

Relatório Final



sara Silva 30042

Universidade Fernando Pessoa

Conteúdo

[Introdução 2](#_Toc93261218)

[Objetivo 3](#_Toc93261219)

[Referências 3](#_Toc93261220)

[Análise do produto 3](#_Toc93261221)

[Planeamento 4](#_Toc93261222)

[Descrição do problema 5](#_Toc93261223)

[Requisitos Funcionais 5](#_Toc93261224)

[Requisitos Não Funcionais 6](#_Toc93261225)

[Requisitos Software 6](#_Toc93261226)

[Arquitetura 7](#_Toc93261227)

[Diagrama de Sequência 8](#_Toc93261228)

[8](#_Toc93261229)

[8](#_Toc93261230)

[Diagrama de Classes 9](#_Toc93261231)

[Casos de Uso 9](#_Toc93261232)

[Restrições gerais 10](#_Toc93261233)

[Testes Unitários 10](#_Toc93261234)

[Conclusão 11](#_Toc93261235)

# Introdução

Pretende-se realizar um projeto tendo em conta o conhecimento adquirido nas aulas de Engenharia de Software. Este trabalho consiste no desenvolvimento de todas as etapas de um projeto de software que seja capaz de armazenar e gerir informação, focando no planeamento, especificação, modelização, implementação, testes e documentação do projeto.

A aplicação vai ser apresentada como requisito de avaliação à disciplina de Engenharia de Software.

# Objetivo

Neste projeto pretende-se que desenvolvam um Web Service que suporte a gestão de restaurantes que uma empresa executa para seus clientes. Um determinado cliente pode ter muitas reservas executados pela empresa e cada restaurante é constituído por muitas reservas que necessitam de um determinado número de horas para serem executadas. Cada Reserva é executada por um empregado do Restaurante e que possuem um dos cargos listados.

O cliente, numa primeira fase, deverá ser capaz de usar o Web Service para consultar informações sobre os seus restaurantes. O Web Service deve ser capaz de gerir as operações necessárias para as funcionalidades descritas. Numa segunda fase, a aplicação deverá ser capaz de representar a execução efetiva das reservas. Os empregados devem incluir períodos de tempo que dedicaram à reserva e o gestor do restaurante deve atribuir um percentual de conclusão à reserva. O cliente deverá poder consultar o estado do restaurante com base na comparação das estimativas iniciais.

# Referências

ESA PSS-05-02, Guide to the user requirements definition phase, Issue 1 Revision 1, Março 1995.

ESA PSS-05-03, Guide to the software requirements definition phase, Issue 1 Revision 1, Março 1995.

# Análise do produto

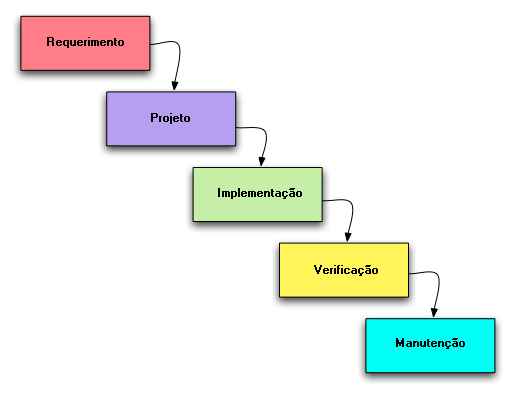
Este tipo aplicação existe para que seja mais fácil a sua gestão, procura gerir os projetos e as respetivas tarefas numa empresa, assim é mais fácil a consulta da informação. Está concentrada e pode ser acedida através do Web Service principal.

# Planeamento

O modelo escolhido para o projeto é o modelo Waterfall (cascata). O modelo Waterfall descreve um método de desenvolvimento que é linear e sequencial.

A vantagem de desenvolvimento em cascata é que ele permite a departamentalização e controlo de gestão. A programação pode ser definida com prazos para cada fase de desenvolvimento e um produto pode prosseguir no processo de desenvolvimento e ser entregue no prazo especificado.

Cada fase do desenvolvimento procede em ordem, sem qualquer sobreposição ou etapas passadas à frente.



# Descrição do problema

## Requisitos Funcionais

|  |  |  |
| --- | --- | --- |
| **Parte do Cliente** | | |
| UR000 | Gerir dados cliente | Obrigatório |
| UR001 | Consultar estado do projeto | Obrigatório |
| UR002 | Consultar informações projeto | Obrigatório |
| UR003 | Autenticação | Obrigatório |
| UR004 | Atualização dados | Obrigatório |
| UR005 |  | Obrigatório |

|  |  |  |
| --- | --- | --- |
| **Parte do Empregado** | | |
| UR000 | Gerir tempo de cada reserva | Obrigatório |
| UR001 | Gerir valor-hora | Obrigatório |
| UR002 | Gerir Cargo | Obrigatório |
| UR003 | Gerir Horário de Trabalho | Obrigatório |

## Requisitos Não Funcionais

|  |  |  |
| --- | --- | --- |
| UR000 | Interface gráfica | Obrigatório |
| UR001 | Garantir a segurança de todos os dados relativos aos utilizadores | Obrigatório |
| UR002 | Garantir o recurso a protocolos de segurança no processo de login | Obrigatório |
| UR003 | Sincronização com as bases de dados dos clientes, empregados, etc. | Obrigatório |
| UR004 | O sistema deve estar disponível em pelo menos dois idiomas, PT e ENG | Obrigatório |

## Requisitos Software

|  |  |  |
| --- | --- | --- |
| URS001 | A aplicação deve estar disponível para Android | Obrigatório |
| URS002 | A aplicação deve estar disponível para IOS | Desejável |
| URS003 | A aplicação deve seguir as boas práticas das Plataformas onde corre (desenho da interface, e utilização de recursos da plataforma). | Obrigatório |
| URS004 | A aplicação deve poder ser utilizada, mesmo com funcionalidades limitadas, quando não há conectividade com o serviço de dados. | Obrigatório |
| URS005 | O sistema deverá correr num navegador web recente. | Obrigatório |

# Arquitetura

Tendo em conta que vamos ter um sistema interativo onde o núcleo funcional do programa é independente da interface do utilizador, o padrão escolhido para projeto é o Model-view-controller (MVC).

Neste padrão temos uma divisão da aplicação em processamento, input e output. As ideias centrais por detrás do MVC são a reutilização de código e separação de conceitos.

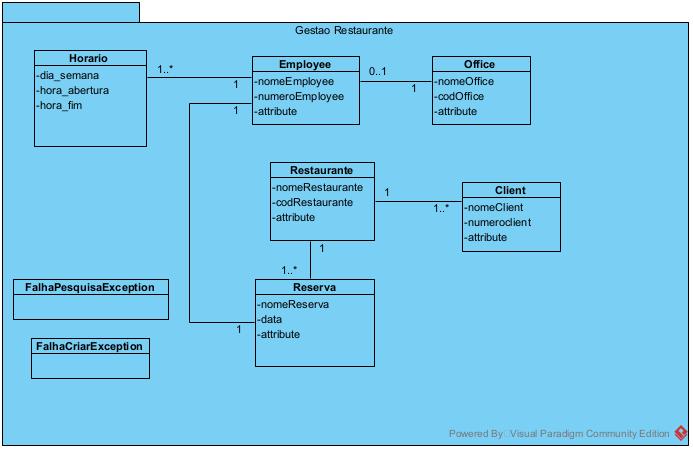


# Diagrama de Sequência

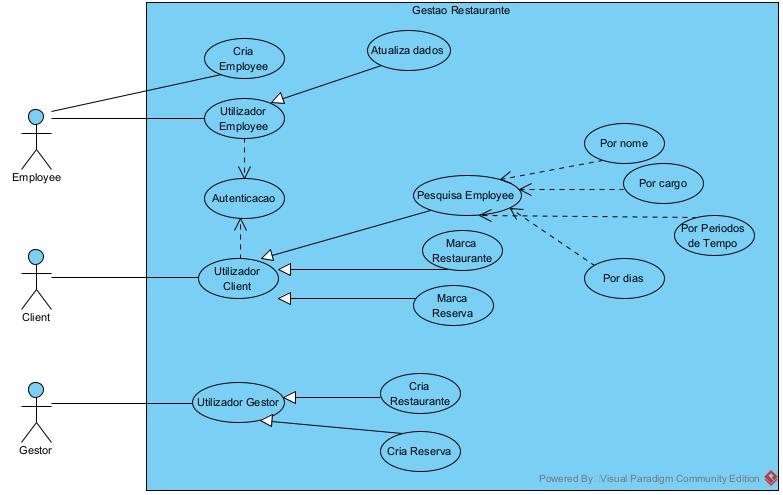
# 

# 

# Diagrama de Classes



# Casos de Uso



# Restrições gerais

Alguns dos critérios de qualidade poderão depender de outros fatores: periodicidade de utilização de dados do serviço de dados, existência de conectividade, etc...

Supõe-se que o serviço de dados tenha uma disponibilidade próxima de 100%, e que a largura de banda disponível seja suficiente para transferir todos dados necessários.

# Testes Unitários

Podem ser encontrados na pasta Tests/Unit alguns dos testes unitários realizados.

# Conclusão

No final do projeto, concluímos que não optamos pelo modelo mais correto para a implementação do projeto. O modelo escolhido foi o de cascata que, é um risco e pode ser um convite a falhas, como foi verificado no nosso projeto, ou seja, requeria um planeamento perfeito antes de se partir para a implementação do código. Pode-se dizer então, que o modelo em cascata é um modelo inflexível.

Para o projeto, poderia ter optado por outros modelos como o de prototipagem ou ágil, que são mais flexíveis e que seriam melhores para engenheiros que não sejam muito experientes.