



**UNIVERSIDADE PAULISTA**

**ICET - INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA**

**CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E  
DESENVOLVIMENTO DE SISTEMAS**

**PROJETO INTEGRADO MULTIDISCIPLINAR**

**PIM IV**

**Desenvolvimento de um sistema integrado para o  
controle de operações em uma fazenda urbana, visando apoiar  
uma startup inovadora na área de segurança alimentar**

<b>Nome</b>	<b>R.A</b>
Bruno Siqueira Rosati	T953EE5
Gabriel Igor Dias Gomes	G873AJ3
Cristielen Fernanda Cardoso da Silva	N295AB6
Guilherme Bordinhon Silva Guimarães	N059CF8
Nicolas Douglas dos Santos	G854BB0

**SÃO JOSÉ DOS CAMPOS – SP**

**NOVEMBRO / 2024**

<b>Aluno</b>	<b>RA</b>
Bruno Siqueira Rosati	T953EE5
Gabriel Igor Dias Gomes	G873AJ3
Cristielen Fernanda Cardoso da Silva	N295AB6
Guilherme Bordinhon Silva Guimarães	N059CF8
Nicolas Douglas dos Santos	G854BB0

**Desenvolvimento de um sistema integrado para o controle de operações em uma fazenda urbana, visando apoiar uma startup inovadora na área de segurança alimentar**

Projeto Integrado Multidisciplinar (PIM) desenvolvido como exigência parcial dos requisitos obrigatórios à aprovação semestral no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da UNIP (Universidade Paulista), orientado pelo corpo docente do curso.

**São José dos Campos – SP**

**NOVEMBRO / 2024**

## RESUMO

Este projeto teve como objetivo o desenvolvimento de uma solução integrada para o controle e gestão das operações na fazenda UrbAgro, projetada para funcionar em três plataformas, sendo elas desktop, mobile e web com foco em garantir acessibilidade e flexibilidade para os responsáveis pela administração da fazenda. A metodologia ágil foi adotada, iniciando com o levantamento de requisitos para compreender as necessidades dos envolvidos e os processos da fazenda. Na modelagem do sistema foi utilizado o UML, com os diagramas de casos de uso, classes e sequência, para mapear as funcionalidades e a arquitetura da solução. A interface teve ênfase na usabilidade, buscando ser intuitiva e de fácil operação, de forma a permitir que os usuários interagissem de maneira eficiente. Como resultado, foram desenvolvidas as três aplicações bem estruturadas, atendendo às necessidades de cada ambiente e proporcionando maior agilidade e eficácia na gestão das operações agrícolas. Em conclusão, a solução implementada representou um avanço significativo, oferecendo uma ferramenta robusta e adaptável que atendeu de forma eficaz às demandas cotidianas, melhorando a produtividade e a tomada de decisões.

Palavras-Chave: Fazenda urbana, gestão, sistema, usuários, aplicações, KPI.

## SUMÁRIO

1. INTRODUÇÃO .....	5
2. PROGRAMAÇÃO ORIENTADA A OBJETOS II.....	6
3. DESENVOLVIMENTO DE SOFTWARE PARA INTERNET .....	7
4. TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS .....	9
5. PROJETO DE SISTEMAS ORIENTADO A OBJETOS .....	10
6. GERENCIAMENTO DE PROJETO DE SOFTWARE .....	11
7. EMPREENDEDORISMO.....	12
8. GESTÃO DA QUALIDADE.....	13
9. DESENVOLVIMENTO DO PROJETO .....	14
9.1 PROGRAMAÇÃO ORIENTADA A OBJETOS II.....	14
9.1.1. Estrutura do Código em C# .....	14
9.1.2. Vantagens da Abordagem.....	15
9.1.3. Implementação e Operações CRUD .....	16
9.1.4. Conformidade com a Modelagem .....	16
9.2 DESENVOLVIMENTO DE SOFTWARE PARA INTERNET .....	19
A estrutura CSS .....	20
A estrutura do JavaScript (JS).....	22
IMPLEMENTAÇÃO DO CRUD.....	24
9.3 TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS .....	25
9.3.1 Implementação de Classes .....	25
9.3.2 Herança.....	26
9.3.3 Abstração .....	26
9.3.4 Polimorfismo.....	27
9.3.5 Teste automatizados .....	28
9.3.6 Testes com Usuários.....	29
9.4 PROJETO DE SISTEMAS ORIENTADO A OBJETOS .....	31
9.4.1 Sistema Hierárquico .....	31
9.4.2 - Diagramas.....	32
9.5 GERENCIAMENTO DE PROJETO DE SOFTWARE.....	34
9.5.1. Iniciação do Projeto .....	34
9.5.1.1. Partes Interessadas.....	34
9.5.2. Planejamento do Projeto .....	35
9.5.2.1. Definição do Escopo do Projeto .....	35
9.5.2.2. Planejamento do Gerenciamento do Cronograma .....	35
9.5.3. Definição do Sistema de Comunicação.....	36
9.5.4. Planejamento do Gerenciamento dos Riscos.....	37
9.5.5. Execução do Projeto .....	37
9.5.5.1. Sistema de Comunicação Utilizado pelo Grupo .....	37
9.5.6. Monitoramento e Controle .....	38
9.5.6.1. Controle do Cronograma .....	38
9.5.7. Gerenciamento da Qualidade.....	38
9.5.8. Controle das Comunicações .....	38
9.5.9. Engajamento das Partes Interessadas.....	39
9.5.10. Encerramento do Projeto.....	39
9.5.11. Documentação .....	39
9.5.12. Avaliação de Desempenho.....	40
9.5.13. Transferência de Conhecimento .....	40
9.6 EMPREENDEDORISMO.....	41

9.7 GESTÃO DA QUALIDADE .....	44
10. CONCLUSÃO.....	47
11. REFERÊNCIAS .....	48
APÊNDICE A – Manual do Usuário da Versão Desktop .....	50
APÊNDICE B – Manual do Usuário da Versão Web .....	67

## 1. INTRODUÇÃO

O presente projeto tem como objetivo desenvolver um sistema integrado que facilitará o controle das operações em uma fazenda urbana, respondendo a uma necessidade crescente em contextos urbanos: a produção de alimentos frescos e saudáveis em meio à urbanização acelerada. À medida que as cidades se expandem, a demanda por práticas agrícolas sustentáveis e acessíveis torna-se cada vez mais relevante. A proposta deste sistema é oferecer uma solução inovadora que não apenas otimize a produção, mas também promova a interação entre produtores e consumidores, fortalecendo a comunidade local.

O sistema será projetado para ser acessível em diversas plataformas — mobile, desktop e web — garantindo que os usuários possam acessar informações e funcionalidades de maneira ampla e dinâmica, independentemente de sua localização. Essa versatilidade é fundamental para atender a um público diverso, que inclui desde agricultores urbanos e empreendedores até consumidores interessados em práticas de cultivo sustentável.

Entre os principais objetivos do projeto, destaca-se a criação de uma interface intuitiva e amigável, que facilite a gestão eficiente das atividades agrícolas, como o plantio, a irrigação, a colheita e a comercialização dos produtos. O sistema incorporará ferramentas avançadas para o monitoramento das condições de cultivo, como temperatura, umidade e nutrientes do solo, permitindo ajustes em tempo real e garantindo a qualidade e a segurança dos alimentos produzidos. Além disso, a análise de dados será uma peça-chave, possibilitando que os usuários tomem decisões informadas com base em informações precisas sobre o desempenho das culturas e as tendências de mercado.

Ao integrar funcionalidades como controle de estoque, gestão de vendas e interação com consumidores, o sistema visa não apenas otimizar as operações internas da fazenda, mas também fomentar um modelo de agricultura urbana mais colaborativo e consciente. A implementação deste sistema representa um passo significativo em direção à inovação no setor agrícola, contribuindo para a sustentabilidade e o fortalecimento da segurança alimentar nas áreas urbanas. Dessa forma, esperamos criar um ambiente em que a agricultura urbana prospere, beneficiando tanto os produtores quanto a comunidade que depende de alimentos frescos e saudáveis.

## 2. PROGRAMAÇÃO ORIENTADA A OBJETOS II

A Programação Orientada a Objetos (POO) é um paradigma no desenvolvimento de sistemas, ela é amplamente adotada em diferentes projetos devido a sua capacidade de modularização, reuso de código e facilidade de manutenção. Segundo Pressman (2016), a Programação Orientada a Objetos organiza o software em torno de objetos que representam todas as entidades do mundo real no contexto desejado, possibilitando que comportamentos e características sejam encapsulados dentro de classes e métodos.

No contexto proposto no PIM (Projeto Integrado Multidisciplinar), a aplicação das técnicas da POO é fundamental para garantir a organização e o desempenho esperado do projeto. Conforme proposto, o desenvolvimento será realizado em C#. Além disso, a implementação utilizará de princípios da orientação a objetos, o que facilitará a manutenção caso necessário do código.

De acordo com Oliveira e Souza (2018), um dos maiores desafios no desenvolvimento de software é alinhar a estrutura do código com a modelagem definida em diagramas de classe e de entidades-relacionamentos. Para isso, o Microsoft SQL Server será utilizado como sistema de gerenciamento de banco de dados (SGDB), garantindo não apenas a integridade dos dados, mas garantindo também a eficiência das operações CRUD (Create, Read, Update, Delete).

Outro ponto importante é a conformidade com a modelagem do software. Para garantir que o código implementado reflita as estruturas estabelecidas na fase de planejamento, é fundamental seguir os diagramas de classes. Gomes e Carvalho (2020) reforçam que, dessa forma, o desenvolvimento não apenas atende as demandas do sistema, mas também proporciona uma base sólida para futuras evoluções e manutenções do sistema.

Assim a POO, aliada as tecnologias descritas, permite a construção de um sistema eficiente e que cumpra os requisitos. A implementação em C#, juntamente com o uso do Microsoft SQL Server, assegura uma solução completa, atendendo as necessidades de um ambiente desktop e a integridade exigida em um sistema.

### 3. DESENVOLVIMENTO DE SOFTWARE PARA INTERNET

Para o desenvolvimento de um sistema integrado destinado ao controle de operações em uma fazenda urbana, a escolha de tecnologias apropriadas é fundamental para garantir que o sistema atenda às demandas de um ambiente inovador e dinâmico. Nesse contexto, o uso de tecnologias web, como HTML, CSS e JavaScript, torna-se imprescindível. A construção de uma plataforma eficaz começa com o HTML, que fornece a estrutura básica e sólida para a criação de sistemas organizados. No livro "HTML & CSS: Design and Build Websites 2011", Jon Duckett explora a importância de uma base bem construída em HTML, destacando como ela é crucial para o desenvolvimento de aplicações funcionais.

O CSS, por sua vez, complementa o HTML ao proporcionar uma apresentação visual refinada, garantindo que a interface do sistema seja tanto atraente quanto prática. *"CSS: The Definitive Guide 2018"*, de Eric A. Meyer, é uma referência essencial para compreender como o CSS pode transformar uma página simples em uma interface de usuário visualmente agradável e funcional. Já o JavaScript traz a interatividade necessária para tornar o sistema mais dinâmico e responsivo. Através de sua capacidade de manipular o DOM em tempo real, o JavaScript otimiza a experiência do usuário e permite atualizações em tempo real, algo fundamental para a automação de processos em uma fazenda inteligente. Em *"Eloquent JavaScript"*, Marijn Haverbeke explora como a linguagem pode ser usada para integrar sensores ambientais e adicionar algoritmos preditivos ao sistema.

Além dessas tecnologias de base, o uso de frameworks modernos como React e Vue.js proporciona um sistema ainda mais responsivo e eficiente. *"Learning React 2018"*, de Alex Banks e Eve Porcello, detalha como o React pode ser utilizado para criar interfaces de usuário que se atualizam dinamicamente, facilitando o controle e monitoramento das operações da fazenda. Da mesma forma, *"Vue.js Up and Running"*, de Callum Macrae, explica como o Vue.js simplifica o desenvolvimento de interfaces interativas, contribuindo para um sistema mais acessível e rápido.

No lado do back-end, o SQL Server é uma ferramenta robusta para gerenciar grandes volumes de dados, essencial para o controle eficiente das informações da fazenda. Combinado com o uso do Windows Server como plataforma de hospedagem, o sistema ganha estabilidade e segurança, fatores importantes para garantir a confiabilidade da solução. A integração dessas tecnologias resulta em uma plataforma escalável e eficiente, voltada para



startups que buscam inovações no campo da segurança alimentar. Ao unir essas ferramentas, cria-se um sistema completo que atende tanto às necessidades operacionais da fazenda quanto às exigências de inovação e sustentabilidade.

#### 4. TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

O desenvolvimento do software para a fazenda urbana fez uso extensivo de C#, uma linguagem de programação robusta e orientada a objetos que possibilitou a implementação eficiente dos princípios de Programação Orientada a Objetos (POO), como herança, polimorfismo, encapsulamento e abstração. Segundo Troelsen e Japikse (2017), C# é uma linguagem com sintaxe clara e forte tipagem, que facilita a criação de aplicações complexas, o que foi fundamental para o projeto.

Embora o JavaScript possa ser utilizado para adicionar funcionalidades avançadas, como algoritmos preditivos, a ênfase deste projeto foi nas capacidades do C# para implementar funcionalidades interativas. Isso otimizou a gestão da produção agrícola, tornando a fazenda mais eficiente e sustentável.

O sistema incorporou padrões de projeto, como Factory e Observer, e aplicou princípios SOLID, assegurando que fosse modular, escalável e de fácil manutenção. A herança, conforme abordado por Gamma et al. (1994), permitiu a criação de uma hierarquia de classes que favoreceu a reutilização de funcionalidades, enquanto o polimorfismo garantiu flexibilidade ao permitir que métodos se comportassem de maneiras distintas em subclasses específicas. A abstração foi aplicada para esconder detalhes complexos, focando nas interações principais das classes.

Testes automatizados foram realizados, assegurando a confiabilidade e robustez do sistema. Essa abordagem garantiu que o sistema atendesse aos requisitos funcionais e não funcionais, resultando em uma ferramenta eficaz para a gestão da fazenda urbana.

## 5. PROJETO DE SISTEMAS ORIENTADO A OBJETOS

O software de gerenciamento projetado da UrbAgro visa otimizar a administração e o controle das operações em um ambiente competitivo. Baseado na abordagem orientada a objetos (OO), o sistema integra funcionalidades essenciais.

Este projeto explora as funcionalidades do Urbagro, destacando sua arquitetura modular que facilita a gestão centralizada e a manutenção. Apresentaremos estratégias para a utilização dos recursos do software, justificando sua importância e vantagens. Além disso, discutiremos a metodologia de projeto adotada para garantir um desenvolvimento alinhado às melhores práticas de engenharia de software.

Nos capítulos 9, serão apresentados os modelos do sistema orientado a objetos, incluindo diagramas de classes, sequência e colaboração, que evidenciam a estrutura e a dinâmica do UrbAgro, ressaltando seu potencial para transformar a administração de pequenos empreendimentos agrícolas.

## 6. GERENCIAMENTO DE PROJETO DE SOFTWARE

O gerenciamento de projetos de software é uma tarefa complexa que requer uma abordagem estruturada para atender às expectativas das partes interessadas. No desenvolvimento de um sistema integrado para o controle de operações em uma fazenda urbana, a metodologia do Project Management Institute (PMI) orienta as equipes em todas as fases do ciclo de vida do projeto: iniciação, planejamento, execução, monitoramento e controle, e encerramento. Essa abordagem melhora a comunicação e aumenta a eficiência na entrega do software, ajudando a mitigar riscos e garantindo que todos os membros da equipe estejam alinhados com os objetivos, especialmente em projetos voltados para a segurança alimentar.

Na fase de iniciação, identificar as partes interessadas e definir o escopo do projeto é fundamental para justificar o desenvolvimento do software, considerando as necessidades de agricultores, gestores e consumidores. Isso facilita o comprometimento e a colaboração de todos os envolvidos.

O planejamento subsequente, que inclui a elaboração do cronograma, a estimativa de custos e a definição do sistema de comunicação, estabelece uma base sólida para a execução do projeto. Um planejamento detalhado permite antecipar desafios específicos, como regulamentações agrícolas e necessidades tecnológicas, criando um roteiro claro para o sucesso do sistema integrado. Como afirmam Kerzner e Saladis (2017), "um bom planejamento é a chave para o sucesso de qualquer projeto, pois fornece uma visão clara do que precisa ser feito e como alcançá-lo".

## 7. EMPREENDEDORISMO

O conceito de empreendedorismo é crucial para o desenvolvimento de um sistema integrado que gerencia o controle de fornecedores, produtos e produção em uma fazenda urbana. Segundo Schumpeter (1934), o empreendedorismo envolve a introdução de inovações que transformam a economia e criam oportunidades. Para a startup, isso significa identificar e explorar oportunidades para criar uma solução tecnológica que integre o gerenciamento de todas as operações da fazenda.

Drucker (2014) destaca que o empreendedorismo é a capacidade de transformar oportunidades em soluções práticas. Assim, a startup deve desenvolver uma plataforma que não apenas otimize o controle das operações internas, mas também integre eficientemente o gerenciamento de fornecedores e produtos. O sistema deve gerenciar desde o recebimento de insumos até o acompanhamento da produção e gestão do estoque, assegurando que todos os processos estejam alinhados para maximizar a eficiência e a segurança alimentar.

A visão empreendedora é igualmente importante para o sucesso deste sistema. Timmons (1999) argumenta que uma visão clara permite ao empreendedor planejar estratégias de longo prazo. Para a fazenda urbana, isso envolve antecipar as necessidades dos usuários e adaptar o sistema para gerenciar operações, fornecedores e produtos de forma eficiente. Essa flexibilidade é vital para se ajustar às mudanças no mercado e às novas demandas do setor de segurança alimentar.

O sistema proposto deve integrar funções como o controle de fornecedores, o gerenciamento da produção e o controle de produtos, garantindo qualidade, eficiência e rastreabilidade. A capacidade de inovar e adaptar o sistema às tendências e requisitos regulatórios emergentes é essencial, assegurando que a solução seja eficaz e competitiva no mercado.

## 8. GESTÃO DA QUALIDADE

A gestão da qualidade é crucial no desenvolvimento de sistemas informatizados, como o sistema integrado para controle de fornecedores, produtos e produção em uma fazenda urbana. Segundo Philip Crosby (1979), a qualidade deve ser incorporada desde o início, definindo requisitos claros para garantir eficiência operacional.

Ferramentas como Six Sigma e o Modelo de Maturidade de Capacidade (CMMI) são fundamentais nesse processo. Michael Hammer e James Champy (1993) destacam que o Six Sigma ajuda a reduzir variabilidade e melhorar a precisão na gestão de fornecedores e produção, enquanto W. Edwards Deming (1986) enfatiza a importância do treinamento da equipe para a aplicação dessas ferramentas.

Portanto, a gestão da qualidade e a utilização eficaz dessas ferramentas são essenciais para o sucesso do sistema integrado, assegurando que atenda às necessidades do projeto, promovendo a segurança alimentar e otimizando as operações da fazenda

## 9. DESENVOLVIMENTO DO PROJETO

### 9.1 PROGRAMAÇÃO ORIENTADA A OBJETOS II

O desenvolvimento do software foi feito utilizando classes independentes, promovendo a modularidade do código e sua organização. A escolha por não utilizar herança e focar em classes específicas se justifica pela clareza e facilidade na manutenção do código que essa abordagem proporciona.

#### 9.1.1. Estrutura do Código em C#

O software é composto por diversas classes, cada uma responsável por uma funcionalidade específica, garantindo assim que o código permaneça organizado e de fácil entendimento. Por exemplo, a classe Centralizador2, representada pela Figura 1, é responsável pela centralização dos elementos na interface gráfica.

Figura 1: Classe Centralizador2.

```
public Centralizador2(Form form)
{
    this.form = form;
}

2 referências
public void CentralizarEmDuasLinhas(Button[] linhaSuperior, Button[] linhaInferior)
{
    int margemVertical = 10; //Margem fixa entre as linhas
    int alturaTotal = linhaSuperior[0].Height + linhaInferior[0].Height + margemVertical; //Altura total

    //Calcula a posição Y para centralizar as linhas
    int posicaoYSuperior = (form.ClientSize.Height - alturaTotal) / 2;
    int posicaoYInferior = posicaoYSuperior + linhaSuperior[0].Height + margemVertical;

    Centralizar(linhaSuperior, posicaoYSuperior); //Posição para a linha superior
    Centralizar(linhaInferior, posicaoYInferior); //Posição para a linha inferior
}

2 referências
private void Centralizar(Button[] botoes, int posicaoY)
{
    int margem = 25; //Margem entre os botões
    int totalBotoes = botoes.Length;
    int larguraTotal = 0;

    //Calcula a largura total dos botões
    foreach (Button botao in botoes)
    {
        larguraTotal += botao.Width;
    }

    //Centraliza os botões na posição Y especificada
    int posicaoX = (form.ClientSize.Width - larguraTotal - (margem * (totalBotoes - 1))) / 2 + 50;

    for (int i = 0; i < totalBotoes; i++)
    {
        botoes[i].Location = new Point(posicaoX, posicaoY); //Usando a posição Y especificada
        posicaoX += botoes[i].Width + margem; //Move a posição X para o próximo botão
    }
}
```

Fonte: Os Autores.

Além dela, existe também, por exemplo, a classe `ColorBar2`, representada pela Figura 2, que é responsável pela implementação de uma barra, de cor verde. Essa separação de funções facilita a identificação de problemas e facilita a aplicação de melhorias.

Figura 2: Classe `ColorBar2`.

```
13 referências
public class ColorBar2
{
    22 referências
    public Panel Panel { get; private set; }

    7 referências
    public ColorBar2(Form form)
    {
        Panel = new Panel
        {
            Size = new Size(125, form.ClientSize.Height), // Largura 125, altura igual à do formulário
            Location = new Point(0, 0), // Posição no canto superior esquerdo
            BackColor = ColorTranslator.FromHtml("#BDDCAD") // Cor da barra
        };
    }
}
```

Fonte: Os Autores.

Além disso, existe também a classe `Form1`, que funciona como ponto de entrada da interface do usuário, utilizando instâncias das classes auxiliares para compor a estrutura visual da aplicação. Este enfoque modular também assegura que mudanças ou adições de funcionalidades possam ser implementadas de maneira isolada, reduzindo o risco de afetar outras partes do sistema em caso de erros.

### 9.1.2. Vantagens da Abordagem

A utilização de classes individuais, apresenta várias vantagens, dentre elas podemos destacar:

- **Simplicidade e Clareza:** O código é mais fácil de entender, pois cada classe tem um propósito claro e definido.
- **Fácil Manutenção:** Em caso de necessidade de manutenção, o desenvolvedor pode focar apenas nas classes necessárias, sem precisar se preocupar em impactar outras partes do sistema.
- **Reutilização de Código:** As classes podem ser reutilizadas em diferentes partes da aplicação, promovendo a eficiência.



### 9.1.3. Implementação e Operações CRUD

É essencial garantir que as operações de Create, Read, Update e Delete (CRUD) sejam implementadas de maneira robusta e eficiente. Para isso, as classes específicas que lidam com essas operações são desenvolvidas assegurando a integridade dos dados e a eficiência nas operações.

Por exemplo, a classe `CadastrarFunc` é responsável por gerenciar as operações relacionadas ao cadastro de funcionários. Essa abordagem permite que cada operação seja encapsulada dentro de métodos específicos, promovendo clareza e eficiência no código.

A implementação adequada das operações CRUD é vital para garantir que os dados sejam manipulados corretamente. Segundo Alencar e Nascimento (2017), um sistema eficiente deve não apenas permitir a execução dessas operações, mas também assegurar que sejam feitas de maneira robusta, evitando a perda de dados e garantindo a integridade da informação.

### 9.1.4. Conformidade com a Modelagem

É crucial que o código implementado siga os diagramas de modelagem rigorosamente. A classe `CadastrarFunc` deve garantir que as operações estejam em conformidade com a modelagem definida, refletindo a estrutura estabelecida na fase de planejamento. De acordo com Oliveira e Souza (2018), a aderência aos diagramas não apenas facilita a manutenção do código, mas também garante que o sistema atenda às necessidades do usuário final, evitando falhas de implementação.

A validação pode ser realizada por meio de testes unitários, que verificaram se cada classe e cada método operam e fornecem os resultados esperados. Essa prática é recomendada por Pressman (2016), que destaca a importância de testar as funcionalidades para garantir a qualidade do software.

## Aplicação Mobile:

Neste projeto, desenvolvemos uma aplicação utilizando Dart e o framework Flutter, que visa facilitar a gestão de pedidos e vendas em um contexto agrícola. A interface do usuário foi projetada para ser intuitiva e responsiva, proporcionando uma experiência agradável. A aplicação inclui uma tela de introdução (SplashScreen), um sistema de login com verificação de código, e uma tela principal que permite o acesso a funcionalidades e visualização dos pedidos, vendas e produção.

O Dart, em conjunto com o Flutter, foi a ferramenta escolhida para o desenvolvimento da interface gráfica devido à sua robustez, alto desempenho e capacidade de criar aplicativos nativos e responsivos para múltiplas plataformas, como web, desktop e dispositivos móveis. A flexibilidade do Flutter e a simplicidade do Dart permitiram uma rápida prototipagem e o desenvolvimento de uma aplicação interativa e de fácil manutenção.

A seguir, vamos explorar o código-fonte, algumas das classes principais:

Figura 2: class LoginScreen

Tela onde o usuário insere o ID e a senha para fazer o login. Após o login bem-sucedido, o código de verificação é enviado e o usuário deve inserir o código para prosseguir.

```
class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final TextEditingController idController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  String? verificationCode;
  bool isPhoneVerified = false;
  String errorMessage = '';

  void _sendVerificationCode() {
    verificationCode = '111111'; // Código fixo para verificação
    print("Código enviado: $verificationCode"); // Simula o envio do código
  }

  void _verifyCode(String code) {
    if (code == verificationCode) {
      setState(() {
        isPhoneVerified = true; // Atualiza o estado do telefone verificado
        errorMessage = ''; // Limpa a mensagem de erro
      });
    }
  }
}
```

Fonte: Os Autores.

Figura 3: class HomeScreen

Tela inicial do aplicativo após o login, que oferece opções de navegação para outras telas, como Pedidos, Vendas e Produção.

```
class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('UrbAgro'),
        actions: [
          PopupMenuButton<String>(
            onSelected: (value) {
              if (value == 'logout') {
                _showLogoutDialog(context);
              }
            },
            itemBuilder: (BuildContext context) {
              return [
                PopupMenuItem<String>(
                  value: 'config',
                  child: Text('Configurações'),
                ), // PopupMenuItem
                PopupMenuItem<String>(
                  value: 'version',
                  child: Text('Versão'),
                ), // PopupMenuItem
              ];
            },
          ),
        ],
      ),
    );
  }
}
```

Fonte: Os Autores.

Figura 2: class MenuButton

Componente reutilizável de botão de menu, usado em várias telas para navegar para diferentes partes da aplicação, como "Pedidos", "Vendas" e "Produção".

```
class MenuButton extends StatelessWidget {
  final IconData icon;
  final String label;
  final VoidCallback onPressed;

  MenuButton({required this.icon, required this.label, required this.onPressed});

  @override
  Widget build(BuildContext context) {
    return Container(
      width: double.infinity,
      height: 80,
      child: ElevatedButton(
        onPressed: onPressed,
        style: ElevatedButton.styleFrom(
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
          ), // RoundedRectangleBorder
          backgroundColor: Colors.white,
        ),
      ),
    );
  }
}
```

Fonte: Os Autores.

## 9.2 DESENVOLVIMENTO DE SOFTWARE PARA INTERNET

### **Testes de Usabilidade:**

Recentemente, realizamos dois testes de usabilidade no sistema para identificar pontos de melhoria na experiência dos usuários. O primeiro ajuste foi direcionado à barra de logout, que apresentava dificuldades de acesso e manuseio, gerando confusão entre os usuários. Após os testes, a barra foi reposicionada e otimizada para que a opção de logout se tornasse mais visível e intuitiva, permitindo uma saída rápida e segura do sistema. Esse ajuste visou minimizar erros e facilitar a navegação, reduzindo o tempo gasto com processos de logout.

Em complemento, foi realizada uma análise detalhada dos rótulos dos botões de ação, que mostravam certa ambiguidade para os usuários, dificultando a compreensão imediata de suas funções. Com o objetivo de aprimorar a clareza, esses botões foram renomeados utilizando termos mais específicos e diretos, esclarecendo de maneira objetiva o propósito de cada ação.

Essas melhorias não apenas tornaram o sistema mais direto, mas também proporcionaram uma interação mais eficiente, na qual os usuários podem navegar com segurança e confiança. A mudança reflete um esforço contínuo em otimizar a interface, priorizando a usabilidade e a satisfação do usuário, além de reduzir possíveis barreiras de acesso dentro do sistema.

### **A estrutura HTML**

é projetada para proporcionar uma navegação intuitiva e uma apresentação clara das informações. Ela é composta por uma barra lateral de navegação e uma área de conteúdo principal, ambas organizadas dentro de um contêiner com a classe container. Esta abordagem permite um layout responsivo e bem estruturado, essencial para a experiência do usuário.

A barra lateral, identificada pela classe sidebar, contém uma seção de perfil que inclui uma imagem do usuário (profile.png) e uma saudação personalizada, além de um logotipo da UrbAgro (logo.png). Essa seção proporciona um toque pessoal ao usuário, facilitando a identificação e o engajamento com a interface.

O conteúdo principal da página, definido pela classe main-content, abriga um menu de navegação que apresenta diversas funcionalidades do sistema, como Fornecedores, Pedidos, Produtos, Funcionários, Vendas e Produção. Os itens do menu são construídos com a classe menu-item, que combina uma imagem de ícone e uma descrição textual. Cada ícone

possui uma imagem associada e um texto alternativo, garantindo acessibilidade e melhorando a usabilidade.

A escolha do HTML como linguagem de marcação é fundamental, pois permite estruturar o conteúdo de maneira semântica e hierárquica. Isso não apenas facilita a leitura e a manutenção do código, mas também assegura que a página seja interpretada corretamente pelos navegadores e mecanismos de busca. O uso de classes e identificadores aprimora a estilização e a navegação, tornando a interface mais agradável e funcional.

Figura 1: Tela inicial (HTML)

```
<div class="container"> <!-- Container principal que envolve todo o conteúdo da página -->
  <div class="sidebar"> <!-- Barra lateral de navegação -->
    <div class="profile"> <!-- Seção de perfil do usuário -->
       <!-- Imagem do perfil do usuário com texto alternativo -->
      <p>Olá! Que bom vê-lo novamente, <span>Gabriel</span></p> <!-- Mensagem de saudação personalizada para o usuário -->
    </div>
    <div class="logo"> <!-- Seção para o logo da UrbAgro -->
       <!-- Imagem do logo da UrbAgro com texto alternativo -->
    </div>
  </div>
  <div class="main-content"> <!-- Conteúdo principal da página -->
    <!-- Itens do menu de navegação -->
    <div class="menu-item"> <!-- Item do menu para Fornecedores -->
       <!-- Ícone para Fornecedores -->
      <p>FORNECEDORES</p> <!-- Texto do menu -->
    </div>
    <div class="menu-item"> <!-- Item do menu para Pedidos -->
       <!-- Ícone para Pedidos -->
      <p>PEDIDOS</p> <!-- Texto do menu -->
    </div>
    <div class="menu-item"> <!-- Item do menu para Produtos -->
       <!-- Ícone para Produtos -->
      <p>PRODUTOS</p> <!-- Texto do menu -->
    </div>
    <div class="menu-item"> <!-- Item do menu para Funcionários -->
       <!-- Ícone para Cadastro de Funcionários -->
      <p>FUNCIONÁRIOS</p> <!-- Texto do menu -->
    </div>
    <div class="menu-item"> <!-- Item do menu para Vendas -->
       <!-- Ícone para Vendas -->
      <p>VENDAS</p> <!-- Texto do menu -->
    </div>
    <div class="menu-item"> <!-- Item do menu para Produção -->
       <!-- Ícone para Produção -->
      <p>PRODUÇÃO</p> <!-- Texto do menu -->
    </div>
  </div>
</div>
```

Fonte: Os autores.

## A estrutura CSS

A estrutura CSS apresentada na Figura 2 é responsável por definir o estilo visual de componentes essenciais, como o perfil do usuário, o logotipo, o conteúdo principal e os itens do menu. Esse uso do CSS é crucial para garantir uma apresentação estética coesa e atrativa, melhorando a experiência do usuário.

A classe “.profile p” ajusta o espaçamento superior do parágrafo e define a cor do texto como cinza escuro (#333), proporcionando uma legibilidade aprimorada. Além disso, o

elemento “span” dentro do perfil recebe um texto em negrito com “font-weight: bold”, destacando informações importantes e ajudando na hierarquia visual do conteúdo.

A imagem do logotipo, na classe “.logo img”, é dimensionada para uma largura de 100px e centralizada com uma margem superior automática, o que contribui para um layout harmonioso e equilibrado. O uso de unidades de medida específicas permite um controle preciso sobre a aparência do logotipo, garantindo que ele se encaixe adequadamente no design.

O conteúdo principal, estilizado pela classe “.main-content”, utiliza o modelo de layout flexbox, ocupando 80% da largura da tela e alinhando seus itens com um espaçamento de 20px entre eles. Essa abordagem flexível permite uma organização dinâmica dos elementos, ajustando-se facilmente a diferentes tamanhos de tela e dispositivos.

Os itens do menu, identificados pela classe “.menu-item”, possuem um fundo cinza claro (“#dcdcdc”), dimensões de 200x150px, bordas arredondadas e centralização de texto. Essas características não apenas conferem um visual moderno e amigável, mas também organizam o conteúdo de forma intuitiva, facilitando a navegação.

A utilização do CSS é fundamental não apenas para a estilização, mas também para a responsividade e acessibilidade da interface. Com o CSS, é possível criar uma experiência visualmente atraente e funcional, onde cada elemento é cuidadosamente projetado para contribuir para a usabilidade e estética geral da aplicação.

Figura 2:Tela Inicial (CSS)

```
/* Estiliza o parágrafo dentro do perfil, adicionando uma margem superior de 10px e mudando a cor do texto para cinza escuro (#333). */
.profile p {
  margin-top: 10px;
  color: #333;
}

/* Estiliza o texto dentro do span no perfil, tornando o texto em negrito. */
.profile span {
  font-weight: bold;
}

/* Define a largura da imagem de logo dentro do container .logo para 100px e adiciona uma margem superior automática para empurrar a imagem para o fim da barra lateral. */
.logo img {
  width: 100px;
  margin-top: auto;
}

/* Estiliza o conteúdo principal, ocupando 80% da largura da tela, com elementos internos organizados de forma flexível, permitindo quebra de linha (flex-wrap). Os itens são centralizados e espaçados por um gap de 20px, com padding interno de 20px. */
.main-content {
  width: 80%;
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  align-items: center;
  gap: 20px;
  padding: 20px;
}

/* Estiliza cada item do menu, definindo um fundo cinza claro (#dcdcdc), tamanho fixo de 200x150 pixels, organizando o conteúdo em coluna e centralizando-o. Além disso, os itens têm bordas arredondadas e o texto é centralizado. */
.menu-item {
  background-color: #dcdcdc;
  width: 200px;
  height: 150px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  border-radius: 10px;
  text-align: center;
}
```

Fonte: Os autores.

## A estrutura do JavaScript (JS)

O JavaScript é uma linguagem de programação robusta e flexível, composta por diversos elementos que trabalham em conjunto para possibilitar o desenvolvimento de aplicações dinâmicas. Na Figura 3, podemos observar a organização básica do código JavaScript, que inclui variáveis, funções e objetos, cada um desempenhando um papel vital na construção da lógica da aplicação.

As variáveis são fundamentais, pois armazenam dados que podem ser manipulados ao longo do programa. Elas são declaradas com palavras-chave como “var”, “let” ou “const”, permitindo uma gestão eficiente do escopo e da imutabilidade dos valores. O uso adequado de variáveis contribui para a clareza do código e facilita a manipulação de dados conforme necessário.

As funções agrupam conjuntos de instruções que podem ser reutilizadas, promovendo a manutenção e a legibilidade do código. Ao encapsular lógicas específicas, as funções não



apenas simplificam o fluxo do programa, mas também permitem que diferentes partes do código sejam testadas e alteradas de forma independente, o que é crucial para o desenvolvimento ágil.

Os objetos, por sua vez, desempenham um papel central na programação JavaScript, permitindo a criação de coleções de propriedades e métodos que representam entidades do mundo real. Essa abordagem orientada a objetos é essencial para modelar aplicações mais complexas, oferecendo uma maneira de organizar e estruturar dados de forma lógica e acessível. Além disso, o JavaScript conta com um sistema de eventos que possibilita a interação com o usuário, respondendo a ações como cliques e pressionamento de teclas, o que enriquece a experiência interativa da aplicação.

Em resumo, a estrutura do JavaScript, conforme ilustrado na Figura 3, combina variáveis, funções e objetos para formar uma base sólida para o desenvolvimento de aplicações dinâmicas e interativas. O uso do JavaScript é fundamental, pois permite que os desenvolvedores criem interfaces responsivas e funcionais, proporcionando uma experiência rica e envolvente aos usuários.

Figura 3: Tela Inicial (JS)

```
// Classe para gerenciar os ícones de navegação
class NavigationIcons {
  constructor(iconConfig) {
    this.iconConfig = iconConfig; // Recebe uma configuração de ícones com seus URLs
    this.initEventListeners(); // Inicializa os ouvintes de eventos
  }

  // Adiciona ouvintes de eventos a cada ícone para redirecionar ao clicar
  initEventListeners() {
    this.iconConfig.forEach(icon => {
      // Seleciona o elemento do ícone pelo atributo 'src'
      const element = document.querySelector(`img[src="${icon.src}"]`);
      if (element) {
        // Adiciona um evento de clique para redirecionar para a URL associada ao ícone
        element.addEventListener('click', () => {
          window.location.href = icon.url;
        });
      }
    });
  }
}

// Inicializa o sistema após o carregamento do DOM
document.addEventListener('DOMContentLoaded', () => {
  // Inicializa o menu lateral passando o seletor da imagem de perfil
  new SideMenu('.profile img');

  // Configuração dos ícones de navegação com suas respectivas URLs
  const icons = [
    { src: 'pedidos-icon.png', url: 'ListaPedidos.html' },
    { src: 'fornecedores-icon.png', url: 'Fornecedores.html' },
    { src: 'vendas-icon.png', url: 'Vendas.html' },
    { src: 'registro.png', url: 'DadosFuncionarios.html' },
    { src: 'producao-icon.png', url: 'Producao.html' },
    { src: 'produtos-icon.png', url: 'Produtos.html' }
  ];

  // Inicializa os ícones de navegação com base na configuração
  new NavigationIcons(icons);
});
```

Fonte: Os autores.



## IMPLEMENTAÇÃO DO CRUD

A implementação do CRUD (Criar, Ler, Atualizar e Deletar) é fundamental em diversas aplicações web, pois fornece um conjunto de operações essenciais para o gerenciamento eficaz de dados. Isso pode ser integrado em aspectos como a navegação e os ícones da interface, conforme mostrado na **Figura 3**. Embora o foco da imagem seja a navegação entre páginas, cada operação CRUD está associada a diferentes partes da aplicação acessadas por meio dos ícones.

Por exemplo, ao clicar no ícone de "Pedidos", o usuário é redirecionado para uma página dedicada onde pode Criar novos pedidos através de um formulário, Ler pedidos já existentes que estão listados na tela, Atualizar pedidos ao editar informações específicas e **Deletar** pedidos indesejados. Essa sequência de operações permite ao usuário gerenciar pedidos de maneira eficiente, garantindo que ele tenha total controle sobre as informações apresentadas.

De maneira semelhante, o ícone de "Fornecedores" direciona o usuário a uma página onde ele pode realizar operações CRUD para gerenciar fornecedores. Isso inclui adicionar novos fornecedores, visualizar a lista de fornecedores cadastrados, atualizar informações e inativar ou excluir registros que não são mais necessários. Essa funcionalidade é vital para manter a base de dados sempre atualizada e organizada.

A classe NavigationIcons, representada na imagem, desempenha um papel crucial ao estruturar a navegação. Ela conecta os ícones a páginas específicas que suportam as operações CRUD. Cada ícone está associado a uma URL que leva a uma página representando uma entidade, como pedidos, fornecedores, vendas ou produtos. Assim, ao acessar essas páginas através dos ícones, os usuários podem executar operações de CRUD de forma intuitiva e eficiente, facilitando o gerenciamento dos dados da aplicação.

A utilização do CRUD não só melhora a funcionalidade da aplicação, mas também proporciona uma experiência de usuário mais fluida e lógica. Com um sistema de navegação claro e operações bem definidas, os usuários podem interagir com a aplicação de maneira mais efetiva, resultando em maior satisfação e eficiência no uso do sistema.

### 9.3 TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

O sistema foi desenvolvido para um sistema de uma fazenda urbana, com foco no monitoramento e controle de operações relacionadas a fornecedores, produtos, pedidos, produção, vendas e cadastro de funcionários. Utilizando a linguagem C# e os princípios da Programação Orientada a Objetos (POO), como herança, polimorfismo e abstração, a arquitetura do sistema foi projetada para ser modular e escalável, permitindo fácil manutenção e futura expansão. O acesso ao sistema é restrito ao suporte de TI e ao gerente, que serão responsáveis por supervisionar e gerenciar as atividades e funcionalidades. Enquanto os funcionários irão visualizar tarefas e produção no mobile

A modelagem do sistema foi baseada em um Diagrama de Classe, que serviu como guia para a implementação das principais entidades e suas relações (Larman, 2004). As classes que representam fornecedores, produtos e o cadastro de funcionários foram implementadas seguindo os conceitos de abstração e polimorfismo, como sugerido por Gamma et al. (1994), que afirmam que o uso de herança permite a criação de uma hierarquia de classes que favorece a reutilização de funcionalidades. Isso garantiu que apenas as operações necessárias fossem expostas, protegendo assim os dados sensíveis. Por exemplo, a classe **Fornecedor** implementa métodos específicos que utilizam o polimorfismo para adaptar o comportamento de acordo com diferentes tipos de fornecedores, enquanto a classe Produto utiliza herança para derivar diferentes categorias de produtos, permitindo uma estrutura organizada e reutilizável.

A implementação de padrões de projeto, como o padrão Factory, facilitou a criação e gerenciamento de objetos, assegurando que o sistema fosse flexível e fácil de expandir, conforme abordado por Troelsen e Japikse (2017), que ressaltam a importância da sintaxe clara e da forte tipagem no desenvolvimento de aplicações complexas. Além disso, foram realizados testes automatizados para garantir a confiabilidade e robustez do software, assegurando que o sistema atenda aos requisitos funcionais e não funcionais.

#### 9.3.1 Implementação de Classes

No sistema, as classes foram utilizadas para representar os principais componentes, como **Fornecedor**, **Produto** e **Pedido**. Cada classe contém atributos e métodos que representam o comportamento e as características desses componentes. Essa estrutura modular permite que cada parte do sistema seja gerenciada de forma independente e organizada, facilitando futuras expansões e modificações.

### 9.3.2 Herança

A herança foi utilizada para ampliar e reaproveitar funcionalidades de maneira eficiente no sistema, promovendo organização e reutilização de código. Por exemplo, a classe `AlteradorFonteFornecedores` herda de uma classe base, `AlteradorFonteMenu`, possibilitando o uso de métodos e propriedades compartilhadas entre as classes relacionadas à interface gráfica. Isso permite implementar funcionalidades específicas para a tela de Fornecedores sem duplicar o código, aproveitando o que já foi definido em `AlteradorFonteMenu`. Com essa abordagem, o sistema mantém uma estrutura modular, facilitando a manutenção, a consistência e a adaptação dos componentes às necessidades do projeto.

Figura: 5 Classe (`Alteradorfontemenu`)

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TelaPimExercicio
{
    3 referências
    internal class AlteradorFonteFornecedores : AlteradorFonteMenu
    {
        //Construtor que chama o construtor da classe base
        1 referência
        public AlteradorFonteFornecedores(Form form) : base(form)
        {
        }

        //Altera o tamanho da fonte dos botões específicos da página de Fornecedores
        1 referência
        public void AlterarFonteFornecedores(Button btnLogout, Button btnRetornar)
        {
            float tamanhoFonte = 10; //Variável que define o tamanho da fonte

            btnLogout.Font = new Font(btnLogout.Font.FontFamily, tamanhoFonte);
            btnRetornar.Font = new Font(btnRetornar.Font.FontFamily, tamanhoFonte);
        }
    }
}
```

Fonte: Os Autores.

### 9. 3.3 Abstração

A abstração foi aplicada para ocultar detalhes complexos e expor apenas as funcionalidades essenciais dos componentes visuais e interativos, como a exibição de logotipo, barra de cor e plano de fundo. No desenvolvimento, foram criadas classes auxiliares, como `Logo`, `ColorBar2` e `ColorBackground`, que encapsulam a lógica específica desses elementos, tornando a tela `TelaProdutos` mais limpa e focada em sua funcionalidade principal.

Por exemplo, a classe Logo permite a adição e o posicionamento da imagem de logotipo sem que a tela precise lidar diretamente com os detalhes de carregamento, dimensionamento e posicionamento do controle PictureBox. Essa abordagem facilita a manutenção e evolução do sistema, permitindo que qualquer modificação nos componentes visuais seja feita diretamente nas classes auxiliares, sem impacto na estrutura principal da aplicação.

Figura: 6 (Tela Produtos)

```
namespace TelaPimExercicio
{
    public partial class TelaProdutos : Form
    {
        private Logo logo;
        private ColorBar2 colorBar;
        private ColorBackground colorBg;
        private Centralizador2 centralizador2;
        private Logout logout;

        public TelaProdutos()
        {
            InitializeComponent();

            //Permitindo o redimensionamento da tela
            this.FormBorderStyle = FormBorderStyle.Sizable;
            this.MaximizeBox = true;

            //Definindo o tamanho mínimo da tela para 800x600
            this.MinimumSize = new Size(800, 600);

            //Criando e adicionando a logo
            logo = new Logo(this);
            this.Controls.Add(logo.Picture);

            // Posicionando a logo no canto inferior esquerdo
            logo.Picture.Location = new Point(20, this.ClientSize.Height - logo.Picture.Height - 10);

            //Criando e adicionando a barra verde superior
            colorBar = new ColorBar2(this);
            this.Controls.Add(colorBar.Panel);

            //Criando e adicionando a cor verde de fundo
            colorBg = new ColorBackground(this);
            this.Controls.Add(colorBg.Panel);

            this.Resize += TelaFornecedores_Resize;

            //Iniciando o Logout
            logout = new Logout(this);
        }
    }
}
```

Fonte: os autores.

### 9.3.4 Polimorfismo

O polimorfismo foi empregado para permitir que métodos e comportamentos diferentes dependam da classe específica que os implementam. No contexto do sistema, as classes `AlteradorFonteCadastrarFuncionario` e `AlteradorFonteFornecedores` herdam da classe base `AlteradorFonteMenu`. Embora não haja um método polimórfico implementado nos exemplos fornecidos, o conceito pode ser aplicado se introduzirmos um método virtual na classe base, como `AlterarFonte()`.

Por exemplo, a classe `AlteradorFonteMenu` poderia definir um método virtual `AlterarFonte(Button btnLogout, Button btnRetornar)`. As subclasses, `AlteradorFonteCadastrarFuncionario` e `AlteradorFonteFornecedores`, poderiam implementar esse método de maneiras diferentes, ajustando o tamanho da fonte ou estilo dos botões conforme necessário para cada página. Isso permitiria que o sistema tratasse essas variações de maneira flexível, usando uma interface comum para alterar a fonte dos botões em diferentes contextos.

Dessa forma, ao invocar **`AlterarFonte()`** em uma instância de **`AlteradorFonteMenu`**, o sistema poderia decidir qual implementação usar, dependendo do tipo de objeto específico que está sendo manipulado. Essa abordagem facilita a extensão do sistema, permitindo que novas classes de alteração de fonte sejam adicionadas sem modificar o código existente, garantindo a manutenção da estrutura modular do software.

Figura: 7 (Alterador fonte cadastrar funcionário)

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TelaPimExercicio
{
    internal class AlteradorFonteCadastrarFuncionario : AlteradorFonteMenu
    {
        // Construtor que chama o construtor da classe base
        public AlteradorFonteCadastrarFuncionario(Form form) : base(form)
        {
        }

        // Altera o tamanho da fonte dos botões específicos da página de Cadastrar Funcionário
        public void AlterarFonteCadastrarFuncionario(Button btnLogout, Button btnRetornar)
        {
            float tamanhoFonte = 10; // Variável que define o tamanho da fonte

            btnLogout.Font = new Font(btnLogout.Font.FontFamily, tamanhoFonte);
            btnRetornar.Font = new Font(btnRetornar.Font.FontFamily, tamanhoFonte);
        }
    }
}
```

Fonte: Os Autores.

### 9.3.5 Teste automatizados

Para assegurar a confiabilidade e robustez do sistema de gestão da fazenda urbana, adotou-se uma estratégia de testes automatizados com a ferramenta NUnit, reconhecida por sua eficácia em ambientes .NET. A escolha do NUnit baseou-se em sua integração fluida com a linguagem C#, além de oferecer um ambiente robusto para a escrita e execução de testes de unidade, essenciais para validar a lógica de negócio nas classes de fornecedores, produtos, pedidos, produção e cadastro de funcionários.

O desenvolvimento dos testes seguiu uma abordagem rigorosa fundamentada nos princípios SOLID, particularmente o Princípio da Responsabilidade Única, que permite que

cada classe mantenha um conjunto de testes especializado, voltado exclusivamente para suas funcionalidades. Dessa forma, os testes automatizados abordaram métodos críticos e validaram a correção de cada funcionalidade em diferentes cenários operacionais, conforme documentado.

Além disso, foi utilizado o conceito de *mocking* para simular o comportamento de dependências, possibilitando o isolamento das funcionalidades em teste e reduzindo o risco de falhas decorrentes de interdependências entre classes. Foram realizadas verificações detalhadas para operações de CRUD (Create, Read, Update, Delete), garantindo a integridade e consistência dos dados ao longo das operações realizadas pelo sistema.

De acordo com a documentação oficial do NUnit, a estrutura modular dos testes proporciona facilidade na expansão e manutenção futura do sistema, permitindo que novos módulos e funcionalidades sejam testados de forma eficiente e integrados sem comprometer a estabilidade do software. A validação automatizada, documentada e continuamente atualizada ao longo do desenvolvimento, é, portanto, essencial para manter a confiabilidade e facilitar a evolução do sistema, atendendo às necessidades da fazenda urbana com precisão e segurança.

### 9.3.6 Testes com Usuários

Para validar a usabilidade e funcionalidade do sistema, foram realizados testes práticos com usuários que não participaram do desenvolvimento e que não tinham familiaridade prévia com o sistema. Durante esses testes, os participantes foram orientados a realizar tarefas específicas, como cadastro de fornecedores, inserção de pedidos e controle de produção. O objetivo foi observar como esses usuários interagiam com o sistema e identificar eventuais dificuldades ou barreiras que poderiam comprometer a experiência de uso.

Esses testes foram fundamentais para avaliar a eficiência operacional e a experiência geral do usuário na interface. Com base no feedback recebido, ajustes foram implementados para otimizar a usabilidade do sistema, garantindo uma interação mais intuitiva e eficiente para os futuros usuários.

Resultados dos testes:

Figura:8 (Planilha de Tarefas)

TAREFA	DESCRIÇÃO	RESULTADO OBITIDO
ACESSAR LOGIN	Dar um login e senha para o usuário para ver se ele obtem sucesso no login	OK
CADASTAR FORNECEDOR	Pedir ao usuário para realizar o cadastro de um novo fornecedor.	OK
TELA INICIAL	Orientar o usuário a retornar à tela inicial.	OK
CADASTRAR PRODUTOS	Solicitar ao usuário que cadastre um novo produto no sistema.	OK
INICIAR UMA PRODUÇÃO	Solicitar ao usuário a produção de 200 unidades de cenouras.	OK
EXCLUIR UM PEDIDO	Solicitar ao usuário excluir o pedido 1	OK
EXCLUIR PRODUTOS	Solicitar ao usuário excluir produto 6	OK
INATIVAR UM FORNECEDOR	Pedir ao usuário para Inativar o fornecedor 2	OK
EDIÇÃO NOS PRODUTOS	Solicitar ao usuário que faça uma edição no produto 3.	OK
SAIR	Orientar o usuário a sair do sistema.	OK

Fonte: Os Autores.

## 9.4 PROJETO DE SISTEMAS ORIENTADO A OBJETOS

Para o desenvolvimento do sistema UrbAgro, destinado à gestão de uma fazenda urbana, foram reaproveitados os diagramas de prototipagem já elaborados, com aprimoramentos e incrementos para maior detalhamento do sistema. O sistema será implementado utilizando código em C# para a aplicação em desktop, linguagem Dart para a aplicação Mobile e Javascript, HTML e CSS para o Web, aplicando os conceitos de herança e polimorfismo para otimizar o reaproveitamento de código.

Em relação aos diagramas utilizados, diversos protótipos foram reaproveitados e aprimorados, tais como o diagrama de casos de uso, o diagrama de classes e o diagrama de sequência. Para um maior detalhamento e compreensão do sistema, foram adicionados novos elementos, como o diagrama de implantação, que ilustra a estrutura física do sistema, e o diagrama de comunicação, que complementa o diagrama de sequência. Esses diagramas, em conjunto, facilitam o entendimento do comportamento dos casos de uso, permitindo uma visualização clara do fluxo de execução por meio do diagrama de comunicação.

### 9.4.1 Sistema Hierárquico

O sistema UrbAgro terá três níveis hierárquicos de acesso:

- **Suporte** – O suporte terá todos os níveis de acesso do sistema, ele gerenciará toda parte de segurança, cuidará de fazer recuperação de login em caso de esquecimento, só ele terá a permissão de excluir algo do sistema caso necessário e fará o monitoramento do sistema UrbAgro utilizando o sistema Desktop.
- **Gerente** – O gerente do setor de produção, usará o sistema Web, para fazer o gerenciamento de toda parte de produção, vendas, pedidos e cadastro de novos produtos, inativações no sistema e cadastro de funcionários.
- **Funcionários** - Os funcionários ficaram na parte da produção dos alimentos, eles terão o acesso mais básico e rápido do sistema, e terão a sua disponibilidade o



sistema Mobile, contando com apenas 3 acessos para sua hierarquia, que será as informações básicas para que a produção caminhe.

#### 9.4.2 - Diagramas

- **Diagrama de caso de uso** – Foram construídos 3 diagramas, para exemplificar os 3 tipos de acessos existentes no sistema, nele apresenta os atores, Suporte do sistema, Gerente e Funcionário. Os diagramas detalham retamente todos as permissões e acessos que cada usuário tem dentro do sistema.
- **Diagrama de Classes** – Na construção estética do sistema, foram utilizados 11 tipos de classes, com seus métodos e atributos apresentados, no diagrama também está presente os tipos de relacionamento com as demais, mostrando sua cardinalidade, algumas existindo com heranças e polimorfismo.
- **Diagrama de Implantação** – Como todo sistema a UrbAgro tem sua representação física do sistema a partir do diagrama de implantação. No diagrama foi-se apresentado todos os componentes utilizados como switch, firewall interno e externo e um roteador de sinal, as linguagens de comunicação entre as plataformas HTML, CSS e Java Script para Website, C# para Desktop e Dart para mobile. As 3 plataformas foram separadas entre os usuários, na qual melhor se encaixava a sua função.

Para melhor segurança de informações, foi escolhido usar um banco de dados em nuvem, conectado com o servidor da UrbAgro para se comunicar com os demais dispositivos da fazenda urbana.

- **Diagrama de Sequências** – Na projeção do sistema, ele foi utilizado para mostrar a interação do usuário com o sistema, a ordem temporal com que ocorre cada ação, mostrando o caminho que ele percorre para chegar ao local necessário.

Para cada funcionalidade do sistema, foi-se feito um diagrama de sequências, junto a um diagrama de classes, para ilustrar uma interface na qual o usuário irá interagir. As representações feitas foram divididas entre as 3 hierarquias presente no sistema, suporte, gerente produção e funcionário.

- **Diagrama de Comunicação** – Para complementar o diagrama de sequência e detalhar de forma mais precisa as mensagens trocadas entre os objetos durante a execução dos casos de uso no sistema da UrbAgro, foi utilizado o diagrama de comunicação. O diagrama permite uma visualização clara das interações entre os diferentes componentes do sistema,

Por exemplo o gerente de produção quando vai adicionar uma nova produção ao sistema, cada botão que ele clica, cada interface que ele entra, gera um tipo de comunicação entre os objetos, e este diagrama destaca como as mensagens são enviadas e recebidas para garantir o correto funcionamento das funcionalidades.

Além disso, com dito anteriormente, o sistema conta com três níveis de hierarquia, o diagrama de comunicação evidencia a maneira como usuários interagem com o sistema, considerando as restrições de acesso e as funcionalidades disponíveis de acordo com o perfil de cada usuário. Dessa forma, é possível compreender como as permissões e os fluxos de informação variam conforme o papel desempenhado por cada usuário dentro da plataforma, assegurando que as operações sejam realizadas de maneira segura e eficiente.

## 9.5 GERENCIAMENTO DE PROJETO DE SOFTWARE

### 9.5.1. Iniciação do Projeto

A fase de iniciação é crucial para o alinhamento das expectativas e a identificação das partes interessadas. No contexto do desenvolvimento de um software para a gestão de uma fazenda urbana, as partes interessadas são:

#### 9.5.1.1. Partes Interessadas

##### **Professores e Alunos**

**Importância:** A interação entre professores e alunos é fundamental para o aprendizado em um contexto de fazenda urbana, onde a prática e a teoria se encontram. Um software que gerencie essa experiência pode facilitar a compreensão de conceitos agrícolas e sustentáveis.

**Justificativa para o Desenvolvimento do Software:** Um sistema que centralize informações sobre a operação da fazenda urbana pode otimizar a aprendizagem prática. Para os alunos, isso significa acesso a dados sobre cultivo, manejo e sustentabilidade, além de ferramentas para registrar e analisar suas atividades. Para os professores, o software proporciona uma maneira de acompanhar o progresso dos alunos, gerenciar tarefas e facilitar a comunicação. Essa plataforma não apenas enriquece o aprendizado prático, mas também promove a colaboração e o engajamento.

##### **Usuários (Produtores Urbanos)**

**Importância:** Os produtores são os principais usuários do software, que precisa oferecer ferramentas para planejar, monitorar e otimizar suas atividades agrícolas. Eles dependem de dados precisos para tomar decisões informadas.

**Justificativa para o Desenvolvimento do Software:** O software pode centralizar informações, melhorando a gestão de recursos e aumentando a produtividade. O acompanhamento de dados permitirá que os produtores analisem suas práticas agrícolas, promovendo a melhoria contínua.

##### **Gestores**

**Importância:** Os gestores têm a responsabilidade de supervisionar e implementar iniciativas de agricultura urbana, garantindo que os objetivos sejam atingidos dentro do cronograma e do orçamento.

**Justificativa para o Desenvolvimento do Software:** O software pode oferecer dados e relatórios que ajudem na avaliação do desempenho e no planejamento estratégico, permitindo decisões mais informadas.

### **9.5.2. Planejamento do Projeto**

A fase de planejamento é fundamental para definir claramente o escopo do projeto e os recursos necessários para sua execução.

#### **9.5.2.1. Definição do Escopo do Projeto**

**Objetivo:** Desenvolver um sistema para a gestão de uma fazenda urbana que permita o gerenciamento de insumos, monitoramento de cultivos e vendas para consumidores.

#### **Escopo Inclui:**

- Cadastro de produtos
- Cadastro de funcionários
- Cadastro de fornecedores
- Acompanhamento da produção
- Acompanhamento dos pedidos
- Monitoramento das vendas

#### **9.5.2.2. Planejamento do Gerenciamento do Cronograma**

Figura 1: Atividades a serem desenvolvidas

Atividade	Duração	Dependências	Custo Estimado
Levantamento de Requisitos	2 semanas		R\$2.000,00
Desenho da Arquitetura do Sistema	1 semana	Levantamento de Requisitos	R\$1.500,00
Desenvolvimento do Front-End	3 semanas	Desenho da Arquitetura do Sistema	R\$4.500,00
Desenvolvimento do Back-End	4 semanas	Desenho da Arquitetura do Sistema	R\$6.000,00
Integração e Testes	2 semanas	Desenvolvimento do Front-end	R\$3.000,00
Treinamento dos Usuários	1 semana	Integração e Testes	R\$1.000,00
Lançamento do Software	1 semana	Treinamento dos Usuários	

Fonte: Os Autores.

Figura 2: CPM – Identificar os caminhos críticos

Atividade	Duração	Dependências	Início Mais Precoce	Término Mais Precoce	Início Mais Tardio	Término Mais Tardio	Margem de Manobra	Caminho Crítico
Levantamento de Requisitos	5 dias		0	5	0	5	0	Sim
Desenvolvimento	10 dias	Levantamento de Requisitos	5	15	5	15	0	Sim
Testes	7 dias	Desenvolvimento	15	22	15	22	0	Sim
Revisão	3 dias	Testes	22	25	22	25	0	Sim
Implementação	5 dias	Revisão	25	30	25	30	0	Sim
Documentação	4 dias	Testes	22	26	26	30	4	Não

Fonte: Os Autores.

Figura 3: Papel equipe

Papel	Quantidade
Gerente de Projeto	1
Analista de Negócios	1
Desenvolvedores (Front-end)	2
Desenvolvedores (Back-end)	2
Testador	1
Instrutor de Treinamento	1

Fonte: Os Autores.

### 9.5.3. Definição do Sistema de Comunicação

O sistema de comunicação do projeto será estruturado da seguinte forma:

- **Reuniões Semanais:** Atualizações de status e resolução de problemas.

- **Ferramenta de Gestão de Projetos:** Uso do Trello para acompanhamento das tarefas.
- **Canal de Comunicação:** Grupo no Google Meet para discussões rápidas e troca de informações.

#### 9.5.4. Planejamento do Gerenciamento dos Riscos

Os riscos identificados incluem:

- Atrasos no desenvolvimento devido a mudanças de requisitos.
- Falta de aceitação do usuário final.
- Problemas técnicos durante a integração.

As estratégias de mitigação incluem:

- Reuniões frequentes com stakeholders para alinhar expectativas.
- Coleta de feedback contínuo dos usuários durante o desenvolvimento.
- Planejamento de um período de testes rigorosos antes do lançamento.

#### 9.5.5. Execução do Projeto

A execução do projeto é onde o planejamento é colocado em prática.

##### 9.5.5.1. Sistema de Comunicação Utilizado pelo Grupo

Figura 1: Ferramentas de Comunicação:

Aspecto	Detalhes
Ferramenta	Google Meet
Uso	Comunicação em tempo real, troca de mensagens instantâneas e discussões rápidas
E-mail	Comunicações formais, compartilhamento de documentos e relatórios.
Frequência de Comunicação	Reuniões Semanais: Realizadas toda segunda-feira, 1 hora, para atualização de status do projeto, discussão de desafios e planejamento das atividades.
	Check-ins Diários: Reuniões de 15 minutos ("stand-ups") pela manhã para compartilhamento de progresso e impedimento
	Relatórios Mensais: E-mails com resumo do progresso do projeto, incluindo métricas de desempenho, atrasos e ajustes no cronograma.
Sistema de Trabalho Colaborativo	
Plataforma Utilizada	Trello
Funcionamento do Trabalho	- Quadros de Tarefas: Criados para cada fase do projeto (Levantamento de Requisitos, Desenvolvimento, Testes, etc.
	Listas e Cartões: Tarefas organizadas em listas com cartões que contêm detalhes sobre a tarefa, responsáveis, prazos e comentários
	Comentários e Feedback: Membros da equipe podem adicionar comentários em cada cartão para troca de ideias e sugestões.
Integração com Outras Ferramentas	Google Drive: Compartilhamento de documentos, especificações e materiais de treinamento; versões mantidas para fácil acesso e revisão.
	GitHub: Controle de versão do código, com commits e pull requests para revisão e colaboração eficaz.

Fonte: Os Autores.

### 9.5.6. Monitoramento e Controle

A fase de monitoramento e controle é vital para garantir que o projeto esteja dentro do cronograma e orçamento, além de atender às expectativas de qualidade.

#### 9.5.6.1. Controle do Cronograma

Ferramentas Utilizadas:

- **Trello:** Atualizações diárias no status das tarefas e comentários para acompanhar o progresso.
- **Google Calendar:** Utilizado para agendar reuniões e prazos, garantindo que todos estejam cientes das datas importantes.

Frequência de Revisões:

- **Revisões Semanais:** Durante as reuniões de status, o progresso é revisado e ajustes são feitos conforme necessário. Se uma tarefa está atrasada, a equipe discute soluções para manter o cronograma.

### 9.5.7. Gerenciamento da Qualidade

Padrões de Qualidade:

- **Testes Unitários:** Cada desenvolvedor é responsável por escrever testes para o código que desenvolve, garantindo que funcionalidades individuais funcionem corretamente.
- **Revisões de Código:** Os pull requests no GitHub são revisados por outro membro da equipe antes de serem mesclados, garantindo que o código siga as melhores práticas e padrões de qualidade.

Feedback dos Usuários:

Durante as fases de desenvolvimento e teste, o feedback dos usuários finais é coletado para garantir que o sistema atenda às suas necessidades. Isso é feito por meio de entrevistas e testes de usabilidade.

### 9.5.8. Controle das Comunicações

Registro de Atas de Reuniões:

As atas das reuniões são documentadas e compartilhadas com todos os membros da equipe e partes interessadas, garantindo que todos tenham acesso à mesma informação e decisões tomadas.

Check-ins Semanais:

Durante as reuniões semanais, as discussões são resumidas e as decisões são documentadas, permitindo um acompanhamento claro das evoluções do projeto.

#### **9.5.9. Engajamento das Partes Interessadas**

Reuniões Mensais:

São organizadas reuniões mensais com todas as partes interessadas para atualizações do projeto e coleta de feedback. Essas reuniões são essenciais para garantir que todos estejam alinhados e que suas preocupações sejam ouvidas.

Pesquisas de Satisfação:

Após entregas significativas, são realizadas pesquisas de satisfação com os usuários para coletar informações sobre a usabilidade e funcionalidades do software, permitindo melhorias contínuas.

#### **9.5.10. Encerramento do Projeto**

A fase de encerramento é fundamental para consolidar os resultados do projeto e garantir que todo o conhecimento adquirido seja documentado e compartilhado entre todos os integrantes do grupo.

#### **9.5.11. Documentação**

Todos os documentos do projeto, incluindo planos, relatórios de progresso, atas de reunião e manuais, são organizados em uma pasta central no Google Drive. Essa estrutura facilita o acesso e a consulta futura.

Relatório Final:

Um relatório final é elaborado, resumindo o projeto, incluindo resultados alcançados, métricas de desempenho e lições aprendidas. Este documento é compartilhado com todas as partes interessadas e servirá como base para projetos futuros.



### **9.5.12. Avaliação de Desempenho**

Antes do lançamento do software, é fundamental realizar uma reunião com toda a equipe para discutir o que funcionou bem e o que poderia ser melhorado. Essa reunião de lições aprendidas permite que a equipe compartilhe experiências e identifique áreas de aprimoramento. As lições discutidas são registradas e documentadas para referência futura, garantindo que o conhecimento adquirido seja utilizado em projetos subsequentes.

#### **Análise de KPIs**

Os indicadores-chave de desempenho (KPIs) (métricas quantificáveis que ajudam a monitorar e avaliar o progresso em relação aos objetivos) são revisados para avaliar o sucesso do projeto, considerando aspectos como o tempo de desenvolvimento, a satisfação do usuário e a eficiência do software, que são medidos para garantir que os objetivos do projeto estejam sendo alcançados.

### **9.5.13. Transferência de Conhecimento**

#### **Criação de Manuais:**

Manuais de operação e manutenção são elaborados para facilitar a utilização do software pelos usuários finais e a manutenção pela equipe técnica.

#### **Treinamento da Equipe de Operações:**

Sessões de treinamento são realizadas com a equipe de operações para garantir que todos estejam familiarizados com o uso do software e suas funcionalidades.

#### **Acesso à Documentação:**

A documentação e os manuais de operação são disponibilizados no Google Drive, garantindo que todos os usuários tenham acesso fácil às informações necessárias.

## 9.6 EMPREENDEDORISMO

### Plano de Negócios

Um Plano de Negócios bem estruturado é essencial para guiar um empreendimento e assegurar sua viabilidade no mercado. Para o projeto do sistema de gestão de uma fazenda urbana, os principais elementos do Plano de Negócios incluem:

#### 1. Objetivo Principal (Missão)

A missão do sistema é fornecer uma plataforma integrada para a gestão eficiente de recursos e atividades agrícolas em ambientes urbanos. O foco é promover a sustentabilidade e a otimização de processos, facilitando a vida dos produtores urbanos e contribuindo para o desenvolvimento de práticas agrícolas mais conscientes e eficazes. Ao fazer isso, o sistema busca não apenas aumentar a produtividade, mas também fomentar a conscientização sobre a importância da agricultura sustentável nas cidades.

#### 2. Objetivos de Curto, Médio e Longo Prazo (Visão)

**Curto Prazo:** O objetivo inicial é lançar uma versão beta do software com funcionalidades básicas para o gerenciamento de cultivos e insumos. Isso inclui a implementação de um sistema simples de cadastro de produtos, acompanhamento da produção e gestão de pedidos. O lançamento da versão beta permitirá a coleta de feedbacks valiosos que guiarão as futuras atualizações.

**Médio Prazo:** Após o lançamento da versão beta, o foco será expandir as funcionalidades do sistema. Isso envolverá a inclusão de ferramentas de análise de dados e geração de relatórios que ajudem os produtores a tomar decisões informadas. A meta é atingir um número significativo de usuários e, ao mesmo tempo, estabelecer uma base sólida de clientes satisfeitos.

**Longo Prazo:** O objetivo a longo prazo é tornar-se a principal plataforma de gestão de fazendas urbanas, reconhecida pela inovação e pela eficácia na maximização da produtividade. Isso implica em se tornar uma referência no mercado, com soluções robustas que atendam às necessidades em constante evolução dos produtores urbanos.

#### 3. Público-Alvo

O público-alvo do sistema abrange principalmente produtores urbanos que buscam eficiência na gestão de suas atividades agrícolas. Este grupo pode incluir tanto pequenos agricultores que cultivam em pequenos espaços, como hortas comunitárias, quanto empresas que operam em escala maior. A localização pode incluir áreas urbanas com forte interesse em práticas agrícolas sustentáveis, como regiões metropolitanas em busca de reduzir a pegada ecológica. A caracterização deste público envolve uma diversidade de recursos financeiros e uma crescente demanda por tecnologias que facilitem a agricultura em ambientes urbanos, além de uma disposição para adotar soluções que otimizem seu trabalho.

#### 4. Principais Concorrentes e Estratégias para Vencê-los

Os principais concorrentes podem incluir plataformas já estabelecidas no mercado de gestão agrícola, que oferecem soluções abrangentes para o setor. Para se destacar, algumas estratégias podem ser adotadas:

**Personalização do Software:** Focar em personalizações que atendam necessidades específicas dos produtores urbanos, como integração com serviços de entrega ou suporte para cultivos urbanos em pequena escala.

**Suporte Técnico Eficiente:** Oferecer um suporte técnico contínuo e eficaz, garantindo que os usuários se sintam apoiados e satisfeitos com o uso do sistema.

**Educação e Treinamento:** Prover treinamento e recursos educacionais que ajudem os usuários a maximizar o uso do software, mostrando seu valor agregado e criando uma comunidade em torno do produto.

#### 5. Estratégia de Comercialização

A comercialização do sistema pode ser realizada através de diversos modelos, incluindo:

**Licenciamento:** Oferecer licenças do software a instituições e organizações que desejam implementar a ferramenta em suas operações.

**Venda Direta:** A venda direta do software ao público-alvo, com opções de compra única ou planos de assinatura mensal/anual.

Suporte e Pós-Venda: Além da venda, será fundamental oferecer suporte técnico e serviços de pós-venda, garantindo que os clientes tenham um suporte contínuo, o que ajudará a fidelizar usuários e aumentar a satisfação.

## 9.7 GESTÃO DA QUALIDADE

Neste capítulo, exploraremos a aplicação de uma ferramenta de gestão da qualidade fundamental para o desenvolvimento do sistema de gestão de uma fazenda urbana. A escolha da ferramenta, suas características principais e os impactos significativos na qualidade do sistema serão discutidos em detalhes, enfatizando como essa abordagem é essencial para atender às necessidades específicas dos produtores urbanos.

### Ferramenta de Gestão da Qualidade: Diagrama de Ishikawa

Para o sistema de gestão de uma fazenda urbana, uma ferramenta de gestão da qualidade que se destaca é o Diagrama de Ishikawa, também conhecido como "Diagrama de Causa e Efeito" ou "Espinha de Peixe". Esta ferramenta é crucial para identificar e organizar as causas de problemas que podem afetar a eficiência e a qualidade do software, garantindo que o sistema atenda de forma eficaz às demandas dos usuários.

### Características Principais do Diagrama de Ishikawa

**Visualização Estrutural:** O Diagrama de Ishikawa proporciona uma representação visual clara das relações entre causas e efeitos. No contexto do sistema da fazenda urbana, as "espinhas" podem representar categorias como:

**Métodos de Cultivo:** Processos e práticas de cultivo que podem impactar a produtividade.

**Gestão de Insumos:** Como a comunicação com fornecedores e a organização de pedidos influenciam a eficiência operacional.

**Tecnologia Utilizada:** Ferramentas e plataformas que suportam a operação da fazenda. Essa visualização facilita a identificação de áreas que requerem atenção para melhorar a gestão da fazenda.

**Colaboração da Equipe:** A construção do diagrama é um esforço colaborativo que envolve desenvolvedores, especialistas em agricultura urbana e potenciais usuários. Essa interação promove a troca de ideias e conhecimentos, levando a uma análise mais completa e eficaz das necessidades do sistema.

**Análise Abrangente:** O diagrama permite a consideração de diversos fatores que podem influenciar a qualidade do sistema de gestão da fazenda. Por exemplo, ao mapear causas potenciais de ineficiências na gestão de cultivos, a equipe pode identificar problemas que não seriam evidentes, como a falta de integração entre dados de produção e vendas.

**Foco na Causa Raiz:** O Diagrama de Ishikawa é especialmente eficaz para identificar causas raízes de problemas relacionados à gestão da fazenda. Essa abordagem é vital para implementar soluções que realmente melhorem a qualidade do software, garantindo que ele atenda plenamente às expectativas dos usuários.

### Impactos na Qualidade do Sistema

A aplicação do Diagrama de Ishikawa traz impactos positivos significativos na qualidade do sistema de gestão da fazenda urbana:

**Redução de Erros:** Ao identificar e tratar as causas raízes dos problemas, a equipe pode minimizar a ocorrência de erros no software. Isso resulta em um sistema mais confiável, essencial para a gestão eficiente de atividades agrícolas, aumentando a satisfação dos produtores urbanos.

**Aumento da Produtividade:** A resolução proativa de problemas permite que a equipe se concentre em desenvolver novas funcionalidades, como ferramentas para monitoramento de cultivos e análises de dados. Isso melhora a eficiência do desenvolvimento e acelera a implementação de soluções que atendam às necessidades dos usuários.

**Melhoria Contínua:** A utilização do Diagrama de Ishikawa incentiva uma cultura de melhoria contínua na equipe de desenvolvimento. Ao revisar regularmente as causas de problemas, a equipe é motivada a buscar soluções inovadoras que melhorem a usabilidade e a eficiência do sistema.

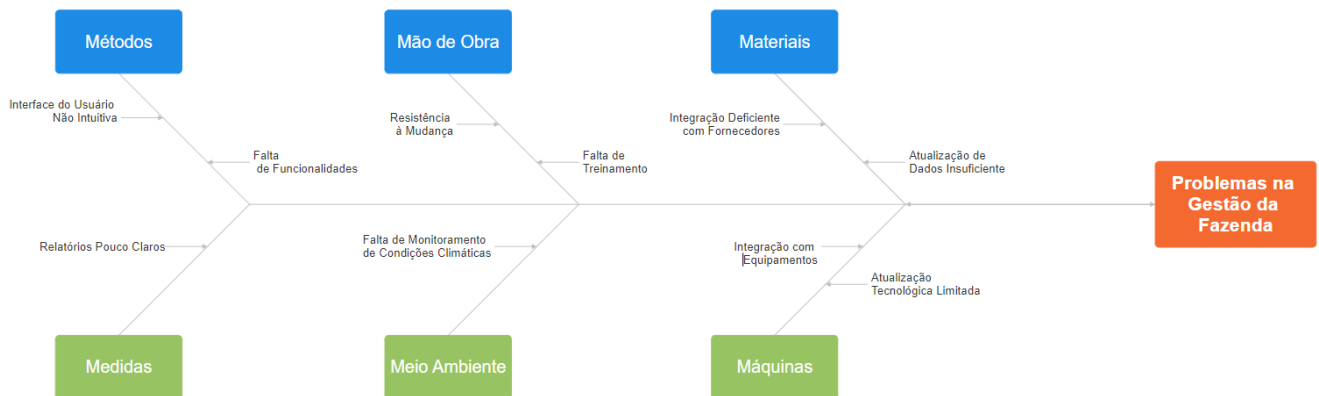
**Satisfação do Cliente:** Com um sistema de maior qualidade e menos falhas, a satisfação dos usuários — os produtores urbanos — tende a aumentar. Isso é crucial para a fidelização e para a construção de uma boa reputação no mercado, resultando em um crescimento da base de usuários do software.

**Documentação e Aprendizado:** O processo de construção do diagrama também serve como documentação valiosa para futuras referências. Ele captura o conhecimento da equipe

sobre as causas de problemas que podem impactar a gestão da fazenda, sendo útil para aprimorar processos em desenvolvimentos futuros.

Abaixo segue um possível Diagrama de Ishikawa relacionado a fazenda.

Figura 1: Diagrama de Ishikawa da UrbAgro



Fonte: Os Autores.

## 10. CONCLUSÃO

Nesse projeto, foi proposto desenvolver três plataformas distintas para os ambientes web, mobile e desktop, visando atender às necessidades para serem feitas a gestão da fazenda urbana de forma eficiente, para isso, decidimos adotar uma abordagem estratégica que considerasse as particularidades de cada ambiente, ao mesmo tempo em que garantisse a integração entre as plataformas e a consistência dos dados. Cada solução foi projetada para oferecer uma ótima experiência eficiente, independentemente do dispositivo utilizado.

Em relação à escolha das tecnologias, optamos por utilizar diferentes linguagens de programação e frameworks, de acordo com as necessidades e limitações de cada plataforma. A integração entre as três plataformas foi uma prioridade, e garantimos que as informações fossem sincronizadas em tempo real, de modo que o usuário pudesse transitar entre as versões sem perda de dados ou funcionalidade. A escolha dessas tecnologias, portanto, não foi apenas uma questão de otimizar o desempenho em cada plataforma, mas também de garantir uma base de código coesa e escalável.

Com isso, as soluções adotadas buscam atender o problema proposto, proporcionando uma infraestrutura sólida. O projeto não apenas resolve as questões atuais, mas também abre portas para futuros aprimoramentos, como a integração de novas funcionalidades e a expansão para outras tecnologias, em busca sempre da melhoria contínua.

Esse projeto serviu como um ponto de partida para a continuidade dos estudos na área e nós como uma equipe, acreditamos que a empresa poderá se beneficiar significativamente das inovações tecnológicas propostas, aumentando sua competitividade e oferecendo uma experiência ainda mais satisfatória para seus usuários.



## 11. REFERÊNCIAS

DENNIS, Alan; WIXOM, Barbara H.; ROTH, Roberta M. **Análise e Projeto de Sistemas**. Grupo GEN, 2014. 978-85-216-2634-3. <<https://integrada.minhabiblioteca.com.br/#/books/978-85-216-2634-3/>>. Acesso em: 06 dez. 2024.

WAZLAWICK, Raul S. **Análise e Design Orientados a Objetos para Sistemas de Informação: Modelagem com UML, OCL e IFML** Grupo GEN, 2014. 9788595153653. <<https://integrada.minhabiblioteca.com.br/#/books/9788595153653/>>. Acesso em: 06 dez. 2024.

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. Grupo GEN, 2014. 9788535226270. <<https://integrada.minhabiblioteca.com.br/#/books/9788535226270/>>. Acesso em: 06 dez. 2024.

PRESSMAN, R. S. **Engenharia de Software**. Porto Alegre: McGraw-Hill, 2016 <<https://integrada.minhabiblioteca.com.br/reader/books/9786558040118>>. Acesso em: 08 dez. 2024.

OLIVEIRA, P. S.; SOUZA, L. F. **Desenvolvimento Orientado a Objetos com C#**. Rio de Janeiro: Editora Campus, 2018.

REIS, M.; LIMA, A. C. **Modelagem de Sistemas e Banco de Dados Relacionais**. São Paulo: Editora Saraiva, 2019.

GOMES, T.; CARVALHO, J. **Projeto de Software: Da Modelagem à Implementação**. Rio de Janeiro: LTC Editora, 2020.

ALENCAR, D. C.; NASCIMENTO, J. A. M. **Desenvolvimento de Sistemas com C# e SQL Server**. 2. ed. São Paulo: Editora Érica, 2017.

GOMES, A. C.; CARVALHO, L. D. **Engenharia de Software: Um Enfoque Prático**. 3. ed. Rio de Janeiro: Elsevier, 2020.

OLIVEIRA, R. R.; SOUZA, P. L. **Modelagem e Implementação de Sistemas: Uma Abordagem Orientada a Objetos**. 1. ed. Porto Alegre: Bookman, 2018.

**C# Orientado a Objetos: Introdução**. Disponível em: <<https://www.devmedia.com.br/csharp-orientado-a-objetos-introducao/29539>>. Acesso em: 08 dez. 2024.

**4 princípios básicos da orientação à objetos com C#**. Disponível em: <<https://marcionizzola.medium.com/4-princ>>. Acesso em: 8 dez. 2024.

Duckett, Jon. **HTML & CSS Design: Design and Build Websites**. Indianapolis, John Wiley & Sons, Inc, 2011.

Haverbeke, Marijn. **Eloquent Javascript, 3rd Edition : A Modern Introduction to Programming**. No Starch Press, 2018.

Meyer, Eric A, and Estelle Weyl. **CSS: The Definitive Guide**. Beijing, China, O'reilly, 2018.

**Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994).** *Padrões de Projeto: Elementos de Software Orientado a Objetos Reutilizáveis*. Editora Bookman.

**Troelsen, A., & Japikse, P. (2017).** *C# 7.0 em uma Casca de Noz: A Referência Definitiva*. O'Reilly Media.

**Documentação do NUnit.** (n.d.). *Documentação do NUnit 3*. Disponível em: <https://nunit.org/>.

**Haverbeke, M. (2014).** *JavaScript Eloquente: Uma Introdução Moderna à Programação*. No Starch Press.

**Larman, C. (2004).** *Utilizando UML e Padrões: Uma Introdução à Análise e Projeto Orientados a Objetos*. Bookman Editora.

**Sommerville, I. (2011).** *Engenharia de Software*. Pearson.

**Fowler, M. (2004).** *Refactoring: Improving the Design of Existing Code*. Addison-Wesley.

## APÊNDICE A – Manual do Usuário da Versão Desktop

Figura 1: Tela de Login.



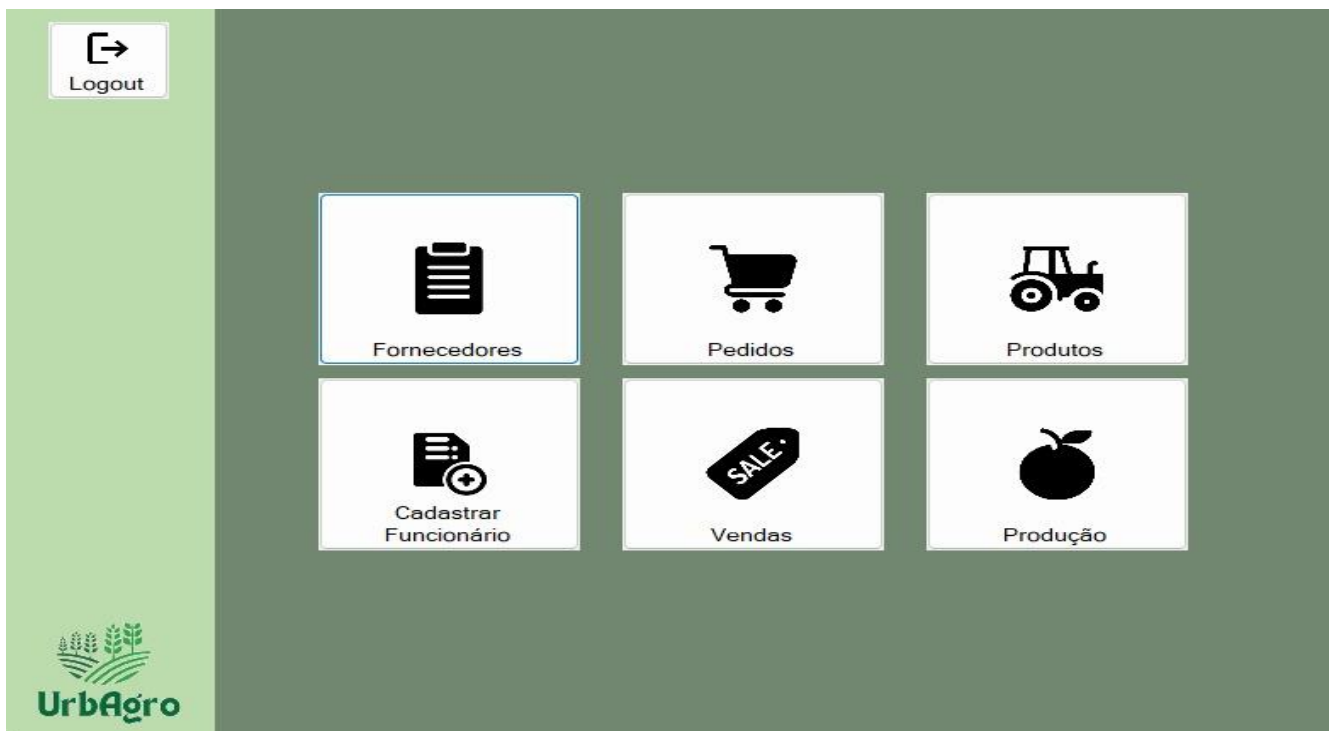
A imagem mostra a interface de login do sistema UrbAgro. No topo, há uma barra verde com o logo 'UrbAgro' (um ícone de plantas verdes) e o nome 'UrbAgro' em verde. Abaixo, no centro, há um formulário cinza com o título 'Login' e a mensagem 'Bem-vindo(a) ao sistema'. O formulário contém dois campos de entrada: 'ID' e 'Senha'. Abaixo dos campos, há dois botões: 'Entrar' e 'Fechar'.

Fonte: Os autores.

A tela de login, representada pela Figura 1, do sistema, conta com 2 botões, o botão “Entrar” que deve ser pressionado após o usuário inserir suas credenciais para fazer login no sistema, e o botão “Fechar”, que fecha o sistema.

Para o primeiro acesso ao sistema, o usuário receberá um login e uma senha. Após inserir esse login, ele deverá clicar na opção desejada.

Figura 2: Tela Principal do Sistema.



Fonte: Os autores.

A Tela Principal do Sistema, representada pela Figura 2, conta com as opções “Fornecedores”, “Pedidos”, “Produtos”, “Cadastrar Funcionário”, “Vendas” e “Produção”.

É para essa tela que o usuário será redirecionado ao fazer login no sistema.

Figura 3: Tela de Fornecedores Cadastrados.

The screenshot shows the 'Fornecedores Cadastrados' (Registered Suppliers) screen. On the left is a green sidebar with a 'Logout' button and the 'UrbAgro' logo. The main area has a title 'Fornecedores Cadastrados' and a search bar with the placeholder 'Digite o CNPJ/CPF do Fornecedor que Deseja Encontrar' and a 'Buscar' button. Below the search bar is a table with the following data:

ID	Nome	Telefone	CNPJ/CPF	Endereço	Email
1	João Sementes	(12) 98888-7777	123.456.789-12	Rua X, Bairro Y	semestes@em.

At the bottom of the main area are three buttons: 'Voltar' (with a left arrow), 'Cadastrar Novo Fornecedor', and 'Inativar Fornecedor'.

Fonte: Os autores.

Ao clicar no botão “Fornecedores”, o usuário será redirecionado para a tela de Fornecedores Cadastrados, representada pela Figura 3, onde todos os fornecedores cadastrados no sistema são exibidos.

A tela de “Fornecedores” conta com os seguintes botões “Logout” que sai do sistema após a confirmação do usuário, “Voltar” que volta para a tela inicial, “Cadastrar Novo Fornecedor”, “Inativar Fornecedor” e “Buscar” que, ao inserir o CPF ou CNPJ do fornecedor que deseja encontrar, exibe as informações do mesmo.

Figura 4: Tela de Cadastro de Fornecedor.

Cadastro de Fornecedor		
Nome do Fornecedor João Sementes	CEP 12345678	Telefone (12) 98888-7777
CPF/CNPJ 123.456.789-12	Endereço Rua X, Bairro Y	E-mail sementes@email.com
Situação do Fornecedor Ativo	Complemento Bloco B	Representante João
ID 1	Cidade São José dos Campos	Matéria-Prima Sementes
Razão Social JoMentes L.T.D.A	Estado São Paulo	

[Voltar](#) [Cadastrar](#)

Fonte: Os autores.

Ao clicar no botão “Cadastrar Novo Fornecedor”, presente na Figura 3, o usuário será redirecionado para a tela de Cadastro de Fornecedor, representada pela Figura 4.

Para realizar um novo cadastro, o usuário deve preencher as informações essenciais requisitadas na tela de cadastro. Após preencher todas as informações, o usuário deverá pressionar o botão “Cadastrar”, e, ao pressioná-lo, deverá confirmar ou não o cadastro.

Figura 5: Inativar Fornecedor.

**Inativação de Fornecedor**

Digite o CNPJ/CPF do Fornecedor que Deseja Inativar

1

Inativar

ID	Nome	Telefone	CNPJ/CPF	Endereço	Email
1	João Sementes	(12) 98888-7777	123.456.789-12	Rua X, Bairro Y	sementes@em.

Sucesso

Fornecedor inativado com sucesso!

OK

Voltar

Fonte: Os autores.

Para inativar um fornecedor, é preciso que o usuário vá até a tela de “Fornecedores Cadastrados”, representada pela Figura 3, e pressione o botão “Inativar Fornecedor”. Após isso, ele será redirecionado para a tela de “Inativar Fornecedor”, representada pela Figura 5, onde ele deverá informar o CPF ou CNPJ do fornecedor que deseja inativar.

Figura 6: Pedidos Cadastrados.

ID	Nome	Quantidade	Valor Unitário	Empresa Responsável pe
1	Batata	50	590	Mercado
2	Brócolis	50	15,90	Tauste
3	Tomate	25	0,99	Piratiniga

Fonte: Os autores.

Na Tela Inicial do Sistema, representada pela Figura 2, caso o usuário pressione o botão de “Pedidos”, ele será redirecionado para a tela de “Pedidos Cadastrados”, onde ele possui as seguintes opções: “Logout”, “Voltar”, “Buscar”, “Cadastrar Novo Pedido”, “Editar um Pedido” e “Excluir um Pedido”.



Figura 7: Cadastro de Pedidos.

The screenshot displays a web interface for registering a new order. On the left, a green sidebar contains a 'Logout' button with a door icon and the 'UrbAgro' logo at the bottom. The main area has a dark green background and is titled 'Cadastro de Pedido'. It features a 'Home' button with a house icon in the top right. The form includes several input fields: 'Nome do Produto' (filled with 'Tomate'), 'ID do Pedido' (filled with '5'), 'Quantidade' (filled with '25'), 'Valor Unitário' (filled with '0,99'), and 'Empresa Responsável pela Compra' (filled with 'Piratinunga'). At the bottom, there are two buttons: 'Voltar' with a left arrow and 'Cadastrar'.

Fonte: Os autores.

Caso o usuário selecione a opção de “Cadastrar Novo Pedido”, ele será redirecionado para a tela de “Cadastro de Pedido”, representada pela Figura 7.

Para cadastrar um novo pedido, o usuário deverá preencher todas as informações necessárias e clicar no botão “Cadastrar”. Ao clicar no botão, uma confirmação será exibida, onde ele deverá ou não confirmar o cadastro do pedido.

Figura 8: Editar Pedido.

The screenshot displays the 'Editar Pedido' (Edit Order) form within the UrbAgro application. The interface features a green sidebar on the left with a 'Logout' button and the 'UrbAgro' logo. The main content area has a dark green background. At the top right, there is a 'Home' button. The form itself is centered and contains several input fields with labels above them: 'Digite o ID do Pedido que Deseja Editar' (Enter the ID of the order you want to edit) with the value '2', 'Nome do Produto' (Product Name) with 'Brócolis', 'ID do Pedido' (Order ID) with '2', 'Quantidade' (Quantity) with '50', 'Valor Unitário' (Unit Value) with '8,99', and 'Empresa Responsável pela Compra' (Company responsible for the purchase) with 'Tauste'. A 'Buscar' (Search) button is located to the right of the first input field. At the bottom of the form, there are two buttons: 'Voltar' (Back) with a left arrow and 'Salvar Informações' (Save Information).

Fonte: Os autores.

Caso o usuário selecione a opção de “Editar Um Pedido”, ele será redirecionado para a tela de “Editar Pedido”, representada pela Figura 8.

Para editar as informações de um pedido, o usuário deverá digitar o ID do pedido em que ele deseja editar e clicar em “Buscar”, após isso, ele poderá editar as informações necessárias e clicar em “Salvar Informações”.

Figura 9: Excluir Pedido.

Logout

Home

Excluir Pedido

Digite o ID do Pedido que Deseja Excluir

2

Excluir

ID	Nome	Quantidade	Valor Unitário	Empresa Responsável pe
2	Brócolis	50	8,99	Tauste

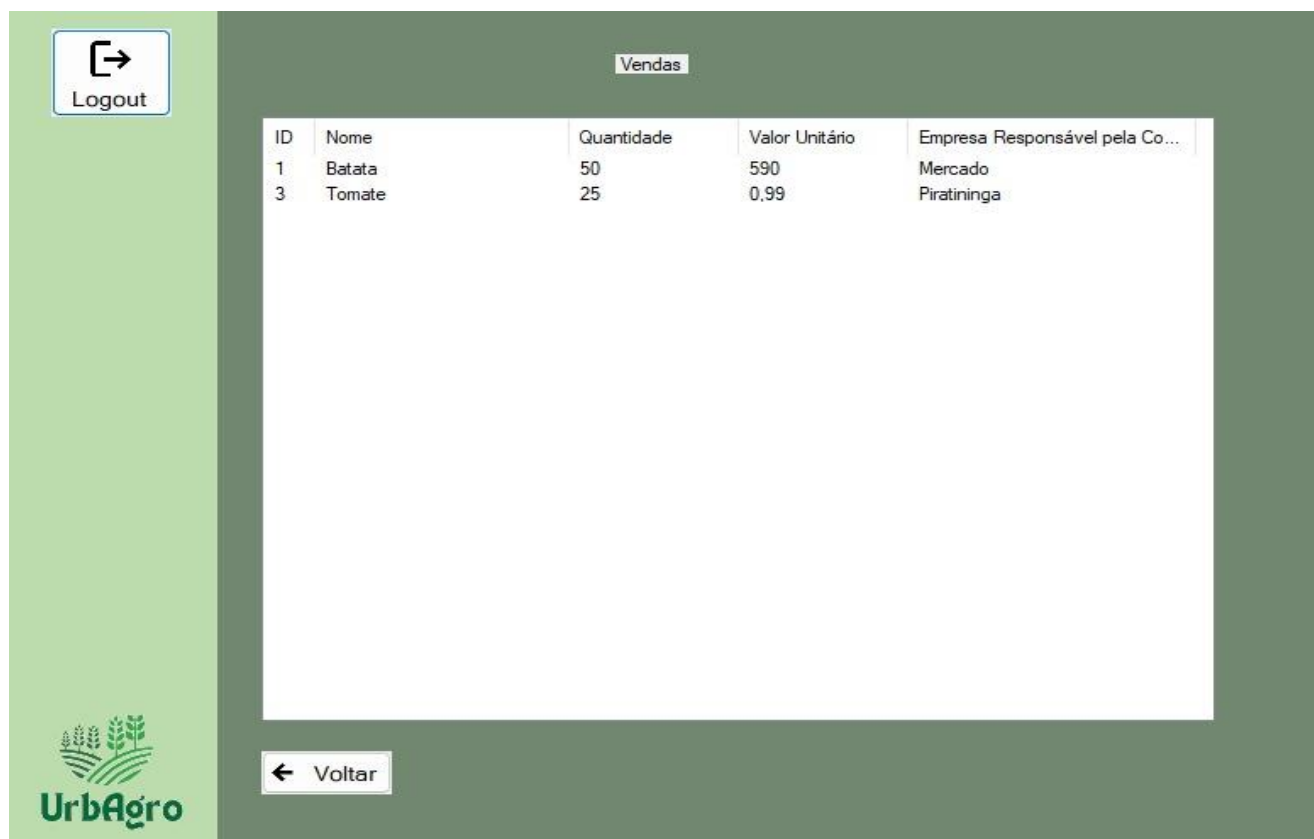
Voltar

Fonte: Os autores.

Caso o usuário selecione a opção de “Excluir Um Pedido”, ele será redirecionado para a tela de “Excluir Pedido”, representada pela Figura 9.

Para excluir um pedido, o usuário deverá digitar o ID do pedido em que ele deseja excluir e clicar em “Excluir”. Após isso, o pedido com o respectivo ID será excluído.

Figura 10: Vendas.



ID	Nome	Quantidade	Valor Unitário	Empresa Responsável pela Co...
1	Batata	50	590	Mercado
3	Tomate	25	0,99	Piratininga

Fonte: Os autores.

Na Tela Inicial do Sistema, representada pela Figura 2, caso o usuário pressione o botão de “Vendas”, ele será redirecionado para a tela de “Vendas”, representada pela Figura 10, onde ele poderá verificar todas as informações de todos os pedidos existentes no sistema.

Nesta tela, caso um pedido seja editado ou excluído, as informações serão automaticamente alteradas.

Figura 11: Produtos Cadastrados.

Produtos Cadastrados

Digite o ID do Produto que Deseja Encontrar

Buscar

ID	Nome	Quantidade	Valor Unitário	Empresa Responsável pe
1	Pêssego	75	6,99	Tauste

Voltar

Cadastrar Novo Produto

Excluir um Produto

Logout

UrbAgro

Fonte: Os autores.

Na Tela Inicial do Sistema, representada pela Figura 2, caso o usuário pressione o botão de “Produtos”, ele será redirecionado para a tela de “Produtos Cadastrados”, representada pela Figura 11, onde é possível consultar as informações de todos os produtos cadastrados no sistema.

Na tela de “Produtos Cadastrados”, existem as opções “Cadastrar Novo Produto” e “Excluir Produto”.

Figura 12: Cadastro de Produto.

The screenshot displays the 'Cadastro de Produto' (Product Registration) form within the UrbAgro application. The interface features a green sidebar on the left with a 'Logout' button and the 'UrbAgro' logo. The main content area has a dark green background and includes a 'Home' button in the top right corner. The form itself is centered and contains several input fields with labels: 'Nome do Produto' (filled with 'Pêssego'), 'ID do Produto' (filled with '1'), 'Quantidade' (filled with '75'), 'Valor Unitário' (filled with '6,99'), and 'Empresa Responsável pela Compra' (filled with 'Taustel'). At the bottom of the form, there are two buttons: 'Voltar' (Back) on the left and 'Cadastrar' (Register) on the right.

Fonte: Os autores.

Caso o usuário selecione a opção de “Cadastrar Novo Produto”, ele será redirecionado para a tela de “Cadastro de Produto”, representada pela Figura 12.

Para cadastrar um produto, o usuário deverá digitar as informações do produto em que ele deseja cadastrar e clicar em “Cadastrar”. Após isso, uma verificação ocorrerá e o produto será cadastrado.

Figura 13: Excluir Produto.

Excluir Produto

Digite o ID do Produto que Deseja Excluir

1

Excluir

ID	Nome	Quantidade	Valor Unitário	Empresa Responsável pe
1	Pêssego	75	6,99	Tauste

Voltar

Fonte: Os autores.

Caso o usuário selecione a opção de “Excluir Um Produto”, ele será redirecionado para a tela de “Excluir Produto”, representada pela Figura 13.

Para excluir um produto, o usuário deverá digitar o ID do produto em que ele deseja excluir e clicar em “Excluir”. Após isso, uma verificação ocorrerá e o produto será excluído.

Figura 14: Funcionários Cadastrados.

ID	Nome	Email	Senha
1	func	func@example.com	1234
2	gerente	gerente@example.com	1234
3	ti	ti@example.com	1234

Fonte: Os autores.

Na tela inicial do sistema, representada pela Figura 2, caso o usuário selecione a opção “Cadastrar Funcionário”, ele será redirecionado para a tela de “Funcionários Cadastrados”, onde é possível verificar todos os funcionários cadastrados no sistema.

Na tela de “Funcionários Cadastrados”, existem as opções de “Cadastrar Novo Funcionário” e “Excluir um Funcionário”.

Importante: Somente usuários com hierarquia de T.I possuem acesso a tela de “Funcionários Cadastrados”.



Figura 15: Cadastro de Funcionário.

Logout

Home

Cadastro de Funcionário

Nome do Funcionário  
Bruno

ID do Funcionário  
4

Email  
bruno@email.com

Senha  
1234

Voltar

Cadastrar

UrbAgro

Fonte: Os autores.

Caso o usuário selecione a opção de “Cadastrar Novo Funcionário”, ele será redirecionado para a tela de “Cadastro de Funcionário”, representada pela Figura 15.

Para cadastrar um funcionário, o usuário deverá digitar as informações do funcionário em que ele deseja cadastrar e clicar em “Cadastrar”. Após isso, uma verificação ocorrerá e o funcionário será cadastrado. Após o cadastro, o novo funcionário poderá utilizar o E-mail e Senha cadastrados para acessar o sistema.

Figura 16: Excluir Funcionário.

Logout

Excluir Funcionário

Digite o ID do Funcionário que Deseja Excluir

4

Excluir

ID	Nome	Email	Senha
4	Bruno	bruno@email.com	1234

Voltar

Home

UrbAgro

Fonte: Os autores.

Caso o usuário selecione a opção de “Excluir Um Funcionário”, ele será redirecionado para a tela de “Excluir Funcionário”, representada pela Figura 16.

Para excluir um funcionário, o usuário deverá digitar o ID do funcionário em que ele deseja excluir e clicar em “Excluir”. Após isso, uma verificação ocorrerá e o funcionário será excluído. Após a exclusão, o funcionário não conseguirá mais acessar o sistema com seu E-mail e Senha.

Figura 17: Produções Cadastradas.

ID	Produto	Quantidade	Data da Produção	Responsável pela Produção
1	Abacaxi	12345	31/10/2024	João

Fonte: Os autores.

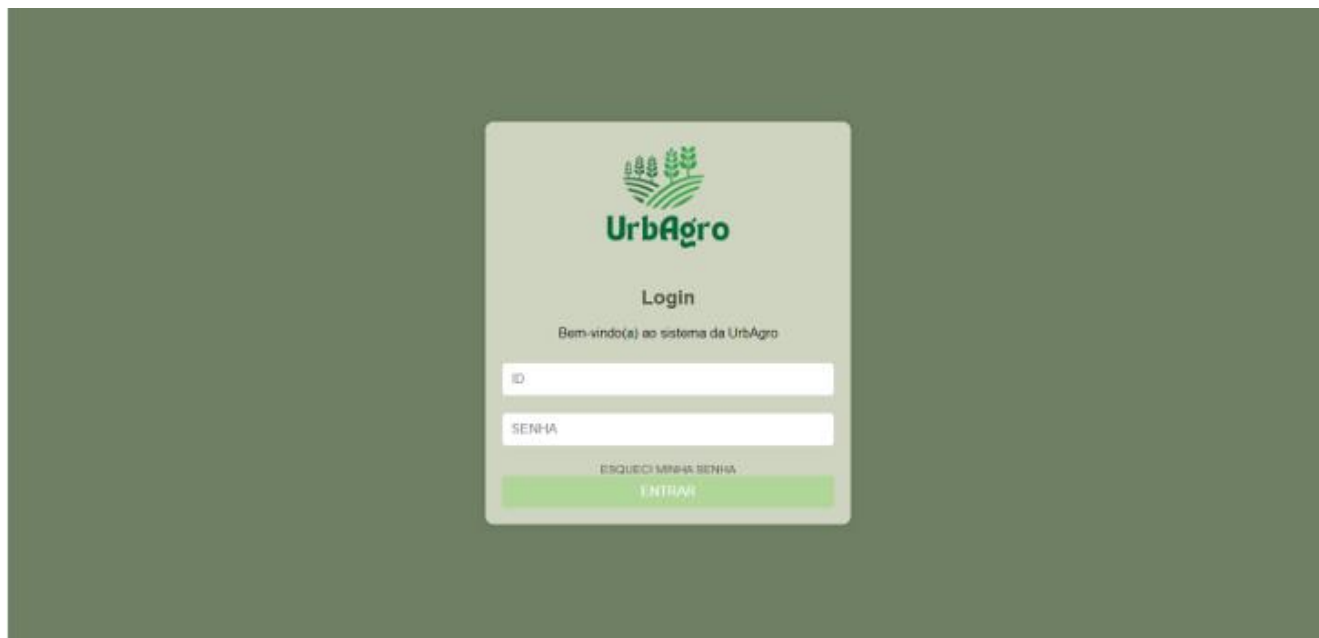
Para cadastrar uma nova produção, o usuário deve preencher os campos obrigatórios e clicar em “Cadastrar”. Após isso, uma mensagem de confirmação será exibida, informando que os dados foram salvos com sucesso.

Para Editar uma Produção, o usuário deverá clicar em “Editar uma Produção”, informar o ID da produção que deseja editar, editar as informações e, editar ou não as informações e clicar em “Salvar”.

Para Excluir uma Produção, o usuário deverá clicar em “Excluir uma Produção”, informar o ID da produção que deseja excluir e clicar em “Excluir” para confirmar a exclusão.

## APÊNDICE B – Manual do Usuário da Versão Web

Figura 1: Tela de Login.

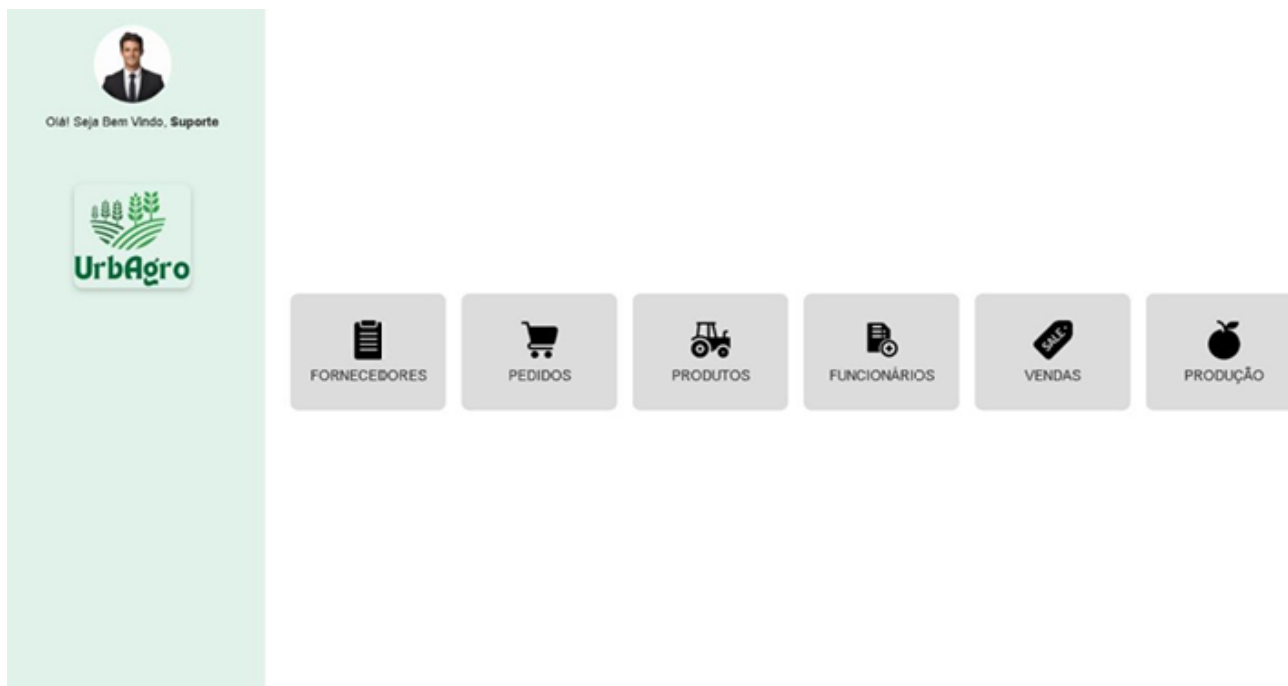


Fonte: Os autores.

A tela de login, representada pela Figura 1, do sistema, conta com 2 botões, o botão “Entrar” que deve ser pressionado após o usuário inserir suas credenciais para fazer login no sistema, e o botão “Esqueci Minha Senha”, que, caso o usuário esqueça a senha, informa-o que ele deve entrar em contato com o suporte.

Para o primeiro acesso ao sistema, o usuário receberá um login e uma senha. Após inserir esse login, ele deverá clicar na opção desejada.

Figura 2: Tela Principal do Sistema.



Fonte: Os autores.

A Tela Principal do Sistema, representada pela Figura 2, conta com as opções “Fornecedores”, “Pedidos”, “Produtos”, “Cadastrar Funcionário”, “Vendas” e “Produção”.

É para essa tela que o usuário será redirecionado ao fazer login no sistema.

É possível clicar na foto do usuário presente no canto superior esquerdo para acessar o menu do usuário, onde é possível realizar o logout do sistema após uma confirmação.

Figura 3: Tela de Fornecedores Cadastrados.

UrbAgro

Digite o CNPJ/CPF do Fornecedor

**Fornecedor 1**  
**Nome:** Guilherme  
**Telefone:** 12997542141  
**CNPJ/CPF:** 111.111.111.11  
**Endereço:** Rua crater, 23  
**Email:** Guilherme\_\_Silva123@gmail.com  
**Cidade:** São José dos Campos  
**Estado:** São Paulo  
**Representante:**  
**Matéria-prima:** Semente de tomate cereja

← Voltar CADASTRAR NOVO FORNECEDOR INATIVAÇÃO DE FORNECEDOR

LISTA FORNECEDORES INATIVADOS

Fonte: Os autores.

Ao clicar no botão “Fornecedores”, o usuário será redirecionado para a tela de Fornecedores Cadastrados, representada pela Figura 3, onde todos os fornecedores cadastrados no sistema são exibidos.

A tela de “Fornecedores” conta com os seguintes botões “Logout” que sai do sistema após a confirmação do usuário, “Voltar” que volta para a tela inicial, “Cadastrar Novo Fornecedor”, “Inativar Fornecedor” e “Buscar” que, ao inserir o CPF ou CNPJ do fornecedor que deseja encontrar, exibe as informações do mesmo.

Figura 4: Tela de Cadastro de Fornecedor.

**Cadastro de Fornecedores**

**Nome**  
Até 50 caracteres

**CEP**  
Formato: 00000-000

**Telefone**  
Formato: (00) 00000-0000

**CNPJ/CPF**  
Até 18 caracteres

**Endereço**  
Até 100 caracteres

**E-mail**  
Até 50 caracteres

**Situação do Fornecedor**  
Até 20 caracteres

**Complemento**  
Até 50 caracteres

**Representante**  
Até 50 caracteres

**Código**  
Até 2 caracteres

**Cidade**  
Até 50 caracteres

**Matéria-Prima**  
Até 50 caracteres

**Razão Social**  
Até 50 caracteres

**Estado**  
Ex: SP

[← Voltar](#) [CADASTRAR](#)

Fonte: Os autores.

Ao clicar no botão “Cadastrar Novo Fornecedor”, presente na Figura 3, o usuário será redirecionado para a tela de Cadastro de Fornecedor, representada pela Figura 4.

Para realizar um novo cadastro, o usuário deve preencher as informações essenciais requisitadas na tela de cadastro. Após preencher todas as informações, o usuário deverá pressionar o botão “Cadastrar”, e, ao pressiona-lo, deverá confirmar ou não o cadastro.

Figura 5: Inativar Fornecedor.

Essa página diz

Informe o ID do fornecedor que deseja inativar:

OK Cancelar

**Fornecedor 1**

**Nome:** Gabriel Igor

**CEP:** 11.111-11

**Telefone:** (12) 98133-9294

**CNPJ/CPF:** 122122222222

**Endereço:** Rua X, Bairro X, Número X

**Email:** teste@email.com

**Situação:** ativo

**Complemento:** bloco b

**Representante:** pablo

**Código:** 123

**Cidade:** Jacareí

**Estado:** Sp

**Matéria-prima:** Alfaca

**Razão Social:** Empresa X

[← Voltar](#) [CADASTRAR NOVO FORNECEDOR](#) [INATIVAÇÃO DE FORNECEDOR](#) [LISTA FORNECEDORES INATIVADOS](#)

Fonte: Os autores.

Para inativar um fornecedor, é preciso que o usuário vá até a tela de “Fornecedores Cadastrados”, representada pela Figura 3, e pressione o botão “Inativar Fornecedor”. Após isso, ele será redirecionado para a tela de “Inativar Fornecedor”, representada pela Figura 5, onde ele deverá informar o ID do fornecedor que deseja inativar.



Figura 6: Pedidos Cadastrados.



Fonte: Os autores.

Na Tela Inicial do Sistema, representada pela Figura 2, caso o usuário pressione o botão de “Pedidos”, ele será redirecionado para a tela de “Pedidos Cadastrados”, onde ele possui as seguintes opções: “Logout”, “Voltar”, “Buscar”, “Cadastrar Novo Pedido”, “Editar um Pedido” e “Excluir um Pedido”.

Figura 7: Cadastro de Pedidos.



**Novo Pedido**

**Nome do Produto**  
Até 50 caracteres

**ID do Pedido**  
Máx. 2 caracteres

**Quantidade**  
Ex: 1000

**Valor Unitário**  
Ex: 1500.00

**Empresa Responsável pela Compra**  
Até 50 caracteres

[← Voltar](#) [ADICIONAR](#)

Fonte: Os autores.

Caso o usuário selecione a opção de “Cadastrar Novo Pedido”, ele será redirecionado para a tela de “Cadastro de Pedido”, representada pela Figura 7.

Para cadastrar um novo pedido, o usuário deverá preencher todas as informações necessárias e clicar no botão “Cadastrar”. Ao clicar no botão, uma confirmação será exibida, onde ele deverá ou não confirmar o cadastro do pedido.

Figura 8: Editar Pedido.

**Lista de Pedidos**

ID do Pedido	Nome do Produto	Valor Unitário: R\$	Quantidade	Empresa Responsável	Ações
123	Alface	2.69	500	Hortifrut Oriente	<a href="#">Editar</a>

**Editar Pedido**

Nome do Produto:  Quantidade:  Empresa Responsável:  Valor Unitário:  [Salvar](#) [Fechar](#)

ID do Pedido	Nome do Produto	Valor Unitário: R\$	Quantidade	Empresa Responsável	Ações
443	Couve	2.80	2000	Hortifrut Oriente	<a href="#">Editar</a>

[CADASTRAR NOVO PEDIDO](#) [Voltar](#) [Excluir Pedido](#)

Fonte: Os autores.

Caso o usuário selecione a opção de “Editar Um Pedido”, ele será redirecionado para a tela de “Editar Pedido”, representada pela Figura 8.

Para editar as informações de um pedido, o usuário deverá digitar o ID do pedido em que ele deseja editar e clicar em “Buscar”, após isso, ele poderá editar as informações necessárias e clicar em “Salvar Informações”.

Figura 9: Excluir Pedido.

Essa página diz

Digite o ID do pedido que deseja excluir:

OK Cancelar

**Lista de Pedidos**

ID do Pedido: 1 Nome do Produto: cebola Valor Unitário: R\$ 22 Quantidade: 232 Empresa Responsável: x [Editar](#)

CADASTRAR NOVO PEDIDO [← Voltar](#) [Excluir Pedido](#)

**UrbAgro**

Fonte: Os autores.

Caso o usuário selecione a opção de “Excluir Um Pedido”, ele será redirecionado para a tela de “Excluir Pedido”, representada pela Figura 9.

Para excluir um pedido, o usuário deverá digitar o ID do pedido em que ele deseja excluir e clicar em “Excluir”. Após isso, o pedido com o respectivo ID será excluído.

Figura 10: Vendas.




### Vendas

ID do Pedido	Data	Produto	Quantidade	Valor Unitário	Total	Empresa Responsável
123	10/11/2024	Alface	500	R\$ 2.69	R\$ 1345.00	Hortifrut Oriente
321	10/11/2024	Cebola	1000	R\$ 6.62	R\$ 6620.00	Seamar Supermercados
443	10/11/2024	Couve	2000	R\$ 2.80	R\$ 5600.00	Hortifrut Oriente
6789	10/11/2024	Rabanete	2500	R\$ 6.45	R\$ 16125.00	Hortifrut Morumbi
0887	10/11/2024	Quiabo	3000	R\$ 0.75	R\$ 2250.00	Shibata Supermercados
8800	10/11/2024	Brócolis	1200	R\$ 8.00	R\$ 9600.00	Shibata Supermercados
578	10/11/2024	Feijão	1000	R\$ 0.15	R\$ 150.00	Conde Atacadista

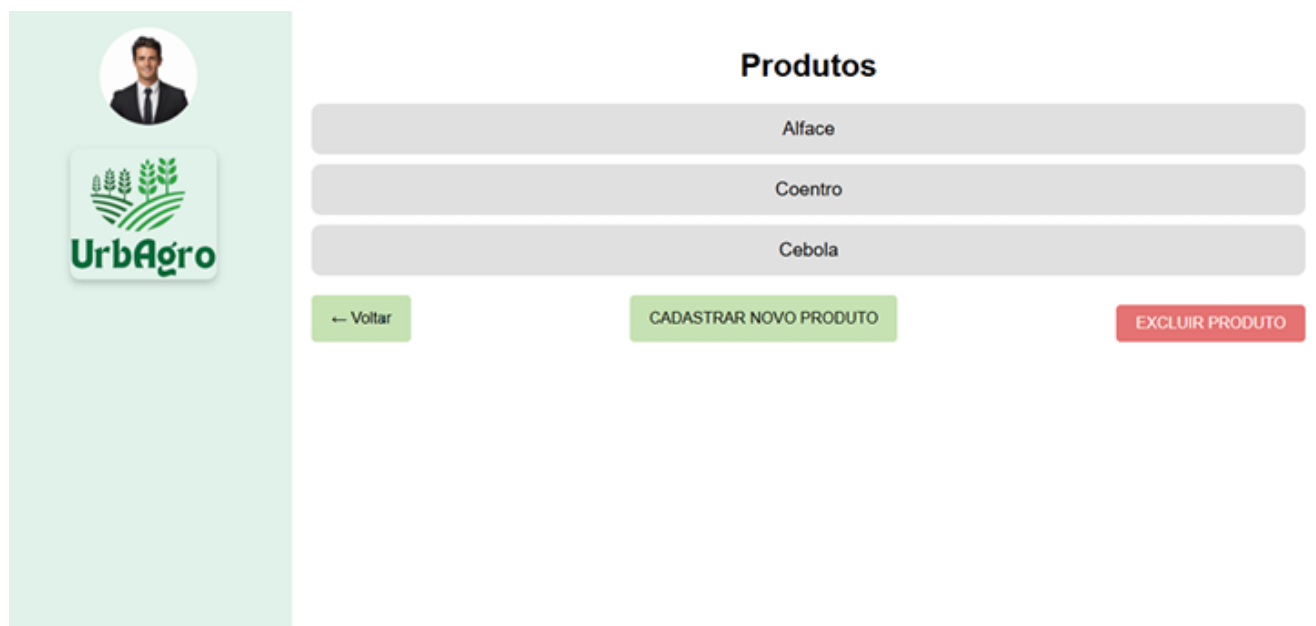
Voltar

Fonte: Os autores.

Na Tela Inicial do Sistema, representada pela Figura 2, caso o usuário pressione o botão de “Vendas”, ele será redirecionado para a tela de “Vendas”, representada pela Figura 10, onde ele poderá verificar todas as informações de todos os pedidos existentes no sistema.

Nesta tela, caso um pedido seja editado ou excluído, as informações serão automaticamente alteradas.

Figura 11: Produtos Cadastrados.

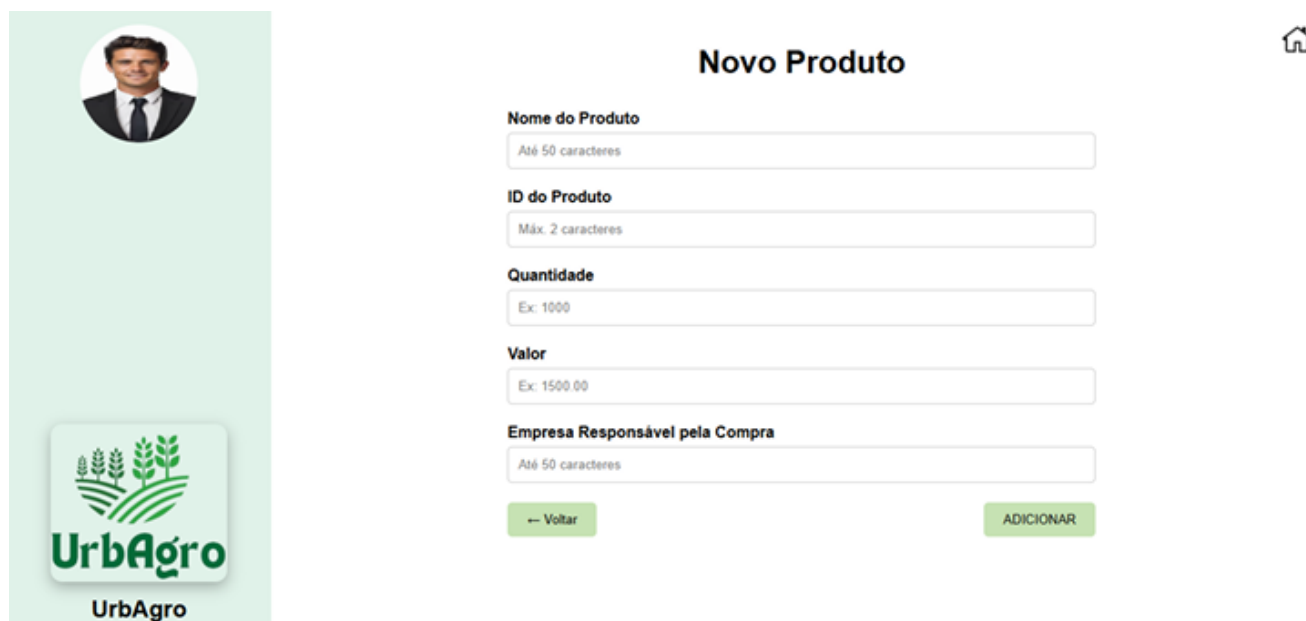


Fonte: Os autores.

Na Tela Inicial do Sistema, representada pela Figura 2, caso o usuário pressione o botão de “Produtos”, ele será redirecionado para a tela de “Produtos Cadastrados”, representada pela Figura 11, onde é possível consultar as informações de todos os produtos cadastrados no sistema.

Na tela de “Produtos Cadastrados”, existem as opções “Cadastrar Novo Produto” e “Excluir Produto”.

Figura 12: Cadastro de Produto.



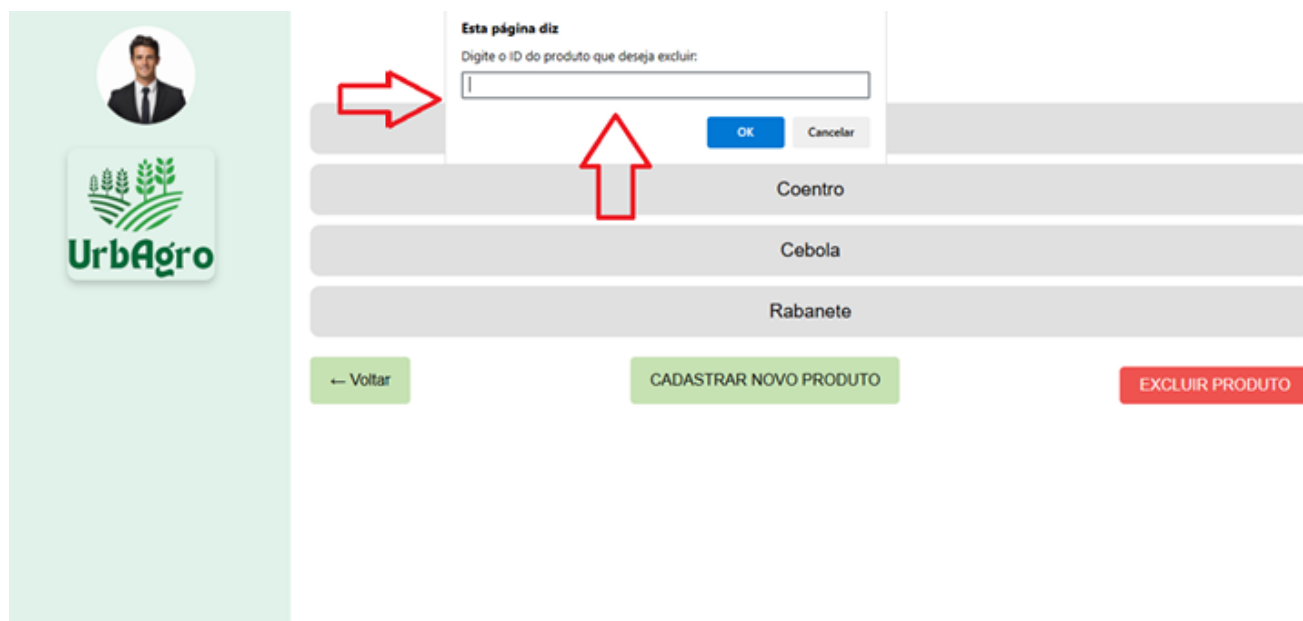
The screenshot displays the 'Novo Produto' (New Product) registration interface. On the left, there is a user profile section with a circular profile picture of a man and the 'UrbAgro' logo, which features a green plant icon and the text 'UrbAgro'. The main form area is titled 'Novo Produto' and contains several input fields: 'Nome do Produto' (Product Name) with a placeholder 'Até 50 caracteres', 'ID do Produto' (Product ID) with a placeholder 'Máx. 2 caracteres', 'Quantidade' (Quantity) with a placeholder 'Ex: 1000', 'Valor' (Value) with a placeholder 'Ex: 1500.00', and 'Empresa Responsável pela Compra' (Company Responsible for Purchase) with a placeholder 'Até 50 caracteres'. At the bottom of the form, there are two green buttons: '← Voltar' (Back) and 'ADICIONAR' (Add). A home icon is visible in the top right corner of the interface.

Fonte: Os autores.

Caso o usuário selecione a opção de “Cadastrar Novo Produto”, ele será redirecionado para a tela de “Cadastro de Produto”, representada pela Figura 12.

Para cadastrar um produto, o usuário deverá digitar as informações do produto em que ele deseja cadastrar e clicar em “Cadastrar”. Após isso, uma verificação ocorrerá e o produto será cadastrado.

Figura 13: Excluir Produto.



Esta página diz

Digite o ID do produto que deseja excluir:

OK Cancelar

Coentro

Cebola

Rabanete

← Voltar CADASTRAR NOVO PRODUTO EXCLUIR PRODUTO

Fonte: Os autores.

Caso o usuário selecione a opção de “Excluir Um Produto”, ele será redirecionado para a tela de “Excluir Produto”, representada pela Figura 13.

Para excluir um produto, o usuário deverá digitar o ID do produto em que ele deseja excluir e clicar em “Excluir”. Após isso, uma verificação ocorrerá e o produto será excluído.



Figura 14: Cadastro de Funcionário.

The screenshot shows a web interface for registering a new employee. On the left, there is a sidebar with a profile picture of a man in a suit and the 'UrbAgro' logo. The main area is titled 'Cadastrar Funcionário' and contains four input fields: 'Nome Funcionário' (filled with 'Guilherme Bordinhon'), 'Id\_Funcionário' (filled with '03'), 'E-mail' (filled with 'Guilherme\_Silva123@gmail.com'), and 'Senha' (masked with '\*\*\*'). At the bottom left of the form is a green button labeled 'Voltar' with a back arrow, and at the bottom right is a green button labeled 'Salvar'. A home icon is visible in the top right corner of the interface.

Fonte: Os autores.

Caso o usuário selecione a opção de “Cadastrar Novo Funcionário”, ele será redirecionado para a tela de “Cadastro de Funcionário”, representada pela Figura 14.

Para cadastrar um funcionário, o usuário deverá digitar as informações do funcionário em que ele deseja cadastrar e clicar em “Cadastrar”. Após isso, uma verificação ocorrerá e o funcionário será cadastrado. Após o cadastro, o novo funcionário poderá utilizar o E-mail e Senha cadastrados para acessar o sistema.

Figura 15: Excluir Funcionário.

Esta página diz  
Digite o ID do funcionário que deseja excluir:

OK Cancelar

**Dados Funcionários**

Maria Aparecida

Guilherme Bordignon

Mateus Jesus

CADASTRAR FUNCIONÁRIO

EXCLUIR FUNCIONÁRIO

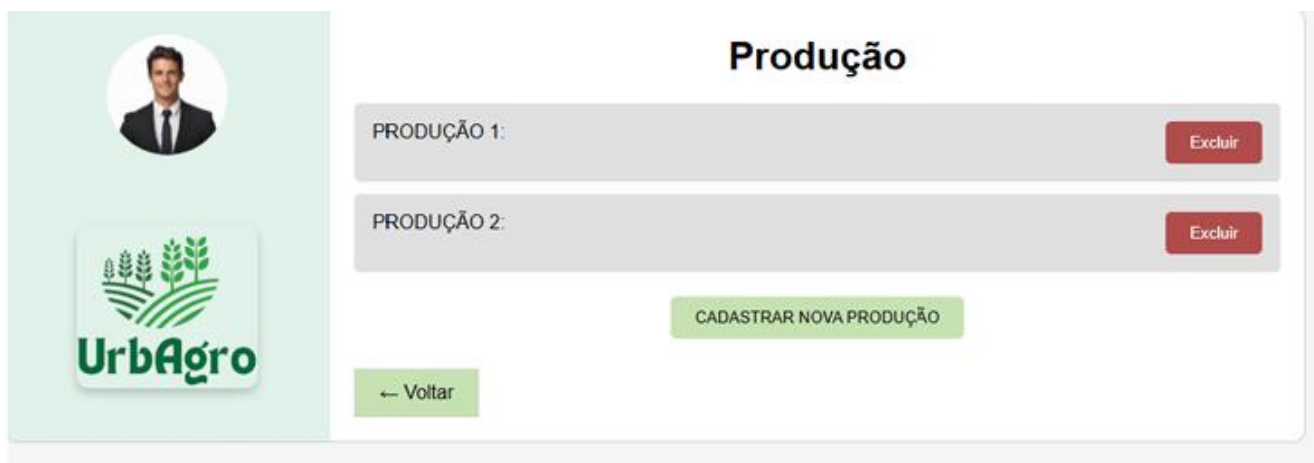
← Voltar

Fonte: Os autores.

Caso o usuário selecione a opção de “Excluir Um Funcionário”, ele será redirecionado para a tela de “Excluir Funcionário”, representada pela Figura 15.

Para excluir um funcionário, o usuário deverá digitar o ID do funcionário em que ele deseja excluir e clicar em “Excluir”. Após isso, uma verificação ocorrerá e o funcionário será excluído. Após a exclusão, o funcionário não conseguirá mais acessar o sistema com seu E-mail e Senha.

Figura 16: Produções Cadastradas.



Fonte: Os autores.

Para cadastrar uma nova produção, o usuário deve preencher os campos obrigatórios e clicar em “Cadastrar”. Após isso, uma mensagem de confirmação será exibida, informando que os dados foram salvos com sucesso.

Para Editar uma Produção, o usuário deverá clicar em “Editar uma Produção”, informar o ID da produção que deseja editar, editar as informações e, editar ou não as informações e clicar em “Salvar”.

Para Excluir uma Produção, o usuário deverá clicar em “Excluir uma Produção”, informar o ID da produção que deseja excluir e clicar em “Excluir” para confirmar a exclusão.

## FICHA DE CONTROLE DO PIM

Grupo Nº \_\_\_\_\_ Ano: 2024 Período: \_\_\_\_\_ Orientador: \_\_\_\_\_

Tema: Sistema Integrado para Controle de uma Fazenda Urbana.

Alunos:

RA	Nome	E-mail	Curso	Visto do aluno
T953EE5	Bruno Siqueira Rosati	Bruno.srosati@gmail.com	ADS	
G873AJ3	Gabriel Igor Dias Gomes	gabrielzinhoigor3@gmail.com	ADS	
N295AB6	Cristielen F. Cardoso da Silva	Crisfernandac10gmail.com	ADS	
G854BB0	Nicolas douglas dos santos	Nicolas.santos@aluno.unip.br	ADS	
N059CF8	Guilherme Bordinhon Silva Guimarães	guilherme.silva945@aluno.unip.br	ADS	

Registros:

Data do encontro	Observações
04/09/2024	Encontro para alinhamento do que precisa ser mudado e o que precisa ser feito no protótipo para que o mesmo se adeque aos novos requisitos.
19/09/2024	Encontro para discussão sobre a monografia.
27/09/2024	Encontro para alinhar sobre como será o desenvolvimento dos projetos Web, Desktop e Mobile
08/10/2024	Encontro para verificarmos o desenrolar do desenvolvimento dos projetos.
14/10/2024	Encontro para discutirmos sobre o design do projeto Web.
17/10/2024	Encontro para revisão em conjunto do estado atual da monografia.
22/10/2024	Encontro para alinhamento da monografia e atualizações gerais.
01/11/2024	Encontro para discussão sobre o Banco de Dados e as dificuldades em realizar a conexão entre os diferentes projetos (Web, Desktop e Mobile).
04/11/2024	Encontro para tentar realizarmos a conexão do banco de dados em conjunto.