

GUSTAVO MATIAS LIRA
JULIA STEFANY MEDEIROS DA SILVA
MATHEUS HENRIQUE LOPES CUTSCHERA
MURILO ANTUNES DE OLIVEIRA
NICOLAS ALVES FACCINELLI
WILGNER KAUE DA SILVA CASTRO

PIM:
TOTEM MUSEU

SOROCABA - SP
2024

GUSTAVO MATIAS LIRA
JULIA STEFANY MEDEIROS DA SILVA
MATHEUS HENRIQUE LOPES CUTSCHERA
MURILO ANTUNES DE OLIVEIRA
NICOLAS ALVES FACCINELLI
WILGNER KAUE DA SILVA CASTRO

PIM:
TOTEM MUSEU

Trabalho de conclusão de
curso para obtenção do título
de graduação em (análise e
desenvolvimento de
sistemas) apresentado à
Universidade Paulista – UNIP.

Orientador: Prof. WALDIR SILVA

SOROCABA – SP
2024

GUSTAVO MATIAS LIRA
JULIA STEFANY MEDEIROS DA SILVA
MATHEUS HENRIQUE LOPES CUTSCHERA
MURILO ANTUNES DE OLIVEIRA
NICOLAS ALVES FACCINELLI
WILGNER KAUE DA SILVA CASTRO

PIM:
TOTEM MUSEU

Trabalho de Conclusão de Curso
para obtenção do título de
Graduação em (análise e
desenvolvimento de sistemas)
apresentado à Universidade
Paulista – UNIP.

Aprovado em:

BANCA EXAMINADORA

_____/ / _____

Prof. Nome do Professor Universidade Paulista – UNIP

_____/ / _____

Prof. Nome do Professor Universidade Paulista – UNIP

_____/ / _____

Prof. Nome do Professor Universidade Paulista – UNIP

RESUMO

Esta documentação detalha o desenvolvimento e implantação de um software interativo para o Museu Espacial, que inclui interativos e um sistema de feedback dos visitantes. O software foi projetado para melhorar a experiência dos visitantes, proporcionando informações em tempo real e atividades interativas que aumentam o engajamento.

Palavras-chave: Software, Site, Museu Espacial.

ABSTRACT

This documentation details the development and implementation of interactive software for the Space Museum, which includes an information totem, a digital map of the museum, interactive questionnaires and a visitor feedback system. The software is designed to improve the visitor experience by providing real-time information and interactive activities that increase engagement.

Keywords: Software, Space Museum.

Sumário

1.INTRODUÇÃO	9
2.MOTIVAÇÃO.....	10
3.OBJETIVO DO SISTEMA	11
4.REQUISITOS	12
4.1 Levantamento de Requisitos.....	12
4.2 Metodologia.....	13
4.3 Critérios de Qualidade.....	13
4.4 Infraestrutura	14
4.5 Requisitos Funcionais Desktop	19
4.6 Requisitos não-funcionais Desktop	21
4.7 Requisitos funcionais web	24
4.8 Requisitos não funcionais WEB	26
4.9 Análise de Requisitos	31
4.10 Análise de Ambiente.....	31
4.11 Análise de Software	31
4.12 Análise de hardware	32
4.13 Análise do Usuário	32
4.14 Gerenciamento de Implantação do Sistema.....	32
4.15 Implantação	32
5 Arquitetura do Sistema	33
5.1 Modelagem de Dados.....	33
5.2 Integração do Banco de Dados.....	33
5.3 Desenvolvimento de Funcionalidades	34
6 INTERFACE DO USUARIO (UI)	34
6.1 Experiência do Usuário (UX)	35
7.TESTE DO SOFTWARES	37

7.1 Teste	38
7.1.2 Codificações testes (web).....	39
7.2 Codificações testes (Desktop)	48
8. TREINAMENTO	52
9. PROTOTIPAÇÃO.....	52
10. CUSTO.....	60
11. DIAGRAMAS	61
11.1 Diagrama MER	62
11.2 Diagrama DER.....	62
12. ANÁLISE DE REQUISITOS	63
12.1 Requisitos de Alto Nível	64
12.2 Características do produto.....	64
12.3 Análise de riscos.	65
12.4 Planejamento.....	65
12.5 Identificação.	65
12.6 Análise de risco do museu.	66
12.7 Análise qualitativa de risco.	71
12.8 Análise quantitativa de risco.	71
12.9 Respostas.....	71
12.10 Controle	71
12.11 Mitigação de riscos. (web).....	72
12.12 Plano de continuidade. (desktop)	73
12.13 Proteções Físicas	73
12.14 Manutenção Preventiva	73
12.15 Fontes de Energia	74
12.16 Monitoramento e Vigilância	74
13. DIAGRAMAS DE CLASSES	74
13.1 Diagrama Sequencia cadastro	75

13.2 Diagramas Sequencia Obras	76
13.3 Diagramas Sequencia Quiz	77
13.4 Diagramas Sequencia Feedback	78
13.5 Diagramas Sequencia Administrador	79
13.6 Diagramas de casos uso	80
14. CRONOGRAMAS	81
14.1 Cronogramas primeiro mês.....	81
14.2 Cronogramas segundo mês	82
14.3 Cronogramas terceiro mês.....	83
14.4 Cronogramas quarto mês	84
15. FERRAMENTAS UTILIZADAS.....	85
16. CONCLUSÃO	86
REFERENCIAS BIBLIOGRAFICA.....	87
ANEXO 1 - MANUAL GUIA AO USUARIO WEB.....	89
ANEXO 2 – GUIA DO USUARIO (DESKTOP)	100
ANEXO 3 – DIAGRAMA TOTEM SOFTWARE	110
ANEXO 4 – DIAGRAMA WEB SOFTWARE	111
ANEXO 5 – ALGORITIMOS WEB	112
ANEXO 6 – ALGORITIMOS DESKTOP	180

1.INTRODUÇÃO

O Museu Stay of mars tem como missão proporcionar uma experiência educativa, envolvente e interativa para todos os seus visitantes. Foi identificado a necessidade de implementar um site para o museu. Este site visa não apenas fornecer informações detalhadas sobre as exposições e a história da exploração espacial em marte, mas também oferecer ferramentas interativas que aumentam o engajamento e a satisfação dos visitantes.

O desenvolvimento deste site foi impulsionado pelas ideias dos professores onde foi debatido e discutido em sala. A solução inclui totens interativos, quiz educacionais como uma interação divertida ao usuário onde tem perguntas a respectivas obras e no final apresenta sua porcentagem de acertos, relatórios onde fica salvo as informações do usuário, mas nada comprometedor apenas nome, idade, sobrenome e seu feedback, o feedback é voltado para a avaliação do usuário onde ele pode expressar a sua satisfação. Cada componente foi projetado para melhorar a experiência do visitante, garantindo que cada momento seja informativo e ilustrativo, visando sempre uma interação simples ao usuário para que não ocorra complicações.

2.MOTIVAÇÃO

Desenvolver um site para o Museu Espacial? É tipo o sonho de qualquer nerd do espaço se tornando realidade! A ideia é criar uma arte que vai além de só informar sobre o museu, será dada uma experiência que faça os visitantes se sentirem parte do universo.

Um quiz que desafiam seu conhecimento sobre o espaço e, o melhor de tudo, um jeito fácil dos visitantes darem feedback sobre o que gostaram (ou não) da visita. A ideia é simples fazer com que cada visita ao museu seja uma aventura espacial em si mesma. E quem sabe, talvez até inspire algumas mentes curiosas a seguirem carreiras na exploração espacial.

3.OBJETIVO DO SISTEMA

Esta documentação detalha o desenvolvimento de um site interativo para o Museu Espacial, quiz interativos e um sistema de feedback dos visitantes. O site foi projetado para melhorar a experiência do público interessado proporcionando informações em tempo real e atividades interativas que aumentam o engajamento.

Oferecer uma maneira intuitiva de explorar as obras do museu, facilitando a navegação e maximizando a eficiência do usuário.

Apresenta quiz interativos que desafiam e entretêm os usuários, oferecendo uma forma divertida de aprender sobre temas relacionados ao espaço, ciência e tecnologia. Esses quiz são adaptados para diferentes faixas etárias e níveis de conhecimento. Um sistema de feedback integrado permite que os visitantes compartilhem suas experiências, sugestões e comentários diretamente pelo site. Isso permite ao museu coletar informações valiosas para avaliar e aprimorar continuamente a qualidade de seus serviços e exposições. O objetivo principal deste software é melhorar a experiência geral dos usuários, proporcionando informações em tempo real, atividades interativas e oportunidades de engajamento que enriquecem e ampliam seu entendimento sobre o espaço. Ao integrar tecnologia inovadora com a riqueza do conteúdo do museu, espera-se criar uma experiência memorável e educativa para todos os usuários.

4.REQUISITOS

Após a aula de Projeto de Software do professor, foram realizadas reuniões no Aplicativo Discord fundamentais para analisar e aprofundar os requisitos do software para o Museu Espacial.

Ficou claro que a interatividade do usuário é o aspecto primordial a ser priorizado, pois é essencial para garantir uma experiência envolvente e educativa para os usuários.

Durante essas discussões, foi identificado os requisitos funcionais e não funcionais essenciais para o projeto. Entre os funcionais, destaca - se a necessidade de um site informativo e intuitivo, quiz educativos e um sistema de feedback dos visitantes. Esses recursos são fundamentais para promover a interação e o engajamento dos usuários durante sua visita ao site do museu. Já os requisitos não funcionais, como desempenho, segurança e usabilidade, também foram cuidadosamente considerados. Sendo assim um site eficiente e confiável, é crucial para garantir uma experiência positiva para os usuários, além de manter a integridade das informações e dos dados do museu. Com base nessas análises e priorizações, a equipe está pronta para avançar para a próxima fase do projeto, focando na implementação e no desenvolvimento de um site que atenda às expectativas e necessidades do Museu Espacial e de seus visitantes.

4.1 Levantamento de Requisitos

A aplicação funcionara em ambiente Windows, a obrigação do museu será a compra de host, domínio e segurança, totens, câmeras de segurança, switch e roteadores. Foi levantado um custo de 1000 Reais mensal que o museu terá que pagar para poder manter o site funcionando.

4.2 Metodologia

Segundo um estudo feito pelo “Research, Society and Development”, o método incremental onde aborda uma maneira estruturada e interativa de entregar produtos de qualidade, foi descrito por “Pinto, Serrador J. K. v. 9, n. 3, 2020”.

No desenvolvimento do software para o museu, a equipe utilizou a metodologia ágil incremental seguindo essa pesquisa para garantir um processo de desenvolvimento flexível e eficiente. Essa abordagem permitiu dividir o projeto em incrementos menores e manejáveis, cada um resultando em uma versão funcional do software. A escolha foi motivada pela necessidade de responder rapidamente as mudanças e melhorias oferecidas pelos usuários e stakeholders. A cada incremento foi coletado os feedbacks dos visitantes permitindo ajustes contínuos e melhorias progressivas no software. Como resultado, entregar um software de alta qualidade de maneira iterativa, mantendo a flexibilidade para adaptações e garantindo que cada versão do software atendesse melhor às necessidades do museu e de seus visitantes.

Segundo Barcelos (2012), “de fato, nenhum conceito parece ser mais tradicional ao estudo da orçamentação pública, do que o incrementalismo, o qual até hoje inspira as pesquisas e reflexões desenvolvidas no campo”.

4.3 Critérios de Qualidade

Segundo a (Marques. Brena 6 a 10 de set, 2008) “a qualidade de software é essencial para o usuário alcançar o resultado final almejado com a adoção de determinado aplicativo, independentemente do tipo e do ramo de atuação da organização”.

Foi desenvolvido o software seguindo a norma ISO/IEC 9126, para garantir a qualidade do produto onde oferece um modelo abrangente de qualidade de software cobrindo características como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade, foram aplicados para garantir que o software atendesse aos mais altos padrões de qualidade. A norma ajudou a definir e medir métricas importantes(testes), permitindo uma avaliação contínua e objetiva do desempenho e da qualidade do software com isso garantindo que o software fosse funcional e intuitivo para os visitantes, facilitando sua manutenção e futuras evoluções.

A escolha foi motivada pela necessidade de um software confiável e de alta qualidade, essencial para proporcionar uma excelente experiência aos usuários do museu e garantir uma eficiência operacional. Assim, podendo desenvolver um produto de software que não apenas atende às necessidades atuais do museu, mas que também está preparado para adaptações e melhorias contínuas.

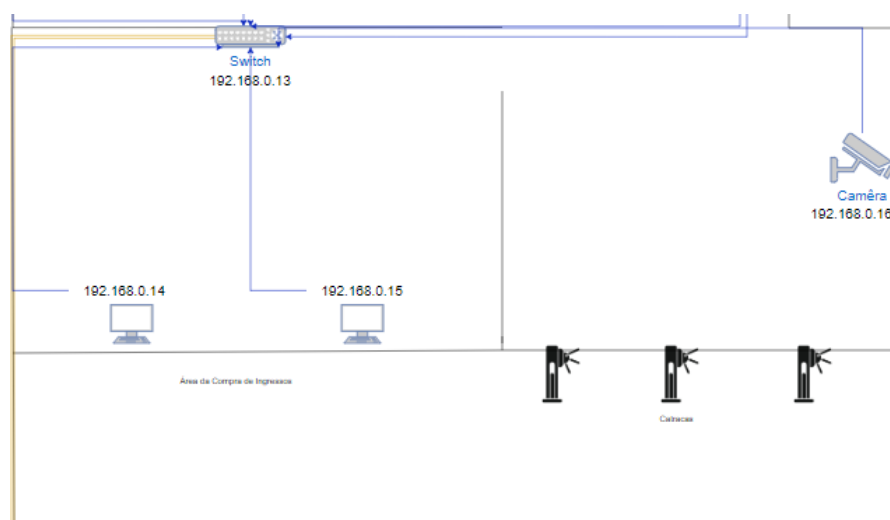
Já na questão segurança o software está seguindo a norma ISO/IEC 27001 para assegurar operações em caso de quebra de equipamentos físicos, como computadores. Esta norma internacional fornece um quadro para garantir a continuidade das operações em caso de danos físicos, por exemplo no caso de um usuário danificar um totem, terá controles e procedimentos de recuperação para garantir que o sistema continue operando sem interrupções significativas, como procedimentos de backup e recuperação de desastres. Envolveu a criação de procedimentos que incluem gestão de riscos, recuperação de desastres e continuidade de negócios. Esses controles são essenciais para assegurar que, mesmo diante de danos físicos aos equipamentos, o software permaneça disponível e funcional mantendo a integridade.

Com isso, conseguindo desenvolver um software que cumpre com os requisitos de segurança e que é resiliente a danos físicos, proporcionando confiança aos usuários e garantindo que as operações do museu não sejam comprometidas por incidentes desse tipo.

4.4 Infraestrutura

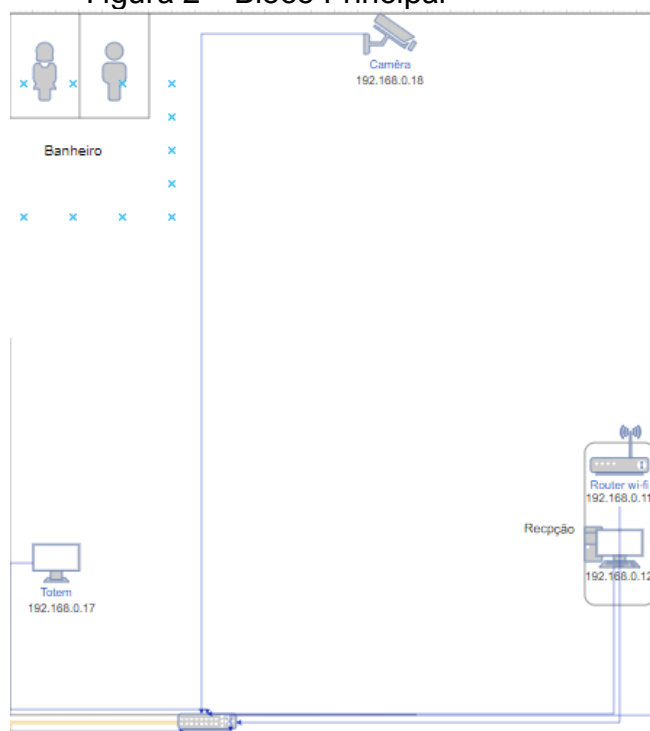
A infraestrutura do museu se distribui por cinco áreas distintas: a entrada, o bloco principal e os dois blocos das atrações, conhecidos como bloco A e B. Além disso, há a sala de T.I. Dentro de cada um desses blocos, há switches, totalizando quatro no total. Esses switches são responsáveis por conectar os totens, as câmeras de segurança e os roteadores wi-fi, formando assim a base da infraestrutura do museu. A seguir, apresentamos imagens detalhadas que ilustram cada parte da topologia, conforme na figura 1, figura 2, figura 3, figura 4, figura 5.

Figura 1 – Entrada museu



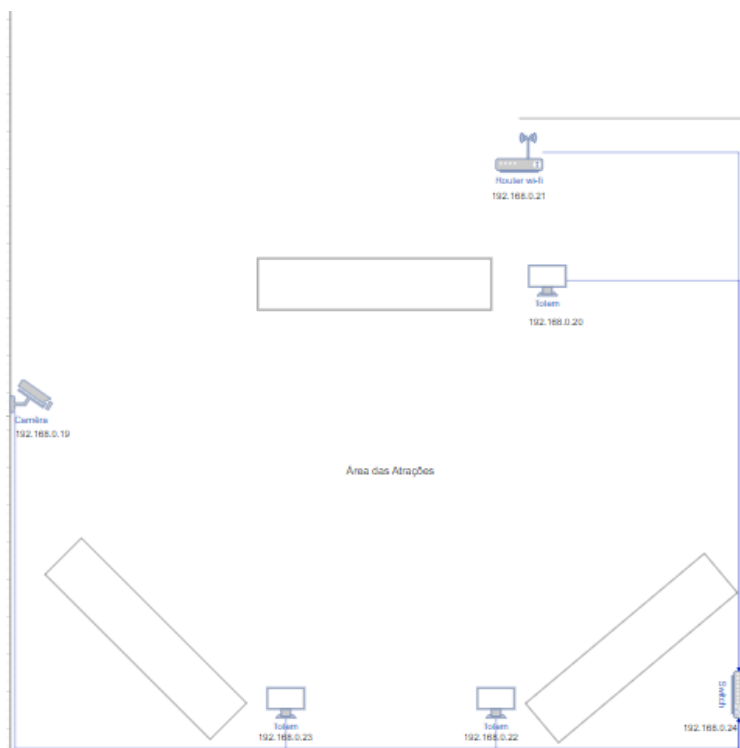
Autor: Autoria própria

Figura 2 – Bloco Principal



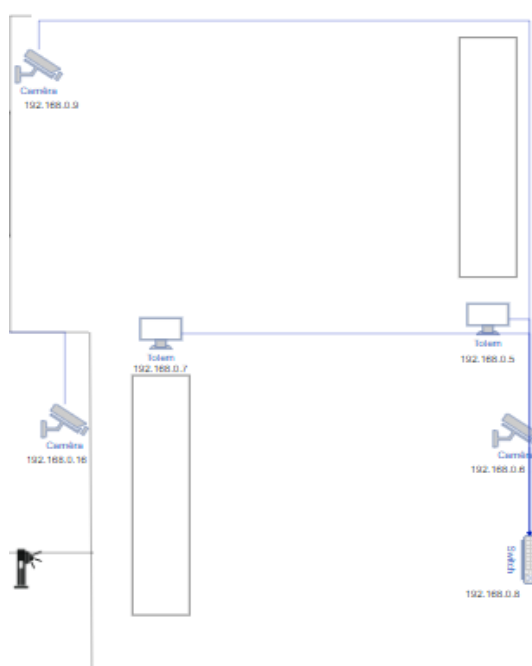
Autor: Autoria própria

Figura 3 – Bloco a



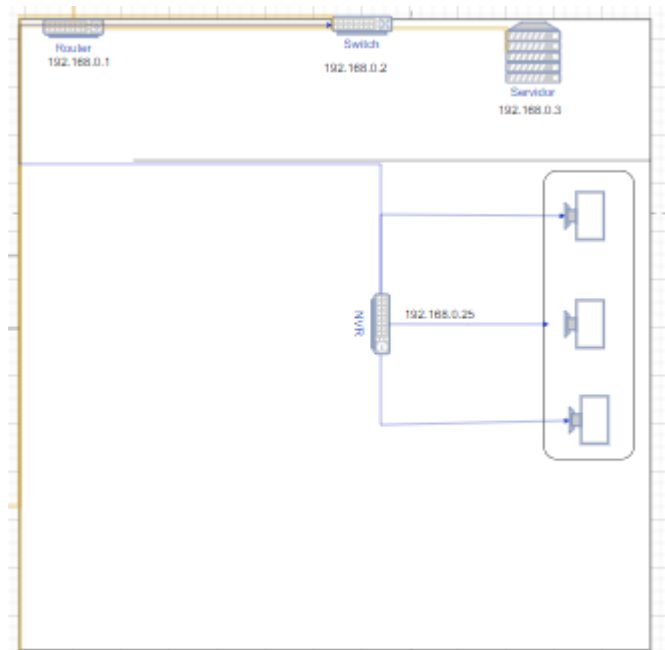
Autor: Autoria própria

Figura 4 – Bloco b



Autor: Autoria própria

Figura 5 – Area de T.I



Autor: Autoria própria

Com base na análise de desempenho do software, para compra dos totens recomendamos os seguintes requisitos de hardware para garantir um funcionamento ideal:

- Processador: Mínimo de 2 GHz.
- Memória RAM: Pelo menos 4 GB.
- Armazenamento: Disco rígido com capacidade mínima de 200 GB.
- Placa de rede: Suporte para 10/100/1000 Mbps ou conexões sem fio.
- Sistema Operacional: Windows 10 ou versão mais recente.
- Conexão com a Internet.
- Monitor Touchscreen para Totem.

Estas especificações são essenciais para assegurar um desempenho fluido e eficiente do software.

4.5 Requisitos Funcionais Desktop

Os requisitos funcionais descrevem o que o sistema deve fazer, ou seja, as funções e características que devem ser implementadas para atender às necessidades dos usuários e aos objetivos do sistema, conforme a tabela abaixo:

Tabela 1 – Requisitos Funcionais

Categoria	Tipo	Requisitos	Descrição
Funcional	Interface	Tela inicial	A tela inicial deve exibir um menu de opções para acessar os feedbacks, quiz, obras, formulários e história das exposições.
Funcional	Feedback	Coleta de feedback	Permitir aos visitantes fornecerem um feedback sobre as exposições e a experiência geral do museu.
Funcional	Feedback	Armazenamento de feedback	Armazena o feedback do visitante no banco de dados.
Funcional	Quiz	Quiz interativo	Oferecer quiz interativos relacionados as exposições para engajar os visitantes.
Funcional	Quiz	Resultado do quiz	Exibir resultados após responder todas as perguntas e apresentar a porcentagem de acerto.
Funcional	Relatórios	Geração de relatórios	Gerar relatório dos feedbacks para o administrador.

Funcional	Obras Espaciais	Exibição de obras especiais	Fornecer informações detalhadas sobre obras espaciais em exibição.
Funcional	Usuario	Cadastro/usuario	O sistema deve permitir que novos usuarios se registrem com informações basicas como nome, email e senha.
Funcional	Usuario	Login	Os usuarios devem fazer login usando suas credencias como login e senha.
Funcional	História das exposições	Informações históricas	Fornecer informações detalhadas e contextuais sobre as exposições.

Autor: Autoria propria

4.6 Requisitos não-funcionais Desktop

Requisitos não funcionais descrevem as qualidades e restrições do sistema que não estão diretamente relacionadas às funções específicas que ele executa, mas que são essenciais para a sua operação eficiente e para proporcionar uma boa experiência ao usuário. Eles especificam os critérios que podem ser usados para julgar a operação de um sistema, ao invés de comportamentos específicos, conforme a tabela abaixo:

Tabela 2 – Requisitos Não Funcionais

Categoria	Tipo	Requisitos	Descrição
Não Funcional	Usabilidade	Intuitividade	O software deve ser fácil de utilizar para visitantes de todas as idades, e níveis de habilidade tecnológica.
Não Funcional	Usabilidade	Interface amigável	A interface deve ser clara e atraente, com menus e botões bem-organizados.
Não Funcional	Usabilidade	Responsividade	O sistema deve ser responsivo, adaptando-se a diferentes tamanhos de tela e dispositivos.

Não Funcional	Desempenho	Tempo de resposta	As operações devem ter um tempo de resposta inferior a 2 segundos.
Não Funcional	Desempenho	Capacidade de carga	O sistema deve suportar vários usuários ao mesmo tempo, sem degradação significativa de desempenho.
Não Funcional	Confiabilidade	Disponibilidade	O software deve estar disponível a todo tempo.

Não Funcional	Confiabilidade	Falha	Em caso de falha o sistema deve se recuperar rapidamente e retomar a operação normalmente sem perda de dados.
Não Funcional	Segurança	Proteção de dados	Dados do usuário guardados em um banco de dados, utilizando criptografia.
Não Funcional	Portabilidade	Compatibilidade com Hardware	O software é Compatível apenas com o Windows.
Não Funcional	Portabilidade	Facilidade de instalação	O software deve ser fácil de instalar e configurar em novos dispositivos.
Não Funcional	Manutenibilidade	Documentação	Deve haver a documentação clara e abrangente para manutenção e atualização do software.
Não Funcional	Manutenibilidade	Modularidade	O design do software deve ser modular, facilitando a adição de novas funcionalidades
Não Funcional	Escalabilidade	Crescimento de dados	O sistema deve lidar com aumento no volume de dados e número de usuários sem degradação de desempenho.
Não Funcional	Escalabilidade	Expansão de funcionalidades	Deve ser possível adicionar novas funcionalidades sem grandes reestruturações.

4.7 Requisitos funcionais web

Categoria	Tipo	Requisitos	Descrição
Funcional	Usuário	Cadastro de Exposições	O sistema deve permitir o cadastro de exposições, com detalhes como nome, descrição, data de início e fim.
Funcional	Obras	Gerenciamento de coleções	O sistema deve permitir o gerenciamento das coleções do museu, incluindo obras de arte, autores e categorias.
Funcional	Obras	Pesquisa de Exposições	O sistema deve permitir que os usuários pesquisem exposições e obras de arte disponíveis no museu.
Funcional	Usuário	Area de login e cadastro do usuário	O sistema deve permitir que usuários e administradores se registrem e façam login para acessar funcionalidades.
Funcional	Usuário	Geração de relatórios	O sistema deve permitir que administradores gerem relatórios sobre visitantes, exposições e feedbacks.

Funcional	Usuário	Sistema de feedback	O sistema deve permitir que os visitantes deixem feedbacks sobre exposições ou a experiência geral no museu.
------------------	---------	---------------------	--

4.8 Requisitos não funcionais WEB

Categoria	Tipo	Requisitos	Descrição
Não Funcional	Usabilidade	Intuitividade	O software deve ser fácil de utilizar para visitantes de todas as idades, e níveis de habilidade tecnológica.
Não Funcional	Usabilidade	Interface amigável	A interface deve ser clara e atraente, com menus e botões bem-organizados.
Não Funcional	Usabilidade	Responsividade	O sistema deve ser responsivo, adaptando-se a diferentes tamanhos de tela e dispositivos.

Não Funcional	Desempenho	Tempo de resposta	As operações devem ter um tempo de resposta inferior a 2 segundos.
Não Funcional	Desempenho	Capacidade de carga	O sistema deve suportar vários usuários ao mesmo tempo, sem degradação significativa de desempenho.
Não Funcional	Confiabilidade	Disponibilidade	O software deve estar disponível a todo tempo.

Não Funcional	Confiabilidade	Falha	Em caso de falha o sistema deve se recuperar rapidamente e retomar a operação normalmente sem perda de dados.
Não Funcional	Segurança	Proteção de dados	Dados do usuário guardados em um banco de dados, utilizando criptografia.
Não Funcional	Portabilidade	Compatibilidade com navegadores	O sistema deve ser compatível com os navegadores mais populares, como Chrome, Firefox, Safari e Edge.
Não Funcional	Portabilidade	Facilidade de instalação	O software deve ser fácil de instalar e configurar em novos dispositivos.
Não Funcional	Manutenibilidade	Documentação	Deve haver a documentação clara e abrangente para manutenção e atualização do software.
Não Funcional	Manutenibilidade	Modularidade	O design do software deve ser modular, facilitando a adição de novas funcionalidades

Não Funcional	Escalabilidade	Crescimento de dados	O sistema deve lidar com aumento no volume de dados e número de usuários sem degradação de desempenho.
Não Funcional	Escalabilidade	Expansão de funcionalidades	Deve ser possível adicionar novas funcionalidades sem grandes reestruturações.

Não Funcional	Processos	Otimização	Algoritmos e processos devem ser otimizados para proporcionar uma experiência rápida e eficiente.
Não Funcional	Experiência do usuário	Feedback	O sistema deve fornecer feedback imediato e claro as ações do usuário.
Não Funcional	Experiência do usuário	Acessibilidade	O software deve ser acessível a pessoas com deficiência, seguindo as diretrizes de acessibilidade como WCAG.
Não Funcional	Legalidade	Conformidade com normas	O software deve estar conformidade com normas e regulamentos aplicáveis, como a ISO9126, ISO25001.
Não Funcional	Eficiência	Consumo de recursos	O software deve ser eficiente no uso de CPU memória e energia.

4.9 Análise de Requisitos

Para analisar os requisitos temos como base um estudo realizado pelo especialista Eduardo cunha sobre os documentos de requisitos.

Foi realizada uma análise detalhada com base no estudo do especialista para garantir que todos os requisitos fossem claros, completos e viáveis dentro do prazo, os integrantes fizeram diversas reuniões para chegar a um definitivo que seja melhor na exploração do software com intuito de facilitar as interações dos visitantes.

4.10 Análise de Ambiente

A análise do ambiente tem como função analisar o ambiente interno e externo do museu, identificando oportunidades e desafios que possam dificultar a implementação e a operação dos totens. Essa análise ajuda a compreender melhor as necessidades dos visitantes o que eles realmente estão buscando, ajuda também na infraestrutura sua eficácia, e sua competitividade no mercado, aborda estratégias que visa a experiencia do usuário, e garantir uma conformidade com as regulamentações, tendo assim um ambiente favorável para o progresso do projeto, essa análise foi feita de acordo com a infraestrutura do museu disponibilizada no título 4.4 Infraestrutura.

O ambiente de desenvolvimento será configurado com as seguintes ferramentas e tecnologias:

- ASP.NET MVC: Utilizado para o desenvolvimento da aplicação web, permitindo a criação de interfaces dinâmicas e escaláveis.
- SQL Server: O banco de dados relacional que armazenará todas as informações do sistema, incluindo dados de usuários, obras e feedback.
- Visual Studio: IDE escolhida para o desenvolvimento, com suporte para C# e ASP.NET, XML oferecendo recursos de depuração e gerenciamento de projetos.

4.11 Análise de Software

A análise de software envolveu a avaliação das necessidades do museu e a seleção de tecnologias adequadas. O sistema foi projetado para ser modular e ser escalável, permitindo futuras expansões e atualizações.

4.12 Análise de hardware

Análise de hardware consiste em saber qual hardware tem melhor custo-benefício, considerando não apenas o preço inicial de aquisição, mas também sua durabilidade, desempenho, eficiência energética e custos de manutenção ao longo do tempo.

4.13 Análise do Usuário

Sem compreender os usuários e as atividades que realizam, não é viável definir quais processos o sistema deve incluir. No desenvolvimento do software para o museu espacial, conduzindo uma análise detalhada dos visitantes e das operações do museu. Essa análise envolveu entrevistas com os visitantes, observação de suas interações com as exposições e coleta de feedback sobre suas experiências. Com essas informações, foi identificada a necessidade e expectativa dos usuários, o que permitiu definir funcionalidades essenciais para o software, como guias interativos, mapas do museu, informações detalhadas sobre as exposições e um quiz. Essa abordagem garantiu que o software fosse desenvolvido com o foco no usuário, podendo proporcionar uma experiência satisfatória e eficiente para todos os visitantes.

4.14 Gerenciamento de Implantação do Sistema

O gerenciamento de implantação foi conduzido em várias fases, incluindo planejamento, execução e monitoramento. Cada fase foi cuidadosamente planejada para minimizar interrupções nas operações do museu.

4.15 Implantação

Depois que o projeto for aprovado, ocorre a implantação, garantindo que todos os recursos necessários estejam disponíveis e que a equipe responsável esteja pronta para executar suas tarefas conforme o planejamento. Durante essa fase, monitoramos de perto o progresso para assegurar que o projeto atenda aos requisitos e prazos estabelecidos, e fazendo ajustes conforme necessário para resolver quaisquer problemas que possam surgir.

5 Arquitetura do Sistema

A arquitetura do sistema será baseada em um modelo cliente-servidor, onde:

- Cliente: O usuário acessa o sistema através de um navegador web, utilizando dispositivos como desktops, tablets e smartphones.
- Servidor: Um servidor web, rodando Internet, hospedará a aplicação ASP.NET. O servidor será responsável por processar as requisições dos usuários e se comunicar com o banco de dados SQL para armazenamento e recuperação de informações.

5.1 Modelagem de Dados

A modelagem de dados envolverá a criação de um esquema de banco de dados que abrange as principais entidades e seus relacionamentos. As entidades a serem modeladas incluem:

- Usuário: Armazena informações como ID (chave primária), nome, e-mail, senha.
- Feedback: Armazena os comentários dos usuários, incluindo ID (chave primária), ID do usuário (chave estrangeira), texto do feedback e data de envio.
- Obra: Contém detalhes sobre as obras do museu, como ID (chave primária), título, descrição, imagens e comentários associados.

5.2 Integração do Banco de Dados

Segundo (*An Introduction to Database Systems*, C. J. Date, 1ª à 8ª edição, 2003)

"As operações de inserção, exclusão e atualização são fundamentais para a manipulação de dados em um sistema relacional. Elas garantem que os usuários possam modificar o estado do banco de dados, enquanto as consultas oferecem uma forma de recuperar informações armazenadas de maneira eficiente."

A integração do banco de dados será realizada através de Entity Framework, que permitirá a manipulação de dados de forma simples e eficiente. As principais etapas incluem:

- Criação do Banco de Dados: A estrutura do banco de dados será criada com as tabelas necessárias, definindo as relações entre elas.

- Configuração da Conexão: A string de conexão será configurada no arquivo de configuração da aplicação para permitir a comunicação com o SQL Server.
- CRUD (Create, Read, Update, Delete): Implementação das operações de CRUD para todas as entidades do sistema, garantindo que os dados possam ser inseridos, lidos, atualizados e excluídos conforme necessário.

5.3 Desenvolvimento de Funcionalidades

O desenvolvimento das funcionalidades principais será realizado em etapas, com foco em testes constantes para garantir a qualidade do código e a usabilidade do sistema. As funcionalidades a serem implementadas incluem:

- Cadastro e Login: Desenvolvimento do sistema de autenticação, incluindo validações de entrada e armazenamento seguro de senhas.
- Quizzes: Criação do mecanismo para adicionar, editar e excluir quizzes, assim como o gerenciamento de perguntas e respostas.
- Feedback: Implementação do sistema que permite que os usuários enviem feedback e que a equipe do museu analise as respostas recebidas.
- Obras do Museu: Desenvolvimento da seção que apresenta as obras, incluindo a capacidade de adicionar novos itens e gerenciar comentários dos visitantes.

6 INTERFACE DO USUARIO (UI)

A interface do usuário será desenvolvida com foco em usabilidade e estética, utilizando boas práticas de design. Elementos importantes incluirão:

- Menu de Navegação: Um menu claro e acessível que permitirá aos visitantes navegarem rapidamente entre as diferentes seções do site, como "Home", "Quizzes", "Obras" e "Feedback".
- Página Inicial: A página inicial será atraente, apresentando um resumo das principais atrações do museu e links diretos para as seções mais relevantes.
- Página de Login e Cadastro: Formulários simples e diretos que guiarão os usuários através do processo de registro e login, com mensagens de erro claras em caso de problemas.

- Seção de Quizzes: Uma interface interativa que apresentará os quizzes de maneira dinâmica, com opções de resposta destacadas e feedback visual sobre o desempenho dos usuários.

- Seção de Obras: Uma galeria de imagens com informações descritivas sobre cada obra.

6.1 Experiência do Usuário (UX)

A seguir, uma abordagem aprofundada para cada aspecto do UX na plataforma, garantindo que o "Museu para Marte" seja interativo, acessível e intuitivo.

- Estrutura de Navegação e Acessibilidade:

Objetivo: Proporcionar uma navegação intuitiva e acessível, para que qualquer usuário, independentemente de idade ou conhecimento técnico, possa explorar o museu com facilidade.

Menu de Navegação Persistente: Um menu fixo será implementado na parte superior, permitindo acesso rápido e constante às principais seções como "Home", "Quizzes", "Obras" e "Feedback". Esse menu será responsivo e adaptável, transformando-se em um ícone de menu expansível em dispositivos móveis.

Organização do Conteúdo: As páginas serão organizadas de forma lógica, com etiquetas claras, garantindo que o visitante encontre facilmente o que busca. Cada seção será apresentada de maneira linear e hierárquica, para que os visitantes possam explorar sem se perder.

Feedback Visual de Interação: Ao passar o mouse sobre links e botões, o visitante verá uma mudança sutil de cor ou sombra, confirmando que o item é interativo. Isso cria uma sensação de controle e reduz a incerteza sobre onde clicar.

- Experiência de Primeira Impressão: Página Inicial

Objetivo: Capturar a atenção dos visitantes imediatamente, dando uma visão geral do museu e facilitando o início da navegação.

- Design Atraente e Limpo: Imagens de Marte e das principais obras serão usadas como destaque visual. Com menos texto e mais visuais, a página inicial evitará uma sobrecarga de informações, permitindo que o visitante se concentre nos pontos principais.

- Processos Diretos e Intuitivos: Login e Cadastro

Objetivo: Tornar o processo de registro e login rápido e simples, eliminando frustrações e erros.

Formulários Simples e Direcionados: Os campos de entrada estarão claramente etiquetados e organizados. O sistema apresentará automaticamente sugestões, como requisitos de senha, e usará validações instantâneas para alertar o usuário em caso de erro.

Mensagens de Erro e Feedbacks Esclarecedores: Se o usuário cometer um erro, uma mensagem descritiva e amigável explicará o que é necessário para corrigi-lo, reduzindo o risco de frustração.

- Engajamento e Feedback Imediato: Seção de Quizzes

Objetivo: Oferecer uma experiência interativa, permitindo que os visitantes aprendam e se divirtam enquanto exploram o museu.

Design Visual Atrativo: Cada quiz será apresentado em uma interface colorida e envolvente, com destaque para as opções de resposta. Elementos visuais, como ícones ou ilustrações, irão enriquecer a experiência.

Feedback Instantâneo: Após cada resposta, o sistema fornecerá feedback imediato, indicando acertos e erros. Isso inclui pequenas animações, como marcações em verde para acertos e em vermelho para erros, mantendo o visitante motivado a continuar.

- Imersão e Engajamento com o Conteúdo: Seção de Obras

Objetivo: Oferecer uma experiência interativa que aprofunde o engajamento dos visitantes com as obras do museu.

Galeria Visual Atraente: A seção apresentará as obras em uma galeria visual com miniaturas que expandem ao clique, permitindo uma navegação rica e uma experiência visual detalhada.

Descrição Interativa e Multimídia: Ao clicar em uma obra, o visitante verá uma página com descrições, e detalhes adicionais sobre cada peça. Ferramentas como zoom e arraste estarão disponíveis para visualizações detalhadas.

- Acessibilidade e Inclusão

Objetivo: Assegurar que todos os visitantes, independentemente de suas habilidades, possam usufruir do museu.

Recursos de Acessibilidade: Opções de ajuste de fonte, cores de alto contraste e navegação por teclado garantirão que usuários com deficiências possam navegar sem dificuldades.

Leitura de Tela e Textos Alternativos: Cada imagem terá descrições alternativas para usuários que dependem de leitores de tela. Os elementos interativos também serão rotulados para facilitar a navegação acessível.

7.TESTE DO SOFTWARES

Segundo o (Roger S. Pressman, 2011) aborda uma ampla gama de tópicos relacionados à engenharia de software. Alguns dos principais conceitos e autores que ele cita incluem técnicas de testes funcionais, unitários e de integração na parte 4.

Teste de software é uma prática essencial no ciclo de desenvolvimento de software, cujo objetivo é garantir a qualidade e a confiabilidade do produto. Consiste em verificar se o software atende aos requisitos especificados, funciona conforme o esperado e está livre de erros ou defeitos.

Existem diferentes tipos de testes de software, incluindo testes de unidade, integração, sistema, aceitação do usuário e regressão, cada um focado em aspectos específicos do software e do processo de desenvolvimento.

Os testes de unidade são realizados para verificar se unidades individuais de código funcionam corretamente.

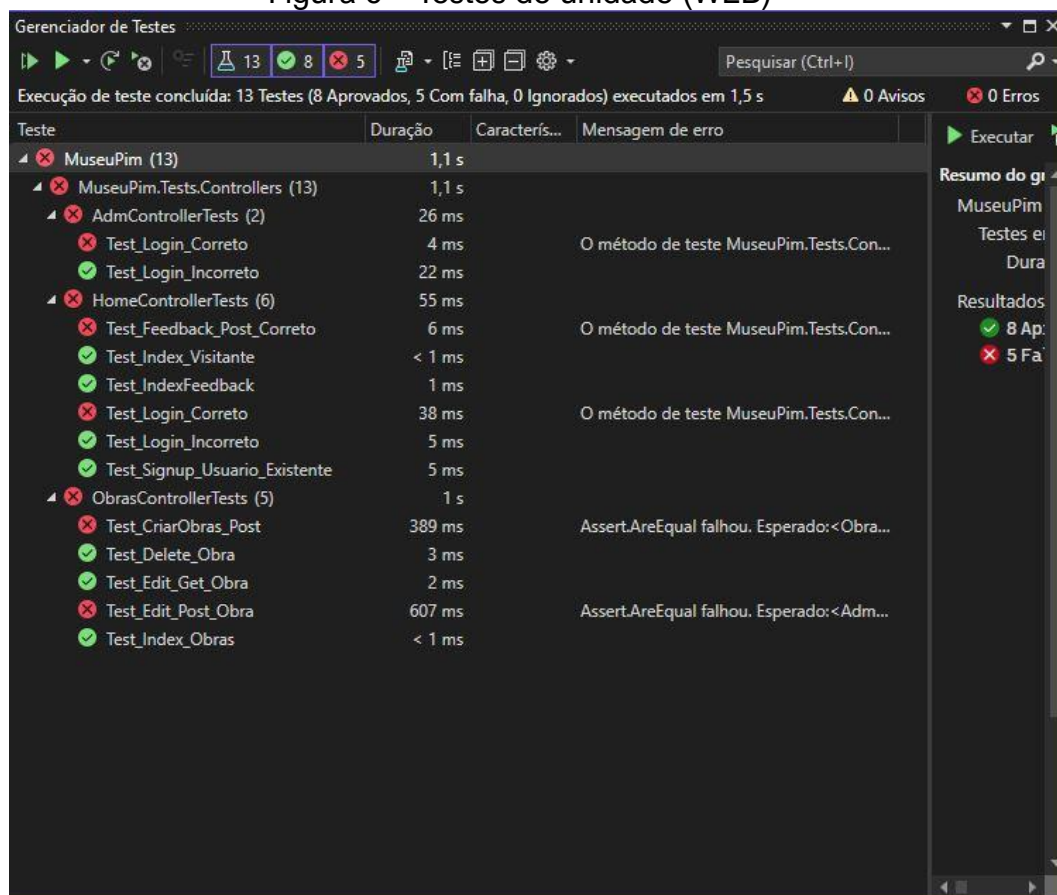
Além disso, os testes de software podem ser realizados manualmente ou automaticamente, sendo este último geralmente preferido devido à sua eficiência e reprodutibilidade. Em resumo, o teste de software desempenha um papel crucial na garantia da qualidade do software, ajudando a identificar e corrigir defeitos antes que o produto seja lançado, garantindo assim uma experiência positiva para os usuários.

7.1 Teste

Depois da fase de implantação do projeto, inicia-se a fase de testes do projeto, onde todas as funcionalidades são rigorosamente avaliadas para garantir que atendem os requisitos. Durante a esta etapa, identifica - se e se corrige quaisquer defeitos, realizando testes de integração, teste unitário, teste de aplicação. Esses processos são cruciais para assegurar que o sistema esteja em conformidade para que possa ser lançado aos usuários finais.

O sistema passou por uma série de testes unitários para garantir sua funcionalidade, conforme na figura 6 mostrado abaixo.

Figura 6 – Testes de unidade (WEB)



Autor: Autoria própria

7.1.2 Codificações testes (web)

Na codificação dos testes foi utilizado o MStest para realizar esses testes unitários, foi realizado 5 tipos de teste entre eles as 3 controller e as duas classes (desktop), já na web foi realizado 3 tipos de testes que são as controller.

Foi feito os testes do administrador conforme na figura 7 (a) e 7(b).

Figura 7 (a) – Teste de AdmController

```

MuseuPim
1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using Moq;
3  using MuseuPim.Controllers;
4  using MuseuPim.Data;
5  using System.Linq;
6  using System.Data.Entity;
7  using System.Web.Mvc;
8  using System.Reflection;
9
10 namespace MuseuPim.Tests.Controllers
11 {
12     [TestClass]
13     public class AdmControllerTests
14     {
15         private Mock<Bd_Pim_MuseuContexto7> mockDb;
16         private Mock<DbSet<Tbl_Administrador>> mockDbSet;
17         private AdmController controller;
18
19         [TestInitialize]
20         public void Initialize()
21         {
22             // Criando um administrador de teste
23             var admin = new Tbl_Administrador
24             {
25                 Nome_ADM = "admin",
26                 Senha_ADM = "senha123"
27             };
28
29             // Simulando um DbSet com um único administrador
30             var admins = new[] { admin }.AsQueryable();
31
32             // Mockando o DbSet
33             mockDbSet = new Mock<DbSet<Tbl_Administrador>>();
34             mockDbSet.As<IQueryable<Tbl_Administrador>>().
35                 Setup(m => m.Provider).Returns(admins.Provider);
36             mockDbSet.As<IQueryable<Tbl_Administrador>>().
37                 Setup(m => m.Expression).Returns(admins.Expression);
38             mockDbSet.As<IQueryable<Tbl_Administrador>>().
39                 Setup(m => m.ElementType).Returns(admins.ElementType);
40             mockDbSet.As<IQueryable<Tbl_Administrador>>().
41                 Setup(m => m.GetEnumerator()).Returns(admins.GetEnumerator());
42
43             // Mockando o DbContext para retornar o DbSet
44             mockDb = new Mock<Bd_Pim_MuseuContexto7>();
45             mockDb.Setup(db => db.Tbl_Administrador).Returns(mockDbSet.Object);
46
47             // Inicializando o controlador
48             controller = new AdmController();
49
50             // Usando reflexão para injetar o mock do DbContext privado
51             var dbField = typeof(AdmController).GetField("db", BindingFlags.NonPublic | BindingFlags.Instance);
52             dbField.SetValue(controller, mockDb.Object); // Injetando o mock diretamente
53         }
54     }

```

Autor: Autoria própria

Figura 7 (b) – Teste de AdmController

```

[TestMethod]
0 referências
public void Test_Login_Correto()
{
    // Arrange: criando um administrador com credenciais corretas
    var admin = new Tbl_Administrador
    {
        Nome_ADM = "admin",
        Senha_ADM = "senha123"
    };

    // Act: chamando o método Login com as credenciais corretas
    var result = controller.Login(admin) as RedirectToRouteResult;

    // Assert: verificando se houve redirecionamento para a ação "Index"
    Assert.IsNotNull(result); // Verificando que houve redirecionamento
    Assert.AreEqual("Index", result.RouteValues["action"]); // Ação correta
}

[TestMethod]
0 referências
public void Test_Login_Incorreto()
{
    // Arrange: criando um administrador com credenciais incorretas
    var admin = new Tbl_Administrador
    {
        Nome_ADM = "admin",
        Senha_ADM = "senhaErrada"
    };

    // Act: chamando o método Login com as credenciais incorretas
    var result = controller.Login(admin) as ViewResult;

    // Assert: verificando se a View retornada possui a mensagem de erro
    Assert.IsNotNull(result); // Verificando que retornou uma View
    Assert.AreEqual("Nome de Usuário ou senha incorretos", result.ViewBag.Notification); // Mensagem de erro esperada
}
}

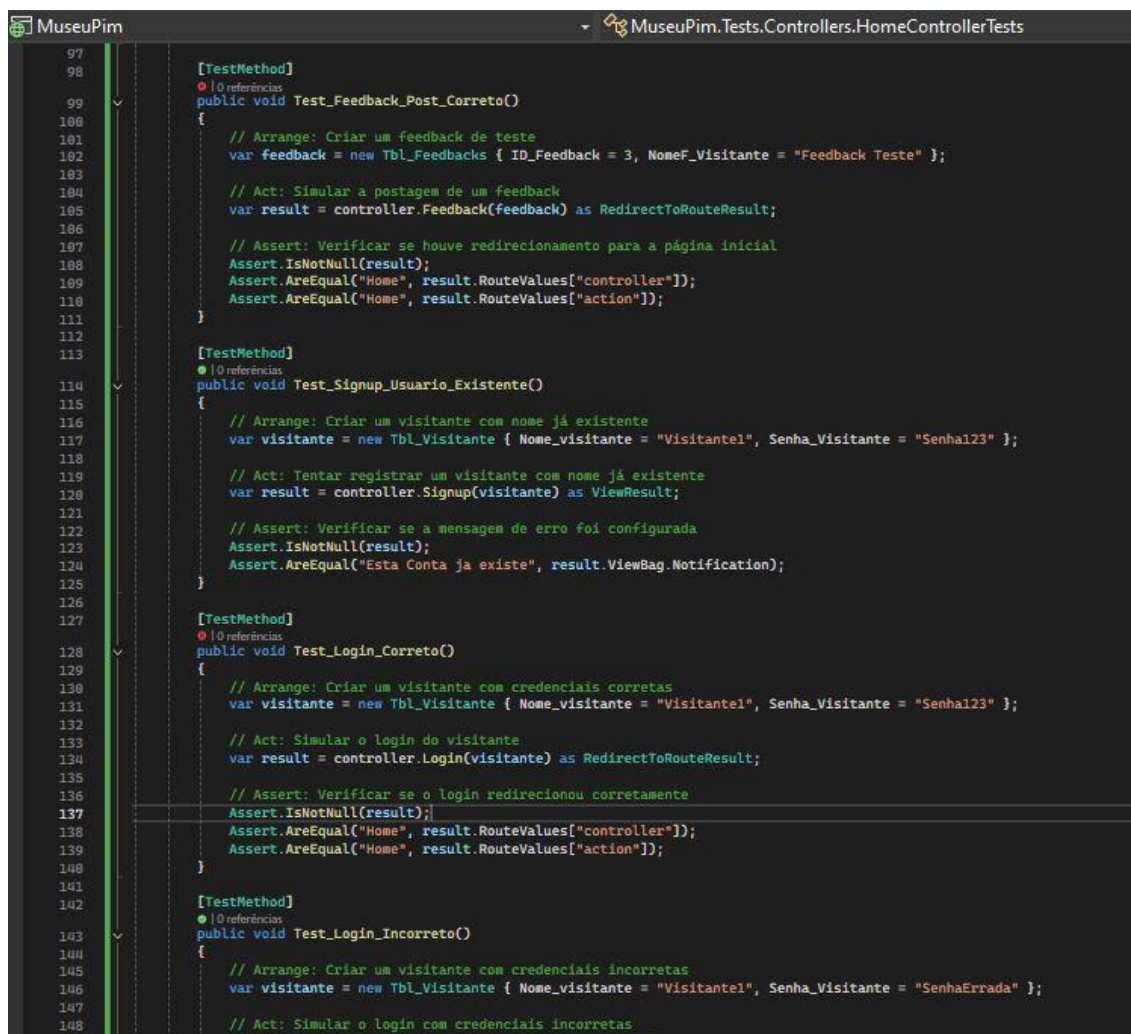
```

Autor: Autoria própria

Esse teste valida a lógica de autenticação no controlador AdmController, simulando cenários de login com credenciais válidas e inválidas. A abordagem de mocking isola o controlador de dependências externas, como o banco de dados, permitindo testes rápidos e confiáveis. Essa prática é essencial para garantir a qualidade do sistema e identificar problemas em estágios iniciais de desenvolvimento.

Foram feitos os testes da home para saber se todos os métodos estão corretos e suas funcionalidades foram atingidas, conforme mostrado nos códigos das figuras 8(a), 8(b) e 8(c).

Figura 8 (a) – Testes de home controller



```

97
98
99 [TestMethod]
100 // 0 referências
101 public void Test_Feedback_Post_Correto()
102 {
103     // Arrange: Criar um feedback de teste
104     var feedback = new Tbl_Feedbacks { ID_Feedback = 3, NomeF_Visitante = "Feedback Teste" };
105
106     // Act: Simular a postagem de um feedback
107     var result = controller.Feedback(feedback) as RedirectToRouteResult;
108
109     // Assert: Verificar se houve redirecionamento para a página inicial
110     Assert.IsNotNull(result);
111     Assert.AreEqual("Home", result.RouteValues["controller"]);
112     Assert.AreEqual("Home", result.RouteValues["action"]);
113 }
114
115 [TestMethod]
116 // 0 referências
117 public void Test_Signup_Usuario_Existente()
118 {
119     // Arrange: Criar um visitante com nome já existente
120     var visitante = new Tbl_Visitante { Nome_visitante = "Visitante1", Senha_Visitante = "Senha123" };
121
122     // Act: Tentar registrar um visitante com nome já existente
123     var result = controller.Signup(visitante) as ViewResult;
124
125     // Assert: Verificar se a mensagem de erro foi configurada
126     Assert.IsNotNull(result);
127     Assert.AreEqual("Esta Conta ja existe", result.ViewBag.Notification);
128 }
129
130 [TestMethod]
131 // 0 referências
132 public void Test_Login_Correto()
133 {
134     // Arrange: Criar um visitante com credenciais corretas
135     var visitante = new Tbl_Visitante { Nome_visitante = "Visitante1", Senha_Visitante = "Senha123" };
136
137     // Act: Simular o login do visitante
138     var result = controller.Login(visitante) as RedirectToRouteResult;
139
140     // Assert: Verificar se o login redirecionou corretamente
141     Assert.IsNotNull(result);
142     Assert.AreEqual("Home", result.RouteValues["controller"]);
143     Assert.AreEqual("Home", result.RouteValues["action"]);
144 }
145
146 [TestMethod]
147 // 0 referências
148 public void Test_Login_Incorreto()
149 {
150     // Arrange: Criar um visitante com credenciais incorretas
151     var visitante = new Tbl_Visitante { Nome_visitante = "Visitante1", Senha_Visitante = "SenhaErrada" };
152
153     // Act: Simular o login com credenciais incorretas
154     var result = controller.Login(visitante) as ViewResult;
155
156     // Assert: Verificar se a mensagem de erro foi configurada
157     Assert.IsNotNull(result);
158     Assert.AreEqual("Credenciais incorretas", result.ViewBag.Notification);
159 }

```

Autor: Autoria própria

Figura 8 (b) – Testes de home controller

```

41
42
43 // Mockando o DbSet<Tbl_Feedbacks> (não é necessário se não for testar diretamente)
44 mockFeedbackDbSet = new Mock<System.Data.Entity.DbSet<Tbl_Feedbacks>>();
45 var feedbacks = new List<Tbl_Feedbacks>
46 {
47     new Tbl_Feedbacks { ID_Feedback = 1, NomeF_Visitante = "Feedback1" },
48     new Tbl_Feedbacks { ID_Feedback = 2, NomeF_Visitante = "Feedback2" }
49 }.AsQueryable();
50
51 mockFeedbackDbSet.As<IQueryable<Tbl_Feedbacks>>().
52     .Setup(m => m.Provider).Returns(feedbacks.Provider);
53 mockFeedbackDbSet.As<IQueryable<Tbl_Feedbacks>>().
54     .Setup(m => m.Expression).Returns(feedbacks.Expression);
55 mockFeedbackDbSet.As<IQueryable<Tbl_Feedbacks>>().
56     .Setup(m => m.ElementType).Returns(feedbacks.ElementType);
57 mockFeedbackDbSet.As<IQueryable<Tbl_Feedbacks>>().
58     .Setup(m => m.GetEnumerator()).Returns(feedbacks.GetEnumerator());
59
60 // Configurando o mock do DbContext para retornar os DbSets
61 mockDb.Setup(db => db.Tbl_Visitante).Returns(mockVisitanteDbSet.Object);
62 mockDb.Setup(db => db.Tbl_Feedbacks).Returns(mockFeedbackDbSet.Object);
63
64 // Inicializando o controlador
65 controller = new HomeController();
66
67 // Usando reflexão para injetar o DbContext no controlador
68 var dbField = typeof(HomeController).GetField("db", BindingFlags.NonPublic | BindingFlags.Instance);
69 dbField.SetValue(controller, mockDb.Object);
70
71
72 [TestMethod]
73 public void Test_Index_Visitante()
74 {
75     // Arrange: Preparar os dados mockados do DbSet de Visitantes
76     var result = controller.Index() as ViewResult;
77
78     // Assert: Verificando se a View foi retornada e a lista de visitantes está correta
79     Assert.IsNotNull(result);
80     var model = result.Model as List<Tbl_Visitante>;
81     Assert.IsNotNull(model);
82     Assert.AreEqual(2, model.Count); // Espera-se que haja 2 visitantes no mock
83 }
84
85 [TestMethod]
86 public void Test_IndexFeedback()
87 {
88     // Arrange: Preparar os dados mockados do DbSet de Feedbacks
89     var result = controller.IndexFeedback() as ViewResult;
90
91     // Assert: Verificando se a View foi retornada e a lista de feedbacks está correta
92     Assert.IsNotNull(result);
93     var model = result.Model as List<Tbl_Feedbacks>;
94     Assert.IsNotNull(model);

```

Autor: Autoria própria

Figura 8 (c) – Testes de home controller

```

MuseuPim
    Microsoft.VisualStudio.TestTools.UnitTesting;
    Moq;
    MuseuPim.Controllers;
    MuseuPim.Data;
    System.Linq;
    System.Web.Mvc;
    System.Collections.Generic;
    System.Reflection;

namespace MuseuPim.Tests.Controllers
{
    [TestClass]
    public class HomeControllerTests
    {
        private Mock<Bd_Pim_MuseuContexto7> mockDb;
        private Mock<System.Data.Entity.DbSet<Tbl_Visitante>> mockVisitanteDbSet;
        private Mock<System.Data.Entity.DbSet<Tbl_Feedbacks>> mockFeedbackDbSet;
        private HomeController controller;

        [TestInitialize]
        public void Initialize()
        {
            // Mockando o DbContext
            mockDb = new Mock<Bd_Pim_MuseuContexto7>();

            // Mockando o DbSet<Tbl_Visitante>
            mockVisitanteDbSet = new Mock<System.Data.Entity.DbSet<Tbl_Visitante>>();
            var visitantes = new List<Tbl_Visitante>
            {
                new Tbl_Visitante { ID_Visitante = 1, Nome_visitante = "Visitante1", Senha_Visitante = "Senha123" },
                new Tbl_Visitante { ID_Visitante = 2, Nome_visitante = "Visitante2", Senha_Visitante = "Senha456" }
            }.AsQueryable();

            mockVisitanteDbSet.As<IQueryable<Tbl_Visitante>>()
                .Setup(m => m.Provider).Returns(visitantes.Provider);
            mockVisitanteDbSet.As<IQueryable<Tbl_Visitante>>()
                .Setup(m => m.Expression).Returns(visitantes.Expression);
            mockVisitanteDbSet.As<IQueryable<Tbl_Visitante>>()
                .Setup(m => m.ElementType).Returns(visitantes.ElementType);
            mockVisitanteDbSet.As<IQueryable<Tbl_Visitante>>()
                .Setup(m => m.GetEnumerator()).Returns(visitantes.GetEnumerator());

            // Mockando o DbSet<Tbl_Feedbacks> (não é necessário se não for testar diretamente)
            mockFeedbackDbSet = new Mock<System.Data.Entity.DbSet<Tbl_Feedbacks>>();
            var feedbacks = new List<Tbl_Feedbacks>
            {
                new Tbl_Feedbacks { ID_Feedback = 1, NomeF_Visitante = "Feedback1" },
                new Tbl_Feedbacks { ID_Feedback = 2, NomeF_Visitante = "Feedback2" }
            }.AsQueryable();

            mockFeedbackDbSet.As<IQueryable<Tbl_Feedbacks>>()
                .Setup(m => m.Provider).Returns(feedbacks.Provider);
            mockFeedbackDbSet.As<IQueryable<Tbl_Feedbacks>>()
                .Setup(m => m.Expression).Returns(feedbacks.Expression);
        }
    }
}

```

Autor: Autoria própria

Esses testes garantem o funcionamento correto do HomeController, cobrindo cenários comuns de interação, como exibição de dados, cadastro e login. A utilização de mocks isola o controlador de dependências externas, como o banco de dados, permitindo a verificação eficiente e confiável da lógica de negócios.

O teste foi realizado para identificar se todos os métodos estavam de acordo com suas funcionalidades, conforme no código abaixo nas figuras 9(a), 9(b) e 9(c).

Figura 9 (a) – Testes de obras controller

```

1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using Moq;
3  using MuseuPim.Controllers;
4  using MuseuPim.Data;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Web.Mvc;
8  using System.Reflection;
9
10 namespace MuseuPim.Tests.Controllers
11 {
12     [TestClass]
13     public class ObrasControllerTests
14     {
15         private Mock<Bd_Pim_MuseuContexto7> mockDb;
16         private Mock<System.Data.Entity.DbSet<Tbl_Obras>> mockObrasDbSet;
17         private ObrasController controller;
18
19         [TestInitialize]
20         public void Initialize()
21         {
22             // Criando o mock do DbContext
23             mockDb = new Mock<Bd_Pim_MuseuContexto7>();
24
25             // Criando o mock do DbSet<Tbl_Obras>
26             mockObrasDbSet = new Mock<System.Data.Entity.DbSet<Tbl_Obras>>();
27             var obras = new List<Tbl_Obras>
28             {
29                 new Tbl_Obras { ID_Obras = 1, Titulo_Obra = "Obra 1", Descricao_Obra = "Descrição 1", Imagem_Obra = "/Image/obra1.jpg" },
30                 new Tbl_Obras { ID_Obras = 2, Titulo_Obra = "Obra 2", Descricao_Obra = "Descrição 2", Imagem_Obra = "/Image/obra2.jpg" }
31             }.AsQueryable();
32
33             mockObrasDbSet.As<IQueryable<Tbl_Obras>>()
34                 .Setup(m => m.Provider).Returns(obras.Provider);
35             mockObrasDbSet.As<IQueryable<Tbl_Obras>>()
36                 .Setup(m => m.Expression).Returns(obras.Expression);
37             mockObrasDbSet.As<IQueryable<Tbl_Obras>>()
38                 .Setup(m => m.ElementType).Returns(obras.ElementType);
39             mockObrasDbSet.As<IQueryable<Tbl_Obras>>()
40                 .Setup(m => m.GetEnumerator()).Returns(obras.GetEnumerator());
41
42             // Configurando o mock do DbContext para retornar o DbSet de Tbl_Obras
43             mockDb.Setup(db => db.Tbl_Obras).Returns(mockObrasDbSet.Object);
44
45             // Inicializando o controlador
46             controller = new ObrasController();
47
48             // Usando reflexão para injetar o DbContext no controlador
49             var dbField = typeof(ObrasController).GetField("_context", BindingFlags.NonPublic | BindingFlags.Instance);
50             dbField.SetValue(controller, mockDb.Object);
51         }
52     }
53 }

```

Autor: Autoria própria

Figura 9 (b) – Testes de obras controller

```

MuseuPim
MuseuPim.Tests.Controllers.ObrasControllerTests

52
53 [TestMethod]
54 ● 0 referências
55 public void Test_Index_Obras()
56 {
57     // Act: Acessar a ação Index
58     var result = controller.Index() as ViewResult;
59
60     // Assert: Verificar se o retorno da View contém as obras
61     Assert.IsNotNull(result);
62     var model = result.Model as List<Tbl_Obras>;
63     Assert.IsNotNull(model);
64     Assert.AreEqual(2, model.Count); // Espera-se que haja 2 obras no mock
65 }
66
67 [TestMethod]
68 ● 0 referências
69 public void Test_Edit_Get_Obra()
70 {
71     // Act: Acessar a ação Edit para a obra com ID = 1
72     var result = controller.Edit(1) as ViewResult;
73
74     // Assert: Verificar se o retorno contém a obra correta
75     Assert.IsNotNull(result);
76     var model = result.Model as Tbl_Obras;
77     Assert.IsNotNull(model);
78     Assert.AreEqual(1, model.ID_Obras);
79 }
80
81 [TestMethod]
82 ● 0 referências
83 public void Test_Edit_Post_Obra()
84 {
85     // Arrange: Criar uma obra modificada
86     var obraEditada = new Tbl_Obras { ID_Obras = 1, Titulo_Obra = "Obra Editada", Descricao_Obra = "Descrição Editada" };
87
88     // Act: Submeter a edição
89     var result = controller.Edit(obraEditada) as RedirectToRouteResult;
90
91     // Assert: Verificar se a ação foi redirecionada para a página correta
92     Assert.IsNotNull(result);
93     Assert.AreEqual("Index", result.RouteValues["action"]);
94     Assert.AreEqual("Adm", result.RouteValues["controller"]);
95 }
96
97 [TestMethod]
98 ● 0 referências
99 public void Test_Delete_Obra()
100 {
101     // Arrange: Definir o ID da obra a ser deletada
102     var idObraParaDeletar = 1;
103
104     // Act: Chamar a ação de deletar
105     var result = controller.Delete(idObraParaDeletar) as RedirectToRouteResult;
106 }
107

```

Autor: Autoria própria

Figura 9 (c) – Testes de obras controller

```
[TestMethod]
[0 referências]
public void Test_CriarObras_Post()
{
    // Arrange: Criar uma obra com dados válidos
    var novaObra = new Tbl_Obras
    {
        Titulo_Obra = "Nova Obra",
        Descricao_Obra = "Nova Descrição",
        Imagem_Obra = "/Image/novaobra.jpg"
    };

    // Act: Submeter a criação da obra
    var result = controller.CriarObras(novaObra) as ViewResult;

    // Assert: Verificar se a obra foi criada e a mensagem configurada
    Assert.IsNotNull(result);
    Assert.AreEqual("Obra inserida com sucesso.", result.ViewBag.Message);
}
```

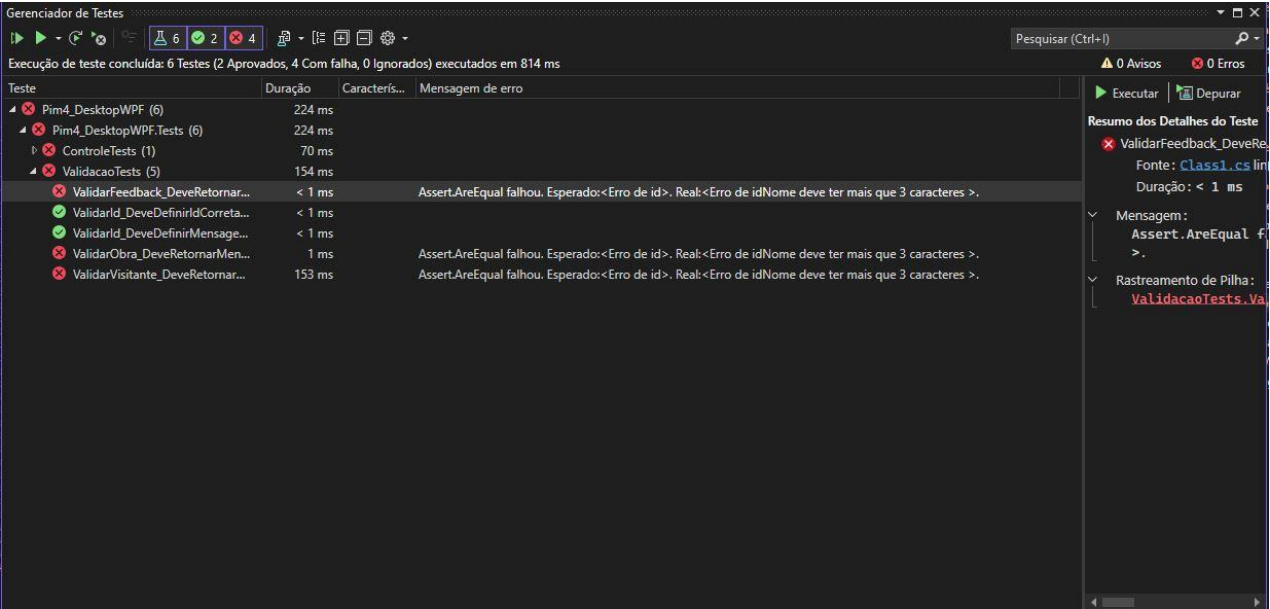
Autor: Autoria própria

Os testes garantem que o controlador ObrasController funciona corretamente ao lidar com obras, incluindo exibição, edição, criação e exclusão. A utilização de mocks para simular o banco de dados permite isolar os testes e verificar a lógica de negócios sem depender do banco real. Dessa forma, é possível detectar falhas rapidamente e assegurar a qualidade do código.

7.2 Codificações testes (Desktop)

Foi feito um diagnóstico para averiguar os métodos e suas funcionalidades conforme na figura 10 abaixo.

Figura 10 – Testes de unidade Controle



Autor: Autoria própria

O teste foi realizado para identificar se todos os métodos estavam de acordo com suas funcionalidades, conforme no código abaixo nas figuras 11.

Figura 11 – Testes de unidade Validação

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using Pim4_DesktopWPF.Modelo;
using Pim4_DesktopWPF.DAL;
using System.Collections.Generic;

namespace Pim4_DesktopWPF.Tests
{
    [TestClass]
    public class ControleTests
    {
        [TestMethod]
        public void CadastrarFeedback_DeveRetornarMensagemDeSucesso_QuandoDadosValidos()
        {
            // Arrange
            var controle = new Controle();

            // Mock de Validação
            var mockValidacao = new Mock<Validacao>();
            mockValidacao.Setup(v => v.ValidarFeedback(It.IsAny<List<string>>()))
                .Callback(() => mockValidacao.Object.mensagem = ""); // Sucesso na validação
            mockValidacao.Setup(v => v.id).Returns(1);

            // Mock de FeedbackDAO
            var mockFeedbackDAO = new Mock<FeedbackDAO>();
            mockFeedbackDAO.Setup(d => d.CadastrarFeedback(It.IsAny<FeedbackClasse>()))
                .Callback(() => mockFeedbackDAO.Object.mensagem = "Sucesso");

            // Substituindo os campos privados via reflexão
            typeof(Controle).GetField("_validacao", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance)
                .SetValue(controle, mockValidacao.Object);
            typeof(Controle).GetField("_feedbackDAO", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance)
                .SetValue(controle, mockFeedbackDAO.Object);

            // Act - Executa o método
            controle.CadastrarFeedback(new List<string> { "1", "Nome", "Feedback", "5" });

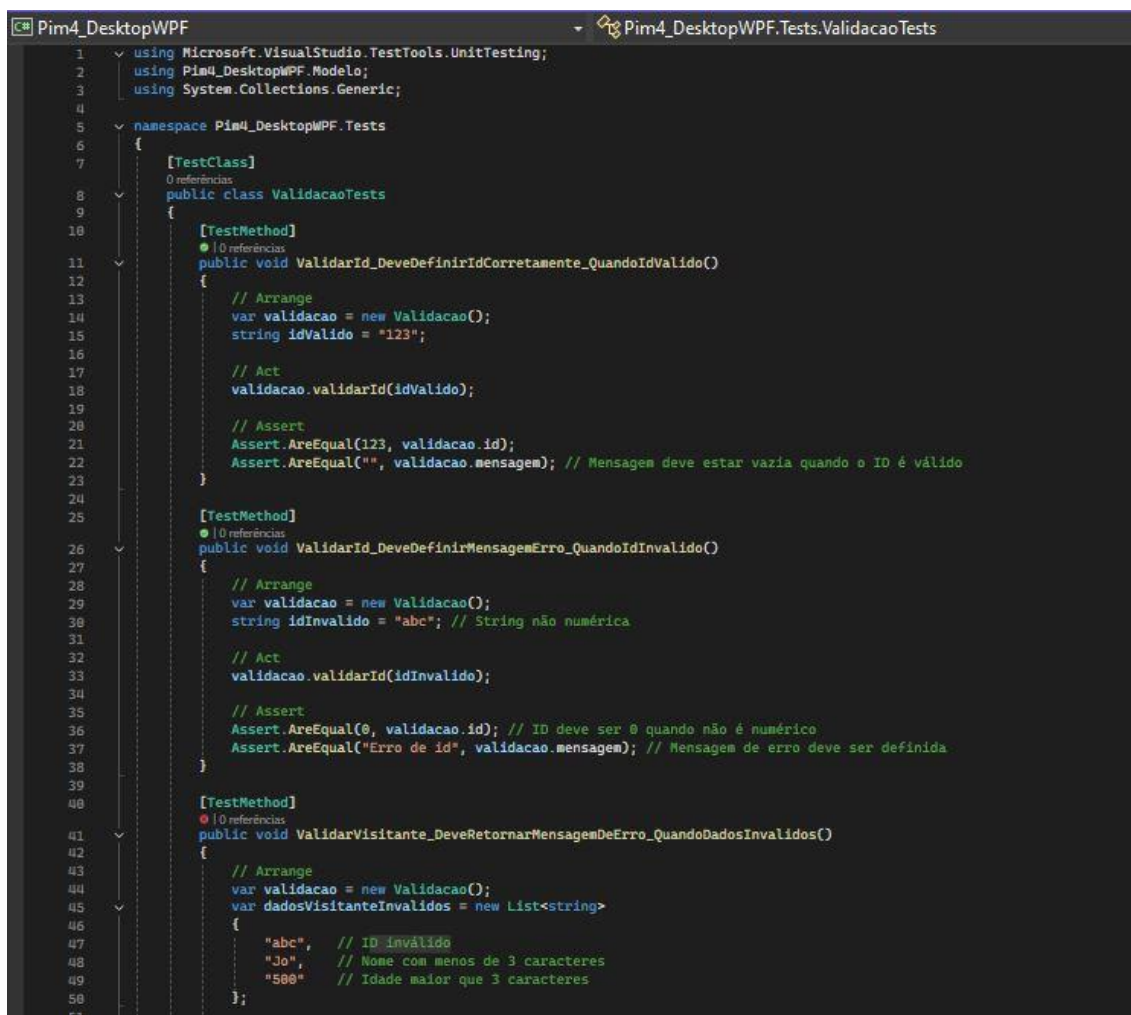
            // Assert - Verifica a mensagem
            Assert.AreEqual("Sucesso", controle.mensagem, "A mensagem não foi alterada para 'Sucesso' após o cadastro.");
        }
    }
}
```

Autor: Autoria própria

Esse teste simula o fluxo completo do método CadastrarFeedback, desde a validação dos dados até a operação de cadastro. Ele garante que, quando os dados fornecidos são válidos, a mensagem de sucesso é retornada corretamente. Essa abordagem é útil para verificar a integração entre diferentes componentes da aplicação sem depender de implementações reais, aumentando a eficiência e confiabilidade dos testes.

O teste foi realizado para identificar se todos os métodos estavam de acordo com suas funcionalidades, conforme no código abaixo nas figuras 12(a) e 12(b).

Figura 12 (a) – Testes de unidade Controle



```

1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using Pim4_DesktopWPF.Modelo;
3  using System.Collections.Generic;
4
5  namespace Pim4_DesktopWPF.Tests
6  {
7      [TestClass]
8      public class ValidacaoTests
9      {
10         [TestMethod]
11         public void ValidarId_DeveDefinirIdCorretamente_QuandoIdValido()
12         {
13             // Arrange
14             var validacao = new Validacao();
15             string idValido = "123";
16
17             // Act
18             validacao.validarId(idValido);
19
20             // Assert
21             Assert.AreEqual(123, validacao.id);
22             Assert.AreEqual("", validacao.mensagem); // Mensagem deve estar vazia quando o ID é válido
23         }
24
25         [TestMethod]
26         public void ValidarId_DeveDefinirMensagemErro_QuandoIdInvalido()
27         {
28             // Arrange
29             var validacao = new Validacao();
30             string idInvalido = "abc"; // String não numérica
31
32             // Act
33             validacao.validarId(idInvalido);
34
35             // Assert
36             Assert.AreEqual(0, validacao.id); // ID deve ser 0 quando não é numérico
37             Assert.AreEqual("Erro de id", validacao.mensagem); // Mensagem de erro deve ser definida
38         }
39
40         [TestMethod]
41         public void ValidarVisitante_DeveRetornarMensagemDeErro_QuandoDadosInvalidos()
42         {
43             // Arrange
44             var validacao = new Validacao();
45             var dadosVisitanteInvalidos = new List<string>
46             {
47                 "abc", // ID inválido
48                 "Jo", // Nome com menos de 3 caracteres
49                 "500" // Idade maior que 3 caracteres
50             };
51         }
52     }

```

Autor: Autoria própria

Figura 12 (b) – Testes de unidade Controle

```

50     };
51
52     // Act
53     validacao.ValidarVisitante(dadosVisitanteInvalidos);
54
55     // Assert
56     Assert.AreEqual("Erro de id", validacao.mensagem); // Mensagem do ID inválido
57     Assert.IsTrue(validacao.mensagem.Contains("Nome deve ter mais que 3 caracteres"));
58     Assert.IsTrue(validacao.mensagem.Contains("Idade elevada digite novamente"));
59 }
60
61 [TestMethod]
62 // 10 referências
63 public void ValidarFeedback_DeveRetornarMensagemDeErro_QuandoDadosInvalidos()
64 {
65     // Arrange
66     var validacao = new Validacao();
67     var dadosFeedbackInvalidos = new List<string>
68     {
69         "abc", // ID inválido
70         "Jo" // Nome com menos de 3 caracteres
71     };
72
73     // Act
74     validacao.ValidarFeedback(dadosFeedbackInvalidos);
75
76     // Assert
77     Assert.AreEqual("Erro de id", validacao.mensagem); // Mensagem do ID inválido
78     Assert.IsTrue(validacao.mensagem.Contains("Nome deve ter mais que 3 caracteres"));
79 }
80
81 [TestMethod]
82 // 10 referências
83 public void ValidarObra_DeveRetornarMensagemDeErro_QuandoDadosInvalidos()
84 {
85     // Arrange
86     var validacao = new Validacao();
87     var dadosObraInvalidos = new List<string>
88     {
89         "abc", // ID inválido
90         "Do" // Nome com menos de 3 caracteres
91     };
92
93     // Act
94     validacao.ValidarObra(dadosObraInvalidos);
95
96     // Assert
97     Assert.AreEqual("Erro de id", validacao.mensagem); // Mensagem do ID inválido
98     Assert.IsTrue(validacao.mensagem.Contains("Nome com mais que 3 caracteres"));
99 }
100 }

```

Autor: Autoria própria

Esse teste verifica a funcionalidade da classe Validação em um projeto WPF, garantindo que a lógica de validação funcione corretamente. Ele utiliza o framework **MSTest** para validar diferentes cenários, como IDs válidos e inválidos, nomes curtos demais e outros dados inconsistentes. Por exemplo, no método validar Id, é testado se um ID válido define a propriedade id corretamente, enquanto um ID inválido gera uma mensagem de erro. Da mesma forma, são testados os métodos que validam dados de visitantes, feedbacks e obras, verificando se mensagens de erro adequadas são retornadas para entradas inválidas. Esses testes ajudam a garantir a confiabilidade e robustez do sistema.

8. TREINAMENTO

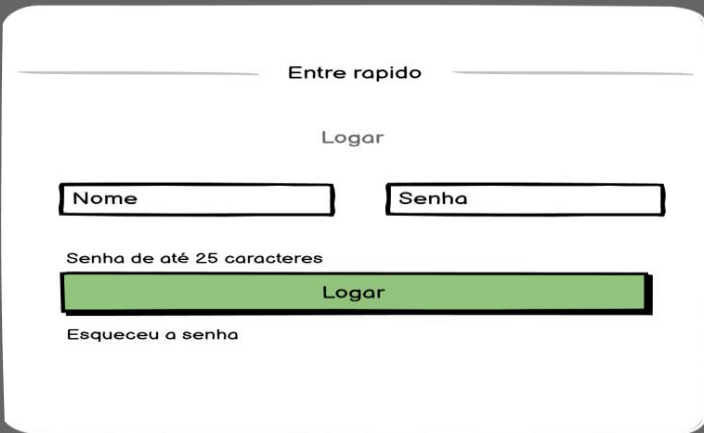
Os funcionários do museu receberam treinamento abrangente para garantir a operação do sistema para poder auxiliar os visitantes, incluindo sessões práticas e materiais de apoio detalhados. Já na função de administrador do site, terá um treinamento para apresentar como incluir obras, editar ou excluir, o funcionário terá monitoramento total do site, onde ele pode ver os feedbacks dos visitantes, e junto do treinamento terá um documento anexado do manual do usuário.

9. PROTOTIPAÇÃO

Aqui foi iniciado as prototipações de telas e antes do criamento de telas e das cores o layout em si, foi realizado reuniões e pesquisas para o melhor desenvolvimento de telas baseando-se em interface ao usuário, deixando uma tela chamativa autoexplicativa. Conseguindo deixar o software fácil de utilizar e com cores confortáveis ao olhar do usuário, conforme as figuras a seguir:

Foi realizado um protótipo da home para o desenvolvimento das telas, conforme nas figuras 13 e 14.

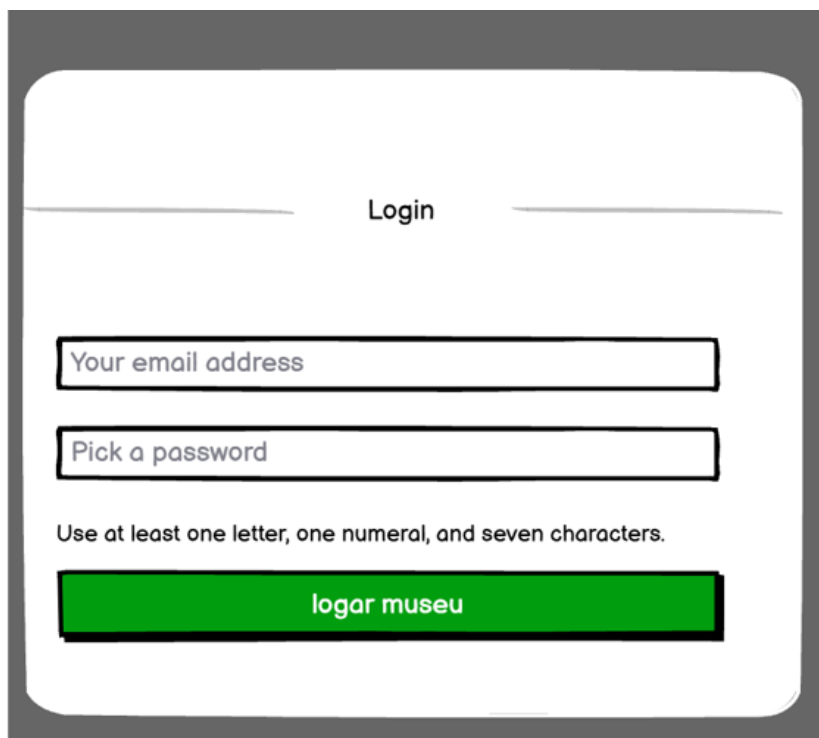
Figura 13 - Login



O protótipo da tela de login apresenta um design limpo e moderno. No topo, o texto "Entre rapido" (com uma correção ortográfica implícita para "rápido") está centralizado. Abaixo dele, o título "Logar" também é centralizado. A interface contém dois campos de entrada: "Nome" e "Senha", ambos com bordas arredondadas. Abaixo do campo "Senha", há uma dica de texto: "Senha de até 25 caracteres". Um botão de login, com o texto "Logar", está destacado por um fundo verde e uma borda preta. Na base da tela, há um link que diz "Esqueceu a senha".

Autor (Auto própria)

Figura 14 - Login

A login form prototype with a white background and rounded corners, set against a dark gray background. At the top, the word "Login" is centered. Below it are two input fields: the first is labeled "Your email address" and the second is labeled "Pick a password". Under the password field, there is a text requirement: "Use at least one letter, one numeral, and seven characters." At the bottom is a green button with the text "logar museu" in white.

Autor (Auto própria)

Versão 1.0

Esse é um protótipo da tela de login do site, simples e minimalista e mais bem-posto. No topo, há um título que diz "Entre rápido". Logo abaixo, há uma seção intitulada "Logar", onde o usuário pode inserir suas credenciais, a interface inclui dois campos de entrada: um para "Nome" e outro para "Senha". Abaixo, há um aviso que menciona que a senha deve ter "até 25 caracteres". O botão de login, destacado em verde e com o texto "Logar" centralizado, está logo abaixo. Na parte inferior da tela, há um link para "Esqueceu a senha", que oferece a opção de recuperar o acesso.

Versão 2.0

Foi alterado o fundo para uma imagem de marte para manter um significado visual, o campo de login foi alterado para e-mail e senha, e foi retirado o "Esqueceu a senha", e adicionado interatividade pelos botões com uma cor laranja, e foi alterado o texto "entre rápido" para "Aventure-se".

Foi realizado um protótipo do cadastro para o desenvolvimento das telas, conforme na figura 15.

Figura 15 - Cadastro

Entre rapido

Cadastro de usuario

Nome

Email opcional

Idade

Senha

Senha de até 25 caracteres

Cadastrar

Email opcional, porem se não preencher caso esqueça a senha não podera recuperar.

Autor (Auto própria)

Versão 1.0

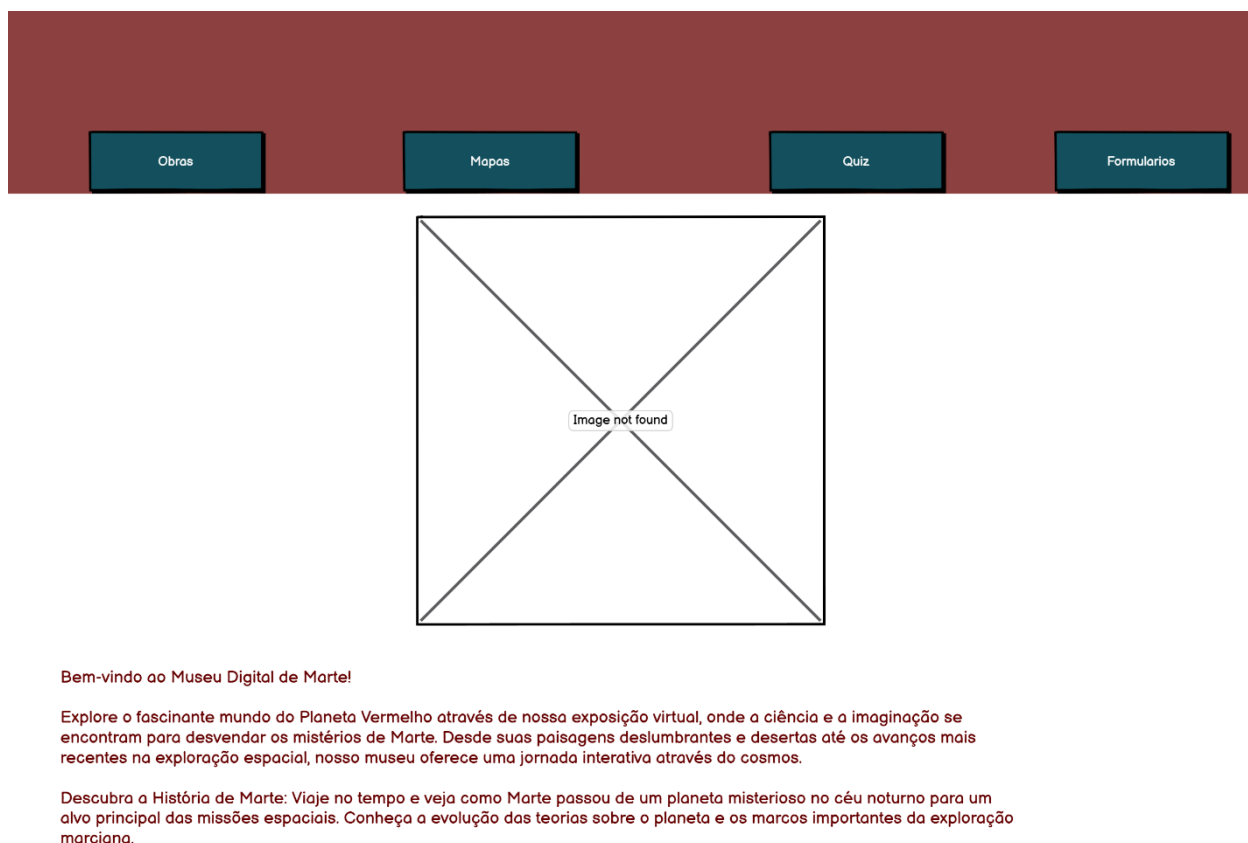
O protótipo ilustra uma interface de cadastro de usuário, com um design limpo e intuitivo. No topo, há uma área para inserir "Nome", "Email opcional", "Idade" e "Senha", com instrução como "Senha de até 25 caracteres". Um botão verde destacado, com o texto "Cadastrar", está posicionado abaixo dos campos, e um fundo cinza.

Versão 2.0

O cadastro foi todo modificado, no fundo foi adicionado uma imagem de um viajante a marte, colocando interações na caixa de texto com uma borda laranja, e um botão laranja de cadastrar.

Foi realizado um protótipo da home para o desenvolvimento das telas, conforme na figura 16.

Figura 16 - Home



Autor (Auto própria)

Versão 1.0:

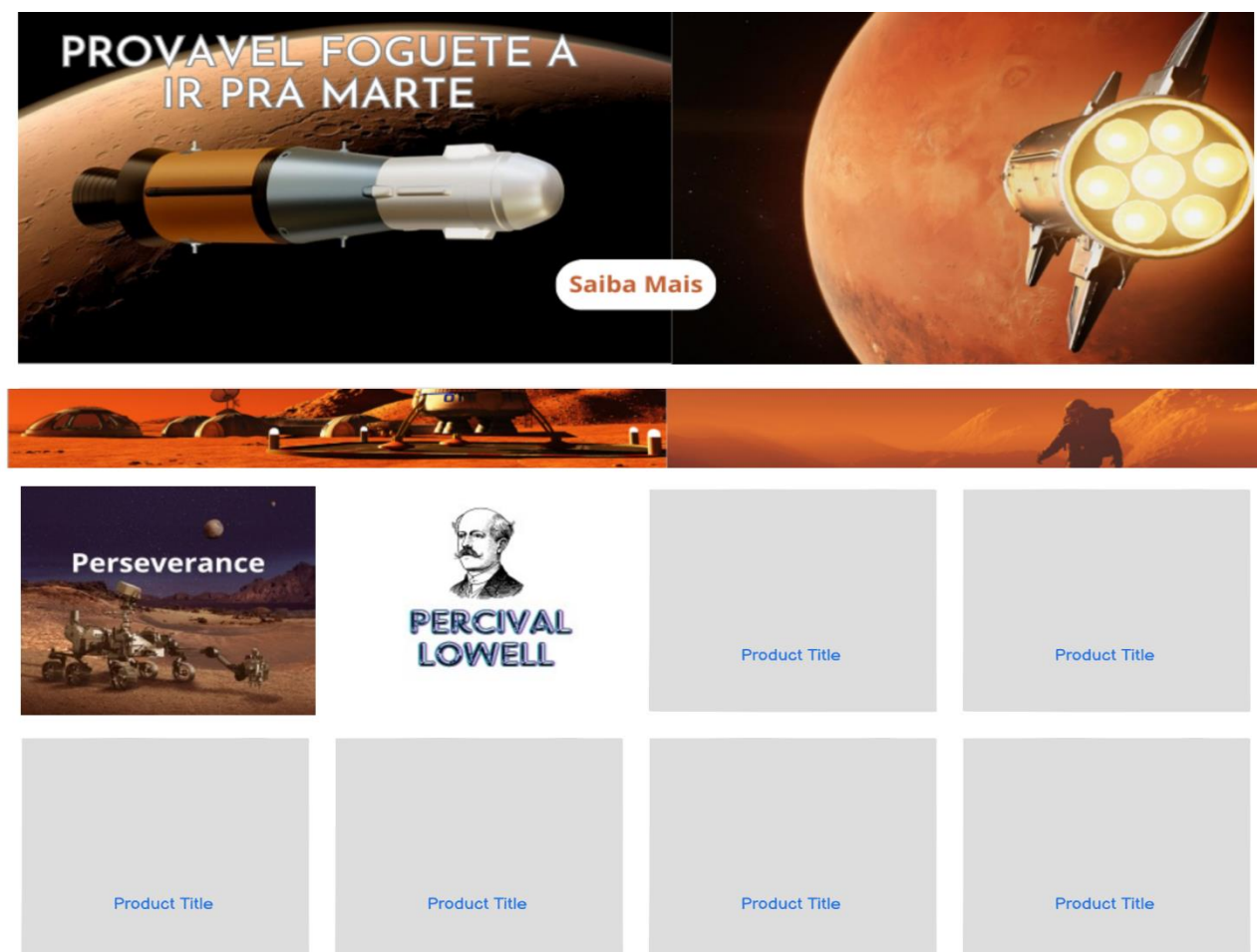
Esse protótipo mostra uma página melhorada do site "Stay of Mars". No topo, há um fundo vermelho com quatro botões rotulados "Obras," "Mapas," "Quiz," e "Formulários". Abaixo desses botões, há um grande espaço reservado para uma imagem ilustrativa. Logo abaixo, há uma mensagem de boas-vindas e uma descrição do museu.

Versão 2.0:

Esta tela apresenta uma seção de informações sobre as obras relacionados a Marte. No espaço da imagem colocamos um logo interativo do museu, mudamos a cor do topo para laranja, deixamos o fundo branco, e removemos a parte visual dos botões no topo e adicionamos interatividade neles, e permaneceu um texto autoexplicativo, e foi colocado feedback ao invés de formulários.

Foi realizado um protótipo das obras para o desenvolvimento das telas, conforme na figura 17.

Figura 17 - Obras



Autor (Auto própria)

Versão 1.0

O protótipo é colocar ilustrações das obras em quadros.

Versão 2.0

Foi adicionado um fundo interativo de marte, e as obras estão em formato de texto e imagens, e foi adicionado imagens para o usuário clicar e poder ver obras em 3D.

Foi realizado um protótipo do quiz para o desenvolvimento das telas, conforme na figura 18.

Figura 18 - Quiz

CATEGORIA

☒ Homem
 ☐ Mulher
 ☐ Criança
 ☐ Idoso

Qual das seguintes descobertas feitas por sondas e rovers em Marte sugere que o planeta poderia ter tido condições favoráveis à vida no passado?

☐ Presença de grandes quantidades de gás oxigênio na atmosfera.
 ☐ Detecção de moléculas orgânicas no solo
 ☐ Formação de grandes tempestades de poeira com alta frequência
 ☒ Presença de uma camada espessa de gelo no polo sul

Qual é a posição de Marte no sistema solar em relação ao Sol?

☐ Primeiro Planeta a partir do sol.
 ☐ Segundo planeta a partir do sol.
 ☐ Nenhuma alternativa.
 ☒ Marte é o quarto planeta a partir do Sol.

Por que Marte é conhecido como o "Planeta Vermelho"?

☐ devido à sua aparência avermelhada, causada pela presença de CO₂ em sua superfície.
 ☐ Por causa do reflexo do sol.
 ☒ Marte é chamado de "Planeta Vermelho" devido à sua aparência avermelhada, causada pela presença de óxido de ferro (ferrugem) em sua superfície.
 ☐ Nenhuma alternativa.

Resultado

Feedback

Sair

Autor (Auto própria)

Versão 1.0

O protótipo foi colocado apenas perguntas sem nenhum tipo visual ilustrativo, o esboço da ideia é apresentar o resultado.

Versão 2.0

Foi adicionado um Container envolvendo as perguntas dentro dele, com um fundo de um viajante de marte, os botões das perguntas são interativos e a cor laranja, e no final de responder uma pergunta aparece uma informação indicando para responde a próxima, e ao finalizar o quis aparecer a porcentagem de acertos.

Foi realizado um protótipo do relatório para o desenvolvimento das telas, conforme na figura 19.

Figura 19 - Relatório



Autor (Auto própria)

Versão 1.0: Este protótipo mostra um painel de relatórios com um fundo vermelho. No topo, há um cabeçalho com a palavra “Relatórios” e um ícone de seta para trás. Abaixo do cabeçalho, o título “Relatórios” é exibido.

O painel é dividido em três seções: “porcentagem de acertos”, “Média de satisfação” e “Comentários”. No rodapé, há um botão cinza com o texto “Início”, para voltar à página inicial do museu.

Versão 2.0

Foi alterado o fundo para cinza, destacando o nome, feedback e o nível de satisfação, e o texto “Relatório” foi alterado para “Feedback do visitante”.

Foi realizado um protótipo do feedback para o desenvolvimento das telas, conforme na figura 20.

Figura 20 - Feedback

Nome de usuário

Feedback

Nível de satisfação

Versão 1.0

O protótipo se iniciou para o usuário conseguir se expressar, com informações diretas e caixas de texto simples, com um fundo branco.

Versão 2.0

Foi adicionado uma imagem interativa de marte, foi aumentado a caixa de texto do feedback, e adicionado estrelas em níveis de satisfação.

10. CUSTO

(O custo total do projeto foi R\$30.000).

Para cobrir:

- Monitoramento e Vigilância;
- Fontes de Energia;
- Manutenção Preventiva;
- Proteções Físicas.

11. DIAGRAMAS

O Diagrama de Entidade-Relacionamento (DER) e o Modelo de Entidade-Relacionamento (MER) são ferramentas utilizadas no projeto de banco de dados para representar a estrutura e os relacionamentos entre os dados de um sistema. O DER é uma representação gráfica que mostra as entidades (objetos ou conceitos sobre os quais deseja-se armazenar informações) e os relacionamentos entre elas. As entidades são representadas por retângulos e os relacionamentos por linhas que conectam as entidades, indicando a natureza e a cardinalidade desses relacionamentos.

Por outro lado, o MER é uma representação conceitual mais abstrata, que descreve as entidades, seus atributos e os relacionamentos entre elas de forma mais detalhada. No MER, as entidades são representadas por retângulos com seus atributos listados dentro deles, e os relacionamentos são representados por linhas conectando as entidades, com indicações de cardinalidade. Ambos os diagramas são importantes para o projeto de banco de dados, pois ajudam a visualizar e compreender a estrutura e os relacionamentos dos dados de um sistema. O DER é geralmente utilizado na fase inicial do projeto, enquanto o MER é mais detalhado e pode ser utilizado em fases posteriores do desenvolvimento do banco de dados.

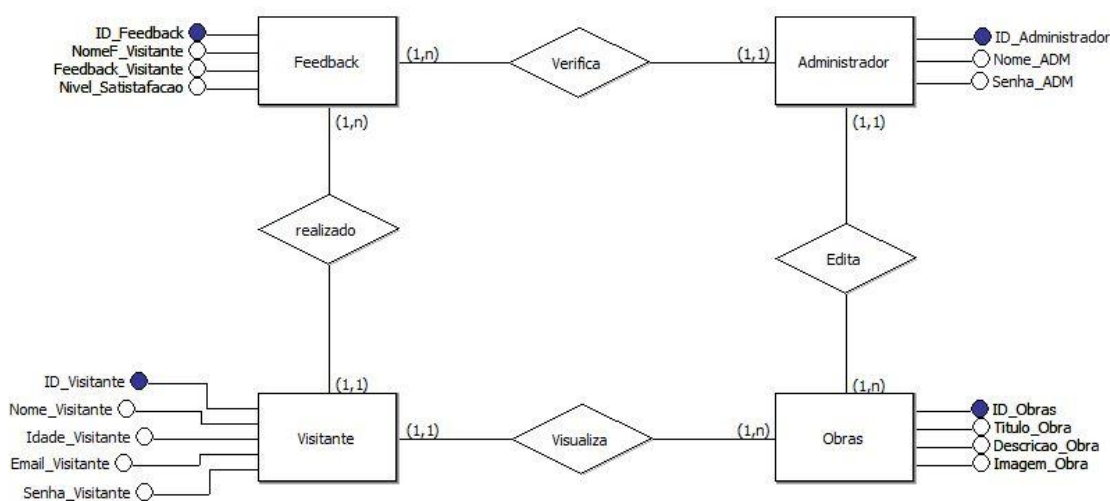
11.1 Diagrama MER

Diagrama (MER) de um sistema de feedback para visitantes de um Museu. No diagrama, temos quatro entidades principais: Visitante, Feedback, Administrador e Obras. Cada uma delas é representada por um retângulo e possui atributos específicos, mostrados em círculos.

11.2 Diagrama DER

A imagem abaixo refere-se ao modelo DER do banco de dados onde será guardado as informações necessárias para contemplar o funcionamento do nosso totem (visitante, relatório, feedback).

Figura 21 - Diagrama DER



Autor: Autoria própria

12. ANÁLISE DE REQUISITOS

A análise de requisitos é um processo fundamental no desenvolvimento de software, onde são identificadas, documentadas e verificadas as necessidades e expectativas dos usuários e outras partes interessadas em relação ao sistema a ser desenvolvido. Essa análise visa entender e definir claramente o que o software deve fazer, quais problemas deve resolver e quais as características são necessárias para atender às demandas dos usuários.

Em resumo, a análise de requisitos significa compreender profundamente o contexto de um projeto de software, identificar e documentar todas as funcionalidades, restrições e objetivos do sistema, garantindo que as necessidades dos usuários sejam atendidas de forma eficaz e que o produto seja desenvolvido conforme o esperado.

Além de identificar e documentar os requisitos, a análise de requisitos também envolve a validação e verificação desses requisitos. Isso significa garantir que os requisitos capturados são precisos, consistentes, completos e compreensíveis. Para isso, são realizadas revisões, entrevistas com os stakeholders, prototipagem e outras técnicas para garantir que todas as necessidades e expectativas sejam corretamente entendidas e representadas.

A análise de requisitos é uma etapa crítica no ciclo de vida do desenvolvimento de software, pois fornece a base para todas as atividades subsequentes, desde o design e implementação até o teste e manutenção. Uma análise de requisitos bem-feita resulta em um software que atende às necessidades reais dos usuários, minimizando retrabalhos e garantindo a satisfação do cliente. Por outro lado, uma análise inadequada ou incompleta pode levar a falhas no produto final, custos adicionais e insatisfação dos usuários, conforme mostrado nas tabelas abaixo:

12.1 Requisitos de Alto Nível

Tabela 3 – Requisitos de alto nível

Atingir a fácil interatividade do usuário ao sistema.	
Acessibilidade	Garantir que o sistema seja acessível a qualquer faixa etária.
Interatividade	Fornecer uma experiência de usuário envolvente e interativa
Escalabilidade	Permitir futuras expansões sem necessidade de grandes reformulações.

Autor: Autoria propria

12.2 Características do produto.

Tabela 4 – Características do produto

Características do produto	Descrição
Manutenibilidade	O software deve ser projetado para permitir a fácil evolução, permitindo a adição de novas funcionalidades e adaptação a mudanças de novas necessidades dos visitantes.
Confiança e proteção	O software deve incluir confiabilidade, proteção e segurança. Um software deve ser desenvolvido para evitar qualquer prejuízo físico ou econômico.
Eficiência	Inclui a capacidade de resposta, tempo de processamento, o uso de memória etc.
	O software deve ser aceitável ao usuário, o que significa que deve ser compreensível uma interatividade onde ele não se perca com as informações, e deixar claro se é compatível.

Autor: Autoria propria

12.3 Análise de riscos.

A análise de riscos é um processo sistemático utilizado para identificar, avaliar e gerenciar riscos potenciais que possam ter um impacto negativo em um projeto, operação ou organização. Esse processo é essencial para a tomada de decisões e para a implementação de estratégias de mitigação eficazes, ou seja, minimizar.

12.4 Planejamento.

No planejamento, busca – se registrar os métodos que serão empregados para administrar os riscos, definindo as estratégias utilizando ferramentas para a medição de riscos.

12.5 Identificação.

Já na identificação de riscos, a prioridade é à detecção de possíveis riscos com o auxílio da ferramenta Power BI, pois ela oferece diversas funcionalidades que facilitam uma análise e auxilia a identificação. Como o Power BI permite a integração de dados de múltiplas fontes onde possibilita uma visão abrangente e detalhada dos riscos potenciais. Sua grande vantagem é seus recursos de visualização interativa e é possível criar dashboards e relatórios dinâmicos que destacam áreas de preocupação e tendências de emergência. OBS. os dados são importados pela planilha do Excel.

Tabela 5 probabilidades

Probabilidade	1	2	3	4	5
1	5	10	15	20	25
2	4	8	12	16	20
3	3	6	9	12	15
4	2	4	6	8	10
5	1	2	3	4	5
Impacto	1	2	3	4	5

Autor: Autoria propria

Níveis de probabilidade:

- 5 Altamente provável;
- 4 Muito provável;
- 3 Provável;
- 2 Pouco improvável;
- 1 Improvável.

Níveis de impacto:

- 5 Crítico. (Perda total);
- 4 Sério. (Grande impacto);
- 3 Moderado (Impacto moderado);
- 2 Menor. (Pequeno impacto);
- 1 Insignificante. (Nenhum impacto).

12.6 Análise de risco do museu.

Para realização do documento do totem, foi realizado um levantamento detectando possíveis ameaças durante o processo de implantação, utilizando indicadores de detecção de riscos. A detecção de riscos foi analisada e priorizada, permitindo a implementação de medidas de mitigação eficazes para garantir a segurança e a funcionalidade dos equipamentos no museu. Foram identificados riscos específicos para cada tipo de equipamento, incluindo switches, roteadores, totens e câmeras de segurança. Conforme e mostrada nas imagens a seguir (Figura 19, 20).

O mesmo para a elaboração do documento sobre o sistema web, onde foi realizado um levantamento identificando possíveis ameaças durante o processo de implantação, utilizando indicadores de detecção de riscos. Esses riscos foram analisados e priorizados, permitindo a implementação de medidas de mitigação eficazes para garantir a segurança e a funcionalidade.

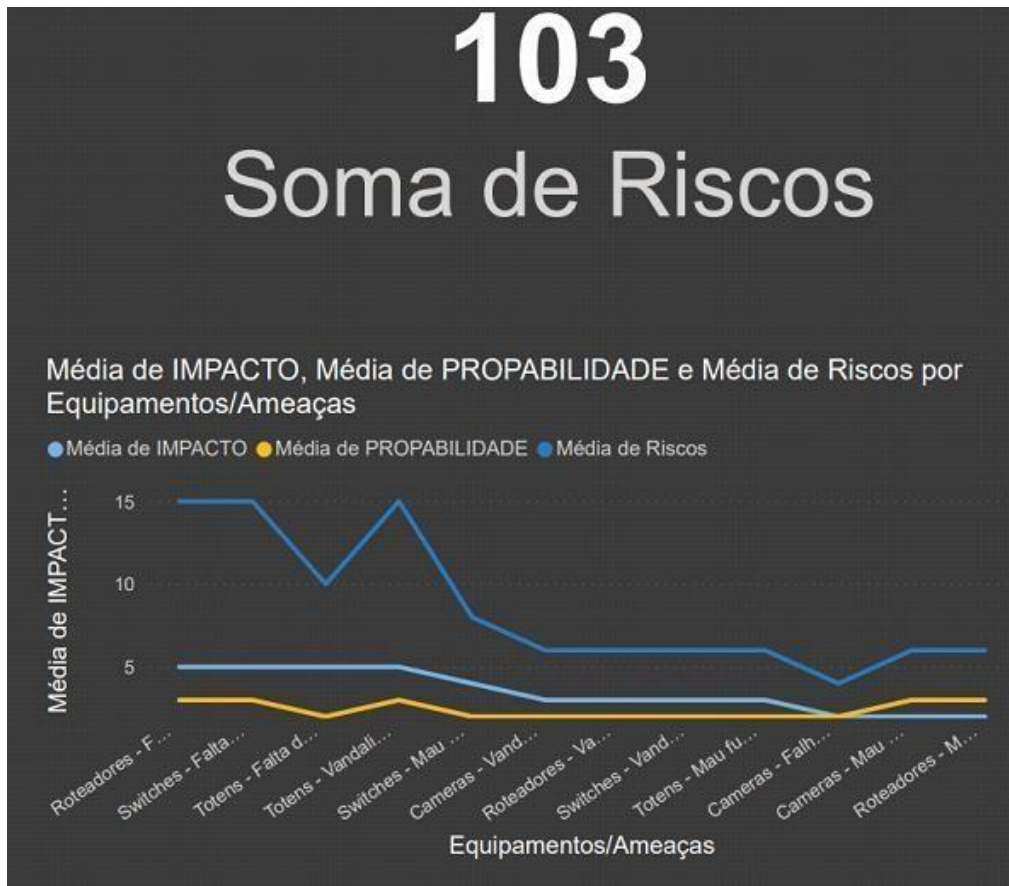
Esses riscos foram analisados e priorizados, permitindo a implementação de medidas de mitigação eficazes para garantir a segurança e a funcionalidade do totem, conforme nas figuras 22, 23.

Figura 22 – Matriz

Equipamentos/Ameaças	Soma de PROPABILIDADE	Soma de IMPACTO	Soma de Riscos
Totens - Vandalismo	3	5	15
Totens - Mau funcionamento	2	3	6
Totens - Falta de energia	2	5	10
Switches - Vandalismo	2	3	6
Switches - Mau funcionamento	2	4	8
Switches - Falta de energia	3	5	15
Roteadores - Vandalismo	2	3	6
Roteadores - Mau funcionamento	3	2	6
Roteadores - Falha de energia	3	5	15
Cameras - Vandalismo	2	3	6
Cameras - Mau funcionamento	3	2	6
Cameras - Falha de Energia	2	2	4
Total	29	42	103

Autor: Autoria própria

Figura 23 - Riscos



Autor: Autoria própria

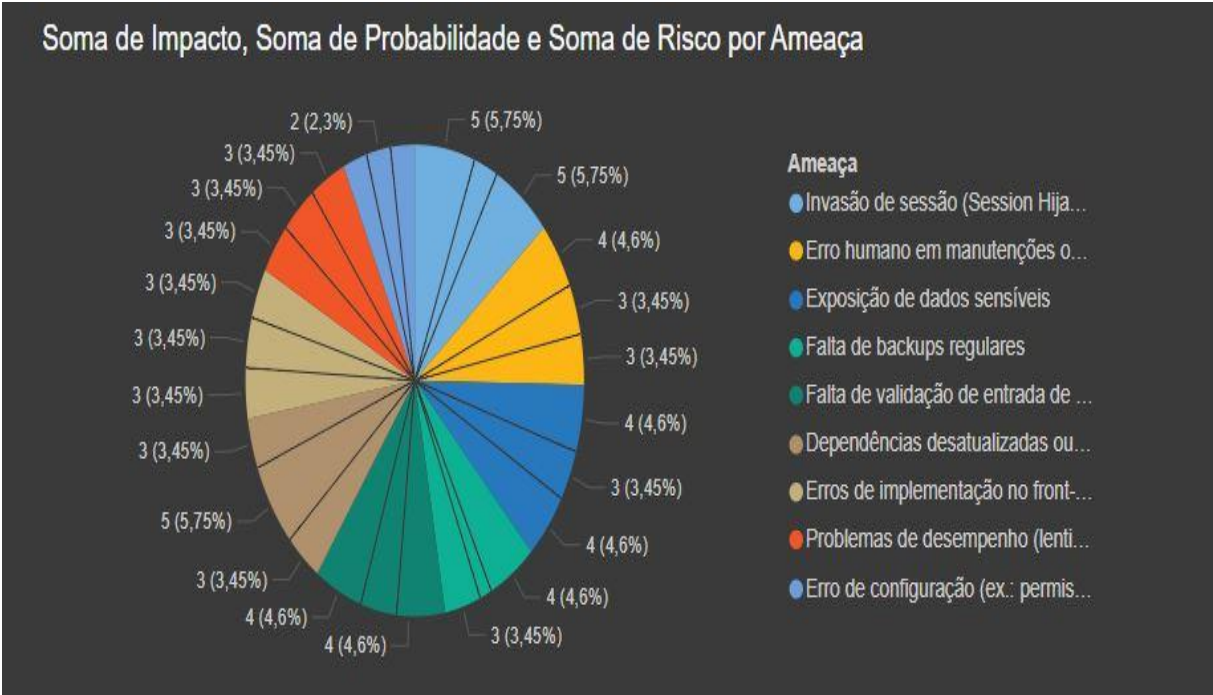
Esses riscos foram analisados e priorizados, permitindo a implementação de medidas de mitigação eficazes para garantir a segurança e a funcionalidade do sistema web, conforme nas figuras 24, 25.

Figura 24 – Matriz (WEB)

Risco			
Ameaça ▼	Soma de Impacto	Soma de Probabilidade	Soma de Risco
Problemas de desempenho (lentidão, falta de escalabilidade)	3	3	3
Invasão de sessão (Session Hijacking)	5	2	5
Falta de validação de entrada de dados	4	3	4
Falta de backups regulares	4	1	3
Exposição de dados sensíveis	4	3	4
Erros de implementação no front-end (ex.: vazamento de tokens em JavaScript)	3	3	3
Erro humano em manutenções ou atualizações	4	3	3
Erro de configuração (ex.: permissões incorretas em arquivos)	2	2	2
Dependências desatualizadas ou vulneráveis	3	5	3
Total	32	25	30

Autor: Autoria própria

Figura 25 – Riscos (WEB)



Autor: Autoria própria

12.7 Análise qualitativa de risco.

Verificar quais são os principais e qual o efeito de cada um, caso venha ocorrer se tornando um problema.

12.8 Análise quantitativa de risco.

Os riscos foram verificados por probabilidades e impactos, analisando todos os possíveis riscos que podem acontecer.

12.9 Respostas

Desenvolver estratégias e técnicas que irão reduzir as ameaças de riscos.

- Aceitar;
- Prevenir;
- Reduzir;
- Delegar.

Aceitar: Desenvolver uma contingência para quando o problema vier a ocorrer.

Prevenir: Nada mais é do que prevenir a probabilidade que o risco ocorra, ou tendo uma proteção segura contra o impacto desse risco.

Reduzir: Tomar ações específicas para a redução do risco, diminuindo o impacto e a probabilidade de acontecer.

Delegar: Realizar a transferência total ou parcial desse risco deixando a ameaça e o impacto por conta de outra parte, um exemplo o Seguro.

12.10 Controle

Segundo (o GuiaPMBOK7 PA: Project Management Institute, 2021). “Monitoramento e o controle do risco é o processo de identificar e de assegurar o controle do risco, monitorando riscos residuais e identificando novos riscos, assegurando a execução dos planos do risco e avaliando sua eficiência na redução dos riscos”.

- Estabelecer exames ou avaliações;
- Execução;
- Plano de emergência;
- Atualizar o plano de contingência.

12.11 Mitigação de riscos. (web)

Erro de configuração (permissões incorretas em arquivos):

Realizar auditorias regulares de configurações de permissões.

Implementar controles de acesso rigorosos e automatizar testes de configurações.

Dependências desatualizadas ou vulneráveis:

Implementar um processo de gerenciamento de dependências, garantindo que todas as bibliotecas e frameworks sejam atualizados.

Utilizar ferramentas como OWASP Dependency-Check para identificar vulnerabilidades conhecidas.

Falta de backups regulares:

Estabelecer uma política de backups automáticos e periódicos.

Armazenar backups de forma segura e garantir que eles sejam facilmente restauráveis.

Erros de implementação no front-end (vazamento de tokens):

Garantir que tokens sensíveis não sejam armazenados no lado do cliente. Usar métodos de criptografia para dados sensíveis e tokens.

Problemas de desempenho:

Realizar testes de carga e identificar gargalos de desempenho. Implementar soluções de escalabilidade, como balanceamento de carga e otimização de consultas de banco de dados.

Erro humano em manutenções ou atualizações:

Adotar práticas de DevOps para garantir testes automatizados e revisões de código rigorosas. Implementar backups e rollback de versões de forma eficiente.

Falta de validação de entrada de dados:

Validar e sanitizar todas as entradas de dados, tanto do lado do cliente quanto do servidor. Utilizar bibliotecas e frameworks que ajudem a evitar injeções de SQL e XSS.

Exposição de dados sensíveis:

Criptografar dados sensíveis, tanto em trânsito quanto em repouso. Utilizar políticas rigorosas de acesso e controle de dados.

Invasão de sessão (Session Hijacking):

Implementar autenticação baseada em tokens seguros (JWT). Forçar a utilização de HTTPS em todas as requisições e garantir que os cookies de sessão sejam protegidos.

12.12 Plano de continuidade. (desktop)

Nada mais é que estratégias abordadas quando esses riscos vêm a ocorrer, foi separado em partes, para garantir a continuidade das operações e a recuperação rápida em caso de interrupções ou desastres.

12.13 Proteções Físicas

Instalações de proteções ao redor dos switches, roteadores e totens para reduzir o risco de vandalismo e danos acidentais.

12.14 Manutenção Preventiva

Programando manutenções regulares para todos os equipamentos, garantindo que continuem funcionando bem e por mais tempo.

12.15 Fontes de Energia

Implementando fontes de energia redundantes e sistemas de backup para os roteadores e switches. Isso assegura que a rede do museu continue operando mesmo em caso de falhas de energia ou problemas com o hardware, minimizando qualquer tempo de inatividade.

12.16 Monitoramento e Vigilância

Instaladas câmeras de segurança extras em áreas críticas para prevenir vandalismo e permitir uma resposta rápida a qualquer incidente. Essas ações são fundamentais para manter o totem seguro e em pleno funcionamento, mesmo diante de imprevistos. Pois temos soluções de emergências prontas.

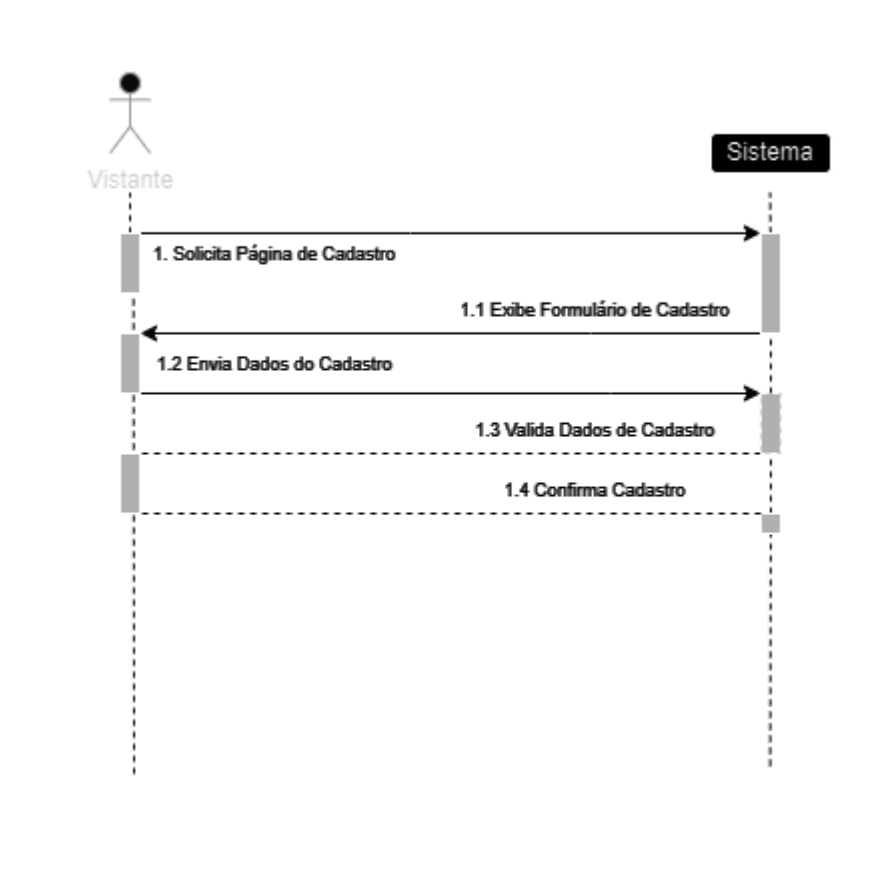
13. DIAGRAMAS DE CLASSES

representação estática utilizada na área da programação para descrever a estrutura de um sistema, apresentando suas classes, atributos, operações e as relações entre os objetos.

13.1 Diagrama Sequencia cadastro

Cadastro O visitante acessa a página de cadastro para criar uma conta. Ele preenche um formulário com informações como nome, e-mail e senha. Após revisar os dados inseridos, o visitante envia o formulário. O sistema processa as informações, realiza as validações necessárias e, se tudo estiver correto, confirma a criação do cadastro, exibindo uma mensagem de sucesso na tela.

Figura 26 – Diagrama de sequencia cadastro

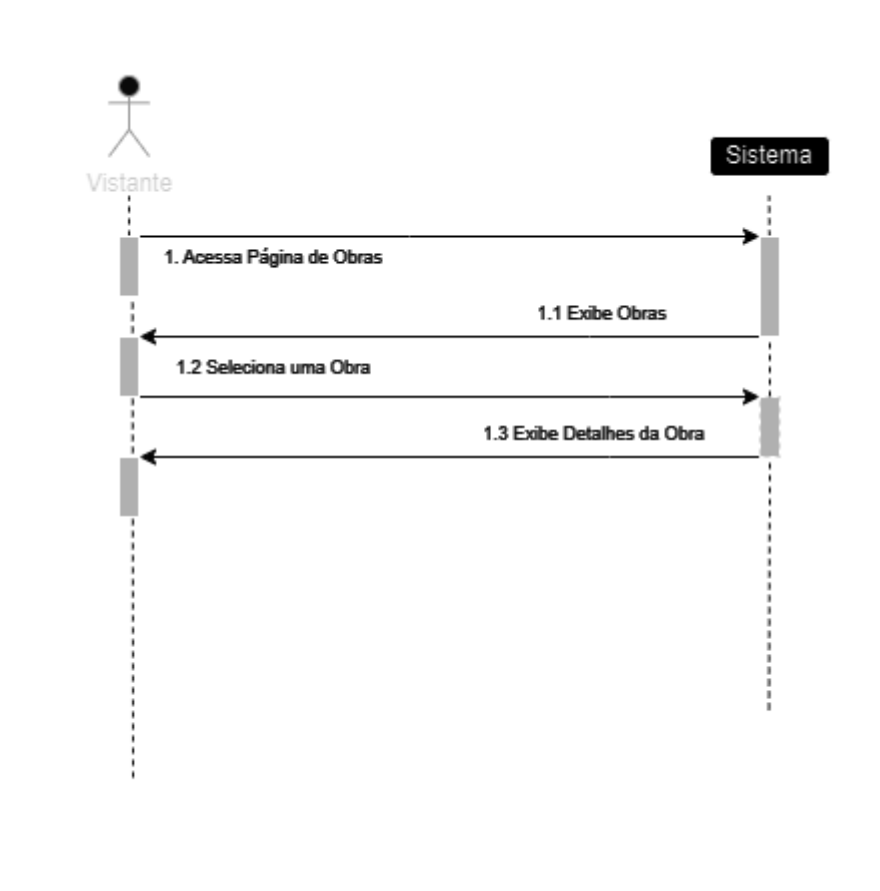


Autor: Autoria própria

13.2 Diagramas Sequencia Obras

Obras O visitante navega até a página de obras, onde são listadas diversas opções de peças e artefatos. Ele pode explorar as obras disponíveis, utilizando filtros e categorias para facilitar a busca. Ao selecionar uma obra de interesse, o visitante é direcionado para uma página de detalhes, onde pode visualizar informações como título, autor, descrição, data de criação e outras curiosidades relacionadas à obra escolhida.

Figura 27 – Diagrama de sequência Obras

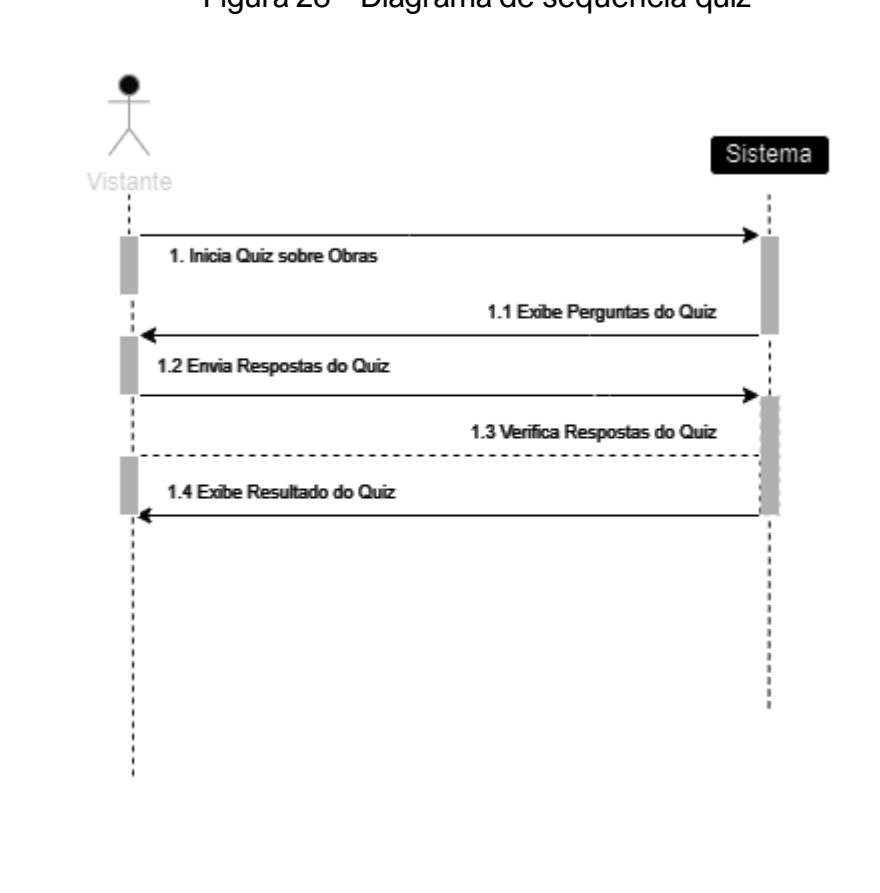


Autor: Autoria própria

13.3 Diagramas Sequencia Quiz

Quiz O visitante inicia um quiz interativo relacionado às obras ou temas do sistema. Ele responde a uma série de perguntas de múltipla escolha, uma de cada vez, dentro de um tempo determinado (se houver). Ao finalizar o quiz, as respostas são enviadas para o sistema, que calcula a pontuação e apresenta o resultado, indicando o desempenho do visitante, juntamente com um feedback detalhado sobre as respostas corretas e incorretas.

Figura 28 – Diagrama de sequência quiz

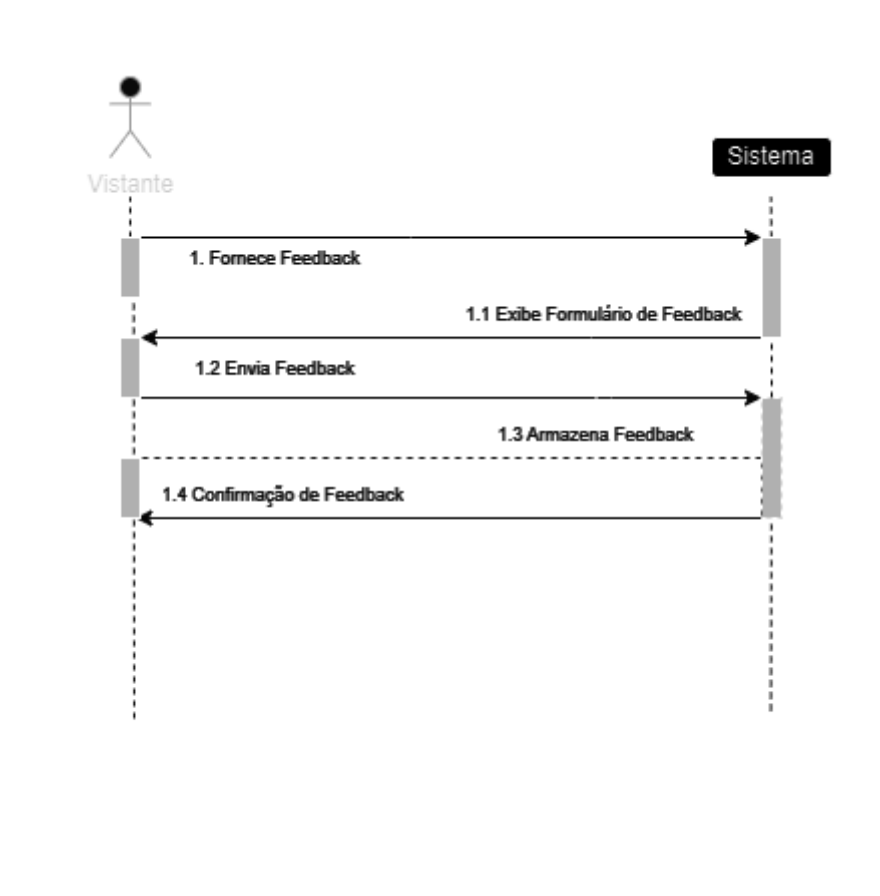


Autor: Autoria própria

13.4 Diagramas Sequencia Feedback

Feedback O visitante acessa a seção de feedback para compartilhar suas opiniões e sugestões sobre a experiência com o sistema ou as obras visualizadas. Ele preenche um formulário, onde pode incluir comentários, críticas construtivas ou elogios. Após o envio, o sistema confirma que o feedback foi recebido com sucesso, agradecendo ao visitante pela sua contribuição, que será analisada pela equipe de administração.

Figura 29 – Diagrama de sequência feedback

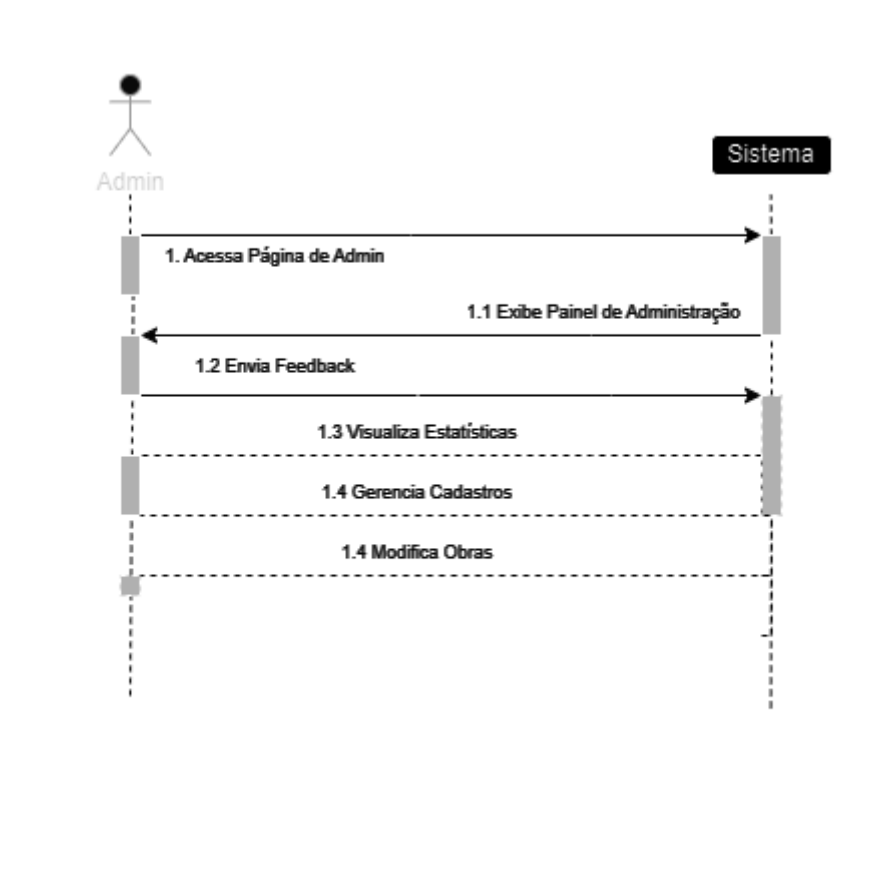


Autor: Autoria própria

13.5 Diagramas Sequencia Administrador

O administrador tem acesso exclusivo à área de gestão de feedbacks. Ele pode visualizar uma lista completa de todos os feedbacks enviados pelos visitantes, organizados por data e relevância. Ao selecionar um feedback específico, o administrador pode acessar detalhes adicionais, revisar os comentários deixados pelos visitantes e tomar ações apropriadas, como responder ao feedback ou realizar melhorias no sistema com base nas sugestões recebidas.

Figura 30 – Diagrama de sequência administrador



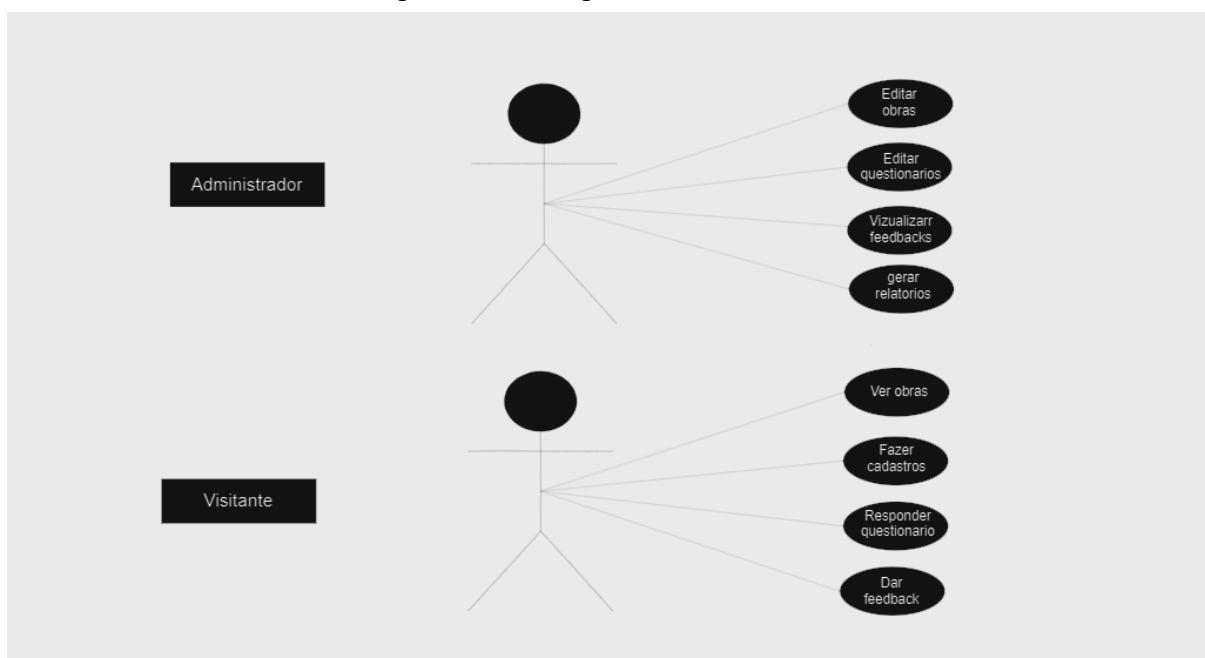
Autor: Autoria própria

13.6 Diagramas de casos uso

Para o Administrador: Editar Obras: Permite adicionar, modificar ou remover informações das obras disponíveis. Editar Questionários: Dá ao administrador a capacidade de criar, atualizar ou excluir quizzes. Visualizar Feedbacks: Permite revisar os feedbacks enviados pelos visitantes. Gerar Relatórios: Possibilita a geração de relatórios analíticos baseados em dados do sistema.

Para o Visitante: Ver Obras: O visitante pode navegar pelas obras e visualizar seus detalhes. Fazer Cadastros: Permite que o visitante se registre no sistema. Responder Questionário: O visitante pode participar de quizzes relacionados ao conteúdo. Dar Feedback: Possibilita ao visitante enviar suas opiniões e sugestões sobre sua experiência.

Figura 31 – Diagrama de casos uso



Autor: Autoria própria

14. CRONOGRAMAS

um documento que descreve todas as tarefas a serem cumpridas dentro de um determinado planejamento do projeto. Atua como um mapa detalhado, delineando o caminho que sua equipe deve seguir para atingir os objetivos estipulados.

14.1 Cronogramas primeiro mês

O primeiro mês do projeto web marcou o início de uma jornada detalhada e organizada, estruturando as bases para o desenvolvimento robusto do sistema. Este mês foi caracterizado pela fase de planejamento e pela criação dos primeiros artefatos essenciais do projeto. Este primeiro mês foi fundamental para estabelecer uma base sólida para o projeto, garantindo que todas as partes interessadas estivessem alinhadas e que a equipe tivesse uma compreensão clara das metas e responsabilidades. As reuniões frequentes e a documentação detalhada foram cruciais para o sucesso contínuo do desenvolvimento. Conforme na figura 32 abaixo:

Figura 32 – Cronograma primeiro mês (Autor: Autoria própria)

NOME DA TAREFA	INICIO	TERMINO
PRIMEIRA SEMANA - Planejamento		
Reunião da equipe (Definição de tarefas - Wilgner)	Seg 05/08/2024	Seg 05/08/2024
Pesquisa de metodologia (Gustavo)	Ter 06/08/2024	Ter 06/08/2024
Reunião de planejamento com os stakeholders (Grupo)	Qua 07/08/2024	Qua 07/08/2024
Planejamento de documentação (Gustavo, Julia)	Qui 08/08/2024	Qui 08/08/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 09/08/2024	Sex 09/08/2024
SEGUNDA SEMANA - Criação		
Reunião da equipe (Definição de tarefas - Wilgner)	Seg 12/08/2024	Seg 12/08/2024
Iniciamento de diagrama Uso (Murilo, Nicolas)	Ter 13/08/2024	Ter 13/08/2024
Criação de algoritmo (Matheus, Wilgner)	Qua 14/08/2024	Qua 14/08/2024
Iniciamento da documentação (Gustavo, Julia)	Qui 15/08/2024	Qui 15/08/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 16/08/2024	Sex 16/08/2024
TERCEIRA SEMANA - Desenvolvimento		
Reunião da equipe (Definição de tarefas - Wilgner)	Seg 19/08/2024	Seg 19/08/2024
Desenvolvimento de diagramas (Murilo, Nicolas)	Ter 20/08/2024	Ter 20/08/2024
Entrega de sprint (Julia, Gustavo)	Qua 21/08/2024	Qua 21/08/2024
Iniciamento da pagina Web (Matheus, Wilgner, Gustavo)	Qui 22/08/2024	Qui 22/08/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 23/08/2024	Sex 23/08/2024
QUARTA SEMANA - Continuação		
Reunião da equipe (Definição de tarefas - Wilgner)	Seg 26/08/2024	Seg 26/08/2024
Criação do diagrama sequencial (Murilo, Nicolas)	Ter 27/08/2024	Ter 27/08/2024
Desenvolvimento pagina Web (Matheus, Wilgner, Gustavo)	Qua 28/08/2024	Qua 28/08/2024
Implementação das ISOS (Julia, Gustavo)	Qui 29/08/2024	Qui 29/08/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 30/08/2024	Sex 30/08/2024

14.2 Cronogramas segundo mês

No segundo mês do projeto, a equipe concentrou seus esforços na finalização do sistema web, garantindo que todas as funcionalidades fossem implementadas e funcionando conforme o esperado. O segundo mês foi marcado por um esforço concentrado da equipe para garantir que o sistema web estivesse totalmente funcional e pronto para a implantação. Através de uma série de reuniões, revisões, e ajustes finais, conseguimos concluir essa importante fase do projeto com sucesso. Conforme na figura 33 abaixo:

Figura 33 – Cronograma segundo mês

NOME DA TAREFA	INICIO	TERMINO
PRIMEIRA SEMANA - Revisão Geral		
Reunião da equipe (Definição de tarefas - Wilgner)	Seg 02/09/2024	Seg 02/09/2024
Reunião de revisão com os stakeholders (Grupo)	Ter 03/09/2024	Ter 03/09/2024
Revisão de tarefas geral (Grupo)	Qua 04/09/2024	Qua 04/09/2024
Revisão de tarefas geral (Grupo)	Qui 05/09/2024	Qui 05/09/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 06/09/2024	Sex 06/09/2024
SEGUNDA SEMANA - Reajuste de Tarefas		
Reunião da equipe (Definição de tarefas - Wilgner)	Seg 09/09/2024	Seg 09/09/2024
Reajuste de tarefas (Murilo, Julia, Gustavo)	Ter 10/09/2024	Ter 10/09/2024
Continuação da pagina Web (Matheus, Wilgner, Gustavo)	Qua 11/09/2024	Qua 11/09/2024
Desenvolvimento de telas (Wilgner)	Qui 12/09/2024	Qui 12/09/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 13/09/2024	Sex 13/09/2024
TERCEIRA SEMANA - Desenvolvimento		
Reunião da equipe (Definição de tarefas - Wilgner)	Seg 16/09/2024	Seg 16/09/2024
Pesquisa do museu (Julia, Murilo)	Ter 17/09/2024	Ter 17/09/2024
Desenvolvimento de programa (Matheus, Wilgner, Gustavo)	Qua 18/09/2024	Sex 18/09/2024
Desenvolvimento de telas (Wilgner)	Qui 19/09/2024	Qui 19/09/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 20/09/2024	Sex 20/09/2024
QUARTA SEMANA - Desenvolvimento		
Reunião da equipe (Definição de tarefas - Wilgner)	Seg 23/09/2024	Seg 23/09/2024
Criação de formulario (Gustavo)	Ter 24/09/2024	Ter 24/09/2024
Conclusão Web (Matheus, Wilgner, Gustavo)	Qua 25/09/2024	Qua 25/09/2024
Implementação do IHC (Murilo, Nicolas)	Qui 26/09/2024	Qui 26/09/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 27/09/2024	Sex 27/09/2024

Autor: Autoria própria

14.3 Cronogramas terceiro mês

Durante o terceiro mês do projeto, a equipe realizou o planejamento inicial, incluindo a definição de tarefas, a equipe deu início à codificação da versão desktop e realizou várias reuniões para discutir e revisar as atividades do mês anterior.

Houve revisões gerais com o grupo para assegurar a qualidade do trabalho entregue, e a equipe se reuniu com os stakeholders para apresentar os resultados e obter feedback. Conforme na figura 34 abaixo:

Figura 34 – Cronograma terceiro mês

NOME DA TAREFA	INICIO	TERMINO
PRIMEIRA SEMANA - Revisão Geral		
Reunião da equipe (Entrega de tarefas, Wilgner)	Seg 30/09/2024	Seg 30/09/2024
Revisão geral (Grupo)	Ter 01/10/2024	Ter 01/10/2024
Revisão geral (Grupo)	Qua 02/10/2024	Qua 02/10/2024
Desenvolvimento programa (Wilgner, Matheus)	Qui 03/10/2024	Qui 03/10/2024
Reunião da equipe (Entrega de tarefas, Wilgner)	Sex 04/10/2024	Sex 04/10/2024
SEGUNDA SEMANA - Reunião		
Reunião da equipe (Entrega de tarefas)	Seg 07/10/2024	Seg 07/10/2024
Conclusão (Grupo)	Ter 08/10/2024	Ter 08/10/2024
Reunião de conclusão com os stakeholders (Grupo)	Qua 09/10/2024	Qua 09/10/2024
Desenvolvimento programa (Wilgner, Matheus, Gustavo)	Qui 10/10/2024	Qui 10/10/2024
Reunião da equipe (Entrega de tarefas, Wilgner)	Sex 11/10/2024	Sex 11/10/2024
TERCEIRA SEMANA - Desenvolvimento		
Reunião da equipe (Definicação de tarefas - Wilgner)	Seg 14/10/2024	Seg 14/10/2024
Implementção Desktop (Matheus, Wilgner, Gustavo)	Ter 15/10/2024	Ter 15/10/2024
Criação diagrama de classe (Nicolas)	Qua 16/10/2024	Qua 16/10/2024
Teste de unidade (Julia, Wilgner)	Qui 17/10/2024	Qui 17/10/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 18/10/2024	Sex 18/10/2024
QUARTA SEMANA - Desenvolvimento		
Reunião da equipe (Definicação de tarefas - Wilgner)	Seg 21/10/2024	Seg 21/10/2024
Desenvolvimento desktop (Matheus, Wilgner, Gustavo)	Ter 22/10/2024	Ter 22/10/2024
Teste de integração (Gustavo, Julia)	Qua 23/10/2024	Qua 23/10/2024
Elaboração de perguntas para o quis (Murilo, Nicolas)	Qui 24/10/2024	Qui 24/10/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 25/10/2024	Sex 25/10/2024

Autor: Autoria própria

14.4 Cronogramas quarto mês

No quarto mês do projeto, a equipe se empenhou em garantir a entrega final e o sucesso completo do projeto.

Durante este mês, foi focado na finalização e entrega do projeto com o máximo de qualidade e atenção aos detalhes. As reuniões serviram para alinhar as últimas tarefas e garantir que todas as etapas fossem concluídas corretamente.

E a conclusão geral feita por todo o grupo, marcaram os últimos esforços antes da entrega do projeto, após a conclusão de todo projeto foi feita uma série de testes.

O quarto mês foi crucial para o fechamento com chave de ouro, culminando em um projeto bem-sucedido e conforme as expectativas dos clientes. Conforme na figura 35 mostrado abaixo:

Figura 35 – Cronograma quarto mês











NOME DA TAREFA	INICIO	TERMINO
PRIMEIRA SEMANA - Desenvolvimento		
Reunião da equipe (Definicação de tarefas - Wilgner)	Seg 28/10/2024	Seg 28/10/2024
Desing da versão desktop (Gustavo, Matheus, Wilgner)	Ter 29/10/2024	Ter 29/10/2024
Testes de automatização (Gustavo, Julia)	Qua 30/10/2024	Qua 30/10/2024
Implantação do diagrama de caso de uso (Murilo, Nicolas)	Qui 31/10/2024	Qui 31/10/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 01/11/2024	Sex 01/11/2024
SEGUNDA SEMANA - Testes		
Reunião da equipe (Definicação de tarefas - Wilgner)	Seg 04/11/2024	Seg 04/11/2024
Teste de Usabilidade (Murilo, Nicolas)	Ter 05/11/2024	Ter 05/11/2024
Revisão de todo conteudo feito (grupo)	Qua 06/11/2024	Qua 06/11/2024
Ajustes finais e testes da versão desktop (Gustavo, Julia)	Qui 07/11/2024	Qui 07/11/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 08/11/2024	Sex 08/11/2024
TERCEIRA SEMANA - Treinamento e Revisão		
Reunião da equipe (Definicação de tarefas - Wilgner)	Seg 11/11/2024	Seg 11/11/2024
Treinamento de suporte da web (grupo)	Ter 12/11/2024	Ter 12/11/2024
Guia de uso da pagina web (grupo)	Qua 13/11/2024	Qua 13/11/2024
Revisão geral (grupo)	Qui 14/11/2024	Qui 14/11/2024
Reunião da equipe (Entrega de tarefas - Wilgner)	Sex 15/11/2024	Sex 15/11/2024
QUARTA SEMANA - Conclusão e entrega		
Reunião da equipe (Entrega de tarefas, Wilgner)	Seg 18/11/2024	Seg 18/11/2024
ABTN (Julia)	Ter 19/11/2024	Ter 19/11/2024
Conclusão geral e entrega (Grupo)	Qua 20/11/2024	Qua 20/11/2024

Autor: Autoria própria

15. FERRAMENTAS UTILIZADAS.

A imagem abaixo é referente a algumas ferramentas que foram utilizadas pelos autores do projeto para a conclusão do totem e da web.

Figura 36 – ferramentas

Ferramentas utilizadas	Imagens
Excel	
Canva	
Power BI	
Balsamiq	
Git hub	
Flowgorithm	
Draw.io	
Discord	
Brmodelo	
Visual studio	

Autor: Autoria própria

16. CONCLUSÃO

O Desenvolvimento do Software para o Museu Espacial "Stay of Mars", marca uma importante evolução na forma como os visitantes interagem e absorvem informações durante sua visita. Nosso software utilizou também ferramentas nunca vista pela equipe, aprendemos a utilizar a plataforma conseguindo absorver 100% através de uma abordagem ágil e colaborativa, conseguimos criar uma interface elegante, fácil de entendimento e interativa, com bibliotecas utilizadas diretamente da Microsoft Visual Studio. Ao dividir o projeto em etapas menores e adaptáveis, pudemos garantir que o software atendesse às necessidades que foi nos passada em sala de aula, e atendesse a expectativa dos usuários com ele. Absorvemos o material de estudo do semestre passado e anterior, foram os fundamentos de Redes onde explicamos toda a Infraestrutura do museu, onde mostramos também a Topologia do museu. Sendo assim, foram aplicadas e feita uma manutenção onde se encontrava possíveis erros. Ao concluir normas de qualidade e segurança, asseguramos que o software seja confiável e recipiente, proporcionando uma visita tranquila e informativa para todos.

REFERENCIAS BIBLIOGRAFICA

Barcelos (2012). O modelo incremental como teoria para o processo orçamentário. https://assecor.org.br/wpfd_file/o-modelo-incremental-como-teoria-para-o-processo-orcamentario/ Acesso em: 10 set 2024

Sommerville, I.; & Graciano, N. (2011) - Engenharia de Software 9 ed. https://www.academia.edu/42787809/Ian_Sommerville_Engenharia_de_Software_9_ed . Acesso em: 19 set 2024

PRESSMAN, Roger S. *Engenharia de software: uma abordagem profissional*. 8. ed. São Paulo: McGraw-Hill, 2011.

PROJECT MANAGEMENT INSTITUTE. *Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK)*. 7. ed. Newtown Square, PA: Project Management Institute, 2021. Disponível em: https://www.trt7.jus.br/files/institucional/governanca_ti/processos/processo-trt7/TRT_Compartilhado/disciplines/8_PMBOK_Gerenciamento_Riscos_Projeto_D20C6DF7.html#:~:text=O%20gerenciamento%20dos%20riscos%20do,de%20riscos%20de%20um%20projeto. Acesso em: 16 out. 2024.

DATE, C. J. *Introdução a sistemas de bancos de dados*. 8. ed. São Paulo: Pearson Addison Wesley, 2003.

W3C. XML: Extensible Markup Language. Disponível em: <https://www.w3.org/XML/>. Acesso em: 18 out. 2024.

MSDN MICROSOFT. *Introdução ao Padrão MVC*. Disponível em: <https://docs.microsoft.com/pt-br/aspnet/mvc>. Acesso em: 18 out. 2024.

SERRADOR, P.; PINTO, J. K. *Métodos ágeis no desenvolvimento de software: uma análise das abordagens incrementais e suas adaptações*. *Research, Society and Development*, v. 9, n. 3, e133932419, 2020. Disponível em: <https://rsdjournal.org/>. Acesso em: 20 out. 2024.

Marques, Brena. Qualidade de Software: Uma Análise a partir dos Critérios da Norma ISO 9126 Enanpad, [6 a 10 de set.], 2008. Disponível em: https://arquivo.anpad.org.br/abrir_pdf.php?e=ODU5Mg==. Acesso em: 20 out. 2024.

ANEXO 1 - MANUAL GUIA AO USUARIO WEB

Tela inicial do site:

Ao acessar o site "Stay Of Mars," o usuário será direcionado automaticamente para a tela inicial. Na parte superior da página (header), encontra-se o nome do site e alguns botões de navegação: "Obras," "Quiz," e "Feedback," que permitem acesso rápido a diferentes áreas do site.

Além disso, no header, há botões para Cadastro e Login para novos usuários, e um botão Desconectar para usuários que desejam fazer logout.

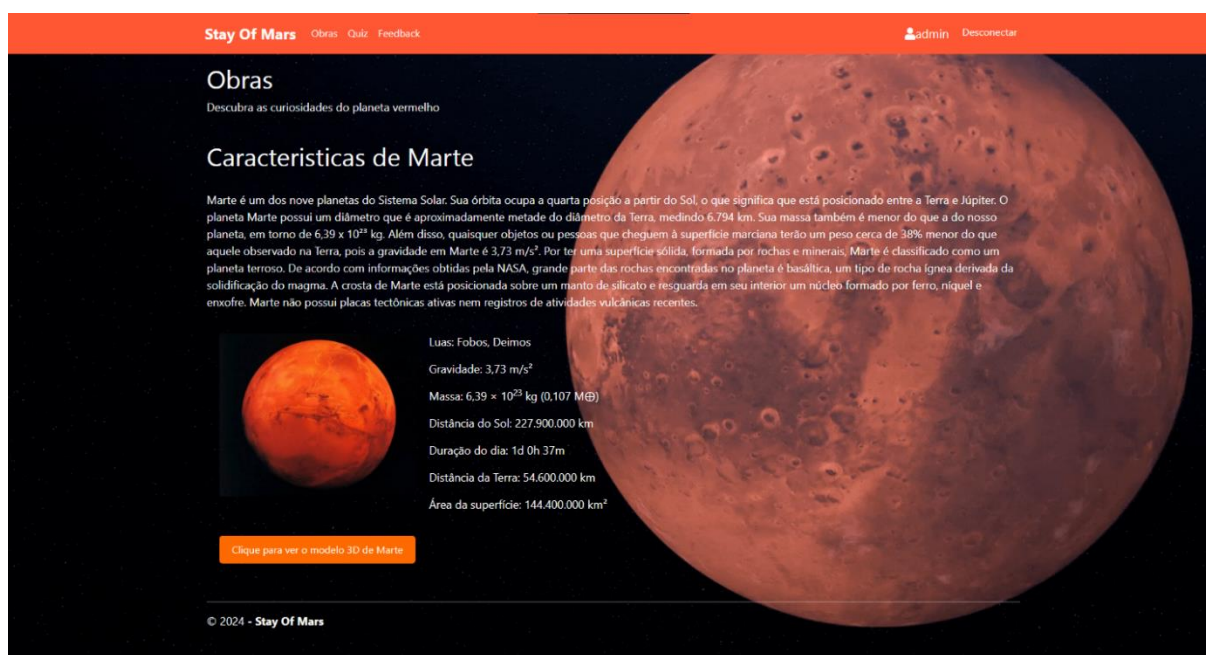
Logo abaixo do header, o usuário verá um GIF animado com o logotipo do museu, acompanhado por uma mensagem de boas-vindas, criando uma experiência inicial acolhedora e informativa, conforme mostrado na imagem abaixo.



Tela de obras do site:

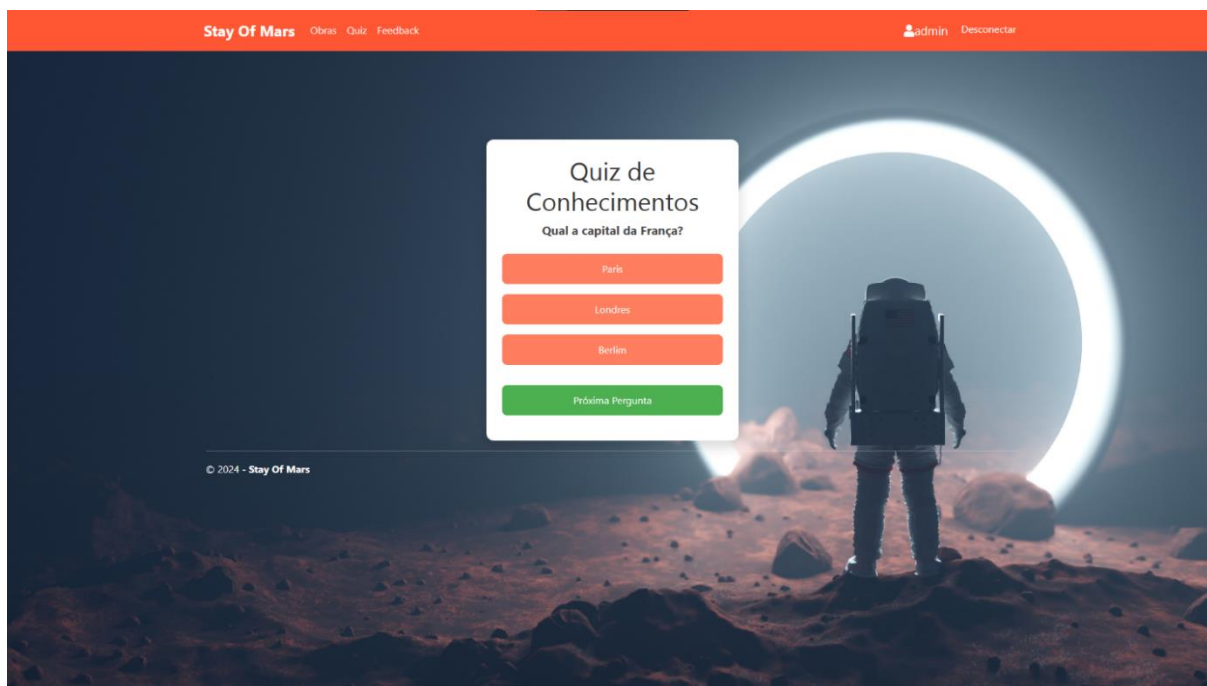
Ao clicar no botão Obras, o usuário será direcionado à seção que exibe as coleções do museu. Nessa página, encontrará um texto introdutório que apresenta o acervo do museu e fornece uma breve explicação.

Cada obra é exibida com um botão específico, que o usuário pode clicar para visualizar mais detalhes sobre aquela peça, conforme mostrado na imagem abaixo.

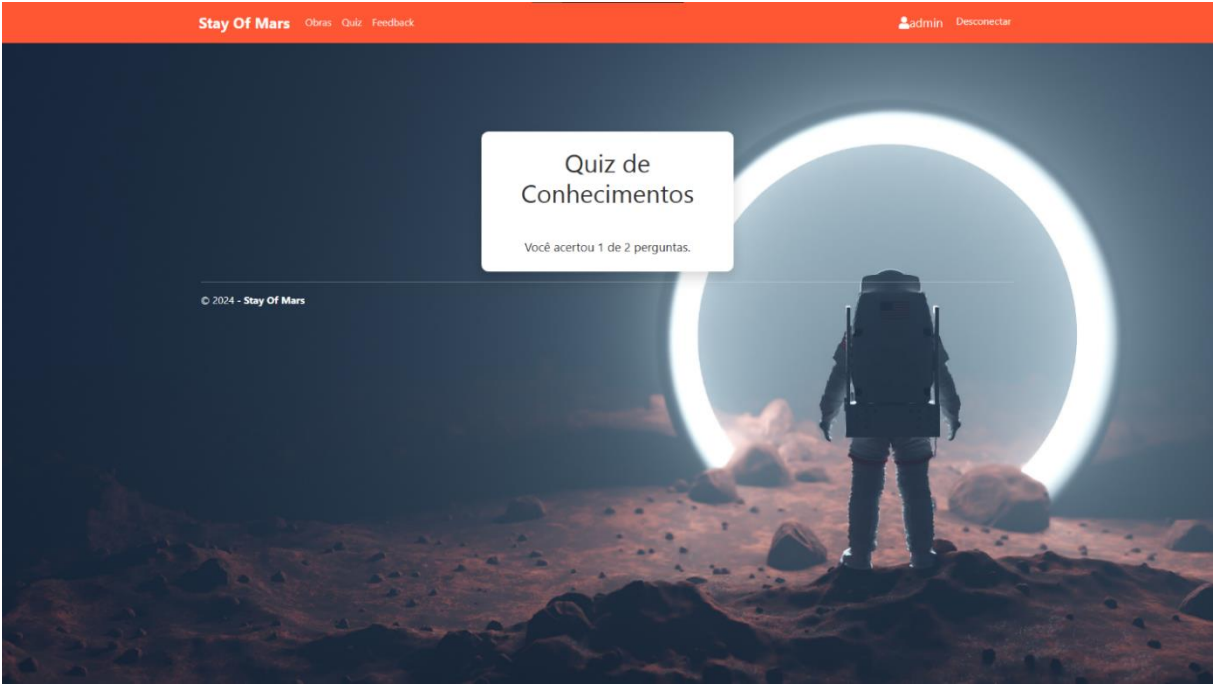


Tela do Quiz do site:

Ao clicar no botão Quiz, o usuário será direcionado a uma tela onde poderá testar seus conhecimentos sobre as obras apresentadas anteriormente. No centro da tela, serão exibidas as perguntas, cada uma com quatro alternativas para o usuário escolher, conforme mostrado na imagem abaixo.



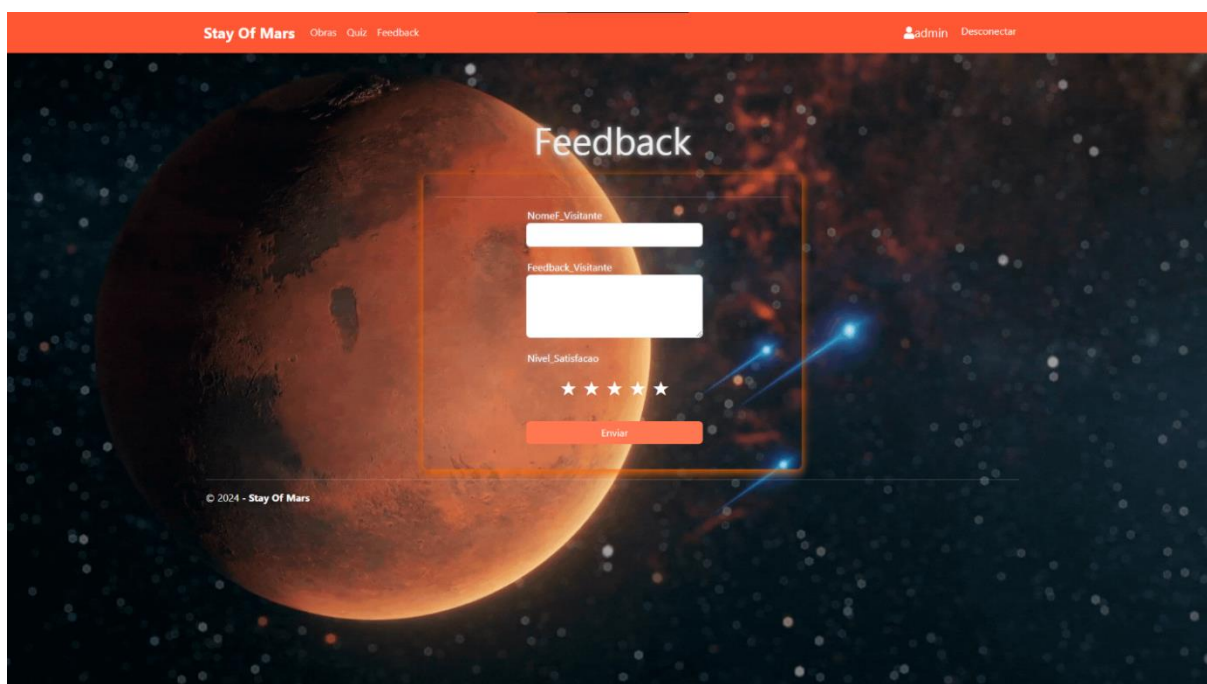
Após selecionar uma resposta, o usuário deve clicar no botão Próxima Pergunta para continuar. Ao finalizar o quiz, uma janela de mensagem (messagebox) exibirá o total de acertos, permitindo que o usuário veja seu desempenho, conforme mostrado na imagem abaixo.



Tela de Feedback:

Após concluir o quiz, o usuário será direcionado para a tela de Feedback, onde poderá compartilhar suas opiniões sobre o site e a experiência de navegação.

Nesta seção, há campos para inserir o Nome e o Feedback. O usuário pode deixar uma avaliação detalhada em forma de texto e atribuir uma pontuação utilizando estrelas para indicar seu nível de satisfação. Ao finalizar, basta clicar no botão “Enviar” para salvar o feedback, conforme mostrado na imagem abaixo.



The screenshot shows a web application titled "Stay Of Mars" with a navigation bar containing "Obras", "Quiz", and "Feedback". The user is logged in as "admin" and can click "Desconectar". The main content area is titled "Feedback" and features a form with the following elements:

- A text input field labeled "Nome Visitante".
- A text area labeled "Feedback Visitante".
- A star rating system labeled "Nível Satisfação" with five stars.
- A red button labeled "Enviar".

The background of the form is a high-quality image of the planet Mars against a starry space backdrop. A copyright notice "© 2024 - Stay Of Mars" is visible in the bottom left corner of the page.

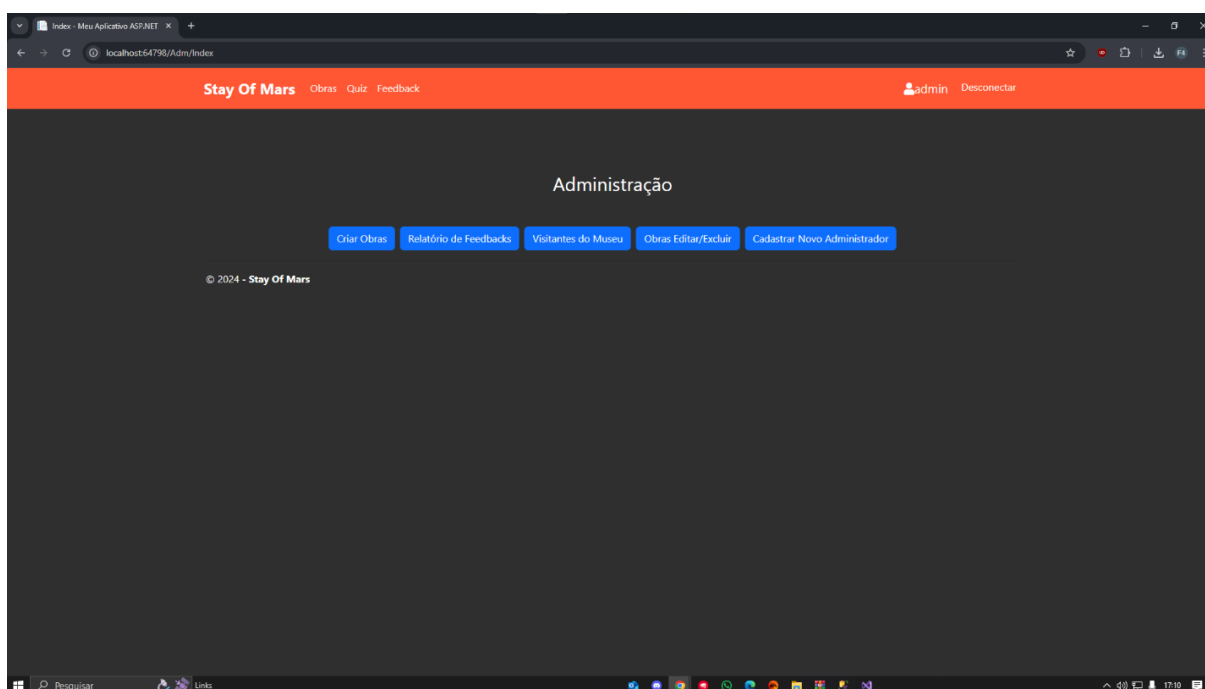
Tela inicial do acesso de administrador:

A tela de administração do site Stay Of Mars oferece cinco principais opções de gerenciamento: Criar Obras, Relatórios de Feedback, Visitantes do Museu, Obras Editar/Excluir, e Cadastrar Novo Administrador. Além disso, há um botão Desconectar** localizado no canto superior direito para encerrar a sessão.

Essas opções permitem que o administrador:

- Crie e edite obras do museu,
- Acesse e visualize relatórios de feedback dos usuários,
- Gerencie informações sobre os visitantes do museu,
- Edite ou exclua obras existentes,
- Cadastre novos administradores no sistema. Essas funcionalidades garantem

controle total sobre o conteúdo e a administração do site, conforme mostrado na imagem abaixo.



Tela de Criar Obras:

Ao selecionar a opção Criar Obras, o administrador pode adicionar uma nova obra ao sistema inserindo o Título e a Descrição da obra. Para incluir uma imagem, o administrador deve clicar em Escolher Arquivo e selecionar a imagem desejada para upload.

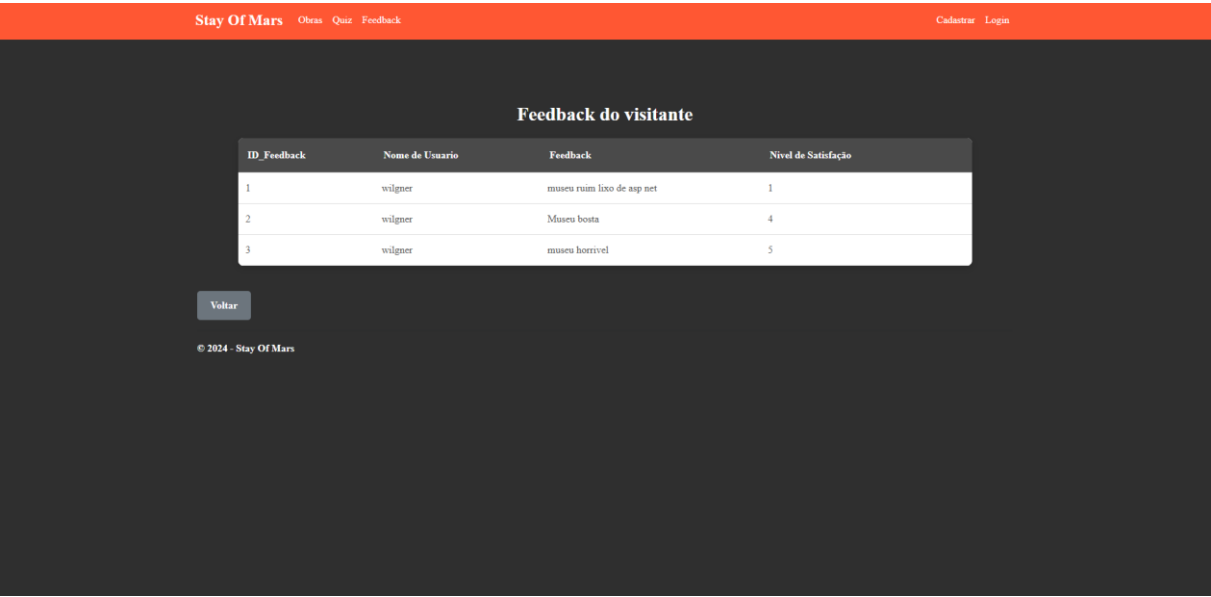
Após preencher todos os campos necessários, basta clicar no botão Criar Obra para concluir a adição da nova obra ao sistema, tornando-a visível aos usuários no site, conforme mostrado na imagem abaixo.

The screenshot displays the 'Criar Obras' form within the Stay Of Mars application. The form is centered on a dark gray background. It features three input fields for 'Título_Obra', 'Descricao_Obra', and 'Imagem_Obra'. Below the 'Imagem_Obra' field, there is a button labeled 'Escolher arquivo' and a text indicator 'Nenhum ... escolhido'. A blue 'Criar Obra' button is positioned at the bottom of the form. The application's header is orange, containing the 'Stay Of Mars' logo and navigation links for 'Obras', 'Quiz', and 'Feedback' on the left, and 'Cadastrar' and 'Login' on the right. A 'Voltar' button is located at the bottom left of the form area, and the footer shows the copyright notice '© 2024 - Stay Of Mars'.

Tela de Relatórios:

Na opção Relatórios de Feedback, o administrador acessa uma tabela intitulada Feedback do Visitante. Esta tabela exibe colunas para ID, Nome, Feedback, e Nível de Satisfação, permitindo ao administrador visualizar os comentários e as avaliações deixadas por usuários.

Através dessa tabela, o administrador pode monitorar o feedback recebido, analisando a satisfação e as sugestões dos visitantes para melhorar a experiência no site, conforme mostrado na imagem abaixo.



Tela de visitantes:

Na aba **Visitantes do Museu**, o administrador encontra uma tabela com informações detalhadas sobre os visitantes. A tabela exibe as colunas **ID**, **Nome**, **Idade**, e **Email**, onde cada linha representa um visitante diferente.

Essa visão facilita o gerenciamento e acompanhamento dos visitantes, permitindo ao administrador acessar dados importantes de forma organizada e prática, conforme a imagem abaixo.

ID_Visitante	Nome de Usuario	Idade Visitante	Email Visitante
1	Wlgnr	14	wlgnrTeste@gmail.com
2	Wlgnr2	14	wlgnrTeste@gmail.com
3	Wlgnr3	14	Wlgnr
4	Wlgnr4	14	Wlgnr
5	Wlgnr5	11	Wlgnr
6	Wlgnr6	14	Wlgnr
7	Wlgnr7	14	Wlgnr
8	Wlgnr8	14	Wlgnr
9	Wlgnr9	14	Wlgnr
10	Wlgnr10	14	wlgnrTeste@gmail.com
11	Wlgnr11	14	wlgnrTeste@gmail.com
1002	matheus3	11	Wlgnr
1003	Murilo	14	wlgnrTeste@gmail.com
1004	gugu	19	lragu@gmail.com

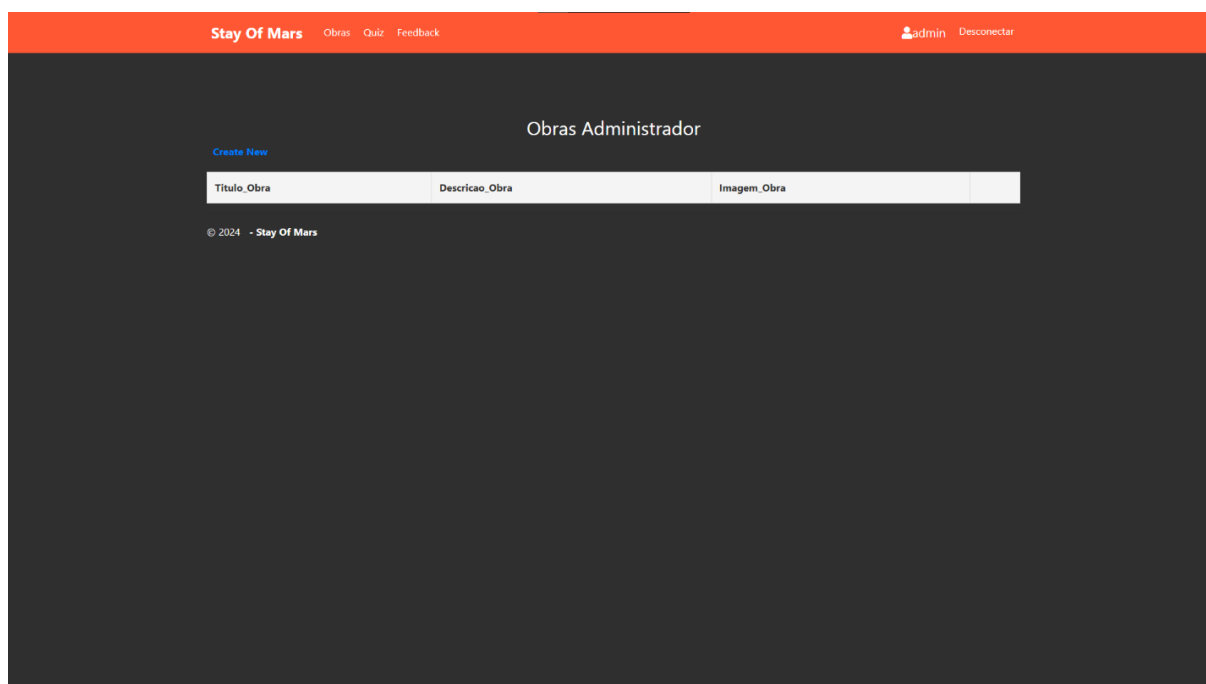
Tela de Obras Editar/Excluir:

Na seção Obras do painel de administração, o administrador encontrará um cabeçalho com o título Obras Administrador. Nessa área, há um botão Create New, que permite ao administrador adicionar novas obras ao sistema.

Abaixo do cabeçalho, existe uma tabela que exibe as informações das obras já cadastradas. As colunas da tabela são:

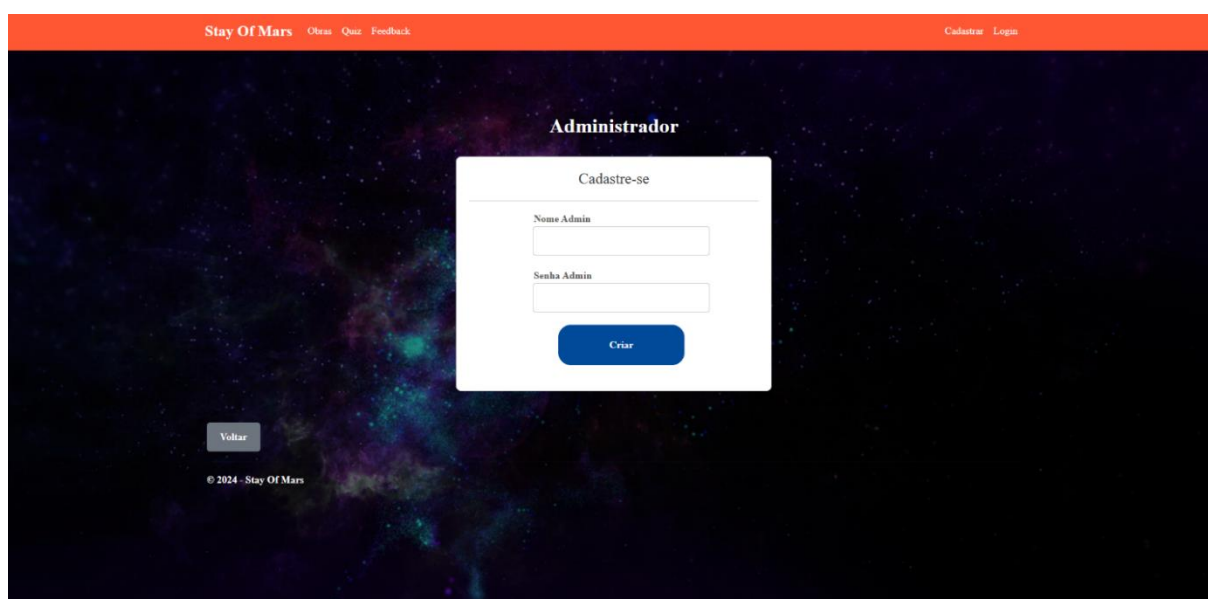
- Título Obra: o nome da obra,
- Descrição Obra: uma breve descrição da obra,
- Imagem Obra: a imagem associada à obra.

Essa estrutura facilita a visualização e o gerenciamento das obras existentes, além de possibilitar a adição de novas obras por meio do botão Create New, conforme a imagem abaixo.



Tela para criar outro Admin:

Na tela de administração, o administrador pode cadastrar novos administradores. Para isso, há campos específicos para inserir o **Nome ADM** e a **Senha ADM**. Após preencher esses campos com as informações do novo administrador, o administrador deve clicar no botão **Create** para adicionar o novo administrador ao sistema, conforme a imagem abaixo.



The screenshot displays the 'Administrador' (Administrator) registration interface. At the top, an orange navigation bar contains the text 'Stay Of Mars' and links for 'Olá', 'Quiz', 'Feedback', 'Cadastrar', and 'Login'. The main content area has a dark, starry background. A white modal box titled 'Administrador' is centered, containing a 'Cadastre-se' (Register) section. This section includes two input fields: 'Nome Admin' and 'Senha Admin'. Below these fields is a blue button labeled 'Criar' (Create). In the bottom left corner of the modal, there is a grey button labeled 'Voltar' (Back) and a copyright notice '© 2024 - Stay Of Mars'.

ANEXO 2 – GUIA DO USUARIO (DESKTOP)

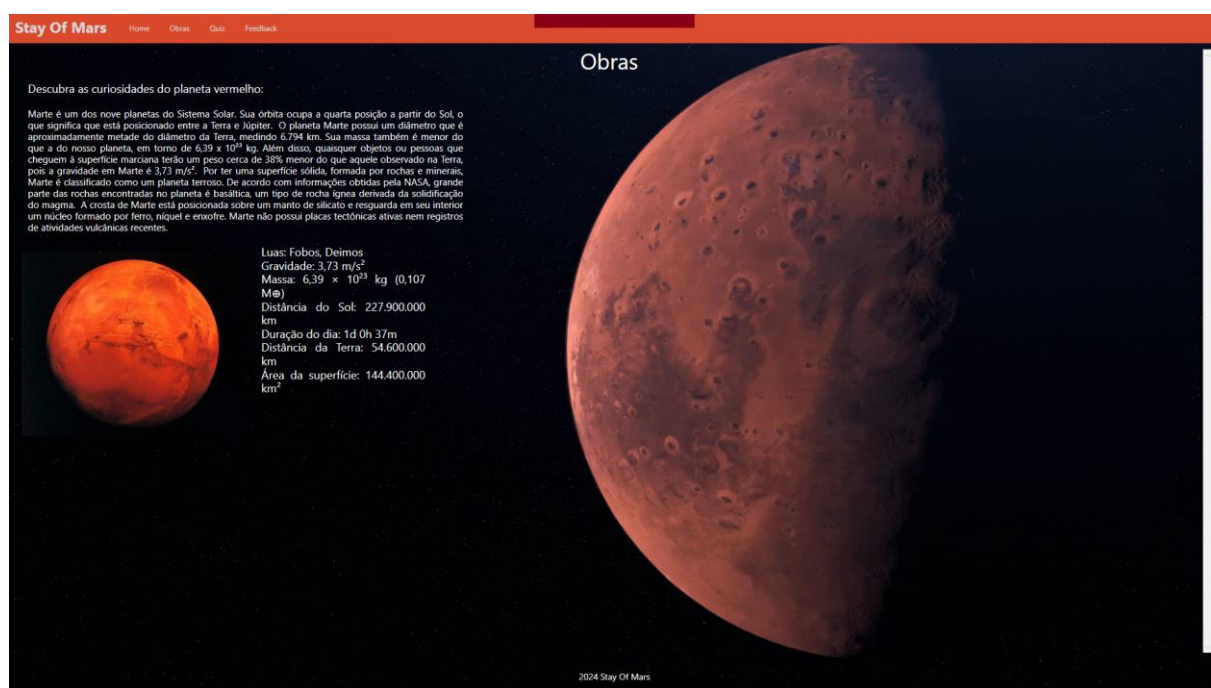
Tela inicial do site:

A tela inicial do software "Stay Of Mars" tem um cabeçalho com o nome do site, links para "Obras," "Quiz," e "Feedback," além de um ícone de usuário e opção de desconexão. Logo abaixo, há um logotipo com uma imagem de Marte. Em seguida, o título "STAY OF MARS" e a descrição "EXPOSIÇÕES ONLINE GUIADAS.". Para acessar as obras, o usuário deve clicar na opção "Obras", respectivamente, no cabeçalho acima.



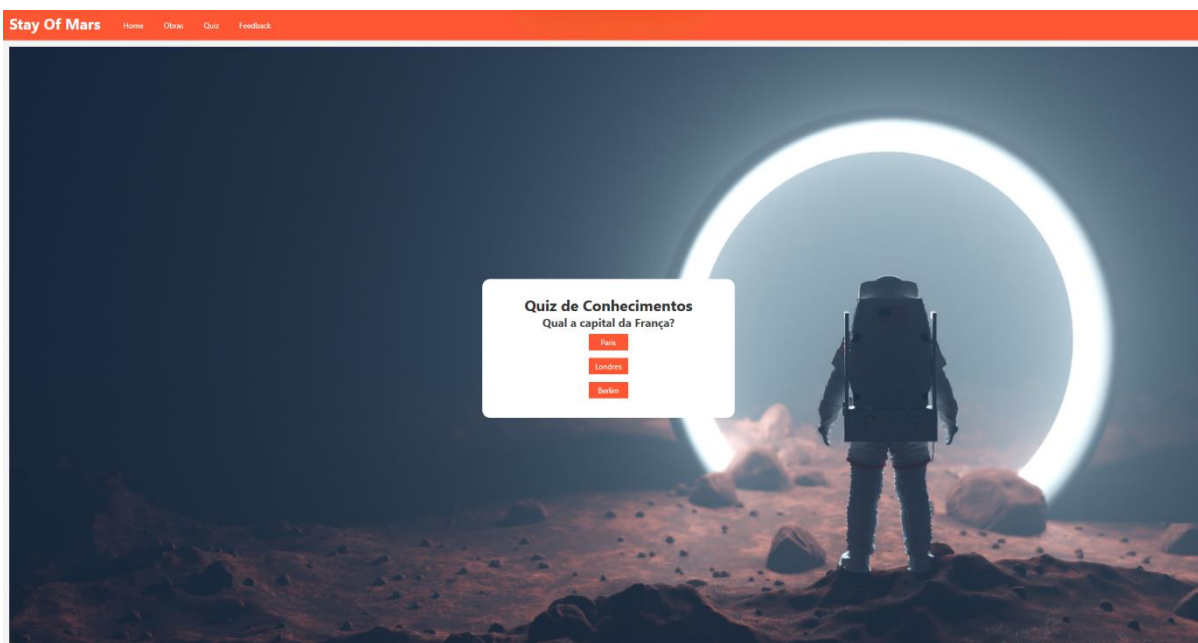
Tela de obras do site:

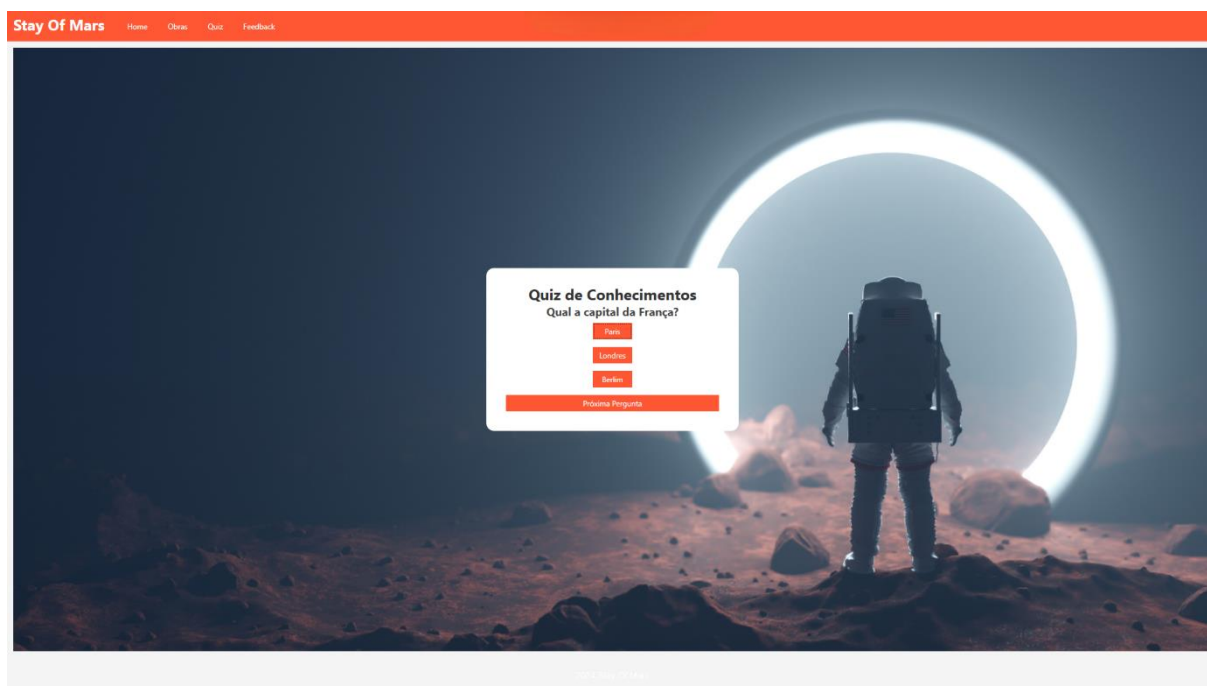
Após sair da tela inicial, ele deverá clicar na opção de “Obras” para dar início a experiência proporcionada pelo software. Esta página destaca as características de Marte, incluindo sua posição no sistema solar, gravidade, luas e dados geológicos. Há uma seção com informações detalhadas e imagens do planeta, além de um botão para visualizar um modelo 3D de Marte.



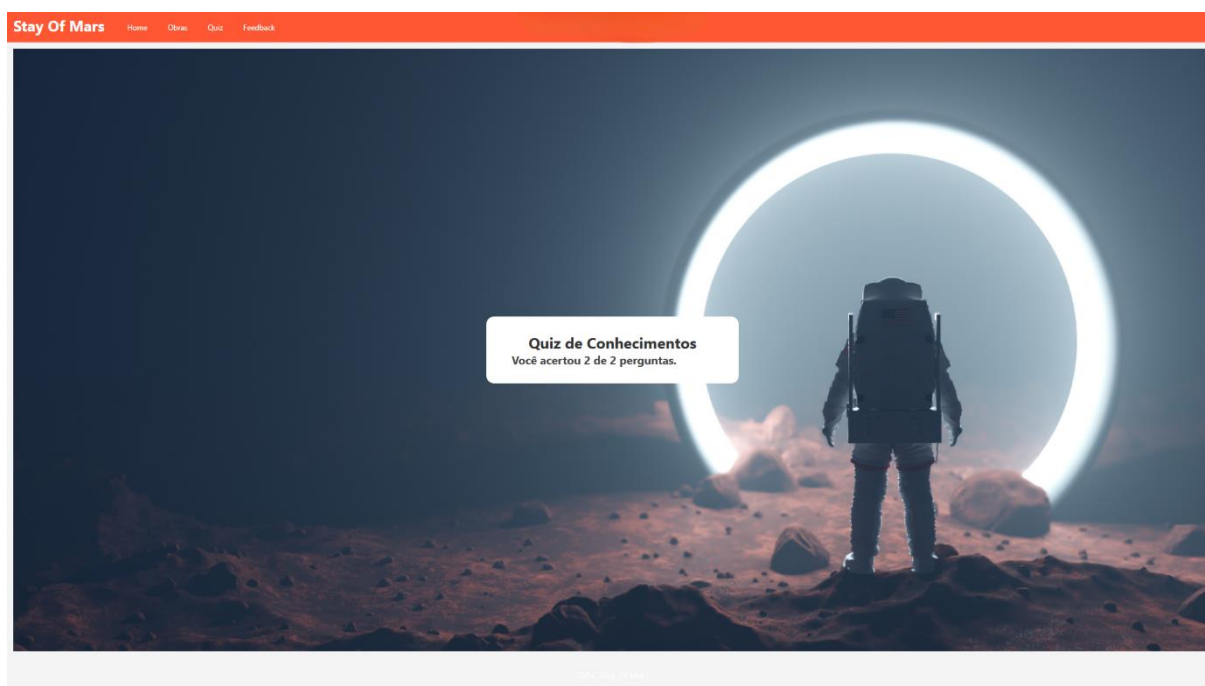
Tela do Quiz do site:

Ao clicar na aba de quiz do software, será aberto uma tela onde permite aos usuários testar seus conhecimentos sobre Marte. No centro da tela, há uma pergunta de múltipla escolha. Abaixo das opções, há um botão verde com o texto "Próxima Pergunta" que, quando clicado, leva o usuário para a próxima questão do quiz.





Ao responder todas as perguntas, apareça uma tela e uma mensagem dizendo quantas perguntas foram respondidas corretamente, como na imagem abaixo:



Tela de Feedback:

O usuário depois de responder o quiz, deverá ir para a tela de feedback do software, onde é possível deixar suas opiniões. Possui campos para nome e feedback, onde é possível deixar uma avaliação de satisfação em forma de texto e outra com estrelas e um botão "Enviar".

Stay Of Mars Home Obras Quiz Feedback

Feedback

Usuário:

Nos deixe seu feedback:

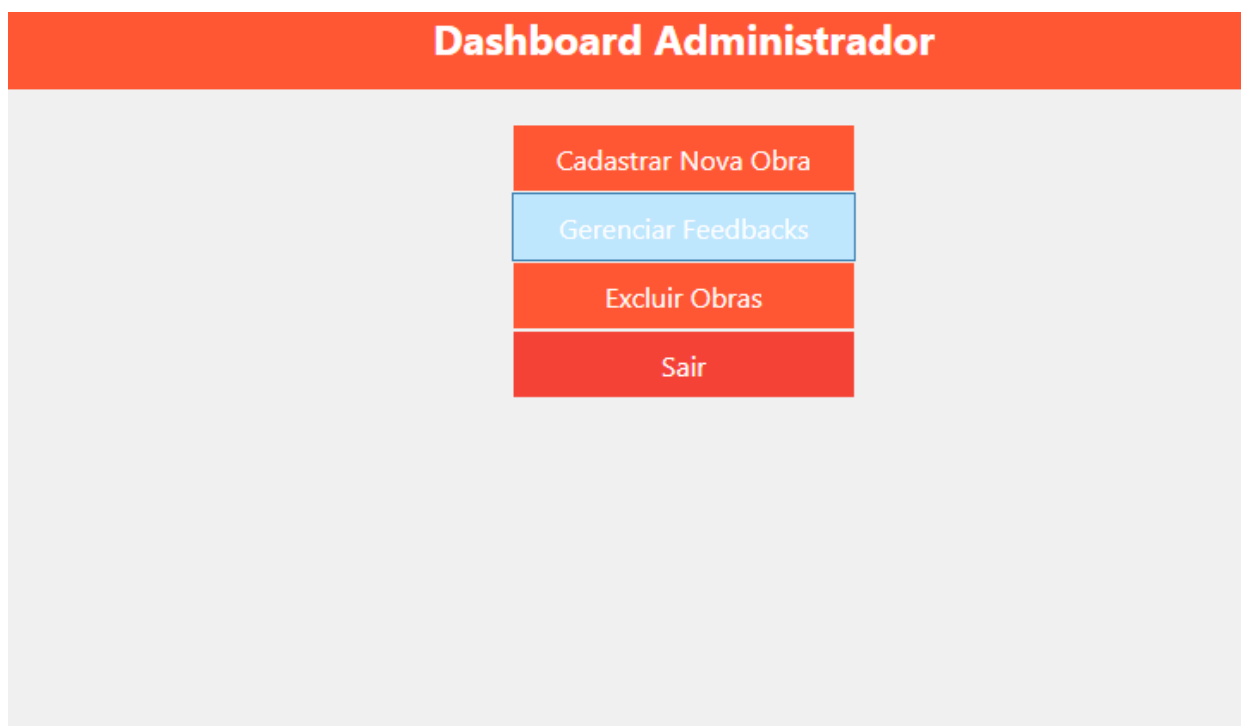
Nível de Satisfação
★ ★ ★ ★ ★

[Enviar Feedback](#)

2024 Stay Of Mars

Tela inicial do acesso de administrador:

A tela de administração do site "Stay Of Mars" contém cinco opções "Cadastrar Nova Obra", "Gerenciar feedback", "Excluir Obras" e "Sair". Essas opções serão para criar e editar obras, visualizar relatórios de feedback e deletar obras.



Tela de Cadastrar Obras:

Ao selecionar a opção de cadastrar obras, o usuário insere o nome da obra e insere uma descrição a ela, além de fazer o upload de uma imagem clicando em "Escolher arquivo", caso não haja nenhuma imagem selecionada, aparecerá uma mensagem dizendo: "Imagem Selecionada: Nenhuma". Após preencher os campos, basta clicar em "Cadastrar Obra" para adicionar a nova obra ao sistema. Há a opção de voltar para a tela inicial.

Cadastro de Obra

Nome Obra:

Descrição Obra:

Imagem Selecionada: Nenhuma

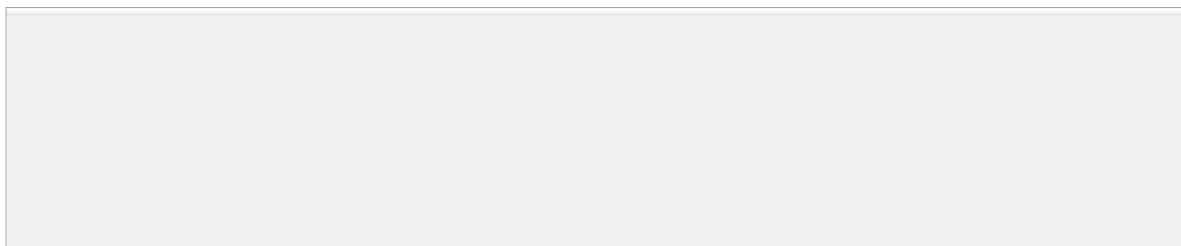
Escolher Imagem

Voltar Cadastrar Obra

Tela de Gerenciar Feedback:

A próxima opção é a aba de “Gerenciar Feedback”. Abaixo, há uma tabela intitulada "Feedback do visitante" com colunas para ID, Nome, Feedback, e Nível de Satisfação. O usuário pode visualizar comentários e avaliações de outros usuários.

Lista de feedbacks



[Voltar para Home](#)

Tela de Exclusão e Edição de Obras

Nesta tela, o administrador tem a funcionalidade de excluir ou editar obras registradas no sistema.

Funcionalidades Disponíveis:

Pesquisar pelo ID da Obra Utilize o campo **ID** para localizar a obra que deseja editar ou excluir e o Nome da obra.

Editar Detalhes da Obra Após localizar a obra, é possível alterar Informações descritivas sobre a obra, caminho ou URL da imagem representativa da obra. Excluir Obra Clique no botão **Excluir** (vermelho) para remover permanentemente a obra do sistema.

Editar Obra: Clique no botão **Editar** (verde) para salvar as alterações feitas nos campos.

Voltar à Tela Principal Utilize o botão **Voltar** para retornar à tela inicial do administrador sem realizar alterações.

Observações: Certifique-se de revisar os dados antes de confirmar a edição ou exclusão.

A exclusão de obras é irreversível, portanto, use esta função com cautela.

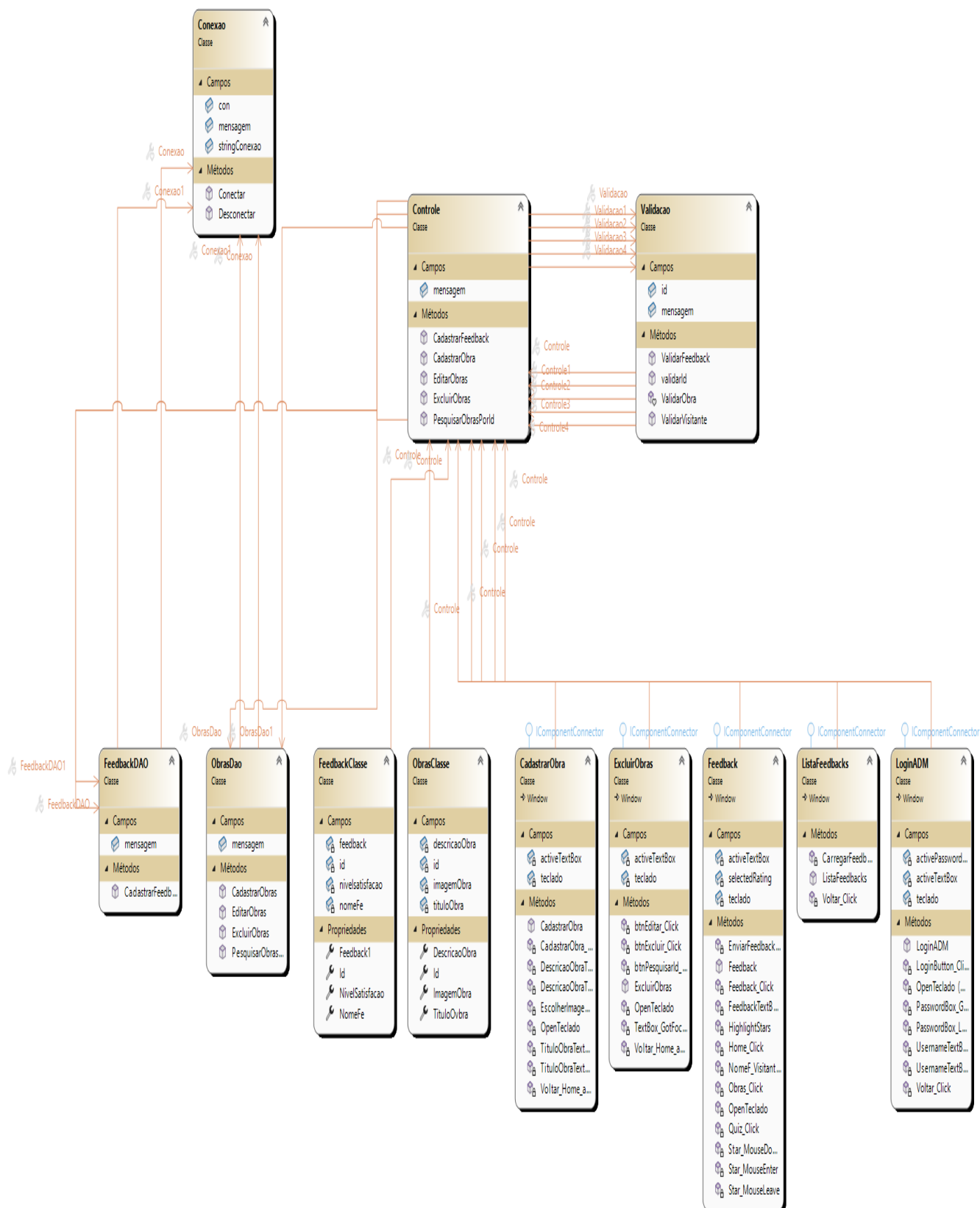
Essa tela foi projetada para simplificar o gerenciamento das obras no sistema.

A interface 'Excluir/Editar Obras' apresenta um formulário com os seguintes elementos:

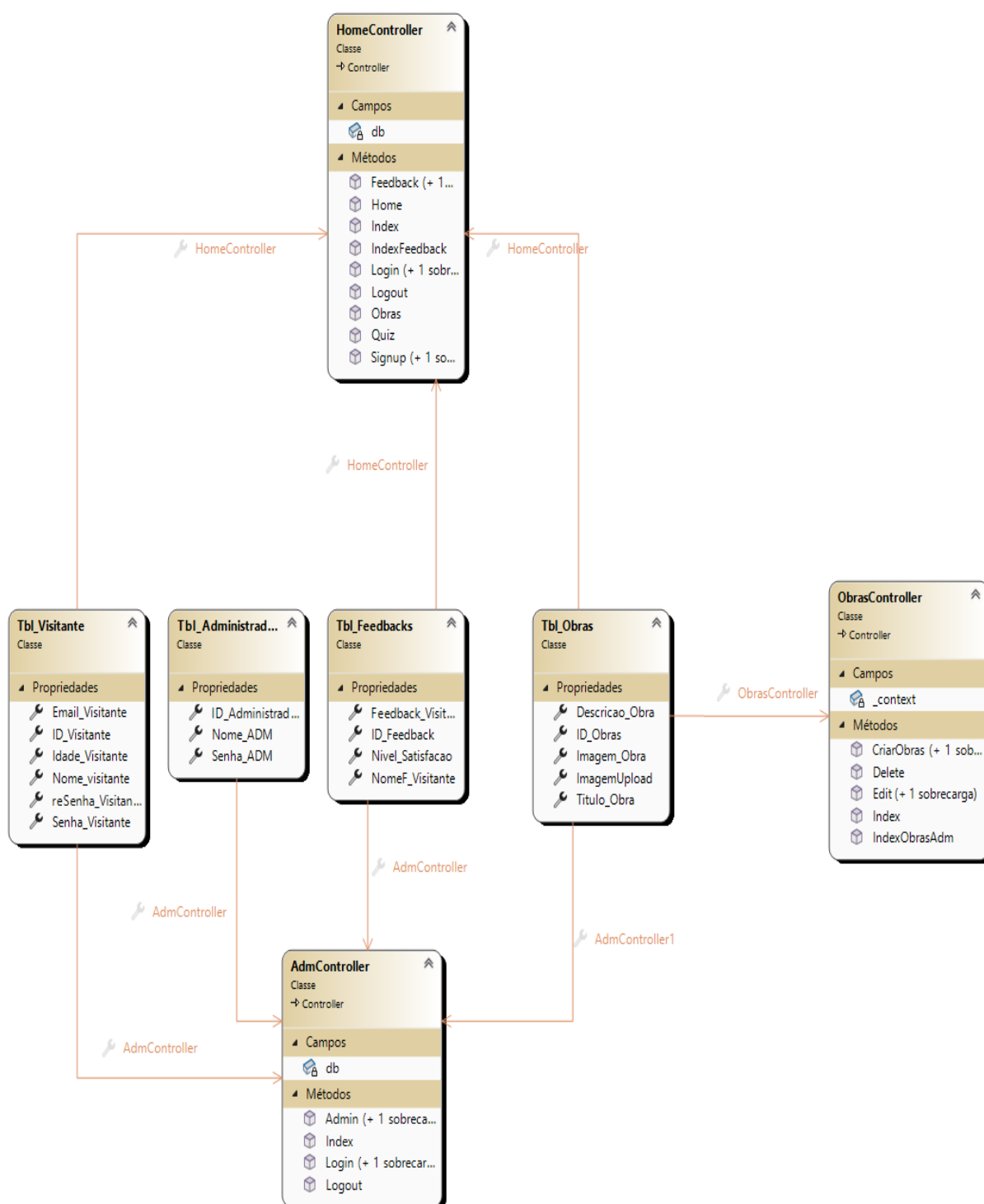
- Um campo de texto para 'ID:' com um botão azul 'Pesquisar ID' ao lado.
- Um campo de texto para 'Obras:'.
- Um campo de texto para 'Título:'.
- Um campo de texto grande para 'Descrição Obras:'.
- Um campo de texto para 'Imagem:'.
- Três botões de ação: 'Voltar' (branco), 'Excluir' (vermelho) e 'Editar' (verde).

ANEXO 3 – DIAGRAMA TOTEM SOFTWARE

A imagem a baixo seria do diagrama do softawre que mostra mais especificado os metodos, funcoes, classes e a comunicacao mdo software.



ANEXO 4 – DIAGRAMA WEB SOFTWARE



ANEXO 5 – ALGORITIMOS WEB

Adm controller

programa

```
{
    // Simula um banco de dados de administradores
    tipo Administrador
    {
        inteiro id
        cadeia nome
        cadeia senha
    }

    // Lista para armazenar administradores cadastrados
    variavel listaAdministradores : vetor[100] de Administrador
    variavel totalAdministradores : inteiro = 0

    // Sessão do administrador logado
    variavel administradorLogado : Administrador

    funcao inicio()
    {
        inteiro opcao
        faca
        {
            escreva("1 - Login\n")
            escreva("2 - Registrar\n")
            escreva("3 - Logout\n")
            escreva("4 - Sair\n")
            escreva("Escolha uma opção: ")
            leia(opcao)

            escolha(opcao)
```



```

{
    caso 1:
        login()
        pare
    caso 2:
        registrar()
        pare
    caso 3:
        logout()
        pare
    caso 4:
        escreva("Encerrando o programa...\n")
        pare
    caso contrario:
        escreva("Opção inválida. Tente novamente.\n")
}
} enquanto (opcao != 4)
}

```

```

funcao registrar()
{
    cadeia nome, senha
    escreva("Digite o nome do administrador: ")
    leia(nome)

    // Verifica se o administrador já existe
    para (inteiro i = 0; i < totalAdministradores; i++)
    {
        se (listaAdministradores[i].nome == nome)
        {
            escreva("Esta conta já existe.\n")
            retorne
        }
    }
}

```

```

    }
}

escreva("Digite a senha do administrador: ")
leia(senha)

// Adiciona o novo administrador à lista
listaAdministradores[totalAdministradores].id = totalAdministradores + 1
listaAdministradores[totalAdministradores].nome = nome
listaAdministradores[totalAdministradores].senha = senha
totalAdministradores++

escreva("Administrador registrado com sucesso!\n")
}

funcao login()
{
    cadeia nome, senha
    escreva("Digite o nome do administrador: ")
    leia(nome)
    escreva("Digite a senha do administrador: ")
    leia(senha)

    // Verifica se o administrador existe e a senha está correta
    para (inteiro i = 0; i < totalAdministradores; i++)
    {
        se      (listaAdministradores[i].nome      ==      nome      &&
listaAdministradores[i].senha == senha)
        {
            administradorLogado = listaAdministradores[i]
            escreva("Login realizado com sucesso! Bem-vindo, ",
administradorLogado.nome, ".\n")

```

```
        retorne
    }
}
    escreva("Nome de usuário ou senha incorretos.\n")
}

funcao logout()
{
```

Home controller

programa

```
{  
    // Tipos simulando tabelas do banco de dados  
    tipo Visitante  
    {  
        inteiro id  
        cadeia nome  
        cadeia senha  
    }  
  
    tipo Feedback  
    {  
        inteiro id  
        cadeia nomeVisitante  
        cadeia mensagem  
    }  
  
    // Listas simulando tabelas do banco  
    variavel visitantes : vetor[100] de Visitante  
    variavel totalVisitantes : inteiro = 0  
  
    variavel feedbacks : vetor[100] de Feedback  
    variavel totalFeedbacks : inteiro = 0  
  
    // Sessão para visitante logado  
    variavel visitanteLogado : Visitante  
  
    funcao inicio()  
    {  
        inteiro opcao  
        faca
```

```
{
    escreva("Bem-vindo ao Museu Virtual!\n")
    escreva("1 - Home\n")
    escreva("2 - Ver Obras\n")
    escreva("3 - Enviar Feedback\n")
    escreva("4 - Cadastro\n")
    escreva("5 - Login\n")
    escreva("6 - Logout\n")
    escreva("7 - Sair\n")
    escreva("Escolha uma opção: ")
    leia(opcao)

    escolha(opcao)
    {
        caso 1:
            home()
            pare
        caso 2:
            verObras()
            pare
        caso 3:
            enviarFeedback()
            pare
        caso 4:
            cadastrarVisitante()
            pare
        caso 5:
            login()
            pare
        caso 6:
            logout()
            pare
    }
```

```

        caso 7:
            escreva("Encerrando o programa...\n")
            pare
        caso contrario:
            escreva("Opção inválida. Tente novamente.\n")
        }
    } enquanto (opcao != 7)
}

```

```

funcao home()
{
    escreva("Bem-vindo à Home do Museu Virtual!\n")
    se (visitanteLogado.nome != "")
    {
        escreva("Visitante logado: ", visitanteLogado.nome, "\n")
    }
    senao
    {
        escreva("Nenhum visitante está logado.\n")
    }
}

```

```

funcao verObras()
{
    escreva("Aqui estão algumas das obras disponíveis no museu:\n")
    escreva("- Obra 1: Mona Lisa\n")
    escreva("- Obra 2: O Grito\n")
    escreva("- Obra 3: Noite Estrelada\n")
}

```

```

funcao enviarFeedback()
{

```

```
cadeia nome, mensagem  
escreva("Digite seu nome: ")  
leia(nome)  
escreva("Digite seu feedback: ")  
leia(mensagem)
```

Obras controller

programa

```
{  
    // Simula a base de dados  
    tipo Obra {  
        inteiro ID_Obras  
        cadeia Titulo_Obra  
        cadeia Descricao_Obra  
        cadeia Imagem_Obra  
    }  
  
    // Lista de obras (representando o banco de dados)  
    lista<Obra> obras <- []  
  
    funcao inicio()  
    {  
        inteiro opcao  
        repita  
        {  
            exibirMenu()  
            leia(opcao)  
  
            escolha(opcao)  
            {  
                caso 1:  
                    listarObras()  
                caso 2:  
                    criarObra()  
                caso 3:  
                    editarObra()  
                caso 4:  
                    deletarObra()  
            }  
        }  
    }  
}
```



```

        caso 5:
            escrever("Saindo do programa...")
            pare
        caso contrario:
            escrever("Opção inválida. Tente novamente.")
        }
    } enquanto (verdadeiro)
}

funcao exibirMenu()
{
    escrever("\n===== Menu =====")
    escrever("1. Listar Obras")
    escrever("2. Criar Obra")
    escrever("3. Editar Obra")
    escrever("4. Deletar Obra")
    escrever("5. Sair")
    escrever("Escolha uma opção: ")
}

funcao listarObras()
{
    escrever("\n===== Lista de Obras =====")
    para inteiro i <- 0 ate tamanho(obras) - 1 faca
    {
        escrever("ID: ", obras[i].ID_Obras, ", Título: ", obras[i].Titulo_Obra)
    }
}

funcao criarObra()
{
    Obra novaObra

```

cadeia titulo, descricao

```
escrever("\n===== Criar Nova Obra =====")
```

```
escrever("Digite o título da obra: ")
```

```
leia(titulo)
```

```
escrever("Digite a descrição da obra: ")
```

```
leia(descricao)
```

```
novaObra.ID_Obras <- tamanho(obras) + 1
```

```
novaObra.Titulo_Obra <- titulo
```

```
novaObra.Descricao_Obra <- descricao
```

```
novaObra.Imagem_Obra <- "Sem imagem" // Imagem fictícia
```

```
obras.adicionar(novaObra)
```

```
escrever("Obra criada com sucesso!")
```

```
}
```

```
funcao editarObra()
```

```
{
```

```
inteiro id
```

```
cadeia novoTitulo, novaDescricao
```

```
escrever("\n===== Editar Obra =====")
```

```
escrever("Digite o ID da obra para editar: ")
```

```
leia(id)
```

```
para inteiro i <- 0 ate tamanho(obras) - 1 faca
```

```
{
```

```
se (obras[i].ID_Obras == id)
```

```
{
```

```
escrever("Digite o novo título: ")
```

```
leia(novoTitulo)
```

```
escrever("Digite a nova descrição: ")  
leia(novaDescricao)
```

```
obras[i].Titulo_Obra <- novoTitulo  
obras[i].Descricao_Obra <- novaDescricao
```

Controle

programa

```
{
```

```
  tipo Visitante {
```

```
    inteiro Id
```

```
    cadeia Nome
```

```
    cadeia Email
```

```
    inteiro Idade
```

```
    cadeia Senha
```

```
  }
```

```
  tipo Validacao {
```

```
    cadeia mensagem
```

```
    inteiro id
```

```
  funcao ValidarVisitante(lista<cadeia> listaDadosVisitante)
```

```
  {
```

```
    mensagem <- ""
```

```
    se (tamanho(listaDadosVisitante) < 5)
```

```
    {
```

```
      mensagem <- "Dados insuficientes para validar o visitante."
```

```
      retorne
```

```
    }
```

```
    // Validações simples para os dados
```

```
    se (listaDadosVisitante[1] == "")
```

```
    {
```

```
      mensagem <- "Nome inválido."
```

```
    }
```

```
    senao se (listaDadosVisitante[2] == "" ou nao
```

```
    contem(listaDadosVisitante[2], "@"))
```

```
    {
```

```

        mensagem <- "Email inválido."
    }
    senao se (converter_inteiro(listaDadosVisitante[3]) <= 0)
    {
        mensagem <- "Idade inválida."
    }
    senao
    {
        id <- aleatorio(1, 1000) // Simula a geração de ID
    }
}

funcao ValidarId(cadeia idTexto)
{
    mensagem <- ""
    se (converter_inteiro(idTexto) <= 0)
    {
        mensagem <- "ID inválido."
    }
    senao
    {
        id <- converter_inteiro(idTexto)
    }
}

tipo VisitanteDAO {
    cadeia mensagem

    funcao CadastrarVisitante(Visitante visitante)
    {
        // Simula a inserção no "banco de dados"
    }
}

```

```

    escrever("Visitante cadastrado com sucesso: ", visitante.Nome)
    mensagem <- "Visitante cadastrado com sucesso!"
}

```

```

funcao EditarVisitante(Visitante visitante)
{
    escrever("Visitante editado: ", visitante.Nome)
    mensagem <- "Visitante editado com sucesso!"
}

```

```

funcao ExcluirVisitante(Visitante visitante)
{
    escrever("Visitante excluído: ID ", visitante.Id)
    mensagem <- "Visitante excluído com sucesso!"
}

```

```

funcao PesquisarVisitantePorId(Visitante visitante): Visitante
{
    escrever("Visitante encontrado: ID ", visitante.Id)
    visitante.Nome <- "Visitante Teste" // Dados fictícios
    visitante.Email <- "teste@email.com"
    visitante.Idade <- 30
    visitante.Senha <- "123456"
    retorne visitante
}
}

```

```

tipo Controle {
    cadeia mensagem

```

```

funcao CadastrarVisitante(lista<cadeia> listaDadosVisitante)
{

```

```
mensagem <- ""  
Validacao validacao <- novo Validacao()  
validacao.ValidarVisitante(listaDadosVisitante)  
mensagem <- validacao.mensagem  
  
se (mensagem == "")
```

Conexão

programa

```
{
```

```
  tipo Conexao {
```

```
    cadeia stringConexao
```

```
    cadeia mensagem
```

```
    inteiro estadoConexao // 0 para Fechada, 1 para Aberta
```

```
    funcao inicializar()
```

```
    {
```

```
        stringConexao <- "FonteDeDados=REVER-
```

```
NOTE\\SQLEXPRESS;Banco=CrudPessoas;Usuario=sa;Senha=rever"
```

```
        mensagem <- ""
```

```
        estadoConexao <- 0 // Inicialmente, conexão fechada
```

```
    }
```

```
    funcao Conectar(): inteiro
```

```
    {
```

```
        mensagem <- ""
```

```
        tente
```

```
        {
```

```
            se (estadoConexao == 0)
```

```
            {
```

```
                escrever("Conectando ao banco de dados com: ", stringConexao)
```

```
                estadoConexao <- 1 // Simula abertura de conexão
```

```
                mensagem <- "Conexão estabelecida com sucesso!"
```

```
            }
```

```
        senao
```

```
        {
```

```
            mensagem <- "Conexão já está aberta."
```

```
        }
```

```
    }
```



```

    pegue (excecao e)
    {
        mensagem <- "Erro ao conectar com BD: " + e
    }
    retorne estadoConexao
}

funcao Desconectar(): inteiro
{
    mensagem <- ""
    tente
    {
        se (estadoConexao == 1)
        {
            escrever("Desconectando do banco de dados...")
            estadoConexao <- 0 // Simula fechamento de conexão
            mensagem <- "Conexão fechada com sucesso!"
        }
        senao
        {
            mensagem <- "Conexão já está fechada."
        }
    }
    pegue (excecao e)
    {
        mensagem <- "Erro ao desconectar do BD: " + e
    }
    retorne estadoConexao
}
}

// Teste da funcionalidade

```

```
funcao inicio()
{
    Conexao conexao <- novo Conexao()
    conexao.inicializar()

    inteiro status <- conexao.Conectar()
    escrever("Status da conexão: ", status, " - ", conexao.mensagem)

    status <- conexao.Desconectar()
    escrever("Status da conexão: ", status, " - ", conexao.mensagem)
}
}
```

VisitanteDAO

```
programa
```

```
{
  tipo Visitante
  {
    inteiro Id
    cadeia Nome
    cadeia Email
    cadeia Idade
    cadeia Senha
  }
}
```

```
tipo VisitanteDAO
```

```
{
  cadeia mensagem
}
```

```
funcao CadastrarVisitante(Visitante visitante)
```

```
{
  cadeia sql <- "INSERT INTO Tbl_Visitante (nome, email, idade, senha)
VALUES (@nome, @email, @idade, @senha)"
  tente
  {
    // Simula execução do comando
    escrever("Executando SQL: ", sql)
    escrever("Parâmetros: Nome = ", visitante.Nome, ", Email = ",
visitante.Email, ", Idade = ", visitante.Idade, ", Senha = ", visitante.Senha)
    mensagem <- "Visitante cadastrada com sucesso."
  }
  pegue (excecao e)
  {
    mensagem <- "Erro no cadastro: " + e
  }
}
```

```

finalmente
{
    escrever("Conexão com banco de dados fechada.")
}
}

funcao PesquisarVisitantePorId(Visitante visitante): Visitante
{
    cadeia sql <- "SELECT * FROM Tbl_Visitante WHERE id = @id"
    tente
    {
        // Simula consulta ao banco
        escrever("Executando SQL: ", sql)
        escrever("Parâmetro: Id = ", visitante.Id)
        // Simula retorno do banco de dados
        se (visitante.Id == 1) // Exemplo: se ID é 1, retornamos dados fictícios
        {
            visitante.Nome <- "João Silva"
            visitante.Email <- "joao@email.com"
            visitante.Idade <- "30"
            visitante.Senha <- "senha123"
        }
        senao
        {
            mensagem <- "Id não encontrado."
        }
    }
    pegue (excecao e)
    {
        mensagem <- "Erro na pesquisa: " + e
    }
    finalmente

```

```

    {
        escrever("Conexão com banco de dados fechada.")
    }
    retorne visitante
}

```

```

funcao EditarVisitante(Visitante visitante)
{
    cadeia sql <- "UPDATE Tbl_Visitante SET nome = @nome, email =
@email, idade = @idade, senha = @senha WHERE id = @id"
    tente
    {
        // Simula execução do comando
        escrever("Executando SQL: ", sql)
        escrever("Parâmetros: Id = ", visitante.Id, ", Nome = ", visitante.Nome,
", Email = ", visitante.Email, ", Idade = ", visitante.Idade, ", Senha = ", visitante.Senha)
        mensagem <- "Visitante editado com sucesso."
    }
    pegue (excecao e)
    {
        mensagem <- "Erro na edição: " + e
    }
    finalmente
    {
        escrever("Conexão com banco de dados fechada.")
    }
}

```

```

funcao ExcluirVisitante(Visitante visitante)
{
    cadeia sql <- "DELETE FROM Tbl_Visitante WHERE id = @id"
    tente

```

```
{  
  // Simula execução do comando  
  escrever("Executando SQL: ", sql)  
  escrever("Parâmetro: Id = ", visitante.Id)  
  mensagem <- "Visitante excluído com sucesso."  
}  
pegue (excecao e)  
{  
  mensagem <- "Erro na exclusão: " + e
```

Admhtml

programa

```
{
    tipo Administrador
    {
        cadeia Nome_ADM
        cadeia Senha_ADM
    }

    // Função para exibir o formulário e capturar os dados
    funcao FormularioAdministrador(): Administrador
    {
        Administrador admin
        admin.Nome_ADM <- ""
        admin.Senha_ADM <- ""

        // Título
        escreva("=====\n")
        escreva("      Cadastro de Administrador\n")
        escreva("=====\n\n")

        // Campo Nome
        escreva("Digite o Nome do Administrador: ")
        leia(admin.Nome_ADM)

        // Validação Nome
        enquanto (admin.Nome_ADM == "")
        {
            escreva("Campo obrigatório! Digite o Nome do Administrador: ")
            leia(admin.Nome_ADM)
        }
    }
}
```

```

// Campo Senha
escreva("Digite a Senha do Administrador: ")
leia(admin.Senha_ADM)

// Validação Senha
enquanto (admin.Senha_ADM == "")
{
    escreva("Campo obrigatório! Digite a Senha do Administrador: ")
    leia(admin.Senha_ADM)
}

escreva("\nAdministrador cadastrado com sucesso!\n")
escreva("=====\n")

retorne admin
}

funcao inicio()
{
    Administrador novoAdmin

    // Chamada do formulário
    novoAdmin <- FormularioAdministrador()

    // Exibir os dados cadastrados
    escreva("\nDados do Administrador Cadastrado:\n")
    escreva("Nome: ", novoAdmin.Nome_ADM, "\n")
    escreva("Senha: ", novoAdmin.Senha_ADM, "\n")

    // Botão Voltar
    escreva("\nDigite 'Voltar' para sair: ")
    cadeia acao

```



```
leia(acao)

enquanto (acao != "Voltar")
{
    escreva("\nOpção inválida! Digite 'Voltar' para sair: ")
    leia(acao)
}

escreva("\nObrigado por usar o sistema de cadastro!\n")
}
}
```

Indexhtml

programa

```
{
    funcao MenuAdministracao()
    {
        inteiro opcao

        // Título do Menu
        escreva("=====\n")
        escreva("    Administração do Museu\n")
        escreva("=====\n\n")

        // Opções Interativas
        escreva("Selecione uma opção:\n")
        escreva("1. Criar Obras\n")
        escreva("2. Relatório de Feedbacks\n")
        escreva("3. Visitantes do Museu\n")
        escreva("4. Editar/Excluir Obras\n")
        escreva("5. Cadastrar Novo Administrador\n")
        escreva("0. Sair\n")
        escreva("=====\n")
        escreva("Digite o número da sua escolha: ")
        leia(opcao)

        escolha opcao
        {
            caso 1:
                escreva("\nOpção selecionada: Criar Obras\n")
                // Lógica para criar obras
                pare

            caso 2:
```

```

    escreva("\nOpção selecionada: Relatório de Feedbacks\n")
    // Lógica para acessar relatórios
    pare

```

caso 3:

```

    escreva("\nOpção selecionada: Visitantes do Museu\n")
    // Lógica para listar visitantes
    pare

```

caso 4:

```

    escreva("\nOpção selecionada: Editar/Excluir Obras\n")
    // Lógica para gerenciar obras
    pare

```

caso 5:

```

    escreva("\nOpção selecionada: Cadastrar Novo Administrador\n")
    // Lógica para cadastrar administradores
    pare

```

caso 0:

```

    escreva("\nSaindo do sistema. Obrigado!\n")
    pare

```

padrao:

```

    escreva("\nOpção inválida! Tente novamente.\n")
    MenuAdministracao() // Recursão para exibir o menu novamente
}

```

```

}

```

```

funcao inicio()

```

```

{

```

```

    MenuAdministracao()

```

```

}

```

```

}

```

Loginhtml

programa

```
{
    funcao LoginAdm()
    {
        caractere nomeAdm, senhaAdm
        inteiro opcao

        // Exibe o título do login
        escreva("=====\n")
        escreva("      Login - Administrador\n")
        escreva("=====\n\n")

        // Solicita o nome do administrador
        escreva("Digite seu nome de administrador: ")
        leia(nomeAdm)

        // Solicita a senha do administrador
        escreva("Digite sua senha de administrador: ")
        leia(senhaAdm)

        // Validação simples (substitua por lógica real de validação, como consulta
        ao banco de dados)
        se nomeAdm = "admin" e senhaAdm = "senha123" então
            escreva("\nLogin bem-sucedido!\n")
            // Lógica para direcionar ao painel do administrador
            escreva("Redirecionando para a página de administração...\n")
            // Simula a navegação
        senao
            escreva("\nNome de usuário ou senha inválidos. Tente novamente.\n")
            LoginAdm() // Recursão para tentar novamente
        fimse
    }
}
```

```
}  
  
funcao inicio()  
{  
    LoginAdm()  
}  
}
```

Criar obras html

programa

```
{
    funcao CriarObra()
    {
        caractere tituloObra, descricaoObra, imagemObra
        caractere mensagem
        inteiro opcao

        // Exibe o título da página de criação de obra
        escreva("=====\n")
        escreva("      Criar Obra\n")
        escreva("=====\n\n")

        // Solicita o título da obra
        escreva("Digite o título da obra: ")
        leia(tituloObra)

        // Solicita a descrição da obra
        escreva("Digite a descrição da obra: ")
        leia(descricaoObra)

        // Solicita o nome do arquivo da imagem da obra
        escreva("Selecione a imagem da obra (nome do arquivo): ")
        leia(imagemObra)

        // Verificação de erros de entrada (simulação, em um sistema real seria
mais complexo)
        se tituloObra = "" ou descricaoObra = "" ou imagemObra = "" então
            mensagem = "Todos os campos devem ser preenchidos corretamente."
            escreva(mensagem)

        // Chama a função novamente em caso de erro
```

```
        CriarObra()
    senao
        mensagem = "Obra criada com sucesso! Título: " + tituloObra
        escreva(mensagem)
    fimse
}

funcao inicio()
{
    CriarObra()
}
}
```

Delete html

programa

```
{
```

```
    tipo
```

```
        Obra
```

```
        {
```

```
            caractere tituloObra
```

```
            caractere descricaoObra
```

```
            caractere imagemObra
```

```
        }
```

```
funcao ExibirDetalhesObra(Obra obra)
```

```
{
```

```
    escreva("=====\n")
```

```
    escreva("        Detalhes da Obra\n")
```

```
    escreva("=====\n\n")
```

```
    escreva("Título da Obra: ", obra.tituloObra, "\n")
```

```
    escreva("Descrição da Obra: ", obra.descricaoObra, "\n")
```

```
    escreva("Imagem da Obra: ", obra.imagemObra, "\n")
```

```
}
```

```
funcao ConfirmarDelecao(Obra obra)
```

```
{
```

```
    inteiro opcao
```

```
    // Exibe os detalhes da obra
```

```
    ExibirDetalhesObra(obra)
```

```
    // Pergunta ao usuário se deseja deletar
```

```
    escreva("\nVocê tem certeza de que deseja deletar esta obra? (1 - Sim, 2
```

```
- Não): ")
```



```
leia(opcao)

se opcao = 1 então
    escreva("Obra deletada com sucesso!\n")
senao
    escreva("Deleção cancelada.\n")
fimse
}

funcao inicio()
{
    Obra obra
    obra.tituloObra = "Obra Exemplo"
    obra.descricaoObra = "Descrição da obra de exemplo."
    obra.imagemObra = "imagem.jpg"

    // Chama a função para confirmar deleção
    ConfirmarDelecao(obra)
}
}
```

Edithtml

programa

```
{
    tipo
        Obra
        {
            inteiro ID_Obras
            caractere tituloObra
            caractere descricaoObra
            caractere imagemObra
        }
}
```

funcao ExibirFormularioEdicao(Obra obra)

```
{
    escreva("=====\n")
    escreva("          Editar Obra\n")
    escreva("=====\n\n")

    escreva("Título da Obra: ", obra.tituloObra, "\n")
    escreva("Descrição da Obra: ", obra.descricaoObra, "\n")
    escreva("Imagem da Obra: ", obra.imagemObra, "\n")
    escreva("\nDigite o novo título da obra: ")
    leia(obra.tituloObra)

    escreva("\nDigite a nova descrição da obra: ")
    leia(obra.descricaoObra)

    escreva("\nDigite o novo caminho da imagem da obra: ")
    leia(obra.imagemObra)

    escreva("\nDeseja salvar as alterações? (1 - Sim, 2 - Não): ")
    inteiro opcao
```

```
leia(opcao)

se opcao = 1 então
    escreva("Alterações salvas com sucesso!\n")
senao
    escreva("Alterações não salvas.\n")
fimse
}

funcao inicio()
{
    Obra obra
    obra.ID_Obras = 1
    obra.tituloObra = "Obra Exemplo"
    obra.descricaoObra = "Descrição da obra de exemplo."
    obra.imagemObra = "imagem.jpg"

    // Chama a função para editar a obra
    ExibirFormularioEdicao(obra)
}
}
```

Algoritmo "exibicao_obras"

inicio

// Exibe o título da página

escreva("Obras de Marte\n")

escreva("Descubra as curiosidades do planeta vermelho\n")

// Exibe as características do planeta Marte

escreva("\nCaracterísticas de Marte\n")

escreva("Marte é um dos nove planetas do Sistema Solar. Sua órbita ocupa a quarta posição a partir do Sol, o que significa que está posicionado entre a Terra e Júpiter.\n")

escreva("O planeta Marte possui um diâmetro de aproximadamente 6.794 km e sua massa é em torno de $6,39 \times 10^{23}$ kg. A gravidade em Marte é $3,73 \text{ m/s}^2$, o que faz com que qualquer objeto tenha um peso 38% menor do que na Terra.\n")

escreva("Marte é classificado como um planeta terroso e sua superfície é composta principalmente por rochas basálticas.\n")

escreva("A crosta de Marte está sobre um manto de silicato e seu núcleo é formado por ferro, níquel e enxofre. Marte não possui placas tectônicas ativas.\n")

// Exibe as informações sobre Marte

escreva("\nInformações adicionais sobre Marte\n")

escreva("Luas: Fobos, Deimos\n")

escreva("Gravidade: $3,73 \text{ m/s}^2$ \n")

escreva("Massa: $6,39 \times 10^{23} \text{ kg}$ \n")

escreva("Distância do Sol: 227.900.000 km\n")

escreva("Duração do dia: 1 dia 0h 37m\n")

escreva("Distância da Terra: 54.600.000 km\n")

escreva("Área da superfície: $144.400.000 \text{ km}^2$ \n")

// Exibe o botão para visualizar o modelo 3D de Marte

escreva("\nClique para ver o modelo 3D de Marte:

https://solarsystem.nasa.gov/gltf_embed/2372/\n")

```
// Exibe as obras de Marte com opção para mostrar a imagem
para cada item em Model
```

```
    escreva("\nTítulo da Obra: ", item.Titulo_Obra, "\n")
    escreva("Descrição da Obra: ", item.Descricao_Obra, "\n")
    escreva("Clique para ver a imagem da obra\n")
    escreva("Imagem da obra: ", item.Imagem_Obra, "\n")
```

```
// Simula o comportamento de mostrar/ocultar a imagem
    escreva("Deseja ver a imagem da obra? (S/N): ")
    leia(resposta)
```

```
    se resposta == "S" ou resposta == "s" então
        escreva("Exibindo imagem: ", item.Imagem_Obra, "\n")
    senão
        escreva("Imagem não exibida.\n")
    fimse
fimpara
```

Web config

```
algoritmo "configuracao_servidor"
```

```
inicio
```

```
    // Definindo variáveis para simular as configurações
```

```
    cadeia configuracao_razor <- "system.web.webPages.razor"
```

```
    cadeia host_factory_type <- "System.Web.Mvc.MvcWebRazorHostFactory"
```

```
    cadeia page_base_type <- "System.Web.Mvc.WebViewPage"
```

```
    cadeia namespaces <- "System.Web.Mvc, System.Web.Mvc.Ajax,
System.Web.Mvc.Html, System.Web.Optimization, System.Web.Routing, MuseuPim"
```

```
    cadeia webpages_enabled <- "false"
```

```
// Exibindo configurações simuladas
```

```
    escreva("Configuração do Razor Pages: ", configuracao_razor, "\n")
```

```

escreva("Tipo de Factory para Host: ", host_factory_type, "\n")
escreva("Tipo base da Página: ", page_base_type, "\n")
escreva("Namespaces utilizados: ", namespaces, "\n")
escreva("Configuração 'webpages:Enabled': ", webpages_enabled, "\n")

// Simulando a remoção de handler e adição de um novo handler
escreva("Removendo o handler 'BlockViewHandler'.\n")
escreva("Adicionando novo handler 'BlockViewHandler' com path '*' e verbo
'*' no modo integrado.\n")

// Simulação de compilação e adição de assemblies
escreva("Compilando com a versão do assembly 'System.Web.Mvc,
Version=5.2.9.0'.\n")
finalgoritmo

```

Singuphtml

algoritmo "Cadastro Visitante"

inicio

// Definindo variáveis para armazenar os dados do visitante

cadeia nome_visitante, email_visitante, senha_visitante, resenha_visitante

inteiro idade_visitante

cadeia mensagem_validacao

// Exibindo o título da página

escreva("Cadastre-se\n")

// Coletando informações do visitante

escreva("Digite seu nome: ")

```
leia(nome_visitante)
```

```
escreva("Digite sua idade: ")
```

```
leia(idade_visitante)
```

```
escreva("Digite seu e-mail: ")
```

```
leia(email_visitante)
```

```
escreva("Digite sua senha: ")
```

```
leia(senha_visitante)
```

```
escreva("Digite a senha novamente: ")
```

```
leia(resenha_visitante)
```

```
// Validação simples
```

```
se nome_visitante = "" ou email_visitante = "" ou senha_visitante = "" ou  
resenha_visitante = "" então
```

```
    mensagem_validacao <- "Todos os campos são obrigatórios."
```

```
    escreva(mensagem_validacao, "\n")
```

```
senao
```

```
    se senha_visitante != resenha_visitante então
```

```
        mensagem_validacao <- "As senhas não coincidem."
```

```
        escreva(mensagem_validacao, "\n")
```

```
senao
```

```
    mensagem_validacao <- "Cadastro realizado com sucesso!"
```

```
    escreva(mensagem_validacao, "\n")
```

```
    fimse
```

```
fimse
```

```
// Simulando um formulário com fundo e estilo (não pode ser implementado  
diretamente em Portugal)
```

```
    escreva("Formulário exibido com estilo, fundo e animação de entrada  
(simulado).\n")
```

```
fimalgoritmo
```


Quizhtml

algoritmo "Quiz_Simples"

inicio

 // Definindo as perguntas e respostas

 perguntas <- [

 {pergunta: "1. Quais são os quatro planetas gasosos do Sistema Solar?",
alternativas: ["a) Mercúrio, Vênus, Terra, Marte", "b) Terra, Vênus, Marte, Júpiter", "c)
Mercúrio, Terra, Júpiter, Saturno", "d) Júpiter, Saturno, Urano, Netuno"], resposta: 3},

 {pergunta: "2. Por que Marte é conhecido como o 'Planeta Vermelho'",
alternativas: ["a) Por causa da lava em sua superfície", "b) Devido à presença de ferro
metálico", "c) Por causa de sua proximidade com o Sol", "d) Devido à presença de
óxido de ferro em sua superfície"], resposta: 3},

 {pergunta: "3. Que erro de tradução incentivou a ideia de canais artificiais
em Marte?", alternativas: ["a) 'Estradas' traduzidas como 'rios'", "b) 'Planícies' traduzidas
como 'montanhas'", "c) 'Vales' traduzidos como 'rios'", "d) 'Canalis' traduzido como
'canais'"]}, resposta: 3},

 // Adicionar o restante das perguntas

]

indice_pergunta <- 1

pontuacao <- 0

funcao exibir_pergunta()

 pergunta_atual <- perguntas[indice_pergunta]

 escreva(pergunta_atual.pergunta)

 para cada opcao em pergunta_atual.alternativas faca

 escreva(opcao)

 fim para

fim funcao

funcao lidar_com_resposta(indice_selecionado)

 se indice_selecionado = perguntas[indice_pergunta].resposta entao

```
        pontuacao <- pontuacao + 1
    fim se

    indice_pergunta <- indice_pergunta + 1
    se indice_pergunta <= comprimento(perguntas) então
        exibir_pergunta()
    senao
        exibir_resultado()
    fim se
fim funcao

funcao exibir_resultado()
    escreva("Você acertou ", pontuacao, " de ", comprimento(perguntas), "
perguntas.")
fim funcao

funcao reiniciar_quiz()
    pontuacao <- 0
    indice_pergunta <- 1
    exibir_pergunta()
fim funcao

// Iniciar o quiz
exibir_pergunta()
fimalgoritmo
```

Loginhtml home

algoritmo "login_visitante"

inicio

 // Declaração de variáveis

 caractere nome_visitante

 caractere senha_visitante

 caractere notificacao

 // Solicitar o nome do visitante

 escreva("Digite seu nome: ")

 leia(nome_visitante)

 // Solicitar a senha do visitante

 escreva("Digite sua senha: ")

 leia(senha_visitante)

 // Verificar as credenciais (validação fictícia)

 se (nome_visitante = "usuario" e senha_visitante = "senha123") então

 notificacao <- "Login bem-sucedido!"

 senao

 notificacao <- "Nome ou senha inválidos."

 fimse

 // Exibir a notificação

 escreva(notificacao)

fimalgoritmo

Indexfeedback

algoritmo "feedback_visitante"

inicio

 // Declaração de variáveis

 tipo Feedback

 inteiro ID_Feedback

 caractere NomeF_Visitante

 caractere Feedback_Visitante

 caractere Nivel_Satisfacao

 fimtipo

 // Lista de feedbacks dos visitantes

 lista<Feedback> feedbacks

 // Preenchendo a lista de feedbacks (simulação)

 adicionar(feedbacks, Feedback{1, "João Silva", "Muito bom!", "Satisfeito"})

 adicionar(feedbacks, Feedback{2, "Maria Oliveira", "Excelente serviço!",
"Muito Satisfeito"})

 adicionar(feedbacks, Feedback{3, "Carlos Souza", "Pode melhorar.",
"Neutro"})

 // Exibindo os feedbacks

 escreva("Feedback do visitante\n")

 escreva("-----\n")

 escreva("ID | Nome do Visitante | Feedback | Nível de Satisfação\n")

 escreva("-----\n")

 para cada feedback em feedbacks faça

 escreva(feedback.ID_Feedback, " | ", feedback.NomeF_Visitante, " | ",
feedback.Feedback_Visitante, " | ", feedback.Nivel_Satisfacao, "\n")

 fimpara

```
// Mensagem final de navegação  
    escreva("\nClique em [Voltar] para retornar.\n")  
finalgoritmo
```

Indexhtml

```
algoritmo "relatorio_visitantes"
```

```
inicio
```

```
    // Definindo o tipo de dados para cada visitante
```

```
    tipo Visitante
```

```
        inteiro ID_Visitante
```

```
        caractere Nome_Visitante
```

```
        inteiro Idade_Visitante
```

```
        caractere Email_Visitante
```

```
    fimtipo
```

```
    // Criando uma lista de visitantes
```

```
    lista<Visitante> visitantes
```

```
    // Preenchendo a lista de visitantes com dados fictícios
```

```
    adicionar(visitantes, Visitante{1, "João Silva", 30, "joao.silva@gmail.com"})
```

```
    adicionar(visitantes,      Visitante{2,      "Maria      Oliveira",      25,
"maria.oliveira@gmail.com"})
```

```
    adicionar(visitantes,      Visitante{3,      "Carlos      Souza",      40,
"carlos.souza@hotmail.com"})
```

```
    // Exibindo o título
```

```
    escreva("Relatório de Visitantes\n")
```

```
    escreva("-----\n")
```

```
    escreva("ID | Nome do Visitante      | Idade | E-mail\n")
```

```
    escreva("-----\n")
```

```
    // Exibindo os dados dos visitantes
```

```
    para cada visitante em visitantes faca
```

```
        escreva(visitante.ID_Visitante, " | ", visitante.Nome_Visitante, " | ",
visitante.Idade_Visitante, " | ", visitante.Email_Visitante, "\n")
```

```
    fimpara
```

```
// Mensagem de navegação  
    escreva("\nClique em [Voltar] para retornar.\n")  
finalgoritmo
```

Home html

algoritmo "home_page"

inicio

// Exibindo a saudação e o título

escreva("Bem-vindo ao Museu Digital de Marte!\n")

escreva("-----\n")

// Exibindo o parágrafo de descrição com indentação

escreva("Explore o fascinante mundo do Planeta Vermelho através de nossa exposição virtual, onde a ciência e a imaginação se encontram para desvendar os mistérios de Marte. Desde suas paisagens deslumbrantes e desertas até os avanços mais recentes na exploração espacial, nosso museu oferece uma jornada interativa através do cosmos. Descubra a História de Marte: Viaje no tempo e veja como Marte passou de um planeta misterioso no céu noturno para um alvo principal das missões espaciais. Conheça a evolução das teorias sobre o planeta e os marcos importantes da exploração marciana.\n")

// Simulando uma indentação do texto

escreva(" - Indentação simulada: Como Marte passou a ser o foco das missões espaciais.\n")

escreva("-----\n")

// Mensagem simulando a logo do museu

escreva("Logo do Museu: [Imagem do Museu não disponível no Portugol]\n")

escreva("-----\n")

// Encerrando o algoritmo

escreva("Obrigado por visitar o nosso museu!\n")

fimalgoritmo

Feedback

algoritmo "feedback"

inicio

 // Exibindo o título e a introdução do feedback

 escreva("Feedback")

 escreva("\n")

 // Simulando a captura de dados do formulário

 nome_visitante: cadeia

 feedback_visitante: cadeia

 nivel_satisfacao: inteiro

 // Entrada do nome do visitante

 escreva("Digite seu nome: ")

 leia(nome_visitante)

 // Entrada do feedback

 escreva("Digite seu feedback: ")

 leia(feedback_visitante)

 // Entrada do nível de satisfação (1 a 5 estrelas)

 escreva("Digite seu nível de satisfação (1 a 5): ")

 leia(nivel_satisfacao)

 // Verificando se o nível de satisfação é válido

 se nivel_satisfacao >= 1 e nivel_satisfacao <= 5 entao

 escreva("Nível de satisfação registrado: ", nivel_satisfacao, " estrelas.\n")

 senao

 escreva("Por favor, insira um nível de satisfação válido entre 1 e 5.\n")

 fimse

 // Simulando o envio do feedback

```
escreva("Feedback enviado com sucesso. Obrigado pelo seu comentário!\n")
```

```
// Estilos para a interface
```

```
escreva("\nEstilos: \n")
```

```
escreva("Fundo: Gradiente de cores de #3c3b3f a #605c3c\n")
```

```
escreva("Cor do texto: Branco\n")
```

```
escreva("Fonte: Montserrat\n")
```

```
escreva("Botão: Cor de fundo #ff7851, com efeito hover para #ff5733\n")
```

```
fimalgoritmo
```

Cadastro html

algoritmo "Cadastro"

inicio

 // Exibir título do formulário

 escreva("Realize seu Cadastro\n")

 // Coletar os dados do usuário

 escreva("Digite seu nome: ")

 leia(nome)

 escreva("Digite seu e-mail (opcional): ")

 leia(email)

 escreva("Digite sua idade: ")

 leia(idade)

 escreva("Digite sua senha: ")

 leia(senha)

 // Processar cadastro

 se nome <> "" e idade <> "" e senha <> "" então

 // Cadastro bem-sucedido

 escreva("Cadastro realizado com sucesso!\n")

 senao

 // Dados inválidos ou incompletos

 escreva("Erro: Por favor, preencha todos os campos obrigatórios.\n")

 fimse

fimalgoritmo

Indexobrasadm

```
programa
```

```
{
```

```
    // Estrutura para armazenar informações sobre as obras
```

```
    tipo Obra {
```

```
        inteiro idObra
```

```
        cadeia titulo
```

```
        cadeia descricao
```

```
        cadeia imagem
```

```
    }
```

```
    // Lista de obras
```

```
    vetor<Obra> obras[3]
```

```
    funcao inicio()
```

```
{
```

```
    // Inicialização dos dados das obras
```

```
    obras[0].idObra <- 1
```

```
    obras[0].titulo <- "Obra 1"
```

```
    obras[0].descricao <- "Descrição da Obra 1"
```

```
    obras[0].imagem <- "imagem1.jpg"
```

```
    obras[1].idObra <- 2
```

```
    obras[1].titulo <- "Obra 2"
```

```

obras[1].descricao <- "Descrição da Obra 2"

obras[1].imagem <- "imagem2.jpg"


obras[2].idObra <- 3

obras[2].titulo <- "Obra 3"

obras[2].descricao <- "Descrição da Obra 3"

obras[2].imagem <- "imagem3.jpg"


// Exibição do título

escreva("=====\\n")

escreva("      Obras Administrador      \\n")

escreva("=====\\n")


// Cabeçalho da tabela

escreva("ID\\tTítulo\\t\\tDescrição\\t\\tImagem\\n")

escreva("-----\\n")


// Exibição das obras na tabela

para (inteiro i = 0; i < comprimento(obras); i++) {

    escreva(obras[i].idObra, "\\t", obras[i].titulo, "\\t", obras[i].descricao, "\\t",
obras[i].imagem, "\\n")

}


escreva("\\n[1] Editar Obra\\n[2] Excluir Obra\\n[3] Voltar\\n")

```

```
    escolhaUsuario()
}

funcao escolhaUsuario()
{
    inteiro opcao

    escreva("\nEscolha uma opção: ")

    leia(opcao)

    escolha (opcao)
    {
        caso 1:

            escreva("Opção Editar selecionada.\n")

            // Implementação para editar uma obra

            pare

        caso 2:

            escreva("Opção Excluir selecionada.\n")

            // Implementação para excluir uma obra

            pare

        caso 3:

            escreva("Voltando ao menu inicial...\n")

            // Implementação para retornar

            pare

        caso contrario:
```

```
        escreva("Opção inválida. Tente novamente.\n")
        escolhaUsuario()
    }
}
}
```

Layout

programa

```
{  
  
    funcao inicio()  
  
    {  
  
        inteiro opcao  
  
  
        // Cabeçalho do "site"  
  
        escreva("=====\n")  
  
        escreva("    Stay Of Mars\n")  
  
        escreva("=====\n")  
  
  
        // Menu de navegação  
  
        escreva("Menu de Navegação:\n")  
  
        escreva("[1] Obras\n")  
  
        escreva("[2] Quiz\n")  
  
        escreva("[3] Feedback\n")  
  
        escreva("[4] Login\n")  
  
        escreva("[5] Cadastro\n")  
  
        escreva("[0] Sair\n")  
  
  
        // Captura de opção do usuário  
  
        escolhaUsuario()  
  
    }
```



```
funcao escolhaUsuario()
{
    inteiro opcao

    escreva("\nEscolha uma opção: ")

    leia(opcao)

    escolha (opcao)
    {
        caso 1:
            exibirObras()

            pare

        caso 2:
            exibirQuiz()

            pare

        caso 3:
            exibirFeedback()

            pare

        caso 4:
            login()

            pare

        caso 5:
            cadastro()

            pare
```

caso 0:

```
    escreva("Saindo do programa...\n")
```

```
    pare
```

caso contrario:

```
    escreva("Opção inválida. Tente novamente.\n")
```

```
    escolhaUsuario()
```

```
}
```

```
}
```

funcao exibirObras()

```
{
```

```
    escreva("Você está na página de Obras.\n")
```

```
    escreva("Retornando ao menu principal...\n")
```

```
    inicio()
```

```
}
```

funcao exibirQuiz()

```
{
```

```
    escreva("Você está na página de Quiz.\n")
```

```
    escreva("Retornando ao menu principal...\n")
```

```
    inicio()
```

```
}
```

funcao exibirFeedback()

```
{  
    escreva("Você está na página de Feedback.\n")  
    escreva("Retornando ao menu principal...\n")  
    inicio()  
}
```

funcao login()

```
{  
    cadeia usuario, senha  
    escreva("==== Login ==== \n")  
    escreva("Usuário: ")  
    leia(usuario)  
    escreva("Senha: ")  
    leia(senha)  
  
    se (usuario == "admin" e senha == "1234") entao  
    {  
        escreva("Login realizado com sucesso!\n")  
    }  
    senao  
    {  
        escreva("Usuário ou senha inválidos. Tente novamente.\n")  
        login()  
    }  
}
```

```
    inicio()
```

```
}
```

```
funcao cadastro()
```

```
{
```

Validação

programa

{

// Variáveis globais

cadeia mensagem

inteiro id

funcao inicio()

{

// Lista de dados simulada (ID, Nome, RG, CPF)

cadeia dadosPessoa[4] = {"123", "João", "123456789", "12345678901"}

mensagem = ""

validarVisitante(dadosPessoa)

// Exibir mensagens de erro, se existirem

se (mensagem != "")

{

escreva("Erros encontrados:\n", mensagem)

}

senao

{

escreva("Todos os dados estão corretos.\n")

}

```
}
```

```
funcao validarId(cadeia identificacao)
```

```
{
```

```
    // Inicializar mensagem como vazia
```

```
    mensagem = ""
```

```
    tente
```

```
    {
```

```
        // Tentar converter para inteiro
```

```
        id = inteiro(identificacao)
```

```
    }
```

```
    // Capturar erro de conversão
```

```
    pega (excecao)
```

```
    {
```

```
        mensagem = mensagem + "Erro de ID: O valor fornecido não é um  
número válido.\n"
```

```
    }
```

```
}
```

```
funcao validarVisitante(cadeia listaDadosPessoa[4])
```

```
{
```

```
    mensagem = ""
```

```
    validarId(listaDadosPessoa[0])
```

```
// Validar o nome

se (comprimento(listaDadosPessoa[1]) < 3)

{

    mensagem = mensagem + "Erro: Nome deve ter mais que 3
caracteres.\n"

}

se (comprimento(listaDadosPessoa[1]) > 50)

{

    mensagem = mensagem + "Erro: Nome não pode ter mais que 50
caracteres.\n"

}


// Validar o RG

se (comprimento(listaDadosPessoa[2]) > 10)

{

    mensagem = mensagem + "Erro: RG deve ter menos que 10
caracteres.\n"

}


// Validar o CPF

se (comprimento(listaDadosPessoa[3]) > 12)

{

    mensagem = mensagem + "Erro: CPF deve ter menos que 12
caracteres.\n"

}
```

Visitante

programa

```
{  
  
    // Estrutura para armazenar os dados do Visitante  
  
    tipo Visitante  
  
    {  
  
        inteiro id  
  
        cadeia nome  
  
        cadeia email  
  
        cadeia idade  
  
        cadeia senha  
  
    }  
  
  
    funcao inicio()  
  
    {  
  
        // Criando um visitante e configurando seus valores  
  
        Visitante visitante  
  
  
        // Configurando valores usando os "setters"  
  
        setId(visitor, 1)  
  
        setName(visitor, "João Silva")  
  
        setEmail(visitor, "joao@email.com")  
  
        setIdade(visitor, "25")  
  
        setSenha(visitor, "senha123")  
    }  
}
```



```
// Exibindo os valores usando os "getters"

escreva("Dados do Visitante:\n")

escreva("ID: ", getId(visitante), "\n")

escreva("Nome: ", getNome(visitante), "\n")

escreva("Email: ", getEmail(visitante), "\n")

escreva("Idade: ", getIdade(visitante), "\n")

escreva("Senha: ", getSenha(visitante), "\n")

}


// Métodos para acessar (get) e modificar (set) os atributos do visitante


funcao setId(Visitante v, inteiro novold)

{

    v.id = novold

}


funcao getId(Visitante v) : inteiro

{

    retorne v.id

}


funcao setNome(Visitante v, cadeia novoNome)

{
```

```
    v.nome = novoNome  
}
```

```
funcao getNome(Visitante v) : cadeia  
{  
    retorne v.nome  
}
```

```
funcao setEmail(Visitante v, cadeia novoEmail)  
{  
    v.email = novoEmail  
}
```

```
funcao getEmail(Visitante v) : cadeia  
{  
    retorne v.email  
}
```

```
funcao setIdade(Visitante v, cadeia novaldade)  
{  
    v.idade = novaldade  
}
```

```
funcao getIdade(Visitante v) : cadeia
```

```
{  
    retorne v.idade  
}
```

funcao setSenha(Visitante v, cadeia novaSenha)

```
{  
    v.senha = novaSenha  
}
```

funcao getSenha(Visitante v) : cadeia

```
{  
    retorne v.senha  
}
```

ANEXO 6 – ALGORITIMOS DESKTOP

Algoritmo "Cadastrar Obra"

```
// Declaração de variáveis
cadeia tituloObra <- ""
cadeia descricaoObra <- ""
cadeia imagemSelecionada <- "Nenhuma"
cadeia resultadoCadastro <- ""

funcao iniciar()
    inteiro opção
    repetir
        limparTela()
        // Interface inicial do sistema
        escreva("\n=== Cadastro de Obra ===\n")
        escreva("1. Inserir Título da Obra (Atual: ", tituloObra, ")\n")
        escreva("2. Inserir Descrição da Obra (Atual: ", descricaoObra, ")\n")
        escreva("3. Selecionar Imagem (Atual: ", imagemSelecionada, ")\n")
        escreva("4. Cadastrar Obra\n")
        escreva("5. Voltar ao Menu Principal\n")
        escreva("Escolha uma opção: ")
        leia(opcao)
        escolha opcao
            caso 1
                tituloObra <- inserirTitulo()
            caso 2
                descricaoObra <- inserirDescricao()
            caso 3
                imagemSelecionada <- selecionarImagem()
            caso 4
                resultadoCadastro <- cadastrarObra(tituloObra, descricaoObra,
imagemSelecionada)
```

```

        escreva("\nResultado: ", resultadoCadastro, "\n")
        escreva("\nPressione qualquer tecla para continuar...")
        leia() // Pausa para o usuário ver o resultado
    caso 5
        escreva("\nVoltando ao menu principal...\n")
        pause(2)
        parar
    caso contrario
        escreva("\nOpção inválida. Tente novamente.\n")
        pause(2)
    fimescolha
ate que falso
fimfuncao

// Função para inserir o título da obra
funcao cadeia inserirTitulo()
    cadeia novoTitulo
    escreva("\nInsira o título da obra: ")
    leia(novoTitulo)
    retorne novoTitulo
fimfuncao

// Função para inserir a descrição da obra
funcao cadeia inserirDescricao()
    cadeia novaDescricao
    escreva("\nInsira a descrição da obra (máx 200 caracteres): ")
    leia(novaDescricao)
    se comprimento(novaDescricao) > 200 entao
        escreva("\nErro: Descrição muito longa. Use no máximo 200
caracteres.\n")
        pause(2)
    retorne descricaoObra

```

```

fimse
retorne novaDescricao
fimfuncao

```

```

// Função para simular a seleção de uma imagem
funcao cadeia selecionarImagem()
    cadeia imagem
    escreva("\nDigite o nome do arquivo da imagem (Ex.: imagem.jpg): ")
    leia(imagem)
    se comprimento(imagem) > 0 entao
        retorne imagem
    senao
        escreva("\nErro: Nenhuma imagem foi selecionada.\n")
        pause(2)
        retorne imagemSelecionada
    fimse
fimfuncao

```

```

// Função para realizar o cadastro
funcao cadeia cadastrarObra(cadeia titulo, cadeia descricao, cadeia imagem)
    se (titulo == "") ou (descricao == "") ou (imagem == "Nenhuma") entao
        retorne "Erro: Todos os campos devem ser preenchidos antes de
cadastrar."
    senao
        // Simula o salvamento dos dados
        escreva("\nCadastrando obra...")
        pause(2)
        retorne "Cadastro realizado com sucesso!"
    fimse
fimfuncao

```

Algoritmo "ExcluirEditarObras"

```

// Declaração de variáveis
cadeia idObra <- ""
cadeia nomeObra <- ""
cadeia descricaoObra <- ""
cadeia imagemObra <- ""
cadeia mensagem <- ""

funcao iniciar()
    inteiro opcao
    repetir
        limparTela()
        // Interface inicial do sistema
        escreva("\n=== Excluir/Editar Obras ===\n")
        escreva("1. Pesquisar Obra por ID\n")
        escreva("2. Inserir Nome da Obra para Pesquisa\n")
        escreva("3. Editar Dados da Obra\n")
        escreva("4. Excluir Obra\n")
        escreva("5. Voltar ao Menu Principal\n")
        escreva("Escolha uma opção: ")
        leia(opcao)

    escolha opcao
        caso 1
            idObra <- pesquisarPorID()
        caso 2
            nomeObra <- pesquisarPorNome()
        caso 3
            editarObra()
        caso 4
            excluirObra()
        caso 5

```

```

        escreva("\nVoltando ao menu principal...\n")
        pause(2)
        parar
    caso contrario
        escreva("\nOpção inválida. Tente novamente.\n")
        pause(2)
    fimescolha
ate que falso
fimfuncao

```

```
// Função para pesquisar por ID
```

```
funcao cadeia pesquisarPorID()
```

```
cadeia id
```

```
escreva("\nDigite o ID da obra para pesquisar: ")
```

```
leia(id)
```

```
// Simulação de busca
```

```
se id == "123" entao
```

```
    escreva("\nObra encontrada: ID = 123\n")
```

```
    nomeObra <- "Obra Exemplo"
```

```
    descricaoObra <- "Descrição Exemplo"
```

```
    imagemObra <- "imagem.jpg"
```

```
    pause(2)
```

```
    retorne id
```

```
senao
```

```
    escreva("\nObra não encontrada.\n")
```

```
    pause(2)
```

```
    retorne ""
```

```
fimse
```

```
fimfuncao
```

```
// Função para pesquisar por nome
```



```

funcao cadeia pesquisarPorNome()
  cadeia nome
  escreva("\nDigite o nome da obra para pesquisar: ")
  leia(nome)

  // Simulação de busca
  se nome == "Obra Exemplo" entao
    escreva("\nObra encontrada: Nome = Obra Exemplo\n")
    idObra <- "123"
    descricaoObra <- "Descrição Exemplo"
    imagemObra <- "imagem.jpg"
    pause(2)
    retorne nome
  senao
    escreva("\nObra não encontrada.\n")
    pause(2)
    retorne ""
  fimse
fimfuncao

// Função para editar os dados da obra
funcao editarObra()
  se idObra == "" entao
    escreva("\nErro: Nenhuma obra selecionada para edição.\n")
    pause(2)
    retorne
  fimse

  escreva("\n=== Editar Obra ===\n")
  escreva("Título atual: ", nomeObra, "\n")
  escreva("Digite o novo título: ")
  leia(nomeObra)

```

```

    escreva("Descrição atual: ", descricaoObra, "\n")
    escreva("Digite a nova descrição: ")
    leia(descricaoObra)

    escreva("Imagem atual: ", imagemObra, "\n")
    escreva("Digite o novo nome da imagem: ")
    leia(imagemObra)

    escreva("\nObra editada com sucesso!\n")
    pause(2)
fimfuncao

// Função para excluir uma obra
funcao excluirObra()
    se idObra == "" entao
        escreva("\nErro: Nenhuma obra selecionada para exclusão.\n")
        pause(2)
        retorne
    fimse

    escreva("\nTem certeza que deseja excluir a obra (ID: ", idObra, ")? (s/n): ")
    cadeia confirmacao
    leia(confirmacao)

    se confirmacao == "s" ou confirmacao == "S" entao
        escreva("\nExcluindo obra...\n")
        pause(2)
        escreva("\nObra excluída com sucesso!\n")
        // Resetando as variáveis
        idObra <- ""
        nomeObra <- ""

```

```
        descricaoObra <- ""  
        imagemObra <- ""  
    senao  
        escreva("\nExclusão cancelada.\n")  
    fimse  
    pause(2)  
fimfuncao  
finalgoritmo
```

```

// Variáveis
cadeia nomeUsuario <- ""
cadeia textoFeedback <- ""
inteiro nivelSatisfacao <- 0

funcao iniciar()
  inteiro opcao
  repetir
    limparTela()
    escreval("\n=== Stay Of Mars ===")
    escreval("1. Home")
    escreval("2. Obras")
    escreval("3. Quiz")
    escreval("4. Feedback")
    escreval("5. Sair")
    escreva("Escolha uma opção: ")
    leia(opcao)

  escolha opcao
    caso 1
      escreverHome()
    caso 2
      escreverObras()
    caso 3
      escreverQuiz()
    caso 4
      menuFeedback()
    caso 5
      escreval("\nObrigado por utilizar Stay Of Mars!")
      parar
    caso contrario
      escreval("\nOpção inválida!")

```

```
        fimescolha
    ate que falso
fimfuncao
```

```
funcao escreverHome()
    escreval("\n=== Home ===")
    escreval("Bem-vindo ao Stay Of Mars!")
    pause()
fimfuncao
```

```
funcao escreverObras()
    escreval("\n=== Obras ===")
    escreval("Informações sobre nossas obras em breve!")
    pause()
fimfuncao
```

```
funcao escreverQuiz()
    escreval("\n=== Quiz ===")
    escreval("Prepare-se para novos desafios em breve!")
    pause()
fimfuncao
```

```
funcao menuFeedback()
    limparTela()
    escreval("\n=== Feedback ===")
    escreva("Digite seu nome: ")
    leia(nomeUsuario)

    escreva("Deixe seu feedback: ")
    leia(textoFeedback)

    escreva("Nível de satisfação (1 a 5): ")
```

```
leia(nivelSatisfacao)
```

```
enquanto nivelSatisfacao < 1 ou nivelSatisfacao > 5 faca
```

```
    escreva("Nível inválido! Escolha entre 1 e 5: ")
```

```
    leia(nivelSatisfacao)
```

```
fimenquanto
```

```
    enviarFeedback()
```

```
fimfuncao
```

```
funcao enviarFeedback()
```

```
    escreval("\nEnviando feedback...")
```

```
    escreval("Obrigado, ", nomeUsuario, "!")
```

```
    escreval("Nível de Satisfação: ", nivelSatisfacao)
```

```
    escreval("Seu Feedback: ", textoFeedback)
```

```
    pause()
```

```
fimfuncao
```

```
fimalgoritmo
```

Algoritmo "Homeadm"

```
// Variáveis
```

```
cadeia nomeUsuario <- ""
```

```
cadeia textoFeedback <- ""
```

```
inteiro nivelSatisfacao <- 0
```

```
funcao iniciar()
```

```
    inteiro opcao
```

```
    repetir
```

```
        limparTela()
```

```
        escreval("\n=== Stay Of Mars ===")
```

```
        escreval("1. Home")
```

```
        escreval("2. Obras")
```

```
        escreval("3. Quiz")
```

```
        escreval("4. Feedback")
```

```
        escreval("5. Sair")
```

```
        escreva("Escolha uma opção: ")
```

```
        leia(opcao)
```

```
    escolha opcao
```

```
        caso 1
```

```
            escreverHome()
```

```
        caso 2
```

```
            escreverObras()
```

```
        caso 3
```

```
            escreverQuiz()
```

```
        caso 4
```

```
            menuFeedback()
```

```
        caso 5
```

```
            escreval("\nObrigado por utilizar Stay Of Mars!")
```

```
            parar
```

```
        caso contrario
```

```
        escreval("\nOpção inválida!")
    fimsecolha
    ate que falso
fimfuncao
```

```
funcao escreverHome()
    escreval("\n=== Home ===")
    escreval("Bem-vindo ao Stay Of Mars!")
    pause()
fimfuncao
```

```
funcao escreverObras()
    escreval("\n=== Obras ===")
    escreval("Informações sobre nossas obras em breve!")
    pause()
fimfuncao
```

```
funcao escreverQuiz()
    escreval("\n=== Quiz ===")
    escreval("Prepare-se para novos desafios em breve!")
    pause()
fimfuncao
```

```
funcao menuFeedback()
    limparTela()
    escreval("\n=== Feedback ===")
    escreva("Digite seu nome: ")
    leia(nomeUsuario)

    escreva("Deixe seu feedback: ")
    leia(textoFeedback)
```



```
escreva("Nível de satisfação (1 a 5): ")
```

```
leia(nivelSatisfacao)
```

```
enquanto nivelSatisfacao < 1 ou nivelSatisfacao > 5 faca
```

```
    escreva("Nível inválido! Escolha entre 1 e 5: ")
```

```
    leia(nivelSatisfacao)
```

```
fimenquanto
```

```
    enviarFeedback()
```

```
fimfuncao
```

```
funcao enviarFeedback()
```

```
    escreval("\nEnviando feedback...")
```

```
    escreval("Obrigado, ", nomeUsuario, "!")
```

```
    escreval("Nível de Satisfação: ", nivelSatisfacao)
```

```
    escreval("Seu Feedback: ", textoFeedback)
```

```
    pause()
```

```
fimfuncao
```

```
fimalgoritmo
```

Algoritmo "ListaFeedbacks"

// Variáveis

vetor[1..100] texto feedbacks // Simulação de uma lista de feedbacks

inteiro totalFeedbacks, i

funcao iniciar()

// Inicializar a lista de feedbacks (simulação)

inicializarFeedbacks()

repetir

limparTela()

// Título

escreval("=====")

escreval(" Lista de Feedbacks ")

escreval("=====")

// Exibir feedbacks

exibirFeedbacks()

escreval("")

escreval("1. Voltar para Home")

escreva("Escolha uma opção: ")

leia(i)

escolha i

caso 1

voltarParaHome()

parar

caso contrario

escreval("Opção inválida! Tente novamente.")

pause()

fimescolha

```
    ate que falso  
fimfuncao
```

```
funcao inicializarFeedbacks()  
    // Simulação de feedbacks adicionados à lista  
    feedbacks[1] <- "Ótimo sistema, muito intuitivo!"  
    feedbacks[2] <- "Precisa de melhorias no desempenho."  
    feedbacks[3] <- "Gostei muito do design e da experiência!"  
    totalFeedbacks <- 3  
fimfuncao
```

```
funcao exibirFeedbacks()  
    se totalFeedbacks > 0 entao  
        para i de 1 ate totalFeedbacks faca  
            escreval(i, ". ", feedbacks[i])  
        fimpara  
    senao  
        escreval("Nenhum feedback disponível.")  
    fimse  
fimfuncao
```

```
funcao voltarParaHome()  
    escreval("Voltando para a tela inicial...")  
    pause()  
fimfuncao  
finalgoritmo
```

Algoritmo "LoginADM"

```
// Variáveis
```

```
texto nomeAdmin, senhaAdmin
```

```
texto nomeCorreto <- "admin" // Nome de usuário pré-definido
```

```
texto senhaCorreta <- "1234" // Senha pré-definida
```

```
inteiro opcao
```

```
funcao iniciar()
```

```
  repetir
```

```
    limparTela()
```

```
    escreval("=====")
```

```
    escreval("      Sistema de Administração      ")
```

```
    escreval("=====")
```

```
    // Entrada do nome do administrador
```

```
    escreva("Digite o Nome do Admin: ")
```

```
    leia(nomeAdmin)
```

```
    // Entrada da senha
```

```
    escreva("Digite a Senha: ")
```

```
    leia(senhaAdmin)
```

```
    // Verificação de credenciais
```

```
    se validarLogin(nomeAdmin, senhaAdmin) entao
```

```
        escreval("")
```

```
        escreval("Login realizado com sucesso!")
```

```
        pause()
```

```
        acessarDashboard()
```

```
        parar
```

```
    senao
```

```
        escreval("")
```

```
        escreval("Nome ou senha incorretos. Tente novamente.")
```

```
        pause()
    fimse

    escreval("")
    escreval("1. Tentar novamente")
    escreval("2. Voltar para o menu anterior")
    escreva("Escolha uma opção: ")
    leia(opcao)

    se opcao = 2 entao
        voltarMenu()
        parar
    fimse
ate que falso
fimfuncao

funcao booleano validarLogin(texto nome, texto senha)
    se nome = nomeCorreto e senha = senhaCorreta entao
        retorne verdadeiro
    senao
        retorne falso
    fimse
fimfuncao

funcao acessarDashboard()
    escreval("Bem-vindo à Dashboard Administrativa!")
    // Aqui adicionaria funcionalidades do painel
    pause()
fimfuncao

funcao voltarMenu()
    escreval("Voltando para o menu inicial...")
```

```
    pause()  
fimfuncao  
finalgoritmo
```

Algoritmo "Obras"

```

// Variáveis

lista Obras

texto tituloObra, descricaoObra, imagemObra

inteiro opcao

// Função principal

funcao iniciar()

    // Exibir cabeçalho

    limparTela()

    escreval("=====")
    escreval("  Stay Of Mars  ")
    escreval("=====")

    // Exibir opções do menu

    escreval("1. Home")
    escreval("2. Obras")
    escreval("3. Quiz")
    escreval("4. Feedback")
    escreva("Escolha uma opção: ")
    leia(opcao)

    // Verificar opção selecionada

    se opcao = 1 entao
        home()
    senao se opcao = 2 entao
        obras()
    senao se opcao = 3 entao
        quiz()
    senao se opcao = 4 entao
        feedback()
    senao

```

```
        escreval("Opção inválida!")
    fimse
fimfuncao

// Função para mostrar informações sobre obras
funcao obras()
    limparTela()

    // Exibir título da página
    escreval("Obras - Descubra as curiosidades do planeta vermelho:")

    // Exibir informações sobre Marte
    escreval("Marte é um dos nove planetas do Sistema Solar. Sua órbita
ocupa a quarta posição a partir do Sol, o que significa que está posicionado entre a
Terra e Júpiter.")
    escreval("Marte possui um diâmetro de 6.794 km e sua massa é  $6,39 \times 10^{23}$ 
kg.")
    escreval("A gravidade em Marte é 3,73 m/s2, cerca de 38% da gravidade
da Terra.")

    // Exibir detalhes sobre Marte
    escreval("Luas: Fobos, Deimos")
    escreval("Distância do Sol: 227.900.000 km")
    escreval("Duração do dia: 1d 0h 37m")
    escreval("Distância da Terra: 54.600.000 km")
    escreval("Área da superfície: 144.400.000 km2")

    // Exibir obras
    escreval("Confira as obras relacionadas a Marte:")

    // Adicionar obras fictícias à lista
    adicionarObra("Obra 1", "Descrição da Obra 1", "imagens/obra1.jpg")
```



```

adicionarObra("Obra 2", "Descrição da Obra 2", "imagens/obra2.jpg")
adicionarObra("Obra 3", "Descrição da Obra 3", "imagens/obra3.jpg")

```

```

// Exibir a lista de obras
para cada obra em Obras faça
    escreval("Título: " + obra.Titulo)
    escreval("Descrição: " + obra.Descricao)
    escreval("Imagem: " + obra.Imagem)
    escreval("=====")
fimpara

```

```

// Voltar ao menu principal
escreval("1. Voltar para o menu")
escreva("Escolha uma opção: ")
leia(opcao)
se opcao = 1 então
    iniciar()
fimse
fimfuncao

```

```

// Função para adicionar uma obra à lista
funcao adicionarObra(texto titulo, texto descricao, texto imagem)
    // Criação de uma nova obra e adição à lista
    novaObra.Titulo <- titulo
    novaObra.Descricao <- descricao
    novaObra.Imagem <- imagem
    adicionar(Obras, novaObra)
fimfuncao

```

```

// Função para voltar ao menu inicial (home)
funcao home()
    escreval("Bem-vindo à Home!")

```

```
// Aqui você poderia colocar a lógica de navegação para a tela inicial
fimfuncao

// Função de Quiz
funcao quiz()
    escreval("Bem-vindo ao Quiz!")
    // Lógica do Quiz
fimfuncao

// Função de Feedback
funcao feedback()
    escreval("Bem-vindo ao Feedback!")
    // Lógica de Feedback
fimfuncao
finalgoritmo
```

Algoritmo "Quiz"

```

// Variáveis
inteiro opcao
texto perguntaAtual
lista alternativas
inteiro respostaCorreta
inteiro perguntaIndex

// Função principal
funcao iniciar()
    // Exibir cabeçalho
    limparTela()
    escreval("=====")
    escreval("  Stay Of Mars  ")
    escreval("=====")

    // Exibir opções do menu
    escreval("1. Home")
    escreval("2. Obras")
    escreval("3. Quiz")
    escreval("4. Feedback")
    escreva("Escolha uma opção: ")
    leia(opcao)

    // Verificar opção selecionada
    se opcao = 1 entao
        home()
    senao se opcao = 2 entao
        obras()
    senao se opcao = 3 entao
        quiz()
    senao se opcao = 4 entao

```

```

        feedback()
    senao
        escreval("Opção inválida!")
    fimse
fimfuncao

// Função do Quiz
funcao quiz()
    limparTela()

    // Título da tela
    escreval("Quiz de Conhecimentos")

    // Inicializar variáveis
    perguntaIndex <- 1
    perguntaAtual <- "Qual é a capital da França?"
    alternativas <- ["Paris", "Londres", "Roma", "Berlim"]
    respostaCorreta <- 1 // "Paris"

    // Exibir a pergunta e alternativas
    exibirPergunta()

    // Esperar pela resposta do usuário
    escreva("Escolha a alternativa (1-4): ")
    leia(opcao)

    // Verificar a resposta
    se opcao = respostaCorreta então
        escreval("Resposta correta!")
    senao
        escreval("Resposta incorreta! A resposta correta é: ",
alternativas[respostaCorreta])

```

```

fimse

// Perguntar se o usuário quer continuar
escreval("Deseja continuar com o quiz? (1 - Sim, 2 - Não)")
leia(opcao)
se opcao = 1 então
    perguntaIndex <- perguntaIndex + 1
    // Chamar próxima pergunta
    quiz()
senao
    escreval("Obrigado por participar do quiz!")
    iniciar()
fimse
fimfuncao

// Função para exibir a pergunta e alternativas
funcao exibirPergunta()
    // Exibir pergunta atual
    escreval(perguntaAtual)

    // Exibir alternativas
    para i de 1 ate comprimento(alternativas) faca
        escreval(i, ". ", alternativas[i])
    fimpara
fimfuncao

// Função para voltar à Home
funcao home()
    escreval("Bem-vindo à Home!")
    // Aqui você poderia colocar a lógica de navegação para a tela inicial
fimfuncao

```

```
// Função de Obras
funcao obras()
    escreval("Bem-vindo à tela de Obras!")
    // Lógica de obras
fimfuncao

// Função de Feedback
funcao feedback()
    escreval("Bem-vindo ao Feedback!")
    // Lógica de Feedback
fimfuncao
fimalgoritmo
```

Algoritmo "Teclado Virtual"

```

// Variáveis
texto textoDigitado
caractere teclaPressionada

// Função principal
funcao iniciar()
    textoDigitado <- "" // Inicializar a variável para armazenar o texto digitado

    // Exibir a interface do teclado
    exibirTeclado()

    // Loop para capturar teclas pressionadas
    enquanto (verdadeiro)
        escreva("Pressione uma tecla: ")
        leia(teclaPressionada)

        se teclaPressionada = "Shift" entao
            // Implementar lógica para Shift (caso seja necessário modificar as
teclas)

            shiftPressionado()
        senao se teclaPressionada = "Caps Lock" entao
            // Implementar lógica para Caps Lock
            capsLockPressionado()
        senao se teclaPressionada = "Space" entao
            textoDigitado <- textoDigitado + " "
            escreval("Texto Atual: ", textoDigitado)
        senao se teclaPressionada = "Backspace" entao
            se comprimento(textoDigitado) > 0 entao
                textoDigitado <- textoDigitado[1..comprimento(textoDigitado)-1] //
Remove o último caractere

```

```

        escreval("Texto Atual: ", textoDigitado)
    fimse
    senao se teclaPressionada = "Enter" entao
        escreval("Texto finalizado: ", textoDigitado)
        pare // Finaliza a captura de texto
    senao
        // Adiciona o caractere pressionado ao texto
        textoDigitado <- textoDigitado + teclaPressionada
        escreval("Texto Atual: ", textoDigitado)
    fimse
fimenquanto
fimfuncao

// Função para exibir as teclas do teclado virtual
funcao exibirTeclado()
    escreval("Teclado Virtual")
    escreval("Teclas: 1 2 3 4 5 6 7 8 9 Q W E R T Y U I O P A S D F G H J K L
Z X C V B N M")
    escreval("Teclas especiais: Shift, Caps Lock, Space, Enter, Backspace")
fimfuncao

// Função para tratar o caso de Shift pressionado
funcao shiftPressionado()
    escreval("Shift pressionado. Agora as teclas podem ter maiúsculas ou
símbolos.")
fimfuncao

// Função para tratar o caso de Caps Lock pressionado
funcao capsLockPressionado()
    escreval("Caps Lock pressionado. Alterando entre maiúsculas e
minúsculas.")
fimfuncao
finalgoritmo

```


Algoritmo "ConexaoBancoDeDados"

```
// Variáveis globais
conexao conexão
mensagem texto
stringConexao texto

// Função para conectar ao banco de dados
funcao conectar()
    mensagem <- "" // Inicializa a mensagem de erro
    conexão <- abrirConexao(stringConexao) // Tenta abrir a conexão com o
banco de dados

    se conexão.estaAberta() entao
        escreval("Conexão estabelecida com sucesso.")
    senao
        mensagem <- "Erro ao conectar com BD."
        escreval(mensagem)
    fimse
fimfuncao

// Função para desconectar do banco de dados
funcao desconectar()
    se conexão.estaAberta() entao
        fecharConexao(conexão) // Fecha a conexão se estiver aberta
        escreval("Conexão fechada com sucesso.")
    senao
        mensagem <- "Erro ao desconectar do BD."
        escreval(mensagem)
    fimse
fimfuncao
```

```
// Função para abrir a conexão (simula a criação de uma conexão com string
de conexão)
```

```
funcao abrirConexao(conexaoString: texto) -> conexao
```

```
// Cria uma conexão com o banco de dados usando a string de conexão
fornecida
```

```
escreval("Abrindo conexão com o banco de dados...")
```

```
// Simula a abertura da conexão
```

```
retorna "Conexão aberta" // Retorna uma conexão aberta
```

```
fimfuncao
```

```
// Função para fechar a conexão (simula o fechamento da conexão)
```

```
funcao fecharConexao(conexao: conexao)
```

```
// Simula o fechamento da conexão
```

```
escreval("Fechando a conexão...")
```

```
fimfuncao
```

```
// Função que verifica se a conexão está aberta
```

```
funcao estaAberta() -> lógico
```

```
// Retorna verdadeiro se a conexão estiver aberta
```

```
retorna verdadeiro
```

```
fimfuncao
```

```
// Função principal (inicializa a conexão e demonstra a desconexão)
```

```
funcao principal()
```

```
stringConexao <- "Data Source=DESKTOP-
86S1QF6\\SQL_BANCOS;Initial Catalog=Bd_Pim_Museu;Integrated Security=True;" //
```

```
String de conexão
```

```
conectar() // Chama a função para conectar ao banco de dados
```

```
desconectar() // Chama a função para desconectar do banco de dados
```

```
fimfuncao
```

```
fimalgoritmo
```

Algoritmo "CadastroFeedback"

// Definição das variáveis

texto mensagem

registro

 NomeFe: texto

 Feedback1: texto

 NivelSatisfacao: inteiro

fimregistro

// Função para simular a conexão com o banco de dados

funcao conectarBancoDeDados() -> conexao

 escreval("Conectando ao banco de dados...")

 retorna "Conexão Aberta"

fimfuncao

// Função para simular a desconexão do banco de dados

funcao desconectarBancoDeDados(conexao: texto)

 escreval("Desconectando do banco de dados...")

fimfuncao

// Função para simular a execução de um comando SQL

funcao executarComando(comando: texto)

 escreval("Executando comando SQL: ", comando)

fimfuncao

// Função para cadastrar feedback no banco de dados

funcao cadastrarFeedback(feedback: registro)

 conexao <- conectarBancoDeDados() // Simula a abertura da conexão

 sql <- "INSERT INTO Tbl_Feedbacks (NomeF_Visitante,
Feedback_Visitante, Nivel_Satisfacao) " &

```
"VALUES ('" & feedback.NomeFe & "', '" & feedback.Feedback1 & "', '"
& feedback.NivelSatisfacao & "')
```

```
// Simula a execução do comando SQL para inserir o feedback
```

```
tente
```

```
    executarComando(sql)
```

```
    mensagem <- "Feedback Registrado"
```

```
exceto
```

```
    mensagem <- "Erro no cadastro"
```

```
fimse
```

```
desconectarBancoDeDados(conexao) // Fecha a conexão
```

```
fimfuncao
```

```
// Função principal
```

```
funcao principal()
```

```
    // Criar um exemplo de feedback
```

```
    feedbackExemplo <- registro
```

```
        NomeFe <- "João"
```

```
        Feedback1 <- "Ótima experiência!"
```

```
        NivelSatisfacao <- 5
```

```
    fimregistro
```

```
// Cadastrar o feedback
```

```
cadastrarFeedback(feedbackExemplo)
```

```
// Exibir a mensagem de sucesso ou erro
```

```
    escreval(mensagem)
```

```
fimfuncao
```

```
fimalgoritmo
```

Algoritmo "ObrasCadastro"

```
// Definição das variáveis
```

```
texto mensagem
```

```
registro
```

```
  Id: inteiro
```

```
  TituloObra: texto
```

```
  DescricaoObra: texto
```

```
  ImagemObra: texto
```

```
fimregistro
```

```
// Função para simular a conexão com o banco de dados
```

```
funcao conectarBancoDeDados() -> texto
```

```
  escreval("Conectando ao banco de dados...")
```

```
  retorna "Conexão Aberta"
```

```
fimfuncao
```

```
// Função para simular a desconexão do banco de dados
```

```
funcao desconectarBancoDeDados(conexao: texto)
```

```
  escreval("Desconectando do banco de dados...")
```

```
fimfuncao
```

```
// Função para simular a execução de um comando SQL
```

```
funcao executarComando(comando: texto)
```

```
  escreval("Executando comando SQL: ", comando)
```

```
fimfuncao
```

```
// Função para cadastrar uma obra
```

```
funcao cadastrarObra(obras: registro)
```

```
  conexao <- conectarBancoDeDados() // Simula a conexão com o banco
```

```
de dados
```

```
sql <- "INSERT INTO Tbl_Obras (Titulo_Obra, Descricao_Obra,
Imagem_Obra) " &
```

```
  "VALUES (" & obras.TituloObra & ", " & obras.DescricaoObra & ", "
& obras.ImagemObra & ")"
```

```
tente
```

```
  executarComando(sql)
```

```
  mensagem <- "Obra Registrada"
```

```
exceto
```

```
  mensagem <- "Erro no cadastro da obra"
```

```
fimse
```

```
desconectarBancoDeDados(conexao) // Simula a desconexão do banco
de dados
```

```
fimfuncao
```

```
// Função para pesquisar uma obra pelo ID
```

```
funcao pesquisarObraPorId(obras: registro) -> registro
```

```
  conexao <- conectarBancoDeDados()
```

```
  sql <- "SELECT * FROM Tbl_Obras WHERE ID_Obras = " & obras.Id
```

```
tente
```

```
  executarComando(sql)
```

```
  // Aqui simulamos a leitura de dados do banco, usando valores fixos
como exemplo
```

```
  // Se a obra for encontrada:
```

```
  obras.TituloObra <- "Título da Obra Encontrada"
```

```
  obras.DescricaoObra <- "Descrição da obra encontrada"
```

```
  obras.ImagemObra <- "Caminho/para/imagem.jpg"
```

```
  mensagem <- "Obra encontrada"
```

```
exceto
```

```
  mensagem <- "Erro na pesquisa da obra"
```

```

fimse

desconectarBancoDeDados(conexao)
retorna obras
fimfuncao

// Função para editar os dados de uma obra
funcao editarObra(obras: registro)
  conexao <- conectarBancoDeDados()
  sql <- "UPDATE Tbl_Obras SET Titulo_Obra = " & obras.TituloObra &
    "", Descricao_Obra = " & obras.DescricaoObra &
    "", Imagem_Obra = " & obras.ImagemObra &
    "" WHERE ID_Obras = " & obras.Id

  tente
    executarComando(sql)
    mensagem <- "Obra editada"
  exceto
    mensagem <- "Erro na edição da obra"
  fimse

  desconectarBancoDeDados(conexao)
fimfuncao

// Função para excluir uma obra
funcao excluirObra(obras: registro)
  conexao <- conectarBancoDeDados()
  sql <- "DELETE FROM Tbl_Obras WHERE ID_Obras = " & obras.Id

  tente
    executarComando(sql)
    mensagem <- "Obra excluída"

```

```

exceto
    mensagem <- "Erro na exclusão da obra"
fimse

desconectarBancoDeDados(conexao)
fimfuncao

// Função principal para executar as operações
funcao principal()
    // Criar uma obra exemplo
    obraExemplo <- registro
        Id <- 1
        TituloObra <- "A Noite Estrelada"
        DescricaoObra <- "Pintura famosa de Van Gogh"
        ImagemObra <- "imagem/anoiteestrelada.jpg"
    fimregistro

    // Cadastrar a obra
    cadastrarObra(obraExemplo)
    escreval(mensagem)

    // Pesquisar a obra pelo ID
    obraPesquisada <- pesquisarObraPorId(obraExemplo)
    escreval("Título: ", obraPesquisada.TituloObra)
    escreval("Descrição: ", obraPesquisada.DescricaoObra)
    escreval("Imagem: ", obraPesquisada.ImagemObra)

    // Editar a obra
    obraPesquisada.TituloObra <- "A Noite Estrelada - Edição"
    editarObra(obraPesquisada)
    escreval(mensagem)

```



```
// Excluir a obra
excluirObra(obraPesquisada)
escreval(mensagem)
fimfuncao

finalgoritmo
```

Algoritmo "ControleObrasFeedback"

```

// Definição das variáveis
texto mensagem

// Função para validar dados do feedback
funcao validarFeedback(listaDadosFeedback: lista de texto) -> texto
    se listaDadosFeedback[0] == "" ou listaDadosFeedback[1] == "" ou
listaDadosFeedback[2] == "" ou listaDadosFeedback[3] == "" então
        retorna "Erro: Todos os campos devem ser preenchidos!"
    senão
        retorna ""
    fimse
fimfuncao

// Função para validar dados da obra
funcao validarObra(dadosObra: lista de texto) -> texto
    se dadosObra[0] == "" ou dadosObra[1] == "" ou dadosObra[2] == "" ou
dadosObra[3] == "" então
        retorna "Erro: Todos os campos devem ser preenchidos!"
    senão
        retorna ""
    fimse
fimfuncao

// Função para validar ID
funcao validarId(id: texto) -> texto
    se id == "" ou id != "Número válido" então
        retorna "Erro: ID inválido!"
    senão
        retorna ""
    fimse

```

```
fimfuncao
```

```
// Função para cadastrar feedback
```

```
funcao cadastrarFeedback(listaDadosFeedback: lista de texto)
```

```
  mensagem <- validarFeedback(listaDadosFeedback)
```

```
  se mensagem == "" então
```

```
    // Criar objeto de feedback
```

```
    feedback <- registro
```

```
      Id <- "NovoID"
```

```
      NomeFe <- listaDadosFeedback[1]
```

```
      Feedback1 <- listaDadosFeedback[2]
```

```
      NivelSatisfacao <- listaDadosFeedback[3]
```

```
    fimregistro
```

```
  // Simular cadastro no banco de dados
```

```
  mensagem <- "Feedback registrado com sucesso!"
```

```
  fimse
```

```
fimfuncao
```

```
// Função para cadastrar obra
```

```
funcao cadastrarObra(dadosObra: lista de texto)
```

```
  mensagem <- validarObra(dadosObra)
```

```
  se mensagem == "" então
```

```
    // Criar objeto de obra
```

```
    obra <- registro
```

```
      Id <- "NovoID"
```

```
      TituloObra <- dadosObra[1]
```

```
      DescricaoObra <- dadosObra[2]
```

```
      ImagemObra <- dadosObra[3]
```

```
    fimregistro
```

```
  // Simular cadastro no banco de dados
```

```

        mensagem <- "Obra registrada com sucesso!"
    fimse
fimfuncao

// Função para pesquisar obra por ID
funcao pesquisarObraPorId(listaDadosObras: lista de texto) -> registro
    mensagem <- validarId(listaDadosObras[0])
    se mensagem == "" então
        // Simular consulta de obra pelo ID
        obra <- registro
        Id <- listaDadosObras[0]
        TituloObra <- "Título da Obra Encontrada"
        DescricaoObra <- "Descrição da obra"
        ImagemObra <- "Caminho/para/imagem.jpg"
    fimregistro

    mensagem <- "Obra encontrada com sucesso!"
    retorna obra
senão
    retorna registro Id <- ""
fimse
fimfuncao

// Função para editar obra
funcao editarObra(listaDadosObras: lista de texto)
    mensagem <- validarObra(listaDadosObras)
    se mensagem == "" então
        // Criar objeto de obra
        obra <- registro
        Id <- listaDadosObras[0]
        TituloObra <- listaDadosObras[1]
        DescricaoObra <- listaDadosObras[2]

```

```

        ImagemObra <- listaDadosObras[3]
    fimregistro

    // Simular edição no banco de dados
    mensagem <- "Obra editada com sucesso!"
    fimse
fimfuncao

// Função para excluir obra
funcao excluirObra(listaDadosObras: lista de texto)
    mensagem <- validarId(listaDadosObras[0])
    se mensagem == "" então
        // Simular exclusão no banco de dados
        mensagem <- "Obra excluída com sucesso!"
    fimse
fimfuncao

// Função principal
funcao principal()
    // Dados para teste
    listaDadosFeedback <- lista de texto
        "1", "Carlos", "Muito bom", "5"
    fimlista
    cadastrarFeedback(listaDadosFeedback)
    escreval(mensagem)

    listaDadosObra <- lista de texto
        "1", "Pintura Imaculada", "Descrição da obra", "imagem.jpg"
    fimlista
    cadastrarObra(listaDadosObra)
    escreval(mensagem)

```

```

listaDadosObraParaPesquisa <- lista de texto
    "1"
fimlista
obraPesquisada <- pesquisarObraPorId(listaDadosObraParaPesquisa)
se obraPesquisada.Id != "" então
    escreval("Obra encontrada: ", obraPesquisada.TituloObra)
    escreval("Descrição: ", obraPesquisada.DescricaoObra)
senão
    escreval("Erro ao encontrar obra")
fimse

listaDadosObraParaEditar <- lista de texto
    "1",    "Pintura    Imaculada    Editada",    "Nova    descrição",
"imagem_editada.jpg"
fimlista
editarObra(listaDadosObraParaEditar)
escreval(mensagem)

listaDadosObraParaExcluir <- lista de texto
    "1"
fimlista
excluirObra(listaDadosObraParaExcluir)
escreval(mensagem)
fimfuncao

finalgoritmo

```

Algoritmo "FeedbackClasse"

```
// Definir a estrutura FeedbackClasse
registro FeedbackClasse
    inteiro id
    texto nomeFe
    texto feedback
    texto nivelSatisfacao
fimregistro

// Função para definir os dados de um feedback
funcao definirFeedback(id: inteiro, nomeFe: texto, feedback: texto,
nivelSatisfacao: texto) -> FeedbackClasse
    feedbackClasse <- FeedbackClasse

    // Atribuir valores aos atributos
    feedbackClasse.id <- id
    feedbackClasse.nomeFe <- nomeFe
    feedbackClasse.feedback <- feedback
    feedbackClasse.nivelSatisfacao <- nivelSatisfacao

    retorna feedbackClasse
fimfuncao

// Função para exibir os dados de um feedback
funcao exibirFeedback(feedbackClasse: FeedbackClasse)
    escreval("ID: ", feedbackClasse.id)
    escreval("Nome do Visitante: ", feedbackClasse.nomeFe)
    escreval("Feedback: ", feedbackClasse.feedback)
    escreval("Nível de Satisfação: ", feedbackClasse.nivelSatisfacao)
fimfuncao
```

```
// Função principal
funcao principal()
  // Criar um feedback com dados fictícios
  meuFeedback <- definirFeedback(1, "Carlos Silva", "Gostei muito da visita",
"5")

  // Exibir os dados do feedback
  exibirFeedback(meuFeedback)
fimfuncao

finalgoritmo
```


Algoritmo "FeedbackClasse"

```
// Definir a estrutura FeedbackClasse
registro FeedbackClasse
    inteiro id
    texto nomeFe
    texto feedback
    texto nivelSatisfacao
fimregistro

// Função para definir os dados de um feedback
funcao definirFeedback(id: inteiro, nomeFe: texto, feedback: texto,
nivelSatisfacao: texto) -> FeedbackClasse
    feedbackClasse <- FeedbackClasse

    // Atribuir valores aos atributos
    feedbackClasse.id <- id
    feedbackClasse.nomeFe <- nomeFe
    feedbackClasse.feedback <- feedback
    feedbackClasse.nivelSatisfacao <- nivelSatisfacao

    retorna feedbackClasse
fimfuncao

// Função para exibir os dados de um feedback
funcao exibirFeedback(feedbackClasse: FeedbackClasse)
    escreval("ID: ", feedbackClasse.id)
    escreval("Nome do Visitante: ", feedbackClasse.nomeFe)
    escreval("Feedback: ", feedbackClasse.feedback)
    escreval("Nível de Satisfação: ", feedbackClasse.nivelSatisfacao)
fimfuncao
```

```
// Função principal
funcao principal()
  // Criar um feedback com dados fictícios
  meuFeedback <- definirFeedback(1, "Carlos Silva", "Gostei muito da visita",
"5")

  // Exibir os dados do feedback
  exibirFeedback(meuFeedback)
fimfuncao

finalgoritmo
```

Algoritmo "ObrasClasse"

```
// Definir a estrutura ObrasClasse
registro ObrasClasse
    inteiro id
    texto tituloObra
    texto descricaoObra
    texto imagemObra
fimregistro

// Função para definir os dados de uma obra
funcao definirObra(id: inteiro, tituloObra: texto, descricaoObra: texto,
imagemObra: texto) -> ObrasClasse
    obra <- ObrasClasse

    // Atribuir valores aos atributos
    obra.id <- id
    obra.tituloObra <- tituloObra
    obra.descricaoObra <- descricaoObra
    obra.imagemObra <- imagemObra

    retorna obra
fimfuncao

// Função para exibir os dados de uma obra
funcao exibirObra(obra: ObrasClasse)
    escreval("ID: ", obra.id)
    escreval("Título da Obra: ", obra.tituloObra)
    escreval("Descrição da Obra: ", obra.descricaoObra)
    escreval("Imagem da Obra: ", obra.imagemObra)
fimfuncao
```

```
// Função principal
funcao principal()
  // Criar uma obra com dados fictícios
  minhaObra <- definirObra(1, "A Noite Estrelada", "Uma das obras mais
famosas de Van Gogh.", "imagem1.jpg")

  // Exibir os dados da obra
  exibirObra(minhaObra)
fimfuncao

finalgoritmo
```

Algoritmo "Validação"

// Função para validar um ID

funcao validarId(identificacao: texto) -> inteiro

inteiro id

id <- -1

tente

id <- inteiro(identificacao) // Tenta converter para inteiro

exceto

escreval("Erro de id") // Mensagem de erro

fimtente

retorna id

fimfuncao

// Função para validar os dados do visitante

funcao validarVisitante(listaDadosPessoa: lista<texto>) -> texto

texto mensagem

mensagem <- ""

// Validando o ID

inteiro idVisitante <- validarId(listaDadosPessoa[0])

se idVisitante = -1 então

mensagem <- "Erro de id\n"

fimse

// Validando o nome

se comprimento(listaDadosPessoa[1]) < 3 então

mensagem <- mensagem + "Nome deve ter mais que 3 caracteres\n"

fimse

se comprimento(listaDadosPessoa[1]) > 50 então

mensagem <- mensagem + "Nome com mais que 50 caracteres\n"

fimse

```

// Validando a idade
se inteiro(listaDadosPessoa[2]) > 120 então
    mensagem <- mensagem + "Idade elevada, digite novamente\n"
fimse

retorna mensagem
fimfuncao

// Função para validar os dados de feedback
funcao validarFeedback(listaDadosFeedback: lista<texto>) -> texto
    texto mensagem
    mensagem <- ""

    // Validando o ID
    inteiro idFeedback <- validarId(listaDadosFeedback[0])
    se idFeedback = -1 então
        mensagem <- "Erro de id\n"
    fimse

    // Validando o nome do visitante
    se comprimento(listaDadosFeedback[1]) < 3 então
        mensagem <- mensagem + "Nome deve ter mais que 3 caracteres\n"
    fimse
    se comprimento(listaDadosFeedback[1]) > 50 então
        mensagem <- mensagem + "Nome com mais que 50 caracteres\n"
    fimse

    retorna mensagem
fimfuncao

// Função para validar os dados da obra

```

```

funcao validarObra(dadosObra: lista<texto>) -> texto
  texto mensagem
  mensagem <- ""

  // Validando o ID
  inteiro idObra <- validarId(dadosObra[0])
  se idObra = -1 então
    mensagem <- "Erro de id\n"
  fimse

  // Validando o título da obra
  se comprimento(dadosObra[1]) < 3 então
    mensagem <- mensagem + "Nome da obra deve ter mais que 3
caracteres\n"
  fimse
  se comprimento(dadosObra[1]) > 50 então
    mensagem <- mensagem + "Nome da obra com mais que 50 caracteres\n"
  fimse

  retorna mensagem
fimfuncao

// Função principal
funcao principal()
  // Exemplo de dados para validar

  // Dados do visitante (ID, Nome, Idade)
  lista<texto> dadosVisitante <- ["1", "João", "25"]
  texto mensagemVisitante <- validarVisitante(dadosVisitante)
  escreval("Validação do Visitante: ", mensagemVisitante)

  // Dados de Feedback (ID, Nome, Feedback)

```

```
lista<texto> dadosFeedback <- ["2", "Maria", "Ótimo"]
texto mensagemFeedback <- validarFeedback(dadosFeedback)
escreval("Validação do Feedback: ", mensagemFeedback)

// Dados da obra (ID, Título, Descrição)
lista<texto> dadosObra <- ["3", "Monalisa", "Descrição da obra"]
texto mensagemObra <- validarObra(dadosObra)
escreval("Validação da Obra: ", mensagemObra)

fimfuncao

finalgoritmo
```