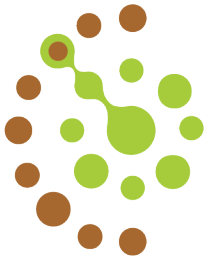

IPBeja
Instituto Politécnico de Beja

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

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

Lista de Figuras

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

[1] Defesa dos Direitos Digitais, 2021.
[2] ANSOL - Associação Nacional para o Software Livre, 2025.
[3] Free Software Foundation, 2025a.
[4] Free Software Foundation, 2025b.
[5] Defesa dos Direitos Digitais, 2023.
[6] Fedigov, 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 puder 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 e 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 perspetivas de evolução futura.

2 Desenvolvimento

2.1 Análise do Sistema

2.1.1 Caracterização dos atores

Como talvez já deu para preceber, só existe um ator, coletivo, universal, e humano. Sem importância para a sua raça, etnia, cultura e identidade. Sem olhar para a carteira. No entanto, é de esperar que a pessoa tenha alguma prática com a Web 2.0 para conseguir usar o software.

Alguém que gosta de conviver com outras pessoas, gosta de partilhar conhecimento e que conhecimento seja partilhado consigo.

Talvez ainda dê para encaixar mais um ator que seria exclusivamente do sistema: o banco de bens.

2.1.2 Diagrama de Casos de Uso

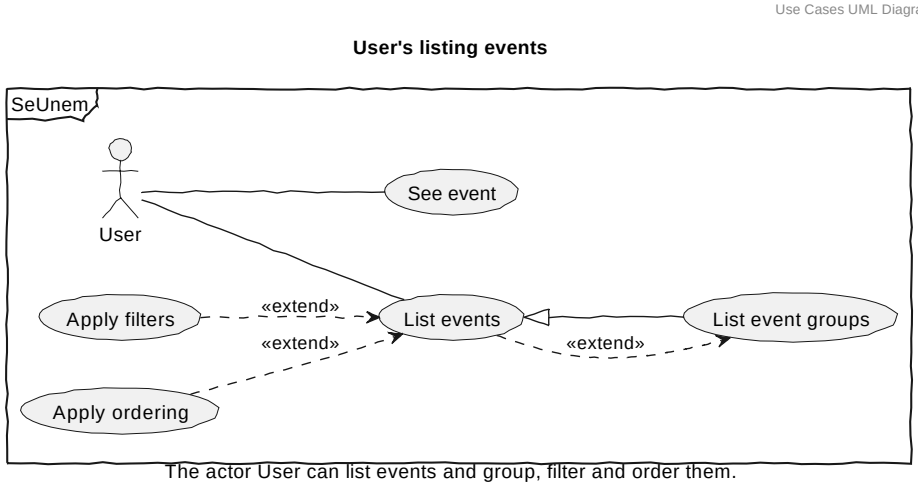

Figura 2.1*Diagrama UML dos casos de uso: O actor User listar eventos e agrupá-los, filtrá-los e ordená-los.*

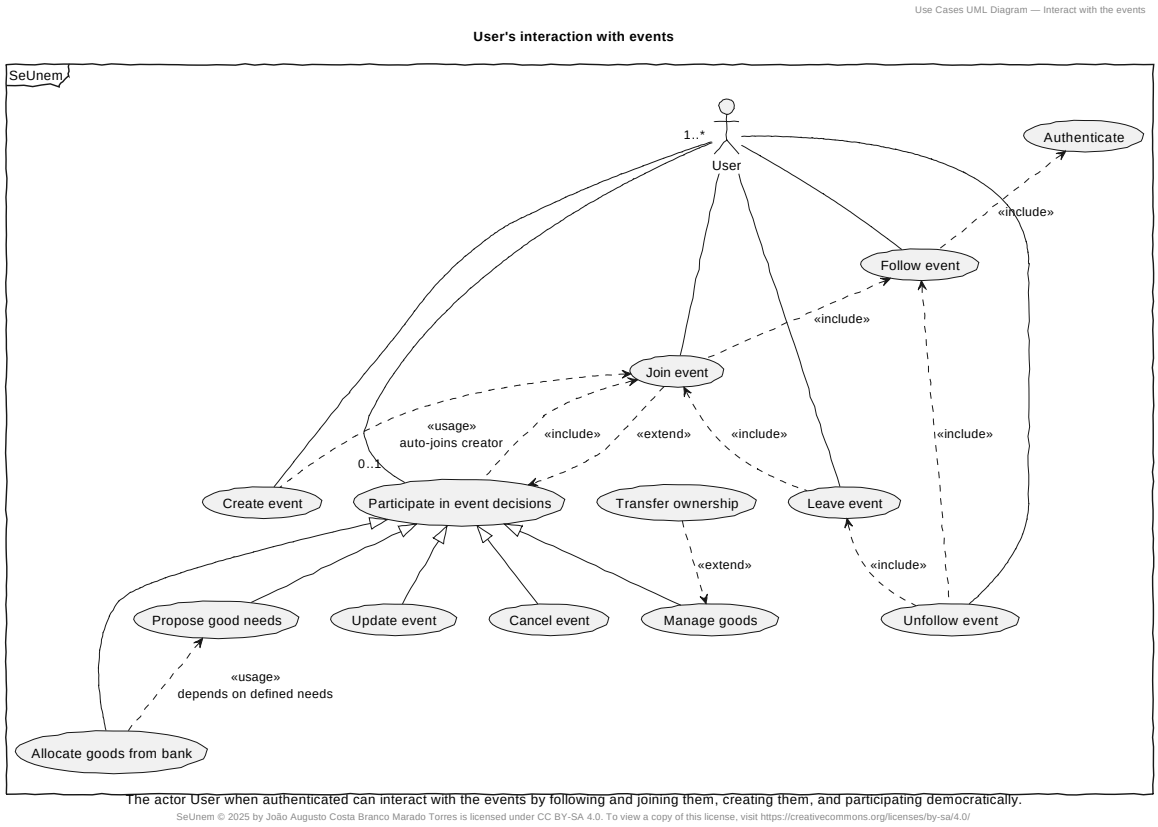

Figura 2.2*Diagrama UML dos casos de uso: O actor User quando autenticado pode interagir com eventos seguindo-os e juntando-se a eles, criando-os, e participando democráticamente.*

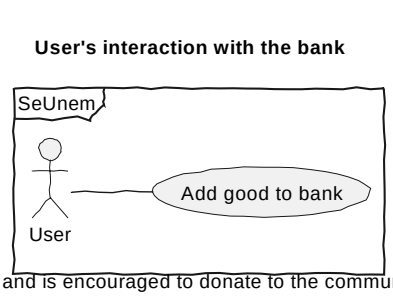

Figura 2.3*Diagrama UML dos casos de uso: O actor User pode e é encorajado a doar para o banco comun de bens.*

2.2 Desenho do Sistema

2.2.1 Modelação da Base Dados

2.2.1.1 Diagrama E/R & Modelo Lógico Relacional

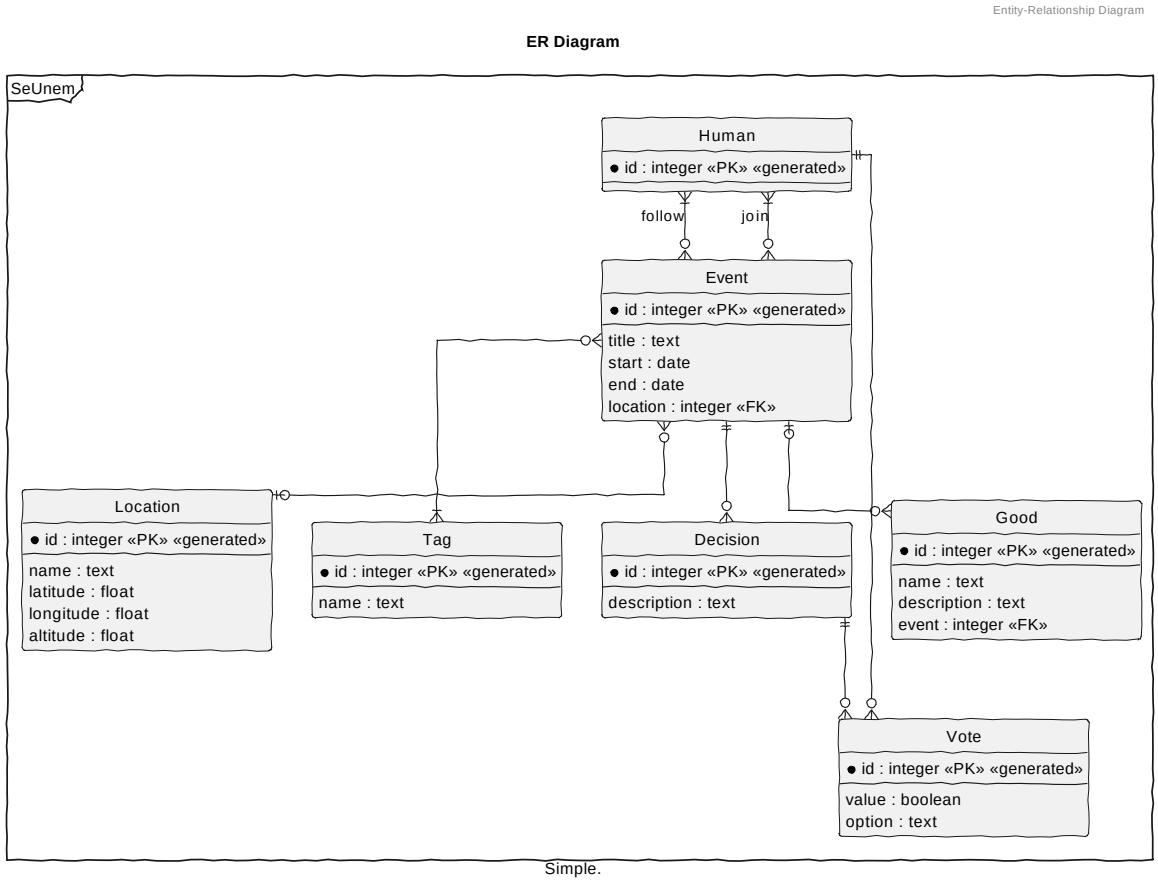

Figura 2.4*Modelo Entidade Relação.*

2.2.2 Modelação de Interfaces Gráficas com o Utilizador

Os protótipos de baixa e média fidelidade (*wireframes*) foram criados diretamente com `Svelte`, de forma a garantir uma representação fiel da interação entre o utilizador e o sistema. Em vez de usar ferramentas externas de prototipagem, optei por começar a desenvolver a estrutura das interfaces reais, seguindo o princípio de que um protótipo executável é preferível a um diagrama estático.

Todo o código-fonte correspondente encontra-se disponível no repositório (Costa Branco Marado Torres, 2025). Para reproduzir localmente os protótipos, podem seguir-se os seguintes passos A.

Após estes comandos, as interfaces podem ser visualizadas localmente através do servidor de desenvolvimento. A versão deste commit (4f0d1c9) contém a estrutura de páginas que serviu de base para a análise e desenho do sistema.

A opção por utilizar `Svelte` e `Inertia.js` (em vez de uma ferramenta de prototipagem tradicional) permitiu uma abordagem mais *pragmática*, poupando tempo e assegurando a consistência entre o protótipo e o código final da aplicação.

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 Linux[1], 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. Mas caraças, na cadeira de Sistemas de Informação também usam (junto com o Microsoft Excel e outro que se chama Microsoft Power BI). Eu não tenho Windows (e ainda bem) no meu portátil por isso só consigo fazer trabalhos que usem isso nos PCs da escola. 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 (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.

[1] Tsoding, 2025.

Outra cena 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* distribuido 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 fácies 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 maestrado (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 formar de dar informação sobre tu e o teu *request*, lembrando que a comunicação servidor–cliente é proibída 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 UML[2] 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

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

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 programas, 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

Para o desenvolvimento do projeto *SeUnem*, foram utilizadas diversas ferramentas livres e de código aberto, tanto para a implementação do sistema como para a documentação e modelação. Entre as principais, destacam-se:

- **Git** — Sistema de controlo de versões utilizado para gerir o histórico de desenvolvimento e facilitar o trabalho colaborativo.
- **Make** — Ferramenta de automatização utilizada para compilar e gerar a documentação técnica (diagramas e relatórios).
- **PlantUML** — Utilizada para a modelação dos diagramas UML, incluindo casos de uso e entidades-relacionamentos.
- **LaTeX** — Usado na redação e formatação do relatório técnico, com o auxílio do pacote `minted` para a inclusão de código.
- **PHP** — Linguagem de programação principal do *backend*.
- **Laravel** — Framework PHP utilizada para estruturar a aplicação no lado do servidor, gerir as rotas e aceder à base de dados.

- **Composer** — Gestor de dependências do ecossistema PHP.
- **Node.js** e **pnpm** — Usados para gerir as dependências do *frontend*.
- **Svelte** — Framework *frontend* escolhida pela sua simplicidade e eficiência, integrada através de `Inertia.js`.
- **Vite** — Empregado como ferramenta de construção (*build tool*) e servidor de desenvolvimento.
- **ESLint** e **Prettier** — Para assegurar a qualidade e consistência do código no lado do cliente.
- **MariaDB** — Base de dados leve usada durante o desenvolvimento e prototipagem inicial.

2.3.2 Arquitetura do Sistema

A aplicação *SeUnem* adota uma arquitetura de três camadas, composta por:

- **Camada de Apresentação (Frontend)** — Implementada em `Svelte`, comunica com o servidor através do `Inertia.js`, permitindo renderização dinâmica sem necessidade de uma API REST completa. Esta camada é responsável pela interação com o utilizador e pela exibição das informações de eventos, decisões e bens comunitários.
- **Camada Lógica (Backend)** — Desenvolvida em `Laravel`, contém a lógica de negócio, autenticação, gestão de permissões e processos democráticos de decisão. O backend também gere a comunicação entre as outras camadas e implementa os princípios de autogestão e participação coletiva.
- **Camada de Dados (Base de Dados)** — `MariaDB`, é responsável pelo armazenamento persistente de entidades como utilizadores, eventos, decisões, votos e bens.

A comunicação entre o *frontend* e o *backend* ocorre através de requisições HTTP geridas pelo `Inertia.js`, evitando a duplicação de lógica e garantindo uma experiência fluida de aplicação de página única (*SPA — Single Page Application*). O sistema foi concebido de forma modular, permitindo a extensão futura para novos tipos de entidades ou modos de participação.

3 Conclusão

O desenvolvimento do projeto *SeUnem* permitiu consolidar conhecimentos adquiridos ao longo da licenciatura, nomeadamente no âmbito da análise, desenho e implementação de sistemas Web baseados em paradigmas modernos de engenharia de software. Esta fase de análise e desenho focou-se sobretudo em definir a estrutura conceitual da aplicação, os atores envolvidos, os casos de uso principais e o modelo de dados subjacente.

O sistema proposto tem como propósito promover a participação democrática, autogestão e partilha comunitária através de software livre e descentralizado. A escolha de tecnologias e ferramentas FLOSS garantiu coerência com a visão ideológica do projeto — de que a tecnologia deve servir o coletivo e não o contrário. A arquitetura em três camadas (frontend, backend e base de dados) possibilita a evolução modular da plataforma, assegurando escalabilidade e manutenção a longo prazo.

Entre as limitações encontradas, destaca-se o tempo reduzido para a implementação efetiva das funcionalidades democráticas mais complexas (como o processo deliberativo e de votação distribuída). Também se reconhece a necessidade de aprofundar mecanismos de autenticação e privacidade que preservem o anonimato sem comprometer a integridade da participação.

De forma geral, os objetivos desta etapa foram atingidos: elaboraram-se os diagramas de casos de uso e E/R, a modelação da arquitetura, e os protótipos de baixa/média fidelidade, demonstrando a viabilidade técnica e conceptual do *SeUnem*.

3.1 Trabalho Futuro

Numa fase futura, pretende-se evoluir o projeto no sentido de:

- Implementar a totalidade da aplicação funcional baseada nos modelos definidos nesta fase;
- Desenvolver mecanismos de deliberação e votação descentralizada, com foco em transparência e auditabilidade pública;
- Melhorar a acessibilidade (*a11y*) e a internacionalização (*i18n*), tornando a plataforma inclusiva e global;

- Integrar protocolos federados para comunicação entre instâncias do *SeUnem*, rumo a uma rede de comunidades autogeridas;
- Criar um módulo de *analytics* participativo para visualizar a utilização dos bens e recursos comunitários;
- Estudar novas formas de autenticação descentralizada e mecanismos de reputação não-hierárquica.

Assim, o *SeUnem* continuará a ser um espaço experimental e tecnopolítico, orientado para repensar as práticas digitais de organização coletiva e participação democrática, sempre com base em software livre e valores de autonomia e partilha.

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/

Costa Branco Marado Torres, J. A. (2025, outubro). *SeUnem* (Versão 0.1.0). https://github.com/torres-engineer/SeUnem/

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

A Apêndice: Linha de comandos

```
# Clonar o repositório
git clone https://github.com/torres-engineer/SeUnem.git seunem
cd seunem

# Alterar para o commit específico dos wireframes
git checkout 4f0d1c9

# Instalar as dependências do frontend e construir os ficheiros
pnpm install
pnpm build

# Iniciar o ambiente de desenvolvimento do Laravel
composer run dev
```