrado no banco de dados. Informar que código já existe. Perguntar ao usuário se ele deseja inserir novo código ou sair.

Adicionais:

1. O usuário pode editar os seguintes atributos:
    a. Código;
    b. Descrição.

4.5.4 Editar Editora

Parâmetros:

- int codigo.

Objetivos:

1. Receber código da editora.
2. Perguntar qual atributo deseja editar.
3. Editar objeto.

Exceptions:

1. Código já pertence a outro objeto.
    a. **Prevenção:** Utilizar o *Buscador* para conferir se o código foi cadastrado no banco de dados. Informar que código já existe. Perguntar ao usuário se ele deseja inserir novo código ou sair.
2. CNPJ já existe.
    a. **Prevenção:** Utilizar o *Buscador* para verificar se o CNPJ digitado já existe no banco de dados. Imprimir mensagem de que já existe uma editora com esse CNPJ. Não fazer alteração. Sair do bloco de instrução.

Adicionais:

1. O usuário pode editar os seguintes atributos:
    a. Código;
    b. Nome;
    c. CNPJ.

4.6 EXCLUIR

As funções de exclusão utilizarão a mesma lógica para cada método, mudando apenas o local de busca no banco de dados.

Parâmetros:

- int codigo.

Objetivos:

1. Receber código.
2. Tentar excluir o objeto.

Exceptions:

1. Código não existe.
    a. **Prevenção:** Utilizar o *Buscador* para conferir se o código foi cadastrado no banco de dados. Informar que código não existe. Perguntar ao usuário se ele deseja inserir novo código ou sair.
2. Não permitida a exclusão por vínculo (Categoria ou Editora).
    a. **Prevenção:** Antes de tentar excluir o objeto, pesquisar na tabela LIVRO quais registros estão associados ao código informado e exibir a lista de livros vinculados a Categoria ou Editora.
3. Não permitida a exclusão por vínculo (Autor).
    a. **Prevenção:** Antes de tentar excluir o objeto, pesquisar na tabela LIVRO_AUTOR quais registros estão associados ao código informado e exibir a lista de livros vinculados ao Autor.

Adicionais:

Não há.

5 METHODS

Cuidará da chamada dos métodos do pacote “Controller”.

5.1 METODOS

Essa classe fará a ligação entre a classe Janela e as classes do pacote Controller.

5.1.1 Adicionar

Parâmetros:

- Não recebe.

Objetivo:

1. Perguntar ao usuário qual objeto ele deseja adicionar.
2. Chamar o método de adicionar.
3. Perguntar ao usuário se ele deseja continuar adicionando outros objetos.

Exceptions:

1. Informar opção inexistente.
   a. **Prevenção:** Verificar se a opção digitada corresponde a uma das opções disponíveis. Informar que a opção digitada não é válida. Repetir o bloco de instrução.

Adicionais:

Não há.

5.1.2 Editar

Parâmetros:

- Não recebe.

Objetivo:

1. Perguntar ao usuário qual objeto ele deseja editar.
2. Chamar o método de editar.
3. Perguntar ao usuário se ele deseja continuar editando outros objetos.

Exceptions:

1. Informar opção inexistente.
   a. **Prevenção:** Verificar se a opção digitada corresponde a uma das opções disponíveis. Informar que a opção digitada não é válida. Repetir o bloco de instrução.

Adicionais:

Não há.

5.1.3 Excluir

Parâmetros:

- Não recebe.

Objetivo:

1. Perguntar ao usuário qual objeto ele deseja excluir.
2. Chamar o método de excluir.
3. Perguntar ao usuário se ele deseja continuar excluindo outros objetos.

Exceptions:

1. Informar opção inexistente.
    a. **Prevenção:** Verificar se a opção digitada corresponde a uma das opções disponíveis. Informar que a opção digitada não é válida. Repetir o bloco de instrução.

Adicionais:

Não há.

5.1.4 Buscar

Parâmetros:

- Não recebe.

Objetivo:

1. Perguntar ao usuário qual objeto ele deseja buscar.
2. Perguntar ao usuário por qual atributo ele deseja buscar.
3. Chamar o método de buscar correspondente.
4. Perguntar ao usuário se ele deseja continuar buscando por outros objetos.

Exceptions:

1. Informar opção inexistente.
    a. **Prevenção:** Verificar se a opção digitada corresponde a uma das opções disponíveis. Informar que a opção digitada não é válida. Repetir o bloco de instrução.

Adicionais:

1. Utilizar o *Listar* para imprimir os resultados retornados.

5.1.5 Listar

Parâmetros:

- Não recebe.

Objetivo:

1. Perguntar ao usuário qual objeto ele deseja listar.
2. Chamar o método de listar.
3. Perguntar ao usuário se ele deseja continuar listando outros objetos.

Exceptions:

1. Informar opção inexistente.
   a. **Prevenção:** Verificar se a opção digitada corresponde a uma das opções disponíveis. Informar que a opção digitada não é válida. Repetir o bloco de instrução.

Adicionais:

Não há.

6 VIEW

Cuidará da interface do usuário.

6.1 JANELA

Tem a função de:

- Cuidar da interface do usuário.
- Chamar os métodos do pacote “Methods”.
- Iniciar conexão com o banco de dados e transmiti-la para outras classes.
- Finalizar conexão com o banco de dados.

É também onde ficará o método main. Seu objetivo é perguntar ao usuário que ação ele quer fazer, podendo escolher entre:

1. Adicionar;
2. Editar;
3. Excluir;

4. Buscar;
5. Listar;
6. Sair.

A opção desejada irá chamar o método da classe “Methods” correspondente. Caso o usuário digite uma opção diferente das disponíveis deverá ser impressa uma mensagem de parâmetro inválido.

7 CLASSES EXTRAS

7.1 CLASSE “MENSAGEM”

Se viu a utilidade da criação de uma classe para centralização de mensagens repetidas dentro do pacote *Controller*. Todos os métodos da classe são void.

Classe que utilizam:

- Adicionar.
- Listar.
- Editar.
- Excluir.
- Janela.

7.2 CLASSE “ADICIONAR RANDOM”

Essa classe foi criada com o intuito de gerar dados aleatórios para o banco de dados mediante validação. Para isso foram feitas alterações na classe *Validar,* criando um método que verifica se a quantidade máxima de objetos já foi adicionada ao banco. Todos os objetos seguem a mesma lógica de criação, sendo que o livro só poderá ser gerado se houver no banco pelo menos um objeto Editora, um objeto Categoria e um objeto do tipo Autor.

7.2.1 Geração de código

A geração de código retorna um número inteiro positivo entre 1000 e 9999. Após a geração desse valor aleatório é verificado sua existência o banco de dados. Caso ele já exista o sistema irá gerar um novo código. Do contrário, ele passa para a próxima etapa.

7.2.2 Geração de nome

A geração de nome recebe por padrão o nome do objeto + Random concatenado com o código de objeto gerado. Por exemplo, se for feita a tentativa de cadastro de uma Editora cujo o código é 2310, seu nome seria “EditoraRandom 2310”.

7.2.3 Geração de CNPJ para Editora

A geração de CNPJ concatena vários números gerados aleatoriamente divididos nas seguintes partes:

1. Gerar um número inteiro positivo entre 10 e 19.
2. Gerar um número inteiro positivo entre 100 e 999.
3. Gerar um número inteiro positivo entre 100 e 999.
4. Gerar um número inteiro positivo entre 1 e 9.
5. Gerar um número inteiro positivo entre 10 e 99.

Após a geração desses números é verificado se possui algum cadastro no banco de dados com o mesmo CNPJ. Em caso positivo, é gerado outro valor de CNPJ. Do contrário, o programa passa para a próxima etapa.

7.2.4 Geração de ISBN para Livro

A geração de ISBN retorna um número inteiro positivo entre 1000000 e 9999999. Após a geração desse valor aleatório é verificado sua existência o banco de dados. Caso ele já exista o sistema irá gerar um novo ISBN. Do contrário, ele passa para a próxima etapa.

7.2.5 Geração de Editora para Livro

Primeiro é feita a verificação para conferir se existem Editoras cadastradas no banco de dados. Caso não exista, o sistema retornará um erro ao usuário informando que existem pré-requisitos para geração de Livro. Do contrário o sistema captura o código de todas as editoras cadastradas no banco de dados e armazenará em um array. Após isso é sorteado um número de 0 à *n*, onde *n* é o tamanho do array. Este número servirá para pegar uma posição aleatória no array e atribuir o código da Editora para o Livro.

7.2.6 Geração de Categoria para Livro

Primeiro é feita a verificação para conferir se existem Categorias cadastradas no banco de dados. Caso não exista, o sistema retornará um erro

ao usuário informando que existem pré-requisitos para geração de Livro. Do contrário o sistema captura o código de todas as Categorias cadastradas no banco de dados e armazenará em um array. Após isso é sorteado um número de 0 à *n*, onde *n* é o tamanho do array. Este número servirá para pegar uma posição aleatória no array e atribuir o código da Categoria para o Livro.

7.2.7 Geração de Autor para Livro

Primeiro é feita a verificação para conferir se existem Autores cadastrados no banco de dados. Caso não exista, o sistema retornará um erro ao usuário informando que existem pré-requisitos para geração de Livro. Do contrário o sistema captura o código de todos os Autores cadastrados no banco de dados e armazenará em um array. Em seguida é gerado um número aleatório entre 1 e *x*, onde *x* é o valor máximo de autores que podem fazer parte de um mesmo livro (valor deve ser previamente estipulado). Caso o número sorteado seja maior do que a quantidade de objetos no array, um novo número deverá ser gerado . Do contrário, é sorteado um número de 0 à *n*, onde *n* é o tamanho do array. Este número servirá para pegar uma posição aleatória no array e atribuir o código do Autor para o Livro. O sistema deverá repetir essa operação para gerar uma quantidade *x* de códigos de Autor para o Livro.

8 LÓGICA DO BANCO

8.1 QUANTO CADASTRO DE OBJETO

Utilizado o comando INSERT para adicionar os dados ao banco.

8.1.1 Livro

A inserção de dados é feita nas tabelas LIVRO e LIVRO_AUTOR, sendo a tabela LIVRO_AUTOR para armazenar a relação entre livros e autores. Os valores são capturados dinamicamente e validados, qualquer inconformidade detectada não faz a inserção. Quando o livro tiver apenas um autor será armazenado uma única vez na tabela LIVRO_AUTOR essa relação. Para livros com n autores será armazenado n vezes na tabela LIVRO_AUTOR.

8.1.2 Autor

A inserção de dados é feita apenas na tabela AUTOR. Os valores são capturados dinamicamente e validados, qualquer inconformidade detectada não faz a inserção.

8.1.3 Categoria

A inserção de dados é feita apenas na tabela CATEGORIA. Os valores são capturados dinamicamente e validados, qualquer inconformidade detectada não faz a inserção.

8.1.4 Editora

A inserção de dados é feita apenas na tabela EDITORA. Os valores são capturados dinamicamente e validados, qualquer inconformidade detectada não faz a inserção.

8.2 QUANTO PESQUISA DE OBJETO

É utilizado o comando SELECT + WHERE para fazer o filtro de objeto mediante parâmetro informado. Para o nome é utilizado o comando LIKE para retornar os resultados que contenham a expressão informada pelo usuário. As informações retornadas são armazenadas nos objetos modelos que constam no pacote “Model”.

8.2.1 Livro

A pesquisa é realizada na tabela LIVRO_AUTOR. É também utilizado o comando INNER JOIN para capturar as informações armazenadas na tabela LIVRO (ISBN, nome do livro, ano de publicação, código da editora e código da categoria), na tabela EDITORA (nome da editora), na tabela CATEGORIA (nome da categoria) e na tabela AUTOR (nome do autor). A filtragem de dados é feita por um dos seguintes atributos:

- cod_livro
- isbn
- nome_livro
- ano
- cod_editora
- cod_categoria
- cod_autor

8.2.2 Autor

A pesquisa é realizada na tabela AUTOR. A filtragem de dados é feita por um dos seguintes atributos:

- cod_autor
- nome_autor

8.2.3 Categoria

A pesquisa é realizada na tabela CATEGORIA. A filtragem de dados é feita por um dos seguintes atributos:

- cod_categoria
- descrição_categoria

8.2.4 Editora

A pesquisa é realizada na tabela EDITORA. A filtragem de dados é feita por um dos seguintes atributos:

- cod_editora
- cnpj
- nome_editora

8.3 QUANTO ALTERAÇÃO DE OBJETO

É utilizado o comando UPDATE para alterar o atributo + WHERE para fazer um filtro dos objeto tendo como parâmetro o código do objeto selecionado. As alterações estão sujeitas a validação para que não haja dois valores iguais de código, ISBN (para livro) e CNPJ (para editora); e certificação que os novos códigos de editora, categoria e autor informados para o livro foram previamente cadastrados no banco de dados.

8.3.1 Livro

As alterações permitidas para a opção de Autor dentro de um Livro são apenas de adicionar ou excluir vínculo, sendo que a exclusão só está permitida para livros com mais de um autor.

Para a parte de “Adicionar autor”, após a validação do código do autor digitado, será realizado inserção na tabela LIVRO_AUTOR desse novo vínculo.

Para a parte de “Exclusão de Autor”, primeiro será feita uma busca na tabela LIVRO_AUTOR utilizando SELECT + WHERE (onde a condicional deve ser o código do livro) para verificar se o livro possui mais de um autor. Como a chamada do método *Buscar* é um array, basta apenas verificar seu tamanho para prosseguir ou não com a exclusão.

Sendo a condição anterior true, deverá ser feita uma nova consulta na tabela LIVRO_AUTOR utilizando SELECT + WHERE (onde a condicional deve ser o código do livro + código do autor) para verificar se existe alguma relação entre os códigos de Autor e Livro informados.

Sendo a condição anterior true deve ser utilizado o DELETE + WHERE, onde a condicional deve ser o código do livro + código do autor.

8.3.2 Autor

Para a alteração do código do autor será feita uma pesquisa na tabela LIVRO_AUTOR utilizando SELECT + WHERE (onde a condicional é o código do autor) afim de verificar se o autor está vinculado a algum livro. Em caso negativo, o código do autor é alterado.

Em caso positivo, será criado um array temporário para armazenar todos livros que possuem relação com o código do autor, utilizando a pesquisa *buscarLivroAutor* da classe *Busca* que retorna um array.

Depois serão excluídos da tabela LIVRO_AUTOR todos os registros vinculados ao código antigo do autor, para isso será utilizado DELETE + WHERE (onde a condicional é o código antigo do autor).

Será utilizado o UPDATE para atribuir valor nulo para o atributo cod_autor de todos os livros vinculados ao código anterior do autor + WHERE (onde a condicional é o código antigo do autor).

Após isso será atualizado o código do autor na tabela AUTOR utilizando UPDATE + WHERE (onde a condicional é o código antigo do autor).

Ao fim, será adicionado a tabela LIVRO_AUTOR os códigos dos livro com o código do autor atualizado.

8.3.3 Categoria

Para a alteração do código da categoria será feita uma pesquisa na tabela LIVRO utilizando SELECT + WHERE (onde a condicional é o código da

categoria) afim de verificar se a categoria está vinculada a algum livro. Em caso negativo, o código da categoria é alterado.

Em caso positivo, será criado um array temporário para armazenar todos livros que possuem relação com o código da categoria, utilizando a pesquisa *buscarLivroCategoria* da classe *Busca* que retorna um array.

Será utilizado o UPDATE para atribuir valor nulo para o atributo cod_categoria de todos os livros vinculados ao código anterior da categoria + WHERE (onde a condicional é o código antigo da categoria).

Após isso será atualizado o código da categoria na tabela CATEGORIA utilizando UPDATE + WHERE (onde a condicional é o código antigo da categoria).

Ao fim, será adicionado a tabela LIVRO o código da categoria atualizado para os livros inseridos no array utilizando UPDATE + WHERE (onde a condicional é o código do livro).

8.3.4 Editora

Para a alteração do código da editora será feita uma pesquisa na tabela LIVRO utilizando SELECT + WHERE (onde a condicional é o código da editora) afim de verificar se a editora está vinculada a algum livro. Em caso negativo, o código da editora é alterado.

Em caso positivo, será criado um array temporário para armazenar todos livros que possuem relação com o código da editora, utilizando a pesquisa *buscarLivroEditora* da classe *Busca* que retorna um array.

Será utilizado o UPDATE para atribuir valor nulo para o atributo cod_editora de todos os livros vinculados ao código anterior da editora + WHERE (onde a condicional é o código antigo da editora).

Após isso será atualizado o código da editora na tabela EDITORA utilizando UPDATE + WHERE (onde a condicional é o código antigo da editora).

Ao fim, será adicionado a tabela LIVRO o código da editora atualizado para os livros inseridos no array utilizando UPDATE + WHERE (onde a condicional é o código do livro).

8.4 QUANTO EXCLUSÃO DE OBJETO

Será usado o comando DELETE + WHERE (onde a condicional é o código do objeto). Para alguns objetos a exclusão só é permitida se não houver vínculo com a tabela LIVRO.

8.4.1 Livro

Será feita a busca para verificar se o código informado existe no banco de dados. Caso exista a exclusão é realizada.

8.4.2 Autor

Será feita a busca para verificar se o código informado existe no banco de dados. Caso exista, é feita outra busca para verificar se o código está vinculado a algum objeto da tabela LIVRO.

Em caso positivo, então a exclusão não é realizada e é exibida a lista de livros que o objeto está vinculada. Em caso negativo, a exclusão é realizada.

8.4.3 Categoria

Será feita a busca para verificar se o código informado existe no banco de dados. Caso exista, é feita outra busca para verificar se o código está vinculado a algum objeto da tabela LIVRO.

Em caso positivo, então a exclusão não é realizada e é exibida a lista de livros que o objeto está vinculada. Em caso negativo, a exclusão é realizada.

8.4.4 Editora

Será feita a busca para verificar se o código informado existe no banco de dados. Caso exista, é feita outra busca para verificar se o código está vinculado a algum objeto da tabela LIVRO.

Em caso positivo, então a exclusão não é realizada e é exibida a lista de livros que o objeto está vinculada. Em caso negativo, a exclusão é realizada.

8.5 QUANTO LISTAGEM DE OBJETO

A pesquisa é feita usando o SELECT + WHERE, onde a condicional é o valor do atributo da tabela informado. Se o método não receber parâmetros deve ser impressa todos os objetos da tabela correspondente, nesse caso o comando WHERE é descartado.

8.5.1 Livro

A pesquisa é feita na tabela LIVRO_AUTOR. É também utilizado o comando INNER JOIN para capturar as informações armazenadas na tabela LIVRO (ISBN, nome do livro, ano de publicação, código da editora e código da categoria), na tabela EDITORA (nome da editora), na tabela CATEGORIA (nome da categoria) e na tabela AUTOR (nome do autor).

8.5.2 Autor

A pesquisa é realizada na tabela AUTOR.

8.5.3 Categoria

A pesquisa é realizada na tabela CATEGORIA.

8.5.4 Editora

A pesquisa é realizada na tabela EDITORA.

8.6 QUANTO GERAÇÃO DE OBJETO

Antes de tudo é feita a validação do objeto que se quer adicionar utilizando SELECT COUNT, onde se é pesquisado o código do objeto e retornado a quantidade de tuplas existentes. Se esse valor for maior ou igual ao valor máximo pré-estabelecido é retornando false e a inserção não acontece.