
Universidade do Minho
Departamento de Informática


Desenvolvimento de Sistema de software

# Simulador de corridas:


**Feito por:**
José Carvalho - A94913
Inês Castro - A95458
João Longo - A95536
Miguel Silva - A97031


GitHub: `https://github.com/JoseBambora/ProjetoDSS`
20 de novembro de 2022

# Conteúdo

# 1 Introdução

Este relatório foi redigido com o intuito de suportar o trabalho prático de Desenvolvimento de Sistemas de Software para o ano letivo 2022/2023. Para este trabalho foi-nos proposto que desenvolvêssemos um sistema de simulação de campeonatos de automobilismo, que visa aplicar os conhecimentos relativos a todas as etapas na realização de um projeto de software.

De modo a facilitar a realização do projeto, este foi divido em três fases, sendo que destas achamos que as 2 primeiras fases de criação são cruciais para o projeto, pois representam boa parte do planeamento do trabalho, facilitando assim a resolução de muitos problemas da 3ª fase.

# 2 Objetivo Global

O simulador desenvolvido suporta vários jogadores ao mesmo tempo, sendo que as decisões tomadas só são possíveis de se realizar em 2/3 das provas e antes do início das mesmas. As decisões afetam diretamente as prestações dos pilotos e veículos selecionados nas provas que posteriormente afetará a classificação final do campeonato e das corridas.

Apenas os administradores são capazes de criar um campeonato, mas apenas jogadores são capazes de o iniciar. Cada campeonato tem um determinado número de circuitos, sendo que cada circuito tem um número de voltas bem definido na sua fase de criação.

Depois desta fase de escolha do campeonato, todos os jogadores serão capazes de escolher o carro e o piloto que preferem, não podendo estes ser alterados durante o campeonato. No final de cada prova, a classificação geral dos jogadores do campeonato sofre alterações, de acordo com as posições dos pilotos que vão sendo registadas ao longo do mesmo.

Após o fim do campeonato o nosso simulador atualiza a pontuação total de cada jogador, pontuações essas que aumentam após a soma dos pontos obtidos com base nas posições alcançadas nas provas simuladas, mas para acontecer esse registo é necessário o login dos jogadores na aplicação, sem esta verificação as pontuações não são atualizadas.

# 3 Primeira Fase

## 3.1 Objetivos

Esta primeira fase consiste em aplicarmos certos conceitos de forma a obtermos uma visão global da nossa solução.

Foi-nos proposto que o desenvolvimento de um modelo de domínio com as entidades relevantes e um modelo de 'Use Cases'. Neste capítulo iremos apresentar e explicar os modelos que construímos com base nos requisitos que levantamos a partir do enunciado.

## 3.2 Entidades relevantes

A nosso ver, o primeiro desafio do projeto consistia em analisar o enunciado ao detalhe e identificar as entidades, que compõem o projeto que iremos realizar bem como as relações entre elas. As entidades principais que conseguimos descortinar imediatamente foram as seguintes:

- Jogadores;
- Administradores;
- Campeonatos;
- Pistas;
- Carros;
- Pilotos.

Classificamos estas entidades como principais pois é a partir delas que se ligam as várias partes do sistema completo. Ou seja, o nosso raciocínio para este projeto passou por relacionar estas entidades de forma coerente entre si. Também é a partir de algumas destas entidades que nascerão outras entidades que implementaremos, tal como possíveis interfaces, coleções, e *Database Access Objects*.

Nos próximos parágrafos desta secção de relatório iremos explicitar o processo que nos levou a criar mais entidades associadas às entidades principais que já listamos.

Relativamente às pistas, lendo o enunciado, conseguimos encontrar referências a parâmetros relevantes que posteriormente se traduzem em novas entidades, nomeadamente:

- Detalhes da pista;
- Clima (pode ser chuvoso ou seco).
- Comprimento da pista

Relativamente aos detalhes da pista, sabemos que numa prova existe um determinado número de curvas, retas e chicanes. Estes "fragmentos" que juntos compõem a pista têm em comum um fator descritivo chamado: grau de dificuldade de ultrapassagem (GDU). De modo a codificar este parâmetro optamos por criar uma entidade genérica, na qual que guardamos o GDU e estabelecemos a relação que enuncia que "cada um dos elementos referidos representam/ são uma caraterística do circuito".

Quanto ao clima, inicialmente ponderamos que poderia ser exclusivamente um atributo da pista, mas como este fator é também relevante para outra entidade (piloto - CTS) e para a posterior implementação em código na qual tencionamos dar a escolha do tipo de pneus a usar ao jogador, decidimos tanto pela relação entre o clima e o piloto como pela compreensão, traduzir a informação numa entidade chamada Clima.

Para fins estatísticos e tornando o nosso simulador mais semelhante ao mundo real, achamos por bem guardarmos as melhores voltas de cada pista. Para isso criamos uma entidade chamada "melhor volta", que estabelece uma relação de um para um em relação ao piloto e ao circuito.

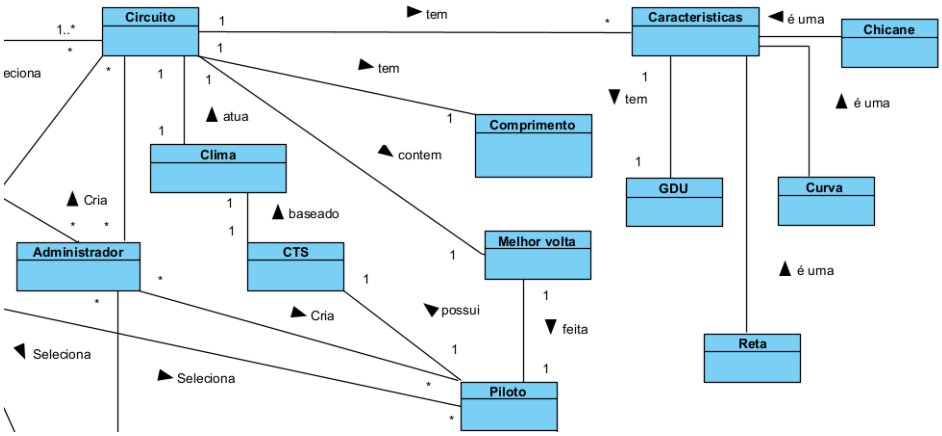


Figura 1: Parte do modelo de domínio referente às pistas

A figura [**1**] representa o modelo de domínio referente às pistas.

Tão importante como as pistas, temos os carros. No que diz respeito aos carros, concluímos que podemos traduzir os seguintes componentes e fatores a ponderar em entidades:

- Pneus
- Perfil Aerodinâmico do Carro (PAC)
- Motor
- Categoria

Apostamos nestas entidades pois todas são relevantes para o desempenho dos carros a competir nos campeonatos.

Relativamente aos pneus, um carro tem associado a si um conjunto de quatro pneus, sendo que um conjunto de pneus está associado a um carro. Um pneu pode ser de três tipos, macio, duro ou chuva. Decidimos trabalhar com o conjunto de pneus em vez de com quatro entidades do tipo pneu porque ao fazer isso reduzimos uma "etapa" na nossa resolução assumindo que todos os pneus que um carro tem equipados são do mesmo tipo. Conjunto de Pneus também está associado a uma capacidade. Este valor de capacidade irá representar a percentagem de pneu não gasto, logo este valor irá ser reduzido ao longo das corridas.

O PAC descreve a forma como o carro lida com fatores como o atrito e resistência do ar, como explicado no enunciado, é considerado como um valor numérico, sendo que se este valor for mais baixo, o carro atinge uma velocidade maior em reta, mas se este valor for menor o carro torna-se mais estável nas curvas. A relação entre carro e PAC é 1 para 1, ou seja, a um carro associa-se um perfil aerodinâmico e um perfil aerodinâmico é relativo a um carro.

Em relação ao motor, um carro tem de ter um motor a combustão e em algumas categorias poderá ter um motor elétrico. Ambos os motores são um

motor. Esta entidade "motor" contém a capacidade do combustível, um modo de corrida, a potência e no caso de ser a combustão, a cilindrada.

Nós pretendemos acrescentar a quantidade e gestão do combustível dos carros à simulação, relatando as alterações do mesmo no decorrer da corrida.

O modo de um motor representa o nível de potência que os motores dispõem para uma determinada corrida, podendo ser de três tipos: agressivo, normal e conservador. No caso de ser agressivo, o motor usa a sua potência máxima, mas por sua vez consome mais combustível e aumenta o risco de colisão ou despiste. O modo conservador é o contrário do agressivo (menos potência, menos consumo e diminui os riscos de colisão) e o normal é um equilíbrio entre os dois modos. A relação do modo com os motores é de um para um.

Existem dois motores no nosso modelo, o elétrico e o de combustão. Ambos tem as mesmas características no nosso modelo, diferenciando apenas que o motor a combustão tem uma componente de cilindrada e o elétrico não.

A cilindrada de um motor a combustão relaciona-se com as capacidades motoras do carro.

Para diferenciar as diferentes categorias entre os diversos carros, criamos uma entidade para cada categoria (C1, C2, GT, SC) sendo que todas são entidades relacionadas com uma entidade "categoria". Cada categoria tem características individuais como a cilindrada e a fiabilidade que estão definidas num determinado intervalo. A fiabilidade está relacionada com a probabilidade de realizar um evento num determinado circuito em segurança e esta depende do motor do carro e do próprio piloto.

A fiabilidade é calculada com base na equação: fiabilidade = (fiabilidade do motor) * (percentagem carro) + (fiabilidade do piloto) * (percentagem piloto). Esta percentagem carro e percentagem piloto dependem da categoria do veículo. No caso da categoria C1, estes valores são 95% e 5% respetivamente, para C2 é 80% e 20%, para a classe GT irá variar ao decorrer da prova, inicialmente será 90%, mas vai diminuir ao longo do decorrer da prova e nos SC os valores serão 25% e 75% respetivamente. A fiabilidade do motor já é definida quando o carro é criado e a fiabilidade do piloto irá depender da decisão do mesmo em cada evento, isto é, se ele tomar uma decisão agressiva a fiabilidade irá ser menor do que se tomasse uma decisão passiva.

Na figura [**2**] encontra-se a parte do modelo de domínio referente aos carros.

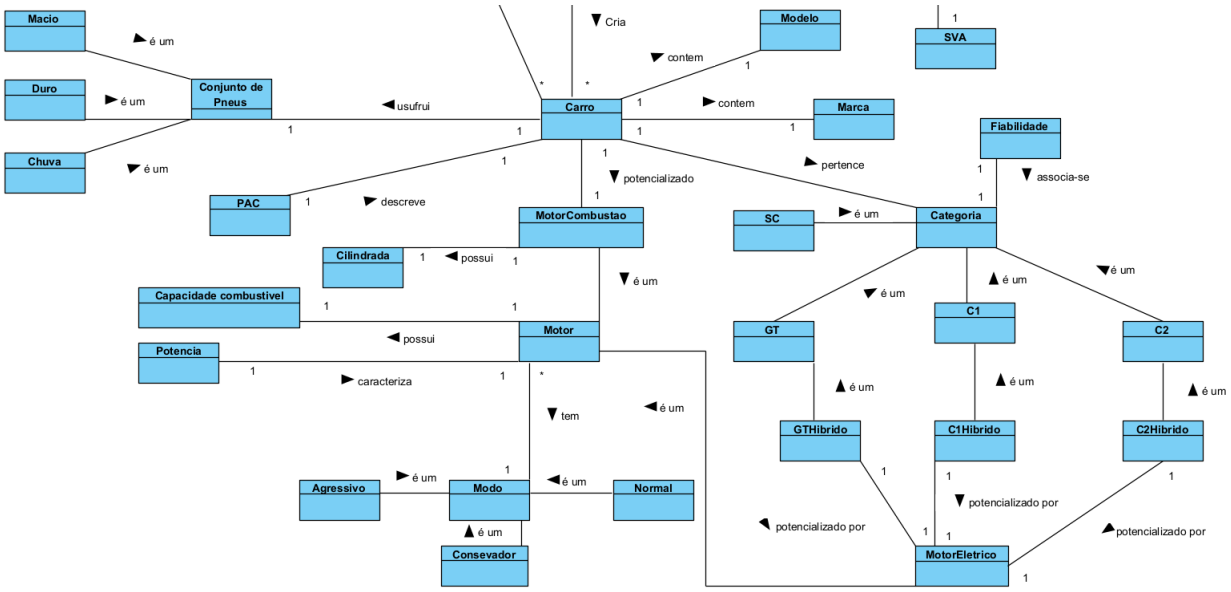


Figura 2: Parte do modelo de domínio referente aos carros

Por último, mas não menos importante, temos os pilotos. Relativamente a esta entidade, temos os seguintes fatores que a afetam:

- Chuva vs. Tempo Seco (CTS)
- Segurança vs. Agressividade (SVA)

Tal como mencionado anteriormente em relação às pistas, a entidade Clima (CTS neste caso) irá ter um impacto no desempenho dos pilotos, sendo que uma maior probabilidade (>0,5) irá traduzir-se numa melhor performance em tempo seco, havendo mais perigo para despistes em circuitos cujo clima se encontra molhado. Por sua vez, Segurança vs. Agressividade (SVA), compreende a pré-disposição que o piloto tem para tomar decisões mais arriscadas, se ultrapassa ou não. Posto isto, para um maior valor (>0,5) de SVA, o piloto possuirá uma atitude mais agressiva, sendo que terá mais probabilidade de ultrapassar e consequentemente uma maior probabilidade de se despistar.

Na figura [**3**] é possível observar o modelo de domínio relativa aos pilotos.

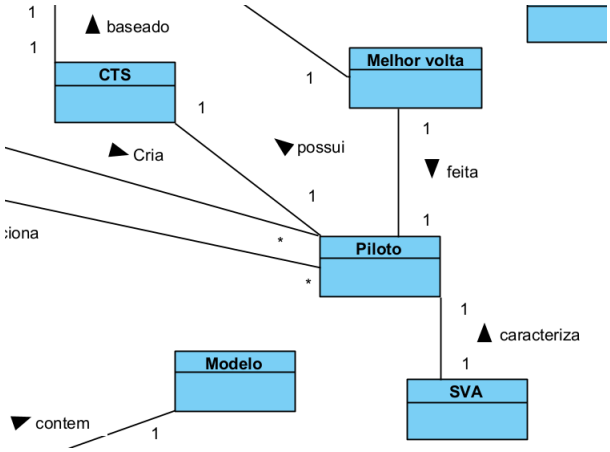


Figura 3: Parte do modelo de domínio referente aos pilotos

Decidimos também que, de modo a armazenar as pontuações obtidas pelos jogadores ao longo das partidas realizadas pudessem ser armazenadas e comparadas, seria boa ideia transformar a classificação em entidades às quais associamos para cada jogador a soma das pontuações que o mesmo pretende registar.

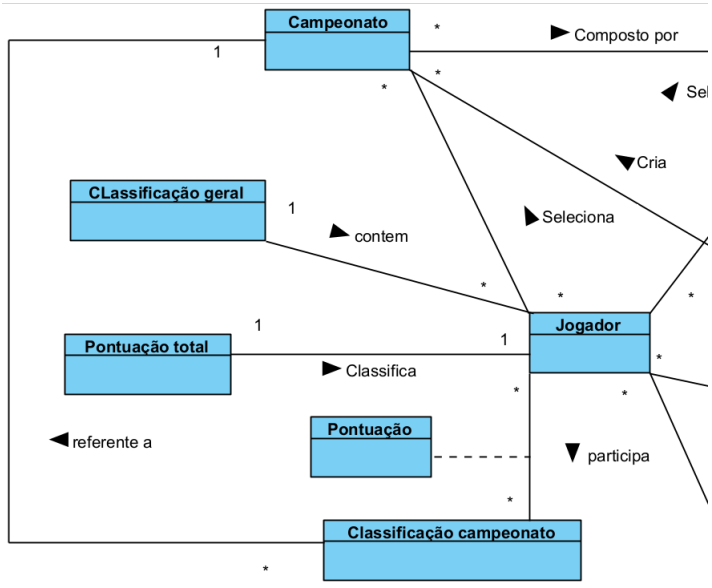


Figura 4: Parte do modelo de domínio referente aos jogadores

## 3.3 Modelo de Domínio

A figura [**5**] representa o nosso modelo de domínio completo, este modelo foi construído com base no conteúdo da secção anterior deste relatório.

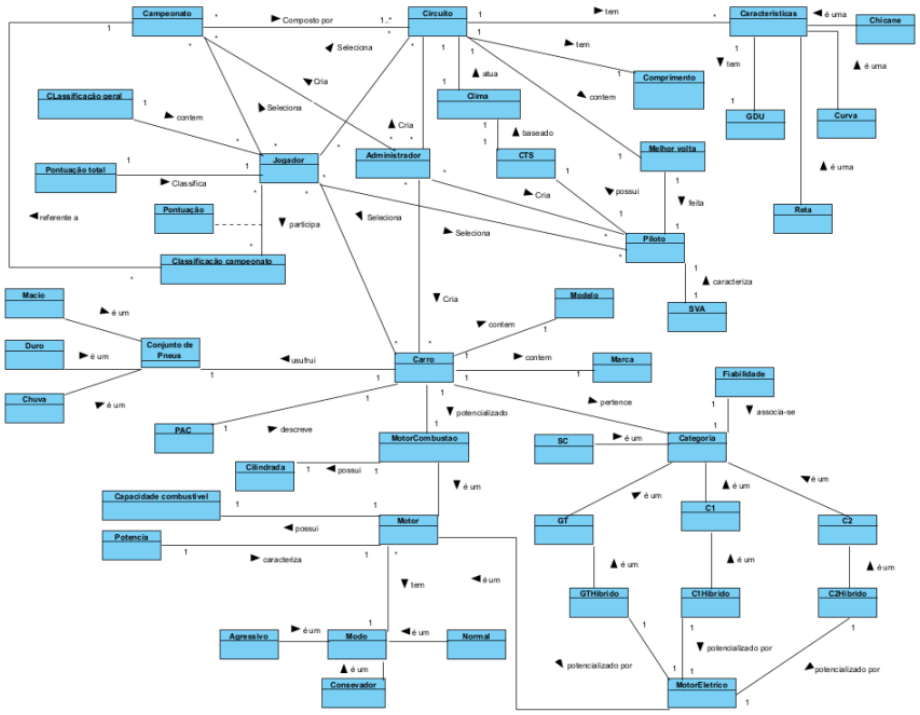


Figura 5: Modelo de Domínio completo

## 3.4 Modelo de Use Case

Para este projeto levantamos os seguintes use cases:

- Fazer o login
- Criar campeonato / carro / piloto / circuito / utilizador
- Aderir a um campeonato
- Configurar campeonato
- Afinar carros
- Simular corridas
- Simular campeonato
- Adicionar pontuações após um campeonato
- Consultar classificação
- Consultar melhores voltas

Os atores usados para os casos de uso são:

- Administrador
- Jogador

Na tabela [**1**] conseguimos observar quais os use cases que cada ator consegue fazer.

| Ator | Use Cases |
|---|---|
| Administrador(a) | Criar campeonato<br>Criar carro<br>Criar piloto<br>Criar circuito<br>Criar jogador ou administrador<br>Fazer Login |
| Jogador(a) | Aderir a um campeonato<br>Configurar campeonato<br>Configurar corridas<br>Fazer Login<br>Simular corridas<br>Simular campeonato<br>Consultar classificação<br>Consultar melhores voltas |

Tabela 1: Tabela dos use cases

Os use cases levantados são visíveis no modelo que consta na figura[**6**] e especificados nas subsecções seguintes.

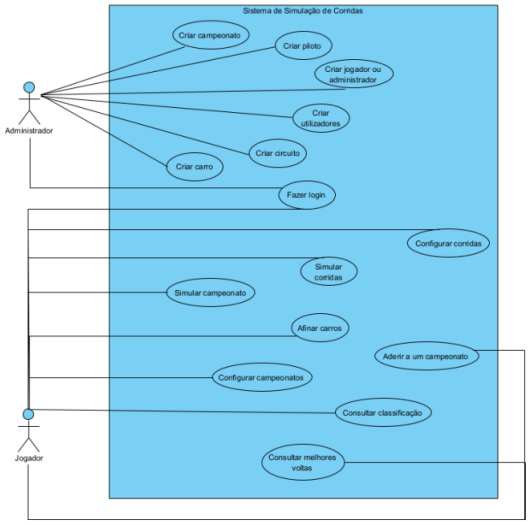


Figura 6: Modelo de use cases

### 3.4.1 Fazer Login

**Descrição**: Utilizador faz login.
**Cenários**: O Miguel entra no sistema e acede à sua conta metendo um nome de utilizador e a sua palavra-passe.
**Pré-Condição**: O sistema tem utilizadores.
**Pós-Condição**: Sistema avança para o menu.
**Fluxo Normal**:

1. Sistema pede os dados de acesso
2. Ator indica os seus dados
3. Sistema valida os dados
4. Sistema apresenta o menu principal

**Fluxo Alternativo** (1): [Dados de acesso incorretos] (passo 3)

3.1 Sistema informa que os dados de autenticação estão incorretos
3.2 Sistema repete o processo de autenticação

### 3.4.2 Criar campeonato

**Descrição**: Administrador cria um campeonato.
**Cenários**: Cenário 1 do enunciado.
**Pré-Condição**: O sistema ser capaz de armazenar a entidade e o administrador estar autenticado.
**Pós-Condição**: A informação relativa ao campeonato está armazenada na base de dados.
**Fluxo Normal**:

1. Administrador seleciona a opção de criar entidade no menu principal
2. Administrador escolhe criar campeonato
3. Administrador insere o nome que pretende dar ao campeonato
4. Administrador seleciona os circuitos que constituem o campeonato
5. Sistema valida os dados
6. Sistema adiciona o campeonato à base de dados
7. Sistema questiona se o campeonato está pronto a ser jogado
8. Administrador responde que não
9. Sai do processo.

**Fluxo Exceção** (1): [circuito não existe] (passo 4)

4.1 Sistema informa que o circuito selecionado não existe na sua base de dados.
4.2 Sai do processo

**Fluxo Exceção** (2): [Não existe nome do campeonato ou não houve seleção de pistas] (passo 6)

5.1 Sistema informa que não existe nome do campeonato ou não houve seleção de pistas.
5.2 Sai do processo

**Fluxo Exceção** (3): [Já existe um campeonato com o mesmo nome] (passo 6)

5.1 Sistema informa que já existe um campeonato com o mesmo nome
5.2 Sai do processo

**Fluxo Alternativo**s (4): [Administrador quer campeonato disponível] (passo 7)

7.1 Sistema altera informação do campeonato, de forma a estar disponível.
7.2 Passo 9.

#### 3.4.3 Criar circuito

**Descrição**: Administrador cria um circuito.
**Cenários**: Cenário 2 do enunciado.
**Pré-Condição**: O sistema ser capaz de armazenar o circuito e administrador está autenticado.
**Pós-Condição**: O circuito consta na base de dados.
**Fluxo Normal**:

1. Administrador seleciona a opção de criar entidade do menu principal.
2. Administrador escolhe criar circuito
3. Administrador insere um nome para o circuito
4. Administrador insere a distância total do circuito
5. Administrador insere o número de curvas.
6. Administrador insere o número de chicanes.
7. Sistema valida o número de curvas e de chicanes.
8. Sistema calcula o número de retas.
9. Administrador seleciona o GDU para cada curva, chicane e reta.
10. Administrador insere o número de voltas ao circuito.
11. Sistema valida os dados.
12. Sistema adiciona o circuito à base de dados.
13. Sai do processo

**Fluxo Exceção** (1): [Nome do circuito já exista no simulador] (passo 10)

7.1 Sistema informa que nome do circuito já existe
7.2 Sai do processo

**Fluxo Exceção** (2): [Distância inserida igual a 0] (passo 10)

10.1 Sistema informa que distância não pode ser 0
10.2 Sai do processo

**Fluxo Exceção** (3): [Número de curvas menor que 2] (passo 7)

7.1 Sistema informa que o circuito tem que ter pelo menos 2 curvas.
7.2 Sai do processo

**Fluxo Exceção** (4): [Número de chicanes negativo] (passo 7)

7.1 Sistema informa que o número de chicanes tem que ser igual ou maior que 0.

7.2 Sai do processo

**Fluxo Exceção** (5): [GDU inválido] (passo 10)

10.1 Sistema informa que o GDU de alguma curva é inválido.
10.2 Sai do processo

**Fluxo Exceção** (6): [Número de voltas igual menor ou igual a 0] (passo 10)

10.1 Sistema informa que número de voltas tem que ser positivo.
10.2 Sai do processo

#### 3.4.4 Criar piloto

**Descrição**: Administrador cria um piloto.
**Cenários**: Cenário 3 do enunciado.
**Pré-Condição**: O sistema ser capaz de armazenar o piloto e o administrador está autenticado.
**Pós-Condição**: O piloto consta na base de dados.
**Fluxo Normal**:

1. Administrador seleciona a opção de criar entidades do menu principal.
2. Administrador escolhe criar piloto.
3. Administrador insere o nome do piloto.
4. Administrador insere um valor de CTS ("Chuva vs. Tempo Seco").
5. Administrador insere um valor para SVA ("Segurança vs. Agressividade").
6. Sistema valida os dados
7. Sistema adiciona o piloto à base de dados
8. Sai do processo

**Fluxo Exceção** (1): [Nome do piloto já existente no sistema] (passo 6)

6.1 Sistema informa que nome do piloto já existe no sistema
6.2 Sai do processo

**Fluxo Exceção** (2): [Valor do CTS menor que 0 ou maior que 1] (passo 6)

6.1 Sistema informa valor do CTS tem que ser entre 0 e 1
6.2 Sai do processo

**Fluxo Exceção** (3): [Valor do SVA menor que 0 ou maior que 1] (passo 6)

6.1 Sistema informa valor do SVA tem que ser entre 0 e 1
6.2 Sai do processo

### 3.4.5 Criar carro

**Descrição**: Administrador cria um carro.
**Cenários**: Cenário 4 do enunciado.
**Pré-Condição**: O sistema ser capaz de armazenar a informação do carro na sua base de dados e administrador autenticou-se.
**Pós-Condição**: O carro consta na base de dados
**Fluxo Normal**:

1. Administrador seleciona a opção de criar entidades do menu principal.
2. Administrador escolhe criar carro.
3. Sistema questiona qual a marca e modelo do carro
4. Sistema questiona qual a categoria do veículo
5. Sistema questiona se o carro é híbrido
6. Administrador indica que não
7. Sistema pede o valor da cilindrada do motor a combustão
8. Sistema pede a potência do motor a combustão
9. Sistema pede o valor do PAC
10. Sistema valida os dados
11. Sistema define modo do motor normal
12. Sistema define a fiabilidade do carro com base na categoria
13. Sistema define a capacidade de combustível a 100
14. Adiciona o carro à base de dados
15. Volta para o menu

**Fluxo Alternativo** (1): [Carro é de categoria SC] (passo 4)

4.1 Avança para 7

**Fluxo Alternativo** (2): [Carro é Hybrid] (passo 5)

5.1 Sistema pede potência do motor elétrico
5.2 Avança para 7

**Fluxo Exceção** (3): [Carro com a mesma marca e o mesmo modelo já existe na base de dados] (passo 10)

10.1 Sistema informa que carro com a mesma marca e o mesmo modelo já existe na base de dados
10.2 Volta ao menu principal

**Fluxo Exceção** (4): [Valor da cilindrada não é compatível com a categoria do carro] (passo 10)

10.1 Sistema informa que valor da cilindrada não é compatível com a categoria do carro
10.2 Volta ao menu principal

**Fluxo Exceção** (5): [Valor da potência de um motor menor ou igual a 0] (passo 10)

10.1 Sistema informa que valor da potência tem que ser maior que 0
10.2 Volta ao menu principal

**Fluxo Exceção** (6): [Valor do PAC menor ou que 0 ou maior que 1] (passo 10)

10.1 Sistema informa que o valor do PAC varia entre 0 e 1
10.2 Volta ao menu principal

### 3.4.6 Criar Utilizador

**Descrição**: Administrador cria um utilizador.
**Cenários**: O José quer criar uma conta utilizador onde insere o nome de utilizador "FastestBug" e introduz a sua palavra-passe "core123".
**Pré-Condição**: O sistema ser capaz de armazenar entidades e o administrador está autenticado.
**Pós-Condição**: As entidades constam na base de dados.
**Fluxo Normal**:

1. Administrador seleciona a opção de criar entidade do menu principal.
2. Administrador seleciona Registar Utilizador.
3. Administrador insere o nome de utilizador.
4. Administrador insere a palavra-passe.
5. Administrador indica se é um jogador ou um administrador.
6. Sistema valida os dados
7. Sistema adiciona a devida entidade à base de dados
8. Sai do processo

**Fluxo Exceção** (1): [Nome de utilizador já existe na base de dados] (passo 6)

6.1 Sistema informa campo inválidos ou que o utilizador já existe
6.2 Sai do processo sem guardar os dados.

### 3.4.7 Aderir a um campeonato

**Descrição**: Um jogador escolhe um carro e um piloto para aderir a um campeonato.
**Cenários**: Cenário 5 do enunciado - Configurar um campeonato.
**Pré-Condição**: O campeonato para qual o participante vai aderir já foi iniciado.
**Pós-Condição**: O participante escolheu um piloto diferente e um carro em relação aos outros participantes do mesmo campeonato.
**Fluxo Normal**:

1. Sistema apresenta o campeonato iniciado
2. Jogador insere o seu nome de utilizador
3. Jogador escolhe um piloto
4. Jogador escolhe um carro
5. Jogador escolhe os pneus do carro
6. Sistema valida os dados
7. Sistema guarda a escolhas.

**Fluxo Exceção** (1): [O piloto selecionado ou o carro escolhido ou os pneus escolhidos não existem no sistema] (passo 6)

6.1 Sistema informa que o piloto ou o carro ou os pneus não existem no sistema.
6.2 Sai do processo sem guardar os dados.

**Fluxo Alternativo**(2); [O piloto selecionado já foi escolhido por outro participante do campeonato] (passo 6)

6.1 Sistema informa que o piloto já foi escolhido por outro jogador para o mesmo campeonato.
6.2 Sai do processo sem guardar os dados.

#### 3.4.8 Configurar campeonatos

**Descrição**: Configuração e começo de um campeonato
**Cenários**: Cenário 5 do enunciado - Configurar um campeonato.
**Pré-Condição**: Iniciador do campeonato autenticou-se
**Pós-Condição**: Campeonato iniciado
**Fluxo Normal**:

1. Jogador seleciona a opção de jogar campeonato
2. Sistema apresenta os campeonatos disponíveis
3. Jogador escolhe qual o campeonato que quer jogar
4. Sistema valida o campeonato escolhido
5. Sistema começa o campeonato

**Fluxo Exceção** (1): [Campeonato não existe] (passo 4)

4.1 Sistema indica campeonato inexistente
4.2 Sai do processo

#### 3.4.9 Configurar corridas

**Descrição**: Jogador decide se pretende afinar o carro antes da corrida.
**Cenários**: Cenário 5 do enunciado - Configurar corridas.
**Pré-Condição**: Situação pré-corrida de um campeonato e o participante não afinou o carro em 2/3 das corridas desse mesmo campeonato e categoria do carro é C1 ou C2.
**Pós-Condição**: Cada carro tem de estar configurado conforme as escolhas do jogador.
**Fluxo Normal**:

1. Sistema apresenta o nome do circuito, o número de voltas e as condições meteorológicas geradas aleatoriamente
2. Sistema questiona o jogador se pretende alterar configuração
3. Jogador responde que sim
4. Jogador escolhe o nível de PAC
5. Jogador escolhe o modo do motor
6. Jogador escolhe os pneus
7. Sistema valida os dados
8. Sistema mete o combustível e os pneus com capacidade a 100 %
9. Sistema guarda as alterações

**Fluxo Exceção** (1): [Nível de PAC inferior a 0 ou maior que 1] (passo 7)

7.1 Sistema informa nível de PAC tem que ser entre 0 e 1
7.2 Sai do processo

**Fluxo Exceção** (2): [O modo do motor ou os pneus escolhidos não existem no sistema] (passo 7)

7.1 Sistema informa que o modo do motor ou os pneus não existem no sistema.
7.2 Sai do processo sem guardar os dados.

**Fluxo Alternativo** (1): [Jogador não pretende alterar configuração] (passo 2)

1.1 Passo 4

**Fluxo Alternativo** (2): [Carro de categoria C2] (passo 2)

2.1 Sistema recalcula fiabilidade do carro com base nas afinações
2.2 Passo 3

#### 3.4.10 Simular corridas

**Descrição**: Simular corridas.
**Cenários**: Cenário 5 do enunciado - Simular corridas.
**Pré-Condição**: Todos os jogadores participantes deste campeonato tem de escolhido um piloto e um carro para participar no campeonato em questão e todos já tiveram a possibilidade de afinar os seus carros.
**Pós-Condição**: A tabela classificativa após a prova está atualizada
**Fluxo Normal**:

1. Sistema apresenta a classificação atual do campeonato
2. Sistema começa a prova
3. Em situações de curva, reta ou chicane, sistema simula a decisão de cada piloto.
4. Sistema calcula a fiabilidade para verificar se os pilotos vão bater ou não.
5. Sistema verifica se há ultrapassagens ou despistes com base nas decisões.
6. Para os pilotos que não se despistam o sistema reduz a quantidade de combustível e diminui a capacidade dos pneus.
7. Repetir desde o ponto 3 para todas as curvas, retas e chicanes para a volta inteira
8. Ao fim da volta o sistema apresenta a tabela classificativa atual
9. Volta a repetir o processo desde o ponto 3 para as próximas voltas até ao fim da prova
10. Fim da prova, sistema apresenta a classificação final, em que contém os melhores tempos de cada piloto, bem como os pontos ganhos da prova especifica.
11. Sistema verifica se tem que atualizar a melhor volta do circuito em questão.
12. Melhor volta antiga é melhor que a melhor volta da corrida, logo o sistema não atualiza
13. Sistema atualiza tabela classificativa do campeonato.

**Fluxo Alternativo** (1): [Pilotos despistaram-se] (passo 5)

5.1 Sistema desqualifica os pilotos que se despeitaram.
5.2 Retorna ao ponto 6

**Fluxo Alternativo** (2): [No caso de o simulador ser a versão Premium] (passo 4)

4.1 Nos casos em que não houve despiste de 2 pilotos e também não houve ultrapassagens entre ambos, o sistema faz também as reduções ou aumentos de tempo, com base nas decisões. Se as decisões tomadas por 2 pilotos forem iguais, o tempo entre ambos não se altera, mas se o piloto que vai à frente optou por arriscar e o outro piloto optou por jogar seguro, a diferença de tempo aumenta enquanto, no caso inverso, a diferença reduz-se.
4.2 Ponto 5

**Fluxo Alternativo** (3): [Combustível acaba ou os pneus desgastaram-se totalmente] (passo 7)

7.1 Sistema desclassifica da prova pilotos que ficarem sem combustível ou sem pneus.
7.2 Ponto 8

**Fluxo Alternativo** (4): [Carros de categoria GT] (passo 8)

8.1 Para cada carro de categoria GT diminuir fiabilidade.
8.2 Ponto 9

**Fluxo Alternativo** (5): [Atualiza melhor volta] (passo 12)

12.1 Sistema verifica que melhor volta antiga é pior que a melhor volta da corrida
12.2 Sistema atualiza melhor volta
12.3 Retorna ao ponto 12

#### 3.4.11 Simular campeonato

**Descrição**: Simular campeonato.
**Cenários**: Cenário 5 do enunciado - Resultado Final.
**Pré-Condição**: Todas as corridas do campeonato já foram simuladas.
**Pós-Condição**: Sistema guarda as pontuações adquiridas no campeonato em questão.
**Fluxo Normal**:

1. Sistema apresenta a classificação final do campeonato
2. Sistema verifica quais os jogadores que tem conta criada
3. Sistema atualiza a pontuação para o iniciador do campeonato
4. Sistema pergunta os dados de acesso a todos os participantes do campeonato que tem conta criada e que pretendem atualizar a sua pontuação no simulador.
5. Sistema valida dados
6. Sistema adiciona pontuação

7. Repetir o ponto 4 para os restantes paticipantes

**Fluxo Alternativo** (1): [Dados de acesso introduzidos não coincidem com os do sistema] (passo 5)

5.1 Sistema informa que dados de acesso não estão corretos.
5.2 Sistema repete o ponto 4 para os restantes participantes que não passaram pelo passo em questão

**Fluxo Alternativo** (2): [Jogador não quer atualizar pontuação] (passo 5)

5.1 Sistema repete o ponto 4 para os restantes participantes que não passaram pelo passo em questão

**Fluxo Alternativo** (3): [Carro usado pelo jogador é híbrido] (passo 6)

6.1 Sistema atualiza pontuações à parte, independentemente da categoria numa tabela própria para estes carros

#### 3.4.12 Consultar Classificação

**Descrição**: Jogador quer consultar Classificação
**Cenários**: Ainda antes de começar o jogo, o Miguel pretende verificar a tabela de classificação.
**Pré-Condição**: Utilizador está autenticado
**Pós-Condição**: É apresentada a tabela de Classificação
**Fluxo Normal**:

1. Jogador seleciona a opção de consultar classificação do menu principal.
2. Sistema apresenta a tabela de classificação
3. Volta ao menu principal.

#### 3.4.13 Consultar melhores voltas

**Descrição**: Jogador quer consultar melhores voltas
**Cenários**: O José pretende verificar os melhores tempos da pista "Autódromo de Braga".
**Pré-Condição**: Utilizador está autenticado
**Pós-Condição**: É apresentada a tabela de melhores tempos da pista em questão
**Fluxo Normal**:

1. Jogador seleciona a opção de melhores tempos.
2. Sistema pergunta qual a pista
3. Sistema valida a pista
4. Sistema vai buscar melhores voltas do circuito
5. Sistema apresenta o melhor tempo da pista
6. Volta ao menu principal.

**Fluxo Exceção**(1) [Pista inexistente]: (passo 3)

3.1 Sistema informa pista inexistente
3.2 Volta ao menu principal

# 4 Segunda Fase

## 4.1 Objetivos segunda fase

Nesta etapa do projeto foi-nos pedido que desenvolvêssemos a arquitetura do sistema a implementar e que descrevêssemos melhor certos comportamentos do sistema. Também foi fornecido pelos docentes código legado, que iremos analisar na secção 4.4, tentando aproveitar ao máximo o código já feito, poupando assim esforço desnecessário.

É importante salientar que, antes de trabalharmos propriamente nos objetivos desta fases, fizemos algumas alterações e correções na primeira fase do projeto. Estas alterações encontram-se na secção 4.2.

Para a definição da arquitetura conceptual do sistema elaboramos seguintes diagramas, tal como foi instruído pela equipa docente:

- Diagrama de componentes
- Diagrama de classes
- Diagrama de pacotes

De modo descrever alguns dos mencionados comportamentos usamos os diagramas de sequência para explicitar o funcionamento dos métodos a desenvolver.

Esperamos que com os diagramas a fazer, consigamos estruturar ainda mais a peça de software a desenvolver, orientando-nos de modo a que na terceira fase consigamos construir um sistema mais rapidamente, mais objetivamente e mais eficientemente, que cumpra com a especificação feita nas etapas prévias.

## 4.2 Alterações primeira fase

Durante a resolução da segunda fase e após uma analise extensa da primeira fase, o grupo reparou que havia certos pormenores que se encontravam incorretos, tendo decidido corrigir e melhorar esses aspetos, que se encontram listados de seguida:

Primeiramente, no modelo de domínio não havia destaque suficiente tendo em conta um carro ser híbrido. Então foram acrescentadas 3 novas entidades, o C1Híbrido, o C2Híbrido e o GTHíbrido. Cada uma destas entidades contem um motor elétrico. Assim, de modo a cumprir com as sugestões dadas pelo docente nas aulas teóricas, alteramos certos aspetos do nosso modelo de domínio, incorporando neste as entidades mencionadas e alterando a forma de algumas setas que simbolizam as relações entre as entidades tornando o nosso modelo um pouco mais legível dado que ele agora tem um tamanho mais extenso. Também alteramos as relações com a entidade "Classificação Campeonato"uma vez que não tinha necessidade de estar relacionada com os pilotos e os carros.

Relativamente aos use cases que se encontravam errados, alguns estavam incompletos, nomeadamente devido à ausência de alguns passos ou fluxos, outros continham passos repetidos e outros que não estavam bem estruturados.

Também alteramos os cenários, enquadrando os nossos use cases com os cenários presentes no enunciado sempre que possível.

As alterações nos use cases foram:

- Criar campeonato: Melhoramos os fluxos de exceção
- Criar circuito: Alteramos alguns passos do Fluxo Normal e melhoramos os fluxos de exceção, que antes não estavam tão explícitos.
- Criar piloto: Indicamos melhor quais os casos de exceção.
- Criar carro: Explicamos melhor quais os casos de exceção.
- Aderir a um campeonato: Mudamos a pré e pós condições. Adicionamos alguns passos ao Fluxo Normal e alteramos os fluxos de exceção.
- Configurar corridas: Alteramos a pré e pós condição e adicionamos passos ao Fluxo Normal.
- Configurar campeonato: Removemos o include de aderir a um campeonato e adicionamos o passo de validar se o campeonato existe.
- Simular corridas: Removemos o include relativo a configurar corrida e passamos para pré condição que os participantes do campeonato já fizeram as configurações
- Simular campeonatos: Removemos o include de simular corridas e metemos como pré condição que as corridas já devem ter sido simuladas, para além disso alteramos o português, trocando o termo "criador"no cenário para "inicializador do campeonato". Adicionamos um fluxo alternativo garantindo a separação das pontuações dos carros híbridos dos movidos a motores de combustão, assim, asseguramos que a pontuação dos carros híbridos é calculada de forma diferente e armazenada numa tabela à parte.
- Consultar melhores voltas: Alteramos o Fluxo Exceção
- Acrescentar pontuação: Interpretamos mal este use case, pois pensamos que o sistema teria que guardar as pontuações ao fim de um campeonato e qualquer participante desse campeonato poderia fazer login quando quisesse para resgatar os ponto. Este processo passou a estar incluído no simular campeonatos, em que no final deste use cases, o sistema passa por um processo de login.

No que diz respeito ao use case de simular corridas, faltou-nos indicar que o sistema no fim das corridas verifica se precisa de atualizar a tabela das melhores provas relativa ao circuito em questão.

Inclusive o grupo teve em consideração a especificação de fluxos alternativos que se encontravam demasiado generalizados, optando por criar fluxos mais específicos à ação a ser realizada pelo sistema.

De resto, fizemos alterações mínimas ao longo do relatório nomeadamente: atribuímos legendas a figuras e adaptamos o texto de modo a referenciar as mesmas.

## 4.3 Diagrama de componentes

A próxima etapa do projeto é fazer uma diagrama de componentes.

Podemos ver um diagrama de componentes como uma breve introdução ao diagrama de classes, pois nesta fase iremos levantar sistemas e sub-sistemas que posteriormente irão resultar em classes.

O primeiro passo para fazer o diagrama de componentes foi passar todos os use cases para um folha de cálculo *Excel* formatada com uma template disponibilizada pela equipa docente.

Após este processo tivemos que associar a cada passo dos casos de uso, uma breve descrição do que é que o sistema deveria fazer, os métodos que originavam-se a partir dessa etapa e o subsistema correspondente em que os métodos atuavam.

Uma boa parte das etapas dos casos de uso estão diretamente relacionados com o utilizador fornecer dados ao sistema. Como este processo está inteiramente ligado a input output, quando estamos perante este tipo de situações, não é necessário identificar nada, pois não está incluído em nenhum subsistema.

O diagrama de componentes que construimos está ilutrado na figura [**7**]

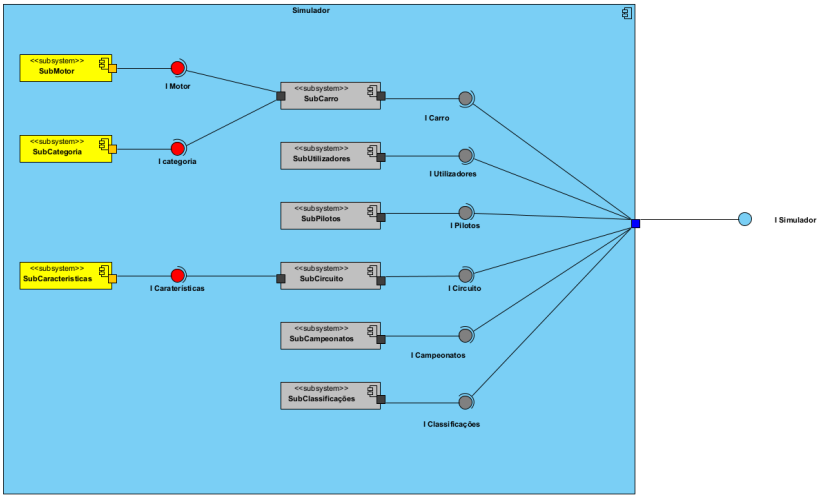


Figura 7: Diagrama de Componentes completo

## 4.4 Código Legado

Para auxílio no desenvolvimento do projeto, a equipa docente forneceu ficheiros com código em Java corresponde a versões posteriores/alternativas do projeto ao que é pretendido realizar, com o propósito de promover a reutilização de código, nesta secção iremos indicar quais dos módulos do código fornecido iremos utilizar e quais módulos iremos descartar.

Classes que iremos utilizar:

- Jogador

- Piloto
- Circuito
- GT
- GTH
- PC1
- PC1H
- PC2
- PC2H
- SC
- Carro
- Campeonato

Algumas destas classes iremos utilizar tal e qual como está no código, outras iremos adaptar, implementando algumas mudanças de forma a seguir a nossa ideia original.

Em relação à classe carro, na nossa implementação não iremos considerar carro como classe abstrata. A nossa ideia é que a categoria seja identificada por um atributo de carro e não como uma subclasse. Ou seja as classes GT, GTH, PC1, PC1H, PC2, PC2H e SC não irão ser sub classes de carro, mas sim atributos do mesmo. O atributo fiabilidade não irá estar na classe carro, mas sim na categoria do próprio carro. Por último, o método tempoProximaVolta não iremos contabilizar, pois forma do cálculo do tempo de cada volta, não irá ser feito dependendo do carro, mas sim com base no conjunto das decisões que o piloto toma para cada curva, reta ou chicane.

A classe jogador na nossa implementação irá ser diferente. Nós consideramos que um jogador é uma subclasse de Utilizador, tanto como a classe Admin. Na nossa classe jogador iremos simplesmente guardar a pontuação global no sistema. A classe utilizador irá conter o seu nome de utilizador e a sua palavra passe que será util para pormenores de autenticação.

A nossa classe circuito não irá diferenciar muito do código legado. A nossa classe circuito irá armazenar as 10 melhores voltas para aquele circuito, ou seja não consideramos apenas o record. Aspetos como tempo médio e desvio não chegamos a considerar, porque achamos que não seja algo muito relevante.

Por último a nossa classe campeonato irá ser diferente. Nós achamos por bem que seja possível jogar-se varias vezes o mesmo campeonato, então nós dicidimos que a classe campeonato simplesmente armazena o seu nome e os circuitos do mesmo. Para a simulação dos campeonatos foi criada a classe ProvaCampeonato que iremos explicá-la mais à frente.

Classes que não iremos utilizar:

- Aposta
- Equipa
- Corrida

Estas classes não serão utilizadas, pois são representativas de ideias que não consideramos ou ideias que tencionamos implementar de forma diferente, para nós mais intuitiva.

A classe corrida é a única que iremos implementar de forma diferente. Nós optamos por em vez de ter uma classe Corrida, temos um classe Campeonato-Prova que irá ela própria irá simular as corridas. Esta classe iremos explicar mais à frente.

As restantes classes, não iremos utilizar, visto que são ideias que não chegamos a considerar.

Classes que iremos utilizar e que não estão no código:

- Prova Campeonato
- Categoria
- Motor
- Categoria
- SimuladorFacade

A classe SimuladorFacade é a classe que irá fazer a ponte do nosso simulador entre a parte lógica de dados com a parte input output.

As classes Motor e Categoria irão resultar ao fim ao cabo em atributos de carro, simplificando de certa forma a classe carro e as suas subclasses do código legado da equipa docente. Ou seja, as classes GTH, PC1H e PC2H, do código legado, irão ser subclasses de GT, PC1 e PC2 respetivamente na nossa solução. Cada uma dessas subclasses irá ter como atributo um motor elétrico.

Em relação à nossa classe Prova campeonato, podemos ver esta classe como uma substituta à classe Corrida, tal como já foi referido. Este módulo armazena os resultados de um conjunto de corridas realizadas para um determinado campeonato. Ou seja, para termos uma prova campeonato, temos ter associações com as classes campeonato, para saber qual campeonato a ser jogado, os pilotos e os carros, para saber as escolhas dos participantes e por último os jogadores que irão jogar o campeonato. Esta classe também armazena a classificação da simulação do campeonato e também guarda as classificações individuais de cada prova. Para simular um campeonato, é preciso criar uma instância desta classe, adicionar todos os jogadores e as escolhas de cada um, e por último invocar o método simular campeonato.

## 4.5 Diagrama de Classes

Tendo em conta o raciocínio explicado na secção anterior a esta, iremos agora apresentar o nosso diagrama de classes, este nada mais é que um esquema que explicita a forma como os diferentes módulo do nosso código se relacionam entre si.

Nota Importante: Por questões de simplicidade e eficiência optamos pela agregação em todo o estado eterno na aplicação, fazendo apenas composição em pormenores do facade.

Optamos por utilizar mapas como estruturas de armazenamento da informação que consta na base de dados que iremos futuramente criar.

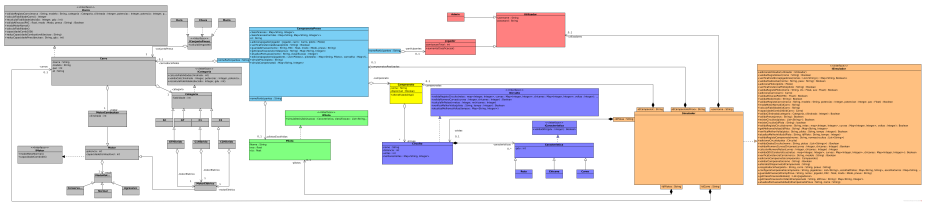

Figura 8: Diagrama de Classes completo

O uso de Interfaces (”ICarro”, ”ICaracteristicas”, ”IConjuntoPneus”, ”ICategoria”, ”IMotor”, ”ICategoria”, ”IPiloto”, ”ICircuito”e ”ISimulador”) foi preferido ao invés de classes abstratas pois a declaração de métodos numa Interface permite que mais do que uma classe os possa implementar tornando o código menos redundante e mais funcional.

A Figura [**9**] apresenta a Interface ICarro implementada pela classe Carro. A Classe carro é composta pelas classes MotorCombustão e Categoria.Por sua vez as classes duro, chuva e macio implementam a Interface IConjuntoPneus. A classe Categoria implementa a interface ICategoria. No que diz respeito às classes SC, GT, C1 e C2, estas são generalizações da classe Categoria. As classes GTHibrido, C1Hibrido e C2Hibrido são extensões das classes GT, C1, C2, respetivamente. o Motor eletrico é extensão da classe Motor e tem uma relação de agregação com as classes GTHibrido, C1Hibrido e C2Hibrido, como mostra a figura em questão. Por outro lado o Motor, para além da relação descrita previamente implementa a interface IMotor; é composta pela classe ModoMotor, que por sua vez heredita três classes: Conservador, Normal e Agressivo; e apresenta-se como extensão do MotorCombustão. Por último devemos referir, como mostra a imagem que as Classes simuladorFacade e CampeonatoProva, assim como a Interface IConjuntoPneus estabelecem relação de agregaç com a classe Carro.

Tomando atenção à Figura[**10**], para além do que foi explicado anteriormente, esta permite-nos saber que a classe CampeonatoProva encontra-se relacionada com as seguintes Classes: Campeonato, simuladorFacade, Jogador, Carro e o Piloto.

Tendo em consideração a Figura[**11**] é de fácil compreensão as relações apresentadas. Conseguimos perceber que a Interface ICaracteristica está relacionada com a classe Caracteristica uma vez que apenas utilizaremos a função presente na interface somente na perspetiva de implementação. Percebemos claramente que as classes Reta, Chicane e Curva são parte da herança das Caracteristica e que por sua vez esta está conectada ao Circuito através de uma relação de agregação. No que diz respeito à classe Circuito, esta está está ligada às classes

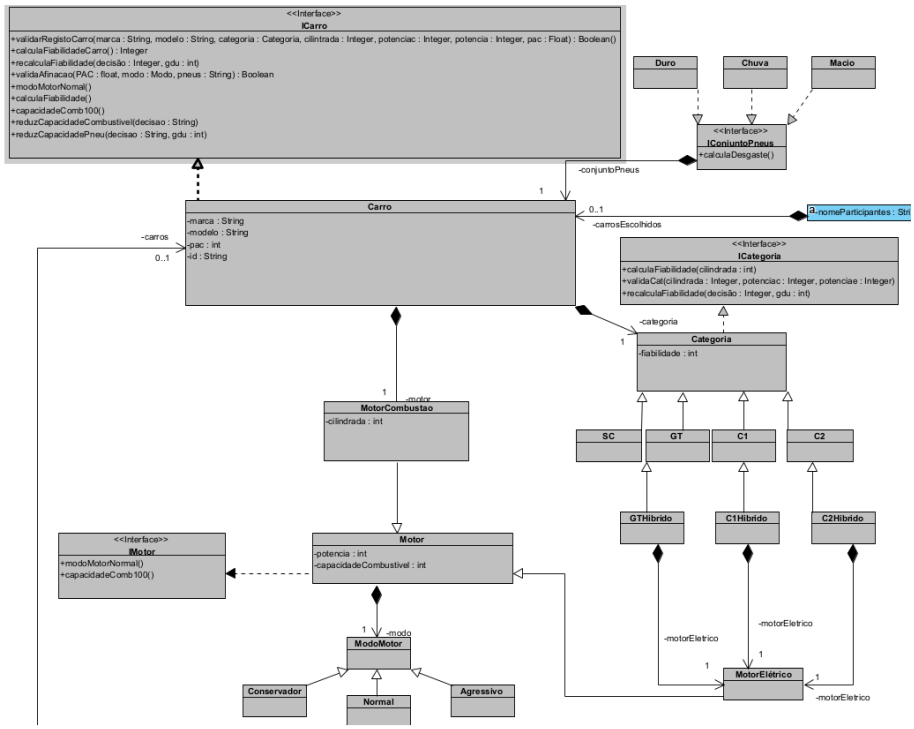


Figura 9: Diagrama de Classes - Carros

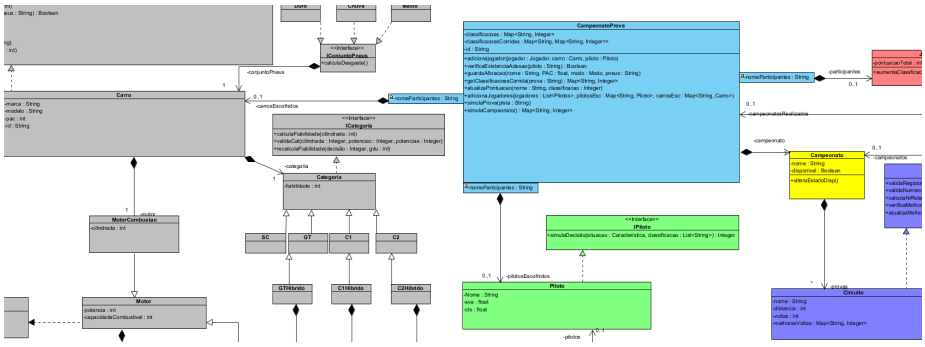

Figura 10: Diagrama de Classes - CampeonatoProva

simuladorFacade e Campeonato por relações de composição e agregação respetivamente.

Observando a Figura[**12**] conseguimos perceber que a classe Piloto é constituída por 3 atributos: nome, sva e cts. Esta classe encontra-se relacionada com simuladorFacade e CampeonatoProva através de uma relação de composição e

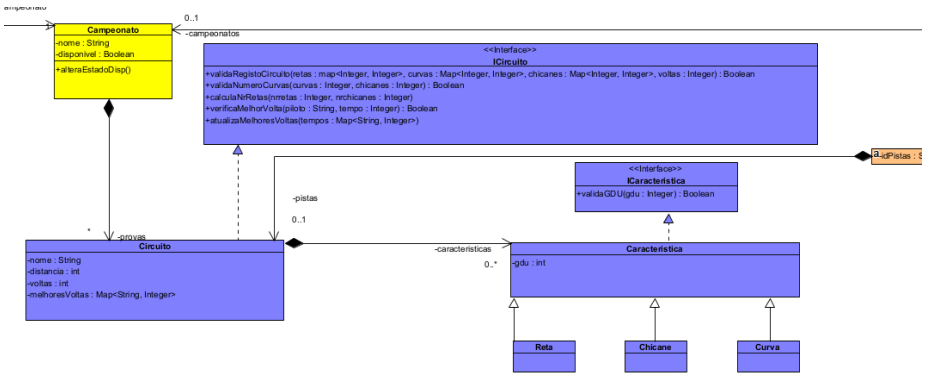


Figura 11: Diagrama de Classes - Circuito

agregação.

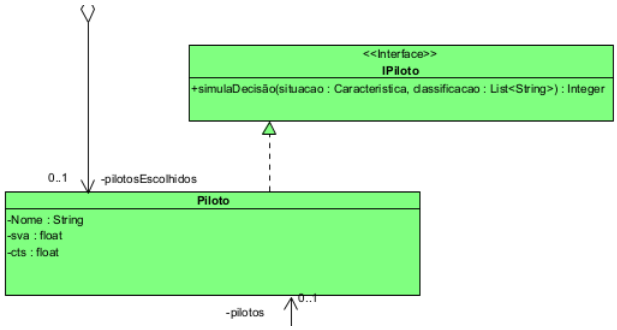


Figura 12: Diagrama de Classes - Piloto

Através da Figura[**13**] podemos observar a classe Admin e Jogador, que fazem parte da herança do utilizador. Este por sua vez está relacionado com o simuladorFacade por uma relação de agregação.

Tal como podemos ver pela Figura[**14**], o simuladorFacade é uma classe que permite uma interface simples para um sistema já complexo em si. A utilização desta classe faculta ao utilizador apenas os recursos com que este vai interagir, permitindo assim uma ligação sistema/utilizador mais conveniente e simples.

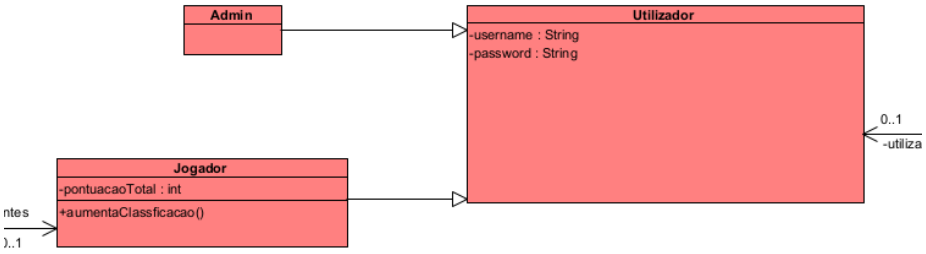


Figura 13: Diagrama de Classes - Utilizador

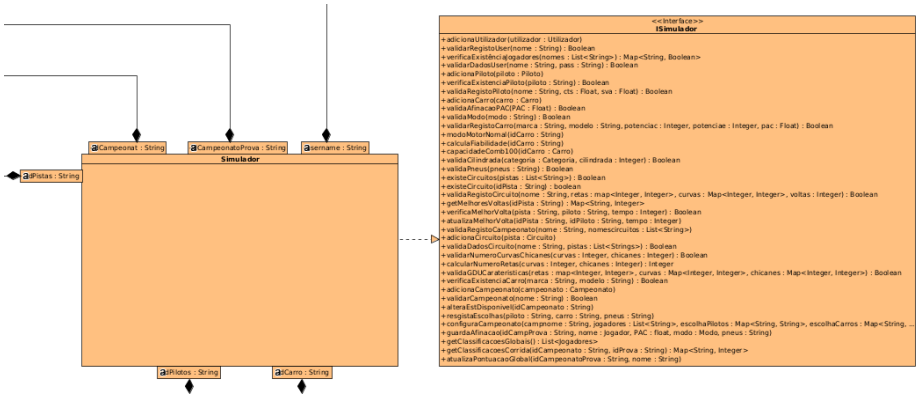


Figura 14: Diagrama de Classes - SimuladorFacade

## 4.6 Diagramas de Sequência

O grupo desenvolveu alguns diagramas de sequência de modo a ilustrar os use cases do cenário 5, como pedido pelos docentes da cadeira. Em seguida estão apresentados os diagramas de sequência para os seguintes use cases: Configurar Campeonato [**15**], Configurar Corridas [**16**], Simular Corridas [**17**], Simula Campeonato [**18**] e Afinar Carro [**19**].

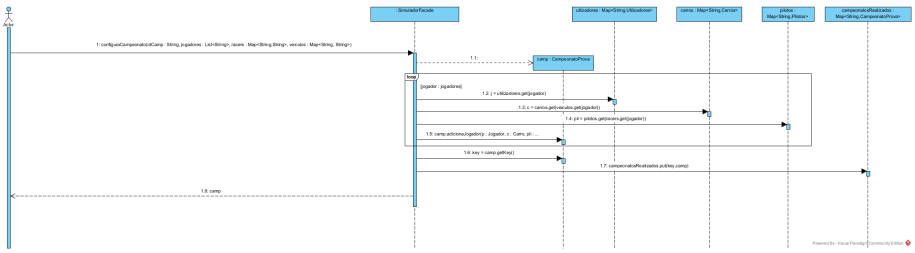


Figura 15: Diagrama de Sequência - Configurar Campeonato

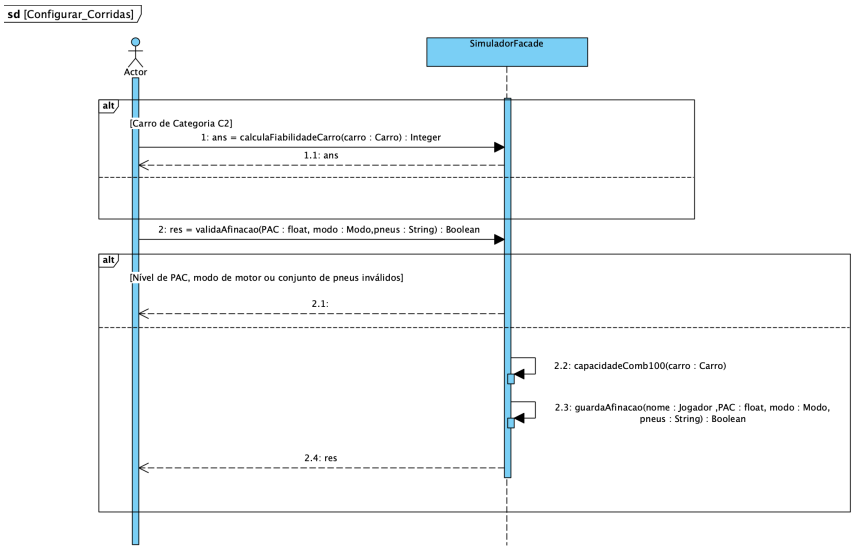


Figura 16: Diagrama de Sequência - Configurar Corridas

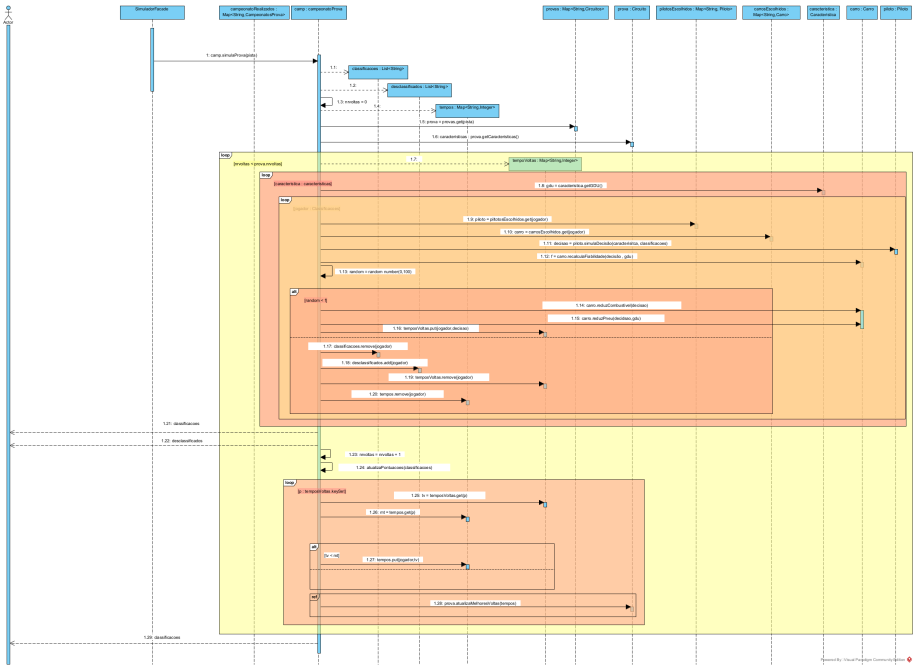

Figura 17: Diagrama de Sequência - Simular Corridas

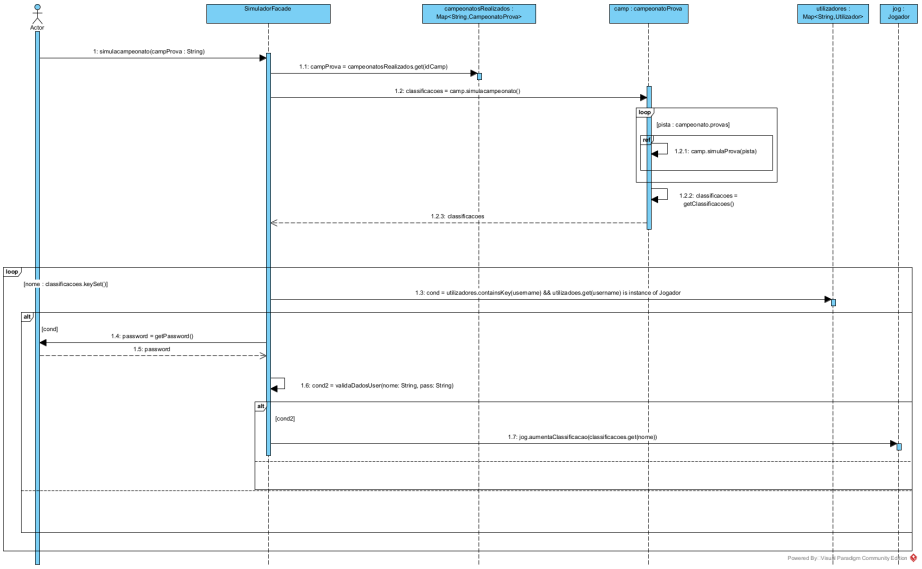


Figura 18: Diagrama de Sequência - Simular Campeonato

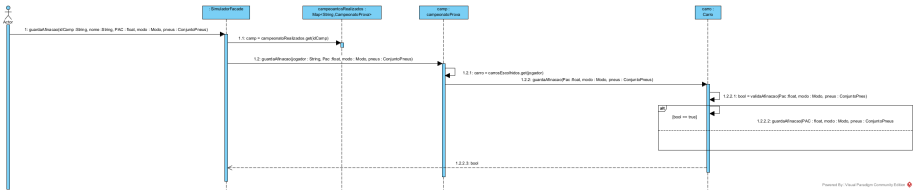

Figura 19: Diagrama de Sequência - Afinação

Decidimos também, para além dos diagramas pedidos adicionar mais alguns que pensamos que são necessários e que nos poderão auxiliar na próxima fase do trabalho de DSS. Entre os diagramas expostos seguidamente fazem parte de extras que consideramos importantes acrescentar. Os diagramas expostos são: Aderir Campeonato [**20**], Consultar Melhores Voltas [**21**], Consultar Classificação [**23**] e o Atualizar Melhores Voltas [**22**].

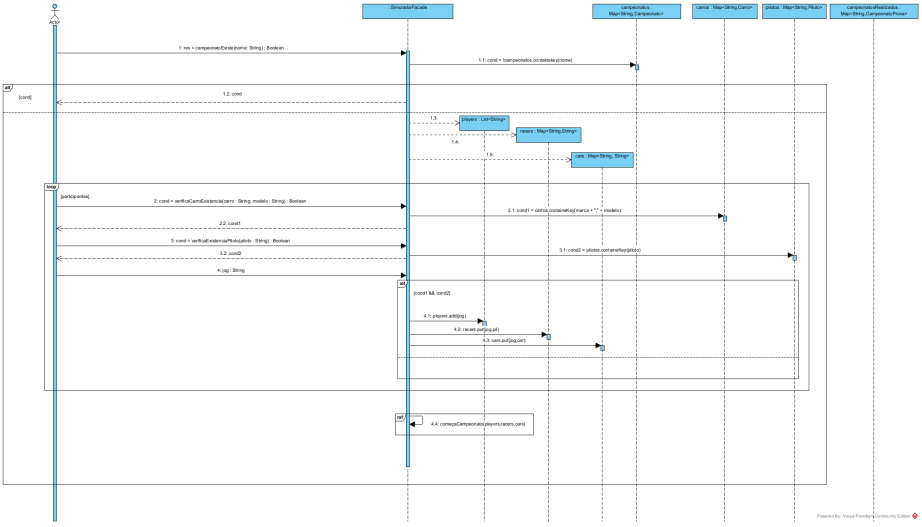

Figura 20: Diagrama de Sequência - Aderir Campeonato

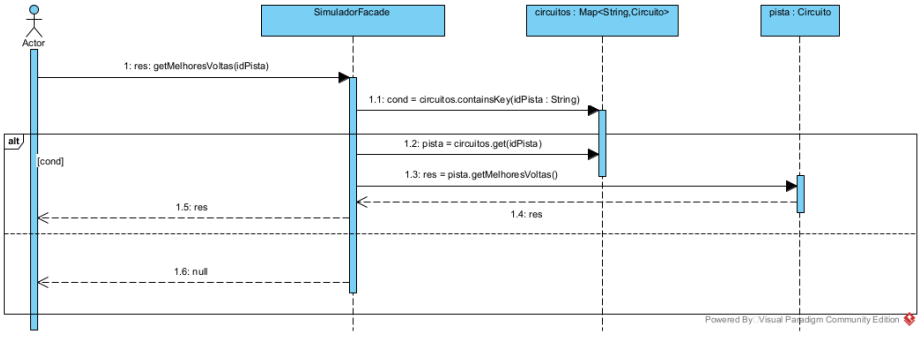


Figura 21: Diagrama de Sequência - Consultar Melhores Voltas

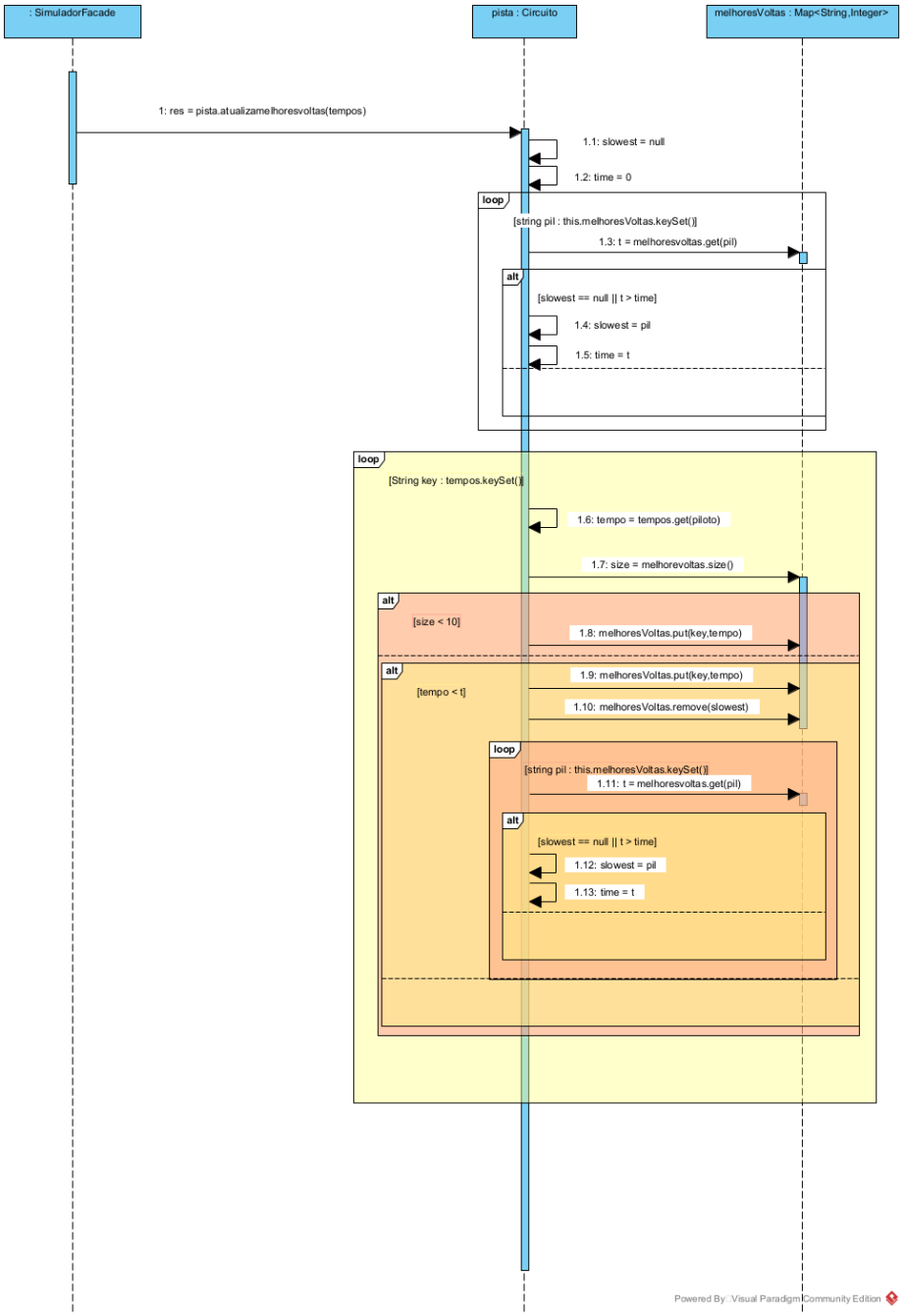


Figura 22: Diagrama de Sequência - Atualiza Melhores Voltas

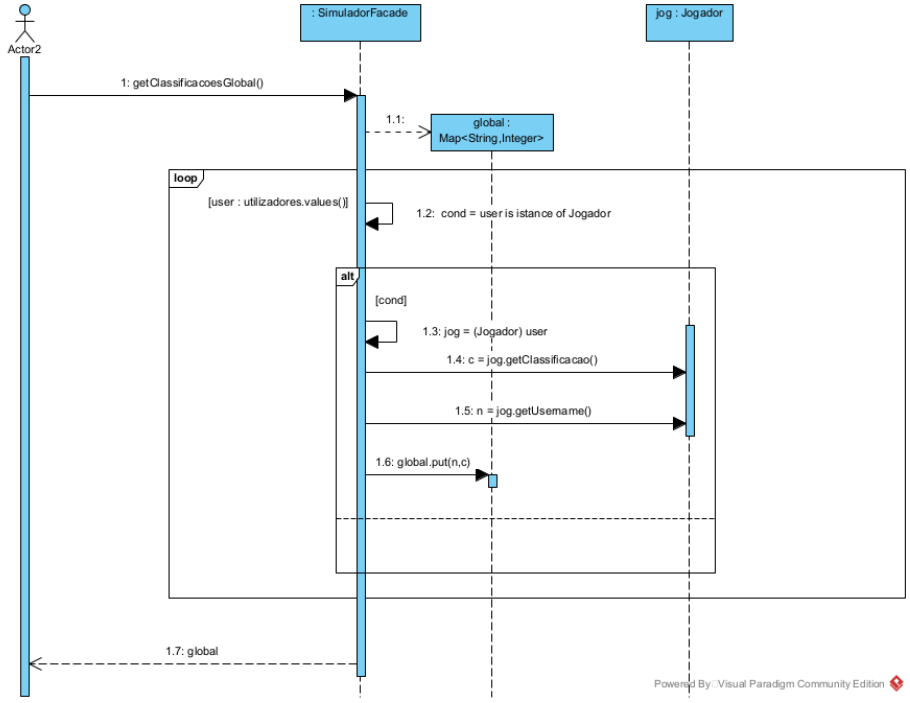


Figura 23: Diagrama de Sequência - Consultar Classificações

## 4.7 Diagrama de Pacotes

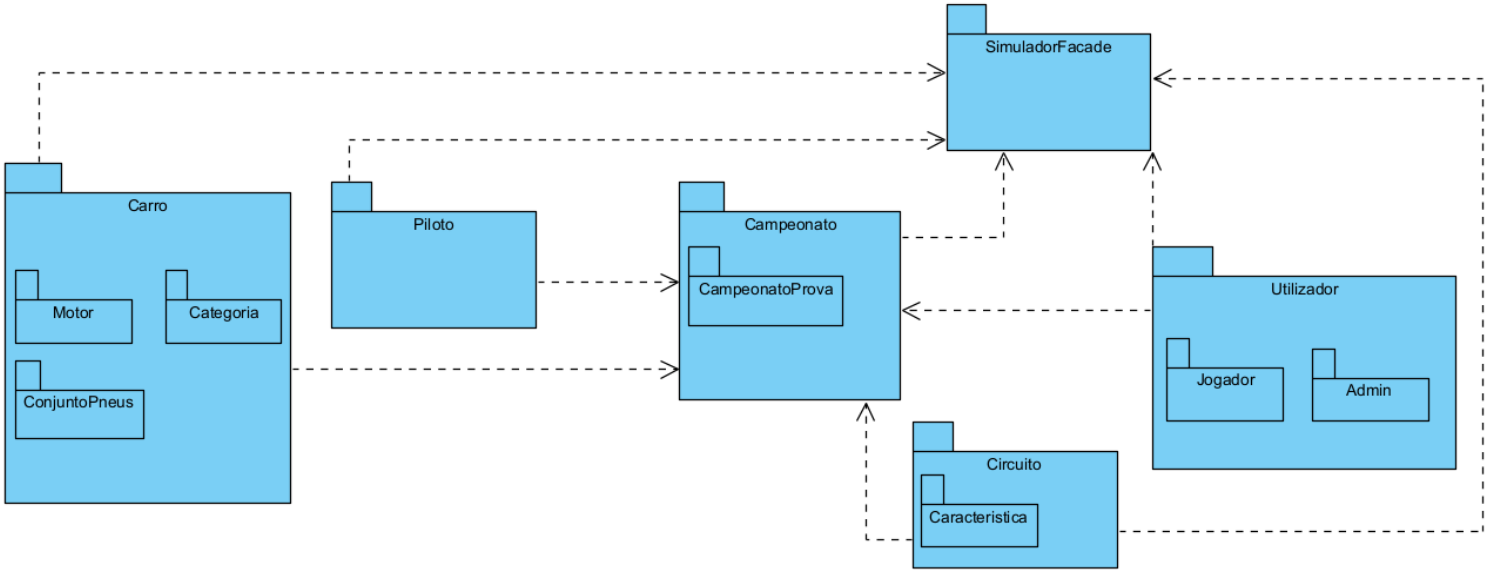


Figura 24: Diagrama de Pacotes da nossa arquitetura

No diagrama de pacotes anexado na figura [**24**] podemos perceber que as classes previamente apresentadas no diagrama de classes se traduzem para seis pacotes, passamos a ter um pacote representativo da parte do sistema correspondente aos carros, a este pertencem os pacotes que representam as partes do sistema que que são utlizadas para definir um carro, ou seja, pacotes para os motores, categoria e o conjuto de pneus; um pacote para os pilotos; um pacote para os campeonatos que contem o pacote referente à classe campeonatoProva; um pacote para os circuitos onde podemos encontrar um pacote para as caracteristicas que definem um circuito; também temos um pacote para representar os utilizadores, onde temos o pacote que define um jogador ou um administrador. Por último, temos também um pacote que representa a Facade do nosso sistema.

A nível das dependências, o pacote representativo do facade depende de todos os outros pacotes, enquanto que o pacote que contem a parte do sistema que retrata os campeonatos depende de todos os pacotes exceto do que simboliza o facade.

## 4.8 Conclusão - 2ª fase

Com a finalização desta etapa do projeto sentimos que estamos muito mais aptos para desenvolver o *software* que desejamos. As representações esquemáticas que desenvolvemos, em especial o diagrama de compone diagrama de classes e o diagrama de pacotes, permitiram que como grupo consigamos dividir o nosso problema em partes, partes essas em que conseguiremos trabalhar de forma síncrona, mais rapidamente e mais eficientemente.

No geral, estamos contentes com os modelos que desenvolvemos, claro, há sempre coisas a melhorar, mas com as correções que fizemos aos problemas apontados pelos docentes relativamente ao que fizemos na primeira parte, juntamente com os modelos e diagramas específicos desta etapa, sentimos que temos uma especificação que de certa forma nos satisfaz.

Assim, achamos que estamos preparados para desenvolver a próxima etapa deste projeto e também estaremos sempre disponíveis para corrigir ou alterar o trabalho realizado nesta fase consoante as futuras indicações da equipa docente.