Escola de Engenharia

Universidade do Minho – Departamento de Informática
Licenciatura em Engenharia Informática (LEI)

3º ano – 1º semestre
Ano Letivo 2022/2023

RELATÓRIO – Trabalho Prático
Entrega Final

Desenvolvimento de Sistemas de Software (DSS)

F1 Manager: Simulador de campeonatos de automobilismo

Grupo 27

A97540, Ana Rita Moreira Vaz
A92847, Guilherme Sousa Silva Martins
A97777, Millena de Freitas Santos
A96794, Ricardo Alves Oliveira

Link repositório GitHub: https://github.com/GuiSSMartins/DSS_Grupo27_2022_23




Conteúdo

Lista de Figuras

1 Introdução

Este relatório é um documento técnico escrito para o trabalho prático desenvolvido no âmbito da unidade curricular de Desenvolvimento de Sistemas de Software do 3º ano - 1º semestre da licenciatura em Engenharia Informática da Universidade do Minho.

O relatório tem como objetivo apresentar uma visão aprofundada do trabalho prático e do processo de desenvolvimento seguido, a fim de alcançar um resultado final robusto, coerente, válido e confiável. Nele está descrita a terminologia específica ao sistema, bem como, os modelos que compõem esta fase final do projeto. O projeto *"Racing Manager"* consiste num sistema de simulação de campeonatos de automobilismo. Os utilizadores podem entrar em campeonatos, escolher carros e pilotos, simular corridas com *bots* e visualizar ranking global das pontuações dos vários jogadores que já participaram em campeonatos.

Utilizou-se a UML, *Linguagem de Modelagem Unificada*, a fim de elaborar os diagramas que compõem esta fase projeto, nomeadamente: diagramas de sequência e diagramas de classes: Para fins de implementação do programa, foi usada a linguagem de programação *Java* e a linguagem *MySQL* para criar uma conexão com a base de dados.

2 Objetivos

A última fase do projeto envolve a criação dos modelos necessários à descrição da implementação do sistema; e a respetiva implementação da solução, segundo os vários modelos e diagramas que foram desenvolvidos nas duas fases anteriores.

Nesta fase o principal objetivo é a implementação do projeto, em especifico a implementação do sistema de simulação de campeonatos e das respetivas corridas.

Para além do ambiente de simulação, temos também como objetivo a implementação de uma base de dados que, de forma segura e acessível, permitam aceder aos dados gerados para o projeto mesmo após o término da execução.

Além da base de dados pretendemos nesta fase a implementação de DAO's que permitam a interação entre a mesma e o ambiente de simulação.

3 Terminologia específica

Os termos seguintes são típicos do "mundo"do Automobilismo. Entender-los pode facilitar na interpretação dos modelos e do sistema em si.

3.1 Cilindrada

Capacidade de um motor de combustão de queimar a soma de combustível e ar nos seus pistões.

3.2 Chicane

Desvio artificial de um circuito, a fim de diminuir a velocidade de quem por ele passa, como medida de segurança.

3.3 Classe 1

Os carros de Classe 1 são protótipos feitos especialmente para o campeonato e apresentam uma cilindrada de 6000cm3.

3.4 Classe 2

Os carros de Classe 2 são veículos de alta performance que podem apresentar cilindradas entre 3000cm3 e 5000cm3.

3.5 Classe GT

Os carros de Classe GT (Grande Turismo) são carros desportivos produzidos em massa que podem apresentar uma cilindrada que varie entre 2000cm3 e 4000cm3.

3.6 Classe SC

Os carros da Classe SC (Stock Cars), são derivados de automóveis quotidianos apresentando, assim, uma cilindrada de 2500 cm3.

3.7 CTS

O CTS, "Chuva vs. Tempo Seco", é um critério que varia de 0 a 1 e indica o nível de perícia de um piloto em relação às condições atmosféricas. Um valor mais baixo indica um melhor desempenho em tempo chuvoso, já o contrário demonstra um melhor desempenho em tempo seco.

3.8 GDU

O GDU, ou Grau de Dificuldade de Ultrapassagem, indica, para cada segmento de um circuito, o quão difícil seria realizar uma ultrapassagem (não tendo em conta diferenças entre carros). Este pode tomar três valores: Possível, Difícil e Impossível.

3.9 PAC

O PAC, ou Perfil Aerodinâmico do Carro, é uma métrica que varia entre 0 e 1, onde um valor próximo de 0 indica um carro mais rápido, enquanto um valor mais próximo de 1 indica um carro mais estável.

3.10 SVA

O SVA, "Segurança vs. Agressividade", é um critério que varia de 0 a 1 e indica a tendência de um piloto em cometer decisões mais arriscadas ao longo das corridas. Isto significando que um piloto com um maior nível de SVA é mais propicio a arriscar uma ultrapassagem perigosa que poderá resultar em despiste.

4 Revisões efetuadas

No último documento referente a este projeto, os diagramas de sequência tiveram muito foco em cenários que não eram tão interessantes ou desafiadores como os referentes ao Administrador adicionar Campeonato, Circuito, Carro e Piloto. O Cenário 5 é o crucial para perceber o funcionamento do projeto, portanto, começamos por implementar diagramas de sequência referentes a este cenário de modo a tornar este relatório mais completo e detalhar como a jogabilidade no geral deve ser implementada.

4.1 primeiroVolta

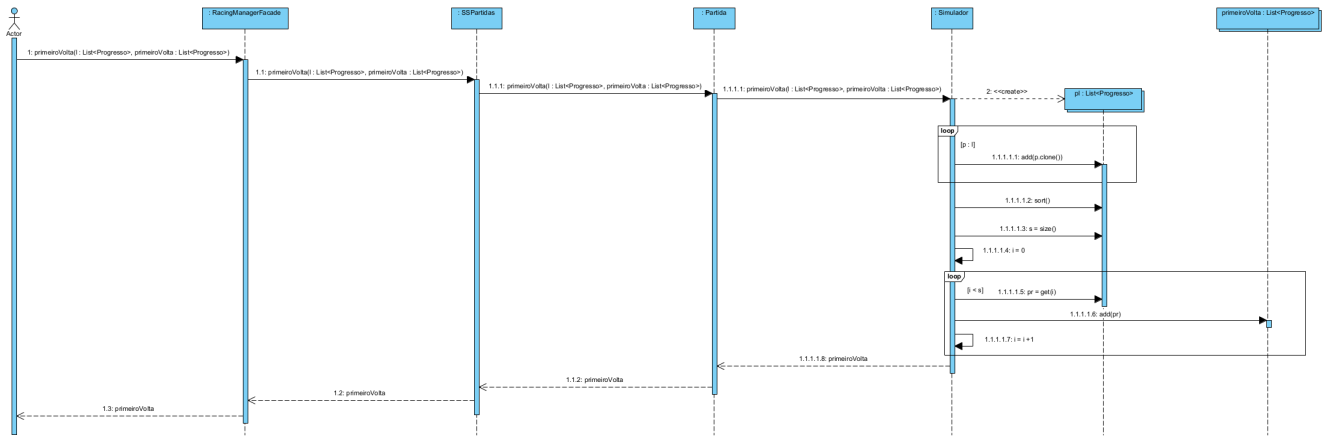

Figura 1: Diagrama de Sequência - método *primeiroVolta*

4.2 getDNF

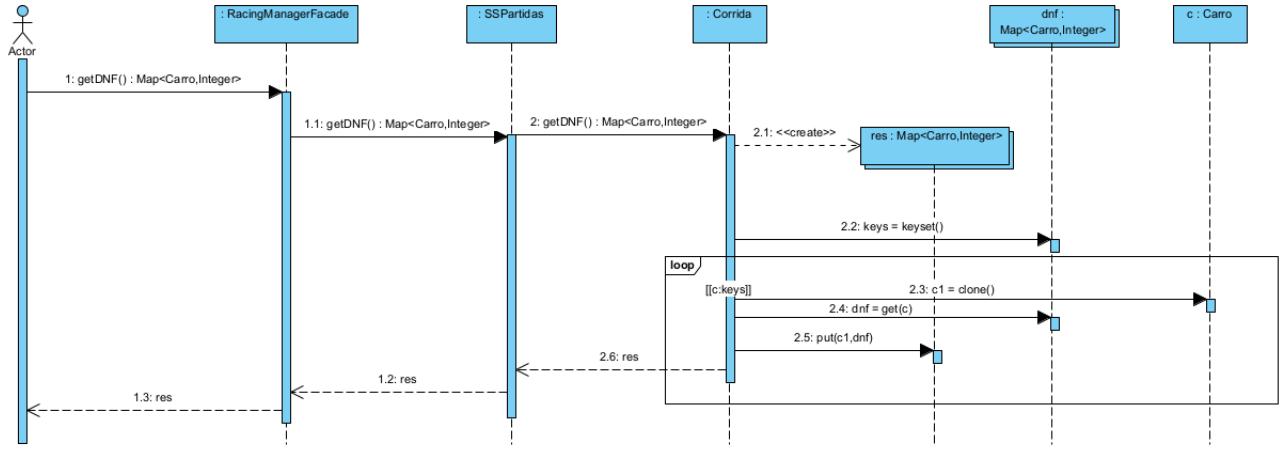

Figura 2: Diagrama de Sequência - método *getDNF*

4.3 hasAfinacoes

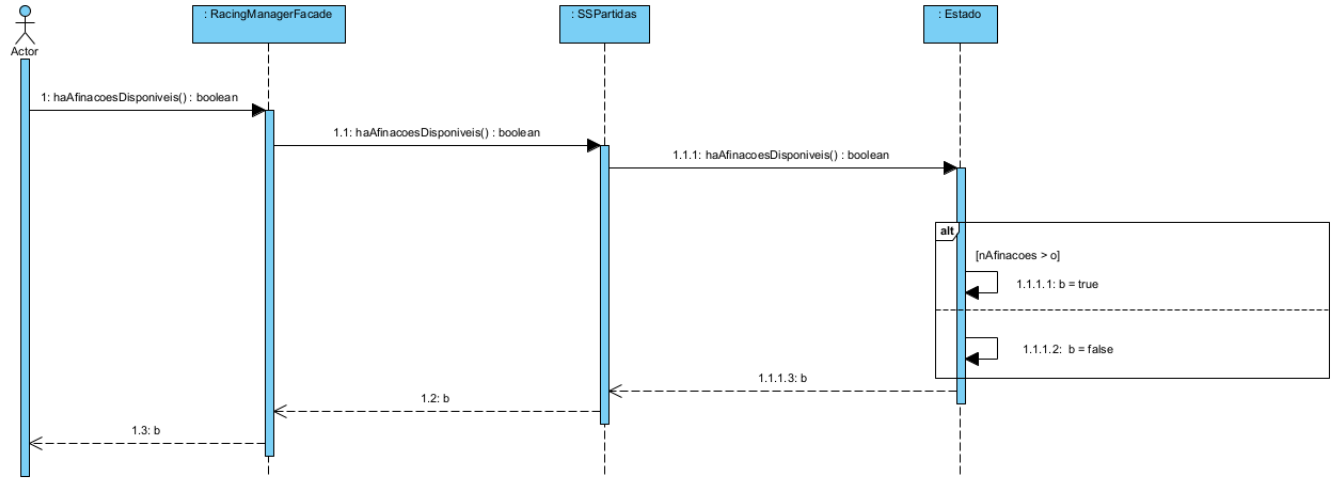

Figura 3: Diagrama de Sequência - método *hasAfinacoes*

4.4 iniciarCorrida

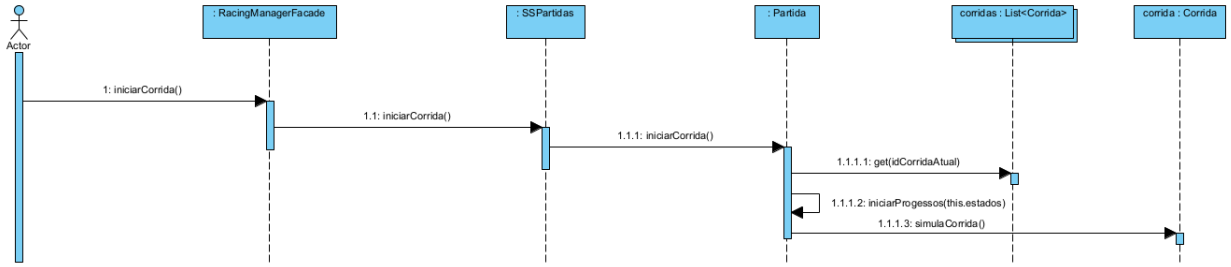

Figura 4: Diagrama de Sequência - método *iniciarCorrida*

4.5 simulaCorrida

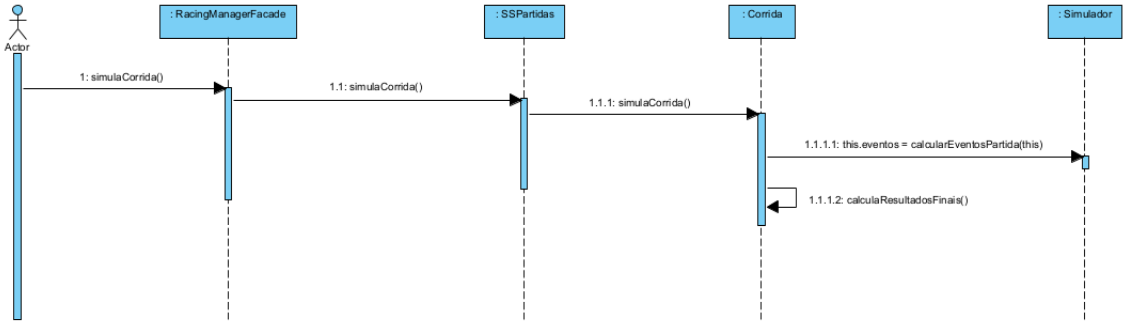

Figura 5: Diagrama de Sequência - método *simulaCorrida*

4.6 calculaResultadoPartida

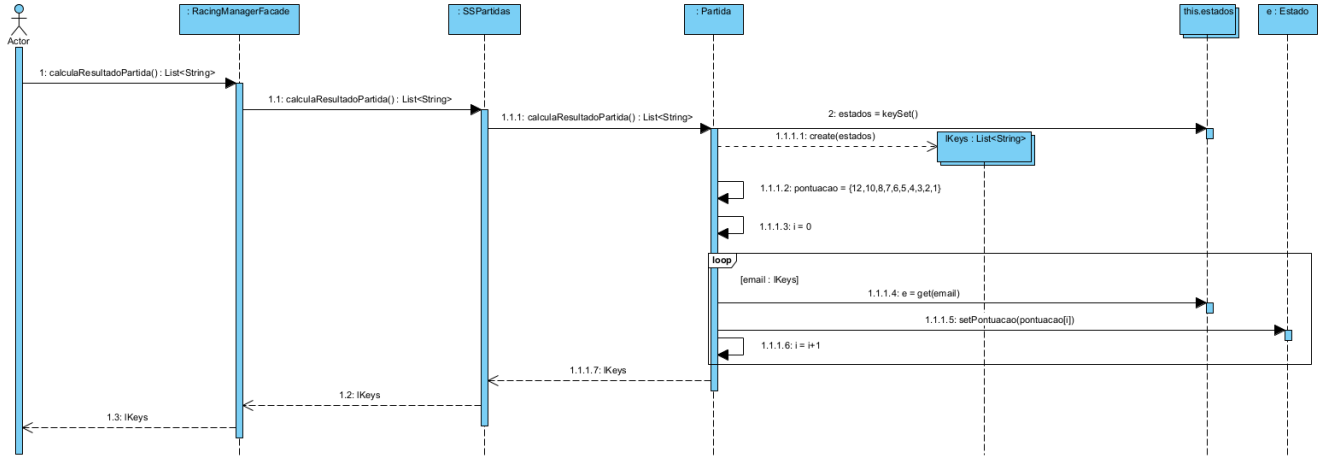

Figura 6: Diagrama de Sequência - método *calculaResultadoPartida*

4.7 Revisões feitas ao Diagrama de Classes - SSPartidas

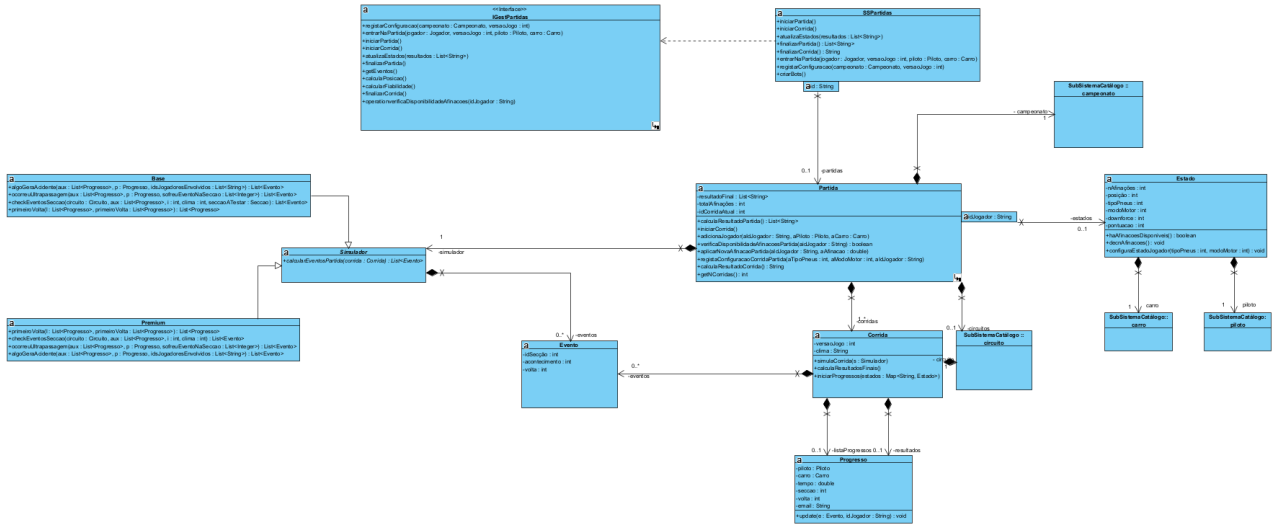

Figura 7: Diagrama de Classes para o SSPartidas

A figura 7 consta as alterações feitas ao diagrama de classes do subsistema Partidas. Algumas estruturas de dados foram modificadas, outras foram adicionadas. O mesmo para operações.

5 Implementação de Persistência de Dados

5.1 Raciocínio

O subsistema SSPartidas é o responsável pela implementação do Cenário 5. Neste projeto consideramos que a simulação de partidas é volátil. Portanto, não há a necessidade de implementar a persistência de dados para este subsistema visto que apenas o resultado de cada jogador na Partida será guardado e isto pode ficar a cargo da tabela que contém informação do jogador.

Portanto, a persistência de dados foi implementada apenas para os subsistemas SSCatálogos e SSUtilizadores.

5.2 Tabelas

Campeonato(Id, Nome, nomeCircuito)

Circuito(nome, nvoltas, comprimento, ncurvas, nretas, nchicanes)

Secção(Id, tipo, posicao, comprimento, GDU, nomeCircuito)

Piloto(nome, cts, sva)

Utilizador(Email, Password, Nome, Jogador, VersaoJogo)

Carreira(Id, Pontuacao, NomeCampeonato, Email)

Carro(Id, Categoria, Marca, Modelo, Cilindrada, Potencia, Pac, Fiabilidade, Hibrido, PotenciaEletrico, TaxaDeterioracao)

5.3 Diagramas de DAO

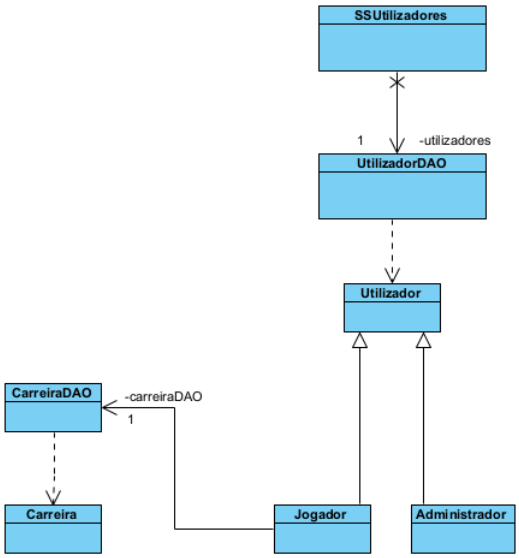

Figura 8: Diagrama DAO - SSUtilizadores

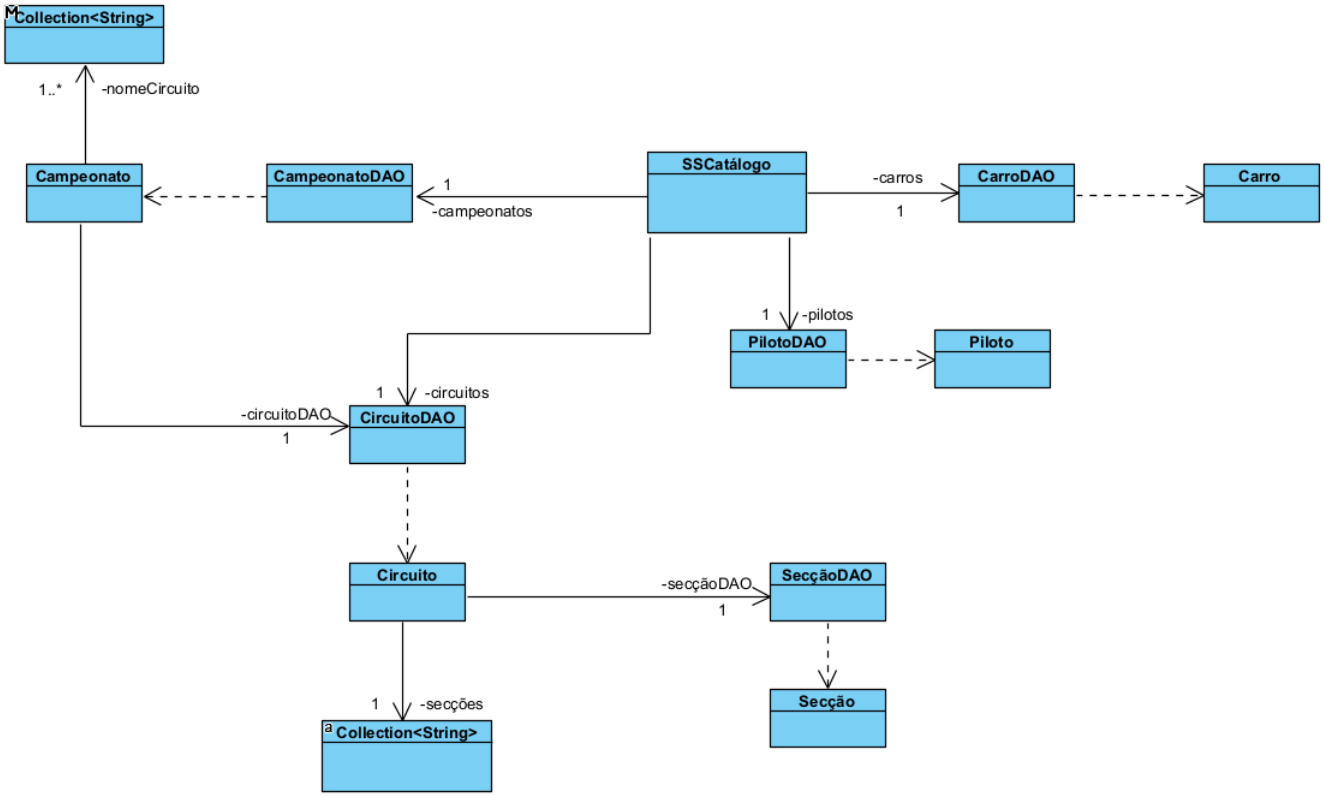

Figura 9: Diagrama DAO - SSCatálogos

6 Máquina de Estados

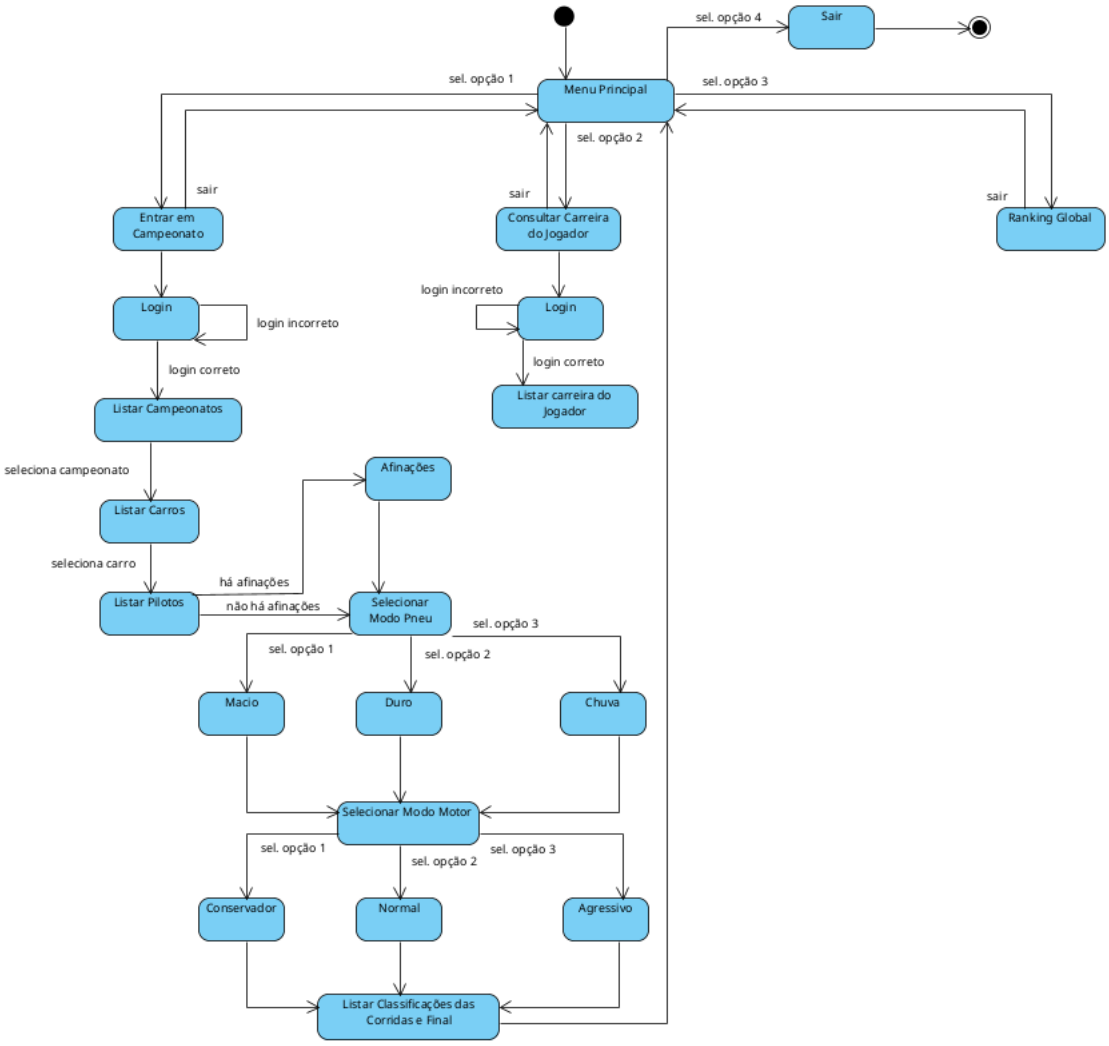

Figura 10: Máquina de Estados para o Menu

Foi desenvolvido este diagrama de máquina de estado da figura 10 para ilustrar como deverá ser o fluxo do Menu a ser implementado na aplicação. É útil pois ajuda a ter uma visão clara e detalhada do passo a passo e facilita o desenvolvimento por parte do programador, bem como reduz ou até mesmo elimina possibilidades de ambiguidade.

7 Visão Geral

No início do projeto foi necessário ler com atenção toda a documentação que especificava a realidade, os requisitos funcionais e os cenários para melhor compreensão do que era desejado. Após uma leitura detalhada, o primeiro passo foi identificar as entidades e desenvolver o Modelo de Domínio que é uma representação da realidade que é relevante para a resolução do projeto apresentado, portanto, ajuda a definir o escopo do problema.

Em seguida, ao identificar os requisitos funcionais, foram desenvolvidos Use Cases relativos aos cenários apresentados e, por fim, o Diagrama de Use Cases. São utilizados durante o processo de análise e projeto de um sistema de software para descrever as funcionalidades do sistema e as interações com os usuários e outros sistemas. Portanto, nos ajudou a entender as necessidades dos usuários e a definir as funcionalidades do sistema de maneira clara e precisa.

Na segunda fase do projeto, começamos por identificar as operações existentes em cada use case de forma a implementar a camada de Lógica de Negócio. Também nos foi fornecido um código legacy, este foi analisado minuciosamente e identificamos o que seria reutilizado e o que não seria, justificando cada decisão. Isto ajudou a ter uma visão mais próxima da implementação e de como esta poderia ser feita, algo que já estávamos mais familiarizados.

Após a identificação das operações, separamos estas em subsistemas pois pertenciam a escopos diferents: SSUtilizadores, SSCatálogos e SSPartidas. Com isto, desenvolvemos o Diagrama de Componentes. Este diagrama permite identificar os componentes do sistema e as interfaces que eles expõem para comunicação com outros componentes, portanto, nos ajudou a visualizar a estrutura do sistema de maneira modular e a perceber como os componentes se comunicam entre si.

Foi então desenvolvido os Diagramas de Classe e de Package. O diagrama de classe é uma representação gráfica da estrutura de uma classe e mostra os atributos e os métodos da classe, bem como as relações entre as classes. O diagrama de package é uma representação gráfica da estrutura de um conjunto de classes agrupadas em um pacote ou namespace. Ele mostra os pacotes e as classes que estão contidos em cada um deles, bem como as relações entre os pacotes. Eles são importantes porque ajudam a entender a organização do código fonte e a identificar as classes e os pacotes que precisam ser desenvolvidos ou modificados.

Sentimos então a necessidade de desenvolver diagramas de atividades de maneira a descrever alguns fluxos pretendidos no projeto. Isto nos ajudou a identificar os passos que são necessários para a realização de uma tarefa e as condições que precisam ser atendidas para que ela seja executada.

Foram desenvolvidos diagramas de sequência para que começássemos a pensar na forma que desejávamos que algumas operações fossem implementadas sem ambiguidade. É utilizado para visualizar e entender como os objetos se comunicam entre si e como são executadas as operações do sistema. Portanto esta fase nos permitiu modelar a lógica das operações, ver como objetos e componentes interagem uns com os outros para concluir uma operação e planear e compreender a funcionalidade detalhada de um cenário existente ou futuro.

Na terceira e última fase do projeto foi necessário analisar e pensar sobre a implementação da persistência de dados. Foram criadas tabelas para estabelecer as relações e os atributos. Em seguida, os diagramas de classe foram adaptados para constar esta persistência de dados. Também foi criado um diagrama de máquina de estados para ilustrar como deveria ser a lógica de navegação do menu. Portanto, ele mostra os diferentes estados que o menu pode ter e as ações que levam de um estado a outro. Este diagrama é útil para entender como um menu funciona e para projetar o comportamento de um menu de forma clara e organizada. Ele também pode ser útil para identificar problemas ou pontos de conflito no comportamento do menu e para resolvê-los antes que o menu seja implementado.

E apenas na última etapa é que foi feita a implementação do código da aplicação.

8 Exemplos de Funcionamento

Nós optamos por recorrer ao *'design pattern'* conhecido como *MVC(Model-View-Controller)*. Assim, este capitulo dedica-se a demonstrar o funcionamento do programa na vista do utilizador.

Para melhor exemplificar estes casos utilizamos a simulação premium, uma vez que nos permite observar melhor as mudanças que ocorrem ao longo da corrida, i.e. as mudanças de posição de uma secção para a seguinte.

8.1 Menu

Menu principal do programa onde o utilizador pode decidir iniciar uma simulação, verificar os rankings de todos os utilizadores ou verificar a sua própria carreira.

```
 *** Racing Manager ***
1 - Entrar em Campeonato
2 - Consultar Carreira do Jogador
3 - Ranking Global
0 - Sair
```

Figura 11: Menu Do Programa

8.2 Ranking Global

Esta opção permite aos utilizadores verificar os rankings globais, utilizando para isso a soma dos pontos de todos os seus campeonatos.

```
 *** Racing Manager ***
1 - Entrar em Campeonato
2 - Consultar Carreira do Jogador
3 - Ranking Global
0 - Sair
Opção: 3

||| RANKING GLOBAL |||

1º: Username: user2 ; Pontuacao: 10

2º: Username: user1 ; Pontuacao: 8
```

Figura 12: Ranking Global

8.3 Consultar Carreira

Operação que permite verificar todos os resultados anteriores de um Jogador nos campeonatos que este participou, sendo que várias participações no mesmo campeonato originam apenas o resultado do melhor dos campeonatos.

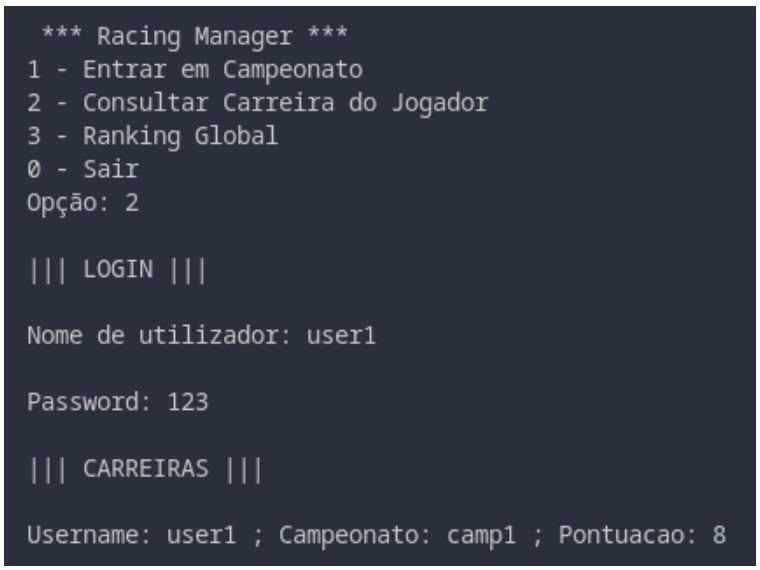

Figura 13: Consultar Carreira de um Jogador

8.4 Configurações

Ao iniciar uma simulação existem vários fatores a considerar por parte do jogador. Dado isso são apresentadas as opções de carros, pilotos, etc. disponíveis para escolha.

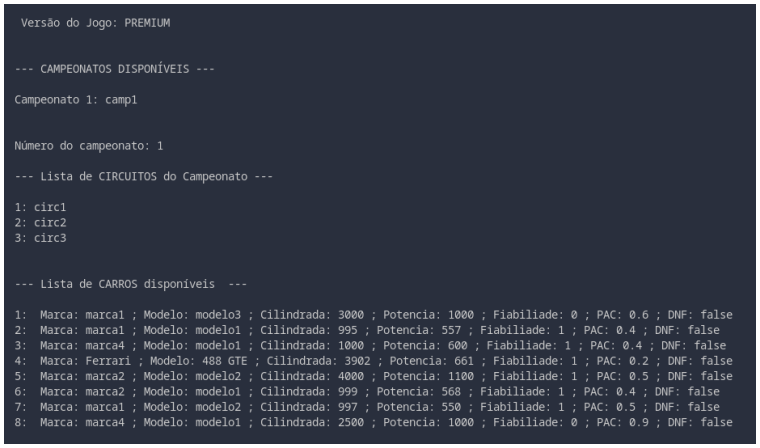

Figura 14: Opções Disponibilizadas

8.5 Ultrapassagem

Durante a simulação de uma corrida existem vários tipos de eventos possíveis que, dependendo da simulação, são demonstrados ao utilizador. O primeiro destes e o mais comum são as ultrapassagens. Estas indicam a visão por parte de ambos os pilotos envolvidos. Abaixo encontra-se o exemplo de uma ultrapassagem onde o piloto 'bot2' que se encontrava atrás na secção 6 conseguiu ultrapassar o piloto 'Charlie Leclerco' na secção 7, ficando com um tempo mais baixo que o anterior.

```
Seccao 6 :
Piloto: Charlie Leclerco; Tempo: 00:07,774
Piloto: bot1; Tempo: 00:02,165
Piloto: bot2; Tempo: 00:12,653

 ---- EVENTOS ------
||| Ultrapassagem[bot2, Charlie Leclerco]
||| Foste ultrapassado[Charlie Leclerco, bot2]
------EVENTOS-------

Seccao 7 :
Piloto: Charlie Leclerco; Tempo: 00:17,681
Piloto: bot1; Tempo: 00:02,601
Piloto: bot2; Tempo: 00:13,095
```

Figura 15: Simulação de Ultrapassagem

8.6 Avaria

Os carros podem ainda sofrer avarias durante a corrida, sendo então incapazes de continuar a mesma. Na imagem a baixo o corredor sofreu uma avaria no carro na secção 13 (última secção do circuito) e, portanto, já não se encontra na dispusta aquando do início da volta seguinte.

```
Seccao 13 :
Piloto: Charlie Leclerco; Tempo: 00:39,258
Piloto: bot1; Tempo: 00:05,515
Piloto: bot2; Tempo: 00:34,179

 ---- EVENTOS ------
||| Avariou[Charlie Leclerco]
------EVENTOS-------

Seccao 0 :
Piloto: bot1; Tempo: 00:05,760
Piloto: bot2; Tempo: 00:34,298
```

Figura 16: Simulação de Avaria

8.7 Avaria com Acidente

Uma avaria pode, para além de tirar da corrida o jogador que a sofre, pode ainda causar acidentes a outros na mesma secção. Isto pode ser verificado no exemplo onde dois corredores acabam por sair da corrida dado a avaria de um deles.

```
Seccao 13 :
Piloto: Charlie Leclerco; Tempo: 00:19,431
Piloto: bot1; Tempo: 00:04,990
Piloto: bot2; Tempo: 00:24,769

 ---- EVENTOS ------
||| Avariou[Charlie Leclerco]
||| Acidente[Charlie Leclerco, bot1]
------EVENTOS-------

Seccao 0 :
Piloto: bot2; Tempo: 00:25,254
```

Figura 17: Simulação de Avaria que gera Acidente

8.8 Acidentes

Os acidentes podem ainda ocorrer por outros motivos, por exemplo a agressividade, ou o estado do tempo. Estes acidentes, assim como as avarias podem gerar acidentes a outros pilotos da corrida. A baixo mostra-se um exemplo de dois acidentes em cadeia logo na primeira secção da corrida.

```
 ---- EVENTOS ------
||| Acidente[Goatifi]
||| Acidente[Goatifi, bot2]
------EVENTOS-------

Seccao 0 :
Piloto: bot1; Tempo: 00:01,345
```

Figura 18: Simulação de Acidente que gera Acidente

8.9 Eventos Base

Na simulação base a visão de eventos é mais limitada, não mostrando os eventos "em tempo real". Isto é, os eventos da corrida são revelados ao utilizador no final de cada volta num agregado de tudo o que aconteceu.

```
 ---- EVENTOS VOLTA 0  ------
||| Ultrapassagem[bot1, Battery Voltar]
||| Foste ultrapassado[Battery Voltar, bot1]
||| Ultrapassagem[bot1, bot2]
||| Foste ultrapassado[bot2, bot1]
||| Ultrapassagem[bot2, Battery Voltar]
||| Foste ultrapassado[Battery Voltar, bot2]
||| Ultrapassagem[Battery Voltar, bot2]
||| Foste ultrapassado[bot2, Battery Voltar]
||| Ultrapassagem[bot2, Battery Voltar]
||| Foste ultrapassado[Battery Voltar, bot2]
||| Ultrapassagem[Battery Voltar, bot2]
||| Foste ultrapassado[bot2, Battery Voltar]
------EVENTOS-------
```

Figura 19: Simulação de Acidente que gera Acidente

8.10 Classificações

No final da simulação são demonstrados ao utilizador os resultados da corrida. As classificações estão separados por carros 'normais' e híbridos. Abaixo encontra-se um log das posições dos carros a cada volta.

```
||||| Classificacoes da corrida |||||
1º: 00:09,025    Categoria: business.subCatálogos.GT     Carro: marcabot1 modelobot1

||||| Classificacoes da corrida Hibridos |||||
1º: 00:21,003    Categoria: business.subCatálogos.GTHibrido      Carro: marca2 modelo2
2º: 00:26,491    Categoria: business.subCatálogos.C2Hibrido      Carro: marcabot2 modelobot2

||||| Primeiro carro a cada volta e desistentes |||||
1ª Volta:
marcabot1 modelobot1
marca2 modelo2
marcabot2 modelobot2
```

Figura 20: Simulação de Acidente que gera Acidente

9 Conclusão e Análise dos resultados obtidos

Este projeto foi com certeza um desafio para o grupo como um todo. É bastante complexo e diferente do que estávamos habituados, visto que é necessário pensar, analisar e modelar o programa antes de o implementar. Contudo, é unânime a opinião de que ajudou a compreender a realidade, comunicar ideias de forma simplificada e documentar de forma mais detalhada e clara. É uma prática que será abordada em todos os projetos daqui para frente.

Ao analisar o resultado final, concluímos que há aspetos que podem ser melhorados. Deveríamos ter feito mais diagramas de sequência de modo a especificar melhor o que cada operação devia fazer e como devia ser feito. Com relação a simulação, gostávamos de ter feito uma interface como seria no mundo real ao invés de recorrer ao terminal. Entretanto, acreditamos que mesmo assim ficou bem detalhada e intuitiva.

Por mais que não tivéssemos conseguido alcançar todos os nossos objetivos, julgamos que o trabalho desenvolvido foi extremamente positivo no âmbito académico. Isto pois fomos capazes de aprender diversos modelos com variadas finalidades, portanto, isto nos ajudará a desenvolver projetos robustos, corretos, mais confiáveis no resto do nosso percurso académico e no mercado de trabalho.

Em conclusão, este trabalho foi muito enriquecedor e nos permitiu adquirir conhecimentos valiosos sobre os modelos e diagramas de desenvolvimento de software, pois conseguimos compreender a importância dos modelos e diagramas no processo de desenvolvimento de software e aplicar esses conhecimentos na desenvolvimento de um projeto. Embora não tenhamos conseguido alcançar todos os objetivos propostos, acreditamos que essa experiência será fundamental para nossa formação e nos ajudará a superar futuros desafios.

10 ANEXOS

10.1 Manual de Utilização

Antes de poder começar a usar a aplicação, deverá confirmar se possui uma conexão *MySQL* ativa. Depois de iniciar o programa, é apresentado um menu com 3 opções:

- **Entrar num Campeonato**: inicia um campeonato e uma simulação de várias corridas, onde escolherá o carro e o piloto a utilizar durante a simulação.
- **Consultar Carreira de um Jogador**: observar as pontuações dos vários jogadores, dos campeonatos que participou.
- **Consultar Ranking Global**: ver o ranking das pontuações de todos os jogadores que estão registados na base de dados da aplicação.

Caso selecione alguma das duas primeiras opções disponíveis, é obrigatório fazer *Login*, indicando o respetivo *username* e *password*. Como o método de registo não foi implementado, dispõe-se de dois utilizadores na base de dados do programa para realizar Login, dependendo da versão de jogo que pretenda utilizar (Base ou Premium):

- **user1** (Versão BASE) - password: 123
- **user2** (Versão PREMIUM) - password: 123

Caso escreva algo incorreto quando pedido, o programa realizará novamente a mesma pergunta e poderá corrigir o(s) valor(es) inseridos. Mas, tenha em atenção que, depois de inserido um valor correto para uma questão do programa, não é possível retroceder na questão. Se desejar corrigir algum valor que tenha selecionado anteriormente, deverá encerrar o programa, reinicializá-lo e escolher as novas opções que deseja colocar/alterar.

Em seguida, basta seguir as indicações do programa para selecionar as opções necessárias para a simulação.