INSTITUTO FEDERAL DO PARANÁ

ANÁLISE E PROJETO DE SISTEMAS

Marina Girolimetto

DOCUMENTAÇÃO APLICATIVO WEB “KITNET”

(Sistema de Locação de Imóveis)

Alunos:

CHARLES EDUARDO

WILIAN FRAGATA

PALMAS JULHO DE 2025

Sumário

Sistema de Locação de Imóveis – “KitNet”

Análise e Projeto de Sistemas
Alunos: Charles Eduardo, Wilian Fragata

Documentação sistema Kitnet

1. Escopo

1.1 Introdução

O presente documento descreve o escopo do projeto de desenvolvimento de um sistema web para locação de imóveis denominado Kitnet, ele terá seu foco em cidades de pequenas onde, geralmente, não são utilizados meios massivos de locação via dispositivos web ou moveis e o contato ainda, na maioria das vezes, se faz pessoalmente. Em resumo, queremos mudar a forma de como fazer negócios imobiliários em pequenas cidades, trazendo mais profissionalidade mesmo em cidades onde o “boca a boca” ainda traz muita dor de cabeça!

1.2 Objetivos do Projeto

Criação de Perfis públicos: Os Usuários poderão criar um perfil seguro possibilitando que as pessoas certas possam, diretamente no site, demostrar interesse através de uma visita. Todo cadastro de perfil possui meios para que sejam confirmadas as identidades legais dos usuários, fazendo com que cada conversa e negócio seja seguro para ambos.
Permitir o cadastro detalhado de imóveis: Para locatários o site funcionara como uma vitrine para seus negócios dando a oportunidade para fazer postagens de suas propriedades bem como descrevê-las, utilizando fotos, adicionando filtros, regras, valores e qualquer descrição que achar necessário. O site também torna mais seguro a forma como podem captar seus clientes, pois ofereceremos a ele uma ficha técnica de seus

locadores, preenchida previamente com toda a sorte de dados possíveis para que se possa escolher dentre muitos, qual será o novo inquilino.

Facilitar a busca personalizada de imóveis: Para Locadores, a plataforma oferece um amplo acervo de casas, dando-lhes a chance de procurar o tipo perfeito que se encaixe em sua necessidade, poderão filtrar por número de cômodos, se possuem ou não garagem, muros, quantidade de banheiros etc. O locador poderá também ter acesso ao manual do inquilinato, contratos feitos bem como todos os documentos, prazos e contratos que existirem na locação.

Criar ambiente seguro e organizado para negociações: Para locatários o site torna mais seguro a forma como podem captar seus clientes, pois ofereceremos a ele uma ficha técnica de seus locadores, preenchida previamente com toda a sorte de dados possíveis para que se possa escolher dentre muitos, qual será o novo inquilino. Para locadores, o site torna seguro a transação, sabendo que onde e com quem fará negócio, podendo fazer reports caso tiver alguma dúvida em relação a fraudes.

1.3 Escopo Funcional

Login e autenticação (autenticar): O sistema deve validar que o Email e senha foram corretamente preenchidos. Se as credenciais estiverem corretas, o usuário deve ser redirecionado para seu perfil. Se as credenciais estiverem incorretas, uma mensagem de erro “Email ou senha inválidos” deverá aparecer. O sistema terá um link “esqueci minha senha” após 3 tentativas de login sem sucesso, a conta deve ser bloqueada temporariamente.

Cadastro e gestão de imóveis (cadastrar): O **locador** poderá cadastrar novos imóveis, associando-os ao seu perfil. O cadastro deve permitir a inclusão de detalhes como fotos, endereço, valor do aluguel, regras da casa e características (nº de quartos, garagem etc.). O locador também poderá editar, desativar (marcar como "alugado") ou remover seus anúncios.

Pesquisa de imóveis por filtros (pesquisar): O **locatário** poderá buscar imóveis disponíveis na plataforma. O sistema deve oferecer filtros para refinar a busca, como:

faixa de preço, número de quartos, banheiros, e se possui garagem. Os resultados devem ser exibidos em formato de lista ou galeria, com foto principal e informações essenciais.

Sistema de favoritos (favoritar): O locatário poderá marcar um imóvel como "favorito" durante a pesquisa. O sistema deve manter uma lista pessoal de favoritos para cada usuário, permitindo que ele acesse rapidamente os imóveis de maior interesse para consultas futuras ou para iniciar uma negociação.

Área administrativa (administrar): Uma área restrita para o administrador do "KitNet". O administrador poderá visualizar e gerenciar os cadastros de usuários (aprovar, bloquear), moderar anúncios de imóveis (remover conteúdo inadequado) e visualizar relatórios básicos de uso da plataforma. Esta área é fundamental para garantir a segurança e a qualidade do sistema.

1.4 Escopo Não Funcional

Usabilidade e Acessibilidade: O sistema deve ser intuitivo e fácil de usar, mesmo para usuários com pouca familiaridade com tecnologia, que é um foco do projeto em cidades pequenas. Um novo locador deve ser capaz de publicar seu primeiro anúncio em menos de 10 minutos após o cadastro. A interface deve seguir padrões de design consistentes em todas as telas.

Desempenho (Performance): O sistema deve ser rápido e responsivo para não frustrar os usuários e transmitir profissionalismo. As páginas de pesquisa e os perfis de imóveis devem carregar completamente em menos de 3 segundos em uma conexão de internet móvel 4G. As buscas com filtros devem retornar resultados em menos de 2 segundos.

Responsividade: A interface deve se adaptar perfeitamente a diferentes tamanhos de tela, garantindo uma boa experiência tanto em computadores (desktops) quanto em celulares (móveis)**:** Todos os elementos da interface, incluindo formulários, botões e imagens, devem ser totalmente funcionais e legíveis em telas com largura a partir de 360 pixels, sem a necessidade de zoom horizontal.

Segurança: A segurança é um pilar do projeto para criar um "ambiente seguro". Isso envolve proteger os dados dos usuários e as comunicações dentro da plataforma. Além da autenticação segura com senhas criptografadas (usando algoritmos fortes como

Bcrypt), todas as comunicações entre o navegador e o servidor devem ser protegidas com certificado SSL/TLS (HTTPS). O sistema deve ter mecanismos de proteção contra ataques comuns como XSS (Cross-Site Scripting) e SQL Injection.

Confiabilidade e Disponibilidade: O sistema deve estar disponível e funcionando corretamente a maior parte do tempo para que os usuários confiem na plataforma. O sistema deve ter uma meta de disponibilidade (uptime) de 99.5% do tempo. Além disso, o processo de backup dos dados críticos (informações de usuários, imóveis e mensagens) deve ser realizado automaticamente a cada 24 horas, com os backups armazenados em um local geograficamente distinto do servidor principal.

1.5 Tecnologias Utilizadas:

- **Linguagem e Plataforma**
  - Node.js – ambiente de execução JavaScript server-side
  - TypeScript – linguagem tipada que compila para JavaScript
- **Framework e Ferramentas Web**
  - Next – framework web para criação de frontend componentizado
- **Banco de Dados e ORM**
  - Hibernate, Mysql:8.0
- **Banco de dados relacional**
  - MySql & Dockerizado(Docker).
- **Segurança e Autenticação**
  - bcryptjs – para hash de senhas
  - jsonwebtoken (JWT) – para autenticação de usuários com tokens

- **Ferramentas de Desenvolvimento**
  - Docker – recarrega automaticamente o servidor em desenvolvimento
  - ts-node-dev – executa arquivos TypeScript com hot reload
  - dotenv – para carregar variáveis de ambiente de um arquivo .env

- **Outras Ferramentas de Desenvolvimento:**
  - Hibernate OR, Gradle, TailWind, Shad CN Ui.

1.6 Stakeholders do Projeto

- Locadores, Locatários, administradores da página, desenvolvedores, Professora Marina, Professor Eduardo.

1.7 Exclusões do Escopo

Integração com meios de pagamento (ex: Pix, cartão): O sistema não realiza transações financeiras, cobranças automáticas ou integração com gateways de pagamento. A negociação e o pagamento entre locador e locatário ocorrem **fora da plataforma**.
Chat interno entre locador e locatário: Não há ferramenta de comunicação direta via chat ou mensagens privadas. Contatos ou dúvidas devem ser tratados por **meios externos**, como e-mail, WhatsApp ou telefone.
Sistema de avaliações ou notas: O sistema ainda **não permite avaliar** ou deixar comentários sobre as kitnets ou seus proprietários. Não há controle de reputação, nota média ou feedback público.
Painel de administração com gráficos e dashboards: Não há dashboards visuais com métricas, gráficos de uso ou relatórios de desempenho. A administração é feita de forma básica, com base em **dados brutos** (listagens simples).
Gerenciamento financeiro (boletos, relatórios contábeis): O sistema **não gera boletos bancários**, notas fiscais, nem relatórios contábeis para locadores. Esse tipo de controle deve ser feito com ferramentas externas.

1.8 Principais Riscos

Tabela de Riscos com Classificação de Impacto:

Nº	Risco	Tipo	Impacto	Justificativa
R1	Falhas na segurança de autenticação	Técnico	Alto	Pode permitir acessos não autorizados a dados sensíveis.
R2	Ausência de backup automatizado	Técnico	Alto	Pode causar perda definitiva de dados importantes do sistema.
R3	Falta de escalabilidade	Técnico	Médio	Pode comprometer o sistema com aumento de usuários.
R4	Validações insuficientes de entrada	Técnico	Médio	Pode gerar dados inconsistentes, são reversíveis manualmente.
R5	Inexistência de testes automatizados	Técnico	Médio	Aumenta o risco de bugs, mas não impede o funcionamento imediato do sistema.
R6	Dependência de configuração correta do ambiente	Técnico	Baixo	Fácil de corrigir, desde que documentado corretamente.
R7	Dependência de poucos locadores ou usuários	Negócio	Alto	Sem adesão mínima, o sistema perde sua utilidade e viabilidade.
R8	Falta de diferenciais competitivos frente ao mercado	Negócio	Médio	Pode impactar a atração e retenção de usuários no médio prazo.
R9	Ausência de modelo de monetização	Negócio	Alto	Torna o projeto insustentável financeiramente.
R10	Riscos jurídicos decorrentes da intermediação imob.	Negócio	Médio	Pode gerar implicações legais em disputas contratuais entre locador e locatário.
R11	Sazonalidade e flutuação de demanda	Negócio	Baixo	Impacto localizado e previsível; pode ser compensado com estratégias de marketing.
R12	Dependência de um único desenvolvedor	Negócio/Técnico	Médio	Pode gerar descontinuidade caso o responsável se ausente.

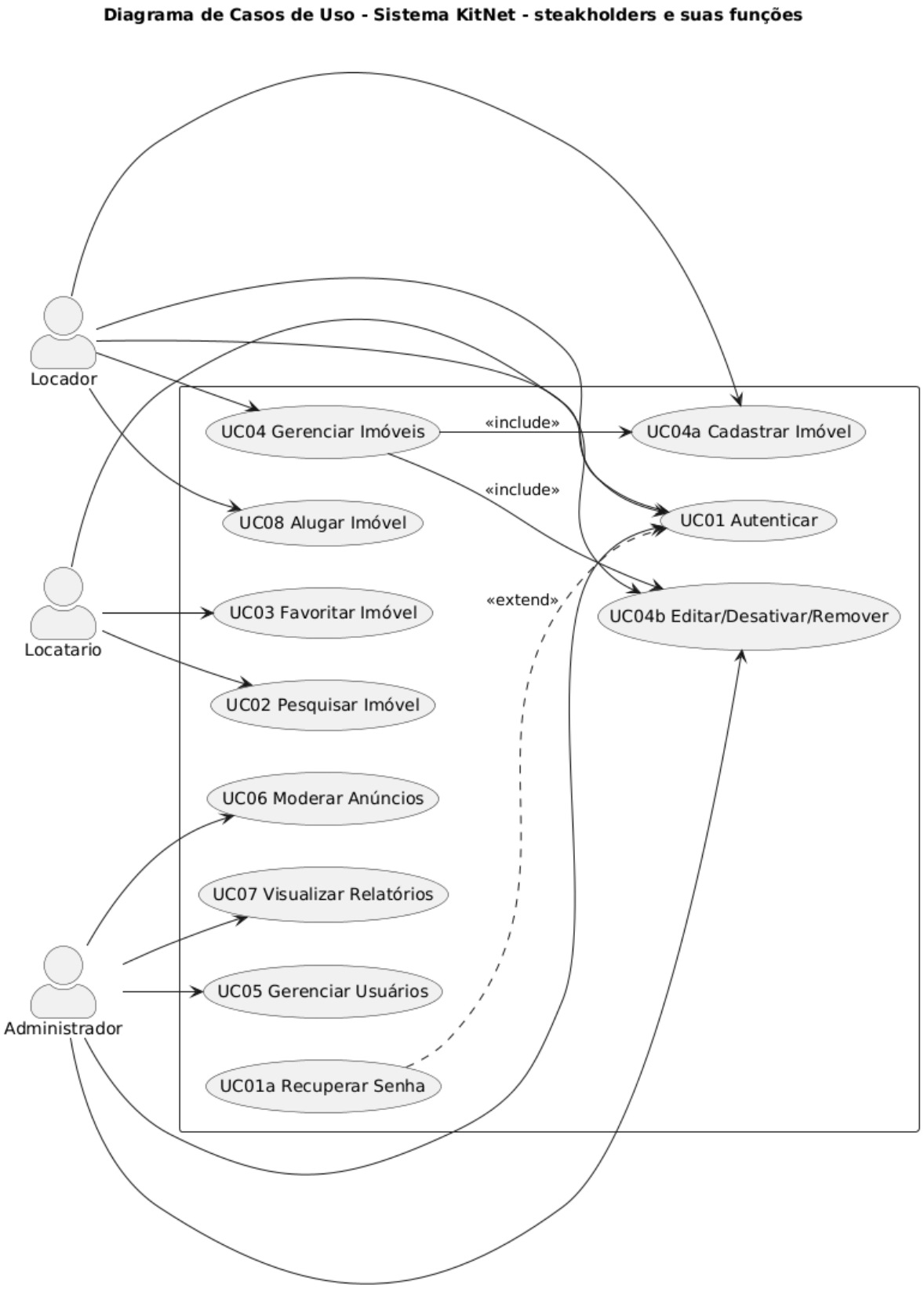
Plano de Mitigação de Riscos:

Risco (ID)	Estratégia de Mitigação
R1	Implementar autenticação com expiração de tokens, uso de HTTPS, logs de acesso e rotação de chaves JWT.
R2	Criar rotinas automáticas de backup e armazenar cópias em amb. externos
R3	Modularizar o sistema para facilitar futura escalabilidade com cache, etc.
R4	Aplicar validações robustas no backend (ex: validações Prisma).
R5	Adotar testes unitários com ferramentas como Jest e integração contínua.
R6	Documentar variáveis obrigatórias no README e criar scripts de verificação de ambiente.
R7	Investir em marketing digital e parcerias locais para atrair usuários iniciais.
R8	Priorizar o desenvolvimento futuro de diferenciais (msg, pagts, avaliações).
R9	Estabelecer modelo de negócio claro (ex: comissão por aluguel, mensalidade de uso da plataforma).
R10	Criar termos de uso e política de privacidade com apoio jurídico, deixando clara a natureza intermediadora.
R11	Planejar ações de divulgação em períodos de baixa demanda.
R12	Registrar conhecimento do projeto e estimular colaboração em equipe.

2. Diagramas

2.1 Diagrama de Casos de Uso:

2.1.1. Stakeholders e suas funções:



Documentação – Diagrama de Casos de Uso: Stakeholders e suas funções

Objetivo: Representar as principais interações dos usuários com o sistema, bem como os fluxos funcionais esperados para cada tipo de ator envolvido na plataforma.

Atores:
Locatário: usuário que acessa a plataforma com o objetivo de pesquisar, visualizar e favoritar kitnets disponíveis para aluguel.
Locador: proprietário que cadastra seus imóveis na plataforma e é responsável pela gestão dos mesmos, além de poder efetuar o aluguel de uma unidade a partir de uma reserva confirmada.
Administrador: usuário com perfil privilegiado que atua na moderação do sistema, podendo remover imóveis suspeitos, visualizar relatórios e gerenciar usuários.

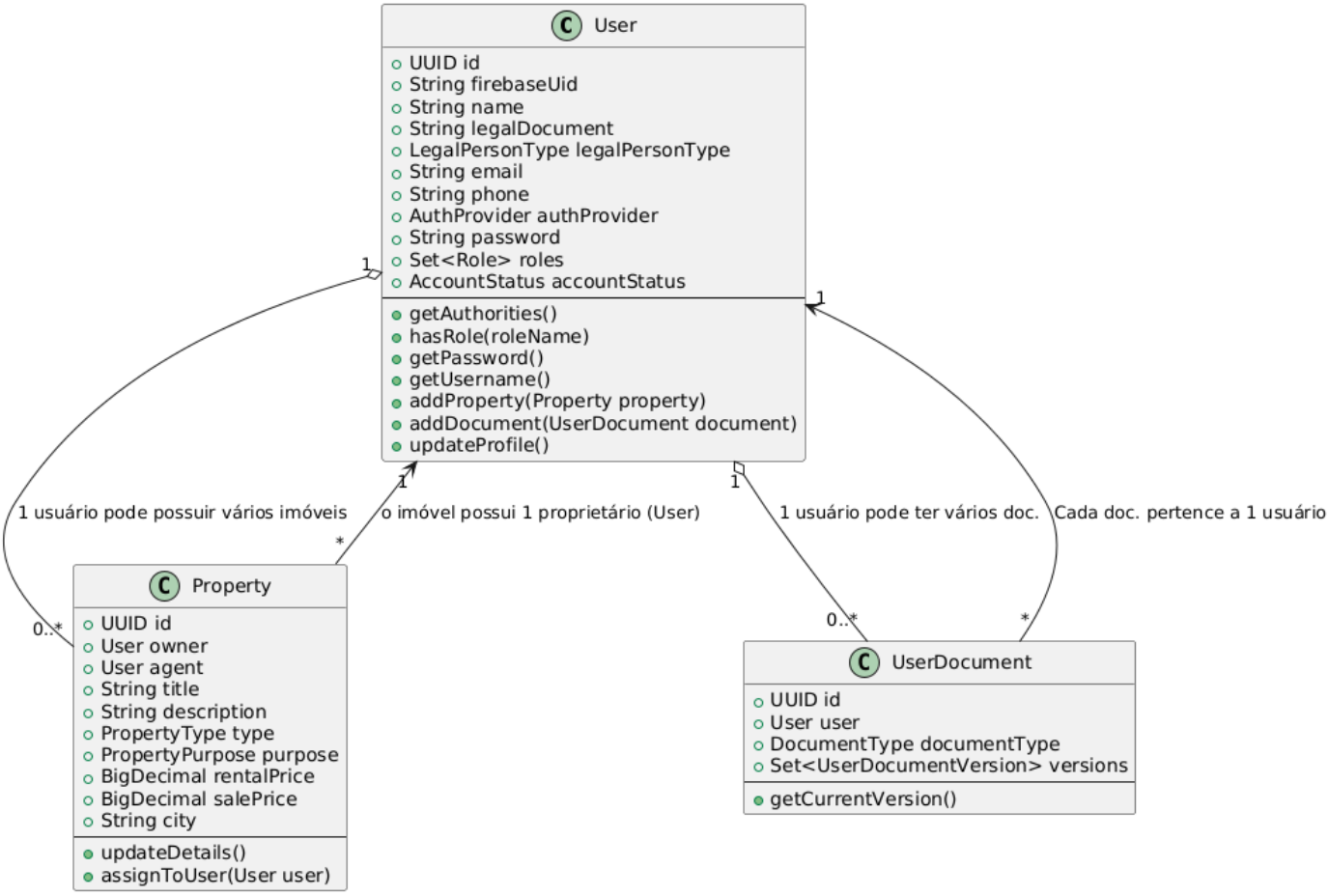
Casos de Uso Principais

UC01 – Autenticar: permite que qualquer usuário (locador, locatário ou administrador) acesse o sistema mediante fornecimento de credenciais válidas.
UC01a – Recuperar Senha: estende o caso de uso de autenticação para situações em que o usuário esqueceu sua senha.
UC02 – Pesquisar Imóvel: permite que locatários busquem imóveis disponíveis por filtros e critérios personalizados.
UC03 – Favoritar Imóvel: locatários podem marcar imóveis como favoritos para facilitar decisões futuras.
UC04 – Gerenciar Imóveis: funcionalidade central para locadores, possibilitando o cadastro, edição, desativação e exclusão de kitnets.
UC04a – Cadastrar Imóvel: inclui o preenchimento dos dados do imóvel, como localização, descrição e preço.
UC04b – Editar/Desativar/Remover: permite modificar, pausar ou excluir imóveis. Este caso de uso pode ser acionado tanto pelo locador quanto pelo administrador (em caso de suspeitas ou denúncias).
UC05 – Gerenciar Usuários: exclusivo do administrador, esse caso de uso permite realizar ações como remoção, bloqueio ou ajuste de permissões dos usuários.
UC06 – Moderar Anúncios: possibilita ao administrador revisar e intervir em conteúdos impróprios ou fraudulentos publicados na plataforma.
UC07 – Visualizar Relatórios: fornece ao administrador acesso a informações gerenciais como número de imóveis cadastrados, reservas efetuadas e acessos.
UC08 – Alugar Imóvel: permite que o locador finalize a locação de uma unidade habitacional após uma solicitação de reserva, caracterizando a efetivação do aluguel.

Relacionamentos

O caso de uso **"Recuperar Senha"** estende **"Autenticar"**, pois é acionado apenas quando a autenticação falha por senha esquecida. O caso de uso **"Gerenciar Imóveis"** inclui os casos de **"Cadastrar Imóvel"** e **"Editar/Desativar/Remover"**, uma vez que essas ações fazem parte da atividade principal. O caso **"Editar/Desativar/Remover"** é acessado tanto pelo locador (para seus próprios imóveis) quanto pelo administrador (para moderação de conteúdos).

2.2 Diagrama de Classes



2.2.1 Descrição simples em linguagem natural dos relacionamentos e métodos principais das classes

Todas as classes e atributos citados podem ser encontrados no diretório s**rc/main/java/com/kitnet/kitnet/model/`** do projeto, basta acessar esse caminho no projeto para visualizar os arquivos `User.java`, `Property.java` e `UserDocument.java`. Os atributos estão definidos diretamente nessas classes, e os métodos de acesso são gerados automaticamente pelo Lombok, facilitando a leitura e manutenção do código e também o que cada método recebe e retorna.

1. Classe User (Usuário):

Local: src/main/java/com/kitnet/kitnet/model/User.java

Descrição: A classe `User` representa os usuários do sistema, sejam eles inquilinos, proprietários, corretores ou administradores. Ela armazena informações pessoais, credenciais de acesso e dados de controle de conta.

Principais métodos:

getAuthorities(): Retorna as permissões do usuário.

Recebe: nada. **Retorna:** Collection<GrantedAuthority> (coleção de permissões do usuário).

hasRole(roleName): Verifica se o usuário possui determinado papel.

Recebe: RoleName roleName (nome do papel). **Retorna:** boolean (true se o usuário possui o papel, false caso contrário).

getPassword(), getUsername(): Métodos de autenticação.

Recebe: nada. **Retorna:** String (senha do usuário). e **Recebe:** nada. **Retorna:** String (e-mail do usuário, usado como nome de usuário).

addProperty(Property property): Associa um imóvel ao usuário (sugerido para clareza).

Recebe: Property property (objeto do imóvel). **Retorna:** void (não retorna valor, apenas associa o imóvel ao usuário).

addDocument(UserDocument document): Associa um documento ao usuário (sugerido para clareza). **Recebe:** UserDocument document (objeto do documento).

Retorna: void (não retorna valor, apenas associa o documento ao usuário).

updateProfile(): Atualiza os dados do perfil do usuário (sugerido para clareza).

Recebe: nada. **Retorna:** void (não retorna valor, apenas atualiza os dados do perfil).

Observação: Os métodos de acesso (getters, setters, equals, hashCode, toString) são gerados automaticamente pelo Lombok, facilitando a manutenção do código.

2. Classe Property (Propriedade)

Local: src/main/java/com/kitnet/kitnet/model/Property.java

Descrição: A classe `Property` representa os imóveis cadastrados no sistema. Cada imóvel possui informações detalhadas como título, descrição, localização, valores de aluguel/venda, entre outros.

Principais métodos:

updateProperty(): Atualiza os detalhes do imóvel (sugerido para clareza). **Recebe:** UUID, PropertyRequestDTO. **Retorna:** PropertyResponseDTO

assignToUser(User user): Atribui um proprietário ao imóvel (sugerido para clareza). **Recebe:** User user (objeto do usuário). **Retorna:** void (não retorna valor, apenas atribui o proprietário ao imóvel).

Observação: Assim como em User, os métodos de acesso são gerados automaticamente pelo Lombok.

3. Classe UserDocument (Documento do Usuário):

Local: src/main/java/com/kitnet/kitnet/model/UserDocument.java

Descrição: A classe `UserDocument` representa documentos enviados pelos usuários, como RG, CPF, comprovantes, entre outros. Cada documento pode ter várias versões (por exemplo, reenvio de um documento corrigido).

Principais métodos:

getCurrentVersion(): Retorna a versão atual do documento.

Recebe: nada. **Retorna:** Optional<UserDocumentVersion> (versão atual do documento, ou vazio se não houver versão).

Relacionamentos entre as classes:

- "Um usuário pode ser proprietário de vários imóveis": O atributo `owner` em `Property` referencia um objeto da classe `User`. Assim, um usuário pode possuir vários imóveis cadastrados.

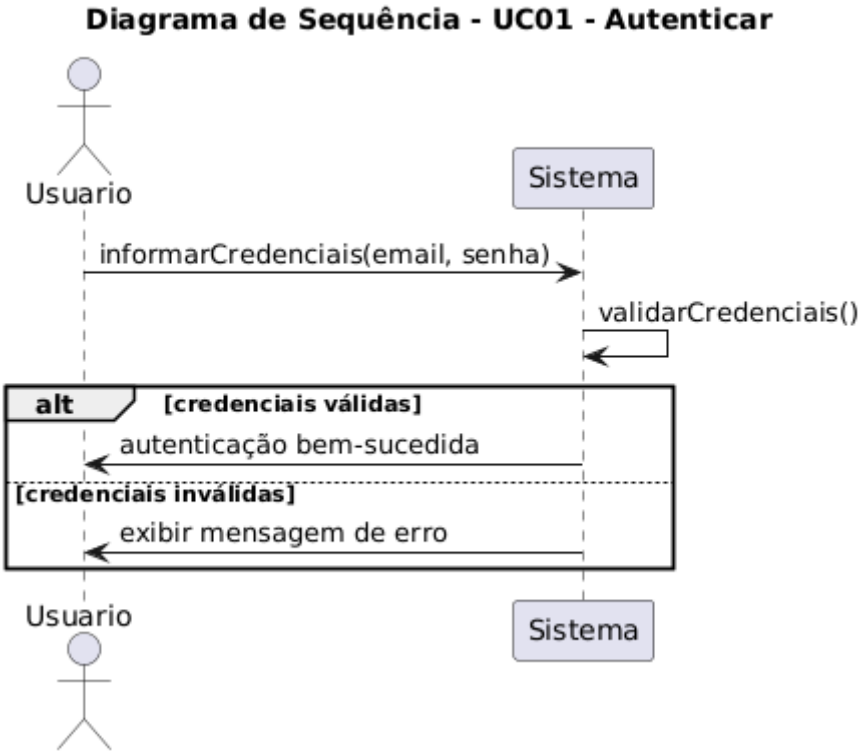
- "Um usuário pode ter vários documentos": O atributo `user` em `UserDocument` referencia um objeto da classe `User`. Dessa forma, cada usuário pode ter múltiplos documentos associados.

- "Cada imóvel tem um proprietário": O relacionamento entre `Property` e `User` é feito pelo atributo `owner`, garantindo que todo imóvel esteja vinculado a um usuário.

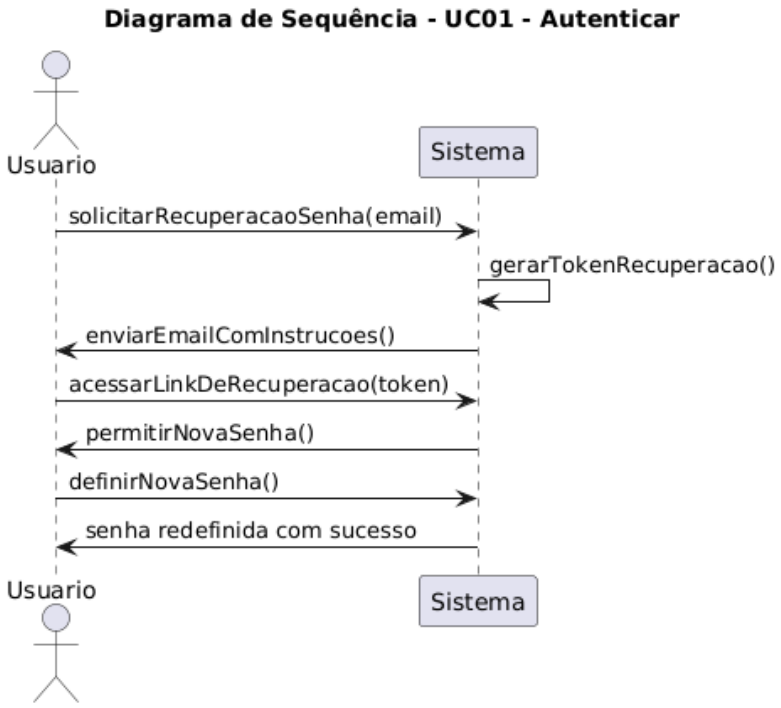
- "Cada documento pertence a um usuário": O relacionamento entre `UserDocument` e `User` é feito pelo atributo `user`, garantindo que todo documento esteja vinculado a um usuário específico.

2.3 Diagrama de Sequência:

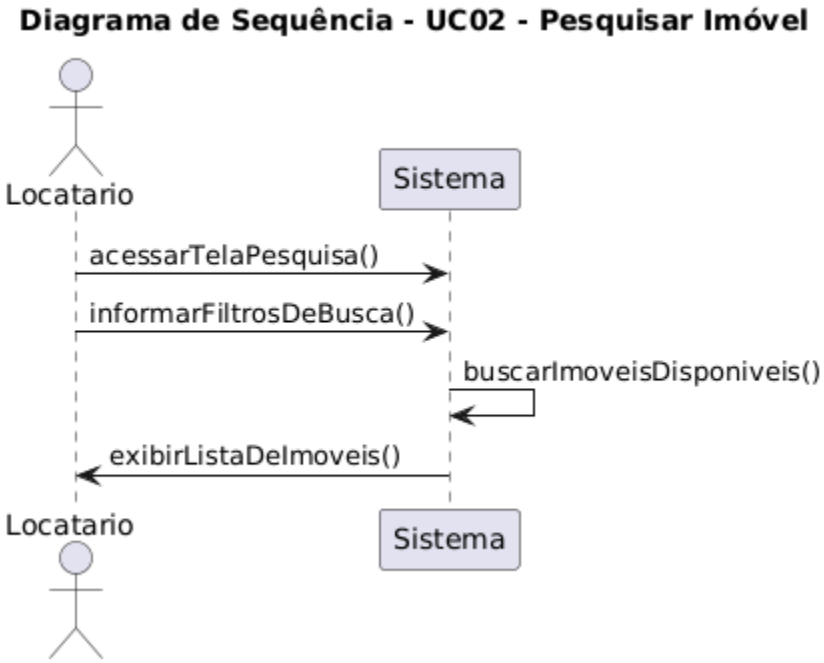
2.3.1: UC01 – Autenticar: permite que qualquer usuário (locador, locatário ou administrador) acesse o sistema mediante fornecimento de credenciais válidas.



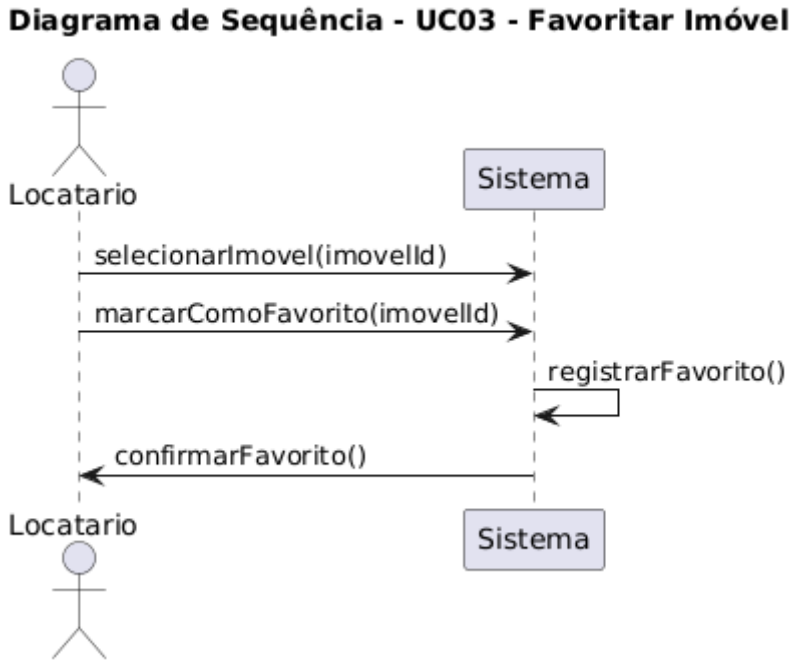
UC01a – Recuperar Senha: estende o caso de uso de autenticação para situações em que o usuário esqueceu sua senha.



UC02 – Pesquisar Imóvel: permite que locatários busquem imóveis disponíveis por filtros e critérios personalizados.

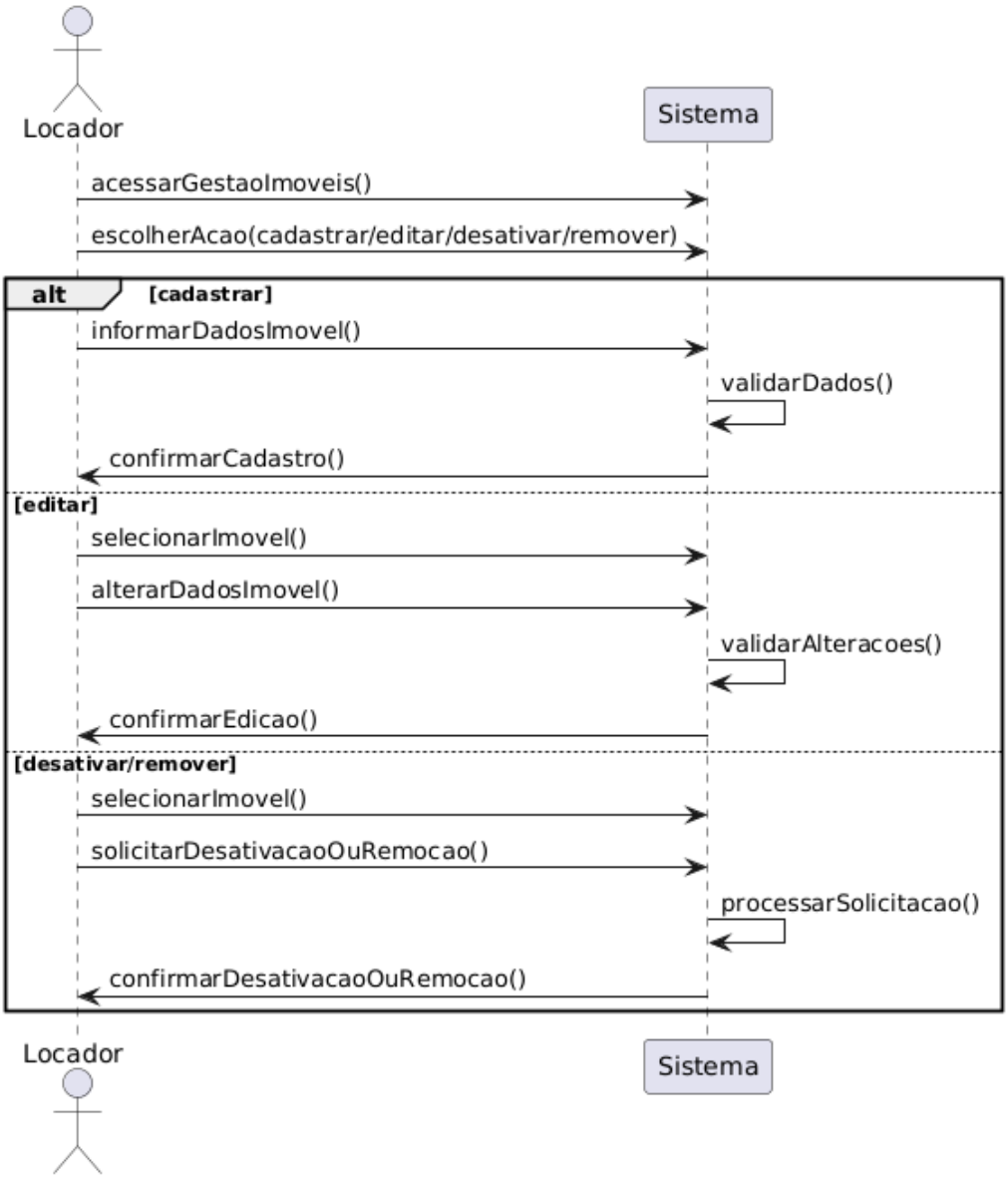


UC03 – Favoritar Imóvel: locatários podem marcar imóveis como favoritos para facilitar decisões futuras.

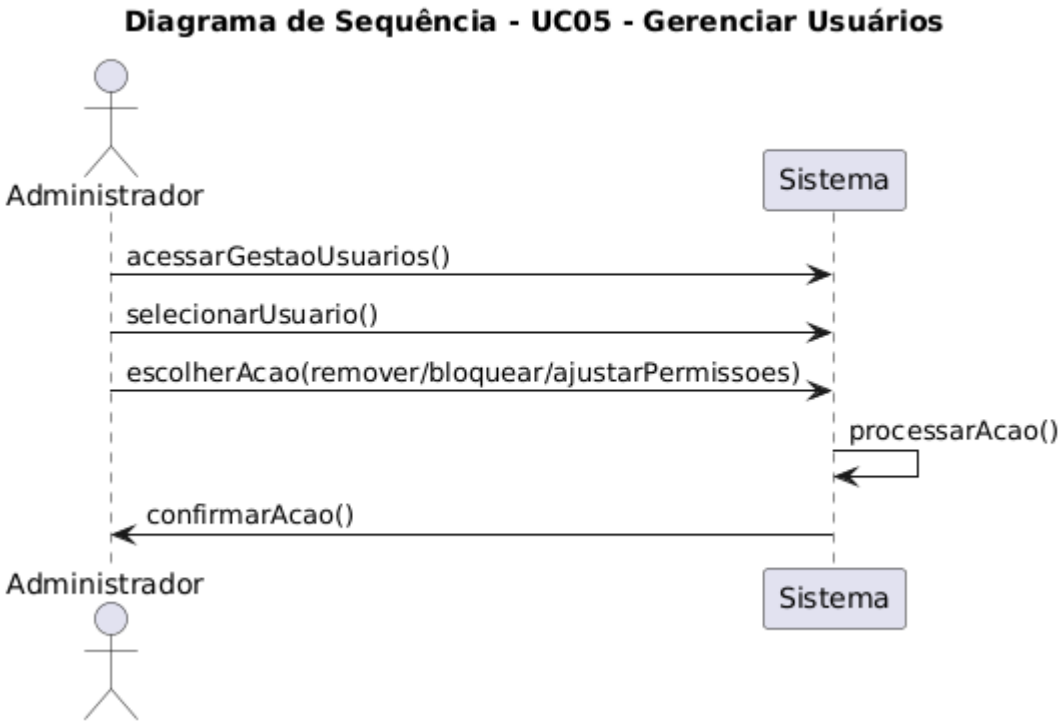


UC04 – Gerenciar Imóveis (A e B): funcionalidade central para locadores, possibilitando o cadastro, edição, desativação e exclusão de kitnets.

Diagrama de Sequência - UC04 - Gerenciar Imóveis (inclui cadastrar, editar, desativar, remover)

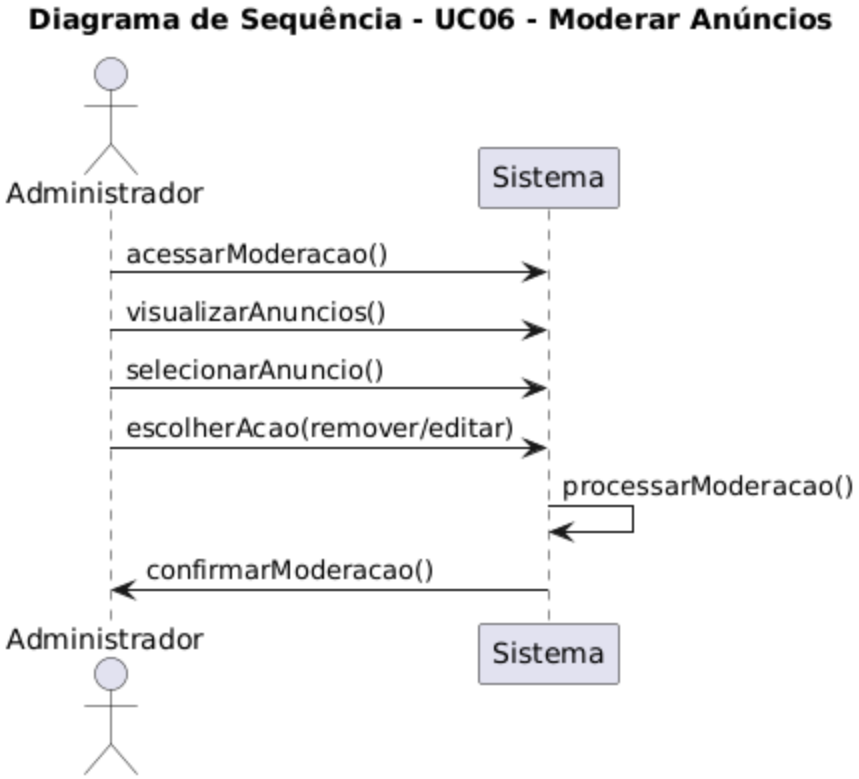


UC05 – Gerenciar Usuários: exclusivo do administrador, esse caso de uso permite realizar ações como remoção, bloqueio ou ajuste de permissões dos usuários.

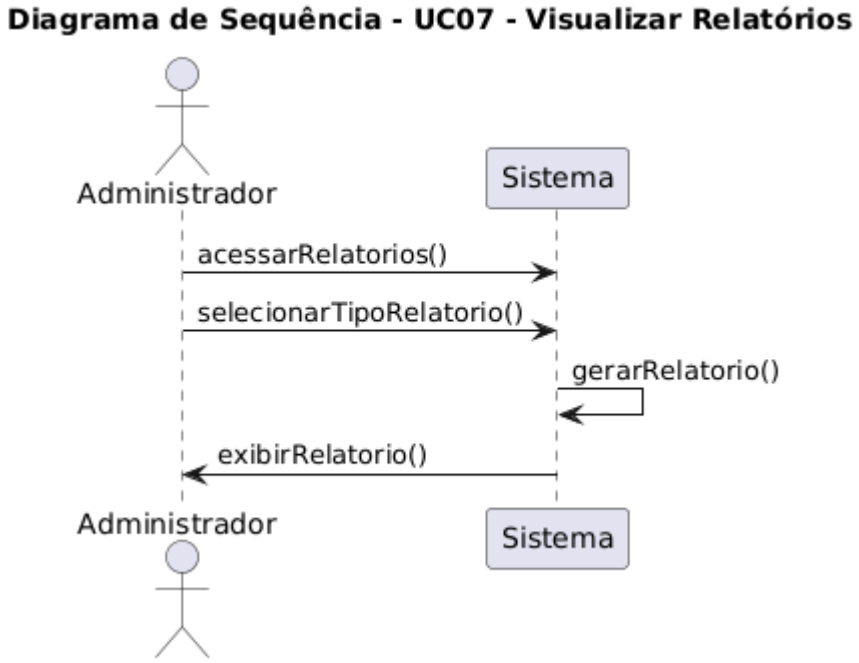


UC06 – Moderar Anúncios: possibilita ao administrador revisar e intervir em conteúdos impróprios ou fraudulentos publicados na plataforma

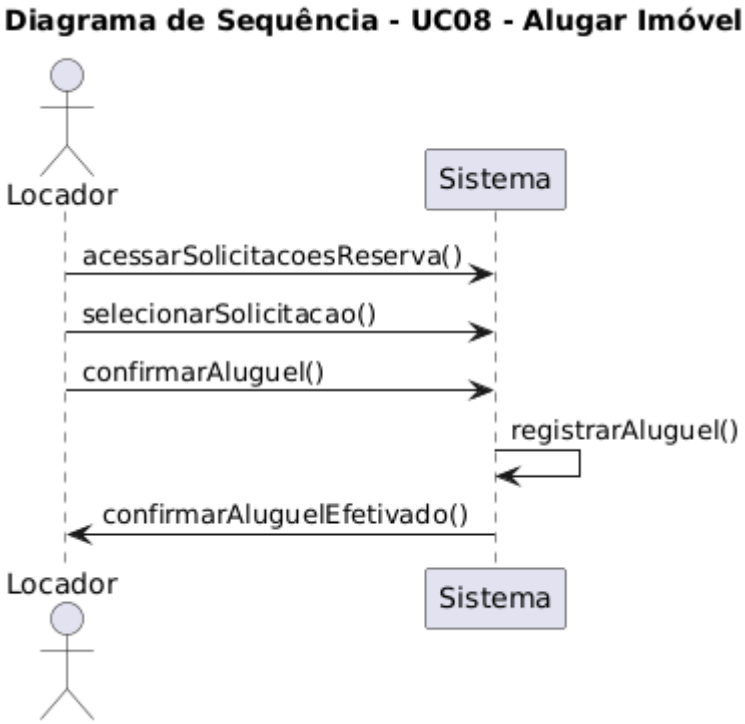
.



UC07 – Visualizar Relatórios: fornece ao administrador acesso a informações gerenciais como número de imóveis cadastrados, reservas efetuadas e acessos.

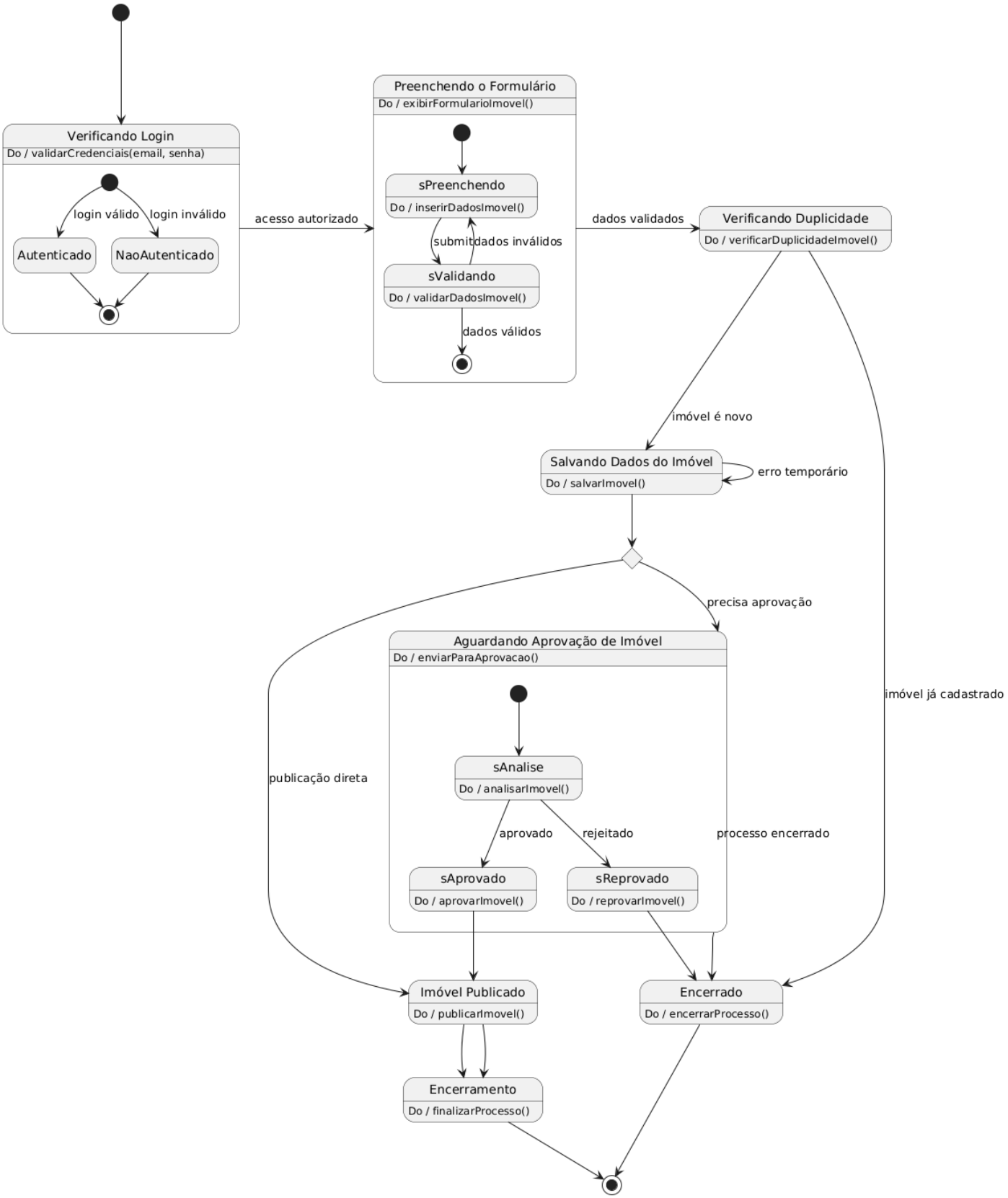


UC08 – Alugar Imóvel: permite que o locador finalize a locação de uma unidade habitacional após uma solicitação de reserva, caracterizando a efetivação do aluguel.



2.4 Diagrama de Máquina:

Caso de Uso: Cadastro de Imóvel



Explicação do Diagrama de Máquina – Cadastro de Imóvel

O diagrama de máquina apresentado descreve, de forma sequencial e visual, os principais estados e transições envolvidos no processo de cadastro de um imóvel na plataforma Kitnet, desde o login do usuário até a publicação ou encerramento do processo.

Principais etapas do fluxo:

Verificando Login: O processo se inicia com a validação das credenciais do usuário. Se o login for válido, o usuário é autenticado e pode prosseguir. Caso contrário, o processo é encerrado.

Preenchendo o Formulário: Após o acesso autorizado, o usuário preenche o formulário de cadastro do imóvel. Os dados inseridos são validados; se houver erros, o usuário é solicitado a corrigir e reenviar. Apenas dados válidos permitem o avanço.

Verificando Duplicidade: Com os dados validados, o sistema verifica se já existe um imóvel cadastrado com as mesmas características. Se o imóvel já existir, o processo é encerrado para evitar duplicidade.

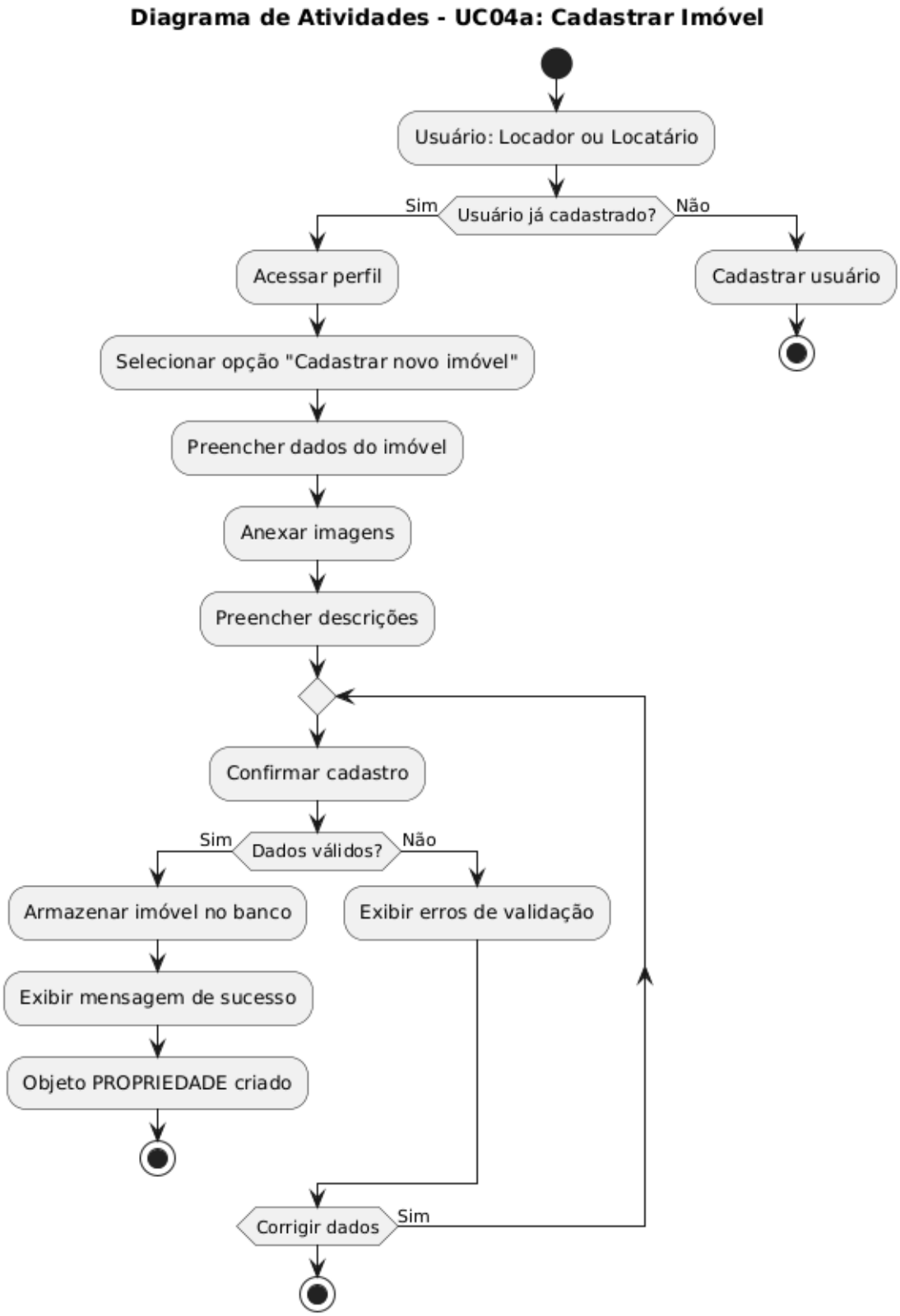
Salvando Dados do Imóvel: Caso o imóvel seja novo, o sistema tenta salvar os dados. Em caso de erro temporário, o sistema tenta novamente até obter sucesso.

Aguardando Aprovação de Imóvel: Dependendo das regras do sistema, o imóvel pode precisar passar por uma etapa de aprovação administrativa. Nessa etapa, um administrador analisa o cadastro e pode aprovar ou rejeitar o imóvel.

Imóvel Publicado ou Encerramento: Se aprovado, o imóvel é publicado e se torna visível na plataforma. Se rejeitado, o processo é encerrado e o proprietário é notificado. Em ambos os casos, o fluxo termina com o encerramento do processo.

2.5 Diagrama de Atividades:

Caso de Uso: Cadastrar imóvel



Explicação Diagrama de atividades: O diagrama de atividades do caso de uso "Cadastrar Imóvel" mostra o passo a passo para um usuário inserir um novo imóvel na plataforma. O fluxo começa verificando se o usuário já está cadastrado. Se sim, ele acessa o perfil, preenche os dados do imóvel, anexa imagens e confirma o cadastro. Caso haja erros de validação, o sistema exibe as mensagens e permite que o usuário corrija os dados até que estejam válidos. Quando tudo está correto, o imóvel é armazenado no sistema e uma mensagem de sucesso é exibida. Se o usuário não estiver cadastrado, ele deve se cadastrar antes de prosseguir.

3. Protótipos:

3.1 três protótipos de telas completos referentes a três casos de uso, usar Figma:

Link Figma: https://www.figma.com/design/DUklTEStoHAlByzUsN4pJJ/Untitled?node-id=0-1

Ao clicar no link, todas as telas e protótipos (presentes no código ou ainda não), solicitadas no exercício estarão no mesmo.
Telas: Caso de usos **UC01 – Autenticar, UC02 – Pesquisar Imóvel e UC04a – Cadastrar Imóvel.**

4. Código:

4.1 Desenvolver funcionalidades completas dos métodos apresentados no diagrama de classes. Codificar quantos conseguir (mínimo um método em cada classe; não será validado Gets e Sets nesse item).

Segue o link do projeto no github: https://github.com/EduardoMG12/kitnet

Obs: Todas as classes e atributos citados podem ser encontrados no diretório s**rc/main/java/com/kitnet/kitnet/model/`** do projeto, basta acessar esse caminho no projeto para visualizar os arquivos `User.java`, `Property.java` e `UserDocument.java`. Os atributos estão definidos diretamente nessas classes, e os **métodos de acesso são gerados automaticamente pelo Lombok, Não há métodos de negócio ou utilitários definidos manualmente na classe Property**. Todos os métodos são gerados automaticamente pelo Lombok, baseados nos atributos. Se você quiser métodos funcionais que fazem parte da regra de negocio estarao no service seguindo os devidos nomes "ProvertyServiceImpl" e se precisar filtrar os retornos ou argumentos das funcoes tem o DTO e o Mappe.

4.2 Implementar um padrão de projeto no seu sistema.

No desenvolvimento do backend do sistema Kitnet, foram aplicados padrões de projeto consagrados para garantir robustez, reutilização e facilidade de manutenção do código. Dentre os padrões utilizados, destacam-se o **Singleton** e o **Factory Method**, ambos amplamente empregados em aplicações baseadas no framework Spring Boot.

Padrão Singleton:

O padrão Singleton tem como objetivo garantir que uma classe possua apenas uma instância durante todo o ciclo de vida da aplicação, fornecendo um ponto global de acesso a essa instância. No contexto do Spring Boot, este padrão é implementado de forma nativa pelo container de injeção de dependências do framework.

Aplicação no Projeto:
No nosso sistema, todas as classes anotadas com `@Service`, `@Repository`, `@Component` ou `@Configuration` são, por padrão, gerenciadas como Singletons pelo Spring. Isso significa que, para cada uma dessas classes, existe apenas uma instância compartilhada entre todos os componentes que dela dependem.
Exemplo:
Local: `src/main/java/com/kitnet/kitnet/service/UserService.java`

```
@Service
public class UserService {
    // O Spring garante que só existe uma instância desta classe durante a execução da aplicação
}
```

Local: `src/main/java/com/kitnet/kitnet/repository/PropertyRepository.java`

```
@Repository
public interface PropertyRepository extends JpaRepository<Property, UUID> {
    // Também é gerenciado como Singleton pelo Spring
}
```

Benefícios:
- Economia de recursos: evita a criação de múltiplas instâncias desnecessárias.
- Consistência de estado: todos os componentes compartilham a mesma instância, facilitando o controle de dados e operações.

Padrão Factory Method

O padrão Factory Method é utilizado para delegar a criação de objetos a métodos especializados, promovendo o desacoplamento entre a lógica de criação e o uso dos objetos. No Spring Boot, este padrão é utilizado na configuração de beans, especialmente em classes anotadas com `@Configuration`.

Aplicação no Projeto:
No nosso sistema, métodos anotados com `@Bean` dentro de classes de configuração funcionam como fábricas de objetos, permitindo a customização e o gerenciamento centralizado das instâncias criadas.

Exemplo:
Arquivo: `src/main/java/com/kitnet/kitnet/config/FirebaseAdminConfig.java`

```
@Configuration
public class FirebaseAdminConfig {
    @Bean
    public FirebaseApp firebaseApp() {
        // Criação e configuração do objeto FirebaseApp
        return FirebaseApp.initializeApp(options);
    }
}
```

Neste exemplo, o método `firebaseApp()` é um Factory Method responsável por criar e configurar a instância do serviço Firebase, que será injetada onde necessário.

Benefícios:
- Desacoplamento: separa a lógica de criação do uso dos objetos.
- Facilidade de manutenção: centraliza a configuração e facilita alterações futuras.

Conclusão:
A utilização dos padrões Singleton e Factory Method no backend do nosso projeto Kitnet, por meio das práticas recomendadas do Spring Boot.

Referência:
[Refactoring Guru – Design Patterns](https://refactoring.guru/design-patterns)