SPOTTER

Documentação do Sprint 0

Discovery, setup e aprendizagem

1 de julho de 2026 a 8 de julho de 2026

Miguel Jordão | Product Owner

Índice

1. Metadados

Campo	Valor
Projeto	Spotter
Sprint	Sprint 0
Período	1 de julho de 2026 a 8 de julho de 2026
Estado no Sprint Tracker	Completed
Tipo de documento	Documentação de sprint
Fonte principal	Notion do projeto Spotter
Autor / Product Owner	Miguel Jordão
Versão	0.5
Última atualização	2 de agosto de 2026

Este relatório documenta o Sprint 0 do Spotter como registo interno, material de apoio à equipa e base para documentação futura de portefólio. O Notion é a fonte de verdade deste documento. Os estados apresentados no inventário referem-se ao fecho do Sprint 0: tarefas concluídas como `Done` e tarefas não assumidas como `Ice Box`.

2. Resumo

O Sprint 0 foi a fase de preparação do Spotter. A equipa partiu de uma ideia ainda aberta, ajudar estudantes do Técnico a encontrar espaços de estudo, e fechou uma base inicial de produto, tecnologia, dados, design, organização e aprendizagem.

No fim do Sprint 0, o projeto tinha um problema validado por dados iniciais, um product brief, personas preliminares, user stories, backlog inicial do Sprint 1, métricas de sucesso, objetivo do Sprint 1, stack técnica escolhida, arquitetura inicial, modelo de dados, contrato de API, exploração da API do Fénix, wireframes, design system base, matriz de estados e regras de trabalho no Notion.

A decisão de produto mais importante foi separar o Sprint 1 do MVP completo. O Sprint 1 ficou focado no Core Room Finder com disponibilidade baseada em dados oficiais. O MVP de verão ficou definido como Sprint 1 + Sprint 2, deixando reports de alunos, recência, expiração e combinação de dados oficiais com reports para a segunda metade do projeto.

O Sprint 0 também expôs riscos importantes. A API do Fénix fornece dados úteis, mas não fornece diretamente "salas livres agora"; a disponibilidade tem de ser calculada a partir de eventos. O TIC é relevante para os alunos, mas não tem dados estruturados suficientes para ocupação real. Alguns eventos aparecem sobrepostos e exigem filtragem/tratamento antes do cálculo de disponibilidade. A amostra inicial também é limitada, por estar muito concentrada na Alameda e em alunos de 1.º ano.

3. Contexto do Projeto

O Spotter é uma web app mobile-first para estudantes do Técnico encontrarem salas e espaços onde possam estudar no campus. O problema inicial é prático: quando o local habitual está cheio, muitos alunos não sabem rapidamente que salas estão livres, onde ficam, durante quanto tempo estarão livres e quão fiável é essa informação.

O projeto de verão foi organizado em três momentos. O Sprint 0 preparou a base: discovery, setup, validação inicial, aprendizagem e planeamento. O Sprint 1 ficou destinado ao Core Room Finder com dados oficiais. O Sprint 2 ficou destinado a reports de alunos e funcionalidades complementares para fechar o MVP.

Esta divisão impediu que a equipa tentasse construir todos os elementos de uma vez. A primeira versão funcional precisava de responder bem a uma pergunta central: "onde posso estudar agora ou num intervalo próximo?". Os reports são relevantes, mas aumentam complexidade de produto, UX, frontend, backend e validação. Por isso, foram planeados para o Sprint 2.

4. Metodologia

O projeto usou uma adaptação prática de Scrum/Agile. A equipa trabalhou com sprints, Product Backlog, tarefas com owners, prioridades, pontos, deliverables e acceptance criteria. Esta estrutura foi especialmente importante porque a equipa era júnior e precisava de transformar trabalho aberto em unidades pequenas, verificáveis e atribuíveis.

No Notion, cada tarefa passou por um ciclo simples: `New Issues` ou planeamento inicial, `Backlog`, `In Progress`, `In Review` e `Done`. O estado `Ice Box` foi usado para tarefas que existiam como hipótese, mas que não entraram no scope concluído do sprint.

Cada tarefa tinha código, título, responsável, tipo, prioridade, pontos, deliverable e acceptance criteria. Esta estrutura serviu para dividir trabalho, explicar o que tinha de ser feito, registar porque certas decisões foram tomadas, guardar evidência do resultado e facilitar revisão posterior. As tarefas também funcionam como histórico do projeto: depois de um sprint terminar, o conteúdo fechado deve manter-se estável; se for preciso corrigir ou refinar uma decisão, cria-se uma nova tarefa de revisão num sprint seguinte.

O processo de review foi parte central da metodologia. Antes de passar para `Done`, uma tarefa devia ser revista por alguém externo à execução. Normalmente, o Product Owner revia as tarefas dos outros membros. As tarefas do Product Owner deviam ser revistas por outro membro da equipa. Esta regra reduziu ambiguidade, aumentou a qualidade da documentação e obrigou cada owner a mostrar evidência clara do que tinha sido feito.

5. Objetivos do Sprint 0

O Sprint 0 foi desenhado como fase de discovery, setup, aprendizagem e aquecimento. A equipa era composta por alunos de 1.º ano, com experiência limitada fora de projetos académicos. Antes de tentar construir uma web app completa, era necessário aprender ferramentas, perceber dados oficiais, experimentar stack e definir responsabilidades para evitar desenvolvimento às cegas.

O objetivo de produto era validar minimamente o problema, definir público-alvo, proposta de valor, personas preliminares, user stories, métricas e scope do MVP de verão. O objetivo não era provar estatisticamente o mercado inteiro, mas confirmar se havia sinais fortes o suficiente para avançar com uma primeira versão.

O objetivo de UX e Frontend era ganhar bases para transformar a ideia em ecrãs e depois em interface funcional. Isto incluiu aprender a utilizar softwares como Figma, React, Vite, Tailwind e aprender termos como mobile-first, wireframes, design system, estados visuais, componentes e QA.

O objetivo de Backend e Fénix Integration era perceber se os dados oficiais podiam suportar o Core Room Finder e definir a stack do projeto. Isto exigiu aprender como funcionam APIs, endpoints, Django, PostgreSQL, modelos de dados, ingestão, normalização e limitações dos dados do Fénix.

O objetivo de equipa era tornar o grupo confortável com uma forma de trabalho mais próxima de produto real: tarefas pequenas, owners claros, reviews, handoffs e documentação contínua. O critério de sucesso do Sprint 0 era chegar ao Sprint 1 com contexto suficiente, tarefas executáveis, stack escolhida, riscos conhecidos e bases mínimas para começar desenvolvimento.

6. Equipa e Síntese do Trabalho por Área

Tabela 1: Equipa do projeto Spotter no Sprint 0

Pessoa	Papel no projeto
Miguel Jordão	Product Owner
Diogo Monteiro	Backend
Ricardo Vicente	Fénix Integration
Francisco Frieza	Frontend Features
Maria Carolina Vidal	Frontend UX

Miguel conduziu o trabalho de Product Owner: discovery, análise das respostas, definição inicial de produto, personas, user stories, backlog inicial do Sprint 1, métricas e objetivo do Sprint 1. Esta frente criou a ligação entre problema, utilizador, scope e execução.

Diogo ficou responsável pela base de Backend: escolha da stack, arquitetura, modelo de dados, estrutura Django, spike técnico e investigação de deploy. Esta frente definiu a base do funcionamento do produto.

Ricardo ficou responsável por Fénix Integration: mapear fontes, testar endpoints, criar dicionário de dados, avaliar qualidade/cobertura, criar script exploratório, propor lógica de disponibilidade e recomendar o primeiro conjunto de dados a suportar. Esta frente reduziu o risco principal do produto: depender de dados oficiais que não vinham prontos para o caso de uso do Spotter.

Francisco ficou responsável por Frontend Features: divisão funcional do trabalho de frontend, estrutura base, matriz de estados, checklist de QA e especificação da página de detalhe. Esta frente tratou do funcionamento do frontend.

Carolina ficou responsável por Frontend UX: referências, fluxo principal, wireframes, design system e estados visuais de disponibilidade. Esta frente desenhou a experiência inicial do utilizador.

7. Produto e Discovery

Discovery e Validação Inicial

O discovery começou com a preparação de um Microsoft Forms no âmbito de PO0-01. A intenção foi recolher evidência rápida, sem tornar o Sprint 0 demasiado pesado. Em vez de tentar uma validação estatisticamente representativa, Miguel procurou perceber se havia sinais suficientes de problema real para justificar o avanço do projeto.

Em PO0-02, foram recolhidas 18 respostas ao formulário. Todas correspondiam a estudantes do Técnico: 17 inquiridos eram da Alameda, 1 do Taguspark, 17 do 1.º ano e 1 do 2.º ano. Esta concentração limita a generalização dos resultados, mas é coerente com o objetivo do Sprint 0: reduzir incerteza inicial.

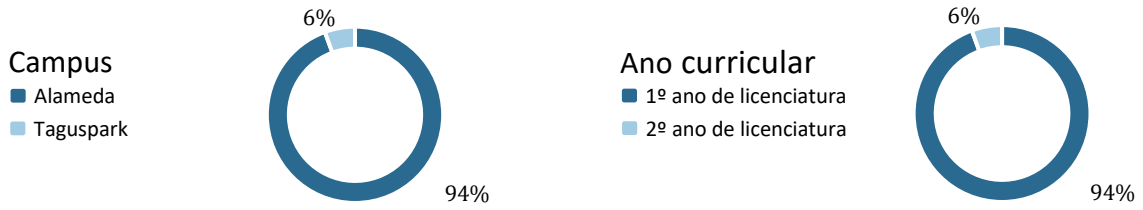

Figura 1: Caracterização da amostra do formulário por campus (gráfico da esquerda) e por ano curricular (gráfico da direita)

No âmbito de PO0-03, Miguel sintetizou os padrões mais relevantes. A dificuldade em encontrar lugar cresce em semanas críticas: nas semanas de MAPs, 11/18 responderam frequentemente ou muito frequentemente; em preparação/exames, o mesmo valor repetiu-se. Em semana normal, a dificuldade é menor, mas ainda aparece: 2/18 frequentemente e 8/18 às vezes.

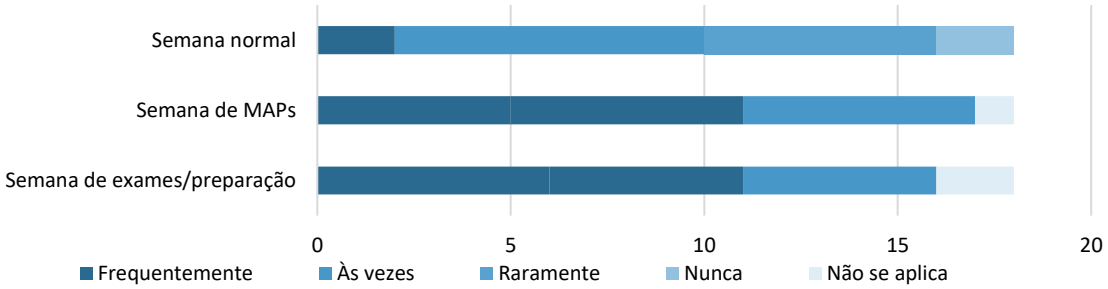

Figura 2: Frequência da dificuldade em encontrar lugar para estudar por período académico

O tempo perdido também apareceu como sinal forte. 9/18 estudantes indicaram perder entre 10 e 20 minutos à procura de lugar, e 1/18 indicou perder mais de 20 minutos. As alternativas atuais mostram que o problema não é apenas falta de espaço: alguns alunos procuram salas ao acaso, outros vão a sítios conhecidos, perguntam a colegas ou desistem de estudar no campus

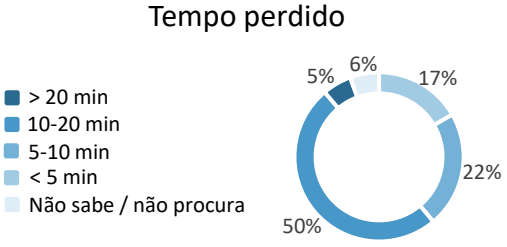

Figura 3: Tempo perdido na procura por lugar

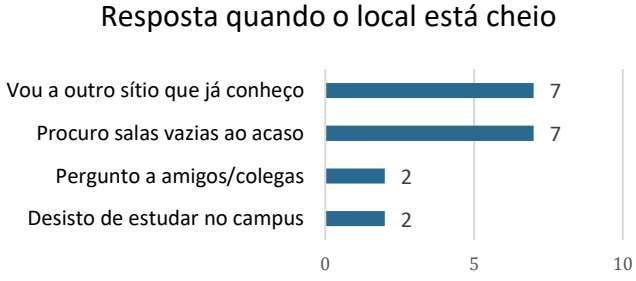

Figura 4: Alternativas usadas quando o espaço escolhido está cheio

O TIC apareceu como local central e problemático. 12/18 indicaram que estudam no TIC e 10/18 apontaram o TIC como problemático. Este dado influenciou decisões posteriores: mesmo com limitações de dados oficiais, o TIC não podia ser ignorado sem reduzir a utilidade percebida do produto.

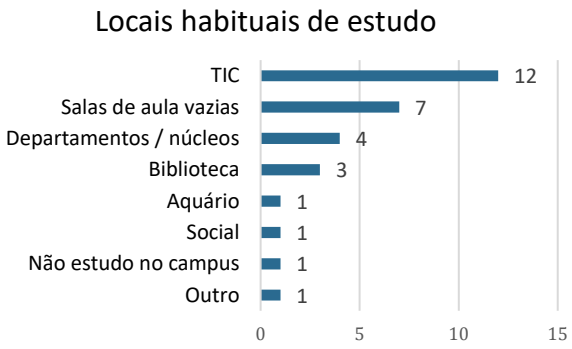

Figura 5: Locais habituais de estudo

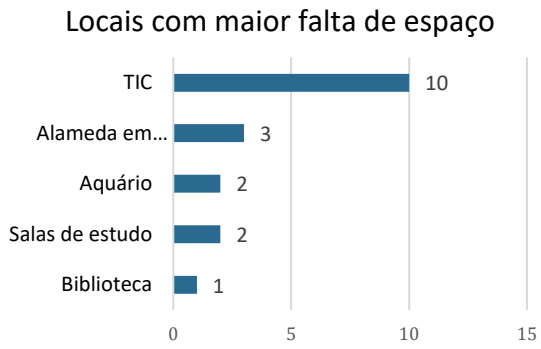

Figura 6: Espaços percecionados como mais problemáticos

A utilidade percebida foi alta. A utilidade de ver salas livres agora teve média 4,56/5, com 17/18 respostas em 4 ou 5. A probabilidade de usar o Spotter teve média 4,17/5, com 16/18 respostas em 4 ou 5. 16/18 estudantes disseram que reportariam ocupação se fosse muito simples.

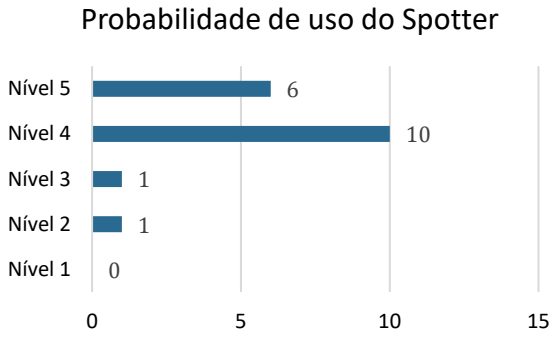

Figura 7: Probabilidade de uso do Spotter

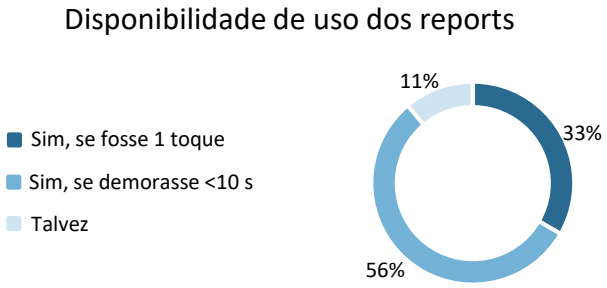

Figura 8: Disponibilidade para reportar ocupação

Definição Inicial do Produto

Em `P00-04`, Miguel consolidou a definição inicial do produto. O problema ficou formulado como perda de tempo e incerteza na procura de sítios para estudar no Técnico. O público-alvo inicial são estudantes que estudam no campus da Alameda, especialmente entre aulas, em semanas de MAPs/exames, quando o sítio habitual está cheio ou quando precisam de estudar em grupo.

A proposta de valor ficou centrada em ajudar estudantes a encontrar rapidamente salas disponíveis para estudar, combinando disponibilidade oficial com reports simples de alunos sobre ocupação real. Esta formulação já continha a visão completa do MVP, mas também obrigou a dividir bem o trabalho por sprints.

O MVP de verão ficou definido como Sprint 1 + Sprint 2. O Sprint 1 entrega a base oficial: lista de salas, estados de disponibilidade, filtros e detalhe. O Sprint 2 acrescenta reports, recência, expiração e combinação entre dados oficiais e reports.

Ficaram fora de scope nesta fase modo offline, integração em tempo real com TIC, restaurantes e espaços não académicos, versão em inglês, login, funcionalidades sociais avançadas e personalização avançada. Estes cortes foram importantes para manter o projeto executável durante o verão.

User Stories e Backlog Inicial

Em PO0-05, Miguel escreveu as user stories iniciais do Sprint 1 a partir do discovery. As histórias cobriam escolha de campus, lista de salas, filtros, pesquisa por intervalo horário, detalhe da sala, compreensão do estado de disponibilidade, estados de loading/vazio/erro e consulta baseada em dados oficiais.

Uma decisão relevante foi não esconder ocupadas/incertas por defeito. A lista deveria mostrar todas as salas relevantes do campus e ordenar por utilidade: livre agora, livre em breve, ocupado e incerto. Isto evita que uma lista vazia seja confundida com "não há salas" quando, na realidade, pode haver erro ou ausência de dados.

No âmbito de PO0-06, Miguel transformou as user stories e dependências técnicas num backlog inicial do Sprint 1. O Product Backlog ficou organizado por áreas, prioridades, pontos, deliverables e acceptance criteria. A distribuição ficou pesada, especialmente para Backend, Frontend Features e UX, mas refletiu a ambição do Sprint 1: entregar uma experiência mobile utilizável, ligada a dados oficiais e com estados robustos.

Ficou registado também que feedback real imparcial seria difícil logo após o Sprint 1 por causa do calendário de verão. Por isso, as métricas de sucesso definidas em PO0-07 foram pensadas como base para validação posterior, sobretudo depois do Sprint 2/beta.

Métricas e Objetivo do Sprint 1

Em PO0-07, Miguel definiu um conjunto curto de métricas: aberturas da app, aberturas de detalhe, escolha de sala e confiança na disponibilidade. Estas métricas não substituem validação real com utilizadores, mas dão uma estrutura mínima para avaliar se o Spotter pode ser usado e se ajuda na decisão de onde estudar.

Em PO0-08, ficou fechado o objetivo do Sprint 1:

> *No fim do Sprint 1, um aluno do Técnico deve conseguir abrir o Spotter no telemóvel, escolher ou usar um campus, ver salas com estado de disponibilidade baseado em dados oficiais, filtrar resultados e abrir o detalhe de uma sala para decidir para onde ir estudar.*

Este objetivo tornou o Sprint 1 mais concreto. A equipa não precisava de entregar o MVP completo, mas precisava de provar que o Core Room Finder funcionava de ponta a ponta: interface mobile, dados oficiais, filtros, estados e detalhe.

8. UX e Frontend

Pesquisa de Referências e Fluxo

O trabalho de UX começou com análise de referências em `UX0-01`. Foram observados produtos e padrões como MIIO e Google Maps, especialmente pela forma como representam lista, estado, filtros e detalhe. Estas referências ajudaram a perceber que o Spotter precisava de ser rápido, legível e orientado para decisão, em vez de parecer uma página informativa genérica.

Em `UX0-02`, Carolina mapeou o fluxo principal: abrir a app, escolher campus, ver salas ordenadas por disponibilidade, aplicar filtros e abrir detalhe. Este fluxo criou uma primeira narrativa de utilização e tornou mais claro o que o frontend teria de implementar no Sprint 1.

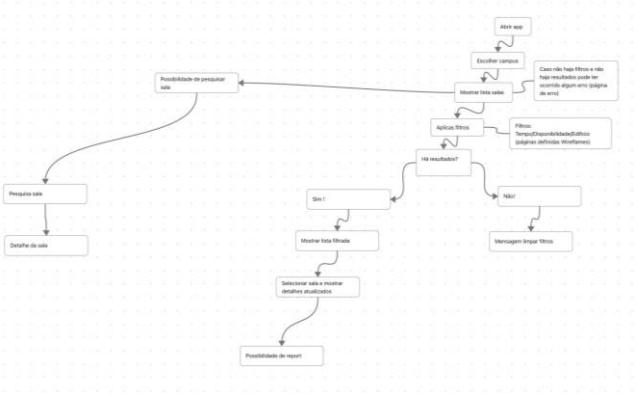

Figura 9: Diagrama do fluxo da aplicação

Wireframes e Protótipo Inicial

Em `UX0-03`, Carolina criou wireframes mobile-first para os ecrãs principais. Estes wireframes devem ser lidos como low-fidelity: serviram para alinhar ecrãs, informação e navegação, não como design final do produto.

O Figma foi o software escolhido para fazer as wireframes, uma vez que permitiria depois evoluir as wireframes para mockups e protótipos realistas e mais completos para o produto.

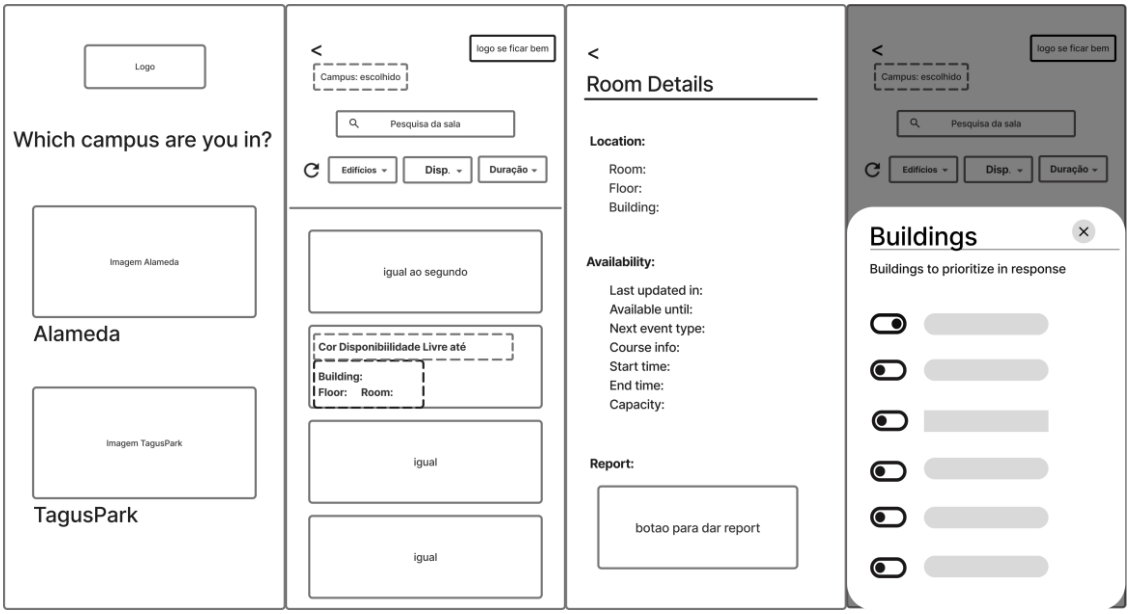

Figura 10: Wireframes do Spotter

Design System e Estados Visuais

Em `UX0-04`, Carolina definiu um design system inicial. A base ficou simples para não aumentar o risco de implementação: branco como cor dominante, azul `#2a6a90` como cor secundária, fonte Inter, cards por sala, faixa lateral de estado e radius de 6px.

Em `UX0-05`, foram definidos os estados visuais de disponibilidade. A interface não deveria depender apenas de cor e por isso cada estado precisava de texto associado. Os estados iniciais foram "Disponível", "Disponível em breve", "Ocupado" e "Incerto". A cor ajuda a leitura rápida, mas o texto protege acessibilidade e reduz ambiguidade.

Frontend Técnico, Repositório e Handoff

Em `FE0-01`, Carolina e Francisco alinharam a divisão entre Frontend UX e Frontend Features. Carolina ficou como owner visual/UX; Francisco ficou como owner funcional/técnico. Esta separação era necessária porque UX e implementação visual se cruzam facilmente em React/Tailwind.

Em `FE0-02`, Francisco criou a estrutura base do frontend. Ficaram definidos assets e componentes, com ficheiros JSX para compor a página principal. Esta base também incluiu aprendizagem prática através de vídeos sobre React, Tailwind, responsividade, navegação e estrutura de interface.

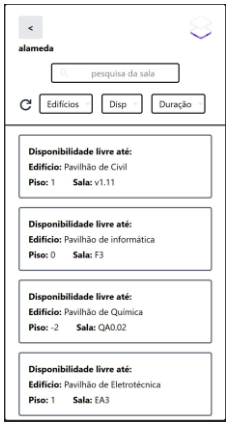

Figura 11: Captura de tela da página da lista de salas

Em `FE0-03`, Francisco criou a matriz de estados da interface, cobrindo loading, vazio, erro, retry, dados incompletos e sucesso. Esta tarefa foi importante porque uma app de disponibilidade pode parecer estragada quando apenas está sem dados ou à espera da API. Em `FE0-04`, Francisco preparou a checklist inicial de QA para antecipar como verificar user stories, estados normais e estados de erro.

Em `FE0-06`, Francisco especificou a página de detalhe para o Sprint 1: nome da sala, edifício/pavilhão, campus, evento atual ou estado livre, capacidade quando disponível e próximo evento. `FE0-05`, analytics, ficou em `Ice Box`: existia como tarefa Should Have, mas não entrou no scope concluído do Sprint 0.

Em `UX0-06`, Carolina e Francisco escolheram React + Vite + Tailwind como base, com shadcn/ui usado seletivamente. Esta decisão protegeu a separação entre frontend e backend e deu à equipa uma stack mais simples para aprender e aplicar no Sprint 1.

9. Backend, Dados e Integração Fénix

Stack e Arquitetura

Em `BE0-01`, Diogo comparou opções de backend e base de dados. A decisão foi usar Python/Django e PostgreSQL. Django foi escolhido pela velocidade de arranque, familiaridade com Python, ORM, admin e segurança base. PostgreSQL foi escolhido por encaixar naturalmente em dados relacionais e consultas temporais.

Em `BE0-02`, Diogo desenhou a arquitetura inicial. O fluxo conceptual ficou: Fénix API -> `ingestion` -> PostgreSQL -> `rooms` API -> frontend -> aluno. Esta arquitetura evita que cada pedido do frontend dependa de chamadas em tempo real ao Fénix. O Spotter deve ingerir dados oficiais em background, guardá-los na base de dados e responder ao frontend a partir da sua própria API.

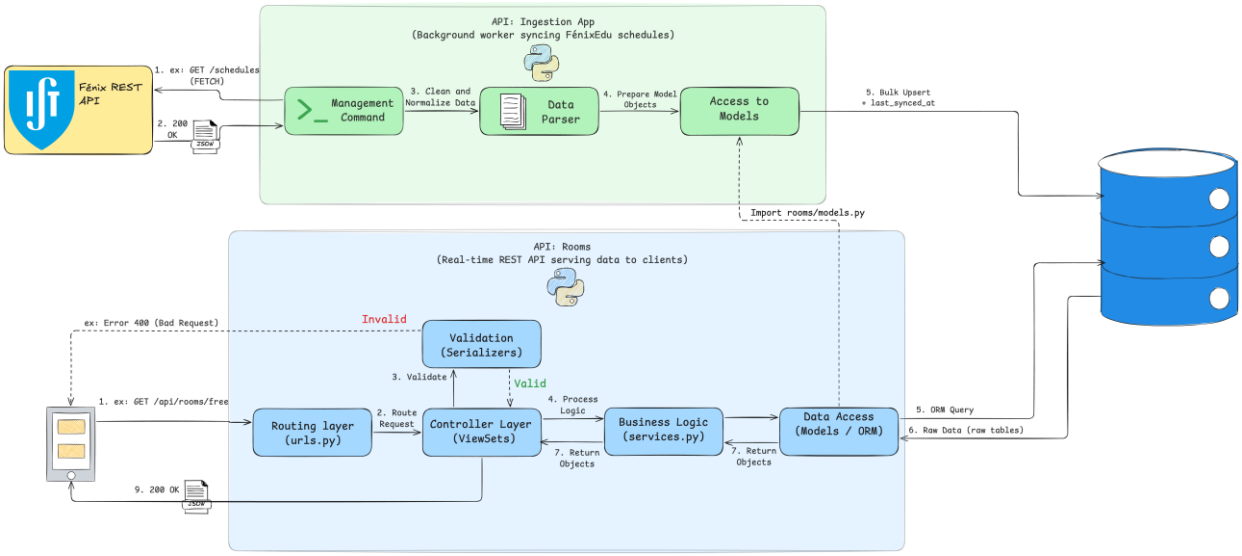

Figura 12: Arquitetura do Spotter

A app `rooms/` ficou associada à API consumida pelo frontend, cujo responsável é Diogo. A app `ingestion/` ficou associada à recolha e preparação dos dados oficiais, cujo responsável é Ricardo.

Modelo de Dados

Em `BE0-03`, Diogo propôs o modelo de dados inicial com `Building`, `Room` e `Event`. O modelo traduziu a estrutura mínima necessária para representar edifícios, salas, capacidade e ocupações oficiais.

Uma decisão importante foi não guardar `status`, `available_from`, `available_until` ou `next_event` como campos fixos. Estes valores dependem da hora consultada e devem ser calculados dinamicamente a partir dos eventos. Esta decisão reduziu risco de dados inconsistentes entre base de dados e resposta da API.

Contrato Inicial da API

Em `BE0-04`, Diogo e Ricardo definiram o contrato inicial da API para o Sprint 1. A tarefa foi partilhada porque os campos disponibilizados pelo backend dependiam diretamente dos dados que a integração Fénix conseguia obter e normalizar. No fim, os responsáveis do Frontend validaram o contrato.

Os endpoints previstos foram `GET /api/rooms/`, para listar salas com filtros e ordenação; `GET /api/rooms/<fenix_id>/`, para obter detalhe de uma sala; e `GET /api/buildings/`, para listar edifícios usados nos filtros. Os parâmetros previstos incluíam `campus`, identificador de edifício ou `fenix_id`, `search`, `status`, `date`, `start_time` e `end_time`.

> **Contrato da API inicial para a lista de salas:**
>
> ```
> {
> "fenix_id":"2448131362371",
> "name":"GA1",
> "building_name":"Pavilhão Central",
> "floor":"Piso 1",
> "campus":"Alameda",
> "normal_capacity":150,
> "status":"LIVRE_AGORA",
> "available_from":"2026-07-05T08:00:00Z",
> "available_until":"2026-07-05T14:30:00Z"
> }
> ```
>

Este contrato reduziu risco de integração com frontend. A equipa passou a ter uma referência comum para lista, detalhe, filtros, estados e campos de resposta. Também deixou claro que a ordenação por utilidade deveria ser responsabilidade do backend, porque depende da lógica de disponibilidade.

Integração Fénix e Qualidade dos Dados

Em `FI0-01`, Ricardo mapeou as fontes candidatas. A FenixEdu API foi identificada como fonte principal para espaços, salas e eventos oficiais. O projeto `ist-space-finder`, disponível em GitHub e associado ao `spaces.leic.pt`, foi identificado como referência técnica: uma solução anterior semelhante que ajuda a perceber como outros lidaram com dados de espaços do IST.

Em `FI0-02`, Ricardo testou o acesso às fontes prioritárias. Os endpoints públicos permitiam obter campus, edifícios, pisos, salas, capacidade e eventos oficiais. O fluxo base validado foi `GET /spaces`, `GET /spaces/{campusId}` e `GET /spaces/{roomId}?day=dd/mm/yyyy`.

Ficou claro que OAuth não era necessário para o Sprint 1. O produto não dependia de login, dados privados, horário pessoal ou cursos do utilizador. OAuth só faria sentido mais tarde se o Spotter passasse a usar dados privados do aluno.

Em `FI0-03`, Ricardo criou um dicionário inicial de dados para alinhar Fénix Integration, Backend e Product Owner. Esse dicionário ajudou a mapear `Building`, `Room` e `Event`, evitando interpretações diferentes dos mesmos campos.

Em `FI0-04`, Ricardo avaliou qualidade, cobertura e limitações. Esta foi uma das tarefas mais importantes do Sprint 0. A API oficial é útil, mas exige processamento: devolve muitos espaços que não são salas de estudo, alguns eventos aparecem sobrepostos, o parâmetro `day` pode devolver informação que precisa de ser filtrada por `event.day`, e pedir eventos sala a sala em tempo real é demasiado lento para a experiência pretendida.

Output script python:

```
--- US1: Escolher Campus ---
Campus disponíveis no Técnico:
  - Taguspark (ID: 2448131360898)
  - Tecnológico e Nuclear (ID: 2448131392438)
  - Alameda (ID: 2448131360897)

--- US2 & US3: Edifícios e Salas no Campus
Alameda ---
Encontrados 23 edifícios em Alameda.
Exemplos de edifícios (Filtros - US3):
  - Secção de Folhas (ID: 2448131377838)
  - Pavilhão do Jardim Norte (ID: 2448131361176)
  - Edifício de Civil (ID: 2448131361042)
  - Pavilhão Central (ID: 2448131361060)
  - Edifício de Minas (ID: 2448131361139)

Procurando salas no Pavilhão Central (ID:
2448131361060)...
Salas encontradas no Pavilhão Central: 107
  - Sala: Depósito (ID: 2448131362542)
  - Sala: Sala de Catalogaçao (ID: 2448131362541)
  - Sala: Sala de Leitura (ID: 2448131362539)
  - Sala: Museu (ID: 2448131362540)
  - Sala: Acesso ao Terraço (ID: 2448131362537)

--- US4, US5, US6 & US8: Detalhe e
Disponibilidade ---
Consultando eventos para a sala GA1 (ID:
2448131362371) no dia 06/10/2025...
[Detalhe da Sala (US5)]
  Nome:       GA1
  Descrição:  GA1
  Capacidade: 130 pessoas
```

```
Total de eventos no dia: 22
   - [LESSON] 16:30 - 18:30 | T
   - [LESSON] 13:00 - 14:00 | T
   - [LESSON] 11:30 - 13:30 | T
   - [LESSON] 10:00 - 11:30 | L
   - [LESSON] 14:30 - 16:00 | TP
   - [LESSON] 17:00 - 18:30 | TP
   - [LESSON] 14:00 - 16:00 | T
   - [LESSON] 08:00 - 10:00 | T
   - [LESSON] 08:00 - 10:00 | T
   - [LESSON] 14:00 - 16:00 | T
   - [LESSON] 13:00 - 15:00 | T
   - [LESSON] 16:00 - 18:00 | T
   - [LESSON] 16:00 - 18:00 | T
   - [LESSON] 13:00 - 14:30 | TP
   - [LESSON] 11:00 - 13:00 | T
   - [LESSON] 10:30 - 12:30 | T
   - [LESSON] 15:30 - 17:00 | TP
   - [LESSON] 08:00 - 09:30 | TP
   - [LESSON] 10:00 - 11:30 | TP
   - [LESSON] 10:00 - 11:30 | TP
   - [LESSON] 11:30 - 13:00 | TP
   - [LESSON] 13:30 - 15:30 | T

[Teste do Motor de Disponibilidade (US6)]
  As 09:00 -> Estado: OCUPADO         | Detalhe:
Ocupado com LESSON até às 10:00
  As 11:30 -> Estado: OCUPADO         | Detalhe:
Ocupado com LESSON até às 13:30
  As 13:45 -> Estado: OCUPADO         | Detalhe:
Ocupado com LESSON até às 14:00
  As 16:00 -> Estado: OCUPADO         | Detalhe:
Ocupado com LESSON até às 18:00
```

Em `FI0-05`, Ricardo criou um script exploratório de ingestão. O valor desta tarefa foi transformar pesquisa em teste prático: recolher amostras, observar a estrutura real dos dados e preparar uma ingestão mais robusta no Sprint 1.

Em `FI0-06`, Ricardo propôs a lógica inicial de disponibilidade. Os estados definidos foram disponível, disponível em breve, ocupado e incerto. O estado disponível exige pelo menos 1 hora de disponibilidade; "disponível em breve" significa ficar livre em até 15 minutos. O estado incerto protege a experiência quando faltam dados ou quando os dados são contraditórios.

Em `FI0-07`, Ricardo recomendou começar pela Alameda. Esta decisão equilibra utilidade, cobertura e risco técnico. A amostra de discovery estava concentrada na Alameda, pelo que o primeiro dataset suporta melhor os utilizadores mais representados. O TIC entra por relevância, mas com tratamento especial. Taguspark fica preparado como expansão posterior.

Repositório, Spike Técnico e Deploy

Em `BE0-05`, ficou criada a estrutura base do backend: projeto Django, gestão de dependências com Pipenv, README básico, repositório GitHub privado, apps `rooms` e `ingestion`, modelos iniciais e `TimeStampedModel`.

Em `BE0-06`, Diogo validou a stack com um spike técnico. Foi criado um endpoint de teste em `/api/rooms/`, ligado a PostgreSQL, com um registo fictício de sala. Desta forma, o spike confirmou que Django, base de dados e resposta JSON podiam funcionar juntos antes do Sprint 1.

Em `BE0-07`, Diogo investigou opções de deploy. A recomendação ficou Supabase para PostgreSQL e Koyeb para alojar Django. Supabase foi recomendado pela compatibilidade PostgreSQL e pelo plano gratuito adequado ao projeto. Koyeb foi recomendado por evitar cold starts no plano analisado.

10. Transição Para o Sprint 1

O Sprint 0 deixou o projeto preparado para começar o Sprint 1 com direção concreta. O problema estava validado inicialmente, o scope estava dividido entre Sprint 1 e Sprint 2, o backlog inicial tinha sido criado, a stack estava escolhida, os dados do Fénix tinham sido testados, a arquitetura estava definida, os repositórios/base tinham sido iniciados e UX/FE tinham uma primeira linguagem comum.

As dependências críticas para o Sprint 1 ficaram claras. Fénix Integration precisava de normalizar dados e fechar a availability engine. Backend precisava de transformar modelo, ingestão e disponibilidade em endpoints utilizáveis. Frontend precisava de deixar de depender apenas de mocks e ligar-se à API. UX precisava de evoluir wireframes, estados e responsividade para uma interface mobile-first coerente.

Os riscos também ficaram explícitos. A cobertura inicial Alameda/TIC podia não representar todos os estudantes. A qualidade dos dados oficiais podia gerar estados errados se eventos sobrepostos não fossem tratados. A carga de trabalho em Backend, Frontend Features e UX era elevada para quatro semanas. A validação real com estudantes ficaria limitada pelo verão, pelo que o feedback mais útil deveria surgir depois do Sprint 2/beta.

11. Aprendizagens

O Sprint 0 foi sobretudo uma fase de aprendizagem aplicada: a equipa teve de transformar uma ideia inicial num produto com âmbito, backlog, protótipos, arquitetura e bases técnicas. As aprendizagens foram diferentes por área, mas tiveram um ponto comum: perceber que construir um produto implica método, comunicação e validação contínua, não apenas execução individual de tarefas.

Enquanto Product Owner, Miguel consolidou aprendizagem prática sobre Scrum e gestão de produto: como transformar problemas em user stories, porque é que cada história precisa de acceptance criteria claros e como usar o backlog para proteger o foco do sprint. A função exigiu também desenvolver competências de liderança: alinhar prioridades, distribuir trabalho, acompanhar progresso, pedir evidência, dar feedback útil, desbloquear decisões e manter a equipa ligada ao objetivo do produto sem perder o histórico das decisões.

Na área de UX, Carolina aprendeu a usar o Figma como ferramenta de desenho e colaboração, e ganhou familiaridade com conceitos como wireframes, protótipos, fluxo de utilização, design mobile-first e estados da interface.

Na frente de Frontend, a aprendizagem de Francisco centrou-se em React, Vite, Tailwind, componentização, routing, responsividade e gestão de estados de loading, erro, vazio e sucesso. A principal evolução foi perceber que a interface precisa de ser desenhada para dados reais e contratos explícitos com o backend, não apenas para mocks. O trabalho também obrigou a transformar protótipos e decisões de UX em componentes reutilizáveis, mantendo coerência visual e preparando a integração com a API.

No Backend, Diogo trabalhou a aprendizagem de Django, PostgreSQL, modelos relacionais, endpoints, serialização e estruturação de um projeto web com separação entre dados, lógica e respostas HTTP. O spike técnico permitiu reduzir incerteza cedo, validando a ligação à base de dados e a devolução de JSON antes de avançar para funcionalidades mais completas.

Na integração com o Fénix, Ricardo aprendeu a lidar com APIs e dados reais: interpretar endpoints, perceber campos disponíveis, filtrar informação relevante, normalizar espaços e eventos, e identificar problemas como lentidão, sobreposição de horários e dados insuficientes.

Como equipa, o Sprint 0 trouxe uma aprendizagem transversal sobre trabalho Agile: planear em ciclos curtos, tornar o trabalho visível no Notion, rever progresso e comunicar bloqueios. O grupo percebeu também que trabalho de equipa é de extrema importância, uma vez que produto, UX, frontend, backend e dados não avançam em paralelo de forma isolada: dependem uns dos outros através de decisões partilhadas, feedback frequente e contratos suficientemente claros para permitir execução coordenada.

12. Inventário do Sprint 0

Área	Código	Título	Responsável(eis)	Estado	Resumo
Product Owner	PO0-01	Preparar as User Interviews	Miguel Jordão	Done	Inclui preparação do formulário e das perguntas de discovery.
Product Owner	PO0-02	Realizar 8-12 entrevistas rápidas	Miguel Jordão	Done	Inclui recolha de 18 respostas e caracterização inicial da amostra.
Product Owner	PO0-03	Sintetizar padrões, dores e alternativas atuais	Miguel Jordão	Done	Inclui síntese estatística e interpretação dos padrões principais.
Product Owner	PO0-04	Definir problema, público e proposta de valor inicial	Miguel Jordão	Done	Inclui product brief, público-alvo, proposta de valor e personas iniciais.
Product Owner	PO0-05	Escrever as User Stories e os Critérios de Aceitação para o Sprint 1	Miguel Jordão	Done	Inclui user stories iniciais e critérios de aceitação do Core Room Finder.
Product Owner	PO0-06	Criar e priorizar o backlog Sprint 1	Miguel Jordão	Done	Inclui backlog inicial do Sprint 1 com prioridades, pontos e acceptance criteria.
Product Owner	PO0-07	Definir métricas iniciais de sucesso (para o Sprint 1)	Miguel Jordão	Done	Inclui métricas curtas de uso, utilidade e feedback futuro.
Product Owner	PO0-08	Fechar o objetivo e o plano para o Sprint 1	Miguel Jordão	Done	Inclui objetivo do Sprint 1, entregáveis, dependências e riscos.
Backend	BE0-01	Comparar e escolher a stack de backend e base de dados	Diogo Monteiro	Done	Inclui escolha de Django e PostgreSQL.
Backend	BE0-02	Desenhar a arquitetura inicial do Spotter	Diogo Monteiro	Done	Inclui arquitetura Fénix -> ingestion -> PostgreSQL -> API -> frontend.
Backend	BE0-03	Propor o modelo de dados inicial	Diogo Monteiro	Done	Inclui entidades `Building`, `Room` e `Event`.
Backend	BE0-04	Definir o contrato inicial da API para o Sprint 1	Diogo Monteiro; Ricardo Vicente	Done	Inclui endpoints, filtros e campos de resposta alinhados com Fénix Integration.
Backend	BE0-05	Criar o repositório e a estrutura base do backend	Diogo Monteiro	Done	Inclui projeto Django, Pipenv, README, apps e modelos iniciais.
Backend	BE0-06	Validar a stack com um spike técnico	Diogo Monteiro	Done	Inclui endpoint de teste ligado a PostgreSQL.
Backend	BE0-07	Investigar opções de deploy	Diogo Monteiro	Done	Inclui recomendação Supabase + Koyeb.
Fénix Integration	FI0-01	Mapear as fontes de dados candidatas	Ricardo Vicente	Done	Inclui FenixEdu API, endpoints `spaces`, ist-space-finder e API própria futura.

Fénix Integration	FI0-02	Testar o acesso às fontes prioritárias	Ricardo Vicente	Done	Inclui chamadas reproduzíveis a espaços, salas e eventos.
Fénix Integration	FI0-03	Criar um dicionário inicial dos dados	Ricardo Vicente	Done	Inclui alinhamento inicial de campos entre Fénix, Backend e Produto.
Fénix Integration	FI0-04	Avaliar qualidade, cobertura e limitações	Ricardo Vicente	Done	Inclui limites dos dados, eventos sobrepostos e lentidão de pedidos sala a sala.
Fénix Integration	FI0-05	Criar um script exploratório de ingestão	Ricardo Vicente	Done	Inclui exploração prática dos dados para preparar ingestão futura.
Fénix Integration	FI0-06	Propor a lógica de disponibilidade para o Sprint 1	Ricardo Vicente	Done	Inclui estados disponível, disponível em breve, ocupado/indisponível e incerto.
Fénix Integration	FI0-07	Recomendar o primeiro conjunto de dados a suportar	Ricardo Vicente	Done	Inclui Alameda primeiro, TIC com tratamento especial e Taguspark depois.
Frontend Features	FE0-01	Fechar a divisão das tarefas entre os dois cargos de Frontend	Maria Carolina Vidal; Francisco Frieza	Done	Inclui separação entre owner visual/UX e owner funcional/técnico.
Frontend Features	FE0-02	Criar a estrutura base do frontend	Francisco Frieza	Done	Inclui estrutura base, assets, componentes e primeira página/lista.
Frontend Features	FE0-03	Criar a matriz de estados da interface	Francisco Frieza	Done	Inclui estados de loading, vazio, erro, retry, dados incompletos e sucesso.
Frontend Features	FE0-04	Preparar a checklist inicial de QA	Francisco Frieza	Done	Inclui critérios iniciais para verificar user stories.
Frontend Features	FE0-05	Avaliar opções de analytics	Francisco Frieza	Ice Box	Inclui tarefa Should Have de analytics que ficou fora do scope concluído do Sprint 0.
Frontend Features	FE0-06	Especificar a página de detalhe para o Sprint 1	Francisco Frieza	Done	Inclui campos e comportamento esperados na página de detalhe.
Frontend UX	UX0-01	Analisar produtos e padrões de referência	Maria Carolina Vidal	Done	Inclui referências visuais e funcionais para lista, filtros e detalhe.
Frontend UX	UX0-02	Mapear o fluxo principal para o Sprint 1	Maria Carolina Vidal	Done	Inclui fluxo campus -> lista -> filtros -> detalhe.
Frontend UX	UX0-03	Criar wireframes mobile-first	Maria Carolina Vidal	Done	Inclui wireframes iniciais em formato low-fidelity.
Frontend UX	UX0-04	Definir o design system inicial	Maria Carolina Vidal	Done	Inclui cores, fonte, cards, radius e base visual.
Frontend UX	UX0-05	Definir os estados visuais de disponibilidade	Maria Carolina Vidal	Done	Inclui mapeamento visual dos estados de disponibilidade.
Frontend UX	UX0-06	Escolher framework e styling com a equipa	Maria Carolina Vidal; Francisco Frieza	Done	Inclui decisão React + Vite + Tailwind + shadcn/ui seletivo.

13. Bibliografia

Notion do Spotter. (2026). Workspace interno do projeto: Product Backlog, Sprint Tracker, user stories, tarefas e decisões. Fonte interna consultada em 2 de agosto de 2026.

Spotter. (2026). Questionário de discovery e respostas recolhidas no Microsoft Forms. Artefacto interno consultado em 2 de agosto de 2026.

Spotter. (2026). Wireframes e design system inicial em Figma. Artefacto interno consultado em 2 de agosto de 2026.

Spotter. (2026). Repositórios internos de frontend e backend. Artefactos de implementação consultados em 2 de agosto de 2026.

Schwaber, K. e Sutherland, J. (2020). The Scrum Guide. Guia metodológico. https://scrumguides.org/scrum-guide.html. Acedido em 2 de agosto de 2026.

FenixEdu e Instituto Superior Técnico. (s.d.). Documentação e endpoints da API Fénix. https://fenixedu.org/dev/api/; https://fenix.tecnico.ulisboa.pt/api/fenix/v1. Acedido em 2 de agosto de 2026.

IST Space Finder. (s.d.). Aplicação web de referência para disponibilidade de espaços no IST. https://spaces.leic.pt/. Acedido em 2 de agosto de 2026.

Documentação técnica consultada. (s.d.). React, Vite, Tailwind CSS, Django e PostgreSQL. https://react.dev/; https://vite.dev/; https://tailwindcss.com/docs; https://docs.djangoproject.com/; https://www.postgresql.org/docs/. Acedido em 2 de agosto de 2026.