



instituto politécnico de gestão e tecnologia

ENGENHARIA INFORMÁTICA

PROGRAMAÇÃO AVANÇADA

TICKET2HELP

ERIC SENA

JOSÉ COSTA

VALÉRIO PINHEIRO

PROGRAMAÇÃO AVANÇADA

VILA NOVA DE GAIA
JUNHO 2024



Índice

Introdução	1
Análise do Sistema	3
Funções mais importantes	5
Cenário de Utilização	8
Teste Unitário	12
Git Hub	14
Conclusão	17

Introdução

Este projeto visa desenvolver um sistema de gestão de tickets de helpdesk para atender solicitações de serviços de hardware (HW) e software (SW) utilizando a linguagem de programação C#. A aplicação será estruturada em uma arquitetura de três camadas (UI, BLL, DAL) e implementada seguindo o padrão de design MVC (Model-View-Controller). A interface do usuário será desenvolvida com WPF (Windows Presentation Foundation), proporcionando uma interação rica e intuitiva para os usuários. O projeto também adotará padrões de design de software para otimizar a criação, estrutura e comportamento dos objetos, assegurando a qualidade e manutenção do código através de metodologias de documentação e testes unitários. Ferramentas de colaboração e gestão de versões de código, como o GitHub, serão utilizadas para facilitar o desenvolvimento colaborativo e o controle de versão. Ao final, um relatório detalhado será redigido para refletir sobre as decisões arquitetônicas, estratégias de design, interação com o usuário, validações e requisitos adicionais.

Requisitos Funcionais

1. Autenticação de Usuários:

- Interface de login que diferencia entre administradores e usuários comuns.
- Redirecionamento para a interface apropriada após a autenticação.

2. Administração de Tickets:

- Visualização de todos os tickets de suporte.
- Filtragem de tickets por tipo de serviço (HW ou SW).
- Geração de relatórios sobre os tickets.

3. Criação e Gerenciamento de Tickets:

- Interface para usuários criarem novos tickets de suporte.
- Acompanhamento do status dos tickets criados.
- Visualização de detalhes específicos de cada ticket.

Requisitos Não Funcionais

1. Arquitetura do Sistema:

- Implementação de uma arquitetura em três camadas: UI (User Interface), BLL (Business Logic Layer) e DAL (Data Access Layer).
- Adoção do padrão de design MVC para uma separação clara das responsabilidades.

2. Tecnologia e Ferramentas:

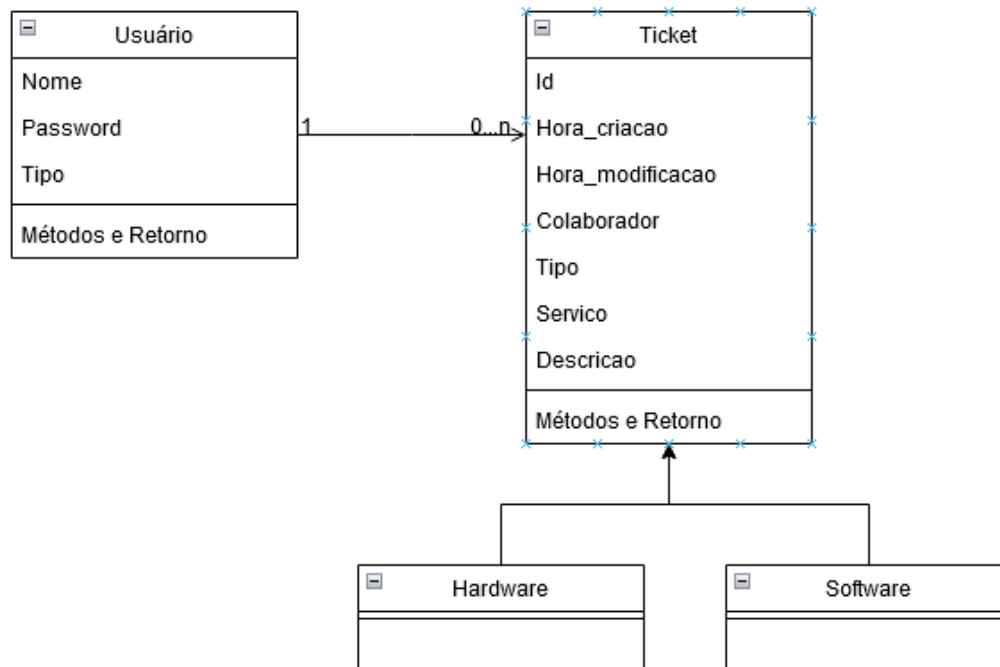
- Utilização de WPF para a interface gráfica.
- Uso do Entity Framework para comunicação com a base de dados, facilitando operações CRUD (Create, Read, Update, Delete).
- Banco de dados SQL Server para armazenamento de dados.
- Ferramentas de colaboração e gestão de versão como GitHub para controle de versão e trabalho colaborativo.

3. Qualidade e Manutenção do Código:

- Aplicação de padrões de design de software para a criação, estrutura e comportamento dos objetos.
- Adoção de metodologias de documentação de software para garantir a clareza e manutenção do código.
- Implementação de testes unitários para assegurar a funcionalidade e integridade do sistema.

Análise do Sistema

Diagrama de Classes



Modelo de Dados

```
create database Ticket2Help
```

```
use Ticket2Help
```

```
go
```

```
create table usuario(
```

```
nome nvarchar(50) primary key,
```

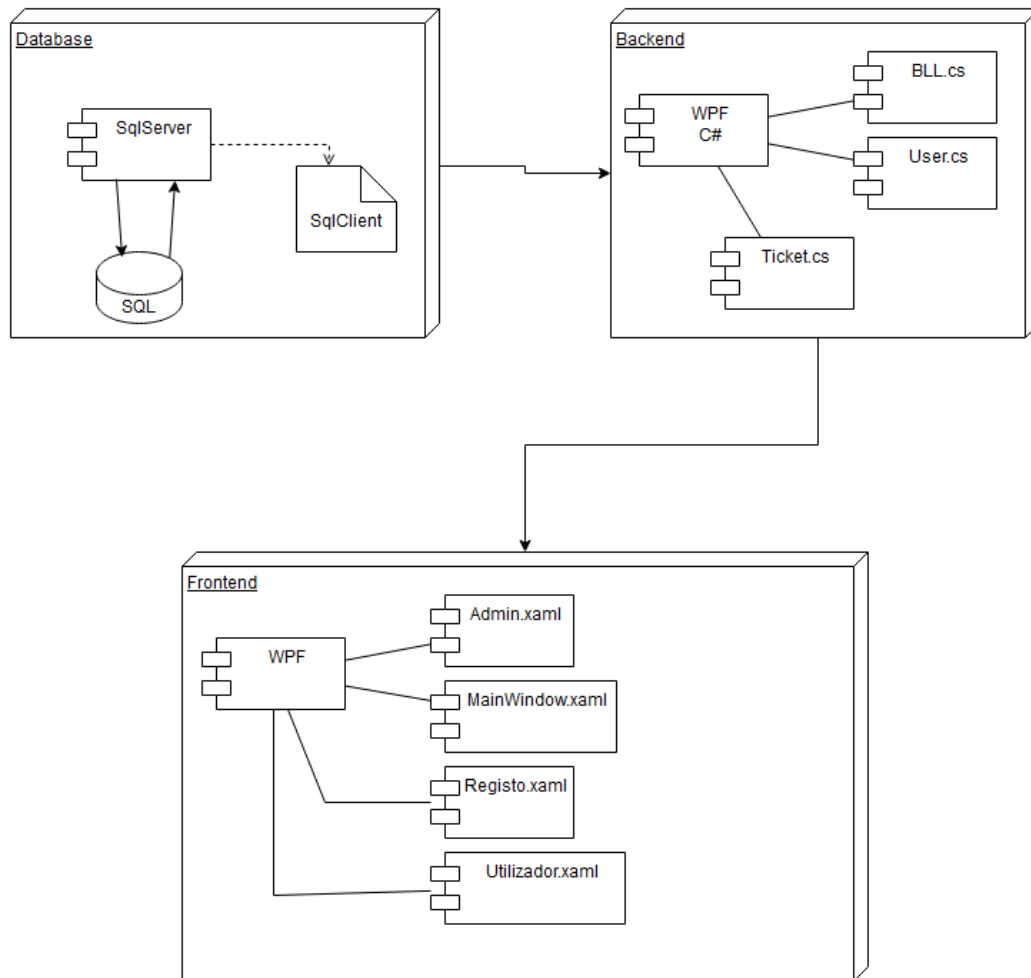
```
password nvarchar(50),
```

```
tipo
```

```
)
```

```
create table Ticket(  
  
id int identity(1,1) not null primary key,  
  
hora_criacao dateTime,  
  
hora_modificacao dateTime,  
  
colaborador nvarchar(50) foreign key references usuario(nome),  
  
tipo nvarchar(50),  
  
servico nvarchar(50),  
  
descricao nvarchar(50)  
  
)
```

Diagrama de Componentes



Funções mais importantes

Camada de Acesso a Dados (DAL)

Responsabilidades da DAL

1. **Configuração do Banco de Dados:**
 - Configurar a conexão com o banco de dados.
 - Definir o mapeamento entre os objetos do modelo e as tabelas do banco de dados (ORM - Object-Relational Mapping).
2. **Operações CRUD:**
 - Fornecer métodos para criar, ler, atualizar e deletar dados.
 - Garantir que as operações de dados sejam realizadas de forma eficiente e segura.
3. **Gerenciamento de Conexões:**
 - Abrir e fechar conexões com o banco de dados conforme necessário.
 - Gerenciar transações para garantir a integridade dos dados.

Camada de Lógica de Negócios (BLL)

Importância da BLL

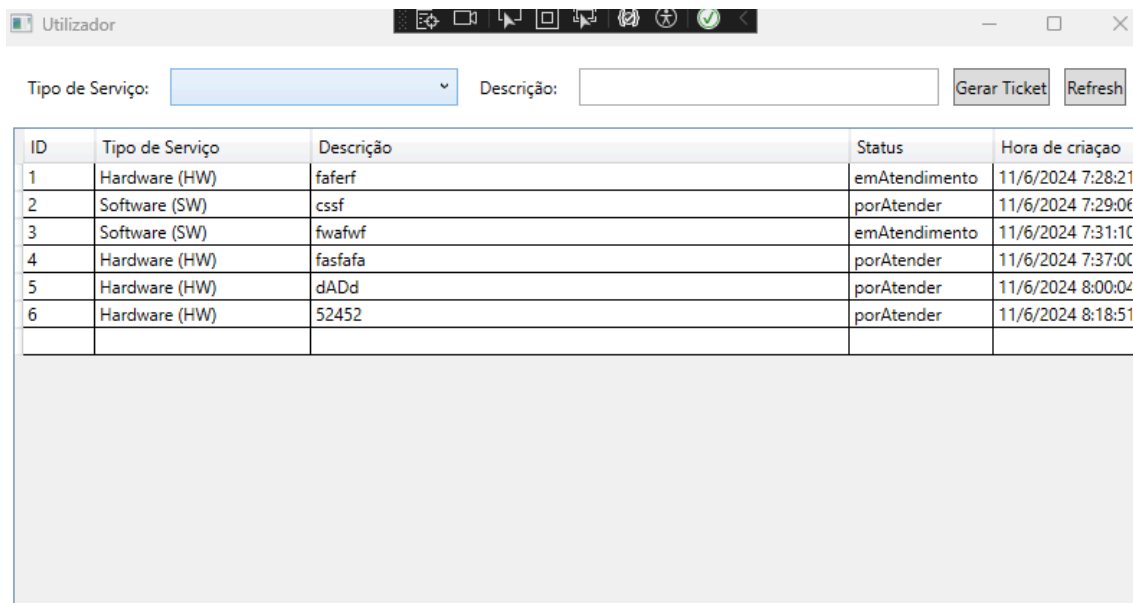
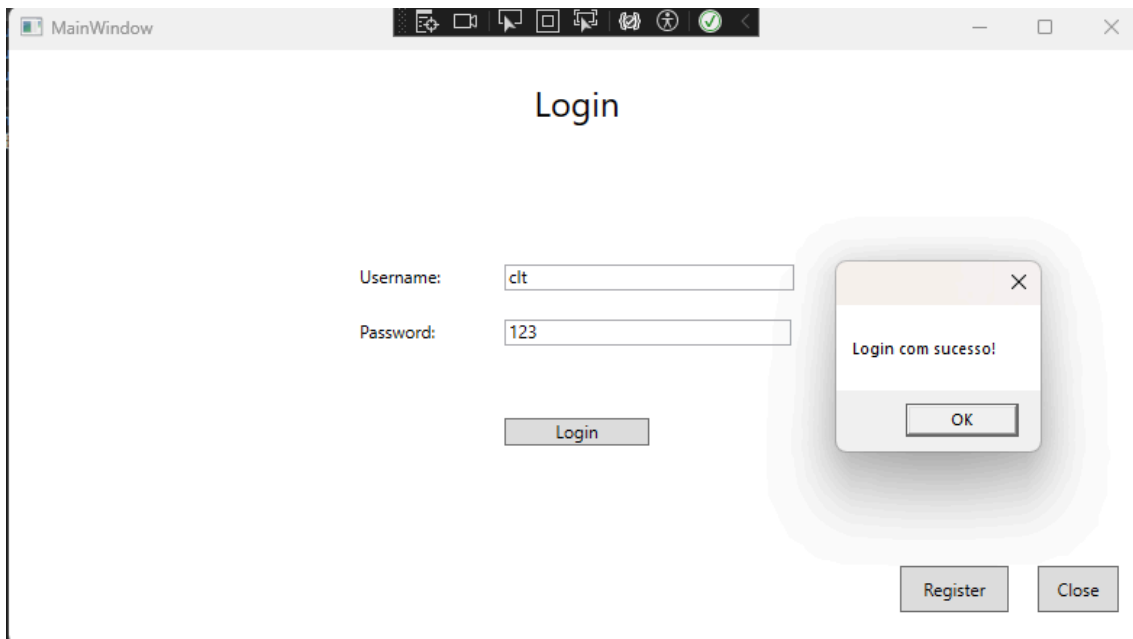
1. **Centralização da Lógica de Negócios:** A BLL centraliza as regras de negócio, facilitando a manutenção e a consistência da aplicação.
2. **Validação e Segurança:** A BLL valida os dados e aplica as regras de segurança antes de enviar ou recuperar dados da DAL.
3. **Reuso e Modularidade:** A lógica de negócios pode ser reutilizada em diferentes partes da aplicação, promovendo modularidade e facilitando testes.
4. **Orquestração de Processos:** A BLL orquestra operações complexas que envolvem múltiplos repositórios ou serviços, garantindo que os processos sejam executados corretamente.

Camada de Interface com o Usuário (UI)

Importância da UI

1. **Experiência do Usuário:** A UI é a face da aplicação para o usuário. Uma interface bem projetada e intuitiva melhora significativamente a experiência do usuário.
2. **Eficiência e Produtividade:** Uma UI eficiente permite que os usuários realizem suas tarefas de forma rápida e precisa, aumentando a produtividade.
3. **Feedback e Interatividade:** Fornece feedback imediato sobre as ações do usuário, melhorando a interação e satisfação do usuário.
4. **Acessibilidade:** Garante que a aplicação seja acessível a todos os usuários, incluindo aqueles com necessidades especiais.

Cenário de Utilização



Utilizador

Tipo de Serviço: Hardware (HW) Descrição: abcd Gerar Ticket Refresh

ID	Tipo de Serviço	Descrição	Status	Hora de criação
1	Hardware (HW)	faferf	emAtendimento	11/6/2024 7:28:21
2	Software (SW)	cssf	porAtender	11/6/2024 7:29:06
3	Software (SW)	fwafwf	emAtendimento	11/6/2024 7:31:10
4	Hardware (HW)	fasfafa	porAtender	11/6/2024 7:37:00
5	Hardware (HW)	dADd	porAtender	11/6/2024 8:00:04
6	Hardware (HW)	52452	porAtender	11/6/2024 8:18:51

Success: - 11/06/2024 23:06:49 - - Hardware (HW) - abcd OK

MainWindow

Login

Username: adm Password: 123 Login

Login com sucesso! OK

Register Close

Admin

Estado do Serviço emAtendimento Atualizar Refresh

ID	Tipo de Serviço	Descrição	Status	Última modificação
1	Hardware (HW)	faferf	emAtendimento	11/6/2024 7:28:21 P
2	Software (SW)	cssf	porAtender	11/6/2024 7:29:06 P
3	Software (SW)	fwafwf	emAtendimento	11/6/2024 9:22:23 P
4	Hardware (HW)	fasfafa	porAtender	11/6/2024 7:37:00 P
5	Hardware (HW)	dADd	porAtender	11/6/2024 8:00:04 P
6	Hardware (HW)	52452	porAtender	11/6/2024 8:18:51 P
7	Hardware (HW)	abcd	porAtender	11/6/2024 11:06:49 P

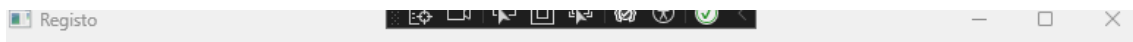
Admin

Estado do Serviço emAtendimento Atualizar Refresh

ID	Tipo de Serviço	Descrição	Status	Última modificação
1	Hardware (HW)	faferf	emAtendimento	11/6/2024 7:28:21 P
2	Software (SW)	cssf	porAtender	11/6/2024 7:29:06 P
3	Software (SW)	fwafwf	emAtendimento	11/6/2024 9:22:23 P
4	Hardware (HW)	fasfafa	porAtender	11/6/2024 7:37:00 P
5	Hardware (HW)	dADd	porAtender	11/6/2024 8:00:04 P
6	Hardware (HW)	52452	porAtender	11/6/2024 8:18:51 P
7	Hardware (HW)	abcd	porAtender	11/6/2024 11:06:49 P

Sucesso

OK



Registo

Username

Password

Tipo

Registrar

Success: adm2 - 123 - Tecnico

OK

Close

Teste Unitário

Este teste unitário em C# foi projetado para garantir o comportamento adequado de uma janela principal (MainWindow) num projeto chamado ProjetoFinal. Vamos analisá-lo em detalhes:

Objetivo do Teste

O teste visa verificar se, ao clicar num botão com um utilizador válido, a janela principal é fechada e uma nova janela (ClientWindow) é aberta corretamente.

Estrutura do Teste

Arrange (Organizar)

Nesta secção, o cenário de teste é preparado:

- Uma instância da **MainWindow** é criada.
- Uma instância falsa (Mock) de **IMessageBoxWrapper** é criada usando Moq para simular a funcionalidade da caixa de mensagem.
- Valores esperados para o nome de utilizador e palavra-passe são definidos.
- Um evento é adicionado para verificar se a janela principal foi fechada.

Act (Atuar)

A ação a ser testada é executada:

- Os campos de texto da janela principal são preenchidos com o nome de utilizador e palavra-passe esperados.
- O evento de clique do botão é acionado chamando **Button_Click** manualmente.

Assert (Afirmação)

Verifica-se se o resultado esperado ocorreu:

- Confirma-se que a janela principal foi fechada (`mainWindowClosed` é verdadeiro).
- Verifica-se se a nova janela aberta é do tipo `Utilizador`.
- Compara-se se o nome de utilizador atual é igual ao esperado (`BLL.colaborador`).
- Garante-se que o método `Show` da caixa de mensagem nunca foi chamado.

Interface IMessageBoxWrapper

Define um contrato para envolver a funcionalidade da caixa de mensagem. No teste, não é utilizada diretamente, mas em vez disso, é simulada para isolar o teste da funcionalidade real da caixa de mensagem.

Conclusão

Este teste unitário oferece uma garantia de que, quando um utilizador válido realiza login, a aplicação responde conforme o esperado, fechando a janela principal e abrindo uma nova janela de cliente. Além disso, demonstra a prática de simular interfaces para isolar o código do teste de dependências externas.

Git Hub


Scorch94 José
 434dfa2 · 9 minutes ago 8 Commits

AvancadoFinal	Jose	last week
AvancadoFinal.sln	José	last week
README.md	José	9 minutes ago


README


Ticket2Help Helpdesk Management Platform


Bem-vindo ao repositório do projeto Ticket2Help! Esta plataforma foi desenvolvida para a gestão eficiente de tickets de helpdesk, focando em solicitações de serviços de hardware (HW) e software (SW). A aplicação é desenhada para funcionar exclusivamente na rede interna da empresa, garantindo segurança e controle sobre os dados e acessos.

Funcionalidades Principais

- Gestão de Tickets**
 - Criação, visualização e atualização de tickets de suporte.
 - Categorização dos tickets em hardware (HW) e software (SW).
 - Acompanhamento do status dos tickets (aberto, em progresso, resolvido, fechado).
- Gestão de Acessos (Login)**
 - Sistema de autenticação de utilizadores com login.
 - Permissões personalizadas para cada utilizador conforme seu papel na organização.
- Permissões de Utilizador**
 - Diferentes níveis de acesso e permissões (usuário padrão, técnico, administrador).
 - Controle sobre quem pode visualizar e editar determinados tickets.


Name	Last commit message	Last commit date
..		
obj	Valério	now
1000021935.jpg	José	2 weeks ago
Admin.xaml	Valério	now
Admin.xaml.cs	Jose	last week
App.xaml	José	2 weeks ago
App.xaml.cs	José	2 weeks ago
AssemblyInfo.cs	José	2 weeks ago
AvancadoFinal.csproj	José	2 weeks ago
AvancadoFinal.csproj.user	Valério	now
Login.xaml	Valério	now
Login.xaml.cs	José	2 weeks ago
Registar.xaml	Valério	now
Registar.xaml.cs	Valério	now
Utilizador.xaml	Valério	now
Utilizador.xaml.cs	Jose	last week

Name	Last commit message	Last commit date
...		
obj	Eric	now
1000021935.jpg	José	2 weeks ago
Admin.xaml	Valério	14 minutes ago
Admin.xaml.cs	Eric	3 minutes ago
App.xaml	José	2 weeks ago
App.xaml.cs	José	2 weeks ago
AssemblyInfo.cs	José	2 weeks ago
MainWindow.xaml	Eric	3 minutes ago
MainWindow.xaml.cs	Eric	3 minutes ago
ProjetoFinal.csproj	Eric	now
ProjetoFinal.csproj.user	Eric	now
Registrar.xaml	Valério	14 minutes ago
Registrar.xaml.cs	Eric	3 minutes ago
TesteUnitario.cs	Eric	3 minutes ago
Utilizador.xaml	Valério	14 minutes ago
Utilizador.xaml.cs	Eric	3 minutes ago

 **Avancadas** Public

Pin Unwatch 1

main 1 Branch 0 Tags Add file <> Code

 **Scorch94** Merge branch 'main' of <https://github.com/Scorch94/Avancadas> c4f37ce · now 14 Commits

.vs	Valério	5 minutes ago
AvancadoFinal	Eric	7 minutes ago
BLL	Eric	7 minutes ago
DAL	Eric	7 minutes ago
ProjetoFinal.sln	Valério	5 minutes ago
README.md	José	1 hour ago
Ticket.sql	Valério	now

Name	Last commit message	Last commit date
...		
html	José	1 minute ago
latex	José	1 minute ago
obj	Eric	15 minutes ago
BLL.cs	Eric	18 minutes ago
BLL.csproj	Eric	15 minutes ago
Ticket.cs	Eric	18 minutes ago
User.cs	Eric	18 minutes ago

EricSena241 José e2f0506 - 1 minute ago History		
Name	Last commit message	Last commit date
..		
html	José	1 minute ago
latex	José	1 minute ago
obj	Eric	15 minutes ago
1000021935.jpg	José	2 weeks ago
Admin.xaml	Valério	29 minutes ago
Admin.xaml.cs	Eric	18 minutes ago
App.xaml	José	2 weeks ago
App.xaml.cs	José	2 weeks ago
AssemblyInfo.cs	José	2 weeks ago
MainWindow.xaml	Eric	18 minutes ago
MainWindow.xaml.cs	Eric	18 minutes ago
ProjetoFinal.csproj	Eric	15 minutes ago
ProjetoFinal.csproj.user	Eric	15 minutes ago
Registrar.xaml	Valério	29 minutes ago
Registrar.xaml.cs	Eric	18 minutes ago

Name	Last commit message	Last commit date
..		
html	José	1 minute ago
latex	José	1 minute ago
obj	Eric	15 minutes ago
BLL.cs	Eric	18 minutes ago
BLL.csproj	Eric	15 minutes ago
Ticket.cs	Eric	18 minutes ago
User.cs	Eric	18 minutes ago

Conclusão

O desenvolvimento do sistema de gestão de tickets de helpdesk para serviços de hardware (HW) e software (SW) atende a uma necessidade crítica no ambiente corporativo moderno, proporcionando uma ferramenta eficaz para o gerenciamento de solicitações de suporte técnico. Este projeto, utilizando a linguagem de programação C#, adota uma arquitetura em três camadas (UI, BLL, DAL) e segue o padrão de design MVC (Model-View-Controller), garantindo uma separação clara das responsabilidades e facilitando a manutenção e escalabilidade do sistema.

A implementação da interface do usuário com WPF (Windows Presentation Foundation) proporciona uma experiência rica e intuitiva, melhorando a usabilidade e eficiência dos usuários. O uso do Entity Framework para a comunicação com a base de dados SQL Server simplifica as operações de CRUD e assegura a integridade e consistência dos dados.

Além disso, a aplicação de padrões de design de software garante a criação de um sistema robusto e bem-estruturado. Metodologias de documentação de software e a implementação de testes unitários asseguram a qualidade e a facilidade de manutenção do código, enquanto o uso de ferramentas de colaboração e gestão de versões, como o GitHub, facilita o desenvolvimento colaborativo e o controle de versão.

Este projeto não apenas cumpre os requisitos funcionais, oferecendo funcionalidades completas para administração de tickets e criação e gerenciamento de solicitações pelos usuários, mas também atende aos requisitos não funcionais, garantindo desempenho, segurança, usabilidade e sustentabilidade do sistema.

O relatório final documenta detalhadamente todas as decisões tomadas durante o desenvolvimento, incluindo as estratégias de padrões e arquitetura, interações com o usuário, validações e requisitos adicionais. Este esforço de documentação assegura que o sistema possa ser facilmente compreendido, mantido e evoluído no futuro, atendendo às necessidades crescentes dos usuários e às exigências tecnológicas em constante mudança.