

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LEANDRO PIRES NUNES

**OBRACERTA: UMA APLICAÇÃO WEB PARA O CONTROLE DE CUSTOS NA
CONSTRUÇÃO CIVIL**

GUARAPUAVA

2025

LEANDRO PIRES NUNES

**OBRACERTA: UMA APLICAÇÃO WEB PARA O CONTROLE DE CUSTOS NA
CONSTRUÇÃO CIVIL**

ObraCerta: A Web Application for Cost Control on Civil Construction

Projeto de Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Roni Fabio Banaszewski

GUARAPUAVA

2025



[4.0 Internacional](https://creativecommons.org/licenses/by/4.0/)

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

RESUMO

Este projeto propõe o desenvolvimento do ObraCerta, uma aplicação web voltada ao controle financeiro de obras residenciais, com foco em construções de pequeno e médio porte. A proposta surge diante da necessidade de uma solução digital que auxilie cidadãos e pequenos construtores a manterem um planejamento financeiro eficiente, especialmente diante do cenário de constantes aumentos nos custos da construção civil. Serão disponibilizadas funcionalidades como o registro de despesas, definição de orçamentos, controle de pagamentos pendentes e a geração de alertas preventivos, proporcionando maior segurança na gestão da obra. Dessa forma, o ObraCerta busca promover organização, previsibilidade e economia ao usuário final, tornando o acompanhamento dos gastos mais acessível e confiável.

Palavras-chave: construção civil; controle financeiro; orçamentação; planejamento de obras.

ABSTRACT

This work proposes the development of ObraCerta, a web application aimed at financial management of residential construction projects, focusing on small and medium-sized works. The proposal arises from the need for a digital solution to assist individuals and small builders in maintaining efficient financial planning, especially considering the scenario of constant cost increases in the construction industry. Features such as expense tracking, budget definition, pending payment control, and generation of preventive alerts will be made available, providing greater security in project management. In this way, ObraCerta seeks to promote organization, predictability, and cost-efficiency for end users, making expense monitoring more accessible and reliable.

Keywords: civil construction; financial control; budgeting; project planning.

LISTA DE FIGURAS

Figura 1 – Método MoSCoW	16
Figura 2 – GitFlow	18
Figura 3 – Logotipo do ObraCerta	31
Figura 4 – Tela de Registro de usuário	33
Figura 5 – Tela de Autenticação	34
Figura 6 – Diagrama de Banco de Dados do ObraCerta	35

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	Interface de Programação de Aplicação, do inglês <i>Application Programming Interface</i>
CBIC	Câmara Brasileira da Indústria da Construção
CI/CD	Integração Contínua e Entrega Contínua, do inglês <i>Continuous Integration / Continuous Delivery</i>
CUB	Custo Unitário Básico
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	Ambiente de Desenvolvimento Integrado, do inglês <i>Integrated Development Environment</i>
MCMV	Minha Casa, Minha Vida
MVP	Produto Mínimo Viável, do inglês <i>Minimum Viable Product</i>
MVCC	Controle de Concorrência Multiversão, do inglês <i>Multiversion Concurrency Control</i>
NF-e	Nota Fiscal Eletrônica
PDF	Formato de Documento Portátil, do inglês <i>Portable Document Format</i>
PIB	Produto Interno Bruto
SEFAZ	Secretaria de Fazenda
SICRO	Sistema de Custos Referenciais de Obras
SINAPI	Sistema Nacional de Pesquisa de Custos e Índices da Construção Civil
SINDUSCON	Sindicato da Indústria da Construção Civil
SPA	Aplicação de Página Única, do inglês <i>Single Page Application</i>
TDD	Desenvolvimento Orientado a Testes, do inglês <i>Test-Driven Development</i>

SUMÁRIO

1	INTRODUÇÃO	8
1.1	Justificativa	9
1.2	Objetivos	10
1.2.1	Objetivo Geral	10
1.2.2	Objetivos específicos	10
1.3	Estrutura do trabalho	10
2	TRABALHOS RELACIONADOS	12
2.1	Preço da Obra	12
2.2	ConstruCalc	13
2.3	OrçaFascio	13
2.4	Considerações	14
3	REFERENCIAL TEÓRICO	15
3.1	Processo de Desenvolvimento	15
3.1.1	Scrum	15
3.1.2	Kanban	15
3.1.3	Moscow	16
3.1.4	Desenvolvimento orientado a testes	16
3.2	Ferramentas de Desenvolvimento	17
3.2.1	Git	17
3.2.2	GitHub	17
3.2.3	GitFlow	18
3.2.4	GitHub Projects	18
3.2.5	Docker	19
3.2.6	DbDiagram	19
3.2.7	Monorepo	19
3.3	Tecnologias ao lado do cliente	20
3.3.1	Angular	20
3.3.2	Tailwind	20
3.3.3	Daisyui	21
3.4	Tecnologias do lado do servidor	21

3.4.1	Spring Boot	21
3.4.2	PostgreSQL	21
3.5	Considerações	21
4	MATERIAIS E MÉTODOS	23
4.1	Materiais	23
4.2	Métodos	23
4.3	Considerações	24
5	ANÁLISE DO SISTEMA	25
5.1	Perfis de Usuários	25
5.2	Funcionalidades Essenciais	25
5.2.1	Cadastro de Usuário (HU01)	26
5.2.2	Autenticação (Login/Logout) (HU02)	26
5.2.3	Cadastro de Despesas (HU03)	26
5.2.4	Edição e Exclusão de Despesas (HU04)	27
5.2.5	Visualização e Filtros de Gastos (HU05)	27
5.2.6	Relatórios e Gráficos de Gastos (HU06)	27
5.2.7	Orçamento Inicial e Comparação com Gastos (HU07)	28
5.2.8	Notificações de Pagamentos Pendentes (HU08)	28
5.2.9	Integração com Nota Fiscal Eletrônica (NF-e) (HU09)	28
5.3	Público-alvo	29
5.4	Considerações	29
6	PROJETO DO SISTEMA	30
6.1	Identidade Visual	30
6.1.1	Cores Principais	30
6.1.2	Tipografia	31
6.1.3	Logotipo	31
6.2	Planejamento da Sprint 0 e da Sprint 1	31
6.2.1	Preparação de Ambiente (Sprint 0)	31
6.2.2	Cadastro e Login (Sprint 1)	32
6.3	Prototipação de Telas	32
6.3.1	Registro de usuário	33
6.3.2	Autenticação	33

6.4	Modelagem do Banco de Dados	34
6.5	Considerações	36
7	CONCLUSÃO	37
	REFERÊNCIAS	38
	 APÊNDICES	 40
	APÊNDICE A – MODELO DE DADOS DETALHADO	42
	A.1–Modelo de Dados	42
A.1.1	Tabela <code>users</code>	42
A.1.2	Tabela <code>suppliers</code>	42
A.1.3	Tabela <code>expense_categories</code>	43
A.1.4	Tabela <code>budgets</code>	43
A.1.5	Tabela <code>expenses</code>	43
A.1.6	Tabela <code>pending_payments</code>	44
A.1.7	Tabela <code>notifications</code>	44
	APÊNDICE B – FUNCIONALIDADES FUTURAS DO SISTEMA OBRA- CERTA	 46
	B.1–Funcionalidades Should Have – Deveria Ter	46
B.1.1	Solicitação de Orçamentos Automatizada	46
B.1.2	Localização de Fornecedores Próximos	46
	B.2–Funcionalidades Could Have – Poderia Ter	47
B.2.1	Banco de Preços Colaborativo	47
B.2.2	Cadastro e Avaliação de Fornecedores	47
	B.3–Funcionalidades Won't Have This Time – Não Terá Desta Vez	47
B.3.1	Integração com WhatsApp dos Fornecedores	47
B.3.2	Atualização Automática de Preços via NF-e	48

1 INTRODUÇÃO

O mercado imobiliário brasileiro destaca-se como um setor fundamental para a economia nacional, com grande capacidade de geração de empregos e impacto direto no desenvolvimento urbano. Segundo dados da Câmara Brasileira da Indústria da Construção (CBIC), em 2024 foram lançadas 383.483 novas unidades habitacionais no país, representando um aumento de 18,6% em relação ao ano anterior e estabelecendo um novo recorde desde o início da série histórica em 2017. As vendas também apresentaram um desempenho expressivo, totalizando 400.547 unidades comercializadas, o que equivale a um crescimento de 20,9% em comparação a 2023 (Câmara Brasileira da Indústria da Construção (CBIC), 2024).

Um dos principais impulsionadores desse crescimento foi o programa Minha Casa, Minha Vida (MCMV), que teve um papel fundamental na expansão do mercado imobiliário nacional. Em 2024, os lançamentos dentro do programa cresceram 44,2% na comparação com 2023, totalizando 187.356 unidades; enquanto as vendas tiveram alta de 43,3%, chegando a 168.773 unidades. No ano, o MCMV teve uma participação de 48,8% do total de lançamentos e 42,1% das vendas no país (Câmara Brasileira da Indústria da Construção (CBIC), 2025).

Esses dados evidenciam uma forte demanda habitacional no país, especialmente concentrada nas regiões Sudeste e Nordeste, onde houve maior número de lançamentos e vendas. Além disso, o estoque disponível de imóveis recuou 7,8% no quarto trimestre de 2024, totalizando cerca de 292 mil unidades (Câmara Brasileira da Indústria da Construção (CBIC), 2024). Este cenário indica um mercado imobiliário aquecido, com demanda crescente por novas unidades e também pela reforma e ampliação de imóveis já existentes.

Dentro desse contexto, a construção civil assume papel estratégico, representando cerca de 6,8% do Produto Interno Bruto (PIB) brasileiro e empregando direta e indiretamente mais de 7 milhões de trabalhadores (Ministério do Trabalho, 2023). Muitos brasileiros optam por construir suas próprias residências ou reformar imóveis já adquiridos como alternativa aos preços elevados encontrados em unidades prontas no mercado. Contudo, construir ou reformar não é tarefa simples e envolve uma gestão financeira complexa e desafiadora.

Indicadores econômicos como o Custo Unitário Básico (CUB), divulgado regularmente pelo Sindicato da Indústria da Construção Civil (Sinduscon), e o Sistema Nacional de Pesquisa de Custos e Índices da Construção Civil (SINAPI), mantido pela Caixa Econômica Federal em conjunto com o Instituto Brasileiro de Geografia e Estatística (IBGE), demonstram uma elevação contínua nos preços relacionados à construção civil. De acordo com o SINAPI, em março de 2025, o custo médio para construir no Brasil foi de R\$1.810,25 por metro quadrado. Em comparação, esse valor era de R\$1.729,25 em março de 2024, e de apenas R\$918,95 em março de 2015, representando um aumento acumulado de 96,99% em dez anos (IBGE - Instituto Brasileiro de Geografia e Estatística, 2025).

A variação significativa desses custos pode ser atribuída a diferentes fatores, tais como a qualidade e tipo dos materiais escolhidos, a mão de obra especializada contratada, a localiza-

ção geográfica da obra, condições específicas do terreno e a sazonalidade do mercado. Itens básicos, como tijolos, cimento, areia, ferramentas e acabamentos, frequentemente apresentam diferenças consideráveis de preços entre diversos fornecedores. Não é incomum que materiais sejam adquiridos diretamente de fábricas ou distribuidores específicos, às vezes em localidades distantes, enquanto itens como escoras e andaimes sejam alugados separadamente, e acabamentos como portas, janelas e vidros sejam adquiridos em fornecedores especializados. Essa fragmentação exige uma atenção redobrada na gestão financeira, especialmente quando a responsabilidade recai sobre proprietários que muitas vezes não possuem conhecimento técnico suficiente na área (LIMMER, 1997).

Na prática, boa parte das obras residenciais ainda é executada de maneira artesanal, com planejamentos informais ou improvisados, sem controle rigoroso de prazos ou orçamento detalhado. Conforme destaca Limmer (1997), a ausência de uma gestão financeira adequada resulta frequentemente em desperdícios, atrasos e até paralisações definitivas das construções. Muitas pessoas ainda recorrem a métodos precários, como anotações em papéis avulsos ou planilhas eletrônicas improvisadas, sujeitas a perdas, erros ou inconsistências que comprometem o controle efetivo das despesas.

Diante deste cenário, caracterizado pela crescente demanda habitacional e pelos elevados custos envolvidos, torna-se fundamental a utilização de ferramentas tecnológicas acessíveis e eficientes para o controle financeiro das obras. Dessa forma, propõe-se neste trabalho o desenvolvimento de uma aplicação web voltada ao planejamento, organização e controle financeiro das despesas associadas à construção civil, buscando proporcionar maior segurança, transparência e eficiência na gestão dos recursos destinados à construção ou reforma residencial.

1.1 Justificativa

Atualmente a área da construção civil enfrenta uma elevação significativa dos custos, o que dificulta cada vez mais a realização de obras, sejam elas de pequeno ou médio porte, tanto em residências já adquiridas quanto naquelas que estão prestes a serem construídas a partir do início.

Além disso, em obras em andamento, muitas vezes o controle financeiro é inexistente, seja na aquisição de materiais, no pagamento de salários, ou até no desconhecimento de taxas legais obrigatórias, o que contribui para que muitas dessas construções não sejam finalizadas conforme o planejado.

Para contribuir com esse controle, é proposta a criação de uma aplicação web voltada à gestão orçamentária da obra. Com ela, o usuário poderá elaborar e acompanhar o orçamento da obra, registrar despesas, visualizar a progressão da obra com base nos valores gastos, importar materiais adquiridos a partir de suas notas fiscais e ainda simular novos orçamentos

com base em dados de construção anteriores cadastrados, promovendo maior entendimento sobre o mercado e custos com obras similares.

Essa aplicação visa auxiliar os usuários a realizarem obras de forma mais planejada, minimizando perdas que muitas vezes são decorrentes da falta de conhecimento sobre o andamento de sua obra. Dessa forma, espera-se que mais obras possam ser finalizadas conforme o previsto, com base em um acompanhamento preciso e contínuo do progresso.

1.2 Objetivos

Nesta seção serão apresentados o objetivo geral e os objetivos específicos do presente trabalho.

1.2.1 Objetivo Geral

Desenvolver uma aplicação web para controle financeiro de obras de pequeno e médio porte, proporcionando aos usuários melhor controle dos valores gastos, valores a pagar e o acompanhamento do progresso financeiro da construção.

1.2.2 Objetivos específicos

- Implementar um módulo para o cadastro e categorização de despesas relacionadas à obra, incluindo materiais, mão de obra e serviços.
- Desenvolver uma funcionalidade que permita a criação e o gerenciamento do orçamento inicial da obra, com a definição de valores previstos por categoria de despesa.
- Integrar a aplicação com a API da Sefaz para possibilitar o registro automático de despesas por meio da chave de acesso da Nota Fiscal Eletrônica (NF-e).
- Criar mecanismos de alerta que notifiquem o usuário sobre o risco de ultrapassagem do orçamento previsto, com base nos gastos registrados.
- Disponibilizar relatórios e indicadores visuais que permitam a comparação entre os valores orçados e os gastos reais, facilitando a análise do progresso financeiro da obra.

1.3 Estrutura do trabalho

Este Trabalho de Conclusão de Curso está estruturado em sete capítulos, conforme descrito a seguir:

O **Capítulo 1** apresenta a introdução ao tema, contextualizando a importância do controle financeiro na construção civil e justificando a necessidade do desenvolvimento de uma solução tecnológica para esse fim. Também são definidos os objetivos geral e específicos do projeto.

O **Capítulo 2** realiza uma revisão de trabalhos relacionados, descrevendo e analisando aplicações existentes voltadas ao orçamento e controle de obras, destacando suas funcionalidades, limitações e público-alvo.

O **Capítulo 3** trata do referencial teórico, abordando as metodologias de desenvolvimento adotadas e as tecnologias empregadas tanto no lado do cliente quanto no servidor, assim como as ferramentas de apoio utilizadas ao longo do projeto.

O **Capítulo 4** descreve os materiais e métodos utilizados, detalhando o ambiente de desenvolvimento, o planejamento das tarefas com base em metodologias ágeis e a organização das etapas de construção do sistema.

O **Capítulo 5** apresenta a análise do sistema desenvolvido, incluindo os perfis de usuários, funcionalidades essenciais organizadas como histórias de usuário e a definição do público-alvo da aplicação.

O **Capítulo 6** contempla o projeto do sistema, com a definição da identidade visual, os protótipos das interfaces e a modelagem do banco de dados que sustenta as funcionalidades implementadas.

Por fim, o **Capítulo 7** apresenta as conclusões do trabalho, destacando os resultados alcançados, as contribuições da aplicação desenvolvida e possibilidades de aprimoramento e expansão futura da solução.

2 TRABALHOS RELACIONADOS

Atualmente, existem diversos aplicativos e sistemas voltados à elaboração de orçamentos, com diferentes funcionalidades e formas de funcionamento. No entanto, muitos desses sistemas são voltados a orçamentos genéricos, ou para orçamentos de obras de grande porte, onde poucos atendem de maneira específica e gratuita ao nicho de obras de pequeno e médio porte. Nesta seção, serão apresentadas algumas soluções utilizadas para orçamentação de obras e controle dos seus custos, que compartilham características semelhantes à proposta deste projeto.

Vale destacar que existem várias soluções no mercado como o OrçaFascio, ConstruCalc, Preço da Obra, Sienge e o Seobra. Destaca-se o Sienge, que, como dito em seu site, é um software onde 72 entre as 100 maiores construtoras do Brasil estão em sua carteira de clientes (Sienge, 2025), tornando-o o principal sistema do mercado, e o Seobra, que também é um sistema de orçamentação que faz uso da base de dados do SINAPI, onde seu foco é para obras públicas.

Embora essas duas soluções sejam amplamente reconhecidas, elas não serão aprofundadas nesta pesquisa. No caso do Sienge, não foi possível obter acesso direto à plataforma, dificultando uma análise prática de suas funcionalidades. Já no caso do Seobra, o acesso foi viável; no entanto, optou-se por dar ênfase ao OrçaFascio, por apresentar funcionalidades mais alinhadas e mais responsivas.

Nessa seção, será abordado e analisado algumas soluções como o OrçaFascio, ConstruCalc e Preço da Obra, que apresentam algumas funcionalidades que apresentam maior coesão.

2.1 Preço da Obra

O Preço da Obra é um software que agiliza orçamentos e propostas com modelos prontos e recursos inteligentes, ajudando construtores a vender mais com rapidez e profissionalismo (Preço Da Obra, 2025). Utilizando a base do SINAPI, eles mantêm uma base de dados de itens, à qual atualizam regularmente e a utilizam como base para a realização de orçamentos de forma rápida para seus modelos prontos de construções.

- **Pontos Positivos:**

- Integração com o SINAPI, garantindo preços atualizados de materiais e serviços.
- Interface Intuitiva, facilitando a navegação e a criação de orçamentos
- Disponibilização de 3 formas de orçamentação, com modelos básicos e avançados.

- Estrutura detalhada de orçamentos, organizando as etapas da obra e os insumos utilizados.

- **Pontos Negativos:**

- Instabilidade e Lentidão em determinadas áreas do sistema, como na sessão de assinaturas.
- Funcionalidade limitada na versão gratuita – Após o uso de um crédito de teste, é necessário contratar um plano pago.
- Foco em profissionais da área, dificultando o uso por usuários leigos ou autônomos sem formação.

2.2 Construcalc

Outro exemplo é o Construcalc, desenvolvido pela empresa Tresium, onde o objetivo deste aplicativo é realizar cálculos precisos de materiais de construção na palma da sua mão, de maneira simples e rápida (Tresium, 2025). O seu foco é no cálculo das áreas de determinadas partes da obra, tornando-as mais precisas. Ao final de cada cálculo, seja ele de uma parede ou de uma laje, ele mostra a quantidade de materiais necessários e dá a possibilidade de salvá-los em uma lista, podendo exportá-los em PDF.

- **Pontos Positivos:**

- Oferece em seus calculo a quantidade de materiais necessários para construção da área selecionada.
- Divisão para cálculos, onde separa em cálculos diferentes para áreas como paredes, lajes, telhas, entre outras áreas.

- **Pontos Negativos:**

- Quantidade alta de anúncios na versão gratuita.
- Foco apenas no calculo dos materiais, sem apresentar os valores em custo das áreas.
- Para realizar o calculo da obra completamente, é necessário adicionar cada calculo individual em uma lista.

2.3 OrçaFascio

Hoje existem vários sistemas para orçamentação utilizados por pequenos e grandes empreendedores, um desses sistemas é o OrçaFascio. O OrçaFascio é um sistema que tem como

principal foco a realização de orçamentos, como dito em seu site, hoje ele oferece integração com mais de 20 tabelas e 85 mil novas composições integradas, sendo a principal plataforma de orçamentos para licitações de obras no Brasil (OrçaFascio, 2025). Além de, claro, possuir um grande número de bases além do SINAPI para apresentar os valores em insumos, ele traz a possibilidade de realizar o acompanhamento da obra e um registro diário sobre o que ocorre no dia a dia dela.

- **Pontos Positivos:**

- Integração com bases do SINAPI, SICRO, entre outras.
- Uso de inteligência artificial para sugestão de composições com base em orçamentos anteriores.
- Possibilidade de geração de relatórios
- App móvel com modo offline

- **Pontos Negativos:**

- Plano gratuito limitado. Funções essenciais necessitam de assinatura.
- Curva de aprendizado alta
- Foco em usuários profissionais do setor.

2.4 Considerações

Hoje o principal foco do mercado é o desenvolvimento de soluções para obras de médio a grande porte, trazendo ferramentas para engenheiros, arquitetos e empreiteiras, a exemplo de sistemas como o Sienge, OrçaFascio e Seobra, o que acaba deixando de lado usuários que estão em busca de uma solução para ajudar nas suas obras de pequeno e médio porte.

Aplicativos como o Preço da Obra e o ConstruCalc conseguem ser mais acessíveis, pois trazem uma maneira de orçamentação rápida, porém que ainda traz detalhes que podem complicar a vida do cidadão que deseja um controle simples e completo das suas obras, para, no mais, realizar o acompanhamento da obra.

Com isso, se faz necessário o desenvolvimento do ObraCerta, como uma solução acessível, trazendo praticidade na gestão de obras.

3 REFERENCIAL TEÓRICO

Este capítulo aborda os principais conceitos, metodologias e tecnologias utilizados no desenvolvimento de sistemas web modernos. São exploradas abordagens de gerenciamento de projetos, práticas de desenvolvimento ágil, técnicas de versionamento de código e integração contínua, bem como frameworks e ferramentas amplamente adotados tanto no front-end quanto no back-end. O objetivo é apresentar uma base teórica sólida que sustente as decisões técnicas adotadas nos capítulos seguintes, proporcionando uma compreensão abrangente do ecossistema tecnológico atual no contexto do desenvolvimento de aplicações web.

3.1 Processo de Desenvolvimento

O processo de desenvolvimento de um sistema abrange as várias etapas do seu desenvolvimento, desde a sua concepção até a sua finalização. Para isso, o planejamento de todo esse processo faz com que seja necessário o uso de metodologias e técnicas, onde elas podem garantir a organização das tarefas, assim como a entrega contínua dentro do prazo estipulado.

3.1.1 Scrum

O Scrum é um framework ágil amplamente utilizado para gerenciamento de projetos complexos. Por se tratar de um framework leve, o Scrum permite que equipes e organizações gerem valor de forma incremental, por meio de soluções adaptativas para problemas complexos (Schwaber, Ken and Sutherland, Jeff, 2020).

Sua estrutura baseia-se na divisão do projeto em ciclos curtos e iterativos chamados sprints, geralmente com duração semanal ou quinzenal. Cada sprint inclui reuniões de planejamento, acompanhamento e revisão, possibilitando a priorização de entregas com base no valor gerado, a experimentação contínua de soluções e o ajuste do desenvolvimento com base no feedback recebido ao longo do processo.

3.1.2 Kanban

O Kanban é uma metodologia visual de gerenciamento de tarefas que busca otimizar o fluxo de trabalho e aumentar a eficiência operacional. Seu foco está na limitação do número de atividades em andamento simultaneamente, o que contribui para a redução do multitarefa e promove maior concentração na conclusão de cada tarefa individualmente (PersonalKanban, 2025).

A técnica utiliza quadros e cartões para representar visualmente o andamento das tarefas, distribuindo-as em colunas que indicam os diferentes estágios do fluxo de trabalho, como "A

Fazer", "Em Progresso" e "Concluído". Essa abordagem permite o monitoramento contínuo das atividades, a identificação de gargalos e o ajuste dinâmico de prioridades, conforme a demanda e o avanço do projeto.

3.1.3 Moscow

O método MoSCoW é uma técnica amplamente utilizada para a priorização de requisitos em projetos, especialmente em contextos ágeis. O seu nome é um acrônimo, derivado da primeira letra de cada uma das quatro categorias. Must Have (Deve Ter), Should Have (Deveria Ter), Could Have (Poderia Ter) e Won't Have (Não Terá). A letra "o" no acrônimo foi adicionada para tornar a palavra pronunciável (Sebrae, 2023).

Essa abordagem permite que equipes definam de forma clara o que é essencial para o sucesso de uma entrega, o que é desejável, mas não imprescindível, e o que pode ser descartado ou postergado. Assim, o método contribui para a gestão eficiente do escopo e a alocação de recursos com base em critérios de valor e viabilidade.

A Figura 1 ilustra graficamente as categorias definidas pelo método MoSCoW, destacando sua aplicação prática na priorização de funcionalidades em projetos de desenvolvimento.



Figura 1 – Método MoSCoW

Fonte: (Sebrae, 2023).

3.1.4 Desenvolvimento orientado a testes

O Test-Driven Development (Desenvolvimento Orientado a Testes) (TDD), é uma metodologia de desenvolvimento de software que tem como principal foco o desenvolvimento de testes antes da implementação de funcionalidades. Em vez de escrever o código primeiro e, depois, criar um teste para validá-lo, o desenvolvimento orientado por testes determina que se escreva o teste primeiro e, em seguida, implemente as mudanças no código até que ele passe no teste que já foi escrito (Steinfeld, Grant, 2023). A escrita do teste segue o processo de Red, Green e Refactor, onde no (Red) o primeiro teste escrito irá falhar; após isso, no (Green) é feita

a escrita de um teste para passar no anterior, e em seguida, no (Refactor) o teste é refatorado para deixar a sua estrutura mais legível.

3.2 Ferramentas de Desenvolvimento

As ferramentas de desenvolvimento desempenham um papel fundamental na produtividade e na organização dos projetos de software. Elas contribuem para a automação de tarefas repetitivas, o controle de versões, a integração de sistemas e a padronização dos processos de trabalho, proporcionando maior eficiência, rastreabilidade e qualidade no ciclo de desenvolvimento.

A seguir, são apresentadas algumas das principais ferramentas frequentemente utilizadas em ambientes de desenvolvimento modernos, destacando suas características e benefícios técnicos.

3.2.1 Git

O Git é um sistema de controle de versão para gerenciamento de repositórios de código-fonte. Ele é um projeto de código aberto maduro e com manutenção ativa, desenvolvido em 2005 por Linus Torvalds, o famoso criador do kernel do sistema operacional Linux (Atlassian, 2025a). Ele permite aos desenvolvedores ter um histórico total de todas as mudanças que ocorreram no código-fonte, indicando onde essa mudança foi realizada, quando ela foi realizada e por quem foi realizada, permitindo um controle total em caso de erros em versões, já que permite mudanças de versões, assim como a clonagem de ramificações, que podem ser unidas para a geração de uma nova versão.

3.2.2 GitHub

O GitHub é uma plataforma baseada em nuvem voltada à hospedagem e ao gerenciamento de repositórios que utilizam o sistema de controle de versões Git. Ela permite armazenar e versionar código-fonte, bem como facilitar a colaboração entre desenvolvedores em projetos compartilhados.

Além das funcionalidades básicas de controle de versões, o GitHub oferece diversos recursos de gerenciamento de projetos, como o controle de pull requests, o acompanhamento de issues e a integração com o GitHub Actions, que possibilita a automação de fluxos de trabalho relacionados à integração e entrega contínuas (CI/CD). A plataforma também oferece suporte à documentação de projetos por meio de wikis.

3.2.3 GitFlow

O GitFlow é uma estratégia de ramificação (branching model) que organiza o fluxo de trabalho com Git por meio de uma estrutura padronizada, facilitando o controle e a manutenção do código-fonte em projetos colaborativos. Essa abordagem define diferentes tipos de ramificações, cada uma com um propósito específico dentro do ciclo de desenvolvimento (Atlassian, 2025b).

As principais branches utilizadas no GitFlow são:

- **main**: armazena a versão estável e pronta para produção da aplicação.
- **develop**: utilizada para integração contínua de novas funcionalidades em desenvolvimento.
- **feature**: criada a partir da branch *develop*, serve para o desenvolvimento de funcionalidades específicas.
- **release**: derivada da *develop*, é usada para preparar uma nova versão antes de ser integrada à *main*.
- **hotfix**: criada a partir da *main*, é destinada à correção de erros críticos em produção.

Esse modelo favorece a organização do processo de versionamento, reduz conflitos entre mudanças simultâneas e contribui para a entrega de versões mais estáveis e confiáveis. A Figura 2 ilustra a estrutura do GitFlow e a interação entre suas ramificações.

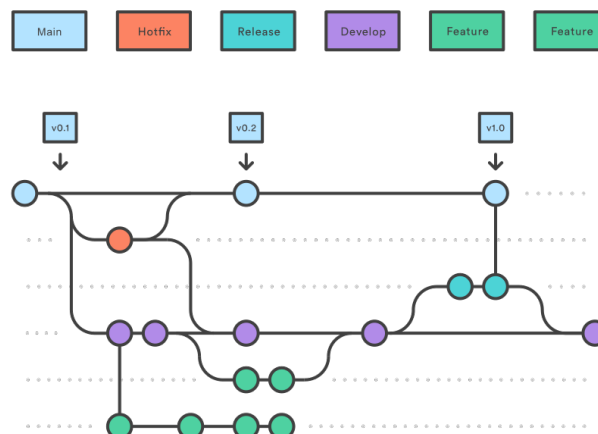


Figura 2 – GitFlow

Fonte: (Atlassian, 2025b).

3.2.4 GitHub Projects

O GitHub Projects é uma ferramenta de gerenciamento de projetos integrada à plataforma GitHub. Trata-se de uma coleção adaptável de itens que podem ser visualizados em

diferentes formatos, como tabelas, quadros no estilo Kanban ou roteiros, permitindo o acompanhamento dinâmico do progresso das atividades (GitHub - GitHub Docs, 2025).

Sua integração nativa com os repositórios Git facilita o gerenciamento de tarefas, permitindo a vinculação direta de issues, pull requests, etiquetas, prazos e responsáveis. Essa abordagem contribui para a organização do fluxo de trabalho, promovendo maior controle e visibilidade sobre o desenvolvimento do projeto.

3.2.5 Docker

O Docker é uma plataforma aberta para desenvolvimento, distribuição e execução de aplicações em contêineres. Ele permite separar as aplicações de seu ambiente de execução, garantindo entrega rápida, isolada e previsível em diferentes sistemas (Docker, 2025).

Ao encapsular dependências e configurações em contêineres padronizados, o Docker facilita o versionamento, a escalabilidade e o gerenciamento de ambientes. Além disso, permite a criação de ambientes distintos para desenvolvimento, testes e produção, contribuindo para a automação de processos e a redução de falhas relacionadas a diferenças de configuração entre máquinas.

3.2.6 DbDiagram

O DbDiagram é uma ferramenta online utilizada para criação de diagramas entidade-relacionamento (ER) de bancos de dados. Ela permite a modelagem visual de tabelas, colunas e relacionamentos, facilitando a compreensão da estrutura de dados. O DbDiagram permite desenhar diagramas relacionais de forma rápida utilizando uma linguagem de marcação própria baseada em DBML (Database Markup Language) (dbdiagram, 2025), assim possibilitando a geração rápida de diagramas e a exportação do modelo para linguagens SQL. Sua praticidade e acessibilidade a tornam uma solução amplamente adotada em ambientes de desenvolvimento ágil, especialmente nas fases iniciais de projeto e validação de modelos relacionais.

3.2.7 Monorepo

Um monorepo (abreviação de monolithic repository) é uma estratégia que consiste em manter múltiplos projetos relacionados em um único repositório, com estruturas e dependências bem definidas entre eles (Monorepo, 2025). Essa abordagem permite que componentes como front-end e back-end compartilhem o mesmo ambiente de versionamento, facilitando a padronização da organização do código.

Entre os principais benefícios do uso de monorepos estão a simplificação da integração contínua, a redução do tempo de configuração de novos projetos e a centralização da gestão de

dependências, testes e pipelines. Essa estratégia tem sido amplamente adotada por grandes organizações que buscam maior consistência e escalabilidade no desenvolvimento de sistemas complexos.

3.3 Tecnologias ao lado do cliente

As tecnologias do lado do cliente são responsáveis pela interface com a qual o usuário interage diretamente. Elas definem os elementos visuais e comportamentais da aplicação, como botões, menus e formulários, promovendo uma experiência de navegação fluida e intuitiva. Essas interações visuais são possíveis graças ao HTML, que visa estruturar o conteúdo das páginas; o CSS, que visa definir o estilo visual e o layout dos elementos; e o TypeScript, uma linguagem que adiciona tipagem estática e facilita a organização e criação de eventos. A seguir, são apresentadas as principais ferramentas utilizadas no desenvolvimento de interfaces web modernas.

3.3.1 Angular

O Angular é um framework front-end de código aberto, desenvolvido e mantido pelo Google, que tem como base a linguagem TypeScript. Ele é amplamente utilizado na construção de aplicações web dinâmicas e interativas, especialmente no modelo de página única, conhecido como Single Page Application (SPA).

Sua arquitetura é orientada a componentes, o que favorece a reutilização de código e a modularização da interface. O Angular oferece diversos recursos integrados, como data binding, sistema de rotas, injeção de dependência e ferramentas de teste, o que o torna uma solução robusta e escalável para projetos de diferentes portes.

3.3.2 Tailwind

O Tailwind CSS é um framework de código aberto para estilização de interfaces web, baseado em classes utilitárias aplicadas diretamente no HTML. Diferentemente de frameworks como o Bootstrap, ele não oferece componentes prontos, mas sim uma abordagem de baixo nível que permite maior flexibilidade na composição do design. Essa característica contribui para a redução da complexidade na implementação e evita a repetição de estruturas visuais entre projetos (Tailwind, 2025).

3.3.3 Daisyui

O DaisyUI é uma biblioteca de componentes visuais construída sobre o Tailwind CSS. Seu principal objetivo é facilitar a criação de interfaces modernas por meio de elementos pré-configurados e personalizáveis, como botões, formulários, modais e menus. A biblioteca mantém a flexibilidade do Tailwind, ao mesmo tempo em que acelera o desenvolvimento ao disponibilizar componentes prontos com suporte a responsividade e estilos consistentes.

3.4 Tecnologias do lado do servidor

As tecnologias do lado do servidor são responsáveis pelo processamento das requisições originadas na interface do usuário. Elas executam a lógica da aplicação, acessam bancos de dados e retornam respostas por meio de APIs. A seguir, são apresentadas as principais ferramentas adotadas nesse contexto, com destaque para suas características técnicas e relevância no desenvolvimento web.

3.4.1 Spring Boot

O Spring Boot é uma ferramenta do ecossistema Spring, desenvolvida para a linguagem Java, que tem como objetivo simplificar a criação de aplicações web e microserviços. Entre suas principais características estão a inicialização automatizada de projetos, o gerenciamento de dependências por convenção e o suporte nativo a recursos como injeção de dependência, segurança e persistência de dados. Essas funcionalidades tornam o Spring Boot uma opção robusta e produtiva para o desenvolvimento de aplicações escaláveis e modulares.

3.4.2 PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados relacional de código aberto, sendo um dos mais utilizados no mercado devido à sua confiabilidade, desempenho e escalabilidade. Seu desenvolvimento teve início em 1986 e, desde então, segue em constante evolução, com recursos como transações ACID, extensibilidade com tipos e funções personalizadas, controle de concorrência multiversão (MVCC) e replicação nativa.

3.5 Considerações

As tecnologias e ferramentas abordadas neste capítulo representam soluções amplamente consolidadas no mercado de desenvolvimento de software. A adoção de frameworks como Angular e Spring Boot, aliada ao uso de metodologias ágeis como o Scrum, demonstra

uma orientação a práticas modernas, seguras e validadas por comunidades técnicas experientes. Essas escolhas contribuem para um ambiente de desenvolvimento mais estável, organizado e eficiente, favorecendo tanto a manutenção da aplicação quanto a experiência do usuário final.

4 MATERIAIS E MÉTODOS

4.1 Materiais

Nesta fase de projeto, foram definidos os materiais e ferramentas que serão utilizados na implementação da aplicação proposta. As tecnologias adotadas já foram descritas no capítulo de Referencial Teórico.

A prototipação das interfaces foi realizada com o Figma, permitindo a simulação do fluxo de navegação e da experiência do usuário. A modelagem do banco de dados foi feita com o DbDiagram, possibilitando a visualização dos relacionamentos e estrutura lógica das tabelas.

O controle de versão será realizado com Git, utilizando o fluxo GitFlow para organização das ramificações, e o repositório será hospedado no GitHub, onde também será utilizado o GitHub Projects para o gerenciamento de tarefas em formato Kanban.

O front-end será implementado com Angular e o back-end com Spring Boot, utilizando PostgreSQL como banco de dados. A arquitetura será preparada para funcionar com contêineres Docker, permitindo simular diferentes ambientes. A estrutura do projeto será organizada como monorepositório (monorepo), centralizando front-end e back-end em um único repositório.

4.2 Métodos

O desenvolvimento da aplicação seguirá as práticas de gerenciamento de projetos com o uso da metodologia Scrum, adaptada no contexto do desenvolvimento de forma individual. O orientador terá o papel de Product Owner e Scrum Master, ou seja, terá o papel na definição dos requisitos e pela orientação técnica e metodológica. O aluno terá o papel de Developer, responsável pela implementação das funcionalidades planejadas na Sprint.

A organização das atividades será definida em sprints quinzenais, realizando reuniões de planejamento no início de cada ciclo, onde serão definidas as tarefas prioritárias. Esse período quinzenal vem em conformidade com a necessidade de adaptação das tarefas durante as sprints, onde podem surgir demandas que podem ter alterações de prioridades e prazos. O acompanhamento e registro de atividades será realizado via GitHub Projects, fazendo a alusão a um quadro Kanban para definir as prioridades e indicar o andamento das atividades.

As tarefas vão ser alocadas em colunas que representam o seu estado dentro da sprint através do GitHub Projects, e serão organizadas da seguinte forma:

- **Backlog** (Pendências): Tarefas que podem ser alocadas a uma Sprint.
- **To do** (A Fazer): Tarefas que estão alocadas a Sprint, mas não foram iniciadas.
- **In Progress** (Em Progresso): Tarefas que estão sendo desenvolvidas na Sprint.
- **Review** (Revisão): Tarefas que estão em etapa de testes, e aguardando revisão.

- **Done** (Concluídas): Tarefas que foram finalizadas durante a Sprint e estão prontas para serem entregues.

Já para a divisão da construção e desenvolvimento do projeto, no Docker o desenvolvimento da aplicação será isolado em containers distintos, permitindo maior controle sobre as dependências de cada frente do sistema, onde a divisão será feita em 3 containers.

- **Backend:** Responsavel pela execução do Spring Boot, e a API de comunicação desenvolvida nele.
- **Frontend:** Responsavel pela execução do Angular, e pela parte visual da aplicação.
- **Banco de Dados:** Responsavel pela execução do PostgreSQL, a parte dos dados da aplicação

Essa separação deve permitir um ambiente de desenvolvimento separado para cada uma das ferramentas, onde cada um não irá interferir em dependências isoladas, além de permitir simulação de ambientes de desenvolvimento, testes e produção.

Em conjunto, deve ser realizada a integração de pipelines de execução e validação de testes via Github Actions, onde assim será permitido que a cada envio de commits em momentos como na realização de pull requests, sejam realizados testes que garantam que os testes unitários e automatizados estão funcionando normalmente, e também que o código-fonte esteja seguindo as boas práticas de implementação.

A organização do código-fonte nos repositórios Git deve ser adotada utilizando a abordagem monorepo, onde tanto o back-end quanto o front-end ficarão no mesmo repositório, garantindo maior facilidade de gestão e manutenção do código-fonte, além de sua implementação e integração, garantindo um ambiente de desenvolvimento que funcione independente do local em que esteja, auxiliando na manutenção e proporcionando facilidade em futuras integrações.

4.3 Considerações

Os materiais e métodos apresentados neste capítulo foram definidos com base nas boas práticas de desenvolvimento de software, buscando garantir organização, clareza no processo e viabilidade de execução do projeto. A adoção dessas abordagens visa facilitar o planejamento, permitir testes e ajustes ao longo do desenvolvimento, além de contribuir para a manutenção e evolução futura do sistema.

5 ANÁLISE DO SISTEMA

Este trabalho propõe o desenvolvimento de uma aplicação web denominada **ObraCerta**, com o objetivo de auxiliar no planejamento e controle financeiro de obras residenciais, especialmente construções autogeridas, ou seja, realizadas diretamente por proprietários sem acompanhamento profissional contínuo.

A aplicação busca oferecer uma solução acessível, prática e digital para que os usuários possam registrar todas as despesas da obra, acompanhar o orçamento previsto em comparação com os gastos reais, receber alertas preventivos de estouro de orçamento e gerar relatórios detalhados sobre o andamento financeiro da construção.

A aplicação web proposta será desenvolvida como um protótipo funcional (Produto Mínimo Viável – MVP), com foco nas funcionalidades essenciais para o gerenciamento orçamentário da construção civil.

5.1 Perfis de Usuários

A aplicação contempla dois perfis principais de usuários, cada um com funções específicas:

- **Usuário Comum:** Responsável pelo cadastro e gerenciamento das despesas da obra. Pode registrar gastos, criar orçamentos, visualizar relatórios e configurar alertas de pagamentos pendentes.
- **Administrador:** Responsável pela gestão de usuários, manutenção e monitoramento do sistema. Possui acesso a métricas, relatórios e configurações.

5.2 Funcionalidades Essenciais

As funcionalidades descritas a seguir representam os requisitos essenciais definidos para a aplicação ObraCerta. Cada funcionalidade está apresentada no formato de história do usuário, uma abordagem comum em métodos ágeis de desenvolvimento de software. A estrutura adotada é: “Como [perfil], quero [ação], para [benefício]”, expressando de forma objetiva quem se beneficia da funcionalidade, o que deseja realizar e qual o propósito ou valor gerado. Para facilitar a priorização das funcionalidades durante o desenvolvimento, foi adotado o método MoSCoW, que as classifica em categorias de importância: Must Have (deve ter), Should Have (deveria ter), Could Have (poderia ter) e Won’t Have This Time (não terá desta vez).

Serão apresentadas as histórias de usuário classificadas como Must Have, que compõem o MVP (Produto Mínimo Viável) da aplicação. As demais funcionalidades classificadas como Should, Could e Won’t Have estão organizadas no Apêndice B, para referência futura e possíveis evoluções do sistema.

5.2.1 Cadastro de Usuário (HU01)

- **História do Usuário:** *Como visitante, quero me registrar no sistema, para poder acessar e utilizar a plataforma.*
- **Critérios de Aceitação:**
 - O usuário deve preencher os campos: nome, e-mail, senha e confirmação de senha.
 - O sistema deve validar a unicidade do e-mail e aplicar regras de senha forte.
 - Após o cadastro bem-sucedido, o usuário deve ser redirecionado para a tela inicial da aplicação.

5.2.2 Autenticação (Login/Logout) (HU02)

- **História do Usuário:** *Como usuário, quero acessar o sistema com meu login e senha, para visualizar meus dados com segurança.*
- **Critérios de Aceitação:**
 - O sistema deve permitir login utilizando e-mail e senha cadastrados.
 - A autenticação deve utilizar tokens JWT (JSON Web Token) para controle de sessão.
 - O usuário deve ter a opção de logout, encerrando sua sessão de forma segura.
 - O sistema deve oferecer funcionalidade de recuperação de senha via e-mail.

5.2.3 Cadastro de Despesas (HU03)

- **História do Usuário:** *Como um usuário, quero cadastrar uma despesa de material, mão de obra ou serviço para que eu possa controlar meus gastos na construção.*
- **Critérios de Aceitação:**
 - O usuário deve informar tipo da despesa, valor, data e fornecedor.
 - O sistema deve calcular automaticamente o valor total da despesa com base na quantidade e preço unitário.
 - Deve ser possível adicionar uma foto da nota fiscal ou recibo.
 - O sistema deve validar campos obrigatórios antes de salvar.

5.2.4 Edição e Exclusão de Despesas (HU04)

- **História do Usuário:** *Como um usuário, quero poder editar ou excluir despesas já cadastradas para corrigir informações ou remover registros errados.*
- **Critérios de Aceitação:**
 - O usuário deve conseguir visualizar todas as despesas cadastradas.
 - O usuário deve poder editar qualquer campo de uma despesa já registrada.
 - O usuário deve poder excluir uma despesa, com um aviso de confirmação.
 - O sistema deve impedir a exclusão de despesas vinculadas a relatórios financeiros fechados.

5.2.5 Visualização e Filtros de Gastos (HU05)

- **História do Usuário:** *Como um usuário, quero visualizar todas as despesas cadastradas e aplicar filtros para encontrar informações específicas.*
- **Critérios de Aceitação:**
 - O usuário deve poder visualizar despesas ordenadas por data, valor ou fornecedor.
 - O sistema deve permitir filtros por período, categoria e fornecedor.
 - O sistema deve exibir um totalizador dos gastos filtrados.

5.2.6 Relatórios e Gráficos de Gastos (HU06)

- **História do Usuário:** *Como um usuário, quero gerar relatórios detalhados dos meus gastos para analisar onde estou gastando mais na minha obra.*
- **Critérios de Aceitação:**
 - O usuário deve poder escolher um período e uma categoria de despesas.
 - O sistema deve gerar um relatório em tabela e em gráfico de pizza/barras.
 - O relatório deve comparar o orçamento planejado com os gastos reais.
 - Deve haver opção de exportação para PDF.

5.2.7 Orçamento Inicial e Comparação com Gastos (HU07)

- **História do Usuário:** *Como um usuário, quero cadastrar um orçamento inicial para poder comparar com os gastos reais ao longo da obra.*
- **Critérios de Aceitação:**
 - O usuário deve informar um valor previsto para cada categoria de despesas.
 - O sistema deve exibir um indicador de progresso da obra, mostrando quanto já foi gasto em relação ao orçamento.
 - O sistema deve gerar alertas se um orçamento for ultrapassado.

5.2.8 Notificações de Pagamentos Pendentes (HU08)

- **História do Usuário:** *Como um usuário, quero receber notificações quando um pagamento estiver próximo do vencimento para evitar atrasos e juros.*
- **Critérios de Aceitação:**
 - O usuário deve poder configurar alertas para pagamentos futuros.
 - O sistema deve enviar notificações visuais e e-mail quando o vencimento estiver próximo.
 - O sistema deve listar pagamentos pendentes na tela principal.

5.2.9 Integração com Nota Fiscal Eletrônica (NF-e) (HU09)

- **História do Usuário:** *Como um usuário, quero cadastrar minhas despesas automaticamente informando a chave de acesso da NF-e para evitar digitação manual.*
- **Critérios de Aceitação:**
 - O usuário deve informar a chave de acesso (44 dígitos) da nota fiscal.
 - O sistema deve buscar automaticamente os dados na API da Sefaz.
 - Os dados da nota devem ser preenchidos nos campos de cadastro de despesas.
 - O usuário deve poder revisar os dados antes de salvar.

5.3 Público-alvo

O principal público-alvo do sistema ObraCerta são indivíduos que estão construindo ou reformando suas próprias residências, sem o suporte de empresas de engenharia ou construtoras.

Além disso, a aplicação também pode beneficiar:

- Profissionais autônomos da construção civil (engenheiros, mestres de obras, pedreiros), que poderão auxiliar clientes no controle de custos.
- Pequenas empresas e escritórios de arquitetura que oferecem consultoria para obras residenciais.
- Estudantes de áreas correlatas (engenharia civil, arquitetura, tecnologia) como ferramenta de apoio à organização de projetos práticos.

5.4 Considerações

A análise das funcionalidades permitiu estruturar as principais demandas do sistema, destacando os perfis de usuários, os objetivos de cada recurso e os benefícios esperados. A abordagem com histórias de usuário tornou possível visualizar os fluxos da aplicação e compreender como cada funcionalidade contribui para o controle financeiro de obras. Esse mapeamento é fundamental para orientar as etapas seguintes do projeto, como a prototipação e a modelagem de dados.

6 PROJETO DO SISTEMA

O projeto do sistema refere-se à definição da estrutura visual e lógica que sustentará a aplicação. Este capítulo apresenta os principais elementos que compõem essa estrutura, como a identidade visual e sua aplicação na prototipação das interfaces, além da modelagem do banco de dados, responsável por organizar as informações que serão manipuladas e exibidas ao longo da utilização do sistema.

6.1 Identidade Visual

Antes do desenvolvimento das interfaces, foram definidas diretrizes visuais fundamentais para garantir a coerência estética da aplicação e reforçar sua proposta funcional. A identidade visual abrange a escolha da paleta de cores, a tipografia e elementos gráficos, como o logotipo, que juntos compõem a aparência do sistema e contribuem para uma navegação intuitiva e agradável.

6.1.1 Cores Principais

A paleta de cores é composta por dois tons principais:

- **Amarelo:** Representando o capacete de obra, um símbolo clássico da construção civil, a cor remete ao trabalho, à segurança e à estruturação física do projeto.
- **Verde petróleo:** Transmite estabilidade, confiança e controle, alinhando-se ao propósito do sistema de oferecer previsibilidade financeira durante o planejamento da obra.
- **Verde claro:** Utilizado como uma cor para destaques leves no sistema, como ao passar o mouse por áreas clicáveis
- **Cinza claro:** Utilizado como cor neutra para planos de fundo e separações visuais, garantindo leveza e foco nos elementos interativos.
- **Branco:** Usado para o fundo principal das telas, trazendo clareza e facilitando a leitura de textos e informações.
- **Cinza escuro ou Preto suave:** Aplicado a textos e ícones, assegurando contraste adequado e boa legibilidade.

6.1.2 Tipografia

A tipografia escolhida para a aplicação é a fonte Inter, amplamente utilizada em interfaces web. Essa escolha se deve à sua legibilidade, modernidade e flexibilidade em diferentes tamanhos e dispositivos, tornando a leitura mais confortável ao usuário.

6.1.3 Logotipo

O logotipo da aplicação reúne dois elementos centrais, o capacete de obra em amarelo e o símbolo de visto (check) em verde, que simboliza segurança e conclusão bem-sucedida. Juntos, esses elementos comunicam visualmente a proposta do sistema de proporcionar o planejamento e a realização de uma obra bem estruturada e financeiramente controlada.



Figura 3 – Logotipo do ObraCerta
Fonte: Autoria própria (2025).

6.2 Planejamento da Sprint 0 e da Sprint 1

Com a conclusão da fase de análise e projeto, foram definidas as primeiras funcionalidades que deverão ser priorizadas para o início do desenvolvimento da aplicação **ObraCerta**. As funcionalidades escolhidas correspondem às histórias de usuário HU01 (Cadastro de Usuário) e HU02 (Autenticação), fundamentais para o controle de acesso ao sistema. O planejamento inicial está estruturado em duas etapas: Sprint 0 e Sprint 1.

6.2.1 Preparação de Ambiente (Sprint 0)

A Sprint 0 tem como foco a configuração do ambiente de desenvolvimento, ferramentas de versionamento e integração, além da organização da estrutura do projeto. Essa sprint tem duração estimada de até 5 dias úteis e não entrega funcionalidades diretamente ao usuário final, mas é essencial para o bom andamento do desenvolvimento. As atividades previstas incluem:

- Configuração dos ambientes de desenvolvimento locais (front-end e back-end).

- Instalação de dependências e bibliotecas essenciais.
- Estruturação inicial do repositório e definição da arquitetura do monorepo.
- Criação de branches conforme o fluxo GitFlow.
- Criação do banco de dados e execução de migrações iniciais.
- Definição de rotas básicas da aplicação e testes de integração entre as camadas.

6.2.2 Cadastro e Login (Sprint 1)

A Sprint 1 terá como meta a entrega de funcionalidades visíveis e funcionais, com foco na criação de contas e autenticação de usuários. Com duração prevista de duas semanas, a sprint será baseada nas histórias HU01 e HU02. As tarefas incluem:

- Implementação do endpoint de criação de usuário com validação de dados e senha segura.
- Criptografia de senhas antes do armazenamento no banco de dados.
- Tela de cadastro com tratamento de erros e redirecionamento ao login.
- Implementação da autenticação via JWT, com geração e validação de tokens.
- Desenvolvimento da tela de login com feedback de erros.
- Lógica de logout.
- Funcionalidade de recuperação de senha por e-mail.
- Testes unitários e de integração para as rotas e interfaces.

Ao final da Sprint 1, espera-se que o sistema permita o cadastro seguro de novos usuários e o acesso à aplicação por meio de login autenticado, servindo como base para as funcionalidades futuras.

6.3 Prototipação de Telas

A prototipação das interfaces foi realizada com a ferramenta **Figma**, seguindo as diretrizes da identidade visual do sistema. Considerando a metodologia ágil adotada, nesta etapa foram desenvolvidos apenas os protótipos relacionados às funcionalidades previstas para a **Sprint 1**, com foco na estrutura de autenticação do sistema.

6.3.1 Registro de usuário

O fluxo inicial da aplicação contempla a possibilidade de criação de conta por novos usuários. Ao acessar a landing page, o visitante pode clicar em “*Registre-se*” e será direcionado à tela de cadastro, ilustrada na Figura 4.

Nessa tela, o sistema solicita os dados essenciais: nome completo, e-mail, senha e confirmação de senha. O botão de voltar permite retornar à tela anterior, caso necessário. Após a validação e envio do formulário, o novo usuário é cadastrado e redirecionado para a tela inicial do sistema, com acesso às funcionalidades do perfil comum.

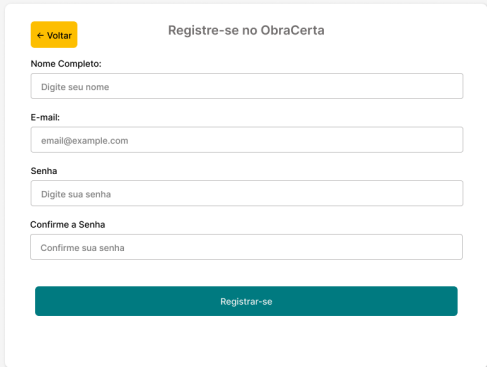


Figura 4 – Tela de Registro de usuário

Fonte: Autoria própria (2025).

6.3.2 Autenticação

Usuários já cadastrados podem realizar login por meio da tela de autenticação, mostrada na Figura 5. Nela, é solicitado o preenchimento das credenciais (e-mail e senha) para acesso ao sistema.

Além da autenticação padrão, a tela também oferece recursos adicionais: acesso ao formulário de registro, para novos usuários, e recuperação de senha, caso o usuário tenha esquecido suas credenciais. Após o login bem-sucedido, o sistema redireciona o usuário para a tela inicial, adaptada ao seu perfil (comum ou administrador).

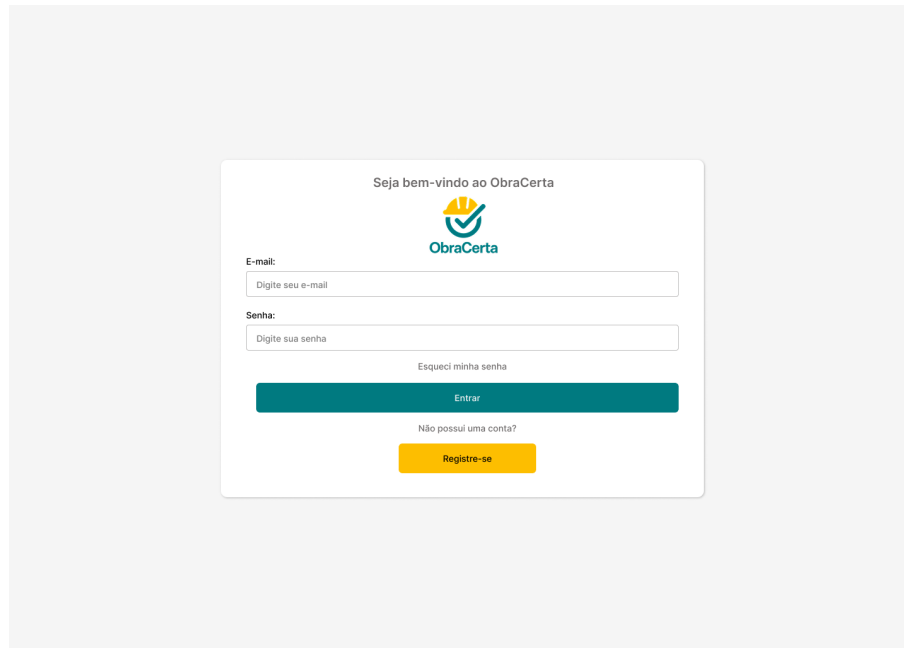


Figura 5 – Tela de Autenticação

Fonte: Autoria própria (2025).

6.4 Modelagem do Banco de Dados

O banco de dados do sistema **ObraCerta** foi projetado para armazenar e organizar as informações essenciais à operação da aplicação, incluindo dados de usuários, orçamentos, despesas, contas a pagar e notificações. A modelagem busca garantir uma estrutura coesa e compatível com as funcionalidades levantadas na análise do sistema.

A Figura 6 apresenta o diagrama relacional desenvolvido para representar a estrutura lógica do banco de dados. Como ilustrado no diagrama, o modelo contempla as principais entidades envolvidas nas operações do sistema e seus respectivos relacionamentos. A seguir, é apresentada uma breve descrição das tabelas que compõem o modelo:

- **users:** Armazena as informações dos usuários do sistema, incluindo nome, e-mail, senha criptografada e tipo de usuário (por exemplo, administrador ou comum).
- **expenses:** Registra os dados das despesas cadastradas pelos usuários, com vínculos para as tabelas de usuários, categorias e fornecedores. Possui campos como o tipo da despesa (material, mão de obra ou serviço), descrição, quantidade, unidade de medida(m², kg, etc), preço unitário, valor total, data da compra, local de armazenamento do comprovante digital(Nota fiscal ou recibo) e chave de acesso da Nota Fiscal Eletrônica (NF-e).
- **budgets:** Armazena os valores orçados pelos usuários para cada categoria de despesa. Possui os campos de id do usuário e categoria, assim como o campo de valor estimado.

- **pending_payments:** Contém os registros de despesas que ainda não foram quitadas, com indicação da despesa associada e sua data de vencimento.
- **suppliers:** Tabela destinada ao cadastro dos fornecedores vinculados às despesas. Armazena informações como o nome, e-mail e telefone.
- **expense_categories:** Responsável pelo armazenamento das categorias que serão atribuídas as despesas e orçamentos, como alvenaria, elétrica, hidráulica, entre outras. Armazena o nome da categoria.
- **notifications:** Armazena as mensagens de notificação geradas para os usuários, como alertas de vencimentos ou estouro de orçamento. Inclui o conteúdo da mensagem, o tipo de notificação(vencimento, orçamento).

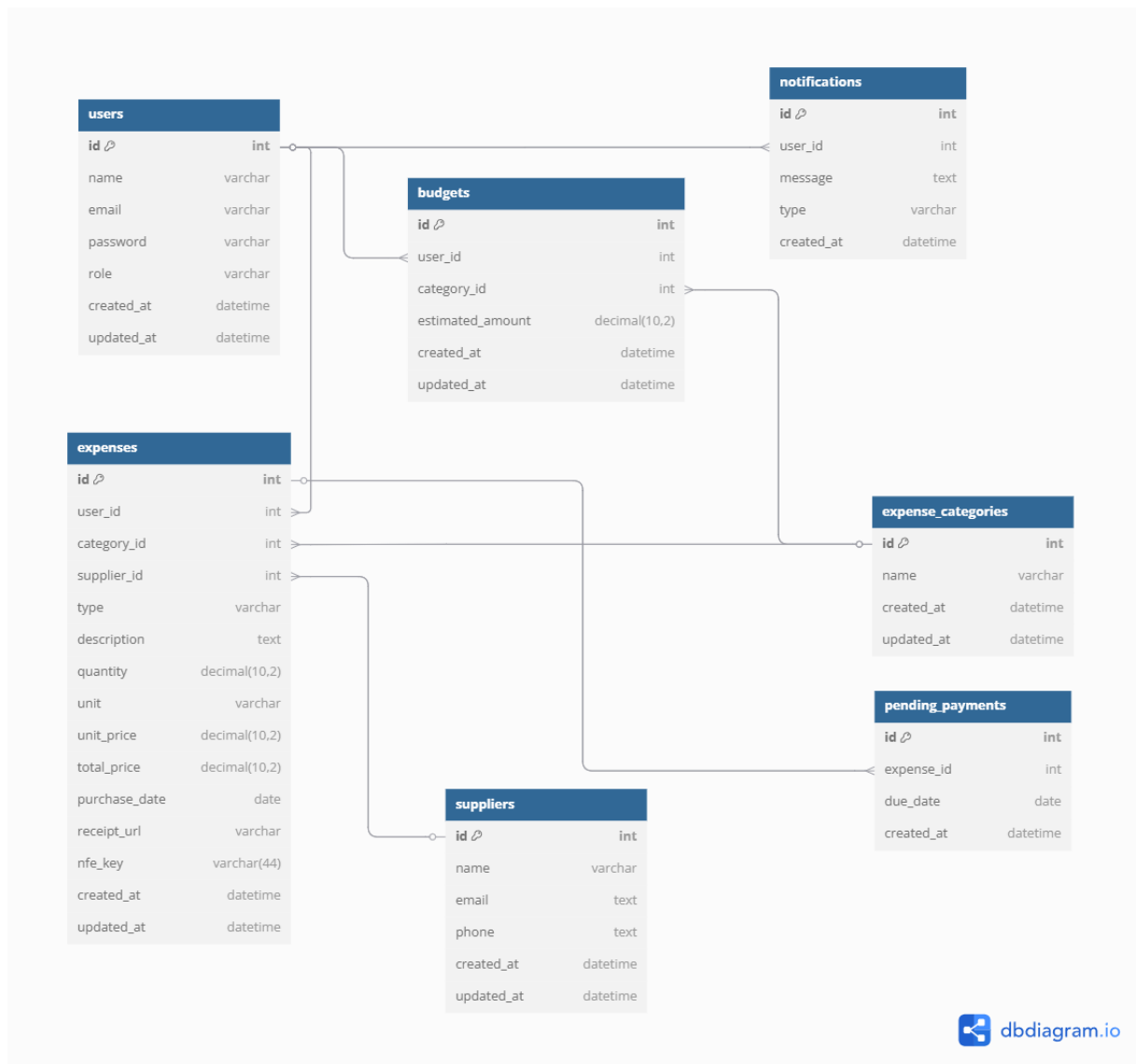


Figura 6 – Diagrama de Banco de Dados do ObraCerta

Fonte: Autoria própria (2025).

6.5 Considerações

O Projeto do Sistema é uma parte importante do desenvolvimento, pois ajuda a organizar como o sistema vai funcionar e como ele vai parecer para o usuário. Nesta etapa, foram definidas as cores, a fonte e o logotipo que dão identidade visual ao sistema, deixando a aparência mais agradável e padronizada.

Essas definições foram usadas para montar os primeiros protótipos das telas, permitindo imaginar como o sistema será na prática. Além disso, foi feito o modelo do banco de dados, que mostra como as informações serão guardadas e organizadas.

Com isso, o projeto já deixa claro como será a navegação do usuário e como os dados serão tratados, facilitando o início da programação e garantindo que o sistema atenda às necessidades planejadas.

7 CONCLUSÃO

A solução **ObraCerta** propõe-se a ser uma alternativa digital acessível e eficiente para o planejamento e o controle financeiro de obras residenciais autogeridas. Diferentemente de muitas ferramentas disponíveis no mercado, que costumam apresentar altos custos e interfaces complexas, o ObraCerta prioriza a simplicidade. Dessa forma, busca atender usuários com diferentes níveis de conhecimento técnico, tornando o gerenciamento de obras mais democrático e intuitivo.

O sistema foi concebido para reduzir um dos problemas mais recorrentes em pequenas obras: o estouro do orçamento. Com a possibilidade de cadastrar um orçamento inicial e registrar as despesas ao longo da execução — seja de forma manual ou por meio da importação de NF-e, o ObraCerta permitirá ao usuário comparar, em tempo real, os valores previstos e os efetivamente gastos. Isso oferecerá uma visão clara da evolução financeira da obra, promovendo maior controle, transparência e tomada de decisão mais assertiva durante todo o processo.

REFERÊNCIAS

- Atlassian. **O que é Git**. 2025. Disponível em: <https://www.atlassian.com/br/git/tutorials/what-is-git>. Acesso em: 16 junho. 2025.
- Atlassian. **Saiba tudo sobre o Gitflow Workflow**. 2025. Disponível em: <https://www.atlassian.com/br/git/tutorials/comparing-workflows/gitflow-workflow>. Acesso em: 16 junho. 2025.
- Câmara Brasileira da Indústria da Construção (CBIC). **Mercado imobiliário bate recorde em 2024 com alta de 18,6% nos lançamentos**. 2024. Disponível em: <https://cbic.org.br/mercado-imobiliario-bate-recorde-em-2024-com-alta-de-186-nos-lancamentos/>.
- Câmara Brasileira da Indústria da Construção (CBIC). **Lançamentos e vendas acumuladas de imóveis em 2024 são os maiores da série histórica, diz CBIC**. 2025. Disponível em: <https://cbic.org.br/lancamentos-e-vendas-de-imoveis-sobem-20-no-3o-trimestre-de-2024/>.
- dbdiagram. **dbdiagram.io: Draw ER diagrams by typing DBML**. 2025. Disponível em: <https://dbdiagram.io>. Acesso em: 17 jun. 2025.
- Docker. **What is Docker? - DockerDocs**. 2025. Disponível em: <https://docs.docker.com/get-started/docker-overview/>. Acesso em: 16 junho. 2025.
- GitHub - GitHub Docs. **Planejamento e acompanhamento com o Projects**. 2025. Disponível em: <https://docs.github.com/pt/issues/planning-and-tracking-with-projects>. Acesso em: 16 junho. 2025.
- IBGE - Instituto Brasileiro de Geografia e Estatística. **Sistema Nacional de Pesquisa de Custos e Índices da Construção Civil - SINAPI**. 2025. Disponível em: <https://www.ibge.gov.br/estatisticas/economicas/precos-e-custos/9270-sinapi.html>.
- LIMMER, C. V. **Planejamento, orçamentação e controle de projetos e obras**. [S.l.]: Editora TLC, 1997. 1–6 p.
- Ministério do Trabalho. **Dados Setoriais - Emprego na Construção Civil**. 2023. Disponível em: <https://www.gov.br/trabalho-e-emprego/>.
- Monorepo. **What is a Monorepo**. 2025. Disponível em: <https://monorepo.tools/#what-is-a-monorepo>. Acesso em: 16 junho. 2025.
- OrçaFascio. **SOBRE A ORÇAFASCIO - OrçaFascio**. 2025. Disponível em: <https://www.orcafascio.com/sobre-a-empresa>. Acesso em: 04 maio. 2025.
- PersonalKanban. **Understanding Personal Kanban - PersonalKanban**. 2025. Disponível em: <https://www.personalkanban.com/pk/>. Acesso em: 16 junho. 2025.
- Preço Da Obra. **O que é o Preço Da Obra e como ele pode me ajudar? - Preço Da Obra**. 2025. Disponível em: <https://precodaobra.com/>. Acesso em: 02 maio. 2025.
- Schwaber, Ken and Sutherland, Jeff. **O Guia do Scrum - O Guia Definitivo para o Scrum: As Regras do Jogo**. 2020. Disponível em: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf>. Acesso em: 16 junho. 2025.
- Sebrae. **O método MoSCoW para definição de prioridades**. 2023. Disponível em: https://sebrae.com.br/Sebrae/Portal%20Sebrae/Arquivos/ebook_sebrae_metodologia_moscow.pdf. Acesso em: 16 junho. 2025.

Sienge. **Demonstração - Sienge**. 2025. Disponível em: <https://sienge.com.br/demonstracao/>. Acesso em: 04 maio. 2025.

Steinfeld, Grant. **5 steps of test-driven development**. 2023. Disponível em: <https://developer.ibm.com/articles/5-steps-of-test-driven-development/>. Acesso em: 16 junho. 2025.

Tailwind. **Tailwind**. 2025. Disponível em: <https://tailwindcss.com/>. Acesso em: 16 junho. 2025.

Tresium. **Por quê usar o ConstruCalc? - Tresium**. 2025. Disponível em: <http://tresium.com.br/>. Acesso em: 02 maio. 2025.

APÊNDICES

APÊNDICE A – Modelo de dados detalhado

A.1 Modelo de Dados

O modelo de dados do *ObraCerta* foi projetado para oferecer suporte completo ao controle financeiro de obras residenciais, com foco em construções de pequeno e médio porte. A estrutura contempla o gerenciamento de usuários, orçamentos, despesas, fornecedores, contas a pagar e notificações, com foco em normalização, integridade e escalabilidade futura.

A.1.1 Tabela `users`

Essa tabela define os usuários do sistema, onde cada perfil pode ter permissões específicas, onde permite acesso a diferentes áreas do sistema e funcionalidades.

- `id` (*int*, PK, UNIQUE): Identificador único do usuário.
- `name` (*varchar*): Nome do usuário.
- `email` (*varchar*, UNIQUE): E-mail de login.
- `password` (*varchar*): Senha criptografada.
- `role` (*varchar*): Tipo de usuário
Exemplo: `usuario`, `admin`.
- `created_at`, `updated_at` (*datetime*): Datas de criação e atualização.

Justificativa: Permite autenticação segura e controle de permissões por tipo de usuário.

A.1.2 Tabela `suppliers`

Armazena informações de fornecedores utilizados nas despesas.

- `id` (*int*, PK, UNIQUE): Identificador do fornecedor.
- `name` (*varchar*): Nome do fornecedor.
- `email` (*text*): E-mail do fornecedor
- `phone` (*text*): Telefone do fornecedor.
- `created_at`, `updated_at` (*datetime*): Datas de registro.

Justificativa: Facilita a reutilização de dados de fornecedores em múltiplas despesas.

A.1.3 Tabela `expense_categories`

Define categorias para as despesas e orçamentos.

- `id (int, PK)`: Identificador da categoria.
- `name (varchar)`: Nome da categoria.
- `created_at, updated_at (datetime)`: Datas de modificação.

Justificativa: Auxilia na organização e análise financeira por tipo de gasto.

A.1.4 Tabela `budgets`

Contém os valores orçados inicialmente por categoria.

- `id (int, PK)`: Identificador do orçamento.
- `user_id (int, FK → users.id)`: Usuário responsável.
- `category_id (int, FK → expense_categories.id)`: Categoria do orçamento.
- `estimated_amount (decimal)`: Valor previsto.
- `created_at, updated_at (datetime)`: Datas de criação.

Justificativa: Permite comparar os gastos reais com os valores previstos.

A.1.5 Tabela `expenses`

Registra as despesas da obra.

- `id (int, PK)`
- `user_id, category_id, supplier_id (FKs)`: Referências externas.
- `type (varchar)`: Tipo de despesa (Material, Mão de Obra, Serviço).
- `description (text)`: Descrição do item.
- `quantity (decimal), unit (varchar), unit_price (decimal)`.
- `total_price (decimal)`: Valor calculado da despesa.
- `purchase_date (date)`: Data da compra.
- `receipt_url (varchar)`: URL do comprovante.

- `nfe_key (varchar(44))`: Chave da nota fiscal eletrônica.
- `created_at, updated_at (datetime)`

Justificativa: Permite detalhamento das despesas com comprovação e automatização de alertas.

A.1.6 Tabela `pending_payments`

Registra pagamentos futuros vinculados a despesas.

- `id (int, PK)`
- `expense_id (int, FK → expenses.id)`
- `due_date (date)`: Data de vencimento do pagamento.
- `created_at (datetime)`

Justificativa: Armazena os pagamentos pendentes, auxiliando na exibição deles para usuários.

A.1.7 Tabela `notifications`

Controla os alertas enviados pelo sistema.

- `id (int, PK)`
- `user_id (int, FK → users.id)`
- `message (text)`: Conteúdo da notificação.
- `type (varchar)`: Tipo de alerta (ex: `vencimento`, `estouro_orcamento`).
- `created_at (datetime)`

Justificativa: Envia comunicações automáticas com base nas ações e estados da obra.

Relacionamentos

- `users → expenses, budgets, notifications (1:N)`
- `suppliers → expenses (1:N)`
- `expense_categories → expenses, budgets (1:N)`
- `expenses → pending_payments (1:1)`

APÊNDICE B – Funcionalidades Futuras do Sistema ObraCerta

Este apêndice apresenta as funcionalidades que **não estão no escopo do presente Trabalho de Conclusão de Curso**. A priorização das funcionalidades do sistema **ObraCerta** foi realizada com base na metodologia **MoSCoW**, sendo que **apenas os requisitos classificados como Must Have** formam o escopo deste trabalho. As demais funcionalidades, identificadas como **Should Have**, **Could Have** e **Won't Have This Time**, estão organizadas nas próximas seções deste apêndice, compondo o escopo de versões futuras da aplicação.

B.1 Funcionalidades Should Have – Deveria Ter

B.1.1 Solicitação de Orçamentos Automatizada

- **História do Usuário:** *Como um usuário, quero enviar automaticamente solicitações de orçamento para fornecedores, a fim de comparar preços de forma prática.*
- **Critérios de Aceitação:**
 - O usuário deve poder selecionar materiais cadastrados para envio.
 - O sistema deve gerar e-mails padronizados contendo a lista de materiais.
 - Deve haver confirmação visual do envio e acompanhamento do status das respostas.
 - As respostas dos fornecedores devem poder ser registradas e comparadas no sistema.

B.1.2 Localização de Fornecedores Próximos

- **História do Usuário:** *Como um usuário, quero visualizar fornecedores próximos no mapa, para facilitar a compra presencial e a logística de entrega.*
- **Critérios de Aceitação:**
 - O sistema deve exibir fornecedores cadastrados em um mapa interativo.
 - Deve permitir busca por categoria de material ou nome do fornecedor.
 - Deve exibir informações como nome, endereço, telefone e horário de atendimento.

B.2 Funcionalidades Could Have – Poderia Ter

B.2.1 Banco de Preços Colaborativo

- **História do Usuário:** Como um usuário, quero acessar um banco de preços colaborativo com base em compras reais, para tomar decisões mais informadas sobre onde comprar materiais.
- **Critérios de Aceitação:**
 - O sistema deve permitir o compartilhamento de preços pagos por usuários.
 - Os dados devem ser organizados por produto, fornecedor e localização geográfica.
 - O sistema deve apresentar média de preços por item.
 - As informações devem ser armazenadas de forma anônima e segura.

B.2.2 Cadastro e Avaliação de Fornecedores

- **História do Usuário:** Como um usuário, quero cadastrar fornecedores e avaliar seu atendimento, para contribuir com a reputação dos parceiros comerciais.
- **Critérios de Aceitação:**
 - O sistema deve permitir o cadastro de novos fornecedores com CNPJ e dados completos.
 - Usuários autenticados devem poder deixar avaliações e comentários.
 - Cada fornecedor deve possuir um perfil com histórico de preços, avaliações e materiais fornecidos.

B.3 Funcionalidades Won't Have This Time – Não Terá Desta Vez

B.3.1 Integração com WhatsApp dos Fornecedores

- **História do Usuário:** Como um usuário, quero poder iniciar conversas com fornecedores via WhatsApp, para facilitar a negociação direta de preços e prazos.*
- **Critérios de Aceitação:**
 - O sistema deve apresentar botão de acesso ao WhatsApp ao lado das informações do fornecedor.

- O número de telefone deve estar armazenado no perfil do fornecedor.
- A abertura da conversa deve incluir uma mensagem padrão de solicitação de orçamento.

B.3.2 Atualização Automática de Preços via NF-e

- **História do Usuário:** Como um usuário, quero que os preços sejam atualizados automaticamente ao importar uma nota fiscal eletrônica, para manter meu histórico de valores sempre atualizado.
- **CrITÉrios de Aceitação:**
 - O sistema deve identificar os itens da nota fiscal e seus respectivos preços.
 - Os valores devem ser comparados com registros anteriores para fins de análise.
 - O usuário deve ser notificado sobre variações significativas de preço.