UNIVERSIDADE
LUSÓFONA

NicePandora

Trabalho Final de curso

Relatório Intercalar 1º Semestre

Francisco Chambel, 22203238, LIG

Orientador: Pedro Serra

Coorientador: Daniel Silveira

Entidade Externa: -

Departamento de Engenharia Informática da Universidade Lusófona

Centro Universitário de Lisboa

2024/2025

Direitos de cópia

Agradecimentos

Resumo

O *Pandora* é uma plataforma educacional inovadora que utiliza inteligência artificial para melhorar a avaliação e compreensão de código em ambientes de aprendizagem. Este sistema foi projetado para resolver problemas enfrentados por professores e alunos durante processos de avaliação em disciplinas de programação, proporcionando feedback detalhado, contextualizado e adaptado ao nível de competência do utilizador.

A solução implementa funcionalidades avançadas, como a análise de código comparativa entre submissões de alunos e códigos de referência, exibição de mensagens de erros detalhadas, e relatórios de progresso para professores. Além disso, o *Pandora* oferece personalização no feedback, incluindo dicas ajustadas, validação estrutural de código e uma interface intuitiva que facilita a navegação e a usabilidade por parte de novos utilizadores.

No desenvolvimento, priorizou-se uma abordagem modular e escalável, permitindo integração com ferramentas educacionais existentes e suporte a múltiplas linguagens de programação. Foram realizados testes em ambiente controlado para validar a eficácia das funcionalidades propostas, demonstrando resultados promissores na melhoria da aprendizagem e eficiência na correção de códigos.

Com potencial para ser implementado em larga escala, o *Pandora* atende aos critérios de um produto viável mínimo, validando os conceitos propostos e destacando-se como uma solução prática e eficiente para desafios reais no ensino de programação. A plataforma está pronta para evoluir para ambientes de produção, demonstrando grande impacto.

Palavras chave: *Pandora*, Inteligência Artificial, Educação, Avaliação de Código, Personalização, Relatórios de Progresso, Feedback.

Abstract

Pandora is an innovative educational platform that leverages artificial intelligence to enhance code evaluation and comprehension in learning environments. This system is designed to address challenges faced by teachers and students during programming assessment processes, providing detailed, contextualized and competence-level-adjusted feedback.

The solution incorporates advanced features such as comparative code analysis between student submissions and reference codes, detailed error messages and progress reports for teachers. Additionally, *Pandora*, offers personalized feedback, including hints, structural code validation and an intuitive interface that simplifies navigation and usability for new users.

The development prioritized a modular and scalable approach, enabling integration with existing tools and support for multiple programming languages. Controlled environment tests validated the effectiveness of the proposed functionalities, demonstrating promising results in improving learning outcomes and efficiency in code assessment.

With the potential for large-scale implementation, *Pandora* meets the criteria of a minimum viable product, validating the proposed concepts and standing out as a practical and efficient solution for real challenges in programming education. The platform is ready to evolve into production environments, showcasing a significant impact on educational contexts.

Key-words: *Pandora*, Artificial Intelligence, Education, Code Assessment, Progress Reports, Feedback.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas

TFC	Trabalho Final de Curso
API	Interface de Programação de Aplicações
HTML	Linguagem de Marcação de Hipertexto
LP	Linguagem de Programação
AATs	Automated Assessment Tools
AI	Artificial Intelligence
ODS	Objetivos de Desenvolvimento Sustentável
EF	Epic Feature
RF	Requisito Funcional
RNF	Requisito Não Funcional
MVP	Minimum Viable Product
ORM	Object-Relational-Mapping
MVC	Model-View-Controller
HTTP	Hyper Transfer Protocol
GHz	Gigahertz
GB	Gigabytes
RAM	Random Access Memory
Mbps	Megabits per second
SSL	Secure Sockets Layer

1 Introdução

Nos últimos anos, a melhoria contínua de ferramentas de avaliação automática (Automated Assessment Tools) tem desempenhado um papel central na educação em programação, especialmente em cursos introdutórios.

Estas ferramentas possibilitam aos alunos submeterem os seus trabalhos de programação, os quais são avaliados automaticamente por meio de testes previamente definidos. A comparação entre os resultados dos testes e as expectativas do professor gera feedback, muitas vezes acompanhado de pontuações. Exemplos de AATs amplamente utilizadas incluem o Drop Project, Replit, Moodle CodeRunner e HackerRank.

No entanto, apesar de proliferação destas ferramentas, questões relacionadas com a adaptabilidade e personalização às necessidades específicas de cursos e instituições permanecem desafios significativos. Foi neste contexto que o Pandora, uma AAT desenvolvida na Universidade Lusófona emergiu como uma solução inovadora, projetada especialmente para a avaliação de projetos em linguagem de programação C.

O Pandora introduziu funcionalidades como a submissão em grupo, configuração flexível de feedback, suporte para leitura e escrita de ficheiros, definição de flags, compilações específicas e controlo administrativo avançado.

Apesar dos avanços introduzidos pelo Pandora, existem ainda oportunidades de melhoria e modernização da aplicação.

O presente trabalho, denominado NicePandora, foca-se na evolução e otimização do Pandora, com o objetivo de expandir as suas funcionalidades, aumentar a sua usabilidade e responder a novas necessidades identificadas pelos utilizadores.

1.1 Enquadramento

As AATs têm ganho destaque nos últimos anos, particularmente no ensino de programação, devido à sua capacidade de avaliar de forma rápida e objetiva os trabalhos dos alunos. Estas ferramentas permitem que os alunos submetam os seus trabalhos de programação e sejam avaliados com base numa séria de testes automáticos, facilitando o processo de feedback.

No entanto, apesar da sua popularidade, muitas AATs enfrentam desafios relacionados com a usabilidade, a personalização e a adaptação às necessidades de cursos específicos.

O Pandora é um exemplo de AAT desenvolvido para a avaliação automática de projetos em C. Contudo, a aplicação enfrenta várias limitações, como uma interface pouco intuitiva e erros frequentes, que comprometam a sua eficiência e experiência do utilizador.

1.2 Motivação e Identificação do Problema

A motivação para o desenvolvimento do NicePandora surge da necessidade de melhorar uma ferramenta existente, que, apesar de inovadora, apresenta sérias falhas em termos de usabilidade e estabilidade.

O Pandora, embora útil, é frequentemente descrito pelos utilizadores como uma aplicação confusa e difícil de navegar, o que compromete a experiência de avaliação tanto para os alunos quanto para os professores. Além disso, a aplicação apresenta falhas técnicas que podem levar a erros nos testes e dificuldades para realizar submissões.

A proposta deste trabalho é solucionar essas questões, tornando o Pandora mais intuitivo, estável e eficaz. O problema a ser resolvido envolve, portanto, a melhoria da usabilidade da aplicação, a resolução de falhas técnicas e a adição de novas funcionalidades, como o AI, para tornar a ferramenta mais eficaz, flexível e inovadora.

1.3 Objetivos

O objetivo principal deste trabalho é melhorar significativamente a plataforma do Pandora, transformando-a numa ferramenta mais intuitiva, robusta e tecnologicamente avançada para a avaliação de projetos de programação.

As melhorias propostas visam otimizar a experiência dos utilizadores (alunos e professores), resolver falhas técnicas e integrar tecnologias modernas, como a inteligência artificial, para aumentar a eficácia e a flexibilidade da plataforma.

Objetivos Gerais

- Melhorar a usabilidade e estabilidade do Pandora, garantindo uma experiência mais fluida e confiável para todos os utilizadores.
- Implementar funcionalidades avançadas baseadas na inteligência artificial para personalizar e enriquecer o feedback das submissões do código.

Objetivos Específicos

- **Reestruturar a interface do Pandora:** Tornar a plataforma mais intuitiva, com uma navegação simplificada e um design que facilite o uso por parte dos alunos e professores.
- **Aumentar a estabilidade da aplicação:** Implementar melhorias técnicas para evitar falhas, como a instabilidade do site durante momentos de alta utilização, garantindo um desempenho consistente mesmo sob grande carga.
- **Integrar inteligência artificial:** Utilizar a API do GPT para fornecer dicas e sugestões de melhorias para os alunos com base no código submetido e nos requisitos fornecidos pelos professores. A implementação incluirá:
    - Geração de prompts otimizados que assegurem que o sistema forneça dicas orientadas, sem oferecer respostas completas, promovendo a aprendizagem ativa dos alunos.
    - Análise automática do código submetido para identificar erros comuns e fornecer feedback personalizado.
    - Integração de uma camada de segurança para evitar abusos no uso da funcionalidade da inteligência artificial.
- **Facilitar a criação e gestão de exercícios pelos professores:** Desenvolver ferramentas administrativas que permitam a configuração de exercícios e testes de forma mais eficiente e intuitiva.
- **Melhorar a documentação e suporte ao utilizador:** Criar guias claros para alunos e professores, incluindo exemplos de como aproveitar as novas funcionalidades.
- **Testar a eficácia das novas funcionalidades:** Realizar testes com utilizadores reais para validar as melhorias, com base em métricas como a satisfação do utilizador, o tempo de execução das submissões e a precisão do feedback fornecido.

1.4 Estrutura do Documento

- Na Secção 2 é apresentada análise da Pertinência e a Viabilidade do trabalho desenvolvido. Esta Secção aborda a necessidade de desenvolver este projeto e o impacto que terá nos alunos, professores e instituição. A análise benchmarking compara a solução proposta com outras existentes no mercado e destaca as suas diferenças numa tabela. É ainda abordado o possível impacto que este projeto poderá ter no mercado de trabalho, juntamente com uma possível continuação após o término do TFC.
- Na Secção 3 é são apresentados os requisitos (funcionais e não funcionais) e os modelos do projeto a desenvolver. Os requisitos são apresentados por meio de uma tabela, havendo ainda a possibilidade de criação, alteração ou remoção consoante o desenvolvimento do trabalho. Foram ainda criados casos de uso para proceder a uma explicação mais detalhada da aplicação, seja no ponto de vista do professor ou do aluno.
- Na Secção 4 são abordados mais a fundo os elementos funcionais e técnicos da plataforma. A arquitetura da plataforma é detalhada, juntamente com as tecnologias utilizadas. Os ambientes de desenvolvimento, teste e produção são abordados com as ferramentas a utilizar.
- Na Secção 6 foi explicada a metodologia de gestão de tarefas usada neste relatório intercalar, o Kanban.

O quadro abaixo é indicativo de entregáveis para cada momento de avaliação, assumindo abordagem sequencial do desenvolvimento de TFC. Podendo ser adoptadas outras abordagens metodológicas (e.g.: metodologias ágeis), serão aceites outras organizações de entregas. Deve-se, no entanto, observar duas condições: (i) a 1ª entrega deverá manter os conteúdos indicados no quadro, por forma a permitir ao júri avaliar a pertinência do tema e a taxa de esforço esperada; (ii) a organização de conteúdos em cada entrega deve ter atenção aos critérios de avaliação de modo a garantir a uniformidade da avaliação

Quadro de conteúdos Desenvolvimento

Avaliação	Tipo	1. Introdução	2. Pertinência e Viabilidade	3.Especificação e Modelação	4. Solução Desenvolvida[1]	5. Testes e Validação	6.Método e Planeamento	7.Resultados	8.Conclusão
1ª entrega Intercalar	Qualitativa; Júri cego	Incluir	Incluir	incluir	incluir	N/A	Inicial	N/D	N/D
2ª Entrega Intercalar	Qualitativa; Júri cego	revisto	revisto	revisto	revisto	Incluir	revisto	N/D	N/D
Final[2]	Quantitativa; Presencial	Final	Final	Final	Solução Proposta	Final	Final[3]	Incluir	Incluir

[1] Sempre que aplicável, inclui código funcional, a disponibilizar em repositório *Git*

[2] O relatório final deverá incluir todos os conteúdos desenvolvidos ao longo do TFC

[3] Para a entrega final sugere-se a inclusão, em anexo, de todo os planos de trabalho anteriores com apreciação de progresso e avaliação das dificuldades encontradas na elaboração e cumprimento do planeamento

2 Pertinência e Viabilidade

2.1 Pertinência

O Pandora, como ferramenta de avaliação automática, desempenha um papel crucial no apoio ao ensino de programação, particularmente em disciplinas introdutórias. No entanto, os problemas identificados na sua interface, usabilidade e estabilidade comprometem a eficácia da plataforma e dificultam a adoção plena pelos seus utilizadores.

Para compreender melhor a opinião dos utilizadores acerca do Pandora, foi realizado um inquérito de pertinência sobre a opinião dos utilizadores em relação à plataforma. Estes foram alguns dos resultados obtidos:

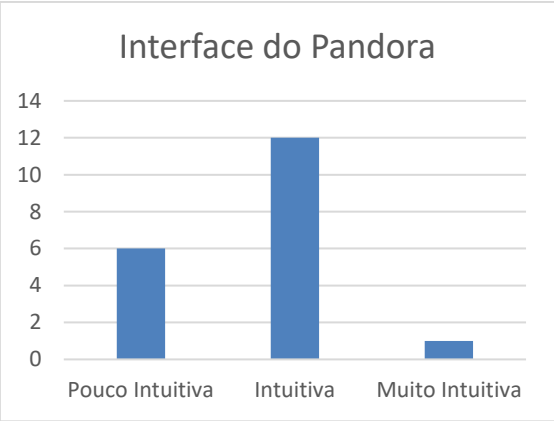

Figura 1 – Interface

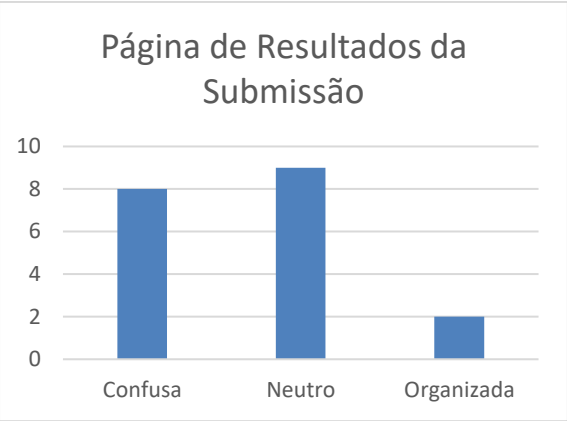

Figura 2 – Página Submissão

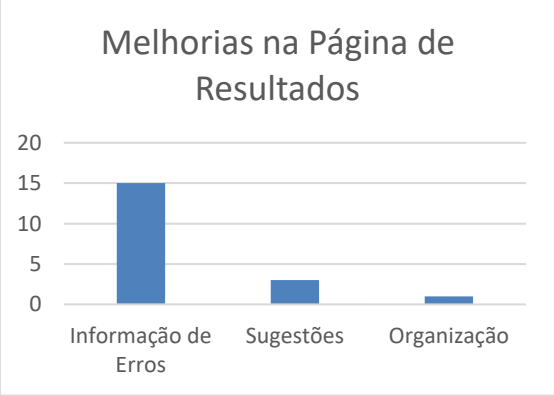

Figura 3 - Melhorias na Página

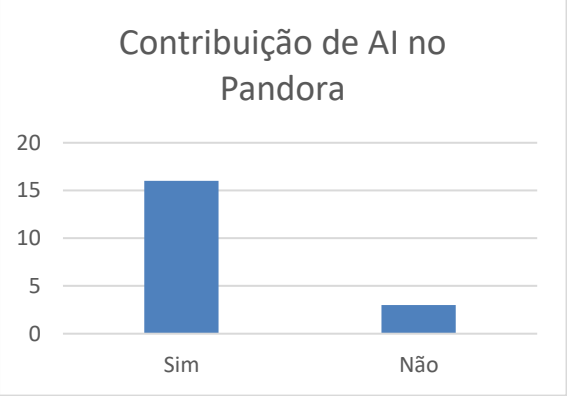

Figura 4 - Contribuição AI

Este trabalho propõe-se a resolver essas questões, apresentando um impacto positivo significativos tanto para alunos quanto para professores.

Impactos da Resolução do Problema

- **Alunos:** As melhorias propostas, como uma interface mais intuitiva e a integração de inteligência artificial para sugestões personalizadas, irão proporcionar uma experiência de utilização mais eficiente e motivadora. Os alunos terão acesso a feedback mais detalhado e relevante, ajudando-os a compreender melhor os seus erros e melhorar o desempenho.
- **Professores:** Com funcionalidades como a criação de exercícios mais simplificada e maior controlo sobre as submissões, o Pandora será um aliado mais poderoso no acompanhamento e avaliação dos alunos. Além disso, a estabilidade da plataforma em momentos de alta utilização garantirá a confiança no uso durante períodos críticos.
- **Instituição:** O reforço do Pandora como ferramenta robusta e eficaz fortalece a posição da universidade na adoção de tecnologias modernas no ensino, contribuindo para a inovação educacional e, possivelmente, atraindo novos alunos.

2.2 Viabilidade

A viabilidade deste trabalho foi avaliada com base em critérios técnicos, económicos, sociais e alinhamento com os Objetivos de Desenvolvimento Sustentável (ODS). A seguir, são apresentados os principais fatores analisados para garantir que o projeto possa implementado e sustentado com sucesso, tanto no contexto académico como potencialmente no mercado.

O projeto do Pandora tem como objetivo principal melhorar a usabilidade e estabilidade de uma ferramenta educacional, promovendo a qualidade da educação (ODS 4). De forma específica, o projeto busca:

- **ODS 4 (Educação de Qualidade):** Contribuir para o ensino superior através de uma plataforma eficiente para a avaliação automática, aumentando a acessibilidade e praticas pedagógicas modernas.
- **ODS 9 (Indústria, Inovação e Infraestrutura):** Promover a inovação no setor de ferramentas tecnológicas aplicadas à educação, com a integração de inteligência artificial para proporcionar uma experiência mais adaptada às necessidades dos utilizadores.
- **ODS 17 (Parcerias e Meios de Implementação):** Estimular colaborações entre instituições educativas e tecnológicas ao oferecer uma ferramenta alinhada às necessidades modernas do ensino.

Viabilidade Técnica

- **Tecnologias Disponíveis:** O Pandora já está desenvolvido com base em linguagens e frameworks amplamente conhecidas. As melhorias propostas, como a integração com inteligência artificial via GPT API, são tecnicamente viáveis e utilizam tecnologias acessíveis e amplamente suportadas.
- **Acessibilidade de Recursos:** Existem ferramentas para testar e validar funcionalidades durante o desenvolvimento, incluindo servidores para simular cargas, frameworks de teste e ferramentas de monitorização de estabilidade.

- **Prototipagem e Testes:** Um protótipo será apresentado como parte deste projeto, validando as novas funcionalidades. A sua escalabilidade será demonstrada através de testes de stress para garantir que o sistema se mantém estável em cenários de alta demanda.

Viabilidade Económica

- **Custos:** A utilização da API do GPT tem custos associados, mas são escaláveis e podem ser ajustados à medida da utilização. A maior parte das melhorias propostas exige desenvolvimento de software, que pode ser implementado em ambiente académico com apoio de recursos existentes.
- **Benefícios:** Um Pandora melhorado pode gerar economia de tempo para professores e estudantes, reduzindo a necessidade de revisão manual de trabalhos e aumentando a eficiência do ensino.
- **Sustentabilidade:** A solução é projetada para ser mantida pela universidade, com custos reduzidos a longo prazo apos a implementação inicial.

Viabilidade Social

- **Aceitação por Utilizadores:** Os dados coletados no questionário indicam que há insatisfação com algumas das funcionalidades do Pandora. As melhorias propostas visam diretamente resolver esses pontos, aumentando a satisfação do utilizador.
- **Beneficiados:** Alunos e professores são os principais beneficiados, sendo que o Pandora oferecerá uma interface mais intuitiva, maior estabilidade e funcionalidades inovadoras como sugestões inteligentes baseadas em inteligência artificial.
- **Engajamento:** O projeto incluirá etapas de coleta de feedback contínuo para garantir que a solução atenda às expectativas e necessidades dos utilizadores.

Impacto Ambiental

O projeto não possui impacto ambiental direto significativo, mas a sua abordagem tecnológica pode contribuir para reduzir o consumo de recursos físicos (por exemplo, papel para tarefas e avaliações), alinhando-se aos objetivos de sustentabilidade.

Com base nestes fatores, conclui-se que o projeto é viável técnica, económica e socialmente. O seu alinhamento com os ODS e a proposta de melhorias inovadoras no Pandora garantem a sua relevância e potencial para continuar após a conclusão do TFC, podendo ser mantido e expandido pela comunidade académica e educacional.

2.3 Análise Comparativa com Soluções Existentes

2.3.1 Soluções existentes

No mercado educacional e tecnológico, diversas ferramentas oferecem funcionalidades que, em diferentes níveis, se assemelham ou complementam as capacidades do Pandora. A seguir, são apresentadas algumas dessas soluções, com uma análise comparativa para contextualizar a proposta e melhoria do Pandora.

- **Moodle:** É uma das plataformas de aprendizagem mais amplamente utilizadas no mundo. É uma solução de código aberto que oferece suporte para a gestão de cursos, avaliação automática, fóruns de discussão e outros recursos educacionais. Embora robusto, a sua interface muitas vezes é considerada pouco intuitiva, e a sua implementação pode ser complexa para algumas instituições.
- **Replit:** É uma plataforma online que permite programar, compilar e executar código diretamente no navegador. Além de suportar várias linguagens de programação, possui funcionalidades para ensino e colaboração, permitindo que professores avaliem os exercícios dos alunos de maneira dinâmica. Apesar disso, o Replit é limitado em termos de integração com plataformas de ensino e não oferece recursos como configuração granular de feedback de nos testes, algo essencial para cursos avançados.
- **Drop Project:** É uma ferramenta de avaliação automática desenvolvida para o contexto académico, focada em facilitar submissões e avaliação de projetos de programação. Suporta múltiplas linguagens e oferece integração com sistemas de autenticação universitários. Contudo, a sua interface ainda pode ser considerada pouco amigável, e a flexibilidade para adaptar os testes e feedback às disciplinas especificas é limitada.
- **HackerRank:** É uma plataforma popular para avaliação de competências em programação, frequentemente utilizada em processos de recrutamento e ensino técnico. A ferramenta oferece uma vasta biblioteca de exercícios e suporte para diversas linguagens de programação. No entanto, a sua abordagem genérica e voltada para empresas limita a personalização para contextos académicos específicos.

O Pandora diferencia-se das soluções existentes ao ser desenvolvido especificamente para o contexto da Universidade Lusófona, com características adaptadas às necessidades do corpo docente e discente. Em comparação:

- **Flexibilidade:** O Pandora permite configurar níveis de feedback, o que oferece uma experiência de ensino mais ajustada às necessidades especificas.
- **Custo-Benefício:** Sendo uma ferramenta interna, reduz significativamente os custos associados a plataformas comerciais.
- **Inovação:** A futura integração com inteligência artificial (GPT API) posiciona o Pandora como uma solução de vanguarda, oferecendo sugestões baseadas no código submetido, sem entregar respostas completas, mas focando na aprendizagem.

As melhorias propostas visam não apenas superar as limitações do Pandora, mas também colocá-lo em posição de competir diretamente com soluções estabelecidas, oferecendo uma alternativa eficiente e ajustada ao contexto local.

2.3.2 Análise de benchmarking

Para avaliar a competitividade do Pandora em relação a outras ferramentas, foi desenvolvida uma análise comparativa. Os critérios analisados foram:

1. **Interface Intuitiva:** facilidade de uso da interface para utilizadores com diferentes níveis de experiência.
2. **Feedback Personalizado:** capacidade de fornecer feedback ajustado às necessidades do utilizador.
3. **Suporte para Avaliações Automáticas:** funcionalidade para corrigir exercícios ou projetos automaticamente.
4. **Integração com Sistemas Académicos:** possibilidade de integração com plataformas de gestão de aprendizagem, como o Moodle.
5. **Custos Acessíveis:** relação custo-benefício competitiva para instituições académicas.
6. **Flexibilidade Técnica:** suporte para diversas linguagens e configurações avançadas.
7. **Uso de Inteligência Artificial:** integração com ferramentas de AI para melhorar a experiência do utilizador.

Critérios	Pandora	Moodle	Replit	Drop Project	HackerRank
Interface Intuitiva			X		X
Feedback				X	X
Avaliações Automáticas	X	X	X	X	X
Integração Académica	X	X		X	
Custos Acessíveis	X	X	X	X	
Flexibilidade Técnica			X	X	X
IA					X

Tabela 1 - Análise Benchmarking

2.4 Proposta de inovação e mais-valias

A proposta de melhoria do Pandora destaca-se por incorporar elementos inovadores que elevam a sua funcionalidade e utilidade no contexto académico, além de proporcionar vantagens que beneficiam tanto os utilizadores quanto a instituições parceiras.

Elementos Inovadores

1. **Integração de Inteligência Artificial**
    - Utilização de modelos de AI (como o GPT) para oferecer sugestões de melhoria nas submissões dos utilizadores.
    - Aplicação de prompts configurados para evitar a resolução direta de problemas, incentivando o raciocínio e a aprendizagem guiada.
2. **Feedback Personalizado e Inteligente**
    - Implementação de um sistema de feedback automático que avalia o código submetido e fornece orientações específicas para cada erro.
    - Ferramenta educativa que não apenas corrige, mas também ensina, promovendo uma curva de aprendizagem mais eficiente.
3. **Estabilidade e Escalabilidade Aprimoradas**
    - Reestruturação técnica para garantir que o sistema suporte um grande volume de utilizadores sem falhas, aumentando a sua confiabilidade em momentos críticos.
4. **Interface Intuitiva e Moderna**
    - Redesenho do layout e da experiência do utilizador para tornar a navegação mais clara e eficiente, promovendo um maior engajamento por parte de alunos e professores.
5. **Flexibilidade Multilinguagem**
    - Expansão do suporte a múltiplas linguagens de programação, permitindo que a plataforma atenda a diferentes áreas de estudo e projetos interdisciplinares.

Diferenciais em Relação às Abordagens Existentes

- **Personalização por IA:** Enquanto ferramentas como o Drop Project e o HackerRank se concentram em avaliações técnicas, o Pandora vai além ao utilizar AI para sugerir melhorias e ensinar com base nas submissões individuais.
- **Integração Académica Focada:** Diretamente de soluções como o Replit, que são mais gerais, o Pandora é desenvolvido para atender às necessidades específicas do ambiente académico, incluindo compatibilidade com sistemas como o Moodle.

Vantagens e Benefícios

1. **Melhoria na Eficiência Académica**
    - Redução de carga para professores, que podem confiar no feedback automático para correções preliminares.
    - Aumento na autonomia dos estudantes, que recebem dicas claras para aperfeiçoamento.
2. **Acessibilidade e Inclusão**
    - Custo competitivo que facilita a adoção por instituições de diferentes portes.
    - Interface acessível para utilizadores com diferentes níveis de experiência técnica.
3. **Impacto Social**
    - Estímulo à educação de qualidade ao oferecer uma ferramenta robusta para ensino de programação.
    - Incentivo ao pensamento crítico e à solução de problemas.
4. **Sustentabilidade e Escalabilidade**
    - Estrutura técnica preparada para crescimento contínuo sem comprometimento da performance.
    - Suporte contínuo à comunidade académica, com potencial de expansão do uso para outros setores, como o empresarial.

Mais-Valias para Parceiros

Para as instituições parceiras, o Pandora oferece um diferencial competitivo ao proporcionar uma plataforma adaptada às suas necessidades locais. A integração de AI e melhorias técnicas garantem que a solução tenha um impacto duradouro, com potencial para evoluir e atender a novas demandas mesmo após o término do TFC.

Com esta abordagem, o Pandora consolida-se como uma ferramenta essencial para o ensino e aprendizagem, destacando-se pela inovação e impacto.

2.5 Identificação de oportunidade de negócio

A evolução do Pandora para uma plataforma comercial viável abre portas para uma oportunidade significativa no mercado de soluções educacionais baseadas em tecnologia, especialmente no contexto da transformação digital no ensino superior.

Contexto de Mercado e Necessidade

O ensino de programação e disciplinas técnicas enfrenta desafios crescentes, como o elevado número de estudantes por turma, dificultando a personalização do ensino, a necessidade de ferramentas acessíveis que facilitem a prática e avaliação de exercícios de programação e a demanda por plataformas que combinem com funcionalidades avançadas com simplicidade de uso.

Estes fatores criam uma lacuna no mercado que o Pandora pode preencher ao oferecer uma solução específica, eficiente e adaptada às necessidades académicas.

Impacto no Empreendedorismo Tecnológico

O Pandora posiciona-se como um exemplo de como projetos académicos podem transcender o ambiente universitário para gerar valor comercial e social. Este projeto tem o potencial de se tornar um ponto de referência no ensino digital, promovendo inovação e acessibilidade no setor educacional.

Com a abordagem certa, o Pandora pode não apenas resolver problemas educacionais, mas também estabelecer-se como um negócio sustentável e inovador no setor tecnológico.

3 Especificação e Modelação

Epic Features:

EF-01: Como professor, quero uma plataforma que permita avaliar códigos de alunos automaticamente para facilitar o processo de correção e acompanhamento de progresso.

EF-02: Como aluno, quero receber feedback detalhado sobre o meu código e dicas contextuais para melhorar a minha aprendizagem.

Features:

Para EF-01 (Professor):

1. Gerar relatórios detalhados do progresso dos alunos.
2. Comparar código dos alunos com códigos de referência.
3. Permitir criar e configurar testes automáticos.
4. Exportar relatórios em formato PDF ou CSV.

Para EF-02 (Aluno):

1. Receber mensagens de erro detalhadas e categorizadas.
2. Obter dicas de melhoria do código submetido.
3. Consultar histórico de submissões e comparar versões.
4. Receber feedback contextual de AI sobre linhas erradas do código.

3.1 Análise de Requisitos

A análise de requisitos é essencial para delinear as funcionalidades e características desejadas para o Pandora. Este levantamento detalhado abrange os requisitos necessários para abordar os problemas identificados e implementar as soluções propostas. Além disso, considera cenários de continuidade do projeto para além do TFC, permitindo escalabilidade e melhorias futuras.

Os requisitos identificados foram categorizados por tipo (funcional, não funcional e de sistema) e priorizados com base no impacto no utilizador e viabilidade técnica. Para cada requisito, são definidos critérios de aceitação, garantindo a clareza na avaliação da sua concretização.

3.1.1 Enumeração de Requisitos

ID	Descrição	Prioridade	Impacto
RF-10	O plugin AI deve pedir a LP.	Alta	Alto
RF-11	O Pandora deve permitir que o utilizador altere a LP selecionada.	Média	Médio
RF-12	O Pandora deve exibir uma lista de LPs mais utilizadas.	Média	Médio
RF-20	O plugin AI deve fornecer a LP ao OpenAI.	Alta	Alto
RF-21	O Pandora deve verificar a compatibilidade da LP com o código fornecido.	Alta	Médio
RF-22	O Pandora deve exibir um alerta em caso de incompatibilidade de linguagem.	Alta	Média
RF-30	O Pandora deve permitir que o utilizador escolha o nível de dificuldade das dicas geradas.	Média	Médio
RF-31	O Pandora deve oferecer níveis predefinidos de dificuldade para dicas.	Média	Médio
RF-32	O Pandora deve lembrar a preferência do utilizador para futuras sessões.	Baixa	Médio
RF-40	O Pandora deve armazenar prompts gerados para consulta posterior.	Baixa	Médio
RF-41	O Pandora deve permitir que o histórico de prompts seja exportado.	Baixa	Médio
RF-42	O Pandora deve organizar o histórico por projeto ou tema.	Média	Médio
RF-50	O Pandora deve exibir dicas contextuais sem fornecer respostas diretas.	Alta	Alto
RF-52	O Pandora deve solicitar feedback sobre a utilidade das dicas fornecidas.	Baixa	Médio
RF-60	O Pandora deve validar a estrutura do código antes de enviá-lo para correção.	Alta	Médio
RF-61	O Pandora deve sugerir correções automáticas para erros comuns.	Alta	Alto
RF-70	O Pandora deve exibir mensagens de erro detalhadas ao utilizador.	Alta	Alto

RF-71	O Pandora deve exibir erros categorizados (sintaxe, compilação, execução).	Alta	Médio
RF-80	O Pandora deve apresentar relatórios detalhados de progresso ao professor.	Alta	Alto
RF-90	O Pandora deve ser capaz de analisar o código fornecido pelos professores.	Alta	Alto
RF-91	O Pandora deve comparar o código dos alunos com o código de referência.	Alta	Alto
RF-100	O Pandora deve identificar e exibir apenas o início do output obtido que diverge do output esperado.	Alta	Alto
RF-101	O Pandora deve permitir que o utilizador visualize o restante do output obtido, caso deseje.	Média	Médio
RF-102	O Pandora deve destacar a primeira linha de código diferente do output esperado.	Média	Alto
RF-103	O Pandora deve permitir que o utilizador filtre os testes por sucesso ou falha.	Média	Média
RF-110	O Pandora deve fornecer um histórico de submissões, permitindo o utilizador comparar diferentes submissões.	Alta	Alto
RF-111	O Pandora deve possibilitar o utilizador fazer download de qualquer submissão presente no histórico.	Média	Alto

Tabela 2 - Requisitos Funcionais

ID	Descrição	Prioridade	Impacto
RNF-10	O Pandora deve fornecer uma interface clara e intuitiva, com navegação simplificada.	Alta	Alto
RNF-20	O Pandora deve permitir que utilizadores novos compreendam a interface e realizem submissões com sucesso em até 5 minutos.	Média	Médio
RNF-30	O Sistema deve ter um uptime de pelo menos 99.9% durante períodos letivos.	Alta	Alto
RNF-40	O Pandora deve armazenar todas as informações sensíveis, como códigos e dados dos utilizadores, de forma criptografada.	Alta	Alto
RNF-50	O sistema deve autenticar todos os utilizadores antes de permitir acesso às funcionalidades.	Alta	Médio
RNF-60	O código do Pandora deve ser documentado seguindo padrões de boas práticas, permitindo que novos desenvolvedores entendam 80% do sistema em até 2 semanas.	Alta	Médio

Tabela 3 - Requisitos Não Funcionais

3.1.2 Descrição detalhada dos requisitos principais

RN-20: O plugin AI deve fornecer a LP ao OpenAI.

- **Objetivo:** Garantir que o modelo de AI tem o contexto correto sobre a LP para gerar feedback relevante e preciso.
- **Dependências:**
  - O código do aluno deve especificar ou inferir a linguagem de programação usada.
  - O modelo OpenAI deve suportar a linguagem fornecida.
- **Critérios de Aceitação:**
  - A LP correta deve ser incluída nos prompts enviados ao OpenAI em 100% das submissões.
  - Testes automáticos devem verificar a formatação do prompt antes do envio.
- **Processos de Negócio:**
  - Durante a submissão, o sistema verifica a linguagem indicada.
  - A LP é então adicionada ao prompt da análise do OpenAI.

RN-40: O Pandora deve armazenar prompts gerados para consulta posterior.

- **Objetivo:** Permitir que professores e administradores revisem prompts para fins de auditoria e análise.
- **Dependências:**
  - Base de dados segura para armazenar históricos de prompts.
  - Sistema de autenticação para acesso aos dados armazenados.
- **Critérios de Aceitação:**
  - 100% dos prompts devem ser armazenados com metadados relevantes (e.g., data, hora, utilizador, LP).
- **Processos de Negócio:**
  - Após a geração, cada prompt é automaticamente enviado para a base de dados com detalhes da submissão.

RF-50: O Pandora deve exibir dicas contextuais sem fornecer respostas diretas.

- **Objetivo:** Oferecer assistência aos alunos sem comprometer a integridade da aprendizagem.
- **Dependências:**
  - AI capaz de gerar feedback contextual sem expor diretamente a solução.
  - Regras definidas para filtrar informações excessivamente específicas.
- **Critérios de Aceitação:**
  - Em 90% dos casos, as dicas devem ser úteis sem fornecer diretamente a solução.
  - Alunos devem classificar a utilidade das dicas como “satisfatória” ou superior em testes.
- **Processos de Negócio:**
  - O código submetido é analisado e o modelo gera sugestões baseadas em padrões de erros comuns.

RF-70: O Pandora deve exibir mensagens de erro detalhadas ao utilizador.

- **Objetivo:** Ajudar o utilizador a identificar e corrigir erros sem dificuldade.
- **Dependências:**
  - Analisador de código que identifica tipos de erros (sintaxe, lógica, etc.).
  - Base de dados com mensagens e erros padronizadas.
- **Critérios de Aceitação:**
  - Todas mensagens devem incluir: descrição do erro, linha afetada e possíveis causas.
  - 90% dos utilizadores devem considerar as mensagens claras e úteis.
- **Processos de Negócio:**
  - Após a execução do código, o sistema gera relatórios com detalhes específicos do erro.

RF-80: O Pandora deve apresentar relatórios detalhados de progresso ao professor.

- **Objetivo:** Fornecer uma visão abrangente do desempenho dos alunos ao longo do tempo.
- **Dependências:**
  - Registro de todas as submissões, incluindo resultados e feedback.
  - Interface que permita visualização gráfica e textual dos dados.
- **Critérios de Aceitação:**
  - 80% dos professores devem considerar dos relatórios úteis e informativos.
- **Processos de Negócio:**
  - Os dados de submissão são agregados em relatórios que incluem métricas como taxa de sucesso e tipos de erros frequentes.

RF-91: O Pandora deve comparar o código dos alunos com o código de referência.

- **Objetivo:** Identificar discrepâncias entre o código do aluno e o código ideal para destacar áreas de melhoria.
- **Dependências:**
  - Base de dados com códigos de referência fornecidos pelos professores.
  - Algoritmo de comparação de semântica e estrutural.
- **Critérios de Aceitação:**
  - 95% das comparações devem identificar diferenças significativas com precisão.
- **Processos de Negócio:**
  - O código submetido é comparado com o de referência, e as diferenças são destacadas num relatório visual.

3.1.3 Casos de Uso/*User Stories*

Caso de Uso 1: Criação de Contest

- **Ator Principal:** Professor
- **Objetivo:** Configurar o Contest antes de disponibilizar para os alunos
- **Cenário:**
  1. O professor acessa o Pandora e faz login no sistema.
  2. O professor seleciona a secção de 'Contests' e seleciona a opção de 'Create Contest'.
  3. O sistema solicita que o professor selecione a LP (RF-10).
  4. O professor seleciona a LP desejada (RF-12).
  5. O professor cria testes, introduzindo o resultado esperado para cada um, com a sua respetiva percentagem.
  6. O sistema solicita que o professor selecione o nível de dificuldade das dicas geradas (RF-30).
  7. O professor seleciona o nível de dificuldade das dicas geradas.
  8. O professor seleciona a/s disciplina/s em que o contest estará disponível, juntamente com a data limite de entrega.
  9. O professor publica o contest.

Caso de Uso 2: Exibição de Dicas Contextuais

- **Ator Principal:** Aluno
- **Objetivo:** Obter dicas sobre o código submetido para facilitar correções sem comprometer a aprendizagem.
- **Cenário:**
    1. O aluno acessa o Pandora e faz login no sistema.
    2. O aluno seleciona a secção de 'Contests' e escolhe o contest pretendido.
    3. O aluno submete o código para análise no Pandora.
    4. O Pandora valida a estrutura do código antes de enviá-lo para o mecanismo de correção (RF-60).
    5. O sistema processa o código, identifica erros comuns e sugere correções automáticas (RF-61).
    6. As dicas geradas são exibidas de forma contextual, sem fornecer a resposta diretamente (RF-50).
    7. O aluno dá feedback sobre a utilidade das dicas, ajudando a melhorar a experiência no futuro (RF-52).

Caso de Uso 3: Análise e Comparação de Código

- **Ator Principal:** Professor
- **Objetivo:** Analisar o progresso dos alunos e verificar a correção dos códigos submetidos.
- **Cenário:**
    1. O professor faz login no Pandora e acessa os relatórios de submissão (RF-80).
    2. O professor seleciona o contest que deseja analisar.
    3. O professor seleciona um aluno específico para analisar a sua submissão.
    4. O Pandora compara o código submetido pelo aluno com o código de referência (RF-91).
    5. O sistema exibe as diferenças entre os códigos, destacando pontos de divergência para facilitar a avaliação (RF-70).

Caso de Uso 4: Exibição e Exploração de Resultados

- **Ator Principal:** Aluno
- **Objetivo:** Visualizar resultados detalhados de uma submissão para corrigir erros.
- **Cenário:**
    1. Após submeter o código, o aluno acessa a página de resultados no Pandora.
    2. O Pandora exibe uma lista de testes com os seus resultados, categorizados por sucesso ou falha (RF-103).
    3. Para os testes falhados, o Pandora exibe apenas a parte inicial do output obtido que difere do esperado (RF-100), destacando a linha inicial com erro (RF-102).
    4. Caso deseje, o aluno expande para visualizar o restante do output obtido (RF-101).
    5. O aluno pode fazer download do código de qualquer submissão para uma análise offline (RF-111).

3.2 Modelação

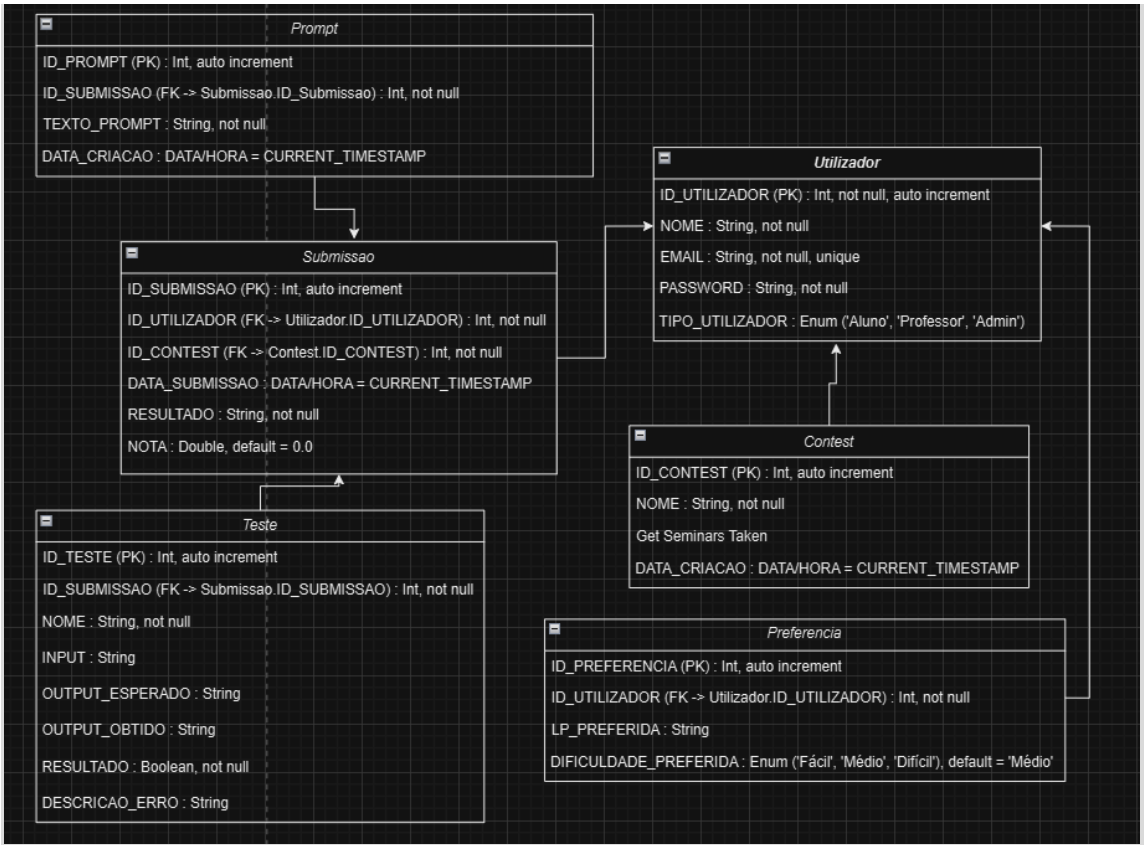

Figura 5 - Modelação

3.3 Protótipos de Interface

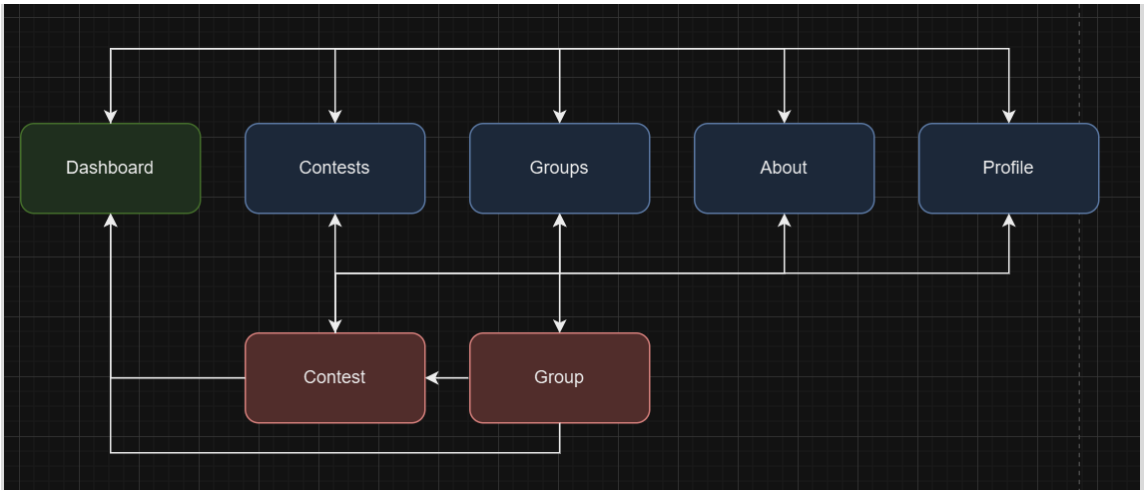

Figura 6 - Mapa Aplicacional

4 Solução Proposta

Nesta secção, descreve-se a solução desenvolvida, incluindo os principais elementos funcionais e técnicos, alterações em relação à proposta inicial, e uma comparação com soluções alternativas destacadas no benchmarking. O objetivo é apresentar uma visão detalhada da aplicação, juntamente com as suas funcionalidades e como elas abordam os problemas identificados.

4.1 Apresentação

A solução proposta consiste no sistema Pandora, uma plataforma que utiliza inteligência artificial para auxiliar alunos e professores na avaliação e compreensão de código. O sistema é projetado para suportar múltiplas linguagens de programação, oferecer feedback contextualizado e relatórios detalhados, e proporcionar um ambiente de aprendizagem e avaliação interativo.

A implementação baseia-se num conjunto de requisitos funcionais que priorizam usabilidade, precisão na análise de código e integração com ferramentas educacionais. Em relação às propostas anteriores, a solução foi ajustada para incluir maior personalização no feedback aos alunos e melhorias no processamento de submissões com erro, baseando-se em resultados obtidos nos testes iniciais do protótipo.

Na entrega final, a análise comparativa entre as propostas de desenvolvimento inicial e a solução implementada será apresentada, destacando quais funcionalidades foram implementadas, quais foram ajustadas e quais permanecem em desenvolvimento. A solução desenvolvida valida o conceito original, ainda que na forma de um MVP, destacando o potencial e aplicação em ambientes educacionais reais.

4.2 Arquitetura

A arquitetura da solução foi projetada para ser modular, escalável e eficiente, utilizando uma combinação de tecnologias bem estabelecidas no mercado para o desenvolvimento de aplicações web e integração com AI. A escolha das tecnologias e ferramentas foi baseada na sua robustez, facilidade de uso e adequação aos requisitos funcionais e não funcionais do projeto.

A arquitetura geral segue um modelo MVC:

- **Frontend (View):**
    - Desenvolvido com HTML, CSS e JavaScript (com Bootstrap).
    - Interage com o backend através de chamadas HTTP.
- **Backend (Controller)**
    - Implementado em Django, gere a lógica dos caminhos, autenticação e a integração com a API GPT.
- **Base de Dados (Model)**
    - Usada para armazenar informações relacionadas aos utilizadores, histórico de interações e dados necessários para a personalização da aplicação.
- **Integração com GPT:**
    - Um script em Python faz as chamadas à API do OpenAI, enviando o código ou pergunta do utilizador e retornando as respostas do AI.

Esta arquitetura modular e baseada em tecnologias amplamente utilizadas garante que a solução seja eficiente, extensível e fácil de manter. A escolha de ferramentas robustas, como o Django e a OpenAI API, permite foco no desenvolvimento da lógica central e garante uma integração confiável entre AI e funcionalidades específicas do projeto.

4.3 Tecnologias e Ferramentas Utilizadas

1. **Django**
    - O Django é a framework principal para o desenvolvimento da aplicação. Ele foi escolhido devido à sua estrutura organizada, suporte integrado a ORM e uma vasta comunidade ativa. Além disso, o Pandora foi previamente desenvolvido utilizando o Django.
    - Facilita a rápida prototipagem, possui recursos de segurança robustos e é ideal para construir aplicações com módulos de autenticação e gestão de utilizadores.
    - **Função na Solução:** Gerir caminhos, a lógica e a interação com a base de dados.
2. **Python**
    - Além de ser a linguagem base do Django, o Python será usado para implementar a interação com o GPT (API do OpenAI).
    - A ampla biblioteca de módulos e a simplicidade da sintaxe tornam o Python ideal para integrar a funcionalidade de AI na aplicação.
    - **Função na Solução:** Gerar prompts, processar respostas do AI e validar código.
3. **HTML/CSS**

- O HTML será utilizado para estruturar os elementos visuais da aplicação, enquanto que o CSS estilizará as páginas para proporcionar uma interface clara e intuitiva.
- **Função na Solução:** Construir uma interface responsiva e amigável ao utilizador.

4. **JavaScript**
    - O JavaScript será usado para melhorar a interatividade e experiência do utilizador, como validações em tempo real e atualizações dinâmicas no front-end.
    - **Função na Solução:** Enriquecer a experiência do utilizador, por exemplo, com feedback instantâneo na submissão de código.
5. **Bootstrap**
    - Esta framework do CSS será utilizada para acelerar o design responsivo e garantir uma interface visual e moderna.
    - **Função na Solução:** Criar páginas que se ajustem automaticamente a diferentes dispositivos e resoluções.
6. **SQLite/MySQL**
    - Durante o desenvolvimento, a base de dados do SQLite será usada para simplicidade e agilidade na fase inicial. Para a fase final, o MySQL será considerado devido à sua escalabilidade e desempenho.
    - **Função na Solução:** Armazenar os dados dos utilizadores, histórico de submissões, prompts gerados e configurações personalizadas.
7. **API do OpenAI**
    - A integração com o modelo GPT será feita através da API oficial do OpenAI.
    - **Função na Solução:** Fornecer feedback aos estudantes e dicas contextuais ajustadas ao nível de dificuldade.
8. **Git/GitHub**
    - O controlo de versão será gerido com o Git, enquanto o código será armazenado e colaborado via GitHub.
    - **Função na Solução:** Versionamento do código.
9. **VSCode**
    - O Visual Studio Code será utilizado como IDE principal devido à sua versatilidade e ampla gama de extensões que suportam desenvolvimento em Python, Django e front-end.
    - **Função na Solução:** Desenvolvimento eficiente e organizado.

4.4 Ambientes de Teste e de Produção

O sucesso do projeto depende de ambientes devidamente configurados para desenvolvimento, testes e produção. Esta secção detalha os requisitos e a configuração de cada ambiente, com foco nos recursos necessários para o funcionamento estável e eficiente da solução.

Ambiente de Desenvolvimento

O ambiente de desenvolvimento é projetado para facilitar a implementação e depuração da solução, permitindo um ciclo de feedback rápido.

- **Ferramentas Utilizadas:**
    - **Sistema Operativo:** Windows10/11 ou Ubuntu 20.04+
    - **IDE:** Visual Studio Code com extensões para Python e Django.
    - **Ambiente Virtual:** Python Virtualenv para isolamento de dependências.
    - **Base de Dados Local:** SQLite para desenvolvimento inicial.
    - **Servidor Local:** Django *runserver*.
    - **Recursos Computacionais Necessários:**
      - Processador: Dual-core 2 GHz ou superior.
      - RAM: 4 GB (mínimo).
      - Armazenamento: 1 GB disponível para código, dependências e base de dados.

Ambiente de Teste

O ambiente de teste é projetado para simular as condições reais de uso da aplicação, garantindo que os componentes sejam avaliados antes da publicação.

- **Configuração do Ambiente:**
    - Implantação do código num servidor interno.
    - Base de dados no MySQL para maior realismo.
    - Simulação de chamadas à API do OpenAI.
    - Servidor web leve configurado para correr a aplicação.
    - Logs habilitados para identificar erros em tempo real.
- **Recursos Computacionais Necessários:**
    - Processador: Quad-core 2.5GHz ou superior.
    - RAM: 8GB
    - Armazenamento: 10GB para a base de dados e logs.
    - Rede: 50 Mbps de conexão para integração com serviços externos.

Ambiente de Produção

O ambiente de produção será configurado para suportar acesso público e garantir alta disponibilidade, segurança e desempenho.

- **Configuração do Ambiente:**
    - Implantação num servidor Cloud.
    - Configuração da base de dados em MySQL para escalabilidade.
    - Uso de um servidor robusto.
    - Integração com a API do OpenAI, com autenticação segura.
    - Certificado SSL configurado para garantir conexões seguras.
- **Recursos Computacionais Necessários:**

- Processador: 4 vCPUs ou superior.
- RAM: 16 GB.
- Armazenamento: 50 GB para base de dados, logs e arquivos estáticos.
- Rede: Conexão estável com largura de banda mínima de 100 Mbps.
- Dependências Externas:
  - Acesso à API do OpenAI com um plano suficiente para o volume de chamadas esperadas.

Critério	Desenvolvimento	Teste	Produção
Base de Dados	SQLite	MySQL	MySQL
Servidor Web	Django runserver	Gunicorn	Gunicorn
Logs	Básico	Habilitado	Detalhado/Monitorado
Rede	Local	Simulado	Alta Performance
Recursos	Limitados	Moderados	Escaláveis

Tabela 4 - Comparação Entre Ambientes

4.5 Abrangência

O desenvolvimento desta solução envolve a aplicação prática de conhecimentos adquiridos em diversas unidades curriculares do curso. Estas unidades forneceram as bases teóricas e práticas necessárias para abordar diferentes desafios do projeto.

Sistemas Operativos

- **Aplicação no Projeto:**
    - Configuração e gestão de servidores para os ambientes de teste e produção, incluindo o uso de sistemas operativos baseados em Linux (como o Ubuntu).
    - Noções sobre multitarefas e escalabilidade para garantir o funcionamento eficiente do sistema num ambiente multitarefa.
    - Gestão de recursos computacionais, como alocação de memória e armazenamento, essenciais para o dimensionamento da solução em produção.

Programação Web

- **Aplicação no Projeto:**
    - Desenvolvimento da aplicação web utilizando Django, uma framework baseada em Python que facilita a criação de sistemas robustos e escaláveis.
    - Integração com tecnologias frontend, como HTML, CSS e JavaScript, para criar uma interface amigável e responsiva.
    - Implementação de comunicação entre o frontend e backend usando APIs REST.

Interação Humano-Máquina

- **Aplicação no Projeto:**
    - Desenho e prototipagem de interfaces baseadas em princípios de usabilidade e experiência do utilizador.
    - Avaliação e validação de interfaces através de testes de usabilidade, garantindo que o sistema seja intuitivo para diferentes tipos de utilizadores.
    - Implementação de funcionalidades acessíveis e consistentes, com navegação clara entre as páginas.

4.6 Componentes

Detalhe de cada um dos componentes, realçando aspectos técnicos de sua implementação
(A desenvolver no seguinte relatório)

4.6.1 Componente 1

4.6.2 Componente n

4.7 Interfaces

Mapa aplicacional composto por *screenshots* dos ecrãs mais representativos da aplicação, com uma descrição do que fazem, como é que foram implementados e que decisões foram tomadas na sua implementação.
(A desenvolver no seguinte relatório)

5 Testes e Validação

Plano de testes para validação prática e operacional da solução construída. Mais do que demonstrar o funcionamento da solução, é importante que os testes demonstrem que ela cumpre os objetivos que se propôs, nomeadamente o de contribuir para a solução de um problema real, demonstrando aplicabilidade, pertinência e relevância.

Os testes a realizar devem incidir sobre qualidade da solução desenvolvida, validação de funcionamento e de operação em contexto produtivo, podendo-se acrescentar outros formatos que se considerem relevantes para a natureza do trabalho.

Nesta secção, deve-se incluir abordagem e justificação para os testes, sempre que possível recorrendo a modelos formais de análise de riscos e de impacto[4]. Em anexo, deve-se apresentar guião detalhado de testes, com descrição de cenários e resultados

Sempre que possível, os testes de validação operacional devem incluir verificação dos recursos indicados (computacionais, armazenamento e rede, assim como serviços *cloud* ou *web services* de terceiros ou artefactos físicos)

Tal como acontece com o formato da solução proposta, também o plano de testes e validação deverá ser coerente com a natureza do trabalho e alinhado com os requisitos definidos, com particular incidência na demonstração de cumprimento dos critérios de aceitação.

Valorizam-se trabalhos com plano de testes e validação a decorrerem em ambientes reais e com forte participação terceiros[5].
(A desenvolver no seguinte relatório).

[4] e.g.: *Fault Tree Analysis*, Diagrama causa-efeito; *Ishikawa*

[5] No caso de validações por terceiros, a demonstração poderá ser realizada por questionários de satisfação, entrevistas ou testemunhos individuais. Nestes casos, o anexo de testes deve incluir inquéritos a aplicar, guiões de entrevistas ou descrição de outros processos de obtenção de avaliação de utilizadores ou outros *stakeholders*

6 Método e Planeamento

6.1 Planeamento inicial

O desenvolvimento do projeto começou com a criação de um Kanban Board no Microsoft Teams, utilizado como ferramenta para organizar e acompanhar as tarefas ao longo do projeto. Este board foi dividido em colunas como o To Do, Doing e Pending, permitindo a visualização clara das tarefas e do progresso do trabalho.

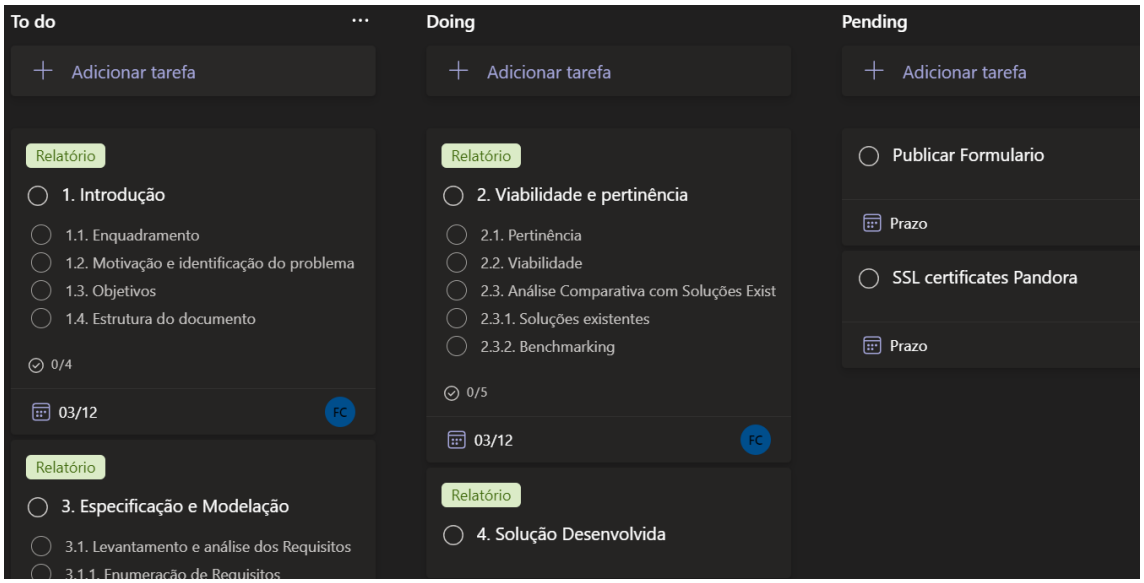

Figura 7 - Kanban Board

O uso do Kanban Board foi essencial para o monitoramento contínuo do progresso, ajudando a priorizar tarefas e garantir que todas as etapas importantes fossem concluídas dentro do prazo. Além disso, permitiu identificar tarefas bloqueadas ou pendentes para resolução em tempo hábil.

6.2 Análise Crítica ao Planeamento

Aconselha-se uma análise critica sobre o cumprimento do calendário e indicações do progresso do trabalho, onde se refira tarefas realizadas, dificuldades mais marcantes e alterações que tenham sido introduzidas ao plano e objectivos anteriores.
Esta secção é aconselhada na segunda entrega e indispensável na final
(A desenvolver no seguinte relatório).

7 Resultados

7.1 Resultados dos Testes

Descrição detalhada de resultados, *outputs* e *outcomes*.
Incluído anexo com os *test cases* e respetivos resultados.
Valorizam-se trabalhos onde a avaliação de critérios de cumprimento de requisitos e test cases seja efetuada por terceiros[6], devendo os resultados de inquéritos, entrevistas e testemunhos serem incluídos em anexo
(A desenvolver no seguinte relatório).

7.2 Cumprimento de requisitos

Apresentar uma tabela com todos os requisitos indicados no capítulo 3 e uma indicação para cada um: realizado, realizado parcialmente, não realizado, abandonado. Nestes 3 últimos casos, apresentar uma justificação.
(A desenvolver no seguinte relatório).

[6] Os testes de aceitação por terceiros podem ser realizados, entre outras hipóteses, por meio de inquéritos de satisfação a potenciais utilizadores reais, realização de *test cases* previamente determinados, utilização efectiva da solução em contexto real, etc.

8 Conclusão

8.1 Conclusão

Analise crítica da realização do TFC, onde se abordem, entre outras, questões como:

- Grau de concretização do plano
- Diferenças entre solução proposta inicialmente e solução desenvolvida
- Evolução do trabalho e conhecimentos ao longo do TFC
- O que se faria diferente se o TFC voltasse ao princípio
- Maiores dificuldades na realização do TFC

(A desenvolver no seguinte relatório)

8.2 Trabalhos Futuros

Assumindo que o trabalho pudesse ser continuados, quais os próximos passos? Como se pode melhorar a solução

Tendo em vista a vertente de inovação e empreendedorismo, o que se pode fazer para aumentar o potencial da solução?

(A desenvolver no seguinte relatório)

Bibliografia

[DEISI24] DEISI, Regulamento de Trabalho Final de Curso, Out. 2024.

[DEISI24b] DEISI, www.deisi.ulusofona.pt, Out. 2024.

[TaWe20] Tanenbaum,A. e Wetherall,D., *Computer Networks*, 6ª Edição, Prentice Hall, 2020.

[ULHT21] Universidade Lusófona de Humanidades e Tecnologia, www.ulusofona.pt, acedido em Out. 2024.

[PANDORA] Saturn Universidade Lusófona, https://saturn.ulusofona.pt/, acedido em Nov. 2024.

Anexo 1 – Recomendações para formatação de um relatório

A escrita do relatório deve seguir o presente template, não mudando nada em termos de formatação (fontes, espaçamentos, tamanhos, etc). **Este anexo exemplificativo deverá ser removido antes de submeter o seu relatório**. Antes de entregar o relatório, exercite a sua capacidade de auto-crítica lendo-o e verificando se está adequadamente redigido.

Na Tabela 2 exemplifica-se uma tabela e a forma como esta deve ser referenciada. Como poderá ver, se passar com o rato por cima da palavra "Tabela 1", neste parágrafo, aparece o hiperlink. Tal é possível se for incluída uma referência da forma que se explica a seguir. As tabelas devem ser apresentadas sempre depois de referenciadas. A legenda da tabela deve ser inserida através da opção do menu *References**Insert caption* (no menu em cima do MS Word), sempre no topo da tabela. A referência a uma tabela insere-se através do comando *References**cross-reference*, sendo a sua numeração automática.

Tabela 2 – Tipos de Selectores existentes.

Tipo	h1, p
Universal	*
Classe	.class1
ID	#element
Atributo	[target=_blank]
Pseudo-classe	div:hover
Pseudo-elemento	p::first-letter

O processo de carregamento de uma página HTML está representado na Figura 8 para exemplificar como se deve inserir uma legenda a uma figura assim como uma referência a esta mesma. Para inserir uma Figura, seleccione *References**Insert Caption* e indique que quer inserir uma Figura. A figura deve sempre aparecer depois de ser referida no texto. Para inserir uma referência a uma figura, utilizar *References**Cross-reference*. O índice e listas de tabelas e figuras (mas páginas iii a v) actualizam-se automaticamente se inseridas desta forma. Para actualizar basta selecionar todo o texto e premir F9.

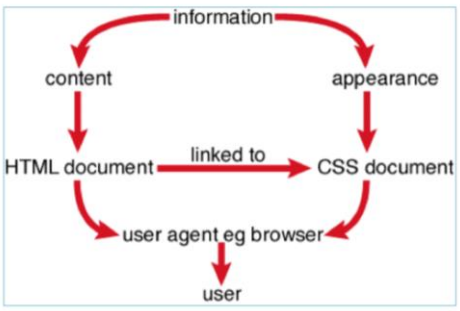

Figura 8 – Processo de carregamento de uma página HTML.

Explica-se de seguida a inserção de referências bibliográficas. Qualquer texto ou ideia que venha de uma referencia bibliográfica deve ser indicada com uma referência. Por exemplo, podemos

referir que este trabalho se enquadra dentro do regulamento do Trabalho Final de Curso [DEISI24]. O hyperlink aponta para a referencia bibliográfica inserida relativa ao regulamento de TFC. Para sua criação deve:

1. escrever o texto que pretende na bibliografia
2. usar uma numeração adequada [], de forma a que respeite a ordem de aparecimento da referencia no texto.
3. selecionar a referência inserida com o rato (por exemplo [2]) e escolher em Insert\Bookmark, criando um nome associado à referencia.

Depois, no texto onde pretender pode inserir a referência através de Insert\Cross-reference.

Glossário

LEI	Licenciatura em Engenharia Informática
LIG	Licenciatura em Informática de Gestão
TFC	Trabalho Final de Curso