LETÍCIA BRUDECK, ROBSON DE ALENCAR, LUCAS BAZAGLIA
UNICURITIBA

ANÁLISE DE PROJETOS: Design e Arquitetura

Curitiba
2023

LETÍCIA BRUDECK, ROBSON DE ALENCAR, LUCAS BAZAGLIA

ANÁLISE DE PROJETOS: Design e Arquitetura

Trabalho de graduação apresentado à universidade Unicuritiba, como parte das exigências para obtenção de nota do curso Análise e Desenvolvimento de Sistemas.

Curitiba
2023

RESUMO

Este trabalho tem como objetivo a revisão e aplicação prática dos conceitos teóricos relacionados à Visão e Análise de Projetos, Design e Arquitetura.

O documento resultante da atividade deverá conter uma descrição concisa do projeto, abordando seu propósito, desafios e objetivos. Além disso, serão elaborados diagramas representativos, tais como diagramas de classe, de pacotes ou de componentes, para melhor visualização e compreensão da arquitetura do sistema.

Uma parte significativa do trabalho será dedicada à apresentação de um código parcial do projeto, implementado em qualquer linguagem de programação. Esse código exemplificará aspectos fundamentais do design e da arquitetura do sistema escolhido.

A atividade também incorporará o uso de ferramentas essenciais, tais como o GitHub para controle de versão, o draw.io para criação de diagramas, e uma IDE ou plataformas online para codificação.

Este projeto visa integrar teoria e prática, proporcionando aos participantes uma experiência abrangente na aplicação dos conceitos aprendidos no contexto de projetos reais, além de promover a colaboração efetiva entre os membros do grupo.

Palavras-chave: GitHub. draw.io. IDE. Diagramas.

Sumário

1. INTRODUÇÃO

O presente projeto congrega esforços em torno da exploração e aplicação dos princípios de Visão e Análise de Projetos, com ênfase no Design e Arquitetura, utilizando como campo de estudo o software "Pro Task". Este aplicativo, dedicado ao gerenciamento de atividades, emerge como um protagonista essencial em ambientes de desenvolvimento de software, proporcionando uma plataforma robusta para a coordenação eficaz de tarefas e projetos. O "Pro Task" se destaca como ponto focal de projeto colaborativo, onde os usuários possuem a possibilidade de contextualizar e aplicar os conceitos teóricos revisados em um cenário de desenvolvimento de software.

Ao longo deste projeto, os participantes serão desafiados a revisitar os conceitos teóricos, selecionar e descrever de forma concisa o "Pro Task", elaborar diagramas representativos (tais como diagramas de classe, de pacotes ou de componentes) específicos para este contexto, e desenvolver um código parcial do projeto, enfocando a implementação de funcionalidades fundamentais para o gerenciamento eficaz de atividades no âmbito do desenvolvimento de software.

A utilização de ferramentas como o GitHub para controle de versão, o draw.io para criação de diagramas, e IDEs ou plataformas online para codificação será fundamental para a eficiência e colaboração do grupo.

O resultado final não só refletirá o entendimento aprofundado dos conceitos teóricos, mas também demonstrará a capacidade prática de aplicá-los de maneira relevante e adaptada ao cenário específico do "Pro Task". Este projeto não apenas busca aprimorar as habilidades técnicas dos participantes, mas também promove uma abordagem integradora e pragmática na concepção e desenvolvimento de soluções para desafios reais em ambientes de desenvolvimento de software.

2. Visão e análise de projeto: Design e Arquitetura.

A base teórica que norteia a Visão e Análise de Projetos, Design e Arquitetura é essencial para a criação de sistemas de software robustos e eficientes. Aqui estão alguns conceitos fundamentais a serem revisados:

2.1 Visão de Projetos

Refere-se à compreensão abrangente do propósito, escopo e metas do projeto. Inclui a identificação de stakeholders, requisitos e restrições que moldarão o design e a arquitetura do sistema.

2.2 Análise de Projetos

Envolve a desagregação e compreensão detalhada dos requisitos do sistema. A análise define as funcionalidades necessárias, identifica padrões e estabelece as bases para as fases subsequentes de design e implementação.

2.3 Design

Refere-se à concepção e especificação detalhada do sistema com base nos requisitos levantados. O design envolve decisões sobre a estrutura, comportamento, interfaces e componentes do sistema.

2.4 Arquitetura:

Representa a estrutura global do sistema, definindo seus componentes principais e suas inter-relações. A arquitetura estabelece os alicerces para as decisões de design e influencia diretamente na qualidade e no desempenho do sistema.

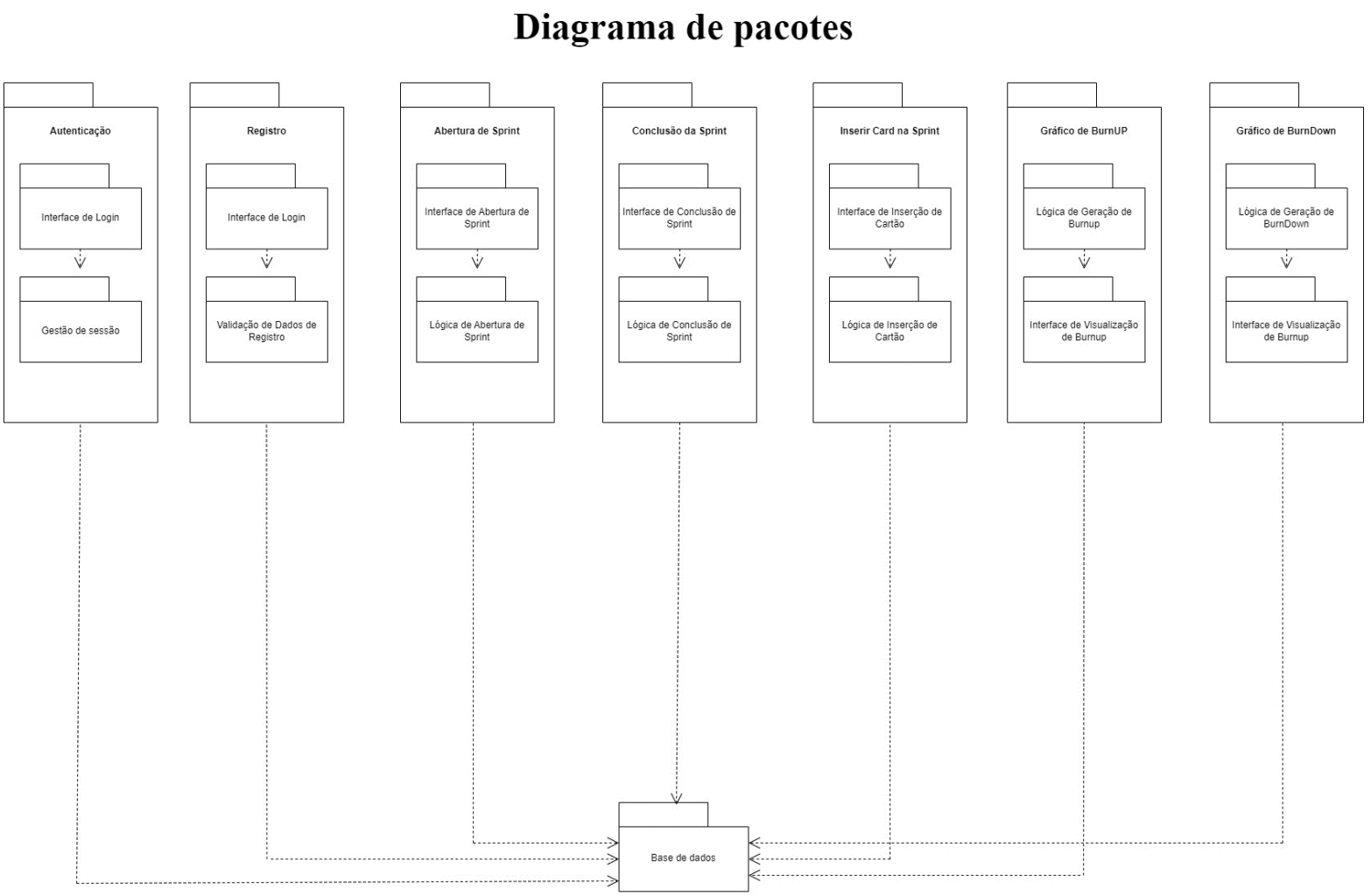
3. Diagramas Utilizados:

3.1 Diagramas de Classe:

Representam a estrutura estática do sistema, identificando as classes, seus atributos e métodos, e as relações entre elas.

3.2 Diagramas de Pacotes:

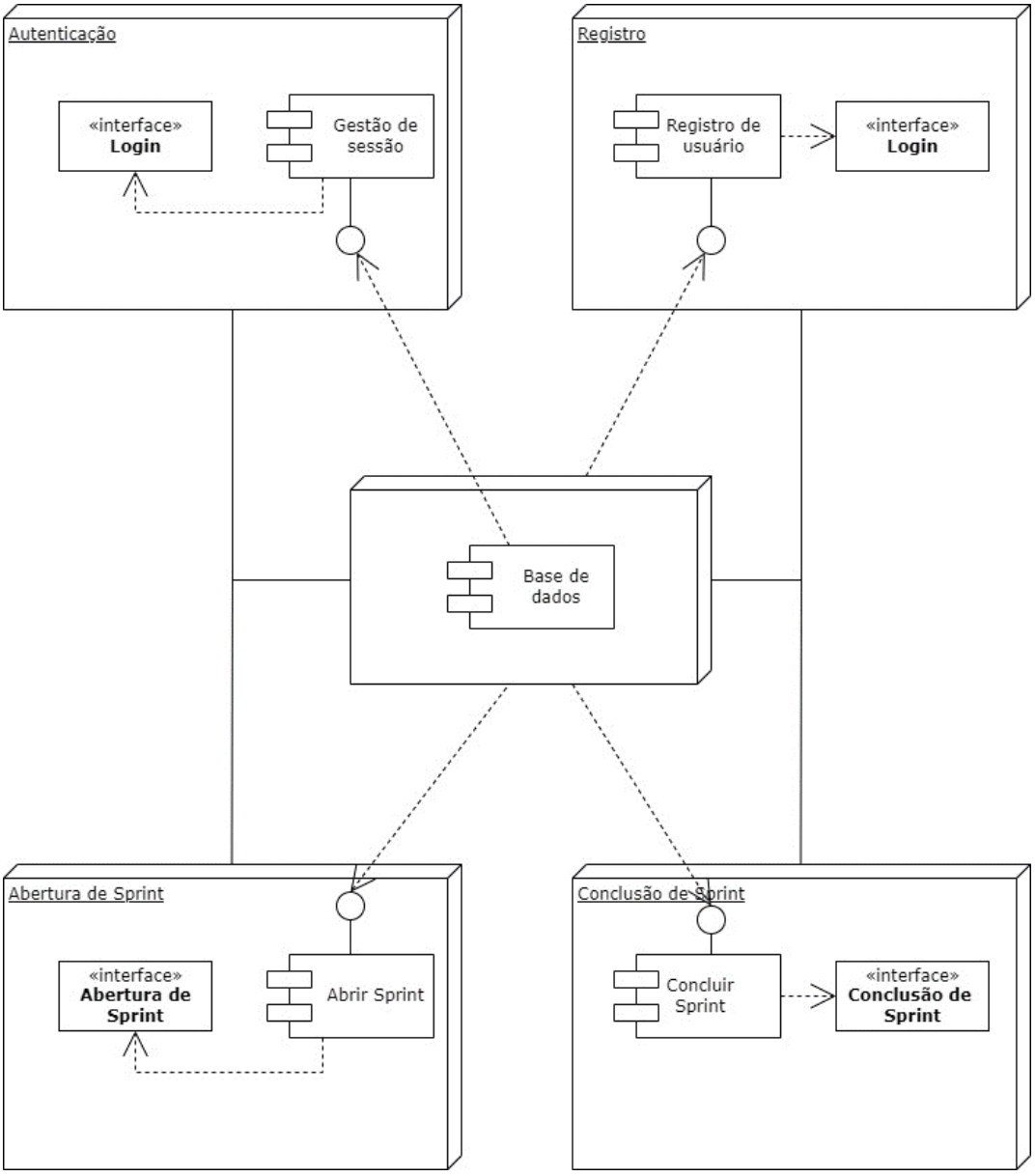
Descrevem a organização modular do sistema, agrupando classes relacionadas em pacotes e mostrando as dependências entre os pacotes.



3.3 Diagramas de Componentes:

Fornecem uma visão de alto nível da organização e das dependências dos componentes do sistema, destacando a distribuição física dos elementos.

Esses diagramas, quando utilizados em conjunto, proporcionam uma representação abrangente e compreensível da visão, análise, design e arquitetura de projetos de software. Eles servem como ferramentas valiosas para a comunicação eficaz entre membros da equipe e stakeholders, contribuindo para o sucesso do desenvolvimento do projeto.



4. Código

```
// Autenticação


public class Autenticacao {
    private ControladorAutenticacao controlador;
    private HistoricoLogin historico;

    public Autenticacao() {
        this.controlador = new ControladorAutenticacao();
        this.historico = new HistoricoLogin();
    }

    public boolean autenticar(String usuario, String senha) {
        boolean autenticado = controlador.verificarCredenciais(usuario, senha);

        if (autenticado) {
            historico.adicionarRegistro(usuario);
        }

        return autenticado;
    }
}

public class ControladorAutenticacao {
    public boolean verificarCredenciais(String usuario, String senha) {
```

```
        return true;
    }
}

public class HistoricoLogin {
    public void adicionarRegistro(String usuario) {
    }
}

public class TokenSessao {
    public String gerarToken(String usuario) {
        return "token_gerado";
    }
}

// Módulo de Registro

public class RegistroUsuario {
    private ControladorRegistro controlador;
    private ValidacaoRegistro validacao;
    private ArmazenamentoUsuario armazenamento;
    private EmailConfirmacao email;

    public RegistroUsuario() {
        this.controlador = new ControladorRegistro();
        this.validacao = new ValidacaoRegistro();
```

```
        this.armazenamento = new ArmazenamentoUsuario();
        this.email = new EmailConfirmacao();
    }

    public void registrarUsuario(DadosUsuario dadosUsuario) {
        if (validacao.validarDados(dadosUsuario)) {
            armazenamento.armazenarDados(dadosUsuario);
            email.enviarConfirmacao(dadosUsuario.getEmail());
        }
    }
}

public class DadosUsuario {
    private String nome;
    private String email;
    private String senha;

    public DadosUsuario(String nome, String email, String senha) {
        this.nome = nome;
        this.email = email;
        this.senha = senha;
    }
}

// TERCEIRO
public class ControladorRegistro {
```

```
}

//QUARTO
public class ValidacaoRegistro {
    public boolean validarDados(DadosUsuario dadosUsuario) {
        return true; //
    }
}

//QUINTO
public class ArmazenamentoUsuario {
    public void armazenarDados(DadosUsuario dadosUsuario) {

    }
}

//SEXTO
public class EmailConfirmacao {
    public void enviarConfirmacao(String email) {

    }
}

// Módulo de Abertura de Sprint
```

```
//PRIMEIRO
public class AberturaSprint {
    private ControladorAberturaSprint controlador;
    private ValidacaoSprint validacao;
    private BancoDadosSprints bancoDados;
    private NotificacaoSprint notificacao;

    public AberturaSprint() {
        this.controlador = new ControladorAberturaSprint();
        this.validacao = new ValidacaoSprint();
        this.bancoDados = new BancoDadosSprints();
        this.notificacao = new NotificacaoSprint();
    }

    public void abrirSprint(DadosSprint dadosSprint) {
        if (validacao.validarDados(dadosSprint)) {
            bancoDados.salvarSprint(dadosSprint);
            notificacao.notificarNovaSprint();
        }
    }
}

//SEGUNDO
public class ControladorAberturaSprint {
}
```

```
//TERCEIRO
public class ValidacaoSprint {
    public boolean validarDados(DadosSprint dadosSprint) {
        return true;
    }
}

//QUARTO
public class BancoDadosSprints {
    public void salvarSprint(DadosSprint dadosSprint) {

    }
}

//QUINTO
public class NotificacaoSprint {
    public void notificarNovaSprint() {

    }
}

//SEXTO
public class DadosSprint {

}
```

CONCLUSÃO

Neste projeto, aplicamos os conceitos teóricos de Visão e Análise de Projetos, Design e Arquitetura no software "Pro Task". A escolha estratégica desse aplicativo proporcionou uma experiência prática e relevante. Os diagramas elaborados ofereceram uma compreensão clara da arquitetura, enquanto a implementação do código parcial demonstrou a tradução eficaz dos conceitos em funcionalidades tangíveis. A colaboração, facilitada por ferramentas como o GitHub e o draw.io, fortaleceu a integridade do projeto. Este trabalho não apenas aprimorou habilidades técnicas, mas também destacou a importância da adaptação e inovação em cenários reais de desenvolvimento de software, preparando-nos para desafios futuros.

Em suma, a conclusão deste projeto reflete não apenas a finalização de uma atividade acadêmica, mas também a consolidação de aprendizados que transcendem o contexto da sala de aula, preparando cada participante para os desafios e oportunidades futuras no campo dinâmico da engenharia de software.

REFERÊNCIAS

DRAW.IO, 2023. Disponível em < https://www.draw.io/>. Acesso em 15 de novembro

GITHUB, 2023. Disponível em < https://github.com/leticiabrus>. Acesso em 15 de novembro