

UNIVERSIDADE VIRTUAL DO ESTADO DE SÃO PAULO

Fabiana Caroline Teixeira
Fernando Piauí da Silva
Marcelo Cornejo Noronha
Pedro Felipe Pimenta
Rodrigo Moreira dos Santos
Tatiana Roberta Greggato

**Gestão eficiente de solicitações condominiais com categorização de
prioridade**

São Paulo - SP
2025

UNIVERSIDADE VIRTUAL DO ESTADO DE SÃO PAULO

Gestão eficiente de solicitações condominiais com categorização de prioridade

Relatório Técnico-Científico apresentado na disciplina de Projeto Integrador para os cursos de Bacharel em Tecnologia da Informação, Ciência de Dados e Engenharia da Computação da Universidade Virtual do Estado de São Paulo (UNIVESP).

São Paulo - SP
2025

TEIXEIRA, Fabiana Caroline; SILVA, Fernando Piauí; NORONHA, Marcelo Cornejo; PIMENTA, Pedro Felipe; SANTOS, Rodrigo Moreira; GREGGATO, Tatiana Roberta. **Gestão eficiente de solicitações condominiais com categorização de prioridade** 36f. Relatório Técnico-Científico. Bacharel em Tecnologia da Informação, Ciência de Dados e Engenharia da Computação – **Universidade Virtual do Estado de São Paulo**. Tutor: (Tiago de Castro Fernandes). Polos (Agudos, Bauru, Lins, Barra Bonita e Pederneiras), 2025.

RESUMO

Relatório parcial descreve o processo do Projeto Integrador I, desenvolvido por estudantes da Universidade Virtual do Estado de São Paulo (UNIVESP), voltado para a criação de uma solução digital que otimize a gestão de solicitações de moradores no Condomínio Bolívar, localizado na cidade de Bauru (SP). Atualmente, o processo de registro e acompanhamento dessas demandas é descentralizado e pouco eficiente, dificultando a atuação do síndico. O projeto propõe a construção de um sistema web utilizando o framework Django, com interface intuitiva e funcionalidade de categorização por prioridade e status. A solução será acessível apenas ao síndico com login e senha para maior segurança das informações, permitindo maior controle, agilidade e organização nas rotinas administrativas. O presente relatório contempla a justificativa, fundamentação teórica, metodologia empregada, além dos resultados preliminares alcançados até o momento, evidenciando o progresso no desenvolvimento da aplicação.

PALAVRAS-CHAVE: condomínio, organização, sistema, demandas, prioridade.

LISTA DE ILUSTRAÇÕES

| | |
|---|----|
| FIGURA 1 - Tela de cadastro de demandas com opções de status | 14 |
| FIGURA 2 - Tela de cadastro de demandas com categorização de tipo e urgência..... | 16 |
| FIGURA 3 - Visualização da classificação por urgência no sistema de demandas | 17 |
| FIGURA 4 - Visualização do trecho que está no arquivo views.py e que define a lógica da filtragem (Back-end)..... | 19 |
| FIGURA 5 - Tela de busca com filtros dinâmicos implementados em HTML, conectados ao back-end (Django) para consultas ao banco de dados com base em status, urgência e tipo de demanda..... | 20 |
| FIGURA 6 - Tela de login desenvolvida em HTML (front-end) com back-end em Django para controle de acesso ao sistema | 21 |
| FIGURA 7 - Tela de Login com autenticação segura (back-end)..... | 22 |
| FIGURA 8 - Definição técnica do modelo Chamado..... | 24 |
| FIGURA 9 - Tela inicial do site desenvolvido para o síndico..... | 27 |

SUMÁRIO

| | |
|---|-----------|
| 1 INTRODUÇÃO | 1 |
| 2 DESENVOLVIMENTO..... | 3 |
| 2.1 OBJETIVOS | 5 |
| 2.1.1 Objetivo Geral | 5 |
| 2.1.2 Objetivos Específicos | 5 |
| 2.2 JUSTIFICATIVA E DELIMITAÇÃO DO PROBLEMA | 7 |
| 2.3 FUNDAMENTAÇÃO TEÓRICA | 8 |
| 2.4 METODOLOGIA..... | 12 |
| 2.4.1. Registrar as solicitações condominiais em um sistema centralizado | 13 |
| 2.4.2. Classificar as demandas por grau de urgência e por tipo (manutenção, reclamação e denúncias de moradores) | 15 |
| 2.4.3. Buscar e filtrar demandas por critérios específicos | 18 |
| Lógica de filtragem no Django (views.py):..... | 19 |
| 2.4.4. Criar um ambiente seguro e confiável para o armazenamento e acesso aos dados. 20 | |
| 2.4.5. Garantir a organização das solicitações e reduzir o risco de esquecimentos ou perdas de informações..... | 24 |
| 2.5 RESULTADOS PRELIMINARES: SOLUÇÃO INICIAL | 25 |
| REFERÊNCIAS | 29 |

1 INTRODUÇÃO

A administração de um condomínio envolve uma série de responsabilidades, entre elas o controle e o acompanhamento de solicitações feitas pelos moradores. Essas demandas que incluem reclamações, solicitações de manutenção, sugestões e outras necessidades, se não forem devidamente registradas e priorizadas, podem comprometer a eficiência da gestão. A descentralização desses registros, somada à ausência de uma ferramenta adequada, dificulta o acompanhamento das solicitações e atrasa sua resolução.

No Condomínio Bolívar, localizado na cidade de Bauru (SP), o síndico Vinícius enfrenta o desafio diário de organizar e acompanhar as diversas solicitações feitas pelos moradores. Atualmente, cerca de 37 a 40 demandas mensais são registradas por meio do aplicativo “Conviver MRV”, disponibilizado pela construtora responsável. Essas solicitações são inicialmente repassadas pela administradora ao síndico por meio de mensagens no WhatsApp e, posteriormente, registradas manualmente na plataforma Bragi — um sistema de comunicação multicanal utilizado para consolidar e organizar as interações com os condôminos. O Bragi permite o gerenciamento centralizado de mensagens recebidas por diversos canais, ferramenta útil como uma forma de manter um controle mínimo das solicitações, uma vez que, até então, essas informações se perdiam facilmente entre conversas informais em aplicativos de mensagens. No entanto, o Bragi não possui funcionalidades específicas para gestão condominial, como controle de chamados internos, divisão por torres ou unidades, nem geração de relatórios personalizados, o que limita sua aplicabilidade no contexto atual. Essa lacuna reforça a necessidade de desenvolvimento de um sistema mais robusto e direcionado às demandas reais da administração de condomínios, proposta que fundamenta este projeto.

Para tentar organizar as informações, o síndico utiliza uma planilha de Excel, no entanto esse método tem se mostrado ineficiente principalmente pela dificuldade em classificar, atualizar e filtrar as demandas de acordo com critérios como urgência e o status de andamento.

Neste sentido, este trabalho se propõe a desenvolver um sistema de web exclusivo para uso do síndico, proporcionando autonomia no acesso à plataforma — especialmente considerando que ele também atua na cidade de Agudos/SP e precisa gerenciar o condomínio à distância, através do uso de frameworks web para otimizar a codificação, software de controle de versão distribuído, para gerenciar as alterações de código; e sistema gerenciador de banco

de dados, para persistir os dados da aplicação. Além disso, para dar sustentação a todo o trabalho, foram utilizados os conceitos vistos nas disciplinas de Leitura e Produção de Textos, Pensamento Computacional, Introdução a conceitos de computação, Projetos e Métodos para a Produção do Conhecimento, Algoritmos e Programação de Computadores, Programação Orientada a Objetos, Banco de Dados, Infraestrutura para Sistemas de Software e Desenvolvimento Web.

2 DESENVOLVIMENTO

A fase inicial do projeto envolveu o levantamento das necessidades reais enfrentadas pelo síndico Vinícius Vieira Missão, do Condomínio Bolívar, localizado em Bauru (SP), no gerenciamento das solicitações dos moradores. Foi identificado que, atualmente, essas demandas são registradas pelos moradores por meio do aplicativo “Conviver MRV” e repassadas à administradora, que as encaminha ao síndico via grupo de WhatsApp e posteriormente, as informações são transferidas para a plataforma online **Bragi** (plataforma que permite o gerenciamento centralizado de mensagens recebidas por diversos canais), que centraliza todas essas informações. No entanto, a forma como os dados são inseridos e exibidos na plataforma é desorganizada e limitada, sem opções de categorização por urgência ou de status de atendimento, como “pendente”, “em andamento” ou “resolvido”. Devido a essas limitações, o síndico adotou o uso de uma planilha Excel como alternativa, embora essa solução não seja intuitiva e exija um tempo considerável para marcar e filtrar informações e a ausência de critérios para priorização por nível de urgência, bem como a falta de controle sobre o status de cada solicitação (como pendente, em andamento ou resolvido), dificulta o acompanhamento eficaz das demandas e compromete a agilidade e competência da gestão.

Diante desse cenário, tornou-se evidente a necessidade do desenvolvimento de um sistema personalizado, que permita ao síndico registrar as solicitações recebidas e gerenciá-las de forma mais estruturada e funcional. A partir disso, a equipe iniciou a modelagem da solução, criando os primeiros esboços das telas e definindo os principais requisitos do sistema, com o objetivo de otimizar a gestão das solicitações feitas pelos moradores.

Com base nesse diagnóstico, iniciou-se a modelagem da solução, com a criação dos primeiros esboços das telas do sistema (protótipos) e a definição dos principais requisitos funcionais. A proposta visa auxiliar na organização e no controle das solicitações recebidas, oferecendo ao síndico uma ferramenta prática, centralizada e de fácil acesso.

Como primeiro passo da implementação, foi elaborada a tela de login, exigindo a autenticação do usuário por meio de usuário e senha, com o objetivo de garantir o acesso restrito apenas ao síndico. Essa medida visa assegurar a confidencialidade dos dados, conforme os princípios de segurança da informação, que envolvem a proteção contra acessos não autorizados

(ISO/IEC 27001, 2013), e evitar alterações ou visualizações não autorizadas, uma vez que as informações tratadas são de uso interno da gestão condominial.

Na sequência, foram definidos os critérios de categorização das demandas, com base em dois aspectos principais: prioridade e status. A prioridade será classificada em três níveis, inspirados na Matriz de Eisenhower, que auxilia na organização de tarefas com base em sua urgência e importância (Covey, 2004).

- Urgente e importante
- Urgente e não importante
- Não urgente

Já o status de cada solicitação permitirá o acompanhamento do andamento das atividades, podendo assumir os seguintes estados:

- Pendente (ainda não iniciada)
- Em andamento (em fase de execução ou acompanhamento)
- Resolvido (solicitação finalizada)

Essas classificações foram pensadas para facilitar a visualização das tarefas prioritárias e oferecer uma visão clara da situação de cada demanda, contribuindo para uma gestão mais eficiente.

Nesta parte, serão expostos os objetivos gerais e específicos, com o propósito de orientar o leitor sobre o propósito deste capítulo. O conteúdo também discute a justificativa, identificando os fatores que influenciaram a escolha da comunidade externa, e a delimitação do problema de pesquisa, esclarecendo a questão central abordada no projeto. Além disso, apresenta-se a metodologia, que descreve as abordagens de pesquisa utilizadas ao longo do desenvolvimento do trabalho.

2.1 OBJETIVOS

2.1.1 Objetivo Geral

O projeto tem como objetivo geral desenvolver um sistema web exclusivo para auxiliar o síndico do Condomínio Bolívar, localizado em Bauru (SP), com o intuito de melhorar a organização, o acompanhamento e a categorização das solicitações enviadas pelos moradores, promovendo maior controle, agilidade e eficiência na gestão condominial. O sistema proposto oferecerá uma interface intuitiva e funcional, permitindo ao síndico registrar manualmente cada solicitação, atribuindo a ela o nível de prioridade (urgência, emergência e transferidas) e o status de atendimento (pendente, em andamento ou resolvido), de acordo com seu conhecimento e julgamento sobre cada caso. Além disso, o sistema garantirá autonomia de acesso ao síndico, que trabalha em outro município, permitindo-lhe acompanhar as demandas e gerenciar as ocorrências de qualquer lugar. A aplicação também possibilitará o acompanhamento de histórico das demandas e facilitará a gestão eficiente das ocorrências diárias do condomínio.

2.1.2 Objetivos Específicos

- **Registrar as solicitações condominiais em um sistema centralizado:** Este objetivo visa garantir que todas as solicitações sejam registradas de forma organizada e acessível em um sistema web, facilitando seu gerenciamento. A substituição do uso de planilhas manuais e anotações informais permitirá a centralização das demandas, proporcionando um armazenamento eficiente em banco de dados relacional, além de possibilitar a filtragem e organização das solicitações de maneira estruturada. Para estruturar o armazenamento das demandas condominiais, foi definido um modelo chamado Demanda, utilizando o ORM do Django, que permite manipular o banco de dados por meio de objetos Python. De acordo com Silberschatz, Korth e Sudarshan (2011), os bancos de dados relacionais permitem a organização eficiente de dados e facilitam a recuperação por meio de consultas estruturadas. O uso de um ORM (Object-Relational Mapping), como o Django ORM, permite integrar a estrutura

relacional ao paradigma orientado a objetos, promovendo maior produtividade no desenvolvimento (HOLANDA, 2021).

• **Classificar as demandas por grau de urgência e por tipo (manutenção, reclamação e denúncias de moradores):** A categorização das solicitações será realizada por meio de campos obrigatórios no momento do cadastro. O grau de urgência será classificado em três níveis — Alta, Média e Baixa —, identificados por cores distintas (vermelho, amarelo e verde, respectivamente) através de códigos de cores, que serão controlados manualmente e de forma intuitiva, de acordo com o conhecimento e julgamento do síndico. Essa codificação visual, baseada nos princípios do Design Centrado no Usuário (DCU), facilita a identificação rápida das demandas mais críticas. Além disso, o sistema permitirá a classificação por tipo de ocorrência, como manutenções, reclamações e denúncias, possibilitando uma organização mais eficiente e o encaminhamento adequado de cada demanda às equipes responsáveis. A categorização das solicitações ajudará a priorizar as ações e a direcionar as demandas para as equipes adequadas. Será desenvolvido também, um processo ou método de acompanhamento do status das solicitações com atualização do status (pendente, em andamento e resolvido) por meio de um botão, um sistema que possibilite visualizar o status das demandas, permite que não perca prazos, consiga passar um feedback para os moradores mantendo a ordem das reclamações/demandas e nada fique pendente ou esquecido. Essa abordagem promove uma gestão condominial mais organizada, visualmente acessível e alinhada às necessidades reais do síndico.

• **Buscar e filtrar demandas por critérios específicos:** Implementar um mecanismo de busca e filtragem das solicitações condominiais por critérios definidos, como status, nível de urgência, tipo de ocorrência e morador envolvido. Essa funcionalidade permitirá ao síndico encontrar informações relevantes de maneira ágil e precisa, melhorando a eficiência do sistema e o tempo de resposta nas ações de gestão. Essa ferramenta se mostra essencial em situações de reincidência de comportamentos indevidos, como reclamações ou advertências já registradas anteriormente, possibilitando o histórico detalhado e a tomada de decisões mais assertivas. Para viabilizar a filtragem das demandas de forma eficiente, o sistema foi desenvolvido utilizando o mapeamento objeto-relacional (ORM) do framework Django, o que permite realizar buscas com base em múltiplos critérios de maneira prática e estruturada. Esses filtros foram incorporados à interface por meio de elementos visuais como campos de pesquisa e botões

interativos, otimizando a experiência do usuário e permitindo ao síndico localizar informações relevantes de forma rápida e precisa.

• **Proporcionar um ambiente seguro com autenticação via Login e Senha:** Para garantir a segurança das informações e restringir o acesso apenas a usuários autorizados, o sistema contará com uma camada de autenticação baseada em login e senha. Essa funcionalidade será implementada utilizando os recursos nativos do framework Django, que oferece um sistema robusto de autenticação com suporte a sessões, controle de usuários e hashes seguros de senhas. A aplicação contará com uma tela de login que validará as credenciais fornecidas pelo usuário. Caso as informações estejam corretas, será iniciada uma sessão autenticada, permitindo o acesso às funcionalidades internas do sistema. Caso contrário, o usuário será redirecionado com uma mensagem de erro apropriada. Esse processo é essencial para evitar o acesso indevido a dados sensíveis, como as demandas dos moradores, o que também está em conformidade com os princípios da LGPD (Lei Geral de Proteção de Dados), que estabelece diretrizes para o tratamento e proteção de dados pessoais no Brasil.

• **Garantir a organização das solicitações e reduzir o risco de esquecimentos ou perdas de informações:** Um sistema estruturado permite que o síndico acompanhe rigorosamente os prazos de cada demanda, evitando falhas críticas na gestão. As imagens das câmeras de segurança são armazenadas em um diretório central do sistema, permanecendo disponíveis por 15 dias – período essencial para a verificação de reclamações que exigem evidências visuais. Essa centralização dos registros contribui para respostas mais consistentes e fundamentadas, assegurando a integridade das informações e fortalecendo uma gestão eficiente, com potencial para reduzir custos operacionais e mitigar riscos.

2.2 JUSTIFICATIVA E DELIMITAÇÃO DO PROBLEMA

Cada vez mais pessoas optam por morar em condomínios, buscando melhor custo-benefício, segurança, lazer e praticidade. No entanto, para que a convivência em comunidade seja harmoniosa, torna-se necessário o estabelecimento de regras claras de convivência e uma convenção condominial, que normatiza tanto os aspectos administrativos quanto os comportamentais. Nesse contexto, o papel do síndico é essencial, atuando como mediador das demandas dos moradores e fiscalizador do cumprimento das normas.

Com o aumento da quantidade de unidades habitacionais e da complexidade das relações internas, a gestão manual das solicitações torna-se ineficiente e propensa a falhas. Segundo Oliveira (2020), "a adoção de sistemas informatizados na administração condominial melhora significativamente a organização dos dados e a capacidade de resposta às demandas". Portanto, a ausência de uma ferramenta própria pode comprometer o controle e a rastreabilidade das solicitações, gerando atrasos, retrabalhos e até consequências legais, principalmente quando se tratam de temas críticos como segurança ou manutenção.

No caso do Condomínio Bolívar, localizado em Bauru (SP), o gerenciamento atual é feito parcialmente pelo aplicativo "Conviver MRV" e complementado com anotações em planilhas, o que torna o processo fragmentado e vulnerável a esquecimentos ou perdas de dados. Situações como denúncias ou ocorrências que exigem acesso às câmeras de segurança são especialmente críticas, pois as imagens se apagam automaticamente após 15 dias. Segundo Reis et al. (2019), "o uso de sistemas centralizados possibilita maior agilidade na gestão e maior controle sobre prazos sensíveis".

Dessa forma, justifica-se o desenvolvimento de um sistema personalizado de apoio ao síndico, com uma interface simples e intuitiva, que permita organizar, categorizar e acompanhar cada solicitação de forma segura e eficiente, independentemente da localização do síndico, que trabalha em outro município. O projeto será exclusivo para o síndico Vinícius, enquanto os moradores continuarão utilizando o aplicativo oficial. O novo sistema funcionará como uma ferramenta complementar de gestão interna, focada na organização, controle e otimização do tempo de resposta.

2.3 FUNDAMENTAÇÃO TEÓRICA

O gerenciamento eficiente de demandas condominiais exige o uso de ferramentas tecnológicas que organizem, priorizem e deem visibilidade às solicitações recebidas. Essa necessidade está diretamente ligada à rotina de síndicos que, diariamente, lidam com um volume considerável de solicitações, as quais devem ser controladas de forma estruturada, segura e ágil.

A Tecnologia da Informação tem se consolidado como um instrumento essencial para a otimização de processos administrativos, melhoria na comunicação e tomada de decisões mais

assertivas. Segundo Rezende e Abreu (2000), “os sistemas de informação proporcionam apoio à administração, possibilitando melhor controle e análise de informações” (REZENDE; ABREU, 2000, p. 42).

No desenvolvimento da solução proposta, utilizou-se a combinação de tecnologias da web como HTML, CSS e JavaScript, que são fundamentais para a criação de interfaces responsivas e intuitivas. De acordo com Miletto e Bertagnolli (2020), “o HTML estrutura o conteúdo, o CSS define o estilo visual, e o JavaScript possibilita a interação dinâmica com o usuário” (MILETTO; BERTAGNOLLI, 2020, p.72).

Além disso, a escolha por um design limpo e minimalista foi intencional, com o objetivo de facilitar a interação do usuário final (o síndico), mesmo com baixo nível de conhecimento técnico. Conforme Nielsen (1993), “a usabilidade está relacionada à facilidade de aprendizado, eficiência de uso, memorização, redução de erros e satisfação do usuário” (NIELSEN, 1993, p. 26).

A construção do sistema seguiu os princípios das metodologias ágeis, com entregas incrementais e foco na colaboração contínua com o usuário. Segundo Schwaber e Beedle (2002), “as metodologias ágeis promovem o desenvolvimento iterativo, com foco na flexibilidade e adaptação às mudanças de requisitos” (SCHWABER; BEEDLE, 2002). No contexto deste projeto, o uso de práticas ágeis permitiu ajustes rápidos durante o desenvolvimento, conforme o síndico identificava novas necessidades ou sugeria melhorias.

Por fim, aspectos legais condominiais foram considerados apenas na definição dos tipos de demandas (manutenção, reclamação, denúncia, entre outros), de forma que o foco do projeto permanecesse voltado à organização interna do síndico, e não à regulamentação jurídica. Com isso, garante-se que o sistema atenda às necessidades práticas de gestão, sem extrapolar o escopo proposto.

O desenvolvimento de sistemas web tem se consolidado como uma prática essencial na transformação digital, permitindo a criação de soluções acessíveis a partir de qualquer navegador com conexão à internet. Segundo Pressman (2016, p. 39), “a engenharia de software para aplicações web combina princípios de engenharia com abordagens adaptativas, priorizando a interatividade, escalabilidade e acessibilidade”.

O uso de frameworks web torna esse processo mais produtivo e seguro. Frameworks são estruturas de desenvolvimento que padronizam a criação de sistemas, promovendo reutilização de código e organização da lógica de programação. De acordo com Sommerville (2011, p. 123), “frameworks oferecem uma base sólida que acelera o desenvolvimento e impõe boas práticas, reduzindo erros comuns”.

No contexto deste projeto, será adotada uma abordagem baseada em Metodologias Ágeis, com foco nas práticas do Scrum, que favorecem o desenvolvimento incremental e a entrega contínua de valor. Segundo Schwaber e Sutherland (2020), criadores do Scrum, essa metodologia divide o desenvolvimento em ciclos curtos chamados sprints, permitindo maior adaptabilidade e resposta rápida a mudanças nos requisitos. Dessa forma, o sistema poderá ser desenvolvido em etapas, com entregas parciais e feedback contínuo, alinhando-se melhor às necessidades reais do síndico e dos moradores.

Durante os sprints definidos no Scrum, serão implementadas funcionalidades fundamentais para o funcionamento do sistema, como as operações CRUD (Create, Read, Update, Delete). Essas operações permitem que o usuário cadastre, visualize, atualize e exclua registros — como chamados ou demandas — garantindo a manipulação eficiente e organizada dos dados no banco. O CRUD representa a base das ações que serão disponibilizadas no sistema, e sua implementação será distribuída estrategicamente ao longo das entregas incrementais do projeto.

Para a construção do sistema "Chama o Síndico 2.0", foram selecionadas ferramentas que atendem aos critérios de simplicidade, eficiência e alinhamento com boas práticas de desenvolvimento web. A seguir, são detalhadas as principais tecnologias adotadas:

- **Django**

O Django foi o framework escolhido para o desenvolvimento do backend do sistema. Escrito em Python, ele adota o padrão de arquitetura Model-View-Template (MVT), que facilita a separação entre as regras de negócio, a camada de dados e a interface com o usuário. Uma de suas maiores vantagens é a robustez no gerenciamento de dados e a segurança oferecida nativamente contra vulnerabilidades comuns, como injeção de SQL e CSRF.

De acordo com Holovaty e Kaplan-Moss (2009, p. 1), criadores do Django, “o framework foi projetado para ajudar os desenvolvedores a criar aplicações web de maneira rápida e com um código limpo e pragmático”. Além disso, a vasta documentação e a comunidade ativa tornam o Django ideal para quem está em processo de aprendizado e construção de projetos acadêmicos.

- **MySQL**

Para o armazenamento dos dados do sistema, será utilizado o MySQL, um Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR). Essa escolha se deve à sua estabilidade, eficiência no tratamento de grandes volumes de dados e ampla compatibilidade com aplicações web. O MySQL permite criar tabelas relacionadas entre si, o que facilita a organização dos dados e garante integridade nas transações.

Segundo Coronel e Morris (2014, p. 12), “os bancos de dados relacionais continuam sendo a base mais comum para aplicações empresariais, dada sua capacidade de manipular grandes volumes de dados com integridade e consistência”. A integração entre o Django e o MySQL é amplamente suportada, o que contribui para uma implementação mais fluida.

- **Git**

O controle de versão do sistema será feito com o uso do Git, uma ferramenta essencial para registrar o histórico de alterações no código-fonte. Com o Git, é possível recuperar versões anteriores do projeto, experimentar novas funcionalidades sem comprometer a base principal e facilitar a colaboração entre desenvolvedores, mesmo que em ambientes remotos.

Conforme Chacon e Straub (2014, p. 5), “o Git é um sistema de controle de versão distribuído, criado para lidar com projetos de forma rápida e eficiente, mesmo quando grandes”. Mesmo em projetos desenvolvidos individualmente, como este, o Git permite uma organização profissional do processo de desenvolvimento, além de possibilitar futura colaboração com outros programadores.

A existência e a observância desses documentos e legislações são imprescindíveis para a correta gestão das demandas condominiais, o que torna essencial sua consideração no

desenvolvimento de soluções digitais de apoio à administração, como o sistema proposto neste projeto.

Além de garantir a organização das solicitações, um sistema digital eficiente auxilia o síndico a cumprir suas obrigações legais de forma mais transparente e sistematizada. Isso é fundamental, pois a negligência no atendimento às responsabilidades legais e administrativas pode gerar sérias consequências jurídicas ao síndico, como advertências, multas e até destituição do cargo em assembleia, conforme previsto no Código Civil. Em casos mais graves, especialmente quando houver omissão em situações de risco à segurança, à saúde ou ao patrimônio, o síndico pode ainda responder civil e criminalmente por danos causados ao condomínio ou aos condôminos.

Portanto, a adoção de uma ferramenta digital que centralize e organize a gestão das demandas e reclamações — com prazos, status e registros claros — contribui não apenas para a eficiência administrativa, mas também para resguardar o síndico de eventuais responsabilizações legais, reforçando sua atuação de forma preventiva e transparente.

2.4 METODOLOGIA

O desenvolvimento do sistema Chama o Síndico 2.0 adotou os princípios do Design Thinking, uma metodologia centrada no ser humano, que promove a inovação por meio da empatia, colaboração e experimentação. Essa abordagem é especialmente eficaz em contextos onde é necessário compreender profundamente os problemas do usuário e construir soluções práticas e criativas. Sobre isso, Tim Brown destaca:

“Design Thinking é uma abordagem centrada no ser humano que busca a inovação integrando as necessidades das pessoas, as possibilidades da tecnologia e os requisitos para o sucesso dos negócios”. (BROWN, 2010, p. 4).

O processo foi estruturado em três grandes etapas:

- **Ouvir e interpretar o contexto:** Por meio de uma reunião realizada na plataforma Google Meet com o síndico Vinícius Vieira Missão, responsável pelo Condomínio Bolívar, foi possível identificar as principais dificuldades enfrentadas na gestão das solicitações condominiais. O síndico relatou que as demandas chegam de múltiplos canais, como por exemplo, aplicativo “Conviver MRV”, mensagens de WhatsApp onde

decidiu aderir plataforma Bragi (um sistema de comunicação multicanal), e também de forma presencial. Essa dispersão faz com que solicitações sejam esquecidas, perdidas ou tratadas fora do prazo, comprometendo a eficiência da gestão e a satisfação dos moradores.

- **Criar e prototipar:** A equipe do projeto realizou sessões de brainstorming para reunir ideias que atendessem diretamente aos problemas apontados. Essa técnica de criatividade em grupo é essencial na fase de ideação do Design Thinking, pois incentiva a geração livre de ideias e a construção colaborativa de soluções. Segundo Osborn (1953), criador da técnica, o brainstorming é mais eficaz quando as ideias são exploradas sem julgamento inicial, permitindo soluções inovadoras que talvez não surgissem em análises convencionais.

Entre as ideias discutidas, destacaram-se: Um formulário centralizado com categorização de demandas; Painéis de status para acompanhamento em tempo real; cores intuitivas destacando urgências e prazos.

- **Implementar e testar:** Com base nas ideias mais viáveis e de maior impacto identificadas nas sessões de brainstorming, iniciou-se a fase de prototipação, etapa essencial do Design Thinking, por meio da construção iterativa da aplicação utilizando o framework Django. Essa etapa foi diretamente vinculada aos objetivos específicos do projeto, sendo cada funcionalidade implementada alinhada a uma metodologia adequada e ferramentas tecnológicas específicas.

Pontos fundamentais para a implementação eficiente de gestão de solicitações condominiais:

2.4.1. Registrar as solicitações condominiais em um sistema centralizado

Instituir registros das solicitações condominiais em um sistema centralizado. A utilização da metodologia ágil Scrum permitiu entregas incrementais e ajustes com base no feedback contínuo do síndico. Aplicou-se a engenharia de requisitos para levantamento das funcionalidades essenciais. A estruturação do banco de dados seguiu os princípios da modelagem relacional, e o conceito de Lean Thinking foi aplicado para eliminar desperdícios

como o uso de planilhas manuais, promovendo a eficiência do fluxo de trabalho (WOMACK; JONES, 2004).

Metodologia aplicada: Classificação de demandas via Matriz de Priorização, baseada em urgência e impacto.

Ferramenta: Para estruturar o armazenamento das demandas condominiais, criando as classes Chamado e Demanda, com campos de categorização e status, utilizando o ORM (Object-Relational Mapping) do Django, que permite manipular o banco de dados por meio de objetos Python.

Justificativa técnica: A modelagem orientada a objetos oferecida pelo Django permite a construção de um banco de dados relacional estruturado, que facilita o armazenamento, filtragem e ordenação de dados relevantes ao síndico.

Modelo de Demanda no Back-End:

Figura 1 - Tela de cadastro de demandas com opções de status

```
1  from django.db import models
2
3  class Demanda(models.Model):
4      URGENCIA = 'urgência'
5      EMERGENCIA = 'emergência'
6      EM_ANDAMENTO = 'em andamento'
7      CONCLUIDA = 'Resolvido'
8
9      STATUS_CHOICES = [
10         (URGENCIA, 'Urgência'),
11         (EMERGENCIA, 'Emergência'),
12         (EM_ANDAMENTO, 'Em Andamento'),
13         (CONCLUIDA, 'Resolvido'),
14     ]
15
16     titulo = models.CharField(max_length=255)
17     descricao = models.TextField()
18     status = models.CharField(
19         max_length=20,
20         choices=STATUS_CHOICES,
21         default=EM_ANDAMENTO, # Status por padrão é 'Em Andamento'
22     )
23
```

Fonte: Autoria própria

A imagem apresenta o formulário onde o síndico preenche os dados da demanda, como título, descrição e status atual. As opções de status são exibidas em uma lista suspensa (dropdown), permitindo a seleção entre *Urgência*, *Emergência*, *Em andamento* ou *Resolvido*, conforme o andamento da resolução. As opções são controladas pelo back-end e refletem os valores definidos no modelo de dados do Django.

2.4.2. Classificar as demandas por grau de urgência e por tipo (manutenção, reclamação e denúncias de moradores)

Classificar as demandas por grau de urgência, origem e tipo, utilizaram-se princípios da engenharia de requisitos funcionais, conforme proposto por Pressman (2016). Foram definidos campos específicos de categorização e filtros no sistema. Além disso, adotou-se a abordagem de Design Centrado no Usuário (DCU), que garantiu uma interface intuitiva, com a utilização de cores (vermelho, amarelo e verde) para representar os diferentes níveis de urgência, definidos com base nas necessidades reais do síndico.

A equipe priorizou, nas primeiras sprints, funcionalidades básicas como o registro da solicitação e atribuição de status (pendente, em andamento, Resolvido), elementos essenciais para o acompanhamento do ciclo de vida de cada demanda. A adoção da lógica CRUD (Create, Read, Update, Delete) viabilizou a criação, visualização, atualização e exclusão das solicitações de forma dinâmica.

Para possibilitar a categorização das solicitações de acordo com seu tipo e grau de urgência, foram adicionados dois novos campos no modelo Demanda: tipo e urgencia. Isso permite que o sistema filtre, agrupe e exiba as demandas com maior clareza, apoiando a tomada de decisões do síndico.

Figura 2 - Tela de cadastro de demandas com categorização de tipo e urgência

```
@property
def is_urgente(self):
    """Verifica se o chamado é urgente"""
    return self.status == self.URGENCIA or self.status == self.EMERGENCIA
def get_flag(self):
    """Retorna a cor da flag dependendo do status"""
    if self.status == self.URGENCIA or self.status == self.EMERGENCIA:
        return 'vermelha' # Flags vermelha para urgência e emergência
    elif self.status == self.EM_ANDAMENTO:
        return 'amarela' # Flag amarela para em andamento
    elif self.status == self.RESOLVIDO:
        return 'verde' # Flag verde para concluída
    return 'neutra' # Para qualquer outro caso
```

Fonte: Autoria própria

A imagem que acompanha essa explicação apresenta o formulário de cadastro, com os campos de seleção para o tipo da solicitação e o grau de urgência, que podem ser definidos manualmente pelo síndico. A interface será construída de forma intuitiva, utilizando menus suspensos (dropdown) e **cores** associadas a cada nível de urgência (vermelho para alta, amarelo para média e verde para baixa), facilitando a visualização e priorização.

Além disso, a aplicação do paradigma de programação orientada a objetos facilitou o mapeamento das entidades do sistema, como “Chamado” e “Demanda”, permitindo que cada instância armazenasse seus próprios atributos e status e o framework Django foi escolhido por sua robustez e segurança, oferecendo suporte nativo para manipulação de dados e integração com banco de dados relacional MySQL. Isso garantiu a consistência e integridade das informações ao longo do processo, além de permitir consultas ágeis para exibir status atualizados em tempo real na interface do usuário. A cada ciclo de desenvolvimento, novas funcionalidades relacionadas ao status foram sendo incorporadas, como a classificação por cor (vermelho, amarelo e verde), filtros por situação da solicitação e ordenação por prioridade. Essas melhorias foram priorizadas no backlog com base no impacto percebido pelo síndico, conforme preconiza o Scrum.

Foi desenvolvido uma página HTML para exibir uma lista de demandas cadastradas no sistema, com classificação visual de urgência, utilizando tecnologias web front-end combinadas com o template engine do Django. As demandas cadastradas são exibidas em formato de lista.

Cada item apresenta o título da solicitação, a descrição, o status atual e uma etiqueta colorida (flag) correspondente ao grau de urgência definido.

Figura 3 - Visualização da classificação por urgência no sistema de demandas

```

1  !DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Demandas - Status</title>
7      <style>
8          .flag-vermelha {
9              background-color: red;
10             color: white;
11             padding: 5px;
12         }
13         .flag-amarela {
14             background-color: yellow;
15             color: black;
16             padding: 5px;
17         }
18         .flag-verde {
19             background-color: green;
20             color: white;
21             padding: 5px;
22         }
23     </style>
24 </head>
25 <body>
26     <h1>Demandas</h1>
27     <ul>
28         {% for demanda in demandas %}
29             <li>
30                 <strong>{{ demanda.titulo }}</strong><br>
31                 Descrição: {{ demanda.descricao }}<br>
32                 Status: {{ demanda.status }}<br>
33                 <!-- Exibe a flag com base na cor retornada pela função get_flag -->
34                 <span class="flag-{{ demanda.get_flag }}">
35                     {{ demanda.get_flag|capfirst }} - {{ demanda.status }}</span><br><br>
36             </li>
37             {% empty %}
38                 <p>Não há demandas para exibir.</p>
39             {% endfor %}
40     </ul>
41 </body>
42 </html>

```

Fonte: Autoria própria

As cores seguem um padrão visual intuitivo:

- Vermelho para demandas de alta urgência
- Amarelo para urgência média / em andamento

- Verde para baixa urgência/ resolvido

Essas etiquetas visuais foram implementadas com CSS interno, utilizando classes customizadas (flag-vermelha, .flag-amarela, .flag-verde) aplicadas dinamicamente via Django. A lógica para atribuição das classes coloridas é feita por uma função (não exibida nesta figura) que retorna o nome da classe correspondente com base na urgência registrada.

O uso dessa sinalização facilita a identificação rápida das prioridades, otimizando o tempo de resposta do síndico e contribuindo para uma gestão mais organizada das solicitações condominiais.

Metodologia aplicada: Modelagem de processos com base em Diagramas de Estado.

Ferramenta: Views baseadas em função (FBV) para exibição dos chamados conforme o status (Urgência, Em andamento, e Resolvido), com lógica de backend escrita em Python.

Justificativa técnica: A separação entre os estados das demandas melhora a rastreabilidade e o controle das ações, otimizando a gestão do tempo e priorização de tarefas.

2.4.3. Buscar e filtrar demandas por critérios específicos

A abordagem de Design Centrado no Usuário (DCU) foi adotada para garantir que o síndico possa interagir com o sistema de forma intuitiva, simples e eficiente. Elementos visuais, como o sistema de cores estilo semáforo para indicar a urgência, filtros de pesquisa por critérios específicos e atualização dinâmica de status foram projetados para facilitar a navegação e o uso cotidiano da ferramenta. Na modelagem do banco de dados, foram definidos índices específicos para campos como tipo de ocorrência, nível de urgência e reincidência, com o objetivo de otimizar as consultas e tornar o processo de filtragem mais rápido e eficiente.

Para atender a este objetivo, o sistema utilizará consultas ao banco de dados relacional por meio do ORM (Object-Relational Mapping) do Django, que permite filtrar registros com base em condições específicas (ex: `Demanda.objects.filter(status='pendente')`). A interface será equipada com campos de seleção (dropdowns), caixas de busca (input) e filtros dinâmicos via JavaScript, proporcionando ao usuário uma navegação mais fluida e intuitiva.

Lógica de filtragem no Django (views.py):

Figura 4 - Visualização do trecho que está no arquivo views.py e que define a lógica da filtragem (Back-end)

```
from django.shortcuts import render
from .models import Demanda

def lista_demandas(request):
    status = request.GET.get('status')
    urgencia = request.GET.get('urgencia')
    tipo = request.GET.get('tipo')

    demandas = Demanda.objects.all()

    if status:
        demandas = demandas.filter(status=status)
    if urgencia:
        demandas = demandas.filter(urgencia=urgencia)
    if tipo:
        demandas = demandas.filter(tipo=tipo)

    return render(request, 'lista_demandas.html', {'demandas': demandas})
```

Fonte: Autoria própria

O sistema utiliza o ORM do Django para acessar e filtrar os registros da tabela Demanda de forma dinâmica. Os filtros são baseados nos valores recebidos por parâmetros de URL (GET), como status=pendente&urgencia=alta.

Utilizamos um formulário HTML com método GET, que envia os filtros diretamente na URL. Os dados preenchidos nos selects são lidos na views.py e usados para refinar a consulta ao banco de dados.

Interface de filtragem (HTML com formulário):

Figura 5 - Tela de busca com filtros dinâmicos implementados em HTML, conectados ao back-end (Django) para consultas ao banco de dados com base em status, urgência e tipo de demanda.

```
<form method="get">
  <label>Status:
    <select name="status">
      <option value="">Todos</option>
      <option value="pendente">Pendente</option>
      <option value="em andamento">Em Andamento</option>
      <option value="resolvido">Resolvido</option>
    </select>
  </label>

  <label>Urgência:
    <select name="urgencia">
      <option value="">Todas</option>
      <option value="alta">Alta</option>
      <option value="media">Média</option>
      <option value="baixa">Baixa</option>
    </select>
  </label>

  <label>Tipo:
    <select name="tipo">
      <option value="">Todos</option>
      <option value="manutenção">Manutenção</option>
      <option value="reclamação">Reclamação</option>
      <option value="denúncia">Denúncia</option>
    </select>
  </label>

  <button type="submit">Filtrar</button>
</form>
```

Fonte: Autoria própria

Se trata de uma solução leve, funcional e intuitiva para o usuário final.

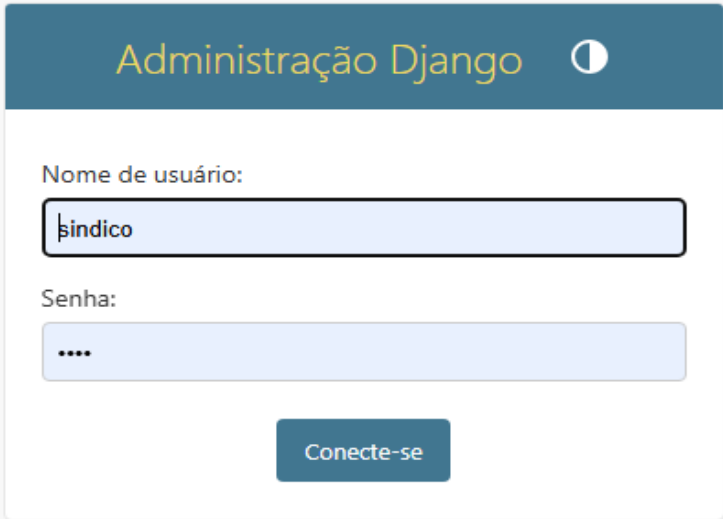
2.4.4. Criar um ambiente seguro e confiável para o armazenamento e acesso aos dados.

A implementação de autenticação com login e senha, aliada ao controle de acesso por perfis, garante que apenas usuários autorizados possam acessar o sistema, visando prevenir vulnerabilidades comuns em aplicações web, como falhas de autenticação e exposição de dados sensíveis. A implementação da autenticação será feita utilizando os recursos nativos do framework Django, que oferece uma estrutura robusta para login, logout, gerenciamento de

sessões e controle de permissões por tipo de usuário (ex: somente o síndico tem acesso ao sistema). Esses recursos evitam falhas comuns de segurança e estão em conformidade com os princípios defendidos pelas metodologias baseada na norma ISO/IEC 2700.

A tela de login abaixo foi desenvolvida para oferecer autenticação segura ao sistema, permitindo somente o acesso a usuários cadastrados. O mecanismo de segurança é baseado no framework Django, que valida as credenciais utilizando a função de autenticação própria do Django e, em caso de erro, informa o usuário de maneira clara e objetiva.

Figura 6 - Tela de login desenvolvida em HTML (front-end) com back-end em Django para controle de acesso ao sistema



A imagem mostra a interface de login do Django Admin. No topo, há uma barra azul com o texto "Administração Django" em amarelo e um ícone de lua. Abaixo, há dois campos de entrada: "Nome de usuário:" com o valor "síndico" e "Senha:" com pontos para ocultar o texto. Um botão azul "Conecte-se" está posicionado abaixo dos campos.

Fonte: Autoria própria

A figura exibe a interface de login, onde o usuário pode inserir seu nome de usuário e senha. A interface utiliza uma combinação de HTML, CSS e Django Template Language para proporcionar uma experiência de uso intuitiva e segura, garantindo que apenas usuários autorizados acessem o sistema.

Código do Template de Login:

Figura 7 - Tela de Login com autenticação segura (back-end)

```

1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Login - Condomínio Bolívar</title>
7      <style>
8          body {
9              font-family: Arial, sans-serif;
10             background-color: #f4f4f4;
11             display: flex;
12             align-items: center;
13             justify-content: center;
14             height: 100vh;
15             margin: 0;
16         }
17         .login-container {
18             background-color: #fff;
19             padding: 30px;
20             border-radius: 8px;
21             box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
22             text-align: center;
23             width: 320px;
24         }
25         .login-container img {
26             width: 120px;
27             margin-bottom: 20px;
28         }
29         .login-container h2 {
30             margin-bottom: 20px;
31         }
32         .input-group {
33             margin-bottom: 15px;
34             text-align: left;
35         }
36         .input-group label {
37             display: block;
38             margin-bottom: 5px;
39         }
40         .input-group input {
41             width: 100%;
42             padding: 8px;
43             border: 1px solid #ccc;
44             border-radius: 4px;
45         }
46         .btn-login {
47             width: 100%;
48             padding: 10px;
49             background-color: #4CAF50;
50             color: white;
51             border: none;
52             border-radius: 4px;
53             cursor: pointer;
54         }
55         .btn-login:hover {
56             background-color: #45a049;
57         }
58         .error {
59             color: #D8000C;
60             margin-bottom: 15px;
61         }
62     </style>
63 </head>
64 <body>

```

```

65     <div class="login-container">
66
67         <h2>Acesso Seguro</h2>
68         {% if error_message %}
69             <div class="error">{{ error_message }}</div>
70         {% endif %}
71         <form method="POST" action="{% url 'login' %}">
72             {% csrf_token %}
73             <div class="input-group">
74                 <label for="username">Usuário</label>
75                 <input type="text" id="username" name="username" required>
76             </div>
77             <div class="input-group">
78                 <label for="password">Senha</label>
79                 <input type="password" id="password" name="password" required>
80             </div>
81             <button type="submit" class="btn-login">Entrar</button>
82         </form>
83     </div>
84 </body>
85 </html>

```

Fonte: Autoria própria

HTML (HyperText Markup Language): O código estrutura a página de login, definindo os elementos básicos, como cabeçalho, campos de entrada e botões.

CSS (Cascading Style Sheets): Estiliza a página, definindo a aparência dos elementos. Foram aplicados estilos para centralizar a interface, criar bordas arredondadas, aplicar sombras e definir cores para botões e textos.

Django Template Language: O template utiliza as tags do Django para incluir a diretiva `{% csrf_token %}` – essencial para prevenir ataques CSRF (Cross-Site Request Forgery). A condicional `{% if error_message %}` permite exibir uma mensagem de erro, caso a autenticação falhe.

Metodologia aplicada: Princípios de Segurança da Informação (baseados na OWASP).

Ferramenta: Implementação de autenticação de usuários via sistema de login do Django com gerenciamento de permissões (`@login_required`), além do uso de token CSRF para evitar ataques de falsificação de requisições.

Justificativa técnica: O Django oferece recursos nativos de segurança que seguem as boas práticas recomendadas para aplicações web, protegendo o sistema contra ataques como XSS, CSRF e SQL Injection.

2.4.5. Garantir a organização das solicitações e reduzir o risco de esquecimentos ou perdas de informações

A categorização das solicitações promove a organização do fluxo de atendimento, com registro claro e rastreável. Neste contexto, os princípios do Lean Thinking foram novamente aplicados para eliminar atividades que não agregam valor, como retrabalho ou buscas manuais. Seu objetivo é aumentar a eficiência e a produtividade, entregando mais valor aos clientes, onde assegura-se que o mesmo terá as facilidades, agilidade e segurança com todas as suas demandas, principalmente as prioritárias. (WOMACK; JONES, 2004).

A imagem abaixo exibe o código do modelo Chamado implementado em Django, que define a estrutura dos registros de solicitações.

Figura 8 - Definição técnica do modelo Chamado

```
class Chamado(models.Model):
    descricao = models.CharField(max_length=255)
    status = models.TextField(max_length=50, choices=[
        ('URGENTE', 'URGENTE'),
        ('EMERGENCIA', 'EMERGENCIA'),
        ('EM ANDAMENTO', 'EM ANDAMENTO'),
        ('CONCLUIDO', 'CONCLUIDO')
    ], default='EM ANDAMENTO')
    data_criacao = models.DateTimeField(auto_now_add=True)
    data_conclusao = models.DateTimeField(null=True, blank=True)
```

Fonte: Autoria própria

Neste modelo há os seguintes tópicos:

- **descricao:** Campo textual que armazena a descrição do chamado.
- **status:** Campo com opções pré-definidas (*choices*) que determina o estado da solicitação, com valores como *URGENTE*, *EMERGENCIA*, *EM ANDAMENTO* *RESOLVIDO*, sendo o padrão *EM ANDAMENTO*.
- **data_criacao:** Campo que registra automaticamente o momento da criação do registro.
- **data_conclusao:** Campo opcional para registrar a data de conclusão do chamado.

Esta definição técnica demonstra como o Django ORM estrutura e gerencia os dados de forma organizada, permitindo a manipulação das informações no banco de dados.

Metodologia aplicada: Controle de versão distribuído.

Ferramenta: Utilização do Git para versionamento e controle de histórico do projeto, com armazenamento remoto no GitHub.

Justificativa técnica: O Git possibilita a rastreabilidade das alterações no código-fonte, facilita o trabalho em equipe e garante recuperação de versões anteriores em caso de falhas, o que é essencial em projetos de software.

Durante toda a fase de desenvolvimento, o síndico Vinícius foi envolvido de forma ativa no processo, sendo consultado para validação de funcionalidades a cada nova entrega. Esse ciclo iterativo de feedback reflete o uso de práticas da metodologia Scrum, que foi aplicada de forma adaptada para ciclos de desenvolvimento curtos com entregas incrementais.

Essa abordagem colaborativa e técnica possibilitou o alinhamento eficaz entre as necessidades reais da gestão condominial e os recursos tecnológicos disponíveis, resultando em um sistema simples, funcional, adaptável e centrado no usuário final, cumprindo os objetivos do projeto com eficiência.

2.5 RESULTADOS PRELIMINARES: SOLUÇÃO INICIAL

O processo de escuta ativa iniciou-se com a coleta de informações por meio de reuniões online com o síndico. Durante esses encontros, foi possível compreender com mais profundidade as reais necessidades do cliente. Inicialmente, o grupo havia proposto o desenvolvimento de um sistema com acesso tanto para moradores quanto para o síndico. No entanto, após uma conversa mais detalhada, identificou-se que os moradores já utilizavam um aplicativo próprio da administradora do condomínio (Conviver MRV) para envio de solicitações.

O verdadeiro desafio estava na dificuldade do síndico em organizar e acompanhar as solicitações recebidas, que chegavam de forma fragmentada pela plataforma Bragi. Não havia uma maneira eficiente de classificar, marcar ou acompanhar o andamento dessas demandas.

Esse cenário causava perda de tempo, sobrecarga no controle manual via planilhas e risco de esquecimentos.

Diante disso, a equipe redefiniu o escopo do projeto e passou a desenvolver uma solução digital de uso exclusivo para o síndico, funcionando como um planner digital personalizado, com foco na organização e acompanhamento das solicitações condominiais.

Como primeiro passo, o grupo optou por uma abordagem prática e objetiva, iniciando o protótipo com as tecnologias HTML, CSS e JavaScript, para estruturar as primeiras interfaces. O desenvolvimento foi realizado no Visual Studio Code (VS Code), um ambiente leve e amplamente utilizado na área de tecnologia por sua flexibilidade, suporte a extensões e integração com sistemas de controle de versão como o Git.

O protótipo inicial possibilitou testes de usabilidade e coleta de feedback, o que orientou a equipe a evoluir para uma solução mais robusta. A versão atual do sistema foi desenvolvida com o framework Django, utilizando Python como linguagem principal no backend, e o banco de dados relacional MySQL para garantir segurança, integridade e organização dos dados.

Até o momento temos funcionalidades essenciais solicitadas pelo síndico, como:

- Cadastro de solicitações com descrição;
- Acompanhamento do status (pendente, em andamento, Resolvido);
- Classificação por nível de urgência (cores: vermelho, amarelo, verde);
- Filtros e ordenações por tipo, data ou prioridade;
- Acesso restrito por login e senha para maior segurança.

A utilização do Design Thinking e da metodologia ágil Scrum permitiu a construção de um sistema centrado no usuário e adaptado ao contexto real do cliente. O feedback contínuo do síndico foi essencial para ajustes nas funcionalidades, resultando em uma ferramenta funcional, acessível e eficiente para a gestão condominial.

O layout inicial foi projetado para facilitar a visualização das demandas. A interface contém os seguintes elementos:

- Campo Origem: permite colocar de qual apartamento e bloco está se originando a solicitação;
- Campo de título: permite ao síndico nomear a demanda de forma clara;
- Campo de descrição: utilizado para detalhar a solicitação;
- Campo de data: registra quando a demanda foi recebida;
- Campo de upload de arquivos: possibilita anexar imagens, documentos ou vídeos que complementam a solicitação;
- Botão “Salvar Demanda”: registra a solicitação no sistema;
- Listagem de demandas: exibe cada solicitação registrada logo abaixo, de forma sequencial;
- Botão “Marcar como Resolvido”: altera o status de solicitação para “Resolvido”.

Figura 9 - Tela inicial do site desenvolvido para o síndico

A interface, intitulada "Chama o Síndico 2.0", apresenta um formulário para registrar uma nova demanda. Os campos incluem: "Origem" (campo de texto), "Título:" (campo de texto), "Descrição:" (campo de texto com ícone de expansão), "Data:" (campo com máscara "dd/mm/aaaa" e ícone de calendário) e "Anexar Arquivo:" (botão "Escolher arquivo" e texto "Nenhum arquivo escolhido"). Abaixo do formulário, há um botão laranja "Salvar Demanda".

Abaixo do botão, a seção "Demandas Salvas" exibe uma única entrada. Os dados desta entrada são:

- academia suja**
- Descrição:** encontramos a academia imunda
- Data:** 2025-04-06
- Arquivo:** Nenhum arquivo anexado
- Status:** resolvido

À direita da entrada, há um botão laranja "Marcar como Pendente". Na base da seção, há um botão laranja "Apagar Demandas".

Fonte: Autoria própria

Essa organização foi pensada para otimizar a gestão de demandas e facilitar o acompanhamento por parte do síndico.

A funcionalidades básicas foram integradas com sucesso, e a exibição das demandas em tempo real foi validada internamente entre os membros do grupo. Essa versão ainda utiliza

armazenamento local (LocalStorage) como base de dados temporária, viabilizando o teste de funcionalidades.

A primeira devolutiva do protótipo foi apresentada ao síndico para coleta de feedback através de uma reunião online através do Google Meet, foi feito a análise e passado as possíveis melhorias e ajustes. Com base na devolutiva, foram implementadas novas etapas, como:

- Classificação por prioridade (através de cores);
- Integração com banco de dados relacional;
- Mudança da cor do site de acordo com a logo do condomínio;
- Ordenação das solicitações que já foram concluídas.
- Filtro de demandas;

A equipe planeja ainda adaptar o layout para garantir total responsividade e experiência amigável ao usuário.

REFERÊNCIAS

NOAH SYSTEMS. **Bragi - Plataforma de gestão condominial**. Disponível em: <https://noahsystems.com.br/bragi/>. Acesso em: 08 abr. 2025.

COVEY, S. R. **Os 7 hábitos das pessoas altamente eficazes**. Rio de Janeiro: BestSeller, 2004.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 27001:2013 - **Tecnologia da informação - Técnicas de segurança - Sistemas de gestão de segurança da informação - Requisitos**. Rio de Janeiro: ABNT, 2013.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

HOLANDA, Márcio de Souza. **Desenvolvimento Web com Django**. São Paulo: Novatec Editora, 2021.

OLIVEIRA, João Carlos de. **Gestão condominial eficiente: tecnologias e boas práticas para síndicos modernos**. São Paulo: Editora Síndico Digital, 2020.

REIS, Marina; SOUSA, Thiago; FONSECA, Luana. **Sistemas de gestão aplicados ao cotidiano: eficiência na administração de demandas**. Belo Horizonte: Editora Ágil, 2019.

REZENDE, Denis A.; ABREU, Aline F. **Tecnologia da Informação aplicada a sistemas de informação empresariais: o papel estratégico da informação e dos sistemas de informação nas empresas**. São Paulo: Atlas, 2000.

MILETTO, Gilberto; BERTAGNOLLI, Luciano. **Desenvolvimento de Interfaces Web**. 1. ed. São Paulo: Editora Exemplo, 2020. p. 72.

NIELSEN, Jakob. **Usability Engineering**. Boston: Academic Press, 1993.

SCHWABER, Ken; BEEDLE, Mike. **Agile software development with Scrum**. Upper Saddle River: Prentice Hall, 2002.

PRESSMAN, Roger S. **Engenharia de Software**. 8. ed. São Paulo: McGraw Hill Brasil, 2016. p. 39.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. p. 123.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide™ – The Definitive Guide to Scrum: The Rules of the Game**. 2020. Disponível em: <https://scrumguides.org>. Acesso em: 08 abr. 2025.

HOLOVATY, Adrian; KAPLAN-MOSS, Jacob. **Django: a web framework for perfectionists with deadlines**. 1. ed. [S.l.]: [s.n.], 2009. p. 1.

CORONEL, Carlos; MORRIS, Steven. **Sistemas de banco de dados: projeto, implementação e administração**. 9. ed. São Paulo: McGraw-Hill, 2014. p. 12.

CHACON, Scott; STRAUB, Ben. **Pro Git**. 2. ed. Rio de Janeiro: Novatec, 2014. Disponível em: <https://git-scm.com/book/pt-br/v2>. Acesso em: 08 abr. 2025.

BROWN, Tim. **Design Thinking: Uma metodologia poderosa para decretar o fim das velhas ideias**. Rio de Janeiro: Alta Books, 2010, p.4.

OSBORN, Alex Faickney. **Imaginação Aplicada: Princípios e Procedimentos para a Solução Criativa de Problemas [Applied Imagination: Principles and Procedures of Creative Problem-Solving]**. New York: Charles Scribner's Sons, 1953.

WOMACK, James P.; JONES, Daniel T. **A mentalidade enxuta nas empresas: elimine o desperdício e crie riqueza**. 2. ed. Rio de Janeiro: Elsevier, 2004.

PRESSMAN, Roger S. **Engenharia de Software**. 8. ed. São Paulo: McGraw Hill Brasil, 2016.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados – Lógica CRUD**. 8. ed. Rio de Janeiro: Campus, 2004.