

**HOUSEM8**

REPORT

Version 1.6

13/01/2021

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Samuel Cunha	07/12/2020	Samuel Cunha	07/12/2020	Introdução, âmbito e introdução das ferramentas
1.1	Jorge Moreira	07/12/2020	Jorge Moreira Samuel Cunha	07/12/2020	Finalização do tópico de ferramentas e âmbito
1.2	Samuel Cunha	07/12/2020	Samuel Cunha	07/12/2020	Adição de mais ferramentas utilizadas Alteração das imagens Pequena conclusão
1.3	Jéssica Coelho	08/12/2020	Jorge Moreira Samuel Cunha	08/12/2020	Alteração de imagens
1.4	Jorge Moreira Jéssica Coelho	08/12/2020	Jorge Moreira Samuel Cunha	08/12/2020	Finalização dos tópicos em falta
1.5	Jéssica Coelho	05/01/2021	Jéssica Coelho	05/01/2021	Atualização do Relatório Final
1.6	Jorge Moreira Jéssica Coelho	12/01/2021	Jorge Moreira	12/01/2021	Atualização das ferramentas Criação do tópico de arquitetura

## Índice

1	Introdução.....	5
2	Âmbito .....	5
3	Arquitetura .....	6
3.1	Padrão MVVM em Flutter .....	6
3.2	Padrão DAO na API .....	6
4	Ferramentas.....	8
4.1	Git / GitLab.....	8
4.2	GitLab Wiki / Documentação .....	9
4.3	.Net 5 e <i>Visual Studio</i> .....	9
4.3.1	Testes .....	10
4.4	Postman .....	10
4.5	Azure .....	11
4.6	Flutter SDK e Dart .....	11
4.7	Android Studio.....	11
5	Metodologia .....	13
5.1	SCRUM .....	13
5.1.1	Sprint #1.....	16
5.1.2	Sprint #2.....	16
5.1.3	Sprint #3.....	16
6	Conclusão.....	17

## Índice de figuras

Figura 1 - Arquitetura alto nível .....	7
Figura 2 - .gitignore .....	8
Figura 3 - Estrutura do projeto.....	9
Figura 4 - Exemplo de testes .....	10
Figura 5 - Code coverage .....	10
Figura 6 - postman logo.....	10
Figura 7 - Azure Logo .....	11
Figura 8 - Flutter e Dart .....	11
Figura 9 - Android Studio .....	11
Figura 10 - Board Backlogs .....	13
Figura 11 - Board Change Requests .....	14
Figura 12 - Board Development .....	14
Figura 13 - Board Priorities.....	15
Figura 14 - Sprints .....	15

# 1 Introdução

Esta secção contém a introdução deste documento. Este documento destina-se a descrever o uso e identificar, as ferramentas usadas para o desenvolvimento do projeto de Laboratório de Desenvolvimento de Software.

Na secção 2 será mencionado o âmbito em que este trabalho se enquadra. Na secção 3 é apresentada uma arquitetura de alto nível da aplicação. Na secção 4 serão identificadas as ferramentas usadas e como é que foram utilizadas no âmbito da metodologia SCRUM. Na secção 5 será explicado de que modo é que a metodologia *SCRUM* foi aplicada usando o *GitLab* da *ESTG*.

# 2 Âmbito

Este projeto desenvolvido no âmbito da disciplina de LDS, dedica-se ao desenvolvimento de uma aplicação (mobile), com o objetivo de fornecer oportunidades de trabalho ou procura de trabalhadores para a realização de certos trabalhos (tais como reparações, canalização, jardinagem, etc.) em casas de clientes. Este sistema será composto por uma aplicação para dispositivos móveis desenvolvida em Flutter. Para a disponibilização dos serviços de negócio, irá ter uma aplicação no *backend* uma *api* desenvolvida em ASP.net core.

Para mais informações sobre os requisitos do software, veio em conjunto com este documento:

- Software Requirements Specification

Junto desta documentação, vem também em anexo, diagramas representantes da arquitetura e cenários a que se aplicam as funcionalidades, isto é:

- Diagramas de Classes
- Diagramas de Componentes
- Diagramas de Use Case

Com este projeto prático, também se pretende que sejam demonstradas competências relativamente ao desenvolvimento ágil de software, com foco em práticas de garantia de qualidade de software, nomeadamente, testes, *Software Configuration Management (SCM)* e uso de padrões de segurança. Pretende-se que seja seguida uma abordagem baseada em *Continuous Integration*, procurando suportar e automatizar o mais possível o processo de construção de software, usando também metodologias ágeis, tal como já dito anteriormente, *SCRUM*.

## 3 Arquitetura

### 3.1 Padrão MVVM em Flutter

O sistema é composto por uma aplicação mobile desenvolvida em flutter, recorrendo ao padrão de software MVVM (*Model-View-ViewModel*).

#### Porque MVVM?

O padrão MVVM separa a *view* do modelo, o que permite que exista um desenvolvimento mais dinâmico da aplicação, onde designers e desenvolvedores conseguem ambos trabalhar na aplicação, visto que a *view* está separada do modelo e não o consegue manipular diretamente. Este padrão não só permite desenvolvimento dinâmico como também permite o reuso do código.

A *view* é simplesmente a camada responsável por mostrar informação aos utilizadores e recolher input.

O modelo representa apenas uma entidade do negócio.

A *viewModel* é a camada de ligação entre a *view* e o modelo. Esta é responsável por expor a informação dos modelos de uma maneira que seja fácil para a *view* apresentar a informação ao utilizador. A *viewModel* expõe também comportamentos, que permitem ser invocados através da *view*, para que sejam obtidas e/ou manipuladas informações relativamente aos modelos. A *viewModel* pode ser vista como o intermediário entre a *view* e o modelo.

Na nossa aplicação temos também a camada de serviços, que é responsável por fazer os pedidos à API e de retornar a informação associada aos modelos.

### 3.2 Padrão DAO na API

O padrão DAO (*Data Access Object*) é usado para separar dados de baixo nível que a API usa ao efetuar operações na base de dados, das operações de serviços de negócios de alto nível. Com este padrão podemos escolher que informação apresentar ao exterior e privar o exterior de ver informação indesejada.

Como consta na figura a seguir, os *controllers* da API recebem os pedidos da app do utilizador. Nestes pedidos pode vir informação correspondente aos modelos que são expostos, que depois são mapeados para entidades internas a que o utilizador não tem acesso.

A partir da informação destas entidades são feitas as operações na base de dados. As entidades e os modelos são os nossos *Data transfer objects* (DTOs).

Para enviar informação de volta ao utilizador, as entidades provenientes da BD são mapeadas para os modelos a serem expostos ao exterior, que por sua vez são enviados pelos *controllers* de volta à aplicação do utilizador.

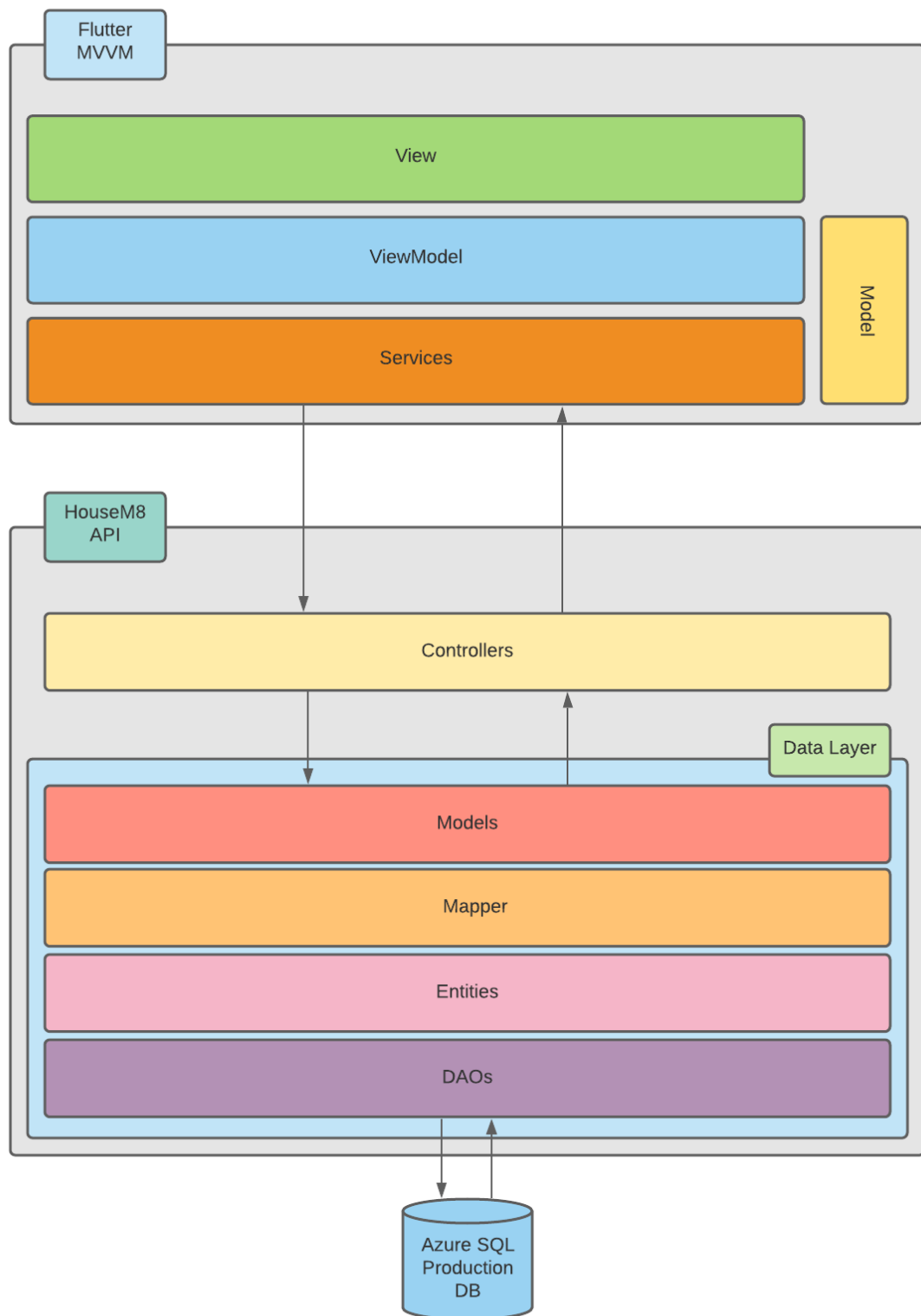


Figura 1 - Arquitetura alto nível

## 4 Ferramentas

### 4.1 Git / GitLab

O projeto desenvolvido foi gerido usando o GitLab da ESTG: <https://gitlab.estg.ipp.pt/8160297/lds-housem8>.

Quando o repositório foi criado, logo de seguida foi colocado um ficheiro “*.gitignore*”, de modo a que só fosse para o repositório o código fonte, os testes e outros ficheiros que não fossem gerados pelos *IDEs* usados pelos membros dos grupos. O ficheiro “*.gitignore*” foi criado usando a ferramenta: <http://www.gitignore.io/>, com as seguintes opções:

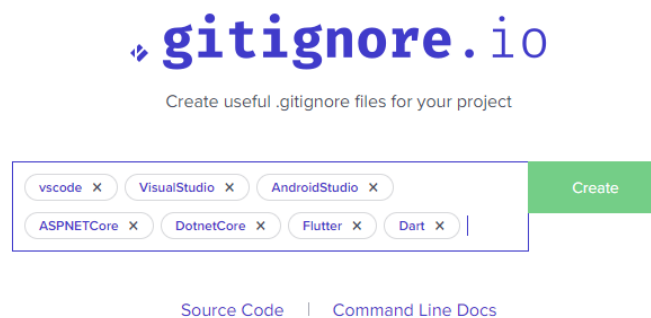


Figura 2 -.gitingore

Depois de criado o repositório com o projeto base, e com o “*.gitignore*”, este estava pronto a ser usado.

A gestão deste projeto foi feita usando, a técnica de *branching* de *GitFlow*. Existe uma *branch* “*master*” que só mantém o código estável, normalmente código proveniente de *releases* estáveis. Existe também uma *Branch* chamada “*Develop*”, que mantém o código em desenvolvimento. Quando foram feitas modificações ao projeto ou desenvolvimento de *features*, estas eram feitas em *branches* originadas a partir da *Develop*, e quando era dado o *merge*, as *branches* eram apagadas e os *commits* passados para a *target branch*, também era feito o *squash* de *commits*.



## 4.2 GitLab Wiki / Documentação

A documentação foi colocada na *wiki* do *GitLab* recorrendo a ajuda a *snippets*. Os *snippets* serviram para anexar ficheiros, usar os links de download dos mesmos na *wiki*. Os ficheiros anexados foram: *SCM Plan*, *Test Design Specification*, *Test Case Outline*, o relatório final (este documento) e as atas de reunião.

Wiki: <https://gitlab.estg.ipp.pt/8160297/lds-housem8/wikis/home>.

## 4.3 .Net 5 e Visual Studio

Este projeto foi desenvolvido utilizando dotNet5 e o *Visual Studio*. Foi com estas ferramentas que também construímos e executamos os testes numa *api* de testes dentro da mesma solução como consta nas seguintes figuras:

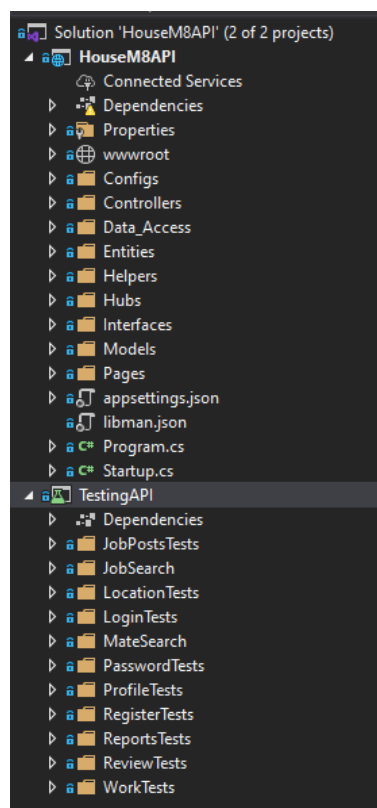


Figura 3 - Estrutura do projeto

#### 4.3.1 Testes

Os testes foram realizados usando a ferramenta *xUnit* para *dotNet* e foi usado o visual studio para recolher informação de *code coverage* como consta nas figuras abaixo, para os testes da funcionalidade em tempo real de Chat foi utilizado uma página Html para simular a conversa entre dois utilizadores.

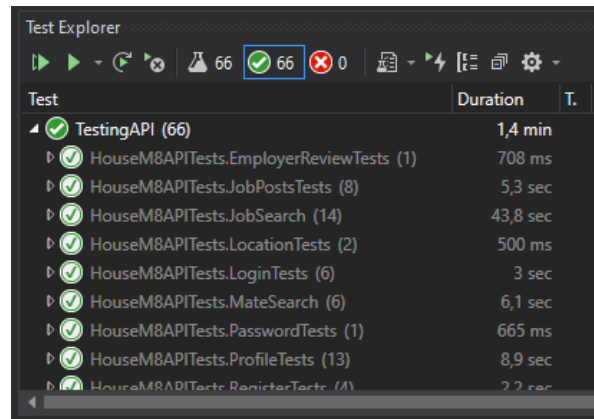
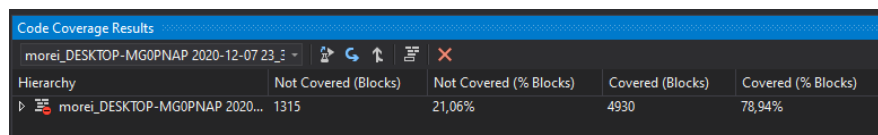


Figura 4 - Exemplo de testes



Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
morei_DESKTOP-MG0PNAP 2020-12-07 23_3	1315	21,06%	4930	78,94%

Figura 5 - Code coverage

Foi realizada documentação para todos os testes feitos:

- Test Design Specification
- Test Case Outline
- Teste de Gestão de imagens

Estes dois documentos foram entregues juntamente com este relatório.

Nota: A figura 3 e figura 4 não representam o estado final dos testes, pois ainda faltam criar mais funcionalidades e excluir classes da code coverage. Apenas foram usadas como exemplos.

#### 4.4 Postman

O Postman foi a ferramenta usada para testar todas as rotas que foram feitas ao longo do trabalho, de modo a verificar o sucesso da implementação das mesmas.



<https://www.postman.com/>

Figura 6 - postman logo

#### 4.5 Azure

A plataforma *Azure* foi a ferramenta utilizada para hospedar a base de dados do sistema, usando os créditos disponíveis no azure para estudantes. Para além da base de dados, foi usado o Azure DevOps para executar as pipelines relativas ao trabalho, e correr testes, visto que as pipelines do gitlab da ESTG estavam com problemas.



<https://azure.microsoft.com/pt-pt/>

*Figura 7 - Azure Logo*

#### 4.6 Flutter SDK e Dart

Para o desenvolvimento da aplicação mobile, foi utilizada a framework flutter para que a aplicação pudesse ser desenvolvida tanto para android, como para iOS, usando a linguagem de programação da google, o Dart.



<https://flutter.dev/>

*Figura 8 - Flutter e Dart*

#### 4.7 Android Studio

Como IDE para a implementação da plataforma mobile, foi utilizado o Android Studio, que nos permitiu também testar a aplicação com recurso a um emulador.



<https://developer.android.com/studio>

*Figura 9 - Android Studio*



## 5 Metodologia

### 5.1 SCRUM

A metodologia ágil usada no desenvolvimento deste produto é *SCRUM*. Ao seguir a metodologia *SCRUM* desenvolvemos o software de uma forma iterativa, usando *sprints*. No caso deste trabalho, os *sprints* foram mencionados no *Gitlab* nas *milestones*. Estes *sprints* foram iniciados depois de um *sprint planning*, uma reunião entre os membros do grupo que definiram um *backlog* priorizado para a equipa de desenvolvimento implementar no trabalho. *Sprint review* também foi feito de modo a perceber o que foi feito ao longo do tempo até a um determinado sprint, para perceber o que se teve de alterar / corrigir, como por exemplo *bugs* ou *features*, que poderiam ser úteis para englobar no planeamento do *sprint* seguinte.

No *gitlab* criamos 4 boards nas *issues* – (*Backlogs*, *Change requests*, *Development* e *Priorities*):

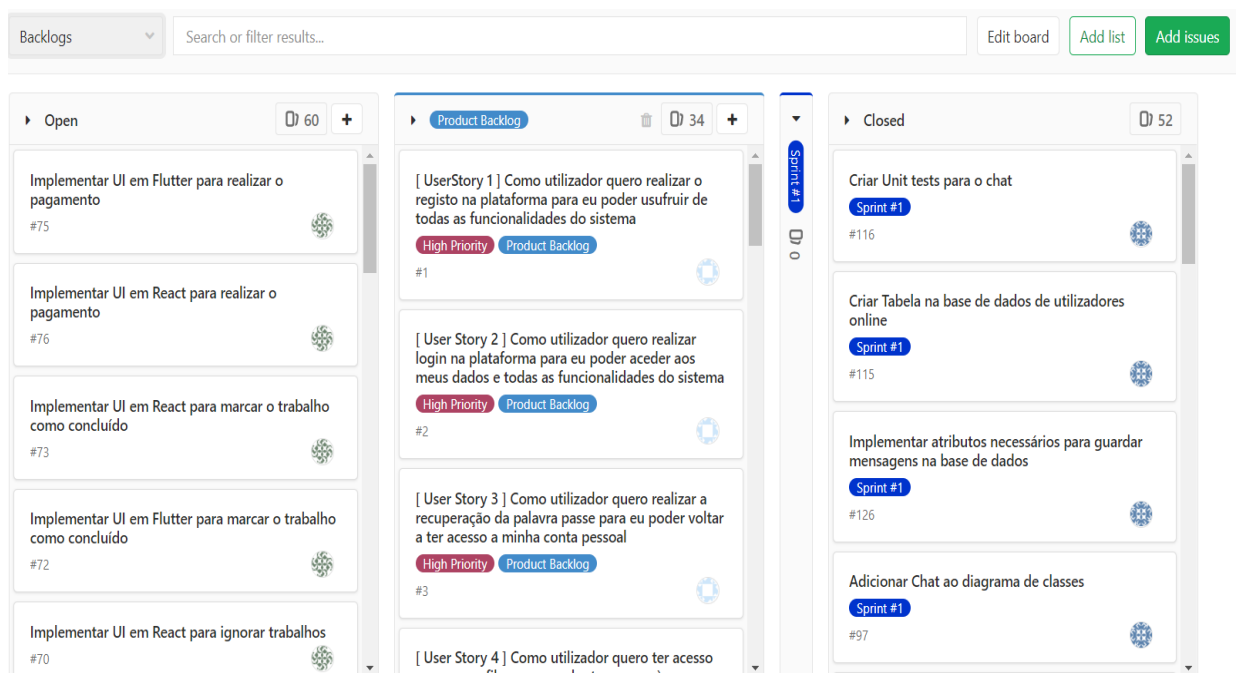


Figura 10 - Board Backlogs

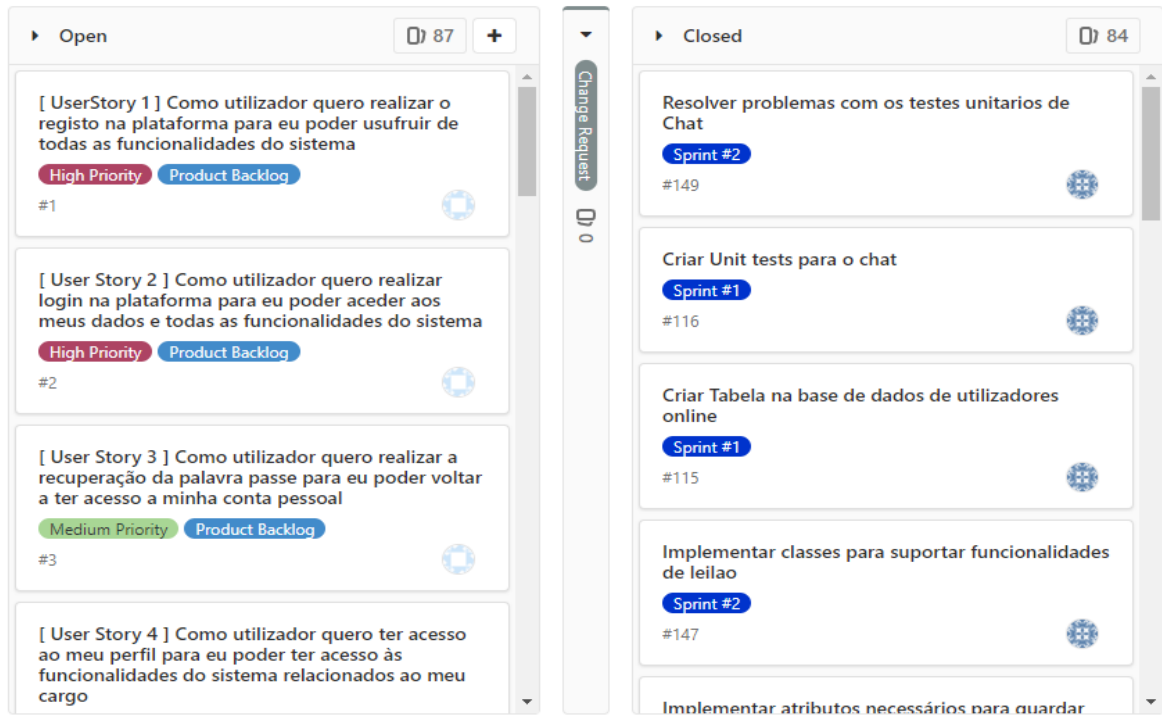


Figura 11 - Board Change Requests

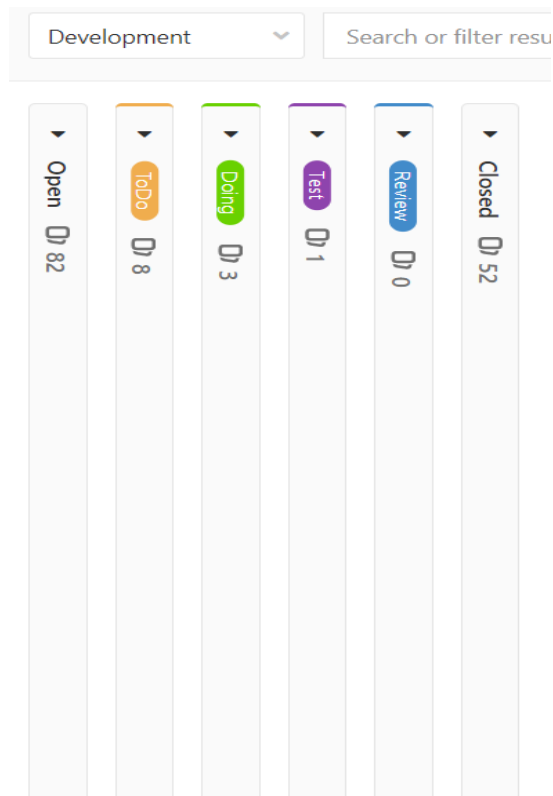


Figura 12 - Board Development

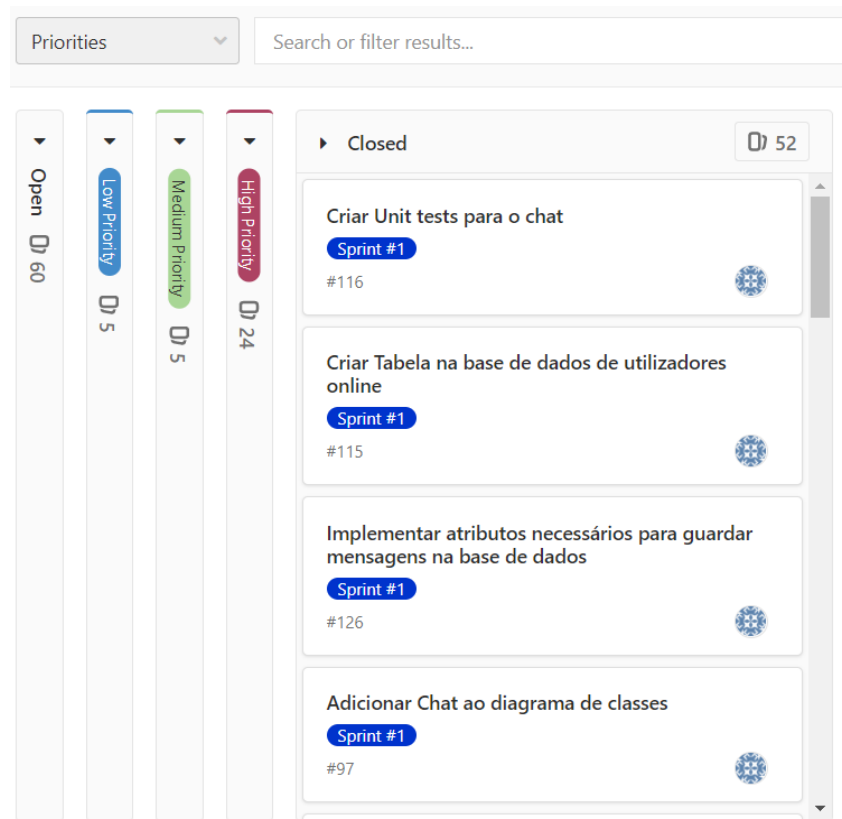


Figura 13 - Board Priorities

### Sprints como milestones:

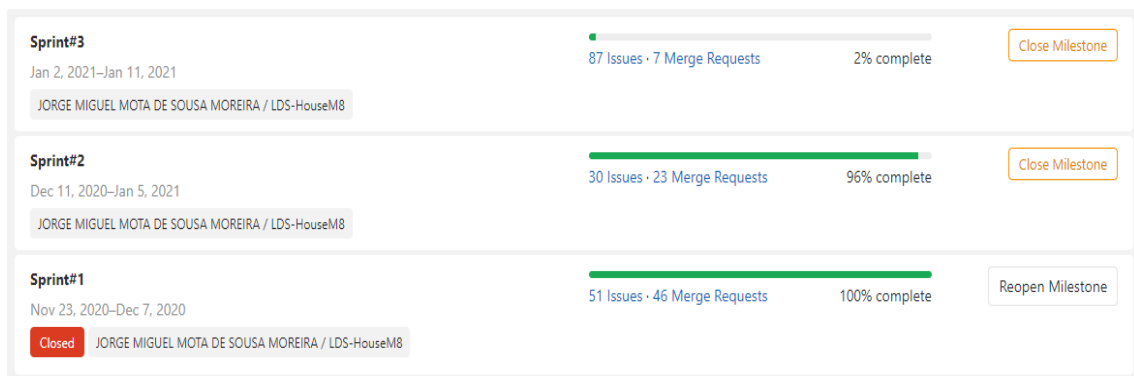


Figura 14 - Sprints

Nas subsecções seguintes serão apresentados os sumários das reuniões a cada sprint.

### 5.1.1 Sprint #1

Durante este sprint, temos como objetivo terminar o core da RestAPI que tem os seguintes requisitos:

- Implementação do Sistema de Login (com autenticação e autorização)
- Implementação das rotas para acesso aos dados dos utilizadores
- Implementação das rotas para acesso aos dados dos trabalhos
- Implementação das rotas para acesso aos dados dos trabalhadores
- Implementação da backend do sistema de chat
- Implementação das rotas para as funcionalidades das avaliações
- Implementação das rotas para as funcionalidades dos *reports*

### 5.1.2 Sprint #2

Durante este sprint, temos como objetivo:

- Melhorias à API e finalização
- Implementação do Sistema de Pagamentos utilizando API's externas
- Implementação da interface gráfica em Flutter com comunicação com a RestAPI

### 5.1.3 Sprint #3

Durante este sprint, temos como objetivo:

- Continuação da implementação da interface gráfica em Flutter para dispositivos móveis, utilizando a Rest API para estabelecer a comunicação com os dados
- Reformulação e finalização da Rest API



## 6 Conclusão

Posto isto, consideramos esta unidade curricular como uma das mais importantes do curso se não a mais importante, visto que foi uma excelente preparação para o que nos espera do estágio/projeto final. Acima de tudo, o trabalho de grupo foi decisivo para a realização do trabalho.

Foi concluído o objetivo principal de termos a HouseM8 completamente funcional onde principalmente os utilizadores podem procurar ou criar postagens de trabalhos e entrar em contacto para a marcação do mesmo e fazer o pagamento.