



ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO
Licenciatura em Engenharia Informática

DESENVOLVIMENTO DE APLICAÇÕES WEB

SEUNEM
Eventos Comunitários, Recursos Partilhados e Decisão
Coletiva — Análise e Desenho

João Augusto Costa Branco Marado Torres



Beja, outubro de 2025

INSTITUTO POLITÉCNICO DE BEJA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO
Licenciatura em Engenharia Informática
DESENVOLVIMENTO DE APLICAÇÕES WEB

SEUNEM
Eventos Comunitários, Recursos Partilhados e Decisão
Coletiva — Análise e Desenho

João Augusto Costa Branco Marado Torres

Trabalho realizado no âmbito da unidade curricular de Desenvolvimento de Aplicações
Web

ORIENTAÇÃO

Dr. Luís Carlos da Silva Bruno

Beja, outubro de 2025

Júri

Responsável: Dr. Luís Carlos da Silva Bruno

Vogal: Dr. Luís Filipe Nobre Horta Baptista Garcia

Vogal: Dr. João Paulo Barros

Resumo

...

Palavras-chave: Gestão comunitária; Eventos; Recursos partilhados; Decisão coletiva; Participação democrática; Software MVC.

Abstract

...

Keywords: Community management; Events; Shared resources; Collective decision-making; Democratic participation; MVC software.

Dedicatória

...

Agradecimentos

...

Conteúdo

| | |
|---------------------------------------|-------------|
| Resumo | i |
| Abstract | ii |
| Dedicatória | iii |
| Agradecimentos | iv |
| Conteúdo | v |
| Lista de Figuras | vii |
| Lista de Tabelas | viii |
| Lista de Abreviaturas e Siglas | ix |
| 1 Introdução | 1 |
| 1.1 Contexto | 1 |
| 1.2 Objetivos | 2 |
| 2 Desenvolvimento | 4 |
| 2.1 Análise do Sistema | 4 |
| 2.2 Desenho do Sistema | 5 |
| 2.3 Metodologia | 5 |
| 2.4 Resultados | 7 |
| 3 Conclusão | 8 |
| 3.1 Trabalho Futuro | 8 |
| Referências Bibliográficas | 9 |
| Licença | 11 |

CONTEÚDO

vi

A Apêndice A: Questionário 12

B Apêndice B: Código de Exemplo 13

Anexos 14

Lista de Figuras

2.1 Diagrama UML dos casos de uso. 4

Lista de Tabelas

Lista de Abreviaturas e Siglas

a11y *Accessibility*

API *Application Programming Interface*

DX *Developer Experience*

FLOSS *Free Libre and Open Source Software*

HTTP *Hypertext Transfer Protocol*

i18n *Internationalization*

MVC *Model-View-Controller*

POO *Programação Orientada a Objetos*

REST *Representational State Transfer*

TDD *Test-Driven Development*

UML *Unified Modeling Language*

1 Introdução

Este relatório é sobre a fase de análise e desenho do SeUnem, que por enquanto é apenas uma plataforma de gestão de eventos democrática que também servirá para eu ter uma ideia de como a tecnologia deve auxiliar a participação democrática. Apenas auxiliar porque tenho a ideia que a participação democrática é um problema social, e não vai ser com a tecnologia que esses problemas vão ser resolvidos.¹ E como deve auxiliar porque só pode existir uma forma correta de fazê-lo: usando *Free Libre and Open Source Software* (FLOSS) (o Software Livre)^{2,3,4}, e serviços descentralizados (protocolos ao invés de plataformas).⁵ Tem que ser possível o *self-host* em que o objetivo final seria ter algo federado.⁶

Recentemente ouvi falar dos projetos (Decidim Community, 2025) e (Voca, 2025), que apesar de nunca ter usado e não saber quase nada acerca deles, tenho a sensação que muitas das ideias que queria aplicar no SeUnem foram também aplicadas lá, e os temas são relativamente parecidos.

1.1 Contexto

Imagina que tu fases parte de uma comunidade. Era muita fixe se conseguissem reunir-se para criar um evento e partilhar mais sobre as vossas paixões entre vós, mas também talvez entre pessoas curiosas que viram um anúncio sobre o vosso evento e quiseram aparecer. Eu nunca estive nessa situação para ser sincero, mas não parece ser algo fácil de fazer. Uma comunidade é feita de pessoas que têm algo em comum, mas não deixam de ser indivíduos com vidas diferentes umas das outras. Eu gostava que local do evento fosse à frente da minha casa, mas como eu vivo na Madeira e o resto da comunidade está espalhada por Portugal continental, eles provavelmente iriam preferir que o evento fosse algures entre a região Centro e a Península de Setúbal. Uns adoravam que as atividades mais fixes fossem antes da hora de almoço, outros depois. É complicado agradar todo o mundo, mas de certeza que com democracia direta, o resultado para a maioria (senão todos) vai ser a satisfação. Como é que de forma simples consegues convidar pessoas fora dessa bolha a fazer uma visita? Mas não acaba por aí! Esses eventos vão requerer coisas, podem ser cadeiras e mesas, ou hardware e software caso seja necessário disponibilizar ao vivo streaming do evento para o bacano lá da ilha. Essa gestão

¹Defesa dos Direitos Digitais, 2021.

²ANSOL - Associação Nacional para o Software Livre, 2025.

³Free Software Foundation, 2025a.

⁴Free Software Foundation, 2025b.

⁵Defesa dos Direitos Digitais, 2023.

⁶Fedigov, 2025.

de equipamentos não é tão fácil, e é capaz até de ter custos um pouco elevados. Bom é que vós não sois a única comunidade a existir em Portugal. Tem aquela comunidade meio esquisita de [...] que no mês passado, fez o seu evento anual, e que por isso já tem cadeiras e mesas disponíveis e que não estão em uso por mais uns 10 meses talvez, e que era fixe eles emprestarem esses produtos. Talvez só precisem de metade das cadeiras que eles têm, bom. Talvez precisam das cadeiras que eles têm e mais 20 (que lá em comunidade vão arranjar essas cadeiras que falam), então arranjam as cadeiras que faltam e quem sabe para o ano essas cadeiras novas vão fazer falta àqueles esquisitos.

Então o projeto vai ser um gestor de eventos para comunidades. Cria um evento para uma comunidade, dá data, hora e local e uma pequena descrição. Depois esses eventos serão listados. Se alguém quiser participar, diz que também vai, e dá "up"/"boost" nesse evento (podendo ser usado para apresentar os eventos mais populares). Quem vai e faz parte desse evento pode dar feedback antes, durante, e depois do evento. Esse feedback pode fazer com que data, hora ou local mudem. Se a mudança ocorrer, quem já ia participar tem de ser notificado sobre a mudança para confirmar se ainda está interessado no evento ou não (como padrão, ele não vai estar interessado até ele dizer que sim). Como eventos normalmente necessitam de bens como cadeiras, mesas, entre outras coisas, que podem muito bem ser reutilizadas de evento para evento, também vai existir forma de "partilhar" esses bens. Se um evento acaba, mas daqui a uma semana tem outro evento, as cadeiras e mesas podem ser reutilizadas nesse outro evento, e é feita uma passagem de "ownership" temporária do bem. Quem participa no evento com bens em falta, vai ter palavra sobre quais bens requisitar, da mesma forma que podem decidir mudanças de horário por exemplo. Se existir bens em excesso, elas continuam no "banco", mas se faltar, um dos organizadores compra o que falta e adiciona ao "banco" para poder ser utilizado em outro evento futuro. Esta transferência de bens serve para fazer *analytics* também. Ver quais mercadorias tem maior disponibilidade e quais são as mais necessárias. Deste modo, até alguém que não organiza os eventos, pode doar mercadorias conforme as necessidades.

1.2 Objetivos

Isto é um trabalho para a escola, e eu acredito que o ensino tem que ser livre, outra razão para o projeto ser FLOSS também. Infelizmente não é isso que eu vejo, preferem «preparar-nos para trabalhar nas empresas». O software proprietário é inerentemente contrário à educação e sabedoria, já que **não deixa o estudante estudar** o programa, o que ele realmente faz e como, modificar para as suas necessidades e a dos seus colegas e partilhar com os mesmos a versão modificada. Nas aulas de Matemática e Física eu sempre tive acesso à fórmulas e aos algoritmos, porquê que nas Ciências da Computação tem que ser diferente? Afinal, (Guimarães, 2025)! Eu vou fazer a minha parte.

Também faz-me sentido que os dados e a informação sejam abertos. Para isso vou ter em conta o (James G. Kim and Michael Hausenblas, 2025) e os (GO FAIR Initiative, 2025).

Temos que pensar na internacionalização (i18n) e ter acessibilidade (a11y) em mente, pensar no offline, proteger a anonimidade dos utilizadores e encriptar comunicações quando elas têm que existir.

Basicamente, a ideia não é existir diferentes tipos de utilizadores, todos os utilizadores vão ter o mesmo “poder” em tudo. A pessoa que cria o evento, de início, vai ter todo o poder sobre aquele evento. Mas depois de pessoas dizerem que vão participar no evento, elas tem tanto quanto o criador. As decisões são feitas por todos de forma democrática. Qualquer pessoa também poderá participar num evento (ao menos que horários colidam, ou sei lá). Qualquer um também consegue doar mercadorias para o banco. Mas pensando bem, era bom existir pessoas que conseguem melhor gerir o banco de mercadorias. Porque após um evento acabar, essas mercadorias não vão estar diretamente acessíveis, primeiro vão precisar serem devolvidas ou banco. Aí era preciso um gajo para trocar o status da mercadoria para “disponível”. Mas pensando mais um pouco, talvez não tenha de ser um tipo de utilizador diferente. Pode existir um sistema de dizer que alguém é o encarregado temporário da mercadoria, e ele tem a responsabilidade de, ou por a mercadoria no banco e marcar o status da mercadoria como disponível, ou entregar diretamente para o outro evento, e o status continua indisponível. Nas duas situações ele deixa de ser o encarregado da mercadoria. É de pensar.

Um dia talvez eu desenvolva o projeto além da ideia atual. Explorar, para além da coordenação e mobilização comunitária e a troca de bens comuns e distribuição circular, a situação do planeamento e orçamentação coletivos e do conhecimento coletivo.

Promover a gestão coletiva, o combate a burocracia, e uma real democracia direta.

Estrutura do documento

O presente relatório encontra-se estruturado da seguinte forma:

- O **Capítulo 1** apresenta o enquadramento, a motivação, os objetivos e a metodologia do projeto;
- O **Capítulo 2.1** descreve a análise de requisitos, os atores e os casos de uso do sistema *SeUnem*;
- O **Capítulo 2.2** apresenta o desenho do sistema, incluindo diagramas UML e protótipos das interfaces;
- Por fim, o **Capítulo 3** sintetiza as conclusões do trabalho e perspectivas de evolução futura.

2 Desenvolvimento

2.1 Análise do Sistema

2.1.1 Caracterização dos atores

2.1.2 Diagrama de Casos de Uso

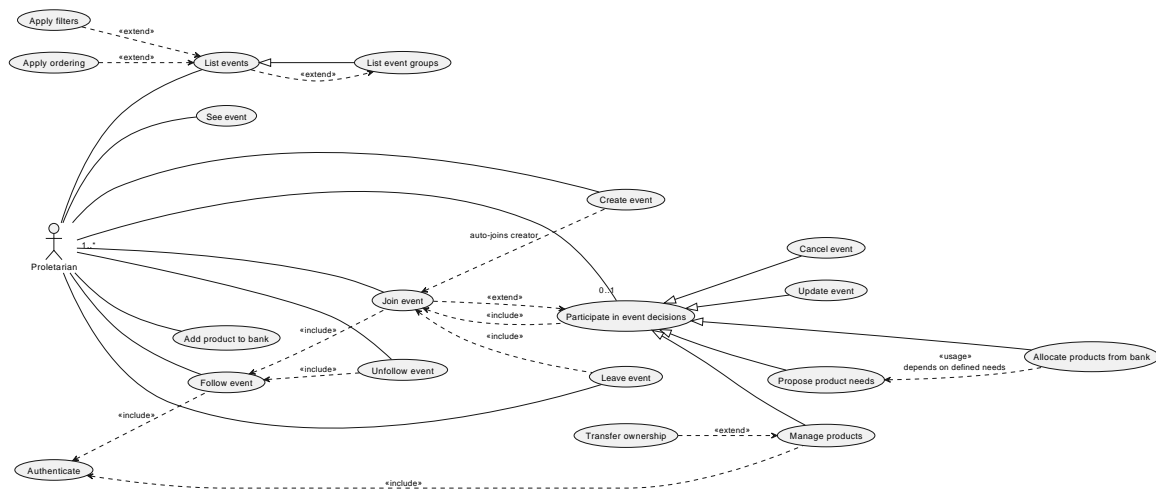


Figura 2.1 Diagrama UML dos casos de uso.

2.2 Desenho do Sistema

2.2.1 Modelação da Base Dados

2.2.1.1 Diagrama E/R

2.2.1.2 Modelo Lógico Relacional

2.2.2 Modelação de Interfaces Gráficas com o Utilizador

2.2.2.1 *Storyboard(s)*

2.2.2.2 Interfaces do caso de uso “1”

2.2.2.3 Interfaces do caso de uso “2”

2.2.2.4 Interfaces do caso de uso “3”

2.3 Metodologia

Como conta para nota, tenho requisitos a cumprir, e por isso, não posso fazer isto de qualquer forma. Vou ter que usar Programação Orientada a Objetos (POO) para criar uma RESTful API e um sistema MVC (se não sabes de o que é que estou a falar, vou tentar explicar no próximo parágrafo). Tive que escolher entre a plataforma (Laravel, 2025) e a (Microsoft, 2025). Apesar de hoje em dia ASP.NET Core ser de código aberto, e eu sei que ia ser possível eu conseguir fazer o trabalho no meu Arch Linux¹, continua a ser do oligarca digital que é a Microsoft. Estarás mais sujeito a depender de algum software ou serviço do ecossistema da Microsoft que não é nem código aberto e muito menos FLOSS. Aliás, o que está no enunciado do projeto e o que dá a perceber é que se a implementação do projeto for com ASP.NET, eu teria que usar — vou ter que censurar — M***soft SQL S***ver. Consegui fazer a cadeira de Bases de Dados 1 sem ter que mexer nesse pedaço de software maligno e escapei de fazê-lo em Bases de Dados 2 porque fiz intercâmbio numa escola que não me obrigou a usar esse software. Nós não queremos Microsoft nas nossas vidas. Não é como se ao escolher Laravel eu estivesse em melhores mãos porque a recomendação é MySQL da Oracle, outro monopólio digital imperialista. MySQL tem a versão *Community* que é FLOSS, e a versão *Enterprise* que é proprietária. A chance de eu ter que usar as extensões proprietárias é mínima, mas esse modelo de múltiplo licenciamento não quero suportar. Pessoas trabalham de graça ao contribuir para a versão FLOSS e depois chega a Oracle que explora esse trabalho em nome do capital. Para quem depende dessas extensões proprietárias, agora estás preso — *vendor lock-in*. Mas bem... vou usar a *fork* (MariaDB Foundation, 2025). Eu também já usei (The PHP Group, 2025) antes. É uma experiência agradável mesmo sem a tipagem estática, e oiço falar de como Laravel tem um ótimo ecossistema e oferece uma ótima *developer experience* (DX), como

¹Tsoding, 2025.

(Inertia.js, 2025), que vai permitir-me usar (Svelte, 2025) para a parte das Views do sistema. Tem que existir autenticação e um sistema de data analytics.

POO é um paradigma onde tu tens objetos, instâncias de classes. Uma classe é como se fosse uma receita para criar as instâncias, que propriedades e comportamentos elas terão. Depois cada instância do objeto é dona do seu estado que não depende do estado dos outros objetos da mesma classe. Eu e tu (assumindo que também és um Homo Sapiens Sapiens) ambos somos instâncias da mesma "classe" mas com propriedades diferentes. Algumas dessas propriedades até podem ser outros objetos, até de classes diferentes. Suponho que tu tenhas um smartphone — eu sei que tenho um. Tem a possibilidade de até termos o mesmo modelo de smartphone, mas as propriedades individuais de ambos os smartphones com certeza já são diferentes. Outra coisa fixe é que objetos conseguem mandar mensagens uns aos outros. Se eu carrego no botão para aumentar o volume do meu telemóvel ele há de reagir de alguma forma (aumentando o volume espero eu). No teu telemóvel deve acontecer o mesmo, mas não significa que aconteça da mesma forma. Cada objeto instância da mesma classe, por ter sido usada a mesma receita, oferecem os mesmos comportamentos, mas as propriedades por terem valores diferentes, podem afetar a execução do comportamento. No final, é esperado o mesmo resultado. Pensa assim, eu e tu provavelmente se nos mandassem vestir as mesmas meias e sapatos, talvez eu comece na esquerda e tu na direita, depois eu meto primeiro todas as meias e finalmente calço os 2 sapatos, e tu calças o sapato logo depois de cada meia. Então essas mensagens podem ser enviadas para qualquer tipo de objeto, instância da sua classe ou não, até para o próprio objeto. As mensagens também podem ser enviadas e recebidas de forma paralela.

API é uma interface que permite comunicação entre várias máquinas e/ou vários programas de software. Essa interface oferece um serviço de um programa de software de forma padronizada. O exemplo do botão do telemóvel, esse botão é como uma interface.

REST é uma lista de princípios de estilo arquitetónico para software. Esses princípios são na realidade restrições que limitam como um sistema de *hypermedia* distribuído deve agir. Podes estar agora a pensar que criar restrições só deve complicar as coisas. Mas a verdade é que estas restrições criam um sistema sem estado (*stateless*) e fiável. Interfaces uniformes, fácil separação de software em componentes menores, mais fáceis de manter. Se o teu sistema não quebra nenhuma das restrições REST, então ele é *RESTful*. Um dos pais do Hypertext Transfer Protocol (HTTP) foi quem surgiu com a ideia do REST, na sua dissertação de mestrado (Fielding, 2000).

Uma API *RESTful* é uma API para a Web que é *RESTful*. Existem muitas formas de criar uma API *RESTful*. O cliente pede um recurso de *hypermedia* e recebe uma representação do estado do recurso que ele quis. Quando a API *RESTful* é desenvolvida em cima do protocolo HTTP, o método e o cabeçalho do *request* pode servir como forma de dar informação sobre tu e o teu *request*, lembrando que a comunicação servidor–cliente é proibida de guardar estado entre *requests*.

MVC é uma padronização de arquitetura de software usando o paradigma de POO falado em cima, que separa partes do código em 3 partes, cada uma com as suas responsabilidades: o modelo, a vista, e o controlador. Um modelo representa algum componente ou parte do sistema. As vistas são representações

visuais dos modelos. O controlador recebe mensagens do utilizador e manda-as tanto para os modelos que possivelmente mandam mensagens para as vistas o que pode modificar a informação que o utilizador vê (manda uma mensagem ao utilizador).

Nesta etapa eu vou caracterizar os atores do sistema e os casos de uso com que eles interagem, usando *Unified Modeling Language* (UML). Para isso, usei o meu fiel (PlantUML Team, 2025) com o documento PDF da última especificação do UML² aberta.

Também querem um protótipo de baixa / média fidelidade das interfaces. Para isso, o mais inteligente é só eu já começar a escrever o meu HTML, ou neste caso, componentes do Svelte. Poupas trabalho para a próxima etapa.

Na minha opinião, quando tu não sabes como vai ser o teu projeto, mais vale começar logo a programar, e não perder tempo com diagramas e coisas que, depois que começas a programar, vais querer trocar. É uma perda de tempo. Tipo *Test-driven development* (TDD).

Para quê que tu queres «uma visão com baixo detalhe do esquema de apresentação das páginas» quando podes ter algo feito com HTML 5 semântico e que já te mostra «a forma como o utilizador interage» com as páginas Web e âncoras?

Matar dois coelhos de uma só cacheirada.

Depois gravo a tela e mando os vídeos como anexo junto ao relatório.

O professor também pede o modelo ER que eu nunca sei como fazer porque cada professor faz de forma diferente e não existe uma padronização para eu seguir. O PlantUML vai ajudar-me outra vez de qualquer forma.

O interessante deste projeto é que só existe um ator. Mas a interação entre vários atores cria uma certa complexidade, especialmente na participação democrática, que trás desafios já que todos os atores têm o mesmo poder.

O que também é de pensar é a questão da autenticação, porque dependendo de como ela é feita, pode acabar com o sistema de democracia direta.

2.3.1 Ferramentas

Exemplo: Git, Linux, L^AT_EX, Python, Docker.

2.3.2 Arquitetura do sistema

2.4 Resultados

Apresente resultados, gráficos e tabelas.

²*Unified Modeling Language: Superstructure, Version 2.5.1, 2017.*

3 Conclusão

3.1 Trabalho Futuro

Referências Bibliográficas

- ANSOL - Associação Nacional para o Software Livre. (2025). *O que é o Software Livre?* Obtido outubro 19, 2025, de <https://ansol.org/software-livre/>
- Decidim Community. (2025). *Decidim: A digital platform for citizen participation*. <https://decidim.org/>
- Defesa dos Direitos Digitais. (2021). *Vamos reflectir sobre... o Voto Electrónico*. Obtido outubro 19, 2025, de <https://direitosdigitais.pt/noticias/76-vamos-reflectir-sobre-o-voto-electronico>
- Defesa dos Direitos Digitais. (2023). *Twitter e Mastodon 101: plataformas vs protocolos*. Obtido outubro 19, 2025, de <https://direitosdigitais.pt/noticias/138-twitter-e-mastodon-101-plataformas-vs-protocolos>
- Fedigov. (2025). *Fedigov*. Obtido outubro 19, 2025, de <https://fedigov.pt/>
- Fielding, R. T. (2000). *Architectural Styles and the Design of Network-Based Software Architectures* [tese de doutoramento, University of California, Irvine] [Doctoral dissertation]. Obtido outubro 19, 2025, de https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- Free Software Foundation. (2025a). *What is free software and why is it so important for society?* Obtido outubro 19, 2025, de <https://www.fsf.org/about/what-is-free-software>
- Free Software Foundation. (2025b). *What is Free Software?* Obtido outubro 19, 2025, de <https://www.gnu.org/philosophy/free-sw.html>
- GO FAIR Initiative. (2025). *FAIR Principles*. <https://www.go-fair.org/fair-principles/>
- Guimarães, G. (2025, outubro). *Pitágoras não cobrou royalties (e outras lições sobre diversidade e partilha)* [Presented at Festa do Software Livre 2025, Auditório, Porto, Portugal, 3 October 2025]. ANSOL - Associação Nacional para o Software Livre. Obtido outubro 19, 2025, de <https://youtu.be/mSrDy4x7Paw?t=24305>
- Inertia.js. (2025). *Inertia.js - Modern monolithic apps without the SPA complexity*. Obtido outubro 19, 2025, de <https://inertiajs.com/>
- James G. Kim and Michael Hausenblas. (2025). *5-star Open Data*. <https://5stardata.info/en/>
- Laravel. (2025). *Laravel - The PHP Framework For Web Artisans*. Obtido outubro 19, 2025, de <https://laravel.com/>
- MariaDB Foundation. (2025). *MariaDB Foundation - MariaDB.org*. Obtido outubro 19, 2025, de <https://mariadb.org/>

- Microsoft. (2025). *ASP.NET Core, an open-source web development framework*. Obtido outubro 19, 2025, de <https://dotnet.microsoft.com/en-us/apps/aspnet>
- PlantUML Team. (2025). *PlantUML - Create UML diagrams from plain text*. <https://plantuml.com/>
- Svelte. (2025). *Svelte - Cybernetically enhanced web apps*. Obtido outubro 19, 2025, de <https://svelte.dev/>
- The PHP Group. (2025). *PHP: Hypertext Preprocessor*. Obtido outubro 19, 2025, de <https://www.php.net/>
- Tsoding. (2025, maio). *I tried .NET on Linux and Died Inside*. Obtido outubro 19, 2025, de <https://www.youtube.com/watch?v=TH5118PhzYs>
- Unified Modeling Language: Superstructure, Version 2.5.1*. (2017). Object Management Group. Needham, MA, USA. <https://www.omg.org/spec/UML/2.5.1/PDF>
- Voca. (2025). *Voca: A democratic space for your community*. <https://voca.city/>

Licença

Este documento está licenciado sob uma Licença Creative Commons Atribuição–Partilha nos Mesmos Termos 4.0 Internacional (CC BY-SA 4.0).

O código fonte (ficheiros `.tex`, `.bib`, `Makefile`, etc.) utilizado para produzir este relatório está licenciado sob a GNU Affero General Public License v3.0 (AGPL v3).

A Apêndice A: Questionário

Conteúdo do apêndice (ex.: questionários, listagens de dados).

B Apêndice B: Código de Exemplo

```
1 def hello():  
2     print("Olá, mundo!")
```

Anexos

Anexo A: Documento Externo

Texto de anexo.