



Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Desenvolvimento de Sistemas de Software

Ano Letivo de 2022/2023

Simulador de Corridas Fase 3 - Implementação da Solução


Grupo 08

a97040 Inês Nogueira Ferreira
a97257 João Miguel Ferreira Loureiro
a91775 José Pedro Batista Fonte
a94942 Miguel Velho Raposo
a94870 Rafael Picão Ferreira Correia

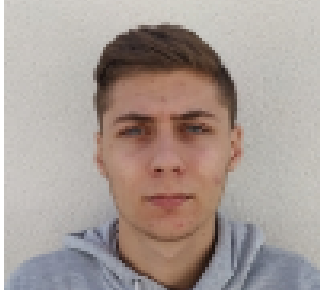
DSS


8 de janeiro de 2023

	Nome	Inês Nogueira Ferreira
	Número	a97040
	GitHub	inesferreira23

	Nome	João Miguel Ferreira Loureiro
	Número	a97257
	GitHub	jmfl27

	Nome	José Pedro Batista Fonte
	Número	a91775
	GitHub	josefonte

	Nome	Miguel Velho Raposo
	Número	a94942
	GitHub	MiguelRaposo

	Nome	Rafael Picão Ferreira Correia
	Número	a94870
	GitHub	rafaelcorreia94870

Resumo

O trabalho desenvolvido no presente relatório visa ao desenvolvimento de um projeto baseado em simuladores de corridas.

A primeira fase é relativa à Análise e Levantamento de Requisitos do Projeto. As ferramentas utilizadas para a identificação de requisitos são os Modelos de Domínio e a construção de Use Cases com a identificação dos seus atores, o modelo geral de use cases e a especificação detalhada dos Use Cases.

A segunda fase é relativa à Modelação Estrutural e Comportamental do Sistema, isto é, a construção de modelos estruturais - através de Diagramas de Componentes e Diagramas de Classes - e de modelos comportamentais - através de Diagramas de Sequência.

A terceira e última fase é relativa à Implementação da Solução, onde foi implementada e desenvolvida uma solução para o projeto proposto, conforme o que foi planeado nas fases anteriores.

Área de Aplicação: Análise e Levantamento de Requisitos, Desenvolvimento de Sistemas de Software, Modelação Estrutural, Modelação Comportamental.

Palavras-Chave: Simulador de Corridas, Diagramas UML, Modelo de Domínio, Modelo de Use Cases, Especificação de Use Cases, API, Lógica de Negócio, Diagramas de Componentes, Diagramas de Classes, Diagramas de Sequência.

 **Link do Repositório do Grupo**

Índice

Lista de Figuras	6
1 Introdução	8
1.1 Contextualização	8
1.2 Caso em Estudo	9
1.3 Objetivos e Motivação	9
1.4 Ferramentas de Desenvolvimento do Software	11
2 Fase 1 - Análise e Levantamento de Requisitos	12
2.1 Modelo de Domínio	12
2.1.1 Descrição do Modelo	12
2.2 Use Cases	13
2.2.1 Identificação dos Atores	13
2.2.2 Modelo dos Use Cases	14
2.2.3 Identificação e Descrição dos Use Cases	14
2.3 Análise dos Use Cases	20
3 Fase 2 - Modelação Estrutural e Comportamental	21
3.1 Modelação Estrutural	22
3.1.1 Diagrama de Componentes	22
3.1.2 Diagrama de Packages	23
3.1.3 Diagrama de Classes	23
3.2 Modelação Comportamental	24
3.2.1 Diagramas de Sequência	24
4 Fase 3 - Implementação da Solução	27
4.1 Alterações feitas às fases anteriores	27
4.2 Estrutura do Código	27
4.3 Implementação da Base de Dados	28
4.4 Manual de Instrução da Interface	30
5 Conclusão	31
6 Anexos	32
6.1 FASE 1 - Análise e Levantamento de Requisitos	32
6.2 FASE 2 - Modelação Estrutural e Comportamental	37
6.2.1 Modelação Estrutural	37

6.2.2	Diagrama de Classes - Em Detalhe	39
6.2.3	Modelação Comportamental	44
6.3	FASE 3 - Implementação da Solução	55

Lista de Figuras

2.1	Modelo de Domínio	12
2.2	Diagrama de Use Cases	14
3.1	Diagrama de Componentes	22
3.2	Diagrama de Packages	23
3.3	Diagrama de Classes	24
4.1	Modelo Relacional da Base de Dados	28
4.2	Diagrama de Classes modificado para incluir os <i>DAO</i>	29
6.1	Diagrama de Use Cases	32
6.2	Modelo de domínio	33
6.3	Parte do modelo de domínio referente ao campeonato	34
6.4	Parte do modelo de domínio referente a uma corridas	34
6.5	Parte do modelo de domínio referente a um carro	35
6.6	Parte do modelo de domínio referente a um piloto	36
6.7	Parte do modelo de domínio referente a um utilizador	36
6.8	Diagrama de Componentes	37
6.9	Diagrama de Classes	38
6.10	Diagrama de Classes do Campeonato	39
6.11	Diagrama de Classes da Corrida	39
6.12	Diagrama de Classes do Circuito	40
6.13	Diagrama de Classes do Participante	40
6.14	Diagrama de Classes do Carro	41
6.15	Diagrama de Classes do Piloto	42
6.16	Diagrama de Classes do Utilizador	42
6.17	Diagrama de Package	43
6.18	Método que adiciona as pontuações de um campeonato ao Ranking global	44
6.19	Método que verifica se um campeonato existe	44
6.20	Método que adiciona Corrida	45
6.21	Método que adiciona Participante ao campeonato	45
6.22	Método que ordena por classificação dos participantes	46
6.23	Método que atualiza o os tempos e pontuações de cada Participante	46
6.24	Método que simula a Campeonato	47
6.25	Método que calcula o numero de afinações máximas e as atribui ao participante	47
6.26	Método que simula a corrida	48
6.27	Método que simula os despistes	48
6.28	Método que simula uma volta	49
6.29	Método que devolve uma lista com as classificações	49
6.30	Método que calcula a distancia dos segmentos	50
6.31	Método que adiciona segmento	50
6.32	Método que adiciona pontuação ao participante	51
6.33	Método que adiciona tempo	51

6.34	Método que calcula a fiabilidade de um carro C1	51
6.35	Método que calcula a fiabilidade de um carro do tipo C2	52
6.36	Método que calcula a fiabilidade de um carro do tipo GT	52
6.37	Método que calcula a fiabilidade de um carro do tipo SC	52
6.38	Método que calcula a potência de um carro do tipo C1	53
6.39	Método que calcula a potência de um carro do tipo C2	53
6.40	Método que calcula a potência de um carro do tipo GT	53
6.41	Método que calcula a potência de um carro do tipo SC	54
6.42	Método que decrementa a Fiabilidade do GT	54
6.43	Método que altera a afinação de um carro	54
6.44	Método que adiciona pontuação ao utilizador	55
6.45	Diagrama de Classes modificado para incluir os <i>DAO</i>	56
6.46	Modelo Relacional da Base de Dados	57

1 Introdução

O presente relatório é referente ao trabalho prático desenvolvido pelo grupo 8 no âmbito da Unidade Curricular de Desenvolvimento de Sistemas de Software, lecionada no curso de Licenciatura em Engenharia Informática no 1º Semestre do ano letivo 2022/2023.

O projeto visa o desenvolvimento de uma aplicação similar aos simuladores de corridas já comercializadas no mercado dos vídeo-jogos.

O trabalho encontra-se dividido em 3 fases:

1. **Análise de Requisitos**
2. **Modelação Conceptual**
3. **Implementação da Solução**

O trabalho apresentado adiante foca-se, primeiramente, numa breve contextualização do tema e na especificação do caso em estudo, assim como os objetivos do grupo e as ferramentas de trabalho a utilizar.

De seguida apresenta-se todo o trabalho referente à primeira fase, nomeadamente, o modelo de domínio com todas as entidades e as relações entre si, os atores do sistema e todos os seus use cases.

Já na segunda fase, irá-se apresentar a arquitetura pensada para o sistema deste projeto, o diagrama de classes e de componentes referentes a esta, tal como os diagramas de sequência que descrevem as várias operações a implementar.

Finalmente, na parte referente à terceira fase, irá-se apresentar a solução final, bem como uma explicação sobre algumas das decisões tomadas com esta.

1.1 Contextualização

Os simuladores são, como o nome indica, um tipo software que permite ao utilizador simular uma atividade em particular. Existem muitos tipos de simuladores, mas os dois tipos mais comuns são aqueles feitos com o intuito de praticar uma determinada habilidade, como os de cirurgias, e, os mais comuns, aqueles que são usados para lazer, como o caso dos de corrida.

Os simuladores já contam com extensos anos de desenvolvimento e os mesmos, em conformidade com a indústria dos video-jogos, sofreram avanços estratosféricos em relação aos motores de física, às interfaces gráficas e às funcionalidades disponíveis. Tais avanços proporcionam uma experiência que simula extraordinariamente bem a realidade, o que se traduz em números de vendas na ordem dos milhões, exemplos disso são o Football Manager e o F1 Manager, que contam com grandes nomes da indústria com estúdios de desenvolvimento dedicados ao lançamento anual dos mesmos.

1.2 Caso em Estudo

O projeto consiste em conceber e implementar um sistema que permita simular campeonatos de automobilismo. Na sua génese a aplicação é similar ao F1 Manager, o que significa que, na verdade, é um jogo onde os utilizadores competem em provas que o software vai simular.

Os simuladores de corrida já existem há muitos anos, e, apesar de, serem introduzidas novas funcionalidades à medida que foram lançadas novas versões, a sua essência manteve-se a mesma: escolhe-se uma série de pilotos, um carro para cada um e uma pista (todos com as suas características únicas), onde depois é simulada uma corrida de **N** voltas onde o resultado depende da simulação dos múltiplos elementos.

O enunciado apresentado também está bastante próximo das funcionalidades mencionadas. Em traços gerais a aplicação funciona do seguinte modo:

O utilizador pode fazer login como **Administrador** ou como **Jogador**. Um **Administrador** pode criar **pilotos, carros, circuitos e campeonatos**, atribuindo a cada um as suas características, que, aquando a simulação, influenciam o resultado final. Um **Jogador** pode configurar campeonatos, configurar corridas e simular as corridas, estas funcionalidades também envolve a escolha de várias propriedades por parte do jogador. O sistema também permite jogadores não autenticados, mas, nesse caso, não usufruem de todas as funcionalidades disponíveis para os jogadores.

1.3 Objetivos e Motivação

Motivação

O grupo encontra-se altamente motivado para cumprir todos os requisitos propostos pelos docentes. Por se tratar de um tema pelo qual existe um interesse pessoal o grupo espera atingir uma série de objetivos traçados.

Objetivos para o Trabalho Geral

- Integrar os conhecimentos adquiridos nas aulas teóricas e práticas no desenvolvimento

do trabalho

- Entregar todos os checkpoints antes da data delineada
- Documentar o trabalho desenvolvido em cada fase
- Implementar todas as funcionalidades descritas
- Implementar uma Interface Gráfica

Objetivos para a 1ª Fase - Análise de Requisitos

- Primeira parte do relatório
- Identificar todas as entidades do Domínio
- Estabelecer as relações corretas entre as entidades
- Identificar todos os atores do Sistema
- Identificar todos os Use Cases associados a cada ator
- Especificar todos os Use Cases identificados
- Analisar criticamente o trabalho desenvolvido

Objetivos para a 2ª Fase - Modelação Estrutural e Comportamental do Sistema

- Segunda parte do relatório
- Identificação dos Métodos e dos Subsistemas através dos Use Cases
- Modelação Estrutural do Sistema - Diagrama de Componentes, Diagramas de Classes
- Modelação Comportamental do Sistema - Diagramas de Sequência

Objetivos para a 3ª Fase - Implementação da Solução

- Terceira parte do relatório
- Implementação de uma Base de Dados e de *DAO's* no projeto
- Atualização do Diagrama de Classes para refletir a implementação de *DAO's*
- Implementação das funcionalidades relacionadas com o "Cenário 5 - Simulação de um campeonato" através da linguagem *Java*
- Conclusões e reflexões finais sobre o trabalho e o produto final

1.4 Ferramentas de Desenvolvimento do Software

Para assegurar as funcionalidades pretendidas, o sistema necessitará do seguinte conjunto de componentes essenciais:

- Modelação e Concepção da Aplicação: Visual Paradigm
- Software de Desenvolvimento: Java, JavaFX/Swing, IntelliJ IDEA
- Software de Gestão do Projeto: Overleaf, Microsoft Office (PowerPoint), GitHub.

- Os jogadores jogam campeonatos compostos por um número de corridas, onde escolhem os pilotos e o seus carros. As corridas são simuladas e no fim de cada corrida atribui-se uma pontuação que contribui para a tabela de classificação de um campeonato.
- Os circuitos são caracterizados por um número de voltas, a distância total e os segmentos de pista (reta, curva, chicane), que definem um grau de dificuldade de ultrapassagem (GDU).
- Os pilotos tem um nome e um nível de perícia, calculado por um fator de capacidade em Tempo Chuvoso vs Tempo Seco (CTS) e fator de Agressividade (SVA).
- Os carros escolhidos são caracterizados por um modelo, uma marca, um perfil aerodinâmico e a categoria onde se inserem (que impõe limites na cilindragem do motor ICE), tendo a particularidade que a categoria GT tem uma taxa de deteiorização. Qualquer carro tem um motor de combustão interna (motor ICE) mas se se inserirem dentro de uma categoria híbrida tem ainda um motor elétrico, que afeta a potência total do carro.
- Antes de cada corrida, o jogador escolhe os pneus do carro e se quer, ou não, fazer uma afinação ao carro (apenas 2/3 das corridas). As afinações permitem alterar o PAC e o modo do motor, o que influencia a performance do carro.

2.2 Use Cases

O foco dos use cases é demonstrar como o sistema implementa as suas ações nucleares. O objetivo não passa por explicar a implementação física mas antes fazer compreender as ideias adjacentes à execução dessas mesmas ações.

2.2.1 Identificação dos Atores

A análise dos cenários possíveis no Modelo de Domínio permite identificar 3 atores:

- **Administrador:** Tem à sua disposição a funcionalidade de criar campeonatos, criar circuitos, criar pilotos e criar carros.
- **Jogador:** Tem as opções de configurar campeonatos, configurar corridas e entrar em campeonatos. No final de cada campeonato os pontos obtidos são guardados na conta do jogador.
- **Convidado:** Tem todas as opções iguais à de um jogador, não podendo, no entanto, configurar campeonatos. No final de cada campeonato se algum utilizador que entrou como convidado, possuir uma conta de jogador, é-lhe dada a opção de acrescentar os pontos que obteve neste campeonato à sua conta de jogador.

2.2.2 Modelo dos Use Cases

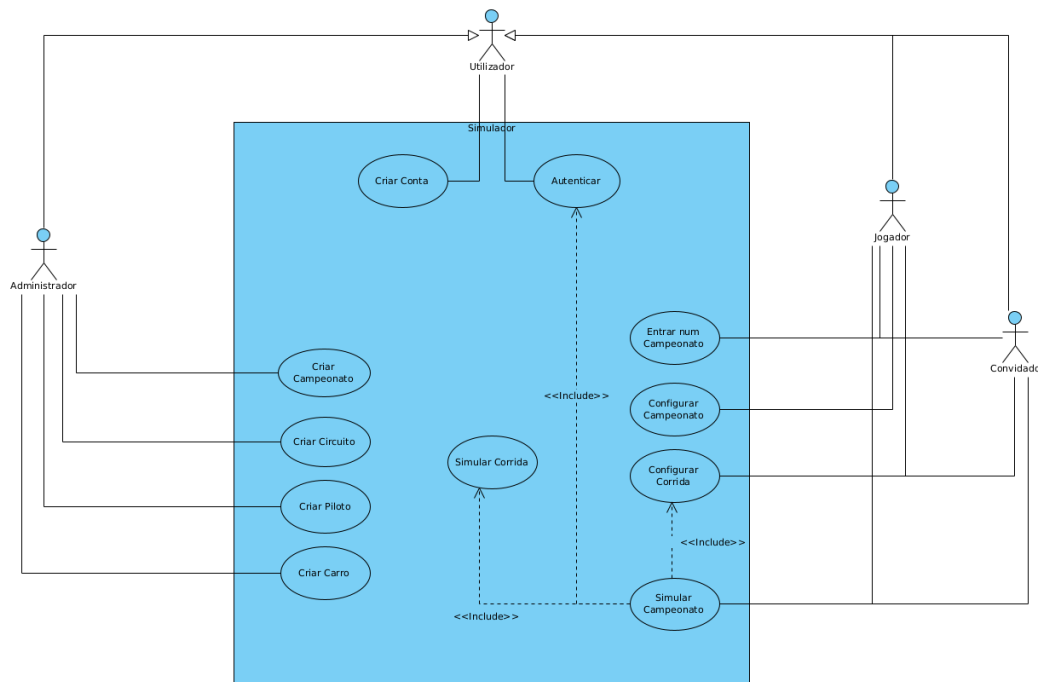


Figura 2.2: Diagrama de Use Cases

2.2.3 Identificação e Descrição dos Use Cases

Abaixo seguem-se todas as descrições de todos os Use Cases do projeto, apresentados no formato de tabela. Cada uma possui o ator do Use Case, uma descrição, a sua pré e pós-condição e os vários fluxos de eventos (normal, alternativo e de exceção).

Criar Campeonato

Cenários:

- Um administrador cria um campeonato.
- Um administrador tenta criar um campeonato com um nome já existente, sendo este descartado.
- Um administrador decidiu não registar o campeonato.

Use Case	Criar um campeonato	
Ator	Administrador	
Descrição	Um administrador cria um campeonato	
Pré-Condição	Estar autenticado no jogo como administrador e existir circuitos disponíveis	
Pós-Condição	Um campeonato novo está disponível	
	Ator	Sistema
Fluxo Normal	1. Administrador indica o nome do novo campeonato a adicionar	2. O nome do novo campeonato é válido.
	3. O administrador escolhe os circuitos que quer que façam parte do campeonato	4. A lista de campeonatos é apresentada
		5. É dada a opção ao administrador de acrescentar o campeonato à lista ou de o descartar
	6. O administrador decide acrescentar o campeonato à lista	
		2.1. O programa avisa o administrador que um campeonato com o mesmo nome
Fluxo de exceção (1) : [Já existe um campeonato com esse nome] (passo 2)		2.2. O sistema termina o processo
Fluxo de exceção (2) : [O administrador decidiu não registar o campeonato] (passo 6)	6.1. O administrador decide descartar o campeonato	

Criar Circuito

Cenários:

- O Vitor faz login como administrador e cria um circuito.
- O Vasco faz login como administrador e tenta criar um circuito com um nome já existente, sendo este descartado.

Use Case	Criar circuito	
Ator	Administrador	
Descrição	Cria um circuito	
Pré-Condição	Estar autenticado no jogo como administrador.	
Pós-Condição	Um circuito novo está disponível para competição.	
	Ator	Sistema
Fluxo Normal	1. Administrador indica o nome do novo circuito a adicionar	2. O nome do novo circuito é válido
	3. O administrador indica o número de curvas.	
	4. O administrador indica o número de chicanes.	
	5. O programa apresenta a distância total mínima.	
	6. O administrador introduz a distância total.	7. O programa apresenta o circuito construído.
	8. O administrador indica o GDU para cada segmento da pista.	
	9. O administrador indica o número de voltas totais.	
		2.1. O programa avisa o administrador que um circuito com o mesmo nome já existe
		2.2. O sistema termina o processo.
Fluxo de Exceção (1) : [Já existe um circuito com esse nome] (passo 2)		

Criar Piloto

Cenários:

- A Carolina faz login como administrador e cria um piloto.
- O Rui faz login como administrador tenta criar um piloto com o mesmo nome de um piloto já existente e o sistema descarta esta tentativa de criação de piloto.

Use Case	Criar piloto	
Ator	Administrador	
Descrição	Cria um piloto	
Pré-Condição	Existir uma conta de administrador e estar autenticado no jogo	
Pós-Condição	O piloto é criado com sucesso e pode depois ser usado nas corridas.	
	Ator	Sistema
Fluxo Normal	1. O administrador escreve o nome do piloto	2. O nome do piloto é válido
	3. O administrador insere o CTS ("Chuva vs. Tempo Seco")	
	4. O administrador insere o SVA ("Segurança vs. Agressividade")	
		2.1. O programa avisa o administrador que um piloto com o mesmo nome já existe.
Fluxo de Exceção (1) : [Já existe um piloto com esse nome] (passo 2)		2.2 O Sistema termina o processo

Criar Carro

Cenários:

- O Rafael faz login como administrador e cria um carro C1 não híbrido.
- O Miguel faz login como administrador e cria um carro C2 não híbrido.
- O José faz login como administrador e cria um carro GT não híbrido.
- O João faz login como administrador e cria um carro SC.
- A Inês faz login como administrador e cria um carro C1 híbrido.
- A Francisca faz login como administrador e cria um carro C2 híbrido.
- A Joana faz login como administrador e cria um carro C2 híbrido.
- A Ana faz login, como administrador e tenta criar um carro C1 mas dá-lhe um valor de cilindrada fora dos parâmetros então o processo falha e ela volta ao menu principal.

Use Case	Criar carro	
Ator	Administrador	
Descrição	É criado um novo carro no sistema	
Pré-Condição	Ator estar autenticado como admin.	
Pós-Condição	O sistema fica com mais um carro disponível para jogar.	
	Ator	Sistema
Fluxo Normal	1. Sistema apresenta categorias disponíveis	
	2. Ator escolhe categoria, marca, modelo, cilindrada e potência	
		3. O sistema verifica se a cilindrada está dentro dos parâmetros para a sua categoria
		4. Sistema verifica que o carro é da categoria C1, C2 ou GT
	5. Ator indica que carro não é híbrido	
	6. Ator indica P.A.C.	
Fluxo Alternativo (1) : [carro é SC] (passo 4)	3.1 Sistema verifica que carro é SC	
Fluxo Alternativo (2) : [carro é híbrido] (passo 5)	6.1 Ator indica que é híbrido e indica potência do motor elétrico	
	6.2 Regressa a 6	
Fluxo alternativo (3) : [A cilindrada está fora dos parâmetros para a sua categoria] (passo 3)		3.1. O sistema informa o utilizador que a cilindrada não está dentro dos parâmetros para a sua categoria
		3.2. O sistema termina o processo

Entrar num Campeonato

Cenários:

- O António, o amigo do Francisco, quer entrar no campeonato que ele configurou, para tal, o António introduz o identificador do campeonato e entra nele.
- O António introduz um identificador inválido e não entra em nenhum campeonato.

Use Case	Entrar num campeonato	
Ator	Jogador \Convidado	
Descrição	Entra num campeonato	
Pré-Condição	Existir campeonato	
Pós-Condição	Jogador entra no campeonato	
	Ator	Sistema
Fluxo Normal	1. O jogador introduz o identificador do campeonato	2. O identificador introduzido é válido
		3. É apresentada a lista de carros da categoria
	4. O jogador escolhe um carro	
		5. É apresentada a lista dos pilotos
	6. O jogador escolhe o piloto	
		7. O jogador encontra-se preparado para o campeonato
Fluxo de exceção (1) [O identificador introduzido não está associado a nenhum campeonato] (passo 2):		2.1. É apresentado uma mensagem de erro sobre o identificador
		2.2. O Sistema termina o processo

Configurar Campeonato

Cenários:

- O Francisco escolhe um campeonato para jogar com os seus amigos

Use Case	Configurar campeonato	
Ator	Jogador	
Descrição	Configura um campeonato para o poder jogar	
Pré-Condição	Existir um campeonato, existem carros e existem pilotos	
Pós-Condição	Uma sala de espera é criada para o Campeonato	
	Ator	Sistema
Fluxo Normal		1. A lista dos campeonatos é apresentada ao jogador
	2. O jogador escolhe o campeonato que quer jogar	
		3. É apresentada a lista dos circuitos
		4. É apresentada a lista de carros
	5. O jogador escolhe um carro	
		6. É apresentada a lista dos pilotos
	7. O jogador escolhe o piloto	
		8. O jogador encontra-se preparado para o campeonato

Configurar Corrida

Cenários:

- O Rafael ao jogar um campeonato está com um carro da categoria C1, depois de uma corrida, ele decide afinar o carro e muda o downforce e escolhe novos pneus.
- O Bernard ao jogar um campeonato está com um carro da categoria GT, ele escolhe os pneus do carro, estando assim pronto para a próxima corrida.
- A Filipa ao jogar um campeonato está com um carro da categoria C2, depois de uma corrida, ela decide não afinar o carro e escolhe novos pneus.
- O Alberto ao jogar um campeonato está com um carro da categoria SC, ele escolhe os pneus do carro, estando assim pronto para a próxima corrida.
- O Luís ao jogar um campeonato está com um carro da categoria C1, após algumas corridas, ele excede o limite de afinações do campeonato, logo ele escolhe os pneus do carro e fica pronto para a próxima corrida.

Use Case	Configurar corrida	
Ator	Jogador \ Convidado	
Descrição	Configurar uma corrida	
Pré-Condição	O campeonato está configurado	
Pós-Condição	O jogador está pronto para simular corrida	
Fluxo Normal	Ator	Sistema
		1. É apresentado o nome do circuito onde se vai realizar a próxima corrida
		2. É apresentada a situação meteorológica
		3. Sistema verifica que o carro é da categoria C1, C2
		4. Sistema verifica se o número de afinações por campeonato já foi excedido
		5. O número de afinações não foi excedido
		6. Sistema dá a opção de fazer a afinação ou não
	7. O utilizador decide fazer a afinação	
	8. O jogador escolhe o downforce e o modo do motor	
	9. O utilizador escolhe os pneus.	
	3.1. O jogador avança para o passo 9	
		5.1 O sistema avisa que o número de afinações foi excedido
		5.2 O sistema avança para o passo 9
	7.1. O jogador escolhe não fazer a afinação	
	7.2. O jogador regressa ao passo 9	

Criar Conta

Cenários:

- O João entra na página de criar uma conta e cria uma conta.
- O Robert por engano, entra na página de criar uma conta em vez de fazer login. Ao introduzir as suas credenciais, aparece um erro a dizer que a conta já existe.
- O Hugo decide criar uma conta, ele escreve uma palavra-passe e ao reintroduzir essa palavra-passe, engana-se. Após tentar criar uma conta, o sistema manda uma mensagem de erro a dizer que a palavra-passe e a confirmação de palavra-passe são diferentes.

Use Case	Criar conta	
Ator	Utilizador	
Descrição	Um utilizador cria uma conta	
Pré-Condição	True	
Pós-Condição	Conta registrada.	
	Ator	Sistema
Fluxo Normal	1. Utilizador introduz o seu nome de jogador	
		2. O sistema verifica a unicidade do nome do jogador
		3. O nome é único
	4. Utilizador introduz a palavra passe	
	5. Utilizador confirma a palavra passe	
		6. É verificado se a palavra passe e a sua confirmação são iguais
Fluxo de exceção (1) : [A conta a tentar ser registada já existe] (passo 3)		3.1 Aparece uma mensagem de erro a dizer que a conta já existe.
		3.2 O Sistema termina o processo
Fluxo de exceção (2): [A confirmação da palavra passe está errada] (passo 6)		6.1. Aparece uma mensagem de erro a dizer que a confirmação e a palavra passe não são iguais.
		6.2 O Sistema termina o processo

Autenticação

Cenários:

- O Francisco autêntica-se para entrar como jogador.
- O Simão ao introduzir a sua palavra passe ou o seu nome engana-se, aparecendo uma mensagem de erro.

Use Case	Autenticação	
Ator	Utilizador	
Descrição	Um utilizador autêntica-se	
Pré-Condição	A conta existe	
Pós-Condição	O utilizador está autenticado	
	Ator	Sistema
Fluxo Normal	1. O utilizador insere o seu nome e a sua palavra passe	
		2. O sistema verifica a palavra passe e o nome
Fluxo de exceção (1): [a palavra passe está errada ou o nome não existe] (passo 2)		2.1. Aparece uma mensagem a dizer que a palavra passe está errada ou o nome não existe
		2.2 O Sistema termina o processo

Simular Campeonato

Cenários:

- O Francisco e a Inês decidem fazer um campeonato entre os dois. Após duas corridas as pontuações de cada um são somadas às pontuações globais e o campeonato termina.
- O Francisco e a Inês decidem fazer um campeonato entre os dois. No final do campeonato a Inês ganhou. Como a Inês entrou no campeonato como convidada, ela decide autenticar-se para adicionar esses pontos à sua conta.
- O Francisco e a Inês decidem fazer um campeonato entre os dois. No final do campeonato a Inês perdeu. Como a Inês entrou no campeonato como convidada, os pontos ganhos nesse campeonato não são somados à sua conta.

Use Case	Simular Campeonato	
Ator	Jogador / Convidado	
Descrição	Jogadores simulam um campeonato	
Pré-Condição	Um jogador ter configurado um campeonato e haver pelo menos mais um jogador pronto para jogar.	
Pós-Condição	Campeonato termina	
	Ator	Sistema
Fluxo Normal	1. <<include>>Configurar corrida	
	2. <<include>>Simular corrida	
		3. O sistema verifica se o campeonato tem mais corridas
		4. O campeonato não tem mais corridas
		5. A pontuação atribuída dos jogadores autenticados é adicionada à pontuação global.
Fluxo alternativo (1) [O campeonato ainda tem mais corridas] (passo 4):		4.1 Volta para o passo 1
Fluxo alternativo (2) [Um dos jogadores está a jogar como convidado](passo 5):	5.2 O convidado escolhe sim	5.1 O sistema pergunta se o convidado quer se autenticar
	5.3 <<include>>Autenticação	
Fluxo alternativo (3) [O convidado escolhe não](passo 5.2)		5.4 Os pontos deste jogador são adicionados à pontuação global
	5.2.1 O convidado escolhe não	

Simular Corrida

Cenários:

- O Francisco e os 3 amigos simulam uma corrida.

Use Case	Simular Corrida	
Ator	Jogador \ Convidado	
Descrição	Simular uma corrida com 1 ou mais jogadores	
Pré-Condição	Os jogadores estão preparados e configurações escolhidas	
Pós-Condição	Classificação dos jogadores e a respectiva pontuação atribuída	
	Ator	Sistema
Fluxo Normal		1. O sistema inicia a corrida com a configurações escolhidas
		2. O sistema simula todos os eventos da corrida
		3. O sistema apresenta a classificação da corrida
		4. O sistema apresenta e atribui a pontuação a todos os jogadores

2.3 Análise dos Use Cases

Um detalhe a notar é o ator **Convidado**. Este é muito semelhante a um **Jogador**, sendo única diferença não estar autenticado, logo não pode executar a opção de **Configurar um Campeonato** nem pode **Acumular Pontos** no fim de uma corrida, e, por consequente, de um campeonato.

Assim, nos Use Cases onde está referido como ator, sempre que um Jogador é mencionado no fluxo, também se refere a um Convidado. Não existem Use Cases diferentes porque seria redundante, no entanto, quando era importante diferenciar os dois (ver **Simular Campeonato**) utiliza-se um Fluxo Alternativo.

3 Fase 2 - Modelação Estrutural e Comportamental

A segunda fase do trabalho foca-se na Modelação Estrutural e Comportamental do Sistema, partindo do trabalho desenvolvido na 1ª Fase do trabalho.

Partindo do processo proposto na aula, o grupo identificou a **API da lógica de negócio e a divisão dos seus subsistemas**. De seguida, definiu a **Arquitetura de cada Subsistema** e o seu **Modelo Comportamental detalhado**. Por último, especificou-se a **interação do utilizador** com tudo o sistema.

Para identificar a API da lógica de negócio o grupo partiu dos use cases criados. Em cada um detetou-se o comportamento da camada lógica e definiu-se os Métodos associados, sendo depois possível agrupá-los por subsistemas de comportamento semelhante.

Com os Subsistemas e os seus Métodos identificados, construi-se um modelo que mostra o relacionamento entre diferentes subsistemas do sistema - **Diagrama de Componentes**. O termo "componente" refere-se a um módulo de classes que representa sistemas ou subsistemas independentes com capacidade de interagir com o restante do sistema.

Já o **Diagrama de Packages** permite gerir mais facilmente as várias classes pertencentes ao sistema, ao agrupar-las em "pacotes", através de divisões lógicas. Permite também identificar as várias dependências entre pacotes e proporcionar um melhor desenvolvimento do sistema.

Dentro de cada módulo de Subsistema, encontra-se o seu **Diagrama de Classes** - que mapeia de forma clara a estrutura do subsistema ao modelar suas classes - *seus atributos e os seus Métodos* - e relações entre objetos. Cada um dos Métodos é detalhado com o seu respetivo **Diagrama de Sequência** - que pode ser descrito como um diagrama de interação, pois descreve como, e em qual ordem, um grupo de objetos trabalha em conjunto- especificando as mensagens que mandam entre si, os métodos que utilizam, etc.

3.1 Modelação Estrutural

3.1.1 Diagrama de Componentes

O grupo interpretou o campeonato como um conjunto de corridas, por sua vez compostas pelo seu circuito, carros e pilotos, desse modo cria-se a associação de subsistemas aninhados em outros subsistemas. Como o utilizador também tem acesso aos dados da sua conta de utilizador, consideramos-lo um subsistema separado. Outro modo de utilização do sistema é como Administrador, e desse modo o utilizador tem de ter acesso direto aos subsistemas piloto, carro e utilizador de forma separada. Assim sendo, o grupo construiu o diagrama de componentes da seguinte forma.

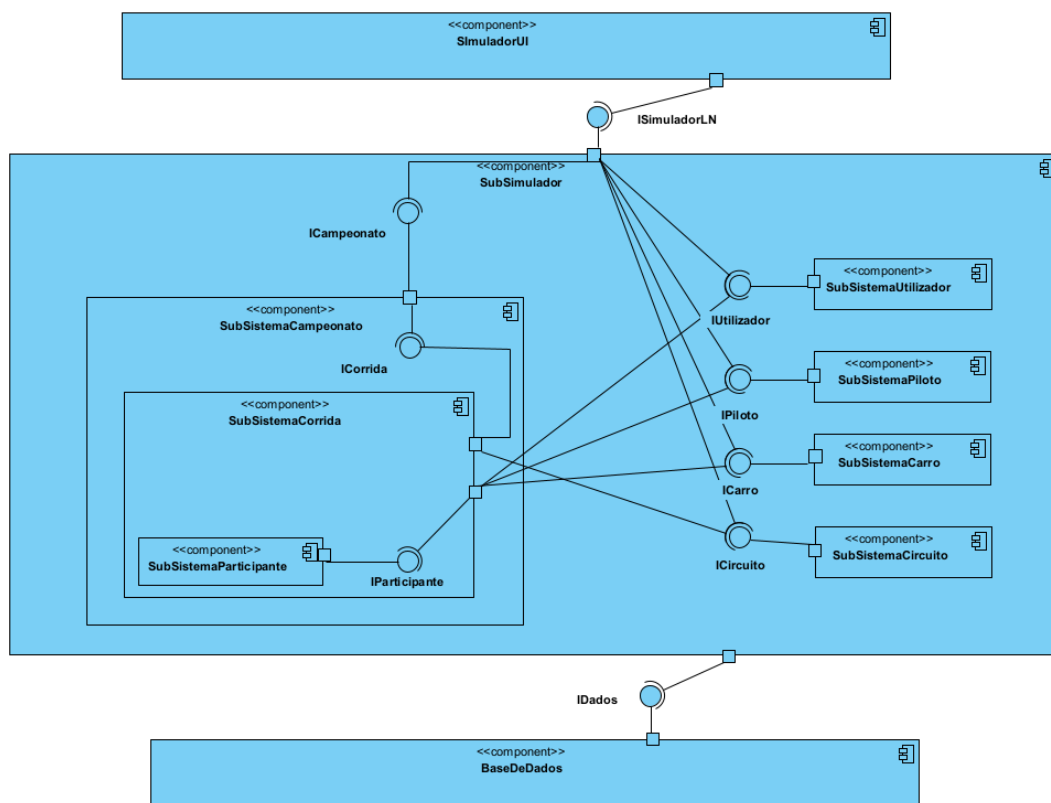


Figura 3.1: Diagrama de Componentes

3.1.2 Diagrama de Packages

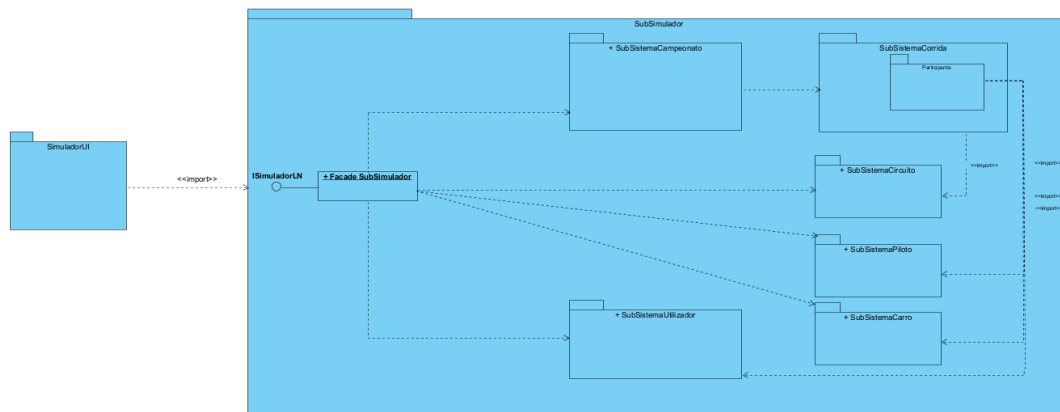


Figura 3.2: Diagrama de Packages

Decidimos dividir as classes em packages de acordo com os subsistemas presentes no Diagrama de Componentes, seguindo a sua lógica.

O package **SimuladorUI**, como elemento "central" do programa, importa todas as outras classes, sendo estas todas incluídas no package **SubSimulador**. Esta, por sua vez, engloba os packages relacionados com a gestão dos campeonatos e dos utilizadores (**SubSistemaCampeonato** e **SubSistemaUtilizador**, assim como, **SubSistemaCircuito**, **SubSistemaPiloto**, **SubSistemaCarro**).

Por último, o Package **SubSistemaCorrida** responsável por todas as classes cujos métodos estão relacionados com a simulação de uma corrida importa o package do **SubSistemaCircuito** e o **SubSistemaParticipante** importa o Package **SubSistemaCarro**, o Package **SubSistemaPiloto** e o o Package **SubSistemaUtilizador**.

3.1.3 Diagrama de Classes

O Diagrama de Classes apresenta 12 classes diferentes que modelam o sistema. Além das relações estabelecidas, dos atributos e dos métodos de cada classe é importante esclarecer algumas decisões tomadas pelo grupo.

- A classe **CARRO** é uma classe abstrata que se estende para 4 classes (4 categorias), que implementam os métodos característicos da sua classe.
- A classe **Corrida** estende a superclasse **Circuito** e adiciona alguns atributos específicos como o número de voltas e o clima.

- [illegible]

- Classificação Final
 - Atualiza Participantes
 - Simula Campeonato
 - Calcula número de Afições Totais
- Métodos da Classe Corrida
 - Simula Corrida
 - Simula Despiste
 - Simula Volta
 - Lista de Classificação
- Métodos da Classe Circuito
 - Adiciona Segmento
 - Calcula Segmento
- Métodos da Classe Participante
 - Adiciona Pontuação
 - Adiciona Tempo
- Métodos da Classe Carro
 - Calcula Fiabilidade C1
 - Calcula Fiabilidade C2
 - Calcula Fiabilidade GT
 - Calcula Fiabilidade SC
 - Calcula Potencia C1
 - Calcula Potencia C2
 - Calcula Potencia GT
 - Calcula Potencia SC
 - Decrementa Fiabilidade GT

- Altera Afinação
- Métodos da Classe Utilizador
 - Adiciona Pontuação ao Utilizador

4 Fase 3 - Implementação da Solução

A terceira, e última, fase deste projeto consiste na Implementação da Solução, sendo o culminar deste projeto inteiro.

Seguindo o que foi estipulado nas fases anteriores, desenvolvemos a componente do programa referente ao "**Cenário 5 - Jogar**" (Simular um Campeonato), como recomendado pelos os docentes. Modificamos também o **Diagrama de Classes**, para ter em conta os **DAO's**, vitais à implementação de uma base de dados no projeto.

Assim, nesta parte do relatório iremos apresentar e justificar as decisões tomadas aquando da implementação do sistema, nomeadamente em relação à estrutura do código, implementação da Base de Dados e, consequentemente, dos DAO's e um manual de instruções para utilizar o produto final.

4.1 Alterações feitas às fases anteriores

Sendo esta a fase final do projeto, fizemos algumas correções de erros presentes em alguns diagramas das fases anteriores que surgiram na implementação da solução:

- foi alterado o **Diagrama de Classes** de forma a eliminar a lista de Utilizadores RankingGlobal, devido á sua redundância.

4.2 Estrutura do Código

O código está dividido em várias classes, cujas variantes e métodos refletem o que foi estipulado pelo o **Diagrama de Classes**. Por sua vez, os métodos relevantes definidos na fase anterior, foram modelados conforme os seus **Diagramas de Sequências**.

Assim, usaremos esta secção para justificar algumas decisões ao nível de código.

A classe **Model** é a base da implementação inteira, sendo nela que são guardados os objetos essenciais à simulação de um campeonato, através de *Maps*, cada um correspondente a um

objeto diferente, onde a chave de uma entrada é o identificador do objeto a guardar.

As restantes classes, em termos de variáveis e métodos, seguem o que foi definido nas fases anteriores, sendo portanto auto-explicativas. É de notar a introdução de classes *DAO*, que serão explicadas com maior detalhe na próxima secção.

O método mais importante, em relação ao cenário 5, é o método **simulaCampeonato**, que como o nome indica, simula um campeonato, através da simulação das corridas que o compõe, apresentando os resultados finais do campeonato, bem como os intermédios das corridas. De forma a calcular os resultados, são comparados as várias características dos participantes, conforme defendido no enunciado.

4.3 Implementação da Base de Dados

Uma das funcionalidades chave deste projeto é a criação de vários objetos (utilizadores, pilotos, carros, etc...) e o seu armazenamento para posterior uso. Para tal efeito, é necessário utilizar uma **Base de Dados**, tal como a API **Java DataBase Connectivity (JDBC)**, que juntamente com os **Data Access Objects (DAO)**, permitem a uma aplicação *Java* comunicar com uma dada base de dados.

Para tal efeito, utilizamos a aplicação MariaDB para criar e gerir a Base de Dados, onde fizemos um **Modelo Relacional** de forma a estruturar e relacionar as várias entidades que irão ser armazenadas.

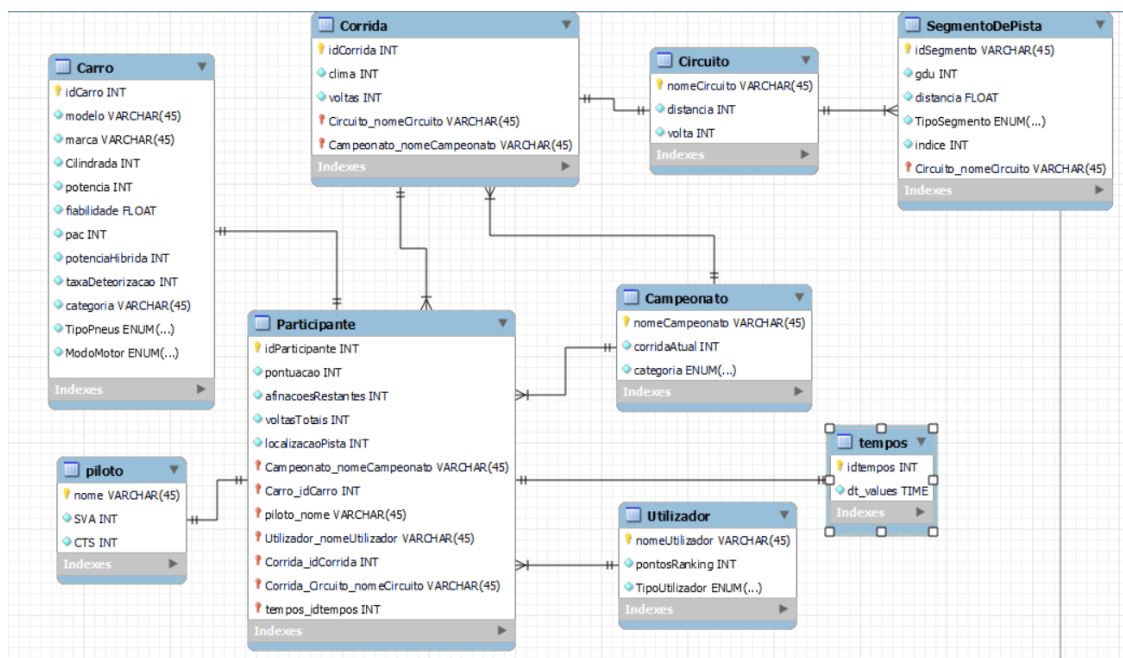
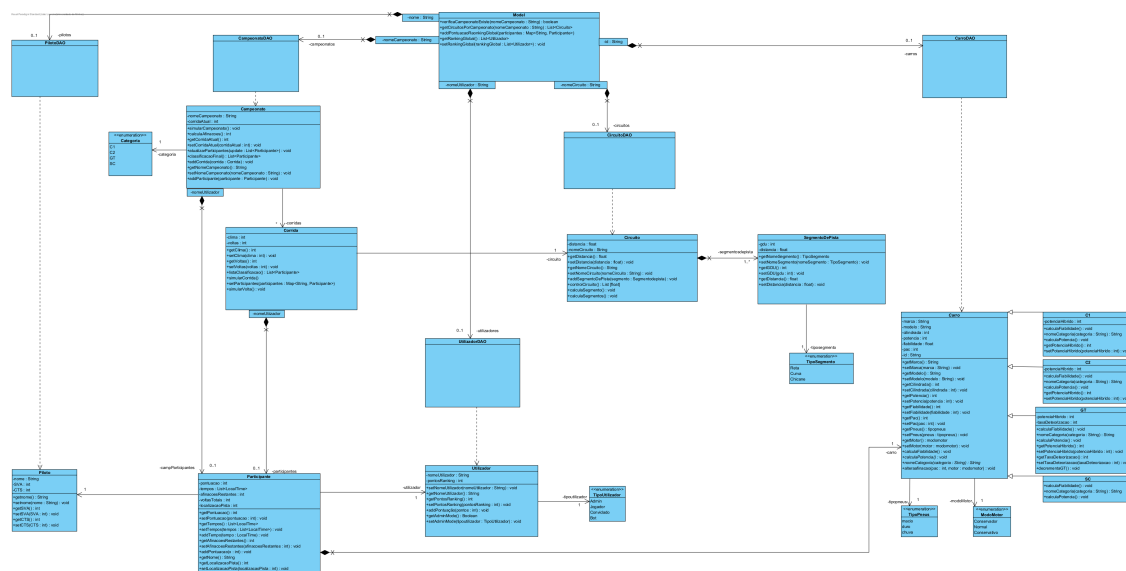


Figura 4.1: Modelo Relacional da Base de Dados

Cada uma das tabelas representa um objeto de tipo diferente cuja informação deve ser armazenada na base de dados, representando cada entrada um dos atributos da classe do respectivo objeto a armazenar.

Tivemos também que modificar as classes relevantes para suportar *DAO's*, que, essencialmente, armazenam informação na Base de Dados e criam objetos no programa a partir da informação nela presente, podendo também fazer *queries SQL*.



Existem classes *DAO* para Campeonato, Carro, Circuito, Piloto e Utilizador, ou seja, as entidades que o utilizador do programa pode criar para utilização no jogo.

Por exemplo, após a criação de um Piloto, este tem a sua informação desconstruída e devidamente inserida na sua tabela correspondente na base de dados, através do método *put*

da *DAO*. Quando é preciso carregar um objeto, o método *get* faz uma query SQL à base de dados, criando um objeto a partir do resultado desta, que é então guardado em memória.

4.4 Manual de Instrução da Interface

Nesta secção, elaboramos um pequeno manual de instruções que explica o comportamento do programa, de modo a proporcionar uma utilização o sistema sem grandes constrangimentos.

Ao iniciar o programa pela primeira vez, o utilizador irá se deparar com uma interface gráfica, o menu, no terminal de onde correu o programa. Inserindo o algarismo da opção correspondente, o utilizador pode seleccionar o modo Admin ou o modo Jogador.

Selecionado o Admin, pode escolher editar Campeonatos, Circuitos, Pilotos ou Carros, seleccionando a opção que desejar. Feita a escolha, pode então decidir Consultar, Adicionar ou Remover a entidade.

Ao Adicionar, o utilizador insere parâmetro a parâmetro o que deseja da entidade seleccionada, podendo escrever *STOP* a qualquer momento para interromper o processo. Se escolher Consultar, é-lhe apresentado o nome de todas as entidades, onde o utilizador pode inserir o nome da que pretende para visualizar os seus detalhes. O Remover segue a mesma lógica do Consultar, sendo a única diferença que o utilizador insere o nome do que pretende remover.

Selecionando o Jogador, o utilizador escolhe as suas definições para a corrida, onde-lhe é então apresentado a simulação da corrida.

5 Conclusão

Com o desenvolvimento deste trabalho prático, conseguimos aplicar e consolidar os conhecimentos abordados na Unidade Curricular de Desenvolvimento de Sistemas de Software, nomeadamente o extensivo planeamento de um projeto através dos vários tipos de diagramas e modelos UML que foram utilizados ao longo das 3 fases.

Nesta fase final conseguimos consolidar, mais concretamente, a utilização de *DAO's* e de Base de Dados na implementação de um sistema em *Java*, bem como o seu desenvolvimento a partir do que foi estabelecido nas fases anteriores.

Apesar das nossas melhores tentativas, não conseguimos por a simulação de um campeonato a funcionar, estando o problema relacionado com o *backend* que não conseguimos identificar. Alguns *DAO* também não estão a funcionar devido a problemas na syntax do *SQL* que não conseguimos identificar a tempo da entrega. Não estamos, portanto, satisfeitos com o resultado final da implementação, mas satisfeitos com o resultado das fases anteriores.

Com base no que foi estabelecido na fase de modelação deste projeto (Fase 2), estamos confiantes que é possível implementar com sucesso a maioria das funcionalidades estabelecidas na fase de requisitos (Fase 1) que estão em falta, dada uma fase de implementação (Fase 3) mais longa.

Assim, o grupo fecha este projeto satisfeito, em geral, com o que foi realizado ao longo deste, que permitiu desenvolver as nossas competências de planeamento de projetos em grande escala, principalmente através do uso de diagramas UML, ferramentas úteis que certamente serão importantes no nosso futuro, tanto académico como profissional, também como um conhecimento mais aprofundado da linguagem *Java* e como utilizar uma base de dados num projeto nela feito.

6 Anexos

6.1 FASE 1 - Análise e Levantamento de Requisitos

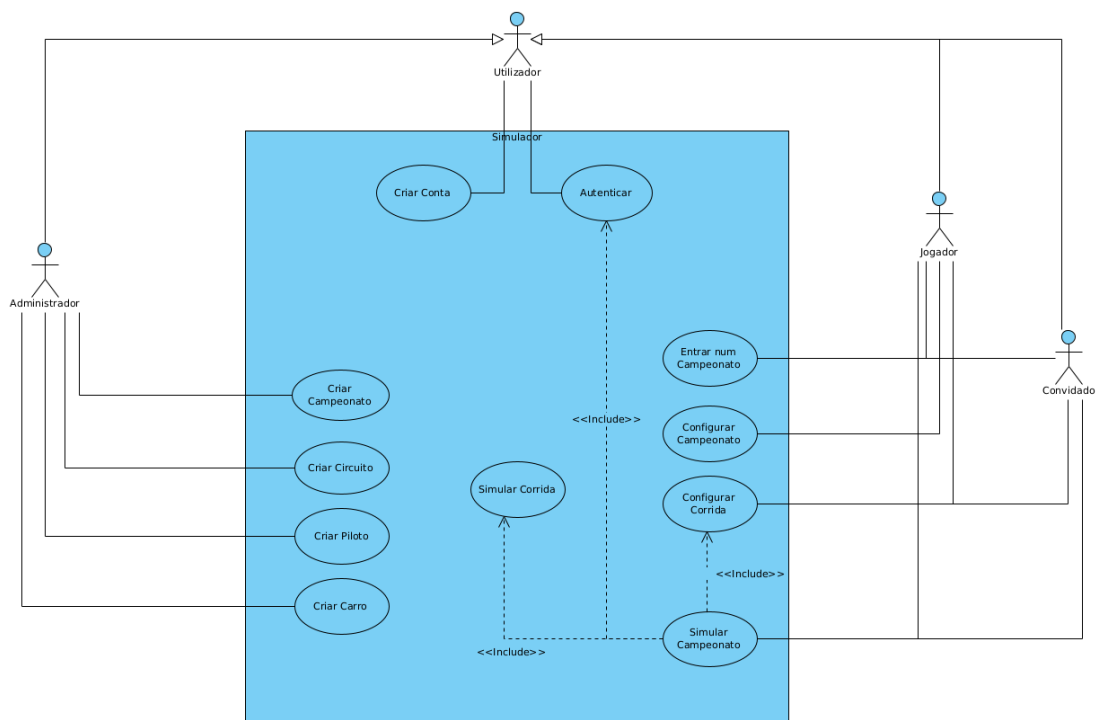


Figura 6.1: Diagrama de Use Cases



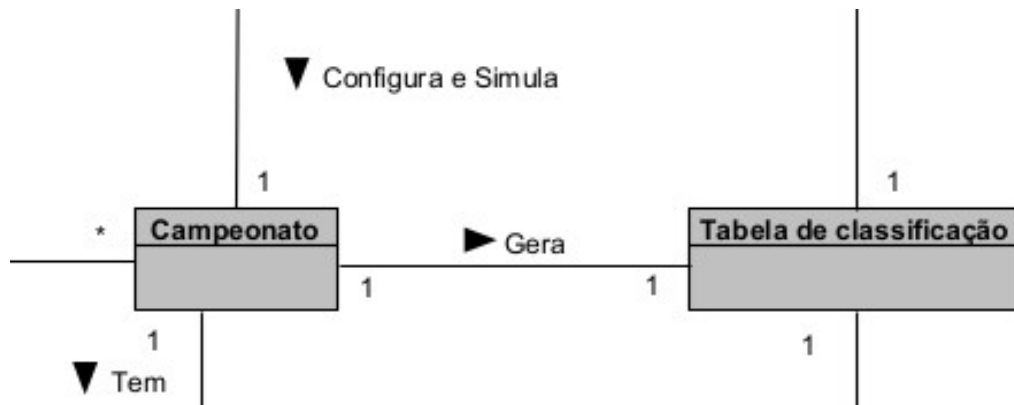


Figura 6.3: Parte do modelo de domínio referente ao campeonato

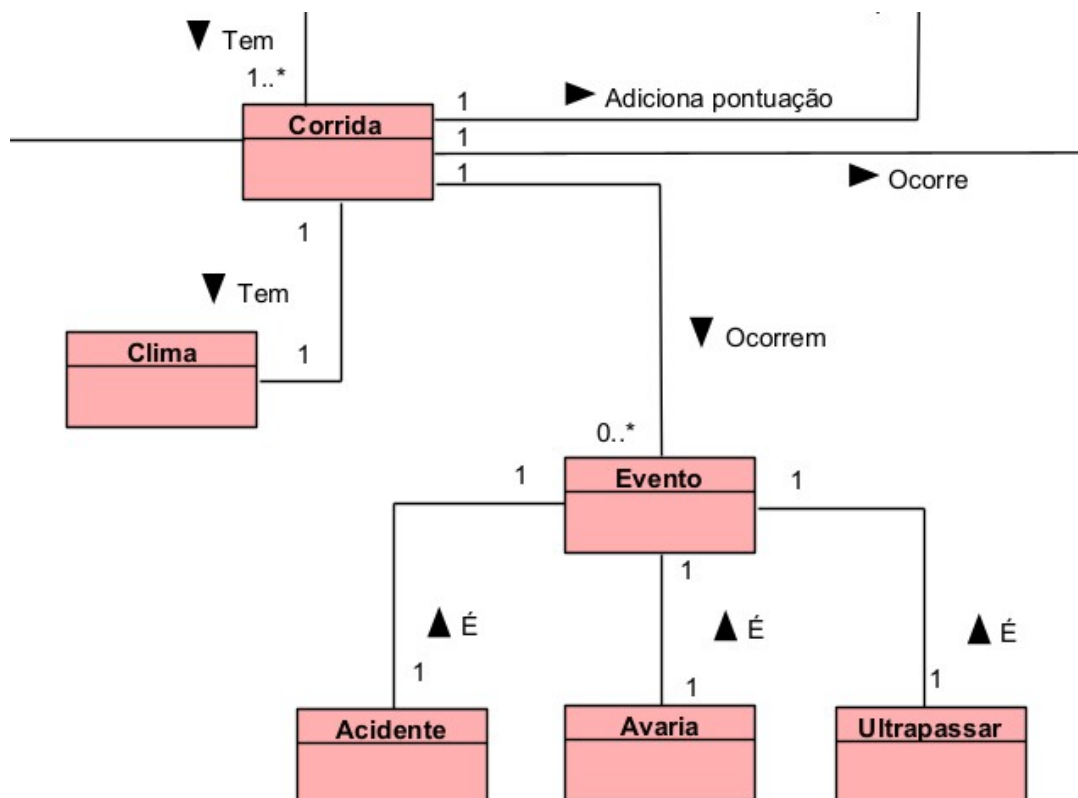


Figura 6.4: Parte do modelo de domínio referente a uma corridas

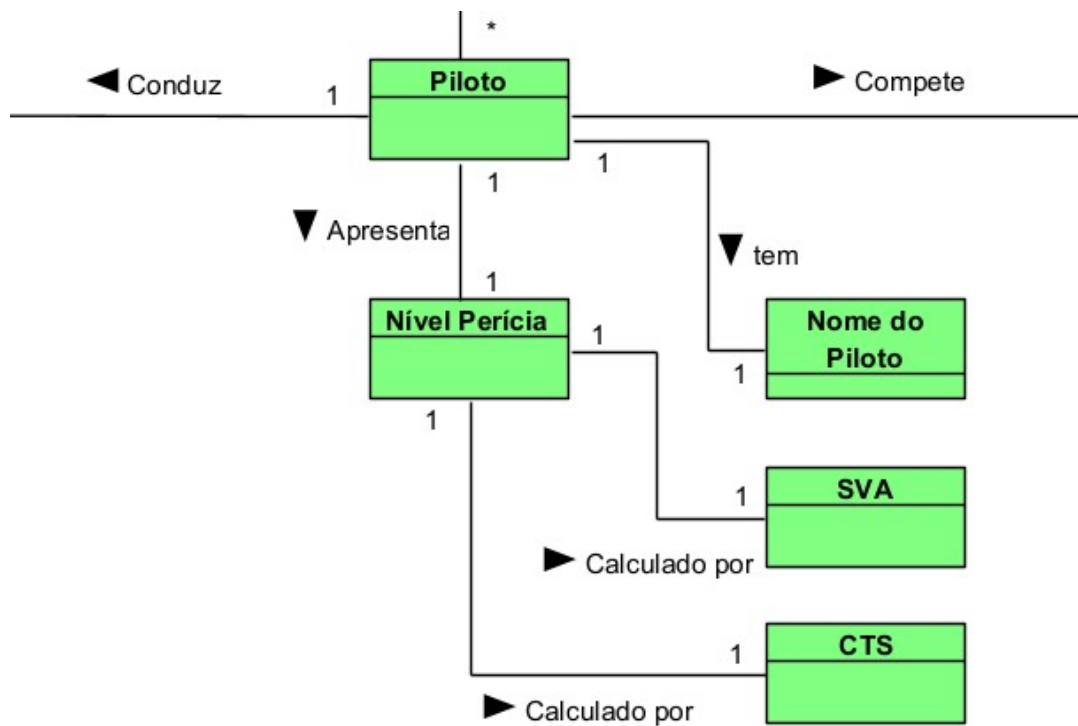


Figura 6.6: Parte do modelo de domínio referente a um piloto

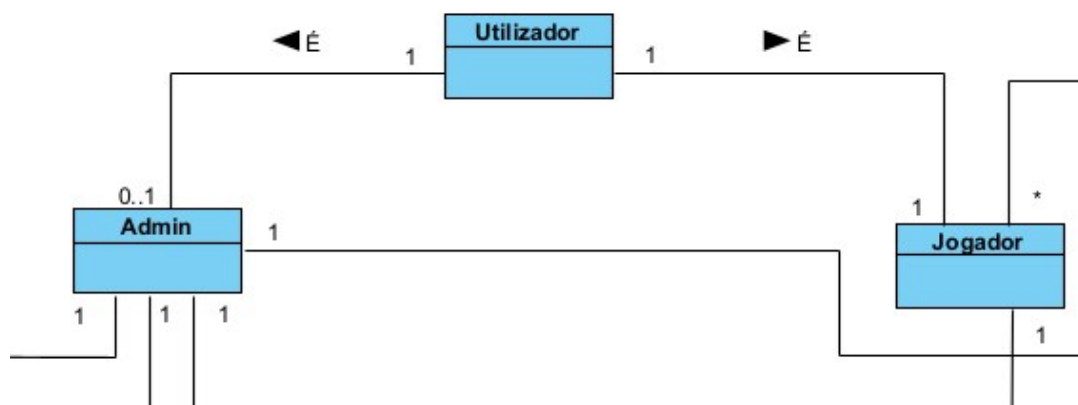


Figura 6.7: Parte do modelo de domínio referente a um utilizador

6.2 FASE 2 - Modelação Estrutural e Comportamental

6.2.1 Modelação Estrutural

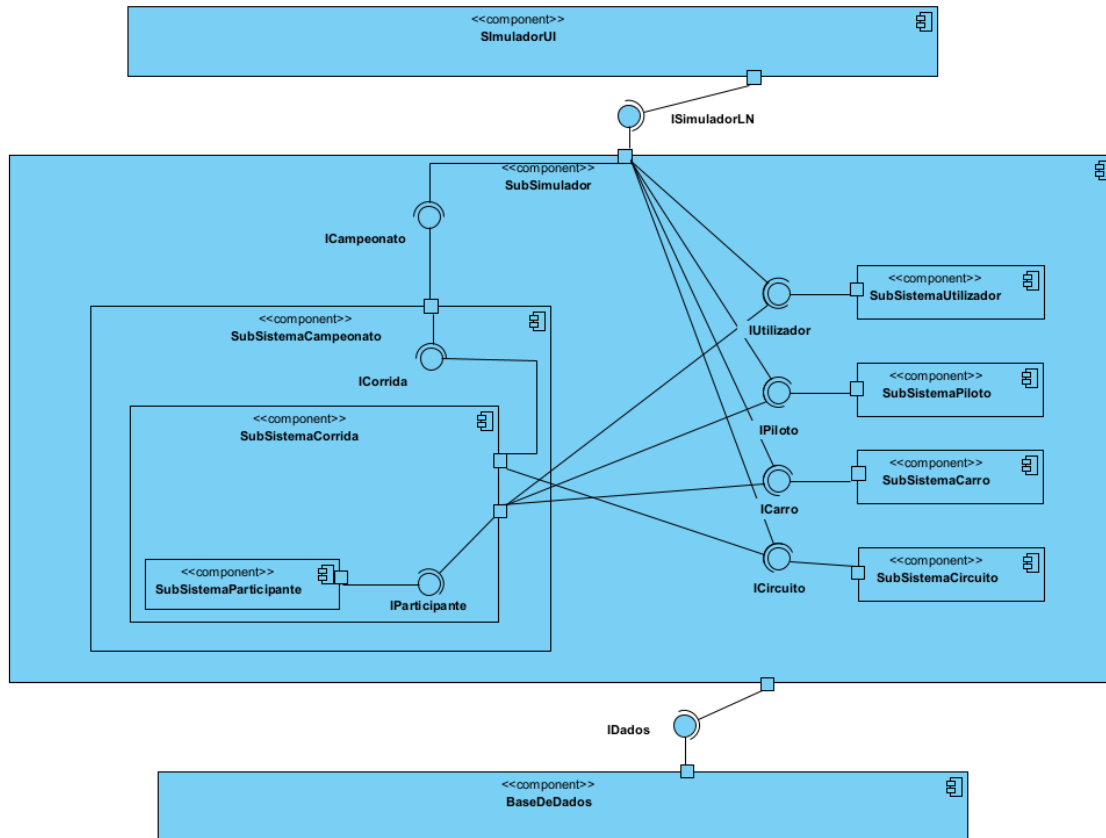


Figura 6.8: Diagrama de Componentes

Figura 6.9: Diagrama de Classes

6.2.2 Diagrama de Classes - Em Detalhe

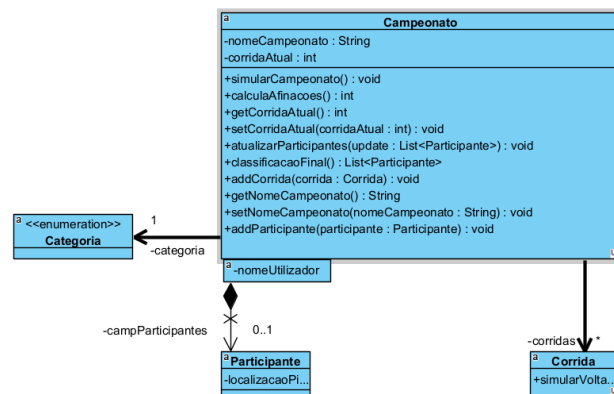


Figura 6.10: Diagrama de Classes do Campeonato

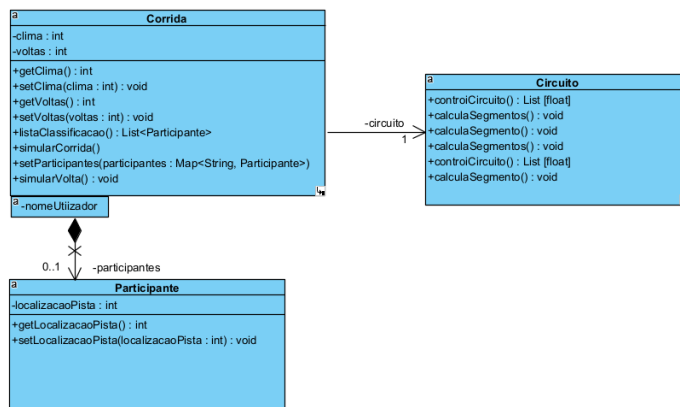


Figura 6.11: Diagrama de Classes da Corrida

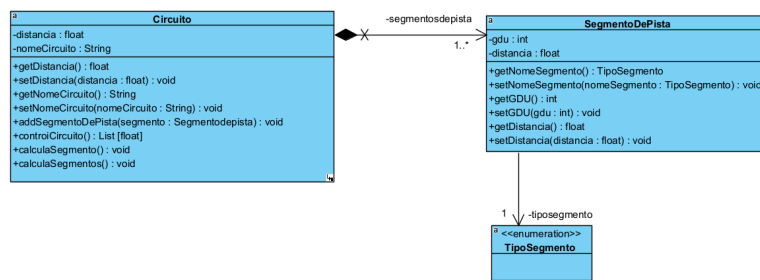


Figura 6.12: Diagrama de Classes do Circuito

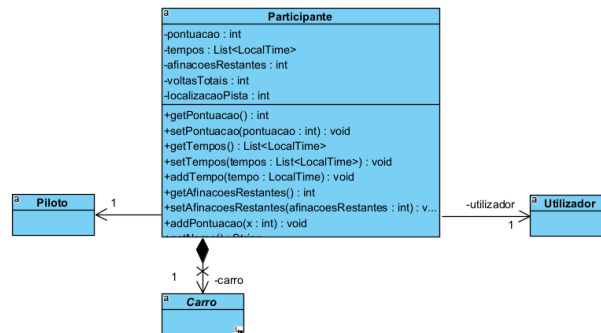


Figura 6.13: Diagrama de Classes do Participante

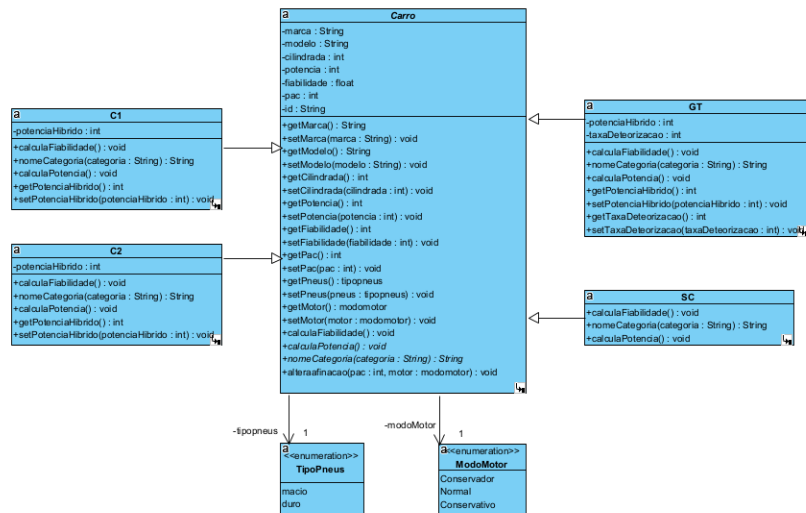


Figura 6.14: Diagrama de Classes do Carro

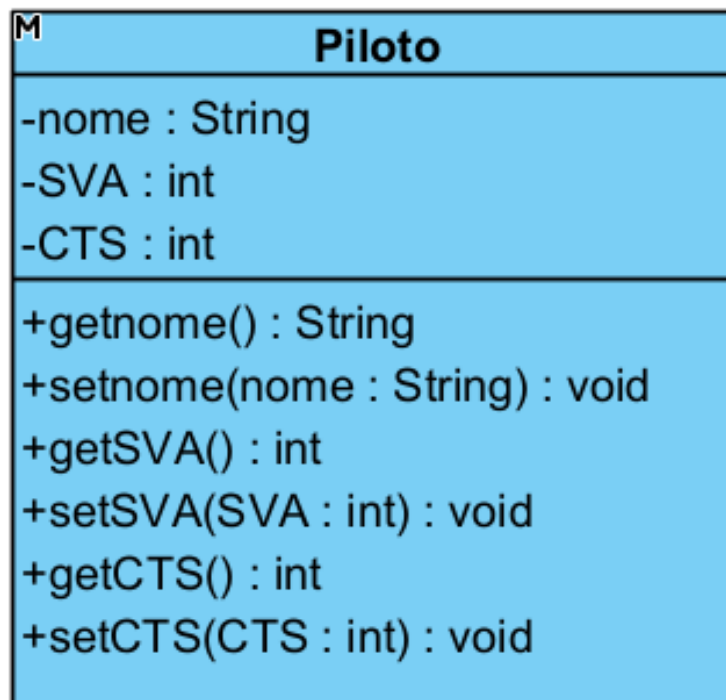


Figura 6.15: Diagrama de Classes do Piloto

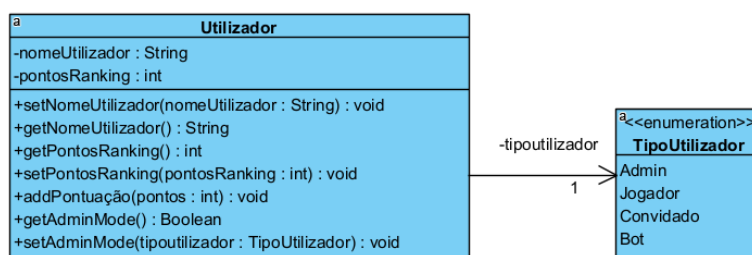


Figura 6.16: Diagrama de Classes do Utilizador

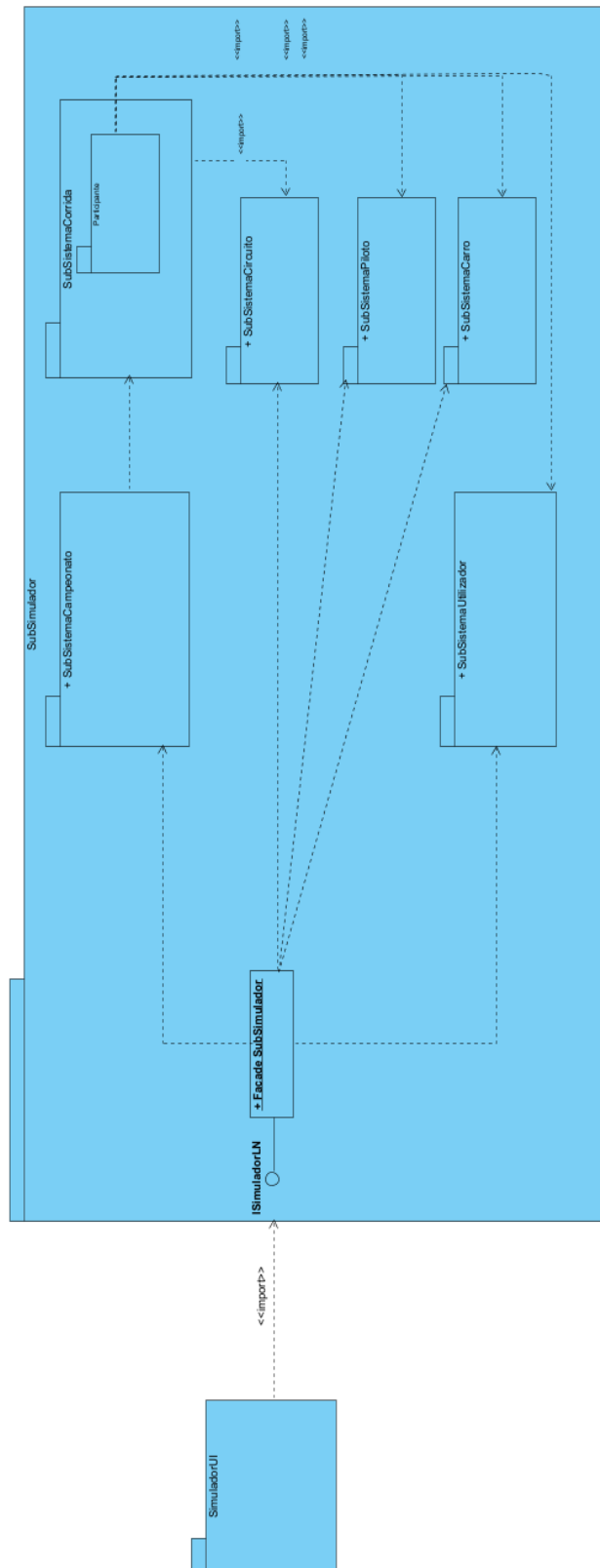


Figura 6.17: Diagrama de Package

6.2.3 Modelação Comportamental

Classe Model

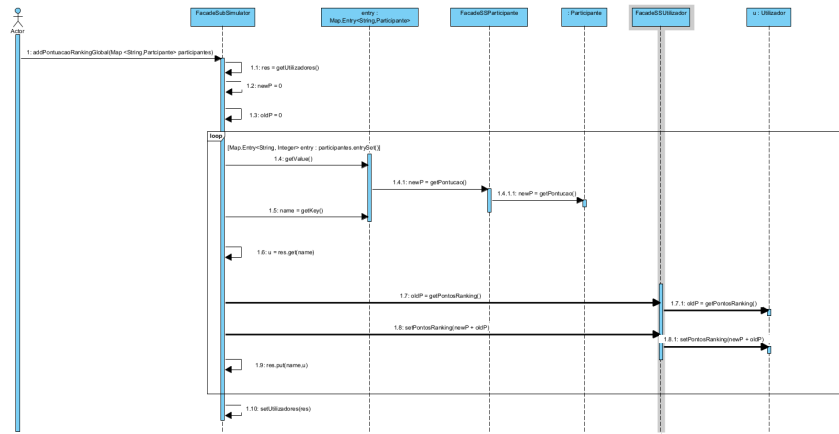


Figura 6.18: Método que adiciona as pontuações de um campeonato ao Ranking global

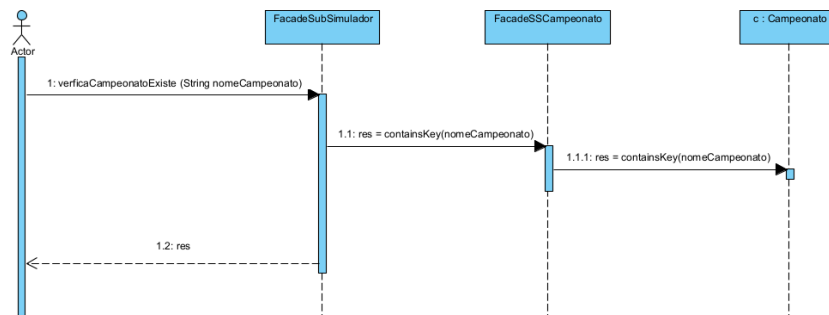


Figura 6.19: Método que verifica se um campeonato existe

Classe Campeonato

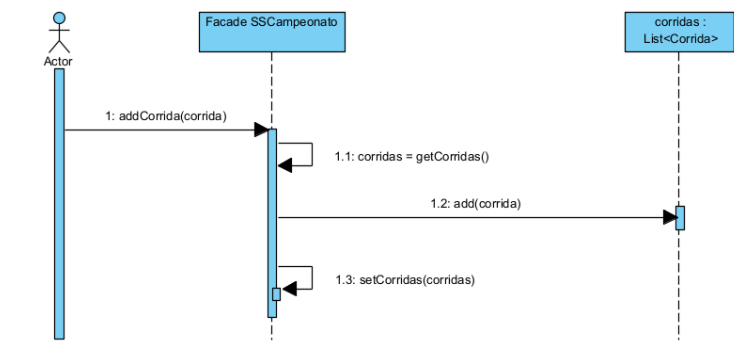


Figura 6.20: Método que adiciona Corrida

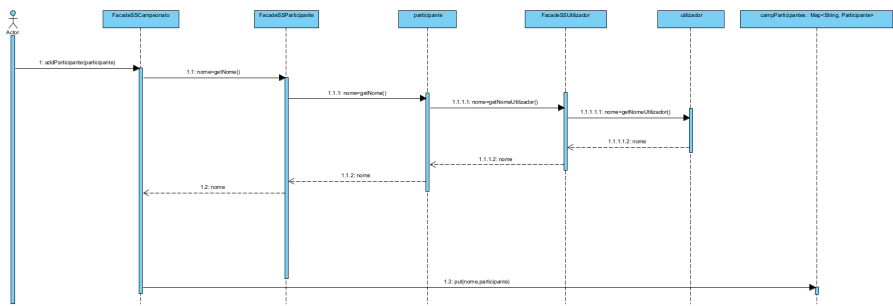


Figura 6.21: Método que adiciona Participante ao campeonato

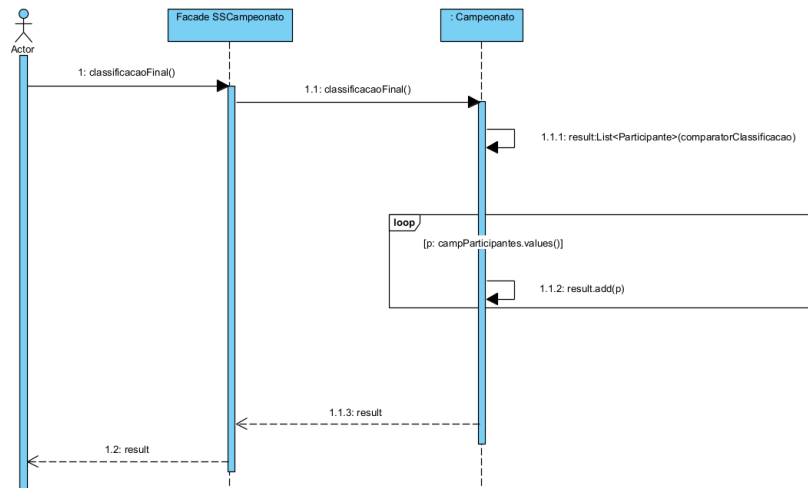


Figura 6.22: Método que ordena por classificação dos participantes

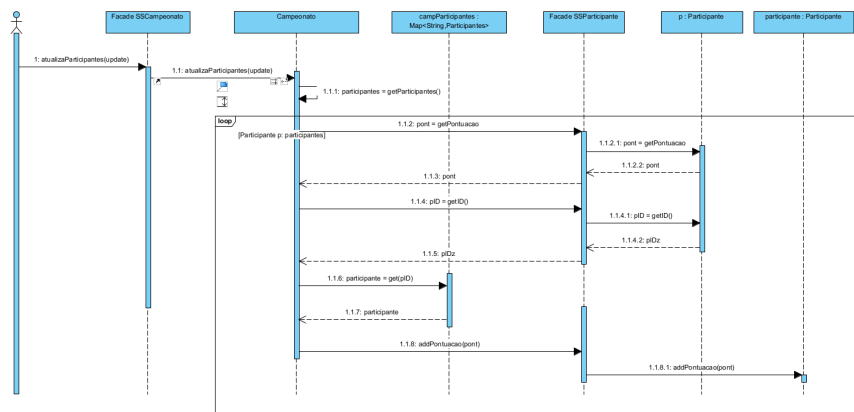
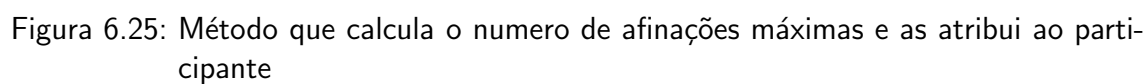
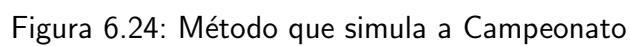


Figura 6.23: Método que atualiza o os tempos e pontuações de cada Participante



Classe Corrida

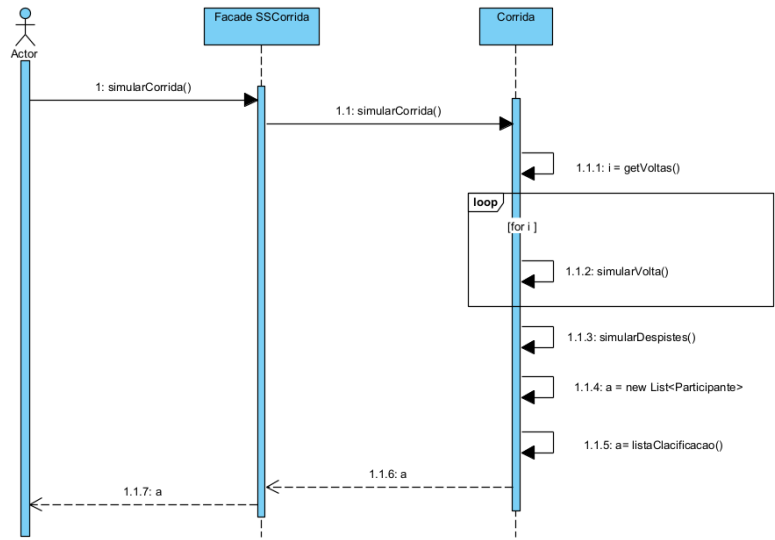


Figura 6.26: Método que simula a corrida

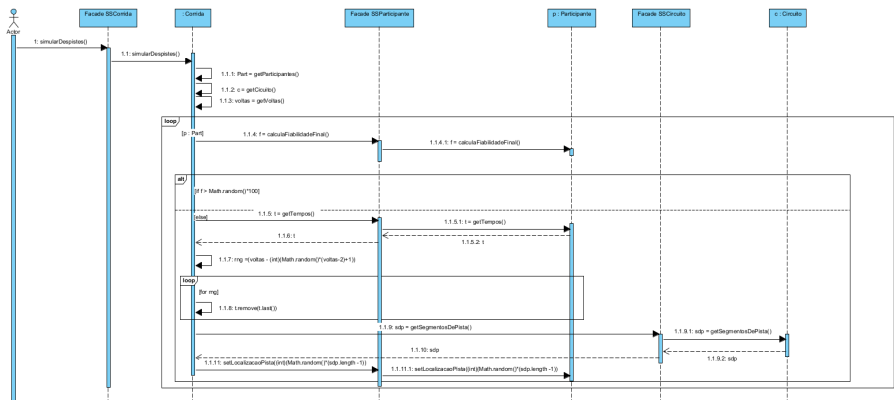


Figura 6.27: Método que simula os despistes

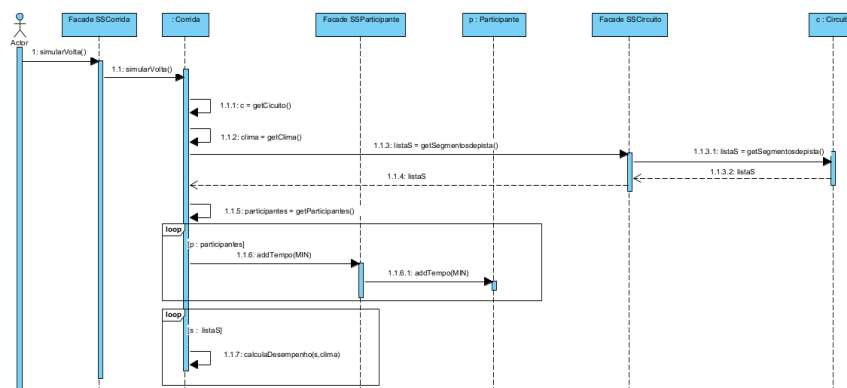


Figura 6.28: Método que simula uma volta

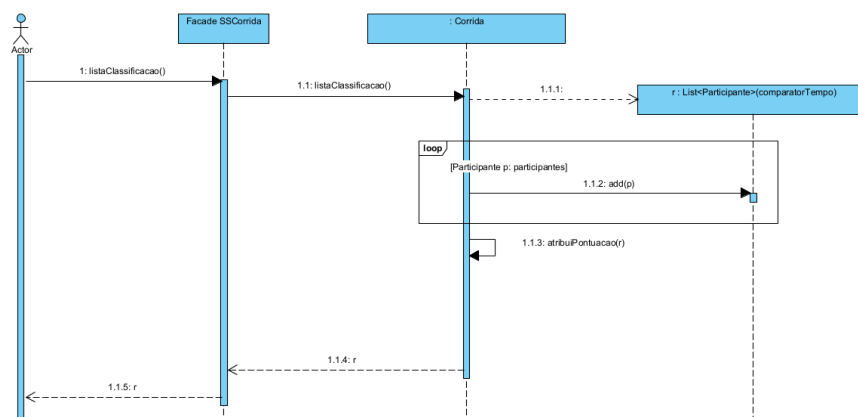


Figura 6.29: Método que devolve uma lista com as classificações

Classe Circuito

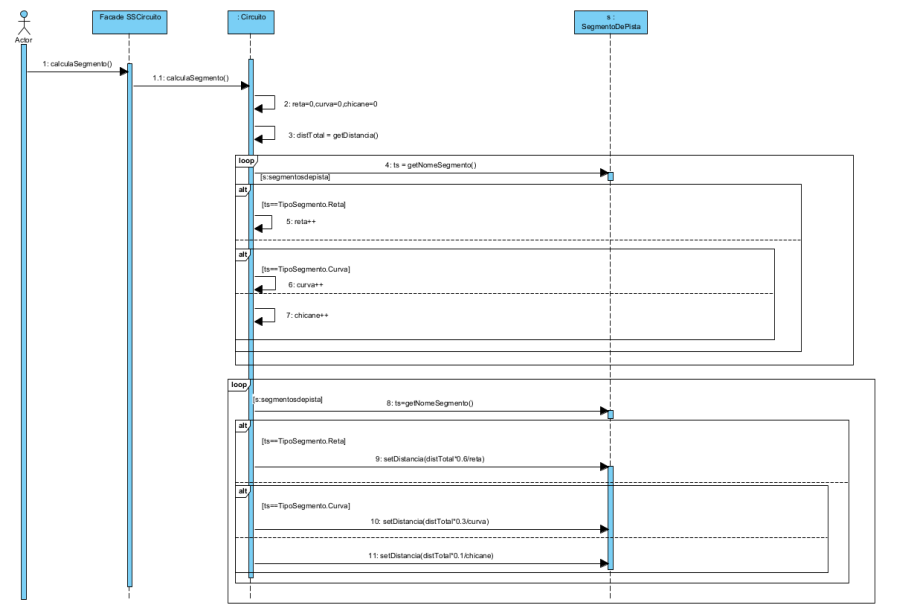


Figura 6.30: Método que calcula a distancia dos segmentos

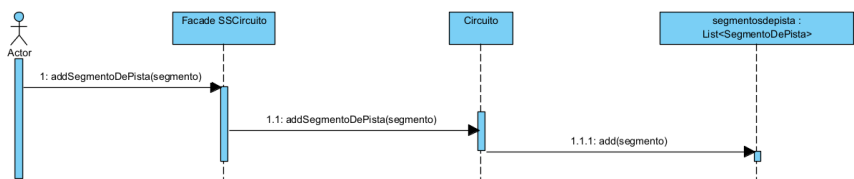


Figura 6.31: Método que adiciona segmento

Classe Participante

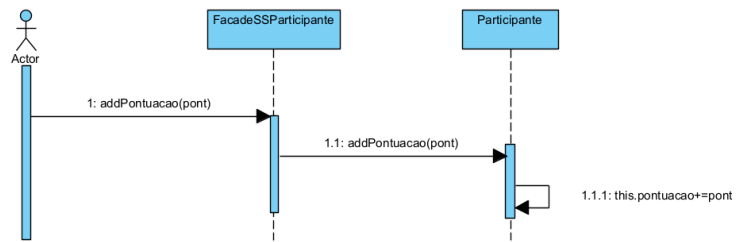


Figura 6.32: Método que adiciona pontuação ao participante

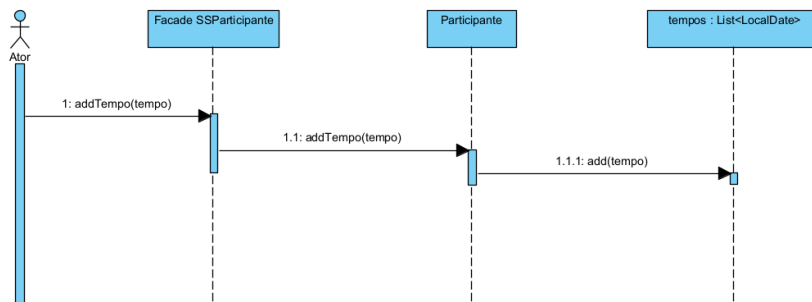


Figura 6.33: Método que adiciona tempo

Classe Carro

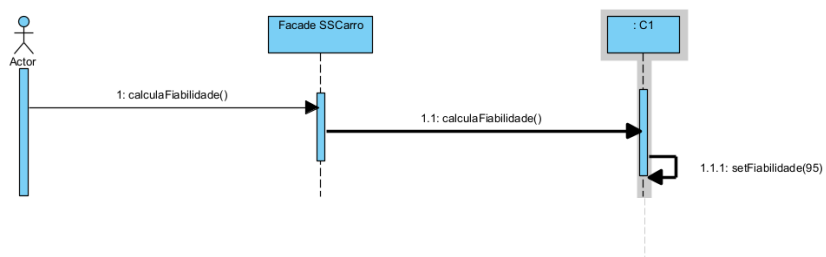


Figura 6.34: Método que calcula a fiabilidade de um carro C1

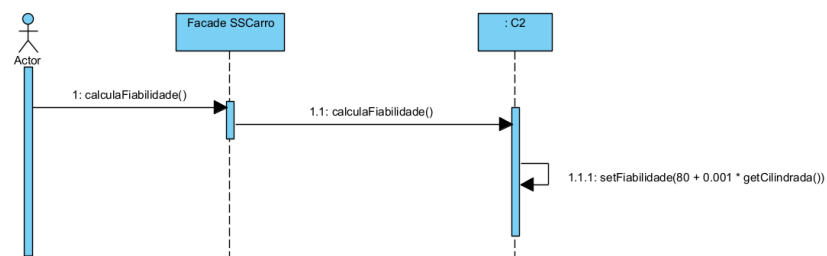


Figura 6.35: Método que calcula a fiabilidade de um carro do tipo C2

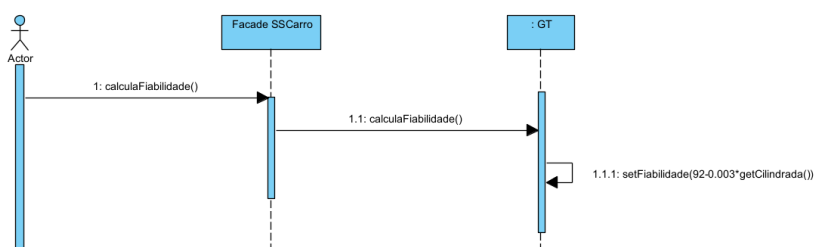


Figura 6.36: Método que calcula a fiabilidade de um carro do tipo GT

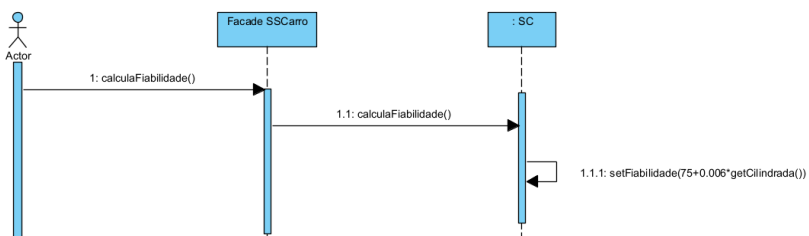


Figura 6.37: Método que calcula a fiabilidade de um carro do tipo SC

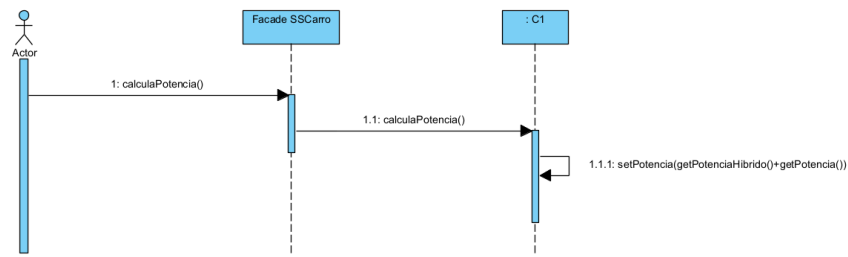


Figura 6.38: Método que calcula a potência de um carro do tipo C1

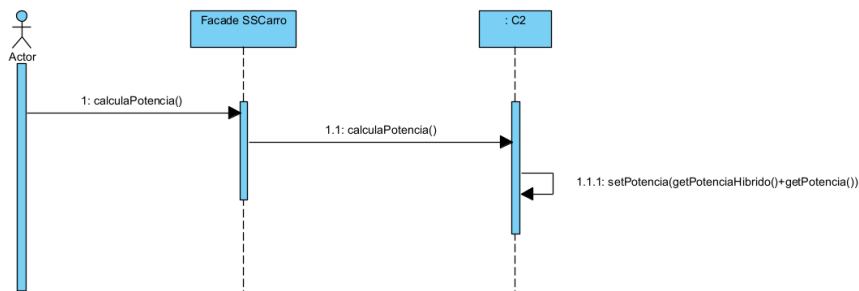


Figura 6.39: Método que calcula a potência de um carro do tipo C2

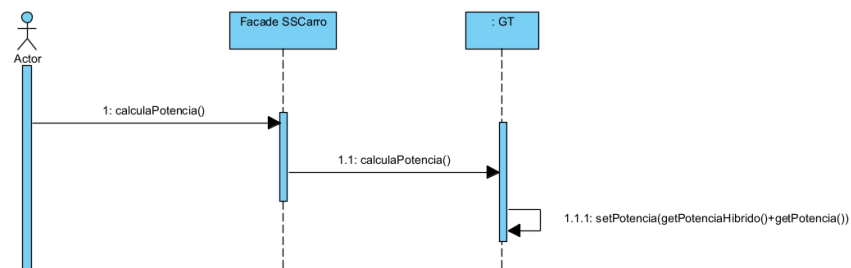


Figura 6.40: Método que calcula a potência de um carro do tipo GT

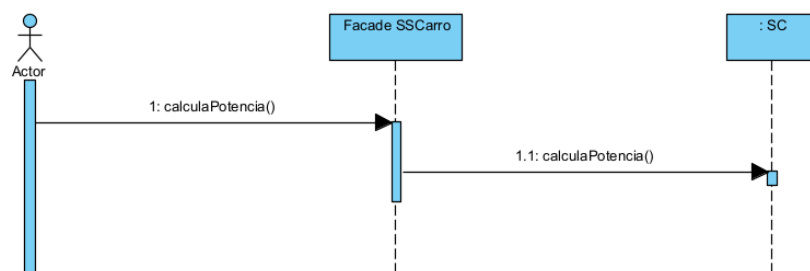


Figura 6.41: Método que calcula a potência de um carro do tipo SC

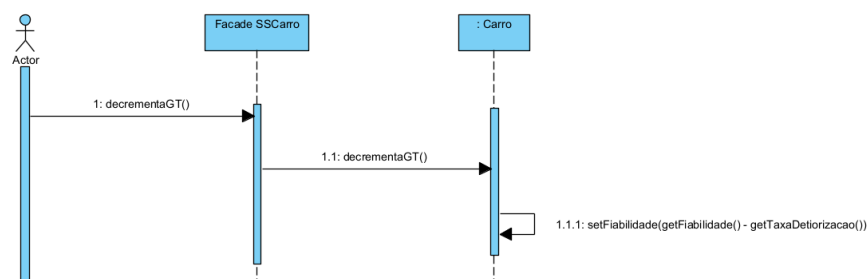


Figura 6.42: Método que decrementa a Fiabilidade do GT

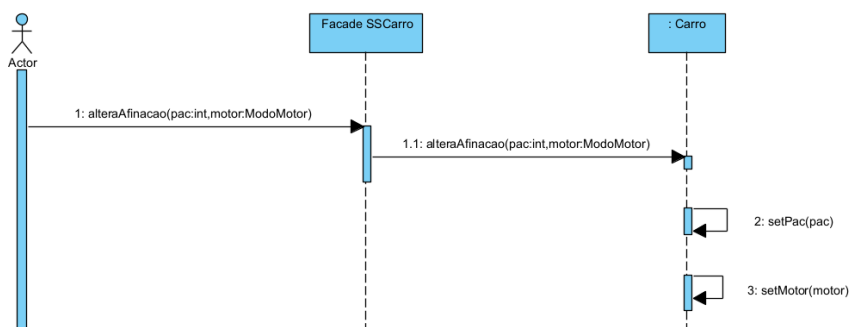


Figura 6.43: Método que altera a afinação de um carro

Classe Utilizador

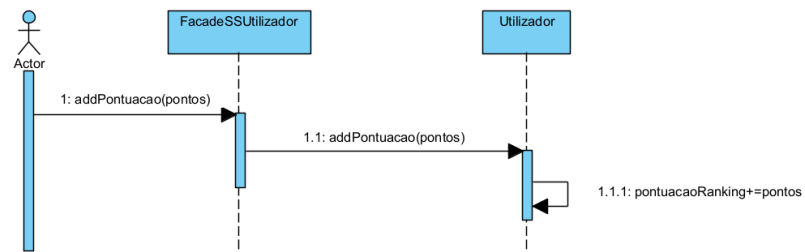
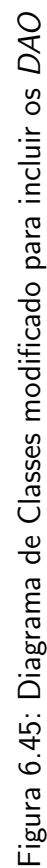


Figura 6.44: Método que adiciona pontuação ao utilizador

6.3 FASE 3 - Implementação da Solução



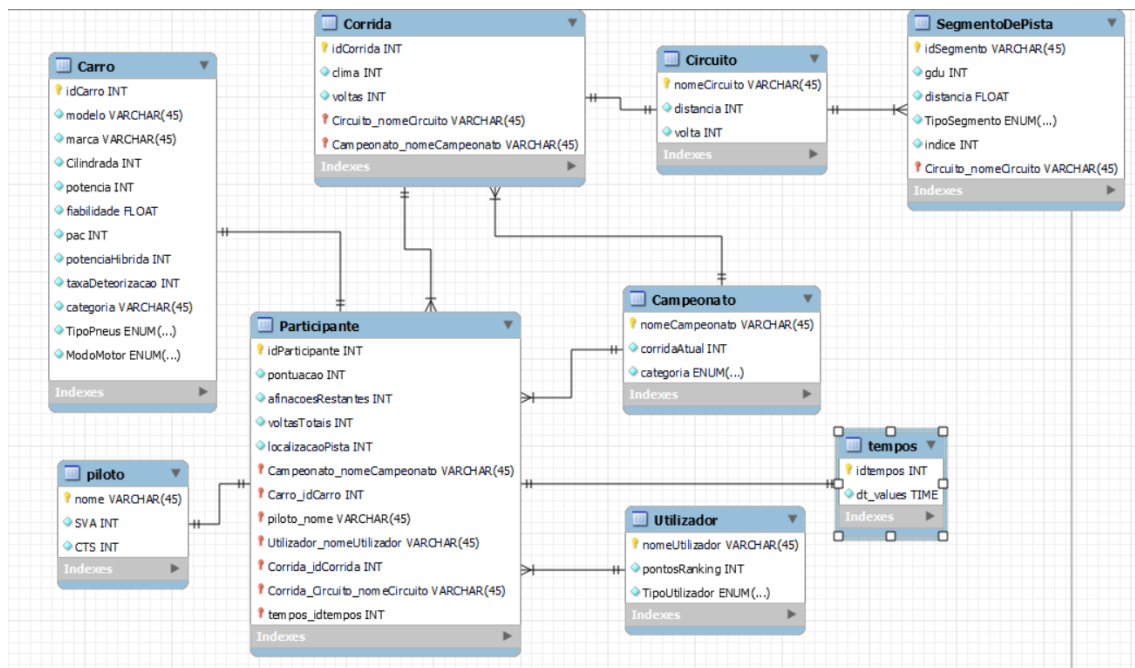


Figura 6.46: Modelo Relacional da Base de Dados