



Universidade do Minho
Escola de Engenharia

Sports Manager

Mestrado Integrado em Engenharia Informática

Laboratórios de Informática IV
(2º Semestre - 2019/2020)

A84003	Beatriz de Freitas Rocha
A79973	Filipa Alexandra da Silva Faria
A85308	Filipe Miguel Teixeira Freitas Guimarães
A80749	Francisco Braga Urbano Rosa
A84073	Gonçalo Nuno Fernandes e Ferreira

Braga,
Maio 2020

Capítulo 1

Resumo

O nosso projeto intitular-se-á *Sports Manager* e tratar-se-á de um sistema de reservas de espaços de desporto e aulas temáticas direcionado a ginásios e espaços de lazer.

O nosso projeto deverá ser capaz de disponibilizar uma ferramenta eficaz para as empresas poderem arrendar espaços aos seus respetivos clientes e, consequentemente, estes poderem alugá-los.

Conteúdo

1	Resumo	1
2	Introdução	5
3	Modelação	6
3.1	Modelo conceptual	6
3.2	Modelo lógico	8
3.3	Diagrama de classes	9
3.4	<i>Use cases</i>	10
3.4.1	Criar conta	10
3.4.2	Apagar conta	10
3.4.3	Disponibilizar espaço	10
3.4.4	Mudar lotação de espaço	11
3.4.5	Mudar preço de aluguer de espaço	11
3.4.6	Autenticar	11
3.4.7	Inscrever em aula	12
3.4.8	Requisitar aluguer de espaço	12
3.4.9	Alterar dia/hora da reserva de um espaço	12
3.4.10	Cancelar reserva	12
3.4.11	Criar perfil de instrutor	13
3.4.12	Criar aula	13
4	Requisitos da aplicação	14
5	<i>Mockups</i>	15
5.1	<i>Landing page</i>	15
5.2	<i>Log in</i>	16
5.3	Registo de conta	16
5.4	Registo de informações pessoais	17
5.5	Contacto	17
5.6	Área de utilizador	18
5.7	Estatísticas do utilizador	18
5.8	Aulas do utilizador	19
5.9	Espaços do utilizador	19
5.10	Definições do utilizador	20
5.11	Instrutor	20
5.12	<i>Loading</i>	21
5.13	Erro	21

6	<i>BackEnd</i>	22
6.1	Autenticação	22
6.2	Criar conta	22
6.3	Comprar bilhete	23
6.4	Disponibilizar espaço	23
6.5	Criação de evento	23
6.6	Requisitar um espaço	23
6.7	Mudar lotação de Espaço	24
6.8	Mudar preço de aluguer de espaço	24
6.9	Alterar dia/hora de reserva	24
6.10	Apagar conta	24
7	<i>BackOffice</i>	25
7.1	Login	26
7.2	Main Page	27
7.3	Utilizadores	27
7.3.1	Utilizadores registados	27
7.3.2	Aulas frequentadas por um utilizador específico	27
7.3.3	Aulas frequentadas pelos utilizadores em média	28
7.3.4	Apresentação da informação	28
7.4	Aulas	28
7.4.1	Total de aulas registadas no sistema	28
7.4.2	Número médio de alunos por Aula	28
7.4.3	Lucro médio por aula	29
7.4.4	Apresentação da informação	29
7.5	Espaços	29
7.5.1	Número de Espaços registados no sistema	29
7.5.2	Nº de alugueres de um Espaço	29
7.5.3	Lucro de um Espaço e lucro médio por Espaço	30
7.5.4	Número de Aulas efetuadas num Espaço e número médio de Aulas efetuada por Espaço	30
7.5.5	Apresentação da informação	30
8	<i>Rest Api</i>	31
8.1	Autenticação	31
8.2	Instrutor	31
8.3	Utilizador	32
8.3.1	Aulas	32
8.3.2	Espaços	32
8.3.3	Definições	32
9	<i>FrontEnd</i>	33
9.1	Web-API	33
9.2	ReactJS	33
9.3	Cliente vs Instrutor	33
9.4	Google Maps API	34
9.5	Deployment	34
10	Conclusão e Análise de Resultados	36

Lista de Figuras

2.1	Logotipo da nossa aplicação	5
3.1	Modelo conceptual	7
3.2	Modelo lógico	9
3.3	Diagrama de classes	9
5.1	<i>Landing page</i>	15
5.2	Página de <i>log in</i>	16
5.3	Página de <i>sign up</i>	16
5.4	Página de registo	17
5.5	Página de contacto da empresa	17
5.6	Página principal de utilizador	18
5.7	Página com as estatísticas do utilizador	18
5.8	Página com as aulas do utilizador	19
5.9	Página com os espaços do utilizador	19
5.10	Página de definições	20
5.11	Página de instrutor	20
5.12	Página de <i>loading</i>	21
5.13	Página de erro	21
7.1	<i>Form Login</i>	26
7.2	<i>Form MainPage</i>	27
7.3	<i>Form Users</i>	28
7.4	<i>Form Aulas</i>	29
7.5	<i>Form Espaços</i>	30
9.1	Google Maps API	34
9.2	Pipeline FrontEnd	34
9.3	Release pipeline FrontEnd	35

Capítulo 2

Introdução

Este relatório tem em vista a documentação do projeto *Sports Manager* tratando-se este de um sistema de reservas de espaços de desporto e aulas temáticas. Este projeto surge no âmbito da disciplina de Laboratórios de Informática IV no 2º semestre do 3º ano do Mestrado Integrado em Engenharia Informática da Universidade do Minho. É uma aplicação baseada em JavaScript e ferramentas Microsoft que tem como público alvo empresas que disponibilizam espaços desportivos e aulas nos mesmos (e.g. ginásios). Assenta a sua importância nas mesmas, visto que possibilita à empresa organizar os seus espaços pelos seus clientes e permite que os mesmos se inscrevam em aulas nesses espaços.

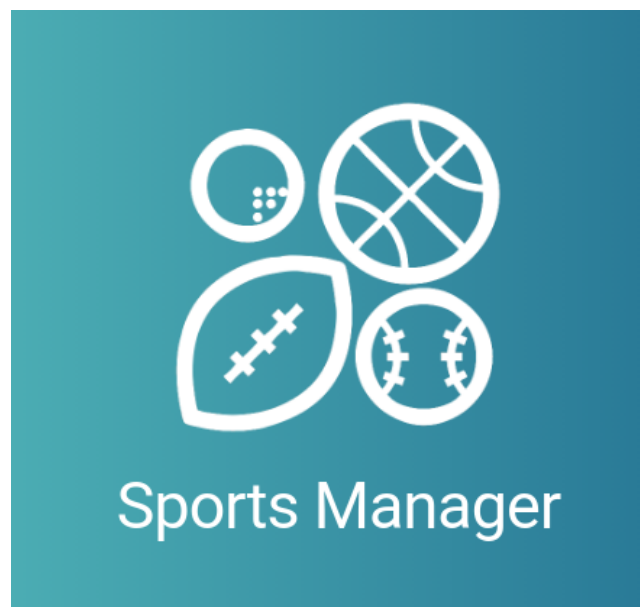


Figura 2.1: Logotipo da nossa aplicação

Capítulo 3

Modelação

3.1 Modelo conceptual

Neste capítulo iremos apresentar a modelação por que passou o nosso projeto. Assim sendo, achamos por bem começar por explicar a modelação da nossa base de dados. Após elaborarmos uma análise de mercado detalhada, chegamos àquilo que seria o modelo conceptual da nossa base de dados (que pode ser visto na figura abaixo). Neste, podemos verificar que irão existir três entidades no nosso projeto, nomeadamente Utilizador, Aula e Espaço. A chave primária do Utilizador será o seu *email* e terá, ainda, outros atributos associados, tais como:

- o seu NIF;
- a sua *password*;
- o seu número de telemóvel;
- o seu nome;
- o seu género;
- a sua morada;
- o seu peso, que se trata de um atributo multivalorado e composto pelos respetivos kilogramas e pela data em que a pesagem foi efetuada;
- a sua altura;
- a sua data de nascimento e
- o seu perfil, que irá fazer a distinção entre um instrutor e um cliente.

Já a chave primária do Espaço será o seu respetivo código e terá, ainda, outros atributos associados, tais como:

- a sua área;
- o seu preço;
- o seu local;

- a sua lotação máxima;
- o seu tipo, ou seja, se é um ginásio, se é um campo de ténis, etc;
- a data e hora em que fica disponível e
- a data e hora em que deixa de estar disponível.

Por último, a chave primária da Aula será também o seu respetivo código e terá, ainda, outros atributos associados, tais como:

- o número de bilhetes que estarão disponíveis;
- a modalidade, ou seja, ténis, basketball, etc;
- a data e hora em que começa;
- a data e hora em que termina e
- o preço de cada bilhete.

Passemos agora à explicação das relações. Um Utilizador poderá frequentar várias aulas e uma aula poderá ser frequentada por vários Utilizadores, caso estes sejam clientes. Contudo, um Utilizador poderá lecionar várias aulas e uma aula só poderá ser lecionada por um Utilizador, caso este seja um instrutor. Um Utilizador poderá, ainda, alugar vários Espaços e um Espaço poderá ser alugado por vários Utilizadores, tendo em atenção que ficará registada a data de início e de fim do aluguer. Por último, uma Aula decorre num Espaço e um Espaço poderá ter várias Aulas a decorrer.

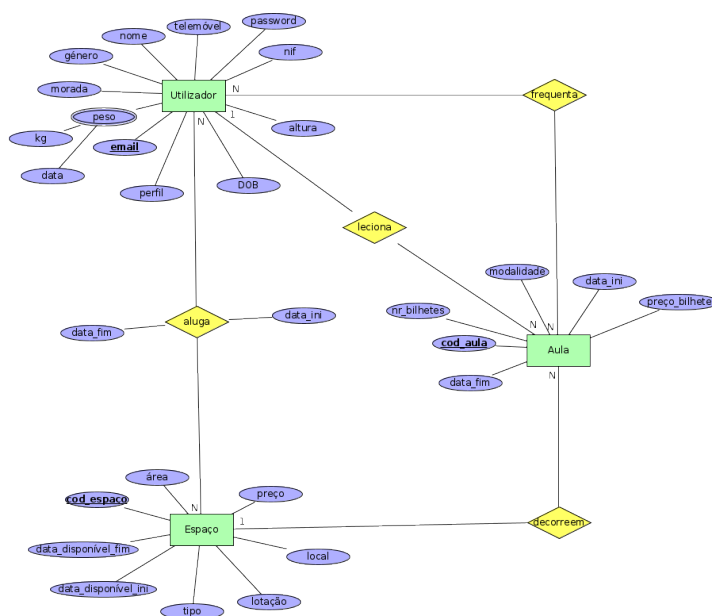


Figura 3.1: Modelo conceptual

3.2 Modelo lógico

Após analisar este modelo e validá-lo, passamos para o desenvolvimento do modelo lógico (que pode ser visto na figura abaixo) recorrendo à ferramenta MySQL Workbench para o efeito.

Começemos por explicar a tabela **UTILIZADOR**. O *email* é um VARCHAR(256) e vai ser usado como chave primária, portanto não pode ser nulo. Para o NIF optamos por um VARCHAR(9), uma vez que cada NIF tem, exatamente, 9 dígitos. O género é um VARCHAR(1), que poderá conter F (Feminino) ou M (Masculino). Os atributos nome, telemóvel, *password*, morada e perfil serão armazenados em VARCHAR(128), VARCHAR(20), VARCHAR(16), VARCHAR(256) e VARCHAR(45), respetivamente. Por último, a data de nascimento e a altura serão armazenadas em DATE e INT, respetivamente.

O código é um INT e vai ser usado como chave primária da tabela **AULA**, portanto, não pode ser nulo. Da mesma maneira, o número de bilhetes também será armazenado num INT. Para armazenar a data de início e a data de fim optamos por DATETIME, pois, desta maneira, conseguimos guardar não só a data, mas também a hora. Por fim, para armazenar o preço de cada bilhete e a modalidade optamos por FLOAT e VARCHAR(36), respetivamente.

O código é um INT e vai ser usado como chave primária da tabela **ESPAÇOS**, portanto, não pode ser nulo. Da mesma maneira, a lotação máxima e a área serão representadas por um INT. Para armazenar a data em que o Espaço fica e deixa de estar disponível optamos por um DATETIME, pois, desta maneira, conseguimos guardar não só a data, mas também a hora. Por último, para armazenar os atributos preço, tipo e local optamos por FLOAT, VARCHAR(45) e VARCHAR(256), respetivamente.

Visto que um Utilizador e uma Aula partilham de uma relação N para N (quando o Utilizador se trata de um cliente), tivemos de criar uma nova tabela designada por **FREQUENTA**. Esta, por sua vez, guarda o código da Aula e o *email* do Utilizador.

Da mesma maneira, um Utilizador e um Espaço também partilham de uma relação N para N, portanto também tivemos de criar uma nova tabela designada por **ALUGA** que, por sua vez, guarda o código do Espaço, o *email* do Utilizador e, ainda, a data de início e de fim do aluguer, que são armazenadas num DATETIME, para que fique registada não só a data, mas também a hora.

Por último, uma vez que o atributo peso da entidade Utilizador é multivalorado, tivemos de criar uma tabela extra designada por **PESO** apenas para ele. A data da pesagem é armazenada em DATE e corresponde à sua chave primária, portanto não pode ser nula. Esta tabela guarda, ainda, os respetivos kilogramas, que são armazenados num FLOAT.

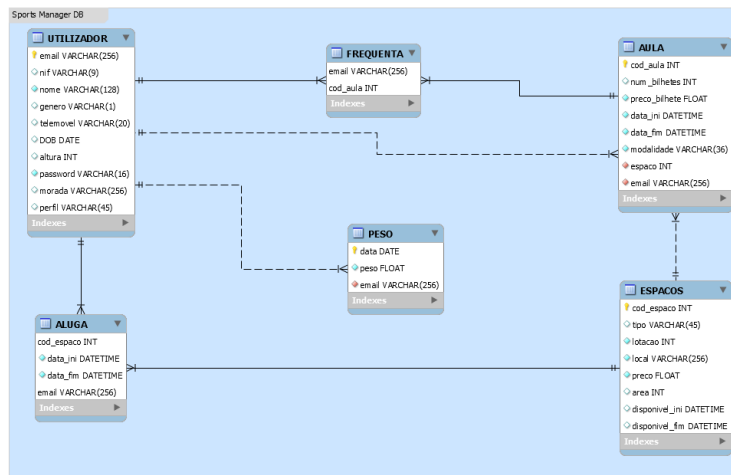


Figura 3.2: Modelo lógico

3.3 Diagrama de classes

Após apostar na eficácia da nossa base de dados, sentimos a necessidade de criar um modelo que representasse as classes (Figura 3.3) que teremos na nossa implementação em C# da web-app.

Posto isto, teremos, então, 8 classes (Administrador, Sports Manager, Utilizador, Instrutor, Aula, Espaço, Aluguer e Peso). Grande parte do que foi dito acima pode ser visto aqui, portanto vamos apenas explicar as classes Administrador, Sports Manager, Instrutor e Utilizador.

A classe Sports Manager guarda um atributo permissão para que seja possível saber quando o Administrador está a fazer alterações na aplicação. Este, por sua vez, guarda o respetivo *email* e *password* da empresa que possui a nossa aplicação. Por fim, tivemos de criar duas classes distintas para o cliente (Utilizador) e instrutor (Instrutor). A classe Utilizador possui os mesmos atributos que já referimos anteriormente, à semelhança da classe Instrutor que apenas descarta os atributos altura, data de nascimento e morada.

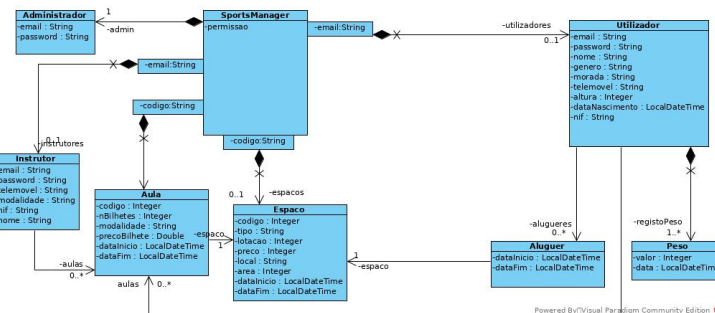


Figura 3.3: Diagrama de classes

3.4 Use cases

Para definir que funcionalidades necessitaríamos de ter na nossa aplicação, criamos alguns *use cases* que podem ser vistos nesta secção.

3.4.1 Criar conta

Descrição: O cliente cria uma conta na aplicação Sports Manager.

Pré-condição: O cliente está na página Sign Up.

Pós-condição: O cliente deverá ter uma conta criada.

Fluxo Normal:

1. O sistema pede ao cliente que insira o *email* da nova conta e a *password*.
2. O cliente insere o *email* e a *password*.
3. O sistema verifica se o *email* está disponível.
4. O sistema pede ao cliente o seu nome, o género, a morada, o número de telemóvel, a altura, a data de nascimento, o NIF e o seu peso atual.
5. O cliente insere as informações que o sistema pediu.
6. O sistema cria a conta com esse *email*.

Fluxo de exceção 1 [O *email* escolhido já está em uso] (passo 3):

- 3.1. O sistema informa que o nome de utilizador já está em uso.

3.4.2 Apagar conta

Descrição: O cliente apaga uma conta na aplicação Sports Manager.

Pré-condição: O cliente deverá estar autenticado na sua conta e na página de definições de conta.

Pós-condição: A conta do cliente é apagada.

Fluxo Normal:

1. O cliente indica que pretende apagar a sua conta.
2. O sistema pede ao cliente que indique a sua *password*.
3. O cliente insere a *password* da conta.
4. O sistema verifica se a *password* inserida corresponde à *password* da conta.
5. O sistema apaga a respetiva conta.

Fluxo de exceção 1 [A *password* introduzida não corresponde à *password* da conta] (passo 4):

- 4.1. O sistema indica que a *password* introduzida não corresponde à *password* da conta.

3.4.3 Disponibilizar espaço

Descrição: A empresa disponibiliza um espaço para aluguer.

Pré-condição: A empresa deverá estar autenticada na sua conta.

Pós-condição: O espaço fica disponível para aluguer.

Fluxo Normal:

1. A empresa indica ao sistema que pretende disponibilizar um novo espaço para aluguer.
2. O sistema pede à empresa que indique o tipo de espaço, a lotação máxima, o preço de aluguer, o local onde se encontra o espaço e a área total.
3. A empresa fornece as informações pedidas pelo sistema.
4. O sistema valida as informações fornecidas.

5. O sistema passa a disponibilizar o novo espaço.
Fluxo de exceção 1 [O utilizador insere um campo igual ou inferior a 0] (passo 4):
4.1. O sistema informa que os campos lotação máxima, preço de aluguer ou área total têm valor igual ou inferior a 0.

3.4.4 Mudar lotação de espaço

Descrição: A empresa muda a lotação de um espaço já existente.

Pré-condição: A empresa deverá estar autenticada na sua conta.

Pós-condição: O espaço tem uma nova lotação máxima.

Fluxo Normal:

1. A empresa indica ao sistema que pretende mudar a lotação de um espaço existente.
2. O sistema pede à empresa que indique a nova lotação do espaço.
3. A empresa indica a nova lotação.
4. O sistema verifica se a lotação é superior a 0.
5. O sistema atualiza a lotação do espaço.

Fluxo de exceção 1 [A lotação é igual ou inferior a 0] (passo 4):

- 4.1. O sistema informa que a lotação indicada é igual ou inferior a 0.

3.4.5 Mudar preço de aluguer de espaço

Descrição: A empresa muda o preço de aluguer de um espaço já existente.

Pré-condição: A empresa deverá estar autenticada na sua conta.

Pós-condição: O espaço tem um novo preço de aluguer.

Fluxo Normal:

1. A empresa indica ao sistema que pretende mudar o preço de aluguer de um espaço existente.
2. O sistema pede à empresa que indique o novo preço de aluguer do espaço.
3. A empresa indica o novo preço.
4. O sistema verifica se o preço indicado é superior a 0.
5. O sistema atualiza o preço de aluguer do espaço.

Fluxo de exceção 1 [O preço indicado é igual ou inferior a 0] (passo 4):

- 4.1. O sistema informa que o preço indicado é igual ou inferior a 0.

3.4.6 Autenticar

Descrição: O cliente efetua a autenticação na aplicação Sports Manager.

Pré-condição: O cliente encontra-se na página Login.

Pós-condição: O cliente fica autenticado na sua conta.

Fluxo Normal:

1. O sistema pede ao cliente que indique o seu *email* e a sua *password*.
2. O cliente indica o *email* e a *password*.
3. O sistema verifica se o *email* existe.
4. O sistema verifica se a *password* está correta.
5. O sistema autentica o cliente.

Fluxo de exceção 1 [O *email* fornecido pelo cliente não existe] (passo 3):

- 3.1. O sistema informa que o *email* introduzido não existe no sistema.

Fluxo de exceção 2 [A *password* introduzida está incorreta] (passo 4):

4.1. O sistema informa que a *password* introduzida está incorreta.

3.4.7 Inscrever em aula

Descrição: O cliente inscreve-se numa aula.

Pré-condição: O cliente deverá estar autenticado na sua conta e na página Manage Classes.

Pós-condição: O cliente fica inscrito numa aula

Fluxo Normal:

1. O cliente seleciona uma aula da lista de aulas disponíveis e indica ao sistema que pretende inscrever-se na mesma.
2. O sistema verifica o número de vagas disponíveis.
3. O sistema indica ao cliente que a inscrição foi feita com sucesso.

Fluxo de exceção 1 [A aula não tem vagas disponíveis] (passo 2):

- 2.1. O sistema informa que a aula já não tem vagas disponíveis.

3.4.8 Requisitar aluguer de espaço

Descrição: O cliente requisita o aluguer de um espaço.

Pré-condição: O cliente deverá estar autenticado na sua conta e na página Manage Places.

Pós-condição: O cliente tem uma reserva de um espaço.

Fluxo Normal:

1. O cliente indica ao sistema que pretende alugar um espaço.
2. O sistema informa que o aluguer foi feito com sucesso.

3.4.9 Alterar dia/hora da reserva de um espaço

Descrição: O cliente altera a data/hora de uma reserva de um espaço.

Pré-condição: O cliente deverá estar autenticado na sua conta e na página Manage Places.

Pós-condição: A reserva do cliente tem uma nova data e hora.

Fluxo Normal:

1. O cliente indica ao sistema que pretende mudar a data/hora de uma reserva.
2. O sistema pede ao cliente que indique a nova data e hora.
3. O cliente indica a nova data e hora para a sua reserva.
4. O sistema verifica se, nessa data e hora, o espaço se encontra disponível.
5. O sistema efetua a alteração de data e hora.

Fluxo de exceção 1 [O espaço não se encontra disponível na data e hora especificadas pelo cliente] (passo 4):

4.1 O sistema informa que, na data e hora indicadas, o espaço está indisponível para aluguer.

3.4.10 Cancelar reserva

Descrição: O cliente cancela uma reserva.

Pré-condição: O cliente deverá estar autenticado na sua conta e na página Manage Places.

Pós-condição: A reserva do cliente é cancelada.

Fluxo Normal:

1. O cliente indica ao sistema que pretende cancelar uma reserva existente.
2. O sistema cancela a reserva.

3.4.11 Criar perfil de instrutor

Descrição: A empresa cria um perfil de instrutor na aplicação Sports Manager.

Pré-condição: A empresa deverá estar autenticada na sua conta.

Pós-condição: É criado um novo perfil de instrutor.

Fluxo Normal:

1. A empresa pede ao sistema para criar um perfil de instrutor.
2. O sistema pede à empresa que insira o nome, a *password*, o telemóvel, o *email*, a modalidade e o NIF do instrutor.
3. A empresa insere as informações que o sistema pediu.
4. O sistema cria o novo perfil de instrutor.

3.4.12 Criar aula

Descrição: O instrutor cria uma nova aula.

Pré-condição: O instrutor deverá estar autenticado na sua conta.

Pós-condição: É criada uma nova aula.

Fluxo Normal:

1. O instrutor indica ao sistema que pretende criar uma nova aula.
2. O sistema pede ao instrutor que indique qual o nome da aula, o espaço onde irá decorrer, o preço, a lotação, a data e hora de início e a data e hora de fim.
3. O sistema verifica se, na data e hora indicadas, o espaço se encontra disponível para a realização da aula.
4. O sistema cria a nova aula.

Fluxo de exceção 1 [O espaço não está disponível na data e hora pretendida] (passo 3):

- 3.1. O sistema informa que o espaço não se encontra disponível na data e hora indicadas.

Capítulo 4

Requisitos da aplicação

Para determinarmos que funcionalidades teria a nossa aplicação, decidimos olhar para cada um dos intervenientes e desenvolver as funcionalidades básicas separando em **empresa**, **instrutor** e **cliente**.

A empresa deverá ter ao seu dispor as seguintes funcionalidades:

- Criação de espaços para aluguer (definindo o local, a que horas está disponível, a lotação máxima e o custo associado)
- Criação de novos perfis para instrutores
- Gestão dos espaços disponíveis
- Criação de eventos utilizando os espaços existentes (definindo a lotação e o preço)
- Consultas estatísticas

O instrutor deverá ter ao seu dispor as seguintes funcionalidades:

- Consultas estatísticas, nomeadamente o lucro que teve com as aulas lecionadas em cada mês
- Criação de aulas
- Edição de aulas
- Cancelamento de aulas

O cliente deverá ter ao seu dispor as seguintes funcionalidades:

- Autenticação
- Requisição de espaços
- Gestão de reservas/aulas (cancelar, trocar)
- Inscrição em aulas
- *Personal trainer* digital
- Consultas estatísticas

Capítulo 5

Mockups

Para termos a perspetiva de como a nossa aplicação se iria comportar construímos os *mockups* iniciais. Para que os pudéssemos usar no projeto final decidimos criá-los utilizando React JS que se trata da *framework* usada na nossa aplicação final. Um dos pontos em que nos quisemos focar nesta fase de modelação foi tornar estes *mockups* responsivos. O nosso objetivo resumiu-se em criar uma *web-app* e, como tal, as páginas devem adaptar-se a qualquer tipo de ecrã, quer se trate de um computador, de um *tablet* ou até mesmo de um telemóvel. De um ponto de vista geral, o processo de tornar estas páginas responsivas consistiu em aplicar sempre valores relativos às margens, ter em atenção a partir de que píxeis as páginas estavam a ser partidas e ser muito cuidadosos na aplicação do preenchimento.

5.1 *Landing page*

A *landing page* serve como cartão de entrada para um utilizador ou instrutor entrar na nossa aplicação.

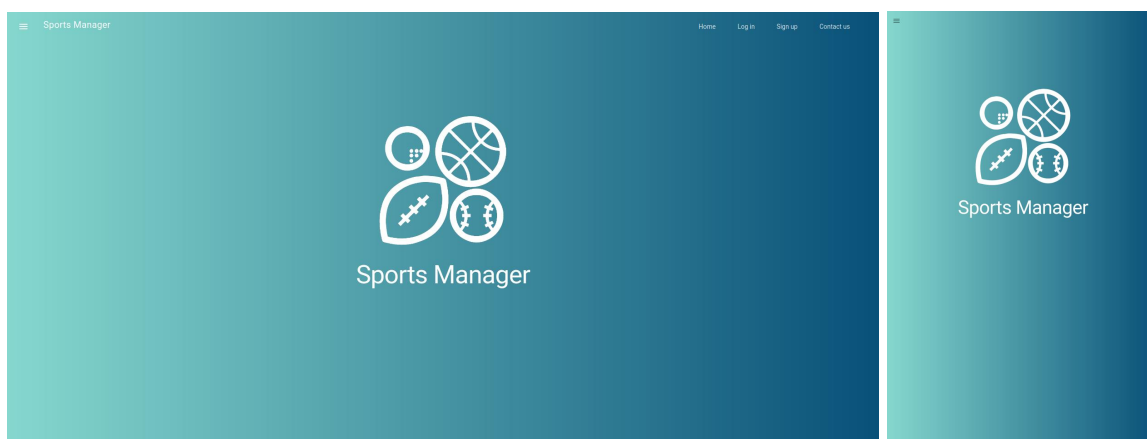
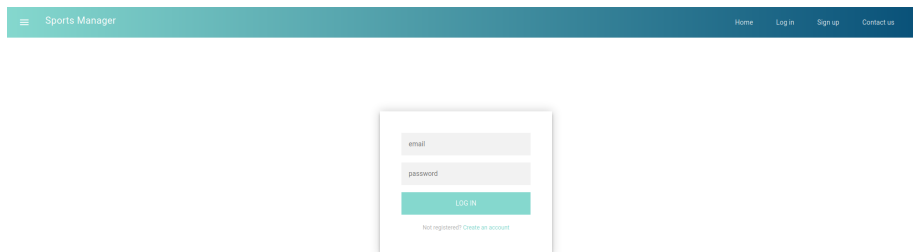


Figura 5.1: *Landing page*

5.2 *Log in*

A página de *log in* serve para utilizadores e instrutores já registados entrarem na sua área pessoal.

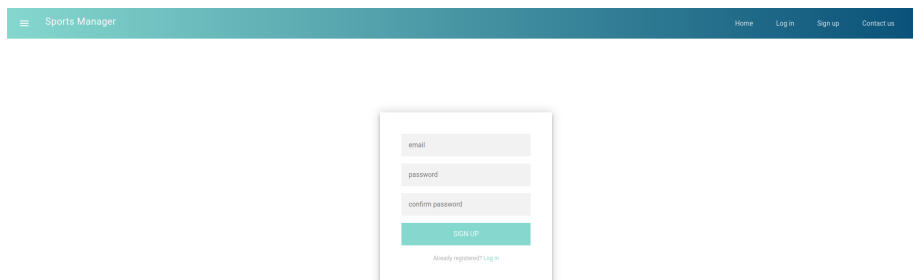


The screenshot shows the 'Log In' page of the 'Sports Manager' application. At the top, there is a teal header bar with the 'Sports Manager' logo on the left and navigation links 'Home', 'Log in', 'Sign up', and 'Contact us' on the right. The main content area features a white login form with two input fields labeled 'email' and 'password'. Below these fields is a teal 'LOG IN' button. At the bottom of the form, there is a link that reads 'Not registered? Create an account'.

Figura 5.2: Página de *log in*

5.3 Registo de conta

É uma página de pré-registo onde o utilizador fornece informações como *email* e *password*.



The screenshot shows the 'Sign Up' page of the 'Sports Manager' application. It has the same teal header bar as the login page. The main content area features a white sign-up form with three input fields labeled 'email', 'password', and 'confirm password'. Below these fields is a teal 'SIGN UP' button. At the bottom of the form, there is a link that reads 'Already registered? Log in'.

Figura 5.3: Página de *sign up*

5.4 Registo de informações pessoais

Esta página é complementar à do registo. É nela que o utilizador insere informações pessoais como o nome, género, etc.

The image shows two versions of a personal information registration form. The left version is a desktop layout with a large profile picture placeholder and a form with fields for Name, Gender (Female/Male), Address, Phone number, Date of birth (Day/Month/Year), Tax ID number, Height, and Weight, followed by a SUBMIT button. The right version is a mobile layout with a smaller profile picture placeholder and a form with the same fields, also followed by a SUBMIT button. Both versions have a 'Sports Manager' header and a 'Home' link.

Figura 5.4: Página de registo

5.5 Contacto

Serve como um meio de contacto entre a empresa e o cliente no caso de algum problema com algum espaço ou até mesmo com a própria aplicação.

The image shows two versions of a company contact form. The left version is a desktop layout with a 'CONTACT US' header, contact information (phone 123 456 789, email someone@example.com), and a form with fields for email, subject, and message, followed by a SUBMIT button. The right version is a mobile layout with the same header and contact information, and a form with the same fields, also followed by a SUBMIT button.

Figura 5.5: Página de contacto da empresa

5.6 Área de utilizador

Página em que o utilizador entra após o *log in*.

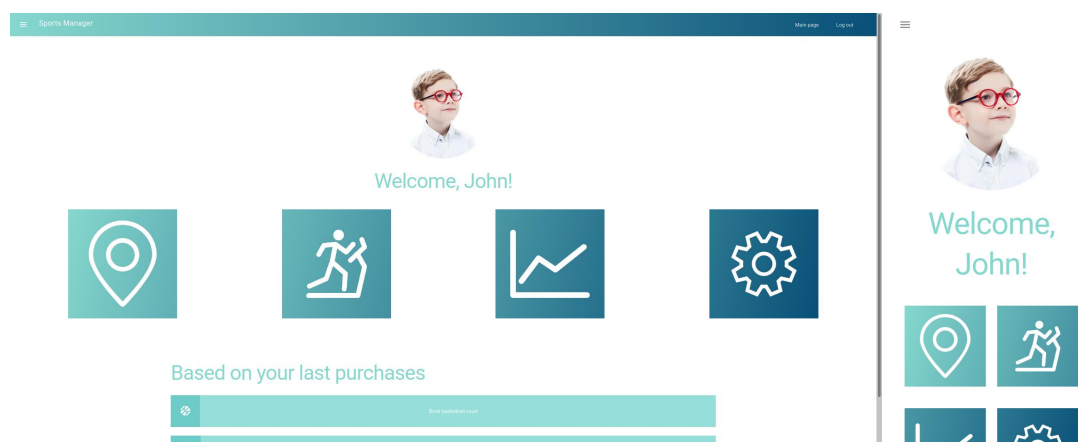


Figura 5.6: Página principal de utilizador

5.7 Estatísticas do utilizador

Página para que o utilizador consiga ver as suas estatísticas, tais como os meses em que gastou mais dinheiro, as aulas que mais frequentou, etc.

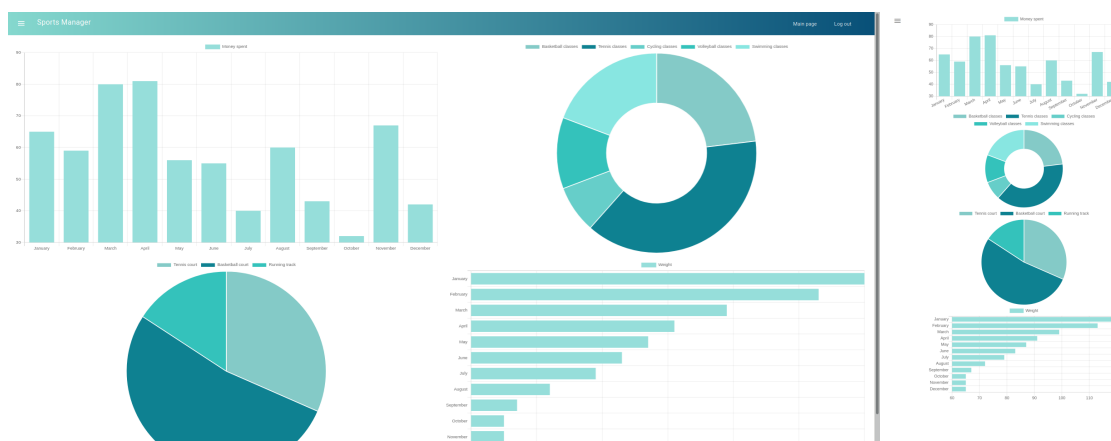


Figura 5.7: Página com as estatísticas do utilizador

5.8 Aulas do utilizador

Área para o utilizador se inscrever em aulas e ver as suas inscrições atuais.

Sports Manager

Main pageLog out

Next classes

Class	Place	Price	Capacity	Begin	End
<input checked="" type="checkbox"/> Aerobics	Gym	2.9	20	12:00	13:00
<input checked="" type="checkbox"/> Cycling	Gym	1.25	15	16:00	17:00
<input checked="" type="checkbox"/> Tennis	Tennis court	2.35	6	18:00	20:00

Available classes

Class	Place	Price	Capacity	Begin	End
<input type="checkbox"/> Aerobics	Gym	2.9	20	12:00	13:00
<input type="checkbox"/> Cycling	Gym	1.25	15	16:00	17:00
<input type="checkbox"/> Tennis	Tennis court	2.35	6	18:00	20:00

Figura 5.8: Página com as aulas do utilizador

5.9 Espaços do utilizador

Área para o utilizador controlar as suas reservas de espaços.

Sports Manager

Main pageLog out

Next appointments

Place	Location	Area (m²)	Capacity	Price	Weather	Begin	End
<input checked="" type="checkbox"/> Gym	Braga	170	20	20	Sunny	16:00	18:00
<input checked="" type="checkbox"/> Tennis court	Guimarães	100	15	10	Rainy	14:00	16:00
<input checked="" type="checkbox"/> Basketball court	Viana do Castelo	120	12	15	Cloudy	10:00	12:00

Available places

Place	Location	Area (m²)	Capacity	Price	Weather	Begin	End
<input type="checkbox"/> Gym	Braga	170	20	20	Sunny	16:00	18:00
<input type="checkbox"/> Tennis court	Guimarães	100	15	10	Rainy	14:00	16:00
<input type="checkbox"/> Basketball court	Viana do Castelo	120	12	15	Cloudy	10:00	12:00

Figura 5.9: Página com os espaços do utilizador

5.10 Definições do utilizador

Área para o utilizador alterar as definições da sua conta.

The screenshot shows the 'Sports Manager' application interface. The top navigation bar includes a hamburger menu, the text 'Sports Manager', and links for 'Main page' and 'Log out'. The main content area is divided into two sections. On the left, there is a large circular placeholder for a user profile picture. On the right, there is a form for updating user information. The form fields are: Email, Password, Name, Gender (with radio buttons for Female and Male), Address, Phone number, Date of birth (with dropdowns for Day, Month, and Year), Height, and Weight. A sidebar on the far right shows a smaller version of the user profile and the settings form.

Figura 5.10: Página de definições

5.11 Instrutor

Será a página em que o instrutor entra após o *log in*.

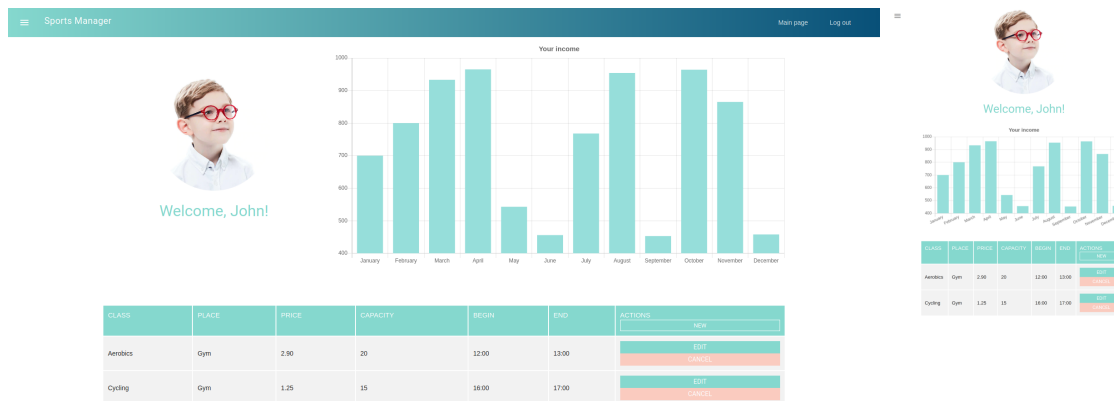


Figura 5.11: Página de instrutor

5.12 *Loading*

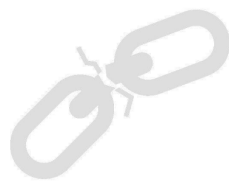
Será a página apresentada caso esteja a carregar alguma informação.



Figura 5.12: Página de *loading*

5.13 Erro

Será a página apresentada caso ocorra algum erro na aplicação.



Oops! It looks like we're having some trouble...

Figura 5.13: Página de erro

Capítulo 6

BackEnd

Nesta parte iremos explicar, com algum detalhe, a implementação do *BackEnd* da nossa aplicação, nomeadamente a implementação dos *use cases* anteriormente descritos. A base de implementação do *BackEnd* são os *use case* e o modelo lógico especificados no capítulo 3.

6.1 Autenticação

O processo de autenticação é bastante simples e começa por utilizar o método `containsKey()` que verifica a existência da componente *email* na base de dados. Em caso afirmativo, o sistema prossegue à validação da combinação *username-password*. Para tal, faz uso do método `passwordMatch()` que interroga a base de dados se o *email* e *password* que o utilizador introduziu correspondem à informação que está presente na base de dados.

O método `containsKey()` tem como argumento o *email* que o utilizador introduz quando faz *log in* e o método `passwordMatch()`, para além do *email*, tem também como argumento a *password* que o utilizador introduz. De referir, que a *password* que é passada à base de dados para verificação é alvo de uma função de *hash* para maior segurança.

6.2 Criar conta

O processo de criar uma conta é algo que não requer uma lógica muito avançada, pelo que simplesmente verificamos se o *email* que o utilizador quer registar já existe na base de dados (utilizando de novo o método `containsKey()`). Se não existir, registamos o utilizador na base de dados através do método `put()` que tem como argumentos o *email*, o nome, a morada, o género, o número de telemóvel, o NIF, a altura, a *password*, a data de nascimento e o perfil do utilizador. Mais uma vez, a *password* que é enviada para a base de dados é alvo de uma função de *hash* para uma maior segurança e proteção dos dados.

6.3 Comprar bilhete

A implementação deste *use case* teve de ser feita com cautela, pois requer uma lógica aplicacional ponderada de forma a garantir o seu correto funcionamento. Desde logo, é necessário garantir que um utilizador nunca consegue comprar um bilhete para uma aula para a qual já se inscreveu (isto é, para uma aula que já comprou bilhete), sendo isto feito através do método `comprouBilhete()` que tem como argumento o código da aula e o *email* do utilizador. Adicionalmente, é preciso garantir que, no momento da compra, a aula ainda tem bilhetes disponíveis, sendo que efetuamos essa verificação com o método `bilhetesDisponíveis()` que tem como argumento o código da aula. Se, depois destas verificações, o sistema permitir a compra do bilhete, é adicionado o código da aula e o *email* do utilizador à tabela **FREQUENTA** que associa que utilizadores participaram em determinada aula. Esta adição é feita utilizando o método `addUserToAula()`, tendo como argumentos o código da aula e o *email* do utilizador.

Um aspeto muito importante deste *use case* é o controlo de concorrência. De facto, é necessário garantir que existe um controlo de concorrência no que toca à compra de bilhetes, por exemplo. Um caso muito prático é aquele em que resta apenas um bilhete para uma aula e dois utilizadores o querem comprar. Sem controlo de concorrência, ambos comprariam o bilhete sem qualquer tipo de problema (procedimento errado). Com controlo de concorrência, apenas um deles compra o bilhete (procedimento correto).

6.4 Disponibilizar espaço

O processo de disponibilização de um novo espaço é bastante simples. O *BackEnd* recebe a informação relativa ao espaço (o tipo, a lotação, o local, o preço, a área, a data de início e a data de fim) do *FrontEnd* e através do método `put()` o espaço é adicionado à base de dados. O atributo `cod_espaco` (identificador do espaço) é auto-incrementável, pelo que também esta operação precisa de controlo de concorrência.

6.5 Criação de evento

O processo de criação de um novo evento é relativamente simples, uma vez que apenas necessitamos de aferir a disponibilidade do espaço a certa hora e dia. De forma a criar um evento, primeiro é necessário verificar a disponibilidade do espaço através do método `spaceAvailable()`. Se o espaço estiver disponível, é inserido um novo registo na tabela **ALUGA** de forma a reservar o espaço para este evento através do método `rent()` e é inserido na tabela **AULA** o registo do evento criado através do método `put()`.

6.6 Requisitar um espaço

O processo de requisição de um espaço é quase igual, logicamente falando, ao da subsecção anterior, com exceção da criação do evento em si. Deste modo, o

processo é idêntico ao descrito anteriormente, sendo que apenas não efetuamos a criação do evento.

6.7 Mudar lotação de Espaço

Este processo é bastante simples e apenas é necessário selecionar na base de dados o registo da tabela **ESPACOS** com o código desejado e alterar o atributo **lotacao** para o seu novo valor. Este processo é executado através do método `update()` que altera o atributo desejado para o novo valor no registo identificado por um determinado código de espaço.

6.8 Mudar preço de aluguer de espaço

Este processo é bastante simples e apenas é necessário selecionar na base de dados o registo da tabela **ESPACOS** com o código desejado e alterar o atributo **preco** para o seu novo valor. Este processo é executado através do método `update()` que altera o atributo desejado para o novo valor no registo identificado por um determinado código de espaço.

6.9 Alterar dia/hora de reserva

A alteração de um dia/hora de reserva pode ser quer no período de início quer no período de fim. Deste modo, interrogamos a base de dados se a nova data (dia e hora) de início ou fim está disponível para determinado espaço através do método `canChangeDate()`. Se tal for possível, altera-se a reserva com a nova data através do método `updateReserva()`.

6.10 Apagar conta

Para apagar uma conta da base de dados, selecionamos o registo identificado pelo *email* do utilizador que faz o pedido e removemos esse registo. Este processo é feito com o método `deleteUser()` que, por sua vez, chama os métodos `remove()` e `logout()`.

Capítulo 7

BackOffice

Neste capítulo iremos detalhar a implementação de uma aplicação de *BackOffice*, mais concretamente uma aplicação de gestão de todo o sistema.

Esta aplicação apresenta estatísticas sobre o estado do sistema, como por exemplo quantos utilizadores temos registados, quantas aulas já existiram, etc. Não é uma aplicação terminada, isto é, tem sempre espaço para crescer, espaço para serem adicionadas mais funcionalidades que os administradores do sistema considerem úteis para a análise do negócio.

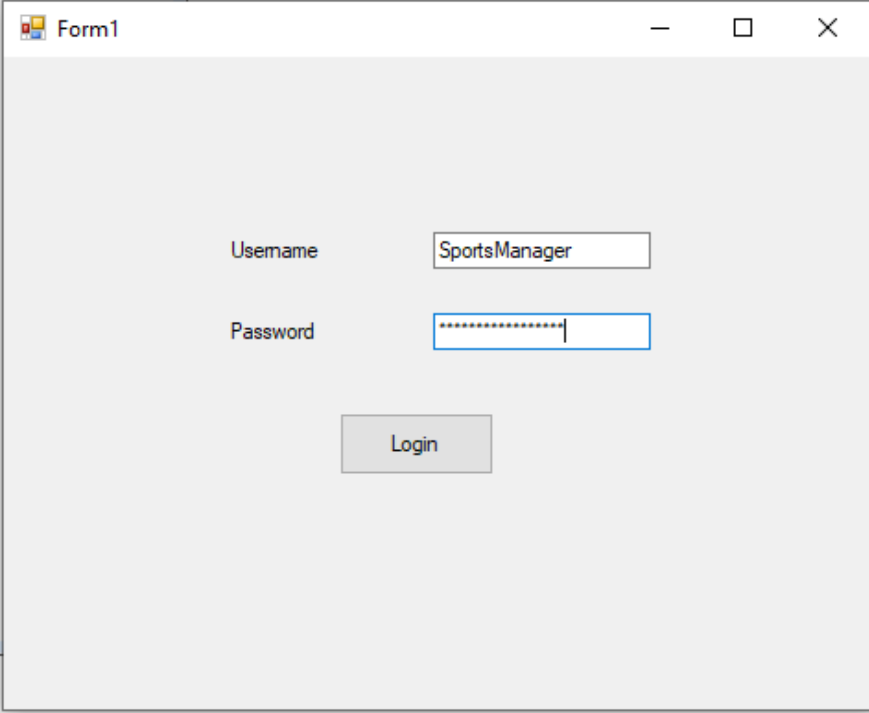
A implementação desta aplicação foi feita através de uma interface gráfica, recorrendo à ferramenta Windows Forms do Visual Studio, que permite ao utilizador visualizar o estado do sistema e ter uma melhor perceção do negócio. Tal como o *BackEnd*, esta aplicação faz uso da base de dados implementada no Azure para efetuar operações de consulta, entre outras.

O administrador que pretender utilizar esta aplicação deverá efetuar o login com o user e password de administrador e a partir daí pode interagir com a aplicação de diversas formas: ver algumas estatísticas sobre os Utilizadores, sobre os Espaços registados no sistema ou sobre as Aulas já dadas.

A apresentação da informação, como o próprio nome da ferramenta utilizada indica, é feita através de Forms. Nestes, a informação que é mostrada ao utilizador (informação que resulta de consulta à Base de Dados e à lógica do programa) é colocada em TextBox em modo de ReadOnly para não permitir alteração do conteúdo. Informação que dependa de uma seleção (por exemplo número de aulas totais frequentadas pelo utilizador X) é apresentada também numa TextBox, mas o seu conteúdo depende do item selecionado na ComboBox onde são apresentadas todas as alternativas possíveis num drop-down. Estas consultas só avançam uma vez pressionado o botão respetivo de confirmação da seleção. A navegação entre Forms é feita através de Buttons, sendo que em todos os Forms onde se justifique existe a presença de um Button de regresso e nos restantes um Button indicativo do Form para o qual vamos navegar.

7.1 Login

Esta aplicação está protegida por um login de administrador para que apenas os administradores do sistema possam ter acesso ao mesmo.



The image shows a screenshot of a Windows-style application window titled "Form1". The window has a light gray background and contains a login form. The form consists of two labels, "Username" and "Password", each followed by a text input field. The "Username" field contains the text "SportsManager". The "Password" field contains a series of dots, indicating a masked password. Below the input fields is a button labeled "Login". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Figura 7.1: *Form Login*

7.2 Main Page

Efetuada o login, o administrador tem a possibilidade de consultar informações acerca dos Utilizadores, das Aulas e dos Espaços existentes no sistema, como podemos comprovar pela figura abaixo:

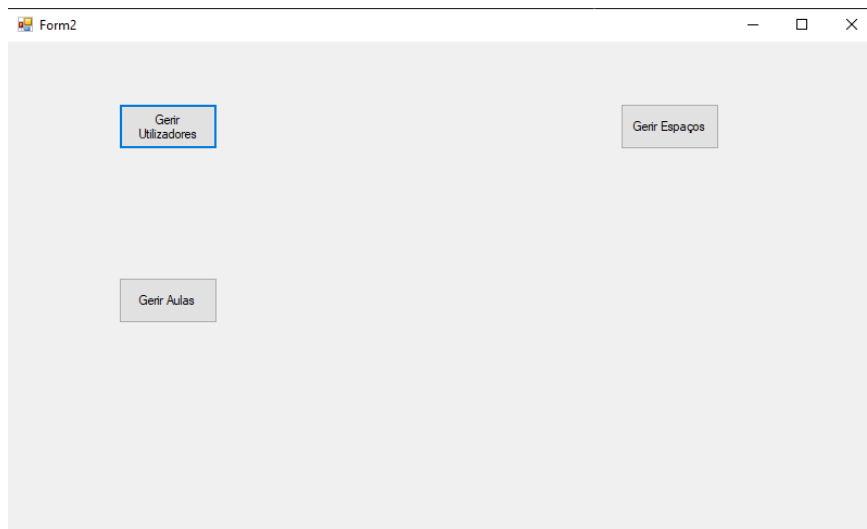


Figura 7.2: *Form MainPage*

7.3 Utilizadores

7.3.1 Utilizadores registados

Uma funcionalidade básica que se pede numa aplicação de administração de sistemas que dependem de utilizadores é quantos utilizadores já se registaram e possuem um registo ativo no sistema. Deste modo, interrogamos a base de dados sobre o número de registos existentes, isto é, quantas chaves primárias da tabela Utilizador existem.

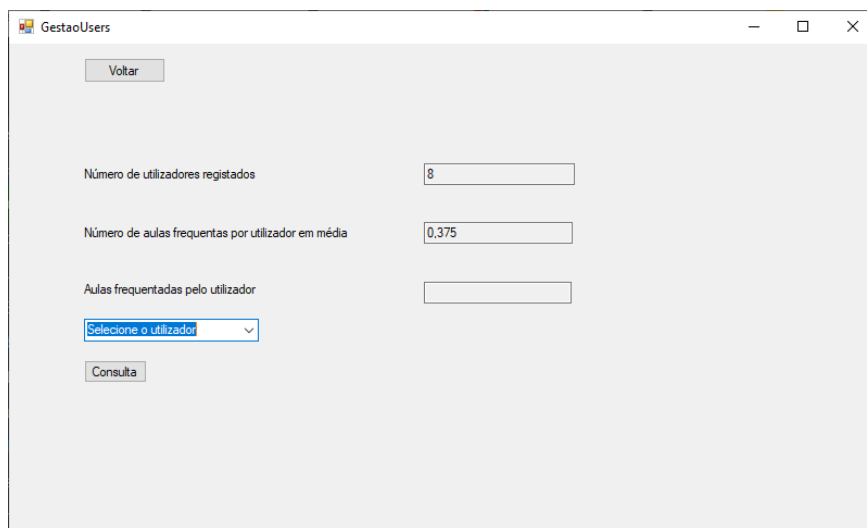
7.3.2 Aulas frequentadas por um utilizador específico

Uma vez que a nossa aplicação principal, a SportsManager, possui a funcionalidade de registo de Aulas e de inscrição nas mesmas, é interessante percebermos o comportamento individual de cada utilizador no que diz respeito à frequência das aulas. Assim, sempre que o administrador pretender, pode consultar o número de aulas que um utilizador específico já frequentou. A nossa Base de Dados possui todos os registos de quais utilizadores frequentaram quais aulas, pelo que apenas necessitamos de pesquisar quais as aulas que o utilizador em específico frequentou.

7.3.3 Aulas frequentadas pelos utilizadores em média

Na subsecção anterior referimos a importância de perceber o comportamento individual de cada utilizador no que diz respeito à frequência das Aulas. Contudo, do ponto de vista do negócio importa também perceber o comportamento geral de todos os utilizadores registados no sistema para percebermos, neste caso, se a adesão às Aulas é grande ou pequena. Este número médio é obtido pela divisão de todas as aulas existentes no sistema pelo número de utilizadores registados. Uma outra opção seria dividir o número de Aulas existentes pelo número de utilizadores distintos que já frequentaram pelo menos uma Aula. Contudo, optámos pela opção mencionada pois achamos que é uma métrica mais realista pois mostra-nos se temos muitos Utilizadores inativos ou não.

7.3.4 Apresentação da informação



The screenshot shows a web application window titled "GestaoUsers". It contains a form with the following elements:

- A "Voltar" button at the top left.
- A label "Número de utilizadores registados" followed by a text input field containing the value "8".
- A label "Número de aulas frequentas por utilizador em média" followed by a text input field containing the value "0,375".
- A label "Aulas frequentadas pelo utilizador" followed by an empty text input field.
- A dropdown menu with the placeholder text "Selecione o utilizador".
- A "Consulta" button at the bottom.

Figura 7.3: *Form Users*

7.4 Aulas

7.4.1 Total de aulas registadas no sistema

Similarmente aos Utilizadores, é sempre importante saber quantas Aulas já foram registadas no sistema. Para tal basta pedirmos à Base de Dados que nos informe sobre o número de códigos de Aula distintos existentes.

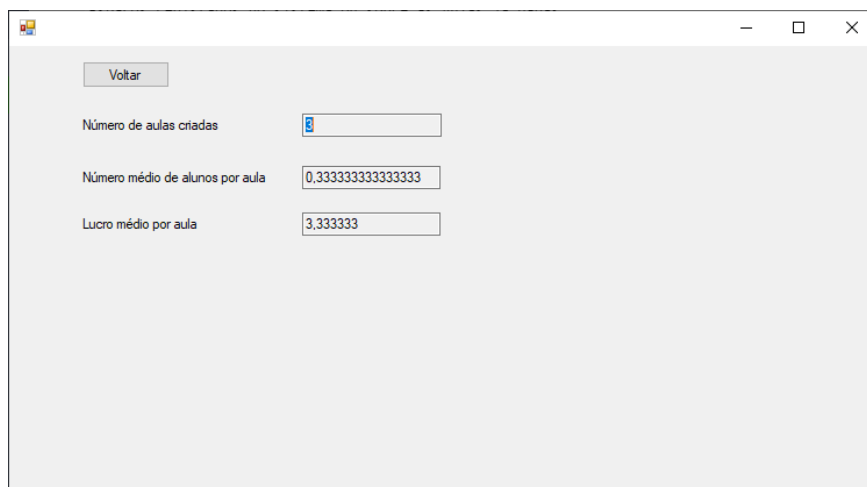
7.4.2 Número médio de alunos por Aula

Uma métrica interessante que importa medir é o número médio de alunos por Aula. É uma métrica um pouco simples, mas que reflete o "sucesso" ou "insucesso" geral das Aulas registadas na nossa plataforma.

7.4.3 Lucro médio por aula

Outra métrica que pode refletir o "sucesso" ou "insucesso" das aulas, mas neste caso monetário, é o lucro médio por Aula. Como sabemos, cada Aula é também caracterizada pelo número de bilhetes disponíveis e pelo preço individual de um bilhete. Para cada Aula um certo número de Utilizadores compram bilhete pelo preço estipulado. Se contabilizarmos todos os bilhetes vendidos de cada aula e o seu preço conseguimos obter um lucro médio por cada Aula.

7.4.4 Apresentação da informação



Label	Value
Número de aulas criadas	3
Número médio de alunos por aula	0.3333333333333333
Lucro médio por aula	3.333333

Figura 7.4: *Form Aulas*

7.5 Espaços

7.5.1 Número de Espaços registados no sistema

Para percebermos qual a adesão por parte das empresas à nossa plataforma, importa perceber quantos espaços no total é que estão registados no nosso sistema. Para tal, o programa acede à Base de Dados para contabilizar o número de registos únicos de Espaços existentes.

7.5.2 N° de alugueres de um Espaço

Uma estatística muito importante na análise do desempenho de um espaço é a de quantas vezes esse mesmo espaço já foi alugado. Uma variação interessante desta estatística que poderão ser implementadas futuramente são o número de alugueres por cada localidade existente ou por exemplo o número de alugueres por tipo de espaço (campo de futebol, sala de ginásio, entre outros).

Esta estatística obtém-se consultando o número de registos existentes na Base de Dados na tabela que guarda esta informação, para o espaço pretendido.

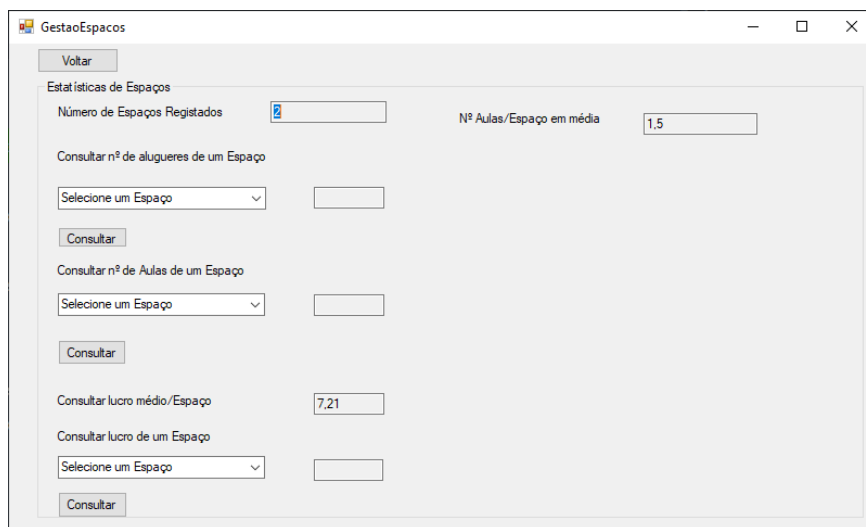
7.5.3 Lucro de um Espaço e lucro médio por Espaço

Numa perspetiva puramente de negócio, o que importa é a rentabilidade, ou seja, o que se fatura consoante o que disponibiliza aos consumidores. Deste modo, precisamos de ter informação sobre a rentabilidade média de todos os espaços bem como a rentabilidade individual de um espaço. Como já foi referido anteriormente, as aulas ocorrem em espaços, logo a rentabilidade de um espaço que permite aulas será a soma da rentabilidade das aulas com o valor total dos alugueres efetuados nesse espaço. De um modo geral, a rentabilidade média por espaço será a soma das rentabilidades individuais a dividir pelo total de espaços existentes.

7.5.4 Número de Aulas efetuadas num Espaço e número médio de Aulas efetuada por Espaço

Na subsecção anterior mencionámos a importância da estatística do lucro de um espaço e do lucro médio por espaço. Contudo, essa estatística não pode ser analisada isoladamente pois pode existir um outlier que simplesmente inflaciona o valor total do lucro. Deste modo, importa também perceber a utilização destes espaços por forma a percebermos se o valor do lucro foi um "acaso" ou resulta de uma utilização contínua do(s) espaço(s).

7.5.5 Apresentação da informação



The screenshot shows a window titled "GestaoEspacos" with a standard Windows interface (minimize, maximize, close buttons). Inside the window, there is a "Voltar" button at the top left. Below it, the section "Estatísticas de Espaços" contains two input fields: "Número de Espaços Registados" with the value "2" and "Nº Aulas/Espaço em média" with the value "1.5".

Below this, there are two sections for consulting data:

- Consultar nº de alugueres de um Espaço:** Includes a dropdown menu labeled "Selecione um Espaço", an empty input field, and a "Consultar" button.
- Consultar nº de Aulas de um Espaço:** Includes a dropdown menu labeled "Selecione um Espaço", an empty input field, and a "Consultar" button.

At the bottom, there are two more sections:

- Consultar lucro médio/Espaço:** Includes an input field with the value "7,21".
- Consultar lucro de um Espaço:** Includes a dropdown menu labeled "Selecione um Espaço", an empty input field, and a "Consultar" button.

Figura 7.5: *Form Espaços*

Capítulo 8

Rest Api

Neste capítulo iremos descrever a implementação da *Rest Api*, que com recurso à lógica do *BackEnd* fornece as funcionalidades necessárias para receber a informação.

O desenvolvimento deste segmento do projecto foi implementado com recurso a ferramentas *Microsoft*, em concreto com a utilização da *framework ASP.NET Web API's*, permite assim fornecer a informação necessária de uma forma mais genérica, mais precisamente *JSON*. Face à utilização da ferramenta *Microsoft* foi publicado no Azure o *layer*, é também usada a funcionalidade disponibilizada *CORS* para controlar a autorização aplicacional.

8.1 Autenticação

A autenticação é um ponto importante neste projeto, pois é necessário validar as permissões da conta que está a usufruir do serviço, deste modo é fornecido um *caching* de sessões dos utilizadores. Para satisfazer este requisito foi especificado que ao efetuar o respetivo *HTTP GET Login* é retornado dois valores ao utilizador que estão associados a sessão, assim permitindo que faça os pedidos de acordo com a sua paleta de permissões. Estes dois valores são posteriormente usados para todas as chamadas à *Web Api* de modo a autenticar a validade do pedido, também atualizada na *cache* o tempo de validade destas chaves para garantir segurança ao utilizador. Para eliminar as chaves do sistema de *caching* foi feito ao respetivo *HTTP GET Logout* que dado estas chaves apaga-as do sistema assim finalizando a sessão.

8.2 Instrutor

Com base na análise da informação nas paginas do instrutor foram criados 4 comportamentos *HTTP GET, PUT, POST e DELETE*. Em primeiro lugar temos o *GET*, que é utilizado para recolher a informação inicial da página, como por exemplo os dados da conta e também as aulas que pertencem a este utilizador. O *PUT, POST e DELETE* são as ferramentas que permitem interagir com as aulas, nas quais podemos adicionar uma nova aula, atualizar os dados da aula e também cancelar uma aula existente a acontecer no futuro, respetivamente.

Para melhor estruturação foi criado um controlador na *framework* que baseado no tipo de pedido feito executa o comportamento desejado, adiciona-se ainda um *model* aos controladores para implementar a partilha de única dessa classe pelos pedidos existentes.

8.3 Utilizador

A informação relativa ao utilizador subdivide-se em algumas paginas as quais aulas, espaços, estatísticas e definições, com esta dimensão de possibilidades optou-se por criar 4 controlados para receber os pedidos *HTTP*, respetivamente. A modulação deve-se também ao facto de existir diversos comportamentos diferentes que exige um pensamento na evolução da API.

8.3.1 Aulas

Para a sub-página das aulas temos como requisitos a possibilidade de compra e reembolso de bilhetes para aulas, a *Web Api* visa complementar o *BackEnd* ao autenticar o utilizador e posteriormente na lógica de atualização de dados. Com este objetivo são utilizados *HTTP GET*, *PUT* e *DELETE*, para retornar os dados iniciais, comprar um bilhete para uma determinada aula e reembolsar um bilhete previamente comprado.

8.3.2 Espaços

A página dos espaços é semelhante à das aulas, referente aos espaços disponíveis no site e também com uma função adicional, que permite observar a ocupação de espaço ao tentar aluga-lo, assim é possível verificar se este está disponível e a que horas. Face a alcançar estas funcionalidades são utilizados *HTTP GET*, *PUT* e *DELETE*, analogamente à as aulas. Para implementar a funcionalidade de ocupação é criado o caso de *HTTP POST*, que em primeiro lugar autentica o pedido e posteriormente recolhe as horas para qual esta o espaço ocupado no dado dia.

8.3.3 Definições

O serviço para as definições visa atualizar a informação do utilizar, assim é necessário passar como parâmetros os respetivos campos que podem ser alterados. É utilizado um *HTTP GET* com os respetivos dados, os quais podem ser nulos, que nesse caso não sofrem alterações, contrariamente os campos não nulos são posteriormente com recurso ao *BackEnd*.

Capítulo 9

FrontEnd

9.1 Web-API

Para maior modularidade do nosso projeto decidimos fazer o nosso FronEnd separado da *Web-API*. Para isso o nosso projeto tem uma classe *WebAPI.js* que trata das chamadas à WebAPI necessárias.

9.2 ReactJS

A construção do nosso frontend, como já tínhamos referido nos mockups, usamos a biblioteca de JavaScript React para, juntamente com diversas *frameworks* como *Material-UI*, *Material-Table*, *React-MDL*, entre outras, nos ajudar no desenvolvimento, facilitando a programação das respectivas funcionalidades.

9.3 Cliente vs Instrutor

A nossa aplicação prevê a existencia de dois tipos de utilizador o cliente e o instrutor. Como maneira de tornar a nossa aplicação mais minimalista temos apenas um local de login sendo a webapi, como já foi referido, que trata de fazer correspondencia e transmitir ao frontend que pagina vai aparecer.

9.4 Google Maps API

No nosso contexto de aluguer de espaços achamos por bem incluir aqui um mapa para melhorar a experiência de utilização. Para isso registamo-nos na *Google Cloud Platform* e pedimos uma *API key* para acesso a esta funcionalidade. Com isto um utilizador consegue saber com precisão onde se situa o espaço que irá alugar.

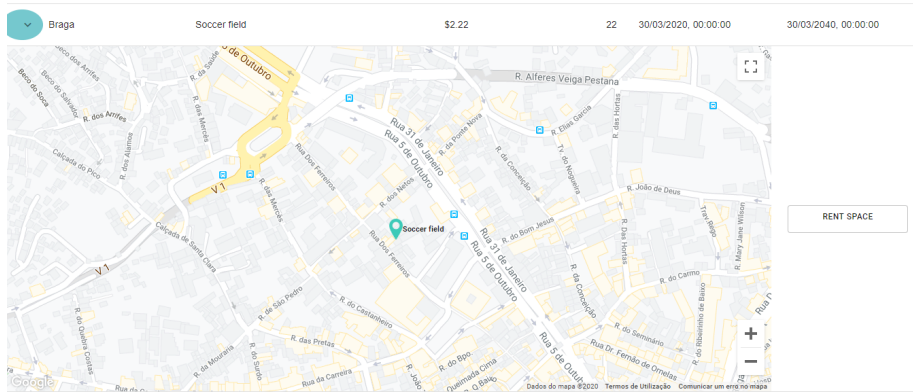


Figura 9.1: Google Maps API

9.5 Deployment

Para a implementação em cloud da nossa aplicação criamos um projeto "Web app" na Azure ficando com acesso ao endereço <https://sportsmanagerwebsite.azurewebsites.net> para a nossa aplicação. Para criar um ambiente automático de *release* usamos o Azure devops, criando um novo projeto e transferindo o nosso código para lá. Para automatizar o workflow criamos uma pipeline como a imagem seguinte.

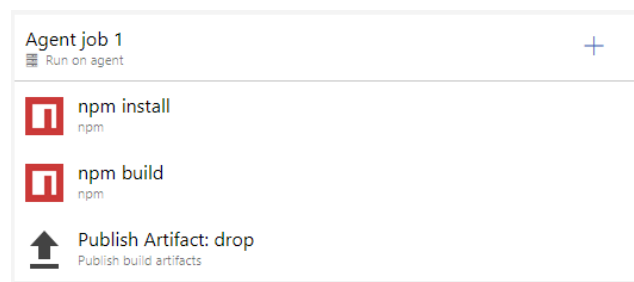


Figura 9.2: Pipeline FrontEnd

Criamos também uma *release pipeline*.

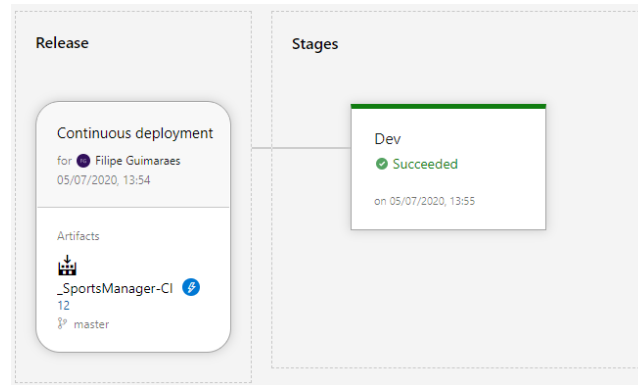


Figura 9.3: Release pipeline FrontEnd

Com estas duas *pipelines* conseguimos, de forma automática, dar deploy ao nosso frontend sempre que fazemos push ao nosso código.

Capítulo 10

Conclusão e Análise de Resultados

Colocando fim a esta fase de desenvolvimento da aplicação, ficamos satisfeitos com o resultado que obtivemos. Nesta fase temos já uma aplicação funcional e pronta para usar. Percebemos que haveria mais funcionalidades que podíamos implementar que poderiam ser feitas numa iteração seguinte a esta data. Com este projeto conseguimos ter uma vista de todas as fases no que toca à conceção de uma aplicação. Conhecemos ferramentas de cloud e api's temos à nossa disposição, bem como em que contexto as usar. Vimos ainda a facilidade e benefício em usar ferramentas baseadas em cloud como a *Azure* e a *Google Cloud* que tornam o nosso *workflow* mais dinâmico e produtivo, oferecendo diversas ferramentas.

A nossa aplicação pode ser consultada em <https://sportsmanagerwebsite.azurewebsites.net/>.