Desenvolvimento *de uma API Escalável para Gestão de Simulados e Avaliações Educacionais*

Gean Neis Lira

Rafael Vieira Coelho

Instituto Federal do Rio Grande do Sul (Câmpus Farroupilha), Farroupilha – RS – Brasil

(geanneislira@hotmail.com)

Resumo:
A crescente demanda por métodos de ensino mais dinâmicos e personalizados, especialmente diante dos desafios enfrentados por estudantes e professores na preparação para exames como o ENEM, evidencia a necessidade de soluções tecnológicas eficazes no ambiente educacional. Este trabalho apresenta o desenvolvimento de uma API RESTful escalável voltada à geração automatizada de simulados educacionais baseados em questões do Exame Nacional do Ensino Médio (ENEM). A proposta visa atender tanto estudantes, que buscam reforço e autoavaliação, quanto professores, que necessitam de agilidade na elaboração de avaliações. O sistema backend foi desenvolvido utilizando tecnologias como Python, Flask, MySQL, Docker e Jupyter Notebook, com foco na modularidade, reprodutibilidade e integração futura com interfaces web ou mobile. A obtenção das questões foi realizada por meio da API pública "enem.dev", garantindo dados estruturados e confiáveis. O sistema permite a geração de simulados personalizados, exportáveis em formato DOCX ou JSON, categorizados por áreas do conhecimento. A arquitetura proposta assegura a integridade dos dados e a escalabilidade da aplicação. Foram analisadas plataformas similares, como Brasil Escola, Beduka e Estuda.com, identificando-se lacunas no atendimento a professores. A solução proposta diferencia-se por contemplar ambos os públicos e por oferecer flexibilidade na criação dos simulados. Os testes foram realizados com ferramentas como Insomnia, validando a funcionalidade dos endpoints. Os resultados demonstram que a aplicação é eficaz e promissora para o uso educacional, com potencial de expansão para novos formatos e funcionalidades. Conclui-se que a tecnologia, quando bem aplicada, pode contribuir significativamente para a melhoria do ensino e da aprendizagem.

Palavras-chave: API RESTful. ENEM. Simulados. Backend. Educação.

Abstract:
This paper presents the development of a scalable RESTful API designed for the automated generation of educational mock exams based on questions from the Brazilian National High School Exam (ENEM). The proposed solution aims to support both students, seeking reinforcement and self-assessment, and teachers, who require agility in creating assessments. The backend system was developed using technologies such as Python, Flask, MySQL, Docker, and Jupyter Notebook, focusing on modularity, reproducibility, and future integration with web or mobile interfaces. The questions were sourced from the public API "enem.dev," ensuring structured and reliable data. The system enables the creation of customizable mock exams, exportable in DOCX or JSON formats, categorized by knowledge areas. The proposed architecture ensures data integrity and application scalability. Several existing platforms, such as Brasil Escola, Beduka, and Estuda.com, were analyzed, revealing a gap in addressing teachers' needs. The proposed solution stands out by serving both audiences and offering flexibility in exam creation. The endpoints were tested using tools like Insomnia, validating the expected functionality. Results show that the application is effective and promising for educational use, with potential for expansion into new formats and features. It is concluded that technology, when properly applied, can significantly enhance teaching and learning processes.

Keywords: RESTful API. ENEM. Mock Exams. Backend. Education.

1. INTRODUÇÃO

A transformação digital tem proporcionado uma reformulação significativa no campo educacional. A reestruturação das práticas de ensino e aprendizagem ampliaram o acesso ao conhecimento por meio de tecnologias que se tornam cada vez mais acessíveis e interativas. Com o avanço dos recursos tecnológicos, ambientes educacionais passaram a incorporar sistemas informatizados para promover metodologias mais dinâmicas e centradas no estudante (Valente, 2018). Observando este cenário, os simulados digitais se destacam como ferramentas pedagógicas eficazes no reforço de conteúdos e no diagnóstico de desempenho individual, sobretudo na preparação para avaliações de larga escala como o ENEM - Exame Nacional do Ensino Médio (MEC, 2024).

Atualmente, o ENEM é consolidado como o principal exame de ingresso ao ensino superior no Brasil. Ele avalia competências e habilidades multidisciplinares que são distribuídas em quatro áreas de conhecimento durante a execução do teste: Ciências Humanas e suas Tecnologias; Ciências da Natureza e suas Tecnologias; Linguagens, Códigos e suas Tecnologias; e Matemática e suas Tecnologias (BRASIL, 2023). A prática sistemática por meio de simulados baseados em provas anteriores permite não apenas a familiarização com o formato do exame, mas também a identificação de lacunas no processo de aprendizagem (Nunes & Gasparin, 2020).

Conforme Moran (2015), o uso de tecnologias digitais no ensino potencializa práticas pedagógicas mais dinâmicas, personalizadas e acessíveis, especialmente quando integradas a metodologias ativas de aprendizagem. Observando este contexto, a criação de simulados de forma automatizada surge como uma estratégia eficaz para preparar os estudantes para exames como o ENEM ou vestibulares. Desta forma, permite identificar lacunas, reforçar conteúdos específicos e melhorar o desempenho individual daqueles que utilizam esta prática.

Tendo em vista a importância dos simulados no reforço dos conteúdos e no diagnóstico de desempenho, este trabalho se propõe a desenvolver um sistema *backend* responsável por organizar, classificar e gerar provas personalizadas com base em questões de edições anteriores do Exame Nacional do Ensino Médio. A solução foi pensada para atender professores que buscam agilidade na elaboração de avaliações e estudantes que buscam meios mais eficazes de autoavaliação. O sistema organiza e classifica as questões de acordo com as quatro áreas de conhecimento estabelecidas pela prova. Foi optado por focar exclusivamente

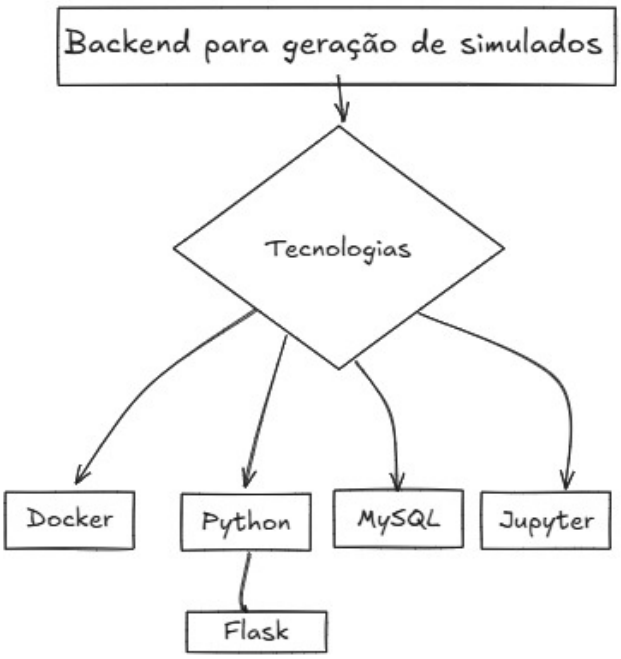
no desenvolvimento do *backend* buscando estabelecer uma base sólida, escalável e reutilizável que possa ser integrada a diversas interfaces e plataformas, como aplicações *web* ou *mobile.*

2. FUNDAMENTAÇÃO TEÓRICA

Para realizar o desenvolvimento de sistemas *backend* robustos, é necessário o domínio de conceitos relacionados a linguagens de programação, bancos de dados e arquitetura de *software*. Este capítulo busca fundamentar as escolhas técnicas adotadas no desenvolvimento do sistema bem como realizar um estudo sobre ferramentas já existentes no mercado.

A base teórica para a realização deste projeto busca abranger os conceitos e justificativas das tecnologias escolhidas para a construção do *backend*, embasando-se em obras e autores conhecidos na área da computação e desenvolvimento de software. Na Figura 1, são listadas em forma de diagrama quais as tecnologias escolhidas para atingir o resultado esperado ao final do desenvolvimento deste trabalho.

Figura 1 - Diagrama de tecnologias necessárias



Fonte - Autoria própria

Para o desenvolvimento deste projeto foi optado por seguir com as tecnologias listadas na Figura 1, sendo o Docker uma plataforma que utiliza contêiner para empacotar e executar aplicações em ambientes isolados e portáveis, promovendo maior agilidade e reprodutibilidade no desenvolvimento (Merkel, 2014), Python é uma linguagem de programação interpretada, de alto nível, MySQL é um sistema de gerenciamento de banco de dados relacional e o Jupyter Notebook é uma ferramenta interativa que integra código, visualizações e textos explicativos, sendo amplamente utilizada em contextos educacionais e científicos (Kluyver *et al*., 2016).

Com base na diagramação apresentada na Figura 1, os tópicos discutidos na fundamentação teórica tratam sobre as tecnologias e justificativas para a realização do trabalho proposto. No desenvolvimento serão apresentadas as tecnologias utilizadas na produção deste trabalho bem como suas principais características, a fim de justificar a utilização e embasar as escolhas.

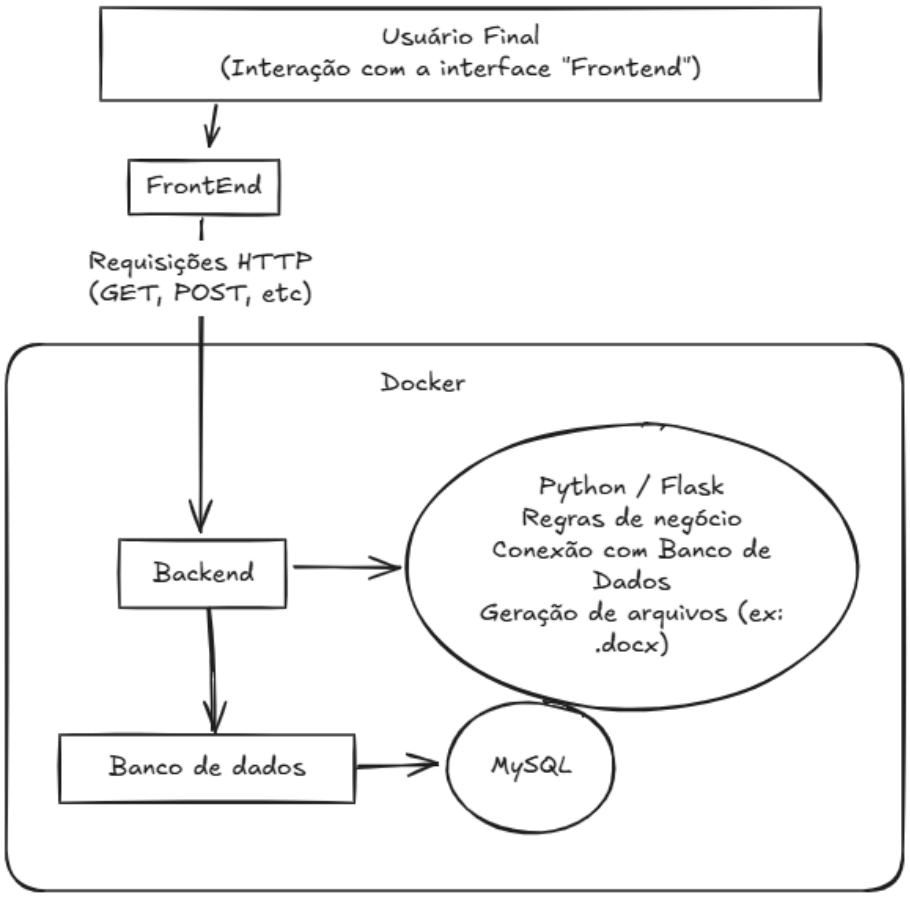
2.1 *Backend* e APIs RESTful

Um sistema geralmente é dividido de forma a segregar as funções que cada área do desenvolvimento busca integrar um projeto, comumente utilizam-se os termos *fontend* e *backend.* O termo *fontend* refere-se a tudo aquilo que é utilizável pelo usuário final da aplicação, ou seja, toda a interação entre o usuário e sistema são efetuadas nas telas desenvolvidas pelo time de *fontend*.

O *backend* é a camada de desenvolvimento de um sistema responsável pelo processamento das regras de negócio, comunicação com banco de dados e fornecimento de dados estruturados para a camada de *fontend*, resumidamente, uma camada transparente para o usuário final. Segundo Sommerville (2011), a separação entre as camadas de apresentação e lógica de negócios contribui para a modularidade e reusabilidade do *software*.

A Figura 2 apresenta uma representação simples e direta sobre como uma aplicação *web* é composta.

Figura 2 - Diagrama *Frontend* E *Backend*



Fonte - Autoria própria

Para o desenvolvimento deste projeto, a comunicação entre cliente e servidor é estabelecida por meio de uma API (*Application Programming Interface*) RESTful (*Representational State Transfer*), padrão amplamente adotado por sua leveza e compatibilidade com diversos clientes HTTP (FIELDING, 2000). Esta estrutura permite que os dados sejam consumidos por meio dos métodos HTTP padronizados, facilitando a utilização do *backend* para a consulta dos dados durante a implementação do *frontend.*

O protocolo HTTP (Hypertext Transfer Protocol) é amplamente adotado em arquiteturas do tipo cliente-servidor, no qual o cliente realiza requisições por meio de uma aplicação — como um navegador ou sistema *fontend* — e o servidor responde entregando os dados ou executando ações. Para cada tipo de operação esperada, deve-se especificar um método HTTP correspondente. Esses métodos padronizados indicam a intenção da requisição e orientam o servidor sobre a ação a ser tomada. No desenvolvimento de APIs, os métodos mais utilizados são:

- GET: Realiza a busca dos dados no servidor;
- POST: Envia novos dados;
- PUT: Atualiza um recurso por completo;
- PATCH: Atualiza parcialmente um recurso;
- DELETE: Remove um recurso do servidor.

2.2 Python

Python é uma linguagem de programação criada por Guido van Rossum no final da década de 1980 e lançada oficialmente em 1991, ela foi projetada com foco na legibilidade do código e na produtividade do desenvolvedor (Downey, 2015). Trata-se de uma linguagem interpretada, de alto nível e com tipagem dinâmica, o que significa que seu código é executado linha a linha por um interpretador, facilitando testes rápidos, depuração e prototipagem (Martins, 2021). Essas características citadas por Martins contribuem para o desenvolvimento ágil de aplicações, especialmente em contextos acadêmicos e educacionais.

A linguagem de programação foi escolhida por conter sintaxe clara, objetiva e ser de fácil aprendizado. Segundo Lutz (2013), a legibilidade também é um dos maiores benefícios da linguagem. Além disto, o uso desta linguagem está alinhado à afinidade do autor com a linguagem, com seu interesse em aprofundar os conhecimentos tendo em vista projetos futuros que necessitarão de um maior domínio da mesma. Segundo Downey (2015), Python é amplamente utilizado tanto na academia quanto na indústria, tornando-se uma excelente escolha para projetos educacionais e de prototipagem rápida.

Além dos pontos levantados anteriormente, a escolha de Python como linguagem de programação se deve à sua simplicidade sintática, vasta comunidade e abundância de bibliotecas voltadas ao desenvolvimento web, automação, aplicação para tratamento de dados e exportação dos mesmos. De acordo com Martins (2021), Python tem se consolidado como uma linguagem de entrada para desenvolvedores iniciantes, ao mesmo tempo em que é suficientemente poderosa para aplicações profissionais.

Python também é uma linguagem de programação que contempla grande integração com ferramentas como Jupyter Notebook e bibliotecas como Flask (vide seção 2.3), tornando a linguagem estratégica para projetos que visam natureza educacional ou científica.

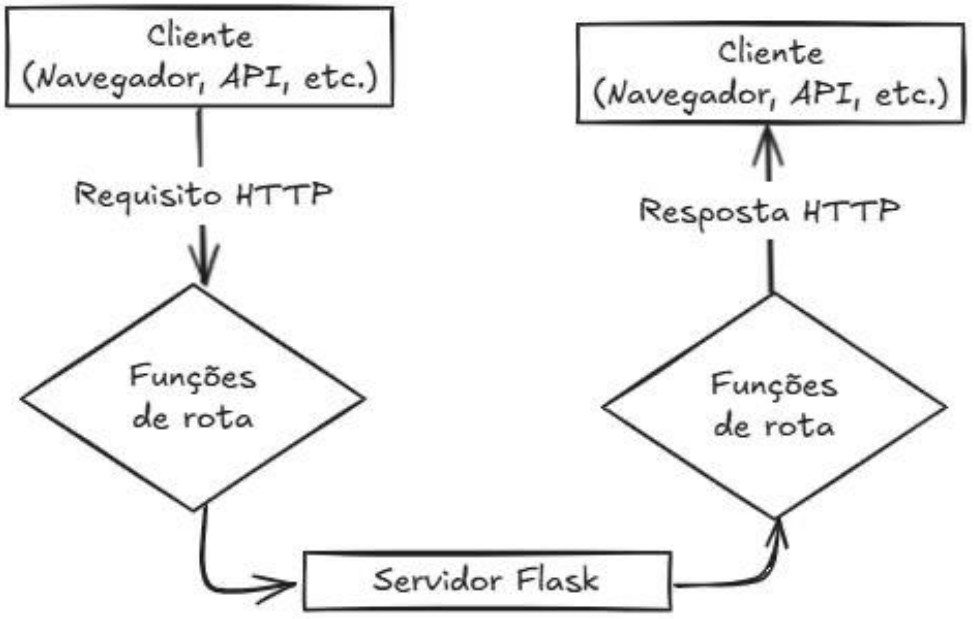
2.3 Flask

Flask é um micro *framework web* desenvolvido em Python. De acordo com Grinberg (2018), sua estrutura minimalista permite que o desenvolvedor tenha maior controle sobre os componentes do sistema, favorecendo a criação de APIs RESTful eficientes e modulares. A simplicidade e estrutura minimalista foi essencial para o desenvolvimento do sistema proposto, tornando a API simples e escalável para futuras integrações com os mais diversos formatos de *frontend*.

O *framework* oferece recursos essenciais para a criação de APIs de forma rápida e modular. Sendo minimalista, ele permite que o desenvolvedor tenha maior controle sobre as estruturas do projeto, diferente de *frameworks* mais complexos como Django. Silva e Souza (2022) destacam que o Flask é indicado para projetos de pequeno a médio porte e para soluções customizadas, tornando o Flask adequado para o escopo deste trabalho.

Observando o diagrama simplificado da Figura 3, é identificado a presença de dois agentes fundamentais em uma arquitetura cliente-servidor. O cliente, representado pelo usuário final da aplicação, realiza solicitações de consulta ou alteração de dados através de interações com a interface da aplicação. Essas solicitações são transmitidas ao servidor por meio de requisições HTTP, utilizando métodos específicos (como GET, POST, PUT, DELETE), que definem a natureza da operação a ser executada. No outro lado o servidor, onde as rotas previamente definidas têm a função de receber, interpretar e direcionar essas requisições para os respectivos manipuladores de lógica de negócio. Após o processamento da solicitação, o servidor retorna uma resposta ao cliente, contendo os dados solicitados ou informações sobre o sucesso ou falha da operação realizada, possibilitando assim a atualização dinâmica da interface apresentada ao usuário.

Figura 3 - Diagrama funcionamento Flask



Fonte - Autoria própria

2.4 MySQL

O MySQL é um sistema gerenciador de banco de dados relacionais *open source* amplamente utilizado. Conforme Elmasri e Navathe (2011), bancos relacionais são adequados para aplicações que exigem consistência e integridade dos dados, além de permitirem a modelagem estruturada de informações.

Para o desenvolvimento do projeto, optou-se pela utilização de uma estrutura de banco de dados relacional MySQL. A escolha desta tecnologia se deu por sua robustez, ampla documentação e compatibilidade com diferentes ambientes de desenvolvimento, bem como sua gratuidade de utilização, isto permitiu uma estruturação eficiente dos dados por meio de entidades e relacionamentos normalizados. Elmasri e Navathe (2011) argumentam que bancos relacionais são indicados quando a integridade dos dados e as operações transacionais são prioridades, como no caso da associação entre questões, alternativas e áreas do conhecimento.

Além dos pontos levantados acima, a escolha por MySQL se deve diante a sua disponibilidade em diversos ambientes de hospedagem, compatibilidade com ferramentas de

orquestração como Docker (explicado na seção 2.5) e sua ampla comunidade, garantindo disponibilidade de documentação e suporte em caso de necessidade.

2.5 Docker

Docker é uma plataforma que permite o encapsulamento de aplicações com todas as suas dependências e o agrupamento em contêineres. Segundo Merkel (2014), a utilização de contêiner facilita a portabilidade entre ambientes e a escalabilidade de sistemas, além de reduzir erros causados por divergências de configuração. Sua adoção no desenvolvimento do *backend* possibilitou a criação de um ambiente padronizado, replicável e facilmente distribuível.

A adoção de tecnologias de encapsulamento e conteinerização tem o intuito de disponibilizar um sistema leve, portátil e altamente escalável por meio da criação de novas instâncias caso haja necessidade. Merkel (2014) apresenta que os contêineres oferecem uma alternativa mais ágil e isolada em comparação com máquinas virtuais, reduzindo conflitos de ambiente e facilitando a escalabilidade horizontal de sistemas.

Por fim, a criação de um ambiente com máquinas virtuais torna o sistema e principalmente o banco de dados mais caro para alocação de recursos para execução da aplicação responsável pela gestão do sistema e para a execução do sistema operacional. A adoção de um ambiente virtualizado geralmente transforma o suporte mais oneroso e não garante a normalização dos sistemas envolvidos para a execução da aplicação.

2.6 Jupyter Notebook

Jupyter Notebook é uma ferramenta amplamente utilizada em ciência de dados e ambientes educacionais, principalmente por sua interatividade. Conforme Kluyver *et al*. (2016), sua capacidade de combinar código executável, visualizações e explicações textuais o torna ideal para análise exploratória de dados e documentação de processos.

No contexto deste trabalho, o Jupyter Notebook foi utilizado como ferramenta de apoio à categorização das questões e também para apoio no gerenciamento inicial do banco

de dados. A interatividade gerada pelo uso de um notebook contribuiu significativamente para a manipulação, visualização e inspeção dos dados de forma prática e interativa, permitindo ajustes rápidos durante a etapa de preparação das questões. Segundo Pérez e Granger (2015), os notebooks são ferramentas poderosas para documentação técnica, reprodutibilidade de experimentos e integração entre código, visualizações e textos descritivos.

Além disso, a adoção do Jupyter Notebook está alinhada às boas práticas em projetos de ciência de dados e desenvolvimento educacional, favorecendo a transparência e organização do código, do mesmo modo, quando planejado próximos passos para este desenvolvimento, a ferramenta oferece organização e colaboração entre os diferentes tipos de usuários, como desenvolvedores e educadores (clientes da ferramenta proposta). Essa característica foi particularmente útil na fase de testes e validação da base de questões, contribuindo para a confiabilidade e consistência do sistema desenvolvido.

2.7 Insomnia

Insomnia é uma plataforma de código aberto voltada a realização de requisições REST, GraphQL e SOAP, e é amplamente utilizada no mundo do desenvolvimento para testar, debugar e organizar APIs. Conforme documentado por Patel (2021), o Insomnia permite a construção de fluxos de requisições com variáveis, autenticação e corpo de mensagens personalizáveis, sendo uma alternativa eficiente para o monitoramento das respostas dos endpoints. Além disso, ela oferece uma interface intuitiva para o acompanhamento detalhado das respostas dos endpoints, incluindo status HTTP, cabeçalhos, tempo de resposta e conteúdo retornado.

Durante o desenvolvimento deste *backend*, a ferramenta foi essencial para validar o funcionamento dos *endpoints* criados, possibilitando principalmente testes rápidos sem a necessidade de construção de uma interface gráfica (*frontend*). Utilizando esta abordagem foi possível agilizar o processo de desenvolvimento, onde a ferramenta permitiu identificar as falhas e comportamentos esperados, além de validar os dados e retornos em tempo real.

Outro diferencial relevante é a integração com ferramentas de versionamento e colaboração, como GitHub, permitindo disponibilidade do código e versionamento da API

para futuros desenvolvimentos. A utilização do Insomnia, portanto, complementa o processo de desenvolvimento *backend*, permitindo uma abordagem mais interativa na validação da API, estando também em conformidade com práticas ágeis de desenvolvimento de software e princípios de desenvolvimento centrado no usuário (SOMMERVILLE, 2011).

2.8 ENEM e o Uso da Tecnologia na Educação

O ENEM (Exame Nacional do Ensino Médio) é uma das principais ferramentas avaliativas para o ensino brasileiro, sendo utilizado para ingresso nas universidades e também para aferimento da qualidade do ensino médio (INEP, 2022). Ele é composto por quatro áreas do conhecimento, com questões que buscam avaliar a capacidade de interpretação, raciocínio lógico, contextualização e aplicação prática do conhecimento.

Segundo Alves (2021), "o vestibular remete a um processo seletivo constituído por prova de redação e caderno de questões sobre assuntos estudados ao longo do ensino médio." Tradicionalmente, os vestibulares são utilizados como forma de classificação para ingresso nas universidades, determinando os candidatos mais aptos a ocuparem as vagas ofertadas.

O ENEM foi criado em 1998 e, inicialmente, visava apenas avaliar a qualidade do ensino médio brasileiro. Conforme Tancredi (2024), no seu lançamento, o exame tinha como principal objetivo realizar a avaliação da qualidade do ensino brasileiro e refletir sobre como é possível melhorá-lo. No ano de 2004, o ENEM passou a ser utilizado também como processo seletivo para o ensino superior, ampliando ainda mais a importância no cenário educacional nacional.

Atualmente, o ENEM é utilizado como critério de seleção para três importantes programas do governo federal: o Sistema de Seleção Unificada (SiSU), o Programa Universidade para Todos (ProUni) e o Fundo de Financiamento Estudantil (Fies) (MEC, 2024).

- **Sistema de Seleção Unificada (SiSU):** O Sistema de Seleção Unificada (Sisu) é o sistema gerenciado pelo Ministério da Educação (MEC) onde as instituições públicas ofertam vagas para os participantes do Exame Nacional do Ensino Médio (ENEM).
- **Programa Universidade para Todos (ProUni):** O Programa Universidade Para Todos (Prouni) oferece bolsas de estudo, tanto integrais ou parciais para cursos de graduação em instituições de ensino superior privadas.

- **Fundo de Financiamento Estudantil (Fies):** O Fundo de Financiamento Estudantil (Fies) é um programa ofertado pelo Ministério da Educação que destina-se a financiar a graduação no ensino superior em instituições privadas de ensino na forma da Lei 10.260/2001.

Com o avanço das Tecnologias, o uso de recursos digitais na educação tem se tornado uma prática comum e eficaz. Moran (2007) ressalta que a integração da tecnologia ao processo de ensino e aprendizagem pode transformar a forma como o conhecimento é construído, permitindo maior interatividade e personalização do ensino. Além disso, Kenski (2012) destaca que as tecnologias digitais ampliam o acesso à informação, promovem a autonomia dos estudantes e possibilitam novas estratégias de ensino.

2.9 Trabalhos Relacionados

2.9.1 Simulador Brasil Escola

O simulador do Brasil escola (Brasil Escola, 2024) permite que os estudantes escolham entre alguns simulados pré-planejados de temas específicos, como é visto na Figura A.1 (Anexo A).

Conforme a Figura A.1 (Anexo A), ainda é observado a possibilidade de simular o primeiro e segundo dia do ENEM ou a prova completa. Além disso o estudante também tem acesso à possibilidade de criar seu próprio simulado, porém neste ponto ele é limitado a criação de provas que contenham única e exclusivamente 180 questões conforme a Figura A.2 (Anexo A).

2.9.2 Simulador Beduka

O simulador Beduka (Beduka, 2024) tem o comportamento semelhante ao simulador do Brasil Escola, com a diferença de apresentar as opções a seguir:

- Enem (Por Caderno): Nessa modalidade, você escolhe o caderno com 45 questões que deseja para fazer a prova. Nesse tipo de simulado estão incluídas as seguintes áreas do conhecimento: (A) Ciências da Natureza e suas Tecnologias; (B) Ciências Humanas e suas Tecnologias; (C) ….

- Enem (Por dia): Nessa modalidade, você escolhe fazer as questões como elas são apresentadas no primeiro ou no segundo dia do ENEM. Neste caso, são 90 questões para cada dia.
- Enem (Simulado modular): Aqui você pode escolher quantas questões fazer em cada área do conhecimento.

Uma ressalva é que o sistema Beduka necessita o login para fazer provas gratuitas, ou seja, não existe a possibilidade de utilizar a ferramenta sem a criação de um acesso pessoal, diferente do simulador Brasil Escola.

2.9.3 Aplicativo ENEM – Provas e Simulados

Aplicativo mobile que apresenta uma plataforma de aprendizado focada em simulados do ENEM, o software está disponível na *App Store* para dispositivos iOS e na *Play Store* para dispositivos Android, e não apresenta a obrigatoriedade de fazer login para começar a utilizar suas funcionalidades, a página inicial listada na Figura A.3 (Anexo A) apresenta a possibilidade de gerar um simulado personalizado ou a possibilidade de estudar por temas.

O aplicativo conta com propagandas no cabeçalho e um pequeno calendário para acompanhar os estudos, bem como uma página para relacionar o histórico de questões respondidas e as estatísticas do estudante.

Além disso, a ferramenta possibilita a criação de simulados personalizados com base nas 4 categorias do ENEM, limitando o usuário a 100 questões, independente de quantas categorias escolher conforme demonstrado na Figura A.4 (Anexo A).

O aplicativo também conta com a possibilidade de estudo por temas específicos, conforme apresentado na Figura A.5 (Anexo A). Nesta tela existe a possibilidade de criar um simulado com 100 questões da área específica ou escolher os cadernos de provas do ENEM anteriores.

Em suma, o aplicativo é bem completo para quem busca estudar com base nas questões do ENEM, porém ele não apresenta a possibilidade de criar provas com base nas questões de vestibulares terceiros.

2.9.4 Estuda.com

Aplicativo mobile que apresenta uma plataforma de aprendizado focada em simulados. O app oferece a opção de treinar com questões por disciplina das mais diversas áreas, como por exemplo, artes ou sociologia. Além disso, ele permite a criação de simulados de até 40 questões, tudo isso apresentando já na sua tela inicial conforme a Figura A.6 (Anexo A).

O grande diferencial deste aplicativo é possibilitar a escolha da origem das perguntas utilizadas no simulado, como podemos observar na Figura A.7 (Anexo A) listada ao final do artigo, o aplicativo permite que seja escolhido questões entre diversas instituições de ensino superior, facilitando assim o estudo para vestibulares específicos.

2.9.5 Considerações

É possível concluir que existem plataformas competentes para criação de simulados para o ENEM e vestibulares. No entanto, todas focam apenas no aluno, sem atender à necessidade de professores que buscam agilidade na criação de provas. O sistema proposto busca preencher essa lacuna, atendendo a ambos os públicos.

Com base na amostragem apresentada acima é conclusivo que existem plataformas competentes para criação de simulados para o ENEM ou para outros vestibulares, no entanto todas elas apresentam particularidades e todas tem o enfoque apenas no aluno, não buscando atender a necessidade de professores que querem agilidade no seu dia a dia para a criação destes simulados.

É importante destacar que o sistema a ser desenvolvido neste trabalho busca criar um centralizador que possa ser utilizado tanto por alunos que buscam aumentar seus conhecimentos como por professores que buscam melhorar a criação e aplicação de simulados na sala de aula.

A análise de outras aplicações permite uma reflexão sobre a arquitetura e o público-alvo dos aplicativos supracitados, como eles atendem às diferentes demandas e sua estrutura. É necessário rastrear formas de inovar e melhorar arquiteturas/aplicações já

existentes, buscando atender às expectativas do público-alvo que respondeu o questionário da melhor forma possível.

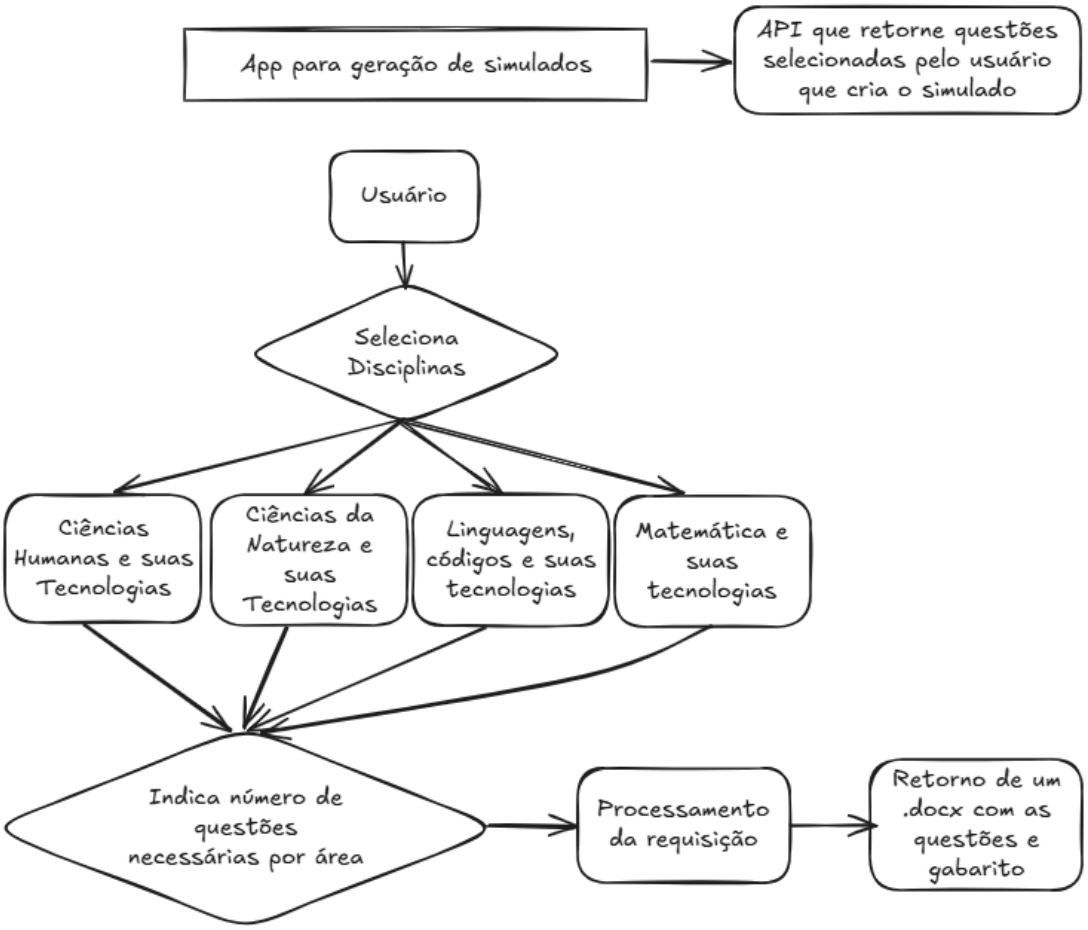
3. DESENVOLVIMENTO

O presente trabalho se caracteriza como uma pesquisa aplicada com enfoque no desenvolvimento de um sistema *backend* que destina-se à geração de simulados baseados nas questões de edições anteriores do ENEM (Exame Nacional do Ensino Médio). A metodologia escolhida é de natureza tecnológica e exploratória, uma vez que é necessário investigar ferramentas existentes, processos voltados à construção de simulados e influência da solução no ambiente em que será aplicada. O desenvolvimento foi dividido nas seguintes etapas:

1. Levantamento de requisitos;
2. Seleção de tecnologias;
3. Tratamento dos dados utilizados:
    a. Classificação das questões;
4. Modelagem do banco de dados;
5. Desenvolvimento da API;
6. Contêinerização;
7. Validação funcional.

Durante o levantamento de requisitos, foi feito um esboço do funcionamento esperado pela API a ser desenvolvida. Desta forma foi possível definir quais as tecnologias necessárias para a codificação e arquitetura do sistema (veja na Figura 4).

Figura 4 - Desenho primário da API



Fonte - Autoria própria

Com base no esboço acima foi possível definir quais seriam as tecnologias utilizadas para atender plenamente o que é esperado da API, que resumidamente é retornar um arquivo de texto no formato .DOCX (Documento de Texto do Microsoft Word) contemplando a quantidade de perguntas solicitadas, separando-as por áreas de conhecimento.

Após a definição das tecnologias e ferramentas essenciais para o desenvolvimento do sistema *backend*, tornou-se necessário estabelecer uma estratégia eficiente para a obtenção das questões aplicadas nas edições anteriores do ENEM. Nessa etapa, foi realizada uma análise das fontes públicas disponíveis, com o objetivo de extrair os dados de forma estruturada, confiável e aderente às necessidades do projeto.

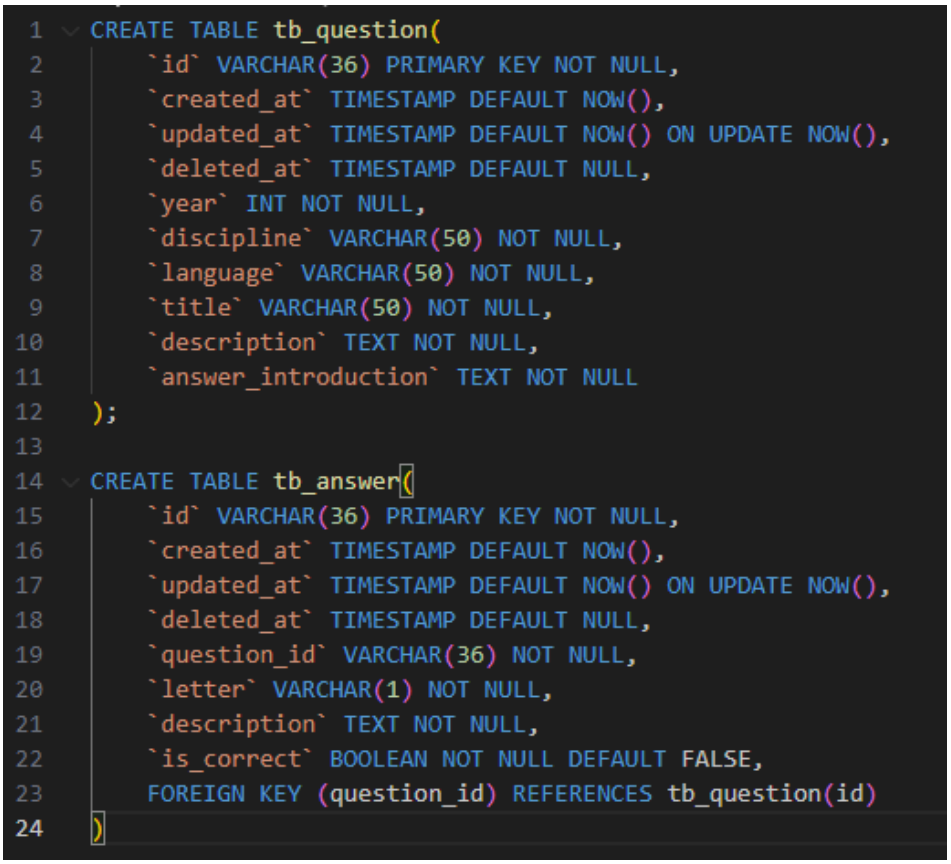
Inicialmente, tentou-se utilizar os arquivos das provas disponibilizadas pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP), disponibilizadas no site e acessíveis em formato PDF. No entanto, a extração das informações utilizando scripts em Python resultou em dados com ruído excessivo, dificultando a separação dos elementos relevantes, como enunciados, alternativas e gabaritos. A conversão dos conteúdos para o formato JSON, necessária para a organização dos dados no banco, gerava inconsistências e exigia um processo de limpeza extenso e pouco eficiente.

Diante dessas limitações, optou-se por buscar fontes alternativas que oferecessem acesso a dados já organizados e tratados, facilitando a integração com o banco de dados relacional do sistema. Essa decisão foi motivada pela necessidade de garantir qualidade, integridade e consistência às informações, reduzindo o tempo gasto com processamento e focando nos aspectos técnicos e funcionais da aplicação.

Durante a pesquisa por fontes alternativas que atendessem à necessidade de obter um conjunto confiável e estruturado de questões do ENEM, foi identificado o projeto “enem.dev”, uma API pública de código aberto voltada à consulta de provas e questões do exame. A solução foi desenvolvida por um usuário da comunidade de programadores e disponibilizado no GitHub (Galete, 2024), que compartilhou, em sua documentação, as dificuldades enfrentadas tanto na localização de fontes consistentes quanto no tratamento dos dados brutos disponibilizados oficialmente. Essa API se destacou por oferecer um acesso facilitado às informações já organizadas, eliminando a necessidade de tratamento manual dos dados e atendendo diretamente aos requisitos do projeto proposto, no que se refere à automação e confiabilidade na coleta de dados educacionais.

Para estruturar o armazenamento das informações referentes às questões e alternativas, foi desenvolvido um banco de dados relacional utilizando MySQL. A modelagem (Figura 5) foi pensada para garantir integridade referencial, facilidade de manutenção e escalabilidade. Duas tabelas principais foram criadas: **tb_question**, responsável por armazenar os dados das questões; e **tb_answer**, destinada ao armazenamento das alternativas associadas.

Figura 5 - Criação das tabelas necessárias



```sql
CREATE TABLE tb_question(
    `id` VARCHAR(36) PRIMARY KEY NOT NULL,
    `created_at` TIMESTAMP DEFAULT NOW(),
    `updated_at` TIMESTAMP DEFAULT NOW() ON UPDATE NOW(),
    `deleted_at` TIMESTAMP DEFAULT NULL,
    `year` INT NOT NULL,
    `discipline` VARCHAR(50) NOT NULL,
    `language` VARCHAR(50) NOT NULL,
    `title` VARCHAR(50) NOT NULL,
    `description` TEXT NOT NULL,
    `answer_introduction` TEXT NOT NULL
);

CREATE TABLE tb_answer(
    `id` VARCHAR(36) PRIMARY KEY NOT NULL,
    `created_at` TIMESTAMP DEFAULT NOW(),
    `updated_at` TIMESTAMP DEFAULT NOW() ON UPDATE NOW(),
    `deleted_at` TIMESTAMP DEFAULT NULL,
    `question_id` VARCHAR(36) NOT NULL,
    `letter` VARCHAR(1) NOT NULL,
    `description` TEXT NOT NULL,
    `is_correct` BOOLEAN NOT NULL DEFAULT FALSE,
    FOREIGN KEY (question_id) REFERENCES tb_question(id)
)
```

Fonte - Autoria própria

A estrutura formulada permite o versionamento, exclusão dos registros e estabelece um relacionamento entre as questões e respostas, garantindo que cada alternativa esteja sempre vinculada a uma pergunta específica. A escolha por utilizar identificadores únicos (UUIDs) como chave primária visa facilitar a interoperabilidade com outros sistemas e serviços futuros.

Após definir e criar a estrutura do banco de dados, foi necessário transformar os dados obtidos por meio da API pública do projeto “enem.dev” para que estivessem em conformidade com o modelo relacional criado . Para isso, foi desenvolvido um código em Python que realiza o processamento das informações de cada questão e suas alternativas (veja a Figura 6). O código percorre os itens retornados pela API, faz a validação se a questão possui arquivos vinculados — caso possua, ela é ignorada pois atualmente o projeto prioriza apenas perguntas textuais. Em seguida é gerado um identificador único (UUID) para cada

questão e os campos relevantes são extraídos e estruturados em um dicionário Python compatível com os atributos da tabela **tb_question**.

Figura 6 - Transformação dos dados adquiridos pela API

```python
for question in questions:
    if len(question['files']) > 0:
        continue

    question_id = str(uuid.uuid4())

    transformed_question = {
        'id': question_id,
        'year': question['year'],
        'discipline': question['discipline'],
        'language': question['language'] or '',
        'title': question['title'],
        'description': question['context'] or '',
        'answer_introduction': question['alternativesIntroduction'] or ''
    }

    temp_answers = []
    for answer in question['alternatives']:
        if answer['file']:
            break

        transformed_answer = {
            'id': str(uuid.uuid4()),
            'question_id': question_id,
            'letter': answer['letter'],
            'description': answer['text'] or '',
            'is_correct': answer['isCorrect']
        }

        temp_answers.append(transformed_answer)

    if len(temp_answers) > 0:
        transformed_questions.append(transformed_question)
        transformed_answers.extend(temp_answers)
```

Fonte - Autoria própria

Para as alternativas das questões, o tratamento ocorre da mesma forma, respeitando sempre o vínculo com a pergunta correspondente e desconsiderando todas que contenham

arquivos em anexo. Cada alternativa é convertida em um dicionário estruturado, também contendo um UUID exclusivo. Caso a questão possua alternativas válidas, ela e suas respostas são adicionadas às listas **transformed_questions** e **transformed_answers**, listas que posteriormente serão utilizadas para a inserção dos dados no banco MySQL.

Após a etapa de transformação, foi garantido a integridade e a consistência dos dados antes de seu armazenamento, evitando assim a inserção de dados incompletos ou em formato inadequado à estrutura desenvolvida para o banco. Além disso, a automatização do processo reduz drasticamente o tempo e esforço para normalização dos dados, contribuindo para a escalabilidade e confiabilidade do sistema.

Com a garantia que os dados estão no formato esperado, iniciou-se o processo de população do banco de dados com as questões transformadas. Para isso foi implementado um script em Python que estabelece a conexão com o MySQL, desabilita o *autocomit* para garantir atomicidade das operações e executa instruções de inserção conforme observado na Figura 7.

Figura 7 - Comandos insert no banco de dados

```
connection.autocommit = False


cursor = connection.cursor()


insert_question_statement = """
    INSERT INTO tb_question(id, year, discipline, language, title,
    description, answer_introduction)
        VALUES (%s, %s, %s, %s, %s, %s, %s);
"""


insert_answer_statement = """
    INSERT INTO tb_answer(id, question_id, letter, description,
    is_correct)
        VALUES (%s, %s, %s, %s, %s);
"""
```

Fonte - Autoria própria

Após realizar a conexão com o banco é repassado o comando se inserção “INSERT” para que os dados sejam populados no MySQL, além disto foi necessário percorrer as listas criadas na etapa de pré processamento para executar as inserções dos dados de forma transacional.

Figura 8 - Execução de forma transacional

```
# with tqdm(total=len(transformed_questions)+len(transformed_answers)) as progress_bar:
try:
    for question in transformed_questions:
        values = tuple(question.values())

        cursor.execute(insert_question_statement, values)
        progress_bar.update(1)

    for answer in transformed_answers:
        values = tuple(answer.values())

        cursor.execute(insert_answer_statement, values)
        # progress_bar.update(1)

    connection.commit()
except Exception as e:
    print(e)

    connection.rollback()
```

Fonte - Autoria própria

Caso ocorra qualquer exceção, o bloco **except** aciona um *rollback* para preservar a consistência do banco. A prática de utilizar *prepared statements* (instruções parametrizadas) mitiga o risco de injeção de SQL e melhora a legibilidade do código.

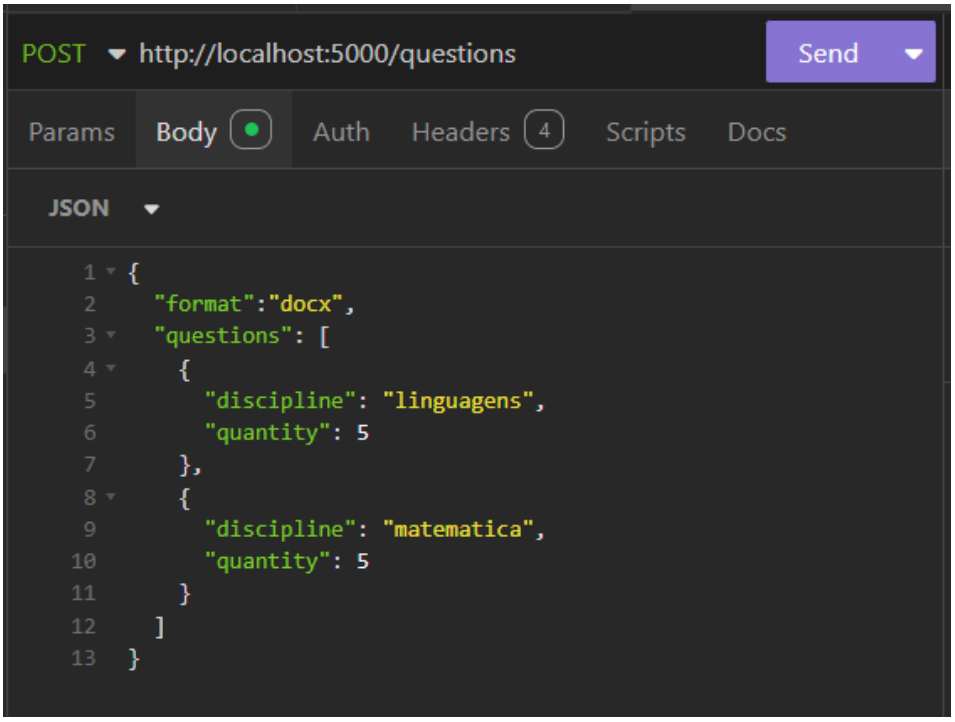
Ao concluir este processo de inserção de dados, a conexão é finalizada e os dados estão posicionados no banco conforme esperado para dar continuidade ao desenvolvimento.

Com os dados populados da forma esperada, deu-se início à criação da aplicação que faz a consulta dos requisitos dos usuário e gera o simulado conforme solicitado. Para isso foi desenvolvido uma aplicação utilizando Flask que resulta em uma API.

Para o funcionamento da API é necessário que o *frontend,* neste caso simulado utilizando a ferramenta Insomnia, envie uma requisição POST contendo um JSON com as disciplinas e também o número de questões correspondente a cada área de conhecimento.

Com as informações repassadas pelo método POST conforme a Figura 9, o sistema faz o tratamento das informações e realiza a consulta ao banco de dados para que seja possível retornar o arquivo gerado contendo o número exato de questões e disciplinas solicitado pelo usuário.

Figura 9 - Exemplo de POST utilizando JSON



Fonte - Autoria própria

Ao utilizar o método POST é enviado ao sistema um JSON contendo as disciplinas e quantidade de questões correspondente a cada área do conhecimento. Desta forma, durante o desenvolvimento do sistema *Frontend* basta que seja enviado estas informações para que o *backend* desenvolvido seja capaz de gerar os simulados.

Quando o *backend* recebe os dados inseridos por meio do método POST ele realiza a consulta ao banco de dados por meio de um "**for**" para que o número de questões solicitadas seja extraído do banco de dados para a geração do simulado. Na figura 10 é possível observar o método criado para realização da consulta ao MySQL.

Figura 10 - Código de tratamento ao método POST

```python
@bp.post("")
def get_questions():
    request_body = request.get_json()

    # TODO:
    format = request_body.get('format')

    questions = {}
    for item in request_body['questions']:
        dicipline = item['discipline']
        current_quantity = questions.get(dicipline, 0)
        questions[dicipline] = current_quantity + item['quantity']
    # ---

    final_questions = []

    for discipline, quantity in questions.items():
        # TODO:
        question_query = """
            SELECT tq.id,
                   tq.title,
                   tq.description,
                   tq.answer_introduction,
                   tq.discipline,
                   tq.language,
                   tq.year
              FROM tb_question tq
             WHERE tq.deleted_at IS NULL
               AND tq.discipline = %s
             ORDER BY RAND()
             LIMIT %s;
        """
        # ---
```

Fonte - Autoria própria

Após executar o método da figura 10 para a consulta das questões, é necessário que sejam também selecionadas as respostas correspondentes a cada questão. Para isso, também é utilizado um método "**for**" que consulta e seleciona as questões já com a resposta sinalizada. Na figura 11 é listado o bloco do código que realiza a consulta das perguntas com as respostas.

Figura 11 - Código de tratamento ao método POST

```
    questions = fetch_all(question_query, (discipline, quantity, ))

    for question in questions:
        answers_query = """
            SELECT ta.id,
                   ta.letter,
                   ta.description,
                   ta.is_correct
              FROM tb_answer ta
             WHERE ta.deleted_at IS NULL
               AND ta.question_id = %s;
             ORDER BY ta.letter ASC;
        """

        question_id = question['id']

        answers = fetch_all(answers_query, (question_id,))

        question['answers'] = answers

        final_questions.append(question)

# TODO: Adicionar constantes
if format == 'json':
    return final_questions
# --

# TODO:
if format== 'docx':
    document = generate_docx(final_questions)
    docx_stream = io.BytesIO()

    document.save(docx_stream)
    docx_stream.seek(0)
```

Fonte - Autoria própria

Após realizar a consulta aos dados necessários para preencher a requisição do usuário que foi feita utilizando o POST para a API, o sistema oferece duas possibilidades de retorno conforme o formato especificado: JSON ou DOCX.

- JSON

Caso o formato especificado na requisição seja JSON, o sistema retorna os dados estruturados diretamente como uma lista de objetos JSON. Cada objeto representa uma questão e inclui, além de seus metadados, a lista de alternativas associadas. Esse formato é particularmente útil para aplicações que pretendem consumir os dados de forma dinâmica, como interfaces web ou mobile (*frontend*), permitindo renderização e manipulação direta dos conteúdos.

- DOCX

Quando o formato requisitado é DOCX, o *backend* utiliza o módulo python-DOCX para montar um documento estruturado contendo as questões e alternativas. Esse documento é armazenado em um objeto do tipo BytesIO, que simula um arquivo binário em memória. O conteúdo do arquivo é então enviado ao cliente por meio de um objeto Response, com os cabeçalhos apropriados que indicam se tratar de um arquivo Word.

Figura 12 - Código de tratamento ao método POST

```
    return Response(
        docx_stream.read(),
        content_type='application/vnd.openxmlformats-officedocument.wordprocessingml.document',
        headers={'Content-Disposition': 'attachment; filename=my_document.docx'}
    )
# --

return '', 204
```

Fonte - Autoria própria

A implementação mostrada na figura 12 garante que, ao consumir a API, o usuário possa baixar automaticamente um arquivo pronto para uso em avaliações impressas ou em ambiente offline, atendendo à necessidade de professores que desejam agilidade na geração de simulados.

Além disto, para gerar o documento com formato Microsoft Word é executado o trecho de código demonstrado na figura 13, que especifica como o arquivo deve ser montado e como as questões serão dispostas no documento.

Figura 13 - Código de tratamento ao método POST

```python
# TODO:
def generate_docx(questions):
    document = Document()

    populate_docx(document, questions, False)
    populate_docx(document, questions, True)

    return document

def populate_docx(document, questions, is_template):
    if not is_template:
        document.add_paragraph('Aluno: _____')
        document.add_paragraph('Professor: _____')
        document.add_paragraph('Disciplina: _____')
        document.add_paragraph('Data: ____/____/____')
        document.add_paragraph('')
    else:
        document.add_page_break()

    for question in questions:
        document.add_heading(question['title'], level=3)

        if question['description']: document.add_paragraph(question['description'])

        document.add_paragraph(question['answer_introduction'])

        for answer in question['answers']:
            paragraph = document.add_paragraph(answer['letter'] + ": " + answer['description'])

            if is_template and answer['is_correct']:
                paragraph.add_run(' (correta)').bold = True

        document.add_paragraph('')
# --
```

Fonte - Autoria própria

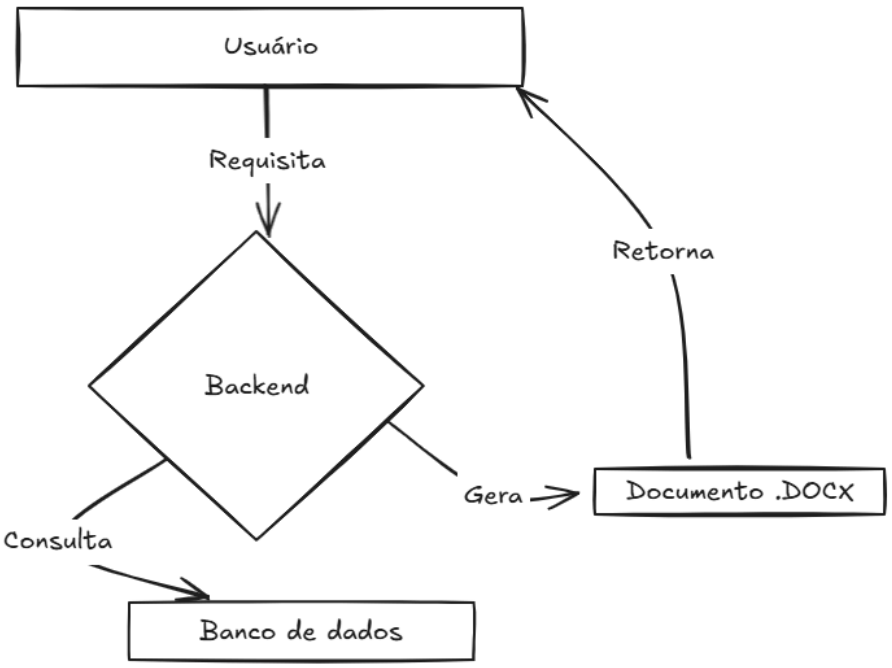
Após a execução completa da API, o usuário recebe como resposta um documento contendo a quantidade de disciplinas e questões previamente especificadas na requisição. Ao final do arquivo, é disponibilizado também o gabarito da prova, possibilitando tanto a aplicação quanto a correção imediata do simulado. Durante o desenvolvimento do sistema,

chegou-se a considerar a implementação da exportação em formato PDF. Contudo, optou-se pelo uso exclusivo do formato DOCX por oferecer maior flexibilidade e facilidade de edição, especialmente para professores que desejam personalizar o conteúdo antes da aplicação.

Adicionalmente, foi implementado o suporte ao formato JSON, visando atender à etapa futura de desenvolvimento de uma interface *frontend* para o sistema. Esse formato permite o retorno das informações de maneira estruturada e digital, dispensando a necessidade de arquivos físicos e viabilizando a integração com aplicações web e mobile.

Ao final do desenvolvimento foi obtida a estrutura final do sistema de acordo com a Figura 14.

Figura 14 - Arquitetura final do sistema



Fonte - Autoria própria

Desta forma é possível entender resumidamente como o sistema foi projetado para integração com sistemas *fontend* por meio de uma API de uso simplificado.

4. RESULTADOS

Ao finalizar o desenvolvimento do sistema proposto, foi obtido uma API RESTful funcional, que tem a capacidade de gerar simulados a partir de questões previamente classificadas de acordo com as áreas de conhecimento do ENEM.

Foi desenvolvido um sistema estruturado que tem capacidade de assegurar a integridade dos dados, a escalabilidade e a possibilidade de futuras integrações com interfaces *frontend*, sejam elas web ou *mobile*.

O *backend* foi desenvolvido em três camadas principais:

- Lógica de negócios - Regras de geração e consulta de questões anteriores;
- Persistência - Utilização de um banco de dados relacional MySQL.
- Controle - Criação de rotas para a API (Vide abaixo);

A rota GET /disciplines foi implementada com o uso de Blueprints do *microframework* Flask, estratégia que favorece a modularização e organização do código-fonte. A rota tem como objetivo fornecer uma listagem das disciplinas disponíveis no banco de dados, funcionando como ponto de consulta para funcionalidades que dependem da categorização das questões.

Figura 15 - Declaração da rota Get

```python
from flask import Blueprint

from app.db import fetch_all

bp = Blueprint('disciplines', __name__, url_prefix='/disciplines')

@bp.get("")
def get_disciplines():
    query = """
        SELECT DISTINCT discipline
          FROM tb_question tq
         WHERE tq.deleted_at IS NULL;
    """

    return fetch_all(query)
```

Fonte - Autoria própria

Esse *endpoint* é essencial para alimentar interfaces do usuário para que ele saiba quais disciplinas estão disponíveis para a filtragem, categorização ou geração de simulados personalizados, contribuindo para a usabilidade do sistema e para a organização pedagógica dos conteúdos.

A rota POST /questions, figura 16, é responsável por gerar e retornar um conjunto de questões baseado em critérios recebidos no corpo da requisição. Essa rota permite ao usuário solicitar questões específicas por disciplina e quantidade, com a opção de receber os dados no formato JSON ou em um documento .DOCX.

Figura 16 - Declaração da rota Post

```python
from flask import Blueprint, Response, request

from app.db import fetch_all

bp = Blueprint('questions', __name__, url_prefix='/questions')

@bp.post("")
def get_questions():
    request_body = request.get_json()

    # TODO:
    format = request_body.get('format')

    questions = {}
    for item in request_body['questions']:
        dicipline = item['discipline']
        current_quantity = questions.get(dicipline, 0)
        questions[dicipline] = current_quantity + item['quantity']
    # ---
```

Fonte - Autoria própria

O endpoint recebe um objeto JSON contendo uma lista de disciplinas e suas respectivas quantidades de questões desejadas, bem como o formato de saída (*format*), que pode ser "JSON" ou "DOCX". Basicamente esse endpoint é possível pois a primeira rota listada traz quais as disciplinas são necessárias para a realização da rota POST.

O desenvolvimento se deu utilizando o *microframework* Flask que possibilitou a criação de *endpoints* leves e responsivos. As funcionalidades básicas da API são: Criação de novos simulados e filtragem de perguntas por área de conhecimento. Os *endpoints* foram testados manualmente utilizando ferramentas como Postman e Insomnia, (já citadas na fundamentação teórica), garantindo o funcionamento esperado para as requisições efetuadas à API.

A utilização do Docker para conteinerização permite a replicação do ambiente e facilita a distribuição do sistema para outros ambientes, facilitando os próximos passos do projeto.

Como principal resultado do desenvolvimento do trabalho foi obtido um sistema que tem capacidade de gerar simulados de forma automática com base em requisitos apontados pelo usuário. O *backend* está devidamente estruturado para atender aplicações web ou mobile para apoiar tanto alunos que buscam aperfeiçoamento pessoal como professores no planejamento de atividades avaliativas ou preparatórias para vestibulares e para o próprio ENEM.

5. CONSIDERAÇÕES FINAIS

A escolha por uma abordagem prática e apoiada em ferramentas de código aberto que são amplamente documentadas visa garantir a acessibilidade, reprodutibilidade e sustentabilidade da solução proposta. A metodologia adotada permite que o sistema seja expandido e integrado a diferentes plataformas sem maiores complicações, além disso ela possibilita a replicação do processo em contextos similares.

O desenvolvimento do sistema *backend* que foi proposto neste trabalho permitiu explorar de maneira técnica a integração entre conceitos do curso de Análise e Desenvolvimento de sistemas e soluções tecnológicas amplamente utilizadas nas mais diversas áreas da TI. Baseando-se nas tecnologias selecionadas para o desenvolvimento deste trabalho, foi possível a construção de uma infraestrutura funcional que será apta a fornecer suporte àquilo que foi proposto no início deste projeto, que é um sistema de apoio aos estudos utilizando questões baseadas no ENEM.

Tecnicamente o projeto demonstrou aderência aos princípios da engenharia de software que são a modularidade, a escalabilidade e a reprodutibilidade. A utilização de uma API RESTful bem definida e a organização concisa da base de dados assegura que o *backend* esteja preparado para ser integrado à aplicações *frontend*.

Observando o sistema desenvolvido do ponto de vista educacional, é possível constatar que a aplicação é uma ferramenta promissora no auxílio de estudantes e professores na preparação para exames de larga escala, como o ENEM, proporcionando acesso facilitado a simulados customizáveis categorizados por áreas do conhecimento.

A ausência de uma interface gráfica não compromete a viabilidade do sistema, mas indica um caminho claro para aprimoramentos futuros. A continuidade do desenvolvimento tende a ampliar de forma positiva o impacto da solução proposta.

Em resumo, o trabalho evidenciou que o uso apropriado de tecnologias pode contribuir de forma significativa no campo da educação, especialmente quando as soluções são focadas na resolução de problemas reais, como o acesso a materiais de estudo e práticas avaliativas personalizadas.

Ao longo do projeto, foi possível aplicar conhecimentos em Python, Flask, MySQL, Docker e Jupyter Notebook. A estrutura modular e a documentação clara permitem que o sistema seja facilmente mantido, expandido ou adaptado para novas demandas.

Para trabalhos futuros é destacado a necessidade de criação de uma interface gráfica, seja ela *web* ou *mobile, para* facilitar a utilização do sistema por usuários não técnicos. Também pode ser realizado um aprimoramento no processo de categorização automática das questões, possibilitando a utilização de um *dataset* maior ou que contemple questões com dados que vão além de textos, como técnicas de aprendizado de máquina.

6. REFERÊNCIAS

ALVES, J. Vestibular e ENEM: diferenças e semelhanças. São Paulo: Revista Educação em Foco, 2021.

ALVES, Rafael. O que é vestibular? Brasil Escola, 2021. Disponível em: https://brasilescola.uol.com.br/enem/o-que-e-vestibular.htm. Acesso em: 6 maio 2025.

BEDUKA. Simulado Beduka. 2024. Disponível em: https://beduka.com/blog/enem/simulado-enem-online. Acesso em: 6 maio 2025.

BRASIL. Ministério da Educação. Exame Nacional do Ensino Médio (ENEM). 2023. Disponível em: https://www.gov.br/mec/pt-br. Acesso em: 10 maio 2025.

BRASIL ESCOLA. Simulado Brasil Escola. 2024. Disponível em: https://brasilescola.uol.com.br/simulado. Acesso em: 6 maio 2025.

DOWNEY, A. *Think Python: how to think like a computer scientist*. 2. ed. Sebastopol: O'Reilly Media, 2015.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. 6. ed. São Paulo: Pearson, 2011.

ENEM - PROVAS E SIMULADOS. Aplicativo educacional. 2024. Disponível em: https://play.google.com/store/apps/details?id=com.aplicativo.enem. Acesso em: 6 maio 2025.

ESTUDA.COM. Simulados por disciplinas e instituições. 2024. Disponível em: https://www.estuda.com. Acesso em: 6 maio 2025.

FERREIRA, A. B. O uso de simulados como estratégia de ensino. *Revista Brasileira de Educação*, v. 25, p. 1–12, 2020.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. 2000. Tese (Doutorado) – University of California, Irvine.

GRINBERG, M. *Flask web development: developing web applications with Python*. 2. ed. Sebastopol: O'Reilly Media, 2018.

GUTIERREZ, C. T. S.; FIALHO, F. A. Tecnologias educacionais: a utilização do ENEM no ensino médio. *Revista Tecnologias na Educação*, v. 14, n. 33, p. 22–35, 2022.

INEP. *Resumo técnico ENEM 2022*. Brasília: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira, 2022.

KENSKI, V. M. *Tecnologias e ensino presencial e a distância*. 7. ed. Campinas: Papirus, 2012.

KLUYVER, T. et al. Jupyter Notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (org.). *Positioning and Power in Academic*

Publishing: Players, Agents and Agendas. Amsterdam: IOS Press, 2016. p. 87–90. Disponível em: https://doi.org/10.3233/978-1-61499-649-1-87. Acesso em: 3 jun. 2025.

LUTZ, M. *Learning Python*. 5. ed. Sebastopol: O'Reilly Media, 2013.

MARTINS, L. Python: uma linguagem acessível para iniciantes. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 11., 2021, Porto Alegre. *Anais* [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021. p. 88–95.

MARTINS, R. L. *Python para desenvolvedores*. São Paulo: Novatec, 2021.

MERKEL, D. Docker: lightweight Linux containers for consistent development and deployment. *Linux Journal*, [S. l.], n. 239, p. 2, mar. 2014. Disponível em: https://linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment. Acesso em: 3 jun. 2025.

MORAN, J. M. *A educação que desejamos: novos desafios e como chegar lá*. 4. ed. Campinas: Papirus, 2007.

MORAN, J. M. *Metodologias ativas para uma aprendizagem mais significativa*. Campinas: Papirus, 2015.

NUNES, M.; GASPARIN, J. Simulados como instrumentos de avaliação formativa no ensino médio. *Revista Educação e Ensino*, v. 31, n. 2, p. 75–92, 2020.

PATEL, A. API Testing and Development with Insomnia. 2021. Disponível em: https://blog.insomnia.rest/api-testing-and-development. Acesso em: 03 jun. 2025.

PATEL, M. Insomnia REST Client: API testing made simple. *Medium*, 2021. Disponível em: https://medium.com/insomnia-rest. Acesso em: 16 maio 2025.

PÉREZ, F.; GRANGER, B. E. Project Jupyter: computational narratives as the engine of collaborative data science. *Technical Report*, 2015.

SANTOS, P. A.; MONTEIRO, L. M. Plataformas digitais de simulado e o papel do professor: análise das funcionalidades voltadas ao ensino. *Revista Brasileira de Tecnologias Educacionais*, v. 7, n. 1, 2021.

SILVA, J. A. da; SOUZA, P. H. de. Desenvolvimento de aplicações web com Flask. *Revista de Tecnologia Aplicada*, v. 15, n. 1, p. 45–59, 2022.

SILVA, L. A.; SOUZA, F. B. *Desenvolvimento web com Flask: fundamentos e aplicações práticas*. Rio de Janeiro: Ciência Moderna, 2022.

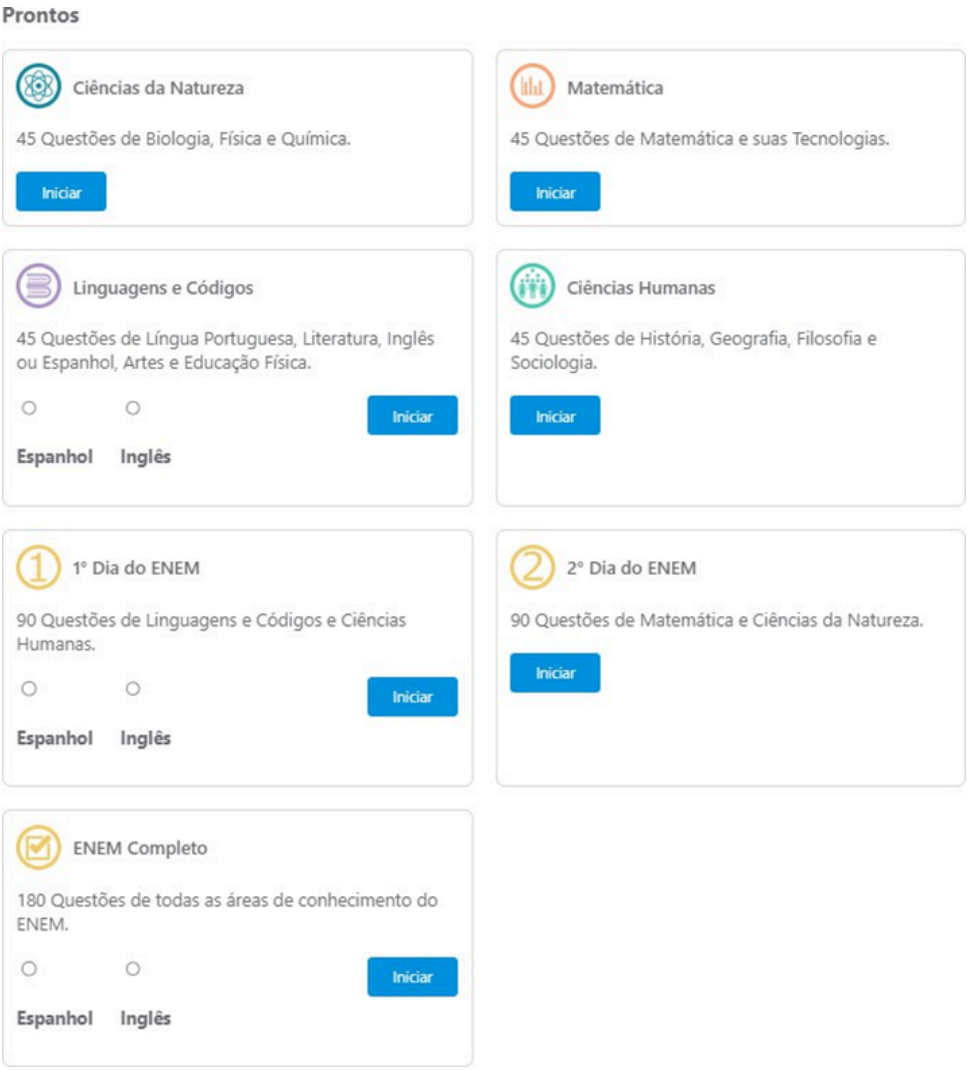
SOMMERVILLE, I. *Engenharia de software*. 9. ed. São Paulo: Pearson, 2011.

TANCREDI, R. História do ENEM: origem e transformações. *Guia do Estudante*, 2024. Disponível em: https://guiadoestudante.abril.com.br/enem/origem-do-enem. Acesso em: 6 maio 2025.

VALENTE, J. A. Tecnologias digitais e a transformação da educação. *Revista Brasileira de Informática na Educação*, v. 26, n. 1, p. 7–28, 2018.

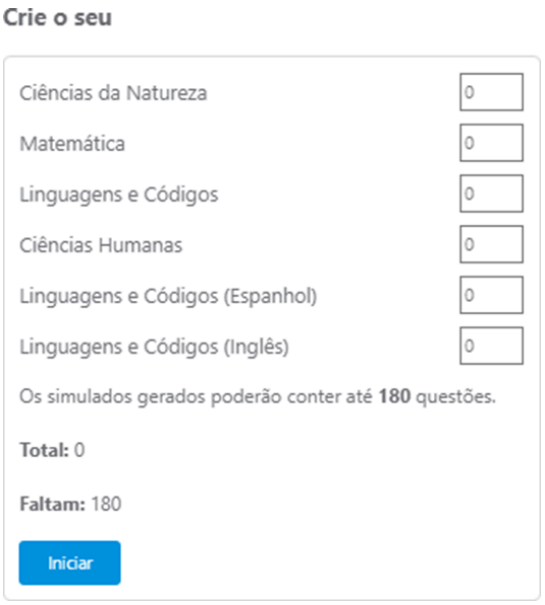
ANEXO A – CAPTURAS DE TELAS DE APLICAÇÕES ANALISADAS

Figura A.1 – Página inicial do Simulador Brasil Escola



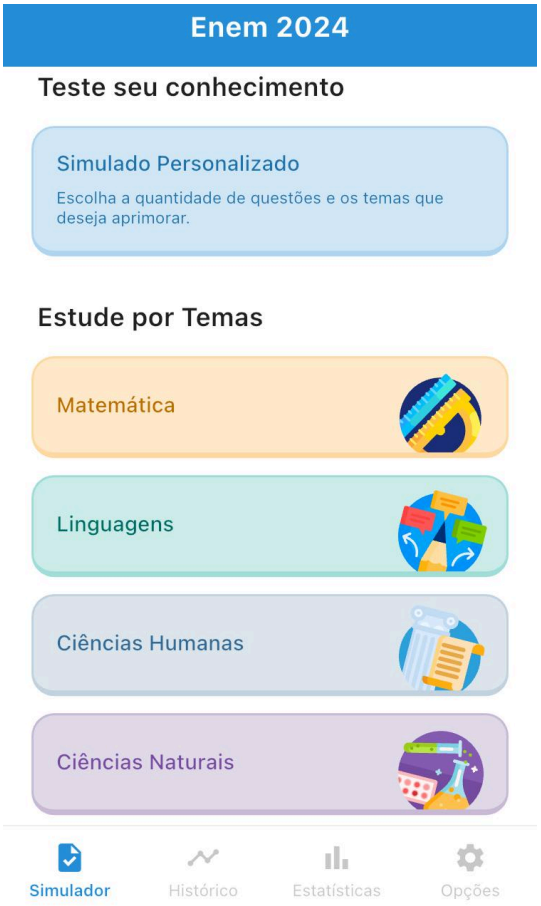
Fonte: Simulador Brasil Escola (2024)

Figura A.2 – Criação de Simulado no Brasil Escola



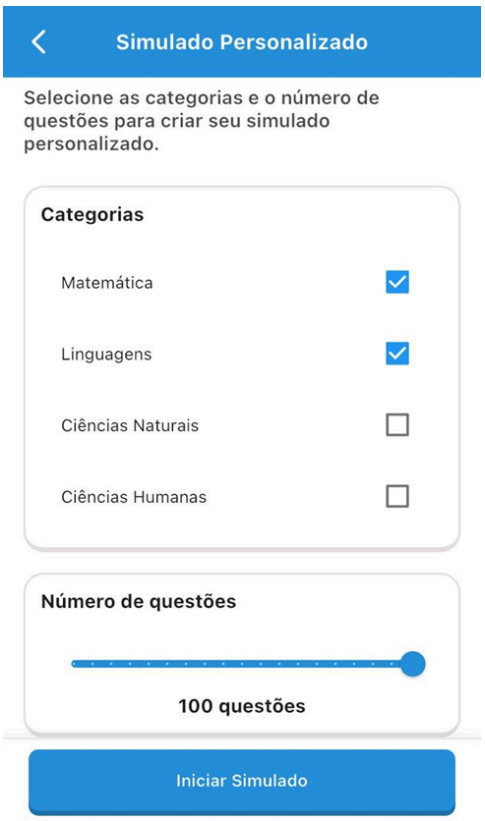
Fonte: Simulador Brasil Escola (2024)

Figura A.3 – Página inicial do aplicativo ENEM – Provas e Simulados



Fonte: ENEM – Provas e Simulados (2024)

Figura A.4 – Simulados personalizados no aplicativo ENEM



Fonte: ENEM – Provas e Simulados (2024)

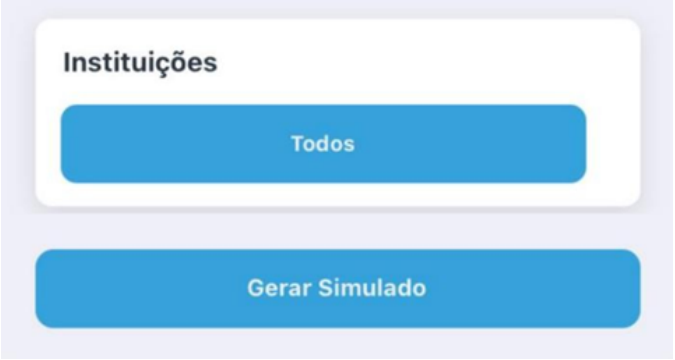
Figura A.5 – Temas específicos no aplicativo ENEM

Fonte: ENEM – Provas e Simulados (2024)

Figura A.6 – Página inicial do aplicativo Estuda.com

Fonte: Estuda.com (2024)

Figura A.7 – Geração de simulados por instituição no Estuda.com



Fonte: Estuda.com (2024)