A black background with green and brown text

Description automatically generated

Escola Superior de Tecnologia e Gestão  
Licenciatura em Engenharia Informática

Engenharia de Software

Plano de Desenvolvimento de Software Projeto Airbnb

**António Pedro Duarte Rosa, 24105**

**Rita Alexandra Lampreia Dias, 23240**

****

Beja, junho de 2025

Instituto Politécnico de Beja

Escola Superior de Tecnologia e Gestão  
Licenciatura em Engenharia Informática

Engenharia de Software

Plano de Desenvolvimento de Software Projeto Airbnb

**António Pedro Duarte Rosa, 24105**

**Rita Alexandra Lampreia Dias, 23240**

ORIENTAÇÃO

Professora Isabel Sofia Sousa Brito

Beja, junho de 2025

Índice

[Abstract 4](#_Toc199496833)

[1 Introdução 5](#_Toc199496834)

[2 Fase de Planeamento Inicial e Gestão do Projeto 6](#_Toc199496835)

[2.1 Definição de objetivos 6](#_Toc199496836)

[2.2 Metodologia de desenvolvimento de Software 6](#_Toc199496837)

[2.2.1 Simulação de Contexto Profissional 7](#_Toc199496838)

[2.2.2 Reflexão sobre Desenvolvimento Real 9](#_Toc199496839)

[2.3 Ferramentas CASE e plataformas de apoio 10](#_Toc199496840)

[2.4 Calendarização e organização da equipa 11](#_Toc199496841)

[2.4.1 Sprint 1 – Planeamento e Iniciação do Projeto (*Duração estimada: 1 semana)* 11](#_Toc199496842)

[2.4.2 Sprint 2 – Fase de análise (*Duração estimada: 1 semana)* 12](#_Toc199496843)

[2.4.3 Sprint 3 –Fase de Desenho (*Duração estimada: 1 semana)* 13](#_Toc199496844)

[2.4.4 Gestão do projeto 13](#_Toc199496845)

[3 Fase de Análise / Modelação do Domínio do Problema 14](#_Toc199496846)

[4 Fase de Desenho / Modelação do Domínio do Sistema (Solução) 15](#_Toc199496847)

[5 Links para plataformas de apoio 16](#_Toc199496848)

[6 Referências Bibliográficas 16](#_Toc199496849)

Índice de Figuras

[Figura 1- Diagrama desenvolvido no Figma para ilustrar o fluxo de trabalho Scrum com iterações por sprint 8](#_Toc199260685)

# Abstract

cxdeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwegdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrktetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhwegdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrktetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfetjrktetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdhektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektra

# Introdução

cxdeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwegdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrktetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhwegdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrktetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdgdffhdfahjfhdrrjbtljkwcxdfcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfetjrktetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektrafcxgdfsdgdffhdfahjfhdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrawjeerfakwnrfjdhektrawjeerfakwnrfjlfalrwlerjlwerjwehdrrjbtljkwdvhetjrkthektrahdrrjbtljkwdvhetjrkthektra

# Fase de Planeamento Inicial e Gestão do Projeto

## Definição de objetivos

O objetivo central deste projeto académico é simular o desenvolvimento de um sistema de reserva de alojamentos, inspirado na plataforma Airbnb, com o intuito de aplicar conceitos e metodologias de Engenharia de Software. Em particular, vamos focar-nos na modelação da fase de análise (domínio do problema) e da fase de desenho (domínio da solução), estruturando o processo de desenvolvimento sem implementar o sistema completo. De forma mais detalhada, pretendemos:

* Escolher a metodologia ágil mais adequada e selecionar as ferramentas CASE que darão suporte à gestão do projeto;
* Definir os papéis e responsabilidades da equipa, planear e calendarizar o projeto;
* Analisar a plataforma Airbnb e similares para levantar e categorizar requisitos funcionais e não funcionais, desenvolver user stories e elaborar diagramas de casos de uso UML, assegurando que os requisitos identificados respondem às necessidades do utilizador e do product owner (domínio do problema);
* Desenhar os modelos UML essenciais, incluindo diagramas de classes e sequência, representando as soluções do sistema (domínio da solução);
* Gerir o projeto com validação contínua das tarefas e entregas, comunicação eficaz e controlo de versões dos documentos e ficheiros.

## Metodologia de desenvolvimento de Software

Neste trabalho académico, propomo-nos a modelar as fases de análise e desenho de um projeto de desenvolvimento de software inspirado na plataforma Airbnb. Desta forma, o foco aqui não é a implementação total do sistema, mas sim a estruturação do processo e dos componentes que o antecedem.

Como o desenvolvimento de software hoje em dia é dinâmico e iterativo, optámos por usar uma metodologia ágil, em vez de modelos mais tradicionais, como o Waterfall, que seguem uma sequência rígida e linear. Estes modelos mais antigos tendem a ser menos flexíveis a mudanças nos requisitos e têm dificuldade em integrar o feedback contínuo do cliente ao longo do projeto.

Em particular, escolhemos a metodologia Scrum porque, na nossa perspectiva oferece uma combinação equilibrada entre estrutura e capacidade de adaptação. Como refere (Valente, 2020), “Scrum é um método ágil para gerenciamento de projetos, que não necessariamente precisam ser projetos de desenvolvimento de software” mostrando sua versatilidade, abrangência e acima de tudo a sua relevância para o contexto académico e para a estrutura que estamos a desenvolver.

### Simulação de Contexto Profissional

A decisão de seguir esta abordagem visa mimetizar o contexto profissional de desenvolvimento de software, permitindo-nos compreender de forma aplicada os princípios, metodologias, papéis e artefactos usados em equipas reais. Assim, definimos:

* **Papéis**:
* Product Owner: Responsável por definir e priorizar os requisitos do sistema e gerir o Product Backlog. Neste projeto, esta função foi assumida pelo António Rosa.
* Scrum Master: Responsável por garantir que o processo Scrum é seguido e que os bloqueios sejam removidos. Desempenhado por Rita Dias.
* Development Team: Ambos os membros da equipa participam ativamente na produção dos artefactos (diagramas, requisitos, documentos).
  + **Artefactos**:
* Product Backlog: Lista completa de funcionalidades desejadas para o sistema.
* Sprint Backlog: Subconjunto do Product Backlog selecionado para cada sprint.
* Increment: Conjunto de entregáveis prontos no final de cada Sprint (neste caso, diagramas, modelos e documentação).

As tarefas são organizadas no Trello, ferramenta CASE escolhida para dar suporte à metodologia Scrum. Cada tarefa ou funcionalidade é representada como um cartão, contendo descrições e atribuições. As tarefas seguem o fluxo: Product Backlog → Sprint Backlog → Em Progresso → Em Revisão/Testes → Concluído. Este processo é representado na figura 1, que ilustra o ciclo iterativo de desenvolvimento Scrum. O Product Backlog centraliza todas as funcionalidades identificadas para o projeto, e durante o planeamento de sprints, um subconjunto dessas funcionalidades é alocado a cada Sprint. Cada Sprint passa pelas fases mencionadas, resultando em incrementos de software. A soma dos incrementos desenvolvidos ao longo dos diferentes sprints contribui para a construção evolutiva e contínua do produto final. Na figura 2, apresentamos um print do quadro trello durante o sprint 2 para ilustrar o processo de organização, evolução e monitorização do progresso das tarefas.

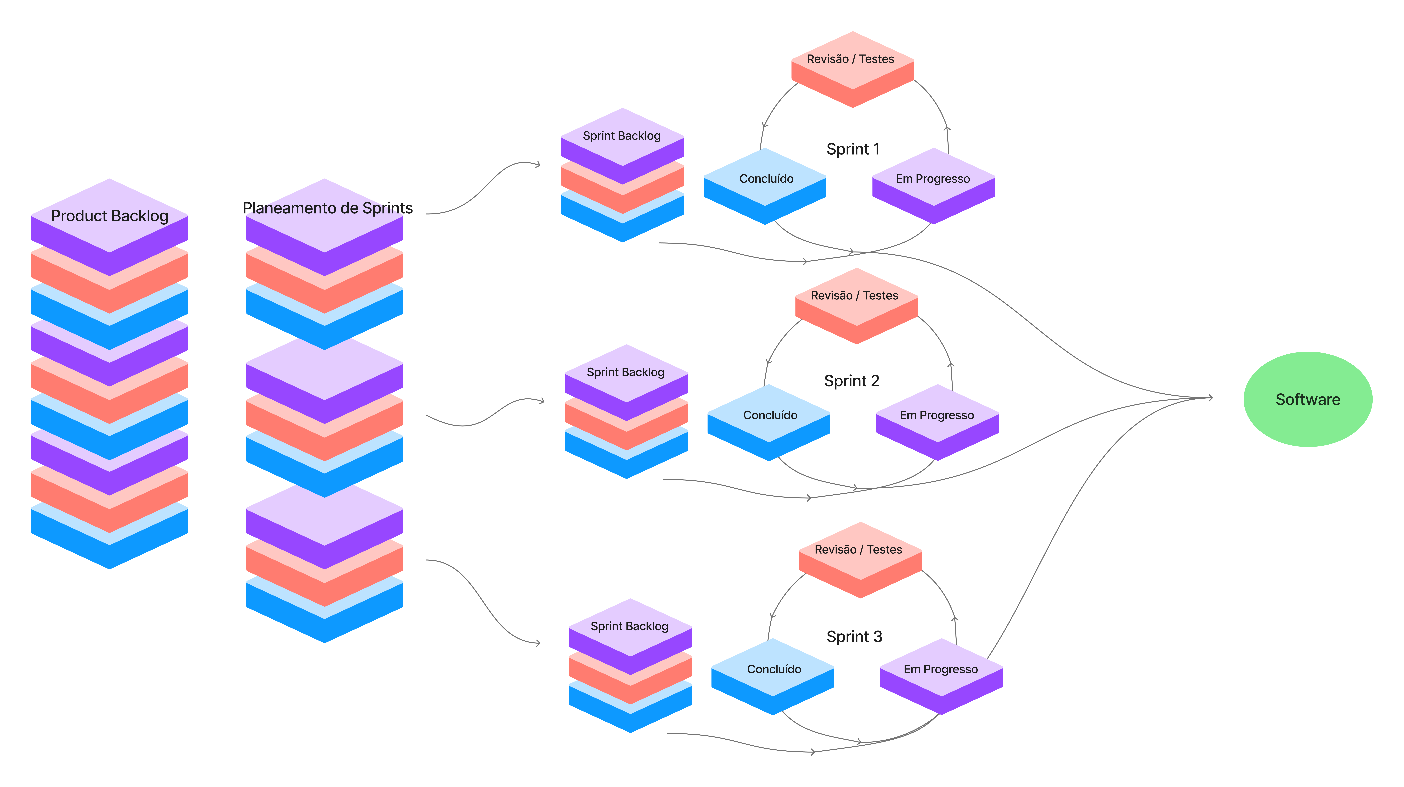


Figura - Diagrama desenvolvido no Figma para ilustrar o fluxo de trabalho Scrum com iterações por sprint

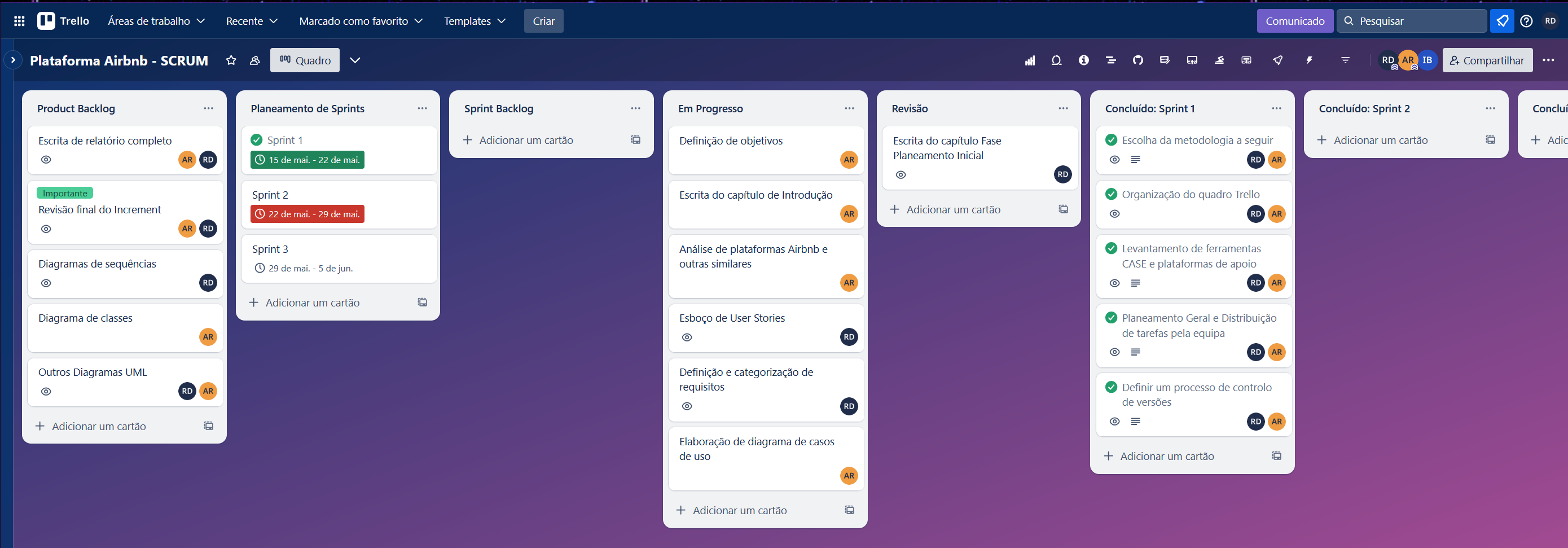


Figura - Quadro Trello durante o Sprint 2

* **Reuniões Scrum**:

Em alinhamento com a metodologia Scrum, também considerámos o conjunto de cerimónias previsto: Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective. Contudo, adaptámos essas reuniões ao nosso contexto académico.

Realizámos reuniões de planeamento no início de cada sprint (Sprint Planning) e as reuniões de revisão e retrospetiva no final de cada fase, foram fundidas num único evento. No entanto, não realizámos reuniões diárias (Daily Scrum), pois a comunicação foi contínua e informal, assegurando alinhamento e progressos sem necessidade de reuniões diárias estruturadas devido ao formato e a escala do projeto, com apenas dois membros.

### Reflexão sobre Desenvolvimento Real

No contexto de um desenvolvimento real de um sistema como o Airbnb, a adoção de uma metodologia ágil como o Scrum também se revelaria adequada, dada a necessidade de iterações constantes, validação com utilizadores e adaptabilidade a novas funcionalidades.

Contudo, dada a complexidade e dimensão técnica de um sistema com tantos utilizadores, seria igualmente importante adotar uma arquitetura baseada em microsserviços. Tal arquitetura permite dividir o sistema em "Certos grupos de módulos que são executados em processos independentes, sem compartilhamento de memória. Ou seja, o sistema é decomposto em módulos não apenas em tempo de desenvolvimento, mas também em tempo de execução." (Valente, 2020) Com isso, reduz-se significativamente o risco de uma alteração num módulo afetar o comportamento de outros.

"Uma segunda vantagem de microsserviços é escalabilidade." (Valente, 2020) Esta modularidade permite que diferentes partes do sistema possam crescer ou ser otimizadas de forma independente. Este princípio é essencial para plataformas como o Airbnb, que processam grandes volumes de dados, operações complexas e requerem alta disponibilidade e desempenho.

Assim, neste trabalho académico, usamos Scrum para entender o processo, os papéis e os artefactos de um projeto real. Mas reconhecemos que num contexto profissional completo, além da metodologia de gestão, é crucial considerar também a arquitetura técnica, como os microsserviços, para garantir robustez e escalabilidade.

## Ferramentas CASE e plataformas de apoio

Para apoiar o desenvolvimento do projeto, foram selecionadas ferramentas CASE e plataformas de gestão que facilitam a modelação, o controlo de versões e a colaboração da equipa, otimizando o fluxo de trabalho e a qualidade dos artefactos. Assim, optámos pelas seguintes soluções:

* + Trello – para gestão de tarefas, organização de sprints e comunicação.
  + Visual Paradigm – para a modelação do sistema através de UML.
  + GitHub – para a produção colaborativa da documentação e do relatório técnico, assegurando também o controlo de versões.
  + Figma – para desenho de diagramas genéricos.

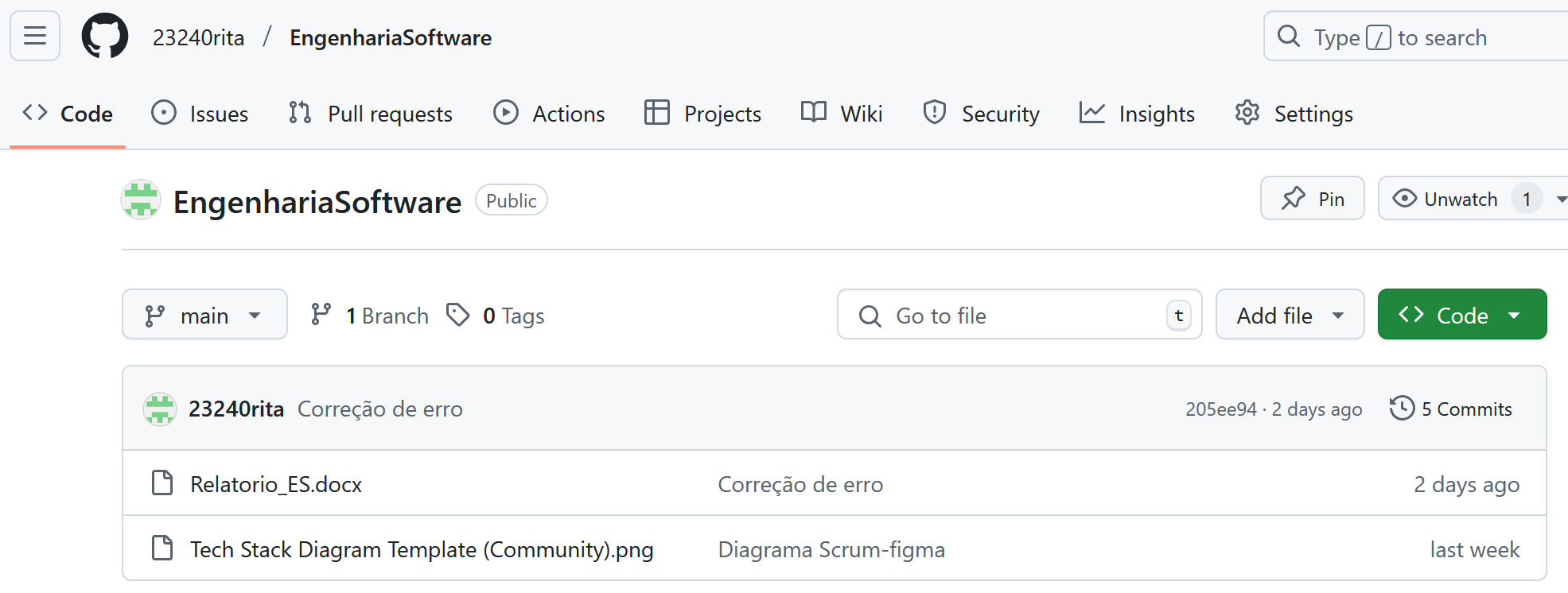


Figura – GitHub para controlo de versões e manutenção de histórico da documentação e diagramas

## Calendarização e organização da equipa

O projeto foi dividido em fases (sprints), com prazos definidos para cada entrega. As responsabilidades encontram-se distribuídas entre os dois membros da equipa, da seguinte forma:

### Sprint 1 – Planeamento e Iniciação do Projeto (*Duração estimada: 1 semana)*

**Objetivos:**

* Escolha da metodologia
* Pesquisa e seleção das ferramentas CASE
* Definição dos papéis e organização da equipa
* Planeamento geral e calendarização do projeto
* Criação do quadro Trello
* Escrita dos capítulos de introdução e Fase de Planeamento Inicial e Gestão do Projeto

**Responsabilidades:**

* **Rita Dias:**
  + Criação e estruturação do quadro Trello
  + Esboço inicial do plano de projeto
  + Redação da secção da fase de planeamento Inicial do relatório final
* **António Rosa:**
  + Definição dos objetivos do projeto
  + Definir um processo de controlo de versões e manter o histórico (usar uma ferramenta para garantir a rastreabilidade)
  + Redação da introdução do relatório final

**Entregáveis:**

* Quadro Trello com tarefas organizadas
* Documento inicial com plano de desenvolvimento
* Descrição da metodologia e ferramentas
* Parte inicial do relatório

### Sprint 2 – Fase de análise (*Duração estimada: 1 semana)*

**Objetivos:**

* Análise da plataforma Airbnb e similares
* Escrita de user stories
* Elaboração do Diagrama de Casos de Uso UML
* Levantamento e categorização de requisitos funcionais e não funcionais
* Validação cruzada dos requisitos (Garantir que os requisitos identificados correspondem realmente às necessidades dos utilizadores e do product owner)
* Escrita dos capítulos da fase de análise no relatório

**Responsabilidades:**

* **Rita Dias:**
  + Escrita das user stories
  + Levantamento e categorização de requisitos
* **António Rosa:**
  + Análise da plataforma Airbnb e similares
  + Elaboração do diagrama de casos de uso
* **Ambos:**
  + Escrita dos capítulos da fase de análise no relatório

**Entregáveis:**

* Lista de requisitos funcionais e não funcionais
* User stories
* Entrevista para utilizadores do sistema Airbnb
* Secção de requisitos do relatório final

### Sprint 3 –Fase de Desenho (*Duração estimada: 1 semana)*

**Objetivos:**

* Criação do Diagrama de Classes UML
* Elaboração dos Diagramas de Sequência
* Criação de diagramas complementares (atividades, estados, componentes)
* Revisão geral e finalização do relatório técnico

**Responsabilidades:**

* **Rita Dias:**
  + Diagramas de sequência UML e outros diagramas UML, se necessário (ex: componentes, estados)
* **António Rosa:**
  + Diagrama de classes e outros diagramas UML, se necessário (ex: componentes, estados)
* **Ambos:**
  + Escrita final e revisão do relatório
  + Validação de todos os artefactos entre os membros
  + Comunicação contínua via Trello

**Entregáveis:**

* Diagramas UML completos – ficheiro Visual Paradigm
* Relatório técnico final

### Gestão do projeto

* + Validação contínua das tarefas e entregas entre sprints.
  + Comunicação ativa entre os membros através de Trello e outros meios.
  + Controlo de versões dos documentos e ficheiros através de backups no GitHub.

# Fase de Análise / Modelação do Domínio do Problema

Esta fase centra-se na compreensão do problema e na identificação dos requisitos do sistema:

* **Recolha de informação:**
  + Análise da aplicação Airbnb real e pesquisa de plataformas semelhantes.
  + Levantamento inicial de requisitos através de User Stories e Casos de Uso.
* **Análise e validação da informação/documentação:**
  + Organização e validação dos requisitos funcionais (ex: "o utilizador pode reservar um alojamento", “o hóspede pode avaliar o alojamento em que esteve”).
  + Identificação dos requisitos não funcionais (ex: desempenho, segurança, usabilidade).
  + Quando possível, conversão de requisitos não funcionais em requisitos funcionais mensuráveis (ex: "a aplicação deve responder em menos de 2 segundos").
* **Modelação de Casos de Uso:**
  + Elaboração do Diagrama de Casos de Uso em UML.
  + Descrição detalhada de cada caso de uso segundo a template fornecida, com atores, fluxo principal, exceções, etc.
  + Especificação e categorização de todos os requisitos levantados.

# Fase de Desenho / Modelação do Domínio do Sistema (Solução)

Esta fase diz respeito à transição da análise para o desenho técnico da solução:

* **Elaboração dos Diagramas UML:**
  + Diagramas de Sequência para representar a interação entre objetos ao longo do tempo em cenários principais (ex: processo de reserva, login, envio de avaliação).
  + Diagrama de Classes para modelar as entidades principais, os seus atributos, métodos e relações.
  + Outros diagramas UML, se necessários, como:
    - Diagrama de estados (ex: estados de uma reserva)
    - Diagrama de atividades (ex: fluxo de registo de utilizador)
    - Diagrama de componentes (para ilustrar arquitetura técnica, se aplicável)

# Links para plataformas de apoio

Trello: <https://trello.com/invite/b/67f6c2436b47e53da2146dac/ATTIaa2ae2d6b8d36d7e16784d61fba9ea092D97186A/plataforma-airbnb-scrum>

GitHub:

<https://github.com/23240rita/EngenhariaSoftware>

# Referências Bibliográficas

Valente, M. T. (2020). *Engenharia de Software Moderna - Princípios e práticas para desenvolvimento de software com produtividade.* Independente.

<https://www.atlassian.com/blog/trello/how-to-scrum-and-trello-for-teams-at-work> (último acesso 06/04/2025)

<https://www.airbnb.com/help/article/2503> (último acesso 09/04/2025)

<https://www.mountaingoatsoftware.com/agile/scrum/meetings> (último acesso 10/04/2025)

<https://www.mountaingoatsoftware.com/agile/agile-planning> (último acesso 10/04/2025)

<https://www.knowledgehut.com/tutorials/scrum-tutorial/scrum-artifacts-overview> (último acesso 10/04/2025)

<https://agilemodeling.com/artifacts/systemUseCase.htm> (último acesso 09/04/2025)

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/> (último acesso 10/04/2025)

**Engenharia de Requisitos**

Os requisitos de um sistema definem *o que* ele deve fazer e *como* ele deve operar. Assim, aEngenharia de Requisitos inclui o conjunto de atividades realizadas com o objetivo de definir,analisar, documentar e validar os requisitos de um sistema. Em uma primeira classificação, osrequisitos podem ser **funcionais** ou **não-funcionais**.

Requisitos funcionais definem *o que* um sistema deve fazer; isto é, quais funcionalidades ou serviçosele deve implementar.

Já os requisitos não-funcionais definem *como* um sistema deve operar, sob quais restrições e comqual qualidade de serviço. São exemplos de requisitos não-funcionais: desempenho, disponibilidade,tolerância a falhas, segurança, privacidade, interoperabilidade, capacidade, manutenibilidade eusabilidade.

Por exemplo, suponha um sistema de *home-banking*. Nesse caso, os requisitos funcionais inclueminformar o saldo da conta, informar o extrato, realizar transferência entre contas, pagar um boletobancário, cancelar um cartão de débito, etc. Já os requisitos não-funcionais, dentre outros, incluem:

Desempenho: informar o saldo da conta em menos de 3 segundos;

Disponibilidade: estar no ar 99% do tempo;

Tolerância a falhas: continuar operando mesmo se um determinado centro de dados cair;

Segurança: criptografar todos os dados trocados com as agências;

Privacidade: não disponibilizar para terceiros dados de clientes;

Interoperabilidade: integrar-se com os sistemas do Banco Central;

Capacidade: ser capaz de armazenar dados de 1 milhão de clientes;

Usabilidade: ter uma versão para deficientes visuais.

-> LIVRO ENGENHARIA DE SOFTWARE MODERNA