**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Engenharia de *Software***

**Projeto Integrado**

**Relatório Técnico**

**EntregaSegura**

Danilo de Carvalho Silva

Jamil Tadeu Zazu

Belo Horizonte

Abril, 2023.

**SUMÁRIO**

[1. Cronograma de Trabalho 4](#_Toc147597730)

[2. Introdução 5](#_Toc147597731)

[3. Definição Conceitual da Solução 7](#_Toc147597732)

[3.1. Diagrama de Casos de Uso 7](#_Toc147597733)

[3.2. Requisitos Funcionais 8](#_Toc147597734)

[3.3. Requisitos Não-Funcionais 10](#_Toc147597735)

[4. Protótipo Navegável do Sistema 11](#_Toc147597736)

[5. Diagrama de Classes de Domínio 12](#_Toc147597737)

[6. Arquitetura da Solução 13](#_Toc147597738)

[6.1. Padrão Arquitetural 13](#_Toc147597739)

[6.2. C4 model - Diagrama de Contexto 14](#_Toc147597740)

[7. Frameworks de Trabalho 15](#_Toc147597741)

[8. Estrutura Base do Front End 16](#_Toc147597742)

[9. Modelo Relacional ou Projeto de Banco de Dados NoSQL 17](#_Toc147597743)

[10. Plano de Testes 18](#_Toc147597744)

[11. Apropriação de Horas no Projeto 19](#_Toc147597745)

[12. Código da Aplicação 20](#_Toc147597746)

[13. Avaliação Retrospectiva 21](#_Toc147597747)

[13.1 Objetivos Estimados 21](#_Toc147597748)

[13.2 Objetivos Alcançados 21](#_Toc147597749)

[13.3 Lições aprendidas 21](#_Toc147597750)

[14. Referências 22](#_Toc147597751)

# Cronograma de Trabalho

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 01/03/2023 | 04/03/2023 | 1. Contextualização e objetivos do trabalho | Escopo do projeto definido |
| 07/03/2023 | 13/03/2023 | 1. Elaboração da proposta de projeto | Proposta do projeto |
| 16/03/2023 | 23/03/2023 | 1. Levantamento de requisitos funcionais | Lista de requisitos funcionais |
| 27/03/2023 | 02/04/2023 | 1. Levantamento de requisitos não-funcionais | Lista de requisitos não-funcionais |
| 04/04/2023 | 10/04/2023 | 1. Documentação dos Requisitos Levantados | Documentação dos requisitos |
| 12/04/2023 | 18/04/2023 | 1. Elaboração do Diagrama de Casos de Uso | Diagrama de casos de uso |
| 20/04/2023 | 26/04/2023 | 1. Desenvolvimento do Diagrama de Classes de Domínio | Diagrama de classes de domínio |
| 28/04/2023 | 04/05/2023 | 1. Definição do Padrão Arquitetural | Padrão arquitetural definido |
| 07/05/2023 | 13/05/2023 | 1. Estrutura de Banco de Dados | Estrutura de banco de dados definida |
| 15/05/2023 | 15/07/2023 | 1. Desenvolvimento do sistema | Sistema desenvolvido |
| 18/07/2023 | 24/07/2023 | 1. Elaboração do Plano de Testes | Plano de testes |
| 27/07/2023 | 02/08/2023 | 1. Revisão da documentação do Projeto | Documentação revisada |
| 05/08/2023 | 10/08/2023 | 1. Criação do repositório no GitHub e deploy do protótipo | Repositório no GitHub e protótipo publicado |
| 20/08/2023 | 26/08/2023 | 1. Revisão da Apropriação de Horas | Registro de horas apropriadas revisto |
| 01/09/2023 | 17/09/2023 | 1. Revisão de Formatação do Relatório Técnico | Relatório Técnico de acordo com as normas |
| 20/09/2023 | 30/09/2023 | 1. Avaliação retrospectiva | Retrospectiva do projeto |
| 03/10/2023 | 03/10/2023 | 1. Entrega final | Projeto entregue |

# Introdução

A era digital tem sido marcada por um notável crescimento no comércio eletrônico no Brasil, impulsionando as vendas online a níveis impressionantes. Em 2022, segundo a Abranet, as vendas online alcançaram a marca de R$ 169,6 bilhões, registrando um crescimento de 5% em relação ao ano anterior. Esse panorama é reflexo da crescente demanda por compras realizadas via dispositivos móveis, que já representam 55% de todas as vendas online. Nesse contexto, a gestão de entregas residenciais em condomínios emerge como um desafio específico que precisa ser enfrentado prontamente.

O cerne deste projeto consiste na resolução de uma dificuldade latente: o gerenciamento seguro e eficaz das entregas em condomínios. Esse desafio envolve assegurar que as entregas alcancem os destinatários corretos, garantir o controle preciso da entrada e saída de pacotes e promover uma comunicação eficiente entre os funcionários do condomínio e os moradores. A "dor do cliente" reside na necessidade de aprimorar esse processo, tornando-o mais eficiente, confiável e seguro, considerando o aumento exponencial da demanda por compras online e o contínuo crescimento do comércio eletrônico.

A motivação deste projeto é oferecer uma solução para um problema que cresce de forma incessante no mercado. Com a implementação do sistema *EntregaSegura*, almeja-se obter uma economia significativa de tempo tanto para os funcionários quanto para os moradores, além de aprimorar a segurança no processo de entrega de encomendas. Essa solução trará benefícios relevantes como: a redução de atrasos nas entregas, a otimização de recursos e a maior satisfação dos residentes, tornando-se uma inovação relevante no contexto atual de crescimento do comércio eletrônico e da crescente demanda por entregas em condomínios.

O objetivo geral deste trabalho é apresentar a descrição detalhada do projeto do sistema *EntregaSegura*, desenvolvido com o propósito de otimizar o gerenciamento de entregas em condomínios.

Objetivos específicos são:

* Descrever minuciosamente os requisitos da aplicação *EntregaSegura*;
* Elucidar como o sistema atua na solução do problema identificado, apresentando suas funcionalidades e vantagens;
* Demonstrar a relevância e aplicabilidade do *EntregaSegura* no atual contexto de crescimento do comércio eletrônico e da demanda por entregas em condomínios.

Ao longo deste trabalho, cada um desses objetivos será explorado em detalhes, evidenciando o valor que o sistema *EntregaSegura* pode proporcionar aos condomínios residenciais no atual cenário da era digital. Com esse projeto, busca-se estabelecer uma base sólida para aprimorar a eficiência e a segurança do processo de entregas em condomínios, beneficiando tanto os prestadores de serviço quanto os moradores.

# Definição Conceitual da Solução

## 3.1. Diagrama de Casos de Uso



**Figura 1**: Casos de Uso

**Fonte:** Os Autores

**O arquivo original pode ser encontrado em:** [Casos de Uso](https://github.com/ddcsilva/EntregaSegura/blob/main/Docs/Diagramas/Diagrama%20de%20caso%20de%20uso%20-%20PlantUML.png?raw=true)

## 3.2. Requisitos Funcionais

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A)\*** | **Prioridade**  **(B/M/A)\*** |
| RF01 | O administrador deve poder listar os funcionários cadastrados no sistema. | B | A |
| RF02 | O administrador deve poder cadastrar novos funcionários no sistema. | B | A |
| RF03 | O administrador deve poder editar as informações dos funcionários cadastrados. | M | A |
| RF04 | O administrador deve poder excluir funcionários do sistema. | M | A |
| RF05 | O administrador deve poder listar os condomínios registrados. | B | B |
| RF06 | O administrador deve poder cadastrar novos condomínios. | M | A |
| RF07 | O administrador deve poder editar informações dos condomínios. | M | B |
| RF08 | O administrador deve poder excluir condomínios do sistema. | M | B |
| RF09 | O administrador deve poder listar as unidades registradas no sistema. | B | B |
| RF10 | O administrador deve poder cadastrar novas unidades. | M | B |
| RF11 | O administrador deve poder editar informações das unidades cadastradas. | M | B |
| RF12 | O administrador deve poder excluir unidades do sistema. | M | B |
| RF13 | O administrador deve poder consultar o endereço através do CEP ao cadastrar ou editar condomínios. | A | A |
| RF14 | O administrador deve poder listar os moradores cadastrados. | B | B |
| RF15 | O administrador deve poder cadastrar novos moradores. | M | B |
| RF16 | O administrador deve poder editar informações dos moradores cadastrados. | M | B |
| RF17 | O administrador deve poder excluir moradores do sistema. | M | B |
| RF18 | O administrador deve poder carregar o dashboard do sistema. | B | A |
| RF19 | O síndico deve poder listar os condomínios registrados. | B | B |
| RF20 | O síndico deve poder cadastrar novos condomínios. | M | B |
| RF21 | O síndico deve poder editar informações dos condomínios. | M | B |
| RF22 | O síndico deve poder excluir condomínios do sistema. | M | B |
| RF23 | O síndico deve poder listar as unidades registradas no sistema. | B | B |
| RF24 | O síndico deve poder cadastrar novas unidades. | M | B |
| RF25 | O síndico deve poder editar informações das unidades cadastradas. | M | B |

\* B = Baixa, M = Média, A = Alta.

## 3.3. Requisitos Não-Funcionais

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | O sistema deve apresentar um tempo de resposta abaixo de 200 ms para 95% das consultas. | A |
| RNF02 | A interface do sistema deve ser responsiva e adaptar-se a diferentes tamanhos de tela. | M |
| RNF03 | O sistema deve ser compatível com os navegadores mais populares, como Chrome, Firefox e Safari. | M |
| RNF04 | A segurança do sistema deve ser robusta, protegendo os dados dos usuários contra acesso não autorizado. | A |
| RNF05 | O sistema deve ser capaz de lidar com um grande volume de usuários simultâneos sem degradação de desempenho. | A |
| RNF06 | A disponibilidade do sistema deve ser de, no mínimo, 99,9%, permitindo um tempo de inatividade programado mínimo. | A |
| RNF07 | O sistema deve ser facilmente escalável, permitindo a adição de recursos conforme necessário para suportar o crescimento do número de usuários. | M |
| RNF08 | O sistema deve ser intuitivo e fácil de usar, exigindo um treinamento mínimo para os usuários aprenderem a utilizá-lo. | M |
| RNF09 | O sistema deve fornecer mensagens de erro claras e úteis para ajudar os usuários a corrigirem problemas que possam surgir durante a utilização. | M |
| RNF10 | O sistema deve ser atualizável, permitindo a adição de novos recursos ou a correção de bugs de forma fácil e rápida. | M |
| RNF11 | O sistema deve ter uma aparência visual atrativa e profissional, promovendo uma experiência de usuário positiva. | B |

\* B = Baixa, M = Média, A = Alta.

# Protótipo Navegável do Sistema

* Link do Protótipo Navegável e interativo: [Protótipo navegável e interativo](https://www.figma.com/proto/R642DaPUr3NuEGA5hpxK0a/EntregaSegura?page-id=0%3A1&type=design&node-id=69-53356&viewport=285%2C116%2C0.14&t=PDiFnGxO6ulmVcmY-1&scaling=min-zoom&starting-point-node-id=69%3A53356&mode=design)
* Link do Vídeo de apresentação: [Vídeo de apresentação](https://github.com/ddcsilva/EntregaSegura/raw/main/Docs/Vi%CC%81deo%20de%20apresentac%CC%A7a%CC%83o.mp4)
* Link do Design System no Figma: [Design System](https://www.figma.com/file/R642DaPUr3NuEGA5hpxK0a/EntregaSegura?type=design&mode=design)

# Diagrama de Classes de Domínio

Interface gráfica do usuário, Diagrama

Descrição gerada automaticamente

**Figura 2:** Diagrama de Classes de Domínio

**Fonte:** Os Autores

**O arquivo original pode ser encontrado em:** [Diagrama de Classes de Domínio](https://github.com/ddcsilva/EntregaSegura/blob/main/Docs/Diagramas/Diagrama%20de%20Classes%20-%20EntregaSegura%20v2.jpg?raw=true)

# Arquitetura da Solução

## 6.1. Padrão Arquitetural

Ao analisar os requisitos funcionais e não funcionais do sistema, optamos pelo padrão arquitetural **Arquitetura em Camadas**, com ênfase na Clean Architecture. Essa combinação promove uma separação clara e intencional das responsabilidades, tornando o código mais limpo, modular e testável.

O frontend foi desenvolvido em Angular 14, seguindo as melhores práticas de Single Page Application (SPA). Nessa camada, utilizamos o padrão MVVM (Model-View-ViewModel) para separar de forma efetiva a lógica da interface dos dados. A integração de bibliotecas como Ngx-mask, Ngx-spinner, Ngx-toastr, Bootstrap 5 e Angular Material proporciona uma interface de usuário rica e responsiva.

No que diz respeito ao backend, temos o projeto **EntregaSegura.Application** que atua como intermediário entre a camada de apresentação e a camada de domínio. Aqui, são gerenciados os DTOs, notificações, interfaces e lógica de negócio de alto nível. A integração com o AutoMapper permite um mapeamento eficiente entre entidades e DTOs.

A essência do sistema está no **EntregaSegura.Domain**. Nesse núcleo, encontramos as entidades, os modelos de domínio, as validações e as interfaces de repositório, garantindo autonomia sem depender de camadas externas. Já o **EntregaSegura.Infra.Data** cuida da persistência e do acesso aos dados, implementando as interfaces do domínio e utilizando o padrão Repository para abstrair o acesso ao banco de dados. A tecnologia subjacente é o Entity Framework Core, que facilita o mapeamento objeto-relacional e a gestão da conexão com o SQL Server.

**EntregaSegura.API** desempenha um papel vital como interface no padrão MVC (Model-View-Controller) para o backend. Ele recebe requisições do frontend e as direciona para as camadas apropriadas. Além disso, a gestão de autenticação e usuários é efetivamente tratada pelo ASP.NET Identity. O Fluent Validation é utilizado para garantir a aplicação rigorosa das regras de negócio, enquanto o **EntregaSegura.Infra.IoC** foi implementado para gerenciar a inversão de controle e a injeção de dependência, aproveitando a Injeção de Dependência Nativa do .NET.

Para armazenar os dados, a escolha do SQL Server como solução de banco de dados foi motivada por sua robustez, segurança e desempenho avançado. Em conclusão, a arquitetura multicamadas adotada, juntamente com os princípios da Clean Architecture, garante a escalabilidade, modularidade e facilidade de manutenção do sistema, preparando-o para futuras extensões e melhorias com um impacto mínimo nos componentes existentes.

## 6.2. C4 model - Diagrama de Contexto

Diagrama

Descrição gerada automaticamente

**Figura 3:** Arquitetura da solução (C4 model)

**Fonte:** Os Autores

**O arquivo original pode ser encontrado em:** Arquitetura da solução (C4 model)

# Frameworks de Trabalho

O projeto "EntregaSegura" adotou uma variedade de frameworks e tecnologias, garantindo um desenvolvimento robusto e eficiente.

**Front-End (EntregaSegura Web):**

* Desenvolvido em **Angular 14**.
* Estilização e interface com **Angular Material**.
* Melhoria na experiência do usuário e interações aprimoradas com **Ngx-mask**, **Ngx-spinner** e **Ngx-toastr**.
* Autenticação gerenciada com **@auth0/angular-jwt**.
* Otimização de recursos e reatividade com **RxJS**.

**Back-End (EntregaSegura API):**

* Desenvolvido em **.NET 6**.
* Autenticação utilizando **JwtBearer**.
* Gestão de identidades com **Identity**.
* Manuseio de dados com **Entity Framework Core**.
* Validadores e regras de negócio com **Fluent Validation**.
* Testes unitários com **FluentAssertions** e **XUnit**.
* Mapeamento com **AutoMapper**.
* Criptografia baseado no algoritmo **Rfc2898DeriveBytes** para garantir a segurança das senhas dos usuários.

**Persistência de Dados:**

* **SQL Server Lite** é utilizado como a base de dados principal, por ser uma solução leve e eficiente para aplicações de menor escala, garantindo a integridade e confiabilidade dos dados sem a sobrecarga de sistemas de gerenciamento de banco de dados mais complexos.
* As consultas de endereço são gerenciadas através da integração com uma API externa chamada ViaCEP.

# Estrutura Base do Front End

*Esta seção será entregue na segunda etapa do projeto integrado, conforme cronograma disponibilizado pela PUC.*

# Modelo Relacional ou Projeto de Banco de Dados NoSQL

Diagrama

Descrição gerada automaticamente

**Figura 4:** Modelo Entidade Relacionamento (MER)

**Fonte:** Os Autores

**O arquivo original pode ser encontrado em:** Modelo Entidade Relacionamento (MER)

# Plano de Testes

*Esta seção será entregue na segunda etapa do projeto integrado, conforme cronograma disponibilizado pela PUC.*

# Apropriação de Horas no Projeto

|  |  |  |
| --- | --- | --- |
| **Histórico de apropriação de horas** | | |
| **Data do registro** | **Atividade** | **Quantidade de horas** |
| 04/03/2023 | Contextualização e objetivos do trabalho | 12:00 |
| 13/03/2023 | Elaboração da proposta de projeto | 10:00 |
| 23/03/2023 | Levantamento de requisitos funcionais | 18:00 |
| 02/04/2023 | Levantamento de requisitos não-funcionais | 8:00 |
| 10/04/2023 | Documentação dos Requisitos Levantados | 10:00 |
| 18/04/2023 | Elaboração do Diagrama de Casos de Uso | 16:00 |
| 26/04/2023 | Desenvolvimento do Diagrama de Classes de Domínio | 16:00 |
| 04/05/2023 | Definição do Padrão Arquitetural | 24:00 |
| 13/05/2023 | Estrutura de Banco de Dados | 18:00 |
| 15/07/2023 | Desenvolvimento do sistema | 200:00 |
| 24/07/2023 | Elaboração do Plano de Testes | 21:00 |
| 02/08/2023 | Revisão da documentação do Projeto | 18:00 |
| 10/08/2023 | Criação do repositório no GitHub e deploy do protótipo | 6:00 |
| 26/08/2023 | Revisão da Apropriação de Horas | 16:00 |
| 17/09/2023 | Revisão de Formatação do Relatório Técnico | 18:00 |
| 30/09/2023 | Avaliação retrospectiva | 06:00 |
| 03/10/2023 | Entrega final | 03:00 |

# Código da Aplicação

*Esta seção será entregue na segunda etapa do projeto integrado, conforme cronograma disponibilizado pela PUC.*

# Avaliação Retrospectiva

*Esta seção será entregue na segunda etapa do projeto integrado, conforme cronograma disponibilizado pela PUC.*

## 13.1 Objetivos Estimados

*Esta seção será entregue na segunda etapa do projeto integrado, conforme cronograma disponibilizado pela PUC.*

## 13.2 Objetivos Alcançados

*Esta seção será entregue na segunda etapa do projeto integrado, conforme cronograma disponibilizado pela PUC.*

## 13.3 Lições aprendidas

*Esta seção será entregue na segunda etapa do projeto integrado, conforme cronograma disponibilizado pela PUC.*

# Referências

Comércio eletrônico gerou R$ 169,6 bilhões em vendas em 2022. **Abranet (Associação Brasileira de Internet),** 25 de janeiro de 2023. Disponível em: <<https://www.abranet.org.br/Noticias/Comercio-eletronico-gerou-R$-169,6-bilhoes-em-vendas-em-2022-4181.html>> Acesso em: 10 de maio de 2023.