Logo

Description automatically generated

Software Project Management 2022/2023

**Gestão de Medicamentos**

D1.2.1 - Plano de Desenvolvimento de Software

### Autores

* Rúben Mendes
* Daniel Albino
* Diogo Silva
* Nuno Domingues
* Miguel Neves

### Revisto por:

* Nuno Domingues

### Aprovado por:

* Miguel Neves

**Indice:**

[1. Introdução](#_Toc117470805)

[2. Organização do projeto](#_Toc117470806)

[2.1. Ciclo de vida do projeto](#_Toc117470807)

[2.2. Entregas](#_Toc117470808)

[2.3. Documentos e registos de suporte](#_Toc117470809)

[2.4. Gestão de projeto](#_Toc117470810)

[3. Plano *Baseline* e Controlo](#_Toc117470811)

[3.1. Estimativas](#_Toc117470812)

[3.2. Escalonamento](#_Toc117470813)

[3.3. Rastreamento e Controlo](#_Toc117470814)

[4. Processo Técnico](#_Toc117470815)

[4.1. Arquitetura do sistema](#_Toc117470816)

[4.2. Ferramentas e Técnicas](#_Toc117470817)

# Introdução

O propósito deste projeto, tem como objetivo fundamental facilitar e auxiliar a gestão de medicamentos em ambiente hospitalar, com dois focos principais, que são as encomendas de novos medicamentos, sendo que esta funcionalidade estará à responsabilidade de um gestor de medicamentos e o aviso/alerta da toma de medicamentos dos vários utentes, funcionalidade que estará a responsabilidade dos enfermeiros.

Com isto pretendemos, tornar estas atividades que antes eram repetitivas e cansativas para as entidades envolvidas, num processo mais automatizado e fluído e assertivo.

Pretendemos, desenvolver uma interface gráfica intuitiva, em java e javafx, onde estarão implementadas todas as funcionalidades descritas no documento visão e âmbito.

# Organização do projeto

## Ciclo de vida do projeto

O nosso projeto baseia-se no modelo de ciclo de vida “*waterfall”,* ou seja, uma metodologia sequencial. Com isto queremos dizer que a gestão do projeto se divide em fases distintas, e que uma fase começa só quando a anterior estiver concluída.

Nesse sentido, descrevemos as três fases fundamentais da execução do projeto:

**Fase 2.1 - Especificação de requisitos**

* Especificação de requisitos de software

A equipa, discute com o cliente para perceber quais são as necessidades do mesmo e define já alguns requisitos iniciais funcionais e não-funcionais.

* Plano de risco

Deve ser identificado potenciais riscos que podem influenciar negativamente o projeto, estimando esse impacto.

* Plano de teste de aceitação

Essencialmente é formulado um plano de teste, com objetivo de testar a aplicação, para saber se esta responde às necessidades do cliente.

* Relatório Milestone

Ponto de reflexão, para perceber como é que a fase decorreu, e identificar pontos a melhorar.

**Fase 2.2 - Desenvolvimento de software**

* Arquitetura e design do projeto

A equipa discute e cria diagramas dos vários modelos de design que irá utilizar na implementação do projeto. O detalhe e visão clara de como irá funcionar é fundamental nesta fase.

* Código fonte

A principal parte da aplicação é desenvolvida, é feito um desenvolvimento modular, vários excertos de código, tendo em vista reduzir a complexidade. Cada membro da equipa é responsável por uma ou mais funcionalidades do projeto para garantir eficiência do desenvolvimento.

* Relatório Milestone

Ponto de reflexão, para perceber como é que a fase decorreu, e identificar pontos a melhorar.

**Fase 2.3 - Aceitação de software**

* Relatório de teste de aceitação

São executados testes ao software, de uma forma exaustiva de forma a assegurarmos qualidade. Relatório desses testes que representa uma descrição dos casos de teste de rendimento e de confiabilidade realizados ao software e descrição de eventuais fragilidades do sistema.

* Relatório de qualidade

Revisão do desenvolvimento do projeto, baseada no plano de garantia de qualidade.

* Relatório Milestone

Ponto de reflexão, para perceber como é que a fase decorreu.

## Entregas

|  |  |  |
| --- | --- | --- |
| **22 de novembro de 2022** | **13 de dezembro de 2022** | **20 de dezembro de 2022** |
| D2.1.1 - Software Requirements Specification | D2.2.1 - Software Architecture and Design | D2.3.1 - Acceptance Test Report |
| D2.1.2 - Risk Plan | D2.2.2 - Source Code | D2.3.2 - Quality Assessment Report |
| D2.1.3 - Acceptance Test Plan | D2.2.3 - Milestone M2.2 Report. | D2.3.3 - Milestone M2.3 Report. |
| D2.1.4 - Milestone M2.1 Report. |  |  |

## Documentos e registos de suporte

* Relatórios de reuniões.
* Relatórios de inspeção.
* GPS2223 - Team Log
* GPS2223 - Baseline Plan
* Mockups.
* GPS2223 - Earned Value Analysis
* GPS2223 - Project Estimation Report #1
* GPS2223 - Change Request
* GPS2223 - Review Report

Link para a pasta suporte, [aqui](https://isecpt.sharepoint.com/:f:/r/sites/GPS2223-Team14/Project%20Files/Support?csf=1&web=1&e=VIxMFC).

## Gestão de projeto

* Responsabilidades
  + Gestor de projeto: Nuno Domingues
  + Gestor de qualidade: Diogo Silva
  + Gestor de risco: Rúben Mendes
  + Gestor técnico: Daniel Albino
  + Gestor de cliente: Diogo Silva
  + Gestor de teste: Miguel Neves

# Plano *Baseline* e Controlo

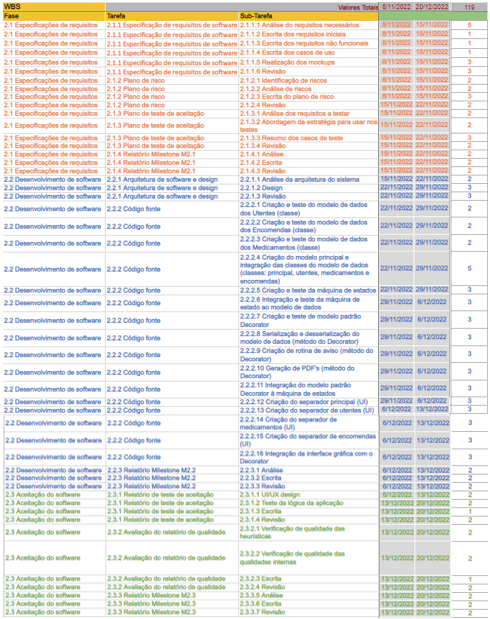
## Estimativas

Para as estimações das tarefas, a equipa decidiu adotar o planning poker, baseando-nos nas informações do documento [GPS2223-5\_Estimation.pdf](https://isecpt.sharepoint.com/:b:/r/sites/GPS2223/Shared%20Documents/General/Slides/GPS2223-5_Estimation.pdf?csf=1&web=1&e=c5xUuc) , através de uma aplicação online, que pode ser acedida [aqui.](https://planningpokeronline.com/)

A equipa decidiu usar as cartas: 0,1,2,3,5 e 8, estimando em horas para cada tarefa.

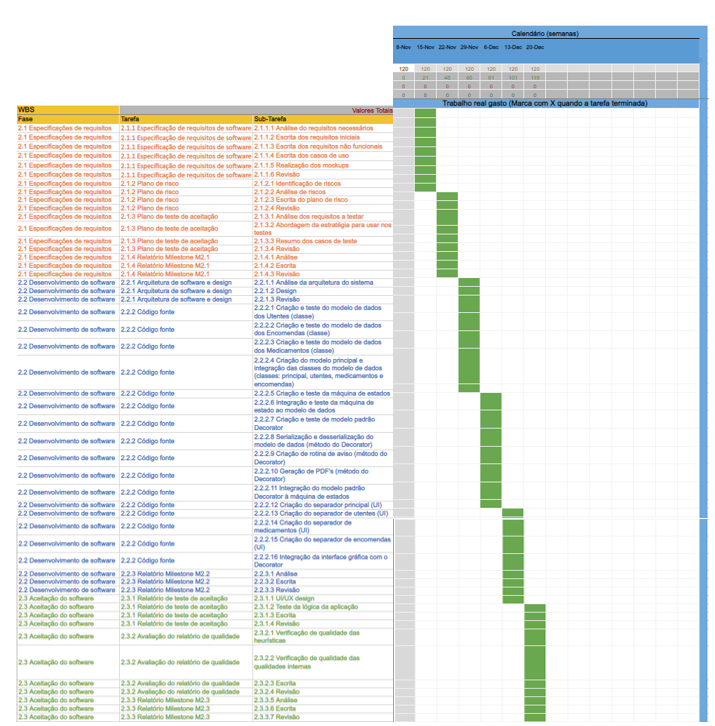
Nos casos de diferentes opinões, a equipa conversa e chega a um consenso, sobre qual carta a aplicar à tarefa.

Todas estas estimações estão presentes nas subtarefas no documento [GPS2223 - Project Estimation Report #1.xlsx.](https://isecpt.sharepoint.com/:x:/r/sites/GPS2223-Team14/Project%20Files/Support/GPS2223%20-%20Project%20Estimation%20Report%20%231.xlsx?d=wb7209ae33b6a4a8e898a14cf1d0e4f68&csf=1&web=1&e=X1STdS)



Link para o documento, [aqui](https://isecpt.sharepoint.com/:x:/r/sites/GPS2223-Team14/_layouts/15/Doc.aspx?sourcedoc=%7BCF8A2FB7-EF79-4592-9A44-4CD1CAE06165%7D&file=GPS2223%20-%20Earned%20Value%20Analysis.xlsx&action=default&mobileredirect=true).

## Escalonamento



Link para o documento, [aqui](https://isecpt.sharepoint.com/:x:/r/sites/GPS2223-Team14/_layouts/15/Doc.aspx?sourcedoc=%7BCF8A2FB7-EF79-4592-9A44-4CD1CAE06165%7D&file=GPS2223%20-%20Earned%20Value%20Analysis.xlsx&action=default&mobileredirect=true).

## Rastreamento e Controlo

De acordo com o nosso planeamento:

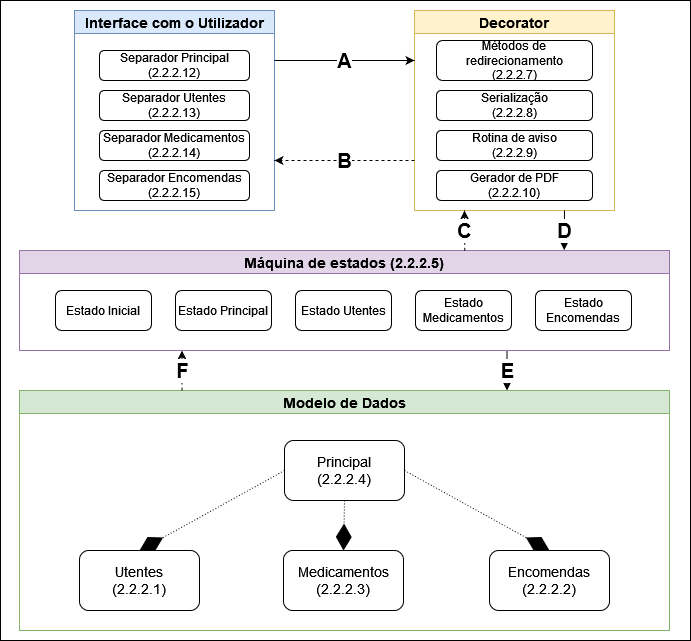
1. Na primeira semana será feita a análise as requisitos funcionais e não funcionais, a realização de casos de uso, mockups e análise de risco, é expectável um custo de 21 horas.
2. Na segunda semana, abordamos os requisitos que são alvo de teste, como também analisamos a arquitetura do sistema, tarefas que serão feitas dentro do “Budget” semanal.
3. Na terceira semana, no seguimento da análise da arquitetura do sistema, realizamos o design da mesma e começamos com a codificação do projeto, esperamos cumprir novamente 20 horas semanais.
4. A Quarta semana é dedicada ao desenvolvimento e teste unitário e de integração de novos módulos, um objetivo de 21 horas
5. Na quinta semana, é expectável terminarmos a codificação do projeto e realizarmos os testes aos dois principais módulos a interface com o utilizador e a parte lógica do trabalho, com 20 horas de trabalho previsto.
6. Na última semana com 18 horas, esperamos fazer e terminar o relatório de qualidade e o milestone.

# Processo Técnico

## Arquitetura do sistema

A arquitetura do sistema do nosso projeto de software, irá aplicar o modelo padrão, o *Finite State Machine* (máquina de estados), o modelo Decorator que irá fazer a ponte de ligação entre a User Interface e a máquina de estados. Ainda temos o modelo PropertyChange, relacionado com a atualização automática da interface gráfica.

|  |  |
| --- | --- |
| A | Envia dados e pede dados |
| B | Recebe dados/informações |
| C | Passagem de dados |
| D | Mediante aquilo que o utilizador pediu em A o decorator redireciona para a função adequada(por exemplo: ver lista de utentes) |
| E | Acede ao modelo de dados para inserir ou retornar informação |
| F | Retorno de informação |



No Diagrama acima, podemos visualizar a arquitetura do nosso sistema:

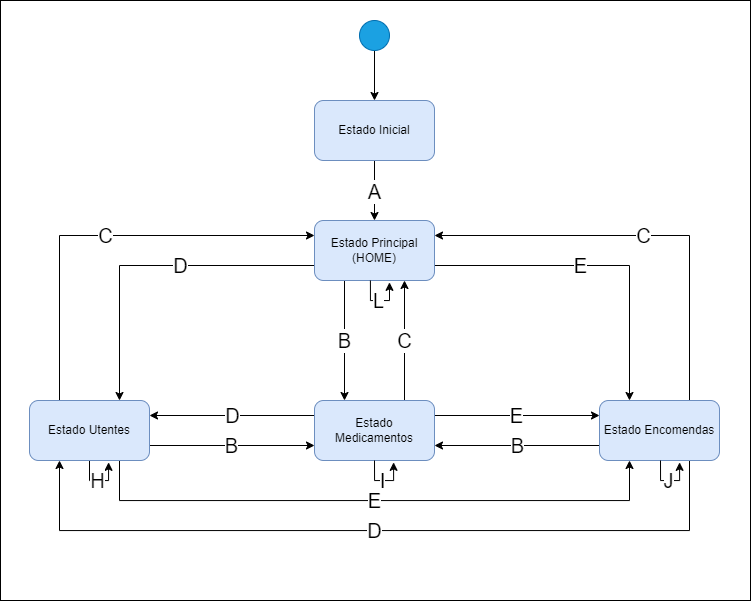
* No Modelo de Dados (caixa verde) irá haver uma classe **“Principal”**, que irá conter uma lista de Utentes (classe “**Utentes**”), uma lista de Medicamentos (classe “**Medicamentos**”) e uma lista de Encomendas (classe “**Encomendas**”). É então fácil de perceber, que os dados da aplicação (isto é, dados dos utentes, medicamentos, encomendas) estarão na classe “**Principal**”.
* Temos também uma Máquina de estados (caixa lilás), que vai permitir controlar que ações se podem fazer no contexto em que a aplicação está (por exemplo, é expectável que seja possível fazer encomendas se a aplicação está no “Estado Encomendas”. Isto já não faria sentido se estivesse no “Estado Utentes”.)
* Por fim temos o *design pattern*, Decorator (caixa amarela) e a Interface com o utilizador (caixa azul).

A interface sempre que necessitar consultar informações/dados do modelo de dados ou enviar dados para o modelo de dados, faz através do Decorator.

O Decorator é que depois trata de redirecionar para as funções adequadas, para devolver a informação ao utilizador ou inserir informação do utilizador no modelo de dados.

**Modelo máquina de estados:**

São definidos um número de estados no modelo de dados do projeto, que definem quais são as operações permitidas em cada um dos estados.



**PropertyChange (Alteração de propriedade):**

O Propertychange, permite a atualização automática da interface, quando há alguma alteração dos dados.

## Ferramentas e Técnicas

* Linguagem de programação:

Java com JavaFX (versão java17).

* Ferramentas:

Intellij (IDE, versão 2022.2.2)

Git

GitHub

StarUML

* Métodos/Técnicas:

Controlo de qualidade fazendo módulos de código, antes de reunir tudo no mesmo.