

**SMART
LEAF**

SOBRE O PROJETO

Desenvolver um software de padronização e interpretação de prontuários médicos entre diferentes especialidades

- Criação e armazenamento de prontuários médicos digitalizados
- Interpretação automatizada de dados clínicos via API de IA (StackpotIA)
- Padronização de registros médicos
- Geração de relatórios e exportação em PDF
- Interface intuitiva e segura, desenvolvida com Java Spring Boot, MongoDB e HTML/CSS/JS

ODS



PROBLEMÁTICA



- Prontuários médicos sem padronização, com registros inconsistentes entre profissionais e especialidades.
- Dificuldade na interpretação e análise dos dados clínicos, gerando retrabalho e risco de erro.
- Tempo excessivo gasto em documentação manual, reduzindo a produtividade dos profissionais de saúde.
- Ausência de ferramentas inteligentes para transformar dados clínicos em informações úteis para médicos.

Lean Canvas



CANVAS

REQUISITOS FUNCIONAIS

Anotações SOAP

Exportação Prontuários

Login Seguro

Gestão de Médicos

Especialidades IA

Histórico de Versões

Edição de Prontuários

Comparação Diff

Colaboração Médica

Integração FHIR HL7

App Mobile Offline

Relatórios de Estatísticas

Recuperação de Senha

REQUISITOS NÃO FUNCIONAIS

Desempenho de 5s

Disponibilidade de 99%

Autenticação Segura

Escalabilidade Container

ArquiteturaMVC

Interface Responsiva

Backup Restore

SuporteMultiplataforma

Conformidade LGPD

DIAGRAMA DE CASO DE USO

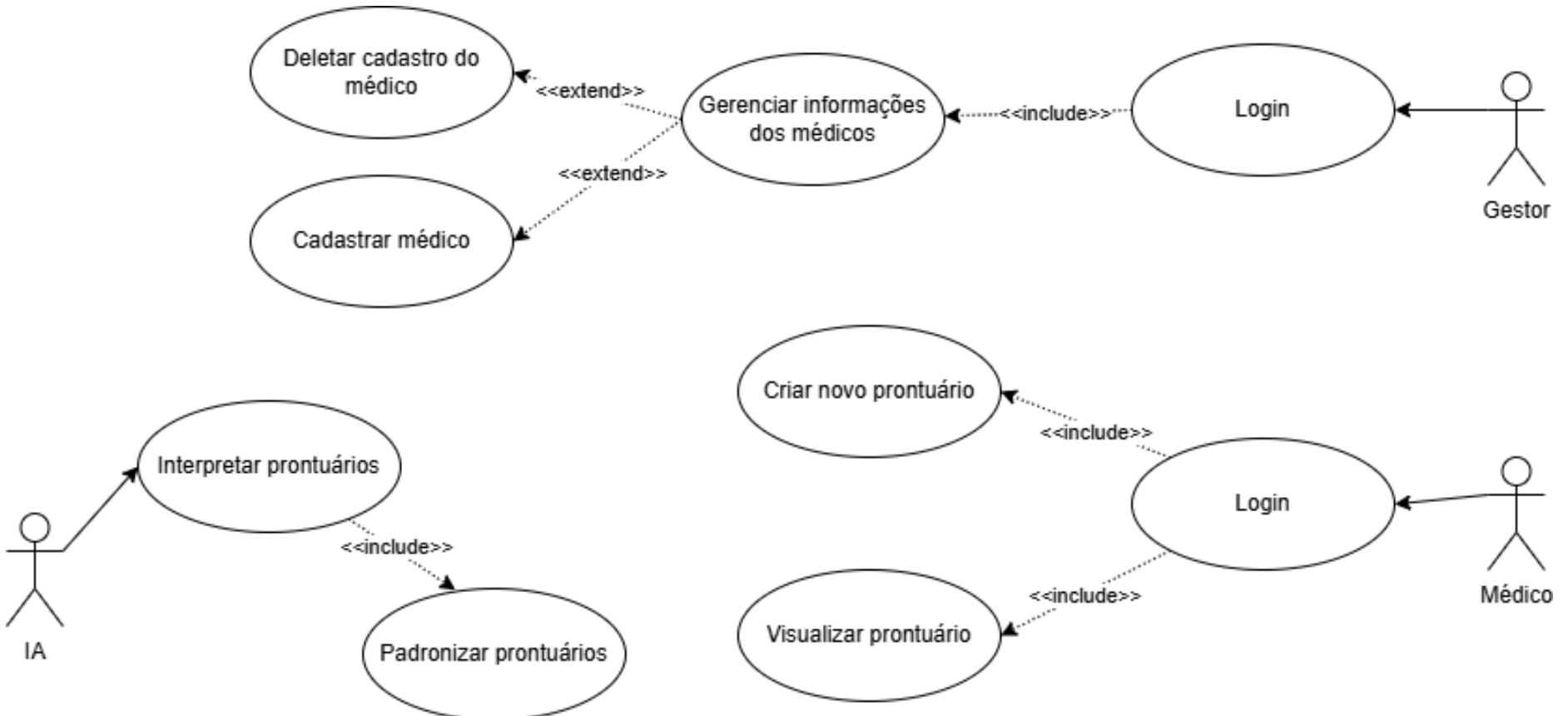
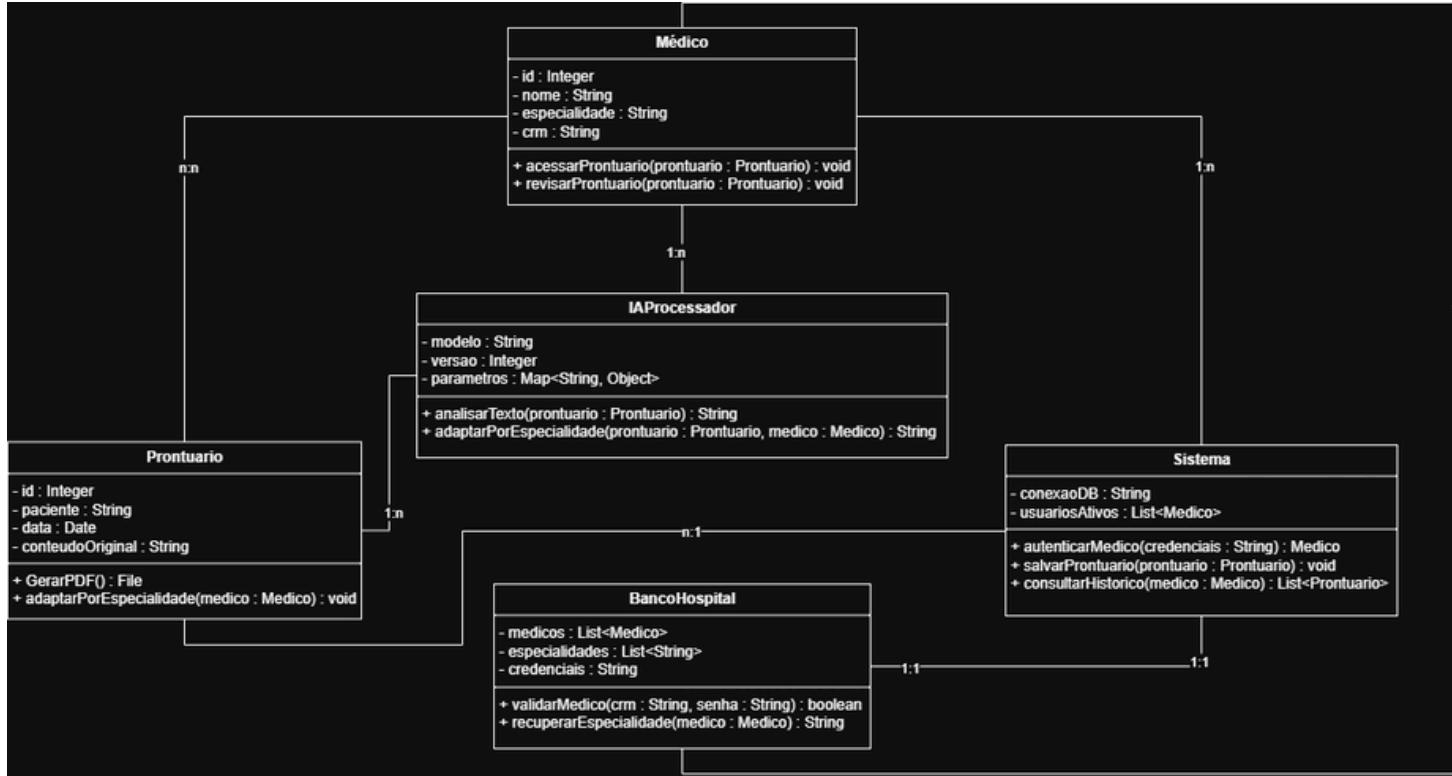
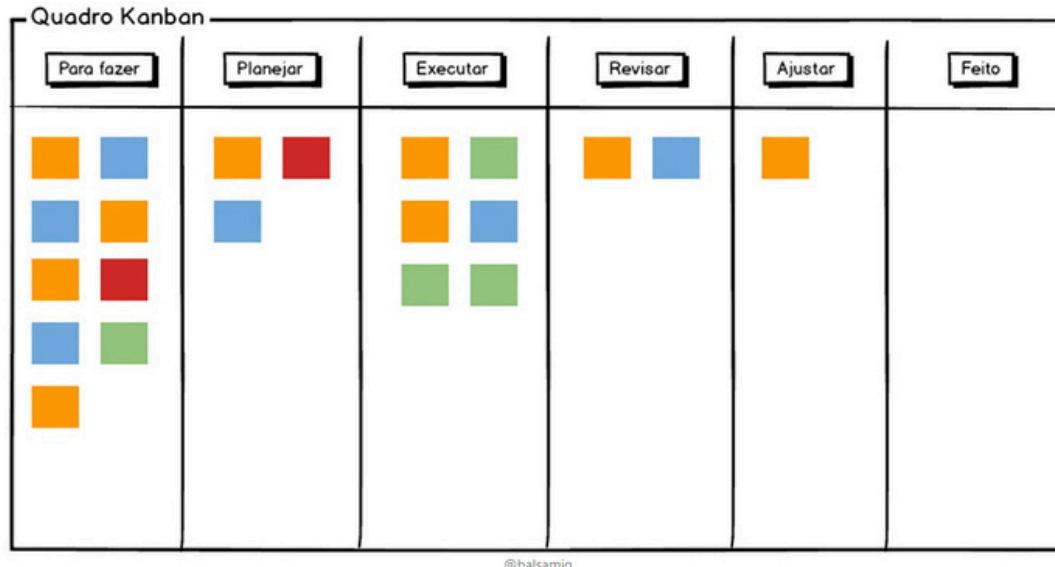


DIAGRAMA DE CLASSE



MÉTODO DE DESENVOLVIMENTO

Kanban



Scrum



PLATAFORMA TRELLO

Projeto_Integrador 000

EA MC BC FF GG +1 ⌂ Compartilhar ...

Sprint 1 - 18/08 - 08/09

- ✓ Reunião Inicial
- ✓ SMART
- ✓ EAP
- ✓ Canvas
- ✓ BPMN
- ✓ Caso de Uso
- ✓ Diagrama de Classe
- ✓ Treino da IA Generativa - Fase 1: Criação
- ✓ Documentação Parte 1
- 
- + Adicionar um cartão

Sprint 2 - 08/09 - 29/09

- ✓ Reunião Inicial
- ✓ WireFrame
- ✓ Desenvolvimento do Leitor de PDF/Arquivo
- ✓ Alterações no Design do Projeto
- ✓ Protótipo - Telas
- ✓ Desenvolvimento do FrontEnd
- ✓ Conexão com o Banco de Dados (MongoDB)
- ✓ Treino da IA Generativa - Fase 2: Implementação
- ✓ Treino da IA Generativa - Fase 2: Machine Learning
- ✓ EAP Atualizado
- ✓ Documentação Parte 2
- + Adicionar um cartão

Sprint 3 - 06/10 - 27/10

- ✓ Reunião Inicial
- Alterações no Design do Projeto - Figma e Front End
- ✓ WireFrame
- Documentação Parte 1 - Final
- Documentação Parte 2 - Final
- 
- Apresentação 13/10
- ✓ Reunião Final
- + Adicionar um cartão

Sprint 4 - 03/11 - 01/12

- Reunião Inicial
- Artigo Científico - Entrega
- Correções de falhas
- Apresentação Final
- Versão final do projeto
- + Adicionar um cartão

+ Adicionar outra lista

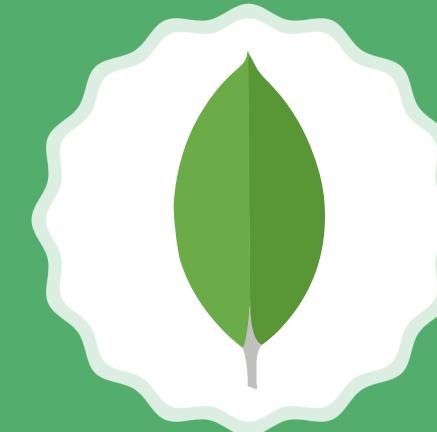
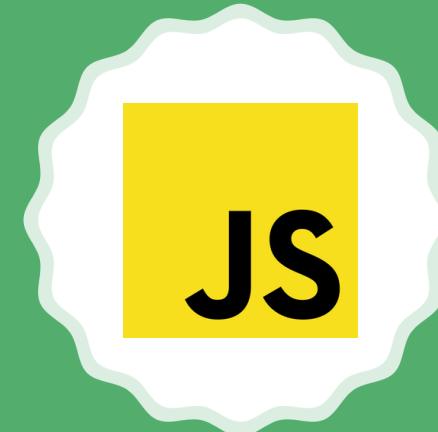
+ Adicionar um cartão

Caixa de entrada Planejador Quadro Mudar de quadros

Jira



TECNOLOGIAS UTILIZADAS



The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with various icons: a file icon, a search icon, a magnifying glass, a gear, a play/pause icon, a file icon, a person icon, a question mark, a gear, and a settings gear.

The main area displays a Python script named `saudade_mais.py`. The code is as follows:

```
saude_mais.py X
src > main > resources > ia-integration > saude_mais.py
18     REALM = "stackspot-freemium"
19
20     CLIENT_ID = "f56b6b65-6488-4401-9a94-c4c211654497"
21     CLIENT_KEY = "y5H1vve2jo2ci23HdzhafYpYni04143BBY12MemAER5LqR5rdfy57ZA5g76CF93o"
22
23     AGENT_URL = "https://genai-inference-app.stackspot.com/v1/agent/01K3GT29GMFY5HMCEM7F8HSNKN/chat"
24
25     def get_jwt():
26
27         url = f"https://idm.stackspot.com/{REALM}/oidc/oauth/token"
28
29         payload = {
30             "grant_type": "client_credentials",
31             "client_id": CLIENT_ID,
32             "client_secret": CLIENT_KEY
33         }
34
35         headers = {"Content-Type": "application/x-www-form-urlencoded"}
36
37         response = requests.post(url, data=payload, headers=headers)
38
39         if response.status_code != 200:
40
41             raise HTTPException(status_code=401, detail="Erro na autenticação")
42
43         return response.json().get("access_token")
44
45
46     @app.post("/chat")
47
48     async def chat(request: Request):
49
50         import datetime
51
52         body raw = await request.body()
```

The status bar at the bottom indicates the file is Java Ready, with tabs for main, Java, and Python, and status for Ln 1, Col 1, Tamanho da Tela Tab: 4, UTF-8, CRLF, Go Live, and a refresh icon.

App controller (model)

The screenshot shows a Java code editor with the file `appController.java` open. The code defines a controller class with three methods: `home()`, `login()`, and `gestor()`. The `home()` and `login()` methods map to the root URL and the login URL respectively. The `gestor()` method maps to the URL `/gestor`. The code uses annotations `@Controller` and `@RequestMapping` from the `org.springframework.web.bind.annotation` package.

```
src > main > java > com > br > iasaude > saudemais > Controller > appController.java > gestor()
```

```
8  @Controller
9
10 public class appController {
11
12     // Rota para a página inicial
13     @GetMapping("/")
14     public String home() {
15
16         return "index";
17
18     // Rota para a página de login
19     @GetMapping("/login")
20     public String login() {
21
22         return "login";
23
24     // Rota para a página do gestor
25     @GetMapping("/gestor")
26     public String gestor() {
27
28         return "gestor";
29     }
30 }
```

Medicorest controller

The screenshot shows a Java code editor with a dark theme. On the left, there's a vertical toolbar with icons for file operations, search, and navigation. The main area displays a Java file named `MedicoRestController.java`. The code implements a REST controller for a medical application:

```
1 package com.br.iasaude.saudemais.Controller;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RestController;
6 import org.springframework.security.access.prepost.PreAuthorize;
7
8 @RestController
9 @RequestMapping("/api")
10 public class MedicoRestController {
11     @GetMapping("/medico")
12     public String medico() {
13         return "Acesso autorizado ao endpoint REST /api/medico";
14     }
15 }
16
```

The code includes imports for Spring annotations and security, defines a `@RestController`, and maps the `/medico` endpoint to the `medico()` method, which returns a success message.

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** MedicoRestController.java 1 < PdfController.java < ...
- Project Path:** src > main > java > com > br > iasaude > saudemais > Controller > PdfController.java > {} com.br.iasaude.saudemais.Controller
- Code Editor:** The PdfController.java file is open. The code handles PDF interpretation and sends requests to an AI API.
- Code Snippet:**

```
24  public class PdfController {  
30      public ResponseEntity<Map<String, Object>> interpretarPdf(@RequestParam("file") MultipartFile file) throws IOException {  
46          // Se não houver texto extraído, retorna erro  
47          if (text == null || text.trim().isEmpty()) {  
48              // (Removido bloco duplicado e corrigido)  
49              logger.warn("Texto extraído do PDF está vazio ou nulo. Não será enviada requisição para IA.");  
50              result.put(key: "erroIA", value: "Texto extraído do PDF está vazio ou nulo.");  
51              return ResponseEntity.ok(result);  
52          }  
53          // Chamada para a API da IA (FastAPI) usando RestTemplate  
54          String iaResponse = null;  
55          try {  
56              ObjectMapper mapper = new ObjectMapper();  
57              Map<String, String> iaPayload = new HashMap<>();  
58              iaPayload.put(key: "user_prompt", text);  
59              String requestBody = mapper.writeValueAsString(iaPayload);  
60              logger.info("Enviando para IA: {}", requestBody);  
61  
62              HttpHeaders headers = new HttpHeaders();  
63              headers.setContentType(MediaType.APPLICATION_JSON);  
64              HttpEntity<String> entity = new HttpEntity<>(requestBody, headers);
```
- Toolbars and Status Bar:** Includes standard IDE icons (File, Edit, View, etc.) and status information like "Java Ready", "edutbx (1 semana atrás)", "Ln 2, Col 35", "Espaços: 4", "UTF-8", "CRLF", "Java", "Go Live", and "Ctrl + F5".

Profile controller

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** The title bar displays "Profile controller".
- File List:** The left sidebar shows the file structure: "src > main > java > com > br > iasaude > saudemais > Controller > ProfileController.java".
- Code Editor:** The main area contains the code for `ProfileController.java`. The code defines a REST controller for retrieving profiles.

```
MedicoRestController.java 1 ProfileController.java X
src > main > java > com > br > iasaude > saudemais > Controller > ProfileController.java > {} com.br.iasaude.saudemais.Controller
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RestController;
6 import java.util.Arrays;
7 import java.util.Map;
8
9 @RestController
10 public class ProfileController {
11     private final Environment env;
12
13     public ProfileController(Environment env) {
14         this.env = env;
15     }
16
17     @GetMapping("/api/profile")
18     public Map<String, Object> getProfile() {
19         String[] profiles = env.getActiveProfiles();
20         return Map.of(
21             k1: "activeProfiles", profiles,
22             k2: "dev", Arrays.asList(profiles).contains(o: "dev")
23         );
24     }
25 }
```

- Bottom Status Bar:** The status bar at the bottom provides information about the project and environment:
 - Project status: "main 0 △ 3 Java: Ready"
 - File status: "edutbx (1 semana atrás)"
 - Editor settings: "Ln 1, Col 35 Espaços: 4 UTF-8 CRLF () Java" (with icons for Go Live and a gear)

OBRIGADO

