

UNIVERSIDADE PAULISTA

LUCAS MATHEUS CAMPESTRINI CERIONE - RA: G856082

MARIA DA GLORIA AYUMI MURAKI NARUSE - RA: N076HC1

NICOLAS KENZO CARVALHO ONISHI - RA: F356CA6

PAULO HENRIQUE MENDES PAIVA - RA: F3576H1

PROJETO DE UM TOTEM DE MUSEU MULTITEMÁTICO

SOROCABA

2024

LUCAS MATHEUS CAMPESTRINI CERIONE - RA: G856082
MARIA DA GLORIA AYUMI MURAKI NARUSE - RA: N076HC1
NICOLAS KENZO CARVALHO ONISHI - RA: F356CA6
PAULO HENRIQUE MENDES PAIVA - RA: F3576H1

PROJETO DE UM TOTEM DE MUSEU MULTITEMÁTICO.

Trabalho de Conclusão de Semestre para
obtenção de pontos na média em (CST em
Análise e Desenvolvimento de Sistemas)
apresentado à Universidade Paulista –
UNIP.

Orientador: Prof. Especialista. Reverdan A.
Sparinger.

SOROCABA

2024

LUCAS MATHEUS CAMPESTRINI CERIONE - RA: G856082
MARIA DA GLORIA AYUMI MURAKI NARUSE - RA: N076HC1
NICOLAS KENZO CARVALHO ONISHI - RA: F356CA6
PAULO HENRIQUE MENDES PAIVA - RA: F3576H1

PROJETO DE UM TOTEM DE MUSEU MULTITEMÁTICO.

Trabalho de Conclusão de semestre para
obtenção de pontos na média em (CST em
Análise e Desenvolvimento de Sistemas)
apresentado à Universidade Paulista –
UNIP.

Aprovado(a) em: ____/____/____

BANCA EXAMINADORA

Me. Waldir Silva
Universidade Paulista - UNIP

Prof. Especialista Alex Machado Ferreira Sampaio
Universidade Paulista - UNIP

Prof. Especialista Reverdan A. Sparinger
Universidade Paulista – UNIP

RESUMO

Neste projeto é apresentado o conceito de um museu de temática múltipla centrado na primeira viagem do ser humano à Lua. O objetivo do museu é oferecer uma experiência educativa e interativa aos seus visitantes, permitindo que eles explorem as obras em exposição e obtenham informações históricas por meio de um totem interativo. Além disso, os visitantes serão convidados a participar de uma pesquisa sobre a exposição, cujos resultados serão destinados a melhorar a experiência e tomar decisões estratégicas sobre a programação do tema em destaque. O sistema de coleta de questionários foi elaborado com a finalidade de obter os dados de maneira dinâmica, assegurando a conformidade com a LGPD. O desenvolvimento do sistema será executado na plataforma .NET com C# e Windows Forms, alinhada aos princípios da gestão estratégica de recursos humanos, economia de mercado e design de interface do usuário, bem como ao paradigma de orientação a objetos. Este projeto propõe uma abordagem inovadora e integrada visando a criação de um museu com tema específico, com o intuito de oferecer uma experiência enriquecedora e inesquecível aos visitantes.

Palavras-chave: Museu multitemático; Primeira viagem à Lua; Totem interativo; Conformidade com a LGPD; Design de interface do usuário.

ABSTRACT

This project presents the concept of a multiple-themed museum centered on the first human trip to the Moon. The objective of the museum is to offer an educational and interactive experience to its visitors, allowing them to explore the works on display and obtain historical information through an interactive totem. In addition, visitors will be invited to participate in a survey about the exhibition, the results of which will be designed to improve the experience and make strategic decisions about the programming of the highlighted theme. The questionnaire collection system was designed with the purpose of obtaining data dynamically, ensuring compliance with the LGPD. The development of the system will be carried out on the .NET platform with C# and Windows Forms, aligned with the principles of strategic human resources management, market economics and user interface design, as well as the object-oriented paradigm. This project proposes an innovative and integrated approach aimed at creating a museum with a specific theme, with the aim of offering an enriching and unforgettable experience to visitors.

Key-words: Multithematic museum; First trip to the Moon; Interactive totem; Compliance with LGPD; User interface design.

LISTA DE ILUSTRAÇÕES

Figura 1 - Planta Baixa Museu	53
Figura 2 – Diagrama de rede.....	54
Figura 3 – Diagrama de sequência totem de entrada	55
Figura 4 – Diagrama de casos de uso.....	56
Figura 5 – Fases da engenharia de usabilidade.....	73
Figura 6 – Fluxograma do totem de entrada	76
Figura 7 – Fluxograma do totem das obras.....	77
Figura 8 – Fluxograma do totem da pesquisa de satisfação	78
Figura 9 – Tela início totem de entrada.....	79
Figura 10 – Tela mapa	80
Figura 11 – Tela login.....	80
Figura 12 – Tela cadastro	81
Figura 13 – Escolhas de ingresso	82
Figura 14 – Tela de compras.....	83
Figura 15 – Tela de login com token	84
Figura 16 – Tela das obras	84
Figura 17 – Tela das obras	85
Figura 18 – Resultado questionário.....	86
Figura 19 – Administrativo.....	87
Figura 20 – Relatório.....	88
Figura 21 – Pesquisa de Satisfação.....	89
Figura 22 – Feedback	90
Figura 23 – Pastas no PostgreSQL.....	91
Figura 24 – Lógico dentro do PostgreSQL	92
Figura 25 – MER	94
Figura 26 – DER.....	95
Figura 27 – Diagrama de classe de vendas	96
Figura 28 – Diagrama de classe de vendas continuação	97
Figura 29 – Diagrama de classe do questionário	98
Figura 30 – Diagrama de classe do questionário continuação	99
Figura 31 – Diagrama de classe da avaliação.....	100
Figura 32 – Diagrama de classe da avaliação continuação	101
Figura 33 – Diagrama de classe do ADM.....	102

Figura 34 – Diagrama de classe do ADM continuação	103
Figura 35 – Teste realizado com a ferramenta xUnit.net.....	105

LISTA DE TABELAS

Tabela 1 – Explicação das funções apresentadas e suas peculiaridades.....	57
Tabela 2 – Explicação dos requisitos funcionais nas aplicações	59
Tabela 3 – Explicação dos requisitos não funcionais nas aplicações	60
Tabela 4 – Modelos e preços utilizados no cabeamento.....	61
Tabela 5 – Modelo e preço utilizado no switch.....	61
Tabela 6 – Modelos e preços utilizados na internet	62
Tabela 7 – Modelo e preço utilizado na segurança do local.....	62
Tabela 8 – Modelo e preço utilizado no computador	62
Tabela 9 – Modelos e preços utilizados no totem	63
Tabela 9 – Código em C# do teclado	106
Tabela 10 – Código em C# da pesquisa de satisfação da PerguntasController	108
Tabela 11 – Código em C# da pesquisa de satisfação da classe MediaAvaliacaoModel	110
Tabela 12 – Questionários das obras construtor.....	112
Tabela 13 - método VerificarResposta	114
Tabela 14 - método PerguntaAtual.....	114
Tabela 15 - métodos AlternativaAtual, ProximaPergunta e AtualizarPergunta	115
Tabela 16 - controle das respostas corretas	116
Tabela 17 – Média de acertos	117
Tabela 18 – armazenamentos de dados	118
Tabela 19 – conexão com o banco de dados.....	119
Tabela 20 – gerenciamento de questionários.....	120
Tabela 21 – Venda de ingressos.....	122
Tabela 22 – gerenciamento de informações de pessoas	124
Tabela 23 - setando ações dos botões	125
Tabela 24 – Conexão com o banco de dados	127
Tabela 25 – Método RegistrarVoto()	127
Tabela 26 – Método RegistrarVoto()	128
Tabela 27 – Método RegistrarVoto()	129
Tabela 28 – Conexão com banco de dados vendas	131

SUMÁRIO

1.	INTRODUÇÃO	8
1.1.	Objetivo geral	8
1.2.	Objetivos específicos	9
2.	PESQUISA BIBLIOGRAFICA	10
2.1.	Museu Interativo: História, Exposições e Infraestrutura Inovadora	10
2.1.1.	Infraestrutura museu	10
2.1.2.	Tema do museu: à primeira viagem do homem à lua	11
2.1.3.	Obras a serem expostas	12
2.2.	Totem Interativo	13
2.2.1.	Hardware do totem	13
2.2.2.	Requisitos mínimos do hardware do totem	13
2.3.	Infraestrutura de rede	14
2.3.1.	Redes de computadores	15
2.4.	Programação e seus princípios	16
2.4.1.	Linguagem da programação	16
2.4.2.	A linguagem C#.....	16
2.4.2.1.	<i>Aplicações Desktop com C#.....</i>	<i>17</i>
2.4.2.2.	<i>Aplicações Web com C#.....</i>	<i>17</i>
2.4.2.3.	<i>Jogos com C#:</i>	<i>17</i>
2.4.2.4.	<i>Ferramentas de Desenvolvimento:</i>	<i>18</i>
2.4.2.5.	<i>Comunidade e Recursos:.....</i>	<i>18</i>
2.4.3.	A biblioteca Windows Forms	18
2.5.	Análise orientada a objetos.....	19
2.6.	Banco de dados.....	20
2.7.	Qualidade de software	21
2.7.1.	Normas ISO	22
2.8.	Princípios de Interfaces com usuário.....	23
3.	INTERAÇÃO HUMANO COMPUTADOR	25
3.1.	Aspectos Fundamentais da Interação Humano-Computador	25
3.2.	Eficácia.....	26
3.3.	Eficiência	26
3.4.	Segurança.....	26
3.5.	Capacidade de aprendizagem.....	27

3.6.	Capacidade de memorização	27
3.7.	Satisfação:	27
3.8.	Ergonomia	28
3.9.	Acessibilidade	28
4.	INTERFACE DE USUÁRIO	29
4.1.	Designer de Interação	29
4.2.	Teoria da ação de Norman, 1986.....	30
4.3.	Percepção	30
4.4.	Interpretação	30
4.5.	Formulação da Intenção	31
4.6.	Especificação da Sequência de Ações	31
4.7.	Execução	31
4.8.	Avaliação	31
4.9.	Ajustes	31
4.10.	Gestão estratégica de RH.....	31
5.	ECONOMIA E MERCADO.....	33
6.	LGPD	35
6.1.	Como a LGPD se aplica em eventos	36
6.2.	Como garantir o consentimento dos usos dos dados em eventos	36
6.3.	Criptografia de dados online.....	38
7.	DESENVOLVIMENTO	40
7.1.	Engenharia de software.....	40
7.1.1.	Processo de desenvolvimento de software	40
7.1.2.	RN, RF e RNF.....	41
7.2.	Levantamento de requisitos.....	41
7.3.	Requisito do negócio.....	42
7.4.	Requisitos funcionais	42
7.5.	Requisitos não funcionais.....	44
7.6.	Como implementar a tecnologia no museu	45
8.	GARANTIA DA QUALIDADE DO SOFTWARE NA EMPRESA	47
8.1.	Padrões e Modelos de Qualidade no Desenvolvimento de Software	47
8.2.	Ciclo de Vida do Software: Garantindo Qualidade e Eficiência	49
8.3.	Gestão estratégica de RH.....	51
8.4.	Infraestrutura de rede	52

8.5.	Estrutura do museu	52
8.5.1.	Cabeamento.....	52
8.5.2.	Diagrama de sequência	54
8.5.3.	Diagrama de casos de uso.....	55
8.5.4.	Custo dos equipamentos.....	61
8.5.4.1.	<i>Cabeamento</i>	<i>61</i>
8.5.4.2.	<i>Switch</i>	<i>61</i>
8.5.4.3.	<i>Internet.....</i>	<i>62</i>
8.5.4.4.	<i>Câmeras</i>	<i>62</i>
8.5.4.5.	<i>Computador para totens</i>	<i>62</i>
8.5.4.6.	<i>Estrutura dos Totens.....</i>	<i>63</i>
8.5.5.	Topologia de rede	63
8.5.5.1.	<i>Topologia em estrela</i>	<i>63</i>
8.5.5.2.	<i>Implementação da rede</i>	<i>64</i>
9.	PRINCIPIOS DE INTERFACE UTILIZADOS.....	65
10.	PROCESSO DE DESIGN EM IHC.....	67
10.1.	Aspectos Fundamentais da Interação Humano-Computador	67
10.2.	Eficácia.....	67
10.3.	Eficiência	68
10.4.	Segurança.....	68
10.5.	Capacidade de aprendizagem.....	68
10.6.	Capacidade de memorização	69
10.7.	Satisfação	69
10.8.	Ergonomia	69
10.9.	Acessibilidade	70
10.10.	Acessibilidade no Totem do Museu Multitemático	70
10.11.	Eficácia e Eficiência na Interação.....	70
10.12.	Segurança e Confiança	71
10.13.	Personalização e Adaptabilidade	71
10.14.	Teoria da Ação na Prática	71
11.	MODELOS DE CICLO DE VIDA.....	72
11.1.	Interação Humano Computador (IHC)	73
11.2.	ISO 9241 (projeto de interface com o usuário)	74
11.3.	Orientações sobre Usabilidade.....	74

12.	REQUISITOS PARA APRESENTAÇÃO VISUAL	75
12.1.	Estrutura Visual.....	75
12.2.	Fluxograma.....	75
13.	FLUXO DE FUNCIONAMENTO SISTEMA	76
14.	CRONOLOGIA DAS TELAS	79
14.1.	Tela totem de entrada	79
14.2.	Tela do totem das obras	83
14.3.	POWER BI	88
14.4.	Pesquisa de satisfação.....	88
14.5.	Banco de dados Postgree	90
15.	MODELAGEM DE DADOS COM MER E DER	93
15.1.	Modelo Entidade-Relacionamento (MER)	93
15.2.	Diagrama Entidade-Relacionamento (DER)	93
15.3.	Diagrama de classe.....	95
15.4.	Programação	104
15.5.	Testes executados nos sistemas	104
15.6.	Testes executados com a xUnit.net.....	104
15.7.	Teclado Virtual	105
15.8.	Pesquisa de Satisfação	108
15.9.	Questionários das obras	112
15.9.1.1.	<i>Método VerificarResposta</i>	<i>114</i>
15.9.1.2.	<i>O método PerguntaAtual</i>	<i>114</i>
15.9.1.3.	<i>Os métodos AlternativaAtual, ProximaPergunta e AtualizarPergunta ..</i>	<i>115</i>
15.10.	Classe Services.....	118
15.10.1.1.	<i>VendasService.....</i>	<i>118</i>
15.10.1.2.	<i>AvaliacaoService</i>	<i>119</i>
15.10.1.3.	<i>QuestionarioService</i>	<i>120</i>
15.11.	Venda de ingressos	122
15.12.	ADM.....	123
15.13.	Conexão com o banco de dados avaliação	126
15.14.	Conexão com o banco de dados questionário.....	128
15.15.	Conexão com o banco de dados vendas.....	130
16.	CONSIDERAÇÕES FINAIS.....	133
	REFERÊNCIAS BIBLIOGRÁFICAS	143

APÊNDICE A - MANUAL DE INSTALAÇÃO E DESINSTALAÇÃO	146
APÊNDICE B – TAP (TERMO DE ABERTURA DE PROJETO).....	149
APÊNDICE C – IMAGENS DO CRONOGRAMA DO PROJETO	155
APÊNDICE D – CÓDIGO FONTE DO PROGRAMA DO QUESTIONÁRIO	158
APÊNDICE E – CÓDIGO FONTE DO PROGRAMA DAS VENDAS	204
APÊNDICE F – CÓDIGO FONTE DO PROGRAMA DO ADM	222
APÊNDICE G – CÓDIGO FONTE DO PROGRAMA DA PESQUISA DE SATISFAÇÃO	240
APÊNDICE H – CÓDIGO FONTE DO PROGRAMA DO TECLADO	251

1. INTRODUÇÃO

Desenvolver um museu com diversas temáticas é sempre um desafio instigante, principalmente quando se aborda um assunto tão marcante e motivador quanto a chegada do homem à Lua. O objetivo deste projeto não se limita em apenas instruir e divertir os visitantes, mas também em oferecer uma vivência envolvente e participativa, permitindo que eles se aprofundem e se informem sobre esse momento histórico tão significativo para a humanidade.

O museu tem como objetivo proporcionar uma experiência singular, onde os visitantes poderão apreciar as obras em exposição e ter acesso a informações históricas sobre cada uma através de um totem interativo. Adicionalmente, os visitantes serão convidados a colaborar em uma pesquisa sobre a exposição, compartilhando suas opiniões que serão destinados melhorar a experiência e realizar escolhas estratégicas com finalidade de futuras exposições.

A estrutura para coleta de questionários foi planejada com objetivo de obter informações de forma dinâmica, garantindo a conformidade com a LGPD (Lei Geral de Proteção de Dados). A tecnologia aplicada no desenvolvimento, *.NET* com *C#* e *Windows Forms*, a fim de assegurar a perfeita conexão com um *display* sensível ao toque, proporcionando uma experiência de uso intuitiva e envolvente aos usuários. Este trabalho será conduzido seguindo os princípios de Projetos de Interface com o Usuário, com documentação alinhada à Engenharia de Software.

Além disso, adotará o paradigma de Orientação a Objetos, conforme aprendido em Programação Orientada a Objetos e Análise de Sistemas Orientado a Objetos, assegurando um desenvolvimento sólido e produtivo do sistema. A gestão estratégica de recursos humanos e a análise da economia de mercado serão integradas visando garantir uma implementação competente do projeto.

1.1. Objetivo geral

O objetivo geral deste trabalho é projetar e desenvolver um totem interativo para um museu multitemático. O totem será uma ferramenta essencial para auxiliar os visitantes, permitindo-lhes consultar obras expostas e obter informações históricas sobre a exposição.

Além disso, o totem incluirá um questionário de múltipla escolha para coletar feedback dos visitantes, visando a melhoria contínua da experiência oferecida pelo museu. O desenvolvimento do totem será realizado em plataforma *.NET* com *C#* e *Windows Forms*, seguindo os princípios de Projetos de interface com o usuário e o paradigma de Orientação a Objetos. O objetivo é proporcionar uma experiência educativa, interativa e imersiva aos visitantes, tornando a visita ao museu mais informativa, envolvente e memorável.

1.2. Objetivos específicos

Os propósitos claros deste projeto estão voltados para a criação e elaboração de um totem interativo destinado a um museu que aborda a primeira missão tripulada à Lua. Inicialmente, procura-se reconhecer e entender as demandas dos visitantes do museu, transformando essas informações em exigências e soluções práticas para o totem.

Adicionalmente, será apresentado o debate e a análise das tecnologias selecionadas para a criação do totem, embasando as escolhas feitas e avaliando sua relação ao propósito do projeto. A colaboração em equipe terá um papel fundamental, incentivando a interação entre diversas áreas de conhecimento na realização do projeto do totem, dessa forma a prática do trabalho em equipe em projetos que envolvem diferentes disciplinas.

Uma outra meta é melhorar a comunicação e capacidade de negociação ao se relacionar com diferentes partes interessadas, tais como gestores do museu e possíveis utilizadores do totem, certificando-se de que as propostas apresentadas atendam às expectativas e necessidades identificadas.

Além disso, o projeto tem como objetivo incentivar o aprendizado constante ao utilizar e colocar em prática novos conceitos e métodos atuais de interação com o usuário, programação orientada a objetos e criação de sistemas computacionais ao conceber e construir o totem. Por último, busca-se estimular a originalidade e inovação para encontrar soluções eficazes e criativas para os desafios enfrentados no desenvolvimento do totem interativo, levando em conta as melhores práticas e tendências do setor.

2. PESQUISA BIBLIOGRAFICA

A pesquisa bibliográfica é o levantamento ou revisão de obras publicadas sobre a teoria que irá direcionar o trabalho científico.

A pesquisa bibliográfica é elaborada a partir de material já publicado, constituído principalmente de: livros, revistas, publicações em periódicos e artigos científicos, jornais, boletins, monografias, dissertações, teses, material cartográfico, internet, com o objetivo de colocar o pesquisador em contato direto com todo material já escrito sobre o assunto da pesquisa. (PRODANOV; FREITAS, 2013, p. 54).

2.1. Museu Interativo: História, Exposições e Infraestrutura Inovadora

ICOM Brasil (*international council of museums*) (2022), fala que é uma Instituição permanente, sem fins lucrativos, ao serviço da sociedade, que estuda, coleciona, preserva, interpreta e expõe o património material e imaterial. Os museus são abertos ao público, acessíveis e inclusivos, promovendo a diversidade e a sustentabilidade, enquanto os museus temáticos trabalham exclusivamente sobre um tema e utilizam qualquer suporte de coleção para atingir esse objetivo.

2.1.1. Infraestrutura museu

O Instituto Brasileiro de Museus (IBRAM) recomenda como etapas para a criação de um museu:

- a) Planejamento de um plano para estabelecer o museu;
- b) Criação da personalidade jurídica da empresa, de acordo com a legislação vigente;
- c) Contratação efetiva de um grupo multidisciplinar, contando até mesmo com o profissional especializado em museologia, para conduzir os procedimentos técnicos necessários no museu;
- d) A validação de um protocolo (normas internas de um museu);
- e) Desenvolvimento do Planejamento Museológico, de acordo com o Artigo 46 da lei nº 11.904, documento fundamental que estabelecerá a missão, os propósitos, o público-alvo e as atividades.

Os museus se definem por diferentes inserções administrativas. Podem ser privados, assim como públicos federais, estaduais ou municipais. Segundo o IBRAM, os museus públicos municipais constituem a categoria de natureza administrativa mais frequente no Brasil, com 41,1% do total segundo. (IBRAM, 2011, p. 27).

Vários museus sentem-se desconectados da pesquisa acadêmica em museologia e não se veem capazes de planejar e gerenciar autonomamente suas atividades. Para melhorar essa situação, publicações como esta obra em foco fazem parte de uma iniciativa para disseminar conhecimento e criar textos direcionados a um público leigo.

Um outro recurso, embora breve, serve como guia para seguir as orientações da área: o livro "Orientações para a implantação de museus locais". De acordo com a obra, são necessários oito elementos fundamentais para garantir o bom desempenho de um museu.

- decreto, lei, portaria, ata ou outro documento legal de sua criação;
- um documento que defina seu estatuto jurídico e sua natureza administrativa;
- um regimento interno que registre objetivos, política institucional, papel e composição da diretoria, assim como formas de manutenção;
- um organograma;
- o Plano Museológico;
- o local de instalação do museu;
- um plano de ocupação dos espaços (por exemplo, salas de exposição, reserva técnica, salas administrativas, espaço de ação educativa e cultural, espaços de serviços, espaços de circulação, sala de segurança e outros que se fizerem necessários);
- E, quando for o caso, identificação de percursos e roteiros no território de atuação do museu (CHAGAS; NASCIMENTO, 2009, p. 14).

2.1.2. Tema do museu: à primeira viagem do homem à lua

No dia 20 de julho de 1969, a histórica missão Apollo 11 da agência espacial NASA entrou para os livros com a pioneira viagem do ser humano até a Lua. Sob o comando do astronauta Neil Armstrong, a espaçonave realizou um pouso bem-sucedido na superfície lunar, possibilitando que Armstrong e Buzz Aldrin se tornassem os primeiros terráqueos a pisarem na Lua. Enquanto isso, Michael Collins permaneceu em órbita ao redor da Lua, conduzindo o módulo de comando da missão.

A investigação lunar foi fruto de extensos períodos de pesquisa, progresso e treinamento rigoroso, contando com a contribuição de diversos especialistas das áreas científica, de engenharia e técnica. A missão Apollo 11 marcou a quinta missão tripulada do programa Apollo e foi a primeira a atingir um pouso controlado na Lua com astronautas a bordo.

A aterrissagem na Lua foi exibida ao vivo para um público mundial, o qual acompanhou com entusiasmo e curiosidade cada momento significativo. Quando Neil Armstrong realizou o icônico passeio, proferindo as palavras "É um pequeno passo para o homem, mas um salto gigantesco para a humanidade", ele simbolizou não apenas a realização na Lua, mas também o triunfo da habilidade e da persistência humanas.

A espaçonave Apollo 11 pousou em segurança na Terra em 24 de julho de 1969, não apenas trazendo os três corajosos astronautas, mas também uma perspectiva renovada sobre nossa localização no universo e as inúmeras possibilidades da exploração espacial. A conquista da Lua pelo homem inaugurou caminhos para futuras expedições no espaço e inspirou inúmeros pesquisadores, profissionais de engenharia e visionários a persistirem na superação dos limites do conhecimento e da criatividade.

2.1.3. Obras a serem expostas

No museu, os objetos em destaque que retratam a primeira viagem tripulada à Lua imergem os visitantes em uma narrativa informativa e envolvente desse momento histórico marcante. Uma réplica fiel da nave Apollo 11 realça a imponência da nave espacial e sua relevância crucial na missão. O módulo lunar, que pode ser uma cópia ou um item autêntico, proporciona aos visitantes uma visão minuciosa da rotina e das atividades dos astronautas na superfície lunar.

As vestimentas especiais em exposição demonstram a tecnologia avançada necessária para garantir a segurança dos astronautas em um ambiente desafiador como o espaço. Instrumentos e equipamentos utilizados na jornada, como computadores e câmeras, ressaltam a sofisticação e a inovação presentes na missão Apollo 11.

Imagens captadas durante a exploração espacial eternizam momentos históricos, como a chegada à Lua e a famosa frase de Neil Armstrong. Documentos antigos, como registros de diálogos da agência espacial dos Estados Unidos, permitem uma visão detalhada dos bastidores dessa jornada épica. Além disso, a contextualização da missão no contexto político, social e cultural da época através de placas informativas e materiais educativos destaca a importância e o impacto significativo desse evento na história.

2.2. Totem Interativo

Um painel digital em um espaço cultural é um dispositivo eletrônico que oferece aos visitantes uma experiência interativa e educativa. Geralmente, o painel é formado por uma tela sensível ao toque, alto-falantes e, em certos casos, câmeras ou leitores de códigos de barras.

As colunas possuem a habilidade de exibir informações sobre exposições, obras de arte, eventos passados ou qualquer outro dado relevante para o museu. Pode incluir vídeos, imagens, textos informativos e até mesmo atividades educativas para interagir de forma mais profunda com os visitantes e com o conteúdo em destaque.

2.2.1. Hardware do totem

Uma das principais vantagens da mesa *touch screen* e do totem interativo está na sua praticidade na hora de ser instalada.

Apesar dos avanços tecnológicos significativos relacionados à eficácia e segurança da tecnologia touchscreen, o desempenho do processamento, softwares e aplicativos *touch* também evoluíram consideravelmente.

2.2.2. Requisitos mínimos do hardware do totem

Telas sensíveis ao toque, telas de toque capacitivo, telas acústicas por superfície (SAW) e telas ópticas são os modelos mais frequentes de tecnologia de tela tátil, embora existam também opções mais peculiares.

Atualmente, os tipos mais populares de monitores sensíveis ao toque são os capacitivos e os resistivos. A grande diferença entre eles está no modo como reagem ao toque do usuário: enquanto o capacitivo responde de forma suave, o resistivo exige uma pressão específica para funcionar.

Esses monitores modernos vêm com diversas opções de entradas de vídeo padrão, como *DP (HDCP)*, *DVI*, *HDMI*, *VGA* e *USB*. São telas planas que utilizam tecnologia *LCD*, *LED* ou uma combinação de ambas. Para rodar aplicativos e vídeos, é necessário um dispositivo de computação comum de uso doméstico. Os requisitos mínimos de hardware recomendados são os seguintes:

- a) **Sistema operacional necessário:** *Windows* 8 ou versão mais recente;
- b) **CPU:** *Intel Core* i3, i5 ou i7;
- c) **Placa de vídeo:** embutida da *Intel* com software atualizado;
- d) **Tela sensível ao toque:** há uma grande variedade de opções disponíveis no mercado com diferentes finalidades (como mesas interativas, paredes interativas, totens interativos, entre outros). uma recomendação seria Totem em MDF (fibra de média densidade) ou plástico (para suportar a tela Touch);
- e) Memória *RAM* 8GB.

2.3. Infraestrutura de rede

A estrutura de comunicação é composta por diversos elementos que possibilitam a interligação de dispositivos entre si e com a rede externa.

Dessa forma, trata-se da organização dos aparelhos e a determinação dos caminhos pelos quais os dados circulam, levando em consideração tanto o hardware quanto o software. Compreende a topologia, os componentes ideais, as restrições características da infraestrutura local e outros aspectos relevantes.

Resumidamente, abrange uma variedade de elementos, como roteadores, switches, servidores, data centers, cabos, links, computadores e pessoas. O intuito é assegurar que todos esses elementos interajam de forma eficiente entre si, garantindo assim a estabilidade e consistência das operações da empresa.

O cabo é o meio mais comum de conexão entre computadores. Principalmente em redes de larga escala, em que a velocidade de comunicação é muito importante, o cabo ainda é o meio mais adequado para transmissão de dados. Existem basicamente três modalidades de cabos para redes de computadores: cabo de par trançado, cabo coaxial e cabo de fibra ótica. (SIQUEIRA, LUCIANO, 2010, p. 10)

Na maioria das vezes, ao abordarmos este tema, estamos nos referindo a um plano organizado que assegura a máxima eficiência. Portanto, é necessário mencionar a infraestrutura de cabos, um sistema avançado de comunicações e novidades para aprimorar a rotina diária.

2.3.1. Redes de computadores

Segundo Emanuel Diniz (2020) em seu artigo observa que uma rede de computadores é constituída por múltiplas conexões estabelecidas entre diversos dispositivos, com o intuito de estimular a troca de recursos e dados diversos, essa interconexão tem como objetivo desempenhar várias funções nos aparelhos, tais como acesso a aplicativos e informações, capacidade de enviar e receber documentos, realização de reuniões e outras funcionalidades.

Dessa forma, a fim de garantir o bom funcionamento das redes de computadores, é fundamental a adoção de protocolos que viabilizem a comunicação desejada pelos usuários.

Os serviços e as facilidades das redes são requisitados pelos usuários por meio de aplicações de software - cada aplicação é um programa aplicativo em um computador que se comunica através da rede com outro programa aplicativo que roda em outro computador. Serviços de rede abrangem um conjunto variado de atividades, tais como e-mail, transmissão e recepção de arquivos, navegação na rede, chamadas telefônicas, acessos a bancos de dados distribuídos e videoconferência. (COMER, DOUGLAS E., 2016, p. 5)

As redes de computadores atuam como uma malha abrangente que possui a habilidade de conectar diferentes sistemas, dispositivos eletrônicos e equipamentos de hardware de computador. Esses elementos desempenham um papel fundamental na troca e armazenamento de informações.

Dessa forma, é fundamental seguir diretrizes e regulamentos para garantir o funcionamento eficiente das redes de computadores, uma vez que é preciso direcionar

as informações, ou seja, estabelecer a forma como serão transmitidas, enviadas e recebidas.

Essas estruturas também têm o importante papel de checar e autenticar as informações dos sistemas. Adicionalmente, elas facilitam a troca de informações entre diversos meios, dispositivos e indivíduos ao redor do planeta.

2.4. Programação e seus princípios

De forma geral, a programação é um processo de escrita, testes e manutenção de programas de computadores. Esses programas, por sua vez, são compostos por conjuntos de instruções determinados pelo programador que descrevem tarefas a serem realizadas pela máquina e atendem diversas finalidades.

Segundo Anita Lopes e Guto Garcia (2002, p. 15) “[...]Lógica de programação e a técnica de encadear pensamentos para atingir determinado objetivo O aprendizado desta técnica é necessário, para quem deseja trabalhar com desenvolvimento de sistemas e programas[...]”.

2.4.1. Linguagem da programação

A Linguagem de Programação consiste em um sistema formal e escrito que define um conjunto de instruções e normas utilizadas para criar programas de computador (*software*). Esses programas podem ser projetados para funcionar em computadores, dispositivos móveis ou em qualquer outro aparelho capaz de executá-los.

Há diversas linguagens de programação disponíveis, cada uma com sua finalidade específica. Algumas têm o objetivo óbvio de desenvolver softwares, enquanto outras são utilizadas de forma menos convencional, como para controlar veículos automotivos ou aparelhos domésticos, como torradeiras.

2.4.2. A linguagem C#

O C# (lê-se "C Sharp") é uma linguagem de programação moderna, orientada a objetos e com forte tipagem, desenvolvida pela *Microsoft*. É amplamente utilizada na criação de diversos tipos de softwares, incluindo aplicativos para *Windows*, *web*,

jogos e dispositivos móveis, principalmente utilizando a plataforma *.NET*. Essa linguagem foi projetada para ser simples, robusta, confiável e eficiente, apresentando recursos avançados como suporte a tipos genéricos, expressões lambda, entre outros.

O *C#* é uma linguagem de programação orientada a objetos com um sistema de tipagem forte, o que significa que os programas são estruturados em objetos que contêm tanto dados quanto métodos, e todos os tipos de dados são determinados durante a compilação. Além disso, o *C#* possui um sistema integrado de gerenciamento de memória, que utiliza um coletor de lixo para controlar a alocação e deslocação de memória, evitando vazamentos de memória.

Outra vantagem do *C#* é a capacidade de interoperar com *C++* e *Component Object Model* (COM), permitindo a conexão com sistemas legados e bibliotecas já existentes. O *.NET Framework* inclui uma ampla biblioteca padrão que oferece suporte para uma variedade de tarefas comuns, como manipulação de arquivos, acesso a bancos de dados e comunicação em rede.

O *C#* oferece várias opções para o desenvolvimento de diferentes tipos de aplicativos:

2.4.2.1. *Aplicações Desktop com C#*

- a) **Windows Forms:** Permite criar interfaces gráficas de usuário (GUI) de forma rápida e fácil;
- b) **Windows Presentation Foundation (WPF):** Oferece recursos avançados de GUI, como animações e gráficos 3D.

2.4.2.2. *Aplicações Web com C#*

- a) **ASP.NET:** Um *framework* para o desenvolvimento de aplicativos *web* dinâmicos e escaláveis.

2.4.2.3. *Jogos com C#:*

- a) **Unity:** Uma *engine* popular para desenvolvimento de jogos que suporta *C#* para criação de jogos 2D e 3D.

2.4.2.4. Ferramentas de Desenvolvimento:

- a) **Visual Studio:** A principal IDE usada para desenvolver aplicativos C#, oferecendo um conjunto abrangente de ferramentas de desenvolvimento e design de interface de usuário.

2.4.2.5. Comunidade e Recursos:

- a) O C# possui uma grande comunidade de desenvolvedores ativos e uma vasta quantidade de recursos disponíveis online, incluindo documentação oficial, fóruns de discussão e tutoriais;
- b) O C# é uma escolha popular entre os desenvolvedores devido à sua combinação de simplicidade, eficiência e suporte da *Microsoft*.

2.4.3. A biblioteca Windows Forms

O *WinForms*, também chamado de *Windows Forms*, é uma biblioteca de GUI desenvolvida pela *Microsoft* para o *.NET Framework*. Com essa ferramenta, os desenvolvedores podem criar de maneira simples interfaces de usuário para aplicativos *Windows* através da inserção de elementos visuais em formulários. Isso facilita a construção rápida e fácil de aplicativos *desktop* para o sistema operacional *Windows*.

As principais características incluem uma grande variedade de controles previamente definidos, como botões, caixas de texto, listas e menus, que podem ser facilmente adicionados aos formulários. Além disso, suporta eventos e delegados, permitindo que os controles reajam às interações do usuário, como cliques do mouse e teclas pressionadas.

As funcionalidades de *layout* automático presentes no *Windows Forms* possibilitam que os componentes se organizem de forma automática ao redimensionar e reposicionar de acordo com as mudanças de tamanho do formulário. Essa característica facilita a criação de interfaces que se adaptam a diferentes tamanhos de tela e resoluções.

Uma característica importante do *Windows Forms* é a sua habilidade em suportar gráficos e multimídia, possibilitando a criação de interfaces visualmente

atrativas e interativas. Além disso, ele oferece recursos de acessibilidade para garantir que os aplicativos sejam acessíveis a todos os usuários, incluindo aqueles com deficiências visuais ou motoras. Essa variedade de funcionalidades torna o *Windows Forms* uma escolha popular para o desenvolvimento de aplicativos desktop *Windows*.

A elaboração de programas com *Windows Forms* envolve a criação de formulários com elementos visuais e funcionalidades do programa, permitindo personalização com diversas cores, tipos de fontes e imagens para criar interfaces de usuário únicas e atrativas. Ele está integrado com a linguagem de programação *C#* para criar aplicações sofisticadas e interativas para ambiente de trabalho *Windows*, respondendo às interações dos usuários. O *Windows Forms* é conhecido pela sua acessibilidade e facilidade de aprendizado rápido, além de oferecer suporte a várias versões do *Windows*, garantindo compatibilidade com sistemas mais antigos. Essas características tornam o *Windows Forms* uma escolha popular tanto para programadores iniciantes quanto experientes que desejam desenvolver programas *Windows* de alta qualidade.

2.5. Análise orientada a objetos

A abordagem de programação orientada a objetos é uma técnica que organiza o sistema de software com base em entidades conhecidas como "objetos". Esses objetos possuem propriedades que armazenam informações e métodos que realizam operações. Por meio da troca de mensagens, eles se comunicam entre si, imitando interações presentes no contexto real.

No contexto do museu, o sistema do totem interativo utiliza a abordagem de orientação a objetos para apresentar os diferentes elementos do sistema de forma mais intuitiva e eficiente. Algumas das vantagens desse modelo incluem:

- a) **Encapsulamento:** Consiste em os objetos preservarem suas funcionalidades e dados, mantendo em sigilo os aspectos internos e disponibilizando somente os métodos acessíveis publicamente. Esse procedimento ajuda na estruturação do sistema e evita consequências indesejadas;
- b) **Reuso de código:** A programação baseada em objetos facilita a reutilização de código por meio da herança e composição. É possível que

classes herdem comportamentos de outras classes e que objetos sejam compostos por outros objetos, possibilitando o desenvolvimento de software de forma modular e com capacidade de ser expandido;

- c) **Polimorfismo:** O polimorfismo é uma característica que permite que objetos de classes diferentes sejam tratados de maneira uniforme. Isso significa que um método pode ter comportamentos diferentes dependendo do tipo de objeto que o chama, o que aumenta a flexibilidade e a capacidade de crescimento do sistema;
- d) **Estrutura e divisão:** A programação baseada em objetos permite a organização do código em classes e objetos que representam entidades e funcionalidades específicas do sistema. Com isso, o código se torna mais legível, simples de manter e de reutilizar.

De forma sucinta, a programação baseada em objetos é uma abordagem para criar software que incentiva a modularidade, a reutilização de código e a capacidade de se adaptar, proporcionando facilidade na criação, manutenção e expansão do sistema de exibição interativa do museu.

2.6. Banco de dados

Um banco de dados é uma estrutura organizada para armazenar e gerenciar conjuntos de dados. Ele é criado para armazenar grandes quantidades de informações de forma organizada, tornando mais fácil o acesso, controle e atualização desses dados. Os bancos de dados são comumente utilizados em programas de computador para armazenar informações como histórico de usuários, transações financeiras, registros médicos, dentre outros.

Existem diversas classificações de bancos de dados, cada uma criada para atender a diferentes necessidades e situações de uso.

Existem várias classes de sistemas de armazenamento de dados, cada uma com suas próprias características e usos específicos. Os sistemas de armazenamento de informações relacionais, como *MySQL*, *SQL Server* e *Oracle Database*, organizam dados em tabelas conectadas, seguindo o conceito relacional. Por outro lado, os sistemas de armazenamento *NoSQL*, como *MongoDB*, *Cassandra* e *Redis*, foram

criados para lidar com grandes volumes de dados não convencionais ou parcialmente estruturados, como documentos e pares chave-valor.

As memórias cacheadas, como *Redis* e *Memcached*, são responsáveis por armazenar dados na memória RAM do computador, proporcionando assim um acesso rápido às informações. Por outro lado, os bancos de dados orientados a objetos, tais como *db4o* e *ObjectDB*, têm a capacidade de armazenar diretamente objetos de programação, sem a necessidade de mapeá-los para uma estrutura de tabela relacional. Por fim, os bancos de dados de grafos, como *Neo4j* e *ArangoDB*, foram desenvolvidos para armazenar e consultar dados na forma de grafos, sendo especialmente úteis para lidar com relações complexas entre entidades.

2.7. Qualidade de software

No processo de criação do software destinado ao totem interativo do museu, são empregados diversos conceitos de qualidade de software visando garantir a eficácia, segurança e usabilidade do sistema. Um dos pilares fundamentais é a transparência do código-fonte, que deve ser claro e de fácil compreensão, agilizando a manutenção e a colaboração entre os desenvolvedores.

Além disso, a praticidade na manutenção é um princípio chave, garantindo que o software seja desenvolvido de forma modular e bem estruturada, permitindo que correções e melhorias sejam feitas de forma eficiente. A confiabilidade também é um aspecto crucial, exigindo que o software seja robusto e capaz de lidar com falhas, funcionando corretamente mesmo em circunstâncias adversas.

A eficiência é um princípio crucial, sendo primordial otimizar o programa para garantir um desempenho eficiente, evitando desperdício de recursos como memória e processamento.

A adaptabilidade também é um aspecto relevante a ser considerado, garantindo que o software possa ser facilmente adaptado a diferentes ambientes e sistemas, possibilitando sua utilização em diversas situações.

A simplicidade na utilização é fundamental, exigindo que a interface do sistema seja clara e de fácil acesso, proporcionando uma experiência positiva aos usuários. A possibilidade de realizar testes é outro ponto importante, sendo necessário que o software seja projetado para possibilitar a realização de testes de maneira simples e eficaz, garantindo a qualidade e a confiabilidade dele.

No fim das contas, a segurança é um princípio essencial, exigindo que o software seja desenvolvido de forma a proteger os dados e garantir a privacidade das pessoas, em conformidade com as regras da LGPD. Ao seguir essas orientações de qualidade do software, é possível garantir que o software destinado ao totem digital do museu seja eficaz, protegido e de fácil utilização, oferecendo uma experiência agradável aos visitantes.

2.7.1. Normas ISO

Na situação da programação do software para o totem de interatividade do museu, é possível aplicar diversas normas ISO importantes para assegurar a eficiência e adequação do sistema. Algumas das normas ISO que podem ser levadas em consideração são:

- a) **Norma ISO/IEC 25010 - SQuaRE (Software product Quality Requirements and Evaluation):** Este modelo define um padrão de excelência de software que aborda questões como performance, confiabilidade, usabilidade, eficiência, manutenção, adaptabilidade, segurança e compatibilidade. Ele é utilizado para estabelecer e validar os requisitos de qualidade do software desenvolvido para o terminal de interação;
- b) **Norma ISO/IEC 9126:** O conteúdo discute sobre a Engenharia de Software e a Qualidade do Produto, trazendo diretrizes para a análise da qualidade de softwares, levando em conta características como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Essas orientações são importantes para garantir que o software utilizado no totem atenda aos requisitos de qualidade estabelecidos.
- c) **Norma ISO 9241:** Explora a ergonomia na interação entre humanos e sistemas, oferecendo diretrizes para a usabilidade de sistemas interativos. Ela aborda aspectos como facilidade de utilização, eficácia, segurança, eficiência, conforto e acessibilidade. Essas recomendações podem ser aplicadas para garantir que a interface do usuário do totem seja amigável e fácil de usar;
- d) **Norma ISO/IEC 27001:** Encarregado de estabelecer os parâmetros para

um sistema de segurança da informação, garantindo a devida proteção dos dados dos clientes coletados pelo totem, em conformidade com as normas estabelecidas pela LGPD;

- e) **Norma ISO/IEC 12119 - Pacotes de software - Exigências de qualidade e teste:** Neste guia são apresentadas diretrizes para definir os padrões de excelência de um programa de computador e métodos de avaliação. É viável utilizar esse padrão para garantir que o software do terminal passe por testes abrangentes e esteja de acordo com os critérios de qualidade estabelecidos.

Ao seguir as orientações da ISO, é possível garantir que o software desenvolvido para o totem interativo do museu esteja em conformidade com os padrões de qualidade necessários, oferecendo uma experiência positiva para os usuários e cumprindo as normas e regulamentos aplicáveis.

2.8.Princípios de Interfaces com usuário

As bases para a elaboração de interface com o usuário, conhecidas como princípios de usabilidade, são orientações que ajudam os designers a criarem interfaces de usuário eficazes e amigáveis.

Ao desenvolver o totem interativo em um museu, é crucial considerar alguns princípios essenciais.

- a) **Retorno:** Ofereça retorno imediato e transparente sobre as ações do utilizador. Isso pode envolver animações, mensagens de confirmação e mudanças na interface para destacar que uma ação foi concluída com êxito;
- b) **Uniformidade:** Preserve a uniformidade na aparência e no comportamento dos elementos da interface. Isso auxilia na familiarização e na interação do sistema pelo usuário;
- c) **Simplicidade:** Priorize a simplicidade na interface, garantindo facilidade de compreensão. Evite sobrecarregar com muitas informações e escolhas, destacando o que realmente importa para o usuário;
- d) **Destaque:** Faça com que as principais opções e funcionalidades sejam

facilmente visíveis e acessíveis. Utilize elementos visuais como cores, ícones e texto em destaque para direcionar o usuário para as informações essenciais;

- e) **Adaptabilidade:** Deixe o usuário ajustar a aparência da interface de acordo com suas escolhas. Isso engloba escolhas relacionadas ao tamanho da fonte, tonalidade das cores e organização dos elementos na tela;
- f) **Resistência a falhas:** Desenvolva a interface visando reduzir os efeitos de equívocos cometidos pelo usuário. Ofereça mensagens de erro objetivas e indique medidas corretivas sempre que viável;
- g) **Efetividade:** Desenvolva a estrutura de modo a possibilitar que o usuário execute suas atividades de maneira ágil e efetiva. Isso envolve reduzir a quantidade de cliques e ações requeridas para finalizar uma tarefa.

Certifique-se de que a plataforma seja acessível a todos os usuários, sem distinção de deficiências visuais, auditivas ou motoras. Isso pode incluir a utilização de cores contrastantes adequadas, legendas em vídeos e suporte para diversos tipos de dispositivos de entrada.

Ao aplicar essas diretrizes de design de interface com o usuário, é possível criar uma experiência satisfatória e eficiente no totem interativo do museu, garantindo que os visitantes consigam interagir de forma natural e simplificada com o dispositivo.

3. INTERAÇÃO HUMANO COMPUTADOR

A Interação Humano-Computador (IHC), também conhecida como *HCI* em inglês, é um campo de estudo multidisciplinar que se dedica à compreensão e aprimoramento da interação entre seres humanos e sistemas computacionais. Esse domínio de pesquisa abrange uma variedade de aspectos, incluindo usabilidade, experiência do usuário e a influência da interação tecnológica nas percepções e experiências individuais dos usuários.

De acordo com Finlay, "Com a popularização dos computadores, um número cada vez maior de pesquisadores se interessou por estudar a interação entre homem e máquina, com interesse nos aspectos físicos, psicológicos e teóricos desse processo. (Finlay, Dix, Abowd, & Beale, 2003, p. 3)

A IHC busca, portanto, investigar e desenvolver métodos, técnicas e teorias que promovam uma interação mais espontânea, acessível e centrada no usuário.

3.1. Aspectos Fundamentais da Interação Humano-Computador

a usabilidade é um atributo de qualidade relacionado à facilidade do uso de algo. Mas especificamente, refere-se à rapidez com que os usuários podem aprender a usar alguma coisa, a eficiência deles ao usá-la, o quanto lembram daquilo, seu grau de indução aos erros e o quanto gostam de utilizá-la (Nielsen, 2007).

Ao aplicar os conceitos de IHC no desenvolvimento do software em conjunto com o totem do museu multitemático, busca-se proporcionar uma vivência agradável, com uma interface intuitiva e fácil interação. Utilizando técnicas de design interativo, buscamos garantir que os usuários possam navegar pelo sistema de forma fluida e eficiente, facilitando a compra de ingressos, avaliação das obras, participação em pesquisas de satisfação e questionários.

Ao projetar a interface do software, consideramos os requisitos ergonômicos a fim de garantir que seja confortável e acessível aos usuários, levando em conta aspectos como o tamanho e a resolução da tela do totem, a disposição dos elementos visuais e a legibilidade do texto.

Dessa forma, a aplicação prática dos princípios de IHC no desenvolvimento do software destinado exclusivamente ao totem do museu visa proporcionar uma experiência de uso agradável e satisfatória aos visitantes, promovendo assim a usabilidade e eficácia do sistema.

3.2. Eficácia

Para garantir uma experiência satisfatória aos usuários, é essencial que o sistema seja capaz de fornecer orientações claras e compreensíveis, permitindo que o usuário aprenda facilmente a navegar e realizar suas tarefas de forma autônoma.

Além disso, é importante que o sistema demonstre uma consistência visual e comportamental para que o usuário seja capaz de reconhecer e aprender padrões com certa facilidade. O sistema deve ser projetado de modo a permitir a realização de diversas atividades, não se limitando apenas ao trabalho.

3.3. Eficiência

A forma como o sistema aborda a gestão de falhas em relação ao usuário é de extrema importância, uma vez que permite que o usuário reverta ações executadas erroneamente. Mesmo que o sistema possua mecanismos para impedir a execução de comandos específicos que possam resultar em ações graves, falhas humanas e sistêmicas ainda podem ocorrer.

Nesse contexto, é essencial que as instruções para corrigir erros sejam comunicadas de maneira transparente e interativa, possibilitando que o usuário compreenda e solucione o problema, restaurando o sistema ao seu estado anterior.

O sistema deve ser projetado para permitir que o usuário controle suas ações de forma clara, compreendendo seus comandos e interações. Isso aumenta a satisfação do usuário e ajuda a alcançar seus objetivos de maneira mais eficiente.

3.4. Segurança

A segurança, em sua essência, refere-se ao nível de proteção de um sistema, ou seja, como o sistema deve prevenir erros para que o usuário não realize ações indesejáveis acidentalmente. Uma maneira de evitar erros é, por exemplo, fornecer

ao usuário um comando que permita a recuperação total de informações em caso de acionamento acidental de um botão que as elimine.

3.5. Capacidade de aprendizagem

Dominar a utilização do sistema por conta própria, sem enfrentar dificuldades, é fundamental para compreender a flexibilidade e a familiaridade que o usuário desenvolverá com o sistema. Por exemplo, é essencial que o sistema seja de fácil utilização, permitindo que o usuário execute suas tarefas de forma tranquila, independentemente da complexidade delas.

Além disso, é importante considerar que usuários de diferentes faixas etárias interagirão com o sistema, e, portanto, o sistema deve oferecer ferramentas que possibilitem a customização de comandos e a criação de teclas de atalho. Isso contribuirá para que o usuário se sinta menos sobrecarregado durante a interação com o sistema.

3.6. Capacidade de memorização

A memorabilidade na utilização do sistema deve ser planejada de modo a garantir que o usuário, mesmo não utilizando o sistema com frequência, possa contar com o suporte necessário para facilitar sua utilização. Além do pré-preenchimento automático de campos de formulário, outras estratégias podem ser adotadas para promover a memorização e familiarização com o sistema.

Além disso, a consistência na organização e rotulagem dos elementos da interface contribui para uma experiência mais intuitiva, permitindo que o usuário associe facilmente ações e funcionalidades aos seus respectivos controles.

3.7. Satisfação:

A satisfação do usuário é um aspecto crucial da usabilidade de um software. Quando o sistema é intuitivo, eficiente e confiável, os usuários se sentem satisfeitos com sua experiência.

3.8. Ergonomia

Pode-se dizer que a ergonomia está na origem da usabilidade, pois ela visa proporcionar eficácia e eficiência, além do bem-estar e saúde do usuário, por meio da adaptação do trabalho ao homem. Isto significa que seu objetivo é garantir que sistemas e dispositivos estejam adaptados à maneira como o usuário pensa, comporta-se e trabalha e, assim, proporcionem usabilidade.

De maneira geral, a ergonomia representa um conjunto de diretrizes e práticas que visam garantir a utilização adequada do hardware e do software. Ela é fundamental para promover a saúde e o bem-estar dos usuários, uma vez que o uso prolongado ou inadequado desses recursos pode resultar em sérios problemas de saúde, como lesões musculoesqueléticas e fadiga ocular.

Além disso, a ergonomia abrange não apenas aspectos físicos, como a disposição dos dispositivos e a postura do usuário, mas também questões relacionadas à interface e à usabilidade do software, contribuindo para uma experiência de uso mais confortável e produtiva. É importante considerar também a adaptação das tecnologias às características individuais dos usuários, como idade, habilidades e necessidades específicas, para garantir uma interação mais eficiente e segura

3.9. Acessibilidade

Durante a interação, o usuário utiliza suas habilidades motoras para operar os dispositivos de entrada, além de seus sentidos visão, audição e tato para identificar as respostas do sistema emitidas pelos dispositivos de saída. Ele também emprega sua capacidade cognitiva, interpretativa e de raciocínio para compreender as respostas do sistema e planejar os próximos passos da interação. Qualquer obstáculo imposto pela interface pode prejudicar a capacidade do usuário de aproveitar o suporte computacional oferecido pelo sistema.

A acessibilidade é fundamental para garantir que pessoas com diferentes capacidades seja de movimento, percepção, cognição ou aprendizado tenham igual oportunidade de perceber, compreender e utilizar o sistema para desfrutar de seu suporte computacional.

4. INTERFACE DE USUÁRIO

Bruno Silva e Simone Barbosa (2010) observam em seu artigo que a interação é um processo de comunicação do usuário com o sistema por meio de interfaces. O engajamento entre a interface e o usuário ocorre quando ambos participam juntos de alguma atividade específica. Esse procedimento vai além da simples realização da tarefa, pois envolve uma comunicação que se assemelha a uma interação verbal ou à manipulação de componentes.

Dentro do cenário da comunicação entre usuário e sistema, a interface resulta da junção de softwares e hardwares fundamentais para facilitar a interação entre o usuário e a aplicação. Ela inclui componentes visuais, como botões, menus e ícones, além de dispositivos de entrada, como teclado, mouse ou tela sensível ao toque. A interface funciona como uma conexão que permite ao usuário interagir com o sistema e obter retorno sobre suas ações.

4.1. Designer de Interação

A fim de assegurar a verdadeira interatividade de um produto, é fundamental adotar uma estratégia de design que leve em conta não só as características do usuário, mas também o ambiente no qual será empregado. Isso requer compreender as atividades particulares que o usuário irá executar ao interagir com o sistema.

Na criação de uma interface interativa, é fundamental levar em conta o perfil do usuário principal e o contexto em que o produto será empregado. Dessa forma, torna-se possível desenvolver um layout que atenda de forma personalizada as exigências e gostos do usuário logo no início da interação com o sistema.

É essencial compreender que o *design* de interação não se limita apenas à beleza visual, abrangendo diversos elementos que influenciam a experiência do usuário. Para além do visual, como organização, cores e fontes, o *designer* de interação precisa levar em conta a praticidade, acessibilidade e efetividade do produto. Isso implica em desenvolver interfaces intuitivas e práticas, que facilitem a interação do usuário e proporcionem uma experiência positiva.

Além disso, é crucial ressaltar que o desenvolvimento de interação precisa ser flexível e customizável, possibilitando aos usuários a capacidade de personalizar a

interface de acordo com suas preferências pessoais. Essa abordagem eleva a experiência do usuário e reforça seu vínculo com o produto.

4.2. Teoria da ação de Norman, 1986.

Entendemos que a engenharia cognitiva, aliada à psicologia cognitiva, se dedica ao estudo da cognição, ou seja, como os indivíduos adquirem conhecimento, e aplicam teorias com o objetivo de compreender a capacidade e as limitações da mente do usuário. Essa abordagem interdisciplinar busca explorar os processos mentais envolvidos na interação humano-computador, fornecendo percepções sobre como projetar sistemas adaptáveis e amigáveis ao usuário. Ademais, compreender melhor a forma como os usuários percebem, processam e respondem às informações apresentadas pelas interfaces visando o aprimoramento da sua experiência de usuário com a tecnologia.

Para utilizar a teoria da ação na prática, são desempenhadas sete etapas que compreendem desde a identificação do problema até a avaliação dos resultados. Essas etapas incluem análise situacional, definição de objetivos, planejamento, implementação, monitoramento, avaliação e ajustes. Cada uma dessas etapas desempenha um papel crucial no processo, influenciando diretamente o sucesso da aplicação da teoria da ação em diversos contextos práticos.

4.3. Percepção

A primeira etapa é a percepção, na qual o usuário percebe os elementos relevantes na interface, como botões, menus ou ícones, por meio dos sentidos visuais, auditivos ou táteis.

4.4. Interpretação

Após perceber os elementos, o usuário interpreta seu significado e função na interface, entendendo como podem ser utilizados para alcançar seus objetivos ou realizar determinadas tarefas.

4.5. Formulação da Intenção

Com base na interpretação dos elementos da interface, o usuário formula sua intenção, ou seja, decide qual ação deseja realizar e como pretende fazê-lo por meio da interação com os elementos disponíveis.

4.6. Especificação da Sequência de Ações

Nesta etapa, o usuário especifica a sequência de ações necessárias para alcançar sua intenção, determinando a ordem e os passos a serem seguidos na interação com a interface.

4.7. Execução

Após planejar a sequência de ações, o usuário executa as operações necessárias na interface, utilizando dispositivos de entrada, como teclado, mouse ou tela sensível ao toque, para realizar suas ações planejadas.

4.8. Avaliação

Após a execução das ações, o usuário avalia os resultados obtidos, verificando se suas expectativas foram atendidas e se alcançou seus objetivos por meio da interação com a interface.

4.9. Ajustes

Por fim, o usuário pode realizar ajustes com base na avaliação realizada, modificando suas ações ou revisando sua intenção, caso necessário, para otimizar sua interação com a interface e alcançar melhores resultados.

4.10. Gestão estratégica de RH

uma organização é uma combinação de esforços individuais que tem por finalidade realizar propósitos coletivos. Por meio de uma organização, torna-se possível perseguir e alcançar objetivos que

seriam inatingíveis por uma pessoa. Uma grande empresa ou uma pequena oficina, um laboratório ou o corpo de bombeiros, um hospital ou uma escola são todos exemplos de organizações. (Maximiano, 2006).

A Gestão Estratégica de Recursos Humanos (GERH) desempenha um papel fundamental no direcionamento das atividades de uma organização, estando ligada ao seu planejamento estratégico. À medida que as empresas crescem e as demandas da gestão de pessoal se tornam mais complexas, a GERH se estabelece como um componente indispensável, evoluindo ao longo da história da administração para atender às exigências tanto das empresas quanto dos colaboradores.

Uma das principais políticas da GERH é garantir que a empresa conte com recursos humanos adequados e motivados para suas operações presentes e futuras. A partir desse princípio, diversas outras políticas e estratégias são formuladas para otimizar o aproveitamento dos recursos humanos existentes, promover seu desenvolvimento contínuo e assegurar a sustentabilidade e expansão da organização.

No âmbito da remuneração, a GERH vai além do salário direto, incluindo benefícios e participação nos lucros, levando em consideração tanto as capacidades financeiras da empresa quanto a dinâmica do mercado de trabalho.

Mudança Organizacional é qualquer transformação de natureza estrutural, estratégica, cultural, tecnológica, humana ou de qualquer outro componente capaz de gerar impacto em partes ou no conjunto da organização. (Wood Jr, 1995, p.190).

O monitoramento compreensível dos funcionários, apoiado por tecnologias da informação, é essencial para garantir um ambiente de trabalho ético e produtivo. Estabelecer uma cultura organizacional fundamentada em valores éticos contribui para a criação de um ambiente saudável e propício ao desenvolvimento individual e coletivo.

Definir claramente a visão, missão e valores da organização é crucial para estabelecer uma base sólida para suas operações e promover relações harmoniosas com todos os *stakeholders* envolvidos.

5. ECONOMIA E MERCADO

A ciência que estuda como os recursos escassos das sociedades é alocada tendo por base as decisões individuais de consumidores, trabalhadores, firmas etc. É a ciência que analisa as escolhas individuais e suas interações (Guimarães; Gonçalves, 2010, p. 184).

Sendo então, o estudo que envolve a gestão de recursos trabalhando com a escassez destes artifícios financeiros, legais e de informação. Portanto, a gestão bem-sucedida desses recursos traz um impacto positivo na economia e na sociedade, determinado assim, o que é produzido, consumido e como são oferecidos. A importância da economia reside na ação fundamental em organizar e manter em funcionamento a sociedade.

O presente trabalho atuou no ramo turístico na economia, onde a interação entre museus e turismo conseguem enriquecer a vida e cultura de uma região, assim que os museus são integrados ao turismo, resultando em uma valorização do patrimônio, contribuindo para o desenvolvimento da identidade do local, de acordo com (Rotman & Castells, 2007, p. 63 - 64), [...] "a revitalização da identidade cultural, para a preservação dos bens culturais e das tradições, operando como uma atividade que pode gerar mecanismos de sustentabilidades próprios para a cidade" [...].

Vale destacar que os projetos de economia criativa desenvolvidos nos museus podem atrair um segmento do turismo ainda crescente: os turistas criativos. Esse tipo de turismo, conforme observado por Richards e Raymond (2000), oferece aos visitantes a oportunidade de explorar seu potencial por meio de experiências de aprendizagem e participação ativa em cursos típicos dos destinos turísticos. O turismo criativo, centrado na experiência e na participação, tem sido cada vez mais valorizado, pois atende à necessidade do setor turístico de se reinventar, às necessidades dos destinos de fazer algo diferente em um mercado saturado e aos desejos dos turistas por experiências significativas.

Com isso, o museu cria oportunidades para o desenvolvimento econômico local atraindo visitantes e gerando empregos. Além disso, ao conseguir promover experiências significativas para os turistas, o museu consegue contribuir para os possíveis investimentos futuros na economia local.

A implementação dos totens interativos envolveu uma série de custos, os quais foram devidamente planejados e gerenciados, sendo a equipe responsável pelo desenvolvimento do software dos totens a mesma envolvida na abertura do museu, permitiu-se a integração mais eficiente no desenvolvimento tecnológico e o conteúdo cultural. Posto isso, foram inclusos a compra dos materiais como os totens e os componentes que fazem o software seguir em funcionamento.

6. LGPD

O principal propósito da Lei Geral de Proteção de Dados (13.709/2018) é garantir os direitos essenciais de liberdade, privacidade e autonomia da pessoa física. Além disso, busca estabelecer um ambiente de segurança jurídica, com a uniformização de normas e procedimentos para assegurar a proteção dos dados pessoais de todos os cidadãos presentes no território nacional, seguindo padrões internacionais estabelecidos.

A legislação determina o que são considerados dados pessoais e destaca que certos tipos de dados requerem cuidados especiais, como os dados sensíveis e os dados de crianças e adolescentes. Além disso, ressalta que todos os dados que são processados, tanto em formato físico quanto digital, devem obedecer a regulação.

A LGPD enfatiza que não importa a localização da sede de uma empresa ou do centro de dados, se houver manipulação de informações de indivíduos, sejam brasileiros ou estrangeiros, dentro do território nacional, a lei deve ser seguida. A legislação também permite o compartilhamento de dados pessoais com organizações internacionais e outros países, contanto que os requisitos nela estabelecidos sejam cumpridos.

Na LGPD, informações sensíveis são definidas como dados pessoais sensíveis relacionadas à origem racial ou étnica, convicção religiosa, opinião política, filiação a sindicato ou organização de caráter religioso, filosófico ou político, dados referentes à saúde ou à vida sexual, dados genéticos ou biométricos, quando vinculados a uma pessoa natural.

No caso específico do projeto, o nome, o e-mail e a idade são considerados dados pessoais, mas não são considerados dados sensíveis pela LGPD. O CEP, por si só, não é considerado um dado sensível, mas pode ser considerado sensível se utilizado de forma a revelar informações sobre a localização geográfica precisa de uma pessoa.

Portanto, para garantir a conformidade com a LGPD, é importante proteger esses dados pessoais e utilizá-los apenas para os fins específicos informados aos visitantes, sem coletar informações sensíveis sem consentimento explícito e específico.

6.1. Como a LGPD se aplica em eventos

Agora que você entendeu o conceito da LGPD, é importante considerar os dados obtidos durante o seu evento. Por exemplo, durante o processo de inscrição, quando são solicitados nome, e-mail e demais informações, a LGPD já está em vigor!

Portanto, é nesse instante em que é fundamental comunicar de maneira clara ao participante a finalidade do uso de seus dados. Por exemplo, se pretende utilizar os dados além da inscrição para enviar e-mails, publicidade segmentada ou qualquer outra finalidade, solicitar o consentimento do participante é necessário.

6.2. Como garantir o consentimento dos usos dos dados em eventos

A concordância é um dos aspectos mais importantes da LGPD. Por isso, é essencial que tudo seja compreendido, permitido e detalhado na Política de Privacidade, atendendo aos critérios estabelecidos.

A concordância deve ser expressa de maneira clara e voluntária; A utilização das informações deve ser totalmente transparente, sem margem para questionamentos.

Se houver um contrato por escrito, é importante destacar a cláusula que trata dos dados em relação às demais. Caso a sua solicitação envolva informações sensíveis - como raça, orientação sexual, saúde, filiação política, entre outros - é necessário ressaltar qual a finalidade do tratamento desses dados. Ou seja, explicar de que forma eles serão utilizados e com qual objetivo, no caso deste projeto não estamos usando dados sensíveis.

Apenas os pais ou tutores legais têm o direito de autorizar a utilização dos dados de menores de idade. Caso haja alguma alteração no processo de tratamento dessas informações, será imprescindível obter uma nova autorização por parte do titular.

De fato, é possível assegurar isso durante o evento seguindo algumas etapas. Primeiramente, certifique-se do consentimento através do opt-in e opt-out. Em termos simples, o opt-in representa a permissão que o participante concede para o uso de seus dados, enquanto o opt-out oferece o direito de cancelamento.

Sabia que isso é um direito assegurado pela LGPD? O usuário possui a prerrogativa de revogar o consentimento, restringi-lo ou até mesmo excluí-lo.

Além disso, os participantes mais comprometidos costumam ser aqueles que genuinamente querem participar e oferecem mais insights ao seu evento. É fundamental ter uma equipe capacitada para cumprir a LGPD, que não só garante o direito de cancelamento, mas também assegura que o participante possa acessar todas as suas informações pessoais, solicitar detalhes sobre o seu uso, fazer correções, estabelecer limites, entre outros.

Contudo, é importante lembrar que a política de privacidade do Evento se limita à maneira como a plataforma de eventos irá utilizar as informações. Esse é um aspecto crucial a ser considerado em uma plataforma de eventos, pois garante a segurança tanto dos organizadores quanto dos participantes.

Contudo, é essencial que o responsável pela organização esclareça de maneira transparente como irá utilizar as informações coletadas dos participantes, especialmente caso haja necessidade de solicitar dados adicionais aos já requeridos no momento da inscrição.

Então chegou a hora de criar a sua política de privacidade! Neste documento, deve estar explícito:

- a) Quais dados pessoais serão tratados;
- b) Para qual finalidade;
- c) Como, onde e por quanto tempo são armazenados;
- d) Com quem os dados são compartilhados e sua responsabilidade.

Além disso, a LGPD estabelece quatro protagonistas no contexto da legislação. Ou seja, indivíduos com responsabilidades bem definidas que devem ser contempladas em seu documento de privacidade. Esses agentes são:

- a) **Titular:** como o nome já diz, é o dono dos dados tratados;
- b) **Controlador:** pessoa ou empresa responsável pelo tratamento dos dados pessoais. É quem coordena como o dado será coletado, usado e armazenado;
- c) **Operador:** É a empresa ou pessoa que realiza o tratamento dos dados pessoais em nome do controlador. Por exemplo, a plataforma de eventos usada pelo organizador;

- d) **Encarregado:** Também conhecido como *Data Protection Officer* (DPO), ele é a pessoa indicada pelo controlador para atuar como canal de comunicação entre ele, os titulares dos dados e a Autoridade Nacional de Proteção de Dados (ANPD).

Contudo, é importante ressaltar que nem todos os indivíduos precisam dar importância ao papel do encarregado. A ANPD (Autoridade Nacional de Proteção de Dados) isenta os microempreendedores e as pequenas empresas da obrigação de ter essa função.

Em resumo, a política de privacidade precisa revelar a forma como as informações do usuário serão manipuladas, quais são os direitos dele, qual é a responsabilidade do responsável pelo processamento de dados e, se for o caso, fornece orientações para o responsável pela conformidade. Dessa maneira, a política de privacidade garante a transparência no uso das informações, mesmo para aqueles que não estão familiarizados com esse processo.

Dessa maneira, a fim de garantir maior acessibilidade, sugiro a busca por ajuda de um especialista qualificado para elaborar uma política de privacidade em conformidade com a LGPD, como por exemplo um advogado.

6.3. Criptografia de dados online

No contexto do museu multitemático sobre a primeira viagem do homem à lua, informações sensíveis poderiam incluir, por exemplo, a opinião política sobre a exploração espacial, convicções religiosas relacionadas à lua ou até mesmo dados biométricos se houver algum tipo de identificação biométrica associada ao questionário.

Para garantir a conformidade com a LGPD, o questionário não deve solicitar ou coletar essas informações sensíveis dos visitantes. Em vez disso, deve-se focar em perguntas relacionadas à experiência da exposição e ao feedback sobre o museu.

Para proteger dados sensíveis, como e-mails e CEP em conformidade com a Lei Geral de Proteção de Dados (LGPD), é recomendável usar criptografia robusta. Isso pode incluir:

- a) **Criptografia de Dados em Repouso:** Garante que os dados armazenados estejam protegidos, mesmo quando não estão em uso. Isso pode ser alcançado por meio de algoritmos de criptografia forte aplicados aos dados armazenados em bancos de dados ou arquivos;
- b) **Criptografia de Dados em Trânsito:** Protege os dados enquanto estão sendo transmitidos pela rede. Usar protocolos de comunicação seguros, como HTTPS para transmissão de e-mails e comunicações online, é essencial;
- c) **Criptografia de Ponta a Ponta:** Para comunicações sensíveis, como troca de e-mails contendo informações pessoais, a criptografia de ponta a ponta garante que apenas o remetente e o destinatário possam acessar o conteúdo das mensagens;
- d) **Gerenciamento de Chaves Seguro:** As chaves de criptografia devem ser armazenadas e gerenciadas de forma segura para evitar acessos não autorizados.

Além disso, é importante adotar práticas de segurança adicionais, como controle de acesso, monitoramento de atividades suspeitas e implementação de políticas de privacidade e segurança de dados. Essas medidas não apenas ajudam a proteger os dados sensíveis, mas também garantem conformidade com as regulamentações de proteção de dados, como a LGPD.

Esses seriam meios a ser aplicados na prática no museu em si, para seguirmos a LGPD e conseguirmos garantir a segurança do público.

7. DESENVOLVIMENTO

A elaboração do plano do projeto se baseia nas informações geradas pelos demais processos, incluindo a estratégia de planejamento, a fim de produzir um documento claro e coeso que sirva de orientação tanto para a realização quanto para o acompanhamento do projeto. Geralmente, este processo é realizado diversas vezes.

7.1. Engenharia de software

Em resumo, esses especialistas planejam e orientam o progresso de softwares, apps e sistemas, garantindo que estejam de acordo com as exigências e desempenhem as funções estabelecidas.

7.1.1. Processo de desenvolvimento de software

A criação do programa para o totem de interação no museu pode ser realizada através de um processo escalonado e progressivo, como o Scrum, seguindo as etapas a seguir:

- a) **Elaboração do planejamento:** Estabelecimento das metas e abrangência do projeto, identificação das necessidades e estabelecimento do cronograma e recursos requeridos;
- b) **Levantamento de Necessidades:** Investigação e estudo das demandas e anseios dos usuários, identificação dos requisitos operacionais e técnicos do sistema;
- c) **Iniciativa:** Elaboração da estrutura do programa, contemplando a organização de classes, interfaces e módulos do sistema, assim como a interação com o usuário;
- d) **Execução:** Elaboração da aplicação de software conforme as diretrizes e plano estabelecidos, empregando a linguagem de programação C# e a ferramenta *Windows Forms*, tal como detalhado na tarefa;
- e) **Testes de software:** Realização de testes individuais, de integração e de aceitação para assegurar que o programa atenda às necessidades e opere de maneira adequada;

- f) **Implementação:** Colocação do programa no terminal interativo do centro cultural, assegurando seu desempenho adequado e preparando-o para ser utilizado pelos frequentadores;
- g) **Controle e Manutenção:** Monitoramento da eficiência do programa em execução, correção de eventuais falhas e implementação de melhorias. A abordagem de desenvolvimento iterativa e incremental, como o Scrum, é recomendada para o projeto devido à entrega de funcionalidades do software em pequenos intervalos de tempo, possibilitando receber retorno dos usuários e adaptar o software de acordo com as necessidades. Esse método oferece uma maneira flexível e adaptável ao processo de criação, fundamental em projetos com requisitos suscetíveis a mudanças ao longo do tempo.

7.1.2. RN, RF e RNF

Um programa de computador contém uma série de exigências classificadas em funcionais (RF) e não funcionais (RNF), além do requisito do negócio (RN) que representa as especificações que definem as necessidades de um negócio em relação a um sistema ou solução de informação. Essas exigências costumam englobar descrições de recursos e funcionalidades, assim como eventuais restrições ou limitações técnicas.

Os requisitos funcionais descrevem o comportamento do sistema, enquanto os requisitos não funcionais estabelecem critérios que podem ser usados para julgar a operação de um sistema, em vez de comportamentos específicos. (Sommerville, 2011, p. 59).

7.2. Levantamento de requisitos

A identificação de necessidades é crucial no processo de criação de programas, como no caso do software destinado ao totem digital do museu. Tal procedimento consiste na coleta, avaliação e registro das exigências e anseios dos usuários, assim como dos requisitos operacionais e não operacionais do sistema.

Essa fase é primordial para assegurar que o programa atenda plenamente às demandas dos usuários e direcione o desenvolvimento de maneira produtiva e eficaz.

7.3. Requisito do negócio

O requisito de negócio para o sistema do totem interativo do museu consistiria em proporcionar aos frequentadores uma experiência educativa e de interação, possibilitando que acessem informações sobre a primeira viagem humana à Lua, visualizem peças relacionadas ao tema, participem de uma pesquisa sobre a exposição e obtenham um retorno sobre suas respostas. Além disso, o sistema precisa possibilitar que a equipe responsável pelo museu colete dados sobre a interação dos visitantes com o totem, a fim de realizar análises e tomar decisões estratégicas.

O software de pesquisa de satisfação deve processar as avaliações de maneira ágil e segura, sendo imprescindível o feedback dos clientes para orientar melhorias contínuas nos serviços, conforme os princípios de gestão da qualidade estabelecidos. Além disso, o sistema irá coletar as informações e dados não sensíveis obtidos por essa pesquisa e irá gerar um relatório consolidado com filtros gerais (como dia, mês, ano, exposição), para análise da equipe e elaboração de planos para futuras melhorias.

Adicionalmente, o totem também será responsável pela venda de ingressos para os interessados nas atrações exibidas, oferecendo opções de meia-entrada, entrada inteira e isenção de taxas.

7.4. Requisitos funcionais

A determinação dos requisitos para a criação de software é influenciada pela espécie de projeto a ser desenvolvido, pelos usuários potenciais envolvidos e pela abordagem geral adotada pela organização ao definir os requisitos. A falta de clareza e consistência nos requisitos pode resultar em insatisfação do cliente com o projeto, pois dificulta a comunicação entre o desenvolvedor e o usuário final.

Em resumo, as interfaces do projeto serão projetadas para melhorar a experiência do visitante. Portanto, a definição dos requisitos funcionais é uma etapa crucial do projeto, com o objetivo de estabelecer funções claras para atender às necessidades do projeto como um todo.

No contexto da Lei Geral de Proteção de Dados (LGPD), os clientes são obrigados a se registrar e fazer login para comprar ingressos, participar da avaliação das obras e da pesquisa de satisfação. Durante esse processo, dados como idade, nome, CEP e e-mail são coletados. Serão coletadas informações válidas e compartilháveis de acordo com as diretrizes estabelecidas pela LGPD (Lei Geral de Proteção de Dados). Garantiremos que os dados sejam protegidos e criptografados, visando preservar a identidade dos usuários.

Será enviado por e-mail ao visitante um token logo após a compra do ingresso. Esse token será utilizado para participar da pesquisa de satisfação e da avaliação das obras, no entanto, sua participação não é obrigatória. O sistema deve oferecer uma página de avaliação com opções que variam de “Excelente” a “Péssimo”. Esta abordagem de pesquisa tem como objetivo incentivar os participantes a expressarem sua satisfação ou insatisfação com a experiência no museu. Ademais, uma avaliação completa do museu será realizada para identificar áreas de melhoria que possam atrair mais clientes.

Os usuários devem ter a capacidade de escolher a opção de compra de ingresso, bem como de avaliar as obras. Ao final da visita, eles serão convidados a participar da pesquisa de satisfação. Após a conclusão, o sistema gerará automaticamente um relatório abrangente sobre a interação do usuário feita no dia, que será enviado para a empresa.

Este relatório desempenhará um papel crucial como um instrumento de controle para avaliar o desempenho do museu. Ele será responsável por armazenar e analisar as interações ocorridas durante a visita ao museu, com o objetivo de fornecer dados sobre a satisfação do cliente, as avaliações das obras, as vendas realizadas no dia e identificar áreas que necessitam de aprimoramento.

As funcionalidades necessárias para o funcionamento do totem interativo do museu englobam:

- a) **Pesquisa de Trabalhos:** Os frequentadores precisam ter a habilidade de buscar dados sobre os trabalhos correlatos à primeira saga do ser humano para a Lua, tais como narrativas e ilustrações;
- b) **Exibição de Arte:** Os espectadores devem ter a possibilidade de apreciar fotos das obras vinculadas ao assunto da mostra;

- c) **Feedback para Avaliação:** Os frequentadores são encorajados a participar de uma avaliação sobre a mostra, respondendo a perguntas de seleção múltipla;
- d) **Resultado da Coleta de Dados:** É imprescindível que o sistema some as respostas dos questionários dos usuários e exiba os dados de maneira objetiva e compreensível;
- e) **Retorno ao Usuário:** Após completar a pesquisa, o sistema irá apresentar um retorno ao usuário, exibindo os resultados obtidos e talvez fornecendo dados extras com base em suas respostas;
- f) **Levantamento de Informações:** O sistema necessita obter informações sobre a forma como os visitantes interagem com o totem, tais como as obras visualizadas e as respostas fornecidas na pesquisa, a fim de realizar uma análise e embasar as decisões da equipe do museu;
- g) **Proteção das Informações:** É fundamental que as informações pessoais dos usuários sejam coletadas e guardadas de maneira segura, seguindo os requisitos da LGPD, assegurando a confidencialidade e a integridade dos dados.

7.5. Requisitos não funcionais

Os critérios não funcionais particulares para o sistema do totem digital do museu podem abranger:

- a) **Performance:** É necessário que o sistema tenha a capacidade de suportar uma grande quantidade de usuários acessando ao mesmo tempo, assegurando respostas ágeis e eficazes às solicitações dos usuários;
- b) **Facilidade de uso:** O design da interface do totem deve ser simples e amigável, de forma a ser compreensível para todos os usuários, independentemente do seu nível de conhecimento em tecnologia;
- c) **Proteção:** É fundamental que o sistema assegure a proteção das informações dos usuários, impedindo que sejam acessadas por pessoas não autorizadas e garantindo o cumprimento da LGPD;
- d) **Credibilidade:** O sistema precisa ser estável e imune a erros, assegurando sua operação adequada ao longo de toda a sua utilização;

- e) **Interoperabilidade:** A plataforma precisa ser apta para operar em variados aparelhos de hardware, como telas sensíveis ao toque e teclados virtuais, assegurando sua utilização em múltiplos contextos;
- f) **Mobilidade:** É essencial que o sistema possa ser facilmente deslocado para várias locações de exposição, possibilitando seu uso em diversos museus ou eventos;
- g) **Manutenção:** É essencial que o sistema possua uma estrutura que facilite sua manutenção e atualização, garantindo a possibilidade de adicionar novas funcionalidades e corrigir possíveis erros com facilidade;
- h) **Documentação obrigatória:** O sistema deve vir acompanhado de uma documentação detalhada e de fácil entendimento, para facilitar a compreensão e futuras manutenções por parte da equipe do museu.

7.6. Como implementar a tecnologia no museu

Para implementarmos ele no estabelecimento é importante seguir um processo cuidadoso e bem planejado. Aqui estão algumas das etapas que devemos seguir:

- a) **Planejamento e análise:** Estabelecer as metas da execução e as demandas do totem digital. Efetuar uma avaliação da viabilidade tecnológica e econômica do empreendimento;
- b) **Design e desenvolvimento:** Escolher as tecnologias corretas para a criação do totem digital interativo. Desenvolver a concepção visual do programa e da interação com o usuário. Criar o programa para o totem digital, obedecendo às diretrizes sugeridas;
- c) **Testes e validação:** Efetuar avaliações de usabilidade, performance e proteção do terminal de autoatendimento interativo. Verificar o programa de computador de acordo com as especificações estabelecidas;
- d) **implantação:** Alocar o quiosque interativo no local de exposição, assegurando sua harmonização com o espaço físico. Executar verificações finais após a conclusão da instalação a fim de assegurar o correto funcionamento;
- e) **Treinamento e capacitação:** Oferecer capacitação para os funcionários do museu acerca da utilização e manutenção do totem digital;

- f) **Monitoramento e manutenção:** Acompanhar o funcionamento do painel interativo após a instalação. Realizar revisões periódicas a fim de assegurar o pleno desempenho do equipamento;
- g) **Avaliação e melhoria contínua:** Analisar a performance do totem digital considerando os objetivos estabelecidos. Implementar aprimoramentos no programa e na usabilidade com base nas sugestões dos frequentadores e dos colaboradores do museu;
- h) **Documentação e registro:** Registrar todos os passos executados durante a implementação, desde os pré-requisitos, planejamento, criação, verificação e execução. Registrar de forma regular todas as informações referentes ao bom funcionamento e à manutenção do totem interativo.

8. GARANTIA DA QUALIDADE DO SOFTWARE NA EMPRESA

Dentro de uma empresa, é fundamental atender às necessidades internas e seguir padrões para melhorar o desempenho do sistema. Para isso, as normas ISO oferecem diretrizes essenciais. Elas incluem características e métricas que garantem um fluxo de funcionamento adequado e uma maior qualidade de software.

Ao seguir as normas ISO, as empresas podem melhorar seus processos internos, garantir a conformidade com padrões internacionais e, conseqüentemente, aumentar a qualidade dos seus produtos e serviços. Isso pode resultar em maior satisfação do cliente, maior eficiência operacional e uma posição mais forte no mercado.

8.1. Padrões e Modelos de Qualidade no Desenvolvimento de Software

- a) **ISO 13407:** a ISO 13407 estabelece diretrizes para o design centrado no usuário. em nosso projeto, estamos incorporando os princípios desse padrão em todas as etapas do desenvolvimento, incluindo pesquisa de usuários, desenvolvimento de perfis de usuário, prototipagem iterativa e testes de usabilidade. essas práticas visam assegurar uma experiência amigável ao usuário e adequada ao usuário final;
- b) **SO/IEC 9126-1:** Estabelece um modelo de qualidade para o software. Os princípios subjacentes a esse modelo de qualidade de software incluem:
- c) **Funcionalidade:** Garantir que o software execute as funções pretendidas de forma precisa e eficaz quando o software estiver sendo utilizado;
- d) **Confiabilidade:** Conforme observado por Sommerville (1995) em seu artigo, a credibilidade se distingue de segurança e proteção, sendo um atributo avaliável do sistema. Refere-se à garantia de que o software opere de maneira estável e consistente, minimizando falhas e erros. No entanto, é crucial exercer vigilância para garantir que a confiabilidade necessária tenha sido alcançada. Dessa forma, mesmo quando muitos usuários interagem com o hardware e o software em grande escala, podemos realizar melhorias constantes com base na avaliação feita pelos próprios usuários finais;
- e) **Usabilidade:** Segundo Nielsen (1993), a usabilidade não é uma propriedade simples da interface com o usuário.

A usabilidade engloba uma série de aspectos e está frequentemente relacionada a diferentes elementos, como a facilidade de aprendizado do usuário, a eficiência na utilização do produto, a capacidade de memorização e recuperação das informações pelo usuário e a frequência de problemas durante o uso do sistema. Um sistema mais amigável e intuitivo facilita significativamente a experiência do usuário, promovendo uma interação mais fluida e eficaz com o software.

Para melhorar a satisfação do usuário ao utilizar o sistema, é fundamental que a interface seja intuitiva e fácil de usar, simplificando o processo para os clientes. As páginas de opções de compra de ingresso, interação com as obras e pesquisa de satisfação devem ser claras e organizadas, com uma disposição visual que facilite a escolha do cliente e deixe clara a finalidade de cada produto. Dessa forma, enfatizamos a importância da avaliação do usuário, visando que gestores e responsáveis pelo estabelecimento possam tomar medidas para implementar melhorias contínuas na empresa:

- a) **Eficiência:** A eficiência no projeto é alcançada garantindo o adequado desempenho do software e dos recursos do sistema. Isso assegura que o software funcione corretamente e disponibilize mais tempo de uso, além de determinar como ele se comportará em relação aos recursos disponíveis. Para que o sistema seja eficiente e atenda às necessidades dos usuários, é necessário garantir um tempo de resposta e processamento ágil. Para alcançar esse resultado, é imprescindível que o usuário cumpra os requisitos para utilizar plenamente o potencial do software;
- b) **Manutenibilidade:** Para garantir a facilidade na manutenção do software e aprimorar a qualidade da análise, é fundamental que ele seja executado corretamente desde o início. Isso permite uma organização compreensível e ajuda a prevenir problemas futuros. O desenvolvimento do software simplifica a análise e compreensão de problemas, auxiliando na identificação de áreas para melhoria durante manutenções ou atualizações. A importância da manutenibilidade no desenvolvimento de software foca na necessidade de um sistema menos complexo e de fácil modificação, mantendo a estabilidade do sistema;
- c) **Gerenciamento e manutenção:** Segundo Chikofsky (1990) em seu artigo

muitos softwares dos quais dependemos hoje têm em média de 10 a 15 anos. Mesmo quando esses programas foram criados, usando as melhores técnicas de projeto e codificação conhecidas na época [muitos não foram], o tamanho do programa e o espaço de armazenamento eram as preocupações principais. Eles então migraram para novas plataformas, foram ajustados para mudanças nas máquinas e na tecnologia dos sistemas operacionais e aperfeiçoados para atender a novas necessidades dos usuários – tudo isso sem grande atenção na arquitetura geral. O resultado são estruturas mal projetadas, mal codificadas, de lógica pobre e mal documentadas em relação aos sistemas de software, para os quais somos chamados a fim de mantê-los rodando.

A manutenção e funcionamento do software são cruciais para preservar sua eficácia e evitar falhas. Estabelecer uma rotina de manutenção é primordial para antecipar problemas durante o uso. Para isso, uma equipe altamente treinada foi designada para assegurar a constância da manutenção do produto, buscando soluções ágeis para corrigir eventuais falhas. Com esse propósito, estabelecemos um departamento dedicado à garantia do pleno funcionamento do software e do totem, contribuindo assim para a viabilidade operacional do projeto.

Adicionalmente, mediante a implementação da LGPD, garantimos a segurança dos dados. Introduzimos recursos de segurança para proteger as informações armazenadas, captadas e processadas localmente, assegurando a proteção de todos os indivíduos.

Igualmente, nomearemos uma equipe para comandar a operação e a usabilidade, com o intuito de alcançar a excelência no serviço. Nossa meta elevar o desenvolvimento contínuo e integrado, refinando incessantemente nossos métodos para proporcionar suporte eficiente a todos os usuários.

8.2. Ciclo de Vida do Software: Garantindo Qualidade e Eficiência

Para garantir que um software tenha um ciclo de vida que assegure qualidade e eficiência desde sua concepção até sua descontinuação, é necessário passar por diversas etapas de aprimoramento. Isso inclui testes, implementação, manutenção

durante o uso e o procedimento de desligamento do software. Com base nesses aspectos, vamos detalhar como esse processo é realizado.

- a) **Planejamento:** Nesta fase inicial, é essencial definir as necessidades do cliente em relação ao projeto. Isso inclui a entrega de um software de qualidade, para o qual é necessário determinar os recursos a serem utilizados, estabelecer prazos de entrega, orçamentos necessários para o projeto e definir requisitos, escopo, cronograma e alocação de recursos. Essas medidas nos permitem uma organização adequada, garantindo uma comunicação clara entre o cliente, o projeto e os desenvolvedores;
- b) **Desenvolvimento:** Desenvolvimento, seguiremos os critérios definidos e o design fornecido pelo cliente. Implementaremos tecnologias essenciais para o desenvolvimento do software e para a estrutura do totem, garantindo que as funcionalidades sejam compatíveis e coesas. Além disso, serão realizados testes e atualizações regulares para evitar qualquer mau uso do produto;
- c) **Testes:** Os testes serão conduzidos considerando a experiência do usuário e suas interações com o sistema. Através da análise das avaliações, nossa equipe de especialistas estará apta a identificar e corrigir de maneira precisa quaisquer falhas no software ou na estrutura do totem. Adicionalmente, asseguraremos que o sistema preserve sua eficiência diante das demandas habituais de uso pelo público e das interações realizadas;
- d) **Implementação:** Para garantir uma implementação bem-sucedida, é fundamental que o ambiente ofereça condições ideais para o funcionamento do totem. Isso inclui um espaço amplo e confortável, estrategicamente posicionados para evitar possíveis danos ou interferências. Além disso, é essencial posicionar o totem e as obras em locais de fácil localização para os visitantes, garantindo uma experiência agradável desde o primeiro contato. Testes regulares são conduzidos para avaliar a adequação do ambiente, assegurando um espaço seguro e acessível para todos os usuários;
- e) **Manutenção:** Após a implementação, o software é monitorado constantemente para evitar bugs, implementar atualizações de segurança,

melhorar o desempenho e adicionar novos recursos conforme necessário, mesmo com muitos usuários utilizando-o. Isso garante que a qualidade do software permaneça cada vez mais estável ao longo do tempo. O totem contará com uma equipe exclusiva de suporte e manutenção designada para auxiliar os visitantes, garantindo que possam desfrutar corretamente do totem. As manutenções serão agendadas regularmente para garantir que o totem preserve sua operacionalidade contínua ao decorrer do uso;

- f) **Descontinuação:** Caso haja uma possível descontinuação, é importante considerar que o software pode se tornar superado e, eventualmente, ser substituído por uma versão mais recente. Nesse caso o software do totem do museu irá se manter até o término no evento após acabar ele ainda pode ser reutilizado em futuras exposições.

8.3. Gestão estratégica de RH

Neste trabalho, concentramo-nos no Museu Multitemático, onde a implementação da GERH foi realizada de forma assertiva. Desde a seleção e capacitação das equipes responsáveis pela operação dos totens interativos até a formulação de políticas de reconhecimento e incentivo, cada etapa foi cuidadosamente planejada e executada para garantir a eficácia e o alinhamento com os objetivos organizacionais.

Ao enfrentar desafios complexos durante os processos de mudança, adotamos uma abordagem minuciosa e estratégica, reconhecendo a importância de um planejamento cuidadoso e do acompanhamento próximo, em uma colaboração conjunta com a administração.

Neste trabalho também explorou o conceito de grupos de trabalho, destacando a importância de objetivos claros, papéis bem definidos e liderança eficiente para o sucesso coletivo. Compreendemos a relevância da diversidade para o pensamento criativo e o desempenho de alta qualidade, e reconhecemos a importância da habilidade de negociação para resolver conflitos e promover relacionamentos sólidos e produtivos.

Com esta abordagem da GERH e sua aplicação prática no Museu Multitemático, buscamos contribuir para o aprimoramento do desempenho organizacional e o bem-estar dos colaboradores.

8.4. Infraestrutura de rede

A infraestrutura de rede é crucial para a coordenação e preservação dos equipamentos tecnológicos de uma empresa, abrangendo desde os computadores até os servidores. Ela inclui redes de comunicação, cabos, roteadores, switches e firewalls, que juntos garantem uma conectividade sólida e eficaz.

Adicionalmente, a rede de comunicação é encarregada de garantir a eficiência no envio de informações de maneira ágil e protegida, viabilizando a interação tanto dentro quanto fora da empresa, a utilização de softwares e sistemas corporativos, e a preservação e recuperação de dados essenciais.

Para garantir o funcionamento adequado da infraestrutura, é fundamental estabelecer políticas de segurança eficazes, realizar monitoramento constante para identificar e solucionar possíveis falhas, além de manter equipamentos e softwares sempre atualizados para acompanhar as mudanças tecnológicas e demandas da empresa.

Uma infraestrutura de rede bem elaborada e administrada é essencial para manter as atividades cotidianas e promover a flexibilidade e capacidade de adaptação da empresa às novas exigências e obstáculos. Dessa forma, torna-se um alicerce crucial para a eficácia operacional e a competitividade organizacional.

8.5. Estrutura do museu

Utilizaremos como guia para o planejamento de nossas ações uma representação gráfica do museu na figura 1, contendo informações sobre seu tamanho e a localização dos equipamentos (Ilustração utilizada apenas como exemplo, para fins de referência).

8.5.1. Cabeamento

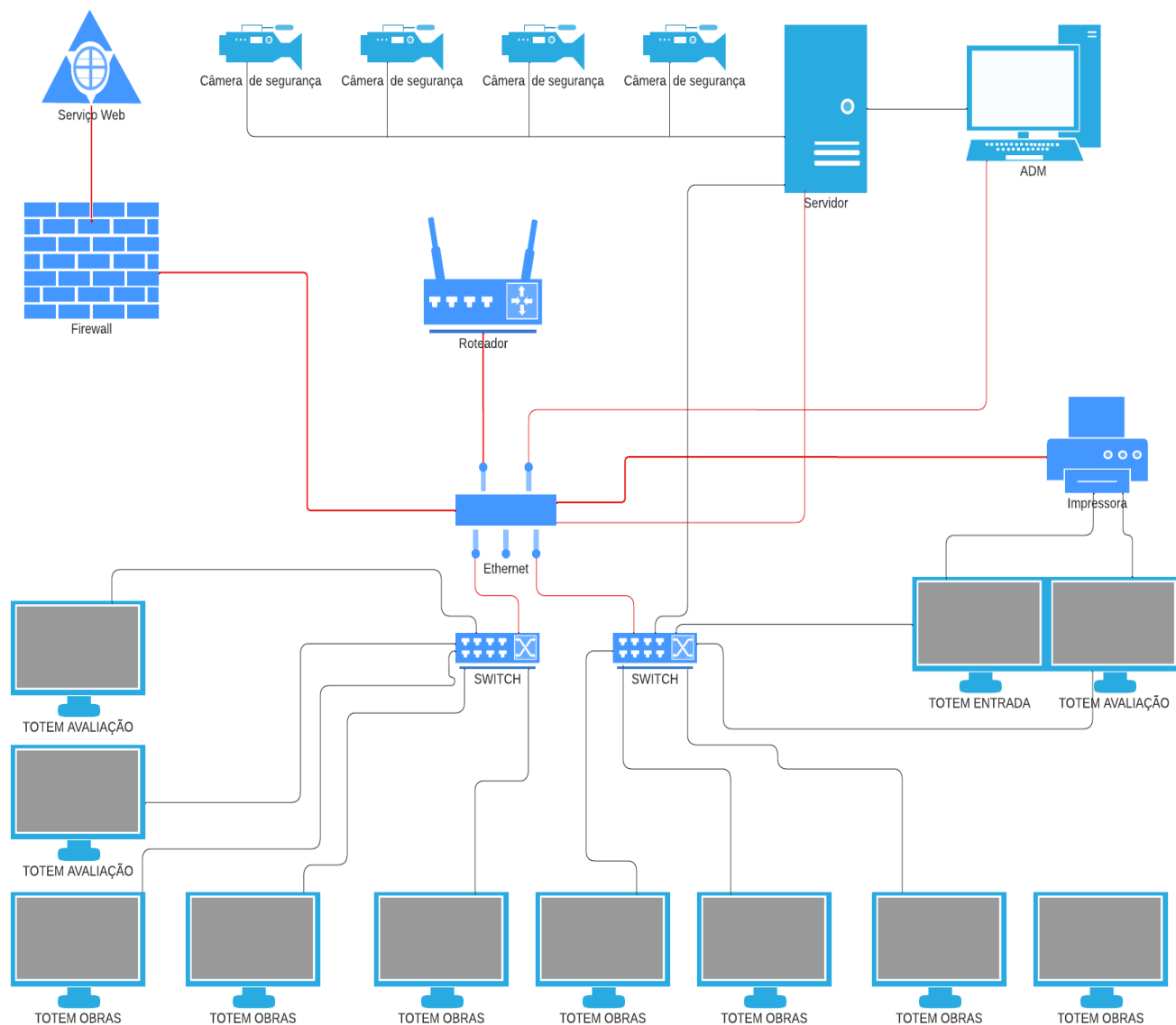
Após a identificação geral dos equipamentos, é possível visualizar, na figura 2, o esquema de ligação em rede deles, juntamente com os componentes que integram essa estrutura.

Figura 1 - Planta Baixa Museu



Fonte: Autoria própria.

Figura 2 – Diagrama de rede



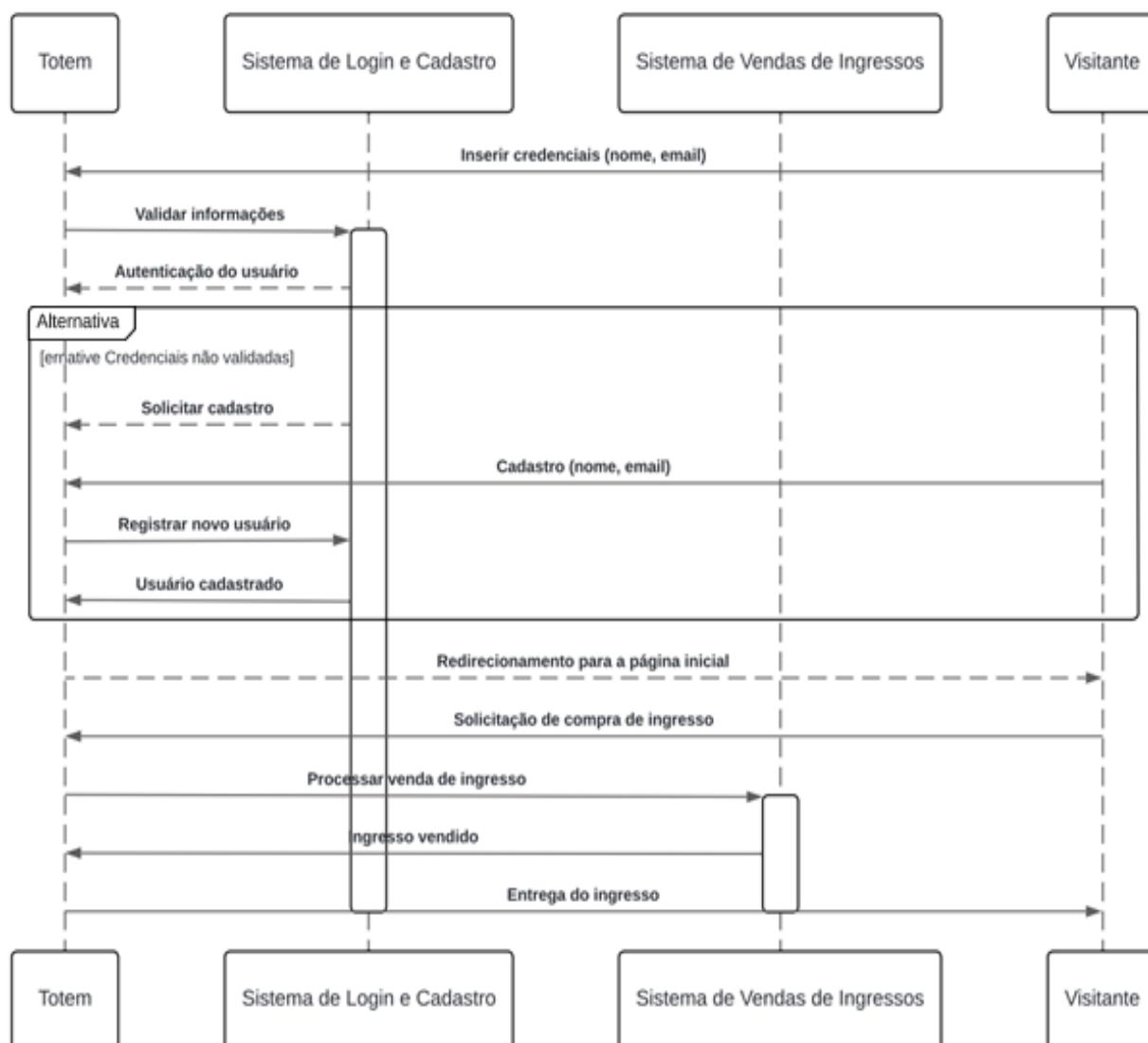
Fonte: Autoria própria.

8.5.2. Diagrama de sequência

Um diagrama de sequência é uma representação visual da interação entre objetos em um sistema de software, mostrando como eles se comunicam ao longo do tempo por meio de mensagens.

O diagrama mostrado na figura 3 mostra como seria o funcionamento do totem de entrada e seu funcionamento referente a pedidos dos visitantes.

Figura 3 – Diagrama de sequência totem de entrada



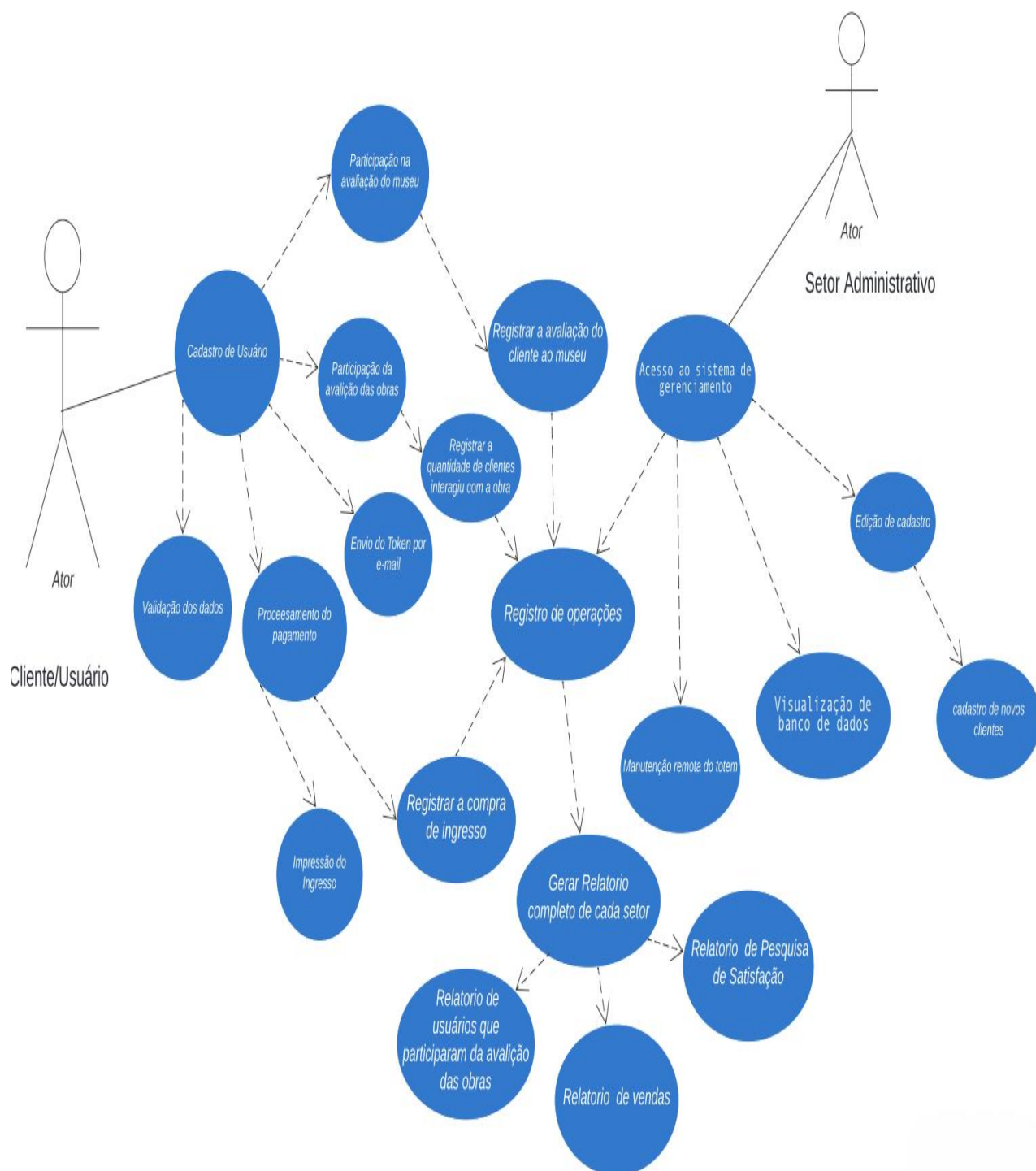
Fonte: Autoria própria.

8.5.3. Diagrama de casos de uso

O diagrama de uso, ou diagrama de casos de uso, representa graficamente a interação entre um sistema e seus atores para alcançar um objetivo específico. Ele documenta e visualiza como um sistema interage com seus usuários e é útil para comunicar requisitos, validar o entendimento dos requisitos, projetar interfaces e identificar casos especiais de uso.

Na figura 4 a seguir podemos ver as interações entre atores e o sistema para melhor entendimento.

Figura 4 – Diagrama de casos de uso



Fonte: Autoria própria.

Suas aplicações em sistema têm suas peculiaridades na qual explicaremos cada uma a seguir na tabela 1.

Tabela 1 – Explicação das funções apresentadas e suas peculiaridades

Caso de Uso: Cadastro de Usuário
Atores: Cliente/Usuário
Interessados: Cliente/Usuário Setor Administrativo do museu
Pós-condições: <ul style="list-style-type: none"> a) O cliente recebe um token por e-mail para participar da avaliação do museu; b) Os dados da compra do ingresso e a avaliação do museu são registrados no sistema.
Requisitos correlacionados: <ul style="list-style-type: none"> a) Sistema de pagamento online habilitado; b) O Banco de dados de clientes serve para armazenar informações de cadastro; c) Sistema de envio de e-mails para entrega do token; d) Sistema de avaliação do museu para coletar e armazenar feedback dos clientes.
Variações Tecnológicas: <ul style="list-style-type: none"> a) Utilização de diferentes provedores de pagamento online (cartão de crédito, débito, PIX); b) Integração com diferentes sistemas de envio de e-mails.
Fluxo Principal: <ul style="list-style-type: none"> a) O Cliente/Usuário acessa o sistema de cadastro de usuário no totem do museu; b) O sistema exibe um formulário de cadastro para o Cliente/Usuário preencher com suas informações pessoais, incluindo e-mail, nome, idade e CEP; c) O Cliente/Usuário preenche o formulário e seleciona a forma de pagamento para a compra do ingresso (cartão de crédito, débito ou PIX); d) O sistema valida os dados inseridos pelo Cliente/Usuário; e) Se os dados estiverem corretos, o sistema processa o pagamento utilizando o sistema de pagamento online habilitado; f) Após o pagamento ser processado com sucesso, o sistema envia um token para o e-mail cadastrado pelo Cliente/Usuário; g) Após o pagamento ser processado com sucesso, o sistema realiza a impressão do ingresso h) O Cliente/Usuário recebe o e-mail com o token e finaliza o processo de cadastro; i) O Cliente/Usuário pode usar o token recebido para participar da avaliação do museu no totem de saída após sua visita; j) Os dados da compra do ingresso e a avaliação do museu são registrados no sistema para fins de análise e gerenciamento pelo Setor Administrativo do museu.
Tratamento de Exceções: <ul style="list-style-type: none"> a) Dados inválidos: Se o Cliente/Usuário inserir dados inválidos (como e-mail mal formatado, nome, idade e CEP inválido) no formulário de cadastro, o sistema exibe uma mensagem de erro e solicita correções; b) Falha no pagamento: Em caso de falha no processamento do pagamento (por exemplo, devido a problemas de conexão ou erro na transação), o sistema informa ao Cliente/Usuário sobre a falha e oferece opções para tentar novamente ou escolher outra forma de pagamento; c) E-mail não enviado: Se ocorrer uma falha no envio do e-mail com o token, o sistema registra o problema e tenta enviar novamente em momento posterior. O Cliente/Usuário é informado sobre a falha e orientado sobre como proceder; d) Token não recebido: Se o Cliente/Usuário relatar não ter recebido o token, o sistema oferece opções para reenviar o e-mail com o token ou fornecer um novo token no totem, após verificação da identidade; e) Problemas técnicos: Em caso de problemas técnicos (como falhas de conexão ou sistema), o sistema exibe uma mensagem de erro e sugere tentar novamente mais tarde ou contatar o suporte técnico.

Caso de Uso: Gerenciamento Administrativo
Atores: Administrativo
Interessados: Setor Administrativo do museu
Pós-condições: O administrador realiza as operações de gerenciamento com sucesso.
Requisitos correlacionados: <ul style="list-style-type: none"> a) Banco de dados de clientes e vendas; b) Sistema de relatórios de vendas e avaliação; c) Sistema de controle remoto para o totem.
Variações Tecnológicas: <ul style="list-style-type: none"> a) Utilização de sistema de autenticação seguro para acesso do administrador; b) Integração com banco de dados para acesso às informações relevantes; c) Implementação de ferramentas de controle remoto para interação com o totem.
Fluxo Principal: <ul style="list-style-type: none"> a) O administrador acessa o sistema de gerenciamento administrativo utilizando suas credenciais; b) Após autenticação bem-sucedida, o administrador é direcionado ao painel de controle; c) No painel de controle, o administrador tem acesso às seguintes funcionalidades: <ul style="list-style-type: none"> a) Visualizar banco de dados de clientes e vendas; b) Gerar relatórios de vendas e avaliação; c) Realizar operações de manutenção remota no totem, como atualizações de conteúdo; d) Editar cadastros de clientes, se necessário; e) Cadastrar novos clientes, se necessário; f) O administrador realiza as operações desejadas, garantindo que as alterações sejam aplicadas corretamente no sistema; g) Após concluir as operações, o administrador encerra a sessão de administração.
Tratamento de Exceções: <ul style="list-style-type: none"> a) Se o administrador inserir credenciais inválidas, o sistema exibe uma mensagem de erro e solicita credenciais válidas; b) Se houver problemas de conexão com o banco de dados, o sistema exibe uma mensagem de erro e sugere tentar novamente mais tarde; c) Se ocorrer uma falha ao tentar acessar o totem remotamente, o sistema registra o problema e oferece opções para solucioná-lo.
Caso de Uso: Pesquisa de Satisfação
Atores: Cliente/Usuário
Interessados: Setor de Gestão de Qualidade do Museu
Precondições: O Cliente/Usuário deve ter visitado o museu e recebido o token de acesso à pesquisa.
Pós-condições: O feedback do Cliente/Usuário é registrado no sistema de avaliação do museu.
Requisitos correlacionados: <ul style="list-style-type: none"> a) Sistema de envio de e-mails para fornecer o token de participação na pesquisa. b) Sistema de avaliação do museu coletar e armazenar feedback dos clientes no banco de dados. c) Variações Tecnológicas: Utilização de provedores de envio de e-mails para garantir a entrega do token através do e-mail. d) Integração com o sistema de avaliação do museu para registrar o feedback dos clientes de forma eficiente.

Fluxo Principal:

- a) O Cliente/Usuário recebe o token de acesso à pesquisa por e-mail logo após a compra do ingresso;
- b) O Cliente/Usuário acessa o totem de pesquisa de satisfação disponível no museu;
- c) No totem, o Cliente/Usuário insere o token recebido por e-mail para iniciar a pesquisa;
- d) O Cliente/Usuário responde às perguntas da pesquisa, fornecendo feedback sobre sua experiência no museu;
- e) Após concluir a pesquisa, o Cliente/Usuário encerra a sessão e o feedback é registrado no sistema de avaliação do museu.

Tratamento de Exceções:

- a) Dados inválidos: Se o Cliente/Usuário inserir dados inválidos (como e-mail mal formatado, nome, idade e CEP inválido) no formulário de cadastro, o sistema exibe uma mensagem de erro e solicita correções.
- b) Falha no pagamento: Em caso de falha no processamento do pagamento (por exemplo, devido a problemas de conexão ou erro na transação), o sistema informa ao Cliente/Usuário sobre a falha e oferece opções para tentar novamente ou escolher outra forma de pagamento.
- c) E-mail não enviado: Se ocorrer uma falha no envio do e-mail com o token, o sistema registra o problema e tenta enviar novamente em momento posterior. O Cliente/Usuário é informado sobre a falha e orientado sobre como proceder.
- d) Token não recebido: Se o Cliente/Usuário relatar não ter recebido o token, o sistema oferece opções para reenviar o e-mail com o token ou fornecer um novo token no totem, após verificação da identidade.
- e) Problemas técnicos: Em caso de problemas técnicos (como falhas de conexão ou sistema), o sistema exibe uma mensagem de erro e sugere tentar novamente mais tarde ou contatar o suporte técnico.

Fonte: Autoria própria.

Em seguida nas tabelas 2 e 3 em referência aos requisitos funcionais e não funcionais no sistema veremos a relação deles aos produtos apresentados e sua explicação da aplicação deles.

Tabela 2 – Explicação dos requisitos funcionais nas aplicações

Requisitos Funcionais		
RF1	Cadastro de Usuário	Permitir que os clientes se cadastrem no sistema do museu para acessar serviços como compra de ingressos e participação em pesquisas de satisfação e Avaliação das obras.
RF2	Login	Solicitar que o usuário faça o login para interagir com menu do visitante.
RF3	Menu	O usuário pode escolher uma tarefa através do menu.
RF4	Cadastro de Banco	Serão feitos os cadastros no banco.
RF5	Compra de ingresso	Permitir que o usuário selecione o tipo de ingresso desejado, como inteira, meia ou isento.
RF6	Avaliação das obras	Após interagir com as obras em exposição, o cliente tem a opção de participar de uma breve avaliação sobre elas. A participação na avaliação é voluntária.
RF7	Pesquisa de Satisfação	Os visitantes do museu serão convidados a participar de uma pesquisa de satisfação voluntária para expressar suas opiniões sobre sua experiência no museu como um todo.
RF8	Relatório Diário de Atividades do Museu	O sistema registra e gera relatório completo das atividades diárias, incluindo compra de ingressos, interações com obras e participação em pesquisas de satisfação

Fonte: Autoria própria.

Tabela 3 – Explicação dos requisitos não funcionais nas aplicações

Requisitos não funcionais			
Externo			
Segurança	RNF 1	Login Administrativo	Só poderão usar o sistema funcionários cadastrados
Segurança	RNF 2	Controle de Usuário	O sistema deve registrar todas as operações que o funcionário fez durante o login
Segurança	RNF 3	Nível de Usuário	O sistema deve possuir níveis de usuário por Padrão: Admin, Usuário.
Segurança	RNF 4	Cópia de segurança	Deverão ser feitos Backups de todos os dados armazenados/atualizados num período mensal
Produto			
Eficiência	RNF 5	Consultas simples ou com filtro	O sistema deve realizar consultas.
Eficiência	RNF 6	Relatórios de gerenciais e análise.	O sistema deve imprimir relatórios em no máximo os relatórios podem ser filtrados conforme as opções
Organizacional			
Implementação	RNF 7	Linguagem de programação	O sistema deve ser desenvolvido em C#
Implementação	RNF 8	Banco de dados	O banco utilizado deverá ser PostgreSQL
Implementação	RNF 9	Sistema	As máquinas deverão estar rodando Windows 8 32/64 Bits ou superior.
Implementação	RNF 10	Internet	A banda de transmissão deverá ser no mínimo de 2 Mbps para download e 700 Kbps para upload

Fonte: Autoria própria.

8.5.4. Custo dos equipamentos

A seguir será mostrado os valores dos equipamentos utilizados nos projetos para conhecimento, pois eles foram doados por outras organizações, pois logo então não teria nenhum custo com os equipamentos nesse projeto.

8.5.4.1. Cabeamento

A seguir será mostrado os valores dos equipamentos utilizados nos projetos para conhecimento, pois eles foram doados por outras organizações, pois logo então não teria nenhum custo com os equipamentos nesse projeto.

Em sequência mostraremos os cabos utilizados para estrutura de redes e seus preços apresentados na tabela 4.

Tabela 4 – Modelos e preços utilizados no cabeamento.

Produto	Modelo	Preços (R\$)
Cabo CAT5e	Furukawa 305M – 2x	1.400,00
Conector	RJ-45 Furukawa – 30Uni	50,00
Valor total Cabeamento:		1.450,00

Fonte: Autoria própria.

8.5.4.2. Switch

Switch utilizado para controle de cabos e para endereçamento deles e seus preços apresentados na tabela 5.

Tabela 5 – Modelo e preço utilizado no switch

Produto	Modelo	Preços (R\$)
Kit 2x Switch 16 Portas	Tp-Link TL-Sg1016d	1.200,00
Valor Total Switch		1.200,00

Fonte: Autoria própria.

8.5.4.3. Internet

Roteador utilizado para WIFI local e plano de internet contratado e seus preços apresentados na tabela 6.

Tabela 6 – Modelos e preços utilizados na internet

Produto	Modelo	Preços (R\$)
Roteador Acess Point	Intelbras AP 310	300,00
Internet Fibra Ótica	Claro Net 350Mb – M	100,00
Valor Total Internet		400,00

Fonte: Autoria própria.

8.5.4.4. Câmeras

Câmeras instaladas para segurança do local e das obras e seus preços apresentados na tabela 7.

Tabela 7 – Modelo e preço utilizado na segurança do local

Produto	Modelo	Preços (R\$)
Kit 4 Câmeras NVR	KaBuM! Smart	950,00
Valor Total Câmeras		950,00

Fonte: Autoria própria.

8.5.4.5. Computador para totens

Os totens do museu requeriam computadores no local para executar seus programas específicos, e esse modelo foi escolhido por sua capacidade robusta de atender aos requisitos, além de ser compacto, ocupando pouco espaço a seguir veremos seus valores apresentados na tabela 8.

Tabela 8 – Modelo e preço utilizado no computador

Produto	Modelo	Preços (R\$)
Kit 11x Mini PC	GMKTec Mini Gaming PC	6.719,02
Valor total computador:		6.719,02

Fonte: Autoria própria.

A configuração desse modelo seria um *GMKTec Mini Gaming PC* que contém Windows 11, memória de 16GB DDR4, armazenamento 512GB NVme SSD, WiFi 6, BT5.2 e um processador da 12th Gen, N100 da *intel*.

8.5.4.6. Estrutura dos Totens

Os totens precisavam de telas touchscreen para interagir com o sistema, além de suportes de chão para sua instalação a seguir veremos seus valores apresentados na tabela 9.

Tabela 9 – Modelos e preços utilizados no totem

Produto	Modelo	Preços (R\$)
11x Monitor Touch Screen	Monitor Dell Touch Screen de 24" USB-C P2424HT (10 toques simultâneos)	25.300
11x Suporte de chão para Monitor 24"	Mercado livre - Suporte	4.290
Valor total da estrutura dos totens:		29.590,00

Fonte: Autoria própria.

8.5.5. Topologia de rede

A configuração de rede é um termo utilizado para descrever a maneira como será organizada a infraestrutura de um sistema de computadores. Ao considerar a interconexão das máquinas, é importante pensar nos diferentes dispositivos utilizados, tais como Hubs e Switches.

O local oferece maior proteção, facilitando a troca de mensagens e arquivos de forma rápida.

8.5.5.1. Topologia em estrela

A estrutura em estrela é a disposição mais frequente. Neste caso, a rede tem sua operação centralizada em torno de uma conexão direta com um hub ou switch que atua como um servidor controlando as transmissões da rede.

8.5.5.2. Implementação da rede

Foi escolhido a topologia em estrela para ser utilizada neste projeto devido às suas características especiais. Esse arranjo oferece diversas vantagens que são especialmente adequadas para as demandas de um ambiente de exposição interativa como o nosso museu com múltiplas obras.

A resiliência é uma das principais vantagens da topologia em estrela. Mesmo que um dos *links* falhe, a rede continuará funcionando normalmente. Isso acontece devido ao fato de que cada dispositivo está conectado a um ponto central, como um switch ou um hub, responsável por controlar o fluxo de dados.

Se um cabo ou dispositivo em particular apresentar falhas, a rede manterá seu funcionamento regular. Tal ponto é de extrema importância em momentos de eventos, garantindo a continuidade do serviço para uma melhor experiência dos visitantes.

A configuração em estrela da rede também torna a manutenção e o crescimento do sistema mais fáceis. É viável adicionar ou remover dispositivos na rede sem causar impacto em seu desempenho como um todo.

Se precisar incluir um novo totem interativo ou retirar um equipamento provisoriamente para manutenção, é possível realizar essas alterações sem interromper a rede. Essa agilidade é essencial em ambientes como museus, que demandam flexibilidade e capacidade de se adaptar constantemente.

9. PRINCIPIOS DE INTERFACE UTILIZADOS

Dentro das aulas, aprendemos que a interface e as interações entre homem e computador trabalham juntas para facilitar o uso do sistema. Isso envolve a aplicação de métodos da engenharia de software, que se concentra na qualidade do sistema, e a expertise em Interação Humano-Computador (IHC), que se concentra nos aspectos da interação entre humanos e máquinas, ou seja, nos usuários.

O campo da IHC se tornou essencial para todos os profissionais de computação, pois trata do design, avaliação e implementação de sistemas interativos. Ele abrange todos os aspectos relacionados à interação entre usuários e computadores, buscando sempre melhorar essa interação. A seguir, são destacados alguns dos principais princípios de interface utilizados no desenvolvimento de sistemas interativos, como o totem interativo para o museu multitemático:

- a) **Usabilidade:** A experiência do usuário é essencial na área de Interação Humano-Computador. Ela diz respeito à facilidade de aprendizado do sistema, à sua eficiência no uso e à satisfação do usuário. No caso do totem interativo, foram aplicadas estratégias de design que promovem uma navegação intuitiva, com menus bem definidos e orientações simples de seguir.
- b) **Acessibilidade:** Assegurar que a plataforma seja acessível a todos os usuários, inclusive os que apresentam deficiências, é fundamental. Isso engloba a utilização de descrições alternativas para imagens, a adequação do contraste entre texto e fundo, e a introdução de ferramentas de navegação acessíveis a indivíduos com variadas limitações físicas ou sensoriais.
- c) **Feedback Imediato:** Na Interação Humano-Computador (IHC), é essencial oferecer retorno rápido sobre as ações dos usuários. No caso do totem interativo, essa interação é feita por meio de feedback visual que confirma quando uma ação foi realizada com êxito, como a escolha de uma opção ou o término de um questionário.
- d) **Consistência:** A coerência no design da interface assegura que os utilizadores consigam antecipar a forma como o sistema irá responder às suas ações, promovendo a eficiência e a confiança na utilização do sistema. Os componentes da interface no totem interativo seguem padrões consistentes em relação a cores, tipografia e disposição.

- e) **Prevenção de Erros:** Reduzir as chances de falhas é fundamental para garantir uma experiência positiva ao usuário. No totem, foram adotadas estratégias como a confirmação de ações importantes, a restrição de opções incorretas por meio de menus direcionados e a oferta de auxílio contextual para guiar os usuários.
- f) **Aprendizagem e Memorização:** A concepção do totem interativo considera a facilidade de aprendizado para novos usuários e a facilidade de memorização das funções mais comuns para usuários frequentes. Interfaces simplificadas e uma navegação lógica são elementos-chave para atingir essas metas.
- g) **Estética e Design Minimalista:** Um *layout* agradável à vista e organizado não apenas torna o sistema mais atraente, como também aprimora o entendimento e a facilidade de uso. No totem interativo, escolhemos um design simples e limpo que realça as informações principais e diminui elementos que possam causar distrações.
- h) **Engajamento e Motivação:** Para tornar a utilização mais prática, a tela do totem foi desenvolvida para atrair a atenção dos frequentadores do museu. Recursos interativos, como questionários e visualização de obras, são empregados para tornar a visita mais interativa e interessante, estimulando um maior envolvimento com a exposição.

Esses princípios de interface não apenas facilitam o uso do sistema, mas também aprimoram a experiência do usuário, tornando a interação mais eficiente, satisfatória e inclusiva. Ao aplicar esses princípios, o totem interativo do museu multitemático não só cumpre seu propósito informativo, mas também contribui para uma visita mais enriquecedora e memorável para todos os visitantes.

10. PROCESSO DE DESIGN EM IHC

No processo de criação da IHC, as normas ISO (Organização Internacional de Normalização) desempenham um papel crucial. Elas definem o processo e abordam a qualidade de uso e os requisitos ergonômicos, fornecendo padrões que ajudam a atender às necessidades do cliente.

10.1. Aspectos Fundamentais da Interação Humano-Computador

Para aplicar os conceitos de IHC no desenvolvimento do software para o totem do museu multitemático, foi priorizada a criação de uma interface intuitiva e fácil de usar. Utilizando técnicas de design interativo, buscamos garantir que os usuários possam navegar pelo sistema de forma fluida e eficiente, facilitando a compra de ingressos, avaliação das obras, participação em pesquisas de satisfação e quizzes.

Ao projetar a interface do software, consideramos os requisitos ergonômicos para garantir que seja confortável e acessível para os usuários, levando em conta aspectos como o tamanho e a resolução da tela do totem, a disposição dos elementos visuais e a legibilidade do texto.

Dessa forma, a aplicação prática dos princípios de IHC no desenvolvimento do software para o totem do museu visa proporcionar uma experiência de uso agradável e satisfatória para os visitantes, promovendo assim a usabilidade e eficácia do sistema.

10.2. Eficácia

Para implementar esses princípios no software do totem do museu multitemático, foi adotada uma abordagem centrada no usuário, com foco na simplicidade e facilidade de uso.

O design da interface foi cuidadosamente planejado para garantir que os usuários possam realizar suas atividades de forma eficaz, sem a necessidade de instruções complexas ou assistência externa. Além disso, foram incorporados elementos visuais consistentes e intuitivos, permitindo que os usuários identifiquem facilmente as funções disponíveis e aprendam a utilizá-las com rapidez.

10.3. Eficiência

No software do totem do museu multitemático, foram implementados mecanismos de feedback e correção de erros para garantir uma experiência de uso fluente e sem interrupções. Os usuários são guiados de forma intuitiva durante todo o processo de interação, e as mensagens de erro são apresentadas de maneira clara e objetiva, promovendo a facilidade de compreensão e resolução dos problemas.

Além disso, foram incorporadas funcionalidades de controle de ações, permitindo que os usuários desfaçam ou refaçam suas ações conforme necessário, garantindo assim uma maior eficiência na realização de suas tarefas.

10.4. Segurança

No contexto do software do totem do museu multitemático, foram implementadas medidas de segurança robustas para proteger os dados dos usuários e garantir a integridade do sistema. Foram adotados protocolos de criptografia avançados para proteger as informações confidenciais dos usuários durante as transações online, e foram implementados controles de acesso rigorosos para garantir que apenas usuários autorizados possam acessar determinadas funcionalidades do sistema.

Além disso, foram incorporados mecanismos de backup e recuperação de dados para garantir que as informações dos usuários estejam sempre seguras e disponíveis em caso de falha do sistema.

10.5. Capacidade de aprendizagem

No software do totem do museu multitemático, foi dada especial atenção à facilidade de aprendizado e adaptabilidade do sistema. Foram desenvolvidos tutoriais interativos e guias de uso para orientar os usuários na utilização das diferentes funcionalidades do software, permitindo que eles aprendam a interagir com o sistema de forma autônoma e sem dificuldades.

Além disso, foram incorporadas opções de personalização da interface, como a possibilidade de ajustar o tamanho do texto e selecionar preferências de exibição, para atender às diferentes necessidades e preferências dos usuários.

10.6. Capacidade de memorização

No software do totem do museu multitemático, foram implementadas técnicas de design que visam facilitar a memorização e a aprendizagem do sistema pelos usuários. Foram adotados padrões de rotulagem e organização consistentes em toda a interface, garantindo que os usuários possam facilmente encontrar e identificar as funcionalidades desejadas.

Além disso, foram incorporados recursos de ajuda e dicas contextuais para orientar os usuários durante a interação com o sistema, ajudando-os a lembrar das ações e procedimentos necessários para realizar suas tarefas com sucesso.

10.7. Satisfação

No software do totem do museu multitemático, foi dada especial atenção à experiência do usuário, visando garantir sua satisfação e engajamento com o sistema. Foram realizados testes de usabilidade e coleta de *feedback* dos usuários durante o processo de desenvolvimento, permitindo identificar e corrigir eventuais problemas e deficiências na interface e nas funcionalidades do software.

Além disso, foram incorporadas opções de personalização e configuração para atender às preferências individuais dos usuários, proporcionando uma experiência mais personalizada e satisfatória.

10.8. Ergonomia

No desenvolvimento do software para o totem do museu multitemático, foram considerados os princípios da ergonomia para garantir que a interação dos usuários com o sistema seja confortável, segura e eficiente. Foram realizadas análises detalhadas das necessidades e características dos usuários finais, levando em conta aspectos como postura, movimentação e conforto durante a utilização do totem.

Além disso, foram adotadas práticas de design que visam reduzir a fadiga e o desconforto físico dos usuários, garantindo uma experiência de uso mais ergonômica e agradável.

10.9. Acessibilidade

No desenvolvimento do software para o totem do museu multitemático, foi dada especial atenção à acessibilidade, visando garantir que o sistema possa ser utilizado por todos os visitantes, independentemente de suas habilidades ou características individuais. Foram adotadas diretrizes e práticas de design que tornam a interface do software mais acessível e utilizável para pessoas com deficiência, incluindo recursos como legendas para conteúdo sonoro, opções de contraste e tamanho de fonte ajustáveis, e suporte para dispositivos de entrada alternativos.

Além disso, foram realizados testes de usabilidade com uma variedade de usuários, incluindo pessoas com deficiência, para identificar e corrigir possíveis barreiras de acessibilidade, garantindo que o software seja verdadeiramente inclusivo e acessível a todos.

10.10. Acessibilidade no Totem do Museu Multitemático

Para garantir que o software do totem do museu multitemático seja verdadeiramente inclusivo, é essencial incorporar princípios de acessibilidade desde o estágio inicial de design. Considerando a diversidade de visitantes que podem interagir com o totem, incluindo aqueles com diferentes capacidades físicas, sensoriais e cognitivas, é crucial que a interface seja projetada para ser acessível a todos.

Isso significa que a interface do totem deve ser cuidadosamente planejada para permitir que usuários com diferentes habilidades e necessidades possam interagir de forma eficaz. Por exemplo, botões e controles devem ser grandes o suficiente e estar espaçados de forma adequada para facilitar o uso por pessoas com dificuldades motoras. Além disso, a interface deve oferecer suporte a tecnologias assistivas, como leitores de tela, para usuários com deficiência visual.

10.11. Eficácia e Eficiência na Interação

Além da acessibilidade, a eficácia e a eficiência da interação também são aspectos fundamentais a serem considerados no desenvolvimento do software do

totem. Isso significa projetar uma interface intuitiva e fácil de usar, que permita aos visitantes realizar suas tarefas de forma rápida e eficiente.

Por exemplo, ao apresentar informações sobre as exposições do museu, a interface do totem deve ser organizada de forma clara e lógica, auxiliando a navegação e a busca por conteúdo específico. Os elementos visuais, como ícones e botões, devem ser consistentes e familiares, para que os usuários possam entender facilmente sua função.

10.12. Segurança e Confiança

Além disso, a segurança é uma preocupação importante no design do software do totem. Isso inclui garantir que os usuários possam interagir com o sistema sem medo de cometer erros irreversíveis. Por exemplo, o software deve incluir mecanismos para impedir a execução de ações que possam causar danos ou comprometer a integridade do sistema.

10.13. Personalização e Adaptabilidade

Para melhor atender às preferências individuais dos visitantes, o software do totem também deve ser personalizável e adaptável. Isso significa permitir que os usuários personalizem a interface de acordo com suas preferências, como escolher o idioma de exibição ou ajustar o tamanho do texto.

10.14. Teoria da Ação na Prática

Integrar a teoria da ação de Norman ao desenvolvimento do software do totem pode ajudar a garantir uma interação mais fluida e natural. Por exemplo, ao projetar a sequência de ações necessárias para realizar uma busca por informações no totem, é importante considerar como os usuários percebem, interpretam e executam essas ações.

Ao integrar esses conceitos teóricos ao desenvolvimento do software do totem do museu multitemático, podemos criar uma interface acessível, eficaz, eficiente e segura para todos os visitantes. Isso resulta em uma experiência de usuário mais positiva e satisfatória, promovendo acessibilidade e inclusão no ambiente do museu.

11. MODELOS DE CICLO DE VIDA

Utilizando o que aprendemos em aula vimos que com o modelo de ciclo de vida que foca no desenvolvimento da interação com o usuário, teríamos quatro opções a utilizar:

- a) Modelo estrela;
- b) Design participativo;
- c) Engenharia de usabilidade;
- d) Projeto centrado no usuário.

Optamos por adotar o modelo de engenharia de usabilidade, que se destaca por criar interfaces fáceis de usar e que atendam às necessidades dos usuários. Esse modelo se baseia no princípio de centrar no usuário para desenvolver interfaces que sejam eficazes, eficientes e satisfatórias de usar.

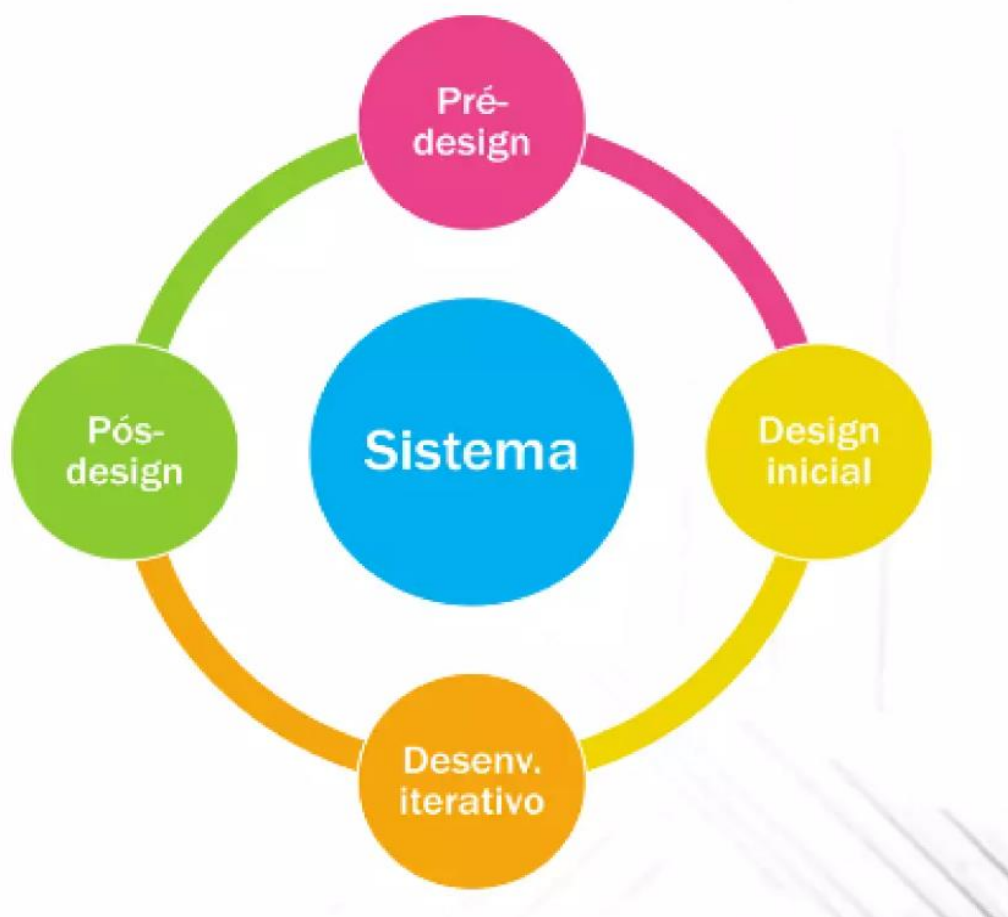
O processo de *design* para engenharia de usabilidade possui 4 fases que podemos ver a seguir os detalhes de cada uma e na figura 5 uma demonstração visual delas:

- a) Pré-design:** Busca de informação e conceituação sobre o usuário; Contexto de trabalho; Sistemas relacionados; Padrões de interface e ferramentas;
- b) Design Inicial:** Primeiro contato do usuário com o que poderá ser a vir o design do sistema. E a partir desse feedback que parte para as implementações no *design*;
- c) Desenvolvimento Iterativo:** Desenvolve a base de prototipagem e testes empíricos (experiências vividas);
- d) Pós-design:** Instalação do sistema no local de trabalho do usuário e acompanhamento com medidas de reação e aceitação do sistema pelo usuário final.

Ao adotar esse ciclo de vida baseado na engenharia de usabilidade, estamos comprometidos em criar um produto de alta qualidade que ofereça uma experiência de usuário excepcional para os visitantes do museu multitemático.

Nossa abordagem centrada no usuário garante que o totem seja intuitivo, fácil de usar e eficaz na entrega das informações e experiências desejadas pelos usuários.

Figura 5 – Fases da engenharia de usabilidade



Fonte: Autoria própria.

11.1. Interação Humano Computador (IHC)

A Interação Humano-Computador é uma área de estudo preocupada com o design, avaliação e implementação de sistemas computacionais interativos para uso humano. Segundo (PREECE et al., 1994), o termo foi cunhado na segunda metade da década de 80, como uma forma de descrever uma nova área de pesquisa que se concentra não apenas no projeto de interface, mas em todos os aspectos relacionados com a interação entre usuários e sistemas.

A Interação Humano-Computador (IHC) é uma disciplina voltada para o projeto, avaliação e implementação de sistemas computacionais interativos para uso humano e com o estudo de fenômenos importantes que os rodeiam (HEWETT et al., 1992).

Através de pesquisas e experimentações, a IHC visa melhorar a experiência geral de usar a tecnologia e torná-la mais acessível a todas as pessoas.

11.2. ISO 9241 (projeto de interface com o usuário)

Trata da ergonomia de software e da usabilidade de interfaces humano-computador. Essas normas fornecem diretrizes e princípios para o design e avaliação de interfaces de usuário, com o objetivo de melhorar a interação entre humanos e sistemas computacionais.

11.3. Orientações sobre Usabilidade

De acordo com Leite e Souza (1999) em seu artigo, quando um indivíduo interage visualmente (ou de forma mais ampla, sensorial) com a interface, ele emprega energia mental para interpretar e compreender o significado de todos os elementos e das informações transmitidas por eles.

Usabilidade refere-se à facilidade com que um sistema de computador pode ser utilizado por usuários específicos para alcançar seus objetivos de forma eficaz, eficiente e satisfatória, dentro de um contexto particular de uso. Para identificar a informação necessária a ser considerada na especificação de usabilidade de um computador em termos de medidas de desempenho e satisfação do usuário, ademais realizar uma análise detalhada das necessidades, expectativas e habilidades dos usuários finais.

Nesse sentido é importante considerar os requisitos ergonômicos para trabalho com telas de visualização, garantindo que o sistema seja projetado levando em conta aspectos como tamanho e resolução da tela, luminância, contraste, legibilidade de caracteres, entre outros. Podemos afirmar que, para proporcionar uma experiência agradável ao usuário, o software deve apresentar um visual robusto, com um layout e cores suaves que facilitem a compreensão do que está sendo feito. É essencial que o design não polua a imagem visual tanto do *layout* quanto do usuário, permitindo que ele execute as tarefas sem qualquer impedimento.

Ao considerar esses aspectos, podemos projetar sistemas de computador que ofereçam uma experiência de uso intuitiva, eficiente e satisfatória para os usuários, promovendo assim a usabilidade do software.

12. REQUISITOS PARA APRESENTAÇÃO VISUAL

A comunicação visual desempenha um papel fundamental na transmissão efetiva de informações. Podemos dizer que uma boa apresentação visual facilita a compreensão e a permanência das informações apresentadas. A capacidade de destacar-se visualmente e transmitir uma mensagem de forma clara e objetiva é crucial para o sucesso de qualquer comunicação.

Além disso, é importante destacar que os aspectos visuais não se limitam apenas a tornar o projeto apresentável. Eles estão diretamente relacionados à capacidade fisiológica e mental de como o usuário irá processar cores, texturas, luminosidade e formas. A maneira como interagimos com esses elementos é fundamental. Eles motivam o usuário a aprender e a sentir-se confortável, impactando diretamente na usabilidade.

12.1. Estrutura Visual

O estilo de uma interface, incluindo formas, fontes, cores e elementos gráficos, influencia significativamente a experiência do usuário. Quando uma interface é visualmente atraente, com gráficos bem elaborados, imagens e cores agradáveis, os usuários tendem a ser mais tolerantes em relação à usabilidade. No entanto, é importante encontrar um equilíbrio entre o design estético e a funcionalidade, pois a interface não apenas deve ser visualmente atraente, mas também deve ser capaz de guiar o usuário no uso correto do sistema.

Dessa forma, é possível combinar efetivamente essas duas ferramentas. Para uma boa estrutura visual é preciso projetar uma interface de modo a ser simples a onde o usuário consiga entender que tipo de atividade estão realizando enquanto estão interagindo com o sistema.

12.2. Fluxograma

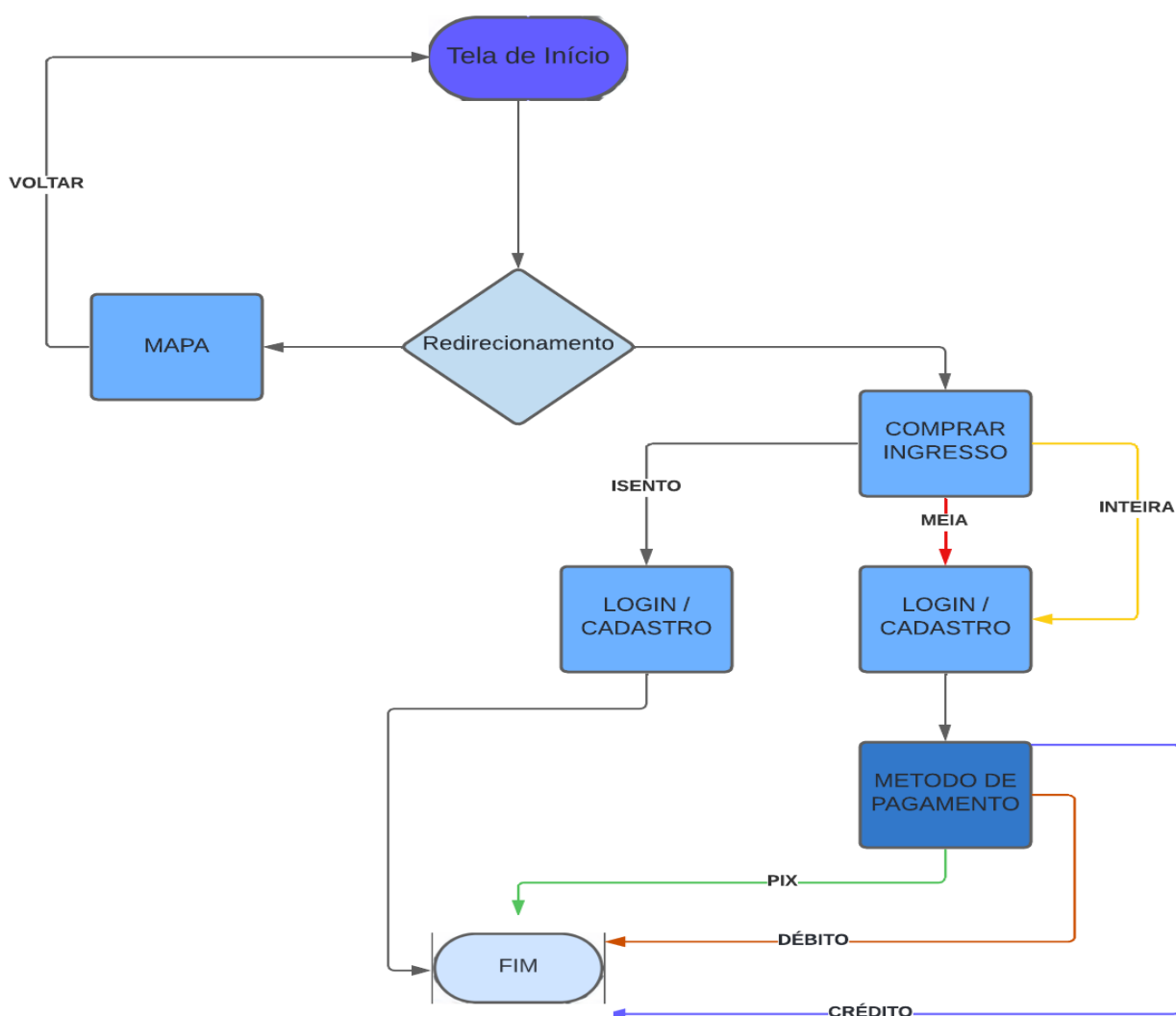
Fluxograma é uma segmentação hierárquica das tarefas do projeto em elementos menores, favorecendo assim o gerenciamento. Seu principal propósito é estruturar as atividades necessárias para alcançar os objetivos do projeto.

13. FLUXO DE FUNCIONAMENTO SISTEMA

Um fluxograma de funcionamento de um sistema é uma representação visual que mostra como o sistema opera, incluindo o fluxo de informações, dados ou processos dentro dele. É útil para entender a lógica do sistema, identificar falhas e documentar o funcionamento para referências futuras.

Pela entrada do museu teremos um totem para demonstrar nosso local, realizar compras e se registrar em nosso sistema seu funcionamento é demonstrado na figura 6.

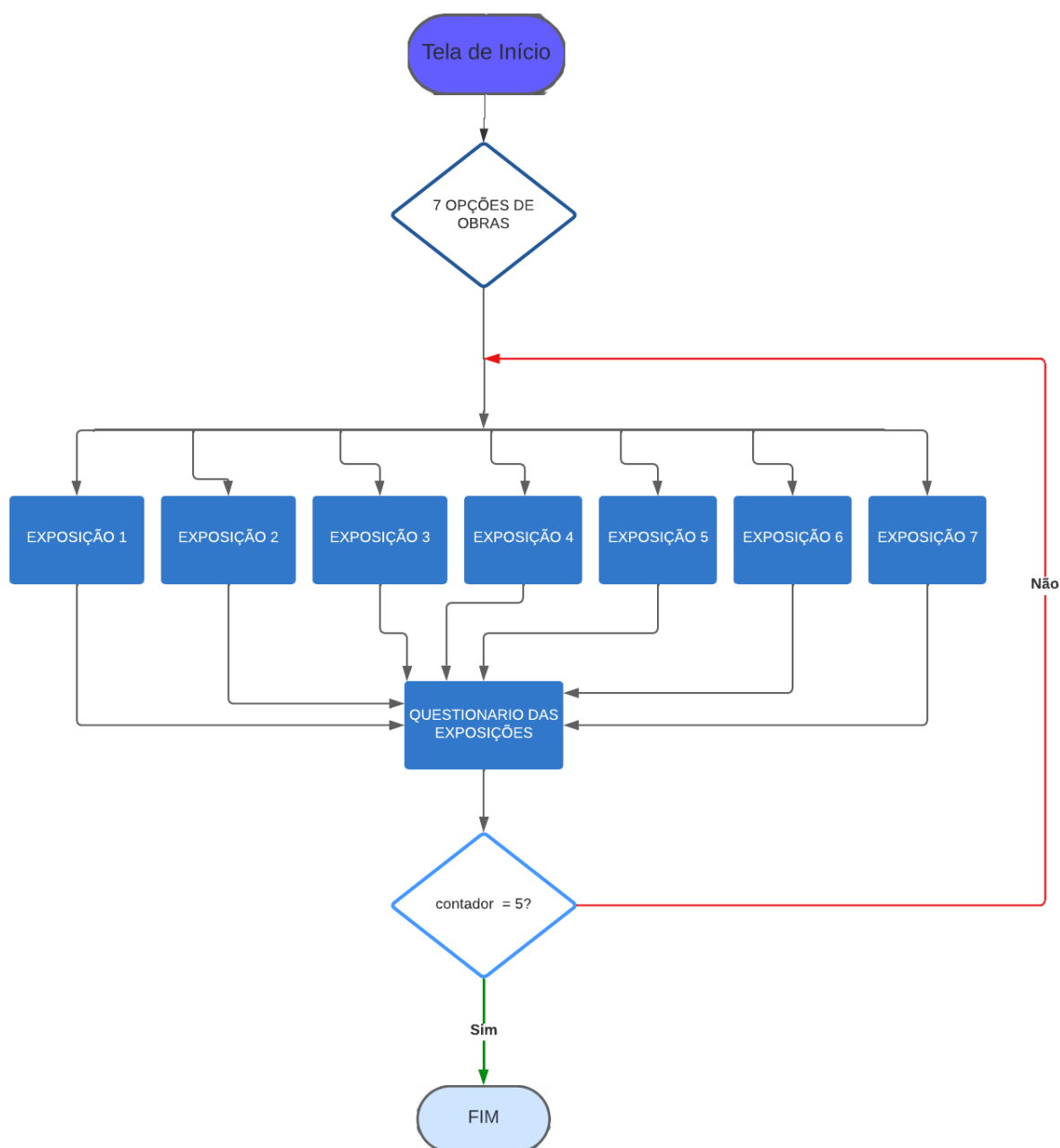
Figura 6 – Fluxograma do totem de entrada



Fonte: Autoria própria.

As obras terão seus próprios totens que serão usados para explicar sua história e incentivar os visitantes a participarem de um questionário de 5 perguntas, para ampliar. Seu conhecimento. Seu fluxo é demonstrado na figura 7 para entendimento.

Figura 7 – Fluxograma do totem das obras

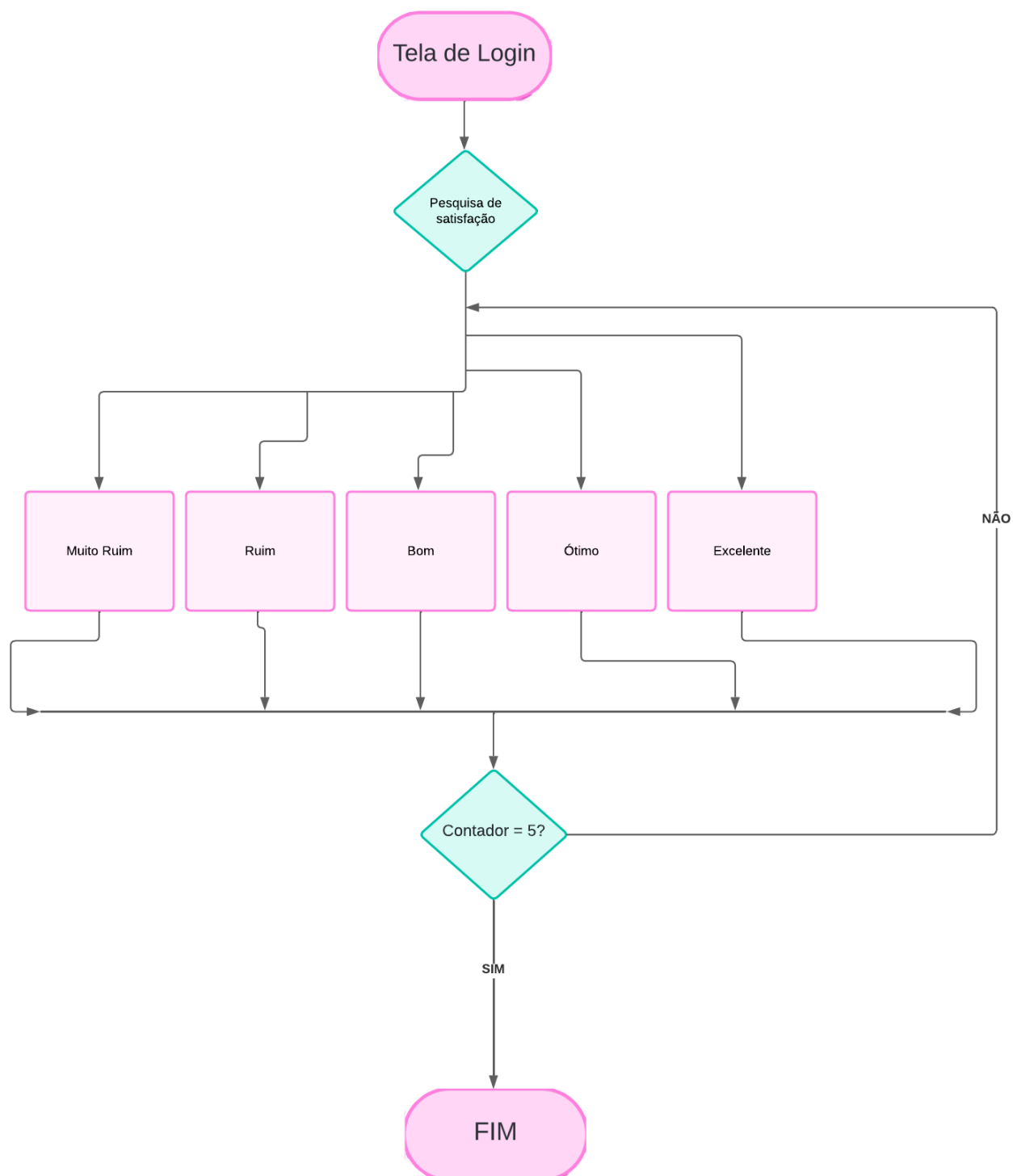


Fonte: Autoria própria.

Ao final da exposição, os visitantes poderão participar de uma pesquisa de satisfação no totem, fornecendo feedback para entendermos melhor o andamento da

exposição e para futuras melhorias no museu. Seu conhecimento. Seu fluxo é demonstrado na figura 8 para entendimento.

Figura 8 – Fluxograma do totem da pesquisa de satisfação



14. CRONOLOGIA DAS TELAS

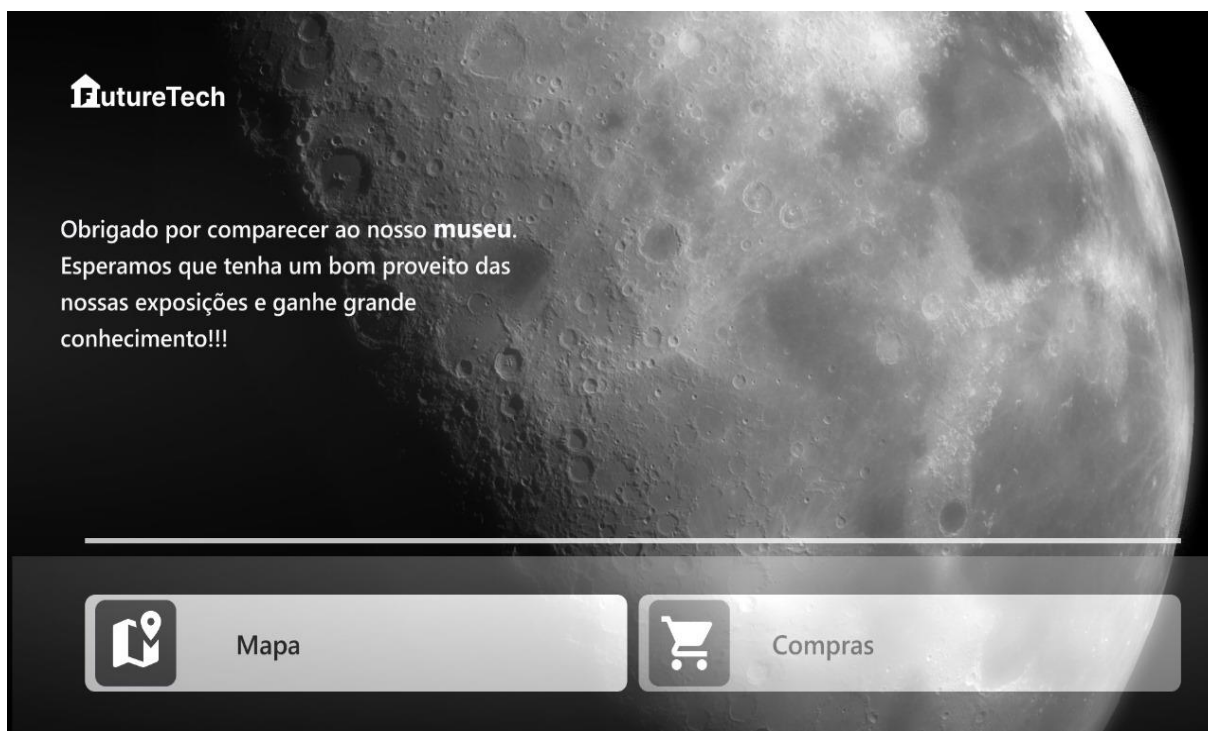
Uma cronologia de telas é uma visualização que mostra a ordem em que as telas ou páginas de um aplicativo, site ou software são acessadas, ilustrando como os usuários progridem de uma tela para a próxima.

14.1. Tela totem de entrada

Seria uma tela representativa como demonstrado na figura 9 que irá conter dois botões, sendo “mapa” e “compras”, que no primeiro botão “mapa” ele será redirecionado para uma visão geral do local que é mostrado na figura 10, que mostrara o que contêm no museu e como poderá prosseguir para conhecê-lo e se localizar.

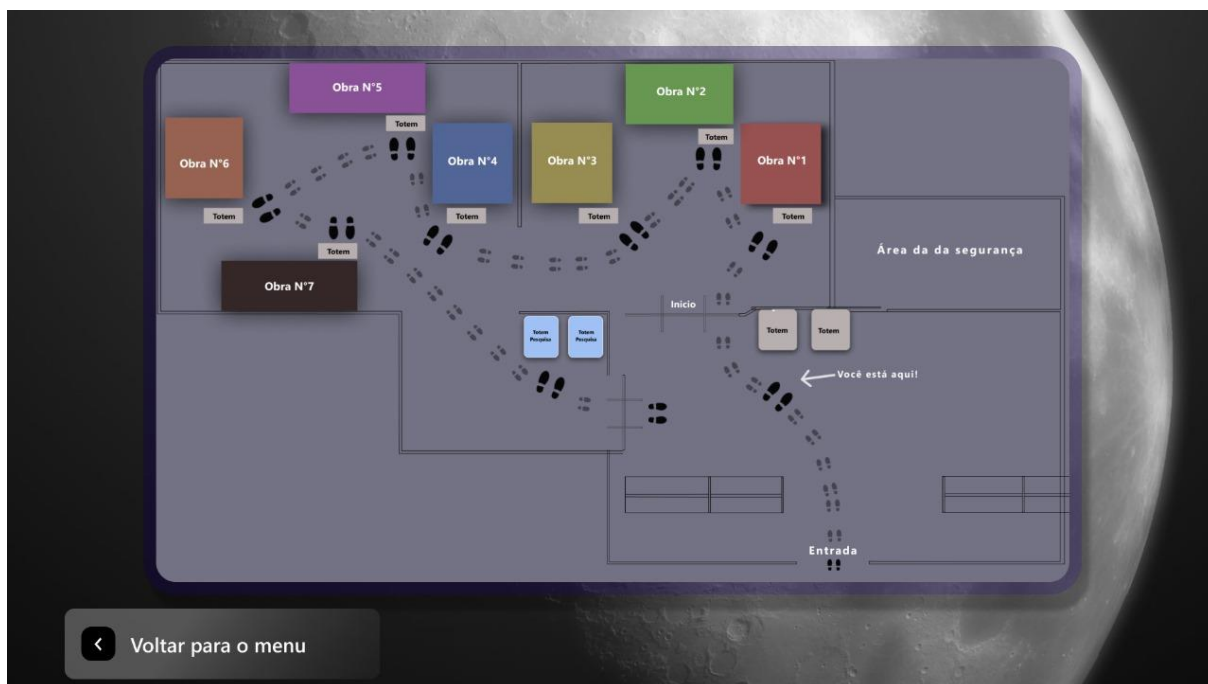
Já no botão “compras” o visitante poderá realizar a compra do ingresso para entrar no museu, mas primeiramente terá que realizar um cadastro básico no sistema ou login caso não seja a primeira vez visitando para realizar as pesquisas de satisfação e questionários no local. Como mostrada na tela inicial de “compras” na figura 11.

Figura 9 – Tela início totem de entrada



Fonte: Autoria própria.

Figura 10 – Tela mapa



Fonte: Autoria própria.

Na tela do mapa seria apenas para o visitante já conhecer um pouco do local e entender como funciona o estabelecimento e voltando para o menu principal clicando no botão “voltar para o menu” ele poderá realizar a compra do ingresso para entrar no local.

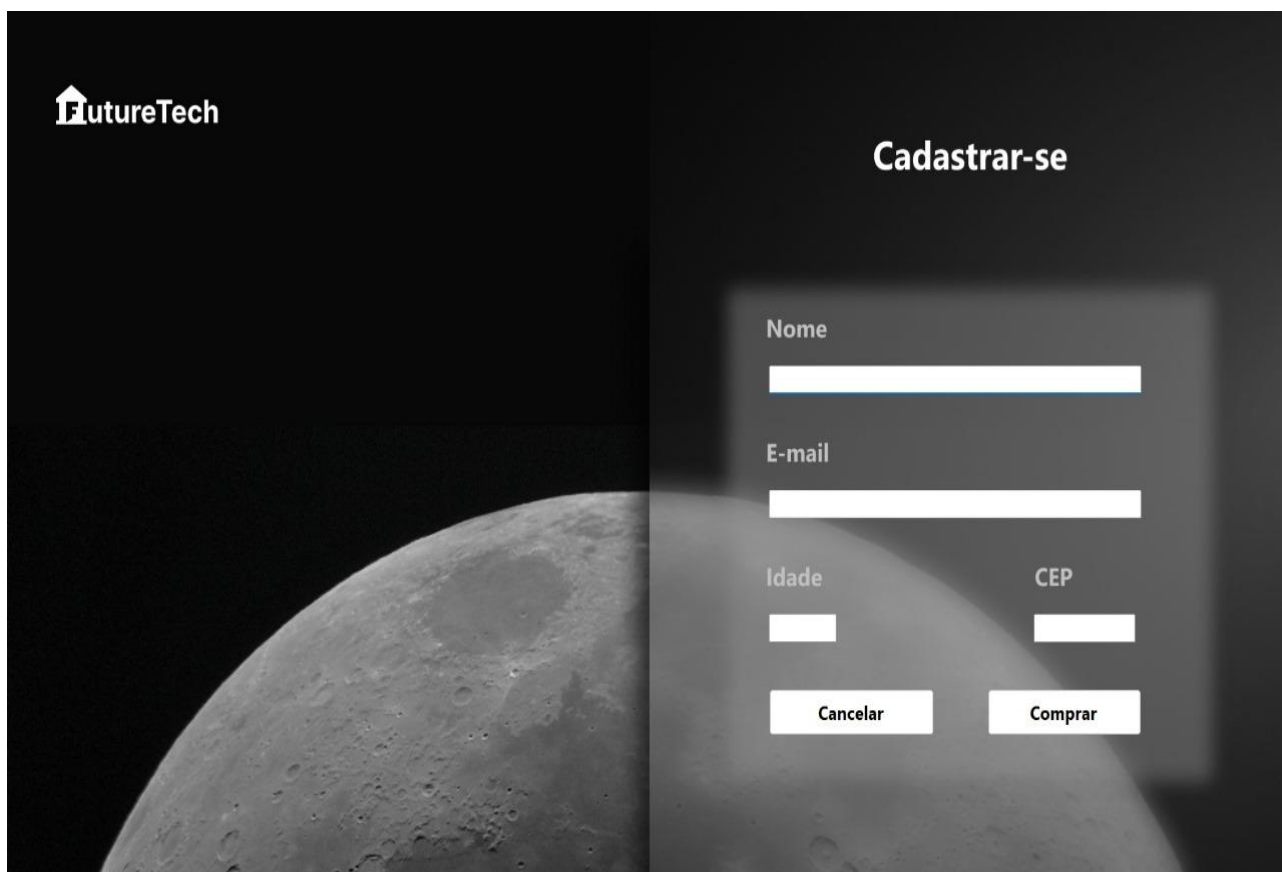
Figura 11 – Tela login



Fonte: Autoria própria.

Como foi comentado caso não tenha o cadastro no sistema será necessário o fazer no qual ao clicar em “cadastre-se” ele será redirecionado para tela de cadastro mostrado na figura 12.

Figura 12 – Tela cadastro



A interface de cadastro do FutureTech apresenta um formulário sobreposto a uma imagem de fundo que mostra a metade escura da Lua. O formulário, intitulado "Cadastrar-se", contém os seguintes campos e elementos:

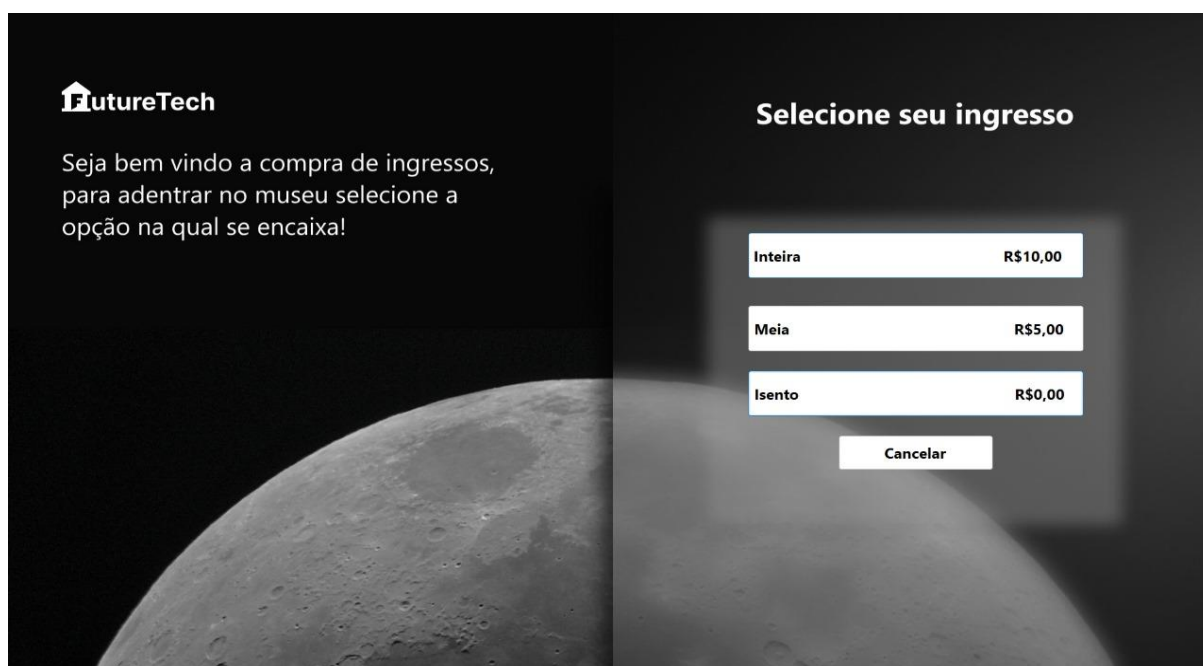
- Nome:** Um campo de texto único.
- E-mail:** Um campo de texto único.
- Idade:** Um campo de texto único.
- CEP:** Um campo de texto único.
- Botões:** "Cancelar" e "Comprar" localizados na base do formulário.

Fonte: Autoria própria.

Como seria a primeira visita do usuário a tela de cadastro desempenha um papel fundamental na intenção de registrar o visitante como um usuário no sistema, solicitando seu nome para identificação como perfil único, é solicitado e-mail para o visitante receber seu token de acesso as obras, a idade é requerida para garantir que o serviço seja adequado para à faixa etária correspondente do visitante e o CEP para requerer informações de localização dos visitantes para o museu.

Em resumo, a tela visa coletar informações essenciais de identificação que garante a adequação dos serviços, em seguida ele será redirecionado para a tela de escolha de ingresso, demonstrado na figura 13, para prosseguir com a compra e adentrar no museu.

Figura 13 – Escolhas de ingresso



Fonte: Autoria própria.

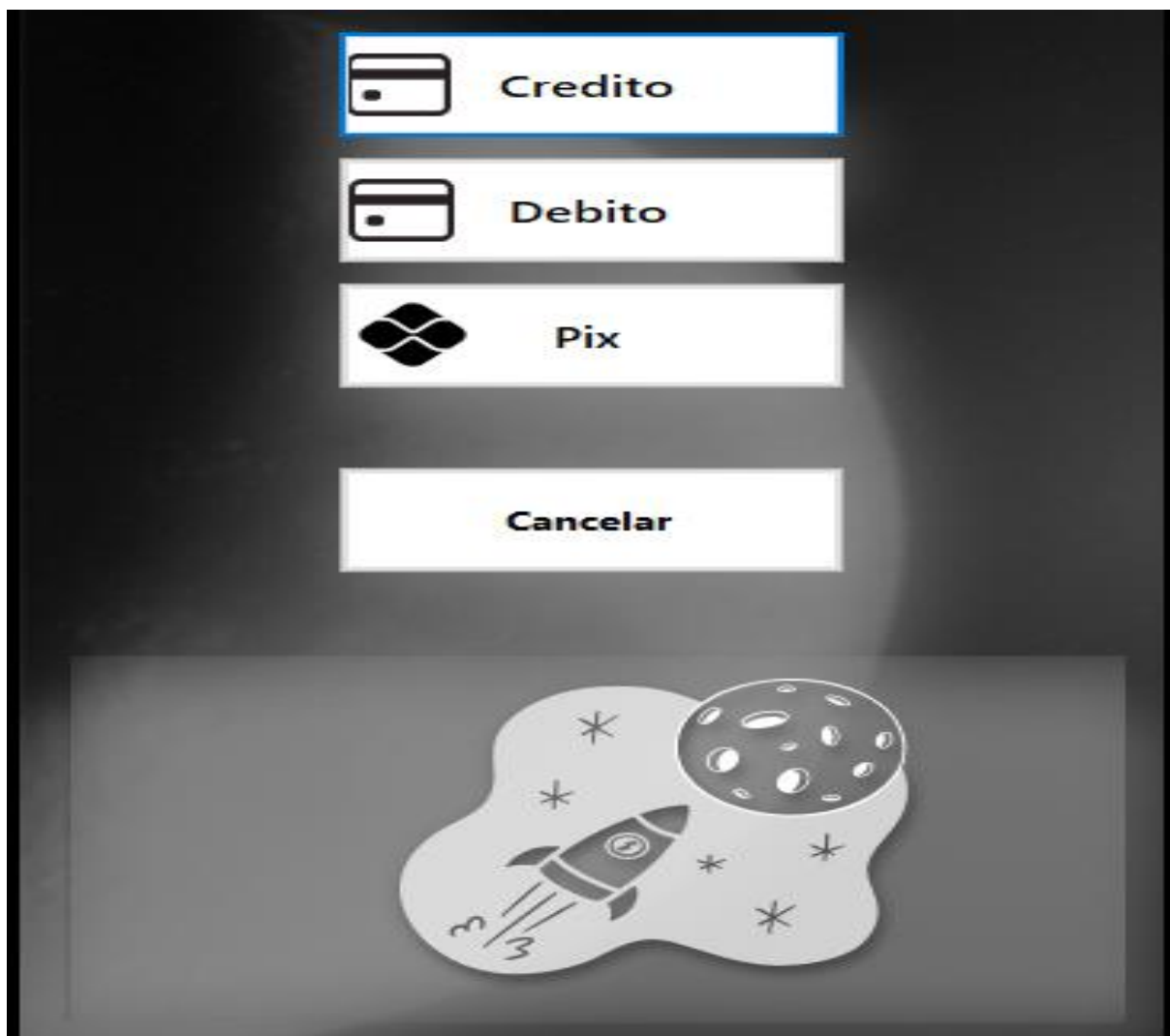
Esta tela oferece ao visitante a possibilidade de comprar o ingresso da forma que seja conveniente, proporcionando diferentes opções de compra que seja referente ao perfil do visitante.

A tela apresenta três opções de compras de ingressos, apresentando botões com os nomes de “Inteira”, “Isento” e “Meia” tendo cada botão correspondente a categoria específica de ingresso, permitindo que o visitante selecione a opção que adequa sua condição. Essa abordagem tem como objetivo atender às diversas necessidades e situações que os compradores desejam, garantindo melhor experiência a todos os que forem interessados pelo museu.

Após a escolha do ingresso, caso não for isento, o visitante é redirecionado a forma de pagamento, como mostrado na figura 14, sendo possível escolher uma de 3 opções:

- a) **Cartão de crédito:** Oferece disponibilidade de parcelamento, sendo possível maior flexibilidade financeira do usuário;
- b) **Cartão de débito:** Garante uma transação imediata e direta, sem necessidade de aguardar a fatura;
- c) **Pix:** Destaca-se na rapidez e praticidade na transação, permitindo que o pagamento seja efetuado de forma instantânea.

Figura 14 – Tela de compras



Fonte: Autoria própria.

Em resumo, as 3 formas de pagamento oferecem vantagens distintas, permitindo que o usuário escolha a melhor forma que o adequa, dependendo de suas necessidades.

14.2. Tela do totem das obras

Ao entrar no museu ele verá sete obras expostas, todas contendo seu totem específico e dentro de cada o visitante fará um login, demonstrado na figura 15, com seu token no qual foi gerado após a compra do ingresso e enviado a seu Email para conseguir acessar uma breve explicação sobre a obra em questão e um incentivo de aprender mais com um jogo de cinco perguntas no botão destacado “teste seu conhecimento”, como será mostrado na figura 16.

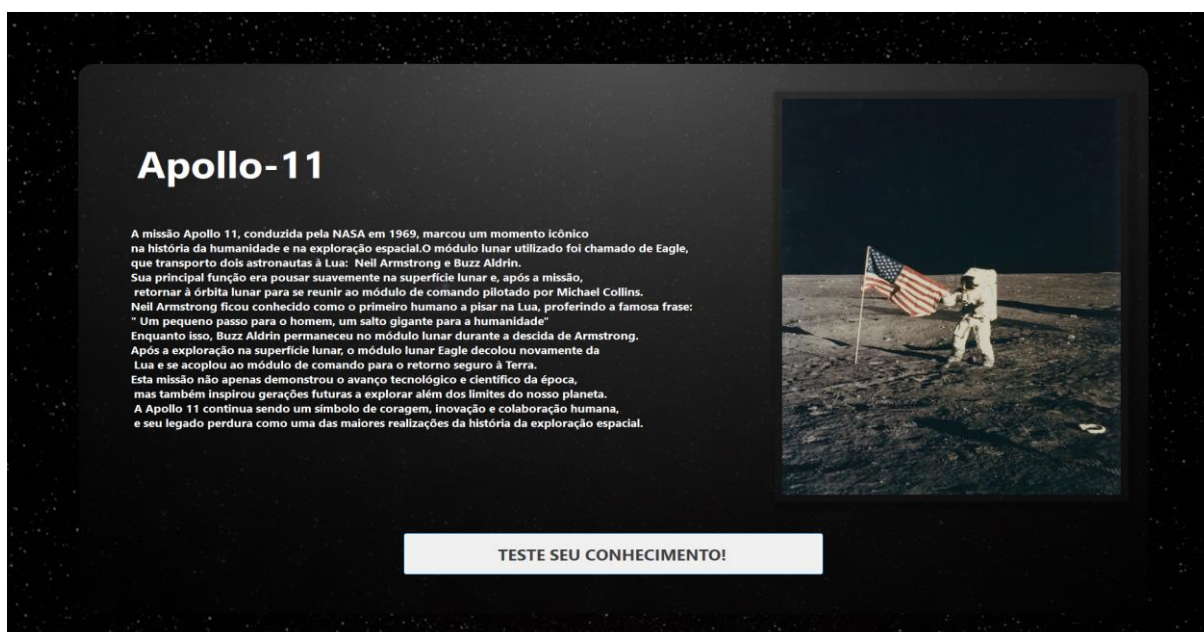
Na qual seria sobre a primeira obra e ela será usada como base para as outras que somente irá mudar nas outras o texto, pois seria outra obra e as perguntas também devido a isso.

Figura 15 – Tela de login com token



Fonte: Autoria própria.

Figura 16 – Tela das obras



Fonte: Autoria própria.

A tela de obras possui a finalidade de introduzir o visitante ao conteúdo principal da exposição, as obras que o farão entender sobre o tema. A tela consiste em dois elementos principais, o título da obra que o totem apresenta e um texto explicativo contando sobre a obra, contextualizando o visitante ao tema abordado.

Além do título e texto explicativo, a tela apresenta um botão que redireciona o visitante para a próxima etapa da experiência, o questionário relacionado a obra, que será mostrado como base somente a primeira pergunta, dentro de cinco, na figura 17.

Figura 17 – Tela das obras

Questionário sobre:

O Módulo Lunar da Apollo 11

Pergunta 1

Qual era o nome do módulo lunar usado na missão Apollo 11?

- A) Eagle
- B) Falcon
- C) Hawk
- D) Sparrow
- E) Osprey

Fonte: Autoria própria.

A tela do questionário apresenta o título referente a obra que o antecede, permitindo uma interação direta sobre o tema. A tela do questionário é composta por 3 elementos, o título, as perguntas e as alternativas.

- a) **O Título:** Coincide com a obra referente a obra;
- b) **As Perguntas:** Aspectos que foram retiradas do texto explicativo em forma de perguntas;
- c) **As Alternativas:** 5 botões referentes as perguntas, sendo apenas 1 com a alternativa correta.

As perguntas formuladas visam explorar os assuntos que foram referenciados nas obras, fornecendo o leitor a oportunidade de refletir e responder de maneira direta, auxiliando igualmente aos envolvidos pela disponibilização do museu, retornando um feedback sobre as obras.

Ao final das 5 perguntas do questionário, o visitante é recebido com uma mensagem informando quantas alternativas estavam certas, contendo as que foram respondidas corretamente, demonstrado na figura 18 e a porcentagem de acertos e média de acertos gerais para a obra. O feedback contribui para o engajamento do leitor, incentivando a refletir sobre os resultados, proporcionando a oportunidade de entender mais profundamente sobre a obra em questão.

Figura 18 – Resultado questionário



Fonte: Autoria própria.

Para administração de tudo e controle do museu em si, foi feito uma tela de ADM, demonstrada na figura 19, para geração de relatórios de pesquisa de satisfação e pessoas que visitaram o lugar e contém Três botões “Inteira”, “Isenta” e “Meia”, que caso ocorra algum erro na compra de ingresso o ADM possa gerar para o visitante conseguir entrar no museu.

O sistema de administração de visitantes é desenvolvido para proporcionar aos administradores melhor gerenciamento capaz de editar as informações dos visitantes de forma ágil e organizada. O Sistema consta com campos que devem ser preenchidos de nome, e-mail, idade, data de nascimento e o código do visitante. Após o preenchimento dos dados, o administrador pode clicar no botão de salvar para cadastrar um novo visitante.

Além disso, o sistema também consta com campos de preenchimentos que possibilita a edição das informações do visitante, podendo o administrador atualizar ou corrigir dados conforme necessário, sem ter que excluir cadastros de usuários e refazer a criação de visitantes.

Figura 19 – Administrativo

The screenshot displays an administrative interface with a table of visitors and a form for adding or editing a visitor.

Código	Nome	E-mail	CEP	Idade	Data
P891B	Paulo	mendespaulo35@gmail.com	18020290	29	23/04/2024
R91G9	Rebeca	rebeca@gmail.com	18020290	23	23/04/2024
B91G9	Bruno	bruno@gmail.com	18020290	21	23/04/2024
A71P7	Ayumi	naruse.ayumi2013@gmail.com	18116901	24	25/04/2024
H57C5	Hugo	hugo@gmail.com	18020290	21	26/04/2024
J40C4	Nicolas	kenzonicolas8@gmail.com	06700190	22	04/05/2024

On the right side, there is a form for adding or editing a visitor. It includes fields for Name, E-mail, Age, and Date, along with buttons for 'Remover' (Remove) and 'Salvar' (Save). Below the form, there is a section for 'Editar' (Edit) with fields for Name and E-mail, and a 'Buscar' (Search) button. At the bottom, there are buttons for 'Inteira' (Full), 'Meia' (Half), and 'Isento' (Exempt), and a 'Relatorios' (Reports) button.

Fonte: Autoria própria.

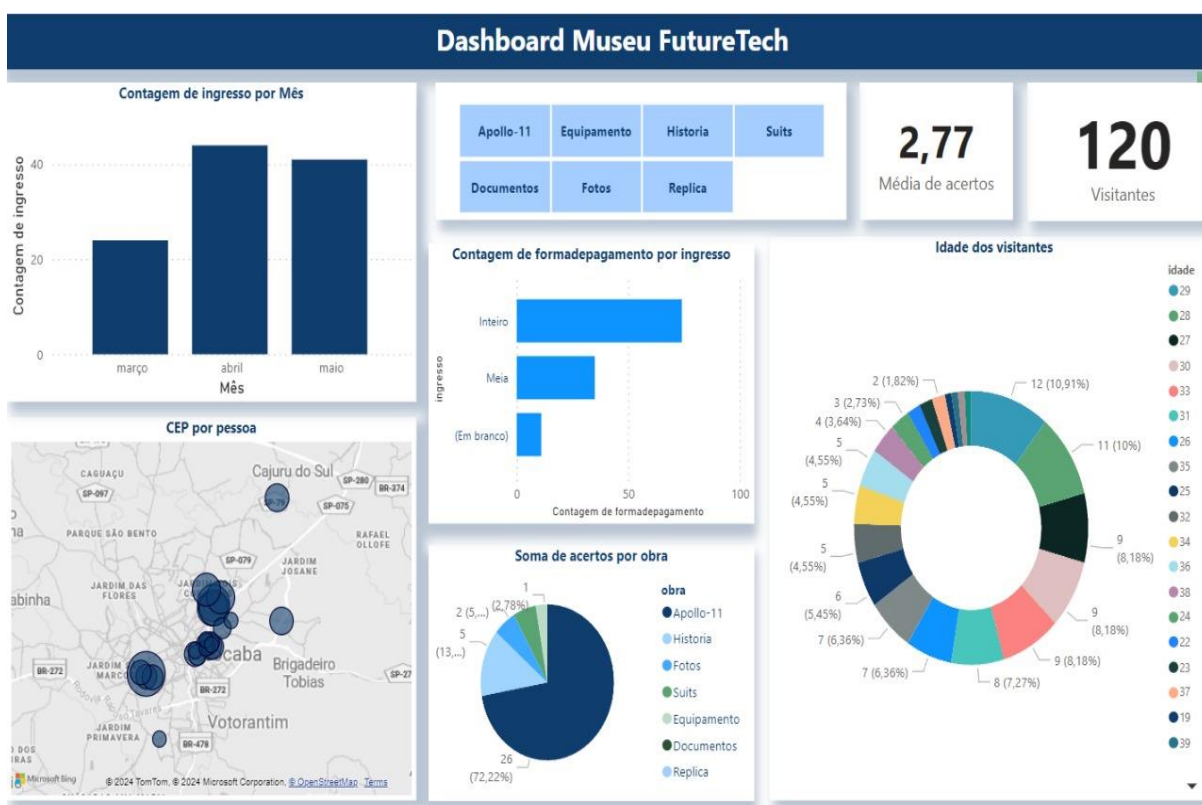
Dentro desta tela podemos ver o botão “Relatório” que com ele será redirecionado para o aplicativo Power BI (*Power Business Intelligence*) e será gerado uma tabela para melhor controle.

14.3. POWER BI

O Power BI é uma ferramenta de análise de negócios da Microsoft que permite transformar dados em informações visualmente impactantes. Ele ajuda a criar painéis e relatórios interativos com visualizações de dados dinâmicas e poderosas.

Com esta ferramenta a utilizamos para gerar uma tabela, demonstrada na figura 20, que assim conseguiríamos visualizar a quantidade de vendas de nossos ingressos e a satisfação de nossos clientes para também termos uma noção de como estamos e se podemos ter melhoria futuras no museu e no sistema.

Figura 20 – Relatório



Fonte: Autoria própria.

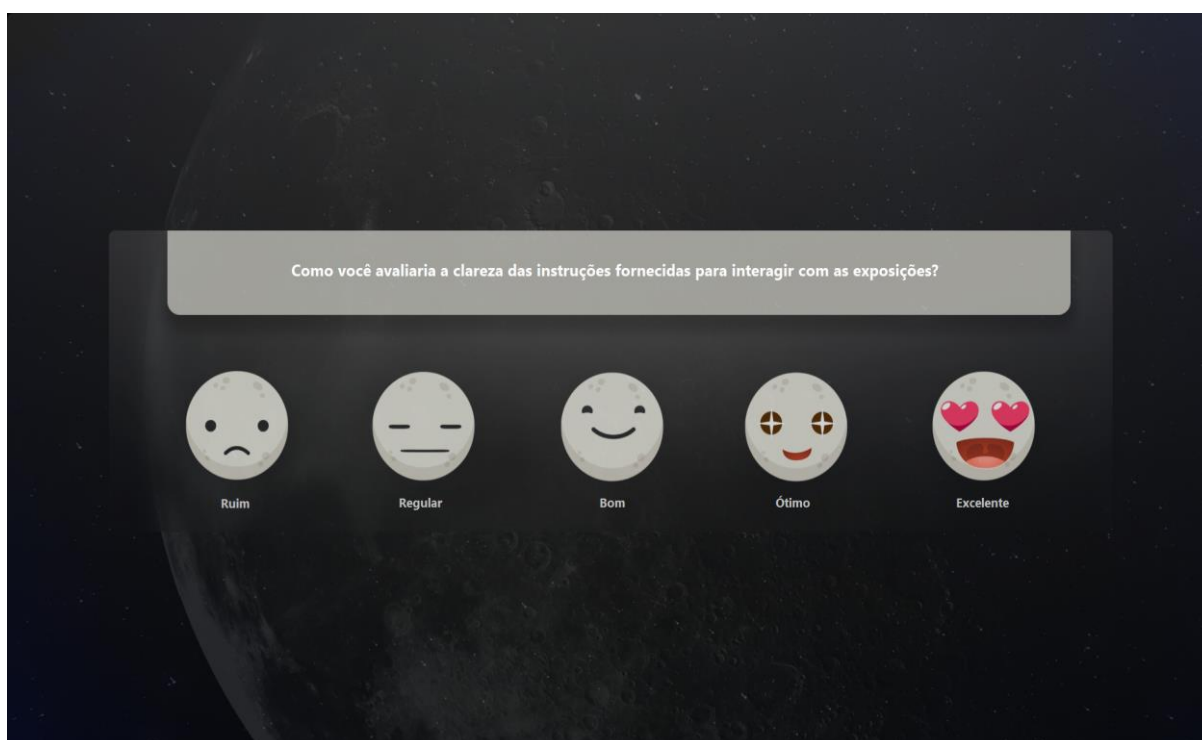
14.4. Pesquisa de satisfação

Realizar uma pesquisa de satisfação é essencial para avaliar a percepção dos clientes em relação ao evento, visando aprimorar constantemente o atendimento em todos os canais de comunicação.

Além disso, é fundamental para avaliar a qualidade dos produtos e serviços, bem como identificar possíveis melhorias nos processos. Abaixo apresentamos a nossa pesquisa de satisfação a fim de entender a opinião dos nossos participantes em relação ao evento. Após entender o que seria a pesquisa de satisfação do usuário, a fizemos para ter como base para melhoria de nosso sistema e do museu.

Após chegar ao totem de avaliação será pedido ao visitante fazer login com seu token para sabermos de sua opinião sobre o evento em si, a tela de login será a mesma já demonstrada anteriormente na figura 14. Em seguida será redirecionado para tela de pesquisa de satisfação mostrada na figura 21 a seguir.

Figura 21 – Pesquisa de Satisfação



Fonte: Autoria própria.

A tela de avaliação mostrada foi desenvolvida para o museu abordando ao visitante perguntas sobre a experiência do mesmo com o museu, proporcionando para os envolvidos da administração do projeto um feedback ao final da visita dos participantes.

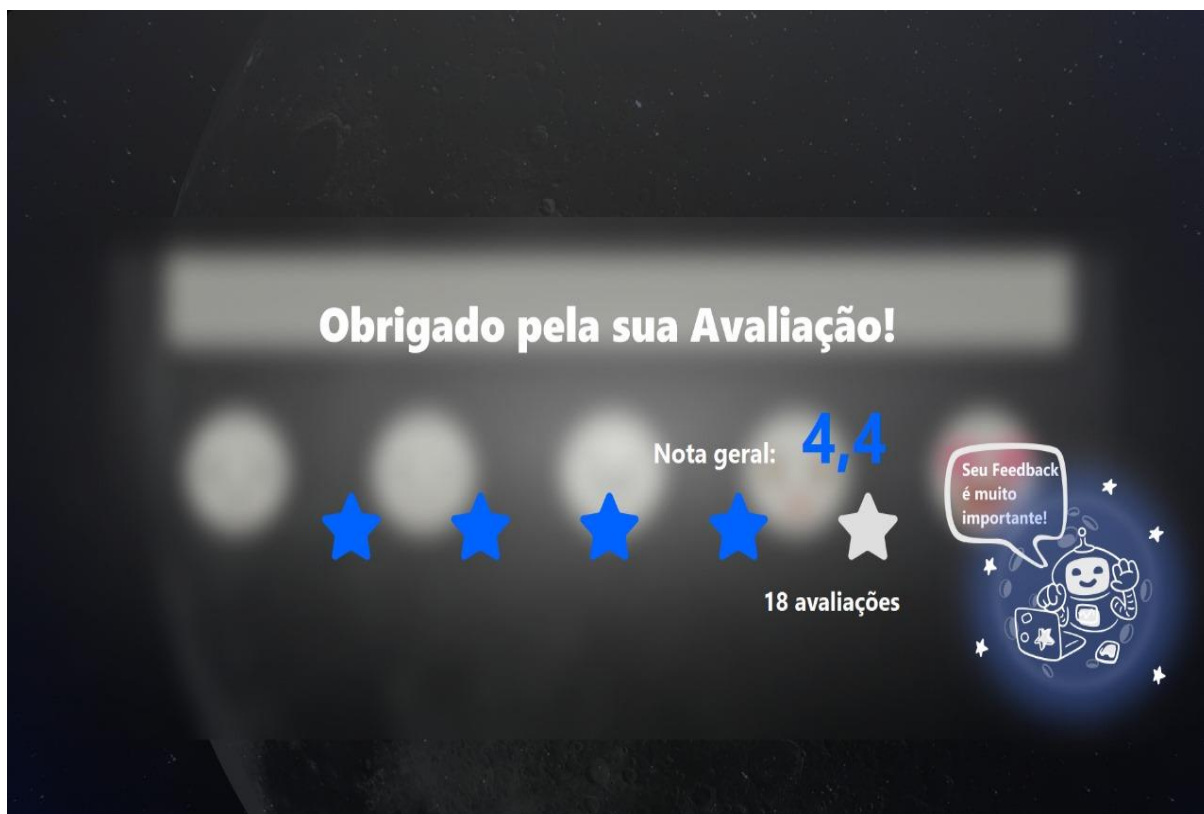
A tela exibe uma pergunta relacionada a experiência do usuário solicitando que o visitante clique em uma das 5 imagens de carinhas com expressões referenciando sua vivência.

As carinhas representam uma escala que vai de ruim a excelente, permitindo que a expressão dos visitantes seja representada de forma rápida e intuitiva.

Sendo assim, a tela de avaliação do museu é representada com uma estratégia eficaz para melhor obtenção dos feedbacks dos visitantes, permitindo análise das experiências, possibilitando ajustes e melhorias contínuas com base nas críticas recebidas dos visitantes do museu.

Após responder todas as perguntas de satisfação, o usuário recebe uma mensagem de agradecimento por ter mandado um feedback sobre as suas experiências, assim como mostra a figura 22.

Figura 22 – Feedback



Fonte: Autoria própria.

14.5. Banco de dados Postgree

O *PostgreSQL* tem o papel de gerenciar os dados desses bancos de maneira organizada e eficaz, rodando e gravando todas as informações que ficam registradas nesses compartimentos. Por meio desse sistema, usuários podem executar consultas de maneira simples, sem precisar acessar diretamente o banco de dados.

O *PostgreSQL* desempenha um papel fundamental no sistema, registrando os cadastros dos visitantes e armazenando suas informações não sensíveis. Além disso, ele se conecta com uma tabela no *POWERBI* para a geração de relatórios. Essa integração permite que o museu analise o desempenho do sistema e identifique oportunidades de melhoria, tanto para a exposição quanto para a experiência dos visitantes.

A seguir na figura 23 mostraremos as tabelas criadas no banco de dados *PostgreSQL* para visualização e curiosidade.

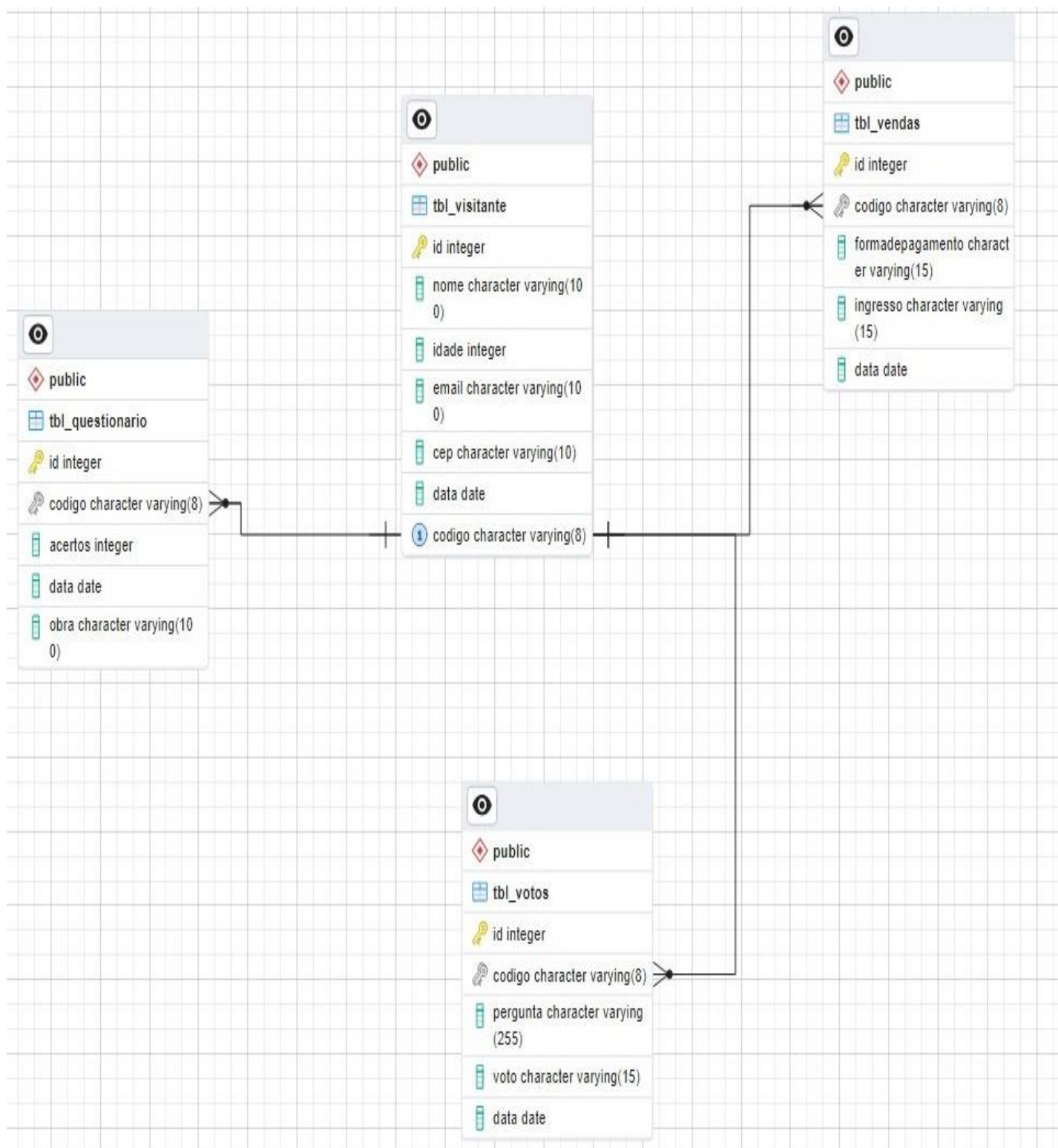
Figura 23 – Pastas no PostgreSQL



Fonte: Autoria própria.

Um exemplo na figura 24 dentro do próprio *PostgreSQL* a comunicação entre tabelas e sua visão em lógico.

Figura 24 – Lógico dentro do PostgreSQL



Fonte: Autoria própria.

15. MODELAGEM DE DADOS COM MER E DER

De acordo com Rob e Coronel (2006), a modelagem conceitual é uma fase muito importante no planejamento de uma aplicação de um banco de dados bem-sucedida. Sendo assim, a utilização do Modelos Entidade-Relacionamento (MER) e Diagramas Entidade-Relacionamento (DER) foi um passo crucial para as definições dos planejamentos de como seriam as funcionalidades do banco de dados.

O museu em questão coleta dados dos visitantes para cadastros, registra vendas de ingressos e informações sobre questionários respondidos pelos visitantes.

15.1. Modelo Entidade-Relacionamento (MER)

Um modelo Entidade-Relacionamento é amplamente utilizado no projeto de aplicações de banco de dados. Esse modelo, juntamente com suas variações, é comumente empregado no projeto conceitual de aplicações de banco de dados, sendo que muitas ferramentas de projeto de banco de dados aplicam seus conceitos. (ROB; CORONEL, 2006, p.35).

No contexto do museu multitemático, as entidades principais são os visitantes, ingressos e respostas e votos, cada uma com seus atributos específicos. Os relacionamentos entre as entidades são essenciais para entender como esses elementos estão conectados e interagem no sistema.

Como visto a seguir na figura 25, um visitante pode comprar vários ingressos, o que é representado por um relacionamento de “1 para muitos” entre entidades de visitante e ingressos.

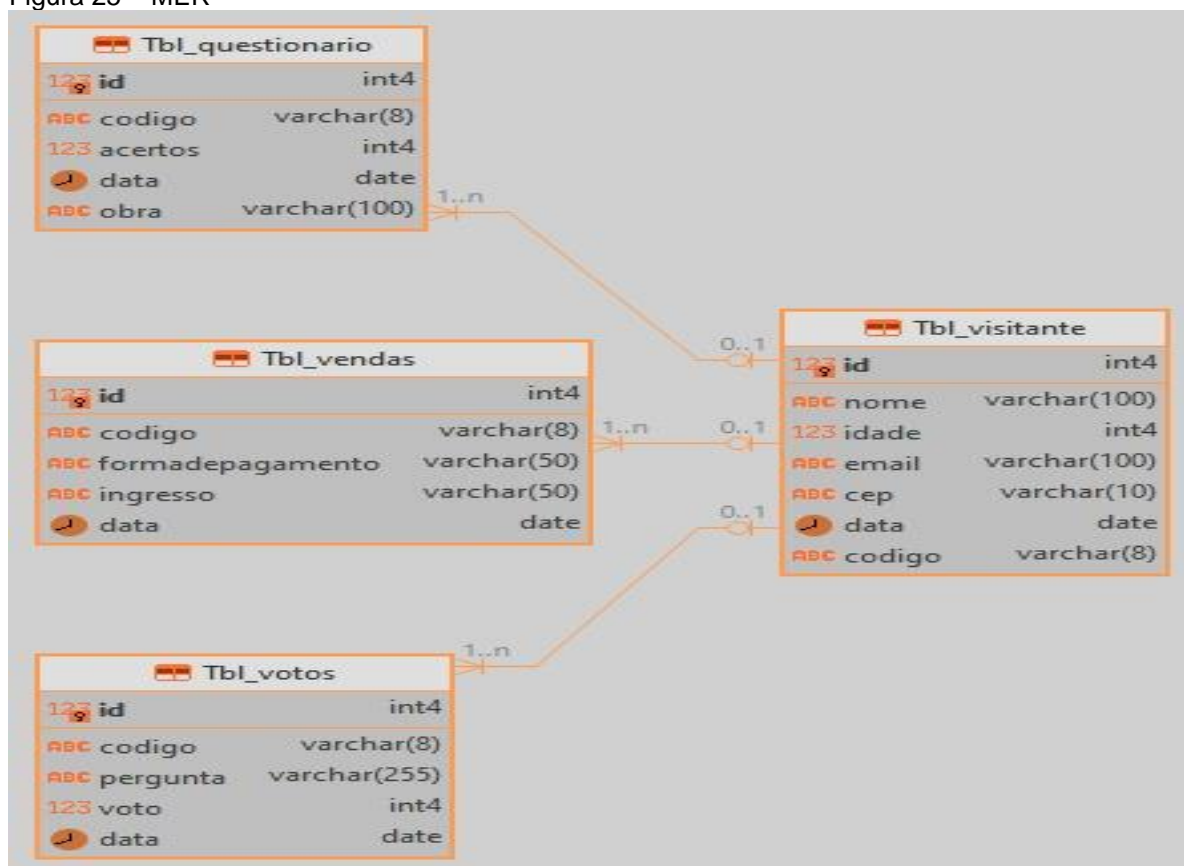
15.2. Diagrama Entidade-Relacionamento (DER)

Heuser (1998) afirma em seu artigo que o diagrama entidade-relacionamento é a representação gráfica do que foi representado no modelo entidade-relacionamento, favorecendo uma visualização e compreensão do banco de dados. Ou seja, o DER descreve as entidades, atributos e relacionamentos do sistema de forma mais detalhada e simples. No caso do presente trabalho, o DER é apresentado na figura 26, onde as entidades são representações dos visitantes, vendas, perguntas de satisfação e as perguntas acertadas do questionário.

- a) **Entidade Visitante:** Possui atributos como Código de identificação, nome, e-mail, idade, CEP e data que fez seu pedido;
- b) **Entidade de Vendas:** É referenciado o código da entidade visitante para saber quais foram as opções do pagamento de seu ingresso, formas de pagamento (inteira, meia ou isenta) e a data que foi realizado;
- c) **Entidade Votos:** A entidade que representa as respostas da pesquisa de satisfação, referenciando o código do visitante, mostrando a pergunta que o visitante respondeu e sua respectiva resposta;
- d) **Entidade Respostas:** Representa a quantidade de respostas o visitante conseguiu responder com assertividade do questionário, referenciando o código da entidade visitante.

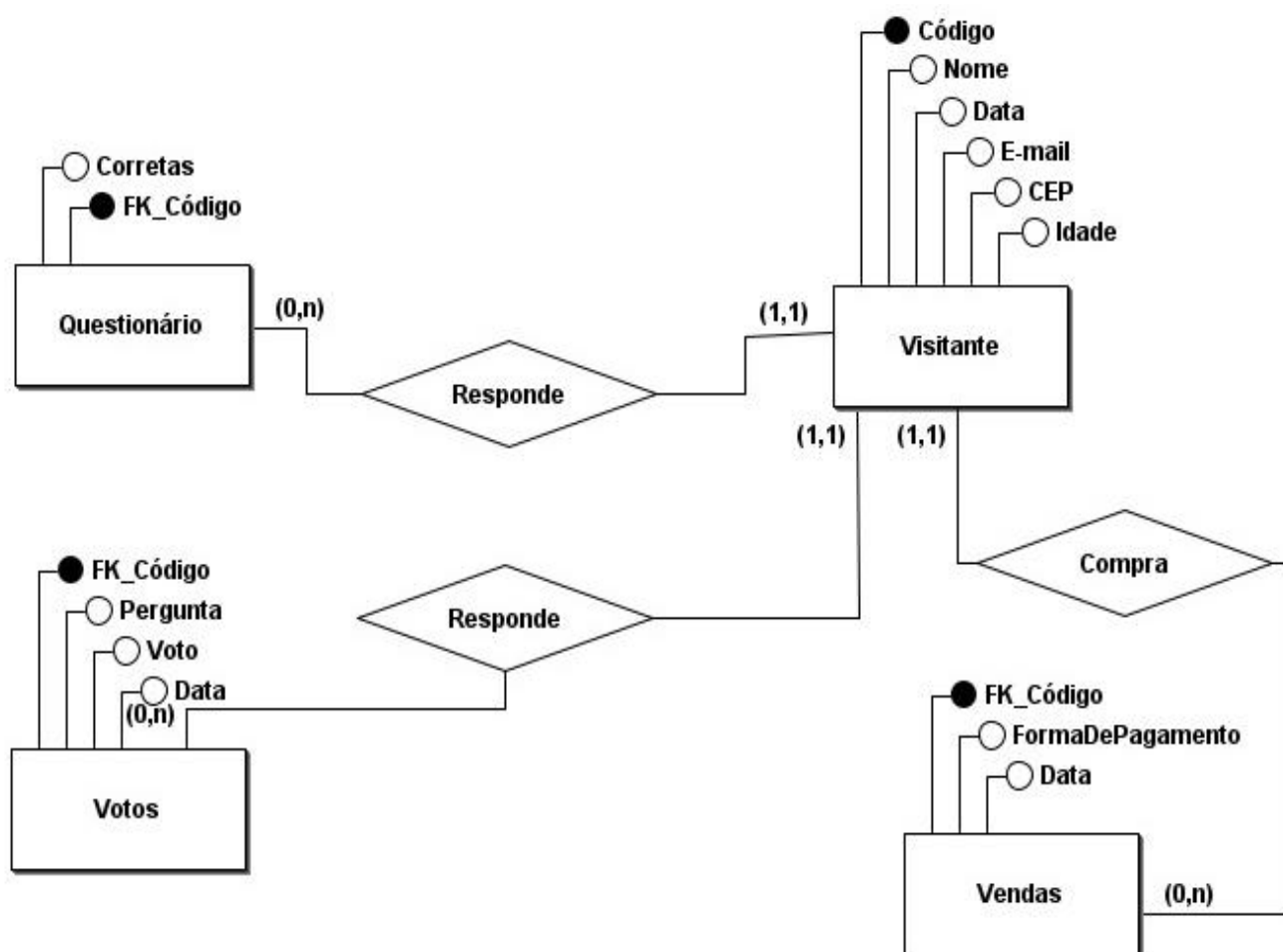
O DER, segundo Rob e Coronel (2006), é útil em projetos de banco de dados porque um esquema de banco de dados raramente é alterado, enquanto os conteúdos dos conjuntos de entidades frequentemente se alteram.

Figura 25 – MER



Fonte: Autoria própria.

Figura 26 – DER



Fonte: Autoria própria.

15.3. Diagrama de classe

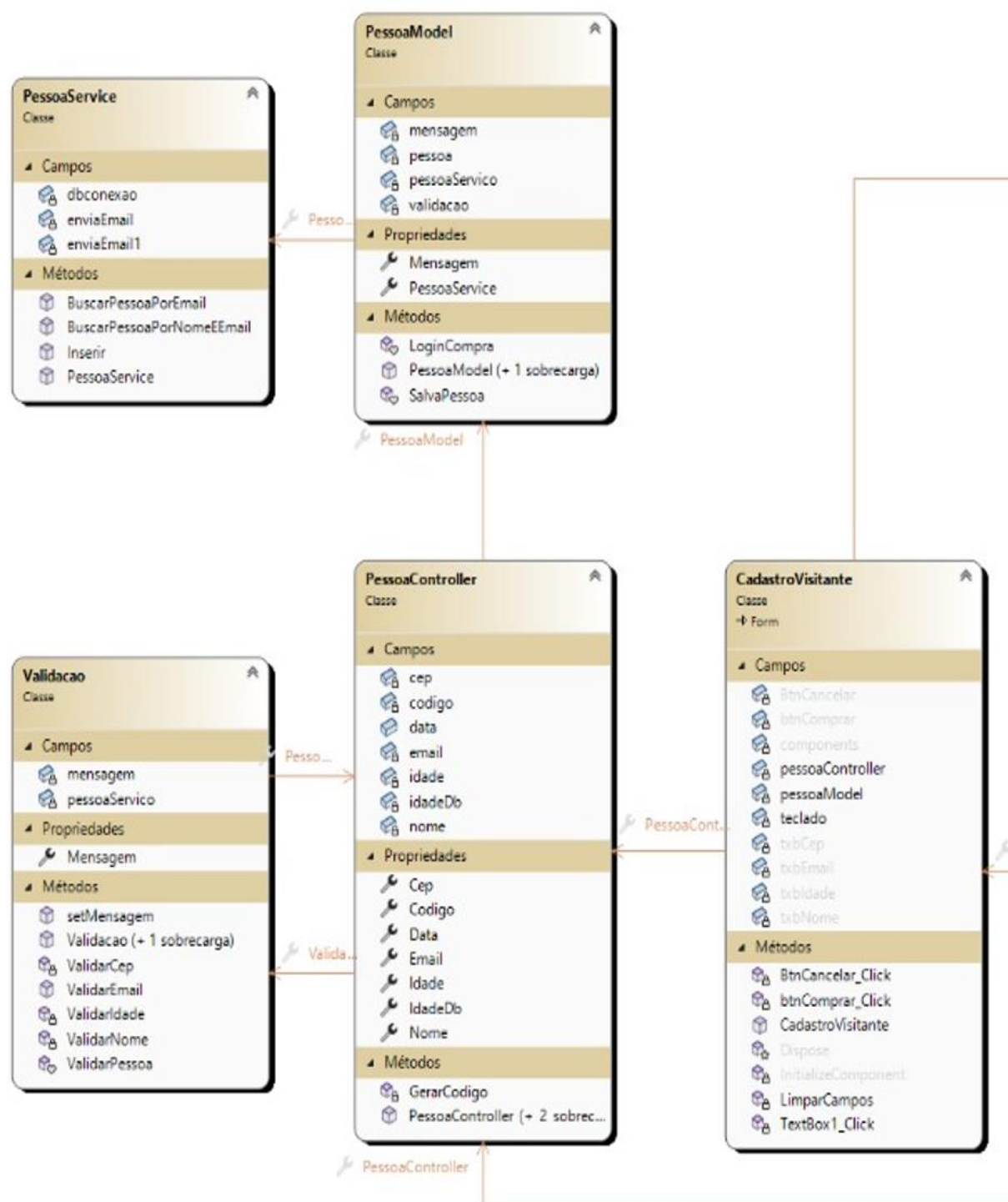
Diagrama de classe é uma ilustração gráfica das categorias de um sistema, suas características, procedimentos e as conexões entre elas.

Essencial na construção de sistemas orientados a objetos, é empregado para detalhar a organização fixa de um sistema.

A seguir serão demonstrados os diagramas dos sistemas criados e sua comunicação. Começaremos com o sistema de vendas que seria o início da entrada do museu que será mostrado na figura 27 e figura 28.

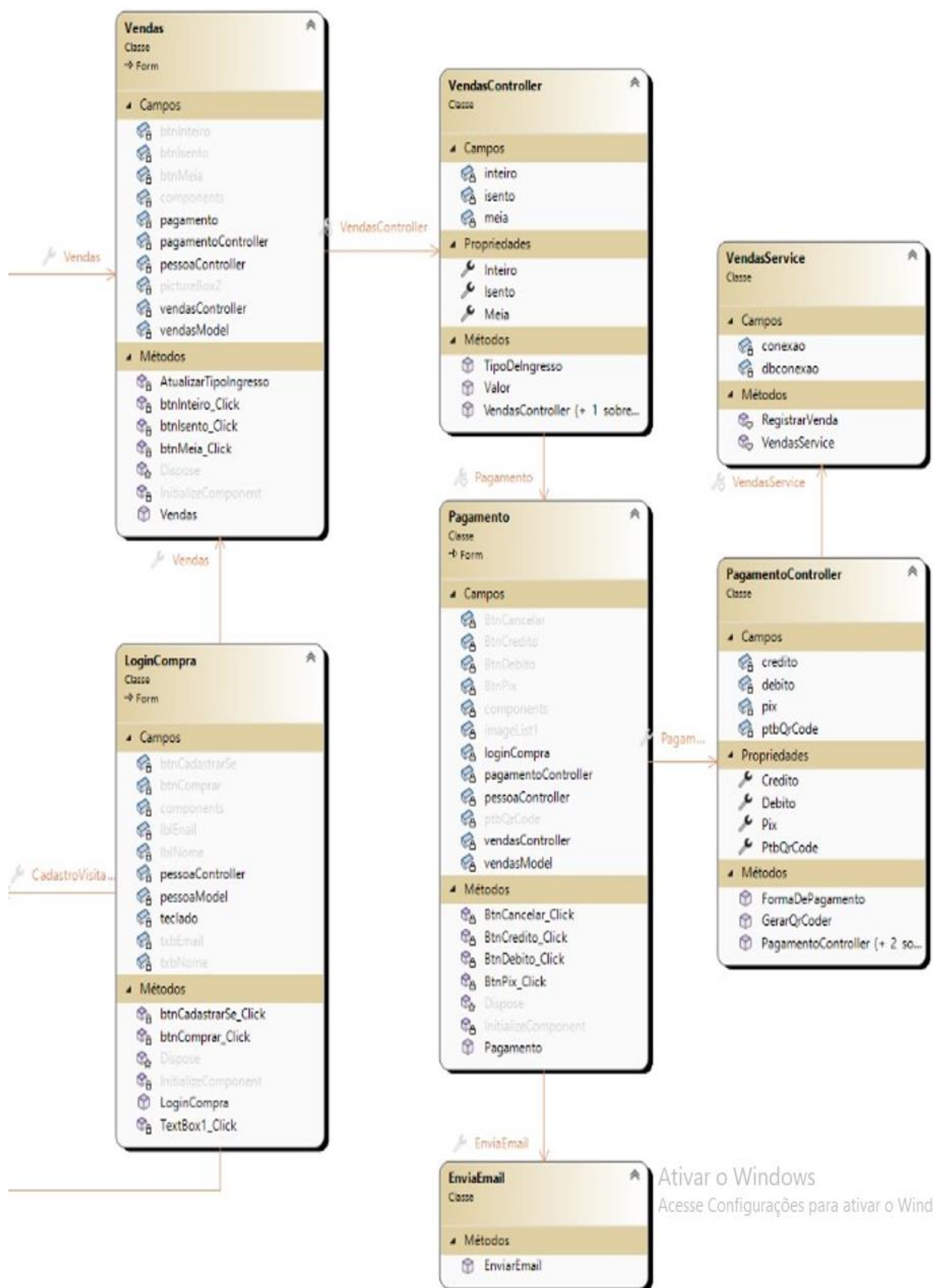
Após visualizarmos o diagrama de vendas será demonstrado em duas partes também na figura 29 e figura 30 o diagrama de classe do questionário das obras.

Figura 27 – Diagrama de classe de vendas



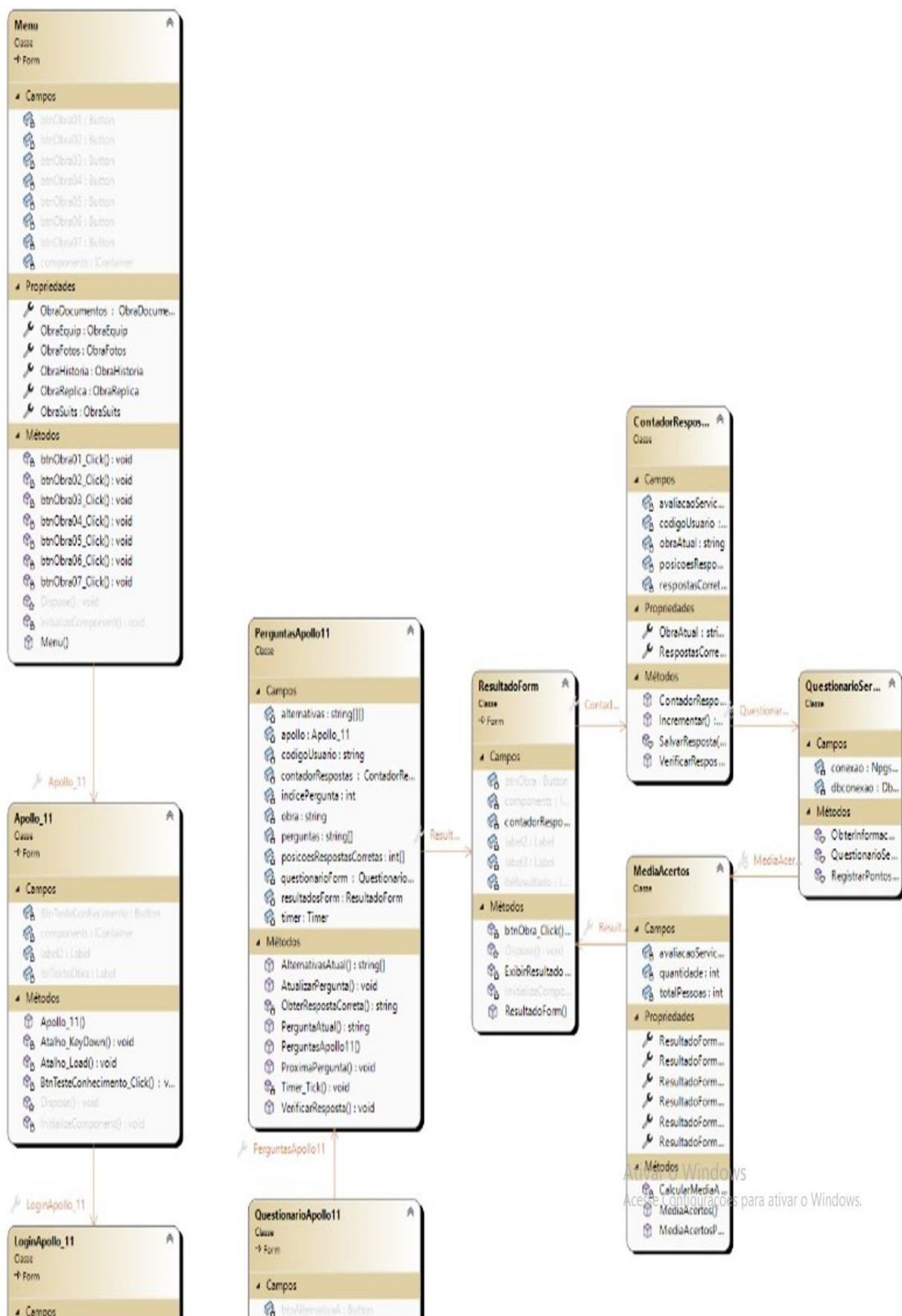
Fonte: Autoria própria.

Figura 28 – Diagrama de classe de vendas continuação



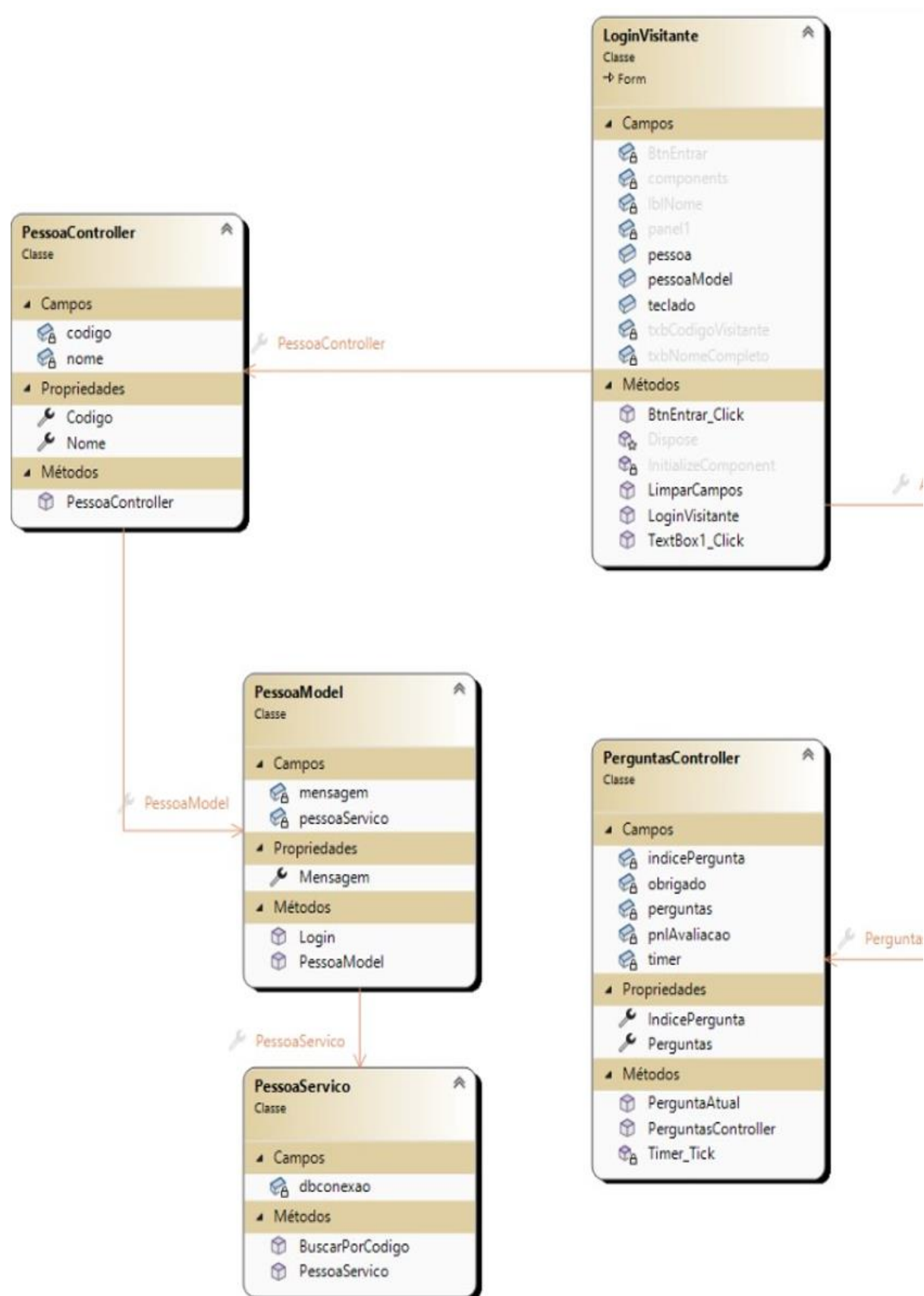
Fonte: Autoria própria.

Figura 29 – Diagrama de classe do questionário



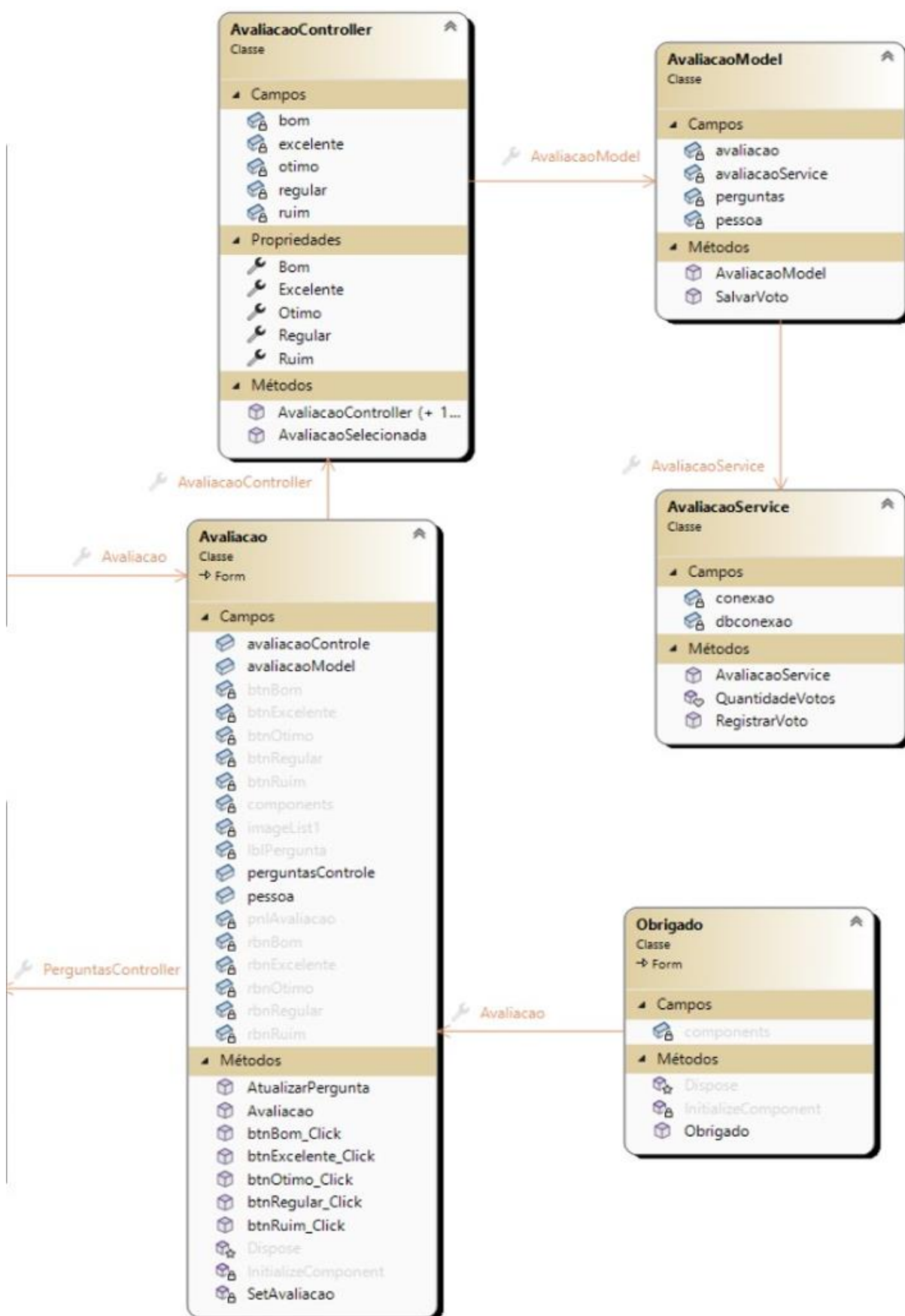
Fonte: Autoria própria.

Figura 31 – Diagrama de classe da avaliação



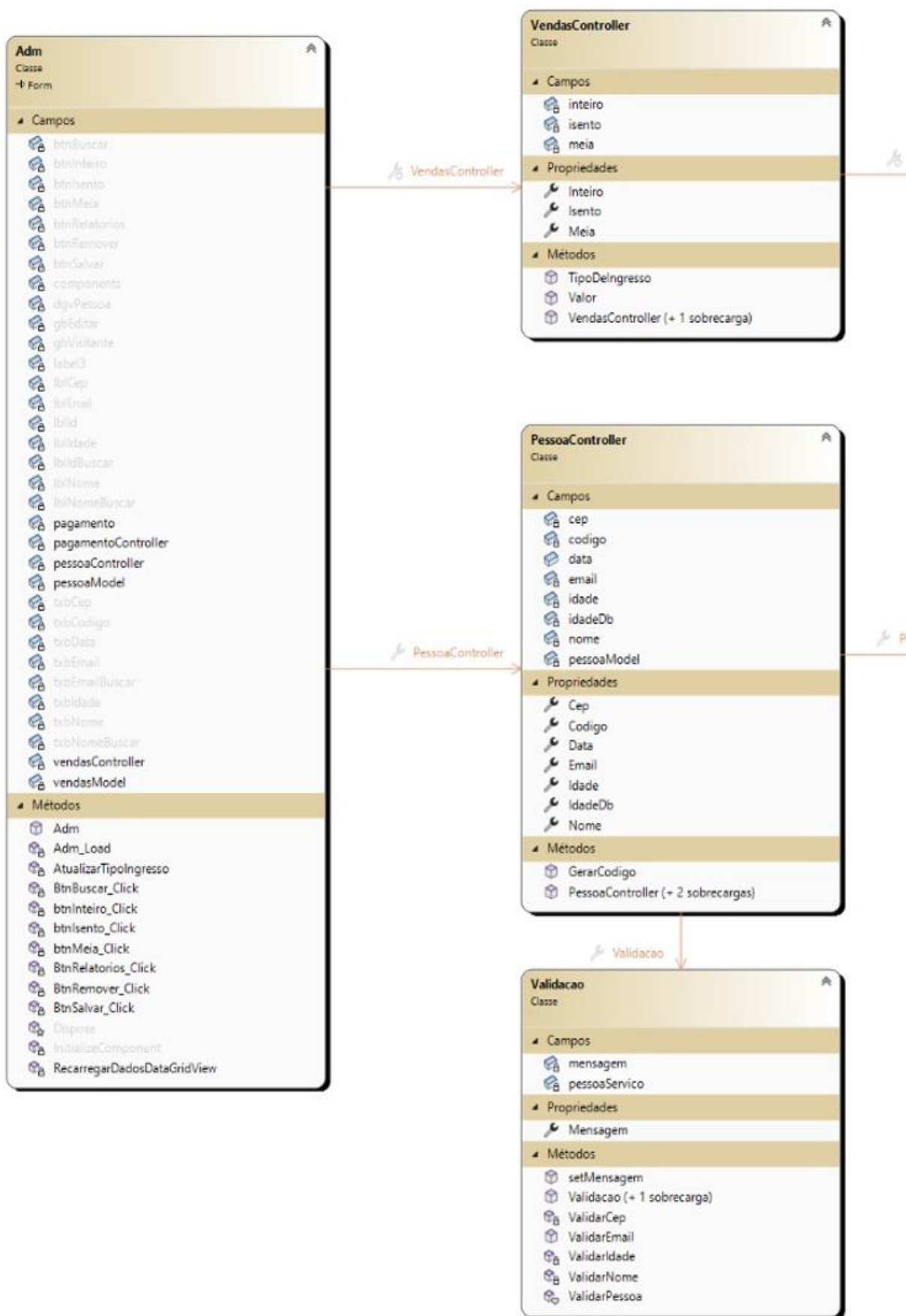
Fonte: Autoria própria.

Figura 32 – Diagrama de classe da avaliação continuação



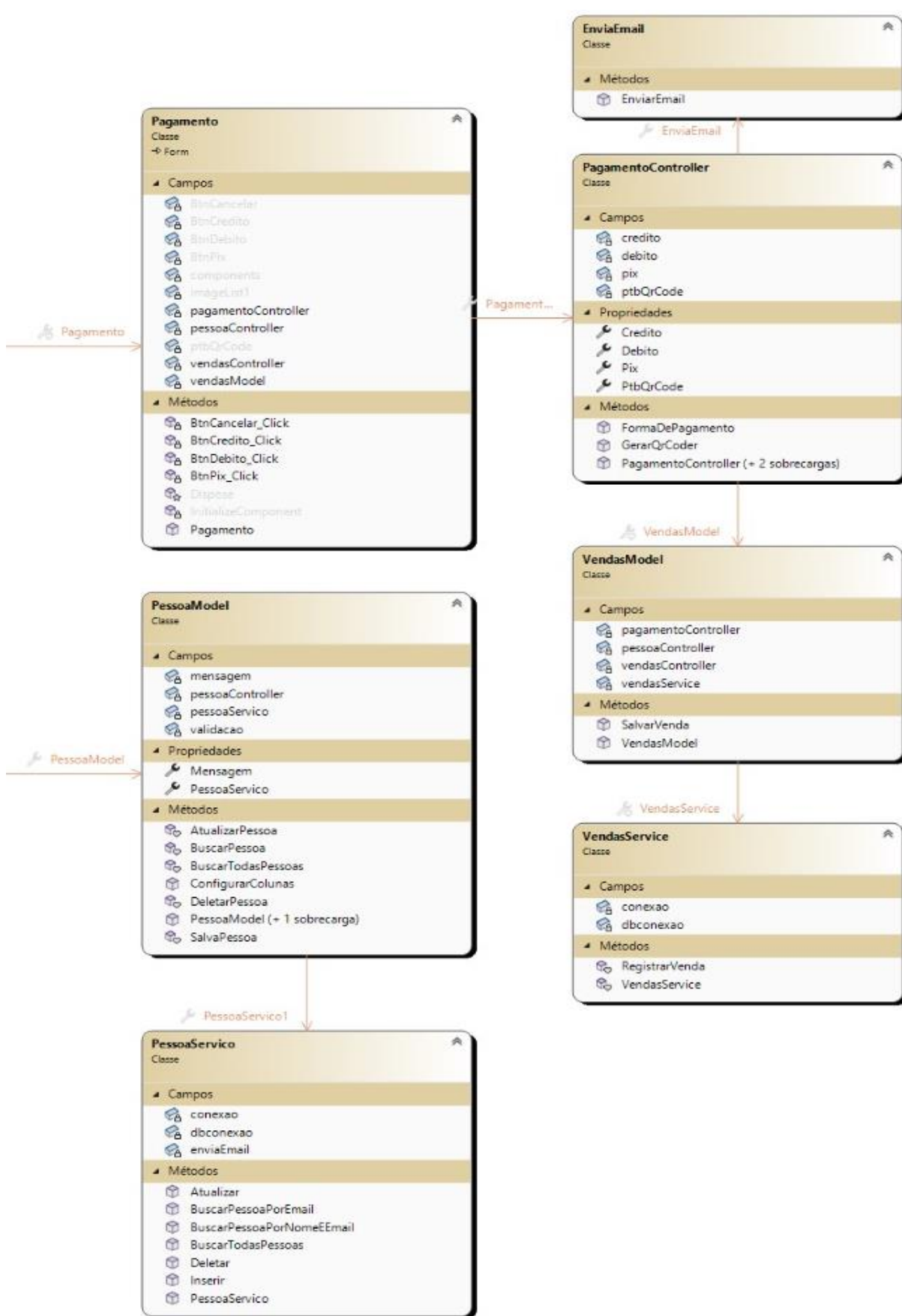
Fonte: Autoria própria.

Figura 33 – Diagrama de classe do ADM



Fonte: Autoria própria.

Figura 34 – Diagrama de classe do ADM continuação



Fonte: Autoria própria.

15.4. Programação

Para o totem interativo do museu, a programação desempenha um papel fundamental na criação de uma experiência envolvente e educativa para os visitantes. A programação envolve a implementação de funcionalidades interativas, a gestão de conteúdo dinâmico e a criação de uma interface intuitiva e atrativa.

Além de um meio para gerir relatórios de pesquisas dos visitantes para melhorias futuras, entre outras funções, a seguir será mostrado o desenvolvimento do sistema, ferramentas utilizadas e logicas na programação, serão pegos trecho principais do programa em si para visualização.

15.5. Testes executados nos sistemas

Os testes desempenham um papel fundamental no processo de criação de software, uma vez que garantem a correta funcionamento do código, identificam possíveis erros antes da fase de produção, possibilitam a refatoração de forma segura, funcionam como uma documentação executável, tornam mais fácil a manutenção do código, aumentam a credibilidade na qualidade do software e oferecem um retorno imediato sobre o código desenvolvido.

15.6. Testes executados com a xUnit.net

xUnit.net é uma ferramenta de teste destinada ao ambiente *.NET* (como *C#*, *F#* e outros), possibilitando a criação e execução de testes de unidade de maneira eficiente. Sua estrutura segue as diretrizes do *xUnit*, um padrão de nomenclatura comum em *frameworks* de teste unitário.

No exemplo a seguir demonstrados na figura 35 podemos ver a execução de teste realizado no software e suas tentativas e falhas. Concluindo um funcionamento normal em quatro tentativas.

Figura 35 – Teste realizado com a ferramenta xUnit.net

```
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v2.4.5+1caef2f33e (64-bit .NET 7.0.18)
[xUnit.net 00:00:00.64] Starting: TestProject1
[xUnit.net 00:00:01.35] Finished: TestProject1
===== Execução de teste concluída: 4 Testes (4 Aprovados, 0 Com falha, 0 Ignorados) executados em 1,4 s =====
Iniciando a detecção do teste para a execução do teste solicitado
```

Fonte: Autoria própria.

15.7. Teclado Virtual

Este código representa um programa em C# que simula um teclado virtual. Foi especificamente desenvolvido para complementar um monitor sensível ao toque, integrando-se diretamente à aplicação do museu *FutureTech*.

A solução oferecida visa facilitar a entrada de texto em uma aplicação interativa projetada para o museu *FutureTech*, proporcionando aos visitantes uma experiência de interação intuitiva e eficiente (que está presente completo no Apêndice).

- a) **Variáveis de Controle:** *shiftAtivado*: Controla se a tecla Shift está ativada.
targetTextBox: Armazena o *TextBox* onde o texto será inserido;
- b) **Construtor e Evento *Load***: No construtor, o formulário é configurado como o mais alto na pilha de z-order e são associados eventos de carregamento (*Load*) e desativação (*Deactivate*) do formulário;
- c) **Métodos:** *FocusTargetTextBox ()*: Foca o *TextBox* alvo para entrada de texto:
 - **Teclado_*Load***: Centraliza o formulário na tela principal quando carregado;
 - **AdicionarLetra (*string* letra)**: Adiciona uma letra ao *TextBox* alvo na posição do cursor. *TextBox_Click*: Define o *TextBox* alvo quando clicado;
 - **SetTargetTextBox (*TextBox* targetTextBox)**: Define o *TextBox* alvo.
- d) **Eventos dos Botões:** *btnLetraA_Click*, *btnShift_Click* etc.: Adicionam letras, alternam entre letras maiúsculas e minúsculas, e adicionam caracteres especiais ao *TextBox* alvo;

- e) **Método `btnBack_Click` e `btnRun_Click`:** *`btnBack_Click`*: Remove o último caractere do *TextBox* alvo; *`btnRun_Click`*: Oculta o teclado.

O teclado virtual é uma ferramenta interativa que permite aos usuários inserirem texto de forma intuitiva em campos de entrada de texto na aplicação. O programa inicia criando um formulário que representa o teclado virtual. Este formulário é configurado para ser exibido como o mais alto na pilha de *z-order*, isso significa será exibido acima de todos os outros elementos na interface gráfica, garantindo sua visibilidade e acessibilidade aos usuários.

Essa configuração é importante para garantir uma interação eficiente, especialmente em dispositivos com telas sensíveis ao toque, como no caso do teclado virtual no código fornecido que garante que permaneça visível durante a interação do usuário. Além disso, são associados eventos importantes, como o carregamento do formulário e a perda de foco, a métodos correspondentes que controlam o comportamento do teclado.

Uma vez que o teclado virtual é exibido, o usuário pode interagir com ele através da tela sensível ao toque. Os métodos implementados no código permitem adicionar letras ao campo de texto alvo, remover caracteres, alternar entre letras maiúsculas e minúsculas, e adicionar caracteres especiais, como *underscore*, espaço, arroba e ".com".

O código também inclui funcionalidades para posicionar o teclado centralizado na tela do dispositivo quando é carregado. Isso garante uma experiência de usuário agradável e consistente, o código em si segue no apêndice e os métodos serão apresentados na tabela 9.

Tabela 9 – Código em C# do teclado

```
// Método para focar o TextBox alvo
public void FocusTargetTextBox()
{
    targetTextBox.Focus();
}

// Método executado quando o formulário é carregado
private void Teclado_Load(object sender, EventArgs e)
{
    // Centraliza o formulário na tela principal
```



```

        int x = (Screen.PrimaryScreen.WorkingArea.Width - this.Width) / 2;
        int y = Screen.PrimaryScreen.WorkingArea.Height - this.Height;
        this.Location = new Point(x, y);
    }
    // Método para adicionar uma letra ao TextBox alvo
    private void AdicionarLetra(string letra)
    {
        if (targetTextBox != null)
        {
            // Obtém a posição do cursor a partir do início e do fim do texto
            int cursorPositionFromStart = targetTextBox.SelectionStart;
            int cursorPositionFromEnd = targetTextBox.TextLength - cursorPositionFromStart;
            // Insere a letra na posição do cursor
            targetTextBox.Text = targetTextBox.Text.Insert(cursorPositionFromStart, letra);
            // Define a nova posição do cursor após a inserção
            targetTextBox.SelectionStart = cursorPositionFromStart + letra.Length;
            targetTextBox.SelectionLength = 0;
        }
    }
    // Método para definir o TextBox alvo
    public void SetTargetTextBox(TextBox targetTextBox)
    {
        this.targetTextBox = targetTextBox;
        this.targetTextBox.HideSelection = false; // Define se a seleção do TextBox será oculta ou
        não
        this.targetTextBox.Focus(); // Dá foco ao TextBox alvo
    }
    // Métodos para adicionar letras ao TextBox alvo, com ou sem shift ativado
    // Métodos para letras maiúsculas e minúsculas
    private void btnLetraA_Click(object sender, EventArgs e) { /* ... */ }
    private void btnLetraQ_Click(object sender, EventArgs e) { /* ... */ }

    // Método para remover o último caractere do TextBox alvo
    private void btnBack_Click(object sender, EventArgs e) { /* ... */ }

    // Método para abrir o teclado numérico
    private void btn123_Click(object sender, EventArgs e) { /* ... */ }

    // Método para alternar entre maiúsculas e minúsculas
    private void btnShift_Click(object sender, EventArgs e) { /* ... */ }

```

```
// Métodos para adicionar caracteres especiais
private void btnUnderline_Click(object sender, EventArgs e) { /* ... */ }
private void btnSpace_Click(object sender, EventArgs e) { /* ... */ }
private void btnArroba_Click(object sender, EventArgs e) { /* ... */ }
private void btnPontoCom_Click(object sender, EventArgs e) { /* ... */ }

// Método para ocultar o teclado
private void btnRun_Click(object sender, EventArgs e)
{
    this.Hide();
}
```

Fonte: Autoria própria.

15.8. Pesquisa de Satisfação

O sistema desenvolvido para a interação com os visitantes do museu com o objetivo de coletar as opiniões dos visitantes sobre aspectos diferentes sobre a exposição, museu, sistema etc. A classe PerguntasController se torna responsável por gerenciar as perguntas, exibindo uma pergunta de cada vez em um painel dinâmico, permitindo que o visitante escolha entre as opções de avaliação disponíveis, como "Ruim", "Regular", "Bom", "Ótimo" e "Excelente" mostrando a tela de agradecimento quando todas as perguntas tenham sido respondidas. Como podemos ver o código no apêndice e a seguir os métodos principais na tabela 10.

Tabela 10 – Código em C# da pesquisa de satisfação da PerguntasController

```
public class PerguntasController
{
private Obrigado obrigado;
    private MediaAvaliacaoModel media;
    private Panel pnlAvaliacao;
    private int indicePergunta = 0;
    private string[] perguntas = {
        "Como você avaliaria a clareza das instruções fornecidas para interagir com as exposições?",
        "Em sua opinião, como as tecnologias utilizadas nas exposições contribuíram para a sua compreensão dos temas abordados?",
        "Como você classificaria a interatividade das exposições?",
```

```

        "Considerando sua visita ao museu, como você classificaria sua satisfação geral?",
        "Como você avalia a facilidade de uso do sistema dos totens para interagir com as
        exposições?"
    }
    public string PerguntaAtual()
    {
        if (indicePergunta < perguntas.Length)
        {
            return perguntas[indicePergunta];
        }
        else
        {
            // Exibindo a tela de agradecimento e finalizando a avaliação
            pnlAvaliacao.Visible = false;
            obrigado.Show();
            media.CalcularMediaAvaliacoes(); // Calcular a média das avaliações
            return "";
        }
    }

    private void Media_AvaliacoesConcluidas(object sender, EventArgs e)
    {
        // Encontrar a instância correta de Obrigado
        Obrigado obrigadoForm =
        (Obrigado)Application.OpenForms.OfType<Obrigado>().FirstOrDefault();
        if (obrigadoForm != null)
        {
            obrigadoForm.SetMediaAvaliacao(media);
        }
    }
}

```

Fonte: Autoria própria.

A classe `MediaAvaliacaoModel` se torna responsável por calcular a média das avaliações fornecidas. Nesse contexto, o Controller é responsável por atualizar a tela de agradecimento com a média calculada.

Além disso, a classe oferece eventos para sinalizar quando as avaliações tenham sido concluídas e métodos para obter a imagem correspondente à média total de avaliação. A imagem é utilizada para representar visualmente a avaliação geral

dos visitantes, permitindo uma rápida e fácil compreensão. Que o código segue no apêndice e na tabela 11 a seguir os métodos principais.

**Tabela 11 – Código em C# da pesquisa de satisfação da classe
MediaAvaliacaoModel**

```
public event EventHandler AvaliacoesConcluidas;
public MediaAvaliacaoModel()
{
    avaliacaoService = new AvaliacaoService();
}
private void OnAvaliacoesConcluidas()
{
    AvaliacoesConcluidas?.Invoke(this, EventArgs.Empty);
}
public void CalcularMediaAvaliacoes()
{
    if (avaliacaoService == null)
    {
        throw new InvalidOperationException("AvaliacaoService não foi inicializado.");
    }
    List<string> resultados = avaliacaoService.QuantidadeVotos();
    CalcularQuantidades(resultados);
    int totalAvaliacoes = Ruim + Regular + Bom + Otimo + Excelente;
    media = (Ruim * 1 + Regular * 2 + Bom * 3 + Otimo * 4 + Excelente * 5) / (double)totalAvaliacoes;
    mediaMensagem = $"{media:F1}";
    mensagem = $"{totalAvaliacoes/5} avaliações";
    OnAvaliacoesConcluidas();
}
public Image ObterImagemMedia()
{
    Image imagemExibir = null;
    if (this.media >= 4.5)
    {
        imagemExibir = Properties.Resources.cincoestrelas;
    }
    else if (media >= 3.5)
    {
        imagemExibir = Properties.Resources.quatroestrelas;
    }
}
```

```

else if (media >= 2.5)
{
    imagemExibir = Properties.Resources.tresestrelas;
}
else if (media >= 1.5)
{
    imagemExibir = Properties.Resources.duasestrelas;
}
else
{
    imagemExibir = Properties.Resources.umaestrela;
}
return imagemExibir;
}

private void CalcularQuantidades(List<string> resultados)
{
    foreach (string resultado in resultados)
    {
        string[] partes = resultado.Split('\t');
        if (partes.Length == 2)
        {
            string voto = partes[0].ToLower();
            int quantidade = int.Parse(partes[1]);
            switch (voto)
            {
                case "ruim":
                    Ruim = quantidade;
                    break;
                case "regular":
                    Regular = quantidade;
                    break;
                case "bom":
                    Bom = quantidade;
                    break;
                case "otimo":
                    Otimo = quantidade;
                    break;
                case "excelente":
                    Excelente = quantidade;
                    break;
            }
        }
    }
}

```

```

    }
  }
}

```

Fonte: Autoria própria.

15.9. Questionários das obras

O sistema elaborado sobre um questionário de múltipla escolha do presente trabalho é implementado pelas classes Perguntas dos programas referentes as obras, que é responsável por gerenciar as perguntas e respostas do questionário.

O construtor da classe recebe parâmetros uma instancia do questionário que o representa, um contador de responder, o código do visitante, que segue no apêndice e o nome da obra, inicializando os campos correspondentes e configurando um timer responsável para fechar a tela de resultados que é instanciada logo quando é finalizado as perguntas do questionário, que podemos ver na tabela 12 a seguir os métodos principais.

Tabela 12 – Questionários das obras construtor

```

private string[] perguntas = {
    "Qual era o nome do módulo lunar usado na missão Apollo 11?",
    "Quantos astronautas a nave Apollo 11 levou para a Lua?",
    "Qual foi o papel do módulo lunar na missão Apollo 11?",
    "Qual era o nome do primeiro astronauta a pisar na Lua a bordo do módulo lunar?",
    "Como o módulo lunar retornou à órbita lunar após a missão na superfície?"
};

private string[][] alternativas = {
    new string[] { "A) Eagle",
        "B) Falcon",
        "C) Hawk",
        "D) Sparrow",
        "E) Osprey" }, // Resposta correta: A
    new string[] { "A) Quatro",
        "B) Dois",
        "C) Três",
        "D) Cinco",

```

```

        "E) Seis" }, // Resposta correta: C
new string[] { "A) Orbitar a Lua e transmitir dados para a Terra ",
        "B) Transportar astronautas até a Lua e mantê-los seguros ", // Resposta correta: B
        "C) Pousar na Lua e retornar à órbita lunar",
        "D) Explorar a superfície lunar em busca de recursos",
        "E) Construir uma base lunar para futuras missões" },
new string[] { "A) Neil Armstrong",
        "B) Buzz Aldrin",
        "C) Michael Collins",
        "D) Alan Shepard",
        "E) John Glenn" }, // Resposta correta: A
new string[] { "A) Não retornou à órbita, permanecendo na superfície lunar ",
        "B) Utilizou foguetes auxiliares para retornar à órbita",
        "C) Foi empurrado por astronautas para voltar à órbita",
        "D) Aterrissou em uma área predestinada para ser recuperado posteriormente",
        "E) Decolou da superfície lunar e se acoplou ao módulo de comando" } // Resposta
correta: E
    };
    private int[] posicoesRespostasCorretas = { 0, 2, 1, 0, 4 };
    public PerguntasApollo11(QuestionarioApollo11 questionarioForm, ContadorRespostas
contadorRespostas, string codigoUsuario, string obra)
    {
        this.questionarioForm = questionarioForm;
        this.contadorRespostas = contadorRespostas;
        this.codigoUsuario = codigoUsuario;
        contadorRespostas.ObraAtual = obra;
        timer = new WinFormsTimer();
        timer.Interval = 10000;
        timer.Tick += Timer_Tick;
    }

```

Fonte: Autoria própria.

É definido *arrays* para armazenar as perguntas e as alternativas de cada pergunta, definindo também as posições das respostas corretas. Sendo assim, é necessário criar métodos para auxiliar a gerenciar como verificar cada resposta do visitante no questionário, sendo eles:

15.9.1.1. Método *VerificarResposta*

Verifica se a resposta fornecida pelo visitante está correta, comparando a posição da resposta escolhida com a posição correta da alternativa, como podemos ver na tabela 13:

Tabela 13 - método *VerificarResposta*

```
public void VerificarResposta(string respostaUsuario)
{
    string respostaCorreta = ObterRespostaCorreta(indicePergunta);

    int posicaoRespostaUsuario = respostaUsuario[0] - 'A';

    if (posicaoRespostaUsuario == posicoesRespostasCorretas[indicePergunta])
    {
        contadorRespostas.Incrementar();
    }
}
```

Fonte: Autoria própria.

15.9.1.2. O método *PerguntaAtual*

Retorna a pergunta que o visitante se encontra de acordo com o índice do *array*, sendo verificado toda vez que o método é chamado tendo uma estrutura de decisão que verifica se existem mais perguntas para o visitante pode responder, se não tiver, ele é redirecionado para o resultado do questionário, como podemos ver na tabela 14.

Tabela 14 - método *PerguntaAtual*

```
public string PerguntaAtual()
{
    if (indicePergunta < perguntas.Length)
    {
        return perguntas[indicePergunta];
    }
    else
    {

```



```

        timer.Start();
        resultadosForm = new ResultadoForm(contadorRespostas);
        resultadosForm.ShowDialog();
        questionarioForm.Hide();
        return "";
    }
}

```

Fonte: Autoria própria.

15.9.1.3. Os métodos *AlternativaAtual*, *ProximaPergunta* e *AtualizarPergunta*

São responsáveis por retornar as alternativas para a pergunta atual, avançar para a próxima pergunta e atualizar o texto referente a pergunta, como podemos ver na tabela 15.

Tabela 15 - métodos *AlternativaAtual*, *ProximaPergunta* e *AtualizarPergunta*

```

public string[] AlternativasAtual()
{
    if (indicePergunta < alternativas.Length)
    {
        return alternativas[indicePergunta];
    }
    else
    {
        return null;
    }
}

public void ProximaPergunta()
{
    indicePergunta++;
}

```

Fonte: Autoria própria.

A classe *ContadorRespostas* é responsável por manter o controle das respostas corretas dadas pelo visitante, fazendo parte do sistema implementado para interação com o público do museu.

Dois construtores são fornecidos, um que recebe apenas o código do usuário e associa à propriedade do código e o outro que recebe um *array* de posições das

respostas corretas e a obra referente ao questionário como segue no apêndice. São utilizados métodos para verificar se a resposta está correta, para incrementar ao contador de respostas corretas e para salvar essas informações no banco de dados. Como podemos ver os métodos na tabela 16.

Tabela 16 - controle das respostas corretas

```
public void VerificarResposta(int indicePergunta, int respostaUsuario)
{
    if (indicePergunta < posicoesRespostasCorretas.Length && respostaUsuario ==
posicoesRespostasCorretas[indicePergunta])
    {
        // Incrementa o contador de respostas corretas.
        Incrementar();
    }
}

internal void SalvarResposta()
{
    avaliacaoService.RegistrarPontos(respostasCorretas, codigoUsuario, ObraAtual);
}

public void Quantidade()
{
    int quant = respostasCorretas;
    // Exibir as informações em uma caixa de mensagem
    Mensagem = ("{quant}");
}
```

Fonte: Autoria própria.

Adicionando a classe MediaAcertos para calcular a média de acertos total dos visitantes que interagiram com o sistema do questionário sobre a obra referente. Exibindo para uma variável de mensagem a média calculada e a porcentagem de acertos. Apresentados na tabela 17 a seguir.

O construtor inicializa a classe que é responsável por trazer as informações do banco de dados do questionário. São utilizados os seguintes métodos:

- a) O método MediaAcertosPorObra que deve receber o ContadorRespostas como parâmetro utilizando esse objeto para obter informações sobre a obra atual, como quantidade de acertos e total de pessoas que participaram do

questionário. Sendo assim, calcula a média de acertos e a percentagem de acertos em relação ao total de perguntas do questionário;

- b) O método `CalcularMediaAcertos` mantém responsável por calcular a média de acertos verificando se o total de pessoas que responderam ao questionário é diferente de zero para evitar uma divisão por zero.

Tabela 17 – Média de acertos

```
public MediaAcertos()
{
    avaliacaoService = new QuestionarioService();
}
public void MediaAcertosPorObra(ContadorRespostas contador)
{
    (quantidade,totalPessoas) =
    avaliacaoService.ObterInformacoesPorObra(contador.ObraAtual);
    double mediaAcertos = CalcularMediaAcertos();
    double totalPerguntasPorQuestionario = 5;
    // Exibir as informações em uma caixa de mensagem
    double percentagemAcertos = (mediaAcertos/totalPerguntasPorQuestionario) * 100.0;
    // Exibir as informações em uma caixa de mensagem
    Mensagem = ("A média de acertos para a obra é: {mediaAcertos:F1} \n\n Porcentagem de
    acertos: {percentagemAcertos:F1} %");
}
private double CalcularMediaAcertos()
{
    if (totalPessoas != 0)
    {
        double mediaAcertos = (double)quantidade/totalPessoas;
        return mediaAcertos;
    }
    else
    {
        return 0;
    }
}
public string Mensagem { get => mensagem; set => mensagem = value; }
}
```

Fonte: Autoria própria.

Em conjunto, a utilização das classes citadas permite não apenas a apresentação das perguntas e o registro das respostas dos usuários, mas também a análise das respostas, fornecendo informações valiosas sobre o conhecimento e o interesse do público em relação ao tema abordado.

15.10. Classe Services

Para realizar a tarefa de armazenar os dados dos sistemas é criado classes que são responsáveis por mandar e receber os dados do banco de dados utilizado no presente trabalho, demonstrados na tabela 18. As classes que foram utilizadas em cada sistema são denominadas *service* para melhor entendimento do programa. Sendo assim, será apresentado a seguir o que cada classe de *service* é responsável:

15.10.1.1. VendasService

É utilizada para gerenciar as vendas realizadas no sistema de venda de ingressos. É utilizado uma conexão com o banco de dados para inserir as informações na tabela que a representa. Cada registro na tabela significa uma forma de pagamento, data da venda, código do visitante e valor do ingresso:

Tabela 18 – armazenamentos de dados

```
internal void RegistrarVenda(string pagamentoAtual, string codigoUsuario, string
vendaAtual,string e_mail, string nome, string valorIngresso)
{
    EnviaEmail enviaEmail = new EnviaEmail();
    using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO
public.tbl_vendas(formadepagamento, data, codigo)
                                VALUES (@formadepagamento, @Data,@CodigoPessoa)",
conexao))
    {
        command.Parameters.AddWithValue("@CodigoPessoa", codigoUsuario);
        command.Parameters.AddWithValue("@ingresso", vendaAtual);
        command.Parameters.AddWithValue("@formadepagamento", pagamentoAtual);
        command.Parameters.AddWithValue("@Data", DateTime.Now);
```

```

        int linhasAfetadas = command.ExecuteNonQuery();

        if (linhasAfetadas < 0)
        {
            MessageBox.Show("Falha ao registrar o voto.");
        }
        enviaEmail.EnviaEmail(nome,e_mail,codigoUsuario, pagamentoAtual, valorIngresso);
    }

```

Fonte: Autoria própria.

15.10.1.2. AvaliacaoService

Pertence ao programa que gerencia as avaliações do sistema, utilizando uma conexão com o banco de dados para inserir as informações sobre as avaliações na tabela de votos, registrando na tabela cada avaliação com os dados como o código do visitante, qual a pergunta que foi avaliada, resposta que foi dada pelo visitante e a data da avaliação, demonstrados na tabela 19.

Tabela 19 – conexão com o banco de dados

```

public void RegistrarVoto(string pergunta, string avaliacao, string codigoUsuario)
{
    using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO
public.tbl_votos (codigo, pergunta, voto, data)
VALUES (@CodigoPessoa, @Pergunta, @Voto, @Data)",
conexao))
    {
        command.Parameters.AddWithValue("@CodigoPessoa", codigoUsuario);
        command.Parameters.AddWithValue("@Pergunta", pergunta);
        command.Parameters.AddWithValue("@Voto", avaliacao);
        command.Parameters.AddWithValue("@Data", DateTime.Now);
        int linhasAfetadas = command.ExecuteNonQuery();
        if (linhasAfetadas < 0)
        {
            MessageBox.Show("Falha ao registrar o voto.");
        }
    }
}

internal List<string> QuantidadeVotos()
{

```

```

List<string> resultados = new List<string>();
using (NpgsqlCommand command = new NpgsqlCommand
    (@"SELECT voto, COUNT(voto) AS quantidade FROM tbl_votos GROUP BY voto",
conexao))
{
    var reader = command.ExecuteReader();
    while (reader.Read())
    {
        string voto = reader.GetString(0);
        int quantidade = reader.GetInt32(1);
        resultados.Add($"{voto}\t{quantidade}");
    }
}

```

Fonte: Autoria própria.

15.10.1.3. *QuestionarioService*

Mantém o serviço responsável por gerenciar os questionários respondido pelos visitantes utilizando uma conexão com o banco de dados para inserir informações sobre a tabela questionário. Cada registro da tabela representa um questionário respondido contendo dados como o código do visitante que respondeu, a quantidade de acertos, a obra que foi referente e a data de realização do questionário, demonstrados na tabela 20.

Tabela 20 – gerenciamento de questionários

```

internal void RegistrarPontos(int acertos,string codigoUsuario,string obraAtual)
{
    using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO
public.tbl_questionario (codigo, acertos,obra, data)
                                VALUES (@CodigoPessoa, @acertos,@obra, @Data)",
conexao))
    {
        command.Parameters.AddWithValue("@CodigoPessoa", codigoUsuario);
        command.Parameters.AddWithValue("@acertos", acertos);
        command.Parameters.AddWithValue("@obra", obraAtual);
        command.Parameters.AddWithValue("@Data", DateTime.Now);
        int linhasAfetadas = command.ExecuteNonQuery();
        if (linhasAfetadas < 0)
        {

```

```

        MessageBox.Show("Falha ao registrar o voto.");
    }
}
}
internal (int, int) ObterInformacoesPorObra(string nomeObra)
{
    int quantidadeAcertos = 0;
    int totalPessoas = 0;
    using (NpgsqlCommand command = new NpgsqlCommand
        (@"SELECT SUM(acertos), COUNT(DISTINCT codigo) FROM public.tbl_questionario
WHERE obra = @NomeObra", conexao))
    {
        command.Parameters.AddWithValue("@NomeObra", nomeObra);
        var reader = command.ExecuteReader();
        if (reader.Read())
        {
            if (!reader.IsDBNull(0))
            {
                quantidadeAcertos = reader.GetInt32(0);
            }
            if (!reader.IsDBNull(1))
            {
                totalPessoas = reader.GetInt32(1);
            }
        }
    }
    return (quantidadeAcertos, totalPessoas);
}
internal bool CodigoAssociadoObra(string codigo, string obra)
{
    bool codigoAssociado = false;
    using (NpgsqlCommand command = new NpgsqlCommand
        (@"SELECT COUNT(*) FROM public.tbl_questionario WHERE codigo = @Codigo AND
obra = @obra", conexao))
    {
        command.Parameters.AddWithValue("@Codigo", codigo);
        command.Parameters.AddWithValue("@obra", obra);
        long count = (long)command.ExecuteScalar();
        // Se rowCount for maior que zero, significa que o código está associado à obra
        if (count > 0)
    }
}

```

```

        {
            codigoAssociado = true;
        }
    }
    return codigoAssociado;
}
}
}

```

Fonte: Autoria própria.

15.11. Venda de ingressos

Este programa em *CSharp* simula o processo de venda de ingressos para o Museu FutureTech (que está presente completo no Apêndice). A ênfase desta etapa é o desenvolvimento de um sistema para gerenciar as vendas de ingressos.

O programa utiliza bibliotecas padrão do *CSharp*, incluindo o *Npgsql*, que permite o acesso a bancos de dados PostgreSQL, e o *namespace* *PIM_III_ADS_VENDAS.Utills*, que contém a classe *Dbconexao*, que podem ser vistas no apêndice.

O método *RegistrarVenda()*: registra uma venda no banco de dados, recebendo informações como método de pagamento, código do usuário, tipo de venda, e-mail, nome e valor do ingresso. Ele instancia a classe *EnviaEmail* para lidar com o envio de e-mails.

Um bloco *using* é utilizado para criar e configurar um comando SQL (*NpgsqlCommand*) para inserir dados na tabela *tbl_vendas*. Parâmetros são adicionados ao comando SQL para realizar a inserção na tabela.

O comando SQL é executado usando *ExecuteNonQuery()* para inserir os dados e obter o número de linhas afetadas. Se o número de linhas afetadas for menor que zero, uma mensagem de falha é exibida e o método *EnviarEmail()* é chamado para enviar um e-mail relacionado à venda. O método em si será apresentado na tabela 21 a seguir.

Tabela 21 – Venda de ingressos

```

internal void RegistrarVenda(string pagamentoAtual, string codigoUsuario, string
vendaAtual, string e_mail, string nome, string valorIngresso)
{

```



```

        EnviaEmail enviaEmail = new EnviaEmail();
        using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO
public.tbl_vendas(formadepagamento, data, codigo)
                        VALUES (@formadepagamento, @Data,@CodigoPessoa)",
conexao))
        {
            command.Parameters.AddWithValue("@CodigoPessoa", codigoUsuario);
            command.Parameters.AddWithValue("@ingresso", vendaAtual);
            command.Parameters.AddWithValue("@formadepagamento", pagamentoAtual);
            command.Parameters.AddWithValue("@Data", DateTime.Now);
            int linhasAfetadas = command.ExecuteNonQuery();
            if (linhasAfetadas < 0)
            {
                MessageBox.Show("Falha ao registrar o voto.");
            }
            enviaEmail.EnviaEmail(nome,e_mail,codigoUsuario, pagamentoAtual, valorIngresso);
        }
    }
}
}

```

Fonte: Autoria própria.

15.12. ADM

Este código representa um programa em C# que simula sistema de administração de dados. Foi especificamente desenvolvido para termos um controle melhor de visitantes no local e uma ajuda extra caso ocorra erros ao gerar ingressos para entrada do museu e gerir relatórios das pesquisas atribuídas no local para futuras melhorias e ver a participação do público.

O sistema de administração de visitantes é desenvolvido também para proporcionar melhor gerenciamento capaz de editar as informações dos visitantes de forma ágil e organizada.

Esse código demonstrado no apêndice é um exemplo de uma classe `PessoaController()` que gerencia informações de pessoas. Ela encapsula as informações de uma pessoa e fornece métodos para manipular essas informações de forma segura.

Método GerarCodigo: Esse método recebe o nome da pessoa como parâmetro e gera um código único para ela. Ele obtém as iniciais do nome, gera um número aleatório entre 10 e 99, uma letra aleatória e concatena essas partes para formar o código. Se o código gerado tiver mais de 5 caracteres, ele é truncado para 5 caracteres como podemos ver na tabela 22.

Tabela 22 – gerenciamento de informações de pessoas

```
public string GerarCodigo(string nome)
{
    // Obtém as iniciais do nome
    string iniciais = string.Join("", nome.Split(' ').Select(s => s[0]))
    // Gera um número aleatório entre 10 e 99
    Random rnd = new Random();
    int numero = rnd.Next(10, 100);
    // Gera uma letra aleatória
    char letra = (char)rnd.Next('A', 'Z');
    // Concatena as partes para formar o código
    string codigo = $"{iniciais}{numero}{letra}{numero}";
    if (codigo.Length > 5)
    {
        codigo = codigo.Substring(0, 5);
    }
    return codigo;
}
```

Fonte: Autoria própria.

No código a seguir na tabela 23 serão demonstrados os métodos que setaram os botões para suas ações no qual podem atualizar, deletar e buscar a pessoa dentro do banco de dados para melhor gerenciamento.

Métodos Internos:

- a) **SalvaPessoa(PessoaController pessoaController):** Verifica se a pessoa é válida (através do método ValidarPessoa da classe Validacao), e se sim, insere a pessoa no banco de dados através do serviço de pessoa (pessoaServico.Inserir);
- b) **AtualizarPessoa(PessoaController pessoaController):** Valida a pessoa e, se encontrada, atualiza seus dados no banco de dados;

- c) **DeletarPessoa(PessoaController pessoaController):** Verifica se a pessoa possui um código válido e, se sim, a deleta do banco de dados;
- d) **BuscarPessoa(PessoaController pessoaController):** Busca uma pessoa por nome e e-mail, e retorna uma mensagem indicando o resultado da operação;
- e) **BuscarTodasPessoas():** Retorna uma lista de todas as pessoas no banco de dados.

Tabela 23 - setando ações dos botões

```

internal void SalvaPessoa(PessoaController pessoaController)
{
    validacao.ValidarPessoa(pessoaController);

    if (validacao.Mensagem.Equals(""))
    {
        pessoaServico.Inserir(pessoaController);
        this.mensagem = "Cadastro realizado com sucesso.";
    }
    else
    {
        this.mensagem = validacao.Mensagem;
    }
}

internal void AtualizarPessoa(PessoaController pessoaController)
{
    validacao.ValidarPessoa(pessoaController);

    if (pessoaController != null)
    {
        pessoaServico.Atualizar(pessoaController);
        this.mensagem = "Pessoa atualizada com sucesso!";
    }
    else
    {
        this.mensagem = "Erro ao atualizar pessoa: pessoa não encontrada";
    }
}

```

```

internal void DeletarPessoa(PessoaController pessoaController)
{
    if (pessoaController.Codigo.Equals(""))
    {
        this.mensagem = "Erro ao Remover pessoa: pessoa não encontrada";
    }
    else
    {
        pessoaServico.Deletar(pessoaController);
        this.mensagem = "Pessoa excluída com sucesso!";
    }
}
internal void BuscarPessoa(PessoaController pessoaController)
{
    pessoaController = pessoaServico.BuscarPessoaPorNomeEEEmail(pessoaController);
    if (pessoaController != null && pessoaController.Nome != null && pessoaController.Email !=
null)
    {
        this.mensagem = "Pessoa encontrada com sucesso!";
    }
    else
    {
        this.mensagem = "Login inválido. Verifique suas credenciais.";
    }
}
internal List<PessoaController> BuscarTodasPessoas()
{
    IEnumerable<PessoaController> todasPessoas = pessoaServico.BuscarTodasPessoas();
    return todasPessoas.ToList();
}
}

```

Fonte: Autoria própria.

15.13. Conexão com o banco de dados avaliação

No código a seguir na tabela 24, a classe *AvaliacaoService* é responsável por lidar com a interação direta com o banco de dados *PostgreSQL*. Vou explicar como é feito o processo de salvamento dos votos no banco de dados:

Antes de tudo, é estabelecida uma conexão com o banco de dados PostgreSQL. Isso é feito através do objeto *NpgsqlConnection*, que é inicializado usando uma instância de *Dbconexao*, responsável por fornecer os detalhes da conexão.

Tabela 24 – Conexão com o banco de dados

```
private Dbconexao dbconexao = new Dbconexao();
private NpgsqlConnection conexao;

public AvaliacaoService()
{
    conexao = dbconexao.GetConnection() as NpgsqlConnection;
}
```

Fonte: Autoria própria.

O método *RegistrarVoto()* é utilizado para inserir um novo registro na tabela de votos do banco de dados. Ele recebe como parâmetros a pergunta, a avaliação e o código do usuário, como podemos ver na tabela 25.

Tabela 25 – Método RegistrarVoto()

```
public void RegistrarVoto(string pergunta, string avaliacao, string codigoUsuario)
{
    using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO public.votos
(codigo, pergunta, voto, data)
VALUES (@CodigoPessoa, @Pergunta, @Voto, @Data)", conexao))
    {
        command.Parameters.AddWithValue("@CodigoPessoa", codigoUsuario);
        command.Parameters.AddWithValue("@Pergunta", pergunta);
        command.Parameters.AddWithValue("@Voto", avaliacao);
        command.Parameters.AddWithValue("@Data", DateTime.Now);

        // Executa o comando SQL para inserir o voto no banco de dados
        command.ExecuteNonQuery();
    }
}
```

Fonte: Autoria própria.

Neste método, é criado um comando SQL de inserção (INSERT INTO) que adiciona um novo registro à tabela de votos. Os valores a serem inseridos (código do usuário, pergunta, avaliação e data) são passados como parâmetros para o comando SQL, o que ajuda a prevenir ataques de injeção de SQL e garante a segurança do sistema.

Após a configuração dos parâmetros, o comando SQL é executado usando o método *ExecuteNonQuery()*, que executa o comando no banco de dados sem retornar nenhum conjunto de resultados.

Portanto, o voto do usuário é salvo no banco de dados *PostgreSQL* através da execução de um comando SQL de inserção, que é construído com os dados fornecidos pelo usuário e enviado para o banco de dados usando a conexão estabelecida. Isso garante que os votos sejam armazenados de forma segura e eficiente para posterior análise.

15.14. Conexão com o banco de dados questionário

No código a seguir na tabela 26, o salvamento no banco de dados ocorre principalmente na classe *QuestionarioService*, dentro do método *RegistrarPontos()*. Vou explicar como é realizado esse processo:

Antes de tudo, é estabelecida uma conexão com o banco de dados *PostgreSQL*. Isso é feito através do objeto *NpgsqlConnection*, que é inicializado usando uma instância de *Dbconexao*, responsável por fornecer os detalhes da conexão.

Tabela 26 – Método RegistrarVoto()

```
private Dbconexao dbconexao = new Dbconexao();
private NpgsqlConnection conexao;

internal QuestionarioService()
{
    conexao = dbconexao.GetConnection() as NpgsqlConnection;
}
```

Fonte: Autoria própria.

O método `RegistrarPontos()` é utilizado para inserir um novo registro na tabela questionário do banco de dados. Ele recebe como parâmetros a quantidade de acertos, o código do usuário e a obra atual, como podemos ver na tabela 27.

Tabela 27 – Método RegistrarVoto()

```
internal void RegistrarPontos(int acertos, string codigoUsuario, string obraAtual)
{
    using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO public.questionario
(codigo, acertos, obra, data)
                VALUES (@CodigoPessoa, @acertos, @obra, @Data)", conexao))
    {
        command.Parameters.AddWithValue("@CodigoPessoa", codigoUsuario);
        command.Parameters.AddWithValue("@acertos", acertos);
        command.Parameters.AddWithValue("@obra", obraAtual);
        command.Parameters.AddWithValue("@Data", DateTime.Now);

        int linhasAfetadas = command.ExecuteNonQuery();

        if (linhasAfetadas < 0)
        {
            MessageBox.Show("Falha ao registrar os pontos.");
        }
    }
}
```

Fonte: Autoria própria.

Neste método, é criado um comando SQL de inserção (*INSERT INTO*) que adiciona um novo registro à tabela questionário. Os valores a serem inseridos (código do usuário, quantidade de acertos, obra e data) são passados como parâmetros para o comando SQL.

Após a configuração dos parâmetros, o comando SQL é executado usando o método *ExecuteNonQuery()*, que executa o comando no banco de dados sem retornar nenhum conjunto de resultados.

Portanto, os pontos do questionário, representando a quantidade de acertos de um usuário em uma determinada obra, são salvos no banco de dados *PostgreSQL* através da execução de um comando SQL de inserção, que é construído com os dados fornecidos e enviado para o banco de dados usando a conexão estabelecida.

Isso garante que os resultados do questionário sejam armazenados de forma segura e eficiente para posterior análise.

15.15. Conexão com o banco de dados vendas

No código apresentado no apêndice, o salvamento no banco de dados é realizado pela classe `PessoaService`. Aqui está o fluxo geral do processo de salvamento:

- a) **Chamada do Método de Inserção:** Quando um novo registro de pessoa é criado e validado, o método `Inserir` da classe `PessoaService` é chamado para salvar esses dados no banco de dados;
- b) **Conexão com o Banco de Dados:** Dentro do método `inserir`, uma conexão com o banco de dados PostgreSQL é estabelecida utilizando a classe `Dbconexao`. Esta classe fornece uma conexão com o banco de dados;
- c) **Execução da Inserção:** Após a conexão ser estabelecida, uma query SQL de inserção é executada na tabela visitante do banco de dados. Esta query inclui os dados da pessoa a serem inseridos, como nome, idade, e-mail, CEP, data e código;
- d) **Execução Segura:** Para evitar injeção de SQL e garantir a segurança dos dados, os parâmetros são passados para a query utilizando métodos seguros oferecidos pela biblioteca *Npgsql*. Isso evita a execução de comandos maliciosos no banco de dados;
- e) **Envio de E-mail:** Após a inserção bem-sucedida dos dados no banco de dados, um e-mail é enviado para a pessoa cadastrada utilizando a classe `EnviaEmail`;
- f) **Feedback ou Tratamento de Erros:** Se a inserção for concluída com sucesso, a operação é considerada bem-sucedida e qualquer tratamento adicional pode ser realizado. Caso contrário, uma mensagem de erro é exibida ou o sistema pode lidar com a falha de outras maneiras, como registrando em logs ou notificando os usuários.

Esse é o fluxo geral do processo de salvamento no banco de dados conforme implementado no sistema fornecido. Na tabela 28 a seguir veremos os métodos principais da classe.

Tabela 28 – Conexão com banco de dados vendas

```

public void Inserir(PessoaController pessoa)
{
    using (NpgsqlConnection conexao = dbconexao.GetConnection() as NpgsqlConnection)
    {
        conexao.Execute(@"INSERT INTO visitante (NOME, IDADE, EMAIL, CEP, DATA,
Codigo)
VALUES (@Nome, @IdadeDb, @Email, @Cep, @Data, @Codigo)",
new
{
    pessoa.IdadeDb,
    pessoa.Nome,
    pessoa.Email,
    pessoa.Cep,
    pessoa.Codigo,
    Data = DateTime.Today
}
    )
}
public PessoaController BuscarPessoaPorEmail(string email)
{
    using (NpgsqlConnection conexao = dbconexao.GetConnection() as NpgsqlConnection)
    {
        return conexao.QueryFirstOrDefault<PessoaController>("SELECT * FROM visitante
WHERE email = @Email", new { Email = email });
    }
}
public PessoaController BuscarPessoaPorNomeEEEmail(PessoaController pessoa)
{
    using (var conexao = new Dbconexao())
    {
        var connection = conexao.GetConnection();
        var resultado = connection.QueryFirstOrDefault(
            "SELECT nome, idade, email, cep, data, codigo FROM public.visitante WHERE nome
= @Nome AND email = @Email",
            new { pessoa.Nome, pessoa.Email });
    }
}

```

```
if (resultado != null)
{
    pessoa.Nome = resultado.nome;
    pessoa.IdadeDb = resultado.idade;
    pessoa.Email = resultado.email;
    pessoa.Cep = resultado.cep;
    pessoa.Data = resultado.data;
    pessoa.Codigo = resultado.codigo;

    return pessoa;
}
```

Fonte: Autoria própria.

16. CONSIDERAÇÕES FINAIS

A elaboração do programa para o terminal interativo do museu foi um desafio estimulante e enriquecedor, permitindo a aplicação prática de conhecimentos e habilidades adquiridas ao longo do curso. A questão central da pesquisa, "Como desenvolver um terminal interativo que ofereça uma experiência educativa e envolvente sobre a primeira viagem do homem à Lua?", foi adequadamente respondida por meio da criação de uma ferramenta interativa e informativa.

A negociação do tema proposto foi crucial para alinhar as expectativas e assegurar que o projeto atendesse às necessidades educacionais e interativas do museu. Houve consenso sobre a importância de destacar a primeira viagem à Lua como um marco histórico e educativo.

Os objetivos de proporcionar uma experiência educativa e interativa foram plenamente atingidos. Os visitantes puderam acessar informações relevantes e participar de atividades que enriqueceram sua compreensão do tema. Os resultados obtidos mostraram-se concordantes com as expectativas iniciais, sendo bem recebidos pelos usuários, que consideraram a experiência educativa e envolvente.

A escolha da linguagem de programação C# e da plataforma Windows Forms revelou-se acertada, permitindo o desenvolvimento de um software robusto e eficiente, adequado para o ambiente do museu. A aplicação dos princípios de qualidade de software, como legibilidade, manutenibilidade e usabilidade, foi essencial para garantir a qualidade e eficiência do sistema desenvolvido.

O projeto destacou a importância de seguir as normas da LGPD para proteger a privacidade e segurança das informações dos visitantes. O registro de informações sobre a interação dos visitantes com o totem, como as obras visualizadas e as respostas à pesquisa, permitirá à equipe do museu avaliar o impacto da exposição e tomar decisões fundamentadas para futuros aprimoramentos.

A elaboração do programa seguiu uma abordagem iterativa e incremental, permitindo a apresentação de partes do sistema em períodos curtos. Isso favoreceu a personalização do software conforme as demandas dos usuários.

A contribuição das disciplinas para o trabalho desenvolvido foi fundamental para o sucesso do projeto, fornecendo conhecimentos técnicos e práticos em desenvolvimento de software, design de interface e conformidade com normas de proteção de dados. Disciplinas como Programação Orientada a Objetos I, Análise de

Sistemas Orientados a Objetos e gestão estratégica de recursos humanos desempenharam um papel crucial na realização deste trabalho.

Em resumo, o programa criado para o terminal interativo do museu constitui uma ferramenta significativa para proporcionar uma vivência enriquecedora aos visitantes, ajudando na divulgação e conservação da história da primeira missão tripulada à Lua. Além disso, a pesquisa de satisfação realizada permitirá melhorias futuras para a instituição, proporcionando um retorno valioso sobre a experiência dos visitantes. O trabalho evidenciou a relevância da utilização de tecnologias apropriadas e da adoção de métodos eficazes de desenvolvimento de software para alcançar as metas estabelecidas.

REFERÊNCIAS BIBLIOGRÁFICAS

Barbosa, Simone Diniz Junqueira; Silva, Bruno Santana da. *Interação humano-computador*. Rio de Janeiro: Elsevier Editora, 2010.

CHAGAS, Mário; NASCIMENTO JÚNIOR, José do (orgs.). **Subsídios para a Criação de Museus Municipais**. Rio de Janeiro: Ministério da Cultura / Instituto Brasileiro de Museu e Centros Culturais / Departamento de Processos Museais, 2009.

COMER, Douglas E. **Redes de Computadores e Internet**. 6. ed. Bookman Editora, 2016.

DE SOUZA, C. S., LEITE, J. C., Projeto de interface de usuário: **Perspectivas cognitivas e semióticas**. In **XIX Congresso Nacional da Sociedade Brasileira de Computação**, vol. 2, Edições Entrelugar (Rio de Janeiro, RJ, Brasil, 1999).

EMANUEL DINIZ SANTOS, A.; REIS DA SILVA, T.; GONÇALVES DOS SANTOS, F.; FONTINELE DE ALMEIDA, F.; RODRIGUES VALÉRIO, J.; HENRIQUE DA SILVA ARANHA, E; **Ensino de Redes de Computadores Mediado por Tecnologias Educacionais: um Mapeamento Sistemático da Literatura**. *Revista Novas Tecnologias na Educação*, Porto Alegre, v. 18, n. 1, 2020. DOI: 10.22456/1679-1916.106043. Disponível em: <https://seer.ufrgs.br/index.php/renote/article/view/106043>. Acesso em: 21 maio. 2024

FINLAY, J.; DIX, A.; ABOWD, G. D.; BEALE, R. **Human-computer Interaction**. Reino Unido: Pearson/Prentice-Hall, 2003.

GUIMARÃES, B.; GONÇALVES, C. E. **Introdução à Economia**. Rio de Janeiro: Campus, 2010.

HEUSER, C. A. **Projeto de Banco de Dados**. Edição. Local: Editora, Ano (1998).

INSTITUTO BRASILEIRO DE MUSEUS. MUSEUS EM NÚMEROS (IBRAM). **[Museus, Educação e Pesquisa]**. Brasília: Instituto Brasileiro de Museus, 2011.

ICOM BRASIL. **Nova definição de museu**. International council of museums, 2023. Disponível em: https://www.icom.org.br/wp-content/uploads/2021/02/ICOM-Define_Methodology_en.pdf. Acesso em: 25 de out. de 2023.

JOSÉ, Wilson. **Introdução à Economia**. Apostila. Universidade Federal de Ouro Preto, Centro de Educação Aberta e a Distância (CEAD), 2011. Disponível em: https://cead.ufop.br/professores/wilsonjose/Introducao_Economia2011/APOSTILA_Introducao_Economia.pdf. Acesso em: 24 de abril de 2024;

LOPES, Anita; GARCIA, Guto. **Introdução à programação**. Rio de Janeiro: Elsevier, 2002 - 15 Reimpressão.

MILANI, André. **PostgreSQL - Guia do Programador**. 1. ed. Novatec Editora, 2008.

MINISTÉRIO DO ESPORTE. **Lei Geral de Proteção de Dados Pessoais (LGPD)**. Gov.br, 2022. Disponível em: <https://www.gov.br/esporte/pt-br/acesso-a-informacao/lgpd>. Acesso em: 25 de out. de 2023.

NIELSEN, Jakob. **Usabilidade na web**. [S. l.]: Elsevier Brasil, 2007. 406 p.

OSBORNE, W. M.; CHIKOFSKY E. J. Fitting Pieces to the Maintenance Puzzle, IEEE Software, Vol. 7, No. 1, 1990. Disponível em: <https://www.proquest.com/docview/215834475?pq-origsite=gscholar&fromopenview=true&sourcetype=Scholarly%20Journals>. Acesso em 28 mar. 2024.

PEREIRA, S. R., & PAIVA, P. B. (2011). **A importância da Engenharia da Usabilidade para a Segurança de Sistemas Informatizados em Saúde**. Journal of Health Informatics, 3(3). Recuperado de <https://jhi.sbis.org.br/index.php/jhi-sbis/article/view/145>.

PREECE, J.; ROGERS, Y.; SHARP, H. **Designer de interação: além da interação homem-computador**. s.l.: Bookman, 2013.

PRODANOV, C. C.; FREITAS, E. C. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico**. Novo Hamburgo, RS: Feevale, 2013.

SERVIÇO FEDERAL DE PROCESSAMENTO DE DADOS (SERPRO). **Serpro e LGPD: segurança e inovação**. SERPRO, 2022. Disponível em: <https://www.serpro.gov.br/lgpd>. Acesso em: 25 de out. de 2023.

SIQUEIRA, Luciano. **Infraestrutura de redes**. 2. ed. Linux New Media do Brasil Editora Ltda., 2010.

MAXIMIANO, A. C. A. **Introdução à administração**. São Paulo: Atlas, 2006

NIELSEN, Jakob. **Usabilidade na web**. [S. l.]: Elsevier Brasil, 2007. 406 p.

RICHARDS, G. & MARQUES, L. (2012). **"Exploring creative tourism: editors introduction"**. In: Journal of Tourism Consumption and Practice. Volume 04, n. 02, 01-11.

ROB, Peter; CORONEL, Carlos. **Sistemas de Banco de Dados: Projeto, Implementação e Gerenciamento**. 4. ed. São Paulo: Editora, 2006.

ROTMAN, M. & CASTELLS, A. N. G. (2007). **"Patrimônio e cultura: processos de politização, mercantilização e construção de identidades"**. In: M. F. L. Filho. Antropologia e patrimônio cultural: diálogos e desafios contemporâneos. (pp. 57-79). Blumenau: Nova Letra.

RUSSO, R. de F. S. G.; SILVA, L. F. da, & LARIEIRA, C. L. C. (2021). **Do manifesto ágil à agilidade organizacional**. Revista de Gestão e Projetos (GeP), 12(1), 1-10

SABADIN, N. M. **Interação Humano-Computador**. Uniasselvi, 2016.

SILVA, Bruno Santana da; BARBOSA, Simone Diniz Junqueira. **Interação Humano Computador: Projetando a Experiência Perfeita**. Rio de Janeiro: Campus, 2010.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

WINCKLER, Marco; PIMENTA, Marcelo. Avaliação de usabilidade de sites Web. Research Gate, 2014. Disponível em: https://www.researchgate.net/profile/Marco-Winckler-3/publication/228816116_Avaliacao_de_usabilidade_de_sites_Web/links/02bfe510a614de7879000000/Avaliacao-de-usabilidade-de-sites-Web.pdf. Acesso em: [11/05/2024].

WOOD JR., T. **Mudança organizacional: aprofundando temas atuais em administração de empresas**. São Paulo: Atlas, 1995.

APÊNDICE A - MANUAL DE INSTALAÇÃO E DESINSTALAÇÃO

Este manual foi elaborado com o objetivo de orientar os usuários, independentemente de seu nível de familiaridade com procedimentos técnicos, por meio de instruções claras e passo a passo. Ao seguir as diretrizes apresentadas neste manual, os usuários poderão instalar e desinstalar o software de forma adequada, aumentando assim sua utilidade e tornando mais fácil sua gestão no ambiente do sistema operacional.

Necessário seguir os seguintes passos para garantir uma instalação bem-sucedida:

- a) Baixar o arquivo de instalação;
- b) Executar o arquivo de instalação;
- c) Aceitar os termos de licença;
- d) Selecionar o local de instalação;
- e) Concluir a instalação;
- f) Iniciar o software após a instalação.

Após a instalação bem-sucedida do software, o próximo passo é configurá-lo para uso adequado. O administrador do sistema é responsável por essa configuração inicial.

Criação de Conta de Usuário

O administrador deve criar uma conta de usuário para ir interagir com o sistema. Serão solicitadas informações básicas, como nome, e-mail e senha. Essas credenciais serão usadas pelos usuários para acessar o sistema posteriormente.

Ativação das Opções de Obter Relatórios

Após criar as contas de usuário, o administrador deve ativar as opções de obtenção de relatórios.

Isso permite que o sistema gere relatórios detalhados sobre as atividades dos usuários, como compra de ingressos, interação com as obras do museu e participação

na pesquisa de satisfação. Esses relatórios serão úteis para monitorar o desempenho do sistema e tomar decisões estratégicas.

Com o software devidamente configurado, os usuários podem começar a utilizar suas funcionalidades principais. Abaixo estão as instruções básicas para usar cada recurso:

Compra e Venda de Ingressos

Para comprar ingressos, o usuário deve fazer login em sua conta e acessar a opção "Compra de Ingressos". Lá, ele poderá visualizar os eventos disponíveis, selecionar a quantidade desejada de ingressos e prosseguir com o pagamento. Após a compra bem-sucedida, os ingressos serão automaticamente gerados e enviados por e-mail para o usuário.

Além disso, será enviado um token exclusivo para cada usuário, que será utilizado para participar da avaliação das obras e da pesquisa de satisfação. Cada usuário receberá seu próprio token por e-mail após a conclusão da compra dos ingressos.

Avaliação das Obras do Museu

Os usuários podem interagir com as obras do museu navegando pela galeria virtual. Ao clicar em uma obra, Eles terão a opção de participar do quiz, respondendo às perguntas sobre as obras. Essas avaliações serão registradas pelo sistema e poderão ser visualizadas posteriormente pelo administrador.

Pesquisa de Satisfação

Ao final da visita ao museu, os usuários serão convidados a participar de uma pesquisa de satisfação. Eles podem acessar a pesquisa por meio de um token enviado por e-mail. A pesquisa consiste em perguntas sobre a experiência do usuário durante a visita e fornecer feedback sobre o museu.

Para desinstalar o software

No Windows, abra o Painel de Controle. Em outros sistemas operacionais, utilize o método apropriado para acessar as configurações de desinstalação.

- a) Localize o software na lista de programas instalados;
- b) Selecione o software que deseja desinstalar;
- c) Siga as instruções na tela para concluir o processo de desinstalação;
- d) Certifique-se de que todas as instâncias do software estejam fechadas antes de prosseguir com a desinstalação.

Informações de Suporte

Em caso de dúvidas ou problemas durante o processo de instalação ou desinstalação, a nossa equipe estará disponível para fornecer suporte. A empresa estará pronta para ajudar e oferecer assistência adicional sempre que necessário. Os usuários podem entrar em contato através do seguinte meio:

- a) **E-mail:** futuretech@gmail.com;
- b) **Telefone:** +55 15 0101-0011.

Estamos comprometidos em garantir uma experiência positiva para nossos usuários e resolver quaisquer questões que possam surgir durante o uso do software.

APÊNDICE B – TAP (TERMO DE ABERTURA DE PROJETO)

Identificação do projeto

Nome do Projeto	Projeto de um totem de museu multitemático		
Responsável pelo projeto	Líder de projeto	Equipe básica	
Prof. Especialista. Reverdan A. Springer	Lucas Matheus Campestrini Cerione	Lucas Matheus Campestrini Cerione	Documentação
		Maria da Gloria Ayumi Muraki Naruse	Documentação
		Paulo Henrique Mendes Paiva	Desenvolvedor
		Nicolas Kenzo Carvalho Onishi	Desenvolvedor

Descrição do projeto

Justificativa do Projeto (por quê?)	Objetivo do Projeto (para que será feito)	Objeto do Projeto (o que será feito)
Uma organização sem fins lucrativos pretende criar um museu multitemático. Com o tema exibido será a primeira viagem do homem que permanecerá em exibição dependendo da reação do público visitante as obras apresentadas. Foi criado o projeto para criação de sistema de um totem	Para melhor funcionamento e controle do museu, para facilitar serviços e tornar a visita mais elaborada e atrair mais o público.	Será feito um programa para apresentação de obras onde cada visitante será motivado a responder um questionário que o resultado será demonstrado para a pessoa e com isso irá gerar um relatório para tomar a decisão de melhorias ou encerramento do evento.

interativo para facilitar os visitantes do museu.		
---	--	--

Estrutura analítica do projeto

Os produtos correspondem às entregas que serão feitas com vistas à consolidação do trabalho. Em outras palavras, o somatório dos produtos corresponde ao escopo do projeto.

Etapas	Produtos	Valor estimado	Prazo de entrega
1	Infraestrutura do museu e redes	R\$ 4.000	3 meses
2	Desenvolvimento do programa em C# do museu e Windows Forms	R\$ 0,00	3 meses
3	Infraestrutura do Totem	R\$ 36.309,02	3 meses
4	Documentação do projeto e impressão	R\$ 100,00	3 meses
VALOR E PRAZO FINAIS		R\$ 40.409,02	3 meses

APROVAÇÃO DO TERMO DE ABERTURA		
Aprovado por	Assinatura	Data
		__/__/__

INFORMAÇÕES PARA A GESTÃO DO PORTFÓLIO

As informações solicitadas na sequência são necessárias para avaliação do portfólio de projetos da organização. Responda as questões que segue com base em sua expectativa para o projeto.

Alinhamento estratégico

Considera uma escala de contribuição do projeto para alcance do objetivo estratégico. A análise estratégica é feita pela ponderação do nível de contribuição do projeto para um determinado objetivo estratégico: nenhuma contribuição; contribuição indireta e forte contribuição para o objetivo estratégico. Avalie o projeto nesse critério marcando sua opção com um “X”.

OBJETIVO ESTRATÉGICO		FORTE	INDIRETA
PERSPECTIVA	OBJETIVO 1: Estimular parcerias entre as Instituições de Pesquisa e Empresas por meio de projetos PD&I;		X
	OBJETIVO 2: Solução de problemas demandados pela sociedade e gestão do estabelecimento;	X	
	OBJETIVO 3: Estabelecer pesquisas de satisfação em cada obra para melhor desenvolvimento futuro	X	

GUT (Gravidade, Urgência e Tendência)

Trata-se de um coeficiente que combina três indicadores: gravidade, urgência e tendência. Gravidade representa o impacto do problema na organização, pode estar ligado a questões legais, recursos ou mesmo da atividade fim. A urgência relaciona-se ao tempo de resposta ao problema, ou seja, projetos urgentes requerem ação e decisão imediata e têm maior prioridade do que projetos não urgentes. Tendência avalia o nível de agravamento ou não do problema com o passar do tempo, ou seja, se nada for feito a tendência é de crescimento, redução ou desaparecimento do problema? Avalie o projeto nesse critério marcando sua opção com um “X”.

NOTA	GRAVIDADE		URGÊNCIA		TENDÊNCIA	
5		EXTREMAMENTE GRAVE		PRECISA DE AÇÃO IMEDIATA		IRÁ PIORAR RAPIDAMENTE
4		MUITO GRAVE		É URGENTE	X	IRÁ PIORAR EM POUCO TEMPO
3		GRAVE	X	O MAIS RÁPIDO POSSÍVEL		IRÁ PIORAR
2		POUCO GRAVE		POUCO URGENTE		IRÁ PIORAR A LONGO PRAZO
1	X	SEM GRAVIDADE		PODE ESPERAR		NÃO IRÁ MUDAR

Comprometimento das partes interessadas

Avalia o nível de comprometimento das partes interessadas com o projeto. Quanto mais alto é o comprometimento com o projeto, mais prioritário o projeto se torna. O comprometimento é avaliado de forma segmentada nos seguintes grupos: usuário final (beneficiário direto); patrocinador do projeto; unidades da organização; equipe do projeto; e gestor (líder) do projeto.

A análise é feita pela ponderação do nível de comprometimento das partes interessadas, com base na seguinte escala: nenhum comprometimento; baixo comprometimento; e alto comprometimento. Avalie o projeto nesse critério marcando uma opção com um “X”.

PARTES INTERESSADAS	MUITO ALTO	ALTO	MODERADO	BAIXO	MUITO BAIXO
USUÁRIO FINAL (BENEFICIÁRIO DIRETO)		X			
UNIDADES DA ORGANIZAÇÃO					X
EQUIPE DO PROJETO			X		
GESTOR (LÍDER) DO PROJETO	X				

Conhecimento técnico

Avalia se o conhecimento técnico disponível na organização é suficiente para realizar o projeto. Quanto maior é o conhecimento técnico disponível, maior será a facilidade de se realizar determinado projeto e, conseqüentemente, menor será o “custo” de sua realização. Avalie o projeto nesse critério marcando uma opção com um “X”.

FAIXA DE AVALIAÇÃO	
Pleno: conhecimento e experiência disponíveis na organização para realizar todas as ações do projeto	
Alto: conhecimento e experiência disponíveis na organização para realizar as ações críticas do projeto	

Moderado: conhecimento e experiência disponíveis na organização para realizar algumas as ações do projeto	X
Baixo: conhecimento e experiência insuficientes na organização para realizar as ações do projeto	
Inexistente: não existe nenhum tipo de conhecimento ou experiência anterior na organização para realizar as ações	

Riscos

Um risco é um evento incerto ou condicionado que, se acontecer, pode impactar negativamente ou positivamente os objetivos do projeto (ações e resultados). A valoração do risco do projeto é dada pela multiplicação entre a probabilidade de ocorrência do risco e seu impacto no projeto. Os impactos serão dimensionados na escala: alto / significativo / moderado / baixo / insignificante. A probabilidade variará da seguinte forma: quase certo / provável / possível / improvável / remoto. A combinação entre as duas variáveis gerará a classificação segundo os tipos: risco alto, risco significativo, risco moderado e risco baixo. De forma a criar uma base comum de comparação entre dos riscos envolvidos nos projetos foi definido um conjunto eventos para que sejam analisados o impacto e a probabilidade de ocorrência. Assinale com um “x” a probabilidade e impacto para os riscos do quadro que segue com base na escala apresentada.

RISCO 01: Descontinuidade do financiamento do projeto		
Trata-se da interrupção total ou parcial do fluxo de recursos financeiros destinados ao projeto. As causas desse evento podem ser variadas, ou seja, contingenciamentos de recursos organizacionais, perda de prioridade do projeto, ineficiência na execução dos recursos, dentre outros. A consequência imediata desse risco pode ser alterações em seu escopo ou qualidade ou mesmo atrasos nas entregas.		
PROBABILIDADE		QUASE CERTO
	X	PROVÁVEL
		POSSÍVEL
		IMPROVÁVEL
		REMOTO
IMPACTO	X	Alto: alterações fatais no escopo, qualidade ou tempo do projeto;
		Significativo: as alterações no projeto são severas, mas podem ser aceitas com nova validação em seu escopo ou prazos;
		Moderado: o impacto pode ser amenizado ampliando significativamente o tempo do projeto;
		Baixo: as consequências do risco podem ser absorvidas pela equipe de gerenciamento do projeto;
		Insignificante: não envolve qualquer impacto que mereça destaque.

Análise da intensidade de gestão

A decisão sobre quais projetos deverão compor a carteira de projetos institucional deve considerar, também, a complexidade de implantação dessas iniciativas, intensidade de gestão. Deve-se mensurar o esforço de gestão envolvido na execução dos projetos por meio de critérios básicos e da expectativa sobre cada empreendimento. Avalie o projeto nesse critério marcando uma opção com um “x”.

FATOR DE AVALIAÇÃO	CRITÉRIO	ANÁLISE	
NECESSIDADE DE ARTICULAÇÃO	Equipe do projeto		Nível 01: Poucas pessoas envolvidas no projeto: gestor e poucos executores da mesma unidade.
		X	Nível 02: Moderado envolvimento de pessoas: gestor e muitos executores da mesma unidade.
			Nível 03: Grande envolvimento de pessoas: gestor e outros executores das unidades distintas.
	Fornecedores		Nível 01: Não envolve contratação de fornecedores.
			Nível 02: Baixa demanda por contratação de fornecedores.
		X	Nível 03: Grande demanda por contratação de fornecedores.
	Parceiros	X	Nível 01: Não existe o envolvimento de parceiros na execução direta de ações.
			Nível 02: Baixo envolvimento de parceiros na execução direta de ações.
			Nível 03: Grande envolvimento de parceiros na execução direta de ações.
	Beneficiários	X	Nível 01: Beneficiários diretos são da própria instituição.
			Nível 02: Facilidade de articulação com os beneficiários diretos.
			Nível 03: Dificuldade de articulação com os beneficiários diretos.
APOORTE DE RECURSOS	Fonte orçamentária	X	Nível 01: Sem uso de fonte orçamentária.
			Nível 02: Utilização de uma fonte orçamentária.
			Nível 03: Utilização de várias fontes orçamentárias.
	Execução orçamentária	X	Nível 01: Facilidade de execução orçamentária.
			Nível 02: Dificuldade de execução orçamentária.
			Nível 03: Muita dificuldade de execução orçamentária.
	Especialidade		Nível 01: As ações não demandam por especialidades técnicas críticas.
			Nível 02: Demandas pontuais por especialidades técnicas críticas.
		X	Nível 03: Grande dependência de RH qualificados.

APÊNDICE C – IMAGENS DO CRONOGRAMA DO PROJETO

CRONOGRAMA

Responsaveis:

Início do projeto: 27/02/202

Semana de exibição

Lucas cerione, Paulo Paiva, Ayumi, Nicolas Konish

[illegible]

CRONOGRAMA

Responsaveis:

Início do projeto: 27/02/2024

Semana de exibição

Lucas cerione, Paulo Paiva, Ayumi, Nicolas Konish

[illegible]

CRONOGRAMA

Responsaveis:

Início do projeto: 27/02/2024

Semana de exibição:

▲▼

Lucas cerione, Paulo Paiva, Avumi, Nicolas Konishi

[illegible]

CRONOGRAMA

Responsaveis:

Início do projeto: 27/02/2024

Semana de exibição:

▲▼

Lucas cerione, Paulo Paiva, Ayumi, Nicolas Konishi

[illegible]

CRONOGRAMA

Responsaveis:

Início do projeto: 27/02/2024

Lucas cerione, Paulo Paiva, Ayumi, Nicolas Konishi

Semana de exibição:

9

[illegible]

APÊNDICE D – CÓDIGO FONTE DO PROGRAMA DO QUESTIONÁRIO

```
namespace TotemPIMApresentacao.View
{
    public partial class Menu : Form
    {
        public Menu()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
        }
        private void btnObra01_Click(object sender, EventArgs e)
        {
            Apollo_11 apollo11 = new Apollo_11();
            apollo11.Show();
        }
        private void btnObra02_Click(object sender, EventArgs e)
        {
            ObraDocumentos obraDocumentos = new ObraDocumentos();
            obraDocumentos.Show();
        }
        private void btnObra03_Click(object sender, EventArgs e)
        {
            ObraEquip obraEquip = new ObraEquip();
            obraEquip.Show();
        }
        private void btnObra04_Click(object sender, EventArgs e)
        {
            ObraFotos obraFotos = new ObraFotos();
            obraFotos.Show();
        }
        private void btnObra05_Click(object sender, EventArgs e)
        {
            ObraHistoria obraHistoria = new ObraHistoria();
            obraHistoria.Show();
        }
        private void btnObra06_Click(object sender, EventArgs e)
        {
            ObraReplica obraReplica = new ObraReplica();
            obraReplica.Show();
        }
        private void btnObra07_Click(object sender, EventArgs e)
        {
            ObraSuits obraSuits = new ObraSuits();
            obraSuits.Show();
        }
    }
}
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class ResultadoFormSuit : Form
    {
        private ContadorRespostas contadorRespostas;
        public ResultadoFormSuit(ContadorRespostas contadorRespostas)
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            this.contadorRespostas = contadorRespostas;
            ExibirResultados();
        }
        private void ExibirResultados()
```

```

        {
            contadorRespostas.SalvarResposta();
            MediaAcertos mediaAcertos = new MediaAcertos();
            mediaAcertos.MediaAcertosPorObra(contadorRespostas);
            lblResultado.Text = $"Você acertou {contadorRespostas.RespostasCorretas} de 5 perguntas";
        }

        private void btnObra_Click(object sender, EventArgs e)
        {
            ObraSuits ObraSuits = new ObraSuits();
            ObraSuits.ShowDialog();
            this.Hide();
        }
    }
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class QuestionarioSuits : Form
    {
        private PerguntasTrajesEspaciais perguntas;

        public QuestionarioSuits(ContadorRespostas contadorRespostas, string codigoUsuario)
        {
            InitializeComponent();
            perguntas = new PerguntasTrajesEspaciais(this, contadorRespostas, codigoUsuario, "Suits");
            this.WindowState = FormWindowState.Maximized;
            AtualizaPergunta();
        }

        private void AtualizaPergunta()
        {
            perguntas.AtualizarPergunta(lblPergunta, lblNumeroPergunta, new Button[]
            {
                btnAlternativaA,
                btnAlternativaB,
                btnAlternativaC,
                btnAlternativaD,
                btnAlternativaE
            });
        }

        private void ProximaPergunta()
        {
            perguntas.ProximaPergunta();
            AtualizaPergunta();
        }

        private void btnAlternativaA_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("A");
            ProximaPergunta();
        }

        private void btnAlternativaB_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("B");
            ProximaPergunta();
        }

        private void btnAlternativaC_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("C");
            ProximaPergunta();
        }

        private void btnAlternativaD_Click(object sender, EventArgs e)

```

```

        {
            perguntas.VerificarResposta("D");
            ProximaPergunta();
        }

        private void btnAlternativaE_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("E");
            ProximaPergunta();
        }
    }
}
namespace TotemPIMApresentacao.View
{
    public partial class ObraSuits : Form
    {
        public ObraSuits()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
        }
        private void Atalho_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.V)
                this.Close();
        }
        private void BtnTesteConhecimento_Click(object sender, EventArgs e)
        {
            LoginSuits loginSuits = new LoginSuits();
            loginSuits.ShowDialog();
            this.Hide();
        }
        private void Atalho_Load(object sender, EventArgs e)
        {
            this.KeyPreview = true;
            this.KeyDown += Atalho_KeyDown;
        }
    }
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;
namespace TotemPIMApresentacao.View
{
    public partial class LoginSuits : Form
    {
        private PessoaModel pessoaModel;
        private PessoaController pessoa;
        private Teclado teclado;
        public string LoginStatus { get; private set; }

        public LoginSuits()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            pessoa = new PessoaController();
            pessoaModel = new PessoaModel();
        }

        private void TextBox1_Click(object sender, EventArgs e)
        {
            // Verifica se o teclado ainda não foi instanciado
            if (teclado == null)
            {

```

```

        // Cria uma nova instância do teclado
        teclado = new Teclado();
    }

    // Define a TextBox clicada como o TextBox de destino do teclado
    teclado.SetTargetTextBox(sender as TextBox);

    teclado.Show();
}
private void Atalho_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Control && e.KeyCode == Keys.V)
        this.Close();
}

private void BtnEntrar_Click(object sender, EventArgs e)
{
    pessoa.Codigo = txbCodigoVisitante.Text;
    pessoaModel.Login(pessoa);

    if (pessoaModel.Mensagem.Equals(""))
    {
        string codigoUsuario = pessoa.Codigo;
        ContadorRespostas contadorRespostas = new ContadorRespostas(codigoUsuario);

        QuestionarioSuits questionarioSuits = new QuestionarioSuits(contadorRespostas, codigoUsuario);
        questionarioSuits.ShowDialog();
        this.Close();
    }
    else
    {
        MessageBox.Show(pessoaModel.Mensagem);
    }
    LimparCampos();
}

private void LimparCampos()
{
    txbCodigoVisitante.Text = string.Empty;
}

private void Atalho_Load(object sender, EventArgs e)
{
    this.KeyPreview = true;
    this.KeyDown += Atalho_KeyDown;
}
}
}
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class ResultadoFormReplica : Form
    {
        private ContadorRespostas contadorRespostas;
        public ResultadoFormReplica(ContadorRespostas contadorRespostas)
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            this.contadorRespostas = contadorRespostas;
            ExibirResultados();
        }
        private void ExibirResultados()
        {
            contadorRespostas.SalvarResposta();
            MediaAcertos mediaAcertos = new MediaAcertos();
            mediaAcertos.MediaAcertosPorObra(contadorRespostas);
        }
    }
}

```

```

        lblResultado.Text = $"Você acertou {contadorRespostas.RespostasCorretas} de 5 perguntas";
    }
    private void btnObra_Click(object sender, EventArgs e)
    {
        ObraReplica ObraSuits = new ObraReplica();
        ObraSuits.ShowDialog();
        this.Hide();
    }
}
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class QuestionarioReplica : Form
    {
        private PerguntasReplica perguntas;

        public QuestionarioReplica(ContadorRespostas contadorRespostas, string codigoUsuario)
        {
            InitializeComponent();
            perguntas = new PerguntasReplica(this, contadorRespostas, codigoUsuario, "Replica");
            this.WindowState = FormWindowState.Maximized;
            AtualizaPergunta();
        }
        private void AtualizaPergunta()
        {
            perguntas.AtualizarPergunta(lblPergunta, lblNumeroPergunta, new Button[]
            {
                btnAlternativaA,
                btnAlternativaB,
                btnAlternativaC,
                btnAlternativaD,
                btnAlternativaE
            });
        }
        private void ProximaPergunta()
        {
            perguntas.ProximaPergunta();
            AtualizaPergunta();
        }

        private void btnAlternativaA_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("A");
            ProximaPergunta();
        }

        private void btnAlternativaB_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("B");
            ProximaPergunta();
        }

        private void btnAlternativaC_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("C");
            ProximaPergunta();
        }

        private void btnAlternativaD_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("D");
            ProximaPergunta();
        }
    }
}

```



```

        private void btnAlternativaE_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("E");
            ProximaPergunta();
        }
    }
}
namespace TotemPIMApresentacao.View
{
    public partial class ObraReplica : Form
    {
        public ObraReplica()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
        }

        private void Atalho_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.V)
                this.Close();
        }

        private void BtnTesteConhecimento_Click(object sender, EventArgs e)
        {
            LoginReplica loginReplica = new LoginReplica();
            loginReplica.ShowDialog();
            this.Hide();
        }

        private void Atalho_Load_1(object sender, EventArgs e)
        {
            this.KeyPreview = true;
            this.KeyDown += Atalho_KeyDown;
        }
    }
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class LoginReplica : Form
    {
        private PessoaModel pessoaModel;
        private PessoaController pessoa;
        private Teclado teclado;
        public string LoginStatus { get; private set; }

        public LoginReplica()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            pessoa = new PessoaController();
            pessoaModel = new PessoaModel();
        }

        private void TextBox1_Click(object sender, EventArgs e)
        {
            // Verifica se o teclado ainda não foi instanciado

```

```

        if (teclado == null)
        {
            // Cria uma nova instância do teclado
            teclado = new Teclado();
        }

        // Define a TextBox clicada como o TextBox de destino do teclado
        teclado.SetTargetTextBox(sender as TextBox);

        teclado.Show();
    }
    private void Atalho_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.Control && e.KeyCode == Keys.V)
            this.Close();
    }

    private void BtnEntrar_Click(object sender, EventArgs e)
    {
        pessoa.Codigo = txbCodigoVisitante.Text;
        pessoaModel.Login(pessoa);

        if (pessoaModel.Mensagem.Equals(""))
        {
            string codigoUsuario = pessoa.Codigo;
            ContadorRespostas contadorRespostas = new ContadorRespostas(codigoUsuario);

            QuestionarioReplica questionarioReplica = new QuestionarioReplica(contadorRespostas,
codigoUsuario);
            questionarioReplica.ShowDialog();
            this.Close();
        }
        else
        {
            MessageBox.Show(pessoaModel.Mensagem);
        }
        LimparCampos();
    }

    private void LimparCampos()
    {
        txbCodigoVisitante.Text = string.Empty;
    }

    private void Atalho_Load(object sender, EventArgs e)
    {
        this.KeyPreview = true;
        this.KeyDown += Atalho_KeyDown;
    }
}
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class LoginHistoria : Form
    {
        private PessoaModel pessoaModel;
        private PessoaController pessoa;
        private Teclado teclado;
        public string LoginStatus { get; private set; }
        public LoginHistoria()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;

```

```

        pessoa = new PessoaController();
        pessoaModel = new PessoaModel();
    }

    private void TextBox1_Click(object sender, EventArgs e)
    {
        // Verifica se o teclado ainda não foi instanciado
        if (teclado == null)
        {
            // Cria uma nova instância do teclado
            teclado = new Teclado();
        }

        // Define a TextBox clicada como o TextBox de destino do teclado
        teclado.SetTargetTextBox(sender as TextBox);

        teclado.Show();
    }

    private void Atalho_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.Control && e.KeyCode == Keys.V)
            this.Close();
    }

    private void BtnEntrar_Click(object sender, EventArgs e)
    {
        pessoa.Codigo = txbCodigoVisitante.Text;
        pessoaModel.Login(pessoa);

        if (pessoaModel.Mensagem.Equals(""))
        {
            string codigoUsuario = pessoa.Codigo;
            ContadorRespostas contadorRespostas = new ContadorRespostas(codigoUsuario);

            QuestionarioHistoria questionarioHistoria = new QuestionarioHistoria(contadorRespostas,
codigoUsuario);
            questionarioHistoria.ShowDialog();
            this.Close();
        }
        else
        {
            MessageBox.Show(pessoaModel.Mensagem);
        }
        LimparCampos();
    }

    private void LimparCampos()
    {
        txbCodigoVisitante.Text = string.Empty;
    }

    private void Atalho_Load(object sender, EventArgs e)
    {
        this.KeyPreview = true;
        this.KeyDown += Atalho_KeyDown;
    }
}
}
namespace TotemPIMApresentacao.View
{
    public partial class ObraHistoria : Form
    {
        public ObraHistoria()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
        }
    }
}

```

```

private void Atalho_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Control && e.KeyCode == Keys.V)
        this.Close();
}

private void BtnTesteConhecimento_Click(object sender, EventArgs e)
{
    LoginHistoria loginHistoria = new LoginHistoria();
    loginHistoria.ShowDialog();
    this.Hide();
}

private void Atalho_Load_1(object sender, EventArgs e)
{
    this.KeyPreview = true;
    this.KeyDown += Atalho_KeyDown;
}
}
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class QuestionarioHistoria : Form
    {
        private PerguntasHistoria perguntas;

        public QuestionarioHistoria(ContadorRespostas contadorRespostas, string codigoUsuario)
        {
            InitializeComponent();
            perguntas = new PerguntasHistoria(this, contadorRespostas, codigoUsuario, "Historia");
            this.WindowState = FormWindowState.Maximized;
            AtualizaPergunta();
        }

        private void AtualizaPergunta()
        {
            perguntas.AtualizarPergunta(lblPergunta, lblNumeroPergunta, new Button[]
            {
                btnAlternativaA,
                btnAlternativaB,
                btnAlternativaC,
                btnAlternativaD,
                btnAlternativaE
            });
        }

        private void ProximaPergunta()
        {
            perguntas.ProximaPergunta();
            AtualizaPergunta();
        }

        private void btnAlternativaA_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("A");
            ProximaPergunta();
        }

        private void btnAlternativaB_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("B");
            ProximaPergunta();
        }
    }
}

```

```

        private void btnAlternativaC_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("C");
            ProximaPergunta();
        }

        private void btnAlternativaD_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("D");
            ProximaPergunta();
        }

        private void btnAlternativaE_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("E");
            ProximaPergunta();
        }
    }
}
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class ResultadoFormHistoria : Form
    {
        private ContadorRespostas contadorRespostas;
        public ResultadoFormHistoria(ContadorRespostas contadorRespostas)
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            this.contadorRespostas = contadorRespostas; // Atribua a instância fornecida ao campo
            contadorRespostas
                ExibirResultados();
        }

        private void ExibirResultados()
        {
            contadorRespostas.SalvarResposta();
            MediaAcertos mediaAcertos = new MediaAcertos();
            mediaAcertos.MediaAcertosPorObra(contadorRespostas);
            lblResultado.Text = $"Você acertou {contadorRespostas.RespostasCorretas} de 5 perguntas"; //
            Atualize o texto da lblResultado com o número atual de respostas corretas
        }

        private void btnObra_Click(object sender, EventArgs e)
        {
            ObraHistoria ObraSuits = new ObraHistoria();
            ObraSuits.ShowDialog();
            this.Hide();
        }
    }
}

using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class LoginFotos : Form
    {
        private PessoaModel pessoaModel;
        private PessoaController pessoa;
        private Teclado teclado;
        public string LoginStatus { get; private set; }
    }
}

```

```

public LoginFotos()
{
    InitializeComponent();
    this.WindowState = FormWindowState.Maximized;
    pessoa = new PessoaController();
    pessoaModel = new PessoaModel();
}

private void TextBox1_Click(object sender, EventArgs e)
{
    // Verifica se o teclado ainda não foi instanciado
    if (teclado == null)
    {
        // Cria uma nova instância do teclado
        teclado = new Teclado();
    }

    // Define a TextBox clicada como o TextBox de destino do teclado
    teclado.SetTargetTextBox(sender as TextBox);

    teclado.Show();
}

private void Atalho_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Control && e.KeyCode == Keys.V)
        this.Close();
}

private void BtnEntrar_Click(object sender, EventArgs e)
{
    pessoa.Codigo = txbCodigoVisitante.Text;
    pessoaModel.Login(pessoa);

    if (pessoaModel.Mensagem.Equals(""))
    {
        string codigoUsuario = pessoa.Codigo;
        ContadorRespostas contadorRespostas = new ContadorRespostas(codigoUsuario);

        QuestionarioFotos questionarioFotos = new QuestionarioFotos(contadorRespostas, codigoUsuario);
        questionarioFotos.ShowDialog();
        this.Close();
    }
    else
    {
        MessageBox.Show(pessoaModel.Mensagem);
    }
    LimparCampos();
}

private void LimparCampos()
{
    txbCodigoVisitante.Text = string.Empty;
}

private void Atalho_Load(object sender, EventArgs e)
{
    this.KeyPreview = true;
    this.KeyDown += Atalho_KeyDown;
}
}
}

namespace TotemPIMApresentacao.View
{
    public partial class ObraFotos : Form
    {

```

```

public ObraFotos()
{
    InitializeComponent();
    this.WindowState = FormWindowState.Maximized;
}

private void Atalho_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Control && e.KeyCode == Keys.V)
        this.Close();
}
private void BtnTesteConhecimento_Click(object sender, EventArgs e)
{
    LoginFotos loginFotos = new LoginFotos();
    loginFotos.ShowDialog();
    this.Hide();
}
private void Atalho_Load_1(object sender, EventArgs e)
{
    this.KeyPreview = true;
    this.KeyDown += Atalho_KeyDown;
}

}
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class QuestionarioFotos : Form
    {
        private PerguntasFotos perguntas;

        public QuestionarioFotos(ContadorRespostas contadorRespostas, string codigoUsuario)
        {
            InitializeComponent();
            perguntas = new PerguntasFotos(this, contadorRespostas, codigoUsuario, "Fotos");
            this.WindowState = FormWindowState.Maximized;
            AtualizaPergunta();
        }
        private void AtualizaPergunta()
        {
            perguntas.AtualizarPergunta(lblPergunta, lblNumeroPergunta, new Button[]
            {
                btnAlternativaA,
                btnAlternativaB,
                btnAlternativaC,
                btnAlternativaD,
                btnAlternativaE
            });
        }
        private void ProximaPergunta()
        {
            perguntas.ProximaPergunta();
            AtualizaPergunta();
        }

        private void btnAlternativaA_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("A");
            ProximaPergunta();
        }
    }
}

```

```

        private void btnAlternativaB_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("B");
            ProximaPergunta();
        }

        private void btnAlternativaC_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("C");
            ProximaPergunta();
        }

        private void btnAlternativaD_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("D");
            ProximaPergunta();
        }

        private void btnAlternativaE_Click(object sender, EventArgs e)
        {
            perguntas.VerificarResposta("E");
            ProximaPergunta();
        }
    }
}
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class ResultadoFormFotos : Form
    {
        private ContadorRespostas contadorRespostas;
        public ResultadoFormFotos(ContadorRespostas contadorRespostas)
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            this.contadorRespostas = contadorRespostas; // Atribua a instância fornecida ao campo
            contadorRespostas
            ExibirResultados();
        }

        private void ExibirResultados()
        {
            contadorRespostas.SalvarResposta();
            MediaAcertos mediaAcertos = new MediaAcertos();
            mediaAcertos.MediaAcertosPorObra(contadorRespostas);
            lblResultado.Text = $"Você acertou {contadorRespostas.RespostasCorretas} de 5 perguntas"; //
            Atualize o texto da lblResultado com o número atual de respostas corretas
        }

        private void btnObra_Click(object sender, EventArgs e)
        {
            ObraFotos ObraSuits = new ObraFotos();
            ObraSuits.ShowDialog();
            this.Hide();
        }
    }
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View

```



```

{
    public partial class LoginEquip : Form
    {
        private PessoaModel pessoaModel;
        private PessoaController pessoa;
        private Teclado teclado;
        public string LoginStatus { get; private set; }
        public LoginEquip()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            pessoa = new PessoaController();
            pessoaModel = new PessoaModel();
        }

        private void TextBox1_Click(object sender, EventArgs e)
        {
            // Verifica se o teclado ainda não foi instanciado
            if (teclado == null)
            {
                // Cria uma nova instância do teclado
                teclado = new Teclado();
            }

            // Define a TextBox clicada como o TextBox de destino do teclado
            teclado.SetTargetTextBox(sender as TextBox);

            teclado.Show();
        }

        private void Atalho_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.V)
                this.Close();
        }

        private void BtnEntrar_Click(object sender, EventArgs e)
        {
            pessoa.Codigo = txbCodigoVisitante.Text;
            pessoaModel.Login(pessoa);

            if (pessoaModel.Mensagem.Equals(""))
            {
                string codigoUsuario = pessoa.Codigo;
                ContadorRespostas contadorRespostas = new ContadorRespostas(codigoUsuario);

                QuestionarioEquip questionarioEquip = new QuestionarioEquip(contadorRespostas, codigoUsuario);
                questionarioEquip.ShowDialog();
                this.Close();
            }
            else
            {
                MessageBox.Show(pessoaModel.Mensagem);
            }
            LimparCampos();
        }

        private void LimparCampos()
        {
            txbCodigoVisitante.Text = string.Empty;
        }

        private void Atalho_Load(object sender, EventArgs e)
        {
            this.KeyPreview = true;
            this.KeyDown += Atalho_KeyDown;
        }
    }
}

```

```

}
namespace TotemPIMApresentacao.View
{
    public partial class ObraEquip : Form
    {
        public ObraEquip()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
        }

        private void Atalho_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.V)
                this.Close();
        }

        private void BtnTesteConhecimento_Click(object sender, EventArgs e)
        {
            LoginEquip LoginEquip = new LoginEquip();
            LoginEquip.ShowDialog();
            this.Hide();
        }

        private void Atalho_Load_1(object sender, EventArgs e)
        {
            this.KeyPreview = true;
            this.KeyDown += Atalho_KeyDown;
        }

    }
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class QuestionarioEquip : Form
    {
        private PerguntasEquip perguntas;

        public QuestionarioEquip(ContadorRespostas contadorRespostas, string codigoUsuario)
        {
            InitializeComponent();
            perguntas = new PerguntasEquip(this, contadorRespostas, codigoUsuario, "Equipamento");
            this.WindowState = FormWindowState.Maximized;
            AtualizaPergunta();
        }

        private void AtualizaPergunta()
        {
            perguntas.AtualizarPergunta(lblPergunta, lblNumeroPergunta, new Button[]
            {
                btnAlternativaA,
                btnAlternativaB,
                btnAlternativaC,
                btnAlternativaD,
                btnAlternativaE
            });
        }

        private void ProximaPergunta()
        {
            perguntas.ProximaPergunta();
            AtualizaPergunta();
        }
    }
}

```

```

private void btnAlternativaA_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("A");
    ProximaPergunta();
}

private void btnAlternativaB_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("B");
    ProximaPergunta();
}

private void btnAlternativaC_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("C");
    ProximaPergunta();
}

private void btnAlternativaD_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("D");
    ProximaPergunta();
}

private void btnAlternativaE_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("E");
    ProximaPergunta();
}
}
}
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class ResultadoFormEquip : Form
    {
        private ContadorRespostas contadorRespostas;
        public ResultadoFormEquip(ContadorRespostas contadorRespostas)
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            this.contadorRespostas = contadorRespostas; // Atribua a instância fornecida ao campo
            contadorRespostas
                ExibirResultados();
        }
        private void ExibirResultados()
        {
            contadorRespostas.SalvarResposta();
            MediaAcertos mediaAcertos = new MediaAcertos();
            mediaAcertos.MediaAcertosPorObra(contadorRespostas);
            lblResultado.Text = $"Você acertou {contadorRespostas.RespostasCorretas} de 5 perguntas"; //
            Atualize o texto da lblResultado com o número atual de respostas corretas
        }

        private void btnObra_Click(object sender, EventArgs e)
        {
            ObraEquip ObraSuits = new ObraEquip();
            ObraSuits.ShowDialog();
            this.Hide();
        }

        private void ResultadoFormEquip_Load(object sender, EventArgs e)
        {

```

```

    }
}

using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class LoginDocumentos : Form
    {
        private PessoaModel pessoaModel;
        private PessoaController pessoa;
        private Teclado teclado;
        public string LoginStatus { get; private set; }
        public LoginDocumentos()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            pessoa = new PessoaController();
            pessoaModel = new PessoaModel();
        }

        private void TextBox1_Click(object sender, EventArgs e)
        {
            // Verifica se o teclado ainda não foi instanciado
            if (teclado == null)
            {
                // Cria uma nova instância do teclado
                teclado = new Teclado();
            }

            // Define a TextBox clicada como o TextBox de destino do teclado
            teclado.SetTargetTextBox(sender as TextBox);

            teclado.Show();
        }

        private void Atalho_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.V)
                this.Close();
        }

        private void BtnEntrar_Click(object sender, EventArgs e)
        {
            pessoa.Codigo = txbCodigoVisitante.Text;
            pessoaModel.Login(pessoa);

            if (pessoaModel.Mensagem.Equals(""))
            {
                string codigoUsuario = pessoa.Codigo;
                ContadorRespostas contadorRespostas = new ContadorRespostas(codigoUsuario);

                QuestionarioDocumentos questionarioDocumentos = new
                QuestionarioDocumentos(contadorRespostas, codigoUsuario);
                questionarioDocumentos.ShowDialog();
                this.Close();
            }
            else
            {
                MessageBox.Show(pessoaModel.Mensagem);
            }
            LimparCampos();
        }

        private void LimparCampos()

```

```

        {
            txbCodigoVisitante.Text = string.Empty;
        }

        private void Atalho_Load(object sender, EventArgs e)
        {
            this.KeyPreview = true;
            this.KeyDown += Atalho_KeyDown;
        }
    }
}
namespace TotemPIMApresentacao.View
{
    public partial class ObraDocumentos : Form
    {
        public ObraDocumentos()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
        }

        private void Atalho_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.V)
                this.Close();
        }

        private void BtnTesteConhecimento_Click(object sender, EventArgs e)
        {
            LoginDocumentos loginDocumentos = new LoginDocumentos();
            loginDocumentos.ShowDialog();
            this.Hide();
        }

        private void Atalho_Load_1(object sender, EventArgs e)
        {
            this.KeyPreview = true;
            this.KeyDown += Atalho_KeyDown;
        }
    }
}
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class QuestionarioDocumentos : Form
    {
        private PerguntasDocumentos perguntas;

        public QuestionarioDocumentos(ContadorRespostas contadorRespostas, string codigoUsuario)
        {
            InitializeComponent();
            perguntas = new PerguntasDocumentos(this, contadorRespostas, codigoUsuario, "Documentos");
            this.WindowState = FormWindowState.Maximized;
            AtualizaPergunta();
        }

        private void AtualizaPergunta()
        {
            perguntas.AtualizarPergunta(lblPergunta, lblNumeroPergunta, new Button[]
            {
                btnAlternativaA,
                btnAlternativaB,
                btnAlternativaC,
            }
        }
    }
}

```

```

        btnAlternativaD,
        btnAlternativaE
    });
}

private void ProximaPergunta()
{
    perguntas.ProximaPergunta();
    AtualizaPergunta();
}

private void btnAlternativaA_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("A");
    ProximaPergunta();
}

private void btnAlternativaB_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("B");
    ProximaPergunta();
}

private void btnAlternativaC_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("C");
    ProximaPergunta();
}

private void btnAlternativaD_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("D");
    ProximaPergunta();
}

private void btnAlternativaE_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("E");
    ProximaPergunta();
}
}
}
}
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class ResultadoFormDocumentos : Form
    {
        private ContadorRespostas contadorRespostas;
        public ResultadoFormDocumentos(ContadorRespostas contadorRespostas)
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            this.contadorRespostas = contadorRespostas; // Atribua a instância fornecida ao campo
            contadorRespostas
                ExibirResultados();
        }
        private void ExibirResultados()
        {
            contadorRespostas.SalvarResposta();
            MediaAcertos mediaAcertos = new MediaAcertos();
            mediaAcertos.MediaAcertosPorObra(contadorRespostas);
            lblResultado.Text = $"Você acertou {contadorRespostas.RespostasCorretas} de 5 perguntas"; //
            Atualize o texto da lblResultado com o número atual de respostas corretas
        }
    }
}

```

```

        private void btnObra_Click(object sender, EventArgs e)
        {
            ObraDocumentos ObraSuits = new ObraDocumentos();
            ObraSuits.ShowDialog();
            this.Hide();
        }
    }
}
using TotemPIMApresentacao.Controller;

namespace TotemPIMApresentacao.View
{
    public partial class Apollo_11 : Form
    {
        public Apollo_11()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
        }

        private void Atalho_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.V)
                this.Close();
        }

        private void BtnTesteConhecimento_Click(object sender, EventArgs e)
        {
            LoginApollo_11 loginApollo_11 = new LoginApollo_11();
            loginApollo_11.ShowDialog();
            this.Hide();
        }

        private void Atalho_Load(object sender, EventArgs e)
        {
            this.KeyPreview = true;
            this.KeyDown += Atalho_KeyDown;
        }
    }
}

```

```

using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

```

```

namespace TotemPIMApresentacao.View
{
    public partial class LoginApollo_11 : Form
    {
        private PessoaModel pessoaModel;
        private PessoaController pessoa;
        private Teclado teclado;
        public string LoginStatus { get; private set; }

        public LoginApollo_11()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            pessoa = new PessoaController();
            pessoaModel = new PessoaModel();
        }

        private void TextBox1_Click(object sender, EventArgs e)

```

```

    {
        // Verifica se o teclado ainda não foi instanciado
        if (teclado == null)
        {
            // Cria uma nova instância do teclado
            teclado = new Teclado();
        }

        // Define a TextBox clicada como o TextBox de destino do teclado
        teclado.SetTargetTextBox(sender as TextBox);

        teclado.Show();
    }
    private void Atalho_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.Control && e.KeyCode == Keys.V)
            this.Close();
    }

    private void BtnEntrar_Click(object sender, EventArgs e)
    {
        pessoa.Codigo = txtCodigoVisitante.Text;
        pessoaModel.Login(pessoa, "Apollo-11");

        if (pessoaModel.Mensagem.Equals(""))
        {
            string codigoUsuario = pessoa.Codigo;
            ContadorRespostas contadorRespostas = new ContadorRespostas(codigoUsuario);

            QuestionarioApollo11 questionarioApollo11 = new QuestionarioApollo11(contadorRespostas,
            codigoUsuario);
            questionarioApollo11.ShowDialog();
            this.Close();
        }
        else
        {
            MessageBox.Show(pessoaModel.Mensagem);
        }
        LimparCampos();
    }

    private void LimparCampos()
    {
        txtCodigoVisitante.Text = string.Empty;
    }

    private void Atalho_Load(object sender, EventArgs e)
    {
        this.KeyPreview = true;
        this.KeyDown += Atalho_KeyDown;
    }
}

using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class QuestionarioApollo11 : Form
    {
        private PerguntasApollo11 perguntas;
        public QuestionarioApollo11(ContadorRespostas contadorRespostas, string codigoUsuario)
        {

```



```

InitializeComponent();

perguntas = new PerguntasApollo11(this, contadorRespostas, codigoUsuario, "Apollo-11");
this.WindowState = FormWindowState.Maximized;
AtualizaPergunta();
}

public PerguntasApollo11 PerguntasApollo11
{
    get => default;
    set
    {
    }
}

private void AtualizaPergunta()
{
    perguntas.AtualizarPergunta(lblPergunta, lblNumeroPergunta, new Button[]
    {
        btnAlternativaA,
        btnAlternativaB,
        btnAlternativaC,
        btnAlternativaD,
        btnAlternativaE
    });
}

private void ProximaPergunta()
{
    perguntas.ProximaPergunta();
    AtualizaPergunta();
}

private void btnAlternativaA_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("A");
    ProximaPergunta();
}

private void btnAlternativaB_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("B");
    ProximaPergunta();
}

private void btnAlternativaC_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("C");
    ProximaPergunta();
}

private void btnAlternativaD_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("D");
    ProximaPergunta();
}

private void btnAlternativaE_Click(object sender, EventArgs e)
{
    perguntas.VerificarResposta("E");
    ProximaPergunta();
}
}
}

```

```

using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Model;

namespace TotemPIMApresentacao.View
{
    public partial class ResultadoForm : Form
    {
        private ContadorRespostas contadorRespostas;

        public ResultadoForm(ContadorRespostas contadorRespostas)
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
            this.contadorRespostas = contadorRespostas;
            ExibirResultados();
        }

        private void ExibirResultados()
        {
            contadorRespostas.SalvarResposta();
            lblResultado.Text = $"Você acertou {contadorRespostas.RespostasCorretas} de 5 perguntas"; //
            // Atualize o texto da lblResultado com o número atual de respostas corretas
            MediaAcertos mediaAcertos = new MediaAcertos();
            mediaAcertos.MediaAcertosPorObra(contadorRespostas);
        }

        private void btnObra_Click(object sender, EventArgs e)
        {
            Apollo_11 Apollo_11 = new Apollo_11();
            Apollo_11.ShowDialog();
            this.Hide();
        }
    }
}

using System.Xml.Linq;
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Service;

namespace TotemPIMApresentacao.Model
{
    public class ContadorRespostas
    {
        private QuestionarioService avaliacaoService = new QuestionarioService();
        private int[] posicoesRespostasCorretas;
        private int respostasCorretas;
        private string codigoUsuario;
        private string obraAtual;
        public ContadorRespostas(string codigoUsuario)
        {
            this.codigoUsuario = codigoUsuario;
        }
        public ContadorRespostas(int[] posicoesRespostasCorretas, string codigoUsuario, string obraAtual)
        {
            this.posicoesRespostasCorretas = posicoesRespostasCorretas;
            this.codigoUsuario = codigoUsuario;
            this.obraAtual = obraAtual; // Aqui atribuímos o valor de obraAtual à propriedade obraAtual
            respostasCorretas = 0;
        }
    }
}

```

```

        public void VerificarResposta(int indicePergunta, int respostaUsuario)
        {
            if (indicePergunta < posicoesRespostasCorretas.Length && respostaUsuario ==
posicoesRespostasCorretas[indicePergunta])
            {
                Incrementar();
            }
        }

        public int RespostasCorretas
        {
            get { return respostasCorretas; }
        }

        public void Incrementar()
        {
            respostasCorretas++;
        }

        public string ObraAtual { get; set; }

        internal void SalvarResposta()
        {
            avaliacaoService.RegistrarPontos(respostasCorretas, codigoUsuario, ObraAtual);
        }

        internal QuestionarioService QuestionarioService
        {
            get => default;
            set
            {
            }
        }
    }
}

using System;
using System.Windows.Forms;
using Npgsql;
using TotemPIMApresentacao.Service;
using TotemPIMApresentacao.Servico;

namespace TotemPIMApresentacao.Model
{
    internal class MediaAcertos
    {
        private QuestionarioService avaliacaoService;
        private int quantidade;
        private int totalPessoas; // Total de pessoas é igual à quantidade de códigos distintos

        public void MediaAcertosPorObra(ContadorRespostas contador)
        {
            // Obter a quantidade de acertos e o total de pessoas (quantidade de códigos distintos)
            (quantidade, totalPessoas) = avaliacaoService.ObterInformacoesPorObra(contador.ObraAtual);

            // Calcular a média de acertos por pessoa
            double mediaAcertos = CalcularMediaAcertos();

            // Exibir as informações em uma caixa de mensagem

```

```

        MessageBox.Show($"Quantidade de Acertos: {quantidade}\nMédia de Acertos por Pessoa: {mediaAcertos}");
    }

    private double CalcularMediaAcertos()
    {
        // Certifique-se de verificar se o total de pessoas é diferente de zero para evitar divisão por zero
        if (totalPessoas != 0)
        {
            // Calcular a média de acertos por pessoa
            double mediaAcertos = (double)quantidade / totalPessoas;
            return mediaAcertos;
        }
        else
        {
            return 0; // Se não houver pessoas, a média é zero
        }
    }
}

using TotemPIMApresentacao.Servico;
using TotemPIMApresentacao.Controller;
using TotemPIMApresentacao.Service;

namespace TotemPIMApresentacao.Model
{
    public class PessoaModel
    {
        PessoaServico pessoaServico = new PessoaServico(new Dbconexao());
        QuestionarioService questionario = new QuestionarioService();

        private string mensagem = "";

        public PessoaModel()
        {
        }

        public bool Login(PessoaController pessoa, string obraAtual)
        {
            var pessoaEncontrada = pessoaServico.BuscarPorCodigo(pessoa);

            if (pessoaEncontrada != null)
            {
                // Verificar se o código está associado à obra atual
                bool codigoAssociadoObra = questionario.CodigoAssociadoObra(pessoa.Codigo, obraAtual);

                if (codigoAssociadoObra)
                {
                    this.mensagem = "O token fornecido já está associado à obra atual.";
                    return false;
                }
            }

            return true;
        }

        public string Mensagem
        {
            get { return mensagem; }
        }
    }
}

```

```

using Npgsql;
using System;
using System.Data;

namespace TotemPIMApresentacao.Servico
{
    public class Dbconexao : IDisposable
    {
        private NpgsqlConnection connection;

        public Dbconexao()
        {
            connection = new NpgsqlConnection("Server=localhost;Port=5432;Database=DB_Museu;User
            Id=postgres;Password=4852;");
        }

        public NpgsqlConnection GetConnection()
        {
            if (connection.State != ConnectionState.Open)
            {
                try
                {
                    connection.Open();
                }
                catch (Exception ex)
                {
                    // Lidar com a exceção ou relançá-la, conforme necessário
                    throw new Exception("Erro ao abrir a conexão com o banco de dados.", ex);
                }
            }

            return connection;
        }

        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (connection != null)
                {
                    if (connection.State != ConnectionState.Closed)
                    {
                        connection.Close();
                    }
                    connection.Dispose();
                    connection = null;
                }
            }
        }
    }
}

using Dapper;
using TotemPIMApresentacao.Controller;

namespace TotemPIMApresentacao.Servico
{

```

```

public class PessoaServico
{
    private readonly Dbconexao dbconexao;

    public PessoaServico(Dbconexao conexao)
    {
        this.dbconexao = conexao;
    }

    public Model.PessoaModel PessoaModel
    {
        get => default;
        set
        {
        }
    }
}

public PessoaController BuscarPorCodigo(PessoaController pessoa)
{
    using (var conexao = new Dbconexao())
    {
        var connection = conexao.GetConnection();

        var resultado = connection.QueryFirstOrDefault(
            "SELECT * FROM visitante WHERE Codigo = @codigo ",
            new { Codigo = pessoa.Codigo });

        if (resultado == null)
        {
            return null;
        }
        pessoa.Codigo = resultado.codigo;
        pessoa.Nome = resultado.nome;
        return pessoa;
    }
}
}
}

```

```

using Npgsql;
using TotemPIMApresentacao.Servico;

namespace TotemPIMApresentacao.Service
{
    public class QuestionarioService
    {
        private Dbconexao dbconexao = new Dbconexao();

        private NpgsqlConnection conexao;

        internal QuestionarioService()
        {
            conexao = dbconexao.GetConnection() as NpgsqlConnection;
        }

        internal Model.MediaAcertos MediaAcertos
        {
            get => default;
            set
            {
            }
        }
    }
}

```

```

internal void RegistrarPontos(int acertos,string codigoUsuario,string obraAtual)
{
    using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO public.questionario
(codigo, acertos,obra, data)
VALUES (@CodigoPessoa, @acertos,@obra, @Data)", conexao))
    {
        command.Parameters.AddWithValue("@CodigoPessoa", codigoUsuario);
        command.Parameters.AddWithValue("@acertos", acertos);
        command.Parameters.AddWithValue("@obra", obraAtual);
        command.Parameters.AddWithValue("@Data", DateTime.Now);

        int linhasAfetadas = command.ExecuteNonQuery();

        if (linhasAfetadas < 0)
        {
            MessageBox.Show("Falha ao registrar o voto.");
        }
    }
}

internal (int, int) ObterInformacoesPorObra(string nomeObra)
{
    int quantidadeAcertos = 0;
    int totalPessoas = 0;

    using (NpgsqlCommand command = new NpgsqlCommand
        (@"SELECT SUM(acertos), COUNT(DISTINCT codigo) FROM public.questionario WHERE obra =
@NomeObra", conexao))
    {
        command.Parameters.AddWithValue("@NomeObra", nomeObra);

        var reader = command.ExecuteReader();
        if (reader.Read())
        {
            if (!reader.IsDBNull(0))
            {
                quantidadeAcertos = reader.GetInt32(0);
            }
            if (!reader.IsDBNull(1))
            {
                totalPessoas = reader.GetInt32(1);
            }
        }
    }

    return (quantidadeAcertos, totalPessoas);
}

internal bool CodigoAssociadoObra(string codigo, string obra)
{
    bool codigoAssociado = false;

    using (NpgsqlCommand command = new NpgsqlCommand
        (@"SELECT COUNT(*) FROM public.questionario WHERE codigo = @Codigo AND obra = @obra",
conexao))
    {
        command.Parameters.AddWithValue("@Codigo", codigo);
        command.Parameters.AddWithValue("@obra", obra);

        long count = (long)command.ExecuteScalar();

        // Se rowCount for maior que zero, significa que o código está associado à obra
        if (count > 0)
        {
            codigoAssociado = true;
        }
    }
}

```

```

    }

    return codigoAssociado;
}

}
}

using TotemPIMApresentacao.View;
using TotemPIMApresentacao.Model;
using WinFormsTimer = System.Windows.Forms.Timer;

namespace TotemPIMApresentacao.Controller
{
    public class PerguntasApollo11
    {
        private string obra = "Apollo-11";
        private Apollo_11 apollo;
        private ContadorRespostas contadorRespostas;
        private ResultadoForm resultadosForm;
        private WinFormsTimer timer;
        private QuestionarioApollo11 questionarioForm;
        private string codigoUsuario;
        private int indicePergunta = 0;

        private string[] perguntas = {
            "Qual era o nome do módulo lunar usado na missão Apollo 11?",
            "Quantos astronautas a nave Apollo 11 levou para a Lua?",
            "Qual foi o papel do módulo lunar na missão Apollo 11?",
            "Qual era o nome do primeiro astronauta a pisar na Lua a bordo do módulo lunar?",
            "Como o módulo lunar retornou à órbita lunar após a missão na superfície?"
        };

        private string[][] alternativas = {

            new string[] { "A) Eagle",
                "B) Falcon",
                "C) Hawk",
                "D) Sparrow",
                "E) Osprey" }, // Resposta correta: A

            new string[] { "A) Quatro",
                "B) Dois",
                "C) Três",
                "D) Cinco",
                "E) Seis" }, // Resposta correta: C

            new string[] { "A) Orbitar a Lua e transmitir dados para a Terra ",
                "B)Transportar astronautas até a Lua e mantê-los seguros ", // Resposta correta: B
                "C) Pousar na Lua e retornar à órbita lunar",
                "D) Explorar a superfície lunar em busca de recursos",
                "E) Construir uma base lunar para futuras missões" },

            new string[] { "A) Neil Armstrong",
                "B) Buzz Aldrin",
                "C) Michael Collins",
                "D) Alan Shepard",
                "E) John Glenn" }, // Resposta correta: A

            new string[] { "A) Não retornou à órbita, permanecendo na superfície lunar ",
                "B) Utilizou foguetes auxiliares para retornar à órbita",

```



```

        "C) Foi empurrado por astronautas para voltar à órbita",
        "D) Aterrissou em uma área predestinada para ser recuperado posteriormente",
        "E) Decolou da superfície lunar e se acoplou ao módulo de comando" } // Resposta correta: E
    };

    private int[] posicoesRespostasCorretas = { 0, 2, 1, 0, 4 };

    public PerguntasApollo11(QuestionarioApollo11 questionarioForm, ContadorRespostas
contadorRespostas, string codigoUsuario, string obra)
    {
        this.questionarioForm = questionarioForm;
        this.contadorRespostas = contadorRespostas;
        this.codigoUsuario = codigoUsuario;
        contadorRespostas.ObraAtual = obra;

        timer = new WinFormsTimer();
        timer.Interval = 4000;
        timer.Tick += Timer_Tick;
    }

    public void VerificarResposta(string respostaUsuario)
    {
        string respostaCorreta = ObterRespostaCorreta(indicePergunta);

        int posicaoRespostaUsuario = respostaUsuario[0] - 'A';

        if (posicaoRespostaUsuario == posicoesRespostasCorretas[indicePergunta])
        {
            contadorRespostas.Incrementar();
        }
    }

    private string ObterRespostaCorreta(int indice)
    {
        if (indice < posicoesRespostasCorretas.Length)
        {
            return alternativas[indice][posicoesRespostasCorretas[indice]].Substring(0, 1);
        }
        else
        {
            return ""; // Sem resposta correta para o índice fornecido
        }
    }

    public string PerguntaAtual()
    {
        if (indicePergunta < perguntas.Length)
        {
            return perguntas[indicePergunta];
        }
        else
        {
            timer.Start();
            resultadosForm = new ResultadoForm(contadorRespostas);
            resultadosForm.Show();
            questionarioForm.Hide();
            return "";
        }
    }

    private void Timer_Tick(object sender, EventArgs e)
    {
        resultadosForm.Close();
        apollo = new Apollo_11();
        apollo.Show();
        timer.Stop();
    }
}

```

```

public string[] AlternativasAtual()
{
    if (indicePergunta < alternativas.Length)
    {
        return alternativas[indicePergunta];
    }
    else
    {
        return null;
    }
}
public void ProximaPergunta()
{
    indicePergunta++;
}
public void AtualizarPergunta(Label lblPergunta, Label lblNumeroPergunta, Button[] botoesAlternativas)
{
    lblNumeroPergunta.Text = "Pergunta " + (indicePergunta + 1);
    lblPergunta.Text = PerguntaAtual();
    string[] alternativas = AlternativasAtual();

    if (alternativas != null && alternativas.Length == 5 && botoesAlternativas.Length >= 5)
    {
        for (int i = 0; i < 5; i++)
        {
            botoesAlternativas[i].Text = alternativas[i];
        }
    }
}
}
}

```

```

using TotemPIMApresentacao.View;
using TotemPIMApresentacao.Model;
using WinFormsTimer = System.Windows.Forms.Timer;

```

```

namespace TotemPIMApresentacao.Controller
{
    internal class PerguntasDocumentos
    {
        private string obra = "Documentos";
        private ObraDocumentos obraDocumentos;
        private ContadorRespostas contadorRespostas;
        private ResultadoFormDocumentos resultadosForm;
        private WinFormsTimer timer;
        private QuestionarioDocumentos questionarioForm;
        private string codigoUsuario;
        private int indicePergunta = 0;
        private string[] perguntas = {
            "Qual foi o principal documento que registrou a missão Apollo 11?", //1
            "Qual era o objetivo principal dos registros de comunicações da NASA durante a missão? ", //2
            "Qual foi a importância dos memorandos internos da NASA para a missão Apollo 11?", //3
            "Quem era o principal destinatário dos memorandos internos da NASA durante a missão?", //4
            "Qual era a natureza dos documentos históricos relacionados à missão Apollo 11?" //5
        };

        private string[][] alternativas = {
            new string[] { "A) Relatório de missão ",
                "B) Registro de comunicações da NASA ",
                "C) Memorando interno da NASA ",
                "D) Diário dos astronautas "
            }
        }
    }
}

```

```

        "E) Carta do presidente dos EUA " }, // Resposta correta: A

        new string[] { "A) Documentar as conversas dos astronautas entre si ",
            "B) Registrar as instruções da NASA para os astronautas ",
            "C) Monitorar a saúde dos astronautas",
            "D) Manter um registro de dados científicos",
            "E) Comunicar com o público em geral" }, // Resposta correta: A

        new string[] { "A) Comunicar decisões e instruções importantes " ,
            "B) Registrar eventos cotidianos da missão ", // Resposta correta: A
            "C) Documentar descobertas científicas",
            "D) Preservar a história da exploração espacial",
            "E) Descrever os procedimentos de segurança " },

        new string[] { "A) Astronautas " ,
            "B) Engenheiros da missão " ,
            "C) Diretores da NASA " ,
            "D) Presidente dos EUA " ,
            "E) Público em geral " }, // Resposta correta: B

        new string[] { "A) Confidenciais " ,
            "B) Secretos " ,
            "C) Públicos " ,
            "D) Privados " ,
            "E) Restritos " } // Resposta correta: C
    };

    private int[] posicoesRespostasCorretas = { 0, 0, 0, 1, 2 };

    public PerguntasDocumentos(QuestionarioDocumentos questionarioForm, ContadorRespostas
    contadorRespostas, string codigoUsuario, string obra)
    {
        this.questionarioForm = questionarioForm;
        this.contadorRespostas = contadorRespostas;
        this.codigoUsuario = codigoUsuario;
        contadorRespostas.ObraAtual = obra;

        timer = new WinFormsTimer();
        timer.Interval = 4000;
        timer.Tick += Timer_Tick;
    }

    public void VerificarResposta(string respostaUsuario)
    {
        string respostaCorreta = ObterRespostaCorreta(indicePergunta);

        int posicaoRespostaUsuario = respostaUsuario[0] - 'A';

        if (posicaoRespostaUsuario == posicoesRespostasCorretas[indicePergunta])
        {
            contadorRespostas.Incrementar();
        }
    }

    private string ObterRespostaCorreta(int indice)
    {
        if (indice < posicoesRespostasCorretas.Length)
        {
            return alternativas[indice][posicoesRespostasCorretas[indice]].Substring(0, 1);
        }
        else
        {
            return "";
        }
    }
}

```

```

public string PerguntaAtual()
{
    if (indicePergunta < perguntas.Length)
    {
        return perguntas[indicePergunta];
    }
    else
    {
        timer.Start();
        resultadosForm = new ResultadoFormDocumentos(contadorRespostas);
        resultadosForm.ShowDialog();
        questionarioForm.Hide();
        return "";
    }
}
private void Timer_Tick(object sender, EventArgs e)
{
    resultadosForm.Close();
    obraDocumentos = new ObraDocumentos();
    obraDocumentos.Show();
    timer.Stop();
}
public string[] AlternativasAtual()
{
    if (indicePergunta < alternativas.Length)
    {
        return alternativas[indicePergunta];
    }
    else
    {
        return null;
    }
}
public void ProximaPergunta()
{
    indicePergunta++;
}
public void AtualizarPergunta(Label lblPergunta, Label lblNumeroPergunta, Button[] botoesAlternativas)
{
    lblNumeroPergunta.Text = "Pergunta " + (indicePergunta + 1);
    lblPergunta.Text = PerguntaAtual();
    string[] alternativas = AlternativasAtual();

    if (alternativas != null && alternativas.Length == 5 && botoesAlternativas.Length >= 5)
    {
        for (int i = 0; i < 5; i++)
        {
            botoesAlternativas[i].Text = alternativas[i];
        }
    }
}
}
}

```

```

using TotemPIMApresentacao.View;
using TotemPIMApresentacao.Model;
using static System.Windows.Forms.DataFormats;
using WinFormsTimer = System.Windows.Forms.Timer;

```

```

namespace TotemPIMApresentacao.Controller
{
    internal class PerguntasEquip
    {

```

```

private ObraEquip obraEquip;
private ContadorRespostas contadorRespostas;
private ResultadoFormEquip resultadosForm;
private WinFormsTimer timer;
private QuestionarioEquip questionarioForm;
private string codigoUsuario;
private int indicePergunta = 0;
private string[] perguntas = {
    "Qual era o nome do computador utilizado na nave Apollo 11 durante a missão à Lua? ", //1
    "Que tipo de câmeras foram usadas para documentar a missão Apollo 11? ? ", //2
    "Qual era a principal função dos dispositivos de comunicação na Apollo 11? ", //3
    "Além do computador de bordo, quais eram os outros tipos utilizados na missão? ", //4
    "Que tipo de instrumentos foram usados para medir a posição da nave durante a missão? " //5
};

private string[][] alternativas = {

    new string[] { "A) Apollo Central Computer (ACC)",
        "B) Lunar Module Guidance Computer (LMGC)",
        "C) Apollo Guidance Computer (AGC)", // Resposta correta: C
        "D) Lunar Module Central Computer (LMCC)",
        "E) Spacecraft Computer System (SCS)" },

    new string[] { "A) Câmeras digitais",
        "B) Câmeras de filme 35mm",
        "C) Câmeras de filme 16mm", // Resposta correta: C
        "D) Câmeras de vídeo analógico",
        "E) Câmeras infravermelhas" },

    new string[] { "A) Transmitir dados para a Terra" ,
        "B) Facilitar a comunicação entre astronautas", // Resposta correta: B
        "C) Receber instruções da NASA",
        "D) Monitorar a saúde dos astronautas",
        "E) Controlar os sistemas da nave" },

    new string[] { "A) Computadores pessoais",
        "B) Computadores de backup",
        "C) Computadores de controle de missão", // Resposta correta: C
        "D) Computadores de navegação",
        "E) Computadores de comunicação" },

    new string[] { "A) Radares",
        "B) Giroscópios",
        "C) Acelerômetros", // Resposta correta: C
        "D) Telescópios",
        "E) Magnetômetros" }

};

private int[] posicoesRespostasCorretas = { 2, 2, 1, 2, 2 };

public PerguntasEquip(QuestionarioEquip questionarioForm, ContadorRespostas contadorRespostas,
string codigoUsuario, string obra)
{
    this.questionarioForm = questionarioForm;
    this.contadorRespostas = contadorRespostas;
    this.codigoUsuario = codigoUsuario;
    contadorRespostas.ObraAtual = obra;

    timer = new WinFormsTimer();
    timer.Interval = 4000;
    timer.Tick += Timer_Tick;
}

public void VerificarResposta(string respostaUsuario)
{
    string respostaCorreta = ObterRespostaCorreta(indicePergunta);
}

```

```

        int posicaoRespostaUsuario = respostaUsuario[0] - 'A';

        if (posicaoRespostaUsuario == posicoesRespostasCorretas[indicePergunta])
        {
            contadorRespostas.Incrementar();
        }
    }

    private string ObterRespostaCorreta(int indice)
    {
        if (indice < posicoesRespostasCorretas.Length)
        {
            return alternativas[indice][posicoesRespostasCorretas[indice]].Substring(0, 1);
        }
        else
        {
            return ""; // Sem resposta correta para o índice fornecido
        }
    }

    public string PerguntaAtual()
    {
        if (indicePergunta < perguntas.Length)
        {
            return perguntas[indicePergunta];
        }
        else
        {
            timer.Start();
            resultadosForm = new ResultadoFormEquip(contadorRespostas);
            resultadosForm.ShowDialog();
            questionarioForm.Hide();
            return "";
        }
    }

    private void Timer_Tick(object sender, EventArgs e)
    {
        resultadosForm.Close();
        obraEquip = new ObraEquip();
        obraEquip.Show();
        timer.Stop();
    }

    public string[] AlternativasAtual()
    {
        if (indicePergunta < alternativas.Length)
        {
            return alternativas[indicePergunta];
        }
        else
        {
            return null;
        }
    }

    public void ProximaPergunta()
    {
        indicePergunta++;
    }

    public void AtualizarPergunta(Label lblPergunta, Label lblNumeroPergunta, Button[] botoesAlternativas)
    {
        lblNumeroPergunta.Text = "Pergunta " + (indicePergunta + 1);
        lblPergunta.Text = PerguntaAtual();
        string[] alternativas = AlternativasAtual();

        if (alternativas != null && alternativas.Length == 5 && botoesAlternativas.Length >= 5)
        {
            for (int i = 0; i < 5; i++)

```

```

        {
            botoesAlternativas[i].Text = alternativas[i];
        }
    }
}
}
}

```

```

using TotemPIMApresentacao.View;
using TotemPIMApresentacao.Model;
using static System.Windows.Forms.DataFormats;
using WinFormsTimer = System.Windows.Forms.Timer;

```

```

namespace TotemPIMApresentacao.Controller

```

```

{
    internal class PerguntasFotos
    {
        private ObraFotos obraSuit;
        private ContadorRespostas contadorRespostas;
        private ResultadoFormFotos resultadosForm;
        private WinFormsTimer timer;
        private QuestionarioFotos questionarioForm;
        private string codigoUsuario;
        private int indicePergunta = 0;
        private string[] perguntas = {
            "Quem foi o astronauta responsável por tirar a famosa foto do Homem na Lua?",
            "Qual foi o formato principal das fotos tiradas durante a missão Apollo 11? ",
            "Que tipo de câmera foi usada para capturar a maioria das fotos da superfície lunar? ",
            "Quantas fotos foram tiradas durante a missão Apollo 11?",
            "Qual foi o formato principal dos vídeos gravados durante a missão Apollo 11? "
        };

        private string[][] alternativas = {

            new string[] { "A) Neil Armstrong",
                "B) Buzz Aldrin",
                "C) Michael Collins",
                "D) Alan Shepard",
                "E) John Glenn" }, // Resposta correta: B

            new string[] { "A) JPEG",
                "B) PNG",
                "C) TIFF",
                "D) GIF",
                "E) BMP" }, // Resposta correta: C

            new string[] { "A) Hasselblad 500EL",
                "B) Nikon D850", // Resposta correta: A
                "C) Canon EOS 5D Mark IV",
                "D) Sony Alpha a7S III",
                "E) Leica M10" },

            new string[] { "A) Cerca de 100 ",
                "B) Cerca de 500",
                "C) Cerca de 1.000",
                "D) Cerca de 1.500",
                "E) Cerca de 2.000" }, // Resposta correta: D

            new string[] { "A) VHS",
                "B) Betamax",

```

```

        "C) Super 8",
        "D) 16mm film",
        "E) Digital " } // Resposta correta: D
    };

    private int[] posicoesRespostasCorretas = { 1, 2, 0, 3, 3 };

    public PerguntasFotos(QuestionarioFotos questionarioForm, ContadorRespostas contadorRespostas,
        string codigoUsuario, string obra)
    {
        this.questionarioForm = questionarioForm;
        this.contadorRespostas = contadorRespostas;
        this.codigoUsuario = codigoUsuario;
        contadorRespostas.ObraAtual = obra;

        timer = new WinFormsTimer();
        timer.Interval = 4000;
        timer.Tick += Timer_Tick;
    }

    public void VerificarResposta(string respostaUsuario)
    {
        string respostaCorreta = ObterRespostaCorreta(indicePergunta);

        int posicaoRespostaUsuario = respostaUsuario[0] - 'A';

        if (posicaoRespostaUsuario == posicoesRespostasCorretas[indicePergunta])
        {
            contadorRespostas.Incrementar();
        }
    }

    private string ObterRespostaCorreta(int indice)
    {
        if (indice < posicoesRespostasCorretas.Length)
        {
            return alternativas[indice][posicoesRespostasCorretas[indice]].Substring(0, 1);
        }
        else
        {
            return ""; // Sem resposta correta para o índice fornecido
        }
    }

    public string PerguntaAtual()
    {
        if (indicePergunta < perguntas.Length)
        {
            return perguntas[indicePergunta];
        }
        else
        {
            timer.Start();
            resultadosForm = new ResultadoFormFotos(contadorRespostas);
            resultadosForm.ShowDialog();
            questionarioForm.Hide();
            return "";
        }
    }

    private void Timer_Tick(object sender, EventArgs e)
    {
        resultadosForm.Close();
        obraSuit = new ObraFotos();
        obraSuit.Show();
        timer.Stop();
    }
}

```



```

        public string[] AlternativasAtual()
        {
            if (indicePergunta < alternativas.Length)
            {
                return alternativas[indicePergunta];
            }
            else
            {
                return null;
            }
        }
        public void ProximaPergunta()
        {
            indicePergunta++;
        }
        public void AtualizarPergunta(Label lblPergunta, Label lblNumeroPergunta, Button[] botoesAlternativas)
        {
            lblNumeroPergunta.Text = "Pergunta " + (indicePergunta + 1);
            lblPergunta.Text = PerguntaAtual();
            string[] alternativas = AlternativasAtual();

            if (alternativas != null && alternativas.Length == 5 && botoesAlternativas.Length >= 5)
            {
                for (int i = 0; i < 5; i++)
                {
                    botoesAlternativas[i].Text = alternativas[i];
                }
            }
        }
    }
}

```

```

using TotemPIMApresentacao.View;
using TotemPIMApresentacao.Model;
using WinFormsTimer = System.Windows.Forms.Timer;

```

```

namespace TotemPIMApresentacao.Controller
{
    internal class PerguntasHistoria
    {
        private ObraHistoria obraEquip;
        private ContadorRespostas contadorRespostas;
        private ResultadoFormHistoria resultadosForm;
        private WinFormsTimer timer;
        private QuestionarioHistoria questionarioForm;
        private string codigoUsuario;

        private int indicePergunta = 0;
        private string[] perguntas = {
            "Qual foi o presidente dos Estados Unidos que anunciou o objetivo de pousar na Lua?", //1
            "Em que ano a missão Apollo 11 foi lançada?", //2
            "Qual foi o nome da corrida espacial entre os Estados Unidos e a União Soviética?", //3
            "Além dos Estados Unidos, qual outro país enviou uma missão tripulada à Lua?", //4
            "Qual foi o nome do primeiro ser humano a viajar para o espaço, em 1961?" //5
        };

        private string[][] alternativas = {
            new string[] { "A) John F. Kennedy",

```

```

        "B) Lyndon B. Johnson ",
        "C) Richard Nixon ",
        "D) Dwight D. Eisenhower ",
        "E) Gerald Ford" }, // Resposta correta: A

new string[] { "A) 1967",
               "B) 1968",
               "C) 1969",
               "D) 1970",
               "E) 1971" }, // Resposta correta: C

new string[] { "A) Corrida Armamentista ",
               "B) Guerra Fria Espacial ", // Resposta correta: D
               "C) Corrida Lunar",
               "D) Corrida Espacial",
               "E) Corrida Tecnológica" },

new string[] { "A) União Soviética",
               "B) China ",
               "C) Índia",
               "D) Japão",
               "E) Alemanha" }, // Resposta correta: A

new string[] { "A) Yuri Gagarin ",
               "B) Neil Armstrong",
               "C) Buzz Aldrin ",
               "D) Alan Shepard ",
               "E) John Glenn" } // Resposta correta: A
};

private int[] posicoesRespostasCorretas = { 0, 2, 3, 0, 0 };

public PerguntasHistoria(QuestionarioHistoria questionarioForm, ContadorRespostas contadorRespostas,
string codigoUsuario, string obra)
{
    this.questionarioForm = questionarioForm;
    this.contadorRespostas = contadorRespostas;
    this.codigoUsuario = codigoUsuario;
    contadorRespostas.ObraAtual = obra;

    timer = new WinFormsTimer();
    timer.Interval = 4000;
    timer.Tick += Timer_Tick;
}

public void VerificarResposta(string respostaUsuario)
{
    string respostaCorreta = ObterRespostaCorreta(indicePergunta);

    int posicaoRespostaUsuario = respostaUsuario[0] - 'A';

    if (posicaoRespostaUsuario == posicoesRespostasCorretas[indicePergunta])
    {
        contadorRespostas.Incrementar();
    }
}

private string ObterRespostaCorreta(int indice)
{
    if (indice < posicoesRespostasCorretas.Length)
    {
        return alternativas[indice][posicoesRespostasCorretas[indice]].Substring(0, 1);
    }
    else
    {

```

```

        return ""; // Sem resposta correta para o índice fornecido
    }
}

public string PerguntaAtual()
{
    if (indicePergunta < perguntas.Length)
    {
        return perguntas[indicePergunta];
    }
    else
    {
        timer.Start();
        resultadosForm = new ResultadoFormHistoria(contadorRespostas);
        resultadosForm.ShowDialog();
        questionarioForm.Hide();
        return "";
    }
}

private void Timer_Tick(object sender, EventArgs e)
{
    resultadosForm.Close();
    obraEquip = new ObraHistoria();
    obraEquip.Show();
    timer.Stop();
}

public string[] AlternativasAtual()
{
    if (indicePergunta < alternativas.Length)
    {
        return alternativas[indicePergunta];
    }
    else
    {
        return null;
    }
}

public void ProximaPergunta()
{
    indicePergunta++;
}

public void AtualizarPergunta(Label lblPergunta, Label lblNumeroPergunta, Button[] botoesAlternativas)
{
    lblNumeroPergunta.Text = "Pergunta " + (indicePergunta + 1);
    lblPergunta.Text = PerguntaAtual();
    string[] alternativas = AlternativasAtual();

    if (alternativas != null && alternativas.Length == 5 && botoesAlternativas.Length >= 5)
    {
        for (int i = 0; i < 5; i++)
        {
            botoesAlternativas[i].Text = alternativas[i];
        }
    }
}
}
}

```

```

using TotemPIMApresentacao.View;

```

```

using TotemPIMApresentacao.Model;
using WinFormsTimer = System.Windows.Forms.Timer;

namespace TotemPIMApresentacao.Controller
{
    internal class PerguntasReplica
    {
        private ObraReplica obraEquip;
        private ContadorRespostas contadorRespostas;
        private ResultadoFormReplica resultadosForm;
        private WinFormsTimer timer;
        private QuestionarioReplica questionarioForm;
        private string codigoUsuario;

        private int indicePergunta = 0;
        private string[] perguntas = {
            "Qual era o principal objetivo da nave Apollo 11 durante a missão à Lua?", //1
            "Quantos estágios principais tinha a nave Apollo 11?", //2
            "Qual era o nome do módulo que permanecia em órbita enquanto o módulo lunar pousava na Lua?",
//3
            "Como era chamado o foguete que lançou a nave Apollo 11 ao espaço? ", //4
            "Qual era a capacidade máxima de tripulação da nave Apollo 11?" //5
        };

        private string[][] alternativas = {
            new string[] { "A) Pousar na Lua e coletar amostras ",
                "B) Orbitar a Lua e fotografar a superfície ",
                "C) Estudar a radiação no espaço profundo ",
                "D) Testar novas tecnologias de propulsão ",
                "E) Realizar experimentos científicos na órbita lunar" }, // Resposta correta: A 0

            new string[] { "A) Dois",
                "B) Três",
                "C) Quatro",
                "D) Cinco",
                "E) Seis" }, // Resposta correta: B 1

            new string[] { "A) Módulo de Comando ",
                "B) Módulo Lunar ", // Resposta correta: A 0
                "C) Módulo de Serviço",
                "D) Módulo de Aterrissagem ",
                "E) Módulo de Retorno" },

            new string[] { "A) Saturn V ",
                "B) Atlas-Agena ",
                "C) Titan II",
                "D) Delta II ",
                "E) Falcon 9" }, // Resposta correta: A 0

            new string[] { "A) Dois astronautas ",
                "B) Três astronautas",
                "C) Quatro astronautas",
                "D) Cinco astronautas",
                "E) Seis astronautas" } // Resposta correta: B 1
        };

        private int[] posicoesRespostasCorretas = { 0, 1, 0, 0, 1 };

        public PerguntasReplica(QuestionarioReplica questionarioForm, ContadorRespostas contadorRespostas,
            string codigoUsuario, string obra)
        {
            this.questionarioForm = questionarioForm;
            this.contadorRespostas = contadorRespostas;
            this.codigoUsuario = codigoUsuario;
            contadorRespostas.ObraAtual = obra;
        }
    }
}

```

```

        timer = new WinFormsTimer();
        timer.Interval = 4000;
        timer.Tick += Timer_Tick;
    }

    public void VerificarResposta(string respostaUsuario)
    {
        string respostaCorreta = ObterRespostaCorreta(indicePergunta);

        int posicaoRespostaUsuario = respostaUsuario[0] - 'A';

        if (posicaoRespostaUsuario == posicoesRespostasCorretas[indicePergunta])
        {
            contadorRespostas.Incrementar();
        }
    }

    private string ObterRespostaCorreta(int indice)
    {
        if (indice < posicoesRespostasCorretas.Length)
        {
            return alternativas[indice][posicoesRespostasCorretas[indice]].Substring(0, 1);
        }
        else
        {
            return ""; // Sem resposta correta para o índice fornecido
        }
    }

    public string PerguntaAtual()
    {
        if (indicePergunta < perguntas.Length)
        {
            return perguntas[indicePergunta];
        }
        else
        {
            timer.Start();
            resultadosForm = new ResultadoFormReplica(contadorRespostas);
            resultadosForm.ShowDialog();
            questionarioForm.Hide();
            return "";
        }
    }

    private void Timer_Tick(object sender, EventArgs e)
    {
        resultadosForm.Close();
        obraEquip = new ObraReplica();
        obraEquip.Show();
        timer.Stop();
    }

    public string[] AlternativasAtual()
    {
        if (indicePergunta < alternativas.Length)
        {
            return alternativas[indicePergunta];
        }
        else
        {
            return null;
        }
    }

    public void ProximaPergunta()
    {
        indicePergunta++;
    }

```

```

    }
    public void AtualizarPergunta(Label lblPergunta, Label lblNumeroPergunta, Button[] botoesAlternativas)
    {
        lblNumeroPergunta.Text = "Pergunta " + (indicePergunta + 1);
        lblPergunta.Text = PerguntaAtual();
        string[] alternativas = AlternativasAtual();

        if (alternativas != null && alternativas.Length == 5 && botoesAlternativas.Length >= 5)
        {
            for (int i = 0; i < 5; i++)
            {
                botoesAlternativas[i].Text = alternativas[i];
            }
        }
    }
}
}
}

```

```

using TotemPIMApresentacao.View;
using TotemPIMApresentacao.Model;
using WinFormsTimer = System.Windows.Forms.Timer;

```

```

namespace TotemPIMApresentacao.Controller

```

```

{
    internal class PerguntasTrajesEspaciais
    {
        private ObraSuits obraSuit;
        private ContadorRespostas contadorRespostas;
        private ResultadoFormSuit resultadosForm;
        private WinFormsTimer timer;
        private QuestionarioSuits questionarioForm;
        private string codigoUsuario;

        private int indicePergunta = 0;
        private string[] perguntas = {
            "Qual era o nome do traje espacial usado pelos astronautas da Apollo 11?",
            "Quantas camadas tinha o traje espacial utilizado na missão Apollo 11?",
            "Qual era a função da camada externa do traje espacial?",
            "Como eram chamadas as luvas dos trajes espaciais da Apollo 11? ",
            "Qual era o principal desafio enfrentado pelos astronautas em relação aos trajes? "
        };

        private string[][] alternativas = {

            new string[] { "A) AstroSuit",
                "B) SpaceGuard",
                "C) LunarSuit",
                "D) Extravehicular Mobility Unit (EMU) ",
                "E) SpaceWalker" }, // Resposta correta: D

            new string[] { "A) Uma camada",
                "B) Duas camadas ",
                "C) Três camadas ",
                "D) Quatro camadas ",
                "E) Cinco camadas " }, // Resposta correta: C

            new string[] { "A) Isolamento térmico",

```

```

        "B) Proteção contra micro meteoroides", // Resposta correta: B
        "C) Controle de umidade",
        "D) Suporte de vida ",
        "E) Comunicação"},

        new string[] { "A) GloveSuits",
            "B) AstroGloves",
            "C) LunarGloves",
            "D) EVA Gloves",
            "E) Space Gloves " }, // Resposta correta: D

        new string[] { "A) Regulação de temperatura",
            "B Movimentação restrita",
            "C) Falta de oxigênio",
            "D) Visibilidade limitada",
            "E) Peso excessivo" } // Resposta correta: B
    };

    private int[] posicoesRespostasCorretas = { 3, 2, 1, 3, 1 };

    public PerguntasTrajesEspaciais(QuestionarioSuits questionarioForm, ContadorRespostas
    contadorRespostas, string codigoUsuario, string obra)
    {
        this.questionarioForm = questionarioForm;
        this.contadorRespostas = contadorRespostas;
        this.codigoUsuario = codigoUsuario;
        contadorRespostas.ObraAtual = obra;

        timer = new WinFormsTimer();
        timer.Interval = 4000;
        timer.Tick += Timer_Tick;
    }

    public void VerificarResposta(string respostaUsuario)
    {
        string respostaCorreta = ObterRespostaCorreta(indicePergunta);

        int posicaoRespostaUsuario = respostaUsuario[0] - 'A';

        if (posicaoRespostaUsuario == posicoesRespostasCorretas[indicePergunta])
        {
            contadorRespostas.Incrementar();
        }
    }

    private string ObterRespostaCorreta(int indice)
    {
        if (indice < posicoesRespostasCorretas.Length)
        {
            return alternativas[indice][posicoesRespostasCorretas[indice]].Substring(0, 1);
        }
        else
        {
            return ""; // Sem resposta correta para o índice fornecido
        }
    }

    public string PerguntaAtual()
    {
        if (indicePergunta < perguntas.Length)
        {
            return perguntas[indicePergunta];
        }
        else
        {
            timer.Start();
        }
    }

```

```

        resultadosForm = new ResultadoFormSuit(contadorRespostas);
        resultadosForm.ShowDialog();
        questionarioForm.Hide();
        return "";
    }
}
private void Timer_Tick(object sender, EventArgs e)
{
    resultadosForm.Close();
    obraSuit = new ObraSuits();
    obraSuit.Show();
    timer.Stop();
}
public string[] AlternativasAtual()
{
    if (indicePergunta < alternativas.Length)
    {
        return alternativas[indicePergunta];
    }
    else
    {
        return null;
    }
}
public void ProximaPergunta()
{
    indicePergunta++;
}
public void AtualizarPergunta(Label lblPergunta, Label lblNumeroPergunta, Button[] botoesAlternativas)
{
    lblNumeroPergunta.Text = "Pergunta " + (indicePergunta + 1);
    lblPergunta.Text = PerguntaAtual();
    string[] alternativas = AlternativasAtual();

    if (alternativas != null && alternativas.Length == 5 && botoesAlternativas.Length >= 5)
    {
        for (int i = 0; i < 5; i++)
        {
            botoesAlternativas[i].Text = alternativas[i];
        }
    }
}
}
}

```

```

using TotemPIMApresentacao.View;
using TotemPIMApresentacao.Model;

```

```

namespace TotemPIMApresentacao.Controller
{
    public class PessoaController
    {
        private string nome;
        private string codigo;

        public PessoaController()
        {

```



```
}

public PessoaController(string nome, string codigo)
{
    this.nome = nome;
    this.codigo = codigo;
}

public string Nome
{
    get
    {
        return nome;
    }
    set
    {
        nome = value;
    }
}

public stringCodigo
{
    get
    {
        return codigo;
    }
    set
    {
        codigo = value;
    }
}

}
```

APÊNDICE E – CÓDIGO FONTE DO PROGRAMA DAS VENDAS

```
using PIM_III_ADS_VENDAS.Controller;
using PIM_III_ADS_VENDAS.Model;

namespace PIM_III_ADS_VENDAS.View
{
    public partial class CadastroVisitante : Form
    {
        private PessoaModel pessoaModel;
        private PessoaController pessoaController;
        private Teclado teclado;

        public CadastroVisitante()
        {
            InitializeComponent();
            this.pessoaModel = new PessoaModel();
            this.pessoaController = new PessoaController();
            this.WindowState = FormWindowState.Maximized;
        }

        private void TextBox1_Click(object sender, EventArgs e)
        {
            if (teclado == null)
            {
                teclado = new Teclado();
            }

            teclado.SetTargetTextBox(sender as TextBox);

            teclado.Show();
        }

        private void BtnCancelar_Click(object sender, EventArgs e)
        {
            LoginCompra loginCompra = new LoginCompra();
            loginCompra.Show();
            this.Close();
        }

        private void btnComprar_Click(object sender, EventArgs e)
        {
            pessoaController = new PessoaController(
                nome: txbNome.Text,
                idade: txbIdade.Text,
                email: txbEmail.Text,
                cep: txbCep.Text,
                codigo: "");

            pessoaModel.SalvaPessoa(pessoaController);

            if (pessoaModel.Mensagem.Equals("Cadastro realizado com sucesso."))
            {
                LimparCampos();
                MessageBox.Show(pessoaModel.Mensagem);
                Vendas vendas = new Vendas(pessoaController);
                vendas.Show();
            }
            else
            {
                MessageBox.Show(pessoaModel.Mensagem);
            }
            LimparCampos();
        }
    }
}
```

```

private void LimparCampos()
{
    txbCep.Text = string.Empty;
    txbEmail.Text = string.Empty;
    txbldade.Text = string.Empty;
    txbNome.Text = string.Empty;
}
}
}

using PIM_III_ADS_VENDAS.Controller;
using PIM_III_ADS_VENDAS.Model;

namespace PIM_III_ADS_VENDAS.View
{
    public partial class LoginCompra : Form
    {
        private PessoaModel pessoaModel;
        private PessoaController pessoaController;
        private Teclado teclado;

        public LoginCompra()
        {
            InitializeComponent();
            pessoaModel = new PessoaModel();
            pessoaController = new PessoaController();
            this.WindowState = FormWindowState.Maximized;
        }

        private void TextBox1_Click(object sender, EventArgs e)
        {
            if (teclado == null)
            {
                teclado = new Teclado();
            }

            teclado.SetTargetTextBox(sender as TextBox);

            teclado.Show();
        }

        private void btnComprar_Click(object sender, EventArgs e)
        {
            pessoaController.Nome = txbNome.Text;
            pessoaController.Email = txbEmail.Text;

            pessoaModel.LoginCompra(pessoaController);

            if (pessoaModel.Mensagem.Equals($"Olá, {pessoaController.Nome}! Bem-vindo(a)!"))
            {
                Vendas vendas = new Vendas(pessoaController);
                vendas.ShowDialog();
                this.Hide();
                MessageBox.Show(pessoaModel.Mensagem);
            }
            else
            {
                MessageBox.Show(pessoaModel.Mensagem);
            }
        }

        private void btnCadastrarSe_Click(object sender, EventArgs e)
        {
            CadastroVisitante cadastroVisitante = new CadastroVisitante();
            cadastroVisitante.ShowDialog();
        }
    }
}

```

```

        this.Hide();
    }
}

using PIM_III_ADS_VENDAS.Controller;
using PIM_III_ADS_VENDAS.Model;

namespace PIM_III_ADS_VENDAS.View
{
    internal partial class Pagamento : Form
    {
        private PagamentoController pagamentoController;
        private VendasModel vendasModel;
        private VendasController vendasController;
        private PessoaController pessoaController;
        private LoginCompra loginCompra;

        public Pagamento(VendasController vendasController, PessoaController pessoaController)
        {
            InitializeComponent();

            pagamentoController = new PagamentoController();
            this.vendasController = vendasController;
            this.pessoaController = pessoaController;

            loginCompra = new LoginCompra();
            vendasModel = new VendasModel(vendasController, pagamentoController, pessoaController);

            this.StartPosition = FormStartPosition.CenterScreen;
            pagamentoController.PtbQrCode = ptbQrCode;
        }

        private void BtnCredito_Click(object sender, EventArgs e)
        {
            pagamentoController.Credito = true;
            pagamentoController.Debito = false;
            pagamentoController.Pix = false;

            vendasModel.SalvarVenda();
            loginCompra.Show();
            this.Close();
        }

        private void BtnDebito_Click(object sender, EventArgs e)
        {
            pagamentoController.Credito = false;
            pagamentoController.Debito = true;
            pagamentoController.Pix = false;

            vendasModel.SalvarVenda();
            loginCompra.Show();
            this.Close();
        }

        private async void BtnPix_Click(object sender, EventArgs e)
        {
            pagamentoController.Credito = false;
            pagamentoController.Debito = false;
            pagamentoController.Pix = true;

```

```

        pagamentoController.GerarQrCoder();

        vendasModel.SalvarVenda();

        await Task.Delay(4000);

        loginCompra.Show();
        this.Close();
    }

    private void BtnCancelar_Click(object sender, EventArgs e)
    {
        loginCompra.Show();
        this.Close();
    }
}

using PIM_III_ADS_VENDAS.Controller;
using PIM_III_ADS_VENDAS.Model;

namespace PIM_III_ADS_VENDAS.View
{
    public partial class Vendas : Form
    {
        private VendasController vendasController;
        private Pagamento pagamento;
        private PessoaController pessoaController;
        private VendasModel vendasModel;
        private PagamentoController pagamentoController;

        public Vendas(PessoaController pessoaController)
        {
            InitializeComponent();

            vendasController = new VendasController();
            pagamentoController = new PagamentoController();
            this.pessoaController = pessoaController;
            this.WindowState = FormWindowState.Maximized;
            pagamento = new Pagamento(vendasController, pessoaController);
            vendasModel = new VendasModel(vendasController, pagamentoController, pessoaController);
        }

        private void btnInteiro_Click(object sender, EventArgs e)
        {
            AtualizarTipoIngresso(true, false, false);
        }

        private void btnMeia_Click(object sender, EventArgs e)
        {
            AtualizarTipoIngresso(false, true, false);
        }

        private void btnIsento_Click(object sender, EventArgs e)
        {
            if (pessoaController.IdadeDb > 70)
            {
                AtualizarTipoIngresso(false, false, true);
            }
        }
    }
}

```

```

        vendasModel.SalvarVenda();
    }
    else
    {
        MessageBox.Show("A opção 'Isento' está disponível apenas para pessoas com mais de 70 anos.");
    }
    this.Close();
}

private void AtualizarTipoIngresso(bool inteiro, bool meia, bool isento)
{
    vendasController.Inteiro = inteiro;
    vendasController.Meia = meia;
    vendasController.Isento = isento;
    pagamento.ShowDialog();
    this.Close();
}
}
}
}

```

```

using Newtonsoft.Json.Linq;
using PIM_III_ADS_VENDAS.Controller;
using PIM_III_ADS_VENDAS.Service;
using System.Net;
using System.Text.RegularExpressions;

```

```

namespace PIM_III_ADS_VENDAS.Utills
{
    public class Validacao
    {
        private readonly PessoaService pessoaService;
        private string mensagem;

        public Validacao()
        {
            pessoaService = new PessoaService(new Dbconexao(), new EnviaEmail());
        }

        public Validacao(PessoaController pessoa)
        {
            ValidarPessoa(pessoa);
        }

        internal void ValidarPessoa(PessoaController pessoa)
        {
            mensagem = "";
            ValidarCep(pessoa);
            ValidarEmail(pessoa);
            ValidarIdade(pessoa);
            ValidarNome(pessoa);
        }

        private void ValidarNome(PessoaController pessoa)
        {
            if (string.IsNullOrEmpty(pessoa.Nome))
            {
                setMensagem("Por favor, insira um nome.");
            }

            if (pessoa.Nome.Any(char.IsDigit))
            {
                setMensagem("O nome não pode conter números.");
            }
        }
    }
}

```

```

        if (!Regex.IsMatch(pessoa.Nome, @"^[A-Z][a-zA-Z\s]*$"))
        {
            setMensagem("O nome deve começar com uma letra maiúscula e conter apenas letras e espaços.");
        }

        pessoa.Nome = pessoa.Nome;
    }

    private void ValidarIdade(PessoaController pessoa)
    {
        if (string.IsNullOrEmpty(pessoa.Idade))
        {
            setMensagem("Por favor, insira uma idade válida.");
            return;
        }
        if (!int.TryParse(pessoa.Idade, out int idade))
        {
            setMensagem("A idade deve ser um número inteiro válido.");
            return;
        }

        if (idade < 12)
        {
            setMensagem("Menores de 12 anos não podem entrar desacompanhados.");
            return;
        }

        if (idade > 120)
        {
            setMensagem("Por favor, insira uma idade válida.");
            return;
        }

        pessoa.IdadeDb = idade;
    }

    public void ValidarEmail(PessoaController pessoa)
    {
        if (string.IsNullOrEmpty(pessoa.Email))
        {
            setMensagem("Por favor, insira um endereço de e-mail.");
            return;
        }

        string pattern = @"^[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+$";
        if (!Regex.IsMatch(pessoa.Email, pattern))
        {
            setMensagem("Por favor, insira um endereço de e-mail válido.");
            return;
        }

        var pessoaExistente = pessoaServico.BuscarPessoaPorEmail(pessoa.Email);
        if (pessoaExistente != null)
        {
            setMensagem("Este endereço de e-mail já está em uso. Por favor, escolha outro.");
        }
        return;
    }

    private void ValidarCep(PessoaController pessoa)
    {
        if (string.IsNullOrEmpty(pessoa.Cep) || !Regex.IsMatch(pessoa.Cep, @"^\d{8}$"))
    }

```

```

        {
            setMensagem("Por favor, insira um CEP válido (formato: 00000000)");
        }
        else
        {
            using (WebClient webClient = new WebClient())
            {
                string url = $"https://viacep.com.br/ws/{pessoa.Cep}/json/";
                string response = webClient.DownloadString(url);
                dynamic jsonObject = JObject.Parse(response);

                if (jsonObject["erro"] != null)
                {
                    setMensagem("CEP não encontrado. Por favor, insira um CEP válido.");
                }
            }
        }

        pessoa.Cep = pessoa.Cep;
    }

    public string Mensagem
    {
        get { return mensagem; }
    }

    public void setMensagem(string mensagem)
    {
        this.mensagem = mensagem;
    }

}

}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Mail;
using System.Net;
using PIM_III_ADS_VENDAS.Service;

namespace PIM_III_ADS_VENDAS.Utils
{
    public class EnviaEmail
    {
        public void EnviarEmail(string nome, string destinatario, string codigo, string formaDePagamento, string
        valorIngresso)
        {
            MailMessage message = new MailMessage();
            try
            {
                SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);
                smtp.EnableSsl = true;
                smtp.Timeout = 60 * 1000;
                smtp.UseDefaultCredentials = false;
                smtp.Credentials = new NetworkCredential("pim3ads@gmail.com", "q n a m n b c j e r x f a h r r");

                message.From = new MailAddress("pim3ads@gmail.com");

                message.Body = message.Body = message.Body = "<html>" +
                "<head>" +
                "<style>" +

```



```

"body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; background-color: #000048;
color: #fff; }" +
".container { max-width: 600px; margin: 0 auto; padding: 20px; background-color: #f8f8f8; border-
radius: 10px; }" +
".header { background-color: #22128e; color: #fff; text-align: center; padding: 10px; border-top-left-
radius: 10px; border-top-right-radius: 10px; }" +
".content { padding: 20px; }" +
".footer { background-color: #22128e; color: #fff; text-align: center; padding: 10px; border-bottom-left-
radius: 10px; border-bottom-right-radius: 10px; }" +
".content p { font-size: 18px; }" +
"</style>" +
"</head>" +
"<body>" +
"<div class='container'>" +
"<div class='header'>" +
"<h1>Experiência Única!</h1>" +
"</div>" +
"<div class='content'>" +
"$"<p>Olá, {nome}</p>" +
"<p>Embarque em uma viagem incrível pela história da conquista da Lua no Museu da Viagem do
Homem à Lua!</p>" +
"$"<p>Seu código de visitante é: {codigo}</p>" +
"$"<p>Forma de pagamento: {formaDePagamento}</p>" +
"$"<p>Valor do ingresso: R${valorIngresso}</p>" +
"</div>" +
"<div class='footer'>" +
"<h2>Não perca esta oportunidade!</h2>" +
"</div>" +
"</div>" +
"</body>" +
"</html>";
message.Subject = "Seu Código de Visitante";
message.IsBodyHtml = true;
message.Priority = MailPriority.Normal;
message.To.Add(new MailAddress(destinatario));

smtp.Send(message);
}
catch (Exception Erro)
{

    MessageBox.Show("Erro ao enviar e-mail!");
}
}
}
}
}

```

```

using Npgsql;
using PIM_III_ADS_VENDAS.Utills;

namespace PIM_III_ADS_VENDAS.Service
{
    internal class VendasService
    {
        private Dbconexao dbconexao = new Dbconexao();
        private NpgsqlConnection conexao;

        internal VendasService()
        {
            conexao = dbconexao.GetConnection() as NpgsqlConnection;
        }
    }
}

```

```

        internal void RegistrarVenda(string pagamentoAtual, string codigoUsuario, string vendaAtual, string
e_mail, string nome, string valorIngresso)
        {

            EnviaEmail enviaEmail = new EnviaEmail();

            using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO
public.vendas(formadepagamento, data, codigo)
                        VALUES (@formadepagamento, @Data, @CodigoPessoa)", conexao))
            {
                command.Parameters.AddWithValue("@CodigoPessoa", codigoUsuario);
                command.Parameters.AddWithValue("@ingresso", vendaAtual);
                command.Parameters.AddWithValue("@formadepagamento", pagamentoAtual);
                command.Parameters.AddWithValue("@Data", DateTime.Now);

                int linhasAfetadas = command.ExecuteNonQuery();

                if (linhasAfetadas < 0)
                {
                    MessageBox.Show("Falha ao registrar o voto.");
                }

                enviaEmail.EnviaEmail(nome, e_mail, codigoUsuario, pagamentoAtual, valorIngresso);
            }

        }
    }
}

```

```

using Npgsql;
using Dapper;
using PIM_III_ADS_VENDAS.Utills;
using PIM_III_ADS_VENDAS.Controller;
using PIM_III_ADS_VENDAS.Service;

```

```

namespace PIM_III_ADS_VENDAS.Service

```

```

{
    public class PessoaService
    {
        private readonly Dbconexao dbconexao;
        private readonly EnviaEmail enviaEmail;
        private EnviaEmail enviaEmail1;

        public PessoaService(Dbconexao conexao, EnviaEmail enviaEmail)
        {
            this.dbconexao = conexao;
            this.enviaEmail = enviaEmail;
        }

        public void Inserir(PessoaController pessoa)
        {
            using (NpgsqlConnection conexao = dbconexao.GetConnection() as NpgsqlConnection)
            {
                conexao.Execute(@"INSERT INTO visitante (NOME, IDADE, EMAIL, CEP, DATA, Codigo)
                VALUES (@Nome, @IdadeDb, @Email, @Cep, @Data, @Codigo)",
                new
                {
                    pessoa.IdadeDb,
                    pessoa.Nome,
                    pessoa.Email,
                    pessoa.Cep,
                    pessoa.Codigo,
                    Data = DateTime.Today
                }
            )
            }
        }
    }
}

```

```

    });

    // enviaEmail.EnviaEmail(pessoa.Email, pessoa.Codigo);
}
}

public PessoaController BuscarPessoaPorEmail(string email)
{
    using (NpgsqlConnection conexao = dbconexao.GetConnection() as NpgsqlConnection)
    {
        return conexao.QueryFirstOrDefault<PessoaController>("SELECT * FROM visitante WHERE email = @Email", new { Email = email });
    }
}

public PessoaController BuscarPessoaPorNomeEEmail(PessoaController pessoa)
{
    using (var conexao = new Dbconexao())
    {
        var connection = conexao.GetConnection();

        var resultado = connection.QueryFirstOrDefault(
            "SELECT nome, idade, email, cep, data, codigo FROM public.visitante WHERE nome = @Nome AND email = @Email",
            new { pessoa.Nome, pessoa.Email });

        if (resultado != null)
        {
            pessoa.Nome = resultado.nome;
            pessoa.IdadeDb = resultado.idade;
            pessoa.Email = resultado.email;
            pessoa.Cep = resultado.cep;
            pessoa.Data = resultado.data;
            pessoa.Codigo = resultado.codigo;

            return pessoa;
        }

        return null;
    }
}
}
}

using Npgsql;
using System;
using System.Data;

namespace PIM_III_ADS_VENDAS.Service
{
    public class Dbconexao : IDisposable
    {
        private NpgsqlConnection connection;

        public Dbconexao()
        {
            connection = new NpgsqlConnection("Server=localhost;Port=5432;Database=DB_Museu;User Id=postgres;Password=4852;");
        }
    }
}

```

```

public NpgsqlConnection GetConnection()
{
    if (connection.State != ConnectionState.Open)
    {
        try
        {
            connection.Open();
        }
        catch (Exception ex)
        {
            // Lidar com a exceção ou relançá-la, conforme necessário
            throw new Exception("Erro ao abrir a conexão com o banco de dados.", ex);
        }
    }

    return connection;
}

public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);
}

protected virtual void Dispose(bool disposing)
{
    if (disposing)
    {
        if (connection != null)
        {
            if (connection.State != ConnectionState.Closed)
            {
                connection.Close();
            }
            connection.Dispose();
            connection = null;
        }
    }
}
}
}
}

using PIM_III_ADS_VENDAS.Controller;
using PIM_III_ADS_VENDAS.Service;
using PIM_III_ADS_VENDAS.View;

namespace PIM_III_ADS_VENDAS.Model
{
    internal class VendasModel
    {
        private VendasService vendasService = new VendasService();
        private PagamentoController pagamentoController;
        private VendasController vendasController;
        private PessoaController pessoaController;

        public VendasModel(VendasController vendasController, PagamentoController pagamentoController,
            PessoaController pessoaController)
        {
            this.vendasController = vendasController;
            this.pagamentoController = pagamentoController;

```

```

        this.pessoaController = pessoaController;
    }

    public void SalvarVenda()
    {
        string pagamentoAtual = pagamentoController.FormaDePagamento();
        string vendaAtual = vendasController.TipoDeIngresso();
        string codigoUsuario = pessoaController.Codigo;
        string e_mail = pessoaController.Email;
        string nome = pessoaController.Nome;
        string valorIngresso = vendasController.Valor();

        vendasService.RegistrarVenda(pagamentoAtual, codigoUsuario, vendaAtual, e_mail, nome,
        valorIngresso);
    }

    }

}

using PIM_III_ADS_VENDAS.Utills;
using PIM_III_ADS_VENDAS.Controller;
using PIM_III_ADS_VENDAS.Service;

using PIM_III_ADS_VENDAS.Service;

namespace PIM_III_ADS_VENDAS.Model
{
    public class PessoaModel
    {
        Validacao validacao = new Validacao();
        PessoaService pessoaServico = new PessoaService(new Dbconexao(), new EnviaEmail());
        PessoaController pessoa;
        private string mensagem = "";

        public PessoaModel()
        {
        }

        public PessoaModel(PessoaController pessoa)
        {
            this.SalvaPessoa(pessoa);
            this.LoginCompra(pessoa);
        }

        internal void SalvaPessoa(PessoaController pessoa)
        {
            validacao.ValidarPessoa(pessoa);

            if (validacao.Mensagem.Equals(""))
            {
                pessoaServico.Inserir(pessoa);
                this.mensagem = "Cadastro realizado com sucesso.";
            }
            else
            {
                this.mensagem = validacao.Mensagem;
            }
        }

        internal void LoginCompra(PessoaController pessoa)

```

```

        {
            pessoa = pessoaServico.BuscarPessoaPorNomeEEmail(pessoa);

            if (pessoa != null && pessoa.Nome != null && pessoa.Email != null)
            {
                this.mensagem = $"Olá, {pessoa.Nome}! Bem-vindo(a)!";
            }
            else
            {
                this.mensagem = "Login inválido. Verifique suas credenciais.";
            }
        }

        public string Mensagem
        {
            get { return mensagem; }
        }
        public PessoaService PessoaService
        {
            get { return pessoaServico; }
            set { pessoaServico = value; }
        }

        public PessoaService PessoaService1
        {
            get => default;
            set
            {
            }
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PIM_III_ADS_VENDAS.Controller
{
    internal class VendasController
    {
        private bool inteiro;
        private bool meia;
        private bool isento;

        public VendasController()
        {
        }

        public VendasController(bool inteiro, bool meia, bool isento)
        {
            this.inteiro = inteiro;
            this.meia = meia;
            this.isento = isento;
        }

        public string TipoDeIngresso()

```

```

    {
        if (Inteiro)
        {
            return "Inteiro";
        }
        else if (Meia)
        {
            return "Meia";
        }
        else
        {
            return "Isento";
        }
    }

    public string Valor()
    {
        if (Inteiro)
        {
            return "10.00";
        }
        else if (Meia)
        {
            return "5.00";
        }
        else if (Isento)
        {
            return "0.00";
        }

        return "0.00";
    }

    public bool Isento
    {
        get
        {
            return isento;
        }
        set
        {
            isento = value;
        }
    }

    public bool Meia
    {
        get
        {
            return meia;
        }
        set
        {
            meia = value;
        }
    }

    public bool Inteiro
    {
        get
        {
            return inteiro;
        }
        set
        {

```

```

        inteiro = value;
    }
}

internal View.Pagamento Pagamento
{
    get => default;
    set
    {
    }
}
}

namespace PIM_III_ADS_VENDAS.Controller
{
    public class PessoaController
    {
        private int idadeDb;
        private string nome;
        private string idade;
        private string email;
        private string cep;
        private string codigo;
        public DateTime data;

        public PessoaController()
        {
        }

        public PessoaController(string nome, string idade, string email, string cep, string codigo, int idadeDb = 0)
        {
            this.nome = nome;
            this.idade = idade;
            this.email = email;
            this.cep = cep;
            this.codigo = GerarCodigo(nome);
            this.idadeDb = idadeDb;
        }

        public PessoaController(string codigo, string email)
        {
            this.codigo = codigo;
            this.email = email;
        }

        private string GerarCodigo(string nome)
        {
            string iniciais = string.Join("", nome.Split(' ').Select(s => s[0]));

            Random rnd = new Random();
            int numero = rnd.Next(10, 100);

            char letra = (char)rnd.Next('A', 'Z');

            string codigo = $"{iniciais}{numero}{letra}{numero}";

            if (codigo.Length > 5)
            {

```



```
        codigo = codigo.Substring(0, 5);
    }
    return codigo;
}

public string Idade
{
    get { return idade; }
    set { idade = value; }
}

public string Codigo
{
    get
    {
        return codigo;
    }
    set
    {
        codigo = value;
    }
}

public string Cep
{
    get { return cep; }
    set { cep = value; }
}

public string Email
{
    get
    {
        return email;
    }
    set
    {
        email = value;
    }
}

public int IdadeDb
{
    get { return idadeDb; }
    set { idadeDb = value; }
}

public string Nome
{
    get { return nome; }
    set { nome = value; }
}

public DateTime Data
{
    get { return data; }
    set { data = value; }
}

}
}
```

```

using QRCode;
using System.Windows.Forms;

using System.Security.Policy;

namespace PIM_III_ADS_VENDAS.Controller
{
    public class PagamentoController
    {
        private bool credito;
        private bool debito;
        private bool pix;

        private PictureBox ptbQrCode;

        public PagamentoController()
        {
        }

        public PagamentoController(PictureBox pictureBox)
        {
            this.ptbQrCode = ptbQrCode;
        }

        public PagamentoController(bool credito, bool debito, bool pix)
        {
            this.credito = credito;
            this.debito = debito;
            this.pix = pix;
        }

        public string FormaDePagamento()
        {
            if (Credito)
            {
                return "Credito";
            }
            else if (Debito)
            {
                return "Debito";
            }
            else if (Pix)
            {
                return "Pix";
            }
            else
            {
                return "Isento";
            }
        }

        public void GerarQrCoder()
        {
            Url generator = new Url("");

            string payload = generator.ToString();

            QRCodeGenerator qRCodeGenerator = new QRCodeGenerator();

```

```

        QRCodeData qRCodeData = qRCodeGenerator.CreateQRCode(payload,
QRCodeGenerator.ECCLevel.Q);
        QRCode qRCode = new QRCode(qRCodeData);
        ptbQRCode.Image = qRCode.GetGraphic(16);
    }

    public PictureBox PtbQRCode
    {
        get { return ptbQRCode; }
        set { ptbQRCode = value; }
    }

    public bool Pix
    {
        get
        {
            return pix;
        }
        set
        {
            pix = value;
        }
    }

    public bool Debito
    {
        get
        {
            return debito;
        }
        set
        {
            debito = value;
        }
    }

    public bool Credito
    {
        get
        {
            return credito;
        }
        set
        {
            credito = value;
        }
    }
    }
}

```

APÊNDICE F – CÓDIGO FONTE DO PROGRAMA DO ADM

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PIM_III_ADS_ADM.Controller
{
    internal class VendasController
    {
        private bool inteiro;
        private bool meia;
        private bool isento;

        public VendasController()
        {
        }

        public VendasController(bool inteiro, bool meia, bool isento)
        {
            this.inteiro = inteiro;
            this.meia = meia;
            this.isento = isento;
        }

        public string TipoDeIngresso()
        {
            if (Inteiro)
            {
                return "Inteiro";
            }
            else if (Meia)
            {
                return "Meia";
            }
            else
            {
                return "Isento";
            }
        }

        public string Valor()
        {
            if (Inteiro)
            {
                return "10.00";
            }
            else if (Meia)
            {
                return "5.00";
            }
            else if (Isento)
            {
                return "0.00";
            }

            return "0.00";
        }

        public bool Isento
```

```

    {
        get
        {
            return isento;
        }
        set
        {
            isento = value;
        }
    }

    public bool Meia
    {
        get
        {
            return meia;
        }
        set
        {
            meia = value;
        }
    }

    public bool Inteiro
    {
        get
        {
            return inteiro;
        }
        set
        {
            inteiro = value;
        }
    }
}
}

using PIM_III_ADS_ADM.Model;

namespace PIM_III_ADS_ADM.Controller
{
    public class PessoaController
    {
        private int idadeDb;
        private string nome;
        private string idade;
        private string email;
        private string cep;
        private string codigo;
        public DateTime data;

        private PessoaModel pessoaModel;

        public PessoaController()
        {
        }

        public PessoaController(string nome, string idade, string email, string cep, string codigo = null, int idadeDb = 0)
        {
            this.nome = nome;

```

```

        this.idade = idade;
        this.email = email;
        this.cep = cep;
        this.codigo = string.IsNullOrEmpty(codigo) ? GerarCodigo(nome) : codigo;
        this.idadeDb = idadeDb;
    }

    public PessoaController(string codigo, string email)
    {
        this.codigo = codigo;
        this.email = email;
    }

    public string GerarCodigo(string nome)
    {
        // Obtém as iniciais do nome
        string iniciais = string.Join("", nome.Split(' ').Select(s => s[0]));

        // Gera um número aleatório entre 10 e 99
        Random rnd = new Random();
        int numero = rnd.Next(10, 100);

        // Gera uma letra aleatória
        char letra = (char)rnd.Next('A', 'Z');

        // Concatena as partes para formar o código
        string codigo = $"{iniciais}{numero}{letra}{numero}";

        if (codigo.Length > 5)
        {
            codigo = codigo.Substring(0, 5);
        }
        return codigo;
    }

    public string Idade
    {
        get { return idade; }
        set { idade = value; }
    }

    public string Codigo
    {
        get { return codigo; }
        set { codigo = value; }
    }

    public string Cep
    {
        get { return cep; }
        set { cep = value; }
    }

    public string Email
    {
        get
        {
            return email;
        }
        set
        {
            email = value;
        }
    }

```

```

    }

    public int IdadeDb
    {
        get { return idadeDb; }
        set { idadeDb = value; }
    }

    public string Nome
    {
        get { return nome; }
        set { nome = value; }
    }

    public DateTime Data
    {
        get { return data; }
        set { data = value; }
    }

    }
}

using QRCode;
using System.Windows.Forms;

using System.Security.Policy;

namespace PIM_III_ADS_ADM.Controller
{
    public class PagamentoController
    {
        private bool credito;
        private bool debito;
        private bool pix;

        private PictureBox ptbQrCode;

        public PagamentoController()
        {
        }

        public PagamentoController(PictureBox pictureBox)
        {
            this.ptbQrCode = ptbQrCode;
        }

        public PagamentoController(bool credito, bool debito, bool pix)
        {
            this.credito = credito;
            this.debito = debito;
            this.pix = pix;
        }

        public string FormaDePagamento()
        {
            if (Credito)

```

```

        {
            return "Credito";
        }
        else if (Debito)
        {
            return "Debito";
        }
        else if (Pix)
        {
            return "Pix";
        }
        else
        {
            return "Isento";
        }
    }

    public void GerarQrCoder()
    {
        Url generator = new Url("");

        string payload = generator.ToString();

        QRCodeGenerator qRCodeGenerator = new QRCodeGenerator();
        QRCodeData qRCodeData = qRCodeGenerator.CreateQrCode(payload,
QRCodeGenerator.ECCLLevel.Q);
        QRCode qRCode = new QRCode(qRCodeData);
        ptbQrCode.Image = qRCode.GetGraphic(16);
    }

    public PictureBox PtbQrCode
    {
        get { return ptbQrCode; }
        set { ptbQrCode = value; }
    }

    public bool Pix
    {
        get
        {
            return pix;
        }
        set
        {
            pix = value;
        }
    }

    public bool Debito
    {
        get
        {
            return debito;
        }
        set
        {
            debito = value;
        }
    }

    public bool Credito
    {
        get
        {
            return credito;
        }
        set
    }

```



```

        {
            credito = value;
        }
    }
}

using PIM_III_ADS_ADM.Controller;
using PIM_III_ADS_ADM.Service;
using PIM_III_ADS_ADM.View;

namespace PIM_III_ADS_ADM.Model
{
    internal class VendasModel
    {
        private VendasService vendasService = new VendasService();
        private PagamentoController pagamentoController;
        private VendasController vendasController;
        private PessoaController pessoaController;

        public VendasModel(VendasController vendasController, PagamentoController pagamentoController,
            PessoaController pessoaController)
        {
            this.vendasController = vendasController;
            this.pagamentoController = pagamentoController;
            this.pessoaController = pessoaController;
        }

        public void SalvarVenda()
        {
            string pagamentoAtual = pagamentoController.FormaDePagamento();
            string vendaAtual = vendasController.TipoDeIngresso();
            string codigoUsuario = pessoaController.Codigo;
            string e_mail = pessoaController.Email;
            string nome = pessoaController.Nome;
            string valorIngresso = vendasController.Valor();

            vendasService.RegistrarVenda(pagamentoAtual, codigoUsuario, vendaAtual, e_mail, nome,
                valorIngresso);
        }

    }
}

using PIM_III_ADS_ADM.Controller;
using PIM_III_ADS_ADM.Service;
using PIM_III_ADS_ADM.Utills;
using PIM_III_ADS_ADM.Service;

namespace PIM_III_ADS_ADM.Model
{
    public class PessoaModel
    {
        Validacao validacao = new Validacao();
        PessoaServico pessoaServico = new PessoaServico(new Dbconexao(), new EnviaEmail());
    }
}

```

```

PessoaController pessoaController;
private string mensagem = "";

public PessoaModel()
{
}

public PessoaModel(PessoaController pessoaController)
{
    this.SalvaPessoa(pessoaController);
    this.AtualizarPessoa(pessoaController);
    this.DeletarPessoa(pessoaController);
    this.BuscarPessoa(pessoaController);
    this.BuscarTodasPessoas();
}

internal void SalvaPessoa(PessoaController pessoaController)
{
    validacao.ValidarPessoa(pessoaController);

    if (validacao.Mensagem.Equals(""))
    {
        pessoaServico.Inserir(pessoaController);
        this.mensagem = "Cadastro realizado com sucesso.";
    }
    else
    {
        this.mensagem = validacao.Mensagem;
    }
}

internal void AtualizarPessoa(PessoaController pessoaController)
{
    validacao.ValidarPessoa(pessoaController);

    if (pessoaController != null)
    {
        pessoaServico.Atualizar(pessoaController);
        this.mensagem = "Pessoa atualizada com sucesso!";
    }
    else
    {
        this.mensagem = "Erro ao atualizar pessoa: pessoa não encontrada";
    }
}

internal void DeletarPessoa(PessoaController pessoaController)
{
    if (pessoaController.Codigo.Equals(""))
    {
        this.mensagem = "Erro ao Remover pessoa: pessoa não encontrada";
    }
    else
    {
        pessoaServico.Deletar(pessoaController);
        this.mensagem = "Pessoa excluída com sucesso!";
    }
}

internal void BuscarPessoa(PessoaController pessoaController)
{
    pessoaController = pessoaServico.BuscarPessoaPorNomeEEEmail(pessoaController);

    if (pessoaController != null && pessoaController.Nome != null && pessoaController.Email != null)
    {
        this.mensagem = "Pessoa encontrada com sucesso!";
    }
}

```

```

    }
    else
    {
        this.mensagem = "Login inválido. Verifique suas credenciais.";
    }
}

internal List<PessoaController> BuscarTodasPessoas()
{
    IEnumerable<PessoaController> todasPessoas = pessoaServico.BuscarTodasPessoas();
    return todasPessoas.ToList();
}

public void ConfigurarColunas(DataGridView dataGridView, List<PessoaController> todasPessoas)
{
    dataGridView.DataSource = todasPessoas;

    dataGridView.Columns["codigo"].HeaderText = "Código";
    dataGridView.Columns["codigo"].DisplayIndex = 0;

    dataGridView.Columns["nome"].HeaderText = "Nome";
    dataGridView.Columns["nome"].DisplayIndex = 1;

    dataGridView.Columns["idadeDb"].HeaderText = "Idade";
    dataGridView.Columns["idadeDb"].DisplayIndex = 2;

    dataGridView.Columns["email"].HeaderText = "E-mail";
    dataGridView.Columns["email"].DisplayIndex = 3;

    dataGridView.Columns["cep"].HeaderText = "CEP";
    dataGridView.Columns["cep"].DisplayIndex = 4;

    // Oculta a coluna que deseja remover
    //dataGridView.Columns["Validacao"].Visible = false;
    dataGridView.Columns["idadeDb"].Visible = false;
}

public string Mensagem
{
    get { return mensagem; }
}

public PessoaServico PessoaServico
{
    get { return pessoaServico; }
    set { pessoaServico = value; }
}
}
}

using Npgsql;
using PIM_III_ADS_ADM.Utills;
using System;
using System.Windows.Forms;

namespace PIM_III_ADS_ADM.Service
{
    internal class VendasService
    {
        private Dbconexao dbconexao = new Dbconexao();
    }
}

```

```

private NpgsqlConnection conexao;

internal VendasService()
{
    conexao = dbconexao.GetConnection() as NpgsqlConnection;
}

internal void RegistrarVenda(string pagamentoAtual, string codigoUsuario, string vendaAtual, string
e_mail, string nome, string valorIngresso)
{
    // Verifica se codigoUsuario não é nulo ou vazio
    if (string.IsNullOrEmpty(codigoUsuario))
    {
        MessageBox.Show("Código do usuário não pode ser nulo ou vazio.");
        return; // Sai do método para evitar a execução do código com um valor inválido
    }

    EnviaEmail enviaEmail = new EnviaEmail();

    using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO
public.tbl_vendas(formadepagamento, data, codigo)
VALUES (@formadepagamento, @Data,@CodigoPessoa)", conexao))
    {
        command.Parameters.AddWithValue("@CodigoPessoa", NpgsqlTypes.NpgsqlDbType.Text,
codigoUsuario);
        command.Parameters.AddWithValue("@vendas", vendaAtual);
        command.Parameters.AddWithValue("@formadepagamento", pagamentoAtual);
        command.Parameters.AddWithValue("@Data", DateTime.Now);

        int linhasAfetadas = command.ExecuteNonQuery();

        if (linhasAfetadas < 0)
        {
            MessageBox.Show("Falha ao registrar o voto.");
        }

        enviaEmail.EnviaEmail(nome, e_mail, codigoUsuario, pagamentoAtual, valorIngresso);
    }
}

}

}

using Dapper;
using Npgsql;
using PIM_III_ADS_ADM.Controller;
using PIM_III_ADS_ADM.Utills;
using System;
using System.Collections.Generic;
using System.Data;

namespace PIM_III_ADS_ADM.Service
{
    public class PessoaServico
    {
        private readonly Dbconexao dbconexao;
        private readonly EnviaEmail enviaEmail;
        private NpgsqlConnection conexao;

        public PessoaServico(Dbconexao conexao, EnviaEmail enviaEmail)
        {
            this.dbconexao = conexao;

```

```

        this.enviaEmail = enviaEmail;

        // Abra a conexão do banco de dados assim que o serviço for instanciado
        this.conexao = dbconexao.GetConnection() as NpgsqlConnection;
        if (this.conexao.State == ConnectionState.Closed)
            this.conexao.Open();
    }

    public void Inserir(PessoaController pessoa)
    {
        if (string.IsNullOrEmpty(pessoa.Codigo))
        {
            pessoa.Codigo = pessoa.GerarCodigo(pessoa.Nome);
        }
        // Use a conexão já aberta
        conexao.Execute(@"INSERT INTO tbl_visitante (NOME, IDADE, EMAIL, CEP, DATA, Codigo)
            VALUES (@Nome, @IdadeDb, @Email, @Cep, @Data, @Codigo)",
            new
            {
                pessoa.IdadeDb,
                pessoa.Nome,
                pessoa.Email,
                pessoa.Cep,
                pessoa.Codigo,
                Data = DateTime.Today
            });

        // enviaEmail.EnviaEmail(pessoa.Email, pessoa.Codigo);
    }

    public void Atualizar(PessoaController pessoa)
    {
        // Use a conexão já aberta
        conexao.Execute(@"UPDATE tbl_visitante SET nome = @Nome, idade = @Idade, email = @Email,
            cep = @Cep, data = @Data WHERE código = @Codigo",
            new
            {
                pessoa.Nome,
                pessoa.IdadeDb,
                pessoa.Email,
                pessoa.Cep,
                Data = DateTime.Today,
                Codigo = pessoa.Codigo
            });
    }

    public void Deletar(PessoaController pessoa)
    {
        // Use a conexão já aberta
        conexao.Execute("DELETE FROM tbl_visitante WHERE Codigo = @Codigo", new { pessoa.Codigo });
    }

    public IEnumerable<PessoaController> BuscarTodasPessoas()
    {
        // Use a conexão já aberta
        return conexao.Query<PessoaController>("SELECT código as Codigo, nome as Nome, idade as Idade, email as Email, cep as Cep, data as Data FROM tbl_visitante");
    }

    public PessoaController BuscarPessoaPorEmail(string email)
    {
        // Use a conexão já aberta
        return conexao.QueryFirstOrDefault<PessoaController>("SELECT * FROM tbl_visitante WHERE email = @Email", new { Email = email });
    }

```

```

public PessoaController BuscarPessoaPorNomeEEmail(PessoaController pessoa)
{
    // Use a conexão já aberta
    var resultado = conexao.QueryFirstOrDefault(
        "SELECT nome, idade, email, cep, codigo FROM public.tbl_visitante WHERE nome = @Nome AND
email = @Email",
        new { pessoa.Nome, pessoa.Email });

    if (resultado != null)
    {
        pessoa.Nome = resultado.nome;
        pessoa.IdadeDb = resultado.idade;
        pessoa.Email = resultado.email;
        pessoa.Cep = resultado.cep;
        pessoa.Codigo = resultado.codigo;

        return pessoa;
    }

    return null;
}
}
}

```

```

using Npgsql;
using System;
using System.Data;

```

```

namespace PIM_III_ADS_ADM.Service
{
    public class Dbconexao : IDisposable
    {
        private readonly string connectionString;
        private NpgsqlConnection connection;

        public Dbconexao()
        {
            connectionString = "Server=localhost;Port=5432;Database=DB_Museu;User
Id=postgres;Password=4852;";
        }

        public IDbConnection GetConnection()
        {
            if (connection == null)
            {
                connection = new NpgsqlConnection(connectionString);
                connection.Open(); // Abre a conexão assim que for criada
            }
            else if (connection.State == ConnectionState.Closed)
            {
                connection.Open(); // Abre a conexão se estiver fechada
            }

            return connection;
        }

        public void Dispose()
        {
            if (connection != null)
            {
                if (connection.State != ConnectionState.Closed)
                {

```

```

        connection.Close(); // Fecha a conexão se estiver aberta
    }
    connection.Dispose();
    connection = null;
}
}
}
}
}
}
}

```

```

using Newtonsoft.Json.Linq;
using PIM_III_ADS_ADM.Controller;
using PIM_III_ADS_ADM.Service;
using PIM_III_ADS_ADM.Service;
using System.Net;
using System.Text.RegularExpressions;

```

```

namespace PIM_III_ADS_ADM.Utils
{
    public class Validacao
    {
        private readonly PessoaServico pessoaServico;
        private string mensagem;

        public Validacao()
        {
            pessoaServico = new PessoaServico(new Dbconexao(), new EnviaEmail());
        }

        public Validacao(PessoaController pessoaController)
        {
            ValidarPessoa(pessoaController);
        }

        internal void ValidarPessoa(PessoaController pessoaController)
        {
            mensagem = "";
            ValidarCep(pessoaController);
            ValidarEmail(pessoaController);
            ValidarIdade(pessoaController);
            ValidarNome(pessoaController);
        }

        private void ValidarNome(PessoaController pessoaController)
        {
            if (string.IsNullOrEmpty(pessoaController.Nome))
            {
                setMensagem("Por favor, insira um nome.");
            }

            if (pessoaController.Nome.Any(char.IsDigit))
            {
                setMensagem("O nome não pode conter números.");
            }

            if (!Regex.IsMatch(pessoaController.Nome, @"^[A-Z][a-zA-Z\s]*$"))
            {
                setMensagem("O nome deve começar com uma letra maiúscula e conter apenas letras e espaços.");
            }
        }
    }
}

```

```

        pessoaController.Nome = pessoaController.Nome;
    }

    private void ValidarIdade(PessoaController pessoaController)
    {
        if (string.IsNullOrEmpty(pessoaController.Idade))
        {
            setMensagem("Por favor, insira uma idade válida.");
            return;
        }
        if (!int.TryParse(pessoaController.Idade, out int idade))
        {
            setMensagem("A idade deve ser um número inteiro válido.");
            return;
        }

        if (idade < 12)
        {
            setMensagem("Menores de 12 anos não podem entrar desacompanhados.");
            return;
        }

        if (idade > 120)
        {
            setMensagem("Por favor, insira uma idade válida.");
            return;
        }

        pessoaController.IdadeDb = idade;
    }

    public void ValidarEmail(PessoaController pessoaController)
    {
        if (string.IsNullOrEmpty(pessoaController.Email))
        {
            setMensagem("Por favor, insira um endereço de e-mail.");
            return;
        }

        string pattern = @"^[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\.$";
        if (!Regex.IsMatch(pessoaController.Email, pattern))
        {
            setMensagem("Por favor, insira um endereço de e-mail válido.");
            return;
        }

        // Verificar se o email já está em uso apenas ao salvar uma nova pessoaController
        if (pessoaController.Codigo == null)
        {
            var pessoaExistente = pessoaServico.BuscarPessoaPorEmail(pessoaController.Email);
            if (pessoaExistente != null)
            {
                setMensagem("Este endereço de e-mail já está em uso. Por favor, escolha outro.");
                return;
            }
        }
    }

    private void ValidarCep(PessoaController pessoaController)
    {
        if (string.IsNullOrEmpty(pessoaController.Cep) || !Regex.IsMatch(pessoaController.Cep,
@"^\d{8}$"))
        {
            setMensagem("Por favor, insira um CEP válido (formato: 00000000)");
        }
        else
    }

```



```

    {
        using (WebClient webClient = new WebClient())
        {
            string url = $"https://viacep.com.br/ws/{pessoaController.Cep}/json/";
            string response = webClient.DownloadString(url);
            dynamic jsonObject = JObject.Parse(response);

            if (jsonObject["erro"] != null)
            {
                setMensagem("CEP não encontrado. Por favor, insira um CEP válido.");
            }
        }
    }

    pessoaController.Cep = pessoaController.Cep;
}

public string Mensagem
{
    get { return mensagem; }
}

public void setMensagem(string mensagem)
{
    this.mensagem = mensagem;
}
}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Mail;
using System.Net;
using PIM_III_ADS_ADM.Service;

namespace PIM_III_ADS_ADM.Utils
{
    public class EnviaEmail
    {
        public void EnviarEmail(string nome, string destinatario, string codigo, string formaDePagamento, string valorIngresso)
        {
            MailMessage message = new MailMessage();
            try
            {
                SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);
                smtp.EnableSsl = true;
                smtp.Timeout = 60 * 1000;
                smtp.UseDefaultCredentials = false;
                smtp.Credentials = new NetworkCredential("pim3ads@gmail.com", "q n a m n b c j e r x f a h r r");

                message.From = new MailAddress("pim3ads@gmail.com");

                message.Body = message.Body = "<html>" +
                "<head>" +
                "<style>" +
                "body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; background-color: #000048; color: #fff; }" +
                ".container { max-width: 600px; margin: 0 auto; padding: 20px; background-color: #f8f8f8; border-radius: 10px; }" +

```

```

        ".header { background-color: #22128e; color: #fff; text-align: center; padding: 10px; border-top-left-
radius: 10px; border-top-right-radius: 10px; }" +
        ".content { padding: 20px; }" +
        ".footer { background-color: #22128e; color: #fff; text-align: center; padding: 10px; border-bottom-left-
radius: 10px; border-bottom-right-radius: 10px; }" +
        ".content p { font-size: 18px; }" +
        "</style>" +
        "</head>" +
        "<body>" +
        "<div class='container'>" +
        "<div class='header'>" +
        "<h1>Experiência Única!</h1>" +
        "</div>" +
        "<div class='content'>" +
        $"<p>Olá, {nome}</p>" +
        "<p>Embarque em uma viagem incrível pela história da conquista da Lua no Museu da Viagem do
Homem à Lua!</p>" +
        $"<p>Seu código de visitante é: {codigo}</p>" +
        $"<p>Forma de pagamento: {formaDePagamento}</p>" +
        $"<p>Valor do ingresso: R${valorIngresso}</p>" +
        "</div>" +
        "<div class='footer'>" +
        "<h2>Não perca esta oportunidade!</h2>" +
        "</div>" +
        "</div>" +
        "</body>" +
        "</html>";
        message.Subject = "Seu Código de Visitante";
        message.IsBodyHtml = true;
        message.Priority = MailPriority.Normal;
        message.To.Add(new MailAddress(destinatario));

        smtp.Send(message);
    }
    catch (Exception Erro)
    {
        MessageBox.Show("Erro ao enviar e-mail!");
    }
}
}
}
}
}

```

```
using PIM_III_ADS_ADM.Controller;
```

```
using PIM_III_ADS_ADM.Model;
```

```

namespace PIM_III_ADS_ADM.View
{
    public partial class Adm : Form
    {
        private PessoaModel pessoaModel;
        // private PessoaController pessoaController;

        private VendasController vendasController;
        private Pagamento pagamento;
        private PessoaController pessoaController;
        private VendasModel vendasModel;
    }
}

```

```

private PagamentoController pagamentoController;

public Adm(PessoaController pessoaController)
{
    InitializeComponent();
    pessoaModel = new PessoaModel();
    // pessoaController = new PessoaController();

    vendasController = new VendasController();
    pagamentoController = new PagamentoController();
    this.pessoaController = pessoaController; // Recebe o objeto PessoaController

    pagamento = new Pagamento(vendasController, pessoaController); // Passa o objeto PessoaController
    vendasModel = new VendasModel(vendasController, pagamentoController, pessoaController);

    this.WindowState = FormWindowState.Maximized;
    txbData.Text = DateTime.Today.ToString("dd/MM/yyyy");
}

private void BtnSalvar_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txbCodigo.Text))
    {
        pessoaController.Nome = txbNome.Text;
        pessoaController.Idade = txbIdade.Text;
        pessoaController.Email = txbEmail.Text;
        pessoaController.Cep = txbCep.Text;

        pessoaModel.SalvaPessoa(pessoaController);
        MessageBox.Show(pessoaModel.Mensagem);
    }
    else
    {
        pessoaController.Nome = txbNome.Text;
        pessoaController.Idade = txbIdade.Text;
        pessoaController.Email = txbEmail.Text;
        pessoaController.Cep = txbCep.Text;
        pessoaController.Codigo = txbCodigo.Text;

        pessoaModel.AtualizarPessoa(pessoaController);
        MessageBox.Show(pessoaModel.Mensagem);
    }
    RecarregarDadosDataGridView();
}

private void BtnRemover_Click(object sender, EventArgs e)
{
    pessoaController.Codigo = txbCodigo.Text;

    pessoaModel.DeletarPessoa(pessoaController);
    MessageBox.Show(pessoaModel.Mensagem);
    RecarregarDadosDataGridView();
}

private void BtnBuscar_Click(object sender, EventArgs e)
{
    pessoaController.Nome = txbNomeBuscar.Text;
    pessoaController.Email = txbEmailBuscar.Text;

    pessoaModel.BuscarPessoa(pessoaController);

    if (pessoaModel.Mensagem.Equals("Pessoa encontrada com sucesso!"))
    {
        txbCodigo.Text = pessoaController.Codigo;
        txbNome.Text = pessoaController.Nome;
        txbIdade.Text = pessoaController.IdadeDb.ToString();
        txbEmail.Text = pessoaController.Email;
    }
}

```

```

        txbCep.Text = pessoaController.Cep;
    }
    else
    {
        MessageBox.Show(pessoaModel.Mensagem);
    }
}

private void BtnRelatorios_Click(object sender, EventArgs e)
{
    string powerBiDesktopPath = @"C:\Program Files\Microsoft Power BI Desktop\bin\PBIDesktop.exe";
    string reportPath = @"C:\Users\kenzo\OneDrive\Documentos\PIM\RelatorioPowerBI\PIM-III-ADS.pbix";

    try
    {
        System.Diagnostics.Process.Start(powerBiDesktopPath, $"{reportPath}\");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro ao abrir o relatorio: " + ex.Message, "Erro", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

private void Adm_Load(object sender, EventArgs e)
{
    List<PessoaController> todasPessoas = pessoaModel.BuscarTodasPessoas();

    pessoaModel.ConfigurarColunas(dgvPessoa, todasPessoas);

    // Configuração para ajustar o tamanho da tabela ao DataGridView
    dgvPessoa.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    dgvPessoa.AutoSizeColumns();
}

private void RecarregarDadosDataGridView()
{
    PessoaController pessoa = new PessoaController();
    List<PessoaController> todasPessoas = pessoaModel.BuscarTodasPessoas();

    dgvPessoa.DataSource = todasPessoas;
}

private void btnInteiro_Click(object sender, EventArgs e)
{
    AtualizarTipoIngresso(true, false, false);
}

private void btnMeia_Click(object sender, EventArgs e)
{
    AtualizarTipoIngresso(false, true, false);
}

private void btnIsento_Click(object sender, EventArgs e)
{
    // Verifica se a idade é maior que 70
    if (pessoaController.IdadeDb > 70)
    {
        // Se a idade for maior que 70, define o tipo de ingresso como isento
        AtualizarTipoIngresso(false, false, true);
        vendasModel.SalvarVenda();
    }
    else
    {
        // Se a idade não for maior que 70, mostra uma mensagem de erro
        MessageBox.Show("A opção 'Isento' está disponível apenas para pessoas com mais de 70 anos.");
    }
}

```

```
    }  
}  
  
private void AtualizarTipoIngresso(bool inteiro, bool meia, bool isento)  
{  
    vendasController.Inteiro = inteiro;  
    vendasController.Meia = meia;  
    vendasController.Isento = isento;  
    pagamento.ShowDialog();  
}  
}  
}
```

APÊNDICE G – CÓDIGO FONTE DO PROGRAMA DA PESQUISA DE SATISFAÇÃO

```
namespace PIM_III_ADS_VISITANTE.Controller
```

```
{  
    public class PessoaController  
    {  
        private string nome;  
        private string codigo;  
  
        public PessoaController()  
        {  
  
        }  
  
        public string Nome  
        {  
            get  
            {  
                return nome;  
            }  
            set  
            {  
                nome = value;  
            }  
        }  
  
        public string Codigo  
        {  
            get  
            {  
                return codigo;  
            }  
            set  
            {  
                codigo = value;  
            }  
        }  
    }  
}
```

```
using PIM_III_ADS_VISITANTE.Model;  
using PIM_III_ADS_VISITANTE.View;  
using System;  
using System.Windows.Forms;
```

```
namespace PIM_III_ADS_VISITANTE.Controller
```

```
{  
    public class PerguntasController  
    {  
        private Obrigado obrigado;  
        private MediaAvaliacaoModel media;  
        private Panel pnlAvaliacao;  
        private int indicePergunta = 0;  
        private string[] perguntas = {  
            "Como você avaliaria a clareza das instruções fornecidas para interagir com as exposições?",  
            "Em sua opinião, como as tecnologias utilizadas nas exposições contribuíram para a sua compreensão dos temas abordados?",  
            "Como você classificaria a interatividade das exposições?",  
            "Considerando sua visita ao museu, como você classificaria sua satisfação geral?",  
            "Como você avalia a facilidade de uso do sistema dos totens para interagir com as exposições?"  
        };  
    }  
};
```

```

public PerguntasController(Panel pnlAvaliacao)
{
    this.pnlAvaliacao = pnlAvaliacao; // Armazene a referência ao Panel

    obrigado = new Obrigado();
    media = new MediaAvaliacaoModel();

    // Adicione um manipulador de eventos para sinalizar a conclusão das avaliações
    media.AvaliacoesConcluidas += Media_AvaliacoesConcluidas;
}

public string PerguntaAtual()
{
    if (indicePergunta < perguntas.Length)
    {
        return perguntas[indicePergunta];
    }
    else
    {
        // Exibindo a tela de agradecimento e finalizando a avaliação
        pnlAvaliacao.Visible = false;
        obrigado.Show();
        media.CalcularMediaAvaliacoes(); // Calcular a média das avaliações
        return "";
    }
}

private void Media_AvaliacoesConcluidas(object sender, EventArgs e)
{
    // Encontrar a instância correta de Obrigado
    Obrigado obrigadoForm = (Obrigado)Application.OpenForms.OfType<Obrigado>().FirstOrDefault();

    if (obrigadoForm != null)
    {
        obrigadoForm.SetMediaAvaliacao(media);
    }
}

public string[] Perguntas
{
    get { return perguntas; }
    set { perguntas = value; }
}

public int IndicePergunta
{
    get { return indicePergunta; }
    set { indicePergunta = value; }
}
}

namespace PIM_III_ADS_VISITANTE.Controller
{
    public class AvaliacaoController
    {
        private bool ruim;
        private bool regular;
    }
}

```

```

private bool bom;
private bool otimo;
private bool excelente;

public AvaliacaoController()
{

}

public AvaliacaoController(bool ruim, bool regular, bool bom, bool otimo, bool excelente)
{
    this.ruim = ruim;
    this.regular = regular;
    this.bom = bom;
    this.otimo = otimo;
    this.excelente = excelente;
}

public string AvaliacaoSelecionada()
{
    if (Ruim)
    {
        return "Ruim";
    }
    else if (Regular)
    {
        return "Regular";
    }
    else if (Bom)
    {
        return "Bom";
    }
    else if (Otimo)
    {
        return "Otimo";
    }
    else if (Excelente)
    {
        return "Excelente";
    }
    else
    {
        return "Nenhuma avaliação selecionada";
    }
}

public bool Excelente
{
    get
    {
        return excelente;
    }
    set
    {
        excelente = value;
    }
}

public bool Otimo
{
    get
    {
        return otimo;
    }
    set
    {
        otimo = value;
    }
}

```



```

    }
}

public bool Bom
{
    get
    {
        return bom;
    }
    set
    {
        bom = value;
    }
}

public bool Regular
{
    get
    {
        return regular;
    }
    set
    {
        regular = value;
    }
}

public bool Ruim
{
    get
    {
        return ruim;
    }
    set
    {
        ruim = value;
    }
}
}
}
}

```

```

using PIM_III_ADS_VISITANTE.Controller;
using PIM_III_ADS_VISITANTE.Service;

namespace PIM_III_ADS_VISITANTE.Model
{
    public class PessoaModel
    {
        PessoaServico pessoaServico = new PessoaServico(new Dbconexao());

        private string mensagem = "";

        public PessoaModel()
        {
        }

        public void Login(PessoaController pessoa)
        {
            var pessoaEncontrada = pessoaServico.BuscarPorCodigo(pessoa);

            if (pessoaEncontrada != null)

```

```

        {
            mensagem = "";
        }
        else
        {
            mensagem = "O token fornecido é inválido. Por favor, verifique e tente novamente.";
        }
    }

    public string Mensagem
    {
        get { return mensagem; }
    }
}

}

using PIM_III_ADS_VISITANTE.Service;
using System;
using System.Collections.Generic;
using System.Drawing;

namespace PIM_III_ADS_VISITANTE.Model
{
    internal class MediaAvaliacaoModel
    {
        private AvaliacaoService avaliacaoService;
        private int ruim;
        private int regular;
        private int bom;
        private int otimo;
        private int excelente;
        private string mensagem;
        private string mediaMensagem;
        private double media;

        public event EventHandler AvaliacoesConcluidas;

        public MediaAvaliacaoModel()
        {
            avaliacaoService = new AvaliacaoService();
        }

        private void OnAvaliacoesConcluidas()
        {
            AvaliacoesConcluidas?.Invoke(this, EventArgs.Empty);
        }

        public void CalcularMediaAvaliacoes()
        {
            if (avaliacaoService == null)
            {
                throw new InvalidOperationException("AvaliacaoService não foi inicializado.");
            }

            List<string> resultados = avaliacaoService.QuantidadeVotos();
            CalcularQuantidades(resultados);

            int totalAvaliacoes = Ruim + Regular + Bom + Otimo + Excelente;

            media = (Ruim * 1 + Regular * 2 + Bom * 3 + Otimo * 4 + Excelente * 5) / (double)totalAvaliacoes;

```

```

mediaMensagem = $"{media:F1}";
mensagem = $"{totalAvaliacoes/5} avaliações";

OnAvaliacoesConcluidas();
}

public Image ObterImagemMedia()
{
    Image imagemExibir = null;
    if (this.media >= 4.5)
    {
        imagemExibir = Properties.Resources.cincoestrelas;
    }
    else if (media >= 3.5)
    {
        imagemExibir = Properties.Resources.quatroestrelas;
    }
    else if (media >= 2.5)
    {
        imagemExibir = Properties.Resources.tresestrelas;
    }
    else if (media >= 1.5)
    {
        imagemExibir = Properties.Resources.duasestrelas;
    }
    else
    {
        imagemExibir = Properties.Resources.umaestrela;
    }

    return imagemExibir;
}

private void CalcularQuantidades(List<string> resultados)
{
    foreach (string resultado in resultados)
    {
        string[] partes = resultado.Split('\t');
        if (partes.Length == 2)
        {
            string voto = partes[0].ToLower();
            int quantidade = int.Parse(partes[1]);
            switch (voto)
            {
                case "ruim":
                    Ruim = quantidade;
                    break;
                case "regular":
                    Regular = quantidade;
                    break;
                case "bom":
                    Bom = quantidade;
                    break;
                case "otimo":
                    Otimo = quantidade;
                    break;
                case "excelente":
                    Excelente = quantidade;
                    break;
            }
        }
    }
}

public int Excelente
{
    get { return excelente; }
}

```

```

        set { excelente = value; }
    }

    public int Otimo
    {
        get { return otimo; }
        set { otimo = value; }
    }

    public int Bom
    {
        get { return bom; }
        set { bom = value; }
    }

    public int Regular
    {
        get { return regular; }
        set { regular = value; }
    }

    public int Ruim
    {
        get { return ruim; }
        set { ruim = value; }
    }

    public string Mensagem
    {
        get { return mensagem; }
        set { mensagem = value; }
    }

    public double Media { get => media; set => media = value; }
    public string MediaMensagem { get => mediaMensagem; set => mediaMensagem = value; }
}
}

```

```

using PIM_III_ADS_VISITANTE.Controller;
using PIM_III_ADS_VISITANTE.Service;

```

```

namespace PIM_III_ADS_VISITANTE.Model

```

```

{
    public class AvaliacaoModel
    {
        private AvaliacaoService avaliacaoService = new AvaliacaoService();
        private PerguntasController perguntas;
        private AvaliacaoController avaliacao;
        private PessoaController pessoa;

        public AvaliacaoModel(PessoaController pessoa, AvaliacaoController avaliacao, PerguntasController perguntas)
        {
            this.avaliacao = avaliacao;
            this.perguntas = perguntas;
            this.pessoa = pessoa;
        }

        public void SalvarVoto()
        {
            string avaliacaoAtual = avaliacao.AvaliacaoSelecionada();

```

```

        string codigoUsuario = pessoa.Codigo;
        string perguntaAtual = perguntas.PerguntaAtual();

        if (perguntaAtual == "Obrigado por participar! Volte sempre.")
        {
            return;
        }

        avaliacaoService.RegistrarVoto(perguntaAtual, avaliacaoAtual, codigoUsuario);

        perguntas.IndicePergunta++;
    }
}
}

```

```

using Dapper;
using PIM_III_ADS_VISITANTE.Controller;

```

```

namespace PIM_III_ADS_VISITANTE.Service
{
    public class PessoaServico
    {
        private readonly Dbconexao dbconexao;

        public PessoaServico(Dbconexao conexao)
        {
            this.dbconexao = conexao;
        }

        public PessoaController BuscarPorCodigo(PessoaController pessoa)
        {
            using (var conexao = new Dbconexao())
            {
                var connection = conexao.GetConnection();

                var resultado = connection.QueryFirstOrDefault(
                    "SELECT * FROM tbl_visitante WHERE Codigo = @codigo ",
                    new { Codigo = pessoa.Codigo });

                if (resultado == null)
                {
                    return null;
                }
                pessoa.Codigo = resultado.codigo;
                pessoa.Nome = resultado.nome;
                return pessoa;
            }
        }
    }
}

```

```

using Npgsql;
using System.Data;

```

```

namespace PIM_III_ADS_VISITANTE.Service

```

```

{
    public class Dbconexao : IDisposable
    {
        private NpgsqlConnection connection;

        public Dbconexao()
        {
            connection = new NpgsqlConnection("Server=localhost;Port=5432;Database=DB_Museu;User
            Id=postgres;Password=4852;");
        }

        public NpgsqlConnection GetConnection()
        {
            if (connection.State != ConnectionState.Open)
            {
                try
                {
                    connection.Open();
                }
                catch (Exception ex)
                {
                    // Lidar com a exceção ou relançá-la, conforme necessário
                    throw new Exception("Erro ao abrir a conexão com o banco de dados.", ex);
                }
            }

            return connection;
        }

        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (connection != null)
                {
                    if (connection.State != ConnectionState.Closed)
                    {
                        connection.Close();
                    }
                    connection.Dispose();
                    connection = null;
                }
            }
        }
    }
}
using Npgsql;

namespace PIM_III_ADS_VISITANTE.Service
{
    public class AvaliacaoService
    {
        private Dbconexao dbconexao = new Dbconexao();
        private NpgsqlConnection conexao;

        public AvaliacaoService()
        {
            conexao = dbconexao.GetConnection() as NpgsqlConnection;
        }
    }
}

```

```

public void RegistrarVoto(string pergunta, string avaliacao, string codigoUsuario)
{
    using (NpgsqlCommand command = new NpgsqlCommand(@"INSERT INTO public.tbl_votos (codigo,
pergunta, voto, data)
                                VALUES (@CodigoPessoa, @Pergunta, @Voto, @Data)", conexao))
    {
        command.Parameters.AddWithValue("@CodigoPessoa", codigoUsuario);
        command.Parameters.AddWithValue("@Pergunta", pergunta);
        command.Parameters.AddWithValue("@Voto", avaliacao);
        command.Parameters.AddWithValue("@Data", DateTime.Now);

        int linhasAfetadas = command.ExecuteNonQuery();

        if (linhasAfetadas < 0)
        {
            MessageBox.Show("Falha ao registrar o voto.");
        }
    }
}

internal List<string> QuantidadeVotos()
{
    List<string> resultados = new List<string>();

    using (NpgsqlCommand command = new NpgsqlCommand
        (@"SELECT voto, COUNT(voto) AS quantidade FROM tbl_votos GROUP BY voto", conexao))
    {
        var reader = command.ExecuteReader();
        while (reader.Read())
        {
            string voto = reader.GetString(0);
            int quantidade = reader.GetInt32(1);
            resultados.Add($"{voto}\t{quantidade}");
        }
    }

    return resultados;
}
}
}

```

```

using PIM_III_ADS_VISITANTE.Controller;
using PIM_III_ADS_VISITANTE.Model;
using WinFormsTimer = System.Windows.Forms.Timer;

namespace PIM_III_ADS_VISITANTE.View
{
    public partial class Avaliacao : Form
    {
        public AvaliacaoController avaliacaoControle;
        public PerguntasController perguntasControle;
        public AvaliacaoModel avaliacaoModel;
        public PessoaController pessoa;
        private WinFormsTimer timer;

        public Avaliacao(PessoaController pessoa)
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
        }
    }
}

```

```

        this.pessoa = pessoa;
        avaliacaoControle = new AvaliacaoController();
        perguntasControle = new PerguntasController(pnlAvaliacao);
        avaliacaoModel = new AvaliacaoModel(pessoa, avaliacaoControle, perguntasControle);

        timer = new WinFormsTimer();
        timer.Interval = 60000;
        timer.Tick += Timer_Tick;

        timer.Start();
        AtualizarPergunta();
    }

    public void AtualizarPergunta()
    {
        lblPergunta.Text = perguntasControle.PerguntaAtual();
        lblPergunta.MaximumSize = new Size(1200, 0);
        lblPergunta.TextAlign = ContentAlignment.MiddleCenter;
        int topPosition = (int)(15 * CreateGraphics().DpiY / 4);
        int leftPosition = (this.ClientSize.Width - lblPergunta.Width) / 2;
        lblPergunta.Location = new Point(leftPosition, topPosition);
    }

    public void btnRuim_Click(object sender, EventArgs e)
    {
        SetAvaliacao(true, false, false, false, false);
    }

    public void btnRegular_Click(object sender, EventArgs e)
    {
        SetAvaliacao(false, true, false, false, false);
    }

    public void btnBom_Click(object sender, EventArgs e)
    {
        SetAvaliacao(false, false, true, false, false);
    }

    public void btnOtimo_Click(object sender, EventArgs e)
    {
        SetAvaliacao(false, false, false, true, false);
    }

    public void btnExcelente_Click(object sender, EventArgs e)
    {
        SetAvaliacao(false, false, false, false, true);
    }

    private void SetAvaliacao(bool ruim, bool regular, bool bom, bool otimo, bool excelente)
    {
        avaliacaoControle.Ruim = ruim;
        avaliacaoControle.Regular = regular;
        avaliacaoControle.Bom = bom;
        avaliacaoControle.Otimo = otimo;
        avaliacaoControle.Excelente = excelente;
        avaliacaoModel.SalvarVoto();
        AtualizarPergunta();
    }

    private void Timer_Tick(object sender, EventArgs e)
    {
        this.Close();
        timer.Stop();
    }
}

```


APÊNDICE H – CÓDIGO FONTE DO PROGRAMA DO TECLADO

```
using TesteTecladoCerto;

namespace Teclado_virtual
{
    public partial class TecladoNumerico : Form
    {
        private TextBox targetTextBox;
        private Teclado teclado;
        public TecladoNumerico(TextBox targetTextBox, Teclado teclado)
        {
            InitializeComponent();
            this.targetTextBox = targetTextBox;
            this.teclado = teclado;
            this.TopMost = true;
            this.Load += Teclado_Load;
            this.Deactivate += Teclado_Deactivate;
        }

        private void Teclado_Deactivate(object sender, EventArgs e)
        {
            this.Hide();
        }

        public void FocusTargetTextBox()
        {
            targetTextBox.Focus();
        }

        private void Teclado_Load(object sender, EventArgs e)
        {
            int x = (Screen.PrimaryScreen.WorkingArea.Width - this.Width) / 2;
            int y = Screen.PrimaryScreen.WorkingArea.Height - this.Height;

            this.Location = new Point(x, y);
        }

        private void BtnTecla_Click(object sender, EventArgs e)
        {
            Button btn = sender as Button;
            if (btn != null)
            {
                string character = btn.Text;
                int cursorPosition = targetTextBox.SelectionStart;
                targetTextBox.Text = targetTextBox.Text.Insert(cursorPosition, character);
                targetTextBox.SelectionStart = cursorPosition + character.Length;
            }
        }

        private void BtnDel_Click(object sender, EventArgs e)
        {
            if (!string.IsNullOrEmpty(targetTextBox.Text))
            {
                int cursorPosition = targetTextBox.SelectionStart;
                if (cursorPosition > 0)
                {
                    targetTextBox.Text = targetTextBox.Text.Remove(cursorPosition - 1, 1);
                    targetTextBox.SelectionStart = cursorPosition - 1;
                }
            }
        }

        private void btnRun_Click_1(object sender, EventArgs e)
        {
            teclado.Show();
        }
    }
}
```

```

        this.Close();
    }
}

using Teclado_virtual;

namespace TesteTecladoCerto
{
    public partial class Teclado : Form
    {
        private bool shiftAtivado = false;
        private TextBox targetTextBox;

        public Teclado()
        {
            InitializeComponent();
            this.TopMost = true;
            this.Load += Teclado_Load;
            this.Deactivate += Teclado_Deactivate;
        }

        public void FocusTargetTextBox()
        {
            targetTextBox.Focus();
        }

        private void Teclado_Load(object sender, EventArgs e)
        {
            int x = (Screen.PrimaryScreen.WorkingArea.Width - this.Width) / 2;
            int y = Screen.PrimaryScreen.WorkingArea.Height - this.Height;
            this.Location = new Point(x, y);
        }

        private void AdicionarLetra(string letra)
        {
            if (targetTextBox != null)
            {
                int cursorPositionFromStart = targetTextBox.SelectionStart;
                int cursorPositionFromEnd = targetTextBox.TextLength - cursorPositionFromStart;
                targetTextBox.Text = targetTextBox.Text.Insert(cursorPositionFromStart, letra);
                targetTextBox.SelectionStart = cursorPositionFromStart + letra.Length;
                targetTextBox.SelectionLength = 0;
            }
        }

        private void TextBox_Click(object sender, EventArgs e)
        {
            TextBox clickedTextBox = sender as TextBox;
            SetTargetTextBox(clickedTextBox);
        }

        public void SetTargetTextBox(TextBox targetTextBox)
        {
            this.targetTextBox = targetTextBox;
            this.targetTextBox.HideSelection = false;
            this.targetTextBox.Focus();
        }

        // Evento para lidar com o teclado perdendo o foco
        private void Teclado_Deactivate(object sender, EventArgs e)
        {
            // Oculta o teclado quando ele perde o foco
            this.Hide();
        }
    }
}

```

```
#region Teclas
private void btnLetraA_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "A" : "a");
    shiftAtivado = false;
}

private void btnLetraQ_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "Q" : "q");
    shiftAtivado = false;
}

private void btnLetraW_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "W" : "w");
    shiftAtivado = false;
}

private void btnLetraE_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "E" : "e");
    shiftAtivado = false;
}

private void btnLetraR_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "R" : "r");
    shiftAtivado = false;
}

private void btnLetraT_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "T" : "t");
    shiftAtivado = false;
}

private void btnLetraY_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "Y" : "y");
    shiftAtivado = false;
}

private void btnLetraU_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "U" : "u");
    shiftAtivado = false;
}

private void btnLetraI_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "I" : "i");
    shiftAtivado = false;
}

private void btnLetraO_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "O" : "o");
    shiftAtivado = false;
}

private void btnLetraP_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "P" : "p");
    shiftAtivado = false;
}
```

```
private void btnLetraS_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "S" : "s");
    shiftAtivado = false;
}

private void btnLetraD_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "D" : "d");
    shiftAtivado = false;
}

private void btnLetraF_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "F" : "f");
    shiftAtivado = false;
}

private void btnLetraG_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "G" : "g");
    shiftAtivado = false;
}

private void btnLetraH_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "H" : "h");
    shiftAtivado = false;
}

private void btnLetraJ_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "J" : "j");
    shiftAtivado = false;
}

private void K_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "K" : "k");
    shiftAtivado = false;
}

private void btnLetraL_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "L" : "l");
    shiftAtivado = false;
}

private void btnLetraZ_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "Z" : "z");
    shiftAtivado = false;
}

private void btnLetraX_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "X" : "x");
    shiftAtivado = false;
}

private void btnLetraC_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "C" : "c");
    shiftAtivado = false;
}
```

```

private void btnLetraV_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "V" : "v");
    shiftAtivado = false;
}

private void btnLetraB_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "B" : "b");
    shiftAtivado = false;
}

private void btnLetraN_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "N" : "n");
    shiftAtivado = false;
}

private void btnLetraM_Click(object sender, EventArgs e)
{
    AdicionarLetra(shiftAtivado ? "M" : "m");
    shiftAtivado = false;
}
#endregion

private void btnBack_Click(object sender, EventArgs e)
{
    if (targetTextBox != null && targetTextBox.Text.Length > 0)
    {
        int cursorPosition = targetTextBox.SelectionStart;
        if (cursorPosition > 0)
        {
            // Move o cursor uma posição para trás
            targetTextBox.SelectionStart = cursorPosition - 1;
            // Remove o caractere na posição anterior ao cursor
            targetTextBox.Text = targetTextBox.Text.Remove(cursorPosition - 1, 1);
            // Define a posição do cursor de volta à posição correta
            targetTextBox.SelectionStart = cursorPosition - 1;
        }
    }
}

private void btn123_Click(object sender, EventArgs e)
{
    TecladoNumerico tecladoNumerico = new TecladoNumerico(targetTextBox, this);
    tecladoNumerico.Show();
}

private void btnShift_Click(object sender, EventArgs e)
{
    shiftAtivado = !shiftAtivado;
}

private void btnUnderline_Click(object sender, EventArgs e)
{
    if (targetTextBox != null)
        targetTextBox.Text += "_";
}

private void btnSpace_Click(object sender, EventArgs e)
{
    if (targetTextBox != null)
        targetTextBox.Text += " ";
}

private void btnArroba_Click(object sender, EventArgs e)

```

```
{
    if (targetTextBox != null)
        targetTextBox.Text += "@";
}

private void btnPontoCom_Click(object sender, EventArgs e)
{
    if (targetTextBox != null)
        targetTextBox.Text += ".com";
}

private void btnRun_Click(object sender, EventArgs e)
{
    this.Hide();
}
}
```