UNIVERSIDADE PAULISTA
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Julia Eduarda Henrique de Souza – RA G749359
Luiz Felipe França de Oliveira – RA G089324
Marcos Vinicius Custodio – RA G824DH8
Milleny Luise Sampaio Evangelista - G74AEC9
Nataly Mariano Pereira - RA G751388

PROJETO DE UM TOTEM DE MUSEU MULTITEMATICO.

SOROCABA

2024

Julia Eduarda Henrique de Souza – RA G749359
Luiz Felipe França de Oliveira – RA G089324
Marcos Vinicius Custodio – RA G824DH8
Milleny Luise Sampaio Evangelista – G74AEC9
Nataly Mariano Pereira – RA G751388

PROJETO DE UM TOTEM DE MUSEU MULTITEMATICO.

Trabalho de conclusão do semestre PIM (Projeto Integrado Multidisciplinar) do curso de Análise e Desenvolvimento de Sistema apresentado à Universidade Paulista – UNIP

SOROCABA
2024

Julia Eduarda Henrique de Souza – RA G749359
Luiz Felipe França de Oliveira – RA G089324
Marcos Vinicius Custodio – RA G824DH8
Milleny Luise Sampaio Evangelista - G74AEC9
Nataly Mariano Pereira – RA G751388

PROJETO DE UM TOTEM DE MUSEU MULTITEMATICO.

Trabalho de conclusão do semestre PIM (projeto integrado multidisciplinar) do curso de Análise e desenvolvimento de sistemas apresentado à Universidade Paulista – UNIP.

Aprovado em:

BANCA EXAMINADORA

Prof. Dr. Marcello Bellodi
Universidade Paulista – UNIP
_____/__/___

Prof. Esp. Fernando A G Bueno
Universidade Paulista – UNIP
_____/__/___

Prof. Me. Richardson Kennedy Luz
Universidade Paulista – UNIP
_____/__/_

RESUMO

Este trabalho descreve o desenvolvimento de um sistema de software destinado a melhorar a gestão de museus multitemáticos através da coleta de feedback dos visitantes sobre as exposições. O sistema utiliza totens interativos, posicionados estrategicamente no museu, permitindo que os visitantes avaliem as obras de arte em exibição de maneira amigável e intuitiva. A principal motivação para este projeto é resolver a dificuldade enfrentada pelos museus em obter feedback detalhado e em tempo real sobre as exposições. Ao coletar opiniões dos visitantes, os administradores podem ajustar e aprimorar as exposições, criando uma experiência mais envolvente e satisfatória para o público. O desenvolvimento do software seguiu metodologias ágeis, incluindo Scrum e Kanban, para garantir uma entrega eficiente e contínua. Além disso, o software foi implementado utilizando a linguagem C# e a plataforma .NET, a avaliação das obras pelos visitantes é realizada através de uma interface digital intuitiva, que coleta dados quantitativos e qualitativos. Esses dados são armazenados e analisados para fornecer insights valiosos aos administradores do museu. O sistema também incorpora medidas de segurança para proteger a privacidade dos dados dos visitantes e garantir a integridade das informações coletadas.

PALAVRAS-CHAVE: Museu Multitemático, Gestão de Exposições, Coleta de Feedback, Metodologias Ágeis, C#, .NET, Totens Interativos, Avaliação de Obras de Arte, Experiência do Visitante, Segurança de Dados

ABSTRACT

This work describes the development of a software system aimed at improving the management of multi-thematic museums through the collection of visitor feedback on exhibitions. The system uses interactive kiosks, strategically positioned within the museum, allowing visitors to evaluate the artworks on display in a friendly and intuitive manner. The main motivation for this project is to address the difficulty museums face in obtaining detailed and real-time feedback on exhibitions. By collecting visitors' opinions, administrators can adjust and enhance the exhibitions, creating a more engaging and satisfying experience for the audience. The software development followed agile methodologies, including Scrum and Kanban, to ensure efficient and continuous delivery. Furthermore, the software was implemented using the C# language and the .NET platform. The evaluation of the artworks by visitors is conducted through an intuitive digital interface, which collects both quantitative and qualitative data. This data is stored and analyzed to provide valuable insights to museum administrators. The system also incorporates security measures to protect visitor data privacy and ensure the integrity of the collected information.

KEYWORDS: Multi-themed Museum, Exhibition Management, Feedback Collection, Agile Methodologies, C#, .NET, Interactive Kiosks, Artwork Evaluation, Visitor Experience, Data Security.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA TABELAS

SUMARIO

1. INTRODUÇÃO

Este documento apresenta o desenvolvimento e a implementação de um sistema de software para avaliação de obras de arte em museus. O projeto busca resolver um problema comum enfrentado pelas instituições culturais: a dificuldade de coletar feedback dos visitantes sobre as exposições em exibição. Sem um mecanismo eficaz de captura de opiniões, torna-se desafiador entender plenamente o impacto e a recepção das coleções exibidas.

Para abordar essa questão, foi desenvolvido um software que permite a coleta de dados em tempo real sobre as percepções dos visitantes. Este sistema utiliza interfaces digitais, como totens, estrategicamente posicionados ao longo das exposições. Os visitantes podem interagir com essas interfaces para deixar suas impressões e avaliações de forma amigável e intuitiva.

A estrutura do documento é organizada para oferecer uma visão detalhada do projeto, começando pela introdução das metodologias que foram utilizadas para gerenciar o desenvolvimento do software. Em seguida, é descrito o problema enfrentado pelos museus e como o software proposto pretende solucioná-lo.

Este documento é essencial para entender o desenvolvimento completo do software de avaliação de obras de arte, desde a concepção inicial até a implementação final, visando melhorar a experiência dos visitantes e otimizar a gestão das exposições nos museus.

1.1. Objetivos

1.1.1. Objetivos do trabalho

Exploração de Metodologias Ágeis: Aplicar e demonstrar a eficácia das metodologias ágeis no gerenciamento do desenvolvimento de software, destacando suas vantagens na adaptação rápida às mudanças e no gerenciamento eficiente do fluxo de trabalho.

Estudo de Caso Aplicado: Realizar um estudo de caso aplicado a um ambiente real de museu, identificando problemas específicos na coleta de feedback dos visitantes e propondo uma solução prática e implementável.

Documentação Completa do Processo: Documentar de maneira detalhada todas as etapas do desenvolvimento do software, incluindo análise de requisitos, design do sistema, desenvolvimento, testes e implantação, servindo como referência para futuros projetos similares.

Contribuição Acadêmica e Profissional: Contribuir para o campo acadêmico e profissional com um trabalho que exemplifica a integração de ferramentas de desenvolvimento moderno e práticas de gerenciamento de projetos, oferecendo insights e práticas recomendadas para a criação de sistemas de avaliação interativos.

Prototipagem e Testes: Desenvolver protótipos das interfaces de usuário e realizar testes de usabilidade para garantir que o sistema atenda às necessidades dos usuários finais, proporcionando uma experiência satisfatória e funcional.

Divulgação dos Resultados: Compartilhar os resultados e as lições aprendidas com a comunidade acadêmica e profissionais da área de museologia e desenvolvimento de software, promovendo a disseminação de conhecimento e a melhoria contínua das práticas de desenvolvimento de sistemas.

1.1.2. Objetivos do software

Coleta Eficaz de Feedback: Desenvolver um sistema que permita a coleta de feedback em tempo real dos visitantes sobre as obras de arte expostas no museu, utilizando interfaces digitais intuitivas e de fácil acesso.

Melhoria Contínua das Exposições: Fornece aos administradores do museu dados quantitativos e qualitativos para identificar quais obras são bem recebidas pelos visitantes e quais necessitam de melhorias ou substituição, facilitando a curadoria dinâmica das exposições.

Engajamento dos Visitantes: Criar uma plataforma que incentive os visitantes a expressarem suas opiniões e avaliações, aumentando o engajamento e a satisfação geral com a experiência no museu.

Interface de Usuário Intuitiva: Garantir que o software possua uma interface de usuário clara e acessível para pessoas de todas as idades e níveis de habilidade técnica, promovendo uma interação amigável e sem barreiras.

Segurança e Privacidade dos Dados: Implementar medidas robustas de segurança para proteger as informações coletadas dos visitantes, assegurando que os dados sejam armazenados e processados de forma segura e conforme as regulamentações de privacidade.

Análise de Dados: Desenvolver funcionalidades para a análise dos dados coletados, permitindo que os administradores do museu visualizem e interpretem as avaliações para tomar decisões informadas sobre as exposições.

Integração e Escalabilidade: Projetar o software de maneira que ele possa ser facilmente integrado com outros sistemas e plataformas utilizadas pelo museu, garantindo escalabilidade para futuros aumentos na quantidade de obras ou visitantes.

2. METODOLOGIA DE ESTUDO

O desenvolvimento eficiente de projetos requer a adoção de metodologias robustas que orientem as etapas de planejamento, execução e avaliação. Este trabalho explora diversas técnicas de gestão e programação que foram essenciais para a estruturação e sucesso do nosso projeto acadêmico. Entre as metodologias aplicadas, destacam-se o 5W1H, que guiou a definição clara dos objetivos e ações; o Scrum, que permitiu uma abordagem ágil e adaptativa ao desenvolvimento; e o ciclo PDCA, que fundamentou o processo contínuo de melhoria. Além disso, a organização e método Kankan foram utilizados para a visualização do fluxo de trabalho, garantindo uma entrega eficaz e sistemática das tarefas. No âmbito do desenvolvimento de software, a programação orientada a objeto em C# através do Visual Studio proporcionou uma plataforma robusta para a implementação de soluções técnicas adequadas às necessidades do projeto. Este documento detalha cada uma dessas metodologias, elucidando como contribuíram conjuntamente para a condução e sucesso do projeto apresentado.

2.1. 5W1H

No desenvolvimento de software, a clareza e a organização são fundamentais. É nesse contexto que a técnica 5W1H se destaca como uma ferramenta poderosa para a comunicação eficaz e a tomada de decisões assertivas. Através de perguntas simples e diretas, o 5W1H permite que desenvolvedores, clientes e stakeholders alinhem suas expectativas, definam objetivos com precisão e naveguem pelos desafios de forma coesa.

Originada da necessidade de responder às perguntas básicas em inglês: What (O que), Why (Porque), Where (Onde), When (Quando), Who (Quem) e How (Como), a 5W1H proporciona uma abordagem sistemática para compreender e abordar os elementos essenciais de uma atividade ou projeto.

- What: Refere-se aos objetivos e requisitos do projeto de software. Ao definir claramente o que precisa ser alcançado, os desenvolvedores

podem estabelecer metas tangíveis e mensuráveis para orientar seu trabalho.

- Why: Esta pergunta aborda a justificativa por trás do projeto de software. Compreender o propósito e os benefícios esperados do software é essencial para manter o foco e a motivação durante o desenvolvimento
- Where: Envolve a identificação dos locais onde o software será utilizado ou implementado. Essa consideração é fundamental para adaptar o software às necessidades específicas do ambiente em que será utilizado
- When: Estabelecer marcos temporais claros ajuda a manter o projeto no caminho certo e a gerenciar efetivamente o tempo e os recursos disponíveis.
- Who: Define as responsabilidades e papéis de cada membro da equipe envolvida no desenvolvimento do software. Clareza nas atribuições de cada indivíduo promove a colaboração e a eficiência no trabalho em equipe.
- How: Esta pergunta se concentra nos métodos e estratégias a serem empregados para atingir os objetivos do projeto. Ao planejar cuidadosamente os processos e abordagens técnicas, os desenvolvedores podem garantir a qualidade e a eficiência do software.

A aplicação do 5W1H no desenvolvimento de software se estende por todas as fases do projeto, desde o planejamento inicial até a implementação e manutenção. Alguns exemplos práticos incluem:

- Elicitação de Requisitos: Ao formular perguntas 5W1H durante a coleta de requisitos, os desenvolvedores obtêm uma compreensão profunda das necessidades dos usuários e stakeholders, garantindo que o software atenda às expectativas de forma eficaz.
- Definição de Escopo: O 5W1H auxilia na delimitação do escopo do projeto, evitando o acúmulo de funcionalidades desnecessárias e garantindo que o foco esteja direcionado aos objetivos principais.
- Gerenciamento de Tarefas: A técnica contribui para a organização e o planejamento de tarefas, definindo responsabilidades, prazos e critérios de entrega, otimizando o tempo e os recursos da equipe.

- Comunicação Eficaz: O 5W1H facilita a comunicação clara e objetiva entre os membros da equipe, clientes e stakeholders, evitando mal-entendidos e garantindo que todos estejam alinhados em relação aos objetivos do projeto.
- Resolução de Problemas: Ao aplicar o 5W1H na análise de problemas, os desenvolvedores podem identificar a raiz das falhas com mais precisão, direcionando os esforços de forma eficiente para a resolução dos desafios.
- A adoção do 5W1H no desenvolvimento de software proporciona diversos benefícios, como:
- Clareza e Precisão: A técnica promove a clareza e a precisão na comunicação, evitando ambiguidades e garantindo que todos os envolvidos estejam na mesma página.
- Alinhamento e Foco: O 5W1H facilita o alinhamento das expectativas entre a equipe, clientes e stakeholders, direcionando o foco para os objetivos principais do projeto.
- Eficiência e Produtividade: A organização e o planejamento proporcionados pela técnica otimizam o tempo e os recursos da equipe, aumentando a eficiência e a produtividade.
- Redução de Riscos: A clareza e a comunicação eficaz contribuem para a redução de riscos, como atrasos, estouros de orçamento e falhas no software.
- Melhoria na Qualidade: O 5W1H auxilia na identificação e resolução de problemas de forma precoce, resultando em um software de maior qualidade.

Ao adotar a metodologia 5W1H, os profissionais de desenvolvimento de software podem minimizar a ambiguidade, identificar potenciais problemas e tomar decisões fundamentadas ao longo do ciclo de vida do projeto. Como destacado por Kim, D. Y. (2002).

> "A aplicação consistente da metodologia 5W1H pode levar a uma melhor compreensão dos requisitos do projeto, maior eficiência na execução das tarefas e, em última análise, à entrega bem-sucedida de um software que atenda às expectativas do cliente".

A tabela 1 apresenta um modelo elaborado pelos colaborardes deste projeto para inicialização do mesmo.

Tabela 1: 5W1H

What	Desenvolvimento de um software que possa ser utilizado para uma melhor experiencia em um museu.
Why	Para auxiliar o museu em facilitar a interação dos visitantes com as obras e a história apresentada na exposição e realizar uma pesquisa durante a visita.
Where	O software será disponibilizado em totens espalhados por toda a estrutura física do museu
When	O software desenvolvido será utilizado durante todo o período de exibição do museu
Who	Equipe desenvolvedora do grupo
How	Será disponibilizado em totens, um software capaz de expor informações sobre a exposição do museu, e colher dados informados pelos visitantes durante a visita por meio de uma pesquisa.

Fonte: Elaborado pelo autor

2.2. PDCA

O ciclo PDCA (Plan-Do-Check-Act) é uma metodologia de melhoria contínua desenvolvida por W. Edwards Deming, amplamente utilizada em diversos setores, inclusive no desenvolvimento de software. O PDCA pode ser uma ferramenta poderosa para garantir qualidade e eficiência no processo de desenvolvimento.

Planejar (Plan): Nesta fase, o objetivo é identificar e analisar os problemas, bem como planejar as soluções e melhorias. No contexto do desenvolvimento de software, isso envolve coletar requisitos, definir metas e criar um plano para alcançar os objetivos do projeto. Como afirmou Deming, "sem dados, você é apenas mais uma pessoa com uma opinião". Portanto, dados precisos e análises detalhadas são fundamentais nesta etapa.

Executar (Do): A fase de execução é onde as atividades planejadas são implementadas. No desenvolvimento de software, isso pode envolver a codificação, o design, os testes iniciais e outras atividades técnicas. É fundamental seguir o plano estabelecido na etapa anterior, mas mantendo a flexibilidade para ajustes quando necessário.

Checar (Check): Durante a fase de verificação, o progresso é avaliado em relação ao plano. Nesta etapa, são conduzidos testes para garantir que o software funcione como esperado. Como disse Deming, "Inspecionar o produto não melhora a qualidade, nem garante a qualidade." O verdadeiro valor desta etapa reside na análise dos resultados para identificar discrepâncias.

Agir (Act): A fase final é sobre agir com base nas lições aprendidas. Aqui, você implementa melhorias ao processo de desenvolvimento de software para corrigir problemas e otimizar futuras iterações. Como Deming afirmou: "A qualidade é a consistência em direção à melhoria".

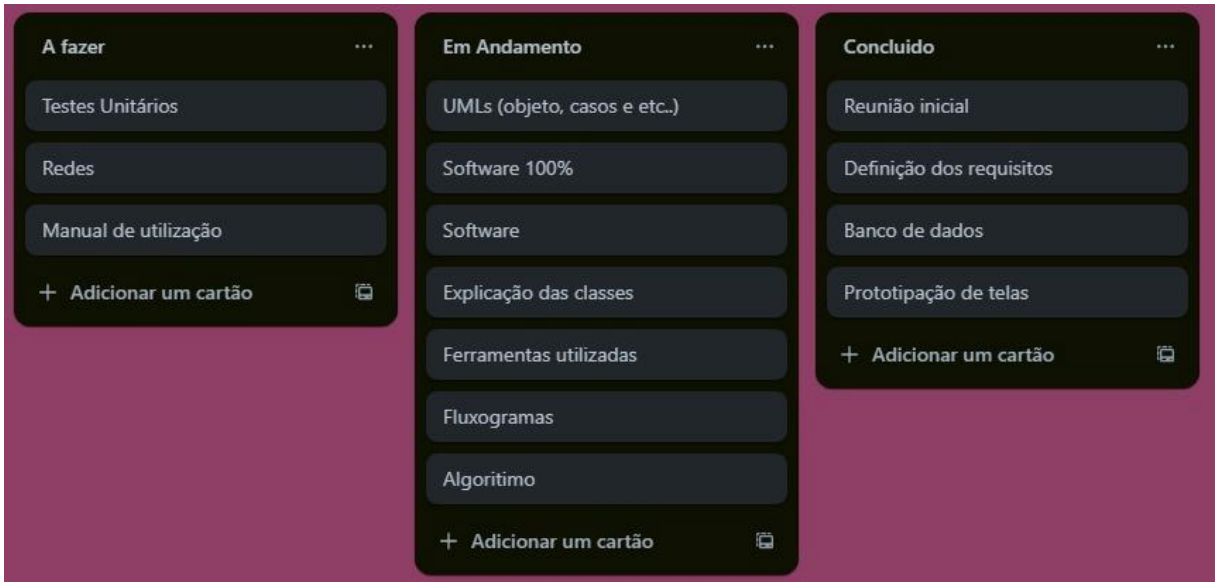
O ciclo PDCA é iterativo e pode ser repetido até que os objetivos desejados sejam alcançados, ajudando as equipes de desenvolvimento a encontrarem problemas, testar soluções e melhorar continuamente o processo de criação de software.

2.3. Kanban

O Kanban é uma metodologia ágil que visa otimizar o fluxo de trabalho e aumentar a eficiência em equipes de desenvolvimento de software. Originado do sistema de produção da Toyota, ele utiliza um quadro visual para gerenciar tarefas e identificar gargalos. Segundo Anderson (2010), "Kanban oferece uma abordagem evolutiva para mudanças incrementais", permitindo que equipes ajustem processos continuamente. A aplicação do Kanban no desenvolvimento de software resulta em maior transparência, melhor colaboração entre membros da equipe e entrega contínua de valor ao cliente (Kniberg & Skarin, 2010). Assim, Kanban promove um ambiente de melhoria contínua e adaptabilidade, essencial para projetos ágeis.

A figura 1 exibe um exemplo do uso do método kanban durante o desenvolvimento deste projeto.

Figura 1: Quadro kanban da equipe



Fonte: Elaborado pelo autor

2.4. Scrum

As empresas vêm mudando o jeito de gerir suas demandas, buscando trabalhar com projetos, por isso faz-se importante compreender a metodologia ágil. A falta de um gerenciamento padronizado para guiar e auxiliar equipes, muitas vezes, causa problemas no repasse de informações, prejudicando a empresa. Assim, este trabalho tem como objetivo apresentar o Movimento Ágil, sua compreensão e finalidade, introduzir as principais metodologias ágeis disponíveis no mercado, focando na metodologia Scrum, suas características, métodos e consequências de utilização no gerenciamento de projetos e os benefícios que proporciona às empresas que a utilizam.

A definição da Metodologia Scrum, nada mais é que uma metodologia ágil usada para a gestão dinâmica de projetos, ela é frequentemente aplicada no desenvolvimento ágil de software. Suas consequências e aplicações por meio de pesquisas bibliográficas em livros, estudos de casos e nas plataformas digitais, resultam eficiência na aplicação do Scrum, todas as áreas da empresa devem estar alinhadas à sua proposta e princípios, seus papéis bem definidos trabalham em conjunto para alcançar os objetivos do projeto de forma eficaz.

No Scrum existem três principais tipos de especificações mais executadas no mercado, são elas:

- Product Owner (PO): O product owner é responsável por representar os interesses dos stakeholders (partes interessadas) e do cliente, mantendo em foco principal a atualização do backlog do produto.
- Scrum Master: O scrum master é o facilitador do processo Scrum, garante a melhoria contínua, seguimento das boas práticas e princípios do Scrum, facilita as reuniões, como o daily scrum também.
- Equipe de Desenvolvimento: É através da equipe que o trabalho acontece! A equipe é um grupo de profissionais que realizam a real do Projeto; eles decidem como organizar as atividades do trabalho e de que forma alcançar os objetivos do Sprint.

3. PROBLEMATICA

O problema abordado pelos museus é a dificuldade em coletar o feedback dos visitantes sobre as obras em exibição. Sem uma maneira eficaz de capturar as opiniões e percepções dos visitantes, é desafiador para as instituições entenderem plenamente o impacto e a recepção de suas coleções. Esse gap de comunicação pode resultar em exposições menos envolventes e uma experiência geral insatisfatória para o público.

Nesse contexto, um software de avaliação de obras pode ser uma solução para essa questão, pois permite coletar dados em tempo real sobre como as obras estão sendo recebidas pelo público. Esse sistema poderia, por exemplo, utilizar interfaces digitais colocadas próximas às obras de arte, onde os visitantes podem deixar suas impressões e avaliações por meio de uma interface amigável e intuitiva. Além disso, o software pode oferecer questionários rápidos e interativos após a visita, incentivando os visitantes a expressarem suas opiniões de forma mais detalhada.

Portanto, será criado um software a fim de avaliar as obras, podendo ser uma ferramenta ótima para qualquer museu que deseja melhorar a capacidade de avaliar e responder às questões dos visitantes. Esse software não só facilitará a coleta de feedback valioso, mas também ajudará os curadores e gestores de museus a adaptarem suas exposições com base nas preferências e feedback do público. Assim, o museu pode se tornar um espaço mais dinâmico e interativo, aumentando a satisfação dos visitantes e enriquecendo a experiência cultural oferecida.

4. INTRODUÇÃO AO SOFTWARE

O software desenvolvido tem como principal objetivo coletar feedback de visitantes sobre as obras exibidas em um museu. Ele facilita a avaliação das exposições, permitindo que os visitantes respondam a perguntas específicas sobre sua experiência com as obras. Essas avaliações são, então, utilizadas pelos administradores do museu para decidir sobre a manutenção, melhoria ou remoção de determinadas exposições.

Durante o processo de desenvolvimento uma de suas primeiras etapas é o levantamento de requisitos, estes requisitos servem para identificar as necessidades dos usuários, as funcionalidades esperadas e os critérios de desempenho. A análise detalhada vai dos conteúdos apresentados da exposição do museu e das interações esperadas com os visitantes.

Os requisitos funcionais tendem a descrever as funções, comportamentos do sistema, suas funcionalidades. Nele define o comportamento com termos de entradas, saídas, coleta de dados que foram determinados com o cliente, assim como podemos ver na tabela 2.

Tabela 2: Requisitos funcionais

Requisitos Funcionais	
Requisitos	**Descrição**
Apresentação de Conteúdo Histórico	O totem é capaz de exibir informações detalhadas sobre a primeira viagem do homem à Lua, inclui o contexto histórico da missão Apollo 11 e detalhes sobre os astronautas envolvidos.
Feedback do Usuário	Inclui mecanismos que coleta o feedback dos usuários, fazendo cálculos de médias gerais e individuais, como pesquisas rápidas, para melhorar de forma constante a experiência do visitante.

Teclado	Teclas alfabéticas, ele é compatível com totem e o sistema operacional, garante segurança de dados para proteger as avaliações respondidas pelos usuários.
Armazenamento dos Dados	Todos os dados coletados vindos das respostas dos visitantes, são armazenados em uma estrutura de banco de dados.

Fonte: Elaborado pelo autor

Os requisitos não funcionais, não determina o que foi feito. Mas são fundamentais para definir como o sistema se comportará em determinadas fases. O objetivo dos requisitos não funcionais aplicados no software é abranger aspectos de segurança, recursos do hardware essenciais, desempenho, portabilidade e além de serem levantados juntamente com os requisitos funcionais, que ambos fazem parte da engenharia de software. De acordo com a tabela 3, está exemplificado como cada aspecto de requisitos não funcionais foi aplicado no software com a utilização da ISO 9126 que é uma norma internacional que fornece um quadro de referência para a avaliação da qualidade de software.

Tabela 3: Requisitos não funcionais

Requisitos Não Funcionais	
Funcionalidades	**Descrição**
Funcionabilidade	Adequação: O software é capaz de avaliar obras oferecendo todas as funcionalidades necessárias como teclado on-screen, perguntas de avaliação e cálculo de médias.
	Acurácia: Os cálculos das médias das respostas do cliente e geral das obras são precisos.
Confiabilidade	Maturidade: O software é vigoroso e não falha frequentemente, garantindo uma boa experiencia aos clientes.

Usabilidade	Inteligibilidade: A interface do usuário é fácil de entender e utilizar, com uma navegação intuitiva e fluida com feedback claro.
	Apreensibilidade: O software é fácil de aprender, permitindo que usuários possam usá-la rapidamente
	Operacionalidade: O software possui uma tabela on-screen que é responsivo e fácil de utilizar.
Manutenibilidade	Modificabilidade: O software é atualizável e modificável presencialmente sem grandes complicações.
	Estabilidade: Mudanças no software foram implementadas sem introduzir novos problemas.
	Testabilidade: O software pode ser facilmente testado garantindo funcionalidades e atualizações estão corretas
Portabilidade	Adaptabilidade: O software é feito para funcionar em Windows 11.
	Instabilidade: O software é fácil de instalar e configurar no hardware do museu.
	Coexistência: O software é capaz de coexistir com outros sistemas no totem sem problemas no funcionamento.

Fonte: Elaborado pelo autor

O software funciona em duas principais fases:

- Coleta de Feedback: Apresenta uma série de perguntas ao visitante sobre uma obra específica. Coleta as respostas fornecidas, que são classificadas em uma escala de 0 a 5.
- Análise de Feedback: A análise de feedback é uma parte crucial do software, pois fornece informações valiosas tanto para os visitantes quanto para os responsáveis do museu.

Este software será utilizado no ambiente de um museu, especificamente nas áreas onde estão localizadas as exposições. Visitantes que desejarem avaliar uma obra podem utilizar o software através de dispositivos eletrônicos (como tablets ou quiosques de autoatendimento) dispostos em pontos estratégicos do museu. Adicionalmente, os administradores do museu usarão o software para visualizar e analisar os feedbacks coletados, permitindo-lhes tomar decisões informadas sobre as exposições. Tendo como principais objetivos:

- Melhoria Contínua das Exposições: Identificar as obras que são bem recebidas pelos visitantes e aquelas que necessitam de melhorias ou substituição.
- Engajamento dos Visitantes: Incentivar os visitantes a expressarem suas opiniões sobre as exposições, aumentando o engajamento e a satisfação geral com a experiência no museu.
- Decisões Informadas: Fornecer dados quantitativos e qualitativos aos administradores para ajudá-los a tomar decisões estratégicas sobre as exposições.
- Facilidade de Uso: Desenvolver uma interface amigável e intuitiva para os visitantes de todas as idades e níveis de habilidade técnica.

4.1. Introdução a linguagem em C#

C# é uma linguagem de programação moderna, orientada a objetos e de tipagem forte, desenvolvida pela Microsoft como parte da plataforma .NET. Lançada no início dos anos 2000, C# combina a flexibilidade e a potência de linguagens como C++ com a simplicidade e a segurança de linguagens como Java, tornando-se uma escolha popular para o desenvolvimento de aplicações desktop, web e móveis. Como citado no livro "Programming C# 8.0: Build Cloud, Web, and Desktop Applications" de Ian Griffiths:

> "C# é uma linguagem que proporciona a flexibilidade e a potência de linguagens como C++ enquanto mantém a simplicidade e a segurança de linguagens como Java. É uma linguagem essencial para qualquer desenvolvedor que deseja criar aplicações robustas e eficientes para a plataforma .NET" (Griffiths, 2019, p. 3).

As principais características do C# incluem:

Orientação a Objetos: C# suporta os principais pilares da programação orientada a objetos, como encapsulamento, herança e polimorfismo.

Sintaxe Simples e Clara: A sintaxe do C# é influenciada por outras linguagens, como C++ e Java, tornando-a fácil de aprender para desenvolvedores familiarizados com essas linguagens.

Segurança e Confiabilidade: C# inclui vários recursos de segurança de tipos e gerenciamento automático de memória, como coleta de lixo, que ajudam a evitar erros comuns de programação.

O projeto utiliza C# para definir e implementar todas as suas funcionalidades principais.

A linguagem C# é usada em diversas partes do software, incluindo:

- Interação com a Interface de Usuário: Utilizamos o Windows Forms para criar a interface gráfica do usuário. O C# permite manipular eventos e interações do usuário com os elementos da interface.
- Lógica de Negócio: O C# é utilizado para implementar a lógica de negócio, como a validação de dados de entrada, processamento de informações e cálculo das médias de feedbacks.
- Acesso a Dados: O software interage com fontes de dados para armazenar e recuperar informações. Utilizamos C# para gerenciar essas operações de maneira eficiente e segura.
- Gerenciamento de Tarefas: O projeto utiliza threads para gerenciar tarefas simultâneas, melhorando o desempenho e a capacidade de resposta do software.

4.2. Introdução à programação a objeto (POO)

A POO é um paradigma de programação que se baseia na criação de classes e objetos. As principais características da POO incluem:

- Classes: São estruturas que definem os atributos e métodos comuns a um grupo de objetos. Elas servem como modelos para a criação de objetos.
- Objetos: São instâncias de classes. Cada objeto possui seus próprios atributos e métodos, mas compartilha o comportamento definido pela classe.
- Herança: É um mecanismo que permite que uma classe herde atributos e métodos de outra classe. Isso promove a reutilização de código e facilita a organização hierárquica das classes.
- Polimorfismo: Permite que objetos de classes diferentes possam ser tratados de maneira uniforme, através de uma referência comum. Isso proporciona flexibilidade e extensibilidade ao código.
- Encapsulamento: É o princípio de ocultar a implementação interna de um objeto e expor apenas uma interface pública para interação. Isso promove a segurança e a manutenibilidade do código.

No software desenvolvido, a POO é amplamente utilizada para modelar e organizar os diferentes componentes do sistema. Por exemplo:

- Classes: São utilizadas para representar entidades como obras de arte, usuários e feedbacks. Cada classe define os atributos e métodos relacionados à sua respectiva entidade.
- Objetos: São criados a partir das classes para representar instâncias específicas das entidades. Por exemplo, um objeto da classe "ExibirObras.cs" representa uma obra específica exibida no museu.
- Herança: É aplicada para representar relacionamentos entre diferentes tipos de obras de arte, como seus nome, descritivos e imagens, que compartilham características comuns.
- Polimorfismo: Permite tratar diferentes tipos de obras de arte de forma uniforme ao calcular médias de feedbacks ou ao exibir informações sobre as obras.

- Encapsulamento: Ajuda a proteger os dados sensíveis e controlar o acesso às propriedades e métodos das classes. Por exemplo, o encapsulamento é utilizado para garantir que apenas as classes necessárias tenham acesso aos objetos da “ListaDeObras.cs”.

4.3. Fluxo

O algoritmo de fluxo de processos descreve o funcionamento básico do processo de interação do usuário com o sistema de avaliação de obras de arte. Ele é parte fundamental do software, pois orienta as etapas que o usuário segue desde o início até a conclusão da avaliação.

O algoritmo foi desenvolvido para guiar o usuário através das seguintes etapas:

- Início: O usuário é solicitado a inserir seu nome.
- Inserção do Nome: O usuário insere seu nome.
- Verificação de Obras Disponíveis: Verifica se há obras disponíveis para exibição.
    - Se houver obras:
        - Exibição das Obras: Exibe a primeira obra disponível e solicita o feedback do usuário.
        - Verificação de Mais Obras: Verifica se há mais obras disponíveis para exibição.
            - Se houver mais obras, o processo continua com a próxima obra.
            - Se não houver mais obras, o processo avança para a etapa de compilação dos feedbacks.
    - Se não houver obras, o processo avança diretamente para a etapa de compilação dos feedbacks.
- Compilação dos Feedbacks: Calcula a média dos feedbacks fornecidos pelo usuário.
- Apresentação do Feedback Pessoal: Apresenta ao usuário a média de seus feedbacks para as obras avaliadas.
- Finalização do Processo: Conclui o processo de avaliação.

Este algoritmo é utilizado internamente pelo software de avaliação de obras de arte para orientar o fluxo de interação do usuário. Ele é executado conforme o usuário interage com a interface do software, guiando-o desde a inserção do nome até a

apresentação de seu feedback médio. O quadro 1 apresenta um exemplo de algoritmo em Portugol.

Quadro 1: Algoritimo Portugol

```
Escreva("Por favor, insira seu nome:")
Leia(nome)
Se (obras_disponiveis for verdadeiro) Então
   obra_atual <- 1

   Repita
      Escreva("Por favor, forneça seu feedback para a obra ", obra_atual, ":")
      Leia(feedbacks[obra_atual])

      Se (mais_obras for verdadeiro) Então
         obra_atual <- obra_atual + 1
      Senão
         mais_obras <- falso
      Fim Se
   Até que (mais_obras seja falso)
Senão
    Passo 8: Ir para o passo 9
Fim Se
total_feedbacks <- 0
Para cada feedback em feedbacks
   total_feedbacks <- total_feedbacks + feedback
Fim Para
media_feedback <- total_feedbacks / tamanho(feedbacks)
Escreva("O seu feedback médio para as obras foi de: ", media_feedback)
Escreva("O processo foi concluído. Obrigado!")
```

Fonte: Elaborado pelo autor

4.4. Protótipo de telas

O objetivo é documentar as telas do protótipo do software para facilitar a compreensão e o desenvolvimento. Esta seção cobre a descrição das três

telasprincipais do protótipo: tela de boas-vindas, tela de avaliação da obra, e tela de feedback final.

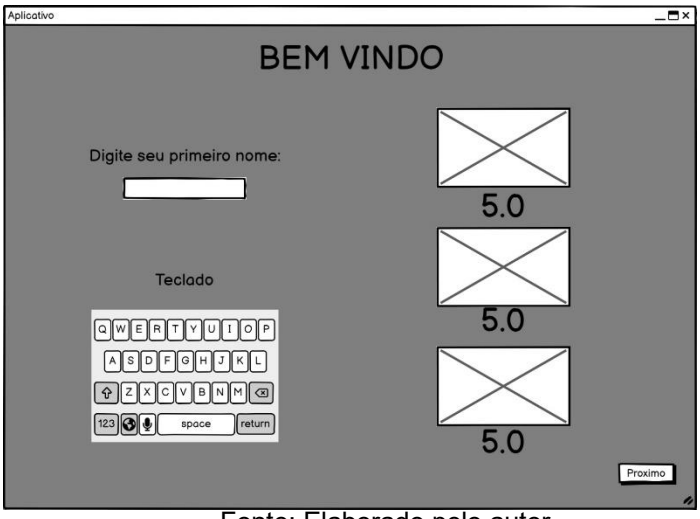
Mapa do Software:

- Tela de Boas-vindas
- Tela de Avaliação da Obra
- Tela de Feedback Final

A primeira tela (figura 2) coleta o nome do cliente e mostra o feedback geral das obras.

- Componentes:
  - Campo de texto para entrada do nome do usuário.
  - Imagem placeholder para "Obra X" com a nota abaixo.
  - Imagem placeholder para "Obra Y" com a nota abaixo.
  - Imagem placeholder para "Obra Z" com a nota abaixo.
  - Botão "Próximo".
  - Teclado para inserir o nome.
- Layout:
  - Título "BEM-VINDO" centralizado no topo da tela.
  - Campo de texto abaixo do título com o rótulo "Digite seu primeiro nome".
  - Três imagens placeholders lado a lado com legendas "OBRA X", "OBRA Y" e “OBRA Z” abaixo delas, cada uma acompanhada de sua nota.
  - Botão "Próximo" no canto inferior direito.
  - Teclado no lado direito.

Figura 2 Protótipo tela inicial



Fonte: Elaborado pelo autor

A segunda tela (figura 3) o cliente vai avaliar as obras respondendo às perguntas de satisfação.

- Componentes:
  - Imagem placeholder da obra de arte.
  - Título da obra ao lado da imagem.
  - Descrição da obra abaixo do título.
  - Quatro perguntas com campos de resposta (escala de 0 a 5).
  - Botão "Próximo".
  - Botões para inserir a resposta.
- Layout:
  - Imagem da obra posicionada no canto superior esquerdo.
  - da obra ao lado direito da imagem.
  - Descrição da obra abaixo do título.
  - Quatro perguntas dispostas verticalmente com campos de resposta ao lado direito.

- Botão "Próximo" no canto inferior direito.

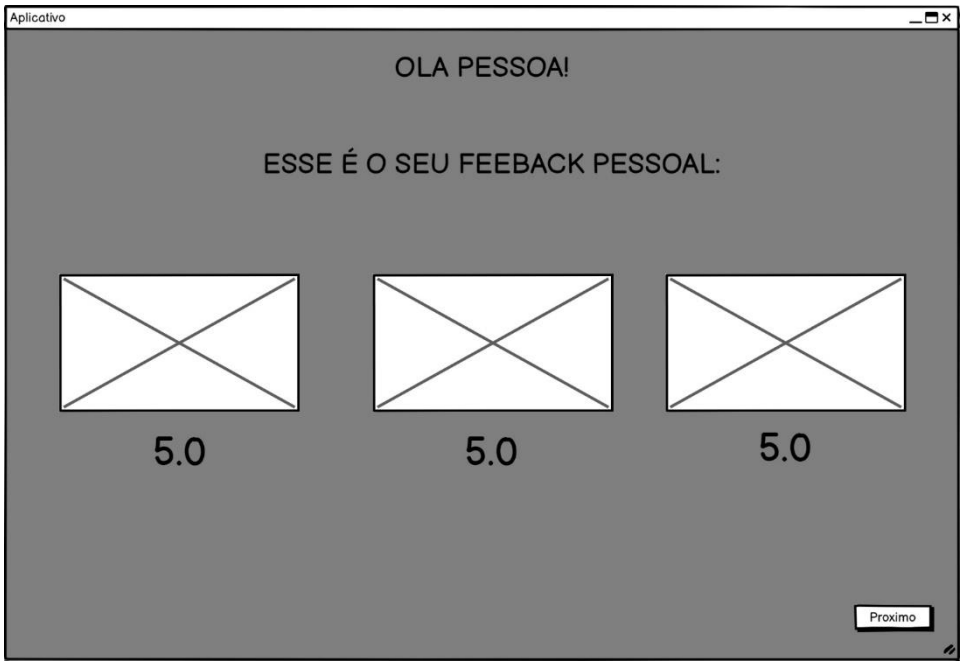
Figura 3 Protótipo tela obras

Fonte: Elaborado pelo autor

A terceira tela (figura 4) mostra ao usuário um resumo do seu feedback fornecido e agradece a participação.

- Componentes:
  - Título para a tela "OLA PESSOA! ESSE É O SEU FEEDBACK PARA AS OBRAS".
  - Imagem placeholder para "Obra X" com a nota abaixo.
  - Imagem placeholder para "Obra Y" com a nota abaixo.
  - Imagem placeholder para "Obra Z" com a nota abaixo.
  - Texto de agradecimento "OBRIGADO POR PARTICIPAR! VOLTE SEMPRE".
  - Botão "Próximo".
- Layout:
  - Título da tela no topo da tela.
  - Duas imagens placeholders lado a lado com legendas "OBRA X", "OBRA Y" e “OBRA Z” abaixo delas, cada uma acompanhada da nota de feedback do cliente.
  - Texto de agradecimento centralizado na parte inferior da tela.
  - "Próximo" no canto inferior direito.

Figura 4 Protótipo tela feedback



Fonte: Elaborado pelo autor

4.5. Fluxo de telas

A tela inicial (figura 5) tem como finalidade apresentar um feedback geral das obras ao cliente. Além disso, o cliente insere seu nome por meio do teclado, essencial para dar continuidade às funcionalidades do software.

- Título "Seja Bem-vindo ao museu do Espaço" centralizado no topo da tela.
- Campo de texto abaixo do título com o rótulo "Digite seu primeiro nome".
- Teclado abaixo da caixa de texto.
- Três imagens das obras lado a lado com suas notas de feedback geral acima.
- Botão "Iniciar" no canto inferior direito.

Figura 5 Tela inicial

Fonte: Elaborado pelo autor

A tela de avaliação (figura 6), o cliente é apresentado a um conjunto de cinco questões de múltipla escolha relacionadas às três obras e organizadas em uma estrutura de loop.

- Imagem da obra posicionada no canto superior esquerdo.
- Título da obra acima da imagem.
- Descrição da obra abaixo da imagem.
- Cinco perguntas dispostas verticalmente com campos de resposta em múltipla escolha abaixo.
- Botão "Avançar" no canto inferior direito.

Figura 6: Tela de avaliação das obras

Fonte: Elaborado pelo autor

A tela de feedback (figura 7) mostra o feedback pessoal do cliente em uma escala de zero a cinco com relação as 3 obras para depois ser adicionado ao feedback geral e voltar a página principal.

- Mensagem com o nome do cliente que foi inserido na tela principal.
- Três imagens das obras lado a lado, cada uma acompanhada da nota de feedback do cliente abaixo.
- Texto de agradecimento na parte inferior da tela.
- Botão "Voltar" no canto inferior direito direcionando de volta a tela principal.

Figura 7 Tela de feedback

Fonte: Elaborado pelo autor

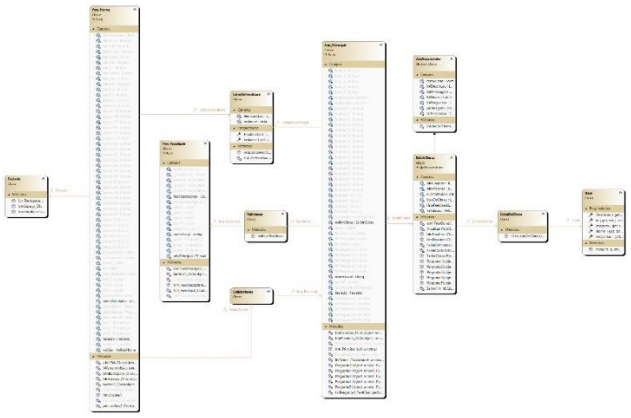
4.6. UML

A UML (Unified Modeling Language) é uma linguagem de modelagem padronizada utilizada para especificar, visualizar, construir e documentar os artefatos de um sistema de software. Ela oferece uma série de diagramas que ajudam a representar diferentes aspectos de um sistema, como a estrutura estática (diagrama de classes), comportamento dinâmico (diagrama de sequência) e casos de uso (diagrama de casos de uso). Criada para unificar diversas metodologias de modelagem, a UML é amplamente utilizada na engenharia de software para facilitar a comunicação entre desenvolvedores, analistas e stakeholders, promovendo uma melhor compreensão e documentação dos sistemas complexos.

4.6.1. Diagrama de classes

O diagrama de classes é uma representação visual utilizada na engenharia de software para descrever a estrutura de um sistema, mostrando suas classes, atributos, métodos e os relacionamentos entre elas. Ele faz parte da UML (Unified Modeling Language) e é fundamental no processo de modelagem orientada a objetos. Com esse diagrama, é possível visualizar a arquitetura do software, facilitando a compreensão, a comunicação entre os membros da equipe e a identificação de possíveis melhorias e erros na estrutura do sistema. A figura 8 apresenta o diagrama de classes do software desenvolvido neste projeto.

Figura 8: Diagrama de classes

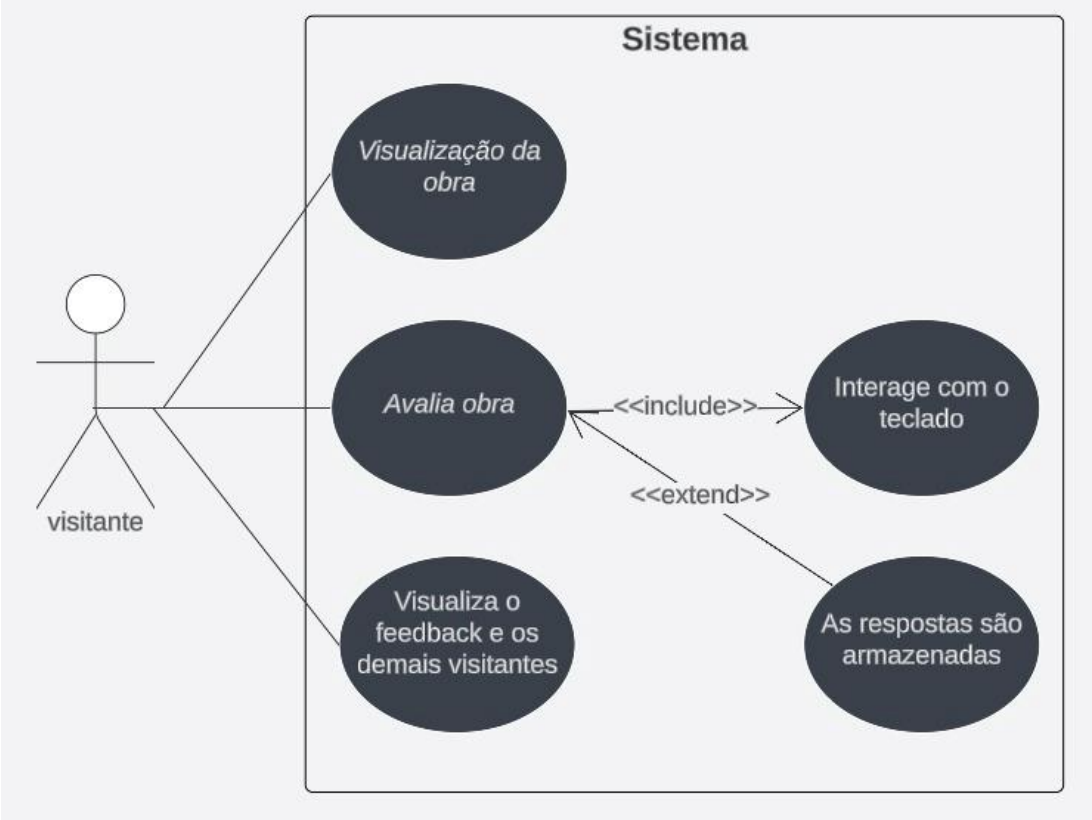


Fonte: Elaborado pelo autor

4.6.2. Diagrama de caso de uso

De acordo com os requisitos do software criado para o engajamento do sistema, segue abaixo na figura 9, o diagrama de caso de uso, onde expõe a execução do sistema, tendo visualizações das obras, avaliações, logo interagindo com o teclado e automaticamente com os dados armazenados, gerando feedbacks avaliados pelos visitantes.

Figura 9 Diagrama de caso de uso

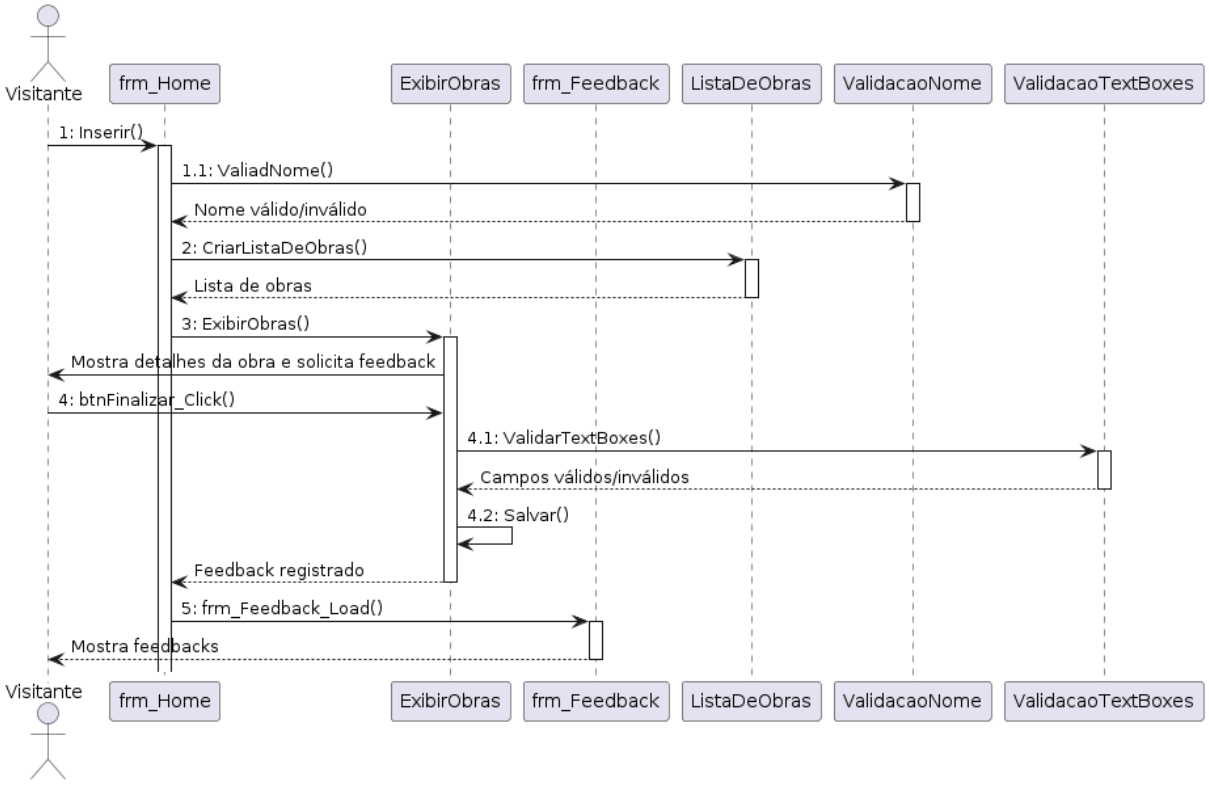


Fonte: Elaborado pelo autor

4.6.3. Diagrama de sequência

O diagrama de sequência é um tipo de diagrama da UML (Unified Modeling Language) que ilustra como os objetos de um sistema interagem entre si ao longo do tempo. Ele foca na ordem cronológica das mensagens trocadas entre os objetos para realizar uma funcionalidade específica. Cada objeto é representado por uma linha vertical, e as mensagens são exibidas como setas horizontais entre essas linhas. O diagrama de sequência é útil para detalhar o fluxo de controle, esclarecer o comportamento dinâmico do sistema e identificar possíveis problemas de comunicação ou dependências. Ele é essencial na fase de design, ajudando desenvolvedores e analistas a visualizarem e validar a lógica das interações do sistema. A figura 10 representa o diagrama de sequência do software desenvolvido.

Figura 10: Diagrama de sequência



Fonte: Elaborado pelo autor

4.7. Explicação das classes

A classe “Teclado” é responsável por fornecer funcionalidades relacionadas à interação do usuário com o teclado virtual durante o processo de avaliação das obras de arte. Ela contém métodos para inserção de caracteres e outras operações relacionadas à entrada de dados.

- Fornece métodos para inserção de caracteres no campo de texto através da função “Inserir”, assim como descrito no Quadro 2.

Quadro 2Codigo C# da funsão Inserir ()

```
public   void   Inserir(object sender, EventArgs e, TextBox txb, CheckBox chk ){
        Button btn = (Button)sender;
        CheckBox chkCapsLock = chk;
        if (chkCapsLock.Checked == true)
        {
            txb.Text = txb.Text + btn.Text.ToUpper();
        }
        else
        {
            txb.Text = txb.Text + btn.Text.ToLower();
        }
    }
```

Fonte: Elaborado pelo autor

- Apagar texto inserido pelo usuário através do teclado, utilizando a função “btnBackspace_Click”, descrito no Quadro 3.

Quadro 3Codigo C# da função btnBackspace_click ()

```
public void btnBackspace_Click(object sender, EventArgs e, TextBox
txb){
    txb.Text = string.Empty;
}
```

Fonte: Elaborado pelo autor

- Validar a entrada de dados do visitante através da classe “ValidaNome”, assim como descrito no Quadro 4

Quadro 4: Codigo C# da classe "ValidaNome"

```
internal class ValidaNome
{
    public string validaNome(string txtNome){
        Regex regex = new Regex(@"^[A-Za-z\s]+$");
        if (regex.IsMatch(txtNome)){
            return "";
        }else{
            return "O texto não é um nome válido. Por favor, insira apenas letras e espaços.";
        }
    }
}
```

Fonte: Elaborado pelo autor

A classe “ExibirObras” é responsável por gerenciar a exibição das obras de arte disponíveis para avaliação no museu. Ela coordena a apresentação das informações sobre cada obra, incluindo nome, descrição, imagem e questionários de feedback.

- Carregar e exibir os detalhes de cada obra de arte, incluindo nome, descrição e imagem presente no Quadro 5, que por sua vez inicia um loop presente em “ExibirDadosObra”, disponível no Quadro 6.

Quadro 5 Codigo C# da função ExibirObras ()

```
public ExibirObras(Form formulario, Button btnProximo, Button btnFinalizar){
    this.btnProximo = btnProximo;
    this.btnFinalizar = btnFinalizar;
    ListaDeObras listaDeObras = new ListaDeObras();
    this.listaDeObras = listaDeObras.CriarListaDeObras();
    listaFeedbackLocal = new List<(int id, int media)>();
    InicializarElementos(formulario);
    Exibir(currentIndex);
    AtualizarVisibilidadeBotoes();
}
```

Fonte: Elaborado pelo autor

- Além de realizar o loop nas obras disponíveis (Quadro 5), “ExibirDadosObra” deve apresentar questionários de feedback para os usuários responderem, presente no quadro 6.

Quadro 6Codigo C# da função ExibirDadosObra ()

```
private void ExibirDadosObra(Item obra){
    lblMensagem.Text = "";
    lblNome.Text = obra.Nome;
    lblDescricao.Text = "Descrição: " + obra.Descricao;
    picImagem.Image = Image.FromFile($"C:\\Images\\{obra.Imagem}");
    for (int i = 0; i < 5; i++){
        if (i < obra.Perguntas.Count){
            lblPerguntas[i].Text = obra.Perguntas[i];
            txtRespostas[i].Text = "";
            lblPerguntas[i].Visible = true;
            txtRespostas[i].Visible = true;
            tbxAtual = txtRespostas[i];
        }else{
            lblPerguntas[i].Visible = false;
            txtRespostas[i].Visible = false;
        }
    }
}
```

Fonte: Elaborado pelo autor

- Coletar e processar os feedbacks fornecidos pelos usuários (Quadro 7).

Quadro 7 Codigo C# da função Salvar ()

```
                    private void Salvar(int id, List<int> respostas){
                      double media = respostas.Average();
                      int mediaInt = Convert.ToInt32(Math.Round(media));
                      ListaDeFeedback.Instance.AdicionarFeedback(id,
mediaInt);
                    }
```

Fonte: Elaborado pelo autor

- Navegar entre as obras de arte disponíveis durante o processo de avaliação (Quadro 8 e 9).

Quadro 8 Codigo C# da função btnProximo_click ()

```
                public void btnProximo_Click(object sender, EventArgs e){
                  if (validacao.ValidarTextBoxes(txtRespostas) == ""){
                    int obraId = listaDeObras[currentIndex].Id;
                    List<int> respostas = txtRespostas.Select(txt
=>int.Parse(txt.Text)).ToList();
                    Salvar(obraId, respostas);
                    double media = respostas.Average();
                    listaFeedbackLocal.Add((obraId, (int)Math.Round(media)));
                    if (currentIndex < listaDeObras.Count - 1){
                      currentIndex++;
                      Exibir(currentIndex);
                      AtualizarVisibilidadeBotoes();
                    }
                  }else{
                    lblMensagem.Text = validacao.ValidarTextBoxes(txtRespostas);
                  }
                }
```

Fonte: Elaborado pelo autor

Quadro 9 Codigo C# da função btnFinalizar_click ()

```
public void btnFinalizar_Click(object sender, EventArgs e, string nome){
    if (validacao.ValidarTextBoxes(txtRespostas) == ""){
        int obraId = listaDeObras[currentIndex].Id;
        List<int> respostas = txtRespostas.Select(txt => int.Parse(txt.Text)).ToList();
        Salvar(obraId, respostas);
        double media = respostas.Average();
        listaFeedbackLocal.Add((obraId, (int)Math.Round(media)));
        formulario.Close();
        Thread telaObra = new Thread(() => abrirTelaObras(nome, listaFeedbackLocal));
        telaObra.SetApartmentState(ApartmentState.STA);
        telaObra.Start();
    }else{
        lblMensagem.Text = validacao.ValidarTextBoxes(txtRespostas);
    }
}
```

Fonte: Elaborado pelo autor

A classe “ExibirObras” depende das classes ListaDeObras e ListaDeFeedback para acessar as informações sobre as obras disponíveis e armazenar os feedbacks fornecidos pelos usuários.

A classe “ListaDeObras” é responsável por gerenciar a lista de obras de arte disponíveis para avaliação no museu. Ela fornece métodos para criar, manter e recuperar informações sobre as obras de arte.

Ela oferece métodos para adicionar, remover e atualizar obras de arte na lista. Além disso, a classe permite acessar informações sobre cada obra de arte, incluindo nome, descrição e imagem, facilitando sua exibição na interface do usuário, Quadro 10.

Quadro 10 Codigo C# da classe ListaDeObras

```
public class Item
{
    public int Id { get; set; }
    public string Nome { get; set; }
    public string Imagem { get; set; }
    public string Descricao { get; set; }
    public List<string> Perguntas { get; set; }
    public Item(int id, string nome, string imagem, string descricao, List<string> perguntas)
    {
        Id = id;
        Nome = nome;
        Imagem = imagem;
        Descricao = descricao;
        Perguntas = perguntas;
    }
}
public class ListaDeObras{
    public List<Item> CriarListaDeObras()
    {
        var listaDeItens = new List<Item>();
        listaDeItens.Add(new Item(1, "NOME", "IMAGEM", "DESCRIÇÃO", new List<string> { "PERGUNTA1", "PERGUNTA2" }));
        return listaDeItens;
    }
}
```

Fonte: Elaborado pelo autor

A classe “ListaDeObras” é utilizada pela classe “ExibirObras” para acessar as informações sobre as obras disponíveis e apresentá-las aos usuários durante o processo de avaliação.

A classe “ListaDeFeedback” é responsável por armazenar os feedbacks fornecidos pelos usuários sobre as obras de arte. Ela mantém uma lista dos feedbacks recebidos, associando cada feedback à obra de arte correspondente.

desempenha um papel fundamental na coleta e gerenciamento dos feedbacks fornecidos pelos usuários durante o processo de avaliação das obras de arte em exposição. Ela é responsável por armazenar de forma organizada os feedbacks associados a cada obra de arte, permitindo uma análise detalhada e posterior processamento. Além disso, a classe oferece métodos para adicionar novos feedbacks, recuperar os feedbacks existentes e realizar operações de manipulação conforme necessário. Essas funcionalidades são essenciais para obter insights sobre a percepção dos visitantes em relação às obras de arte e tomar decisões informadas sobre o gerenciamento da exposição (Quadro 11).

Quadro 11 Codigo C# da classe ListaDeFeedback

```
public class ListaDeFeedback
{
    private static ListaDeFeedback instance;
    private List<(int id, int media)> feedbackList;
    private ListaDeFeedback(){
        feedbackList = new List<(int id, int media)>();
    }
    public static ListaDeFeedback Instance{
        get{
            if (instance == null)
            {
                instance = new ListaDeFeedback();
            }
            return instance;
        }
    }
    public List<(int id, int media)> FeedbackList{
        get { return feedbackList; }
    }
    // Método para adicionar feedback à lista
    public void AdicionarFeedback(int id, int media){
        feedbackList.Add((id, media));
    }
}
```

Fonte: Elaborado pelo autor

A classe ListaDeFeedback é utilizada pela classe ExibirObras para armazenar os feedbacks fornecidos pelos usuários durante o processo de avaliação das obras de arte.

5. BANCO DE DADOS

No banco de dados, estabelecemos um relacionamento entre duas entidades principais: "Obras" e "Feedback". Esse relacionamento é essencial para registrar as interações dos usuários com as obras apresentadas. Quando um usuário interage com uma obra, visualizando sua imagem, lendo sua descrição e título, ele tem a oportunidade de fornecer feedback sobre essa obra respondendo às perguntas sobre ela.

Após coletar o feedback, o sistema calcula uma média para cada obra. Essa média é exibida ao usuário ao final da exibição. Esses dados podem ser valiosos para entender quais obras são mais populares, que geram mais engajamento e quais podem precisar de ajustes ou melhorias.

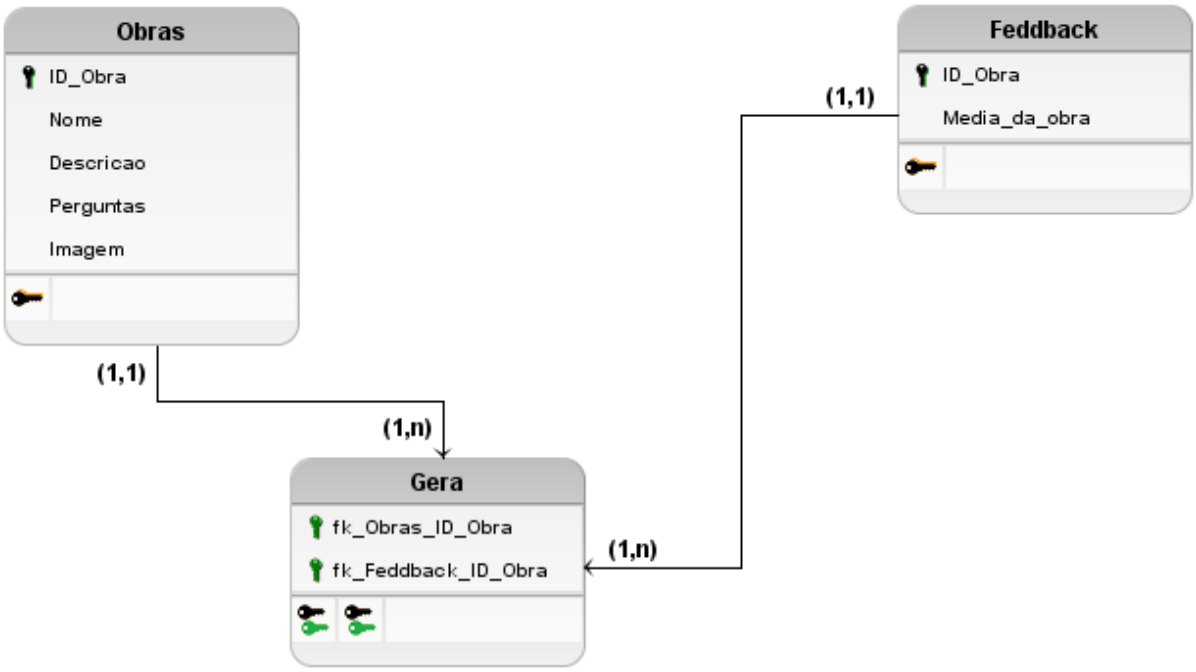
5.1. MER e DER

A modelagem de Entidade-Relacionamento (MER) é fundamental no design de um banco de dados, representando os dados e relacionamentos de forma clara e precisa. O Diagrama de Entidade-Relacionamento (DER) é uma representação mais visual e ampla do MER, que permite descrever a estrutura de um banco em termos de entidades, atributos e relacionamentos entre elas.

No Modelo Entidade-Relacionamento (MER) temos a representação das duas entidades “Obras” e “Feedback”, cada uma com seus atributos e seu relacionamento sendo representado por uma linha conectando-as e indicando a associação entre as duas.

Na figura 11 a entidade “Obras” tem como atributos "id_obra" (chave primária), "nome", "descrição", “perguntas” e “imagem”. Esta é a entidade principal que representa as obras em seu sistema. A entidade “Feedback” tem como atributos "id_obra" (chave estrangeira referenciando a chave primária "id_obra" na entidade "Obras") e "media_da_obra". Esta entidade representa o feedback gerado para cada obra.

Figura 11 Modelo entidade de relacionamento

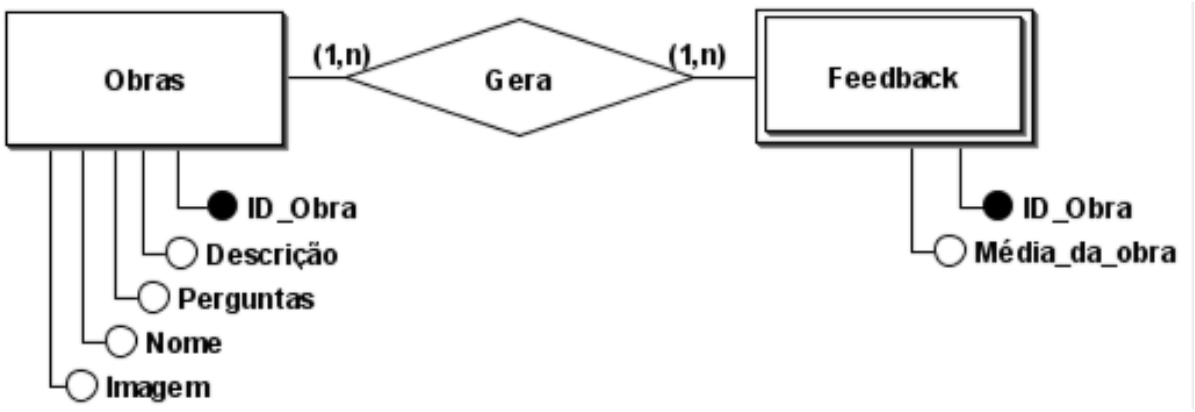


Fonte: Elaborado pelo autor

No DER temos uma representação visual das entidades e seu relacionamento. Cada entidade é representada por um retângulo (“Obras e Feedback), o relacionamento é representado por um losango e linhas fazendo as conexões.

Na figura 12 a entidade “Obras” é representada por um retângulo com os atributos “id_obra", "nome", "descrição", “perguntas” e “imagem” listados dentro dele. A entidade fraca “Feedback” é representada por outro retângulo com os atributos "id_obra" e "média_da_obra" listados dentro dele, uma linha conecta a entidade "Obras" à entidade "Feedback" indicando o relacionamento entre elas de “Gerar”, e uma notação indica suas cardinalidades de uma obra pode ter muitos feedbacks (um-para-muitos).

Figura 12 Diagrama entidade - relacionamento



Fonte: Elaborado pelo autor

Para a criação do Modelo Entidade-Relacionamento e do Diagrama Entidade-Relacionamento utilizamos o BRModelo que é um software de modelagem de dados utilizado para criar Diagramas de Entidade-Relacionamento (DER) e Modelos de Entidade-Relacionamento (MER).

Com o BRModelo é possível utilizar recursos de validação que ajudam a identificar erros e inconsistências no modelo de dados, garantindo a integridade e qualidade do projeto. Ele também permite a geração de scripts SQL a partir do modelo criado, facilitando a implementação do banco de dados em sistemas de gerenciamento de banco de dados como MySQL, PostgreSQL, SQL Server, entre outros.

6. REDES

6.1. Topologia

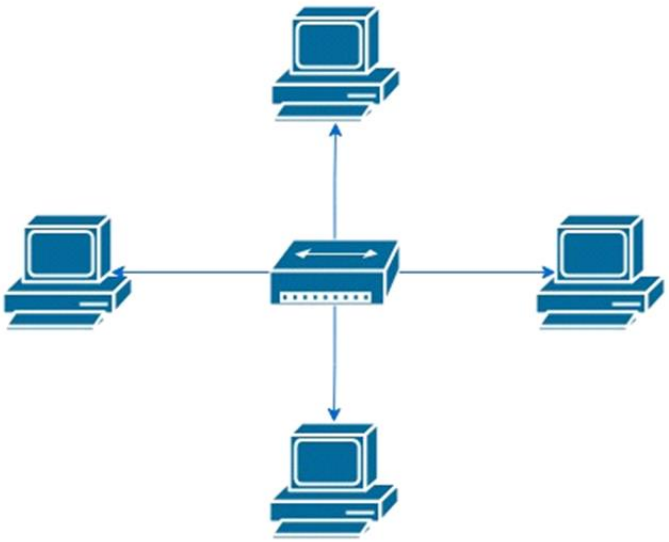
Na infraestrutura de comunicação pretendemos usar arquitetura de rede em estrela (Figura 13), uma configuração consagrada no domínio da informática por suas múltiplas qualidades.

Esta configuração ostenta atributos de confiabilidade, prontidão na identificação de irregularidades, e fortalecimento da segurança. Entre os preceitos distintivos da topologia em estrela, sobressai a confiabilidade, e esse foi o principal motivo para sua escolha. Em situações de falha de um dispositivo ou de um cabo, o funcionamento dos demais dispositivos da rede permanece íntegro, resultado da conexão direta de cada dispositivo ao ponto central (switch).

A configuração descomplicada capacita os administradores de rede a prontamente resolver problemas em dispositivos individuais. Adicionalmente, a implementação de salvaguardas de segurança se torna uma tarefa mais simples, dada a centralização do controle de acesso à rede no ponto central.

“A topologia em estrela é uma abordagem elegante para conectar vários dispositivos em uma rede. Ela proporciona uma estrutura centralizada que simplifica a administração e facilita a detecção e correção de falhas." - Leonard Kleinrock.

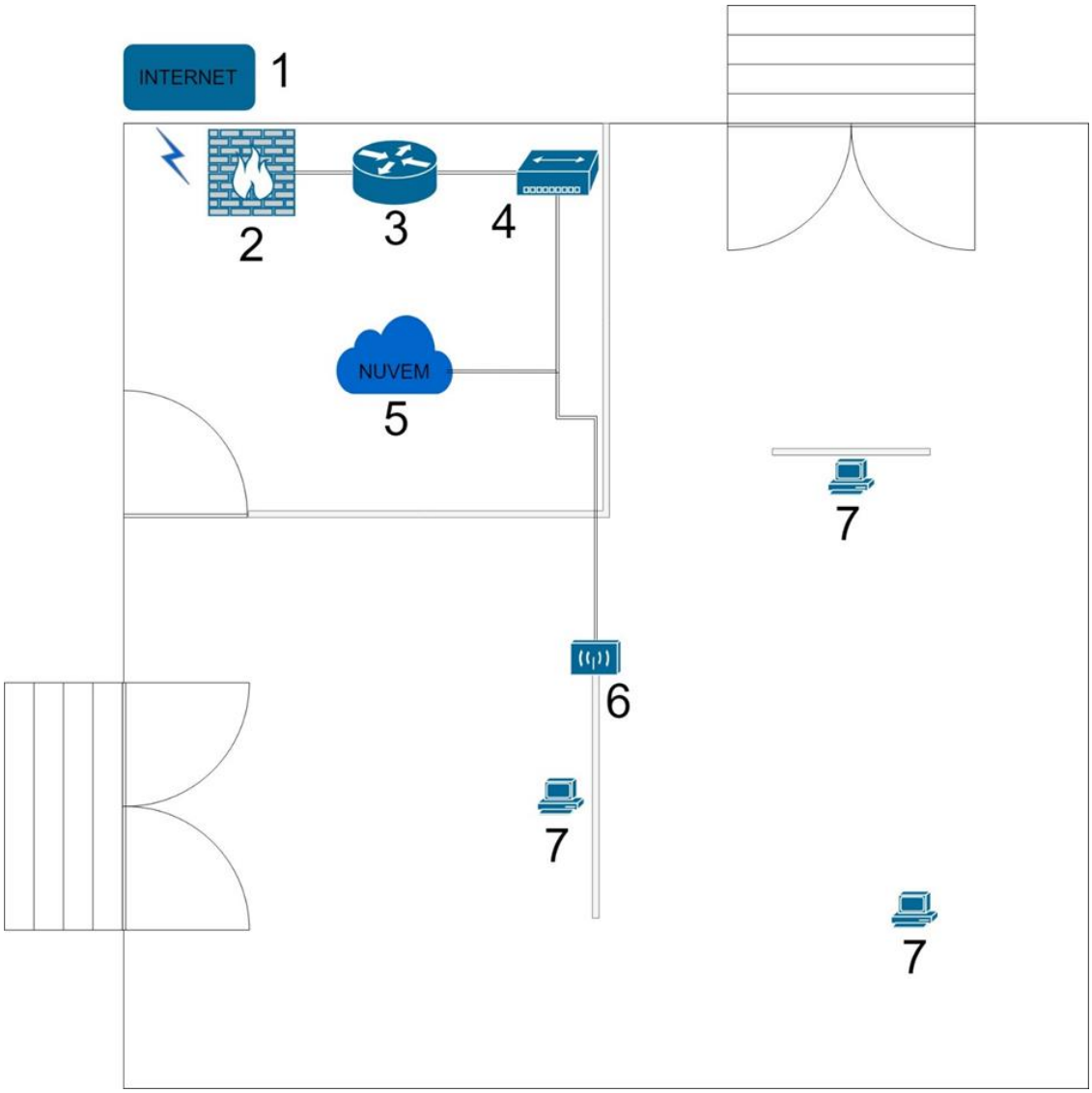
Figura 13 Rede estrela



Fonte: Elaborado pelo autor

A Figura 14 ilustra uma topologia, mostrando a disposição dos equipamentos no ambiente e suas respectivas conexões. A Tabela 4, apresentada abaixo, lista todos os dispositivos juntamente com seus endereços IP individuais ou faixas de IPs atribuídas.

Figura 14 Topologia Museu



Fonte: Elaborado pelo autor

Tabela 4 Legenda topologia

Número	Item	Ranges de ips
1	Internet	-
2	Firewall	192.168.1.1
3	Router	192.168.1.2
4	Switch	-
5	Nuvem	-
6	Access Point	192.168.1.3
7	Totens	192.168.1.4 - 192.168.1.6

Fonte: Elaborado pelo autor

Em equipamentos as recomendações técnicas são:

- Para o router é necessário um router domistico com uma interface WAN gigabit e múltiplas interfaces LAN gigabit, NAT, firewall básico e Wi-Fi integrado
- O switch gerenciável de 5 portas gigabit para conectar todos os dispositivos da rede e sobrar para futuras necessidades ou emergências.
- Os access points de baixo custo com suporte a ethernet gigabit (Gbps) para suas conexões e Wi-Fi 802.11n.
- Para os totens é necessário o mínimo para rodar o software no Windows 11 ou seja: Processadores dual core 1.5GHz ou superior, 4Gb de memória ou superior, sistema de armazenamento por hd ou ssd, placa de rede sem fio e porta USB (Universal Serial Bus).
- O museu utiliza os serviços de armazenamento em nuvem fornecidos pela Amazon Web Services (AWS) devido à sua confiabilidade, segurança e variedade de serviços disponíveis.

6.2. Requisitos

6.2.1. Funcionais

Rede: Cobertura Wi-Fi Total: Garante cobertura Wi-Fi nas áreas das exposições onde os totens estarão instalados.

Conectividade Estável: Oferece conexões estáveis e de alta velocidade para todos os dispositivos conectados.

Gerenciamento Centralizado: Utiliza um controlador de rede para gerenciar todos os APs de forma centralizada.

Armazenamento na Nuvem: Segurança dos Dados: Implementar criptografia e controle de acesso para proteger os dados armazenados

6.2.2. Não funcionais

Para garantir que tudo corra bem temos nos requisitos não funcionais:

Sistema Operacional é um programa como os demais que roda em máquinas. Porém ele é um dos programas cujo diferencial é sua complexidade. O Sistema Operacional é responsável pelo gerenciamento de todo o hardware do computador e as aplicações do usuário, com o propósito de criar um ambiente que permita a execução adequada, ou seja, simples e eficiente ao mesmo tempo. O Sistema Operacional é um grande maestro, pois ele é responsável por designar os recursos do hardware. Como por exemplo: tempo de CPU, memória, dispositivos de I/O (Entrada/Saída) entre outros. É o Sistema Operacional que vai determinar qual recurso será utilizado, qual aplicação e por quanto tempo.

"Lidando com o Windows, é bom lembrar que o conhecimento é poder. Quanto mais você sabe sobre como o sistema operacional funciona, mais controle terá sobre seu computador." (KARP, 2004, p. 45).

O Windows 11 é a mais recente iteração do sistema operacional da Microsoft, lançado com o compromisso de oferecer uma experiência de usuário mais refinada, segura e produtiva introduzindo uma série de atualizações visuais e funcionais. Este sistema operacional da Microsoft representa uma plataforma sólida e moderna para a execução de aplicativos e serviços com alta compatibilidade. Com sua interface

aprimorada e recursos avançados, o Windows 11 será o ambiente onde o software será utilizado.

Uma alta disponibilidade garante que a rede esteja disponível 99,9% do tempo, minimizando o tempo de inatividade para garantir o acesso contínuo dos visitantes e funcionários à rede.

Implementação de medidas de segurança para proteger os dados do museu contra acessos não autorizados, incluindo criptografia, autenticação forte e políticas de acesso baseadas em função.

Aplicação de ferramentas de gerenciamento intuitivas e centralizadas facilitando o gerenciamento da rede, permitindo a configuração, monitoramento e solução de problemas de forma eficiente.

6.2.3. Custos

O custo total estimado para a infraestrutura de rede do museu, incluindo equipamentos de rede, serviços em nuvem, licenças de software e outros custos, é de aproximadamente R$ 8.900,00. Este valor pode variar dependendo da quantidade de totens e serviços em nuvem utilizados, bem como custos de instalação e configuração adicionais.

7. FERRAMENTAS UTILIZADAS

7.1. C#

C# é uma linguagem de programação moderna, orientada a objetos e desenvolvida pela Microsoft. Lançada em 2000, ela foi projetada para ser simples, segura, e eficiente, combinando elementos das linguagens C++ e Java. C# é amplamente utilizada para o desenvolvimento de aplicativos de desktop, jogos, aplicativos web e móveis, além de ser uma linguagem chave para o desenvolvimento na plataforma .NET da Microsoft. O uso do C# garantiu uma base sólida para o desenvolvimento do software.

7.2. Visual Studio

O Visual Studio é um ambiente de desenvolvimento integrado (IDE) criado pela Microsoft, amplamente utilizado para desenvolver aplicativos de desktop, web, móveis e jogos. Suporta várias linguagens de programação, incluindo C#, Visual Basic, C++ e Python. Entre suas principais características estão um editor de código com destaque de sintaxe e autocompletar, um depurador, ferramentas de refatoração e navegação avançada, além de um designer de interface gráfica (Windows Forms Designer) que permite criar interfaces visuais arrastando e soltando.

No desenvolvimento do software foi utilizado o C# e Windows Forms, o Visual Studio foi utilizado para criar a interface gráfica do usuário de forma visual, permitindo a adição e configuração de elementos como botões, caixas de texto, labels e checkboxs diretamente no Windows Forms Designer. O editor de código e o depurador ajudaram no desenvolvimento e na correção do código, facilitando a implementação da lógica do aplicativo e a resolução de bugs. O gerenciamento de projetos do Visual Studio ajudou a organizar os arquivos e dependências do projeto, garantindo um fluxo de trabalho eficiente.

7.3. Github

O GitHub é uma plataforma de hospedagem de código-fonte e colaboração baseada em nuvem, usada principalmente para controle de versão e colaboração em projetos de desenvolvimento de software. Permite que os desenvolvedores armazenem, compartilhem e colaborem em projetos usando o sistema de controle de versão Git. Os recursos incluem controle de acesso flexível, rastreamento de problemas, integração contínua e implantação automatizada, bem como ferramentas para revisão de código e colaboração em equipe.

O GitHub desempenhou um papel fundamental no desenvolvimento do software, destacando-se especialmente por sua capacidade de promover o compartilhamento e a colaboração no projeto.

7.4. BRModelo

O BRModelo é uma ferramenta de modelagem de dados e de processos desenvolvida especialmente para auxiliar profissionais na criação de modelos conceituais, lógicos e físicos de bancos de dados. Com uma interface intuitiva e recursos avançados, o BRModelo permite aos usuários visualizarem e projetar estruturas de banco de dados de forma eficiente e organizada.

O software contou com a utilização do BRModelo para realizar a modelagem tanto do Modelo Entidade-Relacionamento (MER) quanto do Diagrama Entidade-Relacionamento (DER).

7.5. StarUML

O StarUML é uma ferramenta de modelagem UML (Unified Modeling Language) poderosa e versátil, amplamente empregada no desenvolvimento de software. Oferecendo uma variedade de recursos e uma interface amigável, o StarUML permite a criação de diversos tipos de diagramas, como diagramas de

classes, sequência e atividade. Sua flexibilidade e capacidade de extensão o tornam uma escolha popular entre desenvolvedores e equipes de projeto.

No contexto específico, foi utilizado o StarUML para a criação dos diagramas necessários, garantindo uma representação clara e precisa dos requisitos e da arquitetura do sistema em desenvolvimento. Através desses diagramas, os diferentes aspectos do sistema puderam ser visualizados e compreendidos, facilitando a comunicação entre os membros da equipe e auxiliando no planejamento e implementação do software.

7.6. Balsamiq

O Balsamiq é uma ferramenta de prototipagem de interface de usuário que permite aos designers criarem esboços de forma rápida e fácil, simulando a aparência e a interação de um aplicativo ou site. Com uma interface simples e intuitiva, o Balsamiq oferece uma variedade de elementos de design pré-fabricados que podem ser arrastados e soltos para criar protótipos de alta fidelidade.

No contexto específico, o Balsamiq foi utilizado para a prototipação de três telas-chave do projeto, proporcionando uma representação visual e interativa das interfaces do usuário planejadas.

7.7. Draw.io

O Draw.io é uma ferramenta de diagramação online que permite aos usuários criarem uma ampla variedade de diagramas, como fluxogramas, organogramas, mapas mentais, diagramas de rede, entre outros. Ele oferece uma interface intuitiva e fácil de usar, com uma ampla gama de formas, ícones e ferramentas de edição para personalizar os diagramas conforme necessário. O Draw.io é totalmente baseado na web e pode ser acessado em navegadores modernos sem a necessidade de download ou instalação de software adicional.

7.8. Trello

O Trello é uma ferramenta de gerenciamento de projetos baseada na web que utiliza o método Kanban para ajudar equipes a organizar tarefas e fluxos de trabalho. Lançado em 2011 pela empresa Fog Creek Software, e posteriormente adquirido pela Atlassian, o Trello permite aos usuários criarem quadros, listas e cartões para representar diferentes etapas e atividades de um projeto. Cada cartão pode ser personalizado com descrições, checklists, prazos e anexos, proporcionando uma visão clara e detalhada das tarefas.

Uma das principais vantagens do Trello é sua interface intuitiva e fácil de usar, que facilita a colaboração entre membros da equipe, mesmo em ambientes remotos. Segundo Trello (2021), a ferramenta é utilizada por milhões de pessoas ao redor do mundo em diversas indústrias, desde desenvolvimento de software até planejamento de eventos. O Trello oferece integrações com outras ferramentas populares, como Slack, Google Drive e Jira, ampliando suas funcionalidades e tornando-o uma solução versátil para a gestão de projetos. Além disso, com funcionalidades como automação de tarefas repetitivas através do Butler, o Trello aumenta a eficiência e permite que as equipes se concentrem em atividades de maior valor.

CONCLUSÃO

O desenvolvimento de um software para avaliação de obras de arte em museus, utilizando totens interativos, mostrou-se uma solução eficaz para melhorar a gestão de exposições e a experiência dos visitantes. Este projeto, embasado em metodologias ágeis como Scrum e Kanban, proporcionou uma estrutura eficiente e adaptativa para o desenvolvimento contínuo do sistema. A coleta de feedback dos visitantes em tempo real permitiu aos administradores dos museus obterem dados valiosos sobre as percepções das obras expostas. A análise desses dados forneceu insights fundamentais para ajustes e aprimoramentos das exposições, promovendo uma experiência mais envolvente e satisfatória para o público. A utilização da linguagem C# e da plataforma .NET garantiu a robustez e a escalabilidade necessárias ao software, enquanto a interface digital intuitiva assegurou uma interação amigável com os visitantes.

Além disso, a implementação de medidas de segurança para proteger a privacidade dos dados dos visitantes reforçou a confiabilidade do sistema. O projeto não apenas atendeu às necessidades dos museus em obter feedback detalhado e imediato, mas também contribuiu para a valorização da cultura e da arte, oferecendo aos visitantes uma plataforma para expressarem suas opiniões de maneira eficaz.

Em suma, o software desenvolvido não só resolveu um problema significativo enfrentado pelos museus, mas também estabeleceu um novo padrão para a gestão de exposições, demonstrando a importância da integração entre tecnologia e cultura na era digital.

REFERENCIAS BIBLIOGRAFICAS

KLEINROCK, Leonard. Redes de Computadores: Princípios e Práticas. 1. ed. São Paulo: Editora ABC, 2020.

KARP, David A. Windows XP Annoyances for Geeks. 2. ed. Sebastopol: O'Reilly Media, 2004.

ANDERSON, D. J. Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2010.

KNIBERG, H.; SKARIN, M. Kanban and Scrum - Making the Most of Both. C4Media, 2010.

TRELLO. About Trello. Disponível em: <https://trello.com/about>. Acesso em: 25 maio 2024.

GRIFFITHS, I. Programming C# 8.0: Build Cloud, Web, and Desktop Applications. O'Reilly Media, 2019.

FERNANDES, J. C. UX Design - Desenho de Interfaces.

COMER, D. Redes de Computadores e Internet. 6. ed.

TANENBAUM, A. S. Redes de Computadores.

RATHBONE, A. Windows 10 Para Leigos.

APÊNDICE A

MANUAL DE UTILIZAÇÃO

1. Passo: Inserção do Nome
    - Ao abrir o software, você será recebido pela tela inicial. Insira seu primeiro nome no campo designado.
    - Utilize o teclado virtual fornecido na tela para inserir seu nome com facilidade.
    - Após inserir seu primeiro nome, você poderá visualizar a classificação média dada por outros visitantes para as obras de arte que serão exibidas.
    - Clique no botão "Próximo" para confirmar e prosseguir para a visualização das obras.
2. Passo: Visualização e Feedback das Obras
    - Na tela seguinte, você verá a primeira obra disponível, descrição, foto e questionários.
    - Leia atentamente a descrição da obra e, em seguida, realize seu feedback respondendo aos 5 questionários fornecidos, com feedbacks de 0 a 5.
    - Após responder aos questionários, você pode prosseguir para visualizar outras obras clicando no botão "Próximo".
3. Passo: Visualização de Mais Obras
    - Prossiga para visualizar mais obras seguindo o mesmo processo: leia a descrição, responda aos questionários e clique em "Próximo" para avançar.
4. Passo: Finalização e Visualização dos Feedbacks
    - Após interagir com todas as obras disponíveis, o software irá compilar os feedbacks fornecidos por você.
    - Na tela final, você poderá visualizar seu próprio feedback fornecido a cada obra.

APÊNDICE B

Codigo do Software

1.1 ListaDeFeedback

```
public class ListaDeFeedback
{
   private static ListaDeFeedback instance;

   private List<(int id, int media)> feedbackList;

   private ListaDeFeedback()
   {
      feedbackList = new List<(int id, int media)>();
   }

   public static ListaDeFeedback Instance
   {
      get
      {
         if (instance == null)
         {
            instance = new ListaDeFeedback();
         }
         return instance;
      }
   }

   public List<(int id, int media)> FeedbackList
   {
      get { return feedbackList; }
   }

   // Método para adicionar feedback à lista
   public void AdicionarFeedback(int id, int media)
   {
      feedbackList.Add((id, media));
   }
}
```

1.2 Teclado

```
namespace PIM_3º_SEMESTRE.Model_fmr_Home
{
   internal class Teclado
   {
      public void Inserir(object sender, EventArgs e, TextBox txb, CheckBox chk )
      {
         Button btn = (Button)sender;
         CheckBox chkCapsLock = chk;

         if (chkCapsLock.Checked == true)
         {
            txb.Text = txb.Text + btn.Text.ToUpper();
         }
         else
         {
            txb.Text = txb.Text + btn.Text.ToLower();
         }
      }

      public void btnBackspace_Click(object sender, EventArgs e, TextBox txb)
      {
         txb.Text = string.Empty;
      }

      public void btnEspaço_Click(object sender, TextBox txb)
      {
         txb.Text = txb.Text + " ";
```

```
        }

    }
}
```

1.3 ValidaNome

```
namespace PIM_3º_SEMESTRE.Model_fmr_Home
{
    internal class ValidaNome
    {

        public string validaNome(string txtNome)
        {
            Regex regex = new Regex(@"^[A-Za-z\s]+$");
            if (regex.IsMatch(txtNome))
            {

                return "";
            }
            else
            {
                return "O texto não é um nome válido. Por favor, insira apenas letras e espaços.";
            }
        }
    }
}
```

1.4 AbsPropriedades

```
namespace PIM_3º_SEMESTRE.Model_fmr_Principal
{
    public abstract class AbsPropriedades
    {
        protected Form formulario;
        protected Label lblNome;
        protected Label lblDescricao;
        protected PictureBox picImagem;
        protected Label[] lblPerguntas;
        protected TextBox[] txtRespostas;
        protected Label lblMensagem;

        protected void InicializarElementos(Form formulario)
        {
            this.formulario = formulario;

            lblNome = formulario.Controls.Find("lblNome", true)[0] as Label;
            lblDescricao = formulario.Controls.Find("lblDescricao", true)[0] as Label;
            picImagem = formulario.Controls.Find("picImagem", true)[0] as PictureBox;
            lblMensagem = formulario.Controls.Find("lblMensagem", true)[0] as Label;

            lblPerguntas = new Label[5];
            txtRespostas = new TextBox[5];

            for (int i = 0; i < 5; i++)
            {
                lblPerguntas[i] = formulario.Controls.Find("lblPergunta" + (i + 1), true)[0] as Label;
                txtRespostas[i] = formulario.Controls.Find("txtPergunta" + (i + 1), true)[0] as TextBox;
            }
        }

    }
}
```

1.5 ExibirObras

```
namespace PIM_3º_SEMESTRE.Model_fmr_Principal
{
    internal class ExibirObras : AbsPropriedades
    {
        private List<Item> listaDeObras;
        private List<(int id, int media)> listaFeedbackLocal;
```

```
        private int currentIndex = 0;
        private Validacao validacao = new Validacao();

        private Button btnProximo;
        private Button btnFinalizar;

        public ExibirObras(Form formulario, Button btnProximo, Button btnFinalizar)
        {
            this.btnProximo = btnProximo;
            this.btnFinalizar = btnFinalizar;

            ListaDeObras listaDeObras = new ListaDeObras();
            this.listaDeObras = listaDeObras.CriarListaDeObras();

            listaFeedbackLocal = new List<(int id, int media)>();

            InicializarElementos(formulario);
            Exibir(currentIndex);
            AtualizarVisibilidadeBotoes();
        }


        private void Exibir(int index)
        {
            if (index >= 0 && index < listaDeObras.Count)
            {
                ExibirDadosObra(listaDeObras[index]);
            }
            else { MessageBox.Show(""); }
        }

        private void ExibirDadosObra(Item obra)
        {
            lblMensagem.Text = "";
            lblNome.Text = obra.Nome;
            lblDescricao.Text = "Descrição: " + obra.Descricao;
            picImagem.Image = Image.FromFile($"C:\\Users\\mvc11\\OneDrive\\Área de Trabalho\\PIM 3º
SEMESTRE\\SOFTWARE\\PIM 3º SEMESTRE\\Images\\{obra.Imagem}");

            for (int i = 0; i < 5; i++)
            {
                if (i < obra.Perguntas.Count)
                {
                    lblPerguntas[i].Text = obra.Perguntas[i];
                    txtRespostas[i].Text = "";
                    lblPerguntas[i].Visible = true;
                    txtRespostas[i].Visible = true;
                }
                else
                {
                    lblPerguntas[i].Visible = false;
                    txtRespostas[i].Visible = false;
                }
            }
        }

        // abaixo os eventos dos botoes
        private void AtualizarVisibilidadeBotoes()
        {
            btnProximo.Visible = (currentIndex < listaDeObras.Count - 1);
            btnFinalizar.Visible = (currentIndex == listaDeObras.Count - 1);
        }

        public void btnProximo_Click(object sender, EventArgs e)
        {
            if (validacao.ValidarTextBoxes(txtRespostas) == "")
            {
                int obraId = listaDeObras[currentIndex].Id;
                List<int> respostas = txtRespostas.Select(txt => int.Parse(txt.Text)).ToList();
                Salvar(obraId, respostas);

                double media = respostas.Average();
                listaFeedbackLocal.Add((obraId, (int)Math.Round(media)));

                if (currentIndex < listaDeObras.Count - 1)
```

```
        {
            currentIndex++;
            Exibir(currentIndex);
            AtualizarVisibilidadeBotoes();
        }
    }
    else
    {
        lblMensagem.Text = validacao.ValidarTextBoxes(txtRespostas);
    }
}

public void btnFinalizar_Click(object sender, EventArgs e, string nome)
{
    if (validacao.ValidarTextBoxes(txtRespostas) == "")
    {
        int obraId = listaDeObras[currentIndex].Id;
        List<int> respostas = txtRespostas.Select(txt => int.Parse(txt.Text)).ToList();
        Salvar(obraId, respostas);

        double media = respostas.Average();
        listaFeedbackLocal.Add((obraId, (int)Math.Round(media)));

        formulario.Close();
        Thread telaObra = new Thread(() => abrirTelaObras(nome, listaFeedbackLocal));
        telaObra.SetApartmentState(ApartmentState.STA);
        telaObra.Start();
    }
    else
    {
        lblMensagem.Text = validacao.ValidarTextBoxes(txtRespostas);
    }
}

private static void abrirTelaObras(string nome, List<(int id, int media)> listaFeedbackLocal)
{
    Application.Run(new frm_Feedback(nome, listaFeedbackLocal));
}

private void Salvar(int id, List<int> respostas)
{
    double media = respostas.Average();
    int mediaInt = Convert.ToInt32(Math.Round(media));
    ListaDeFeedback.Instance.AdicionarFeedback(id, mediaInt);
}

public void Pergunta1(object sender, EventArgs e, TextBox txb)
{
    Button btn = (Button)sender;
    string tecla = btn.Text;
    txb.Text = tecla;
    btn.Focus();
}
public void Pergunta2(object sender, EventArgs e, TextBox txb)
{
    Button btn = (Button)sender;
    string tecla = btn.Text;
    txb.Text = tecla;
    btn.Focus();
}
public void Pergunta3(object sender, EventArgs e, TextBox txb)
{
    Button btn = (Button)sender;
    string tecla = btn.Text;
    txb.Text = tecla;
    btn.Focus();
}
public void Pergunta4(object sender, EventArgs e, TextBox txb)
{
    Button btn = (Button)sender;
    string tecla = btn.Text;
    txb.Text = tecla;
    btn.Focus();
}
public void Pergunta5(object sender, EventArgs e, TextBox txb)
{
```

```
            Button btn = (Button)sender;
            string tecla = btn.Text;
            txb.Text = tecla;
            btn.Focus();
        }

    }
}
```

1.6 Validacao

```
namespace PIM_3º_SEMESTRE.Model_fmr_Principal
{
    public class Validacao
    {

        public string ValidarTextBoxes(TextBox[] textBoxes)
        {
            for (int i = 0; i < textBoxes.Length; i++)
            {
                if (string.IsNullOrWhiteSpace(textBoxes[i].Text))
                {
                    return $"Por favor, preencha todos os campos!";
                }
                else
                {
                    if (!int.TryParse(textBoxes[i].Text, out int resposta))
                    {
                        return $"Por favor, insira apenas números inteiros!";
                    }
                    else
                    {
                        if (resposta < 0 || resposta > 5)
                        {
                            return $"Por favor, insira valores entre 0 e 5!";
                        }
                    }
                }
            }
            return "";
        }
    }

}
```

1.7 frm_Home

```
namespace PIM_3º_SEMESTRE
{
    public partial class frm_Home : Form
    {
        ValidaNome validar = new ValidaNome();
        Teclado teclado = new Teclado();

        string nomeValidado;
        public frm_Home()
        {
            InitializeComponent();

            List<(int id, int media)> feedbacks = ListaDeFeedback.Instance.FeedbackList;

            var feedbacksPorID = feedbacks.GroupBy(f => f.id);

            foreach (var grupo in feedbacksPorID)
            {
                double mediaGeral = grupo.Select(f => f.media).Average();
                int mediaInteira = (int)Math.Round(mediaGeral);

                string nomeLabel = $"lblId{grupo.Key}";
                Label? label = Controls.Find(nomeLabel, true).FirstOrDefault() as Label;
                label.Text = $"O Feedback atual é de: {mediaInteira:F2}";

                PictureBox star = Controls.Find($"star{grupo.Key}", true).FirstOrDefault() as PictureBox;
```

```
            star.Image = Properties.Resources.ResourceManager.GetObject($"estrela{mediaInteira}") as Image;

        }
    }

    private void button1_Click(object sender, EventArgs e)
    {
        string validacao = validar.validaNome(txtNome.Text);

        if (validacao.Equals(""))
        {
            nomeValidado = txtNome.Text;

            this.Close();
            Thread telaObra = new Thread(() => abrirTelaObras(nomeValidado));
            telaObra.SetApartmentState(ApartmentState.STA);
            telaObra.Start();
        }
        else
        {
            lblResposta.Text = "Nome invalido! Tente novamente.";
        }
    }
    private static void abrirTelaObras(string nome)
    {
        Application.Run(new frm_Principal(nome));
    }
    private void pictureBox2_Click(object sender, EventArgs e)
    {

    }

    private void Alfabeto(object sender, EventArgs e)
    {
        teclado.Inserir(sender, e, txtNome, chkCapsLock);
    }

    private void btnBackspace_Click(object sender, EventArgs e)
    {
        teclado.btnBackspace_Click(sender, e, txtNome);
    }

    private void btnEspaço_Click(object sender, EventArgs e)
    {
        teclado.btnEspaço_Click(sender, txtNome);
    }

  }
}
```

1.8. frm_Principal

```
namespace PIM_3º_SEMESTRE
{
  public partial class frm_Principal : Form
  {
    private ExibirObras exibirObras;

    string nomeLocal;
    public frm_Principal(string nome)
    {
        InitializeComponent();
        nomeLocal = nome;
        exibirObras = new ExibirObras(this, btnProximo, btnFinalizar);
    }

    private void btnProximo_Click(object sender, EventArgs e)
    {
        exibirObras.btnProximo_Click(sender, e);
    }

    private void btnFinalizar_Click(object sender, EventArgs e)
```

```
        {
            exibirObras.btnFinalizar_Click(sender, e, nomeLocal);
        }

        private void txtPergunta1_TextChanged(object sender, EventArgs e)
        {

        }
        private void Pergunta1(object sender, EventArgs e)
        {
            exibirObras.Pergunta1(sender, e, txtPergunta1);
        }
        private void Pergunta2(object sender, EventArgs e)
        {
            exibirObras.Pergunta2(sender, e, txtPergunta2);
        }
        private void Pergunta3(object sender, EventArgs e)
        {
            exibirObras.Pergunta3(sender, e, txtPergunta3);
        }
        private void Pergunta4(object sender, EventArgs e)
        {
            exibirObras.Pergunta4(sender, e, txtPergunta4);
        }
        private void Pergunta5(object sender, EventArgs e)
        {
            exibirObras.Pergunta5(sender, e, txtPergunta5);
        }

        private void lblNome_Click(object sender, EventArgs e)
        {

        }
    }
}
```

1.9 frm_Feedback

```
namespace PIM_3º_SEMESTRE
{
    public partial class frm_Feedback : Form
    {
        Thread telaPrincipal;
        string nomeLocal;
        List<(int id, int media)> feedbacksLocal;

        public frm_Feedback(string nome, List<(int id, int media)> feedbacks)
        {
            InitializeComponent();
            nomeLocal = nome;
            feedbacksLocal = feedbacks;
        }

        private void frm_Feedback_Load(object sender, EventArgs e)
        {
            lblNome.Text = "Olá! " + nomeLocal + " este foi seu Feedback para as obras";

            foreach (var feedback in feedbacksLocal)
            {
                string nomeLabel = $"lblId{feedback.id}";
                Label label = Controls.Find(nomeLabel, true).FirstOrDefault() as Label;


                label.Text = $"Seu Feedback para esta obra foi de: {feedback.media}";


                string nomeStar = $"estrela{feedback.id}";
                PictureBox star = Controls.Find(nomeStar, true).FirstOrDefault() as PictureBox;
                star.Image = Properties.Resources.ResourceManager.GetObject($"estrela{feedback.media}") as Image;
            }
        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
            telaPrincipal = new Thread(abrirTelaPrincipal);
            telaPrincipal.SetApartmentState(ApartmentState.STA);
            telaPrincipal.Start();
        }

        private void abrirTelaPrincipal()
        {
            Application.Run(new frm_Home());
        }
    }
}
```