**Entrega Trabalho Prático 5**

**Projeto Aplicado**

**Aluno/os:**

21140 - Pedro Vieira Simões

21145 – Gonçalo Moreira da Cunha

21152 – João Carlos da Costa Apresentação

**Professor/es: Eduardo Peixoto**

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, janeiro de 2023

IPCA GYM

# Resumo

Este trabalho prático, relativo à unidade curricular de **Projeto Aplicado,** propende a melhorar a performance de trabalho em equipa num desafio que irá explorar as necessidades de um smart campus, no IPCA e demonstrar técnicas e conceitos abordados inter e extracurricular.

A ideia do projeto será uma aplicação dividida em um sistema que pretende atingir principalmente docentes e universitários como público-alvo, e este divide-se em um login para ginásio (inclui gerente e funcionários) e outro para os clientes do ginásio. Ambas serão partilhadas em uma única aplicação com autorizações através do cadastro.

É pretendido através desta App satisfazer os utilizadores ao nível de consumos energéticos e uma boa experiência ao nível de funcionalidades que ajudem no percurso físico, através de multifuncionalidades que ajudem o cliente no treino e na alimentação, além de marcações de consultas e loja online.

Já o funcionário poderá gerir todas essas situações, tendo o posto de Gerente/Admin como uma autenticação superior na qual esse pode gerir os próprios funcionários e conta do ginásio.

Este é uma aplicação android desenvolvida essencialmente com base de dados Microsoft SQL Server, C# para API com Tokens de sessão e sistema de autenticação e Kotlin para Frontend.

Índice

[Resumo 3](#_Toc125064052)

[Índice de Imagens 7](#_Toc125064053)

[1. Introdução 9](#_Toc125064054)

[1.1. Contextualização 9](#_Toc125064055)

[1.2. Motivação e Objetivos 9](#_Toc125064056)

[1.3. Estrutura do Documento 9](#_Toc125064057)

[2. Produto 10](#_Toc125064058)

[2.1. Visão do Produto 10](#_Toc125064059)

[3. Processos gerais do funcionamento do produto 11](#_Toc125064060)

[3.1. Requisitos Funcionais 11](#_Toc125064061)

[3.1.1. Ginásio (Sistema) 11](#_Toc125064062)

[3.1.2. Cliente 11](#_Toc125064063)

[3.2. Requisitos Não Funcionais 12](#_Toc125064064)

[4. Organização do Grupo 12](#_Toc125064065)

[5. Stakeholders 13](#_Toc125064066)

[6. Partes não interessadas 13](#_Toc125064067)

[7. Benefícios 13](#_Toc125064068)

[8. Impacto Diário 14](#_Toc125064069)

[9. Cliente 14](#_Toc125064070)

[10. Utilizadores 14](#_Toc125064071)

[11. Modelo Canvas 14](#_Toc125064072)

[11.4.1. Ginásio e seus funcionários 15](#_Toc125064073)

[11.4.2. Utilizadores/Clientes do ginásio 15](#_Toc125064074)

[12. Grelha de autoavaliação 16](#_Toc125064075)

[13. Atas 17](#_Toc125064076)

[13.1. Entrega #1 17](#_Toc125064077)

[13.1.1. Ata #1 17](#_Toc125064078)

[13.2. Entrega #2 17](#_Toc125064079)

[13.2.1. Ata #1 17](#_Toc125064080)

[13.2.2. Ata #2 18](#_Toc125064081)

[13.2.3. Ata #3 18](#_Toc125064082)

[13.2.4. Ata #4 19](#_Toc125064083)

[13.3. Entrega #3 19](#_Toc125064084)

[13.3.1. Ata #1 19](#_Toc125064085)

[13.3.2. Ata #2 19](#_Toc125064086)

[13.3.3. Ata #3 20](#_Toc125064087)

[13.4. Entrega #4 20](#_Toc125064088)

[13.4.1. Ata #1 20](#_Toc125064089)

[13.4.2. Ata #2 20](#_Toc125064090)

[13.4.3. Ata #3 21](#_Toc125064091)

[13.4.4. Ata #4 21](#_Toc125064092)

[13.5. Entrega #5 21](#_Toc125064093)

[13.5.1. Ata #1 21](#_Toc125064094)

[14. Diagramas 22](#_Toc125064095)

[14.1. Diagrama Entidade-Relação 22](#_Toc125064096)

[14.2. Diagrama de Casos de Uso 24](#_Toc125064097)

[14.2.1. Caso de uso 1 – Gestão de Conta 24](#_Toc125064098)

[14.2.2. Caso de uso 2 – Loja online 25](#_Toc125064099)

[14.2.3. Caso de uso 3 – Aplicação 26](#_Toc125064100)

[14.2.4. Caso de uso 4 – Gestão de lotação 27](#_Toc125064101)

[14.3. Diagrama de atividades 28](#_Toc125064102)

[14.3.1. Cancelamento de serviço 28](#_Toc125064103)

[14.3.2. Reagendamento/Cancelamento de serviço 28](#_Toc125064104)

[14.4. Diagrama de estados 29](#_Toc125064105)

[14.5. Diagrama de sequência por ecrã 30](#_Toc125064106)

[15. Mockups 31](#_Toc125064107)

[15.1. Cliente 31](#_Toc125064108)

[15.1.1. Login 31](#_Toc125064109)

[15.1.2. Menu Principal 32](#_Toc125064110)

[15.1.3. Clique na foto 33](#_Toc125064111)

[15.1.4. Conta do utilizador 34](#_Toc125064112)

[15.1.5. Definições 35](#_Toc125064113)

[15.1.6. Editar perfil 36](#_Toc125064114)

[15.1.7. Atividade 37](#_Toc125064115)

[15.1.8. Treinos 38](#_Toc125064116)

[15.1.9. Nutrição 38](#_Toc125064117)

[15.1.10. Loja online 39](#_Toc125064118)

[15.2. Sistema do Ginásio 40](#_Toc125064119)

[15.2.1. Capacidade 40](#_Toc125064120)

[15.2.2. Gestão dos clientes 41](#_Toc125064121)

[15.2.3. Página inicial 42](#_Toc125064122)

[15.2.4. Loja online 43](#_Toc125064123)

[15.2.5. Atividade 44](#_Toc125064124)

[16. Cronograma 45](#_Toc125064125)

[17. Código implementado para o projeto 46](#_Toc125064126)

[17.1. Base de Dados 46](#_Toc125064127)

[17.2. Back-End 46](#_Toc125064128)

[17.2.1. Programação por camadas 47](#_Toc125064129)

[17.2.2. JWT Tokens e Authentication 48](#_Toc125064130)

[17.2.3. Funcionalidade de Login (Encriptação de password) 50](#_Toc125064131)

[17.2.4. Documentação de código com OpenAPI 51](#_Toc125064132)

[17.3. Front-end 53](#_Toc125064133)

[17.3.1. Activities 53](#_Toc125064134)

[17.3.2. Objeto request 55](#_Toc125064135)

[17.3.3. Custom View 56](#_Toc125064136)

[17.3.4. Requests 58](#_Toc125064137)

[17.3.4.2. Post 61](#_Toc125064138)

[17.3.4.3.Patchs 62](#_Toc125064139)

[17.3.4.4.Delete 63](#_Toc125064140)

[17.3.4.5.Login 64](#_Toc125064141)

[17.3.5. Data Classes 65](#_Toc125064142)

[17.3.6. Activity Classes 67](#_Toc125064143)

[17.3.6.1. Cliente 67](#_Toc125064144)

[17.3.6.2. Funcionário 72](#_Toc125064145)

[17.3.7. Resources 78](#_Toc125064146)

[17.4. Componentes físicos 79](#_Toc125064147)

[18. Avaliação 80](#_Toc125064148)

[19. Conclusão 80](#_Toc125064149)

[20. Bibliografia 81](#_Toc125064150)

[21. Assinaturas 82](#_Toc125064151)

# Índice de Imagens

[Figura 1 - Diagrama de Entidade-Relação 22](#_Toc125063972)

[Figura 2 - Diagrama Casos de Uso 1 (Gestão de Conta) 24](#_Toc125063973)

[Figura 3 - Diagrama de Casos de Uso 2 (Loja Online) 25](#_Toc125063974)

[Figura 4 - Diagrama de Casos de Uso 3 (Aplicação) 26](#_Toc125063975)

[Figura 5 - Diagrama de Casos de Uso 4 (Gestão de lotação) 27](#_Toc125063976)

[Figura 6 - Diagrama de Atividades - Cancelar Serviço 28](#_Toc125063977)

[Figura 7 - Diagrama de Atividades - Reagendamento de Serviço 29](#_Toc125063978)

[Figura 8 - Diagrama de Estados - Marcação de Consulta 29](#_Toc125063979)

[Figura 9 - Login do Utilizador 31](#_Toc125063980)

[Figura 10 - Utilizador - Página inicial 32](#_Toc125063981)

[Figura 11 - Utilizador - Menu 33](#_Toc125063982)

[Figura 12 - Utilizador - Conta 34](#_Toc125063983)

[Figura 13 - Utilizador - Definições 35](#_Toc125063984)

[Figura 14 - Utilizador - Editar perfil 36](#_Toc125063985)

[Figura 15 - Utilizador - Atividade 37](#_Toc125063986)

[Figura 16 - Utilizador - Treino 38](#_Toc125063987)

[Figura 17 - Utilizador - Nutrição 38](#_Toc125063988)

[Figura 18 - Utilizador - Loja Online 39](#_Toc125063989)

[Figura 19 - Sistema - Capacidade 40](#_Toc125063990)

[Figura 20 - Sistema - Gestão de clientes 41](#_Toc125063991)

[Figura 21 - Sistema - Página inicial 42](#_Toc125063992)

[Figura 22 - Sistema - Loja online 43](#_Toc125063993)

[Figura 23 - Sistema - Atividade 44](#_Toc125063994)

[`Figura 24 - Diagrama de Gantt 45](#_Toc125063995)

[Figura 25 - Microsoft SQL Server Management Studio (SSMS) 46](#_Toc125063996)

[Figura 26 - Dependências API Layer 47](file:///C:\Users\joaoc\OneDrive\Ambiente%20de%20Trabalho\Ipca_Gym\Organização\Entregas%20PA\Entrega_5\Especificação_Projeto_Aplicado.docx#_Toc125063997)

[Figura 27 - Dependências BLL Layer 47](file:///C:\Users\joaoc\OneDrive\Ambiente%20de%20Trabalho\Ipca_Gym\Organização\Entregas%20PA\Entrega_5\Especificação_Projeto_Aplicado.docx#_Toc125063998)

[Figura 28 - Dependências DAL Layer 47](file:///C:\Users\joaoc\OneDrive\Ambiente%20de%20Trabalho\Ipca_Gym\Organização\Entregas%20PA\Entrega_5\Especificação_Projeto_Aplicado.docx#_Toc125063999)

[Figura 29 - Login de um funcionário (Back-End) 48](#_Toc125064000)

[Figura 30 - Método auxiliar CreateTokenFuncionário() 49](#_Toc125064001)

[Figura 31 - Headers de Authorize num Request HttpPatch 49](#_Toc125064002)

[Figura 32 - Métodos auxiliares CreatePasswordHash() e VerifyPasswordHash() 50](#_Toc125064003)

[Figura 33 - Tratamento de password no Request HttpPost de um Funcionário 50](#_Toc125064004)

[Figura 34 - Verificação de palavra-passe no Login de um Funcionário 50](#_Toc125064005)

[Figura 35 - Geração documentação no Swagger 51](#_Toc125064006)

[Figura 36 - Propriedades da Solução do projeto 51](#_Toc125064007)

[Figura 37 - Documentação de métodos 52](#_Toc125064008)

[Figura 38 - Documentação de modelos de dados 52](#_Toc125064009)

[Figura 39 - Resultado documentação no Swagger 52](#_Toc125064010)

[Figura 40 – Atividade Cliente Classes 53](#_Toc125064011)

[Figura 41 - Atividade Funcionaria classes 54](#_Toc125064012)

[Figura 42 - Data classes 54](#_Toc125064013)

[Figura 43 – Requests 55](#_Toc125064014)

[Figura 44 - Custom View 56](#_Toc125064015)

[Figura 45 - Custom View - Valores reproduzidos 56](#_Toc125064016)

[Figura 46 - Custom View - Código desenvolvido 56](#_Toc125064017)

[Figura 47 - Custom View – Demonstração 57](#_Toc125064018)

[Figura 48 - get all 58](#_Toc125064019)

[Figura 49 – GetByID 59](#_Toc125064020)

[Figura 50 - Get By Token 60](#_Toc125064021)

[Figura 51 - Post Funcionario Request 61](#_Toc125064022)

[Figura 52 - Post Horario Funcionario request 61](#_Toc125064023)

[Figura 53 - Patch Funcionario Request 62](#_Toc125064024)

[Figura 54 - Patch Loja Request 62](#_Toc125064025)

[Figura 55 - Delete Produto 63](#_Toc125064026)

[Figura 56 - Login Request 64](#_Toc125064027)

[Figura 57 - Data Classe - Cliente 65](#_Toc125064028)

[Figura 58 - Data Classe - Cliente (construtor) 65](#_Toc125064029)

[Figura 59 - Propriedade da classe Cliente (enviar dados) 66](#_Toc125064030)

[Figura 60 - Propriedade da classe do Cliente (receber dados) 66](#_Toc125064031)

[Figura 61 - Login Cliente 67](#_Toc125064032)

[Figura 62 - Recover pass (Cliente) 68](#_Toc125064033)

[Figura 63 - Página Inicial Cliente Exemplo Gets 69](#_Toc125064034)

[Figura 64 - Post Classificação 70](#_Toc125064035)

[Figura 65 - Patch cliente 70](#_Toc125064036)

[Figura 66 - Delete pedido loja 71](#_Toc125064037)

[Figura 67 – Session Token 72](#_Toc125064038)

[Figura 68 - Login funcionário 72](#_Toc125064039)

[69 - Change Password (Funcionário) 73](#_Toc125064040)

[Figura 70 - Post cliente pelo funcionário 74](#_Toc125064041)

[Figura 71 - Get produtos 74](#_Toc125064042)

[Figura 72 - Patch exercicio (1) 75](#_Toc125064043)

[Figura 73 - Patch exercicio (2) 75](#_Toc125064044)

[Figura 74 - Patch exercicio (3) 76](#_Toc125064045)

[Figura 75 - Patch exercicio (4) 76](#_Toc125064046)

[Figura 76 - Delete de um exercício 77](#_Toc125064047)

[Figura 77 - Organização do Front-End (Resources) 78](#_Toc125064048)

[Figura 78 - Componente físico, Arduino com Sensor RFID-RC522 79](#_Toc125064049)

[Figura 79 - Autoavaliação 80](#_Toc125064050)

[80 - Assinaturas 82](#_Toc125064051)

# Introdução

## Contextualização

Este projeto tem como objetivo implementar soluções e desafios propostos pelo professor Eduardo Peixoto, a fim de consolidar os conhecimentos adquiridos durante a unidade curricular do projeto e em outras disciplinas como ISI através da API e PDM através do Front-End, entre muitos outros conhecimentos adquiridos ao longo do percurso acadêmico.

Enquadra-se em uma ideologia de Smart Campus, com foco na área da saúde e atividade física, mas também tem o objetivo indireto de reduzir gastos energéticos dos usuários da aplicação.

O projeto será documentado detalhadamente para mostrar todas as abordagens utilizadas para sua realização.

## Motivação e Objetivos

A ideia de um sistema para o ginásio foi originada pela ideia de futuramente o IPCA vir a ter mais instalações à medida que este vai crescendo e desta forma existir uma forma de gerir o mesmo e ainda ajudar os clientes.

Temos por objetivos pessoais:

* Cimentar conhecimentos obtidos ao longo do percurso académico;

Objetivos do projeto:

* Criar um sistema que ajude a gerir um ginásio, ao nível de:
  + Lotação;
  + Utilizadores;
  + Artigos de venda;
  + Fluxo de entradas/saídas;
  + Etc;
* Criar uma aplicação que ajude os utilizadores, ao nível de:
  + Procurar as melhoras horas de utilização do ginásio;
  + Facilitar o treino;
  + Entregar uma gama de produtos e planos (nutrição e treino);
  + Etc;

## Estrutura do Documento

O documento está estruturado de forma que seja de simples leitura. Existe recurso a referências de material fornecido pelo professor Eduardo Peixoto e/ou referências a excertos de Web grafia.

Este trabalho encontra-se dividido em grupos, de forma a facilitar a procura e associação face ao material fornecido pelo docente.

# Produto

## 2.1. Visão do Produto

Dentro dos subtópicos possíveis encaixados no Smart Campus vai ser abordado a Saúde. Foi decidido toda uma construção em torno do desenvolvimento android que visa à nossa universidade acompanhar a vida saudável e atlética dos estudantes.

O IPCA GYM nasce após notar-se a necessidade desse mesmo acompanhamento e a falta de um setor que permita a atividade aos jovens, no sentido de incentivar aos estudantes a realizar um estilo de vida saudável.

Será então possível aos estudantes terem um acompanhamento mobile da sua atividade física, tal como os diferentes exercícios que pode fazer ao longo do seu treino.

Os gestores do ginásio conseguirão fazer uma monitorização de todas as pessoas inscritas no ginásio, já que, em conjunto com outra unidade curricular, irá ser implementado um sistema externo para gestão de acesso através de um chip/cartão eletrónico.

Este projeto visa alcançar este objetivo através da implementação de uma aplicação Mobile e de hardware de gestão de acessos para que se torne mais cómoda a utilização da mesma.

# Processos gerais do funcionamento do produto

### Requisitos Funcionais

### Ginásio (Sistema)

* Gestão da lotação atual
  + Quantidade atual (avisos caso esteja perto do limite)
  + Quantidade de pessoas que foram nesse dia
  + Quantidade de pessoas que foram nesse mês
  + Quantidade de pessoas que já saíram
* Gestão de clientes
  + Registo
  + cancelar "contrato"
  + alterar dados biométricos
* Gestão de avaliações
* Gestão de pedidos da loja online
* Histórico de entradas e saídas de cada user (data passagem, dizer se é entrada ou saída, id da pessoa que entrou ou saiu, nome dessa pessoa)

### Cliente

* Login
* Dados biométricos
* Planos de treino
  + planos
    - emagrecer
    - massa muscular
    - definir
    - pernas, braços, peito, …
* Planos de nutrição
* Pode avaliar o ginásio/aplicação
* Marcar consulta de medições/nutrição
* Ver evolução ao nível das medidas musculares, …
* Gestão de conta
  + editar alguns dados
  + cancelar "contrato"
* Loja online

### Requisitos Não Funcionais

* **Implementação:** linguagens Kotlin e C#, com recurso a base de dados SQLServer e hardware Arduino; (é provável que sofra alterações tendo em conta que estamos a ter cadeiras em simultâneo com esta)
* **Usabilidade:** aplicação de uso simples bastante didática;
* **Interface:** aplicação mobile para uma utilização mais flexível quando comparada com desktop, etc;
* **Compatibilidade:** o sistema deverá ser compatível com a versão atual ou mais recente do sistema operativo Android;
* **Eficiência no software:** o software será eficaz sem causar muito tempo de espera entre ações;
* **Interoperabilidade:** sistema terá comunicação com a base de dados.

# Organização do Grupo

O grupo 7 da cadeira de Projeto Aplicado da Licenciatura de Engenharia de Sistemas Informáticos pós-laboral (LESI-PL) é constituído pelos seguintes membros de grupo: Pedro Simões, Gonçalo Cunha e João Apresentação.

O grupo tem como orientador o professor Eduardo Peixoto, que poderá participar nas reuniões de grupo, para auxiliar na concretização dos objetivos do projeto.

A nível de cargos distribuídos pelo grupo, temos:

* Gonçalo Cunha - Product Owner, Programador
* João Apresentação - Scrum Master, Programador
* Pedro Simões - Programador

**Product Owner** tem o papel de fazer a definição da visão do projeto, supervisionamento e avaliação das etapas de desenvolvimento.

**Scrum Master** tem o papel de facilitar as sprints, ajudar a equipa a manter-se concentrada nas sprints e fazer o planeamento e organização da mesma.

**Programador** tem a função desenvolver o software em si, obedecendo às tarefas fornecidas pelo Scrum master e de realizar as suas tarefas até ao fim de cada sprint.

Não está projetado a alteração de cargos entre membros de grupo, exceto em ocasiões especiais, das quais serão devidamente registadas com os membros que sofreram a alteração, data e razões da mesma.

# Stakeholders

* **Ginásio (IPCA):** interesse em oferecer uma plataforma aos seus membros e um sistema de gestão desses aos seus funcionários;
* **Cliente/Membro**: interesse em usar a aplicação e usufruir do ginásio;
* **Funcionário (Personal Trainers/Gerente):** interesse em gerir o estado e propriedades do ginásio;
* **Nutricionista:** interesse pelo fornecimento de dados nutricionais e marcação de serviços relacionados com o mesmo;
* **Fornecedores Comerciais:** interesse pelo fornecimento de roupa e alimentos a serem disponibilizados na loja online;
* **IPCA (ou outro estabelecimento académico):** interesse pela atração de novos estudantes e progressão do estabelecimento.

# Partes não interessadas

Todo o tipo de empresas e estabelecimentos não conectados à indústria da saúde, desporto, alimentar (alimentos de treino) e têxtil (roupa de treino).

# Benefícios

De uma forma generalizada o projeto a ser desenvolvido pretende fundamentar a ideia de uma vida saudável em cooperativo com campus universitário, que não só irá atrair os estudantes como irá tornar este politécnico mais destacado e importado com a saúde dos seus alunos.

De uma forma mais ao nível do que será desenvolvido serão beneficiados os stakeholders pelos seguintes aspetos:

* Aplicação mobile com o intuito de:
  + Promoção de uma vida saudável;
  + Facilitar na ida ao ginásio através da visualização de lotação do estabelecimento;
  + Ajudar no treino através dos planos fornecidos;
  + Ajudar na alimentação/nutrição;
  + Fornecer uma loja online de roupa e alimentação;
  + Sistema de ginásio que facilita a gestão de:
    - Lotação;
    - Experiência dos clientes;
    - Contratos;
    - Dados dos clientes;

# Impacto Diário

Atualmente muitas das vezes consome-se tempo e dinheiro a ir a determinado estabelecimento sendo que este poderá ter atingido a sua lotação máxima ou mesmo ter um número elevado de utilizadores, o que pode ser resolvido através de uma gestão desse tipo de dados. De certa forma este sistema vem como forma de resolver e criar um impacto positivo na sociedade ao nível de desperdício de tempo e recursos.

Além disso com a existência de uma App mobile é possível visualizar todo esse acesso de forma simples e eficaz. O cliente ainda terá facilidade nos seus treinos e progressos o que torna toda esta experiência bastante satisfatória.

Apesar de tudo isto, o projeto tem como maior impacto a aposta numa vida diariamente saudável de cada estudante.

# Cliente

O cliente do nosso projeto será o ginásio pertencente ao instituto académico (IPCA). Irá lhe ser disposto um sistema de gestão de acessos (cartões e leitor de cartões) juntamente com uma aplicação a ser oferecida aos seus membros/utilizadores.

# Utilizadores

Os utilizadores serão única e exclusivamente aqueles que sejam membros do ginásio e seus clientes. Estes terão um cartão de acesso ao recinto e ainda uma conta para uso da aplicação.

# Modelo Canvas

* 1. Parcerias-Chave
* Ginásios;
* Personal Trainers;
* Empresas para fornecimento de roupa e alimentos desportivos;
* Institutos académicos (no caso IPCA);
  1. Atividades-Chave
* Marketing;
* Desenvolvimento e manutenção do software;
* Desenvolvimento e manutenção do hardware;
* Suporte.
  1. Recursos-Chave
* Desenvolvedores de software;
* Desenvolvedores de hardware;
* Marcas;
* Contratos com os parceiros-chave.
  1. Proposta de valor

### Ginásio e seus funcionários

* Experiência facilitada na gestão de lotação, dados biométricos e avaliações dos seus clientes.

### Utilizadores/Clientes do ginásio

* Forma mais eficiente e fácil de verificar a lotação do ginásio;
* Experiência personalizada da sua alimentação e treino;
* Existência de loja online de pesquisa simples.
  1. Relacionamento com clientes
* Sistema de avaliações;
* Suporte técnico.
  1. Canais
* Aplicação mobile;
* Parcerias;
* Marketing;
* Mídia social.
  1. Segmentos de clientes
* Ginásios;
* Funcionários do ginásio;
* Clientes do ginásio.
  1. Custos
* Desenvolvedores de software;
* Desenvolvedores de hardware;
* Equipa de instalação do sistema;
* Hardware;
* Marketing.
  1. Receitas
* Percentagem de cada contrato individual;
* Percentagem de vendas da loja online.

# Grelha de autoavaliação

A autoavaliação do grupo será feita em cada entrega, até ao final deste projeto.

As notas atribuídas para cada membro de grupo serão discutidas entre os mesmo em reuniões semanais, estas serão calculadas tendo em conta um sistema de pontuação que apresenta diversos parâmetros e os seus pesos.

O documento oficial que contém as avaliações encontra-se num ficheiro externo ao relatório.

Parâmetros de avaliação de cada membro de grupo:

* Assiduidade (**5** pontos)
* Participação (**2.5** pontos)
* Comunicação (**1** ponto)
* Respeito entre colegas (**1** ponto)
* Cumprimento de prazos (**2.5** pontos)
* Organização (**2** pontos)
* Iniciativa (**3** pontos)
* Qualidade de trabalho (**2** pontos)
* Autonomia (**1** ponto)

# Atas

Esta secção apresentará todas as atas feitas até à data atual, identificando o que foi feito ao longo de cada sprint em uma reunião, indicando o local da reunião e hora, todos os membros de grupo devem estar presentes em cada reunião. Neste documento estarão apresentadas as atas de forma mais compacta.

As atas apresentam-se também em documentos externos, estando estas devidamente separadas.

## Entrega #1

### Ata #1

**Local da Reunião:** Online (via Discord)

**Hora de Início:** 22:30 **Hora de Fim:** 23:45

**Ordem de trabalhos:**

1. Fez-se um brainstorming sobre a ideia do projeto que envolvesse o tema Smart Campus, que foi decidida nesta reunião.
2. Escolha de um nome para a aplicação a ser desenvolvida, o grupo escolheu o nome IPCA GYM
3. Foi feita a distribuição de tarefas para a 1ª entrega do projeto, nos quais foi tratado o documento do mesmo, Mockups, business Model Canvas e levantamento de requisitos

## Entrega #2

### Ata #1

**Local da Reunião:** Online (via Discord)

**Data:** 22 de outubro de 2022

**Hora de Início:** 23:30 **Hora de Fim:** 01:30

**Ordem de trabalhos:**

1. Foi efetuada a distribuição de tarefas para a próxima entrega, nos quais estão incluídos a resolução de:
   1. Diagrama de Componentes;
   2. Diagrama de casos de uso;
   3. Diagrama de classes;
   4. Diagrama de Business Process Model and Notation (BPMN);
   5. Diagrama de Atividades;
   6. Diagrama de Estados;
   7. Diagrama de Entidade-Relação.
2. Discussão de ideias para o futuro diagrama Entidade-Relação, que seria a base do grupo para fazer a Datava-se para a aplicação, foi usado como referência o Diagrama de Classes.
3. Foi pensado também ideias de logótipos para o IPCA GYM.

### Ata #2

**Local da Reunião:** Online (via Discord)

**Data:** 31 de outubro de 2022

**Hora de Início:** 23:30 **Hora de Fim:** 01:30

**Ordem de trabalhos:**

1. Foi efetuada a revisão e análise dos diagramas feitos até à data presente, nos quais estão incluídos:
   1. Diagrama de Componentes;
   2. Diagrama de casos de uso;
   3. Diagrama de classes;
   4. Diagrama de Business Process Model and Notation (BPMN);
   5. Diagrama de Atividades.

Ficaram alguns diagramas em falta, que ficou decidido previamente que iriam ser feitos mais tarde.

1. Início do diagrama de Entidade-Relação.
2. Foi distribuída as tarefas para a semana seguinte de forma que todos os membros consigam tirar o melhor das suas capacidades para as mesmas.

### Ata #3

**Local da Reunião:** Online (via Discord)

**Data:** 08 de novembro de 2022

**Hora de Início:** 23:30 **Hora de Fim:** 01:30

**Ordem de trabalhos:**

1. Foi efetuada a revisão e análise dos diagramas feitos até à data presente, com objetivo de melhorar os mesmos, nos quais estão incluídos:
   1. Diagrama de Componentes;
   2. Diagrama de casos de uso;
   3. Diagrama de classes;
   4. Diagrama de Business Process Model and Notation (BPMN);
   5. Diagrama de Atividades;

Após esta ação, foi inicializada a discussão de ideias para os seguintes diagramas:

1. Diagrama de Estados;
2. Diagrama de Entidade-Relação.
3. Início de implementação do programa no qual foi feito apenas a página de Login, sem qualquer funcionalidade.
4. Foi feita também uma procura de tutoriais e documentação para servirem de apoio para o desenvolvimento da aplicação.

### Ata #4

**Local da Reunião:** Online (via Discord)

**Data:** 12 de novembro de 2022

**Hora de Início:** 23:30 **Hora de Fim:** 01:30

**Ordem de trabalhos:**

1. Foi efetuada a revisão e análise dos diagramas feitos até à data presente, com objetivo de melhorar os mesmo, nos quais estão incluídos:
   1. Diagrama de Componentes;
   2. Diagrama de casos de uso;
   3. Diagrama de classes;
   4. Diagrama de Business Process Model and Notation (BPMN);
   5. Diagrama de Atividades;
   6. Diagrama de Estados;
   7. Diagrama de Entidade-Relação.
2. Tendo em conta o adiantamento de tarefas por parte dos membros de grupo, não foi realizado mais nada nesta reunião.

## Entrega #3

### Ata #1

**Local da Reunião:** Online (via Discord)

**Data:** 19 de novembro de 2022

**Hora de Início:** 16:30 **Hora de Fim:** 17:00

**Ordem de trabalhos:**

1. Tendo em conta a semana ocupada com avaliações de outras cadeiras, não houve muita coisa a ser tratada nesta sprint;
2. Análise e possível melhoramento de diagramas previamente feitos;
3. Discussão de grupo em relação ainda às linguagens a serem utilizadas, tendo em conta o decorrer de outras cadeiras.

### Ata #2

**Local da Reunião:** Online (via Discord)

**Data:** 26 de novembro de 2022

**Hora de Início:** 15:30 **Hora de Fim:** 16:30

**Ordem de trabalhos:**

1. Análise critica e construção dos diagramas de atividades e estados;

2. Melhoria e ajustes nos Mockups de acordo com alterações sugeridas pelo grupo;

3. Discussão de grupo em relação a como fazer o diagrama de sequência por ecrã.

### Ata #3

**Local da Reunião:** IPCA (com docente)

**Data:** 3 de dezembro de 2022

**Hora de Início:** 21:00 **Hora de Fim:** 21:30

**Ordem de trabalhos:**

1. Conclusão dos diagramas de sequência por ecrã;

2. Melhoria e alterações do relatório face a observações feitas pelo docente (diagrama de Gantt, legendas e introduções a tópicos);

3. Realização do dossier de organização do grupo;

4. Discussão em grupo para planejamento da próxima entrega e aspetos a melhorar.

## Entrega #4

## Ata #1

**Local da Reunião:** Online (via Discord)

**Data:** 16 de Dezembro de 2022

**Hora de Início:** 15:30 **Hora de Fim:** 16:00

**Ordem de trabalhos:**

1. Planeamento da construção do projeto
2. Análise da entrega passada e melhorias do projeto
3. Discussão de grupo sobre tarefas individuais a realizar
4. Planeamento do Poster

## Ata #2

**Local da Reunião:** Online (via Discord)

**Data:** 23 de Dezembro de 2022

**Hora de Início:** 15:30 **Hora de Fim:** 16:30

**Ordem de trabalhos:**

1. Parte da aplicação Back-End\* concluída
2. Análise critica sobre aspetos a melhorar no programa para futuras sprints
3. Reorganização do calendário e ajustes de tarefas individuais

\*Back-End (em contexto de desenvolvimento de software) é a parte de um sistema que gere a lógica e o armazenamento de dados, é normalmente responsável por tarefas como armazenar e recuperar dados, processar pedidos e implementar a lógica de negócio. O Back-End é geralmente executado num servidor e pode ser construído com uma variedade de linguagens de programação e tecnologias

## Ata #3

**Local da Reunião:** Online (via Discord)

**Data:** 30 de Dezembro de 2022

**Hora de Início:** 15:30 **Hora de Fim:** 16:30

**Ordem de trabalhos:**

* 1. Revisão e análise critica sobre o que foi implementado no Back-End
  2. Começo da implementação do Front-End \*
  3. Construção do Poster publicitário
  4. Ajustes e correção de problemas

\*Front-End em contexto de desenvolvimento de software, o Front-End é a parte de um sistema que apresenta a interface de utilizador e gere a interação do utilizador. O Front-End é normalmente responsável por tarefas como renderizar a interface de utilizador, processar a entrada do utilizador e apresentar dados ao utilizador.

## Ata #4

**Local da Reunião:** Online (via Discord)

**Data:** 5 de Janeiro de 2022

**Hora de Início:** 21:00 **Hora de Fim:** 22:30

**Ordem de trabalhos:**

1. Ponto de situação do que foi implementado até agora (em função do tempo restante).
2. Implementação do Front-End quase concluído
3. Revisão e finalização do poster publicitário
4. Começo da ligação entre o Front-End e o Back-End
5. Atualização do relatório

## Entrega #5

## Ata #1

**Local da Reunião:** Discord

**Data:** 9 de Janeiro de 2022

**Hora de Início:** 10:00 **Hora de Fim:** 11:00

**Ordem de trabalhos:**

1. Revisão do ponto de situação atual do projeto num todo.
2. Estabelecimento de melhorias finais.
3. Estabelecimento de pontos extra a acrescentar ao projeto (se sobrar tempo).
4. Atualização do relatório

# Diagramas

## Diagrama Entidade-Relação

Segue-se abaixo o diagrama de entidade-relação da base de dados do IPCA GYM:

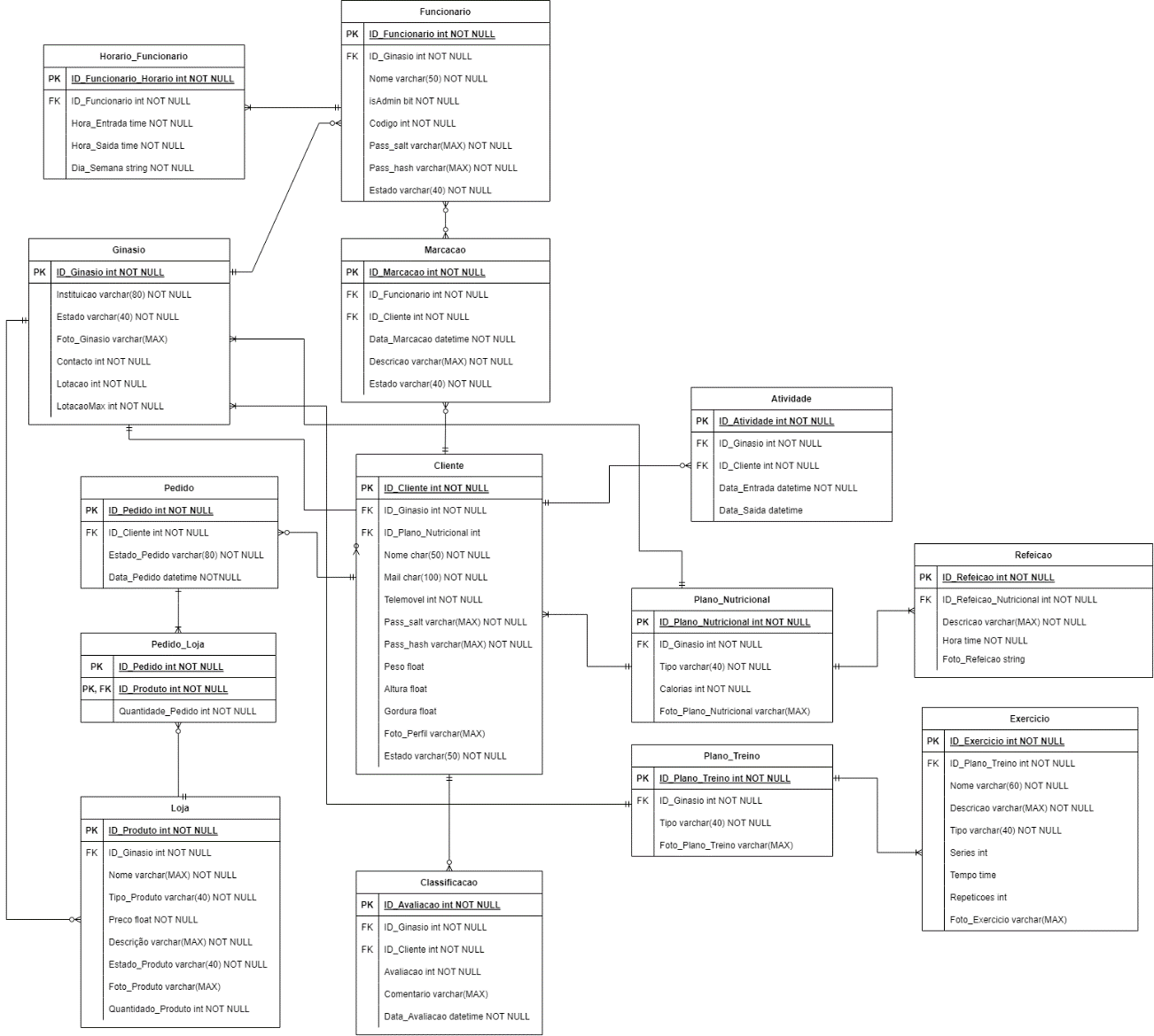


Figura 1 - Diagrama de Entidade-Relação

Como entidades principais este diagrama possuí:

* **­Cliente** – dados de um cliente que está a utilizar a aplicação;
* **Funcionário** – dados de um funcionário do ginásio em causa, possuí o atributo “isAdmin” para determinar se este tem como role Gerente ou não;
* **Ginásio** – dados do ginásio em causa, entidade criada de forma que o projeto, mais tarde, tenha suporte para várias instituições académicas
* **Loja** – possui dados de todos os produtos disponíveis e indisponíveis na loja de cada ginásio
* **Atividade** – entidade criada com o propósito de analisar as entradas e saídas de cada cliente no ginásio (recebe informação do Arduíno)

De forma que fosse possível suportar alguns dados sobre outras funcionalidades, foram adicionadas as seguintes entidades:

* **Plano\_Nutricional** e **Refeição** – entidades que possuem dados sobre diferentes refeições e seus horários, cada ginásio define o seu plano nutricional;
* **Plano\_Treino** e **Exercício** – entidades que possuem dados sobre diferentes exercícios e suas descrições, cada ginásio define o seu plano de treino;
* **Pedido** e **Pedido\_Loja** – entidade que possui dados de cada encomenda feita pelo utilizador na loja do ginásio no qual este está inscrito;
* **Horario\_Funcionario** – regista o horário de cada funcionário, de forma a verificar a sua disponibilidade para as diferentes marcações;
* **Marcação** – possui a informação de todas as marcações marcadas pelo cliente com o funcionário, associadas a cada ginásio;
* **Classificação** – contém todas as avaliações feitas pelos clientes a cada ginásio.

## Diagrama de Casos de Uso

Segue-se abaixo os diferentes casos de usos criados, os diagramas demonstram de uma forma generalizada o uso da aplicação.

### Caso de uso 1 – Gestão de Conta

Qualquer utilizador da plataforma poderá gerir a sua conta de uma forma geral, sendo que o funcionário e o gerente ainda poderão manipular dados do cliente, e ainda que o gerente poderá manipular as de qualquer funcionário.

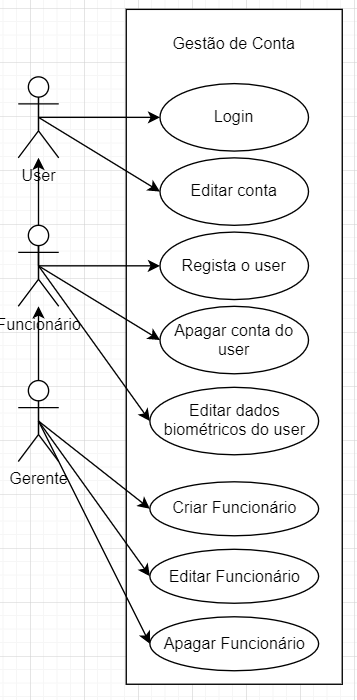


Figura 2 - Diagrama Casos de Uso 1 (Gestão de Conta)

* **CdU 1.1 Login** - O utilizador introduz os seus dados para entrar na nossa aplicação;
* **CdU 1.2 Editar Conta** - O utilizador é capaz de editar o seu perfil;
* **CdU 1.3 Regista o user** - O funcionário do ginásio é capaz de registar um utilizador na aplicação;
* **CdU 1.4 Apagar conta do user** - O funcionário é capaz de apagar a conta de um utilizador e respetivamente todos os dados associados a ele;
* **CdU 1.5 Editar dados biométricos do user** - O funcionário, durante uma consulta , é capaz de alterar os dados do utilizador (altura, peso, massa muscular gorda/magra);
* **CdU 1.6 Criar funcionário** - O gerente é capaz de criar um utilizador com a propriedade de ser funcionário também;
* **CdU 1.7 Editar Funcionário** - O gerente é capaz de editar todos os dados relativos a um funcionário;
* **CdU 1.8 Apagar Funcionário** - O gerente é capaz de apagar a conta de um funcionário, eliminando também todos os dados a ele.

### Caso de uso 2 – Loja online

A loja online irá oferecer ao utilizador a hipótese de visualizar a lista de artigos do ginásio ao qual pertencem, adicionar ao carrinho e finalizar a compra. Depois estas serão adicionadas a lista de pedidos na qual

poderá cancelar.

O funcionário gere esses pedidos, mas em relação a loja online ele apenas poderá gerir o stock.

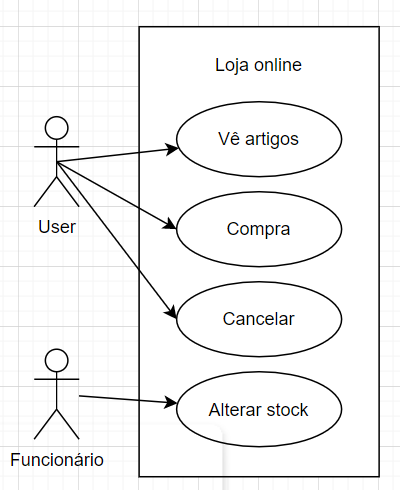


Figura 3 - Diagrama de Casos de Uso 2 (Loja Online)

* **CdU 2.1 Vê artigos** - O utilizador é capaz de ver os artigos na nossa loja online;
* **CdU 2.2 Compra** - O utilizador é capaz de comprar artigos da nossa loja online;
* **CdU 2.3 Cancelar** - O utilizador é capaz de cancelar uma encomenda (desde que essa já esteja registada na nossa loja online);
* **CdU 2.4 Alterar Stock** - O Funcionário é capaz de alterar o stock existente na nossa loja.

### Caso de uso 3 – Aplicação

Neste aqui mostra de forma generalizada o funcionamento da App com as suas multifuncionalidades.

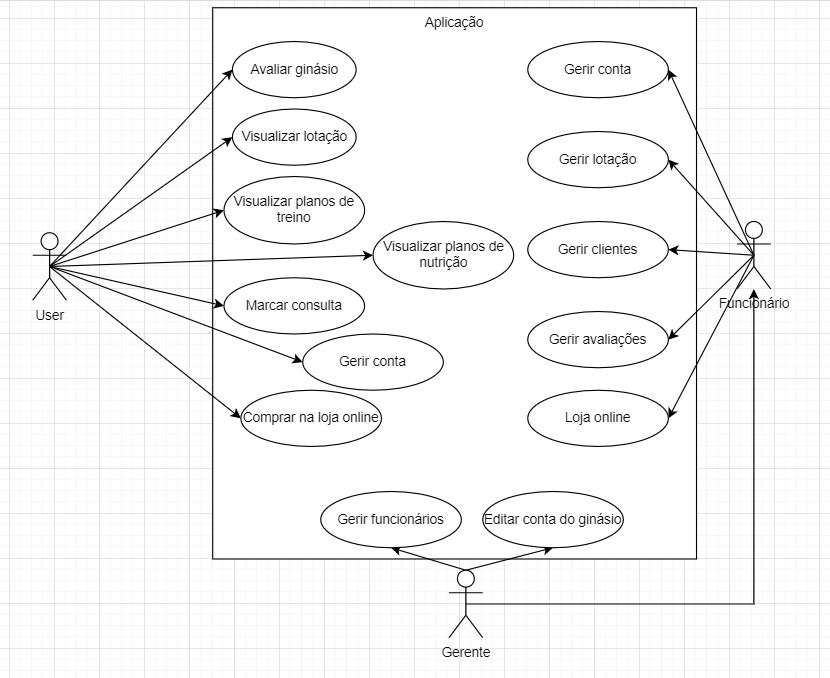


Figura 4 - Diagrama de Casos de Uso 3 (Aplicação)

* **CdU 3.1 Avaliar ginásio** - O utilizador da aplicação poderá usufruir de um sistema de avaliações relativamente ao ginásio/aplicação;
* **CdU 3.2 Visualização lotação** - O utilizador poderá através da página inicial visualizar a lotação atual do ginásio;
* **CdU 3.3 Visualizar planos de treino** - O utilizador poderá visualizar todos os planos de treino que o ginásio tem para oferecer;
* **CdU 3.4 Visualizar planos de nutrição** - O utilizador poderá visualizar todos os planos de nutrição que o ginásio tem para oferecer;
* **CdU 3.5 Marcar consulta** - O utilizador poderá marcar uma consulta nutricional ou física;
* **CdU 3.6 Gerir Conta** - O utilizador poderá alterar os dados da sua conta;
* **CdU 3.7 Comprar na loja online** - O utilizador poderá realizar compras na loja online;
* **CdU 3.8 Gerir funcionários** - O gerente poderá gerir os funcionários (remover, criar, editar);
* **CdU 3.9 Editar conta do ginásio** - O gerente poderá alterar os dados do ginásio;
* **CdU 3.10 Gerir conta** - O funcionário poderá editar os seus dados de conta;
* **CdU 3.11 Gerir lotação** - O funcionário poderá monitorizar a lotação de uma forma mais detalhada;
* **CdU 3.12 Gerir clientes** - O funcionário poderá gerir dados dos seus clientes;
* **CdU 3.13 Gerir avaliações** - O funcionário poderá monitorizar as avaliações feitas á sua ginásio/aplicação;
* **CdU 3.14 Loja online** - O funcionário poderá adicionar/remover/editar produtos á loja online.

### Caso de uso 4 – Gestão de lotação

O principal foco desta App era o facto do utilizador poder ver a capacidade atual do ginásio e assim decidir se lhe apetece ir ou não ao estabelecimento,

O Funcionário irá gerir essas estatísticas mais detalhadamente e ainda irá picar o cartão do user.

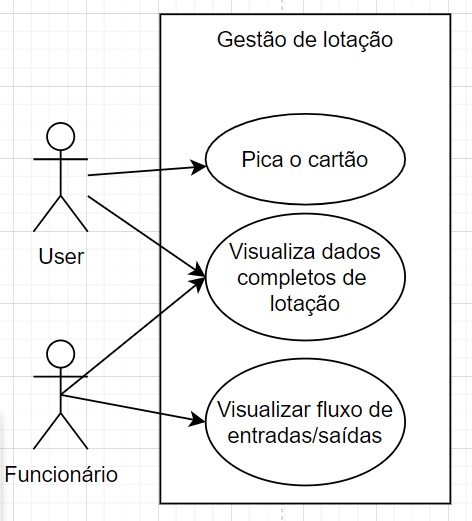


Figura 5 - Diagrama de Casos de Uso 4 (Gestão de lotação)

* **CdU 4.1 Picar o cartão** - O utilizador terá de picar o cartão no leitor do ginásio;
* **CdU 4.2 Visualiza dados completos de lotação** - O utilizador e o funcionário poderão visualizar a lotação detalhada do ginásio;
* **CdU 4.3 Visualizar fluxo de entradas/saídas** - O funcionário poderá monitorizar todas as entradas e saídas do ginásio desde o início da sua existência.

## Diagrama de atividades

Foram desenvolvidos 3 diagramas de atividades.

### Cancelamento de serviço

Para o cancelamento de um serviço por parte de um funcionário, é envolvida a participação do cliente, gerente e funcionário. O funcionário então cancela e notifica o gerente do incidente. O gerente recebe essa notificação e toma uma decisão. Rejeitando esse pedido, o funcionário é notificado e termina a atividade.

No caso de aceitar, o cliente é notificado e aí esse decide remarcar ou cancelar. Caso este não aceite termina a tarefa. Caso aceite, é remarcada e termina com o funcionário a ser notificado.

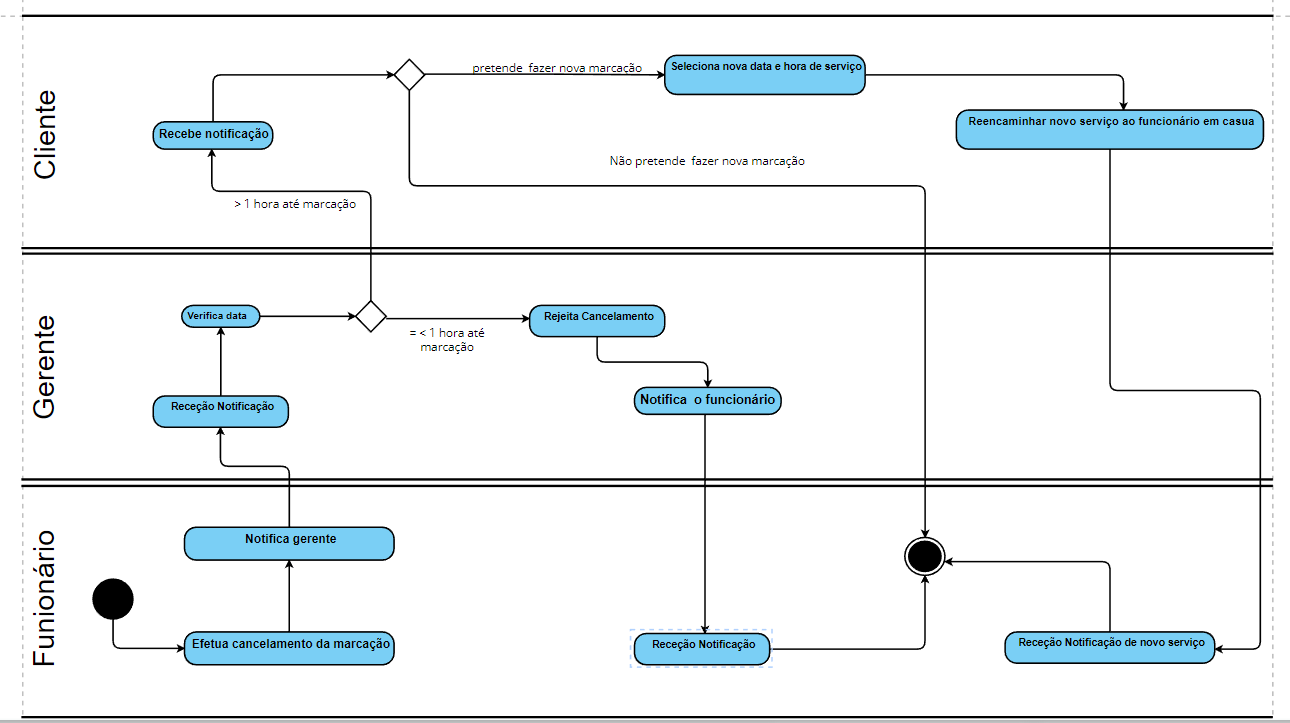


Figura 6 - Diagrama de Atividades - Cancelar Serviço

### Reagendamento/Cancelamento de serviço

Para o reagendamento/cancelamento de um serviço é envolvida a participação do cliente e do funcionário. O cliente altera os dados da sua marcação, onde este decide se pretende remarcar ou cancelar. No caso desta ser cancelada, o funcionário é notificado e termina a atividade.

Por outro lado, o reagendamento leva a definição de uma nova data e dependendo da disponibilidade esta pode ou não ser aceite e desta forma termina com o funcionário sendo notificado.

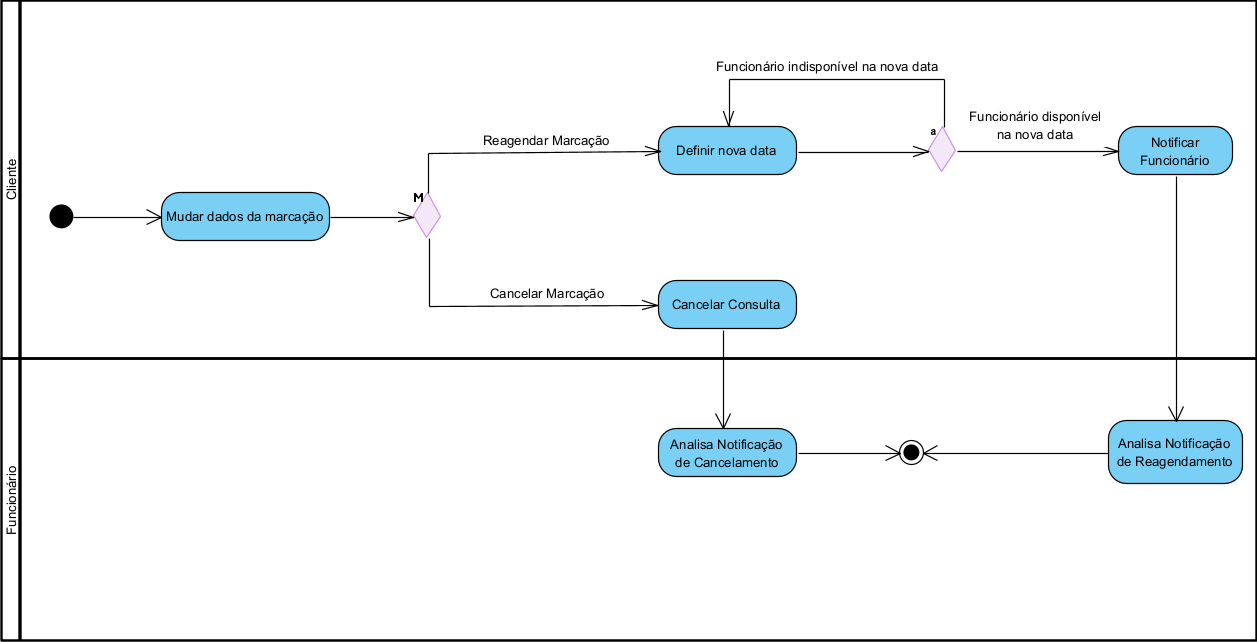


Figura 7 - Diagrama de Atividades - Reagendamento de Serviço

## Diagrama de estados

Foi desenvolvido apenas um diagrama de estados para analisar o estado da marcação de uma consulta. Dentre os estados possível este pode ser dado como:

* **Pendente:** quando este se encontra a espera de uma resposta;
* **Cancelada:** quando não é possível marcar para uma determinada data, ou cancelado pelo próprio cliente que a marcou;
* **Agendada:** quando esta é marcada definitivamente.

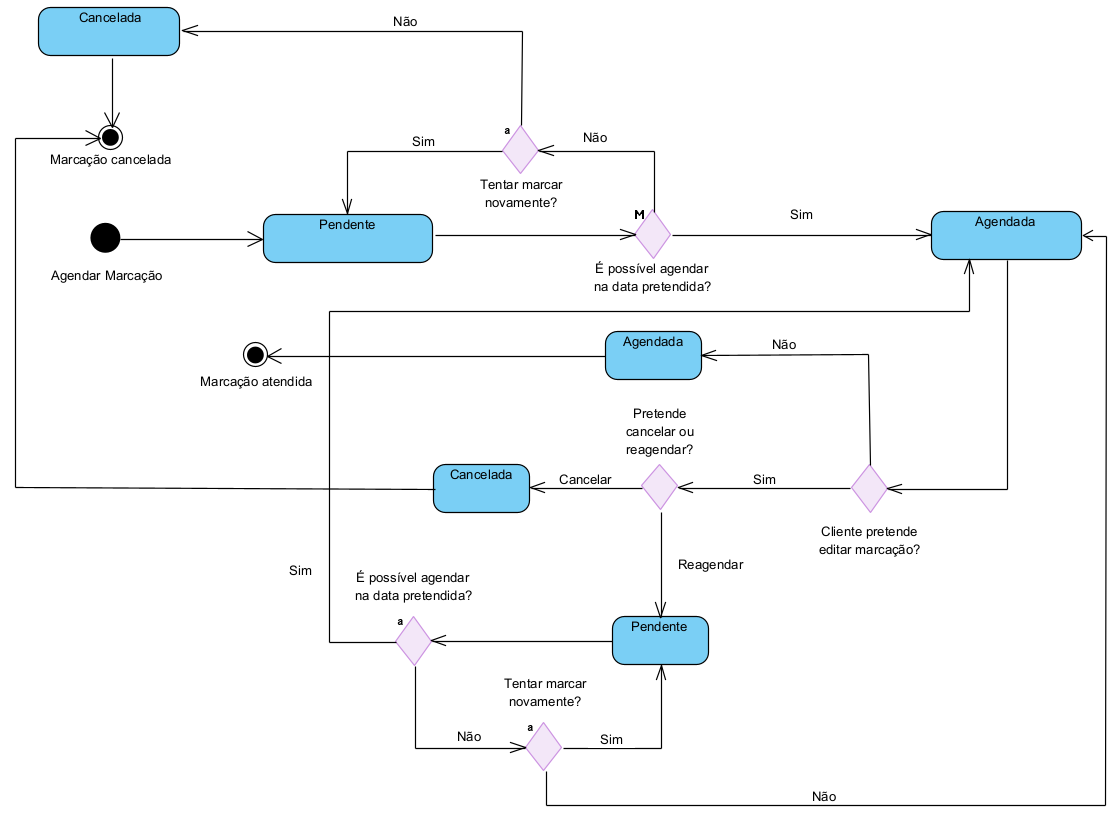


Figura 8 - Diagrama de Estados - Marcação de Consulta

## Diagrama de sequência por ecrã

O diagrama de sequência por ecrã referente aos Mockups realizados recorrendo ao Figma encontram-se no seguinte link:

<https://www.figma.com/file/Q4tM34gl91b9fhrGvUeXRs/MileriuPT's-team-library?node-id=0%3A1&t=a7h8lpVPLF9lSzXo-1>

# Mockups

## Cliente

### Login

Login do utilizador no qual é inserido o código de utilizador (inteiro) e password. Caso este não saiba a password terá de recorrer ao link para recuperar a conta.

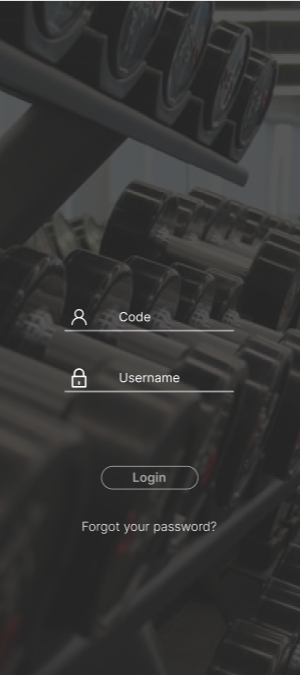


Figura 9 - Login do Utilizador

### Menu Principal

Menu de entrada do utilizador, onde este poderá saber mais informações sobre o ginásio, informações de nutrição, equipa pertencente ao ginásio além de dados sobre a capacidade do estabelecimento. Este ainda poderá aceder a um menu através do clique da foto e tem outras funcionalidades na barra inferior horizontal.

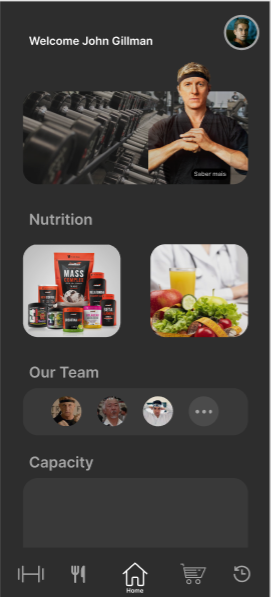


Figura 10 - Utilizador - Página inicial

### Clique na foto

Clicando na foto aparece um menu com algumas opções. Pode aceder a conta, definições da app e terminar a sessão.



Figura 11 - Utilizador - Menu

### Conta do utilizador

Acedendo a conta do utilizador, este pode editar a conta, marcar uma consulta, ver informações acerca das suas medidas, estatísticas, detalhes dos seus treinos e contrato.



Figura 12 - Utilizador - Conta

### Definições

Nas definições este poderá ver algumas das informações acerca da app.

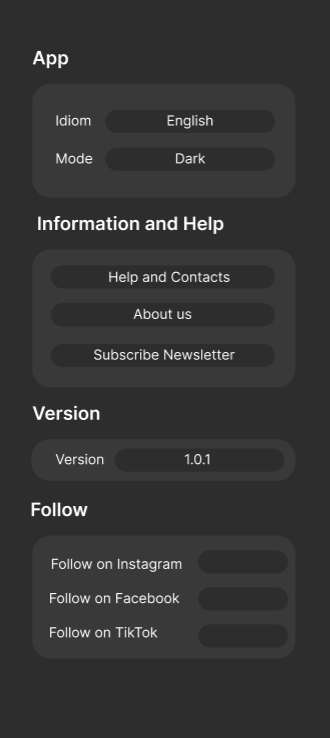


Figura 13 - Utilizador - Definições

### Editar perfil

O utilizador poderá alterar os seus dados principais da conta (nome, email, contacto e data de nascimento).

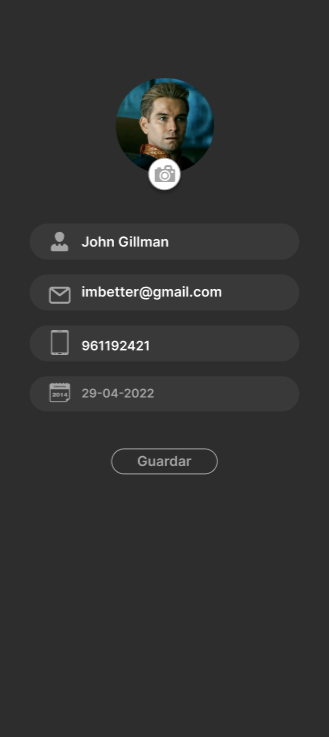


Figura 14 - Utilizador - Editar perfil

### Atividade

Aqui ele poderá ver toda a sua atividade ao nível de treinos executados, além de os poder ver detalhadamente.

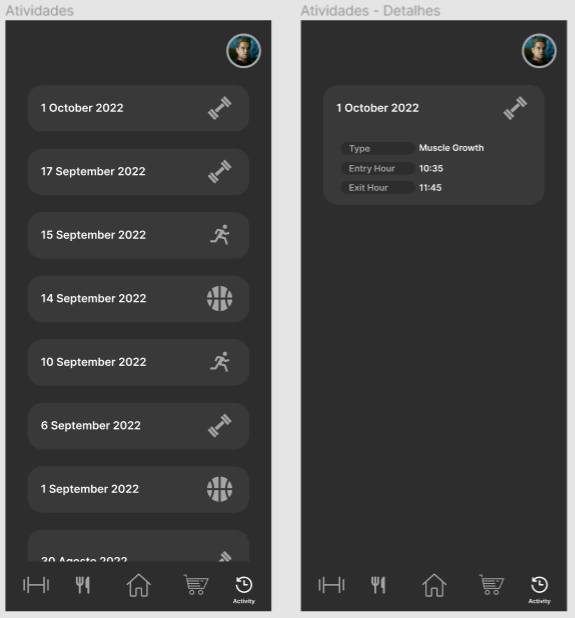


Figura 15 - Utilizador - Atividade

### Treinos

A app oferece ao utilizador os planos de treino que se seguem. Um desses tipos de plano é treinar separadamente cada membro que nos leva ao 2º Mockups. Para o caso de um desporto segue para o 3º Mockups.

O 4º Mockups representa um treino selecionado com os seus exercícios.

Uma imagem com texto, diferente, vários

Descrição gerada automaticamente

Figura 16 - Utilizador - Treino

### Nutrição

O utilizador poderá também aceder aos seus planos de nutrição onde poderá consultar o que deve comer a uma determinada hora e alterar os seus objetivos de nutrição.

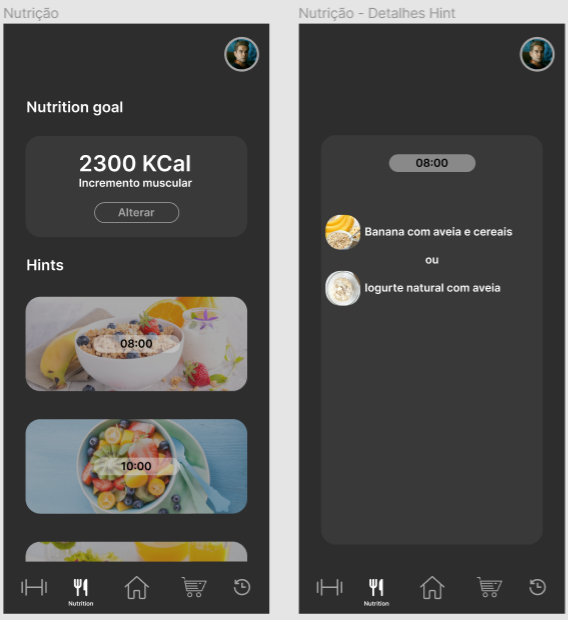


Figura 17 - Utilizador - Nutrição

### Loja online

Loja online para compra de equipamento de treino, alimentos e outros recursos.

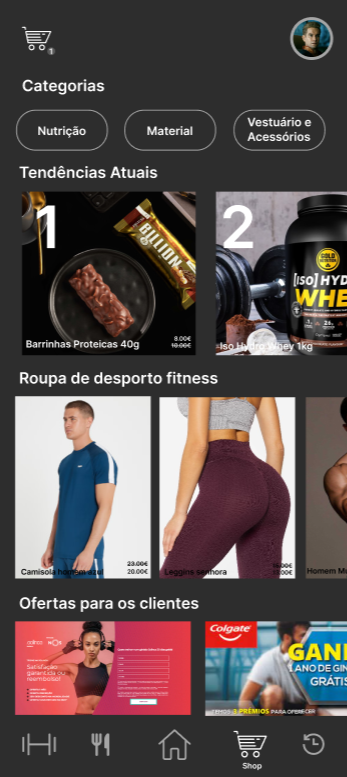


Figura 18 - Utilizador - Loja Online

## Sistema do Ginásio

### Capacidade

O funcionário do ginásio poderá gerir todos os dados de capacidade, estatísticas e outros recursos de monitorização através da secção de capacidade da sua app.

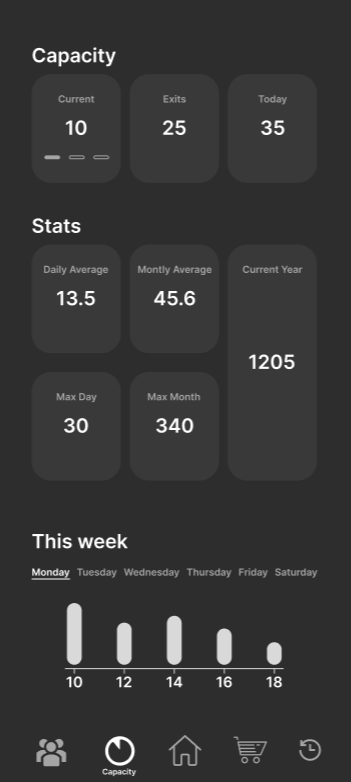


Figura 19 - Sistema - Capacidade

### Gestão dos clientes

Este poderá consultar os clientes, além de visualizar e editar os seus dados métricos. Ainda poderá inserir novos clientes ao sistema.

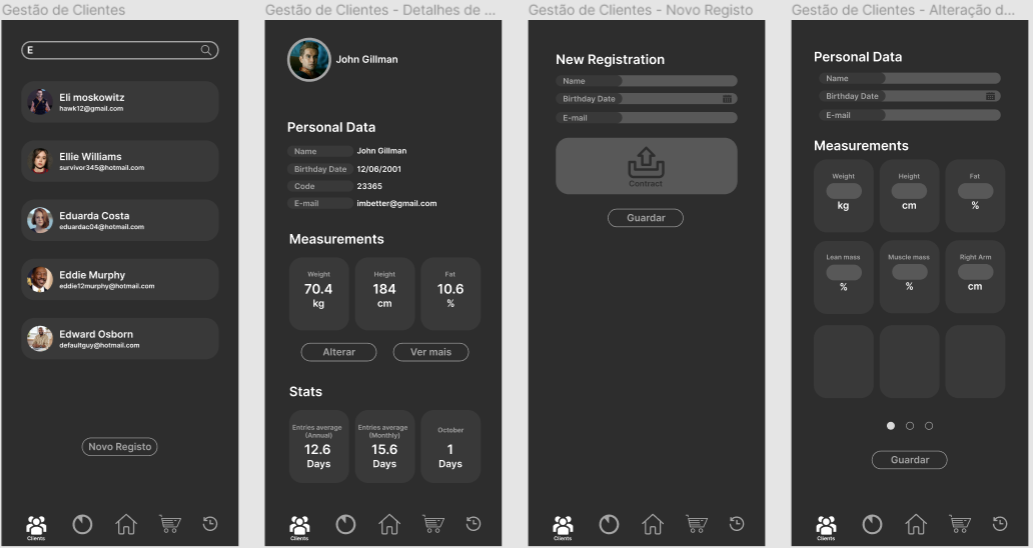


Figura 20 - Sistema - Gestão de clientes

### Página inicial

Através da página inicial este poderá ver a avaliação atual do seu ginásio, além de comentários.

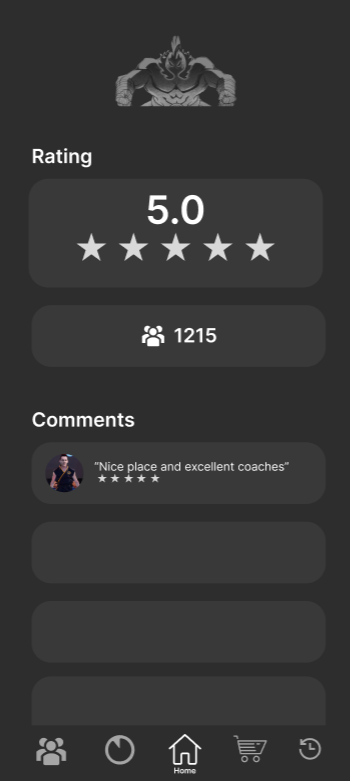


Figura 21 - Sistema - Página inicial

### Loja online

Página para gerir os pedidos de material da loja online por parte dos utilizadores.

Uma imagem com texto

Descrição gerada automaticamente

Figura 22 - Sistema - Loja online

### Atividade

Aqui poderá gerir os fluxos de entrada e saída do ginásio ao longo de todo o tempo.

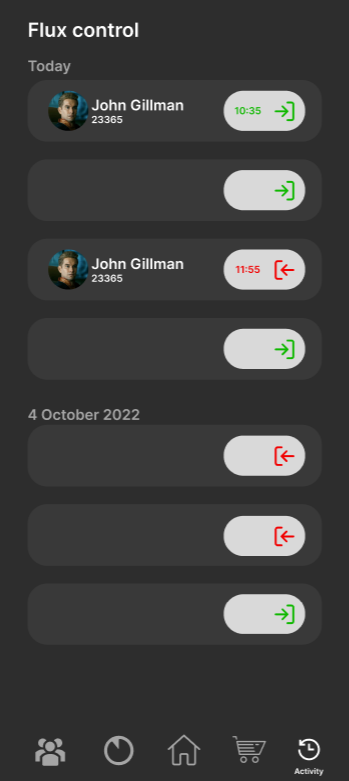


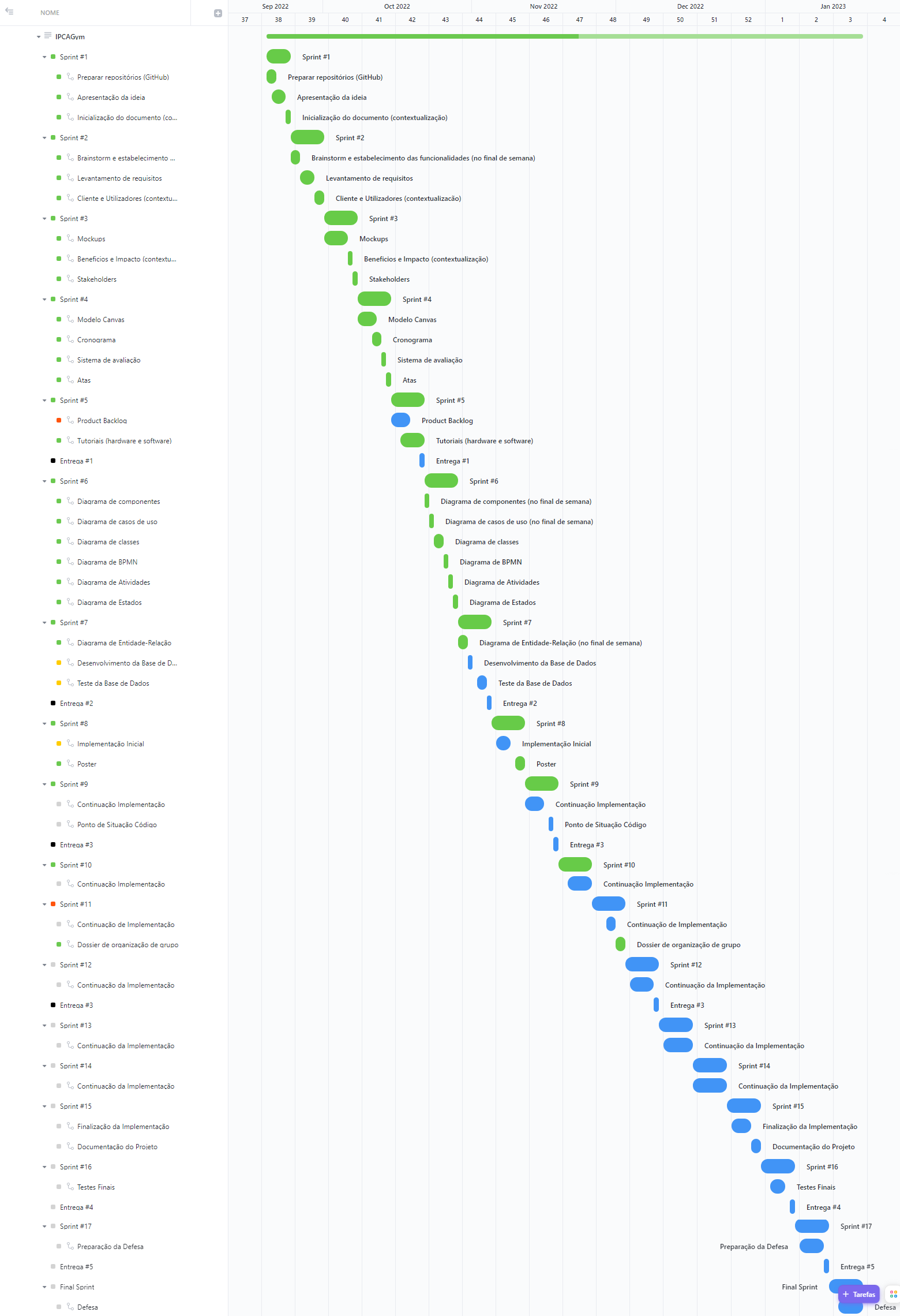
Figura 23 - Sistema - Atividade

# Cronograma

O cronograma do projeto encontra-se num documento externo chamado “Cronograma.xlsx” juntamente com o planeamento das sprints até ao final do projeto. (estas podem sofrem alterações ao longo do desenvolvimento do mesmo, sendo essas alterações devidamente documentadas).

Este mesmo cronograma também foi apresentado em um diagrama de Gantt, identificando assim as tarefas e subtarefas que serão feitas ao longo do projeto, identificando também assim a data e duração das mesmas. Este diagrama será atualizado a cada entrega.

Segue-se o Diagrama de Gantt:



`Figura 24 - Diagrama de Gantt

# Código implementado para o projeto

Este projeto foi implementado com o suporte de um Front-End, Back-End e uma base de dados para guardar a informação.

De forma, a haver uma interação fora da aplicação com o cliente, será implementado também um Arduino com um sensor de cartões elétricos.

## Base de Dados

Como base de dados foi usado o SQL Server, linguagem esta da qual o grupo já tem experiência tanto a nível de comandos (queries) como a nível de IDE, neste caso foi utilizado o SSMS 18 (Microsoft SQL Server Management Studio).

O projeto foi iniciado com uma base de dados local na máquina de cada membro de grupo, com a previsão de no final ser escolhida a base de dados mais povoada e com dados mais coerentes para fazer deployment.

Uma imagem com texto, mesa

Descrição gerada automaticamente

Figura - Microsoft SQL Server Management Studio (SSMS)

## Back-End

A nível de Back-End, de forma ao grupo seguir a metodologia da cadeira de ISI e também de forma a trabalhar com algo que tivesse mais experiência, foi escolhido trabalhar com .NET Framework (C#), utilizado o IDE Visual Studio 2022.

Segue-se a abordagem dos pontos principais do Back-End, onde será especificado o que feito em cada um:

* Programação por camadas
* JWT Tokens e Authentication
* Funcionalidade de Login (Encriptação de passwords)
* Documentação de código com OpenAPI

### Programação por camadas

De forma a organizar melhor a solução do Back-End e também contemplar os objetivos propostos para a entrega final de ISI, esta ficou dividida por diferentes camadas.

A atribuição de dependências foi a seguinte:

* Camada do Backend\_IPCA\_Gym (camada da API) depende da camada de Business Logic (BLL)
* Camada do Business Logic depende da camada de Data Access (DAL)
* Camada de Data Access depende da camada de Business Object (BOL)

Uma imagem com texto

Descrição gerada automaticamente

Figura - Dependências API Layer

Uma imagem com texto

Descrição gerada automaticamente

Figura - Dependências BLL Layer

Uma imagem com texto

Descrição gerada automaticamente

Figura - Dependências DAL Layer

### JWT Tokens e Authentication

De forma a manter o programa mais seguro e fazer a verificação de quem tem acesso a cada request, foram implementadas tokens de sessão com JWT Tokens, fazendo assim a autenticação do utilizador.

Com isto é possível obter informações sobre o utilizador que está a utilizar a aplicação e se este tem as permissões necessárias para aceder a dados específicos ou não.

A token é atribuída ao utilizador dependendo dos dados de Login, segue-se um exemplo para um login de um funcionário:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Login de um funcionário (Back-End)

Como é possível visualizar, o funcionário insere os seus dados e estes são mandados por parâmetro, é obtida a informação da query de busca do funcionário e é verificada a palavra-passe, no caso de sucesso é atribuída a token de sessão a partir do seguinte método auxiliar:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Método auxiliar CreateTokenFuncionário()

Neste método o que irá diferenciar entre os utilizador a sua token é a Role atribuída, há a possibilidade do funcionário ser um “Funcionário” ou um “Gerente”, foi criada uma Role à parte para testes e para efetuar qualquer operação sem problema chamada de “Admin”, que é atribuída a funcionários com o nome “adminaccount”

Para fazer a Autenticação dos requests, o utilizador que solicita um serviço precisa de ter uma token de sessão em que a Role corresponda com as Roles permitidas no serviço, segue-se a seguir um exemplo:



Figura - Headers de Authorize num Request HttpPatch

### Funcionalidade de Login (Encriptação de password)

Como já foi mencionado previamente, o programa apresenta uma funcionalidade de login e registo, da qual a palavra passe do utilizador será encriptada a partir do namespace fornecido pelo System, System.Security.Criptography.

Para fazer a encriptação foram criados dois métodos auxiliares, utilizados no login e registo do utilizador:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Métodos auxiliares CreatePasswordHash() e VerifyPasswordHash()

O CreatePasswordHash() é utilizado no registo de um cliente ou funcionário para encriptar a palavra-passe inserida pelo utilizador:

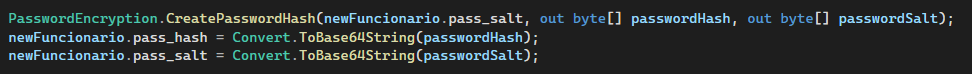


Figura - Tratamento de password no Request HttpPost de um Funcionário

Já o VerifyPasswordHash, como já foi possível ver no Login do ponto 17.2.2. deste projeto, é utilizado para verificar se uma palavra-passe inserida corresponde à palavra-passe registada na base de dados desencriptada:

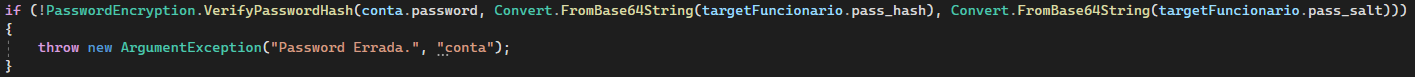


Figura - Verificação de palavra-passe no Login de um Funcionário

### Documentação de código com OpenAPI

De forma a fazer a documentação correta da API, foi utilizado o SwaggerUI, ferramenta que foi sugerida pelo professor de ISI (Óscar Ribeiro).

Começou-se por fazer a implementação do gerador do Swagger no Program.cs (que funciona como Startup da API):

Uma imagem com texto

Descrição gerada automaticamente

Figura - Geração documentação no Swagger

Foi também exportada toda a documentação para um ficheiro .xml externo que se encontra no seguinte diretório:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Propriedades da Solução do projeto

A forma que foi usada para a documentação de métodos e classes foi a seguinte:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Documentação de métodos

Uma imagem com texto

Descrição gerada automaticamente

Figura - Documentação de modelos de dados

Sendo este o resultado no Swagger:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Resultado documentação no Swagger

## Front-end

O Front-End do projeto foi implementado na linguagem Kotlin, utilizando o IDE Android Studio, seguindo assim a metodologia da unidade curricular de Programação de Dispositivos Móveis (PDM).

O projeto ficou dividido em duas partes:

* Activities – tratam-se das classes que fazem o funcionamento do Front-End de forma interativa e também o tratamento de dados recebidos e/ou enviados para a API externa (Back-End falado no ponto 17.2)
* Resources – tratam-se de ficheiros XML compostos pelo código necessário para a construção do visual do software a nível físico, estes apresentam Layouts, Menus e Drawables (package com todas as imagens e ícones utilizados na aplicação)

### Activities

As activities estão divididas por 3 partes distintas:

* Classes de tratamento de dados e funcionamento do Front-End (cliente\_classes e funcionários\_classes): cada um desses packages possui as activities necessárias para cada funcionalidade da aplicação, sendo que cada uma está associada a um layout
* Modelos de dados (data\_classes): possuí os formatos de cada objeto a ser enviado e/ou recebido da API
* Requests (requests): possuí objetos que tratam de fazer os requests todos do Front-End para a API e espera uma resposta
* Views: possuí o código correspondente a uma custom view criada por nós

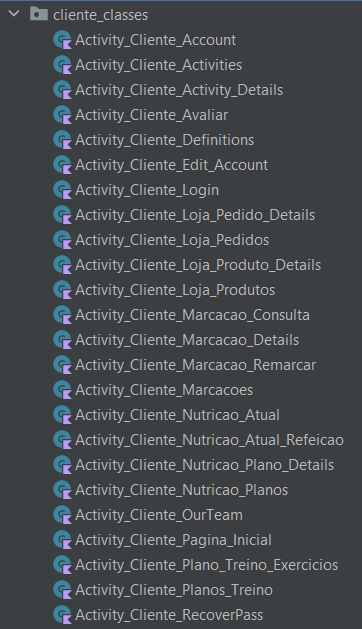


Figura – Atividade Cliente Classes

Uma imagem com texto

Descrição gerada automaticamente

Figura - Atividade Funcionaria classes

Uma imagem com texto

Descrição gerada automaticamente

Figura - Data classes

### Objeto request

Foram montados Objects que possuem métodos que chamam requests desenvolvidos em Back-End. Estes serão chamados pelas activities principais dependo das ações a serem desencadeadas.

Uma imagem com texto

Descrição gerada automaticamente

Figura – Requests

Os pedidos (ou requests) são utilizados para aceder e manipular bases de dados através de uma aplicação. Eles permitem a nós, como desenvolvedores, enviar comandos para a base de dados, tais como realizar consultas, inserir, atualizar e apagar dados (como iremos apresentar nas imagens abaixo). Os pedidos da base de dados são feitos através de uma linguagem de consulta estruturada (neste caso utilizamos o SQL, como já foi referido).

Utilizando estes requests foi então possível manipular dados provenientes da nossa base de dados, de forma precisa e eficiente.

Importante mencionar também que a estrutura dos requests é, na maior parte dos casos, semelhante aos demais.

### Custom View

Foi desenvolvida uma Custom view para numa determinada página da aplicação ser geradas barras de percentagem de estatísticas de capacidade do ginásio.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Custom View

Relativamente ao código desenvolvido é possível visualizar abaixo a sua montagem e atribuição de valores que serão reproduzidos na página:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Custom View - Valores reproduzidos

Uma imagem com texto

Descrição gerada automaticamente

Figura - Custom View - Código desenvolvido

Este em uma situação real do uso da aplicação demonstra-se da seguinte forma:

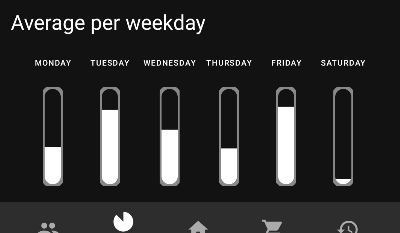


Figura - Custom View – Demonstração

### Requests

#### Get

Os pedidos GET são um dos métodos de requisição HTTP mais comuns e são utilizados para retornar informações, iremos agora apresentar exemplos de diversos “gets” utilizados no programa.

* GetAll

Neste request estamos a apresentar todos os registos cliente inseridos na base de dados. É passado por parâmetro a token de sessão do utilizador atual para verificar se este tem autorização e é retornada a lista pedida.

Uma imagem com texto

Descrição gerada automaticamente

Figura - get all

* GetById

Neste request estamos a apresentar um registo de cliente através do seu id, a diferença entre o anterior é que precisamos de enviar um targetID no request para ele saber a qual ID pertence.

Uma imagem com texto

Descrição gerada automaticamente

Figura – GetByID

* GetByToken

Neste request estamos a apresentar o registo de um cliente através do seu token de sessão, este request é importante para que seja possível listar a sua informação durante a execução da aplicação. Uma imagem com texto

Descrição gerada automaticamente

Figura 50 - Get By Token

### Post

A requisição HTTP POST é usada para enviar dados para serem processados pelo recurso identificado pela URL.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Post Funcionario Request

Uma imagem com texto

Descrição gerada automaticamente

Figura - Post Horario Funcionario request

Como se pode observar nas imagens acima, os requests HTTP POST são idênticos para todas as entidades da nossa aplicação.

O método utilizado envia os dados a serem processados pela identificação da URL. Podemos ver também que os dados são incluídos no corpo da mensagem, utiliza o launch do coroutines para fazer a requisição em segundo plano e usa o Dispatchers.IO para processar o request. Por fim faz uma validação se a reposta é bem-sucedida, caso contrário, ele envia uma mensagem de erro para a função de callback.

### 17.3.4.3.Patchs

O request HTTP PATCH é usado para fazer atualizações parciais em um recurso existente identificado pela URI. A diferença entre o PATCH e o POST é que o POST espera que o cliente envie todos os dados do objeto, enquanto o PATCH espera que o cliente envie apenas as informações que deseja atualizar.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Patch Funcionario Request

Uma imagem com texto

Descrição gerada automaticamente

Figura - Patch Loja Request

Tal como o método POST, os requests HTTP PATCH são idênticos para todas as entidades, este atualiza parcialmente um objeto existente identificado pela URL. Ele envia os dados para serem atualizados em formato json e especifica o ID do objeto que deseja atualizar. Ele utiliza a função launch do coroutines para fazer o request em segundo plano e usa o Dispatchers.IO para processar o mesmo.

### 17.3.4.4.Delete

O request, semelhante aos anterior, também é idêntico para todas as entidades da nossa aplicação, o HTTP DELETE é usado para excluir um objeto específico identificado pela URL. Ele não requer um body na mensagem e o objeto é excluído do servidor.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Delete Produto

Este código faz um request HTTP DELETE usando a biblioteca requests em Kotlin. Ele exclui um objeto específico identificado pela URL. Ele especifica o ID do objeto que deseja excluir.Utiliza também a função launch do coroutines para fazer a requisição em segundo plano e usa o Dispatchers.IO para processar a requisição.

### 17.3.4.5.Login

Uma imagem com texto

Descrição gerada automaticamente

Figura - Login Request

Este código faz um request HTTP POST para o endpoint de login de funcionário. Ele envia o código e a senha do funcionário no body do request em formato json. Ele utiliza a função launch do coroutines para fazer o request em segundo plano e usa o Dispatchers.IO para processar a requisição. Ele verifica se a resposta é bem-sucedida e se o statusCode é 200, em seguida extrai a resposta e transforma-a num objeto LoginReceiverModel usando a função fromJson(). Ele envia então o objeto para a função de callback. Caso contrário, ele envia um objeto nulo para a função de callback.

### Data Classes

Data classes são classes específicas do Kotlin que possuem apenas dados e nenhuma lógica adicional de negócios. Eles são usados ​​para representar dados simples e possuem algumas características úteis, como a capacidade de gerar automaticamente métodos de comparação, cópia e representação de string.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Data Classe - Cliente

Uma imagem com texto

Descrição gerada automaticamente

Figura 58 - Data Classe - Cliente (construtor)

Uma imagem com texto

Descrição gerada automaticamente

Figura - Propriedade da classe Cliente (enviar dados)

Uma imagem com texto

Descrição gerada automaticamente

Figura - Propriedade da classe do Cliente (receber dados)

Como exemplo estamos a apresentar a classe Cliente, é uma data\_classe que representa os dados de um cliente, como id, nome, email, peso, altura, etc. Ela tem dois métodos importantes:

* ToJson(): este método converte os dados da classe para um objeto JSON, que pode ser usado para enviar esses dados para a API.
* FromJson(jsonObject: JSONObject) : este método é um método auxiliar que converte um objeto JSON para uma instância da classe Cliente. A classe Cliente também tem um construtor que inicia todos os dados com os valores passados.

### Activity Classes

### Cliente

Como foi mencionado anteriormente neste documento, a aplicação encontra-se dividida em três roles principais, o Cliente, Funcionário e Gerente. Abrindo o explorador de ficheiros deste projeto podemos observar todas as activities do cliente e do funcionário.

Como o código é muito extenso devido à quantidade de coisas implementadas, iremos apenas apresentar excertos de código implementado que achamos mais importante, tanto para cliente como para funcionário.

* Login Cliente

Uma imagem com texto

Descrição gerada automaticamente

Figura - Login Cliente

Para a autentificação deste projeto estamos a usar tokens de sessão, os tokens de sessão são utilizados para identificar de forma única um utilizador na aplicação web. Eles são armazenados num cookie no navegador do utilizador e são enviados para o servidor com cada solicitação subsequente.

Ele verifica se o utilizador já está autenticado e, se sim, inicia uma nova atividade. Se não, ele configura um listener clicável para o botão de login. Quando o botão de login é selecionado, ele verifica se os campos de email e senha estão preenchidos. Se sim, ele chama uma função de login numa classe de solicitações de cliente, passando o email e a senha como parâmetros. Esta função de login chama uma API para autenticar o utilizador.

Se a autenticação for bem-sucedida, a função de retorno é chamada com um objeto do resultado que contem o token de sessão e o papel utilizador.

O token de sessão e o papel do utilizador são armazenados em preferências compartilhadas e a nova atividade é iniciada. Se a autenticação falhar, uma mensagem é exibida informando o utilizador de que as credenciais estão incorretas.

* Recover Password

Uma imagem com texto

Descrição gerada automaticamente

Figura - Recover pass (Cliente)

Este está a verificar se as senhas escritas são iguais. Ele está a comparar os valores dos campos de senha nova e repetir senha, se não forem iguais exibe uma mensagem "Passwords don't match!".

Se as senhas escritas forem iguais, verifica se os campos de senha nova, repetir senha e email estão preenchidos. Se algum deles estiver vazio, exibe uma mensagem "Insert all fields!". Se os campos estiverem preenchidos, ele faz uma chamada para a função ClienteRequests.recoverPasswordCliente() passando o scope de email e a nova senha, se a recuperação da senha for bem-sucedida, exibe uma mensagem "Password changed!" e vai para o ecrã de login, caso contrário, exibe uma mensagem de erro "Error: Recover Client password fails".

* Get token e os dados do perfil

Uma imagem com texto

Descrição gerada automaticamente

Figura - Página Inicial Cliente Exemplo Gets

Este código parece esta a fazer uma chamada para obter informações de um cliente utilizando o ClienteRequests.GetByToken() com um token de sessão. Se as informações do cliente são encontradas e possuem uma foto de perfil, ela é decodificada de Base64 para um array de bytes, convertida em um Bitmap e exibida numa ImageView. Em seguida, efetua uma chamada para obter informações sobre um ginásio utilizando GinasioRequests.GetByID() com o id do ginásio do cliente. Caso o ginásio estiver com a capacidade máxima, cor do texto que indica a lotação é mudada para vermelho. No final é atribuído a capacidade atual e máxima do ginásio numa TextView.

* Post criar uma avaliação para o ginásio

Uma imagem com texto

Descrição gerada automaticamente

Figura - Post Classificação

Tal como nos restantes requests, começamos por fazer uma chamada para obter informações de um cliente utilizando o ClienteRequests.GetByToken(). Se as informações do cliente são encontradas, é criado um objeto de classificação com os dados do cliente, id do ginásio, rating, descrição e data atual. Em seguida, é feita uma chamada para inserir a classificação utilizando o ClassificacaoRequests.Post() e passando o objeto novo “classificação” e o token de sessão. Se a inserção da classificação for bem-sucedida, a activity é finalizada e iniciada novamente, caso contrário, é exibida uma mensagem de erro "Error on create an rating".

* Patch editar conta cliente

Uma imagem com texto

Descrição gerada automaticamente

Figura - Patch cliente

Este código está a criar um objeto de cliente chamado "editCliente" com vários campos, incluindo o id do cliente, id do ginásio, id do plano nutricional, nome, email, telefone, senha, peso, altura, percentual de gordura, foto e estado.

Depois, efetua uma chamada para atualizar essas informações do cliente no servidor usando ClienteRequests.Patch(). Se a atualização for bem-sucedida, a activity é finalizada e iniciada novamente, caso contrário, é exibida uma mensagem de erro "Error on editting client account".

* Delete pedido loja

Uma imagem com texto

Descrição gerada automaticamente

Figura - Delete pedido loja

O está a impelementar um listener para um botão chamado "botao\_cancelar". Quando o botão é clicado, é criado um Intent para uma activity chamada "Activity\_Cliente\_Loja\_Pedidos", em seguida é feita uma chamada para a função PedidoLojaRequests.DeletePedidoLoja() passando o scope, o token de sessão e o id do pedido a ser cancelado. Se a exclusão do pedido for bem-sucedida, a activity atual é finalizada e a próxima activity é iniciada, caso contrário, exibe uma mensagem de erro "Error: Delete PedidoLoja fails".

### Funcionário

Para todas as activities do funcionário é importante restringir que o utilizador-funcionário não está a entrar com uma conta que tenha uma token de cliente.

Uma imagem com texto

Descrição gerada automaticamente

Figura – Session Token

* Login Funcionário

Uma imagem com texto

Descrição gerada automaticamente

Figura - Login funcionário

Este código está configurado para um listener de um botão chamado "button". Quando o botão é clicado, ele verifica se os campos de código e senha foram preenchidos. Se algum dos campos estiver vazio, exibe uma mensagem "Insert all fields!". Se os campos estiverem preenchidos, verifica se o código é menor que 10000000, se for menor ele faz a chamada para a função FuncionarioRequests.loginFuncionario() passando o scope, o código e a senha. Se a autenticação for bem-sucedida, é criado um SharedPreference chamado "my\_preferences" para armazenar o token e o “papel” do funcionário, em seguida, é criado um Intent para a Activity "Activity\_Funcionario\_Pagina\_Inicial" e finaliza a atual activity. Se o resultado for igual a "Wrong" é exibida uma mensagem "Wrong credentials", caso contrário, exibe uma mensagem "Wrong credentials text"

* Change password funcionário

Uma imagem com texto

Descrição gerada automaticamente

- Change Password (Funcionário)

Este código está a realizar uma chamada para obter informações de um funcionário usando a função FuncionarioRequests.GetByToken() com um token de sessão e um scope. Em seguida, ele faz outra chamada para verificar a senha atual usando FuncionarioRequests.loginFuncionario(), passando o escopo de vida, o código do funcionário e a senha atual. Se a autenticação for bem-sucedida, ele verifica se a nova senha e a confirmação da nova senha são iguais, se não forem iguais, exibe uma mensagem "Passwords don't match".

Caso as senhas forem iguais, verifica se os campos de senha nova e repetir senha estão preenchidos, se algum estiver vazio, exibe uma mensagem "Fill in all the fields". Se os campos estiverem preenchidos, faz uma chamada para alterar a senha usando a função FuncionarioRequests.recoverPasswordFuncionario(), passando o scope de código do funcionário e a nova senha. Se a mudança de senha for bem-sucedida, exibe uma mensagem "Password changed successfully", em seguida finaliza a atual activity e inicia a "Activity\_Funcionario\_Perfil\_Edit". Caso contrário, exibe uma mensagem "Error on changing password". Se a autenticação não for bem-sucedida, exibe uma mensagem "Incorrect password in Current Password".

* Post adicionar cliente por funcionário

Uma imagem com texto

Descrição gerada automaticamente

Figura - Post cliente pelo funcionário

Este código faz uma chamada para obter informações de um funcionário usando a função FuncionarioRequests.GetByToken() com um token de sessão e um scope. Então cria então um objeto do tipo Cliente com alguns valores específicos, como o nome, email, contacto, e o ID do ginásio que é retornado da chamada anterior. Em seguida, ele faz uma chamada para adicionar o novo cliente utilizando a função ClienteRequests.Post(), passando o scope, o token de sessão e o objeto cliente criado anteriormente. Se a adição for bem-sucedida, a atividade atual é finalizada e uma nova atividade é iniciada. Caso contrário, exibe uma mensagem "Error on create an client".

* Get produtos de uma determinada loja

Uma imagem com texto

Descrição gerada automaticamente

Figura - Get produtos

Este código faz uma chamada para obter todos os produtos de uma determinada loja de acordo com o ID do ginásio utilizando a função LojaRequests.GetAllByGinasioID() passando o scope, o token de sessão e o ID do ginásio. Em seguida, ele verifica se a lista de resultados não está vazia e, se não estiver, ele percorre a lista de produtos e adiciona cada produto que estiver com o estado "Ativo" para uma lista "produtos\_list" e Notifica o adaptador da lista para atualizar a lista de exibição de produtos.

* Patch editar exercício do plano de treino



Figura - Patch exercicio (1)

Uma imagem com texto

Descrição gerada automaticamente

Figura - Patch exercicio (2)

Uma imagem com texto

Descrição gerada automaticamente

Figura - Patch exercicio (3)

Uma imagem com texto

Descrição gerada automaticamente

Figura - Patch exercicio (4)

Este código tem como objetivo editar um exercício existente. Ele primeiro usa a função "ExercicioRequests.GetByID" para obter o exercício específico que será editado.

Em seguida, verifica se a opção de "isSet" está marcada. Se sim, ele armazena as informações de séries e repetições na intenção "intentEditIn" e usa a função "ExercicioRequests.Patch" para enviar as informações atualizadas para o servidor e editar o exercício.

Se a opção "isTime" estiver marcada, ele armazena as informações de tempo em minutos e segundos na intenção "intentEditIn" e usa a função "ExercicioRequests.Patch" para enviar as informações atualizadas para o servidor e editar o exercício.

Se a operação de edição for bem-sucedida, a atividade é finalizada e o resultado é retornado como RESULT\_OK. Se houver algum erro, uma mensagem de erro é exibida.

* Delete exercício

Uma imagem com texto

Descrição gerada automaticamente

Figura - Delete de um exercício

Este código está a declarar um evento de clique para um botão com o id "apagarExercicioButton".

Quando o botão é clicado, uma função é chamada, que faz uso da classe ExercicioRequests. A função delete é chamada e passa como parâmetros o scope, o token de sessão e o id de um exercício específico. Dentro dessa função é realizada um request para apagar o exercício e a resposta é verificada. Caso a resposta seja "Error: Delete Exercise fails", uma mensagem de erro ao remover o exercício é apresentada. Caso contrário, uma mensagem é exibida que informa de que o exercício foi removido com sucesso.

### Resources

Esta parte contém a parte física do software, composto por vários packages, no entanto, apenas 3 estão a ser utilizadas:

* Menus: possuí os menus utilizados na aplicação, neste caso foram apenas utilizados duas barras de navegação inferiores que têm o funcionamento de menu
* Layouts: possuí os ficheiros responsáveis pela montagem da parte física e interativa do utilizador com o software, cada layout está associado a uma activity apenas
* Drawables: possuí todos os recursos utilizados na aplicação, essencialmente imagens e ícones

Uma imagem com texto

Descrição gerada automaticamente

Figura - Organização do Front-End (Resources)

## Componentes físicos

O projeto apresenta apenas um componente físico externo à aplicação, que se trata de um leitor de cartões em Arduino (Sensor RFID).

O objeto deste aparelho seria fazer o registo de uma entrada e saída do ginásio IPCAGym a partir de cartões/pins eletrónicos, sendo assim registado na base de dados cada entrada e saída.

Para fazer a implementação deste projeto foi utilizado como IDE o Arduino IDE e Visual Studio Code, o mesmo foi desenvolvido na linguagem C, de forma a consolidar as metodologias da cadeira de Sistemas Embebidos e de Tempo Real (SETR).

O código desenvolvido encontra-se numa pasta externa a este documento.

Devido a danos de hardware vimo-nos impossibilitados de terminar este desenvolvimento.

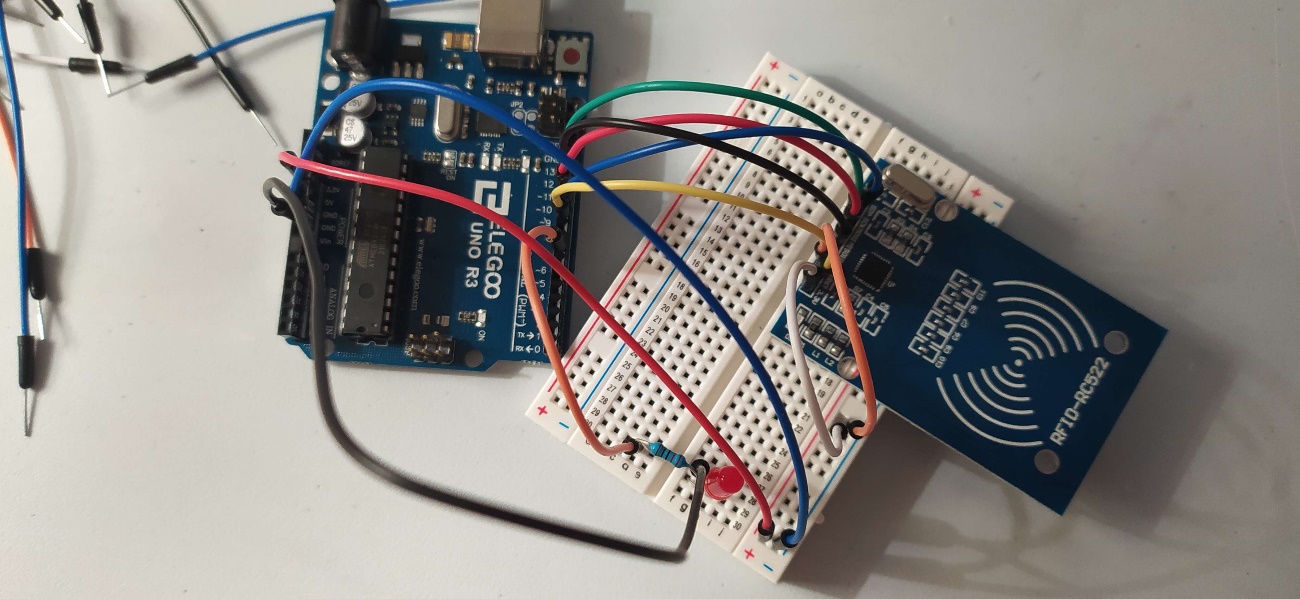


Figura - Componente físico, Arduino com Sensor RFID-RC522

# Avaliação

Segue-se a avaliação da entrega 4 do grupo de trabalho:

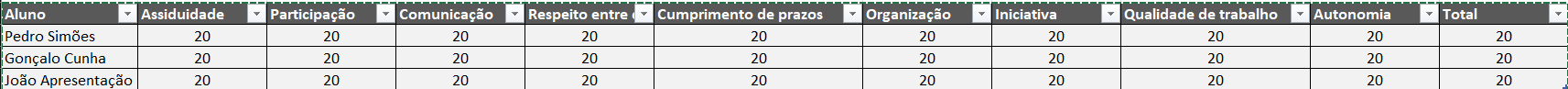


Figura 79 - Autoavaliação

# Conclusão

O progresso atual do projeto foi possível com o recurso a ferramentas de suporte, com uma distribuição por sprints semanais e organização de reuniões para estruturação das tarefas.

Com a 5ª entrega e finalização do projeto, foi possível observar toda uma trajetória de crescimento ao nível de projeto e ao nível do nosso conhecimento e raciocínio.

Ao nível do projeto vimos que foi possível concretizar praticamente tudo o que foi idealizado exceto a implementação do Arduino anteriormente justificada, mas sem qualquer tipo de problemas a equipa encontrou uma solução e é o que se espera em situações problemáticas como estas.

De forma resumida o grupo encontra-se satisfeito com o resultado e com a própria unidade curricular, por proporcionar este acréscimo ao nosso percurso académico.

# Bibliografia

**Repositório GitHub**

<https://github.com/Presentation12/Ipca_Gym>

**Figma**

<https://www.figma.com/file/Q4tM34gl91b9fhrGvUeXRs/MileriuPT's-teamlibrary?node-id=0%3A1&t=p9cWit1VJHUNwf2m-1>

**Sites para produção dos diagramas:**

<https://app.diagrams.net>

<https://online.visual-paradigm.com/pt/login.jsp>

**Sites para pesquisa e suporte no desenvolvimento de código:**

<https://stackoverflow.com>

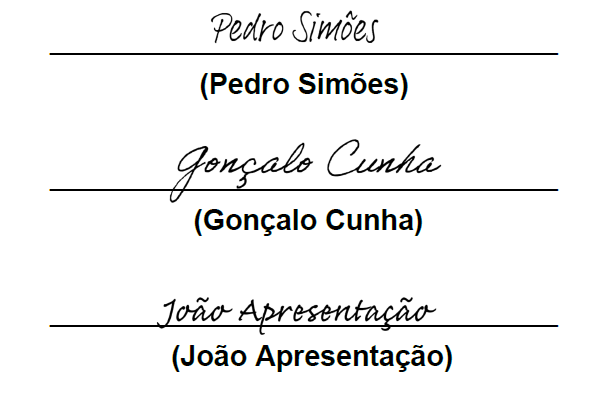
<https://dev.to>

<https://www.youtube.com>

**Material solicitado pelos docentes:**

* **ISI:**
  + <https://elearning2.ipca.pt/2223/pluginfile.php/601521/mod_resource/content/0/ESI-ISI%202022-23%20-%2005%20-%20Cloud%20-%2002%20-%20Web%20API%20Exemplo%20-%20ASP.NET%20Core.pdf>
  + <https://elearning2.ipca.pt/2223/pluginfile.php/604923/mod_resource/content/0/ESI-ISI%202022-23%20-%2005%20-%20Cloud%20-%2005b%20-%20OAuth%20RESTful%20Services_v2.pdf>
  + <https://elearning2.ipca.pt/2223/pluginfile.php/604543/mod_resource/content/0/ESI-ISI%202022-23%20-%2005%20-%20Cloud%20-%2006%20-%20Web%20API%20Documentação%20API%20.pdf>
  + <https://elearning2.ipca.pt/2223/pluginfile.php/604542/mod_resource/content/0/ESI-ISI%202022-23%20-%2005%20-%20Cloud%20-%2003%20-%20Cliente%20HTTP%20-%20ASP.NET%20Core.pdf>
* **PDM:**
  + <https://github.com/lgleto?tab=repositories>

# Assinaturas



80 - Assinaturas