FACULDADE DE TECNOLOGIA DE FRANCA

LENILSON AVELINO DOS SANTOS
RENAN CHRISTIAN DA SILVA CORDEIRO
VICTOR HUGO AMARAL PIMENTEL
VITOR SATURI

Sistema de organização da Sala de Leitura da Escola Estadual Agostinho Lima de Vilhena

UCE – ENGENHARIA DE SOFTWARE II

FRANCA
2025

LENILSON AVELINO DOS SANTOS
RENAN CHRISTIAN DA SILVA CORDEIRO
VICTOR HUGO AMARAL PIMENTEL
VITOR SATURI

Sistema de organização da Sala de Leitura da Escola Estadual Agostinho Lima de Vilhena

UCE – ENGENHARIA DE SOFTWARE II

Relatório Final de Unidade Curricular de Extensão – UCE de Engenharia de Software II, apresentado ao curso de Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Franca, para atender às disposições da Resolução nº 7 de 18 de dezembro de 2018, que estabelece as Diretrizes para a Extensão na Educação Superior.

Prof. Responsável: Prof. Me. Carlos Alberto Lucas

FRANCA
2025

SUMÁRIO

1. INTRODUÇÃO

O coordenador do curso de Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Franca no cumprimento de suas atribuições, apresentou às disposições da Resolução nº 7, e para atender tal demanda, os envolvidos optaram pela construção de uma Landing Page para Instituições Filantrópicas, Organização Não Governamental, Institutos e afins.

No início das aulas regulares, o projeto nos foi apresentado [alunos do 1º semestre do curso de Análise e Desenvolvimento de Sistemas = matutino e noturno].

Em seguida, nos organizamos em grupos com as respectivas responsabilidades, e na sequência buscamos a definição do 'cliente'.

Após nos reunirmos com os responsáveis [cliente], foi elaborada e apresentada uma lista de 'problemas', que necessitavam de um recurso [software] sistêmico. Em seguida, estas questões problemas foram divididas em possíveis soluções sistêmicas.

Tivemos que analisar e identificar as regras do negócio da Instituição, e em seguida, iniciamos o desenvolvimento dos artefatos [a documentação completa de uma solução sistêmica] de Engenharia de Software.

É importante ressaltar que atenderemos com este projeto, as indicações da ONU através da articulação com os Objetivos do Desenvolvimento Sustentável [ODS]: a ODS 1 (erradicação da pobreza), a ODS 8 (trabalho decente e crescimento econômico), a ODS 9 (indústria, inovação e infraestrutura), a ODS 10 (redução das desigualdades) e a ODS 11 (cidades e comunidades sustentáveis).

2. ITINERÁRIO DO PROJETO DE UNIDADE CURRICULAR DE EXTENSÃO – UCE

O presente projeto descreve o projeto desenvolvido por um grupo de estudantes do curso de Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Franca, designado para criar um sistema que administre e organize a entrada e saída de livros da sala de leitura da escola Agostinho Lima de Vilhena. A iniciativa visa melhorar significativamente a organização interna da instituição, alinhando-se aos valores de eficiência e desenvolvimento sustentável que são fundamentais para a sociedade.

2.1 OBJETIVO GERAL

O principal objetivo do projeto é implementar um sistema que otimize a gestão do acervo da Sala de Leitura da Escola Estadual Agostinho Lima de Vilhena, proporcionando maior controle sobre os livros disponíveis, os empréstimos realizados e o estado de conservação das obras. Isso não apenas contribuirá para uma administração mais eficaz dos recursos educacionais, como também promoverá uma operação interna mais organizada e eficiente no ambiente escolar.

A justificativa para a realização deste projeto esta na necessidade identificada de melhorar a infraestrutura tecnológica da Sala de Leitura. A ausência de um sistema informatizado dificulta o acompanhamento do acervo e a rastreabilidade dos livros emprestados. A implementação de um sistema de gestão adequado permitirá a digitalização de processos como, registro, empréstimo e devolução de livros, reduzindo perdas e otimizando a organização das atividades da sala.

Além disso, o projeto busca apoiar os valores educacionais básicos de qualquer escola, que incluem o incentivo à leitura, a valorização do patrimônio público e o compromisso com a formação cidadã dos alunos. Ao fornecer uma solução tecnológica adaptada às necessidades específicas da Sala de Leitura, espera-se não apenas melhorar a qualidade do atendimento aos estudantes, mas também fortalecer o papel da biblioteca como espaço de acesso ao conhecimento e de preservação do acervo literário.

A Sala de Leitura enfrenta uma série de desafios que impactam diretamente sua capacidade de funcionar de forma eficiente e atender às necessidades da comunidade escolar. Um dos principais problemas identificados é a ausência de um sistema de gestão de acervo. Sem um controle digital eficiente, os registros de entrada, saída e estado dos livros são feitos manualmente, o que compromete a precisão dos dados e dificulta a localização das obras.

Além disso, a falta de um processo padronizado para verificar o estado dos livros no momento da devolução gera incertezas quanto à responsabilidade por danos ou perdas. Isso pode resultar em acervos desatualizados, livros danificados circulando entre os alunos e dificuldades em responsabilizar usuários por mau uso.

Diante desses desafios, fica evidente que a Sala de Leitura se beneficiaria a de um sistema informatizado de gestão. Um sistema digital pode automatizar processos, controlar com precisão os empréstimos, devoluções e estados dos livros, além de fornecer dados analíticos por meio de relatórios gerenciais. O desenvolvimento de competências tecnológicas dentro do ambiente escolar também será essencial para garantir a operação contínua do sistema por parte da equipe da escola

Para superar essas dificuldades e promover o crescimento e a sustentabilidade da Sala de Leitura, é essencial investir em tecnologia e capacitação. Ao fazer isso, a escola estará mais bem preparada para oferecer um espaço de leitura eficiente, organizado e acessível a todos, contribuindo de forma direta para o aprendizado dos alunos, a preservação do acervo e o fortalecimento da cultura escolar.

2.1.1 Objetivos específicos

A Sala de Leitura da Escola Estadual Agostinho Lima de Vilhena enfrenta uma série de desafios operacionais que afetam diretamente sua capacidade de oferecer um serviço organizado, transparente e acessível aos estudantes e demais

membros da comunidade escolar. Entre os principais pontos críticos identificados estão a ausência de um controle digital de acervo, a fragilidade no processo de tombamento, a falta de rastreabilidade dos empréstimos e devoluções e a inexistência de uma rotina formal para verificação de danos em livros devolvidos.

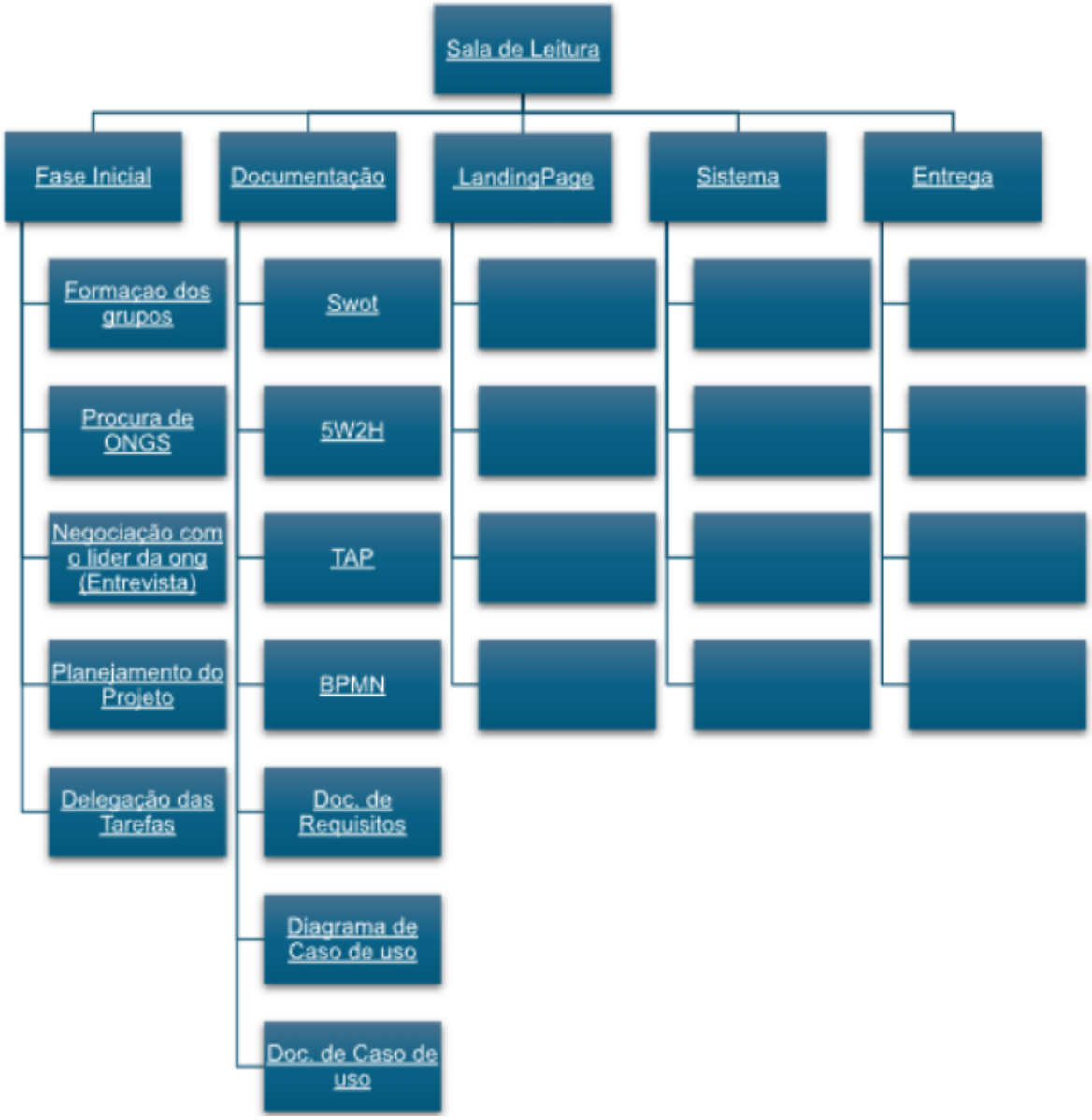
A falta de um sistema de controle informatizado impede que os responsáveis pela sala tenham uma visão clara e atualizada do acervo disponível. Isso dificulta o planejamento e o acompanhamento do uso dos livros, podendo resultar em perdas, extravios e desorganização do espaço físico. Além disso, a ausência de relatórios automatizados compromete a capacidade de tomada de decisões com base em dados confiáveis

Outro problema recorrente é a devolução de livros em mau estado, sem que haja um procedimento sistematizado de avaliação e responsabilização. Devido à incerteza referente ao mau uso isso pode gerar conflitos de diversas formas, sendo assim com uma matriz de rastreabilidade ofertada pelo sistema a margem de erro deste processo seria extremamente baixa.

Diante dessas fragilidades, torna-se evidente a necessidade de desenvolvimento de um sistema informatizado que abranja todas as etapas da gestão da Sala de Leitura, desde o tombamento até a devolução e controle de estado do livro.

Com a implementação deste projeto, espera-se fortalecer a gestão do espaço de leitura da escola, valorizar o acervo, incentivar o hábito da leitura entre os alunos e consolidar um ambiente educativo mais eficiente, acessível e sustentável.

2.2 METODOLOGIA


Sala de Leitura
Fase Inicial
Documentação
LandingPage
Sistema
Entrega
Formaçao dos grupos
Procura de ONGS
Negociação com o lider da ong (Entrevista)
Planejamento do Projeto
Delegação das Tarefas
Swot
5W2H
TAP
BPMN
Doc. de Requisitos
Diagrama de Caso de uso
Doc. de Caso de uso

FORÇAS (STRENGTHS)

- Facilidade de acesso: Fácil contato e rápido feedback referente a qualquer ideia ou prática durante o projeto. Além disso, como dito no 2º artefato uma comunidade disposta abraçar tal projeto.
- Infraestrutura existente: Há infraestrutura para início imediato
- Parcerias institucionais: Conexão com a Diretoria de Ensino de Franca e projetos com APM, Grêmio Estudantil e equipe gestora.
- Foco em inclusão: A escola é referência em educação inclusiva, o que pode atrair apoio institucional.

FRAQUEZAS (WEAKNESSES)

- Processos manuais: Devido a ausência de um sistema prévio tudo é feito e organizado de forma manual.
- Falta de padronização: Empréstimos sem controle eficiente e acervo desorganizado.
- Limitações técnicas: Equipe com pouca experiência em desenvolvimento de sistemas.
- Recursos limitados: Hardware/software básicos podem restringir funcionalidades complexas.

OPORTUNIDADES (OPPORTUNITIES)

- Visibilidade institucional: A land page pode destacar a escola como modelo de gestão educacional.
- Integração com a comunidade: Melhorar a comunicação da sala de leitura com os alunos de forma direta.
- Apoio Geral: Caso seja rentável e de interesse mútuo podemos expandir o projeto de diversas formas.
- Incentivo à leitura: Aumentar empréstimos de livros e participação em eventos culturais.

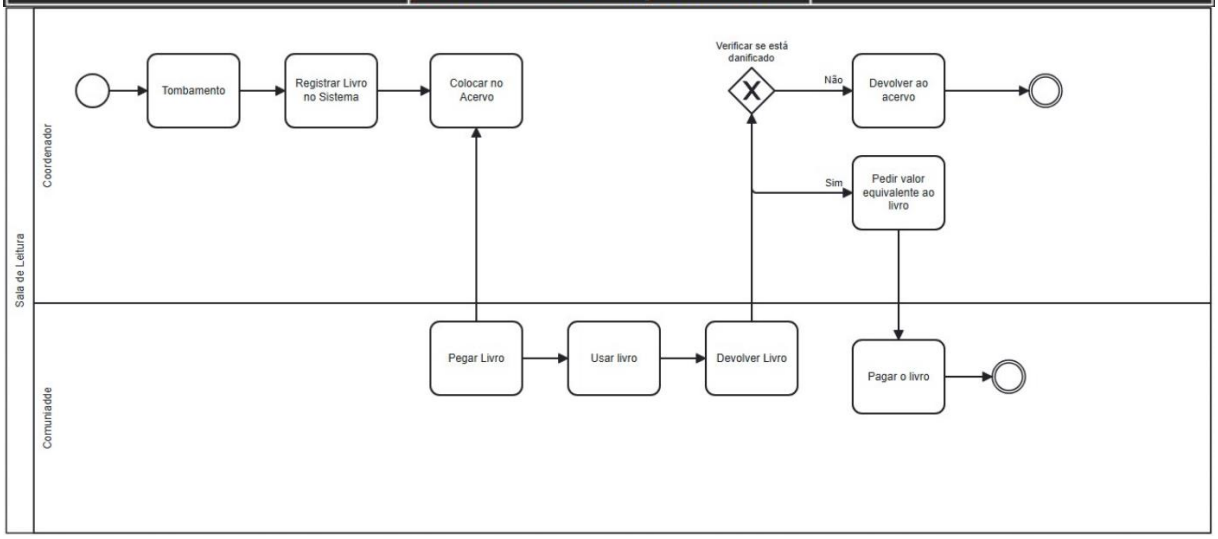
AMEAÇAS (THREATS)

- Falta de engajamento: Colaboradores podem resistir à mudança de processos manuais para digitais.
- Problema com o sistema de empréstimo
- Perda de dados

5W2H	Descrição
What (O quê?)	Desenvolver uma land page e um sistema de gestão para a Sala de Leitura, integrando controle de acervo, empréstimos e comunicação com a comunidade.
Why (Por quê?)	Melhorar a eficiência dos processos, aumentar a visibilidade da escola e otimizar os processos que antes eram manuais perante a sala de leitura.
Where (Onde?)	Sala de Leitura da Escola Estadual Agostinho Lima de Vilhena (Bairro Noêmia, Franca/SP).
When (Quando?)	Início: 13/03/2025
Who (Quem?)	Equipe: Heitor, Rogério, Renan, Victor, Vitor. Orientador: Carlão.
How (Como?)	Etapas: Diagnóstico organizacional, modelagem de requisitos, prototipagem e validação. Tecnologias: HTML, CSS, JavaScript.
How Much (Quanto?)	Até o momento, não existe custo algum.

What (O que será feito?)	Why (Por quê?)	How (Como?)
Desenvolvimento de um sistema interno para o controle de entrada e saída dos livros da sala de leitura.	Para informatizar o processo de funcionamento da sala de leitura.	Através da aplicação dos conhecimentos desenvolvidos em sala de aula.

What (O que será feito?)	Why (Por quê?)	How (Como?)
Logística dos empréstimos	Para obtermos uma melhor rastreabilidade dos livros durante o processo de empréstimo	Adicionando funções de registro ao sistema.



Id: RF001 | Nome: Tombamento Categoria: Evidente | Prioridade: Essencial Descrição: Tombar todos os livros dentro do acervo da biblioteca para garantir a segurança de tal. Informações: - Cada livro registrado deve receber um número sequencial e único. - Deve ser possível incluir informações como título, autor, editora, ano de publicação e localização na biblioteca. -O sistema deve permitir a busca e edição dos registros já tombados. Regra do Negócio: Não tem

Id: RF002 | Nome: Registrar Livro no Sistema Categoria: Evidente | Prioridade: Essencial Descrição: Cadastrar as informações detalhadas dos livros no sistema após o tombamento. Informações: -Incluir dados como título, autor, ano, editora e gênero literário. Associar o livro ao número de tombamento previamente gerado. -Permitir a atualização e edição das informações, se necessário. Regra do Negócio: Não tem

Id: RF003 | Nome: Colocar Livro no Acervo Categoria: Evidente | Prioridade: Essencial Descrição: Disponibilizar o livro registrado para a comunidade no acervo da biblioteca. Informações: O livro deve estar fisicamente disponível nas estantes conforme a organização da biblioteca. -O sistema deve marcar o status do livro como "Disponível para empréstimo". Regra do Negócio: Não tem

Id: RF004 | Nome: Pegar Livro
Categoria: Evidente | Prioridade: Importante
Descrição: Permitir que membros da comunidade retirem livros do acervo para leitura ou empréstimo.
Informações: - O sistema deve registrar a retirada do livro vinculada ao usuário. - Deve haver um controle de prazo para devolução.
Regra do Negócio: Não tem

Id: RF005 | Nome: Usar Livro
Categoria: Evidente | Prioridade: Desejável
Descrição: Utilizar o livro para fins de leitura local ou empréstimo domiciliar.
Informações: - Durante o uso, o livro permanece associado ao usuário até a devolução. - O sistema deve permitir consultas ao histórico de empréstimos.
Regra do Negócio: Não tem

Id: RF006 | Nome: Devolver Livro
Categoria: Evidente | Prioridade: Essencial
Descrição: Receber o livro de volta após o uso, registrando a devolução no sistema.
Informações: - O sistema deve registrar a data de devolução.
O livro deve ser encaminhado para verificação de estado físico.
Regra do Negócio: Não tem

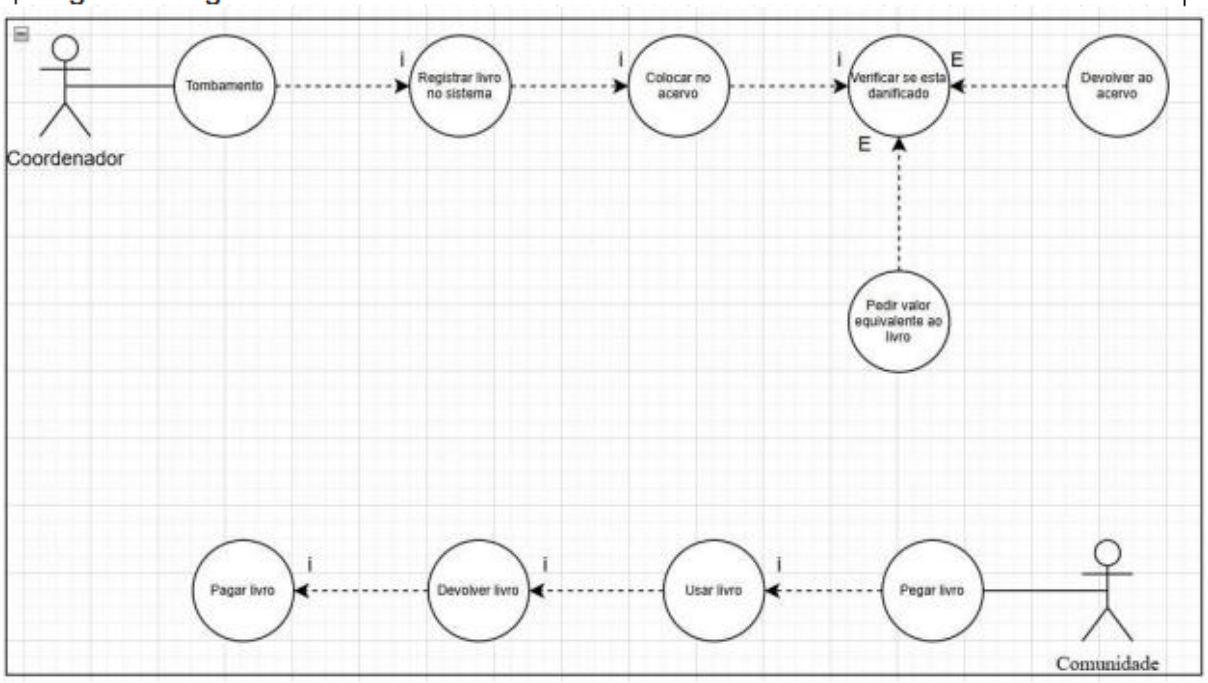
Id: RF007 | Nome: Verificar Estado do Livro
Categoria: Evidente | Prioridade: Essencial
Descrição: Avaliar as condições físicas do livro devolvido para identificar possíveis danos.
Informações: -Caso o livro esteja danificado, deve-se acionar o processo de compensação. - O registro de danos deve ser armazenado no histórico do livro.
Regra do Negócio: Não tem

Id: RF008 | Nome: Devolver Livro ao Acervo
Categoria: Evidente | Prioridade: Essencial
Descrição: Recolocar o livro sem danos no acervo da biblioteca para novo uso.
Informações: - O status do livro deve voltar a "Disponível" no sistema.
Regra do Negócio: Não tem

Id: RF009 | Nome: Solicitar Pagamento pelo Livro Danificado
Categoria: Evidente | Prioridade: Essencial

Descrição: Notificar o usuário para ressarcimento do valor do livro danificado.
Informações: - O sistema deve calcular e informar o valor a ser pago.
O caso deve ser registrado no perfil do usuário.
Regra do Negócio: Não tem

Id: RF010 | Nome: Pagamento de Livro Danificado
Categoria: Evidente | Prioridade: Essencial
Descrição: Receber o pagamento do valor correspondente ao livro danificado.
Informações: - Confirmar o pagamento no sistema e dar baixa na dívida.
Encerrar o processo de compensação para o usuário.
Regra do Negócio: Não tem



ID - UC001

Descrição- Este caso de uso tem por objetivo a chegada dos livros na sala de leitura atendendo o RF001

Ator Primário - Coordenador

Pré-Condição - Não Possui

Cenário Principal - 1 - Os coordenadores solicitam os livros para o estado

2- O estado manda a verba para a compra dos livros

3- Os coordenadores compram os livros

4- Os livros chegam na Sala de Leitura

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - UC002- Registrar livro no Sistema

Extensão - Não Possui

Caso de Uso - Registar Livro no Sistema

ID - UC002

Descrição- Este caso de uso tem por objetivo após a chegada dos livros registrar eles no sistema atendendo o RF002

Ator Primário - Coordenador

Pré-Condição - Tombamento

Cenário Principal - 1 - O Use Case inicia quando os livros chegam na sala de leitura

2- Após a chegada o os coordenadores realizam a contagem dos livros

3- Depois da contagem os coordenadores registram os livros no sistema

Pós Condição - Colocar no Acervo

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - UC003- Colocar no Acervo

Extensão - Não Possui

Caso de Uso - Colocar no Acervo

ID - UC003

Descrição- Este caso de uso tem por objetivo tirar os livros da embalagem após a chegada e colocar nas prateleiras atendendo o RF003

Ator Primário - Coordenador

Pré-Condição - Registar Livro no Sistema

Cenário Principal – 1 - O Use Case inicia quando o coordenador registra o livro no sistema

2- O coordenador Coloca os livros nas prateleiras

Pós Condição - Pegar livro

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - UC004- Pegar livro

Extensão - Não possui

Caso de Uso - Pegar livro

ID - UC004

Descrição- Este caso de uso tem por objetivo a comunidade pegar o livro na Sala de Leitura a atendendo o RF003

Ator Primário - Comunidade

Pré-Condição - Colocar no acervo

Cenário Principal - 1 - O Use Case inicia quando os livros estão na Sala de Leitura

2- A comunidade vai para a Sala de Leitura

3- Um integrante da comunidade escolhe um livro

Pós Condição - Usar livro

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - UC004- Usar livro

Extensão - Não Possui

Caso de Uso - Usar Livro

ID - UC005

Descrição- Este caso de uso tem por objetivo a comunidade pegar o livro na Sala de Leitura e Ler Durante o período estipulado a atendendo o RF004

Ator Primário - Comunidade

Pré-Condição - Pegar Livro

Cenário Principal – 1 - O Use Case inicia quando o integrante da comunidade pega um livro na Sala de Leitura

Pós Condição - Devolver livro

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - UC005- Devolver Livro

Extensão - Não Possui

Caso de Uso - Devolver Livro

ID - UC006

Descrição- Este caso de uso tem por objetivo Após o tempo de leitura consiste no integrante da comunidade devolver o livro atendendo o RF005

Ator Primário - Comunidade

Pré Condição - Usar Livro

Cenário Principal - 1 - O Use Case inicia quando o integrante da comunidade pega um livro na Sala de Leitura

2- O integrante da comunidade lê o livro durante o tempo estipulado

3- O integrante da comunidade devolve o livro para a Sala de Leitura

Pós Condição - Verificar se está danificado

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - Não Possui

Extensão - UC006- Verificar se está danificado

Caso de Uso - Verificar se está danificado

ID - UC007

Descrição- Este caso de uso tem por objetivo após o tempo de leitura consiste no integrante da comunidade devolver o livro atendendo o RF007

Ator Primário - Coordenador

Pré-Condição - Devolver Livro

Cenário Principal - 1 - O Use Case inicia quando o coordenador recebe o livro devolvido pelo integrante da comunidade

2- O coordenador verifica se o livro foi danificado no tempo de empréstimo

Pós Condição - Se não - Devolver ao acervo

Se sim - Pedir o valor equivalente ao livro

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - Não Possui

Extensão - UC008 - Devolver ao acervo, UC009 - Pedir o valor equivalente ao livro

Caso de Uso - Devolver ao acervo

ID - UC008

Descrição- Este caso de uso tem por objetivo após a devolução do livro o coordenador colocar nas prateleiras atendendo o RF008

Ator Primário - Coordenador

Pré-Condição - Verificar se está danificado

Cenário Principal - 1 - O Use Case inicia quando o integrante a comunidade devolve o livro emprestado

2- O coordenador coloca na prateleira de volta

Pós Condição - Não Possui

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - Não Possui

Extensão - UC007- Verificar se está danificado

Caso de Uso - Devolver ao acervo

ID - UC009

Descrição- Este caso de uso tem por objetivo após a devolução do livro o coordenador verifica a qualidade e solicita o valor equivalente ao livro para o integrante da comunidade por mau uso atendendo o RF009

Ator Primário - Coordenador

Pré-Condição - Verificar se está danificado

Cenário Principal – 1 - O Use Case inicia quando o integrante a comunidade devolve o livro emprestado

2- O Verifica a qualidade do livro e verifica defeitos

3- O coordenador pede o valor equivalente ao livro para o integrante da comunidade que teve o livro emprestado

Pós Condição - Pegar o Livro

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - UC010- Pegar o Livro

Extensão - UC007- Verificar se está danificado

Caso de Uso - Pegar livro

ID - UC010

Descrição- Este caso de uso tem por objetivo a comunidade pegar o livro na Sala de Leitura a atendendo o RF010

Ator Primário - Comunidade

Pré-Condição - Devolver ao acervo

Cenário Principal - 1 - O Use Case inicia quando os livros estão na Sala de Leitura

2- A comunidade vai para a Sala de Leitura

3- Um integrante da comunidade escolhe um livro

Pós Condição - Não Possui

Cenário Alternativo - Em qualquer momento pode não ter chegado de livros para a execução do Use Case

Inclusão - Não Possui

Extensão - Não Possui

3. REFERENCIAL TEÓRICO

1)COHN, Mike. Desenvolvimento de Software com Scrum. Editora Bookman. 1ª. Edição. (2023). ISBN: 9788577808076
Concepções/estudos empregados: Leitura e prática = desenvolvimento dos artefatos de software

2)VALENTE, Marco Túlio. Engenharia de Software Moderna. Editora: Independente. 1ª edição (2022). ISBN-10: 6500019504
Concepções/estudos empregados: Leitura e prática = desenvolvimento dos artefatos de software

3)SUTHERLAND, Jeff. Scrum: A arte de fazer o dobro do trabalho na metade do tempo. (2019); Editora Sextante. ISBN-10: 8543107164
Concepções/estudos empregados: Leitura e prática = desenvolvimento dos artefatos de software

4)PRESSMAN, Roger. Engenharia de Software. 9ª Edição. (2021). ISBN-10: 6558040107
Concepções/estudos empregados: Leitura e prática = desenvolvimento dos artefatos de software

4. PRINCIPAIS RESULTADOS

Modelagem de Dados (Contexto: Biblioteca E. E. Agostinho de Vilhena)
A modelagem de dados é o primeiro passo para organizar a informação da biblioteca. O objetivo é criar uma "planta baixa" de como os dados dos alunos, dos livros e dos empréstimos serão guardados e conectados.
Para o projeto da biblioteca, o processo de modelagem esta da seguinte forma:

1. **Entidades Principais (Tabelas):**

usuarios: Quem pega os livros (alunos).
livros: O acervo da biblioteca.
emprestimos: O registro que conecta um usuário a um livro e registra datas de saída e devolução.

2. **Atributos de Cada Entidade (Colunas):**

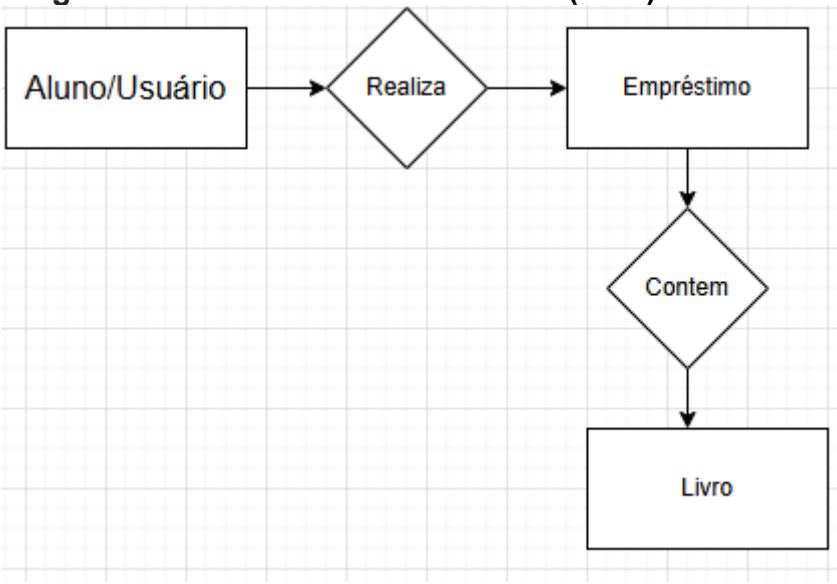
Para usuarios, guardamos nome, email e telefone.
Para livros, guardamos titulo, autor e o status (se está 'disponivel' ou 'emprestado').
Para emprestimos, guardamos as datas (saída e devolução) e o status ('disponível' ou 'emprestado').

3. **Relacionamentos entre Entidades:**

Um usuario pode realizar vários empréstimos, porém não pode ter mais de um empréstimo ativo simultaneamente.
Um livro pode estar em vários emprestimos porém nunca ao mesmo tempo.
A tabela emprestimos é a "ponte": ela usa o livro_id para apontar para um livro específico e o usuario_id para apontar para um usuário específico, registrando quem pegou o quê.

Modelo Entidade-Relacionamento (MER)
O requisito onde “O aluno realiza o empréstimo de um livro”, temos usuário, biblioteca e empréstimo, o usuário (id, nome, telefone) pega um livro (id, titulo, autor, ano_publicacao, editora, status) e para que isso possa entrar no sistema temos que constatar o empréstimo(id, livro_id, usuário_id, data_emprestimo, data_devolucao, status)

Diagrama Entidade-Relacionamento (DER)



Scripts SQL

```
CREATE TABLE usuarios (
    id SERIAL PRIMARY KEY,
    nome TEXT NOT NULL,
    email TEXT UNIQUE,
    telefone TEXT
);
CREATE TABLE livros (
    id SERIAL PRIMARY KEY,
    titulo TEXT NOT NULL,
    autor TEXT,
    ano_publicacao INT,
```

```
    editora TEXT,
    status TEXT CHECK (status IN ('disponivel', 'emprestado')) DEFAULT 'disponivel'
);

CREATE TABLE emprestimos (

 id SERIAL PRIMARY KEY,

livro_id INT NOT NULL REFERENCES livros(id),

 usuario_id INT NOT NULL REFERENCES usuarios(id),

 data_emprestimo DATE NOT NULL DEFAULT CURRENT_DATE,

 data_devolucao DATE,

 status TEXT CHECK (status IN ('ativo', 'concluido')) DEFAULT 'ativo' );
```

CRUD

Biblioteca:

```
package org.example;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class BibliotecaDAO {


    public void inserir(String titulo, String autor, int ano, String editora, boolean status)
{
        String sql = "INSERT INTO livro (titulo, autor, ano, editora, status) VALUES (?,
?, ?, ?, ?)";

        try (Connection conexao = ConnectionFactory.getConnection();
             PreparedStatement stmt = conexao.prepareStatement(sql)) {

            stmt.setString(1, titulo);
            stmt.setString(2, autor);
            stmt.setInt(3, ano);
            stmt.setString(4, editora);
            stmt.setBoolean(5, status);

            stmt.executeUpdate();
            System.out.println(">>> Livro '" + titulo + "' inserido com sucesso!");
```

```
    } catch (SQLException e) {
        System.out.println("Erro ao inserir livro: " + e.getMessage());
    }
}

public List<String> listar() {
    List<String> livros = new ArrayList<>();
    String sql = "SELECT * FROM livro ORDER BY id";

    try (Connection conexao = ConnectionFactory.getConnection();
         Statement stmt = conexao.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {

        while (rs.next()) {
            boolean status = rs.getBoolean("status");
            String statusTexto = status ? "Disponível" : "Emprestado";

            livros.add(rs.getInt("id") + " - " +
                    rs.getString("titulo") + " - " +
                    rs.getString("autor") + " - " +
                    rs.getInt("ano") + " - " +
                    rs.getString("editora") + " - " +
                    statusTexto);
        }

    } catch (SQLException e) {
        System.out.println("Erro ao listar livros: " + e.getMessage());
    }

    return livros;
}


public void remover(int id) {
    String sql = "DELETE FROM livro WHERE id = ?";

    try (Connection conexao = ConnectionFactory.getConnection();
         PreparedStatement stmt = conexao.prepareStatement(sql)) {

        stmt.setInt(1, id);
        int qtdLinhasAfetadas = stmt.executeUpdate();

        if (qtdLinhasAfetadas > 0) {
            System.out.println(">>> Livro removido com sucesso!");
        } else {
            System.out.println(">>> Livro não encontrado!");
        }

    } catch (SQLException e) {
```

```
            System.out.println("Erro ao remover livro: " + e.getMessage());
        }
    }


    public void atualizar(int id, String novoTitulo, String novoAutor, int novoAno, String
novoEditora) {
        String sql = "UPDATE livro SET titulo = ?, autor = ?, ano = ?, editora = ?
WHERE id = ?";

        try (Connection conexao = ConnectionFactory.getConnection();
             PreparedStatement stmt = conexao.prepareStatement(sql)) {

            stmt.setString(1, novoTitulo);
            stmt.setString(2, novoAutor);
            stmt.setInt(3, novoAno);
            stmt.setString(4, novoEditora);
            stmt.setInt(5, id);

            int linhasAfetadas = stmt.executeUpdate();

            if (linhasAfetadas > 0) {
                System.out.println(">>> Livro atualizado com sucesso!");
            } else {
                System.out.println(">>> Livro não encontrado!");
            }

        } catch (SQLException e) {
            System.out.println("Erro ao atualizar livro: " + e.getMessage());
        }
    }


    public void atualizarStatus(int id, boolean novoStatus) {
        String sql = "UPDATE livro SET status = ? WHERE id = ?";

        try (Connection conexao = ConnectionFactory.getConnection();
             PreparedStatement stmt = conexao.prepareStatement(sql)) {

            stmt.setBoolean(1, novoStatus);
            stmt.setInt(2, id);

            int linhasAfetadas = stmt.executeUpdate();

            if (linhasAfetadas > 0) {
                String textoStatus = novoStatus ? "disponível" : "emprestado";
                System.out.println(">>> Status do livro atualizado para: " + textoStatus);
            } else {
                System.out.println(">>> Livro não encontrado!");
            }
```

```
        } catch (SQLException e) {
            System.out.println("Erro ao atualizar status do livro: " + e.getMessage());
        }
    }
}
```

Usuario:

```
package br.edu.fatecfranca;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class UsuariosDAO {

    public void inserir(String nome, String email, String telefone) {
        String sql = "INSERT INTO usuarios (nome, email, telefone) VALUES (?, ?, ?)";

        try (Connection conexao = ConnectionFactory.getConnection();
             PreparedStatement stmt = conexao.prepareStatement(sql)) {

            stmt.setString(1, nome);
            stmt.setString(2, email);
            stmt.setString(3, telefone);

            stmt.executeUpdate();
            System.out.println(">>> Usuário '" + nome + "' inserido com sucesso!");

        } catch (SQLException e) {
            System.out.println("Erro ao inserir usuário: " + e.getMessage());
        }
    }

    public List<String> listar() {
        List<String> usuarios = new ArrayList<>();
        String sql = "SELECT * FROM usuarios ORDER BY id";

        try (Connection conexao = ConnectionFactory.getConnection();
             Statement stmt = conexao.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                usuarios.add(rs.getInt("id") + " - " +
                        "Nome: " + rs.getString("nome") + " - " +
                        "Email: " + rs.getString("email") + " - " +
                        "Telefone: " + rs.getString("telefone"));
            }
```

```
        } catch (SQLException e) {
            System.out.println("Erro ao listar usuários: " + e.getMessage());
        }

        return usuarios;
    }

    public void remover(int id) {
        String sql = "DELETE FROM usuarios WHERE id = ?";

        try (Connection conexao = ConnectionFactory.getConnection();
             PreparedStatement stmt = conexao.prepareStatement(sql)) {

            stmt.setInt(1, id);
            int qtdLinhasAfetadas = stmt.executeUpdate();

            if (qtdLinhasAfetadas > 0) {
                System.out.println(">>> Usuário removido com sucesso!");
            } else {
                System.out.println(">>> Usuário não encontrado!");
            }

        } catch (SQLException e) {
            System.out.println("Erro ao remover usuário: " + e.getMessage());
        }
    }

    public void atualizar(int id, String novoNome, String novoEmail, String
novoTelefone) {
        String sql = "UPDATE usuarios SET nome = ?, email = ?, telefone = ? WHERE
id = ?";

        try (Connection conexao = ConnectionFactory.getConnection();
             PreparedStatement stmt = conexao.prepareStatement(sql)) {

            stmt.setString(1, novoNome);
            stmt.setString(2, novoEmail);
            stmt.setString(3, novoTelefone);
            stmt.setInt(4, id);

            int linhasAfetadas = stmt.executeUpdate();

            if (linhasAfetadas > 0) {
                System.out.println(">>> Usuário atualizado com sucesso!");
            } else {
                System.out.println(">>> Usuário não encontrado!");
            }

        } catch (SQLException e) {
```

```
            System.out.println("Erro ao atualizar usuário: " + e.getMessage());
        }
    }

}
```

Emprestimos:

```
package br.edu.fatecfranca;

import java.sql.*; import java.time.LocalDate; import java.util.ArrayList; import
java.util.List;

public class EmprestimosDAO {

public void inserir(int livro_id, int usuario_id, LocalDate
data_emprestimo, LocalDate data_devolucao) {
   String sql = "INSERT INTO emprestimos (livro_id, usuario_id,
data_emprestimo, data_devolucao, status) VALUES (?, ?, ?, ?, ?)";

   try (Connection conexao = ConnectionFactory.getConnection();
        PreparedStatement stmt = conexao.prepareStatement(sql)) {

       stmt.setInt(1, livro_id);
       stmt.setInt(2, usuario_id);
       stmt.setDate(3, Date.valueOf(data_emprestimo));
       stmt.setDate(4, Date.valueOf(data_devolucao));
       stmt.setString(5, "ativo");

       stmt.executeUpdate();
       System.out.println("Empréstimo adicionado com sucesso!");

   } catch (SQLException e) {
       System.out.println("Erro ao inserir empréstimo: " +
e.getMessage());
   }
}

public List<String> listar() {
   List<String> emprestimos = new ArrayList<>();
   String sql = "SELECT * FROM emprestimos ORDER BY id";

   try (Connection conexao = ConnectionFactory.getConnection();
        Statement stmt = conexao.createStatement();
        ResultSet rs = stmt.executeQuery(sql)) {

       while (rs.next()) {
           String saida = String.format("ID: %d - Livro ID: %d -
Usuário ID: %d - Empréstimo: %s - Devolução: %s - Status: %s",
                   rs.getInt("id"),
                   rs.getInt("livro_id"),
```

```
                    rs.getInt("usuario_id"),
                    rs.getDate("data_emprestimo").toString(),
                    rs.getDate("data_devolucao").toString(),
                    rs.getString("status"));
            emprestimos.add(saida);
        }
    } catch (SQLException e) {
        System.out.println("Erro ao listar empréstimos: " +
e.getMessage());
    }
    return emprestimos;
}

public void remover(int id) {
    String sql = "DELETE FROM emprestimos WHERE id = ?";

    try (Connection conexao = ConnectionFactory.getConnection();
         PreparedStatement instrucao = conexao.prepareStatement(sql))
{

        instrucao.setInt(1, id);
        int nroLinhasAfetadas = instrucao.executeUpdate();

        if (nroLinhasAfetadas > 0) {
            System.out.println("Empréstimo removido com sucesso!");
        } else {
            System.out.println("Empréstimo não encontrado (ID: " + id
+ ").");
        }
    } catch (SQLException e) {
        System.out.println("Erro ao remover empréstimo: " +
e.getMessage());
    }
}

public void atualizar(int id, LocalDate data_devolucao, String
status) {
    String sql = "UPDATE emprestimos SET data_devolucao = ?, status =
? WHERE id = ?";

    try (Connection conexao = ConnectionFactory.getConnection();
         PreparedStatement stmt = conexao.prepareStatement(sql)) {

        stmt.setDate(1, Date.valueOf(data_devolucao));
        stmt.setString(2, status);
        stmt.setInt(3, id);

        int nroLinhasAfetadas = stmt.executeUpdate();
```

```
        if (nroLinhasAfetadas > 0) {
            System.out.println("Empréstimo atualizado com sucesso!");
        } else {
            System.out.println("Empréstimo não encontrado (ID: " + id
+ ").");
        }

    } catch (SQLException e) {
        System.out.println("Erro ao atualizar empréstimo: " +
e.getMessage());
    }
}


}
```

DOCUMENTAÇÃO LINGUAGEM DE PROGRAMAÇÃO

1. Linguagem Java

Conceitos

A linguagem Java foi escolhida como a tecnologia principal para o desenvolvimento do sistema de gestão da biblioteca. Sendo uma linguagem de programação orientada a objetos, robusta e com forte adoção acadêmica e de mercado, ela se alinha perfeitamente ao contexto da disciplina de Linguagem de Programação 1.

Para este projeto, ela forma a base onde toda a lógica de negócio e acesso a dados é construída.

2. Protocolo JDBC

Conceitos

O JDBC (Java Database Connectivity) é uma API padrão do Java que define como a aplicação se comunica com um banco de dados. Ele é a "ponte" entre o nosso código Java (as classes DAO) e o banco de dados PostgreSQL.

O JDBC funciona fornecendo um conjunto de classes e interfaces (como Connection, Statement, PreparedStatement e ResultSet) que nos permitem:

Estabelecer uma conexão com o banco (ConnectionFactory).

Enviar instruções SQL para o banco (PreparedStatement).

Receber e processar os resultados dessas instruções (ResultSet).

Sem o JDBC, nossa aplicação Java não teria como "conversar" com o banco de dados para salvar ou consultar informações.

3. Banco de Dados PostgreSQL

Conceitos

O PostgreSQL é um sistema gerenciador de banco de dados relacional (SGBDR) de código aberto e amplamente robusto. Ele foi escolhido para ser o "cérebro" do armazenamento de dados do nosso sistema.

Enquanto o Java cuida da lógica da aplicação, o PostgreSQL é responsável por armazenar, organizar e garantir a integridade dos dados de forma persistente. Todas as informações de livros, usuários e empréstimos são guardadas em tabelas (livro, usuarios, emprestimos) dentro do banco PI-biblioteca, permitindo consultas rápidas e seguras.

4. Padrão de Projeto DAO (Data Access Object)

Conceitos

O DAO (Data Access Object) é um padrão de projeto que utilizamos para organizar o código. A principal função desse padrão é separar a lógica de negócios da lógica de acesso a dados.

Lógica de Negócios (Classes App): É o que o usuário faz, como "cadastrar um novo usuário" ou "listar livros". Essas classes (ex: AppUsuarios) sabem o que fazer, mas não como fazer.

Lógica de Acesso a Dados (Classes DAO): É onde fica todo o código SQL e JDBC. Essas classes (ex: UsuariosDAO) sabem como se comunicar com o banco de dados para inserir, atualizar, deletar ou buscar dados.

Usar o padrão DAO torna o sistema muito mais fácil de manter. Se no futuro quisermos trocar o PostgreSQL por outro banco de dados, só precisaremos modificar as classes DAO, sem mexer em nenhuma linha das classes App.

Explicação do código-fonte desenvolvido

1. Classe ConnectionFactory

Esta classe tem uma responsabilidade única e fundamental: centralizar e fornecer a conexão com o banco de dados. Ao invés de escrever a URL, usuário e senha em todas as classes DAO, criamos um método estático (getConnection) que faz isso por nós.

Isso garante que, se a senha do banco de dados mudar, só precisamos alterá-la em um único lugar. O método também garante que o driver JDBC do PostgreSQL seja carregado (Class.forName) antes de tentar a conexão.

Exemplo de código-fonte (ConnectionFactory.java):

Java

```
package br.edu.fatecfranca;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnectionFactory {

  // URL de conexão com o banco de dados PI-biblioteca no PostgreSQL
  static String URL = "jdbc:postgresql://localhost:5432/PI-biblioteca";
  static String USER = "postgres";
  static String PASSWORD = "fatec123*";

  // função de conexão com o banco de dados
  public static Connection getConnection(){
     try {
        // Garante que o driver do PostgreSQL seja carregado
        Class.forName("org.postgresql.Driver");
        return DriverManager.getConnection(URL, USER, PASSWORD);
     }
     catch (SQLException e){
        throw new RuntimeException("Erro ao conectar ao banco de dados: " +
e.getMessage());
     } catch (ClassNotFoundException e) {
        throw new RuntimeException("Driver PostgreSQL não encontrado: " +
```

```
e.getMessage());
        }
    }
}
```

2. Classes DAO (ao menos 3) com seus métodos de CRUD

As classes DAO implementam as operações básicas de CRUD (Create, Read, Update, Delete) para cada entidade do nosso sistema.

Entidade 1: UsuariosDAO (CRUD de Usuários) Esta classe é responsável por gerenciar a tabela usuarios. O método inserir, por exemplo, recebe os dados do usuário (vindos da classe App), abre uma conexão, prepara uma instrução SQL segura (PreparedStatement) e a executa.

Exemplo - Método inserir de UsuariosDAO.java:

Java

```
// ... dentro da classe UsuariosDAO ...
public void inserir(String nome, String email, String telefone) {
    // SQL de inserção com parâmetros (?) para segurança
    String sql = "INSERT INTO usuarios (nome, email, telefone) VALUES (?, ?, ?)";

    // try-with-resources garante que a conexão e o statement sejam fechados
    try (Connection conexao = ConnectionFactory.getConnection();
         PreparedStatement stmt = conexao.prepareStatement(sql)) {

        // Substitui os '?' pelos valores reais
        stmt.setString(1, nome);
        stmt.setString(2, email);
        stmt.setString(3, telefone);

        stmt.executeUpdate(); // Executa o comando no banco
        System.out.println(">>> Usuário '" + nome + "' inserido com sucesso!");

    } catch (SQLException e) {
        System.out.println("Erro ao inserir usuário: " + e.getMessage());
    }
}
```

Entidade 2: BibliotecaDAO (CRUD de Livros) Esta classe gerencia a tabela livro. O método listar demonstra como os dados são lidos do banco. Ele executa uma consulta SELECT, percorre os resultados (ResultSet) linha por linha e formata uma List<String> para ser exibida na classe AppBiblioteca.

Exemplo - Método listar de BibliotecaDAO.java:

Java

```
// ... dentro da classe BibliotecaDAO ...
public List<String> listar() {
    List<String> livros = new ArrayList<>();
    String sql = "SELECT * FROM livro ORDER BY id";

    try (Connection conexao = ConnectionFactory.getConnection();
         Statement stmt = conexao.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {
```

```
        while (rs.next()) { // Itera por cada linha de resultado
            boolean status = rs.getBoolean("status");
            String statusTexto = status ? "Disponível" : "Emprestado";

            livros.add(rs.getInt("id") + " - " +
                    rs.getString("titulo") + " - " +
                    rs.getString("autor") + " - " +
                    statusTexto);
        }
    } catch (SQLException e) {
        System.out.println("Erro ao listar livros: " + e.getMessage());
    }
    return livros;
}
```

Entidade 3: EmprestimosDAO (CRUD de Empréstimos) Esta classe gerencia a tabela emprestimos. O método atualizar mostra como modificar um registro existente. Ele permite alterar a data de devolução e o status de um empréstimo específico, identificado pelo seu id.

Exemplo - Método atualizar de EmprestimosDAO.java:

Java

```
// ... dentro da classe EmprestimosDAO ...
public void atualizar(int id, LocalDate data_devolucao, String status) {
    String sql = "UPDATE emprestimos SET data_devolucao = ?, status = ? WHERE id
= ?";

    try (Connection conexao = ConnectionFactory.getConnection();
         PreparedStatement stmt = conexao.prepareStatement(sql)) {

        stmt.setDate(1, Date.valueOf(data_devolucao));
        stmt.setString(2, status);
        stmt.setInt(3, id); // O 'id' é usado no 'WHERE'

        int nroLinhasAfetadas = stmt.executeUpdate();

        if (nroLinhasAfetadas > 0) {
            System.out.println("Empréstimo atualizado com sucesso!");
        } else {
            System.out.println("Empréstimo não encontrado (ID: " + id + ").");
        }
    } catch (SQLException e) {
        System.out.println("Erro ao atualizar empréstimo: " + e.getMessage());
    }
}
```

3. Classe App

As classes App (como AppUsuarios, AppBiblioteca e App de empréstimos) são a camada de interface com o usuário (UI). Elas não contêm nenhuma lógica de banco de dados.

Sua responsabilidade é:
Exibir um menu de opções.
Usar a classe Scanner para ler a escolha do usuário e os dados necessários (nome, título, etc.).
Instanciar a classe DAO correspondente.
Chamar o método apropriado no DAO (ex: dao.inserir(), dao.listar()) e passar os dados que o usuário digitou.
Exibir as respostas (mensagens de sucesso, erro ou listas) vindas do DAO.
Exemplo de código-fonte (AppUsuarios.java):
Java

```
package br.edu.fatecfranca;

import java.util.List;
import java.util.Scanner;

public class AppUsuarios {
  // Instancia o Scanner e o DAO uma vez
  private static Scanner entrada = new Scanner(System.in);
  private static UsuariosDAO dao = new UsuariosDAO();

  public static void main(String[] args) {
     int opcao;
     do {
        // 1. Exibe o menu
        System.out.println("\n\n<<<[MENU USUÁRIOS]>>>");
        System.out.println("1_[Cadastrar usuário]");
        // ... outras opções ...
        System.out.println("0_[Sair]");
        System.out.print("Escolha: ");

        // 2. Lê a opção
        opcao = entrada.nextInt();

        switch (opcao) {
           case 1:
              entrada.nextLine(); // Limpa o buffer
              System.out.print("Insira o nome do usuário: ");
              String nome = entrada.nextLine();
              // ... pede email e telefone ...

              // 4. Chama o DAO
              dao.inserir(nome, email, telefone);
              break;
           // ... outros cases ...
        }
     } while (opcao != 0);
  }
}
```

Descrição de testes

Para validar o funcionamento do sistema, foi realizado um ciclo de testes CRUD para cada uma das três entidades principais. Os testes são executados através das classes App no console.

(Nota: Como não é possível anexar imagens, o processo será descrito textualmente.)

1. Apresentação de um exemplo de CRUD de uma entidade A (Usuários)

Aplicação de Teste: AppUsuarios.java

Create (Cadastrar):

Executar AppUsuarios.java.

Digitar 1 (Cadastrar).

Inserir os dados: Nome: Vitor Saturi, Email: vitor@email.com, Telefone: 16999998888.

Resultado Esperado: Mensagem >>> Usuário 'Vitor Saturi' inserido com sucesso!.

Read (Listar):

No menu, digitar 4 (Listar).

Resultado Esperado: A lista no console deve exibir: 1 - Nome: Vitor Saturi - Email: vitor@email.com - Telefone: 16999998888.

Update (Atualizar):

No menu, digitar 3 (Atualizar).

Informar o ID 1.

Inserir os novos dados: Nome: Vitor Saturi RA, Email: vitor.ra@email.com, Telefone: 16123456789.

Resultado Esperado: Mensagem >>> Usuário atualizado com sucesso!.

Listar (opção 4) novamente para confirmar a alteração dos dados.

Delete (Remover):

No menu, digitar 2 (Remover).

Informar o ID 1.

Resultado Esperado: Mensagem >>> Usuário removido com sucesso!.

Listar (opção 4) novamente para confirmar a remoção.

2. Apresentação de um exemplo de CRUD de uma entidade B (Livros)

Aplicação de Teste: AppBiblioteca.java

Create (Cadastrar):

Executar AppBiblioteca.java.

Digitar 1 (Cadastrar).

Inserir os dados: Título: Java para Iniciantes, Autor: Herbert Schildt, Ano: 2015, Editora: Sextante, Status: 0 (Disponível).

Resultado Esperado: Mensagem >>> Livro 'Java para Iniciantes' inserido com sucesso!.

Read (Listar):

No menu, digitar 4 (Listar).

Resultado Esperado: A lista exibe: 1 - Java para Iniciantes - Herbert Schildt - 2015 - Sextante - Disponível.

Update (Mudar Status):

No menu, digitar 5 (Mudar estado).

Informar o ID 1.

Digitar 1 (Ocupado).

Resultado Esperado: Mensagem >>> Status do livro atualizado para: emprestado.

Listar (opção 4) para confirmar: 1 - Java para Iniciantes - ... - Emprestado.

Delete (Remover):

No menu, digitar 2 (Remover).
Informar o ID 1.
Resultado Esperado: Mensagem >>> Livro removido com sucesso!.
3. Apresentação de um exemplo de CRUD de uma entidade C (Empréstimos)
Aplicação de Teste: App.java (Menu de Empréstimos)
(Pré-requisito: Devem existir um usuário com ID 1 e um livro com ID 1 no banco)
Create (Cadastrar):
Executar App.java.
Digitar 1 (Cadastrar).
Inserir os dados: ID Livro: 1, ID Usuário: 1, Data Empréstimo: 2025-11-17, Data Devolução: 2025-11-24.
Resultado Esperado: Mensagem Empréstimo adicionado com sucesso!.
Read (Listar):
No menu, digitar 4 (Listar).
Resultado Esperado: A lista exibe: ID: 1 - Livro ID: 1 - Usuário ID: 1 - Empréstimo: 2025-11-17 - Devolução: 2025-11-24 - Status: ativo.
Update (Atualizar):
No menu, digitar 2 (Alterar).
Informar o ID do empréstimo: 1.
Inserir Nova Data Devolução: 2025-11-25, Novo Status: concluido.
Resultado Esperado: Mensagem Empréstimo atualizado com sucesso!.
Listar (opção 4) para confirmar: ... Devolução: 2025-11-25 - Status: concluido.
Delete (Excluir):
No menu, digitar 3 (Excluir).
Informar o ID 1.
Resultado Esperado: Mensagem Empréstimo removido com sucesso!.

5. CONSIDERAÇÕES FINAIS

Este semestre nos deu a oportunidade de evoluir muito o projeto em um contexto geral, abordamos o projeto de forma mais “real’’ e conseguimos entender mais como é o processo de um produto mais consistente doque foi abordado no primeiro semestre, isso pois no começo do projeto não existia essa visão de desenvolvimento e como tratar as próximas etapas, creio que esse semestre abriu diversos horizontes para o entendimento da área de sobre como lidar com projetos reais.

6 CONTRIBUIÇÕES DA UCE PARA A FORMAÇÃO DISCENTE

Como da disse anteriormente esse semestre nos permitiu ver algo mais concreto sobre o que é um projeto na área, creio que isso é de extrema importância para escolher uma direção e entender como vamos realmente gerenciar esses projetos, sendo assim quanto mais o curso mais concreto a estrutura que a engenharia de software criou vai perdurar, tal que esse projeto ajuda de diversas formas tanto como individuo tanto quanto profissional.

REFERÊNCIAS

COHN, Mike. Desenvolvimento de Software com Scrum. Editora Bookman. 1ª edição. 2023. ISBN: 9788577808076

VALENTE, Marco Túlio. Engenharia de Software Moderna. Independente. 1ª edição. 2022. ISBN: 6500019504.

SUTHERLAND, Jeff. Scrum: A arte de fazer o dobro do trabalho na metade do tempo. Editora Sextante. 2019. ISBN: 8543107164.

PRESSMAN, Roger. Engenharia de Software. 9ª edição. 2021. ISBN: 6558040107.

COUTINHO. Guia Prático - Engenharia de Software - Projeto. 2020. Disponível online.

BRUM, Bruno Conde Perez; PENA, Leandro. Material de apoio didático. (Utilizado como base para os conceitos de Engenharia de Software e documentação de projetos).

PMI – Project Management Institute. Guia PMBOK – Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos. (Referência utilizada para o Termo de Abertura do Projeto - TAP).

ARTIA. Ferramentas de Gerenciamento de Projetos. 2024. (Utilizado como base teórica para estruturação da EAP - Estrutura Analítica do Projeto).