

Requisitos

- [REQUISITOS](#)
- [PRIORIZAÇÃO DE REQUISITOS](#)
- [CASOS DE USOS](#)

REQUISITOS

Documento de Requisitos do Projeto: Kronos

1. Visão Geral do Projeto

O **Kronos** é um sistema web projetado para transcrever o conteúdo de ficheiros de vídeo e áudio. O utilizador poderá fazer o upload de um ficheiro de multimédia, o sistema extrairá o áudio, enviará para um serviço de inteligência artificial (AssemblyAI) para transcrição e, por fim, armazenará e exibirá o texto resultante. O sistema deve garantir que um mesmo ficheiro não seja processado e armazenado múltiplas vezes.

2. Requisitos Funcionais (RF)

Os Requisitos Funcionais descrevem **o que** o sistema deve fazer.

| ID | Requisito | Descrição |

| :--- | :--- | :--- |

| **RF01** | Upload de Mídia | O sistema deve permitir que o utilizador selecione e faça o upload de um ficheiro de vídeo (ex: .mp4, .mov, .avi) ou áudio (ex: .mp3, .wav) através de uma interface web. |

| **RF02** | Extração de Áudio | O sistema deve ser capaz de extrair a faixa de áudio de um ficheiro de vídeo enviado. Esta operação deve ocorrer no lado do cliente (frontend) para otimizar o envio de dados para o backend. |

| **RF03** | Iniciar Transcrição | O utilizador deve poder iniciar o processo de transcrição após o upload do ficheiro. O sistema deve enviar o áudio extraído para o backend. |

| **RF04** | Integração com a API de Transcrição | O backend deve comunicar com a API do AssemblyAI, enviando o ficheiro de áudio e tratando a resposta para obter o texto transcrito. |

| **RF05** | Armazenamento da Transcrição | O texto da transcrição, juntamente com metadados do ficheiro original, deve ser armazenado na base de dados PostgreSQL. |

| **RF06** | Prevenção de Duplicidade | O sistema não deve permitir a transcrição de um vídeo que já foi processado. A verificação deve ser feita antes de enviar o ficheiro para a API do AssemblyAI para evitar custos desnecessários. |

| **RF07** | Exibição do Resultado | Após a conclusão do processo, a transcrição de texto deve ser exibida de forma clara para o utilizador na interface. |

| **RF08** | Feedback de Processamento | O utilizador deve receber feedback visual durante as etapas do processo, como "A extrair áudio...", "A enviar para o servidor...", "A transcrever...". |

| **RF09** | Listagem de Transcrições | O sistema deve exibir uma lista com o histórico de transcrições já realizadas, permitindo ao utilizador visualizá-las novamente. |

| **RF10** | Tratamento de Erros | O sistema deve informar o utilizador de forma amigável caso ocorra um erro em qualquer etapa (ex: formato de ficheiro inválido, falha na API, erro de rede). |

3. Requisitos Não Funcionais (RNF)

Os Requisitos Não Funcionais descrevem **como** o sistema deve operar, focando nas suas qualidades.

| ID | Requisito | Descrição |

| :--- | :--- | :--- |

| **RNF01** | Desempenho | A extração de áudio no frontend para um vídeo de 5 minutos deve ser concluída em menos de 30 segundos numa máquina comum. A interface deve permanecer responsiva durante o processo. |

| **RNF02** | Usabilidade | A interface deve ser limpa, intuitiva e minimalista. O processo de upload e transcrição deve ser realizado em no máximo 3 cliques. |

| **RNF03** | Segurança | A chave da API do AssemblyAI deve ser armazenada de forma segura no backend (variáveis de ambiente) e nunca exposta no frontend. |

| **RNF04** | Confiabilidade | O sistema deve registar logs de erros no backend para facilitar a depuração. A aplicação deve ser resiliente a falhas na API externa, com mecanismos de nova tentativa ou notificação de falha. |

| **RNF05** | Escalabilidade | A arquitetura dockerizada deve permitir que as instâncias do backend e da base de dados sejam escaladas de forma independente, se necessário. |

| **RNF06** | Manutenibilidade | O código-fonte, tanto no frontend quanto no backend, deve seguir as melhores práticas de desenvolvimento com TypeScript, incluindo tipagem forte e modularização. |

4. Histórias de Utilizador

Uma forma ágil de descrever a funcionalidade da perspetiva do utilizador.

HU1: Transcrever um novo vídeo

- **Como um** utilizador,
- **Eu quero** fazer o upload de um ficheiro de vídeo,
- **Para que** eu possa obter a transcrição em texto do seu conteúdo.

Critérios de Aceite:

1. Dado que estou na página inicial, vejo um botão ou área para "Selecionar Ficheiro".
2. Quando eu seleciono um ficheiro de vídeo válido, o sistema carrega-o e extrai o áudio.
3. Enquanto o sistema processa, eu vejo um indicador de progresso.
4. Quando a transcrição estiver pronta, o texto é exibido no ecrã.
5. Se eu enviar um vídeo que já foi transcrito, o sistema informa-me e exibe a transcrição existente sem reprocessá-la.

HU2: Consultar transcrições antigas

- **Como um** utilizador,
- **Eu quero** ver uma lista das minhas transcrições anteriores,
- **Para que** eu possa aceder e reler os textos sem precisar de fazer o upload novamente.

Critérios de Aceite:

1. Dado que estou na aplicação, existe uma secção ou página chamada "As Minhas Transcrições".
2. Nessa página, vejo uma lista de todas as transcrições concluídas, identificadas pelo nome do ficheiro original.

3. Quando eu clico num item da lista, sou direcionado para um ecrã que exhibe o texto completo daquela transcrição.

5. Proposta de Modelo de Dados (PostgreSQL)

Para atender aos requisitos, uma tabela principal pode ser criada para armazenar as transcrições.

****Tabela: `transcriptions`****

Nome da Coluna	Tipo de Dado	Restrições	Descrição
`id`	`UUID`	`PRIMARY KEY`	Identificador único para cada registo de transcrição.
`original_filename`	`TEXT`	`NOT NULL`	Nome do ficheiro que o utilizador enviou.
`file_hash`	`VARCHAR(64)`	`NOT NULL, UNIQUE`	Hash (SHA-256) do conteúdo do ficheiro para garantir a unicidade.
`transcription_text`	`TEXT`		O texto completo gerado pela API de transcrição.
`status`	`VARCHAR(20)`	`NOT NULL`	Estado do processo (ex: 'processing', 'completed', 'failed').
`error_message`	`TEXT`		Mensagem de erro, caso o processo falhe.
`created_at`	`TIMESTAMPTZ`	`NOT NULL`	Data e hora de quando o registo foi criado.
`updated_at`	`TIMESTAMPTZ`	`NOT NULL`	Data e hora da última atualização do registo.

****Notas Técnicas para Implementação:****

- **Extração de Áudio no Frontend:**** Considere usar a biblioteca `ffmpeg.wasm`, que compila o FFmpeg para WebAssembly, permitindo a manipulação de multimédia diretamente no navegador.
- **Cálculo do Hash:**** Para o ****RF06****, o hash do ficheiro deve ser calculado no frontend (usando a API `SubtleCrypto` do navegador) e enviado ao backend. O backend consultará a tabela `transcriptions` pela coluna `file_hash` antes de prosseguir.
- **Transferência de Ficheiros:**** Pesquise a conversão do áudio extraído para Base64 ou o envio como `FormData` (multipart/form-data) para o backend. `FormData` costuma ser mais eficiente para ficheiros binários.

PRIORIZAÇÃO DE REQUISITOS

Priorização de Requisitos do Projeto Kronos (Método MoSCoW)

Este documento utiliza o método MoSCoW para classificar os requisitos funcionais do projeto Kronos, com o objetivo de definir o escopo do Produto Mínimo Viável (MVP) e planejar as fases seguintes do desenvolvimento.

1. Must-Have (Tem que ter)

Estes requisitos são **essenciais e inegociáveis** para a primeira versão do produto. Sem eles, o sistema não funciona ou não entrega o seu valor principal. O foco aqui é construir o fluxo completo da **História de Utilizador 1 (HU1)**.

ID	Requisito	Justificação
RF01	Upload de Mídia	Ponto de partida de toda a operação. Sem upload, nada acontece.
RF02	Extração de Áudio	Lógica central para processar vídeos, que é o caso de uso principal.
RF03	Iniciar Transcrição	Ação fundamental do utilizador para começar o processo.
RF04	Integração com a API de Transcrição	O coração técnico do projeto; sem a API externa, não há transcrição.
RF05	Armazenamento da Transcrição	É crucial persistir o resultado para que o utilizador possa vê-lo.
RF06	Prevenção de Duplicidade	Requisito de negócio chave para otimizar custos e evitar reprocessamento.
RF07	Exibição do Resultado	O utilizador precisa de ver o produto final do seu pedido.
RF10	Tratamento de Erros	Garante uma experiência minimamente funcional, informando o utilizador sobre falhas básicas (ex: formato inválido).

2. Should-Have (Deveria ter)

Estes requisitos são muito importantes, mas não são vitais para a primeira versão. Devem ser implementados logo após o MVP para enriquecer a experiência do utilizador. Correspondem, em grande parte, à **História de Utilizador 2 (HU2)**.

ID	Requisito	Justificação
RF09	Listagem de Transcrições	Adiciona um grande valor, permitindo que os utilizadores revejam trabalhos anteriores, mas o sistema funciona sem isso inicialmente.

| **RF08** | Feedback de Processamento | Melhora significativamente a usabilidade, mas um simples "a carregar..." é suficiente para o MVP. Um feedback mais detalhado pode vir depois. |

3. Could-Have (Poderia ter)

Estes requisitos são desejáveis, mas menos importantes. Podem ser vistos como melhorias que seriam implementadas se houvesse tempo e recursos disponíveis, após a entrega dos "Must" e "Should".

| ID | Requisito (Sugestão) | Justificação |

| :--- | :--- | :--- |

| **CH01** | Editar Transcrição | Permitir que o utilizador corrija pequenos erros no texto gerado pela IA. |

| **CH02** | Exportar para .txt ou .docx | Adicionar a funcionalidade de exportar o texto para diferentes formatos de ficheiro. |

| **CH03** | Suporte a mais formatos de multimédia | Expandir a compatibilidade para além dos formatos mais comuns. |

| **CH04** | Notificação de Conclusão | Enviar uma notificação no navegador quando uma transcrição longa for concluída. |

4. Won't-Have (Não terá - por agora)

Funcionalidades que foram consideradas, mas que estão explicitamente fora do escopo do projeto inicial para evitar o "scope creep" (aumento descontrolado do escopo).

| ID | Requisito (Excluído) | Justificação |

| :--- | :--- | :--- |

| **WH01** | Contas de Utilizador e Autenticação | Aumenta muito a complexidade. A versão inicial será de uso público e anónimo. |

| **WH02** | Transcrição em Tempo Real (Live) | Requer uma arquitetura completamente diferente (websockets, streaming) e está fora do objetivo inicial. |

| **WH03** | Funcionalidades Colaborativas | Partilhar e editar transcrições em equipa está fora do escopo. |

| **WH04** | Suporte a Múltiplos Idiomas | O foco inicial será apenas num idioma (ex: português ou inglês) para simplificar a integração com a API. |

Conclusão do Planeamento

O plano de desenvolvimento deve focar-se em entregar todos os requisitos **Must-Have** primeiro para construir um MVP robusto. Em seguida, o foco deve passar para os itens **Should-Have** para completar as funcionalidades mais pedidas.

CASOS DE USOS

DIAGRAMA CASO DE USO

