UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CAMPUS PAU DOS FERROS
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA

ALAN ALMEIDA DA SILVA - 2024011434
BRUNO VICTOR PAIVA DA SILVA - 2020011045
CAIO VINICIUS PESSOA GOMES - 2024011364
DALTON FIRMINO CAMPOS - 2024011414
PAULINA JULIA COSTA DE OLIVEIRA - 2023023644

QUARTZO - SISTEMA DE GERENCIAMENTO DE IMOBILIÁRIA

PAU DOS FERROS – RN
2025

1. INTRODUÇÃO

Este documento apresenta o plano de testes para um sistema voltado para imobiliárias. O objetivo é garantir que o sistema atenda aos requisitos funcionais e não funcionais, detectando possíveis falhas e garantindo que todos os componentes funcionem corretamente, tanto isoladamente quanto em conjunto.

2. OBJETIVOS

O principal objetivo dos testes é verificar:

- A garantia do comportamento em conformidade com as restrições do sistema.
- O funcionamento adequado das unidades de código (testes unitários).
- A integração correta entre diferentes componentes (testes de integração).
- A funcionalidade do sistema sob cenários reais de uso (testes de sistema).

3. ESCOPO

Os testes cobrirão:

1. **Testes Unitários**: Foco em *models*, *forms* e *views* do sistema.
2. **Testes de Integração**: Foco na interação entre diferentes componentes como Imóveis, Contratos, pagamento e usuários.
3. **Testes de Sistemas**: Foco em avaliar o sistema como um todo, verificando não apenas os aspectos funcionais, mas também não funcionais.

4. ABORDAGEM DOS TESTES

4.1 Testes Unitários

Os testes unitários visam testar cada componente do sistema de forma isolada. As seguintes áreas serão cobertas:

- **Modelos**: Testar Cadastros de usuários com emails iguais

  Exemplo de teste: um usuário não pode ser cadastrado com o mesmo email já cadastrado no banco.

- **Validações**: Garantir que formulários e validações funcionam conforme esperado, como impedir o cadastro de emails iguais.

  Exemplo de teste: Testar se um formulário de cadastro de usuário falha ao tentar salvar um usuário duplicado.
- **Views**: Garantir que as *views* da API retornem o código de *status* HTTP correto e realizem a ação apropriada.

  Exemplo de teste: Verificar se uma requisição POST para cadastrar um novo imóvel retorna um status “201 *created*” e se o formulário é salvo no banco de dados.

4.2 Testes de Integração

Os testes de integração garantem que diferentes partes do sistema funcionem corretamente juntas. A integração entre *models* e *views* será testada em cenários reais de uso:

- **Cadastro e Gerenciamento de Usuários Administradores**: Verificar a integração entre o módulo de autenticação, o banco de dados de usuários e a interface de gerenciamento de usuários.

  Exemplo de teste: Criar um novo usuário administrador e verificar se as informações são corretamente armazenadas no banco de dados e refletidas na interface.
- **Cadastro e Gerenciamento de Imóveis**: Assegurar que o processo de cadastro, edição e remoção de imóveis funcione de maneira integrada entre a interface do sistema e o banco de dados.

  Exemplo de teste: Editar os dados de um imóvel cadastrado e confirmar se as alterações são exibidas corretamente na lista de imóveis.
- **Renovação de Contratos:** Validar a interação entre os módulos de contratos, cálculo de prazos e atualização no banco de dados.

  Exemplo de teste: Renovar um contrato e verificar se o novo prazo e as condições de pagamento são atualizados corretamente e exibidos no sistema.

- **Geração de Relatórios Financeiros:** Testar a integração entre os dados financeiros armazenados e a funcionalidade de geração e exportação de relatórios.

  Exemplo de teste: Gerar um relatório financeiro com filtro por período e verificar se os valores de aluguéis, vendas e despesas refletem os dados cadastrados no sistema.

4.3 Testes de Sistema

Os testes de sistema têm como objetivo validar o sistema como um todo, garantindo que todos os componentes e módulos funcionem corretamente quando integrados. Nesse tipo de teste, verifica-se se o sistema atende tanto aos requisitos funcionais (o que o sistema deve fazer) quanto aos requisitos não funcionais (como o sistema deve se comportar).

- **Cadastro de Usuários**: Garantir que o fluxo de registro, edição e exclusão de usuários funcione conforme esperado.

  Exemplo de teste: Registrar um novo usuário, editar suas informações (como *e-mail* ou nível de acesso) e excluir o usuário.
- **Registro de Contratos**: Verificar o fluxo completo de criação, edição e cancelamento de contratos imobiliários.

  Exemplo de teste: Criar um contrato com um imóvel e um cliente, editar o prazo do contrato e cancelá-lo, confirmando que o status no sistema é atualizado corretamente.
- **Consulta de Imóveis Disponíveis**: Testar o fluxo de busca e filtro de imóveis, garantindo que os resultados exibidos correspondam aos critérios selecionados.

  Exemplo de teste: Realizar uma busca filtrando por tipo de imóvel, e localização, verificando se os resultados apresentados estão corretos.
- **Renovação de Contratos:** Simular a renovação de um contrato imobiliário, garantindo que os novos prazos e valores sejam aplicados.

  Exemplo de teste: Renovar um contrato de aluguel e verificar se o prazo e o valor mensal atualizados aparecem corretamente no sistema.

- **Geração de Relatórios:** Validar se o sistema permite a geração de relatórios conforme os filtros escolhidos.

  Exemplo de teste: Gerar um relatório de movimentação financeira de um período específico e verificar se os dados apresentados correspondem às transações registradas no sistema.

5. CRITÉRIOS DE ACEITAÇÃO

- **Testes Unitários**:

  Todos os métodos testados devem retornar os resultados esperados;

  O código deve passar por validações sem erros em todos os testes de *model*, *form* e *view*.
- **Testes de Integração**:

  Devem validar o funcionamento correto de todos os fluxos de negócio críticos, incluindo o registro de imóveis, a geração de contratos de aluguel sem inconsistências e o cálculo correto de pagamentos e taxas administrativas.

  A integração entre os módulos do sistema deve ocorrer sem falhas, garantindo que os dados fluam corretamente entre as diferentes partes do sistema, como cadastro de imóveis, gestão de contratos e relatórios financeiros.
- **Testes de Sistema**:

  O sistema deve executar todos os fluxos principais sem erros, como o cadastro, edição e exclusão de imóveis, o gerenciamento de contratos de aluguel e a geração e exportação de relatórios em diferentes formatos.

  As permissões de acesso devem estar configuradas corretamente, permitindo ou restringindo ações de acordo com o perfil do usuário, como, por exemplo, acesso total para administradores e acesso limitado para agentes imobiliários.

6. FERRAMENTAS

Para a execução dos testes, serão utilizadas as seguintes ferramentas:

- **Django Test Framework**: Para escrever e rodar testes unitários, de integração e de sistema.
- **Pytest**: Como alternativa ao sistema de testes padrão do Django.

7. CONCLUSÃO

O plano de testes descrito visa garantir que o sistema de gerenciamento imobiliário (Quartzo) funcione corretamente em todos os níveis. Com a implementação de testes unitários, de integração e de sistema, será possível assegurar a qualidade do código e a funcionalidade do sistema, além de detectar e corrigir falhas durante o desenvolvimento.