Plagiarism Detector AI - Documentação Técnica Completa

Desenvolvido por: Lucas Andre S

Visão Geral do Sistema

O Plagiarism Detector AI é uma plataforma avançada de detecção de plágio e conteúdo gerado por IA, projetada para garantir a integridade acadêmica e profissional de documentos. O sistema utiliza algoritmos sofisticados de análise textual e integração com modelos de linguagem de grande escala (LLM) para fornecer resultados precisos e confiáveis.

Arquitetura do Sistema

Stack Tecnológico

Backend:

- Node.js com TypeScript
- Express.js para servidor HTTP
- tRPC para comunicação type-safe entre frontend e backend
- Drizzle ORM para acesso ao banco de dados
- PostgreSQL (MySQL/TiDB) como banco de dados principal
- AWS S3 para armazenamento de arquivos

Frontend:

- React 19 com TypeScript
- TailwindCSS 4 para estilização
- shadcn/ui para componentes de interface

- Wouter para roteamento
- TanStack Query (React Query) para gerenciamento de estado

Inteligência Artificial:

- Integração com LLM para detecção de conteúdo gerado por IA
- Algoritmos proprietários de similaridade textual
- Análise de padrões linguísticos e perplexidade

Estrutura do Banco de Dados

Tabela: users

Armazena informações dos usuários autenticados.

Campos:

- `id` (INT, PK): Identificador único do usuário
- `openId` (VARCHAR): ID do OAuth Manus
- `name` (TEXT): Nome do usuário
- `email` (VARCHAR): Email do usuário
- `loginMethod` (VARCHAR): Método de login utilizado
- `role` (ENUM): Papel do usuário (user, admin)
- `createdAt` (TIMESTAMP): Data de criação
- `updatedAt` (TIMESTAMP): Data de atualização
- `lastSignedIn` (TIMESTAMP): Último login

Tabela: documents

Armazena metadados dos documentos enviados para análise.

Campos:

- `id` (INT, PK): Identificador único do documento

- `userId` (INT, FK): ID do usuário proprietário
- `filename` (VARCHAR): Nome do arquivo no sistema
- `originalFilename` (VARCHAR): Nome original do arquivo
- `fileType` (VARCHAR): Tipo MIME do arquivo
- `fileSize` (INT): Tamanho do arquivo em bytes
- `s3Key` (TEXT): Chave do arquivo no S3
- `s3Url` (TEXT): URL pública do arquivo
- `extractedText` (TEXT): Texto extraído do documento
- `status` (ENUM): Status do processamento (pending, processing, completed, failed)
- `createdAt` (TIMESTAMP): Data de upload
- `updatedAt` (TIMESTAMP): Data de atualização

Tabela: analyses

Armazena resultados das análises de plágio e IA.

Campos:

- `id` (INT, PK): Identificador único da análise
- `documentId` (INT, FK): ID do documento analisado
- `userId` (INT, FK): ID do usuário
- `plagiarismPercentage` (FLOAT): Percentual de plágio detectado
- `aiContentPercentage` (FLOAT): Percentual de conteúdo gerado por IA
- `confidenceScore` (FLOAT): Score de confiança da análise
- `totalSources` (INT): Número total de fontes encontradas
- `status` (ENUM): Status da análise
- `errorMessage` (TEXT): Mensagem de erro, se houver
- `analysisData` (TEXT): Dados completos da análise em JSON
- `createdAt` (TIMESTAMP): Data de criação

- `completedAt` (TIMESTAMP): Data de conclusão

Tabela: plagiarism_sources

Armazena fontes de plágio identificadas.

Campos:

- `id` (INT, PK): Identificador único
- `analysisId` (INT, FK): ID da análise
- `sourceUrl` (TEXT): URL da fonte
- `sourceTitle` (TEXT): Título da fonte
- `sourceType` (VARCHAR): Tipo da fonte (database, internet, academic)
- `similarityScore` (FLOAT): Score de similaridade (0-1)
- `matchedText` (TEXT): Texto correspondente encontrado
- `originalText` (TEXT): Texto original do documento
- `startPosition` (INT): Posição inicial do trecho
- `endPosition` (INT): Posição final do trecho
- `createdAt` (TIMESTAMP): Data de criação

Tabela: ai_detection_results

Armazena resultados da detecção de conteúdo gerado por IA.

Campos:

- `id` (INT, PK): Identificador único
- `analysisId` (INT, FK): ID da análise
- `textSegment` (TEXT): Segmento de texto analisado
- `aiProbability` (FLOAT): Probabilidade de ser gerado por IA (0-1)
- `startPosition` (INT): Posição inicial do segmento
- `endPosition` (INT): Posição final do segmento
- `detectionMethod` (VARCHAR): Método de detecção utilizado

- `createdAt` (TIMESTAMP): Data de criação

Módulos do Sistema

1. Módulo de Extração de Texto (textExtractor.ts)

Responsável por extrair texto de múltiplos formatos de arquivo.

Formatos Suportados:

- PDF: Utiliza `pdftotext` para extração
- DOCX: Utiliza biblioteca `python-docx`
- TXT: Leitura direta de arquivo
- PPT/PPTX: Utiliza biblioteca `python-pptx`

Funcionalidades:

- Extração de texto com limpeza automática
- Contagem de palavras e caracteres
- Divisão de texto em chunks para processamento

Exemplo de Uso:

```
const result = await extractTextFromFile('/path/to/file.pdf',
'application/pdf');
console.log(result.text); // Texto extraído
console.log(result.wordCount); // Número de palavras
```

2. Módulo de Detecção de Plágio (plagiarismDetector.ts)

Implementa múltiplos algoritmos para detecção de plágio.

Algoritmos Implementados:

a) Jaccard Similarity Calcula similaridade baseada em conjuntos de palavras.

- Fórmula: |A ∩ B| / |A ∪ B|

- Ideal para textos curtos

b) Cosine Similarity Calcula similaridade usando vetores de frequência de palavras.

- Utiliza produto escalar e magnitude de vetores
- Eficaz para documentos longos

c) N-gram Similarity Compara sequências de N palavras consecutivas.

- Padrão: trigramas (n=3)
- Detecta paráfrases e reorganizações

d) Semantic Similarity (LLM) Utiliza modelos de linguagem para análise semântica.

- Detecta similaridade de significado
- Identifica paráfrases sofisticadas

Fontes de Comparação:

- Base de dados interna de documentos
- Busca simulada em fontes da internet
- Análise com IA para identificação de fontes conhecidas

Exemplo de Uso:

```
const result = await detectPlagiarism(text);
console.log(result.overallPercentage); // % de plágio
console.log(result.matches); // Fontes encontradas
console.log(result.confidenceScore); // Score de confiança
```

3. Módulo de Detecção de IA (aiDetector.ts)

Detecta conteúdo gerado por inteligência artificial.

Métodos de Detecção:

a) Análise de Padrões Textuais

- Consistência no comprimento de frases

- Uso de frases comuns em IA
- Frequência de pronomes pessoais

b) Análise de Perplexidade e Burstiness

- Variância no comprimento de frases
- Diversidade vocabular
- Padrões de repetição

c) Detecção com LLM

- Análise semântica profunda
- Identificação de estilo de escrita
- Score de probabilidade de IA

Score Final:

- Média ponderada dos três métodos
- Padrões: 20%, Perplexidade: 20%, LLM: 60%

Exemplo de Uso:

```
const result = await detectAIContent(text);
console.log(result.overallPercentage); // % de conteúdo IA
console.log(result.segments); // Segmentos detectados
console.log(result.confidenceScore); // Confiança
```

API tRPC

Rotas de Autenticação (auth)

auth.me

- Tipo: Query
- Descrição: Retorna informações do usuário autenticado
- Retorno: User | null

auth.logout

- Tipo: Mutation
- Descrição: Realiza logout do usuário
- Retorno: { success: boolean }

Rotas de Documentos (documents)

documents.upload

- Tipo: Mutation
- Input: { filename, fileType, fileSize, fileData (base64) }
- Descrição: Faz upload de documento e extrai texto
- Retorno: { success, documentId, extractedText, wordCount }

documents.list

- Tipo: Query
- Descrição: Lista todos os documentos do usuário
- Retorno: Document[]

documents.get

- Tipo: Query
- Input: { id: number }
- Descrição: Retorna detalhes de um documento
- Retorno: Document | null

Rotas de Análise (analysis)

analysis.create

- Tipo: Mutation
- Input: { documentId: number }
- Descrição: Cria nova análise de plágio e IA

- Retorno: { success, analysisId, plagiarismPercentage, aiContentPercentage, confidenceScore }

analysis.get

- Tipo: Query
- Input: { id: number }
- Descrição: Retorna análise completa com fontes e resultados de IA
- Retorno: Analysis com sources e aiResults

analysis.getByDocument

- Tipo: Query
- Input: { documentId: number }
- Descrição: Retorna análise mais recente de um documento
- Retorno: Analysis | null

analysis.list

- Tipo: Query
- Descrição: Lista todas as análises do usuário
- Retorno: Analysis[]

Rotas de Dashboard (dashboard)

dashboard.stats

- Tipo: Query
- Descrição: Retorna estatísticas gerais do usuário
- Retorno: { totalDocuments, totalAnalyses, completedAnalyses, avgPlagiarismPercentage, avgAIPercentage }

Interface do Usuário

Página Home (/)

Landing page profissional com:

- Hero section com gradiente indigo/purple
- Grid de features destacando funcionalidades
- Seção “How It Works” com 3 etapas
- Lista de recursos principais
- Call-to-action para registro

Página Dashboard (/dashboard)

Dashboard principal do usuário com:

- Cards de estatísticas (documentos, análises, médias)
- Botão de ação rápida para upload
- Lista de análises recentes com métricas
- Design responsivo com gradientes

Página Upload (/upload)

Interface de upload de documentos com:

- Drag-and-drop funcional
- Validação de tipo e tamanho de arquivo
- Preview de arquivo selecionado
- Lista de funcionalidades de análise
- Suporte para PDF, DOCX, TXT, PPT (max 10MB)

Página Analyze (/analyze/:id)

Visualização detalhada de análise com:

- Cards de métricas principais (plágio, IA, confiança)
- Progress bars coloridas
- Tabs para fontes de plágio e segmentos de IA
- Badges de severidade
- Links para fontes externas

Fluxo de Análise

1. **Upload**: Usuário faz upload do documento
2. **Extração**: Sistema extrai texto do arquivo
3. **Armazenamento**: Arquivo salvo no S3, metadata no banco
4. **Análise de Plágio**: Execução de algoritmos de similaridade
5. **Análise de IA**: Detecção de conteúdo gerado por IA
6. **Armazenamento de Resultados**: Salvamento de fontes e segmentos
7. **Cálculo de Métricas**: Agregação de percentuais e scores
8. **Exibição**: Apresentação dos resultados no dashboard

Segurança

Autenticação

- OAuth2 com Manus
- JWT para sessões
- Cookies HTTP-only e secure

Armazenamento

- Arquivos criptografados no S3
- Acesso restrito por usuário
- URLs presignadas para download

Validação

- Validação de tipos de arquivo
- Limite de tamanho (10MB)
- Sanitização de inputs

Performance

Otimizações

- Processamento assíncrono de análises
- Cache de resultados no banco
- Lazy loading de componentes
- Otimização de queries com índices

Escalabilidade

- Arquitetura stateless
- Separação de concerns (frontend/backend)
- Possibilidade de processamento distribuído
- S3 para armazenamento escalável

Testes

Testes Unitários

Implementados com Vitest:

- Testes de autenticação
- Testes de rotas tRPC
- Testes de dashboard stats
- Testes de operações de documentos

Executar testes:

```
pnpm test
```

Deployment

Requisitos

- Node.js 22+
- PostgreSQL/MySQL/TiDB
- AWS S3 ou compatível
- Python 3.11+ (para extração de texto)

Variáveis de Ambiente

- `DATABASE_URL`: String de conexão do banco
- `JWT_SECRET`: Segredo para assinatura de tokens
- `VITE_APP_ID`: ID da aplicação OAuth
- `OAUTH_SERVER_URL`: URL do servidor OAuth
- Credenciais S3 (injetadas automaticamente)

Comandos

```
# Desenvolvimento
pnpm dev

# Build
pnpm build

# Produção
pnpm start

# Migração do banco
pnpm db:push
```

Métricas e KPIs

Métricas de Qualidade

- **Plagiarism Percentage**: 0-100% (quanto menor, melhor)
- **AI Content Percentage**: 0-100% (quanto menor, melhor)
- **Confidence Score**: 0-100% (quanto maior, melhor)
- **Total Sources**: Número de fontes encontradas

Interpretação

- **Plágio < 15%**: Aceitável (verde)
- **Plágio 15-40%**: Atenção (amarelo)
- **Plágio > 40%**: Crítico (vermelho)
- **IA < 20%**: Aceitável (verde)
- **IA 20-50%**: Atenção (amarelo)
- **IA > 50%**: Crítico (vermelho)

Limitações Conhecidas

1. **Formatos de Arquivo**: Limitado a PDF, DOCX, TXT, PPT
2. **Tamanho de Arquivo**: Máximo 10MB
3. **Idioma**: Otimizado para textos em português e inglês
4. **Fontes de Comparação**: Base de dados limitada (simulada)
5. **Tempo de Processamento**: Varia conforme tamanho do documento

Roadmap Futuro

Funcionalidades Planejadas

- Exportação de relatórios em PDF
- Integração com APIs de busca acadêmica (Google Scholar, PubMed)
- Suporte para mais formatos (ODT, RTF, HTML)
- Análise de imagens e gráficos
- API pública para integração
- Sistema de notificações em tempo real
- Comparação entre múltiplos documentos
- Detecção de tradução automática

Melhorias Técnicas

- Cache distribuído com Redis
- Processamento em background com filas
- Websockets para atualizações em tempo real
- Otimização de algoritmos de detecção
- Treinamento de modelos próprios de IA

Suporte e Manutenção

Desenvolvedor: Lucas Andre S

Tecnologias Utilizadas:

- TypeScript, React, Node.js
- PostgreSQL, S3, tRPC
- TailwindCSS, shadcn/ui
- Python (extração de texto)
- LLM (detecção de IA)

Licença: Proprietário

Documentação criada em: 14 de Dezembro de 2024 Versão do Sistema: 1.0.0 Desenvolvido por: Lucas Andre S