

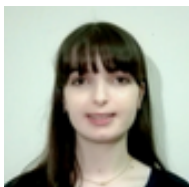


UNIVERSIDADE DO MINHO

DESENVOLVIMENTO DE SISTEMAS DE SOFTWARE LICENCIATURA EM ENGENHARIA INFORMÁTICA

ENTREGA INTERMÉDIA 3

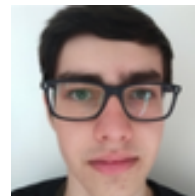
GRUPO 30



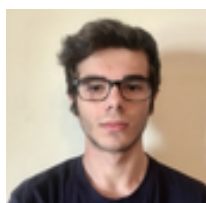
Ana Rita Poças
(a97284)



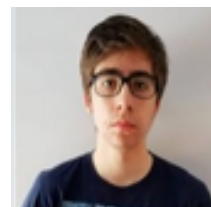
João Pedro Braga
(a97368)



Miguel Pinto
(a96106)



Orlando Palmeira
(a97755)



Pedro Miguel Martins
(a97613)

URL do repositório:

<https://github.com/orlandopalmeira/Trabalho-DSS-2022-2023>

1. Introdução	2
2. Alterações das Fases anteriores	3
3. Modelos Implementados	3
3.1. Diagrama de Instalação	3
4. Implementação do Sistema	4
4.1. Simular corridas	4
4.2. Configurar corridas	5
4.3. Configurar Campeonatos	6
4.4. Classificações	9
5. Base de Dados do Sistema	10
5.1. Tabelas do Programa	10
5.2. DAOs implementados no programa	12
6. Conclusão	15

1. Introdução

Este relatório surge no âmbito da Unidade Curricular de Desenvolvimento de Sistemas de Software, na qual nos foi proposto o desenvolvimento de um sistema que permita simular campeonatos de automobilismo, algo similar a um F1 Manager.

Nesta terceira fase, temos como objetivo fazer a modelação conceptual e a implementação da solução modelada nas fases anteriores. Desta forma, vamos elaborar os modelos necessários à descrição da implementação do sistema e, finalmente, a implementação do sistema. De forma a garantirmos a persistência de dados, procedemos à criação de uma base de dados que fosse capaz de armazenar as informações necessárias para o correto funcionamento do nosso programa. Assim, recorreremos à criação de DAOs que permitem estabelecer uma conexão com o *MySQL*. Assim sendo, os dados do programa ficam guardados na nossa base de dados.

2. Alterações das Fases anteriores

Ao longo da realização da fase 3, vimos que seria preciso fazer alguns ajustes ao planeamento que fizemos nas fases 1 e 2, por forma a tornar a implementação do sistema o mais completa e funcional possível. Mais concretamente, tivemos de fazer alterações nos Diagramas de Classes por forma a integrar a representação dos DAOs, esses diagramas irão ser demonstrados e explicados mais à frente.

3. Modelos Implementados

3.1. Diagrama de Instalação

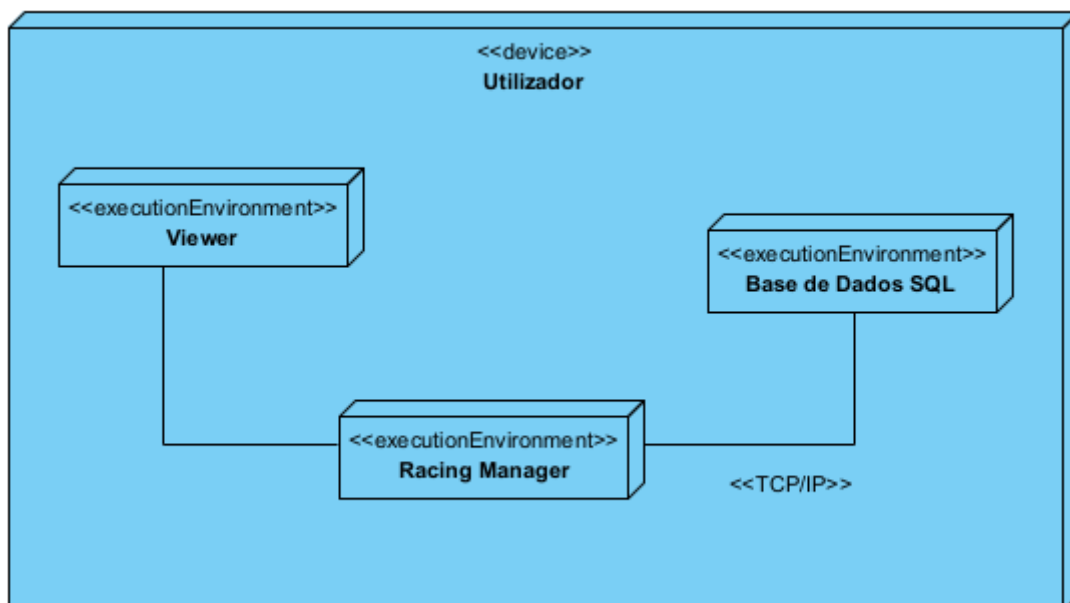


Figura 3.1.1: Diagrama de Instalação

Este diagrama de instalação apresenta a componente física do sistema implementado. O sistema executa na máquina do utilizador, onde são executados todos os componentes para o seu funcionamento: o simulador de corridas, que comunica com a base de dados via TCP e com o viewer (janela) para apresentação dos resultados.

4. Implementação do Sistema

A implementação do sistema começou com o foco no cenário 5 onde o sistema passaria a conseguir simular corridas, configurar corridas e configurar campeonatos.

4.1. Simular corridas

A implementação das simulações da corrida começou por analisar a estrutura planeada nas fases anteriores para obter uma ideia inicial de como deveríamos implementar. Ao analisar o diagrama de classes do componente SubCampeonatos percebemos que a simulação de uma corrida englobaria 5 classes do componente sendo elas o **SSCampeonatosFacade**, **Campeonato**, **Circuito**, **Setor** e **Jogador**.

A classe **SSCampeonatosFacade** que irá conter todos os campeonatos disponíveis para jogar e que permite a escolha ao utilizador. Depois da escolha do utilizador um objeto da classe **Campeonato** ficará disponível para ser jogado.

A classe **Campeonato** contém a lista dos jogadores que irão participar no campeonato e uma lista com os circuitos a serem jogados. A lista dos jogadores estará ordenada pela ordem que cada jogador está nas corridas, e essa lista irá ser passada como argumento para os métodos de simulação da corrida, onde será alterada conforme as ultrapassagens que aconteçam.

A classe **Circuito** ao receber a lista dos jogadores irá fazer a simulação da corrida volta a volta, onde a cada volta fazemos uma verificação se os jogadores têm um acidente ou avaria, com o método **dnf(volta)** da classe **Jogador** e no caso de o jogador parar na pista ele será retirado da lista dos jogadores ativos na corrida e só será colocado de novo no final para poder participar na próxima corrida. A simulação das ultrapassagens será feita em todos os **setores** onde os jogadores que ainda estejam a participar na corrida irão disputar por posições em cada **setor**.

A classe **Setor** estará responsável por receber os jogadores participantes na corrida e as informações da corrida como o **clima** e se a corrida já passou da metade ou não (**halfdistance**) por forma a decidir a performance dos pneus. As ultrapassagens são decididas segundo o tempo estimado que cada jogador demora a completar o setor, sendo esse tempo calculado pelo método **calcularTempo()** da classe **Jogador**. Para verificar se há ultrapassagem fazemos uma comparação entre os tempos dos jogadores. No caso do setor ter uma dificuldade **possível** a diferença entre o tempo de jogadores consecutivos precisa de ser maior do que 1 segundo, se a dificuldade for **difícil** o tempo precisa de ser maior do que 2 segundos e se a dificuldade for **impossível** a ultrapassagem não é efetuada.

A classe **Jogador** irá calcular o tempo estimado para o jogador completar um setor através do método **calcularTempo()** e dos seguintes fatores:

- Potência do motor ICE do carro (potência);
- Potência do motor elétrico do carro (potênciaH);
- Funcionamento do motor (conservador-0seg, normal-1seg ou agressivo-2seg);
- Pneu atual do carro (macio, duro ou chuva);
- Perfil aerodinâmico do carro (afeta a performance dependendo do tipo de setor);
- CTS do piloto (afeta a performance dependendo do clima da corrida)
- SVA do piloto;

Se o **tipo de setor** for “reta” os carros com um **pac** menor terão uma melhor performance já se for “curva” ou “chicane” os carros com **pac** maior terão vantagem.

Se o **clima** for de chuva os carros com pneus de chuva terão vantagem em comparação aos carros com pneus macios ou duros, já se for seco a performance dos pneus macios é melhor na primeira metade da corrida e pior na segunda metade em comparação aos pneus duros.

O tempo estimado é também afetado por um **fator_sorte** que é um valor gerado de forma aleatória entre 0 e 5 segundos.

Com todos esses valores conseguimos calcular a performance esperada de cada jogador.

```
||||| Posições na volta 1 |||||
1° -> Pedro
2° -> Miguel
3° -> João
4° -> Rita
5° -> Orlando
0 jogador Orlando parou na volta 1
0 jogador Pedro ultrapassou o jogador Rita no 1° setor!
0 jogador Pedro ultrapassou o jogador João no 2° setor!
0 jogador Pedro ultrapassou o jogador Miguel no 3° setor!
```

Figura 4.1.1: Posições durante a corrida e acontecimentos.

4.2. Configurar corridas

Com a implementação da simulação das corridas terminada, começamos a pensar nas alterações que os jogadores poderiam efetuar nos seus carros antes da corrida.

Os jogadores irão poder alterar o tipo de **pneu** do seu carro, o perfil aerodinâmico do seu carro (**PAC**) e o modo de funcionamento do seu motor (**funcionamentoMotor**).

Antes de cada corrida, o programa pergunta a cada jogador se pretende alterar o seu carro apresentando a informação do seu carro:

```
Jogador: Pedro
Carro: Ferrari-F8
      Pneu:Macio
      Pac:0.5
      MotorICE: Cilindrada:6000 | Potência:1000 | Funcionamento:Normal
      Classe:C1
Dejesa fazer alterações ao carro? S/N
```

Figura 4.2.1: “Deseja fazer alterações ao carro?”.

Se o jogador responder “S”, o programa entra no menu de modificação do carro:

```
0 que deseja alterar no seu carro?
1 -> Alterar os pneus
2 -> Alterar o perfil aerodinâmico do carro
3 -> Mudar o modo do motor
0 -> Sair
```

Figura 4.2.2: Menu de alteração do carro.

Neste menu o jogador poderá escolher o **pneu** que vai utilizar na corrida (macio, duro ou chuva), alterar o **PAC** se o seu carro pertencer à classe C2 ou C2H (valor entre 0 e 10 que depois é traduzido para um float entre 0 e 1) e mudar o modo de funcionamento do motor (conservador, normal ou agressivo).

Quando todos os jogadores terminarem as suas alterações o programa faz a simulação da corrida.

4.3. Configurar Campeonatos

Para terminar a implementação do cenário 5 só faltava a configuração dos campeonatos onde escolhemos o campeonato que queremos jogar e quais corridas irão ser jogadas nesse campeonato.

```
Qual campeonato pretente jogar?
0 -> Campeonato Teste
1 -> BragaCup
2 -> MinhoPrize
3 -> RallyMinho
4 -> MinhoRace
5 -> CompetitionBraga
6 -> CampeUMnato
```

Figura 4.3.1: Escolha do campeonato.

```
Qual circuito pretende adicionar ao Campeonato?  
1 -> Parque Rodovia  
2 -> Gualtar Campus  
3 -> Azurém Campus  
4 -> Avenida Central  
5 -> Sameiro - Bom Jesus  
6 -> Palmeira - Estação CF  
Escolha 0 se não pretender adicionar mais circuitos!
```

Figura 4.3.2: Escolha das corridas a serem jogadas.

Tanto os campeonatos como as corridas são retiradas da Base de Dados e apresentadas ao utilizador e vale a pena notar que uma corrida poderá ser jogada várias vezes no mesmo campeonato.

Com o campeonato escolhido passamos para a adição dos jogadores ao campeonato.

```
Insira o seu nome:  
miguel  
Dejesa fazer login em uma conta existente? S/N  
s  
Insira um username ou X para cancelar o login:  
miguel  
Insira a sua password:  
miguelgrupo30  
Bem Vindo miguel!
```

Figura 4.3.3: Escolha do nome do jogador e login.

O utilizador ao escolher o seu nome e fazer o login (opcional) é apresentado com todos os carros disponíveis na base de dados:


```

1 ---> Mazda-Miata
      Pneu:Macio
      Pac:0.5
      MotorICE: Cilindrada:2500 | Potência:150 | Funcionamento:Normal
      Classe:SC
2 ---> Toyota-GR86
      Pneu:Macio
      Pac:0.5
      MotorICE: Cilindrada:2500 | Potência:200 | Funcionamento:Normal
      Classe:SC
3 ---> Toyota-Supra
      Pneu:Macio
      Pac:0.5
      MotorICE: Cilindrada:3000 | Potência:400 | Funcionamento:Normal
      Classe:GT
4 ---> Chevy-Camaro
      Pneu:Macio
      Pac:0.5
      MotorICE: Cilindrada:2500 | Potência:350 | Funcionamento:Normal
      Classe:GT

```

Figura 4.3.4: Parte dos carros disponíveis para jogar.

Depois de escolher um carro o jogador deverá escolher um piloto:

```

Escolha um piloto para utilizar no campeonato:
1 ---> Piloto - Yuki - CTS:0.7 - SVA:0.2
2 ---> Piloto - Charles - CTS:0.4 - SVA:0.6
3 ---> Piloto - Kate - CTS:0.5 - SVA:0.7
4 ---> Piloto - Grace - CTS:0.6 - SVA:0.6
5 ---> Piloto - Max - CTS:0.4 - SVA:0.3
6 ---> Piloto - Lewis - CTS:0.8 - SVA:0.5
7 ---> Piloto - Lando - CTS:0.6 - SVA:0.4
8 ---> Piloto - Juliana - CTS:0.8 - SVA:0.5
9 ---> Piloto - Lucy - CTS:0.3 - SVA:0.2
10 ---> Piloto - Maria - CTS:0.6 - SVA:0.8

```

Figura 4.3.5: Pilotos disponíveis para jogar.

E com a escolha do carro e piloto o jogador fica registrado no campeonato e pronto para jogar.

O campeonato para poder ser jogado necessita de pelo menos 2 jogadores porém aceita vários jogadores se o utilizador assim pretender.

Quando o utilizador adicionar todos os jogadores que pretende ao campeonato, a simulação do campeonato começa.

4.4. Classificações

As classificações dos jogadores no Campeonato são mostradas no fim do mesmo, onde aparecem a soma das classificações das corridas para todos os jogadores.

```
Classificação dos carros ICE!  
1° Pedro : 30 pontos  
2° Miguel : 30 pontos  
3° João : 23 pontos  
4° Rita : 23 pontos  
5° Orlando : 23 pontos
```

Figura 4.4.1: Classificação do Campeonato.

Apenas é mostrado a classificação dos jogadores com Carros ICE porque não existiam jogadores com carros híbridos mas a sua classificação seria feita à parte.

No caso em que os jogadores estejam autenticados a sua posição final no campeonato irá ser adicionada à sua pontuação Geral na Base de Dados.

E o programa possui a opção de visualização do ranking global dos utilizadores:

```
||||| Classificação Global |||||  
1° miguel : 109 pontos  
2° pedro : 108 pontos  
3° orlando : 104 pontos  
4° joão : 103 pontos  
5° anarita : 97 pontos
```

Figura 4.4.2: Classificação Global.

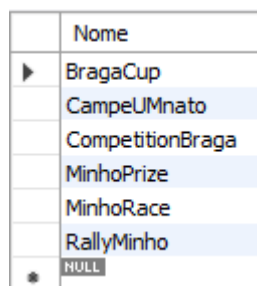
5. Base de Dados do Sistema

De forma a conseguirmos armazenar todos os dados relevantes e necessários ao funcionamento do nosso sistema, decidimos implementar uma base de dados, que fosse capaz de guardar toda a informação a fim de sermos, assim, capazes de garantir a persistência de dados no nosso sistema.

Como sistema de gestão de base de dados, elegemos o *MySQL*.

5.1. Tabelas do Programa

- **Campeonatos**

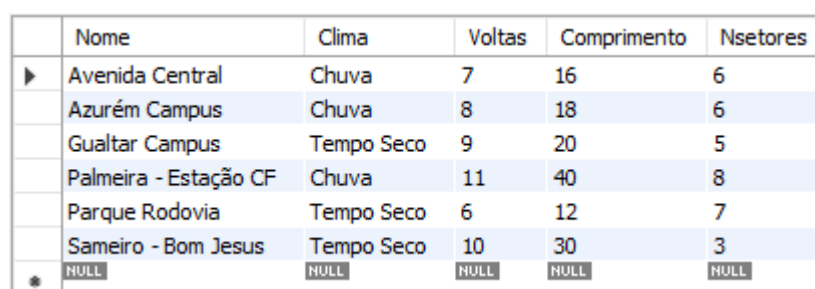


Nome
BragaCup
CampeUMnato
CompetitionBraga
MinhoPrize
MinhoRace
RallyMinho
NULL

Figura 5.1.1: Tabela `Simulação`.`Campeonatos`

Contém como **chave primária** o **nome** do campeonato. Não tem mais nenhum atributo. Permite-nos armazenar os dados de todos os campeonatos registados no programa.

- **Circuitos**



Nome	Clima	Voltas	Comprimento	Nsetores
Avenida Central	Chuva	7	16	6
Azurém Campus	Chuva	8	18	6
Gualtar Campus	Tempo Seco	9	20	5
Palmeira - Estação CF	Chuva	11	40	8
Parque Rodovia	Tempo Seco	6	12	7
Sameiro - Bom Jesus	Tempo Seco	10	30	3
NULL	NULL	NULL	NULL	NULL

Figura 5.1.2: Tabela `Simulação`.`Circuitos`

Contém como **chave primária** o **nome** do circuito. Contém os seguintes atributos: clima (a situação meteorológica do circuito - “Tempo Seco” ou “Chuva”), voltas (número de voltas associado ao circuito), comprimento (kms do circuito) e Nsetores (o número de setores do circuito, que pode ser uma curva, reta ou chicane). Permite-nos armazenar os dados relativos aos circuitos registados no nosso sistema.

- **Classe dos Carros**

	Id	minCilindrada	maxCilindrada
▶	C1	6000	6000
	C2	3000	5000
	GT	2000	4000
	SC	2500	2500
*	NULL	NULL	NULL

Figura 5.1.3: Tabela `Simulação`.`ClassesCarros`

Contém como **chave primária** o **Id** do da Classe, isto é a sua designação (C1-a Classe 1, C2- a Classe 2, GT- a classe Grande Turismo e SC- os Stock Cars.) Contém os seguintes atributos: minCilindrada (representa a cilindrada mínima associada a essa classe) e maxCilindrada (representa a cilindrada máxima associada a essa classe). Caso a Classe tivesse associada a uma única cilindrada e não um intervalo, como é o caso da C1 e SC, o valor da minCilindrada = maxCilindrada. Permite-nos armazenar os dados relativos às classes registadas no nosso sistema.

- **Carros**

	Id	Marca	Modelo	Classe	Cilindrada	Potência
▶	1	Mazda	Miata	SC	2500	150
	2	Toyota	GR86	SC	2500	200
	3	Toyota	Supra	GT	3000	400
	4	Chevy	Camaro	GT	2500	350
	5	Ford	Mustang	GT	4000	600
	6	Porsche	Cayman	C1	6000	800
	7	Porsche	911	C1	6000	900
	8	Ferrari	F8	C1	6000	1000
	9	Maserati	MC20	C2	4000	750
	10	Chevy	Corvette	C2	3000	650
*	NULL	NULL	NULL	NULL	NULL	NULL

Figura 5.1.4: Tabela `Simulação`.`Carros`

Contém como **chave primária** o **Id** do carro. Contém os seguintes atributos: Marca, Modelo, Classe, Cilindrada e Potência. Permite-nos armazenar os dados relativos aos carros registados no nosso sistema.

- **Pilotos**

	Nome	CTS	SVA
▶	Charles	0.40	0.60
	Grace	0.60	0.60
	Juliana	0.80	0.50
	Kate	0.50	0.70
	Lando	0.60	0.40
	Lewis	0.80	0.50
	Lucy	0.30	0.20
	Maria	0.60	0.80
	Max	0.40	0.30
	Yuki	0.70	0.20
*	NULL	NULL	NULL

Figura 5.1.5: Tabela `Simulação`.`Pilotos`

Contém como **chave primária** o **Nome** do piloto. Contém os seguintes atributos: CTS (nível de perícia do piloto no critério “Chuva vs. Tempo Seco” e SVA (nível de perícia do piloto no critério “Segurança vs. Agressividade). Permite-nos armazenar os dados relativos aos pilotos registados no nosso sistema.

- **Utilizadores**

	user	pass	pontuaçãoGeral	isAdmin
▶	anarita	ritagrupo30	97	1
	joão	joaogrupo30	103	1
	miguel	miguelgrupo30	109	1
	orlando	orlandogrupo30	104	1
	pedro	pedrogrupo30	108	1
*	NULL	NULL	NULL	NULL

Figura 5.1.6: Tabela `Simulação`.`Utilizadores`

Contém como **chave primária** o **user** do jogador. Contém os seguintes atributos: pass (a password definida pelo utilizador), pontuaçãoGeral (a pontuação geral do utilizador no sistema) e isAdmin (que apenas pode ter o valor 1, caso seja um administrador do sistema ou 0, caso contrário). Permite-nos armazenar os dados relativos aos utilizadores registados no nosso sistema.

5.2. DAOs implementados no programa

Para conseguirmos implementar a camada de persistência, procedemos à criação das seguintes classes DAO's:

- CampeonatoDAO;
- CarrosDAO;
- CircuitoDAO;
- ClassesCarrosDAO;
- PilotosDAO;
- UtilizadoresDAO;

Estas classes DAOs no nosso programa vão permitir que sejamos capazes de conseguir persistir objetos na base de dados criada, criar objetos a partir da informação que temos na nossa base de dados e encapsular queries SQL.

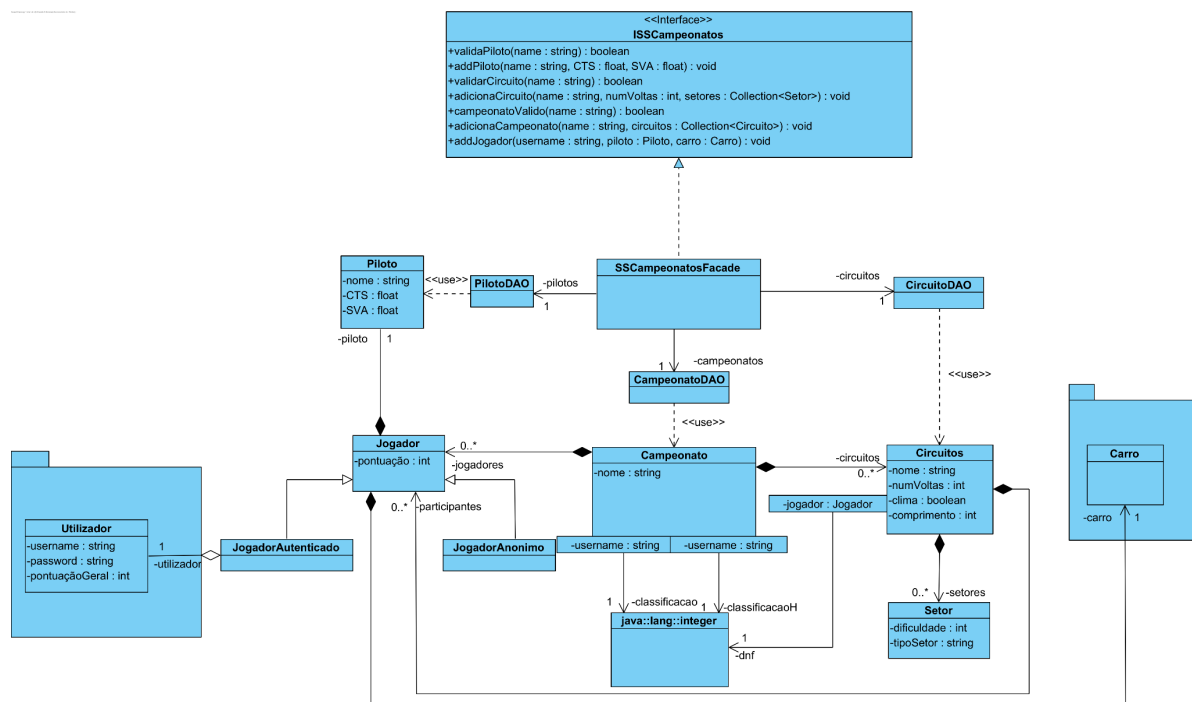


Figura 5.2.1: Diagrama de classes do SSCampeonatos com os DAO's

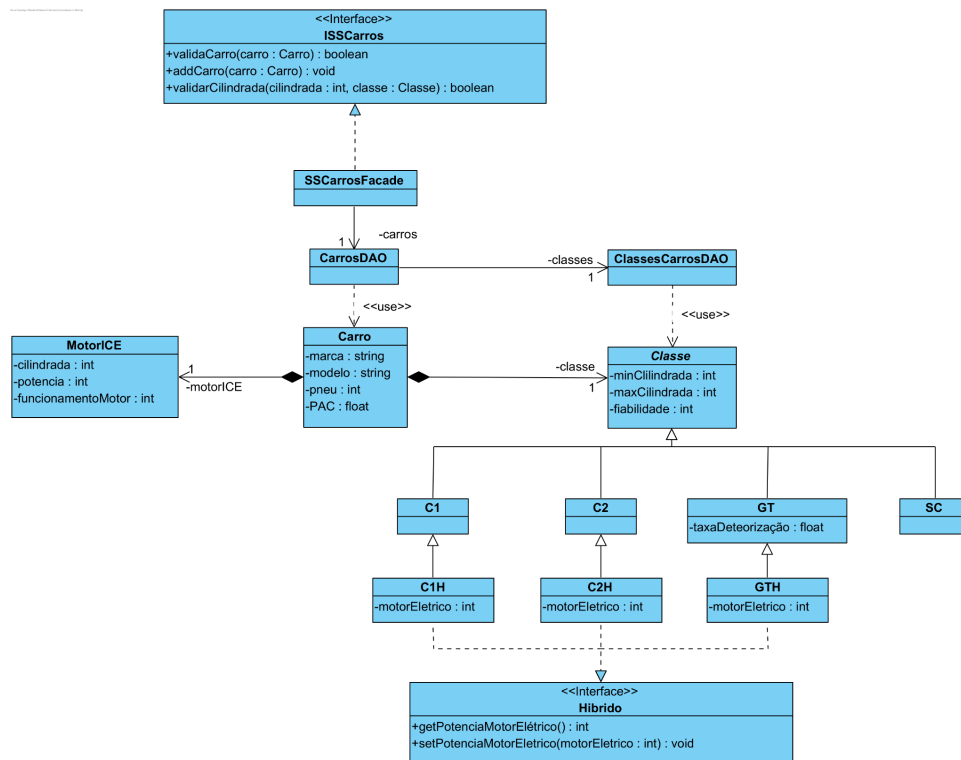


Figura 5.2.2: Diagrama de classes do SSCarros com os DAO's

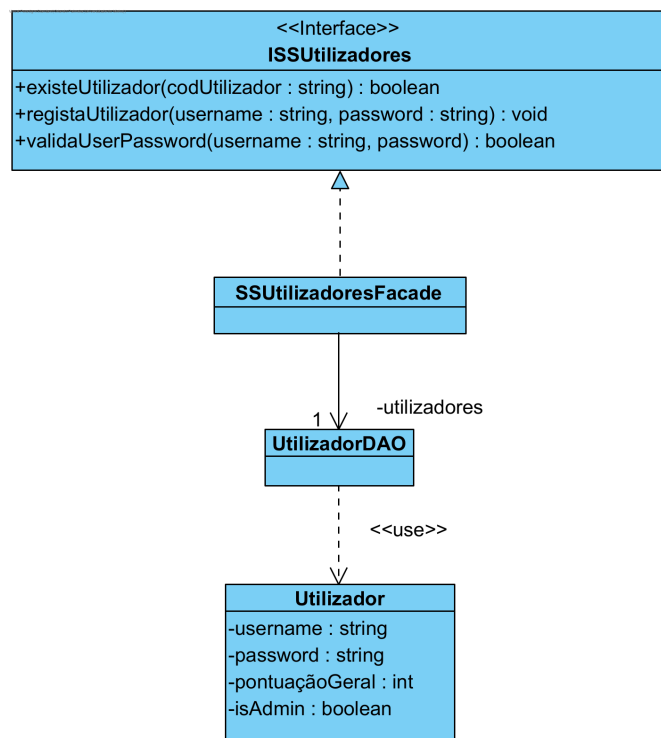


Figura 5.2.3: Diagrama de classes do SSUtilizadores com os DAO's

A inclusão dos DAO's no sistema implicou as alterações evidenciadas nas figuras 5.2.1, 5.2.2 e 5.2.3 relativamente à segunda fase.

Todos os DAO's implementam a interface Map para podermos abstrair as classes que os utilizam da sua verdadeira implementação (execução de *queries*

SQL na base de dados). Desta maneira, a utilização dos DAO's torna-se mais simples e assemelha-se à utilização de uma estrutura de dados comum em memória.

6. Conclusão

O facto de o projeto ter sido realizado por fases permitiu que conseguíssemos, de forma progressiva, desenvolver o sistema final, modificando certos aspetos sempre que considerarmos necessário.

No nosso ponto de vista, consideramos que fomos capazes de implementar os objetivos que nos foram propostos para esta fase, uma vez que completamos a implementação dos requisitos descritos no cenário 5 do enunciado e conseguimos criar a ligação entre o programa e uma Base de Dados através dos objetos DAO's.

Em suma, apesar de não termos conseguido implementar todos os requisitos presentes no enunciado, como a criação dos carros, pilotos, circuitos, etc., conseguimos fazer uma implementação do que consideramos mais importante e temos uma modelação e estrutura definida que no futuro permitiria uma implementação acessível do resto do projeto.