Documentação do Projeto: Sistema de Gerenciamento de Atividades Diversificadas (GAD) com IA

1. Introdução

1.1. Visão Geral do Projeto

Este Trabalho de Conclusão de Curso (TCC) propõe o desenvolvimento do Sistema de Gerenciamento de Atividades Diversificadas (GAD), uma plataforma web para o Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso do Sul (IFMS) – Câmpus Naviraí. O objetivo principal é automatizar e otimizar o processo de submissão, análise, classificação, validação e contabilização de horas de atividades complementares realizadas pelos alunos.

O projeto será desenvolvido em colaboração:

- **Vitor D. (Você):** Responsável pelo desenvolvimento do backend, incluindo a arquitetura do sistema, a lógica de negócios, a integração com o banco de dados e, crucialmente, a implementação dos componentes de Inteligência Artificial (IA) para processamento e análise dos certificados.
- **Mateus Anacleto de Araujo:** Responsável pelo desenvolvimento do frontend (em Angular) e pela pesquisa geral do projeto, incluindo a documentação mais burocrática, conforme orientação do Professor Laurentino (Áudio 03, Áudio 04).

O prazo para conclusão e apresentação do software é julho (Áudio 01), o que exige foco na implementação prática e na entrega de um produto funcional (MVP - Minimum Viable Product) que atenda aos requisitos essenciais.

1.2. Problema a Ser Solucionado

Atualmente, o gerenciamento de certificados e atividades complementares no IFMS Campus Naviraí é realizado de forma manual, frequentemente através do envio de documentos para pastas em serviços de nuvem (Google Drive), conforme descrito por Mateus. Este processo é suscetível a:

- Perda de arquivos.
- Dificuldades de organização.

- Demora na validação e contabilização das horas.
- Falta de um sistema centralizado e padronizado para acompanhamento por parte dos alunos e professores.
- Dificuldade em aplicar consistentemente as regras de classificação e limites de carga horária para diferentes tipos de atividades.

1.3. Solução Proposta: O Sistema GAD

O Sistema GAD visa resolver esses problemas através de uma plataforma web que permitirá:

- **Alunos:** Fazerem upload de seus certificados (PDF, JPG, PNG), sugerirem a classificação da atividade, acompanharem o status de validação e visualizarem o progresso de suas horas complementares.
- **Professores/Validadores:** Analisarem os documentos submetidos, validarem as informações extraídas pela IA, confirmarem ou corrigirem a classificação das atividades e aprovarem/reprovarem as horas, com justificativas.
- **IA:** Processar automaticamente os documentos enviados, extraindo informações chave (nome do aluno, título da atividade, carga horária, instituição, data), classificando a atividade conforme as normas do IFMS e auxiliando na validação.
- **Administradores:** Gerenciarem usuários, configurarem parâmetros do sistema (como tipos de atividades e limites de horas) e gerarem relatórios.

O foco deste documento está no componente backend e na integração de IA.

1.4. Objetivos do TCC (Backend e IA)

- Desenvolver um backend robusto e escalável utilizando Spring Boot (Java) e PostgreSQL.
- Implementar um módulo de IA capaz de:
    - Realizar OCR (Reconhecimento Óptico de Caracteres) em documentos PDF e imagens.
    - Extrair informações relevantes dos certificados.

    - Classificar automaticamente as atividades com base no conteúdo e nas regras do IFMS.
    - Validar a autenticidade dos certificados (quando possível, por exemplo, através de QR Codes).
- Implementar a lógica de negócios para gerenciamento de horas, incluindo a distribuição de horas excedentes.
- Garantir a segurança e integridade dos dados.
- Fornecer APIs RESTful bem definidas para comunicação com o frontend.
- Focar na implementação prática e demonstrar o domínio das técnicas envolvidas, conforme orientação do Professor Laurentino (Áudio 08, Áudio 10).

2. Requisitos do Sistema (Foco no Backend e IA)

2.1. Requisitos Funcionais Essenciais

- **RF-B001:** Permitir upload de arquivos de certificado (PDF, JPG, PNG) pelos alunos.
- **RF-B002:** Processar os arquivos enviados para extrair texto (OCR para imagens e PDFs escaneados, extração direta para PDFs textuais).
- **RF-B003:** Extrair automaticamente informações chave dos certificados: nome do aluno (se presente e necessário para cruzamento), nome da atividade/curso, instituição emissora, data de realização/emissão, carga horária.
- **RF-B004:** Permitir que o aluno sugira o tipo de classificação da atividade no momento do upload (conforme orientação do Professor Laurentino, Áudio 05).
- **RF-B005:** Classificar (ou validar a sugestão do aluno) a atividade com base no seu conteúdo e na tabela de classificação do IFMS, utilizando IA.
    - Siglas: UC, PE, PP, PD, PA, PV, TCC.
- **RF-B006:** Implementar a lógica de contabilização de horas, considerando:
    - Carga horária máxima por tipo de atividade (conforme regras da instituição).
    - Distribuição de horas remanescentes para outras classificações de atividades compatíveis, caso o certificado se enquadre em mais de uma ou o limite de

uma seja atingido.

- **RF-B007:** Validar a autenticidade de certificados através da leitura e processamento de QR Codes (se presentes e contiverem URLs ou dados verificáveis) (Professor Laurentino, Áudio 06, Áudio 10).
- **RF-B008:** Armazenar os documentos originais e as informações extraídas e validadas no banco de dados.
- **RF-B009:** Permitir que professores/validadores revisem as informações extraídas pela IA, a classificação sugerida e aprovem ou reprovem as horas, registrando justificativas.
- **RF-B010:** Fornecer APIs para o frontend exibir o status das atividades (pendente, em análise, aprovado, reprovado) e o total de horas contabilizadas por categoria.
- **RF-B011:** Gerenciar usuários (Alunos, Professores, Administradores) com diferentes níveis de acesso (autenticação e autorização).

2.2. Requisitos Não Funcionais

- **RNF-B001 (Desempenho):** O sistema deve processar os certificados em tempo hábil para não prejudicar a experiência do usuário. A extração e classificação por IA devem ser eficientes.
- **RNF-B002 (Segurança):** Proteger os dados dos alunos e os documentos enviados. Implementar autenticação e autorização robustas (Spring Security).
- **RNF-B003 (Manutenibilidade):** O código backend deve ser modular, bem documentado e seguir boas práticas de desenvolvimento para facilitar a manutenção e futuras expansões.
- **RNF-B004 (Eficiência de Recursos):** Evitar consumo excessivo de processamento, especialmente na leitura de documentos, conforme sua preocupação inicial. O uso de modelos de IA locais (via Ollama, por exemplo) pode ajudar a controlar custos e dependência de APIs externas.
- **RNF-B005 (Escalabilidade):** A arquitetura deve permitir escalabilidade futura, embora o foco inicial seja um MVP para o TCC.

3. Arquitetura do Backend e Tecnologias

3.1. Arquitetura Geral

Será adotada uma arquitetura monolítica modular para o backend, o que é adequado para a complexidade e o prazo de um TCC. Os principais componentes serão:

1. **API Layer (Spring MVC):** Exporá endpoints RESTful para interação com o frontend (Angular) e, potencialmente, outros sistemas no futuro.
2. **Service Layer:** Conterá a lógica de negócios principal, orquestrando as interações entre a API, os componentes de IA e a camada de persistência.
3. **AI/ML Layer:** Módulos dedicados ao processamento de documentos:
    - OCR e Extração de Texto.
    - Extração de Informações Estruturadas.
    - Classificação de Atividades.
    - Validação de QR Code.
4. **Data Access Layer (Spring Data JPA):** Responsável pela interação com o banco de dados PostgreSQL.
5. **Domain Layer:** Entidades que representam os dados do sistema (Aluno, Certificado, Atividade, etc.).

3.2. Estrutura do Monorepo (Conforme Sugerido)

A estrutura de monorepo no GitHub é uma boa abordagem para gerenciar os diferentes artefatos do projeto:

```
gad-ifms/
├── frontend_client/      # Angular (Responsabilidade do Mateus)
├── backend_api/          # Spring Boot (Sua Responsabilidade)
│   ├── src/
│   │   ├── main/
│   │   │   ├── java/
│   │   │   │   └── br/edu/ifms/gad/
│   │   │   │       ├── config/      # Configurações do Spring (Security, AI, etc.)
│   │   │   │       ├── controller/  # Controladores REST API
│   │   │   │       ├── domain/      # Entidades JPA, Enums
│   │   │   │       ├── dto/         # Data Transfer Objects
│   │   │   │       ├── repository/  # Interfaces Spring Data JPA
│   │   │   │       ├── service/     # Lógica de negócios
│   │   │   │       │   ├── ai/        # Sub-pacote para serviços de IA
│   │   │   │       │   │   ├── OcrService.java
│   │   │   │       │   │   ├── InformationExtractorService.java
│   │   │   │       │   │   ├── ClassificationService.java
│   │   │   │       │   │   └── QrCodeService.java
│   │   │   │       │   └── impl/      # Implementações dos serviços
│   │   │   │       └── exception/   # Handlers de exceção
│   │   │   └── resources/
│   │   │       ├── application.properties
│   │   │       └── static/      # Recursos estáticos (se houver)
│   │   └── test/          # Testes unitários e de integração
│   └── pom.xml            # Arquivo Maven
├── doc_site/              # Documentação do usuário/sistema (MkDocs Material)
└── pesquisa/              # Documentos da pesquisa, artigos, TCC em LaTeX (Mateus e você)
```

3.3. Tecnologias Propostas para o Backend e IA

- **Linguagem:** Java (JDK 17+ recomendado)
- **Framework Backend:** Spring Boot 3.x (com Spring MVC, Spring Data JPA, Spring Security)
- **Build Tool:** Apache Maven
- **Banco de Dados:**
  - Produção: PostgreSQL (versão recente)
  - Desenvolvimento/Testes: H2 Database (em memória)
  - **Extensão PostgreSQL:** pgvector para capacidades de busca semântica, caso necessário para classificação avançada ou busca de certificados similares. O link sobre **Data Visualization in PostgreSQL With Apache Superset** (Timescale Blog) mostra como PostgreSQL é versátil, e pgvector adiciona uma camada poderosa para IA.
- **OCR (Reconhecimento Óptico de Caracteres):**
  - **Tesseract OCR:** Open-source, pode ser usado via wrapper Java como Tess4J ou executando um script Python.
  - **Apache PDFBox:** Para extração de texto de PDFs que já contêm texto (não escaneados).
  - O artigo **Extracting Structured Data From Images Using Spring AI** (Baeldung) demonstra como Spring AI pode interagir com modelos de IA (como os da OpenAI) para analisar imagens, o que é diretamente aplicável a certificados em formato de imagem ou PDFs baseados em imagem.
- **Processamento de Linguagem Natural (PLN) e Modelos de IA:**
  - **Spring AI:** Abstração da Spring para trabalhar com vários modelos de IA. Facilita a integração de LLMs.
    - O vídeo sobre **Spring AI com Ollama** (https://www.google.com/search?q=youtube.com/2) é crucial aqui, mostrando como usar modelos locais.
    - O artigo da Baeldung sobre **Spring AI para extrair dados de imagens**

(Baeldung) é altamente relevante.

- **Ollama:** Para rodar LLMs open-source localmente (ex: Llama 3, Mistral, Phi-3). Isso garante privacidade dos dados dos certificados e reduz custos. O vídeo tutorial sobre **Ollama** (youtube.com/3) fornece um bom ponto de partida.
- **Hugging Face Transformers:** Acesso a uma vasta gama de modelos pré-treinados que podem ser usados via Ollama ou diretamente (se rodados em Python e acessados via API interna).
- **LangChain:** Framework para construir aplicações com LLMs. Embora o artigo da Baeldung sobre **Java LangChain** (Baeldung) indique que uma versão Java oficial robusta poderia não estar madura, Spring AI tem evoluído e pode cobrir muitas das funcionalidades de orquestração. A ideia de RAG (Retrieval Augmented Generation) discutida no vídeo sobre **LangChain & RAG** (https://www.google.com/search?q=youtube.com/0) e no artigo **Advanced RAG techniques** (MachineLearningMastery) é muito pertinente para a IA entender o contexto dos certificados.
- **Bibliotecas para extração de entidades (NER):** spaCy (se acessado via script Python) ou modelos NER da Hugging Face.

- **Leitura de QR Code:**
  - **ZXing (Zebra Crossing):** Biblioteca Java popular para decodificar QR codes e outros formatos de barcode.
- **Gerenciamento de Modelos de ML (Opcional para TCC, mas bom saber):**
  - **MLFlow:** Para rastrear experimentos, empacotar modelos e gerenciar o ciclo de vida de modelos de ML, se você treinar modelos customizados.
- **CI/CD (Automação):**
  - **GitHub Actions:** Para build, teste e deploy automatizados, como demonstrado no vídeo sobre **Deploying Spring Boot Backend** (youtube.com/1). O artigo sobre **CI/CD for LLM applications** (CircleCI) também oferece insights relevantes.
- **Segurança:**

    - **Spring Security:** Para autenticação e autorização. O artigo do **GeeksforGeeks sobre Spring Security** e o guia da **Permit.io sobre planejamento de autorização** (Permit.io) são referências úteis.
- **Outras Ferramentas Consideradas (Pelo Usuário):**
    - **Weka/Orange:** O professor mencionou Weka para pesquisa opcional (Áudio 09, Áudio 10). Podem ser usados para análise exploratória de dados ou para testar algoritmos de classificação clássicos, mas a integração direta no backend Spring Boot pode ser menos prioritária que LLMs via Spring AI.
    - **Scikit-learn, PyTorch, TensorFlow:** Mais relevantes se você estiver treinando modelos de ML customizados em Python. Para o TCC, focar em modelos pré-treinados e LLMs via Spring AI/Ollama pode ser mais eficiente.
    - **Mixtral:** Um LLM que pode ser rodado via Ollama.
    - **Google Colab:** Excelente para experimentação e treinamento de modelos em Python.

4. Detalhamento do Componente de IA

O fluxo de processamento de um certificado pela IA será:

4.1. Ingestão do Documento

- O aluno faz upload do arquivo (PDF, JPG, PNG) através da interface frontend.
- O backend recebe o arquivo via API REST.

4.2. OCR e Extração de Texto

- **Se PDF:**
    - Tentar extrair texto diretamente usando Apache PDFBox.
    - Se for um PDF baseado em imagem (escaneado) ou a extração direta falhar/for de baixa qualidade, proceder como imagem.
- **Se Imagem (JPG, PNG) ou PDF de Imagem:**
    - Aplicar pré-processamento na imagem se necessário (ex: binarização, remoção de ruído – OpenCV pode ser usado se um microserviço Python for

considerado, ou bibliotecas Java de imagem).
    - Utilizar Tesseract OCR (via Tess4J) para converter a imagem em texto.
    - Alternativamente, para extração estruturada direta de imagens, explorar as capacidades do Spring AI com modelos multimodais (conforme Baeldung sobre Spring AI e imagens).

4.3. Extração de Informações Chave (IE)

Com o texto extraído:

1. **Identificação de Campos:**
    - **Nome do Aluno:** Se necessário e não implicitamente ligado à sessão do usuário.
    - **Título da Atividade/Curso:** Crucial para classificação.
    - **Instituição Emissora:** Pode ajudar na validação e classificação.
    - **Data de Realização/Emissão:** Importante para verificar a validade temporal da atividade.
    - **Carga Horária:** Essencial para a contabilização.
2. **Técnicas:**
    - **Expressões Regulares (Regex):** Útil para padrões previsíveis (datas, formatos de carga horária).
    - **Modelos de Reconhecimento de Entidades Nomeadas (NER):** Modelos da Hugging Face (ex: BERTimbau para português) ou spaCy, acessados via Spring AI/Ollama ou um serviço Python, podem identificar entidades como EVENTO, ORGANIZACAO, DATA, DURACAO.
    - **LLMs (via Spring AI + Ollama):** Usar prompts estruturados para que o LLM extraia as informações desejadas em um formato específico (ex: JSON). Spring AI suporta mapeamento de saída para POJOs. A técnica de RAG pode ser empregada aqui para fornecer contexto ao LLM sobre o que procurar. O artigo da Baeldung sobre extração de dados de imagens com Spring AI é um bom exemplo.

4.4. Classificação da Atividade

1. **Input:** Texto extraído, informações chave (especialmente o título da atividade e instituição).
2. **Lógica de Classificação:**
    - O aluno poderá sugerir uma categoria (UC, PE, PP, PD, PA, PV, TCC).
    - A IA validará essa sugestão ou proporá uma classificação.
    - **Abordagem Baseada em LLM:** Enviar o texto/descrição da atividade para um LLM (via Spring AI + Ollama) e pedir para ele classificar com base na tabela de categorias e suas descrições fornecidas pelo IFMS. O prompt pode incluir as descrições de cada sigla.
    - **Abordagem Híbrida (Opcional/Avançado):**
        - **Palavras-chave e Regras:** Para classificações óbvias.
        - **Modelo de Classificação de Texto:** Um modelo mais leve (ex: um BERT fine-tuned para classificação de texto ou até mesmo algoritmos clássicos como SVM/Naive Bayes se houver dados de treinamento) pode ser usado se a performance do LLM for um problema ou para uma primeira triagem.
    - O sistema deve ser capaz de atribuir múltiplas classificações se a atividade for compatível com mais de uma categoria, conforme os exemplos fornecidos.

4.5. Validação de QR Code (Professor Laurentino, Áudio 06, 10)

- Se o documento contiver um QR Code, tentar decodificá-lo usando ZXing.
- Analisar o conteúdo do QR Code:
    - Se for uma URL, o sistema pode (opcionalmente, e com cuidado para evitar SSRF) tentar acessar a URL para verificar a validade do certificado (requer análise de segurança e viabilidade).
    - Se contiver dados estruturados, comparar com as informações extraídas do texto do certificado.
- Esta funcionalidade é para tentar identificar certificados inválidos, mas o professor ressaltou que não é obrigatório se aprofundar excessivamente.

4.6. Lógica de Gerenciamento de Horas

- Após a classificação (e validação pelo professor), o sistema aplicará as regras de negócio:
    - Verificar a carga horária máxima permitida para cada categoria (conforme "PLANILHA DE PONTUAÇÃO.pdf" ou regras atualizadas do IFMS).
    - Se a carga horária do certificado exceder o limite da categoria principal, e se o certificado for elegível para outras categorias, distribuir as horas excedentes para essas outras categorias, respeitando seus respectivos limites.
    - Exemplo: "Se uma atividade possuir 200 horas feitas pelo aluno e o mesmo anexar o certificado para PA, os outros 100 vão sobrar. Logo Se a atividade (certificado) é tida como PA e PV ao mesmo tempo pois essas duas classificações possuem muita semelhança, as horas podem ser distribuídas totalizando 200h."

5. Gerenciamento de Dados (PostgreSQL)

5.1. Modelo de Dados Preliminar (Entidades Principais)

- **Usuario**: (id, nome, email, senha_hash, tipo_usuario (ALUNO, PROFESSOR, ADMIN), matricula/siape)
- **Curso**: (id, nome_curso, horas_complementares_totais_exigidas)
- **CategoriaAtividade**: (id, sigla, descricao, limite_max_horas_curso)
- **Certificado**: (id, aluno_id (FK Usuario), data_upload, nome_arquivo_original, path_arquivo_armazenado, texto_extraido_ocr TEXT, status_processamento_ia (PENDENTE, PROCESSADO, ERRO), status_validacao_professor (PENDENTE, APROVADO, REPROVADO), professor_validador_id (FK Usuario, nullable), data_validacao_professor, justificativa_reprovacao TEXT)
- **InfoExtraidaCertificado**: (id, certificado_id (FK Certificado), titulo_atividade, instituicao, data_atividade, carga_horaria_declarada, dados_qr_code TEXT)
- **ClassificacaoSugerida**: (id, certificado_id (FK Certificado), categoria_id (FK

CategoriaAtividade), origem_sugestao (ALUNO, IA), confianca_ia (float, nullable))

- **HorasComputadas**: (id, certificado_id (FK Certificado), aluno_id (FK Usuario), categoria_id (FK CategoriaAtividade), horas_validadas_para_categoria, data_registro)

5.2. Uso do pgvector (Opcional, mas recomendado para exploração)

- Se a classificação baseada em LLM precisar de busca por similaridade (ex: encontrar exemplos de certificados já classificados para ajudar na decisão) ou se houver necessidade de encontrar atividades duplicadas/similares, pgvector pode ser usado para armazenar embeddings do texto dos certificados.
- O artigo sobre **Data Prep Kit** (Heidloff.net) menciona a preparação de dados para RAG e fine-tuning, o que pode envolver a geração de embeddings.

6. Fluxo de Trabalho do Sistema (Workflow)

Conforme solicitado pelo Professor Laurentino (Áudio 06, 07):

1. **Login:** Aluno, Professor ou Administrador acessa o sistema com suas credenciais.
2. **Upload pelo Aluno:**
   - Aluno faz o upload do arquivo do certificado (PDF, JPG, PNG).
   - Aluno preenche informações básicas (ex: nome da atividade, data) e pode sugerir uma ou mais categorias para a atividade (Áudio 05).
3. **Processamento pela IA (Assíncrono):**
   - O sistema enfileira o certificado para processamento.
   - **OCR & Extração de Texto:** O texto do documento é extraído.
   - **Extração de Informações:** A IA identifica título, carga horária, datas, instituição.
   - **Classificação pela IA:** A IA analisa o conteúdo e a sugestão do aluno, validando ou propondo as categorias apropriadas (com nível de confiança).
   - **Leitura de QR Code:** Se existir, o QR Code é lido e seus dados são extraídos.
   - As informações extraídas e as sugestões de classificação são armazenadas. O

status do certificado é atualizado (ex: "Aguardando Validação do Professor").

4. **Dashboard do Aluno:**
    - Aluno visualiza seus certificados enviados, o status de cada um (incluindo "horas não avaliadas" ou "em processamento") e o total de horas já validadas por categoria (Áudio 05).
5. **Validação pelo Professor:**
    - Professor acessa uma lista de certificados pendentes de validação.
    - Visualiza o documento original, as informações extraídas pela IA, a classificação sugerida pela IA e/ou pelo aluno.
    - Professor pode corrigir as informações extraídas ou a classificação.
    - Professor aprova ou reprova o certificado (com justificativa em caso de reprovação).
    - Ao aprovar, as horas são calculadas e distribuídas conforme as regras do sistema.
6. **Finalização:**
    - O status do certificado é atualizado para "Aprovado" ou "Reprovado".
    - As horas aprovadas são adicionadas ao histórico do aluno.
    - Aluno é notificado (opcional).
7. **Relatórios (Administrador/Professor):**
    - Visualização do progresso geral dos alunos, gargalos no processo de validação, etc.

7. Segurança

- **Autenticação:** Spring Security para gerenciar logins. Senhas devem ser armazenadas com hashing seguro (e.g., bcrypt).
- **Autorização:**
    - Spring Security para controle de acesso baseado em papéis (ALUNO, PROFESSOR, ADMIN) para diferentes funcionalidades e APIs.
    - Considerar os princípios de planejamento de autorização do guia da Permit.io.

- **Proteção de Dados:** Cuidado com o armazenamento e acesso aos arquivos de certificados, que podem conter dados pessoais.
- **Validação de Inputs:** Validar todos os dados recebidos pelas APIs para prevenir injeções e outros ataques.
- **Comunicação Segura:** Usar HTTPS para toda a comunicação entre frontend e backend. O vídeo sobre **Deploying Spring Boot Backend** (youtube.com/1) cobre a configuração de HTTPS com Nginx e Let's Encrypt.

8. Desenvolvimento e Implementação

8.1. Foco na Implementação Prática

Conforme o Professor Laurentino enfatizou (Áudio 08, Áudio 10, Áudio 11, Áudio 12), o objetivo é desenvolver e demonstrar a técnica. Replicar soluções existentes ou usar ferramentas prontas é aceitável e encorajado para cumprir o prazo. A complexidade deve ser de um trabalho de graduação.

8.2. Marcos de Desenvolvimento Sugeridos (para o Backend)

- **Mês 1: Configuração e Módulos Base**
  - Semana 1-2: Configuração do projeto Spring Boot, estrutura do monorepo, conexão com banco de dados (PostgreSQL/H2). Definição das entidades JPA básicas. Implementação da autenticação e autorização inicial com Spring Security.
  - Semana 3-4: Implementação do endpoint de upload de arquivos. Desenvolvimento do serviço de OCR básico (Tesseract/PDFBox) para extrair texto de PDFs e imagens. Testes iniciais de extração.
- **Mês 2: IA - Extração e Classificação Inicial**
  - Semana 5-6: Integração com Ollama e Spring AI. Implementação da extração de informações chave (regex, e início da exploração com LLM para campos mais complexos).
  - Semana 7-8: Desenvolvimento do serviço de classificação de atividades

usando LLM (prompt engineering inicial). Implementação da lógica de sugestão do aluno e validação pela IA.

- **Mês 3: Validação, Regras de Negócio e Finalização**
  - Semana 9-10: Implementação da lógica de contabilização e distribuição de horas. Implementação da leitura de QR Code (ZXing).
  - Semana 11-12: Refinamento dos módulos de IA (melhoria de prompts, tratamento de erros). Testes de integração completos. Desenvolvimento das APIs para o frontend (colaboração com Mateus). Documentação da API (Swagger/OpenAPI).
- **Julho: Testes, Ajustes e Preparação para Apresentação**
  - Testes end-to-end (com o frontend).
  - Ajustes de performance e usabilidade.
  - Preparação da documentação final do TCC e da apresentação.

8.3. Pesquisa com Weka (Opcional)

O Professor Laurentino mencionou o Weka como uma ferramenta para uma pesquisa adicional, possivelmente para analisar o comportamento de diferentes algoritmos de classificação nos dados dos certificados (Áudio 09, Áudio 10). Esta pode ser uma atividade paralela ou posterior ao desenvolvimento do MVP, focada mais na parte de pesquisa do TCC e não necessariamente integrada ao sistema de produção.

9. Considerações Finais

Este projeto é desafiador, mas viável dentro do escopo de um TCC, especialmente com o foco na implementação prática e no uso de ferramentas e modelos existentes. A colaboração eficaz com seu colega Mateus e a comunicação contínua com o Professor Laurentino serão cruciais para o sucesso.

Lembre-se de que a documentação do processo de desenvolvimento, as escolhas de design e os desafios encontrados (e como foram superados) são tão importantes

quanto o software final em um TCC.

Este documento serve como um guia inicial e deve ser iterativamente refinado à medida que o projeto avança. Boa sorte!