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

Nome	R.A
EDUARDO SALGADO JUC	N072369
JOÃO CARLOS KENJI HECHT KAMINOBO	G773CB8
LUIS FELIPE SILVA DELLU	G758692
MARCOS VINÍCIUS DO SANTOS MOREIRA	N240HA0
MATHEUS DE LIMA MOISÉS	G7650H2
YGOR MELO RODRIGUES DA SILVA	G7640C5

SÃO JOSÉ DOS CAMPOS – SP

NOVEMBRO / 2024

	RA
João Carlos Kenji Hecht Kaminobo	G773CB8
Luís Felipe Silva Dellú	G758692
Marcos Vinícius dos Santos Moreira	N240HA0
Matheus de Lima Moisés	G7650H2
Ygor Melo Rodrigues da Silva	G7640C5
Eduardo Salgado Juc	N072369

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 trabalho teve como objetivo desenvolver um sistema integrado para otimizar o controle de operações em uma fazenda urbana, com foco em apoiar uma *startup* voltada à segurança alimentar. A crescente demanda por alimentos frescos e sustentáveis nas áreas urbanas exigiu soluções tecnológicas inovadoras, capazes de automatizar processos e melhorar a eficiência na produção de alimentos. O sistema proposto incluiu o uso de sensores inteligentes, automação de irrigação, controle de temperatura e iluminação, além de monitoramento remoto em tempo real, com o objetivo de promover a sustentabilidade e a gestão eficiente dos recursos. Este projeto contribuiu para o aumento da produtividade e para a redução de desperdícios, fortalecendo a cadeia de suprimentos de alimentos saudáveis e acessíveis em áreas urbanas.

Palavras-Chaves: sistema, fazenda urbana, segurança, sustentáveis, sustentabilidade.

SUMÁRIO

1. INTRODUÇÃO

A crescente urbanização e o aumento da população mundial trouxeram desafios significativos para a produção e distribuição de alimentos, especialmente nas grandes metrópoles. A falta de espaço para cultivo, os altos custos logísticos e a necessidade de práticas sustentáveis colocaram em evidência a importância de soluções inovadoras para garantir a segurança alimentar nas áreas urbanas. Nos últimos anos, observou-se uma demanda crescente por alimentos frescos, saudáveis e produzidos de forma ambientalmente responsável, o que impulsionou a busca por novas abordagens tecnológicas na agricultura.

Nesse contexto, este trabalho teve como objetivo desenvolver um sistema integrado para otimizar o controle de operações em fazendas urbanas, focado em apoiar uma *startup* voltada à segurança alimentar. O sistema foi projetado para responder à necessidade de automatização de processos agrícolas, visando aumentar a eficiência na produção de alimentos e reduzir o desperdício de recursos naturais. A proposta atendeu especialmente ao desafio de fornecer soluções tecnológicas adaptadas às limitações de espaço e às exigências ambientais das áreas urbanas.

O projeto foi delimitado à criação de um sistema que integrasse diversas tecnologias, como sensores inteligentes para monitoramento das condições climáticas e do solo, automação de sistemas de irrigação, controle de temperatura e iluminação, além de um módulo de monitoramento remoto em tempo real. Essas funcionalidades foram pensadas para facilitar o gerenciamento da produção agrícola em ambientes controlados, proporcionando aos agricultores urbanos um maior controle sobre os fatores que influenciam diretamente a qualidade e a quantidade de alimentos produzidos.

A motivação para o desenvolvimento desse sistema surgiu da crescente demanda por fazendas urbanas sustentáveis, que são vistas como uma solução promissora para o abastecimento de alimentos frescos nas cidades. Além disso, o projeto visou contribuir para a melhoria da gestão de recursos hídricos e energéticos, promovendo práticas agrícolas mais sustentáveis e menos dependentes de intervenções manuais. A utilização de tecnologias de automação e monitoramento em tempo real buscou tornar o processo produtivo mais eficiente, reduzindo os custos operacionais e o impacto ambiental.

Portanto, este projeto se propôs a oferecer uma solução prática e inovadora para fortalecer a cadeia de suprimentos de alimentos em áreas urbanas, com foco na

sustentabilidade e na eficiência operacional. Ao desenvolver um sistema que integra automação e monitoramento inteligente, o trabalho contribuiu para o avanço das práticas agrícolas urbanas, facilitando a adaptação das fazendas às demandas do mercado por alimentos saudáveis, acessíveis e sustentáveis.

2. PROGRAMAÇÃO ORIENTADA A OBJETOS II

O paradigma da programação orientada a objetos foi essencial para a criação do sistema, pois proporcionou diversas vantagens no desenvolvimento, como melhor organização, facilidade de manutenção e escalabilidade do código. A orientação a objetos possibilitou uma estruturação mais clara e eficiente do sistema, utilizando conceitos chaves como encapsulamento, herança, abstração, polimorfismo e composição, que facilitaram a modularização, reutilização de código e a adaptação do sistema a novas demandas.

A tecnologia orientada a objetos é fundamentada no que, coletivamente, chamamos de modelo de objetos, que engloba os princípios da abstração, hierarquização, encapsulamento, classificação, modularização, relacionamento, simultaneidade e persistência. (FARINELLI, 2007).

3. DESENVOLVIMENTO DE SOFTWARE PARA INTERNET

A matéria de Desenvolvimento de Software para Internet aborda os fundamentos essenciais para a criação de páginas *web* modernas, acessíveis, responsivas e funcionais. A partir dela, foi possível entender como aplicar as ferramentas e técnicas que permitem construir interfaces adaptáveis, bem estruturadas e alinhadas com as necessidades dos usuários e as demandas tecnológicas atuais.

Com o crescente uso de dispositivos móveis, tornou-se fundamental garantir que os sites sejam responsivos, ou seja, que se ajustem automaticamente a diferentes tamanhos de tela e tipos de dispositivos. Hoje em dia, os dispositivos móveis são a principal forma de acesso à *web*, já que os usuários estão conectados praticamente 24 horas por dia por meio de s*martphones* e *tablets*. No entanto, muitos sites ainda não são preparados para essa diversidade de dispositivos, prejudicando a experiência do usuário e, consequentemente, a eficiência do acesso à informação (MONTEIRO, 2020).

Dentro dessa perspectiva, a matéria também destacou a importância do uso estratégico de tecnologias como HTML e CSS, que vão além da estética e contribuem para uma estrutura sólida e acessível. Foi enfatizado como a aplicação de uma marcação semântica clara e organizada facilita a interpretação do conteúdo por diferentes dispositivos e ferramentas, promovendo a acessibilidade e melhorando a experiência do usuário.

Além disso, a disciplina permitiu a compreensão de heurísticas de usabilidade, que são essenciais para garantir que a interface seja intuitiva, clara e eficiente. Como destacado por NETO (2013), “Do ponto de vista do usuário, a interface é uma das partes mais importantes dos sistemas computacionais, porque por meio dela o usuário vê, ouve e sente.” Essa citação sublinha a importância de projetar interfaces que considerem as necessidades do usuário, permitindo uma interação mais fluida e sem frustrações. As heurísticas aplicadas, como a consistência visual, o *feedback* claro e a simplicidade, ajudaram a criar interfaces mais amigáveis, proporcionando uma experiência satisfatória e fácil de navegar.

A matéria também proporcionou uma compreensão profunda das técnicas e princípios que equilibram *design*, funcionalidade e eficiência. Esses conceitos garantem que as páginas *web* não apenas atendam aos padrões contemporâneos, mas também se destaquem pela qualidade e adaptabilidade. Assim, a disciplina ajudou a formar uma base sólida para o desenvolvimento de soluções tecnológicas robustas, flexíveis e inclusivas,

essenciais para atender às expectativas de um público cada vez mais exigente e diversificado.

4. TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

O sistema da fazenda urbana foi projetado para atender às necessidades das plataformas *PC*, *Web* e *Mobile*, utilizando métodos e tecnologias de programação orientada a objetos (*POO*). A base do sistema foi desenvolvida empregando métodos como encapsulamento, herança, polimorfismo e composição.

O sistema foi organizado com base em um diagrama de classes, que estruturou os módulos e suas relações, simbolizando classes como “Planta”, “Nutrientes” e “Ambientes”.

Encapsulamento: “O conceito de encapsulamento é decorrente do fato de se combinar os dados (atributos) e o código que manipula estes dados (métodos) em um único objeto. Ele garante que a única forma de acesso aos dados é através dos métodos disponíveis ao usuário (chamados públicos). Os demais métodos e os atributos da classe ficam sendo privados, ou seja, apenas funções membro da classe têm acesso direto aos mesmos.” (TAKASHI, 1990).

As classes foram criadas com o objetivo de limitar o acesso direto aos dados, empregando métodos para a leitura e modificação de informações, o que reforçou a segurança e a integridade do sistema.

Herança: “Por meio dessa característica do paradigma OO, um objeto filho herdará características e comportamentos do objeto pai. Quando estiver criando classes, você vai perceber que essa possibilidade permite o reaproveitamento de código e torna o trabalho mais racional e otimizado.” (NETO, 2002).

A herança foi usada para melhorar a reutilização do código, criando classes derivadas que compartilhavam atributos e métodos similares. Por exemplo, a classe “Planta” atuou como base para classes mais específicas de plantas, simplificando a criação de novas variedades sem reescrever o código.

Polimorfismo: “Polimorfismo é a propriedade de uma ou mais classes responderem à mesma mensagem, cada uma de uma forma diferente. Numa linguagem orientada a objeto, uma referência polimórfica é tal que, no decorrer do desenvolvimento do *software*, refere-se a mais de uma classe.” (TAKASHI, 1990).

A possibilidade de criar métodos que se ajustaram ao contexto de utilização foi crucial para a adaptabilidade do sistema.

Composição: A composição foi utilizada para estruturar pacotes. Esses pacotes continham classes independentes que garantiram uma organização clara e o isolamento de responsabilidades, facilitando futuras manutenções.

Testes de Unidade: Teste de unidades são testes automatizados de pequenas unidades de código, normalmente classes, as quais são testadas de forma isolada do restante do sistema. Foram realizados testes unitários automatizados para assegurar a operação das classes principais e confirmar a interação entre os métodos. Esses testes validaram a eficácia dos procedimentos de registro de plantas, acompanhamento do ambiente e demais ajustes.

5. PROJETO DE SISTEMAS ORIENTADO A OBJETOS

A engenharia de software tornou-se essencial nesse contexto, pois forneceu uma base estruturada para que equipes multidisciplinares colaborassem de forma eficiente e produtiva. Como a engenharia de software criou métodos, técnicas e ferramentas que tornaram o processo mais econômico e eficiente, foi um fator fundamental para projetos complexos e de longo alcance (SEABRA, 2013).

A metodologia "*Scrum* Ágil" foi utilizada para organizar o trabalho em estágios e atividades distintas, começando com o envio de requisitos e terminando com a aceitação do produto pelo usuário. Isso proporcionou ciclos de S*print*, entregas incrementais e revisão contínua de resultados, facilitando a adaptação constante e a satisfação do usuário.

O processo, dividido em etapas e apoiado por tecnologias específicas, estruturou as atividades de desenvolvimento, facilitando o gerenciamento do sistema. Técnicas para a Internet, como a criação de um sistema *Web*, implementação de testes de usabilidade e mecanismos de segurança, garantiram que o produto final atendesse às expectativas do usuário e estivesse alinhado às práticas de engenharia de software (SEABRA, 2013).

A UML oferece uma variedade de diagramas, como o diagrama de classes, de casos de uso e de sequência, que permitiu mapear a arquitetura do sistema, suas interações e a dinâmica de processos. Sua importância está na capacidade de organizar e documentar de maneira compreensível as complexidades do sistema, melhorando o entendimento, a tomada de decisões e a colaboração durante todas as fases do desenvolvimento (BOOCH, 2005).

6. GERENCIAMENTO DE PROJETO DE SOFTWARE

O gerenciamento de projetos de software, quando orientado pela metodologia do *Project Management Institute* (*PMI*), estruturou-se em um ciclo de vida que abrangeu cinco processos: iniciação, planejamento, execução, monitoramento e controle, e encerramento. Cada um desses processos desempenhou um papel crucial no desenvolvimento do software, promovendo um gerenciamento mais eficaz e eficiente. Para Charvat (2003), “Uma metodologia é um conjunto de orientações e princípios que podem ser adaptados e aplicados em uma situação específica”. Dessa forma, uma metodologia de gerenciamento de projetos representava um conjunto de processos, métodos e ferramentas destinados ao alcance dos objetivos do projeto.

O projeto envolveu *stakeholders* como clientes, usuários, equipe financeira, desenvolvedores e alta gerência. O envolvimento deles foi crucial para justificar a necessidade do desenvolvimento do software, pois suas expectativas e requisitos moldaram a visão do projeto. Um alinhamento claro com os *stakeholders* garantiu que os objetivos do projeto fossem compreendidos e apoiados desde o início, conforme descrito por Kerzner (2009).

7. EMPREENDEDORISMO

A *internet* se consolidou como o principal meio de comunicação entre empresas e indivíduos, proporcionando a interatividade e a troca de informações em uma escala global. Sua utilidade não se restringe a uma área geográfica específica, permitindo uma expansão econômica e do conhecimento, conforme destacado por Castells (2003). Esse ambiente digital abriu oportunidades para os empreendedores, que passaram a atender às necessidades de milhões de pessoas em todo o mundo, aproveitando-se da conectividade global para expandir seus negócios.

A visão empreendedora se mostrou essencial para o sucesso de qualquer negócio, especialmente em um cenário dinâmico e incerto. Ela envolve a capacidade de antecipar tendências, adaptar-se rapidamente às mudanças do mercado e identificar novas oportunidades. Empreendedores visionários buscam lacunas de mercado e, a partir disso, planejam e executam ações estratégicas que garantem o sucesso da empresa. Essa abordagem se torna ainda mais importante em ambientes em constante evolução, onde a flexibilidade e a inovação são fundamentais.

Um dos principais instrumentos para a estruturação de negócios é o plano de negócios, que serve como um guia para tomar decisões e gerenciar as operações de forma eficiente. No caso do sistema de negócios para processamento urbano de alimentos, a análise de mercado foi um ponto crucial. Compreender as necessidades do mercado permitiu posicionar o sistema de maneira competitiva, além de identificar potenciais clientes e tendências emergentes.

A estrutura organizacional do sistema envolve a equipe que participa do desenvolvimento e a governança da empresa, garantindo a eficiência e o controle adequados durante o desenvolvimento e gestão do projeto. Além disso, a viabilidade do sistema proposto depende de diversos fatores, como a inovação tecnológica, que permite a criação de soluções mais eficazes, e a demanda crescente por soluções sustentáveis, que é um motor importante no setor de processamento urbano de alimentos. Também são fundamentais as parcerias estratégicas com fornecedores e distribuidores, que asseguram a qualidade e a continuidade das operações.

8. GESTÃO DA QUALIDADE

A gestão da qualidade no desenvolvimento de sistemas informatizados envolveu princípios e práticas voltados para garantir que o *software* atendesse aos requisitos funcionais e de desempenho estabelecidos, mantendo-se eficiente, seguro e adaptável a mudanças. A qualidade de *software* foi definida como a conformidade aos requisitos explícitos e implícitos de um sistema, e essa conformidade foi assegurada por meio de processos padronizados que abrangeram todo o ciclo de vida do desenvolvimento (PRESSMAN , 2021).

Entre os aspectos principais da gestão da qualidade aplicados ao desenvolvimento de sistemas informatizados, o Ciclo *PDCA* (*Plan-Do-Check-Act*) destacou-se como uma metodologia de melhoria contínua amplamente utilizada. Ele ofereceu uma abordagem estruturada e iterativa para identificar problemas, implementar soluções, monitorar resultados e realizar os ajustes necessários, promovendo um processo de aprimoramento constante. O *PDCA* foi aplicado em diversas áreas, incluindo o desenvolvimento de sistemas informatizados, garantindo que o *software* fosse produzido com eficiência, qualidade e adaptabilidade.

9. DESENVOLVIMENTO DO PROJETO

9.1 PROGRAMAÇÃO ORIENTADA A OBJETOS II

O paradigma de Programação Orientada a Objetos (*POO*) foi amplamente utilizado no desenvolvimento do sistema devido às suas principais características: encapsulamento, herança, polimorfismo e abstração. Essas características permitiram uma modelagem mais próxima da realidade, o que facilitou tanto o desenvolvimento quanto a manutenção do sistema.

O encapsulamento foi aplicado para proteger dados, agrupando atributos e métodos em classes específicas, como Produto, Insumo e Fornecedor. Cada classe foi desenvolvida com métodos que manipularam diretamente seus dados, promovendo maior segurança e controle. A herança possibilitou a criação de uma estrutura hierárquica entre classes, permitindo que classes especializadas herdassem atributos e métodos de uma classe genérica, o que favoreceu a reutilização de código e facilitou a expansão do sistema. O polimorfismo, por sua vez, permitiu que uma operação assumisse múltiplas formas, o que trouxe flexibilidade ao código, permitindo aplicar o mesmo método em várias classes e simplificando sua manutenção. A abstração possibilitou a criação de representações simplificadas de objetos, focando nas características principais para o sistema. Isso auxiliou na criação de interfaces, como a interface Cadastro, que padronizou os métodos essenciais.

A aplicação foi implementada como um sistema *Desktop* em *C#*, utilizando *Windows Forms* para o desenvolvimento de uma interface gráfica adaptada para *Desktop*, o que facilitou o *design* de uma interface intuitiva, com elementos como botões de navegação e formulários para operações de *CRUD* (criação, leitura, atualização e exclusão). No desenvolvimento, a programação de eventos em *C#* foi crucial, pois permitiu acionar funções específicas, como “AdicionarProduto”, “AtualizarFornecedor” e “RegistrarVenda”, conforme a interação do usuário com a interface.

O banco de dados foi estruturado em *SQL Server*, com tabelas relacionadas às entidades do sistema, como Produto, Cliente e Venda. Para garantir a integridade dos dados, foram implementadas regras de consistência, como o uso de chaves primárias e estrangeiras. Por exemplo, a tabela de Venda possuía uma chave estrangeira que referenciava a tabela de Cliente, garantindo que cada venda estivesse associada a um cliente válido.

Para a implementação de operações *CRUD*, cada entidade no sistema teve métodos específicos para criação, leitura, atualização e exclusão de registros. O *Entity Framework* facilitou a criação desses métodos, permitindo que a aplicação se comunicasse com o banco de dados utilizando *LINQ* e expressões *lambda*, promovendo uma camada de acesso a dados robusta e eficiente. Além disso, foram aplicadas transações para garantir a integridade dos dados em operações complexas, como o registro de uma venda e a atualização simultânea do estoque. Caso uma etapa falhasse, as demais eram revertidas, garantindo a consistência dos dados.

A conformidade com a modelagem foi assegurada com a criação de um diagrama de classes que representou a estrutura do sistema, incluindo as classes e suas relações, o que facilitou a manutenção e compreensão da arquitetura. Cada classe no diagrama refletiu uma entidade do sistema, com métodos e atributos mapeados de acordo com os requisitos. O Diagrama de Entidade e Relacionamentos (DER) foi utilizado para representar a estrutura do banco de dados, demonstrando as tabelas e os relacionamentos entre elas, garantindo que o banco de dados suportasse as operações necessárias e refletisse o *design* de alto nível.

9.2 DESENVOLVIMENTO DE SOFTWARE PARA INTERNET

No desenvolvimento de *software* para a Internet, utilizamos tecnologias como HTML e CSS de maneira estratégica, indo além da simples estética para garantir uma estrutura sólida e acessível às páginas *web*. Trabalhamos com práticas que priorizaram a organização do código e a responsividade, criando sistemas funcionais e adaptáveis.

No HTML, aplicamos uma marcação semântica, utilizando elementos como <header>, <main>, <section> e <footer> para organizar as páginas de forma clara e significativa. Evitamos o uso excessivo de , o que nos permitiu criar um código mais limpo, legível e acessível. Essa abordagem também facilitou a manutenção do sistema, além de melhorar o ranqueamento em motores de busca, uma vez que a semântica ajudou a descrever melhor o conteúdo para essas ferramentas.

Com o CSS, garantimos que as interfaces fossem responsivas, utilizando técnicas como *Flexbox* e *Grid Layout* para ajustar os layouts a diferentes dispositivos e tamanhos de tela. Isso assegurou uma experiência de usuário consistente, independentemente do meio de acesso. Além disso, otimizamos o desempenho ao evitar redundâncias no código CSS, criando um design eficiente e escalável.

A responsividade foi um dos aspectos mais importantes do nosso trabalho, pois a diversidade de dispositivos utilizados pelos usuários cresceu exponencialmente nos últimos anos. Criar sistemas que se adaptassem automaticamente a diferentes telas foi fundamental para garantir acessibilidade e usabilidade. Por meio de media queries, ajustamos elementos como tamanhos de fonte, margens e espaçamentos, assegurando que o conteúdo permanecesse legível e esteticamente agradável em qualquer ambiente, seja em um *desktop* de alta resolução ou em um smartphone com tela menor.

Para alcançar essa responsividade, utilizamos valores relativos em vez de absolutos no CSS, como porcentagens (%), unidades relativas à *viewport* (vw e vh) e em ou rem para tamanhos de fonte e espaçamentos. Por exemplo, definimos larguras de elementos usando *width*: 50% para garantir que ocupassem metade da tela disponível, independentemente do tamanho total. Essa abordagem tornou o design fluido, adaptando-se automaticamente a diferentes resoluções. A **Figura 1** ilustra um exemplo aplicado, em que a largura de um container foi configurada de forma relativa, demonstrando a capacidade de redimensionamento dinâmico.

Figura 1 – Código Responsivo

```css
.search-container {
    display: flex;
    align-items: center;
    margin-top: 2rem; /* Usando rem para espaçamento */
    flex-wrap: wrap; /* Permite que os elementos se ajustem em telas menores */
}

.search-input {
    padding: 0.8em; /* Usando em para o preenchimento interno */
    border: 1px solid #ccc;
    border-radius: 5px;
    width: 100%; /* Largura total em telas menores */
    max-width: 20rem; /* Limita a largura máxima em telas maiores */
    margin-right: -22rem; /* Pequeno espaço entre o input e o botão */
    box-sizing: border-box; /* Inclui o padding no cálculo da largura */
}
```

Fonte: Digital Nexus (2024)

Além disso, implementamos layouts flexíveis usando *Flexbox* e *Grid Layout*, que nos permitiram criar estruturas fluídas, capazes de se reorganizar dependendo do tamanho da tela ou das orientações do dispositivo. Por exemplo, menus horizontais em telas maiores se transformavam em menus verticais em dispositivos menores, enquanto imagens eram dimensionados automaticamente para evitar cortes ou distorções.

A utilização de valores relativos também teve um impacto direto na experiência do usuário. Garantimos que as interações fossem intuitivas e sem frustrações, eliminando problemas como o zoom manual em telas pequenas ou elementos fora de alcance em dispositivos móveis. Essas práticas não apenas melhoraram a usabilidade, mas também contribuíram para métricas de retenção e engajamento, já que os usuários podiam navegar com facilidade.

O uso de imagens SVG (*Scalable Vector Graphics*) foi uma das práticas exploradas no desenvolvimento das páginas web, permitindo uma abordagem mais eficiente e flexível para a inserção de gráficos e ícones. Ao contrário de formatos de imagem rasterizados, como JPEG ou PNG, as imagens SVG são baseadas em vetores, o que significa que podem ser escaladas infinitamente sem perda de qualidade. Isso se mostrou especialmente útil para garantir que os elementos gráficos se ajustassem de maneira perfeita em diferentes

dispositivos e resoluções, mantendo a clareza e nitidez. Além disso, as imagens SVG são leves em termos de tamanho de arquivo, o que contribui para a performance do site. Sua flexibilidade também permitiu que fossem facilmente manipuladas com CSS, possibilitando animações e alterações dinâmicas sem necessidade de novos arquivos. Essa combinação de escalabilidade, leveza e personalização deu maior controle sobre o design da interface, melhorando a experiência do usuário e a eficiência do sistema como um todo.

No contexto de engenharia de *software*, a responsividade foi tratada como um aspecto essencial do design centrado no usuário. Ao focarmos nela e adotarmos valores relativos no CSS, entregamos soluções modernas e inclusivas, que se mantiveram relevantes em um mercado onde a variedade de dispositivos e tamanhos de tela é uma realidade incontornável.

9.3 TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

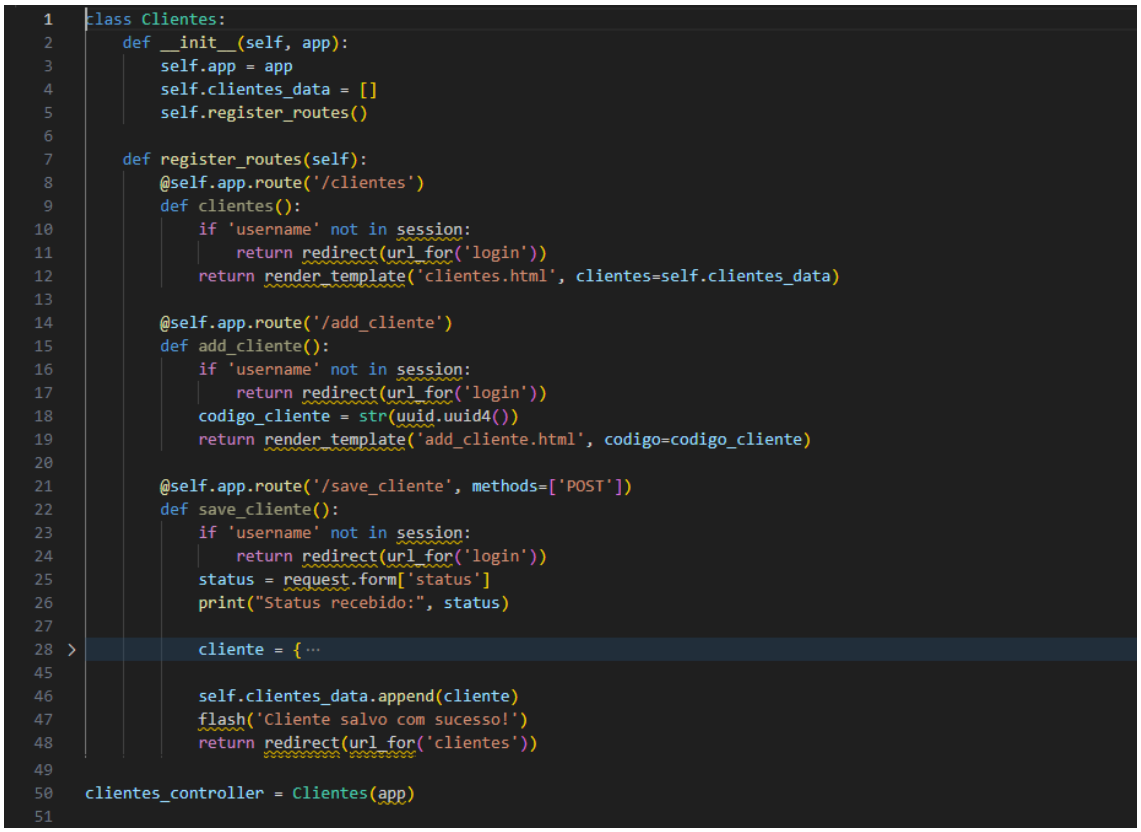
O *Flask* foi escolhido como *framework backend* devido à sua flexibilidade e leveza, o que possibilitou a criação de uma aplicação completa e organizada. A estrutura modular do *Flask* facilitou a implementação de funcionalidades complexas de forma simplificada e escalonável. Além disso, sua alta compatibilidade com *Python*, uma linguagem que oferece clareza e eficiência na construção de lógica de *back-end*, foi um fator determinante. O *Flask* também se integrou a sistemas poderosos de controle de sessão e autenticação de usuários, protegendo os dados operacionais e do usuário.

Os dados inseridos no sistema foram salvos em um banco de dados, garantindo a persistência e a integridade das informações. Isso permitiu que o sistema suportasse grandes volumes de dados, incluindo detalhes sobre clientes, status de produção, informações sobre vendas e muito mais. O uso de um banco de dados facilitou a atualização, consulta e remoção de registros, transformando a aplicação em uma ferramenta de gerenciamento completa.

Para assegurar uma estrutura organizada e de fácil manutenção, foi seguida uma abordagem de Orientação a Objetos (*OO*). Cada área do sistema, como clientes, funcionários e produção, foi desenvolvida como uma classe independente, com suas próprias rotas e métodos. Esse encapsulamento facilitou a manutenção e a expansão do sistema, permitindo que melhorias e correções fossem feitas de forma localizada, sem interferir nas demais áreas. A aplicação dos princípios de *OO* também proporcionou um código mais limpo e reutilizável, facilitando a compreensão e a evolução do sistema.

A **Figura 2** mostrada abaixo, exemplifica a classe responsável pela área de clientes do sistema. Nela, é possível observar como as rotas e a lógica de cada área foram encapsuladas em uma classe específica, permitindo um código modular e de fácil manutenção.

Figura 2 – Classe Clientes Python



```python
class Clientes:
    def __init__(self, app):
        self.app = app
        self.clientes_data = []
        self.register_routes()

    def register_routes(self):
        @self.app.route('/clientes')
        def clientes():
            if 'username' not in session:
                return redirect(url_for('login'))
            return render_template('clientes.html', clientes=self.clientes_data)

        @self.app.route('/add_cliente')
        def add_cliente():
            if 'username' not in session:
                return redirect(url_for('login'))
            codigo_cliente = str(uuid.uuid4())
            return render_template('add_cliente.html', codigo=codigo_cliente)

        @self.app.route('/save_cliente', methods=['POST'])
        def save_cliente():
            if 'username' not in session:
                return redirect(url_for('login'))
            status = request.form['status']
            print("Status recebido:", status)

            cliente = {···

            self.clientes_data.append(cliente)
            flash('Cliente salvo com sucesso!')
            return redirect(url_for('clientes'))

clientes_controller = Clientes(app)
```

Fonte: Digital Nexus (2024)

A interface foi desenvolvida com *HTML* e *CSS*, combinados com o sistema de *templates* do *Flask* (*Jinja*), para proporcionar uma experiência de usuário interativa e visualmente agradável. O *Jinja* permitiu a renderização de dados do *backend* diretamente nas páginas *HTML*, o que garantiu que qualquer informação armazenada no banco de dados fosse imediatamente refletida na interface *web*. A escolha por *HTML* e *CSS* possibilitou uma apresentação clara das informações e uma navegação intuitiva, elementos fundamentais para um sistema de gestão que buscou a eficiência operacional.

A aplicação utilizou *Python* para lidar com toda a lógica de *backend*, incluindo validação de dados, controle de fluxo e cálculos necessários para o bom funcionamento do sistema. *Python* forneceu uma base sólida para todas as funcionalidades, garantindo segurança e precisão nos processos.

Em resumo, este sistema foi concebido como uma solução centralizada e completa para a gestão de fazendas urbanas. A escolha do *Flask* e a aplicação de uma arquitetura orientada a objetos, junto ao uso de *HTML* e *CSS*, resultaram em uma aplicação robusta e escalável, que atendeu a todas as necessidades de gestão de uma fazenda urbana. A versão *web*, sendo a mais abrangente, ofereceu aos usuários o controle total das operações, posicionando-se como a principal ferramenta de gestão dentro do sistema.

9.4 PROJETO DE SISTEMA ORIENTADO A OBJETOS

O sistema de administração para uma pequena empresa foi desenvolvido utilizando recursos de sistemas de objetos orientados a projetos. Esses elementos foram aplicados para estruturar um sistema robusto, modular e escalável. Com base em técnicas de desenvolvimento e práticas de modelagem, métodos foram aplicados para facilitar a manutenção e a evolução do sistema.

Um dos primeiros modelos elaborados foi o diagrama de classes, que foi utilizado para organizar e representar a estrutura do sistema. Esse diagrama demonstrava a estrutura projetada com suas responsabilidades e relações com outras informações, refletindo funcionalidades oportunas, fornecendo a definição de atributos e métodos eficazes, além de facilitar a reutilização e alteração de código em um sistema evolutivo. Segundo Furlan (1998), o diagrama de classes representava uma estrutura lógica estática que exibia uma coleção de elementos declarativos do modelo, como classes, tipos e seus respectivos conteúdos e relacionamentos.

9.5 GERENCIAMENTO DE PROJETO DE SOFTWARE

9.5.1. DECLARAÇÃO DE ESCOPO

Na fase de iniciação, a equipe de gerenciamento de projetos começou com a definição do escopo do projeto, com base nas necessidades do cliente, que deseja um software multiplataforma para o gerenciamento de uma fazenda urbana.

Durante o planejamento, foram realizadas diversas atividades essenciais:

• Definição do Escopo do Projeto: O escopo foi delineado claramente, especificando funcionalidades, entregáveis e limites do projeto. Isso evitou ambiguidades e reduziu o risco de escopo, um conceito discutido por PMBOK (2017).

• Planejamento do Cronograma: As atividades foram definidas e sequenciadas usando técnicas como a Decomposição e o Método do Caminho Crítico (CPM). A estimativa de duração das atividades foi baseada em experiências anteriores e na consulta com especialistas, resultando em um cronograma realista.

• Estimativa de Custos: Foi elaborado um orçamento detalhado que considerou todos os recursos necessários, com base nas estimativas de custo das atividades e alocação de pessoal.

• Planejamento dos Recursos Humanos: As responsabilidades foram atribuídas com base nas habilidades e experiências dos membros da equipe, garantindo que as pessoas certas estivessem nos lugares certos.

• Sistema de Comunicação: Um plano de comunicação foi desenvolvido para assegurar que todas as partes interessadas recebessem informações relevantes de forma oportuna. As reuniões semanais de status e relatórios mensais foram implementados como ferramentas-chave de comunicação.

• Gerenciamento de Riscos: Identificaram-se riscos potenciais, que foram avaliados em termos de impacto e probabilidade. Um plano de mitigação foi elaborado para os riscos mais significativos, conforme as diretrizes do PMI (2017).

Execução

Na execução do projeto, o sistema de comunicação previamente estabelecido foi fundamental para a colaboração efetiva. A equipe utilizou ferramentas como Slack e Trello, que permitiram um acompanhamento em tempo real das tarefas e facilitaram a interação contínua. A frequência de reuniões foi mantida semanalmente, e relatórios diários garantiram que todos estivessem atualizados sobre o progresso e os desafios enfrentados.

Monitoramento e Controle:

O monitoramento e controle foram realizados em paralelo à execução. A equipe utilizou métricas para controlar o cronograma, comparando o progresso real com o planejado. O gerenciamento da qualidade foi implementado por meio de revisões de código e testes constantes, assegurando que os entregáveis atendessem aos padrões estabelecidos. O engajamento das partes interessadas foi monitorado através de feedback contínuo e atualizações regulares, permitindo ajustes proativos conforme necessário.

Encerramento:

No encerramento do projeto, a documentação foi organizada e arquivada, incluindo relatórios finais e registros de comunicação. Uma avaliação de desempenho foi realizada, onde a equipe discutiu sucessos e áreas de melhoria em uma reunião de "lições aprendidas". Isso proporcionou um espaço para a reflexão sobre as práticas adotadas e a identificação de melhorias para projetos futuros. A transferência de conhecimento foi assegurada por meio de documentação detalhada e sessões de treinamento para a equipe de operações, garantindo que o aprendizado acumulado fosse aproveitado no futuro.

Na fase de iniciação, a equipe de gerenciamento de projetos iniciou com a definição do escopo do projeto, baseando-se nas necessidades do cliente, que desejava um software multiplataforma para o gerenciamento de uma fazenda urbana.

Objetivo principal: Desenvolver um sistema de gestão para uma fazenda urbana.

Funcionalidades chaves, entregáveis:

- **Gerenciamento da produção**: Permitiu o controle da produção, incluindo data de início e previsão de colheita; informações sobre condições ideais de plantio, como luz, umidade e temperatura.
- **Módulo de Fornecedores**: Permitiu o cadastro, edição e consulta dos fornecedores no sistema, incluindo informações como Razão Social, Nome Fantasia, CNPJ, endereço e contatos.
- **Módulo de Insumos**: Permitiu o cadastro, edição e consulta dos insumos no sistema, incluindo informações como nome, tipo e quantidade.
- **Módulo de Equipes**: Permitiu o gerenciamento de equipes para tarefas específicas na empresa.
- **Módulo de Clientes**: Permitiu o cadastro, edição e consulta dos clientes, incluindo informações como Razão Social, Nome Fantasia, CNPJ, endereço e contatos.
- **Módulo de Funcionários**: Permitiu o cadastro, edição e consulta de um funcionário na empresa.
- **Controle de Compras**: Permitiu a solicitação de compra e definiu seu estado.
- **Relatórios de vendas**: Permitiu relatórios quantitativos referentes às vendas de um produto específico.
- **Interface Intuitiva**: O sistema teve uma interface minimalista e eficiente para gerenciar as informações.
- **Controle de Atividade**: Permitiu o controle de ativar ou inativar um produto, produção, fornecedor, cliente, funcionário e equipe.
- **Controle de acesso**: O sistema teve um controle de permissões baseado na hierarquia do usuário no sistema.
- **Critérios de aceitação:** A interface deveria ser acessível em dispositivos móveis (*Android*), *Web* e *Desktop*.

9.5.2. NECESSIDADES E EXPECTATIVAS DOS *STAKEHOLDERS*

***Stakeholder* 1**: Equipe de desenvolvimento

Necessidades:

- O projeto precisava ser escalável e capaz de suportar o crescimento da empresa.
- O sistema deveria possuir segurança, com controle de usuários baseado na hierarquia de cargos na empresa.

Expectativas*:*

- Utilização de tecnologia atual, que fosse fácil de manter e de atualizar.
- Performance do sistema: o sistema deveria ser rápido e responder imediatamente às solicitações feitas pelo usuário.
- Documentação que fosse clara e objetiva para futuras atualizações no sistema.

***Stakeholder* 2: Fornecedores**

Necessidades:

- Pagamentos em dia: qualquer atraso de pagamento poderia comprometer o fluxo de caixa da empresa.
- Requisitos claros: os fornecedores necessitavam de requisitos bem definidos para garantir a qualidade e atender às expectativas esperadas do cliente.

Expectativas*:*

- Parcerias estratégicas: estabelecer uma comunicação mais aberta e colaborativa, alcançando soluções inovadoras e promovendo o crescimento de ambas as partes.
- Competitividade e flexibilidade: oferecer condições comerciais competitivas e flexibilidade diante de mudanças no mercado.
- Prazos e condições de entrega: cumprir os prazos acordados e garantir a qualidade do produto na entrega.

***Stakeholder* 3: Funcionários**

Necessidades:

- Ambiente de trabalho seguro: oferecer condições físicas e psicológicas adequadas.
- Oportunidades de crescimento: permitir o desenvolvimento de habilidades por meio de treinamentos e oportunidades de promoção.

Expectativas*:*

- Ambiente organizacional inclusivo: os funcionários desejavam trabalhar em ambientes respeitosos, que valorizassem a diversidade e criassem um clima de colaboração para o alcance dos objetivos.
- Reconhecimento: serem reconhecidos pelos desafios enfrentados, participando de projetos que impactassem o crescimento da empresa.

***Stakeholder* 4: Comunidade**

Necessidades*:*

- Educação e capacitação: investir em programas educacionais e de treinamentos para aumentar a taxa de empregabilidade na região.

Expectativas*:*

- Sustentabilidade e meio ambiente: minimizar os impactos ambientais, como o controle de gastos de água, o tratamento adequado de recursos e a preservação de áreas verdes.

***Stakeholder* 5: Governo**

Necessidades*:*

- Cumprimento das leis: atender às obrigações fiscais, trabalhistas e ambientais estabelecidas pela legislação.

Expectativas*:*

- Pagamento dos impostos: quitar todos os impostos estabelecidos dentro do prazo, fundamentais para o financiamento de serviços públicos em prol da sociedade.

9.5.3. LIMITAÇÕES E EXCLUSÕES

Limitações*:*

- O sistema não gerenciaria as compras e os fornecedores diretamente.

Exclusões*:*

- Não haveria integração com sistemas externos de logística.
- Não haveria funcionalidades de gestão financeira no projeto.

9.5.4. CRONOGRAMA E ORÇAMENTO

Sequenciamento das atividades*:*

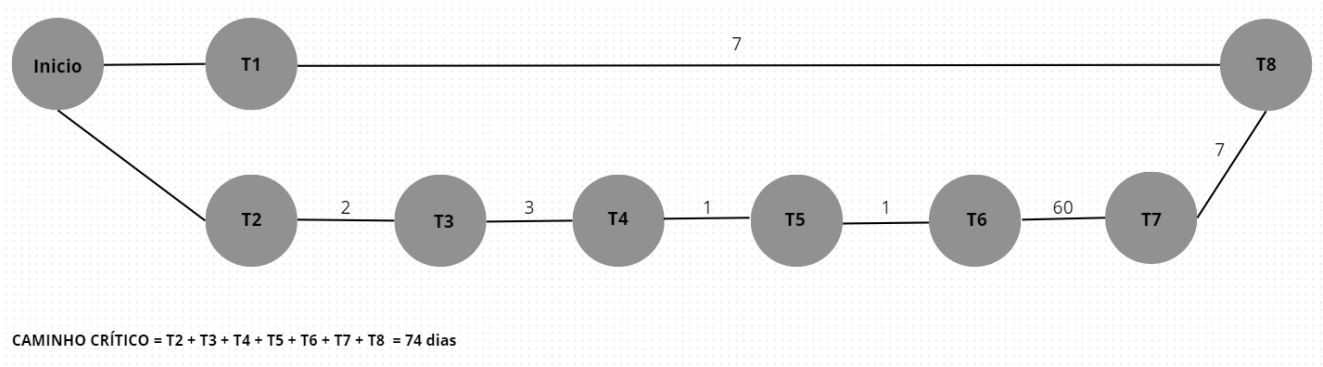
- Foram definidas as atividades dependentes do projeto, a fim de controlar os prazos de execução e evitar atrasos nas etapas futuras.

Figura 3 – Cronograma e Orçamento

T1	Revisão do Documento	7 dias	
T2	Revisão do Protótipo	2 dias	
T3	Ajustes dos Requisitos	3 dias	2TI
T4	Validação dos Requisitos	1 dia	3TI
T5	Definição das Sprints	1 dia	4TI
T6	Desenvolvimento	60 dias	5TI
T7	Testes	10 dias	6TI
T8	Entrega do Sistema	0 dias	7TI

Fonte: Digital Nexus (2024)

Figura 4 – Sequenciamento das Atividades



Fonte: Digital Nexus (2024)

Gestão de Riscos: A gestão de riscos foi uma prática essencial, pois ajudou a identificar e mitigar riscos que poderiam afetar ou atrasar o objetivo final, além de aprimorar o processo de tomada de decisão e responder a ações inesperadas de forma eficaz.

Figura 5 – Tabela de Riscos

Risco	Probabilidade	Impacto	Prioridade	Resposta
Integrante da equipe não comparecer na reunião	Alto	Alto	Médio	Agendamento antecipado, reuniões gravadas.
Falhas de Comunicação	Alto	Alto	Alto	Feedback contínuo, documentação clara e objetiva
Atraso na entrega nas Sprints	Alto	Alto	Crítico	Reuniões diárias, planejamento, acompanhamento e transparência
Equipamento apresentar mal funcionamento	Baixo	Alto	Alto	Manutenção preventiva regular, backup, contratos de suporte e garantia

Fonte: Digital Nexus (2024).

Durante todo o projeto, manteve-se a frequência de reuniões semanais no final de cada semana, o que garantiu o acompanhamento das atividades e o alinhamento das metas entre os membros da equipe.

As reuniões foram essenciais para garantir a comunicação e o progresso dos projetos. Para facilitar a interação e troca de informações, utilizou-se o *Microsoft Teams* como plataforma principal.

O *Teams* proporcionou um ambiente centralizado para videoconferências, troca de mensagens, compartilhamento de arquivos e organização de tarefas, otimizando a colaboração entre os participantes. A utilização dessa ferramenta contribuiu para a manutenção de uma comunicação clara e produtiva ao longo de todo o semestre.

Na *Digital Nexus*, a equipe utilizou um sistema colaborativo que combinava diversas ferramentas digitais para otimizar o planejamento e a execução de tarefas de forma eficiente.

O *Microsoft Teams* foi o principal meio de comunicação da equipe, sendo onde ocorreram as reuniões para organização e priorização das Sprints. Utilizou-se a metodologia ágil Scrum, onde as Sprints foram separadas com duração de 10 dias cada.

O *Github* foi utilizado para controle das versões do código. Essa ferramenta permitiu à equipe colaborar simultaneamente no desenvolvimento do projeto, além de corrigir *bugs* por meio de *pull requests*, antes de serem integrados à versão principal.

Os benefícios do sistema colaborativo foram a transparência, pois a equipe tinha acesso ao projeto em tempo real, e a comunicação eficiente, com as reuniões frequentes para acompanhamento das tarefas.

9.5.5 AVALIAÇÃO DE DESEMPENHO DO PROJETO

Objetivo: Foi realizada uma avaliação do projeto, que destacou os pontos que funcionaram bem e os aspectos que poderiam ser melhorados em novas iniciativas. Esse levantamento buscou consolidar as boas práticas e promover ajustes para o desempenho em projetos futuros.

1. Resultados Positivos:

- **Realização das Metas:** Verificou-se se o sistema de administração da fazenda urbana atendeu às necessidades do projeto, como automação, monitoramento em tempo real e práticas sustentáveis. Essa avaliação buscou garantir que o sistema estivesse alinhado com os requisitos, promovendo a produtividade e o uso sustentável dos recursos.
- **Eficiência no Desenvolvimento:** Avaliou-se se as técnicas e tecnologias empregadas, incluindo a orientação a objetos, contribuíram para a rapidez e escalabilidade do sistema. Graças a essa verificação, foi possível identificar se as escolhas técnicas atenderam às expectativas de desempenho e suportaram o crescimento do projeto.

2. Identificação de Áreas para Melhorias:

- **Gestão de Recursos e Tempo:** Examinou-se o cronograma do projeto e a distribuição de recursos, verificando se houve desvio de prazos ou recursos.
- **Desempenho e Usabilidade:** Considerou-se o *feedback* dos usuários finais para identificar melhorias na interface e na experiência do usuário.

3. Reunião de Lições Aprendidas:

- **Objetivo:** Reunir a equipe para discutir os obstáculos superados e as lições aprendidas, incentivando um aprimoramento contínuo.
- **Pontos Abordados:**
    - Avaliação dos processos de comunicação e colaboração.
    - Exame de erros técnicos ou de integração, buscando soluções para prevenir situações similares em projetos futuros.
    - *Feedback* dos envolvidos sobre o desenvolvimento e implantação do sistema para aprimorar o próximo ciclo de projetos.

Em geral, concluiu-se que a avaliação desempenhou um papel importante no registro, servindo como base para futuros projetos, orientando ajustes e aperfeiçoamentos na execução, garantindo as melhores práticas e maior satisfação dos *Stakeholders*.

Foi utilizada a linguagem *C#* para o desenvolvimento da aplicação *Desktop*, pois ela ofereceu desempenho e eficiência para utilizar o paradigma orientado a objetos, além de possuir facilidade na manutenção e escalabilidade do código, tornando-o mais modular e reutilizável, utilizando os conceitos de Herança, Polimorfismo, Abstração, Encapsulamento e Composição. A linguagem também ofereceu compatibilidade com a plataforma *Windows*, o que facilitou o desenvolvimento para sistemas operacionais dessa família, garantindo uma execução otimizada e acessos a possíveis *APIs* do *Windows*. Além disso, ela tem excelente suporte para integração com outras tecnologias, como banco de dados e sistemas externos, permitindo que as aplicações se comunicassem mais facilmente com outras partes do sistema, caso fosse necessário.

9.6 EMPREENDEDORISMO

A iniciativa se baseia em uma visão empreendedora, identificando uma lacuna no mercado e propondo uma solução inovadora e escalável para atender às necessidades de um público urbano cada vez mais exigente. Através do empreendedorismo, o projeto busca antecipar tendências, adaptar-se rapidamente às mudanças e explorar novas oportunidades de negócios, garantindo competitividade e sucesso no longo prazo.

A primeira etapa envolve uma análise detalhada do mercado urbano de alimentos, identificando as necessidades dos consumidores e as falhas existentes. A partir disso, o projeto buscará integrar tecnologias avançadas, como automação e sistemas de gestão de resíduos, para otimizar o processo de produção e reduzir impactos ambientais.

O plano de negócios incluirá a definição de uma estrutura organizacional sólida, com uma equipe multidisciplinar, incluindo especialistas em tecnologia, sustentabilidade e gestão de operações. Também serão desenvolvidas parcerias estratégicas com fornecedores e distribuidores, fundamentais para garantir a viabilidade do sistema e sua expansão.

Além disso, a sustentabilidade será um pilar central, com foco em práticas ecológicas ao longo de toda a cadeia produtiva. O sistema será projetado para se adaptar rapidamente às mudanças do mercado e aos avanços tecnológicos, posicionando a empresa de forma competitiva e permitindo que ela atenda às novas demandas de um mercado dinâmico.

Em resumo, o projeto busca criar uma solução inovadora para o processamento urbano de alimentos, combinando uma visão empreendedora com tecnologia, sustentabilidade e uma estratégia de negócios bem estruturada. Com isso, pretende-se atender de forma eficiente e responsável às necessidades de alimentação nas cidades, ao mesmo tempo em que se aproveita as oportunidades oferecidas pelo ambiente de negócios em constante evolução.

9.7 GESTÃO DA QUALIDADE

A gestão da qualidade no desenvolvimento de sistemas informatizados foi baseada na aplicação de práticas e princípios que garantiram que o software atendesse aos requisitos funcionais e de desempenho definidos. Esse processo buscou assegurar que o sistema fosse eficiente, seguro e adaptável a mudanças, alinhando-se ao conceito de qualidade de software de Pressman, que a definiu como a conformidade aos requisitos explícitos e implícitos do sistema. Para alcançar essa conformidade, foram seguidos processos padronizados que abrangeram todas as etapas do ciclo de vida do desenvolvimento.

O Ciclo PDCA (Plan-Do-Check-Act) desempenhou um papel essencial na gestão da qualidade, sendo aplicado como uma metodologia de melhoria contínua. Essa abordagem estruturada e iterativa possibilitou a identificação de problemas, a implementação de soluções, o monitoramento dos resultados e os ajustes necessários. Durante o desenvolvimento do sistema, o PDCA foi empregado em diferentes etapas, promovendo o aprimoramento constante e a manutenção de elevados padrões de qualidade.

Na fase de planejamento (Plan), os objetivos do projeto foram definidos, os requisitos do sistema foram levantados e as métricas de qualidade a serem alcançadas foram estabelecidas. Ferramentas de análise de requisitos e de levantamento de riscos foram utilizadas para antecipar desafios e alinhar o projeto às expectativas dos stakeholders.

Na etapa de execução (Do), as soluções planejadas foram implementadas por meio de processos padronizados que incluíram boas práticas de programação, uso de ferramentas de controle de versão e aplicação de metodologias ágeis. As funcionalidades principais foram integradas ao sistema, priorizando a conformidade com os requisitos previamente estabelecidos.

Durante a verificação (Check), foram realizados testes contínuos para garantir que o sistema atendesse às especificações definidas. Testes funcionais, de desempenho e de segurança foram conduzidos para identificar e corrigir erros antes da entrega. Métricas como

tempo de resposta e taxa de erros corrigidos foram monitoradas para avaliar a eficácia das soluções implementadas.

Por fim, na etapa de ação (Act), melhorias e alterações foram feitas com base nos resultados dos testes. Problemas identificados foram resolvidos, e o desempenho do sistema foi otimizado. As lições aprendidas ao longo do processo foram documentadas e incorporadas aos projetos futuros, reforçando o ciclo de melhoria contínua.

Ao final do projeto, a aplicação do Ciclo PDCA demonstrou ser uma abordagem eficaz na gestão da qualidade. O software entregue atendeu aos requisitos funcionais e de desempenho, apresentando adaptabilidade e conformidade às expectativas dos stakeholders. O uso do PDCA permitiu a identificação e resolução proativa de problemas, reduzindo retrabalho e otimizando os recursos. Com isso, o projeto foi concluído com sucesso, consolidando-se como um modelo de desenvolvimento de sistemas informatizados de alta qualidade.

10. CONCLUSÃO

A conclusão evidenciou a dedicação da equipe em abordar um problema relevante em áreas urbanas: a gestão eficaz e sustentável de fazendas urbanas. Por meio da combinação de programação orientada a objetos e métodos avançados de desenvolvimento de *software*, foi desenvolvido um sistema robusto, capaz de atender às demandas dinâmicas do mercado.

O projeto destacou a importância de tecnologias como sensores avançados, automação da irrigação e monitoramento em tempo real para atender às necessidades de uma startup voltada para fazendas urbanas. A escolha de ferramentas e estruturas como *C#* e *Flask* possibilitou o desenvolvimento de um aplicativo multiplataforma, abrangendo dispositivos *desktop*, móveis e acesso web. Além disso, a utilização de abordagens metodológicas como o *Scrum* contribuiu para a organização e gestão das etapas do desenvolvimento.

Os resultados obtidos demonstraram que as soluções implementadas influenciaram positivamente a eficiência operacional, reduziram desperdícios e incentivaram práticas sustentáveis, estabelecendo um modelo promissor para o setor. Apesar dos desafios enfrentados, a integração de testes unitários e as avaliações periódicas asseguraram que o sistema atendesse aos padrões de qualidade e aos objetivos definidos.

Após uma análise detalhada do trabalho realizado, concluiu-se que o projeto não apenas atendeu às demandas atuais, mas também lançou bases sólidas para melhorias futuras e possíveis expansões. De maneira semelhante, o projeto evidenciou o valor da tecnologia na concepção de soluções para os desafios da segurança alimentar urbana, promovendo novas possibilidades de estudo e avanços nesse campo.

11. REFERÊNCIAS

TEIXEIRA, Rubens. EMPREENDEDORISMO DIGITAL: UM ESTUDO DESCRITIVO UTILIZANDO GOOGLE ADS DIGITAL ENTREPRENEURSHIP: A DESCRIPTIVE STUDY USING GOOGLE ADS. Disponível em: <https://fatece.edu.br/arquivos/arquivosrevistas/empreendedorismo/volume10/Rubens%20Teixeira.pdf>. Acesso em: 20 out 2024.

CASTELLS, M. A galáxia da Internet: reflexões sobre a internet, os negócios e a sociedade. Rio de Janeiro: Jorge Zahar, 2003.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software-9**. McGraw Hill Brasil, 2021.

Seabra, D. (2013). Engenharia de Software: Fundamentos, Métodos e Aplicações.

SEABRA, João. UML: Uma ferramenta para o design de software. Rio de Janeiro : Ciência Moderna, 2013. Disponível em: <https://projecaociencia.com.br/index.php/Projecao4/article/view/497/463>. Acesso em: 26 out 2024.

FURLAN, Davi. Modelagem de Objetos Através da UML. São Paulo: Makron Books, 1998.

DE ARAÚJO, ALTEMIR FERNANDES et al. MODELAGEM ORIENTADA A OBJETOS APLICADA À ANÁLISE E AO PROJETO DE SISTEMA DE VENDAS. Disponível em: <https://www.maxpezzin.com.br/aulas/2_ESW_Tecnologia_OO/Modelagem_OO_sistema_vendas.pdf>. Acesso em: 02 nov 2024.

KERZNER, Harold. Project Management: A Systems Approach to. 2009.

CIN, Universidade Federal de Pernambuco. Programação Orientada a Objetos. Disponível em: <https://www.cin.ufpe.br/~acaj/PDF's%20uteis/Programação%20Orientada%20a%20Objetos.pdf>. Acesso em: 09 nov 2024.

KLS, Programação Orientada a Objetos. Disponível em: <https://s3.amazonaws.com/cm-kls-content/201801/INTERATIVAS_2_0/PROGRAMACAO_ORIENTADA_A_OBJETOS/U1/LIVRO_UNICO.pdf>. Acesso em: 09 nov 2024.

ENGSOFTMODERNA. Testes de Unidade. Disponível em: https://engsoftmoderna.info/cap8.html. Acesso em: 09 nov 2024.

MONTEIRO, Enio de Jesus Pontes et al. Recomendações para adoção de padrões de usabilidade e responsividade no desenvolvimento de aplicações web. **Brazilian Journal of Development**, v. 6, n. 5, p. 24790-24819, 2020.

NETO, Machado; JOSÉ, Olibario. **Usabilidade da interface de dispositivos móveis: heurísticas e diretrizes para o design**. 2013. Tese de Doutorado. Universidade de São Paulo.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. The Unified Modeling Language User Guide. 2. ed. Addison-Wesley, 2005.

FICHA DE CONTROLE DO PIM

Grupo Nº _____ Ano_____ Período: _____ Orientador _____

Tema:_____

Alunos:

RA	Nome	E-mail	Curso	Visto do aluno
N072369	Eduardo Salgado Juc	eduardo.juc@aluno.unip.br	**ADS**	
G773CB8	João Carlos Kenji Hecht Kaminobo	joao.kamino@aluno.unip.br	**ADS**	
G758692	Luis Felipe da Silva Dellu	luis.dellu@aluno.unip.br	**ADS**	
N240HA0	Marcos Vinícius dos Santos Moreira	marcos.mor@aluno.unip.br	**ADS**	
G7650H2	Matheus de Lima Moisés	matheus.moises@aluno.unip.br	**ADS**	
G7640C5	Ygor Melo Rodrigues da Silva	ygor.silva30@aluno.unip.br	**ADS**	

Registros:

Data do encontro	Observações
07/09/2024	Reunião para revisão e correção do projeto
14/09/2024	Reunião para separação dos membros para o desenvolvimento das aplicações
21/09/2024	Reunião para criação dos cronogramas
28/09/2024	Reunião para definição e prioridades de cada Sprint
05/10/2024	Início da Sprint 1
19/10/2024	Sprint review e fim da Sprint 1
20/10/2024	Início da Sprint 2
03/11/2024	Sprint review e fim da Sprint 2
09/11/2024	Revisão e correção da documentação do projeto