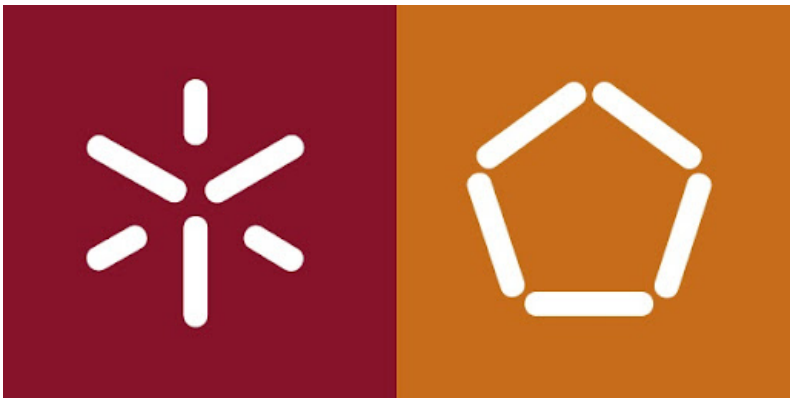
# Universidade do Minho

## Departamento de Informática
## Curso : MIEI / LEI



# Entrega final - Relatório de DSS

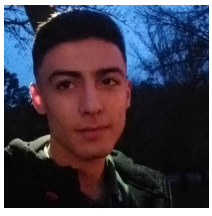
Grupo nº2

Alexandre Eduardo Vieira Martins A93242
Guilherme Rodrigues do Outeiro Cunha Marques A94984
Hugo dos Santos Martins A95125
José Eduardo da Cunha Rocha A97270
João Bernardo Teixeira Escudeiro A96075

https://github.com/Eduard0Rocha/ProjetoDSSGrupo2

7 de janeiro de 2023

Hugo Martins



José Rocha

João Escudeiro

Alexandre Martins

Guilherme Marques

# Índice

# Introdução

Nesta fase final do projeto tivemos como principal foco a implementação do sistema com base nos diagramas realizados nas fases anteriores, alguns modelos dos desenvolvidos previamente foram alterados ou apagados completamente de maneira a ficarem de acordo com o sistema realizado. Tendo isto em conta, desenvolvemos os problemas dados e concluímos totalmente este projeto.

# Objetivos

Nesta fase final do projeto, foi pedido e passo a citar “Os modelos necessários à descrição da implementação do sistema; Implementação do sistema; Terceira parte do relatório”. Tendo isto em conta, e sendo que nas fases anteriores desenvolvemos os modelos de descrição da implementação do sistema, o nosso foco foi dado quase na totalidade à implementação do sistema, e fizemos isto, como pedido, com base nos diagramas desenvolvidos anteriormente. Tivemos também que desenvolver uma base de dados para servir de apoio ao código, gerindo os dados de forma eficiente.

# Diagrama de classes

Após a análise da segunda fase, mantivemos os subsistemas criados nessa fase com algumas alterações, e adicionamos 2 packages novos.

### Subsistema Piloto

Foram alteradas as funções presentes na interface *SGestPiloto* assim como foram adicionados atributos e operações que se encontram agora presentes no *PilotosFacade*. Está agora também presente neste diagrama o *PilotoDAO* onde se encontram mencionadas as conexões com as outras classes.

### Subsistema Carro

Foram alteradas as funções presentes na interface SGestCarro assim como foram adicionados atributos e operações que se encontram agora presentes no CarrosFacade. Está agora também presente neste diagrama o CarroDAO onde se encontram mencionadas as conexões com as outras classes. Por fim foram adicionadas as funções e os atributos relacionados com as diferentes categorias dos carros(C1, C2, GT, SC) assim como os seus equivalentes híbridos.

### Subsistema Campeonato.

Foram alteradas as funções presentes na interface *SGestCampeonato* assim como foram adicionados atributos e operações que se encontram agora presentes no *CampeonatosFacade*. Está agora também presente neste diagrama o *CampeonatoDAO* e o *CircuitoDAO* onde se encontram mencionadas as conexões com as outras classes.

### Subsistema Users.

Foram alteradas as funções presentes nas interfaces *SGestUsers* e SGestUser assim como foram adicionados atributos e operações que se encontram agora presentes no *UsersFacade* e no *UserFacade*. Está agora também presente neste diagrama os *DAOs AdminDAO*, *GuestDAO*, *JogadorAutenticadoDAO*, *JogadorDAO* onde se encontram mencionadas as suas conexões com as outras classes.

### Subsistema Circuitos

Foram alteradas as funções presentes na interface *SGestCircuitos* assim como foram adicionados atributos e operações que se encontram agora presentes no *CircuitosFacade*. Está agora também presente neste diagrama o *CircuitoDAO* onde se encontram mencionadas as conexões com as outras classes.

### Subsistema Data

Neste subsistema encontram-se descritas as operações e os atributos de cada DAO (AdminDAO, CampeonatoDAO, CarroDAO, CircuitoDAO, DAOConfig, GuestDAO, JogadorAutenticadoDAO, JogadorDAO e PilotoDAO).

### Subsistema Business

Neste subsistema encontra-se definida a interface *F1Manager* onde se encontram definidas múltiplas funções relacionadas com a aplicação. Também está definida a *LogicaNegocio* onde estão definidas as operações mais importantes do sistema como por exemplo simulação de corridas. Encontram-se aqui também a relação da *LogicaNegocio* com todos os DAOs existentes.

# Diagramas de sequências

Nesta fase final, para complementar a modelação comportamental, adicionamos um novo diagrama de sequência, e alteramos outro. Adicionamos então um para simular uma corrida e e que simula campeonatos. O alterado foi o que adiciona registo.

O Adicionar Registo foi alterado de forma a suportar os *DAO's* que foram implementados no sistema. Assim, neste modelo é clara a separação de uma camada de base de dados e do resto do sistema. Foram feitos dois diagramas para esta função um referente à operação encontrada na classe *CampeonatosFacade* e outra de mesmo nome que se encontra na classe *LogicaNegocio*.

O novo Diagrama Simula Corrida adicionado modela o comportamento do programa numa das suas funções mais importantes, o de simular uma corrida. O segundo diagrama que foi adicionado é semelhante ao anterior visto que este refere-se à simulação de campeonatos.

# Diagrama Máquina de Estado

Nesta fase final, para complementar a modelação comportamental, adicionamos um novo diagrama de máquinas de estado. Este novo diagrama indica todas as interações possíveis a utilizar pelo utilizador num ciclo de vida dos menus.Através do mesmo consegue facilmente perceber-se de que forma se comporta o produto final , mediante os diferentes inputs do utilizador , bem como a ligação do mesmo com cada um dos packages

# Estratégias utilizadas

Para o desenvolvimento do sistema recorremos a uma metodologia simples metodologia. Começamos por criar packages relativos a cada um dos subsistemas. Cada um destes packages contava com classes facades e interfaces. Estes packages iriam trabalhar com a informação antes de ser inserida na base de dados, e eram aqui efetuadas todas as operações lógicas.

De seguida criamos um package que serviria para auxiliar nas interações com a base de dados. Cada módulo criado interagia diretamente com a tabela a que estava associado. Neste package criamos os *DAO's* que acediam as respectivas tabelas e que geriam as mesmas.

Após estes dois passos completos, passamos então à povoação da base de dados. Para isso limitámo-nos a inserir novos pilotos, carros, jogadores e circuitos, alguns deles reais, outros completamente inventados. A inserção da informação na base de dados deu-se com auxílio de funções criadas nos *DAO's*.

Enquanto a parte de base de dados era desenvolvida, criamos também o package de lógica de negócio, que possui todas as facades, o que facilita as interações entre a interface e o código.

Para a simulação do campeonato , após a mesma ser realizada , o campeonato não pode voltar a ser simulado , pois é ativa uma flag que não o permite. Esta opção foi pensada ponderadamente pelo grupo pois não faria sentido após duas simulações do mesmo campeonato um jogador conseguir o dobro dos pontos.

A simulação assenta nas características quer do piloto , quer do carro , quer das condições atmosféricas , bem como no circuito e respectivos graus de dificuldade.

O grupo optou por apenas solicitar aos usuários que pretendiam realizar alterações antes da simulação do campeonato , devido à escassez de tempo.

Por fim foi feita a interface principal, interface com a qual o cliente irá interagir. Não utilizamos os handlers como foi lecionado por falta de tempo, utilizamos então o *println*.

No desenvolvimento deste projeto são respeitados os principais paradigmas de Java, o encapsulamento e modularidade tal como seria de esperar.

Devido a falta de tempo , não conseguimos criar uma versão premium do programa. Em relação às afinações do carro apesar de ser pedido no enunciado que seja possível realizá-las antes das corridas, o grupo optou por escolher apenas ser possível fazê-las antes do campeonato começar, novamente pelo mesmo problema não conseguimos efetuar a implementação pedida.

# Manual de utilização da aplicação

A aplicação baseia-se numa réplica de um *F1Manager* , em que há uma conexão a uma base de dados local que ajuda a gerir os dados , bem como a guardar todas as informações relativas a cada uma das classes do programa . Quando o programa é iniciado , e após se conseguir estabelecer uma conexão à base de dados , o mesmo verifica se as tabelas existem ou se é necessário criá-las . Após isto , é possível obter as listas quer de jogadores (Autenticados ou Guests) quer de administradores, bem como fazer login ou povoar a base de dados com um conjunto de entidades que o grupo selecionou para ajudar na demonstração da aplicação. Para o administrador , aparece um menu em que pode fazer todas as operações possíveis relativas a carros, pilotos e circuitos como adicionar ou remover , e para os campeonatos consegue obter classificação para os mesmos , adicionar ou remover registos e assim .Para os jogadores é possível obter os carros/pilotos/circuitos/campeonatos disponíveis ,criar um campeonato ou simular um deles.

De uma forma geral a aplicação comporta-se como o esperado , em que é possível fazer um leque de alterações relativas a todas as entidades.

Após uma simulação do programa é possível obter as classificações dos campeonatos , bem como ocorre a atribuição dos pontos.

# Análise crítica dos resultados obtidos

Após a conclusão desta etapa e consequentemente do trabalho final o grupo autoavalia o seu trabalho em positivo considerando todos os requisitos definidos para cada etapa e trabalho final.

A frequentação das aulas práticas e teóricas ajudou os elementos do grupo a ultrapassar barreiras e dificuldades que foram surgindo ao longo da realização do projeto, tendo estas sido consideradas fundamentais para a conclusão do mesmo.

O grupo considera que podia ter tido uma melhor performance em alguns aspetos como por exemplo: inclusão da versão premium, inclusão de handlers em vez de prints e incluir a possibilidade de realizar afinações antes de cada corrida.

Apesar de o maior objetivo ser baseado no cenário 5 o grupo começou optou começar pela criação das entidades pois não faria sentido começar a simulação de campeonatos sem a criação das mesmas. Este processo foi bastante demorado e trabalhoso pois para além de extenso está propenso a bastantes erros.

# Conclusão

Após a conclusão da terceira fase do trabalho prático, e consequentemente da componente prática da UC consideramos ainda que a mesma foi bastante importante para aprimorarmos os nossos conhecimentos relativamente à transferência dos conhecimentos obtidos na modelação concetual para código Java.Desta maneira, foi-nos possível implementar os conceitos abordados nas aulas práticas e teóricas.

Quanto às dificuldades sentidas nesta última fase do projeto, foram sobretudo problemas na conexão com a Base de Dados, falta de conhecimento relativa ao uso dos "*Result Sets*" e a divisão de tarefas e tempo disponível.

No geral , este trabalho, ajudou bastante a perceber a modelação conceptual assim como a implementação do código baseando-nos em diagramas.Sentimos que nos será bastante útil na vida profissional vindoura, permitindo a criação de código mais eficiente, aumento do tempo útil de programação e sobretudo a descoberta de erros grandes numa fase ainda precoce dos projetos.

# Anexos

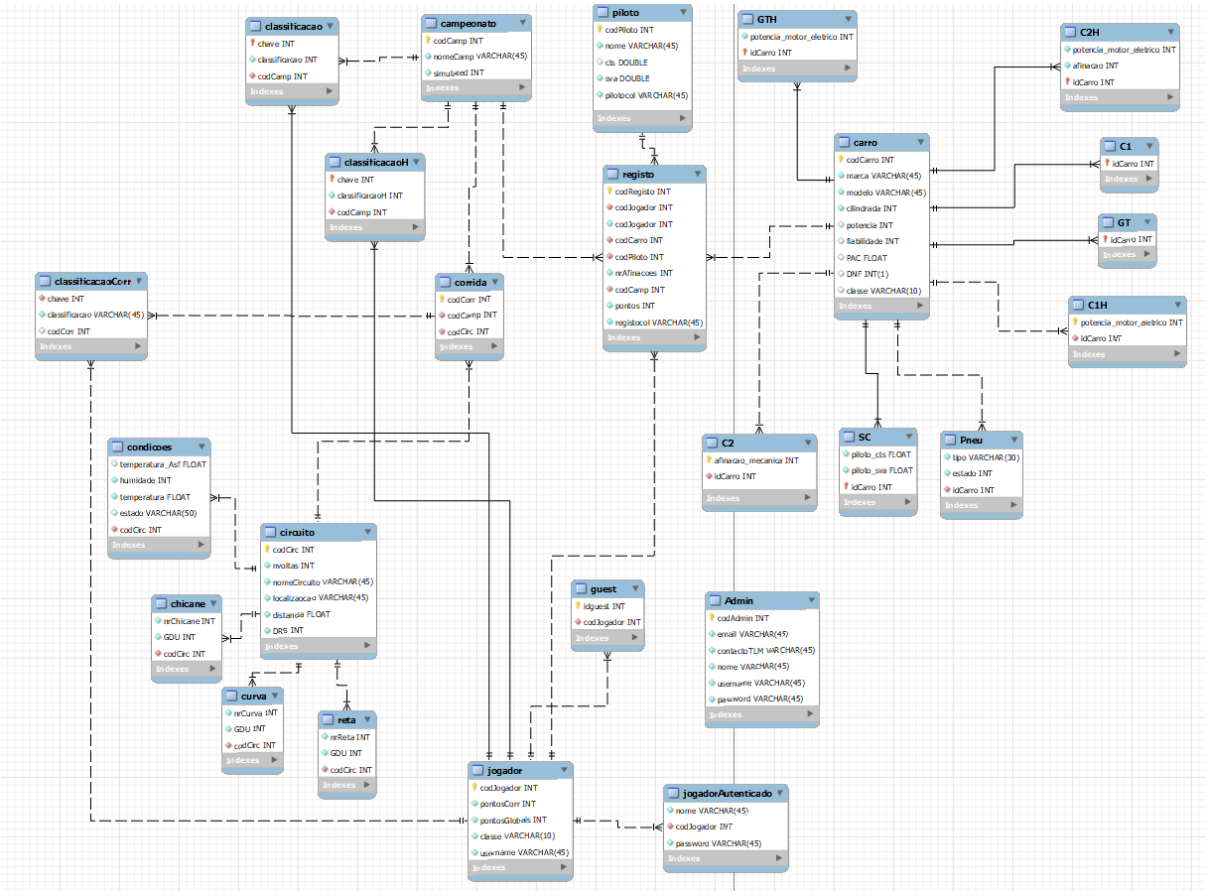



Figura 1 - Modelação Lógica da base de dados.

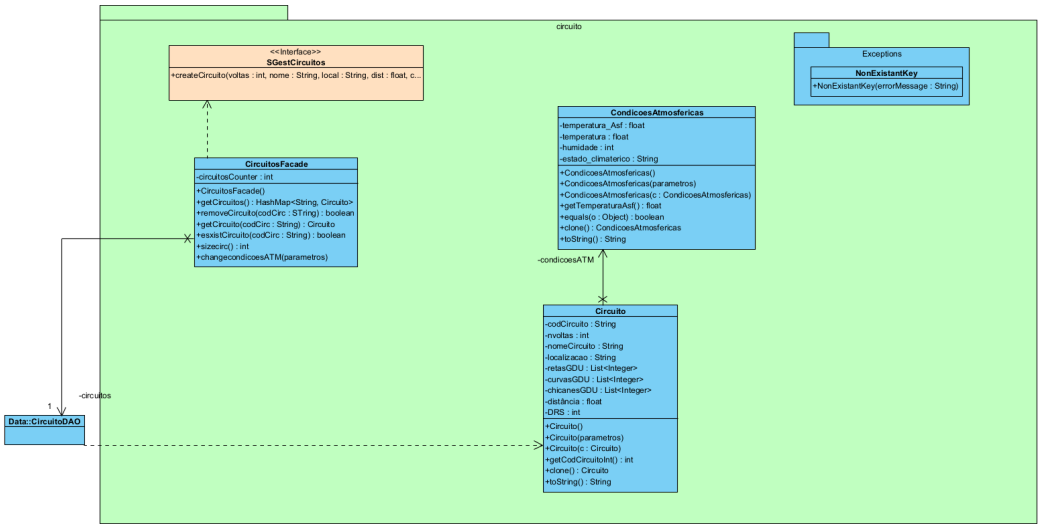



Figura 2 - Diagrama Subsistema Circuito.

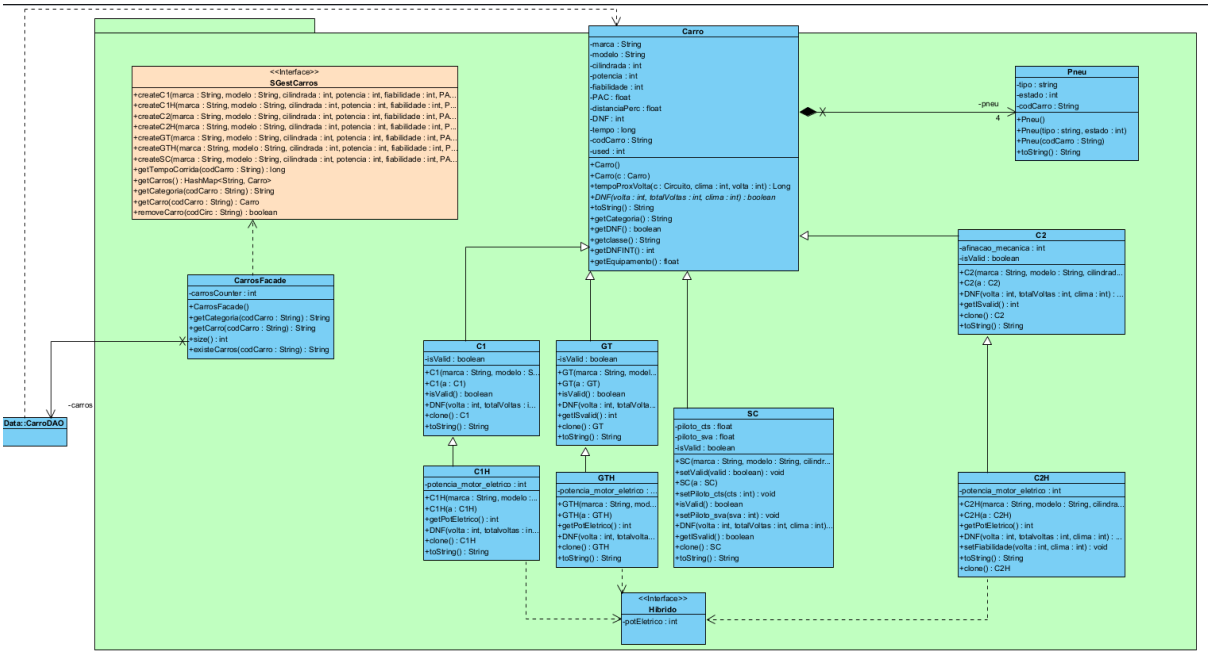



Figura 3 - Diagrama Subsistema Carro.

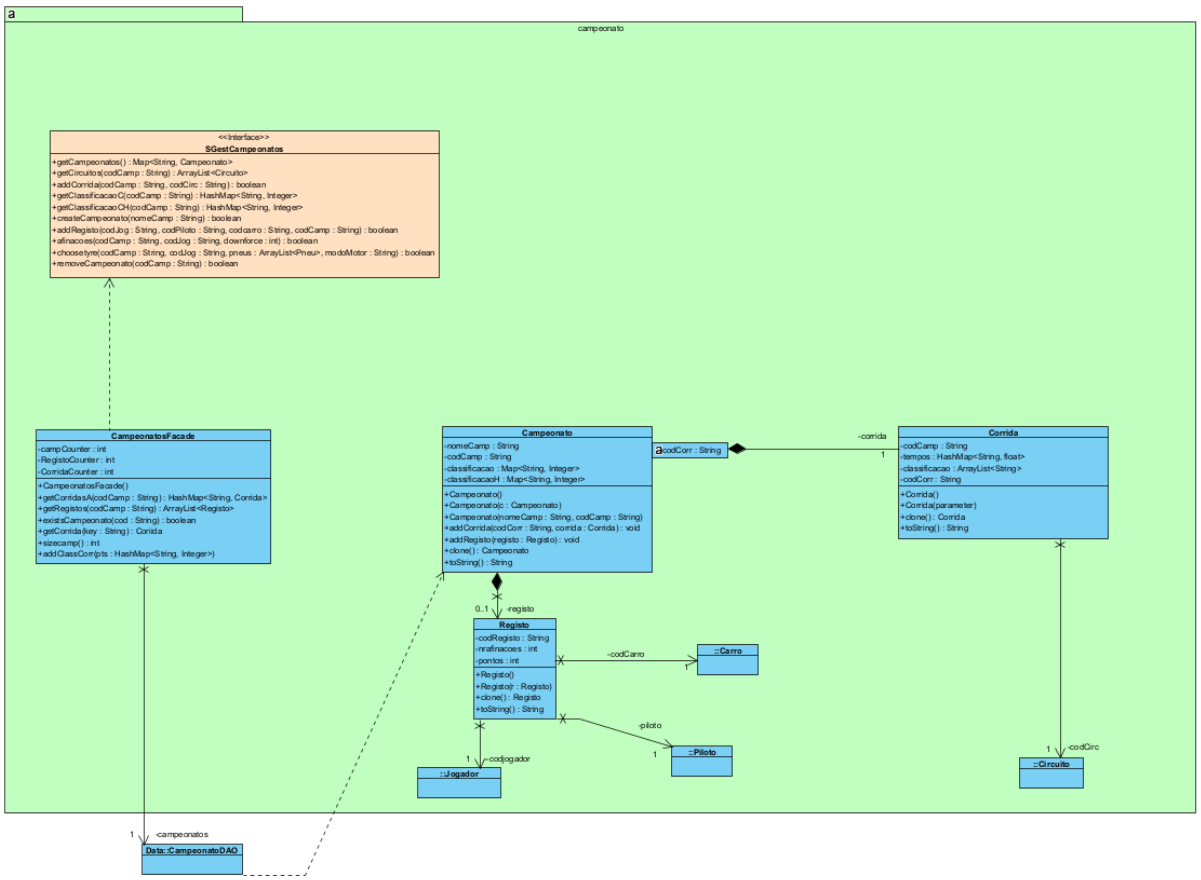



Figura 4 - Diagrama Subsistemas Campeonatos.

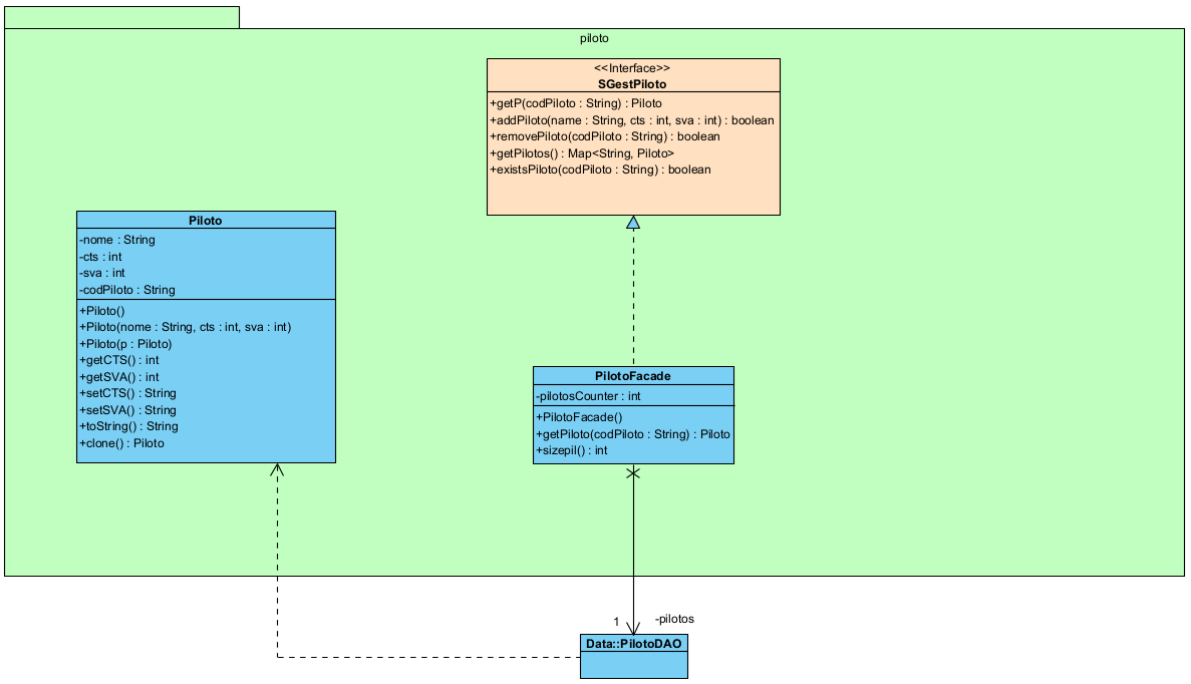



Figura 5 - Diagrama Subsistema Piloto.

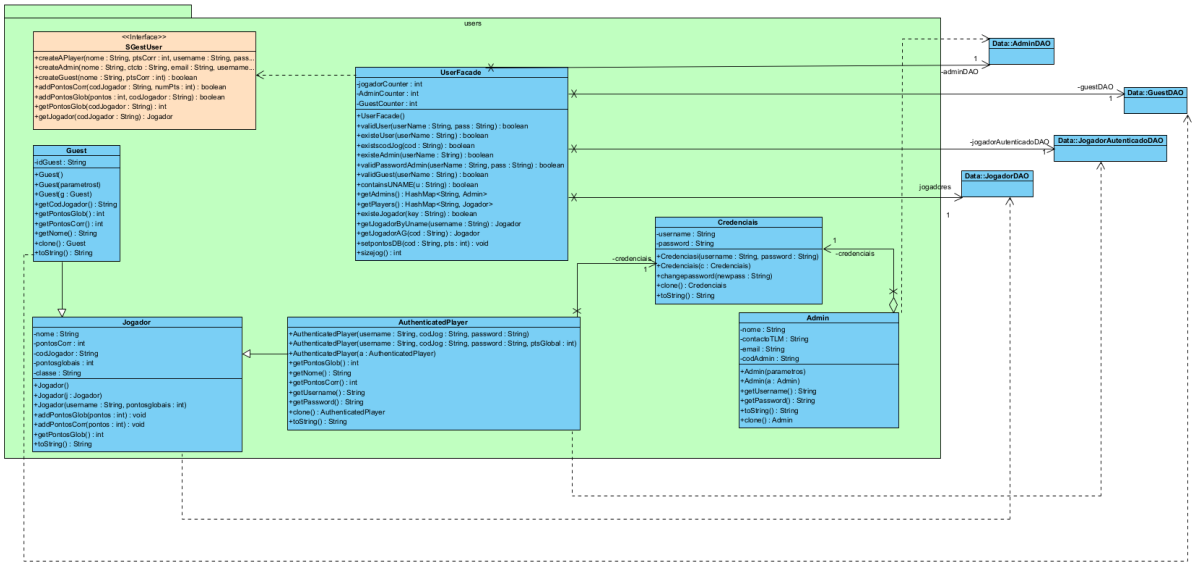



Figura 6 - Diagrama Subsistema User.

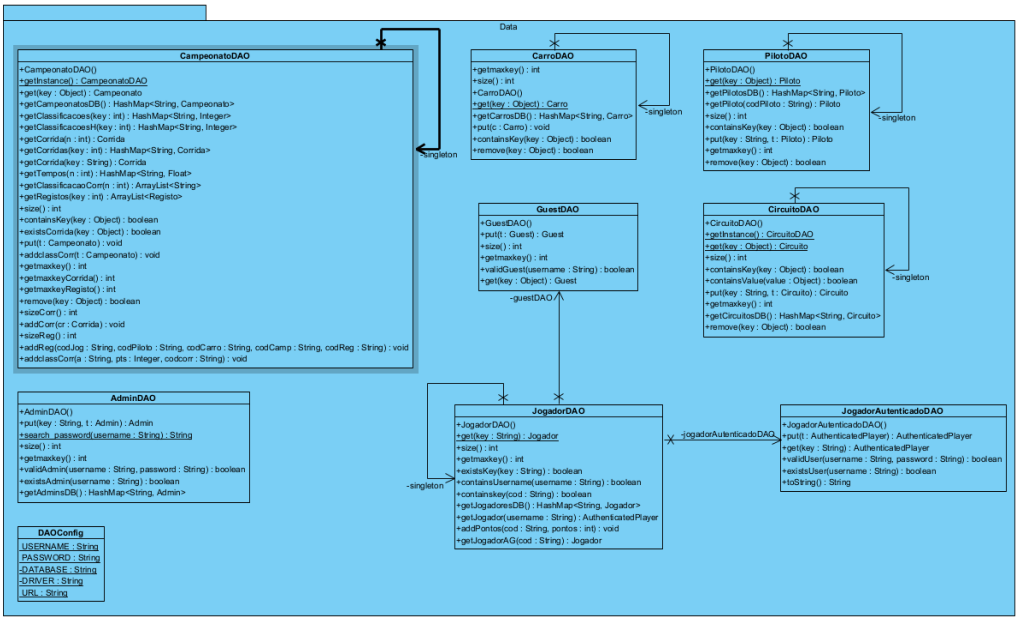



Figura 7 - Diagrama Subsistema Data.

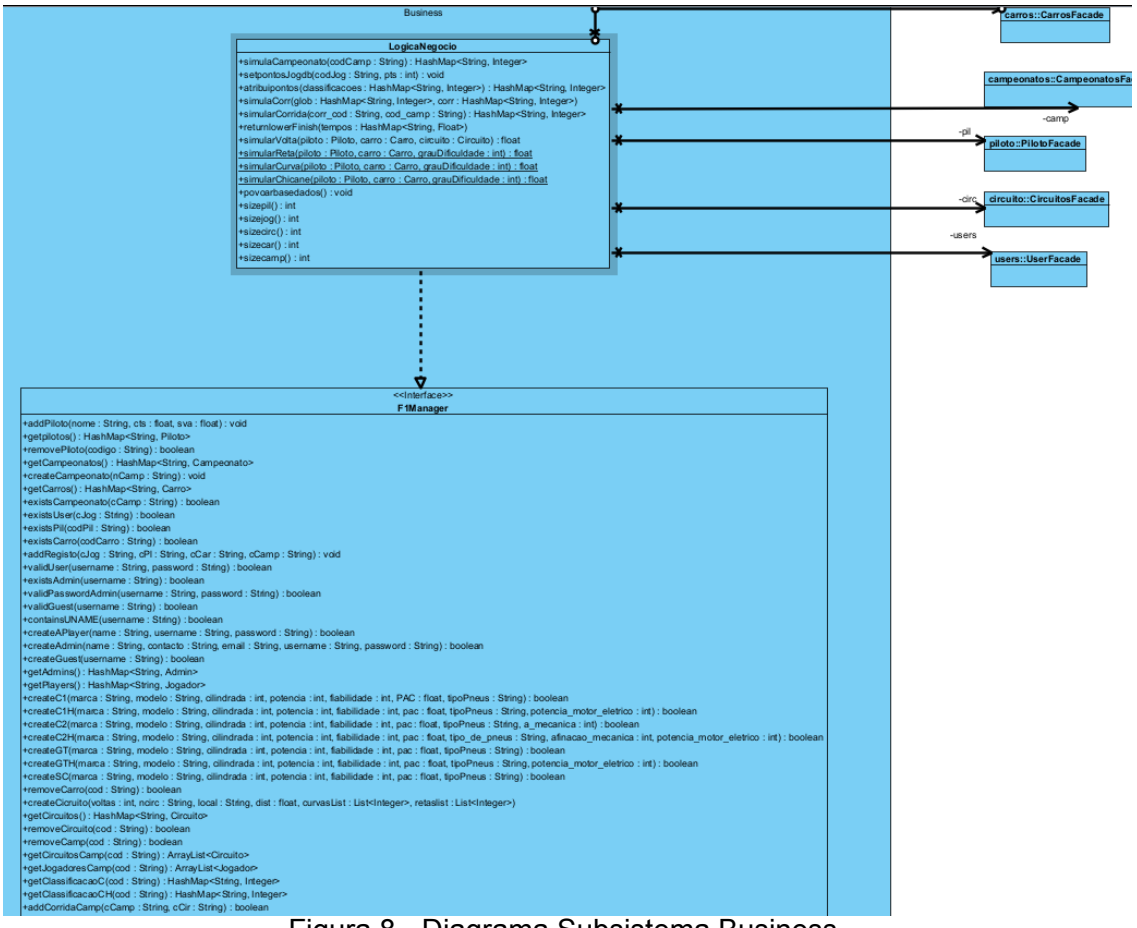



Figura 8 - Diagrama Subsistema Business.

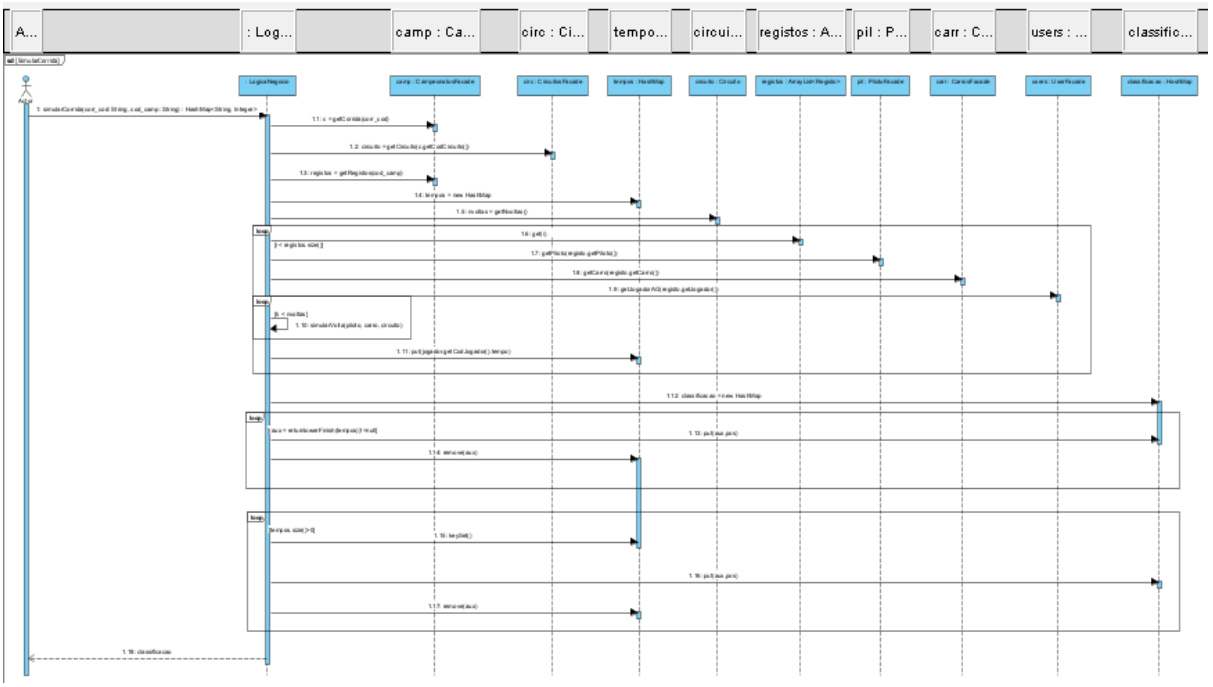



Figura 9 - Diagrama de Sequência Simular Corrida.

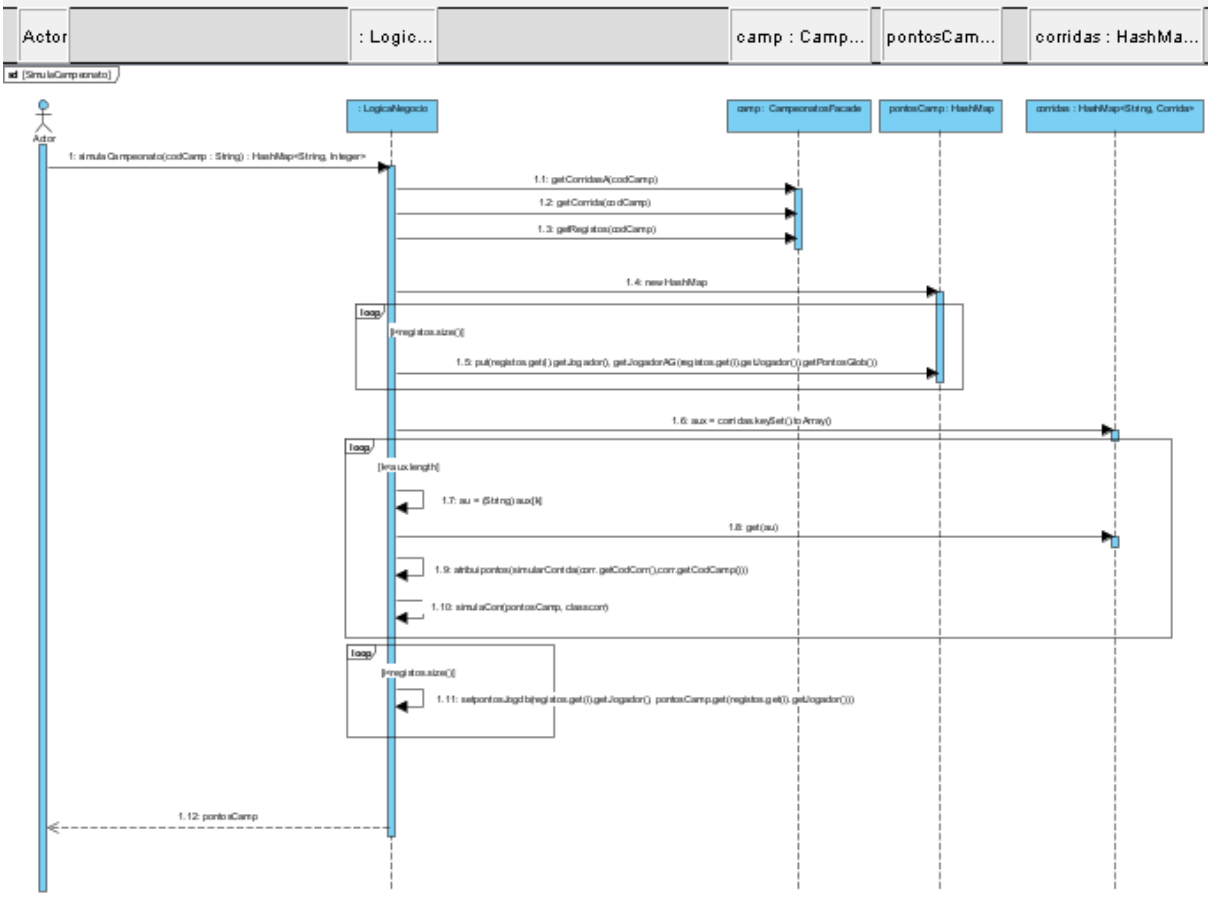


Figura 10 - Diagrama de Sequência Simular Campeonato.

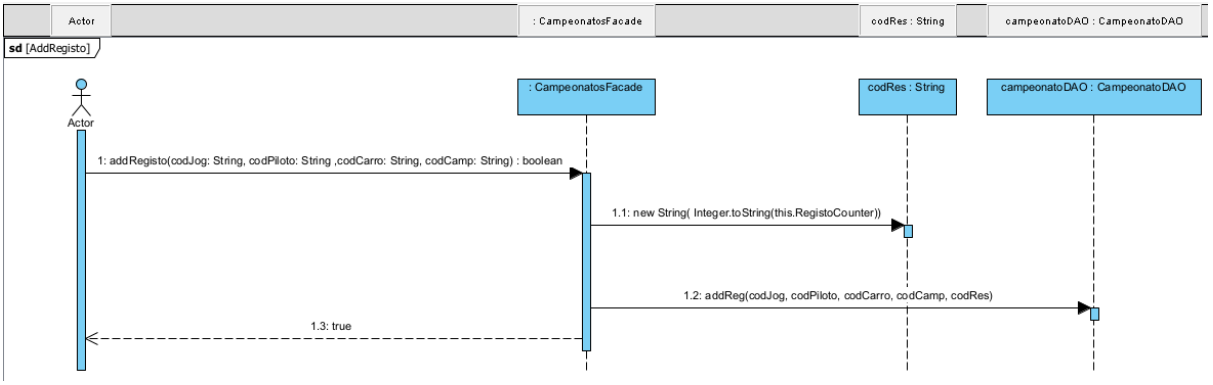



Figura 11 - Diagrama de Sequência AddRegisto da classe CampeonatoFacade.

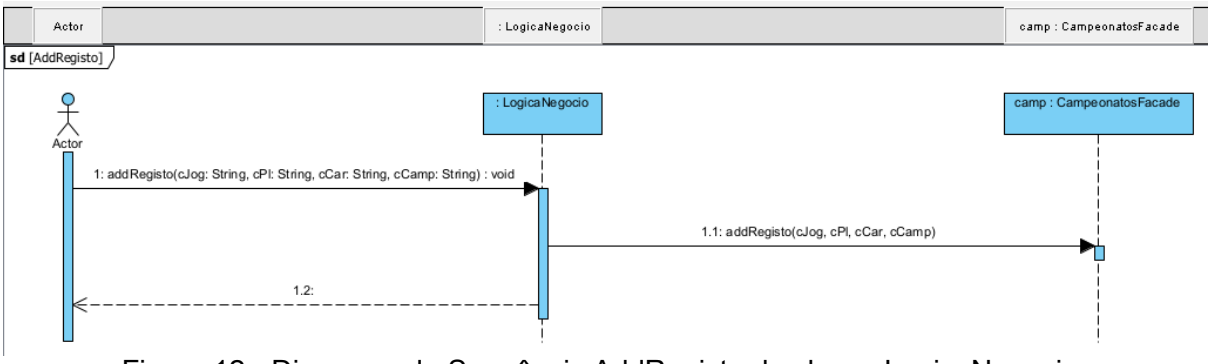



Figura 12 - Diagrama de Sequência AddRegisto da classe LogicaNegocio.

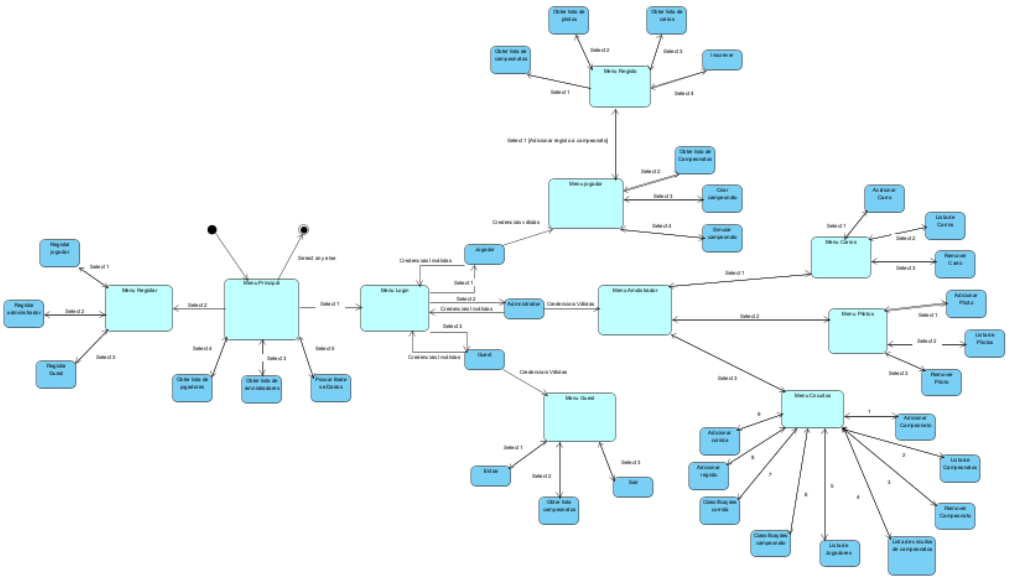


Figura 13 - Diagrama de Máquina de Estado(Menus).