




Documentação de Software

Projeto Integrador

SENIOR CARE



Colaboradores:

TARCIO TELES

DAVI SOUZA

LUCAS RAMOS

PEDRO LACERDA

ROBERTH RHUAN



Documentação de Software Senior Care

1 Ferramentas, Linguagens e Softwares que Serão Utilizados

A escolha das ferramentas, linguagens e softwares para o desenvolvimento do sistema Senior Care foi baseada em critérios como escalabilidade, facilidade de manutenção, compatibilidade com tecnologias modernas, e suporte a requisitos específicos do projeto. Abaixo estão detalhadas as decisões tomadas para cada componente do sistema.

a. Linguagem de Back-End:

- **Linguagem:** Java
- **Justificativa:** Java foi escolhido devido à sua robustez e escalabilidade, além de ser amplamente utilizado em sistemas de grande porte e aplicações empresariais. Ele oferece um vasto ecossistema de bibliotecas e frameworks que suportam o desenvolvimento de sistemas complexos e de alta disponibilidade. A linguagem também é conhecida por sua portabilidade e desempenho confiável.
- **Referência:** "Java tem sido uma escolha popular para back-end em grandes empresas devido à sua estabilidade, escalabilidade e vasta comunidade de desenvolvedores que garantem suporte contínuo." (Fonte: Why Java is Still a Great Choice for Enterprise Applications)

b. Framework de Back-End:

- **Framework:** Spring Boot
- **Justificativa:** Spring Boot é um framework poderoso e flexível para o desenvolvimento de aplicações Java. Ele simplifica a configuração e o desenvolvimento, permitindo a construção rápida de aplicações seguras e escaláveis. O Spring Boot inclui suporte integrado para APIs RESTful, segurança robusta e uma forte integração com sistemas de banco de dados, o que o torna ideal para nosso projeto.
- **Referência:** "O Spring Boot é escolhido por sua capacidade de simplificar a criação de aplicações Java robustas e escaláveis, com uma configuração mínima e uma vasta gama de funcionalidades integradas." (Fonte: [Spring Boot in Microservices](#))

c. Framework de Front-End:

- **Framework:** JavaScript (com Bootstrap)
- **Justificativa:** JavaScript é a linguagem de escolha para o desenvolvimento de front-end devido à sua ubiquidade e flexibilidade. Com o uso do Bootstrap, podemos garantir uma interface de usuário responsiva e moderna, facilitando o desenvolvimento rápido de componentes visuais consistentes. O uso de JavaScript permite a criação de interfaces dinâmicas e interativas, essenciais para uma boa experiência do usuário.
- **Referência:** "JavaScript, em conjunto com Bootstrap, é amplamente utilizado para construir interfaces de usuário responsivas e interativas de forma eficiente, devido à sua popularidade e vasto suporte da comunidade." (Fonte: Why Use JavaScript for Front-End Development)

d. Sistema Gerenciador de Banco de Dados:

- **Banco de Dados:** MySQL

Senior Care

- **Justificativa:** MySQL é um banco de dados relacional amplamente utilizado e reconhecido pela sua robustez e desempenho eficiente em diversas aplicações. Ele é ideal para nosso projeto devido à sua facilidade de uso, suporte avançado para transações, e escalabilidade. A popularidade de MySQL garante uma ampla comunidade e suporte contínuo, além de ferramentas robustas para gerenciamento de dados.
- **Referência:** "MySQL é escolhido por sua eficiência e confiabilidade, sendo uma opção preferida para aplicações web que requerem uma gestão eficaz de grandes volumes de dados." (Fonte: [The Advantages of Using MySQL](#))

e. Versionador de Código:

- **Ferramenta:** GitHub
- **Justificativa:** GitHub continua sendo uma escolha popular para controle de versão devido à sua facilidade de uso e integração com várias ferramentas de desenvolvimento. Ele permite uma colaboração eficaz entre equipes, gerenciamento de branches, e revisão de código, promovendo práticas de desenvolvimento modernas e colaborativas.
- **Referência:** "GitHub é amplamente adotado por sua facilidade de uso e suporte para práticas de desenvolvimento colaborativas, essencial para a gestão eficaz de projetos de software." (Fonte: Benefits of Using GitHub)

f. Ferramentas de Diagramação UML:

- **Ferramenta: PlantUML**
 - **Justificativa:** PlantUML é uma ferramenta de código aberto que permite a criação de diagramas UML a partir de uma linguagem de marcação simples. É especialmente útil para desenvolvedores, pois permite a criação de diagramas diretamente no código, facilitando a documentação contínua e a integração com sistemas de controle de versão. PlantUML é ideal para projetos que requerem uma geração rápida e automatizada de diagramas.
 - **Referência:** "PlantUML facilita a criação e manutenção de diagramas UML diretamente no fluxo de desenvolvimento, promovendo uma documentação consistente e integrada ao processo de codificação." (Fonte: [Why PlantUML Is the Ideal Tool for Developers](#))
- **Ferramenta: Lucidchart**
 - **Justificativa:** Lucidchart é uma ferramenta baseada na web que oferece uma interface intuitiva para a criação de diagramas UML e outros tipos de diagramas. É ideal para colaboração em equipe, permitindo que vários usuários trabalhem simultaneamente em um mesmo diagrama. Lucidchart é uma excelente escolha para equipes que necessitam de uma solução visual abrangente e fácil de usar para a documentação e planejamento de sistemas.
 - **Referência:** "Lucidchart é uma ferramenta poderosa para a criação de diagramas colaborativos, suportando uma vasta gama de diagramas e facilitando a comunicação visual entre equipes." (Fonte: Collaborative Diagramming with Lucidchart)

g. Outras Ferramentas:

- **VSCode:** Editor de código leve e extensível, escolhido pela sua popularidade e vasta gama de extensões.
-

- **Figma:** Ferramenta de design para criação de protótipos e interfaces de usuário, escolhida pela sua capacidade de colaboração em tempo real e facilidade de uso.

2.2 Resolução dos Desafios Técnicos

Desafio Técnico: Implementação de Algoritmos de Recomendação Personalizados

Descrição do Desafio: O Senior Care precisa oferecer recomendações personalizadas para os horários de medicação e consultas dos usuários, com base em seus comportamentos e preferências. O objetivo é melhorar a aderência ao tratamento e a experiência do usuário, fornecendo notificações oportunas e relevantes.

Estratégia de Resolução:

1. Coleta de Dados: O primeiro passo será a coleta de dados de uso, como horários preferidos para notificações, histórico de tomadas de medicação, frequência de consultas, entre outros. Isso será realizado por meio do banco de dados PostgreSQL, que armazenará todas as interações dos usuários com o sistema.

2. Análise de Dados: Utilizando ferramentas como Pandas e NumPy, os dados coletados serão analisados para identificar padrões de comportamento. Isso ajudará a entender melhor as necessidades e preferências dos usuários.

3. Desenvolvimento dos Algoritmos:

- **Content-Based Filtering:** Serão utilizados algoritmos que analisam as características dos medicamentos e consultas, recomendando horários com base em preferências anteriores dos usuários.
- **Collaborative Filtering:** Algoritmos que analisam o comportamento de usuários semelhantes para fornecer recomendações baseadas em padrões de grupo.
- **Hybrid Systems:** Combinação de técnicas de filtragem baseadas em conteúdo e colaborativas para aumentar a precisão das recomendações.

4. Testes e Validação: Os algoritmos serão testados em um ambiente de desenvolvimento para validar sua eficácia e precisão. Serão realizadas simulações com dados de teste para ajustar os modelos e garantir que as recomendações sejam úteis e relevantes.

Ferramentas Utilizadas:

- **Python:** Para o desenvolvimento dos algoritmos.
- **Jupyter Notebook:** Para a análise de dados e desenvolvimento iterativo dos modelos.
- **Scikit-learn e TensorFlow:** Bibliotecas para a implementação de algoritmos de machine learning.

Fontes de Consulta:

Senior Care

- **Livro:** "Machine Learning Yearning" por Andrew Ng.
- **Artigo:** "A Comprehensive Guide to Machine Learning" por Sebastian Raschka.
- **Curso:** "Machine Learning" da Coursera por Andrew Ng.

5. Diagramas e Desenhos Arquiteturais

5.1 Diagrama de Caso de Uso:

Descrição:

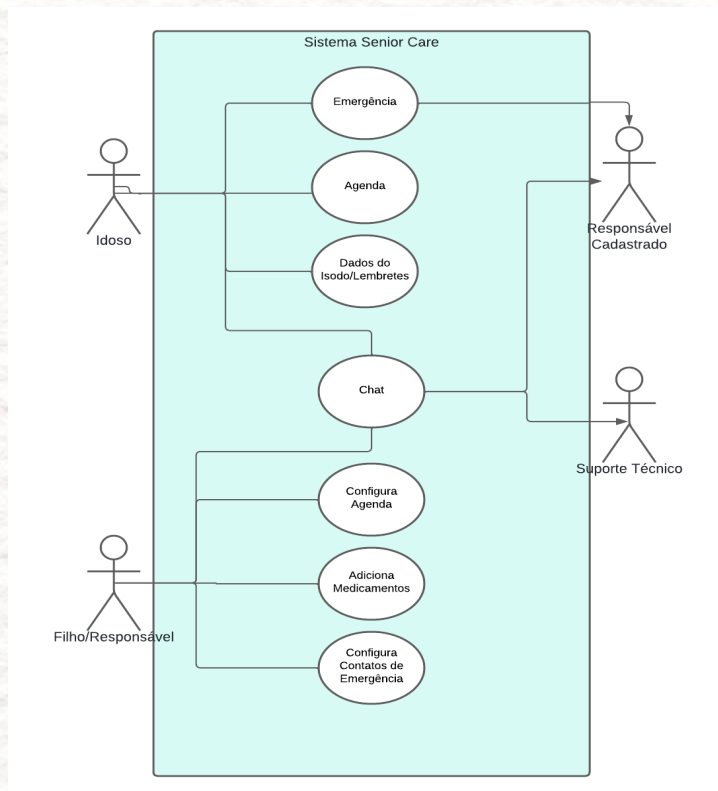
Ator Principal:

- **Idoso:** Usuário do sistema que recebe notificações sobre medicamentos e pode acessar sua agenda e dados pessoais.

Ator Secundário:

- **Responsável (Filho/Responsável):** Pessoa cadastrada como responsável pelo idoso, que administra a agenda, medicamentos e consultas do idoso.

Descrição Geral: Este caso de uso descreve como um idoso interage com o sistema Senior Care para gerenciar notificações de medicamentos, acessar sua agenda e dados pessoais, e utilizar a funcionalidade de botão de emergência. O responsável cadastrado no sistema tem a capacidade de administrar a agenda, os medicamentos e os dados de consulta do idoso, garantindo uma supervisão eficaz e suporte contínuo.

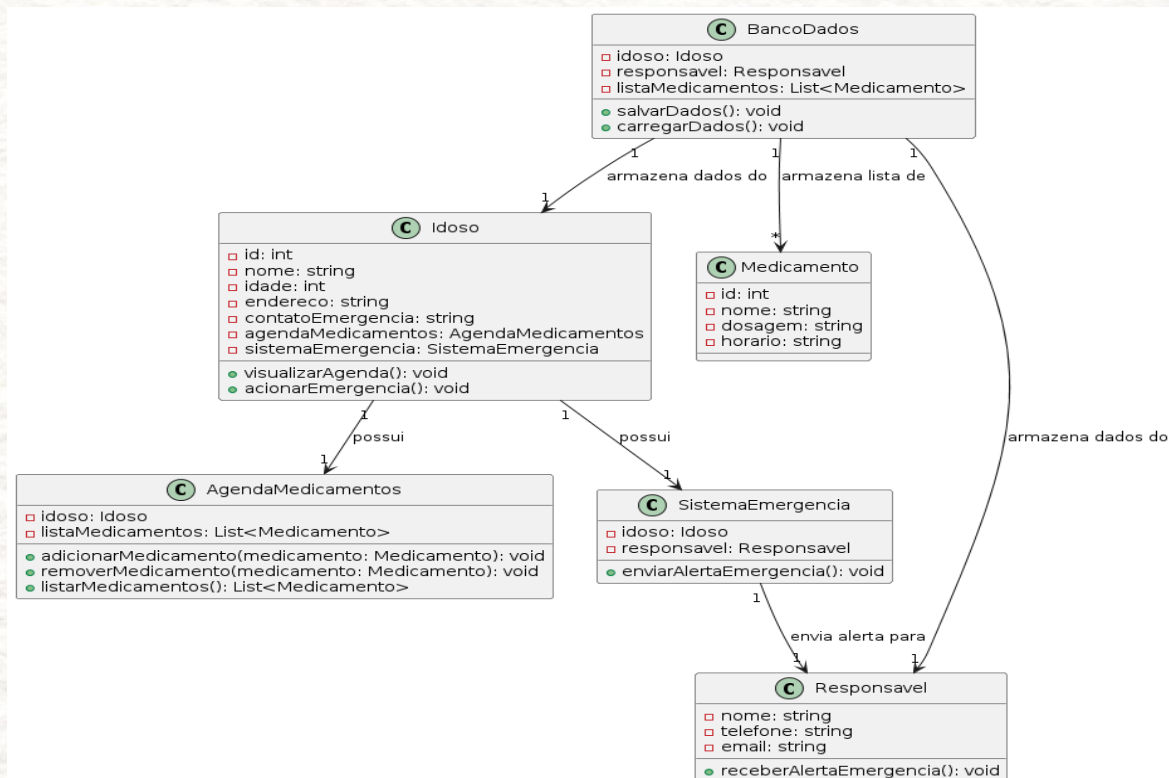


Senior Care

.5.2 Diagrama de Classe

O diagrama de classes do sistema "Senior Care" modela as principais entidades e suas interações, proporcionando uma visão estruturada e organizada do funcionamento do sistema projetado para cuidados de idosos. Abaixo estão as principais classes e suas responsabilidades:

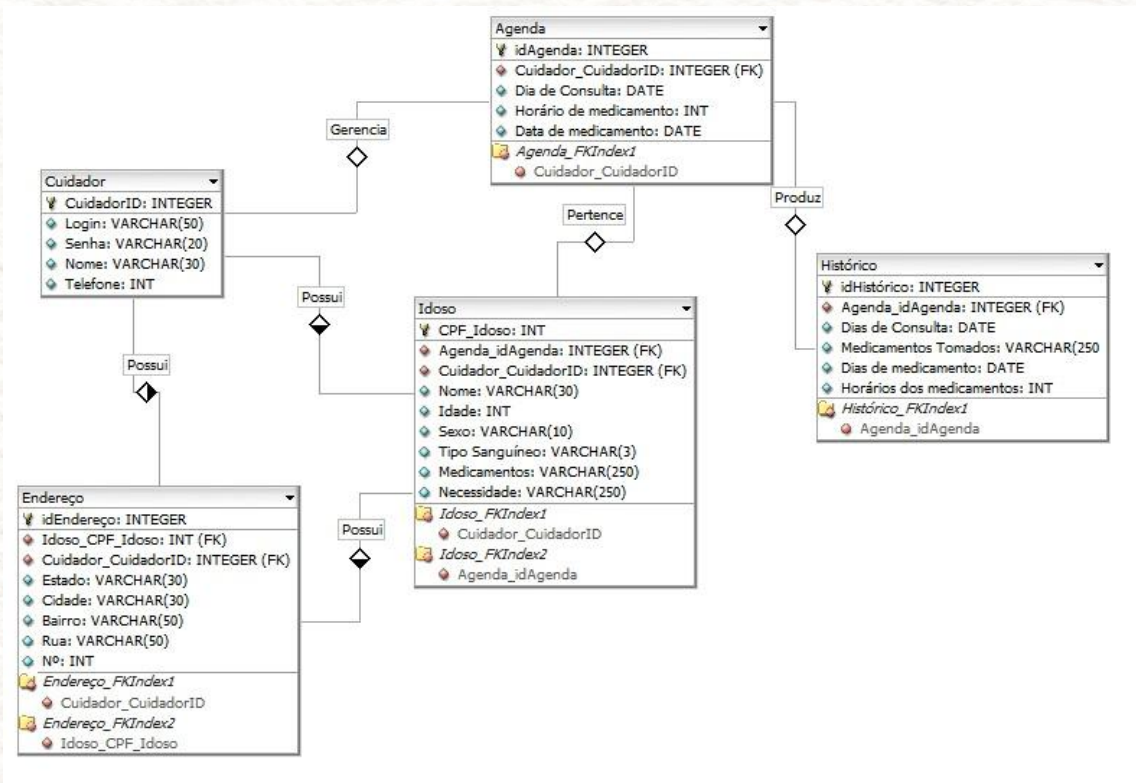
- **Idoso:** Representa os dados essenciais do usuário idoso no sistema, incluindo nome, idade, endereço e informações de contato de emergência. O idoso possui uma agenda de medicamentos e um sistema de emergência integrado.
- **Agenda Medicamentos:** Gerencia a lista de medicamentos prescritos para o idoso, permitindo adicionar, remover e listar os medicamentos associados.
- **Medicamento:** Representa cada prescrição médica, contendo atributos como nome, dosagem, horário de administração e um identificador único (id). Os medicamentos são armazenados no banco de dados junto com outras informações do sistema.
- **Sistema Emergência:** Responsável por enviar alertas de emergência para o responsável cadastrado quando acionado pelo idoso.
- **Responsável:** Representa o responsável pelo idoso, contendo informações como nome, telefone e e-mail. Recebe alertas de emergência do sistema de emergência.
- **Banco Dados:** Classe responsável pela interação com o banco de dados, armazenando dados do idoso, do responsável e a lista de medicamentos prescritos. Gerencia o armazenamento e recuperação dos dados do sistema "Senior Care".



5.3 Diagrama de Entidade-Relacionamento (DER) para o Sistema Senior Care

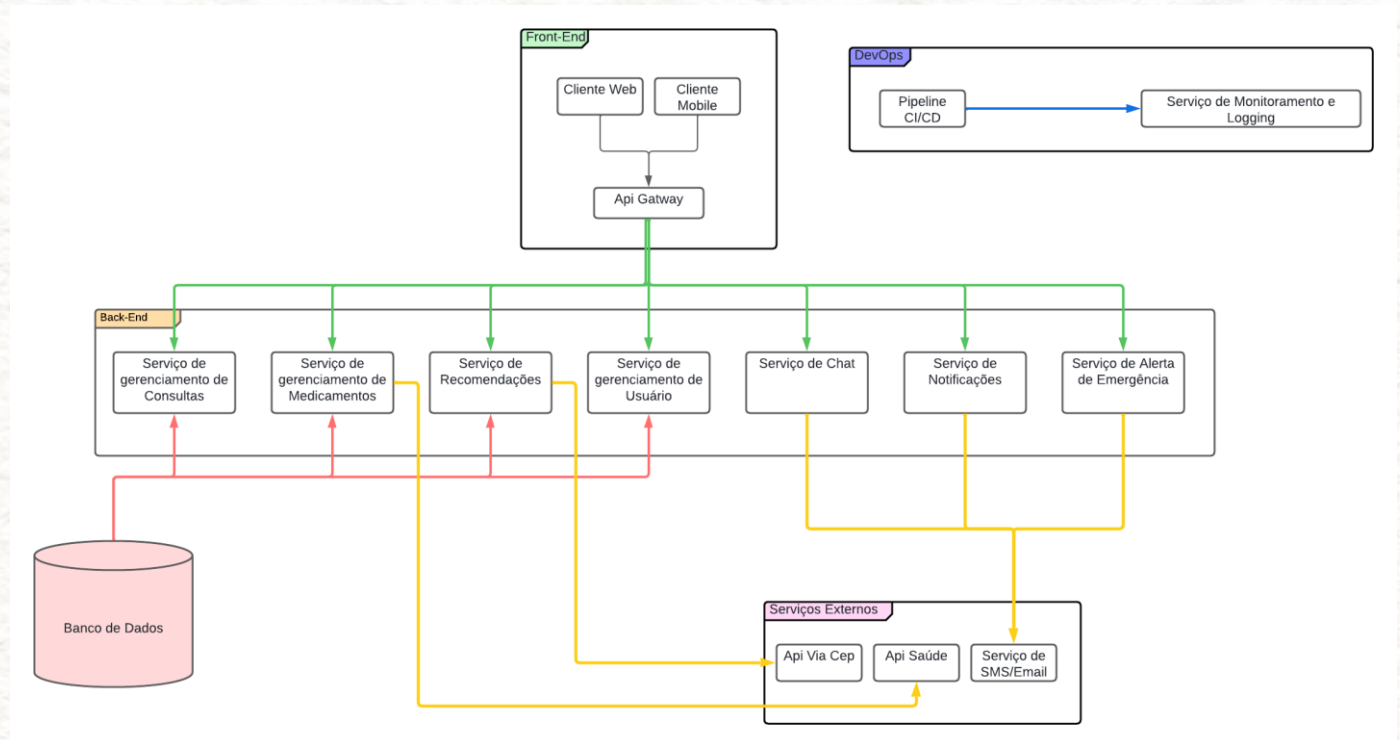
O Diagrama de Entidade-Relacionamento (DER) para o sistema "Senior Care" modela as entidades principais e seus relacionamentos, fornecendo uma estrutura clara para o armazenamento e gerenciamento dos dados essenciais do sistema. Abaixo estão os principais componentes do DER:

| Entidade | Relacionamento | Atributo Chave |
|-----------|-----------------------------------------|-------------------------|
| Cuidador | Um Cuidador possui um ou mais Idosos | Cuidador = CuidadorID |
| Idoso | Um Cuidador possui um ou mais Endereços | Endereço = idEndereço |
| Endereço | Um Cuidador gerencia uma Agenda | Idoso = CPF_Idoso |
| Agenda | Um Idoso possui um ou mais Endereços | Agenda = idAgenda |
| Histórico | Uma Agenda pertence a um idoso | Histórico = idHistórico |
| | Uma agenda produz um Histórico | |



Senior Care

5.5 Desenho de Arquitetura de Software



6. Projeto

Link do GitHub

Confira o repositório do nosso projeto no GitHub para acessar o código-fonte completo, documentação detalhada e colaborar conosco: [GitHub](#).

Link do Site

Visite nosso site para conhecer mais sobre o projeto, explorar suas funcionalidades e ficar por dentro das últimas atualizações: [Site do Projeto](#).

7. Controle de Versão

O controle de versão deste documento é essencial para garantir a rastreabilidade e a integridade das informações ao longo de seu ciclo de vida. A tabela de controle de versão detalha cada atualização feita no documento, incluindo a data da alteração, o autor responsável e uma breve descrição das mudanças implementadas.

Cada versão do documento reflete uma etapa de desenvolvimento e revisão, permitindo que todas as modificações sejam registradas e consultadas facilmente. Isso é fundamental para manter a consistência das informações, evitar a perda de dados e assegurar que todos os envolvidos no projeto tenham acesso à versão mais recente e precisa do documento.

Recomenda-se que todas as atualizações sejam documentadas de forma clara e concisa, destacando os aspectos principais de cada mudança. Essa prática contribui para a transparência do processo de revisão e facilita a colaboração entre diferentes membros da equipe, garantindo que o documento esteja sempre atualizado e alinhado com os objetivos do projeto."

7.1 Tabela de Controle de Versão

[illegible]