

Plano de Desenvolvimento de Software (PDS)

Projeto: "Dimensão Desconhecida"

Empresa: Quantum Leap Studios

Versão: 1.3

Data: 27 de junho de 2025

Autor: Stuart (Líder Técnico)

Stakeholder Principal: Ruan Batista Rodrigues (Desenvolvedor)

1. Resumo Executivo

O projeto "Dimensão Desconhecida" visa desenvolver uma aplicação web completa e imersiva como um tributo à série clássica "The Twilight Zone". A aplicação servirá como um guia definitivo, permitindo aos utilizadores explorar todos os episódios, ler sinopses detalhadas, análises críticas, filtrar conteúdo e interagir através de avaliações e comentários.

O desenvolvimento seguirá as melhores práticas da indústria de software, utilizando metodologias ágeis (sprints semanais) e um fluxo de trabalho profissional (Git Flow, Pull Requests). O projeto será construído integralmente com recursos dos pacotes **GitHub Student Developer Pack** e **Azure for Students**, garantindo um custo de infraestrutura de **zero dólares**. O objetivo final não é apenas entregar um produto funcional, mas também proporcionar uma experiência de aprendizagem prática e profunda sobre o ciclo de vida do desenvolvimento de software.

2. Análise e Escopo do Projeto

2.1. Visão do Produto

Uma plataforma digital elegante e intuitiva onde fãs, novos e antigos, podem mergulhar no mundo de "The Twilight Zone". A aplicação deve ser uma fonte centralizada de informações, apresentada de forma atrativa e de fácil navegação, funcionando perfeitamente em dispositivos móveis e desktop, e promovendo uma comunidade de fãs através de funcionalidades interativas.

2.2. Público-Alvo

- **Fãs da série:** Pessoas que já conhecem e amam a série e procuram um local para revisitar episódios e aprofundar os seus conhecimentos.
- **Novos espectadores:** Pessoas curiosas sobre a série que precisam de um guia para começar a assistir.

- **Estudantes e entusiastas de cinema/TV:** Utilizadores interessados na análise crítica, roteiro e impacto cultural da obra de Rod Serling.

2.3. Funcionalidades Chave (MVP - Produto Mínimo Viável)

- **Listagem de Episódios:** Página principal com a exibição de todos os 243 episódios em formato de cartões.
- **Página de Detalhes:** Ao clicar num episódio, o utilizador será levado para uma página dedicada com sinopse, análise crítica completa e data de lançamento.
- **Busca Simples:** Uma barra de pesquisa para encontrar episódios por nome.
- **Páginas de Conteúdo Adicional:**
 - Página sobre o criador, Rod Serling.
 - Secção de curiosidades e "easter eggs".
- **Sistema de Filtros Avançados:** Capacidade de filtrar episódios por temporada, tema ou avaliação da comunidade.
- **Sistema de Avaliação e Comentários:** Os utilizadores poderão avaliar os episódios e deixar comentários, necessitando de um sistema de autenticação simples.
- **Design Responsivo:** A aplicação deve ser totalmente funcional e esteticamente agradável em telemóveis, tablets e computadores.

2.4. Funcionalidades Futuras (Pós-MVP)

- Notificações sobre novidades ou episódios em destaque.
- Integração com APIs de streaming para indicar onde assistir.
- Perfis de utilizador avançados.

3. Arquitetura e Stack Tecnológico

3.1. Visão Geral da Arquitetura

A aplicação seguirá uma arquitetura moderna de três camadas, desacoplada, que é padrão na indústria:

1. **Frontend (Cliente):** Uma aplicação web, construída com Flutter, responsável por toda a interface do utilizador.
2. **Backend (API):** Um servidor intermediário que lida com a lógica de negócio e comunica de forma segura com a base de dados.
3. **Base de Dados (Persistência):** Um serviço de base de dados na nuvem para armazenar e servir os dados.

Utilizador -> Navegador (Frontend Flutter/Web) -> Requisição HTTP -> API (Python/Django no Azure) -> Consulta -> Base de Dados (Cosmos DB)

3.2. Stack Selecionado

|

| Componente | Tecnologia Escolhida | Justificativa |

| Ambiente de Dev | GitHub Codespaces | Ambiente de desenvolvimento na nuvem, poderoso e consistente. Elimina a necessidade de configuração local e permite o desenvolvimento a partir de qualquer máquina. Disponível no GitHub Student Pack. |

| Base de Dados | Azure Cosmos DB (API NoSQL) | Ideal para os nossos dados em formato JSON. Oferece alta performance, escalabilidade e um plano gratuito robusto (1000 RU/s, 25 GB) que cobre integralmente as nossas necessidades. |

| Backend (API) | Python com Django | Framework Python "batteries-included" que acelera o desenvolvimento. O seu sistema de autenticação, admin e ORM (Object-Relational Mapper) integrados são perfeitos para implementar as funcionalidades de comentários e utilizadores do nosso escopo. |

| Hospedagem do Backend | Azure App Service | Plataforma totalmente gerida para hospedar aplicações web. Possui um runtime nativo para Python e o plano gratuito (F1) é suficiente para a nossa API. Integra-se perfeitamente com o GitHub Actions para deploy contínuo. |

| Frontend | Flutter (compilado para Web) | Framework moderno da Google que permite criar interfaces de utilizador bonitas e performáticas a partir de uma única base de código. O suporte à web permite-nos criar uma PWA (Progressive Web App) que pode, no futuro, ser facilmente compilada para mobile. |

| Hospedagem do Frontend | Azure Static Web Apps | Serviço otimizado para hospedar aplicações estáticas. Perfeito para o build web gerado pelo Flutter. Oferece um plano gratuito generoso, deploy direto do GitHub, SSL gratuito e alta performance global. |

3.3. Alternativas Consideradas

- **Flask para Backend:** Framework Python mais minimalista. Exigiria mais configuração manual para funcionalidades como autenticação, tornando o Django uma escolha mais produtiva para o nosso escopo.
- **React.js/Node.js:** A stack original. É uma combinação extremamente poderosa e popular. Foi preterida em favor da stack Python/Flutter a pedido do desenvolvedor para fins de aprendizagem e exploração de novas tecnologias.

4. Análise de Custos e Utilização de Recursos

O custo total de desenvolvimento e hospedagem deste projeto será de **\$0,00**, utilizando os seguintes benefícios:

- **GitHub Student Developer Pack:**
 - **GitHub Pro:** Permite repositórios privados e funcionalidades avançadas.
 - **GitHub Codespaces:** Fornece horas mensais gratuitas para o nosso ambiente de desenvolvimento na nuvem, mais do que suficiente para o projeto.


- **Microsoft Azure for Students:**


- **Crédito de \$100:** Não será necessário tocar neste crédito.
- **Serviços Gratuitos:**
 - **Azure Cosmos DB:** Utilizaremos o plano gratuito permanente (1000 RU/s, 25 GB).
 - **Azure App Service:** A nossa API Python/Django será hospedada no plano gratuito F1.
 - **Azure Static Web Apps:** O plano gratuito cobre a hospedagem do nosso frontend Flutter.


5. Roteiro de Desenvolvimento (Roadmap) e Cronograma


O desenvolvimento será dividido em "Sprints" semanais para acomodar o novo escopo e stack.


| Sprint | Foco Principal | Período Estimado | Status |


| Sprint 0: Planeamento e Infraestrutura | Setup do projeto, Git, BD e importação de dados. | 27 de junho de 2025 |  Concluído |


| Sprint 1: Backend - API Básica com Django | Configuração do projeto Django, conexão com o BD, criação dos endpoints GET /api/episodes e GET /api/episodes/:id com Django REST Framework. | 28 de junho - 04 de julho de 2025 |  A Iniciar |

| Sprint 2: Backend - API Avançada | Models e endpoints para filtros, comentários e avaliações. Setup do sistema de autenticação do Django. | 05 de julho - 11 de julho de 2025 |  A Iniciar |

| Sprint 3: Frontend - Base Flutter e Listagem | Setup do projeto Flutter (web), criação dos widgets visuais (Header, Card), página principal com lista e UI dos filtros. | 12 de julho - 18 de julho de 2025 |  A Iniciar |

| Sprint 4: Frontend - Páginas de Conteúdo | Página de detalhes do episódio, página de Rod Serling e secção de curiosidades. | 19 de julho - 25 de julho de 2025 |  A Iniciar |

| Sprint 5: Frontend - Interatividade | Implementação de UI para login/registo, formulário e exibição de comentários e avaliações, conectando à API. | 26 de julho - 01 de agosto de 2025 |  A Iniciar |

| Sprint 6: Testes, Automação e Lançamento | Testes manuais e automatizados (E2E), configuração do CI/CD com GitHub Actions para Python e Flutter, deploy final no Azure. | 02 de agosto - 08 de agosto de 2025 |  A Iniciar |

Data de Entrega Estimada (MVP): 08 de agosto de 2025

6. Riscos e Mitigação

| Risco Potencial | Mitigação |

| Dificuldades Técnicas Inesperadas | O uso de tecnologias populares com vasta documentação e o formato de mentoria deste projeto permitem uma rápida resolução de problemas. |

| "Scope Creep" (Aumento do Escopo) | O PDS e o quadro Kanban definem claramente as

funcionalidades do MVP. Novas ideias serão adicionadas ao backlog para fases futuras, mantendo o foco na entrega inicial. |

| Gestão de Tempo | A divisão em Sprints semanais com metas claras ajuda a manter o ritmo e a medir o progresso de forma consistente. |

7. Critérios de Sucesso

O projeto será considerado um sucesso quando os seguintes critérios forem atendidos:

- A aplicação web (MVP) está no ar e acessível publicamente.
- Todas as funcionalidades definidas para o MVP estão a operar corretamente.
- O custo total de infraestrutura manteve-se em \$0,00.
- O desenvolvedor adquiriu experiência prática em todo o ciclo de vida do desenvolvimento, da configuração à entrega final, utilizando a stack Python/Flutter.