Proposta de Projeto Final:

BrazucaTalks: Fluência Personalizada Impulsionada por GenAI.

21/12/2025

—

Proponente:

Yuri Matheus Sousa dos Santos

Papel: Líder do Projeto e Arquiteto de Software

Destinatário: Prof. Me. Adriano Antunes Prates

Versão: 1.0

Visão geral

O BrazucaTalks emerge como uma solução tecnológica projetada para atender à crescente demanda por ensino de idiomas personalizado. O cenário educacional atual enfrenta duas barreiras críticas: a predominância de materiais didáticos estáticos que, embora eficazes para a fixação gramatical, mostram-se insuficientes para o desenvolvimento de competências orais e a inviabilidade logística de fornecer tutoria humana individualizada em larga escala. Diante desse contexto, o desenvolvimento do BrazucaTalks justifica-se como uma resposta estratégica a essas limitações. A plataforma utiliza tecnologias avançadas, especificamente Inteligência Artificial e recursos de conversão texto-fala (Text-to-Speech), para simular a interação com um tutor nativo, preenchendo a lacuna entre a teoria dos livros e a prática da conversação.

Tecnologias adotadas

A arquitetura do BrazucaTalks será desenvolvida utilizando uma abordagem moderna baseada em microsserviços e separação de responsabilidades entre Frontend (Interação Visual) e Backend (Lógica de Negócio e IA). Abaixo estão listadas as principais tecnologias, frameworks e protocolos utilizados:

- Backend e API
    - O núcleo do sistema foi desenvolvido em Python, escolhido por sua robustez no ecossistema de Inteligência Artificial e facilidade de prototipagem rápida.
    - Linguagem: Python 3.10+
    - Framework Web: FastAPI - Utilizado para construção da API RESTful. Escolhido por sua alta performance (assíncrona), suporte nativo a concorrência (async/await) e geração automática de documentação (Swagger UI).
    - Validação de Dados: Pydantic - Responsável pela modelagem e validação rigorosa dos dados de entrada e saída (ex: AskRequest, TranscribeResponse), garantindo a integridade da comunicação entre front e back.
    - Gerenciamento de Ambiente: Python-dotenv - Para segurança e configuração de variáveis de ambiente sensíveis.

    - Middleware: CORSMiddleware - Configurado para permitir a comunicação segura entre o cliente (React) e o servidor, gerenciando as origens permitidas (CORS_ORIGINS).
    - Manipulação de Arquivos: Bibliotecas nativas (pathlib, os, shutil) para gerenciamento de arquivos de áudio temporários e estáticos.
- Armazenamento e Cache Distribuído
    - Para garantir baixa latência e persistência de contexto em um ambiente distribuído, foi utilizada a tecnologia Redis Stack:
    - Banco de Dados em Memória: Redis - Atua como a camada de dados de alta performance.
    - Vector Database (RediSearch): Utilizado para armazenar embeddings (vetores matemáticos) das perguntas e respostas. Isso habilita o Cache Semântico, permitindo que o sistema identifique perguntas similares (mesmo com palavras diferentes) e responda instantaneamente sem reprocessar no LLM.
    - Gestão de Sessão (State Management): Como os LLMs são nativamente stateless (não têm memória), o Redis é responsável por armazenar o Histórico da Conversa de cada usuário (identificado via uuid). Isso permite que o "BrazucaTalks" lembre do nome do aluno e do contexto das interações anteriores, simulando uma memória de longo prazo.
- Frontend e Interface Interativa
    - A interface do usuário foi construída visando uma experiência imersiva, utilizando renderização 3D para o avatar do tutor.
    - Biblioteca de UI: React.js - Biblioteca JavaScript para construção de interfaces reativas baseadas em componentes.
    - Renderização 3D: Three.js via React Three Fiber (R3F) - Utilizado para renderizar o ambiente 3D e o modelo do avatar (.gltf/.glb) diretamente no navegador através de WebGL.
    - Abstrações 3D: Drei - Biblioteca auxiliar utilizada para carregamento otimizado de modelos (useGLTF) e gerenciamento de animações (useAnimations).
    - Animação Facial (Lip-Sync): Algoritmo customizado utilizando Web Audio API (AudioAnalyser). O sistema analisa a frequência do áudio em tempo real para

calcular a energia da fala e modular os Morph Targets (ex: mouthOpen, viseme_aa) da malha 3D, criando uma sincronização labial dinâmica.

- Inteligência Artificial e Processamento
    - Embora encapsulado no código apresentado, o sistema integra três pilares de IA Generativa:
    - LLM (Large Language Model): Integração com modelos de linguagem (via Ollama) para geração de respostas contextuais e humanizadas.
    - RAG (Retrieval-Augmented Generation): Implementado no endpoint /ask para fornecer contexto cultural brasileiro à IA antes da geração da resposta.
    - STT (Speech-to-Text): Módulo de transcrição de áudio (endpoint /transcribe), convertendo a fala do aluno em texto processável.
    - TTS (Text-to-Speech): Módulo de síntese de voz, responsável por gerar o arquivo de áudio (.wav/.mp3) que o avatar "fala".
- Protocolos e Padrões de Comunicação
    - HTTP/1.1: Utilizado para comunicação síncrona via verbos REST (POST para envio de áudio/texto, GET para verificação de saúde).
    - JSON: Padrão de troca de mensagens textuais.
    - Multipart/Form-Data: Protocolo utilizado para o upload eficiente de arquivos de áudio binários (Blobs) do navegador para o servidor.

Justificativa

O desenvolvimento do BrazucaTalks justifica-se como objeto de estudo prático para a disciplina, pois sua arquitetura foi desenhada para enfrentar desafios inerentes à computação distribuída moderna. O sistema não opera de forma monolítica, mas sim através da orquestração de componentes independentes que se comunicam via rede.

Abaixo, detalha-se a relação entre a implementação do projeto e os conceitos teóricos da disciplina:

- Arquitetura Cliente-Servidor Desacoplada
    - O sistema implementa estritamente o modelo Cliente-Servidor. O Frontend (React/Three.js) atua como um cliente "magro" responsável apenas pela renderização e captura de input, enquanto o processamento pesado (Inferência de IA, Transcrição, Síntese de Voz) é delegado ao Backend (FastAPI). Essa separação permite a evolução independente das camadas e a

Transparência de Localização, onde o cliente desconhece a infraestrutura física do servidor.

- Gestão de Estado em Ambientes Distribuídos (Stateless vs. Stateful)
    - Um dos maiores desafios de microsserviços e LLMs é que eles são nativamente stateless (não mantêm memória da interação anterior). Para resolver isso, o projeto utiliza o Redis como um armazenamento de estado externo e compartilhado.
    - Isso demonstra o conceito de Externalização de Estado, permitindo que, em um cenário de escalabilidade horizontal (várias instâncias do Backend), qualquer servidor possa atender a requisição do usuário recuperando o contexto da conversa no Redis, garantindo consistência na sessão.
- Otimização de Desempenho e Latência (Caching)
    - Modelos de IA Generativa são intensivos em CPU e inerentemente lentos (alta latência), o que representa um gargalo (bottleneck) clássico.
    - O projeto implementa uma estratégia de Cache Semântico distribuído via Redis. Diferente de caches tradicionais (chave-valor exata), o sistema utiliza vetores matemáticos para identificar requisições similares. Isso ilustra a aplicação de Estratégias de Replicação de Dados para reduzir a carga no componente de processamento principal (Ollama) e diminuir drasticamente o tempo de resposta (Throughput), um pilar de qualidade em sistemas distribuídos.
- Comunicação e Interoperabilidade
    - O sistema exemplifica a comunicação entre processos heterogêneos. O Backend atua como um Middleware, orquestrando chamadas síncronas (REST/HTTP) para o cliente e interagindo com módulos internos de IA. A utilização de padrões como JSON e Multipart/Form-Data garante a Interoperabilidade entre a interface JavaScript no navegador e a lógica Python no servidor, independentemente do sistema operacional ou hardware do usuário.

Equipe e Responsabilidades Técnicas

O projeto BrazucaTalks foi concebido e executado integralmente por um único membro, que assumiu uma abordagem multidisciplinar para cobrir todas as fases do desenvolvimento de software. As responsabilidades foram estruturadas de acordo com os papéis técnicos descritos abaixo:

Executante: Yuri Matheus Sousa dos Santos

1. Líder do Projeto e Product Owner:

- Responsabilidades: Definição do escopo, dos objetivos e dos requisitos funcionais do projeto. Realização da pesquisa de mercado para identificar as lacunas no ensino de idiomas. Elaboração da documentação da proposta, cronograma de entregas e apresentação final do produto.

2. Arquiteto de Software:

- Responsabilidades: Desenho da arquitetura de sistemas distribuídos, baseada no modelo Cliente-Servidor. Seleção do stack tecnológico principal (Python/FastAPI para o backend, React/Three.js para o frontend). Definição estratégica do Redis como camada de cache semântico para otimização de latência e como repositório de estado para a gestão da memória de conversação (Long-Term Memory), garantindo a escalabilidade da solução.

3. Desenvolvedor Backend:

- Responsabilidades: Implementação completa da API RESTful utilizando FastAPI. Desenvolvimento dos endpoints /transcribe e /ask, incluindo a lógica para processamento de áudio e texto. Integração com o Ollama para inferência do LLM e implementação da lógica de RAG (Retrieval-Augmented Generation) para contextualização cultural. Codificação das interações com o Redis para armazenamento e recuperação de dados de sessão e cache.

4. Desenvolvedor Frontend:

- Responsabilidades: Desenvolvimento da interface do usuário (UI) com React.js. Integração do avatar 3D na cena utilizando React Three Fiber (R3F) e Drei. Implementação do algoritmo de lip-sync em tempo real, utilizando a Web Audio API para analisar o áudio e modular os morph targets do modelo 3D. Gerenciamento do estado do cliente e da comunicação assíncrona com o backend.

5. Engenheiro de DevOps e Infraestrutura:

- Responsabilidades: Containerização completa da aplicação (Backend, Redis, Ollama) utilizando Docker e Docker Compose para garantir a portabilidade e a consistência do ambiente. Configuração e implantação da solução na nuvem (Amazon Web Services - AWS), incluindo o provisionamento da instância EC2, configuração de Security Groups para liberação de portas e associação de Elastic IP para acesso estável.