Uma imagem com texto

Descrição gerada automaticamente

# **Projeto de Desenvolvimento de Software**

## Análise de Requisitos e Modelação

## Alunos

21140 - Pedro Vieira Simões

21149 - Duarte Ribeiro de Melo

10411 - Nuno Filipe Ribeiro Veloso

21152 - João Carlos da Costa Apresentação

21145 – Gonçalo Moreira da Cunha

### Docentes

Nuno Rodrigues

Óscar Ribeiro

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, Junho de 2022

[Figura 1 - Sprint Backlog (Alpha Phase - Sprint #3) 4](file:///C:\TrabalhosPraticos\PDS\Feedy\Relatório\Relatório%20Final.docx#_Toc100164618)

Conteúdo

[**Projeto de Desenvolvimento de Software** 1](#_Toc100164592)

[Análise de Requisitos e Modelação 1](#_Toc100164593)

[Alunos 1](#_Toc100164594)

[Docentes 1](#_Toc100164595)

[Especificações 4](#_Toc100164596)

[Sprint #1 4](#_Toc100164597)

[Sprint #2 4](#_Toc100164598)

[Release to Alpha 4](#_Toc100164599)

[Sprint #1 4](#_Toc100164600)

[Sprint #2 4](#_Toc100164601)

[Sprint #3 4](#_Toc100164602)

[Sprint Backlog 4](#_Toc100164603)

[Distribuição de Tarefas 5](#_Toc100164604)

[Progresso do Desenvolvimento da Sprint 5](#_Toc100164605)

[O que correu mal e melhoramentos 5](#_Toc100164606)

[Previsão da próxima Sprint 5](#_Toc100164607)

[Sprint #4 6](#_Toc100164608)

[Sprint #5 6](#_Toc100164609)

[Sprint #6 6](#_Toc100164610)

[Release to Beta 6](#_Toc100164611)

[Sprint #1 6](#_Toc100164612)

[Sprint #2 6](#_Toc100164613)

[Sprint #3 6](#_Toc100164614)

[Sprint #4 6](#_Toc100164615)

[Sprint #5 6](#_Toc100164616)

[Sprint #6 6](#_Toc100164617)

# Requests da API – O que está a vermelho NÃO está em EntityFramework

## Animal

* Get() – Request que devolve a informação de todos os animais da base de dados
* GetAnimalById(id\_animal) – Request que devolve a informação de apenas um animal, selecionando-o pelo seu ID.
* Post() – Request que adiciona/regista um novo animal na base de dados
* AddAnimal(idCliente, animal) – Request que permite adicionar um novo animal e cria uma nova ligação entre o mesmo e o cliente ao qual ele está associado
* SaveFile() – Request que permite associar uma imagem a um animal
* Patch(animal) – Request que permite a alteração da informação de um animal
* DelAnimal(id) – Request que permite arquivar um animal da base de dados
* VerifyAnimal(id) – Request que valida o animal na aplicação

## AvaliaçãoEstabelecimento

* Get() – Request que devolve a informação de todas as avaliações da base de dados
* GetAtivas() – Request que devolve apenas as avaliações de estabelecimentos ativos
* CheckAvaliacoesGerente(idFuncionario) – Request que permite um gerente visualizar todos as avaliações do seu estabelecimento
* Post() – Request que permite adicionar uma nova avaliação na base de dados
* AvaliarClinica(idUser, avaliacao) – Request que permite um cliente avaliar uma certa clinica

--- TUDO UPDATED CONSOANTE O CÓDIGO (14/04/2022 01:24) FALTA ATUALIZAR O RESTO

* Put() – Request que permite atualizar a informação de uma avaliação na base de dados
* Delete() – Request que permite remover uma avaliação da base de dados

## ClienteAnimal

* Get() – Request que devolve a informação de todas as ligações entre Cliente e Animal da base de dados
* Post() – Request que permite adicionar uma nova ligação entre Cliente e Animal na base de dados
* Put() – Request que permite atualizar a informação de uma ligação entre Cliente e Animal na base de dados
* Delete() – Request que permite apagar uma ligação entre Cliente e Animal da base de dados

## Cliente

* Get() – Request que permite devolver informação de todos os clientes na base de dados
* GetClienteByID() – Request que permite devolver informação de um cliente específico na base de dados
* Login() – Request que permite fazer a verificação se uma conta de cliente existe
* CreateToken() – Método que gera uma token de sessão para o utilizador ao fazer login
* Post() – Request que permite adicionar/registar um novo cliente na base de dados
* SaveFile() – Request que permite associar uma imagem a um cliente
* ChangeStatus() – Request que permite alterar o estado de um cliente
* Patch() – Request que permite atualizar/alterar informação de um cliente na base de dados
* ChangePlan() – Request que permite atualizar o plano do utilizador
* CancelPlan() – Request que permite o utilizador cancelar o seu plano de subscrição
* RescheduleService() – Request que permite o utilizador mudar a data de um serviço
* CancelDelivery() – Request que permite um utilizador cancelar uma encomenda
* DelCliente() – Request que permite arquivar um cliente da base de dados

## Encomenda

* Get() – Request que devolve a informação de todas as encomendas da base de dados
* Post() – Request que permite adicionar uma nova encomenda na base de dados
* Put() – Request que permite atualizar a informação de uma encomenda na base de dados
* Delete() – Request que permite apagar uma encomenda da base de dados

## EncomendaStock

* Get() – Request que devolve a informação de todas as ligações entre Encomenda e Stock da base de dados
* Post() – Request que permite adicionar uma ligação entre Encomenda e Stock na base de dados
* Put() – Request que permite atualizar a informação de uma ligação entre Encomenda e Stock na base de dados
* Delete() – Request que permite apagar uma ligação entre Encomenda e Stock da base de dados

## Estabelecimento

* Get() – Request que devolve a informação de todas os estabelecimentos da base de dados
* SearchEstabelecimentos - Request que devolve a informação de todas os estabelecimentos da base de dados [REPETIDA COM O REQUEST Get()]
* GetEstabelecimentoByID – Request que retorna a informação de um estabelecimento específico da base de dados
* GetAllEstabelecimentosNames() – Request que retorna os nomes de todos os estabelecimentos na base de dados
* Post() – Request que permite adicionar um novo estabelecimento na base de dados
* SaveFile() – Request que permite associar uma imagem a um estabelecimento
* Put() – Request que permite alterar toda a informação de um estabelecimento na base de dados
* Patch() – Request que permite alterar alguma informação de um estabelecimento na base de dados
* ChangeStatusEstabelecimento() – Request que permite atualizar o estado de um certo estabelecimento na base de dados
* CancelarParceria() – Request que permite apagar um estabelecimento da base de dados

## EstabelecimentoGerente

* Get() – Request que devolve a informação de todas as ligações entre Estabelecimento e Gerente da base de dados
* Post() – Request que permite adicionar uma ligação entre Estabelecimento e Gerente na base de dados HARDCODED
* Put() – Request que permite atualizar a informação de uma ligação entre Estabelecimento e Gerente na base de dados HARDCODED
* Delete() – Request que permite apagar uma ligação entre Estabelecimento e Gerente da base de dados HARDCODED

## EstabelecimentoHorario

* Get() – Request que devolve a informação de todos horários de cada estabelecimento da base de dados
* Post() – Request que permite adicionar um horário de um estabelecimento na base de dados
* Put() – Request que permite atualizar a informação de um horário de um estabelecimento na base de dados
* Delete() – Request que permite apagar um horário de um estabelecimento da base de dados

## Funcionario

* Get() – Request que retorna informação de todos os funcionários na base de dados
* GetFuncionarioByID() – Request que retorna a informação de um funcionário específico da base de dados
* Login() – Request que permite fazer a verificação se uma conta de funcionário existe
* CreateToken() – Método que gera uma token de sessão para o funcionário ao fazer login
* Post() – Request que adiciona/regista um funcionário na base de dados
* AddFuncionario() – Request que adiciona um funcionário num estabelecimento, caso este seja o primeiro do estabelecimento também tem o papel de gerente do mesmo
* SaveFile() – Request que permite associar uma imagem a um funcionário
* ChangeStatus() – Request que atualiza o estado de um funcionário na base de dados
* Patch() – Request que atualiza a informação de um funcionário na base de dados
* Put() – Request que altera toda a informação de um funcionário na base de dados
* Delete() – Request que apaga um funcionário da base de dados

## FuncionarioEstabelecimento

* Get() – Request que devolve as ligações entre Funcionário e Estabelecimento da base de dados
* Post() – Request que permite adicionar uma ligação entre Funcionário e Estabelecimento na base de dados
* Put() – Request que permite atualizar a informação de uma ligação entre Funcionário e Estabelecimento nto na base de dados
* Delete() – Request que permite apagar um horário de um estabelecimento da base de dados

## FuncionarioHorario

* Get() – Request que devolve a informação de todos horários de cada funcionário da base de dados
* Post() – Request que permite adicionar um horário de um funcionário na base de dados
* Put() – Request que permite atualizar a informação de um horário de um funcionário na base de dados
* Delete() – Request que permite apagar um horário de um funcionário da base de dados

## Lembrete

* Get() – Request que devolve a informação de todos os lembretes da base de dados
* Post() – Request que permite adicionar um lembrete na base de dados
* Put() – Request que permite atualizar a informação de um lembrete na base de dados
* RemoverReminder() – Request que permite remover um lembrete

## Morada

* Get() – Request que devolve a informação de todas moradas da base de dados
* Post() – Request que permite adicionar uma morada na base de dados
* Put() – Request que permite atualizar a informação de uma morada na base de dados
* Delete() – Request que permite apagar uma morada da base de dados

## Notificação

* Get() – Request que devolve a informação de todas as notificações da base de dados
* Post() – Request que permite adicionar uma notificação na base de dados
* Put() – Request que permite atualizar a informação de uma notificação na base de dados
* Delete() – Request que permite apagar uma notificação da base de dados

## Prescrição

* Get() – Request que devolve a informação de todas as prescrições da base de dados
* Post() – Request que permite adicionar uma precrição na base de dados
* Put() – Request que permite atualizar a informação de uma precrição na base de dados
* Delete() – Request que permite apagar uma precrição da base de dados

## ServicoCatalogo

* Get() – Request que devolve as ligações entre Serviço e Catálogo da base de dados
* Post() – Request que permite adicionar uma ligação entre Serviço e Catálogo na base de dados
* Put() – Request que permite atualizar a informação de uma ligação entre Serviço e Catálogo na base de dados
* Delete() – Request que permite apagar de uma ligação entre Serviço e Catálogo na base de dados

## Servico

* Get() – Request que devolve a informação de todos os serviços da Base de Dados
* Get\_Historico\_Cliente() – Request que retorna a informação de um serviço específico
* Post() – Request que adiciona/regista informação de um serviço na base de dados
* MarcarServico() – Request que adiciona/regista informação de um serviço na base de dados [REPETE COM Post()]
* Put() – Request que altera toda a informação de um certo serviço
* ChangeStatusServico() – Request que altera o estado de um certo serviço
* Delete() – Request que apaga um certo serviço da base de dados

## ServicoPrescricao

* Get() – Request que devolve as ligações entre Serviço e Prescrição da base de dados
* Post() – Request que permite adicionar uma ligação entre Serviço e Prescrição na base de dados
* Put() – Request que permite atualizar a informação de uma ligação entre Serviço e Prescrição na base de dados
* Delete() – Request que permite apagar de uma ligação entre Serviço e Prescrição na base de dados

## StockEstabeleciemento

* Get() – Request que devolve as ligações entre Stock e Estabelecimento da base de dados
* Post() – Request que permite adicionar uma ligação entre Stock e Estabelecimento na base de dados
* Put() – Request que permite atualizar a informação de uma ligação entre Stock e Estabelecimento nto na base de dados
* Delete() – Request que permite apagar de uma ligação entre Stock e Estabelecimento nto na base de dados

## Tratamento

* Get() – Request que devolve a informação de todas os tratamentos da base de dados
* Post() – Request que permite adicionar um tratamento na base de dados
* Put() – Request que permite atualizar a informação de um tratamento na base de dados
* Delete() – Request que permite apagar um tratamento da base de dados

# Especificações

## Sprint #1

TODO

## Sprint #2

TODO

# Release to Alpha

## Sprint #1

TODO

## Sprint #2

TODO

## Sprint #3

### Sprint Backlog

Uma imagem com texto, interior, captura de ecrã, armário

Descrição gerada automaticamente

Figura 1 - Sprint Backlog (Alpha Phase - Sprint #3)

### 

### Distribuição de Tarefas

* Melhoria de Diagrama de Classes – Pedro e Nuno
* Desenvolvimento final da Base de Dados – Pedro, Duarte e João
* Iniciação do Projeto API e associação da Base de Dados ao mesmo – Pedro, Duarte e Nuno
* Desenvolvimento de Controllers e criação de Requests base – João e Gonçalo
* Autenticação com Encriptação de Passwords e Tokens de Sessão – Duarte e João

Nesta sprint foram atribuídos mais membros de equipa por diferentes tarefas tendo em conta que foi necessário efetuar pesquisas com o âmbito de aprendizagem dos pontos referidos acima.

### Progresso do Desenvolvimento da Sprint

Esta sprint teve dois pontos distintos a focar: melhoramento de diagramas desenvolvidos em sprints anteriores e iniciação da API com associação à base de dados capaz de fazer manipulação de dados básica.

Os diagramas de classe e de entidade-relação apresentaram-se incompletos e com erros na última sprint, após uma breve pesquisa sobre as classes utilizadas em APIs (como referência, decidimos pesquisar por Entity Framework) e notou-se que seria necessário criar classe para cada entidade da Base de Dados, com isto, foi desenvolvido um novo diagrama de Classes no qual estão apresentadas todas as classes (principais) do programa. Em relação à base de dados, foi adicionada a cardinalidade que tinha em falta. Foi feita também uma pequena alteração no Cliente nos atributos relativos à password (Adicionada uma PassSalt, para fazer a encriptação e desencriptação da mesma)

Relativamente à API, inicialmente fizemos queries a partir de strings que eram inseridas na query da Base de Dados, todavia foram feitas pesquisas sobre a montagem da Entity Framework num projeto API e assim adaptamos alguns métodos e classes para a mesma.

Por fim, falando agora da autenticação e tokens de sessão, após algum estudo sobre o mesmo decidimos abordar a utilização de JWT (Json Web Token) na qual foram adicionas classes de autenticação e métodos de encriptação de password para as entidades respetivas.

### O que correu mal e melhoramentos

Apesar de um ótimo arranque na sprint, tendo a equipa finalizado o melhoramento de diagramas no primeiro dia da sprint, houve alguma dificuldade na adaptação de Entity Framework no projeto, o que nos levou a fazer os métodos de HTTP Requests a partir de strings de queries, o que consumiu algum tempo.

Também houve alguns conflitos na alteração da Base de Dados para adaptar os atributos da palavra-passe do utilizador ao fazermos Update-Database no projeto (a partir de Migration), o que nos levou a apagar a base de dados (SQL) e voltarmos a montá-la, desta vez em Code-First.

Por outro lado, notou-se a meio da sprint que a equipa de desenvolvimento desenvolveu novas “mecânicas” de pesquisa de informação com o fim de aprendizagem “self-learning”, o que motivou a mesma para futuros sprints.

### Previsão da próxima Sprint

Relativamente à próxima sprint (Alpha #4), espera-se continuar com o desempenho desta e até acrescentar mais algumas tarefas / casos de uso, conferindo mais alguma carga de trabalho, dado que durante a semana que se aproxima os membros constituintes terão mais disponibilidade. Apesar de ainda não se ter realizado a Sprint Planning com o cliente, para a Alpha #4 planearam-se as seguintes tarefas:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Previsão de Tarefas para Phase Alpha Sprint #4

## Sprint #4

## Sprint #5

# Release to Beta

## Sprint #1

## Sprint #2

## Sprint #3

## Sprint #4

## Sprint #5

## Sprint #6