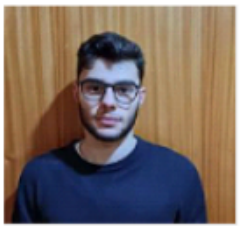
Universidade do Minho
Escola de Engenharia

Desenvolvimento de Sistemas de Software

Trabalho Prático - Fase 1

Grupo 10

Link Github: `https://github.com/LEI-DSS/DSS2425-Grupo-10`



Marco Gonçalves
(a104614)

Leonardo Alves
(a104093)

Salvador Barreto
(a104520)

André Pereira
(a104275)

Índice

1 Introdução

No presente relatório, começaremos por descrever o nosso software através de um **modelo de domínios** e um **modelo de use cases**. De seguida faremos a especificação de cada um dos **Use Case**, que facilitarão na modelação conceptual, onde apresentaremos o nosso **diagrama de classes** dividido por subsistemas. Apresentaremos ainda **diagramas de sequência** relativos aos métodos de API mais importantes da lógica de negócio. Por fim, falaremos da solução efetivamente implementada, mostrando o **diagrama de componentes** e respetivos **diagramas de classes** e **diagrama de pacotes**, uma vez que nesta fase houve a implemetação do **ORM**, e houve alteração da modelação lógica. É de realçar ainda que, nem todos as funcionalidades definidas na modelação conceptual foram implementadas.

2 Análise de requisitos

2.1 Modelo de Domínio

Um **modelo de domínio** é uma abstração da realidade que permite descrever tudo aquilo que é essencial a um problema, escondendo tudo o que for detalhes irrelevantes. Tendo isto em conta, este foi o modelo de domínios que decidimos elaborar para representar o problema que nos foi fornecido:

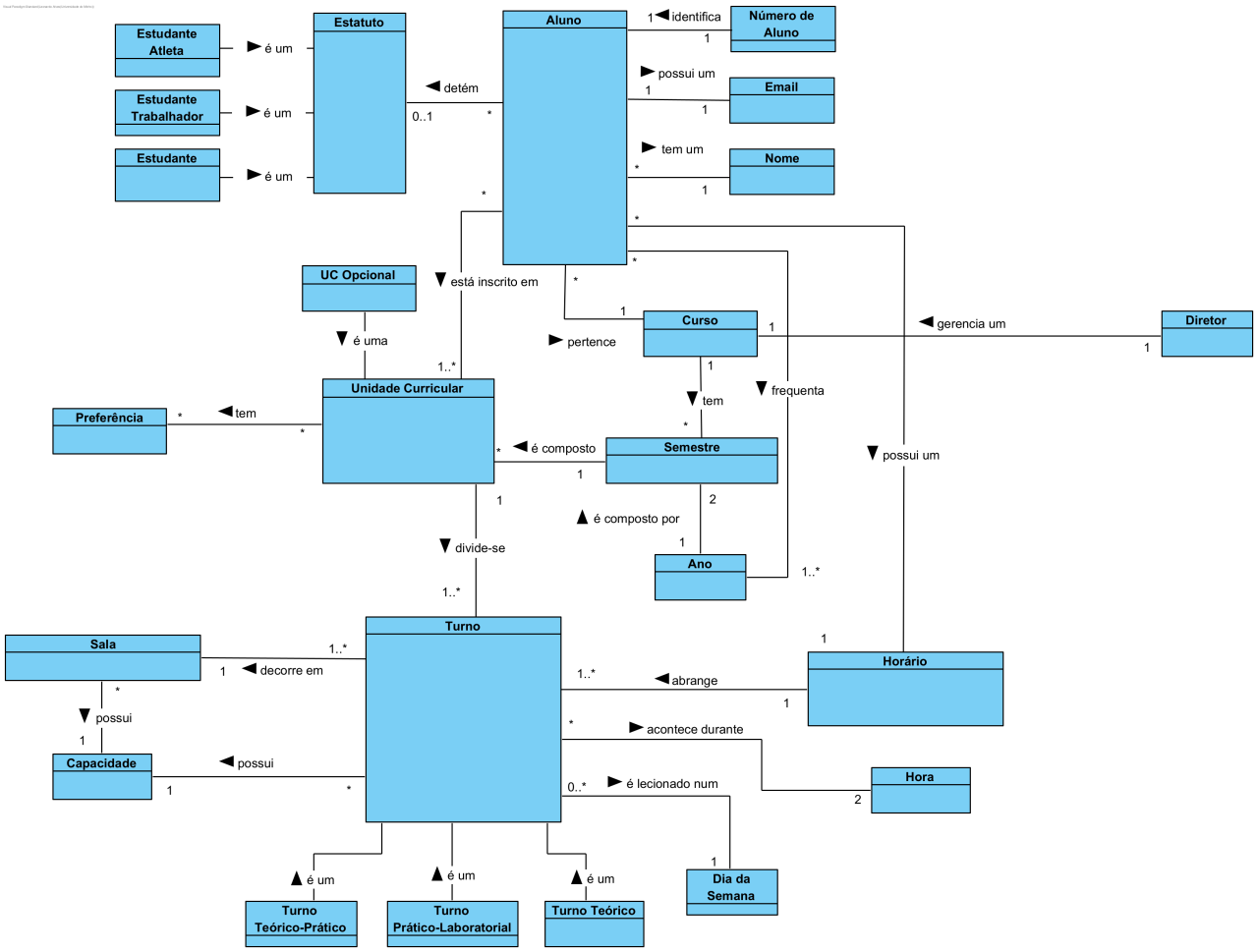
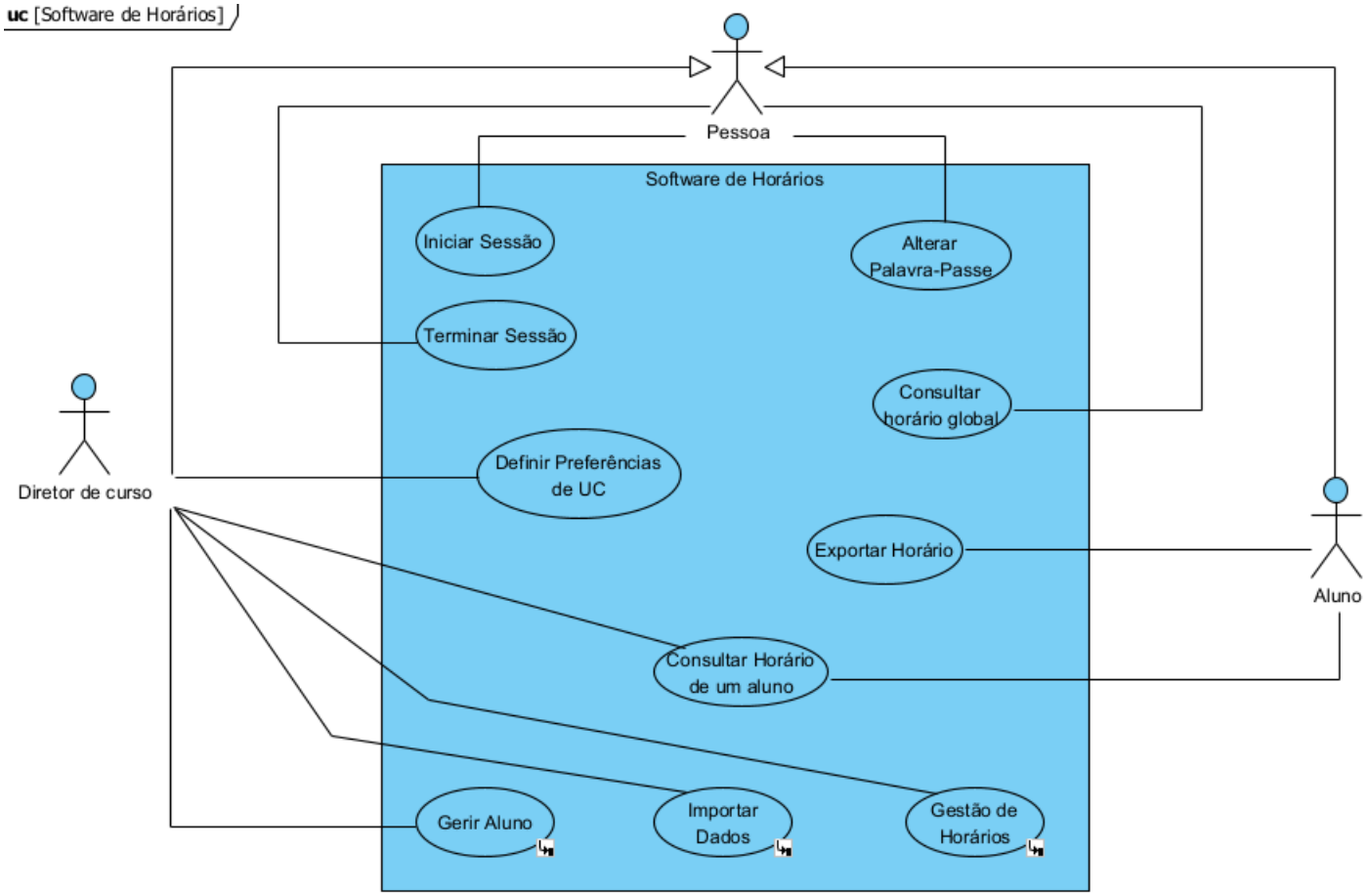
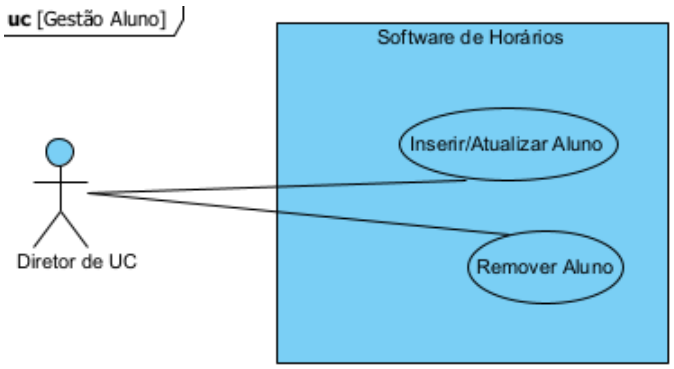


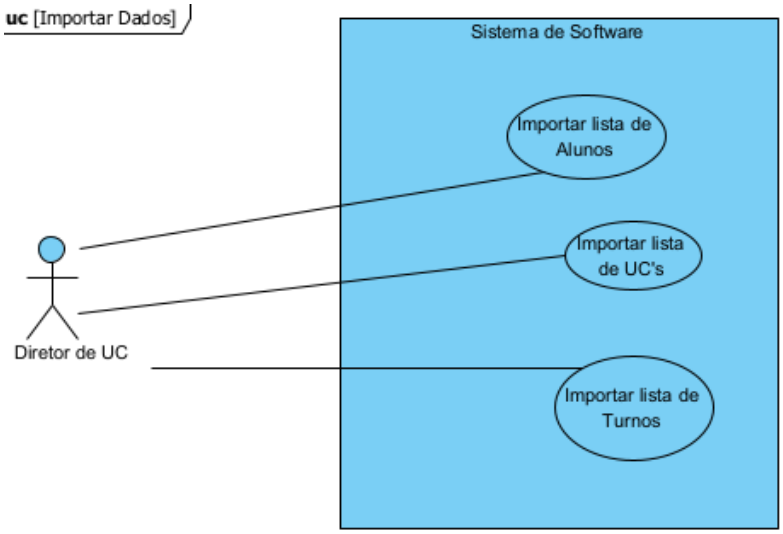
Figura 1: Modelo de Domínio

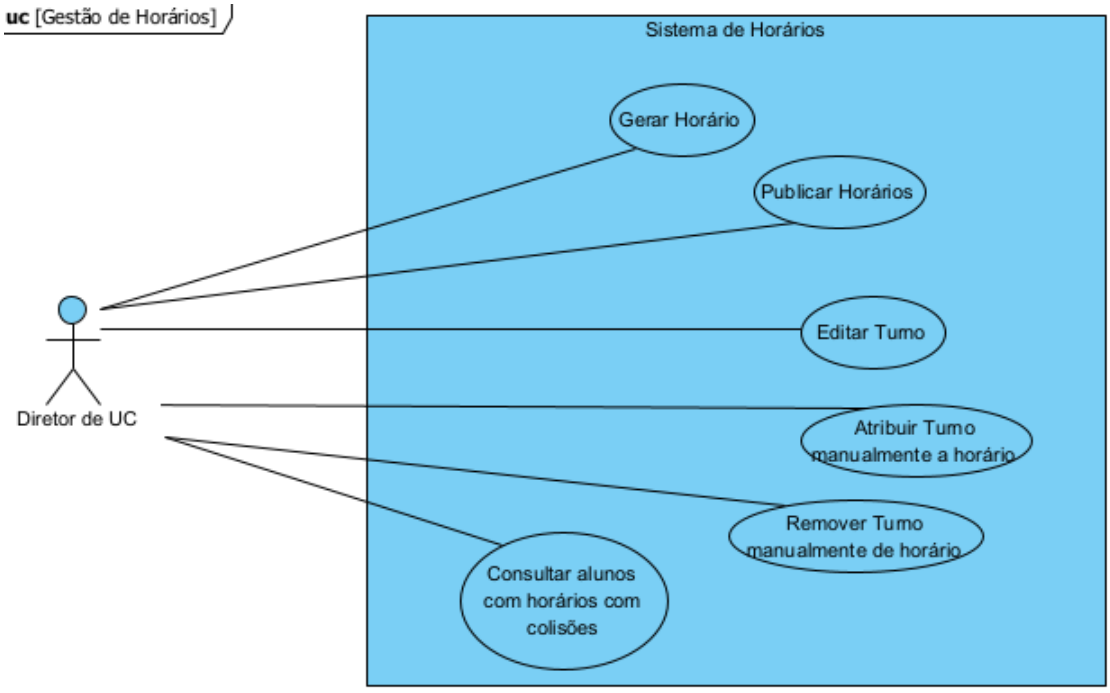
2.2 Modelo de Use Case

Um **modelo de use case** descreve como atores atingem objetivos utilizando o sistema e especificam o comportamento típico esperado bem como as suas variantes.









USE CASE: Iniciar Sessão
Ator: Utilizador.
Descrição: Utilizador inicia uma sessão no software de horários.
Cenário: Cenário 1 e 2
Pré-condição: Utilizador não está autenticado
Pós-condição: Utilizador fica autenticado.

Fluxo Normal:

1. Sistema solicita ao utilizador que insira o seu número de utilizador e palavra-passe.
2. Utilizador insere número de utilizador e palavra-passe.
3. Sistema valida número de utilizador e palavra passe e autentica o utilizador no sistema.

Fluxo de Exceção 1: [Validação de utilizador e palavra-passe falhou] (passo 3)

3.1 Sistema informa que número de utilizador e/ou palavra-passe não são válidos.

USE CASE: Terminar Sessão
Ator: Utilizador.
Descrição: Utilizador termina uma sessão no software de horários.
Cenário: Cenário 1 e 2
Pré-condição: Utilizador está autenticado.
Pós-condição: Utilizador deixa de estar autenticado.

Fluxo Normal:

1. Utilizador solicita o término de sessão.
2. Sistema termina sessão corretamente.

USE CASE: Alterar Palavra-passe
Ator: Utilizador.
Descrição: Utilizador altera a sua password no software de horários.
Cenário: Cenário 1 e 2
Pré-condição: Utilizador está autenticado.
Pós-condição: Password do ator fica alterada.

Fluxo Normal:

1. Ator insere a sua palavra-passe antiga.
2. Sistema valida palavra-passe anterior.
3. Sistema pede ao utilizador para inserir a nova palavra-passe.
4. Utilizador especifica a nova palavra-passe.
5. Sistema regista e associa a nova palavra-passe ao utilizador.

Fluxo de Exceção 1: [Validação da palavra-passe antiga falhou] (passo 2)

2.1 Sistema informa que a palavra-passe anterior é inválida.

USE CASE: Consultar horário de um aluno
Ator: Utilizador.
Descrição: Utilizador consulta um horário de um aluno.
Cenário: Cenário 1 e 2
Pré-condição: Utilizador autenticado e horários foram publicados.
Pós-condição: Sistema apresenta horário.

Fluxo Normal:

1. Utilizador fornece o número do aluno cujo horário pretende consultar.
2. Sistema verifica que o aluno existe.
3. Sistema apresenta horário associado ao aluno.

Fluxo de Exceção 1: [Aluno não existe] (passo 2)

2.1 Sistema informa que o aluno não foi encontrado.

USE CASE: Consultar Horário global
Ator: Utilizador.
Descrição: Utilizador consulta o horário global com todos turnos de cada disciplina de um certo ano do curso.
Cenário: Cenário 1 e 2
Pré-condição: Utilizador autenticado e horários foram publicados.
Pós-condição: Sistema apresenta horário.

Fluxo Normal:

1. Diretor de curso insere o ano do horário que pretende consultar.
2. Sistema apresenta horário.

USE CASE: Exportar Horário
Ator: Aluno.
Descrição: Aluno exporta para fora do sistema o seu horário.
Cenário: Cenário 2
Pré-condição: Horário foi publicado.
Pós-condição: Aluno possui ficheiro de horário.

Fluxo Normal:

1. Sistema exporta horário.

USE CASE: Importar Lista de Alunos (Importar Dados)
Ator: Diretor de Curso.
Descrição: O Diretor de Curso importa para o sistema uma lista de alunos.
Cenário: Cenário 1
Pré-condição: Diretor de Curso está autenticado.
Pós-condição: Novos alunos são adicionados ao sistema.

Fluxo Normal:

1. Diretor de Curso insere a pasta e o nome do ficheiro a importar
2. Sistema seleciona ficheiro e importa os dados.
3. Sistema notifica o Diretor de Curso que o ficheiro foi importado com sucesso.

Fluxo de Exceção 1: [Erro ao ler o ficheiro] (passo 2)

2.1 Sistema notifica o Diretor de Curso que houve um erro a ler o ficheiro.

USE CASE: Importar Lista de UC's (Importar Dados)
Ator: Diretor de Curso.
Descrição: O Diretor de Curso importa para o sistema uma lista de UC's.
Cenário: Cenário 1
Pré-condição: Diretor de Curso está autenticado.
Pós-condição: Novas UC's são adicionadas ao sistema.

Fluxo Normal:

1. Diretor de Curso insere a pasta e o nome do ficheiro a importar
2. Sistema seleciona ficheiro e importa os dados.
3. Sistema notifica o Diretor de Curso que o ficheiro foi importado com sucesso.

Fluxo de Exceção 1: [Erro ao ler o ficheiro] (passo 2)

2.1 Sistema notifica o Diretor de Curso que houve um erro a ler o ficheiro.

USE CASE: Importar Lista de Turnos (Importar Dados)
Ator: Diretor de Curso.
Descrição: Diretor de Curso importa para o sistema uma lista de turnos.
Cenário: Cenário 1
Pré-condição: Diretor de Curso está autenticado.
Pós-condição: Novos turnos são adicionados ao sistema.

Fluxo Normal:

1. Diretor de Curso insere a pasta e o nome do ficheiro a importar.
2. Sistema seleciona ficheiro e importa os dados.
3. Sistema notifica o Diretor de Curso que o ficheiro foi importado com sucesso.

Fluxo de Exceção 1: [Erro ao ler o ficheiro] (passo 2)

2.1 Sistema notifica o Diretor de Curso que houve um erro a ler o ficheiro.

USE CASE: Gerar Horário (Gestão de Horários)
Ator: Diretor de Curso.
Descrição: Sistema gera um horário para cada um dos alunos.
Cenário: Cenário 1
Pré-condição: Existem turnos e alunos no sistema.
Pós-condição: Horário fica alocado a um conjunto de alunos.

Fluxo Normal:

1. Sistema aloca horários aos alunos.
2. Sistema notifica o Diretor de Curso que os horários foram gerados e atribuídos com sucesso.

USE CASE: Publicar Horários (Gestão de Horários)
Ator: Diretor de Curso.
Descrição: Sistema torna os horários visíveis para os alunos.
Cenário: Cenário 1
Pré-condição: Horário tem de estar atribuído a pelo menos um aluno.
Pós-condição: Horário fica disponível para ser consultado pelo aluno.

Fluxo Normal:

1. Sistema torna os horários gerados visíveis.
2. Sistema notifica o Direto de Curso que os horários foram publicados.

USE CASE: Atribuir Turno Manualmente a horário (Gestão de Horários)
Ator: Diretor de Curso.
Descrição: Sistema permite atribuir um turno a um certo aluno a pedido do diretor de curso.
Cenário: Cenário 1
Pré-condição: Horário tem de estar atribuído a um aluno.
Pós-condição: Horário atualizado pelo diretor de curso.

Fluxo Normal:

1. Diretor de curso insere o número do aluno cujo horário deseja alterar.
2. Sistema verifica que o aluno existe.
3. Diretor de curso insere o identificador do turno que pretende inserir no horário.
4. Sistema verifica que o turno existe.

5. Sistema atualiza o horário do aluno.
6. Sistema atualiza o número de inscritos no turno.
7. Sistema informa o ator que o horário do aluno e o turno foram atualizados.

Fluxo de Exceção 1: [Número de aluno não existe] (passo 2)

2.1 Sistema notifica o diretor de curso que o número de aluno não existe.

Fluxo de Exceção 2: [Turno não existe] (passo 4)

4.1 Sistema notifica o diretor de curso que o número de turno não existe.

Fluxo de Exceção 3: [Aluno já está inscrito no turno] (passo 5)

5.1 Sistema notifica o diretor de curso que o aluno já se encontra nesse turno.

USE CASE: Remover Turno Manualmente de horário (Gestão de Horários)
Ator: Diretor de Curso.
Descrição: Sistema permite remover um turno a um certo aluno a pedido do diretor de curso.
Cenário: Cenário 1
Pré-condição: Horário tem de estar atribuído a um aluno.
Pós-condição: Horário atualizado pelo diretor de curso.

Fluxo Normal:

1. Diretor de curso insere o identificador do turno que pretende remover do horário.
2. Sistema verifica se o turno existe.
3. Diretor de curso insere o número do aluno cujo horário deseja alterar.
4. Sistema verifica que o aluno existe.
5. Sistema atualiza o horário do aluno.
6. Sistema atualiza o número de inscritos no turno.
7. Sistema informa o diretor de curso que o horário e turno foram atualizados.

Fluxo de Exceção 1: [Turno não existe] (passo 2)

2.1 Sistema notifica o diretor de curso que o número de turno não existe.

Fluxo de Exceção 2: [Número de aluno não existe] (passo 4)

4.1 Sistema notifica o diretor de curso que o número de aluno não existe.

Fluxo de Exceção 3: [Aluno não está inscrito no turno] (passo 5)

4.1 Sistema notifica o diretor de curso que o aluno não pertence a esse turno.

USE CASE: Consultar alunos com horários com colisões (Gestão de Horários)
Ator: Diretor de Curso.
Descrição: Diretor de Curso consulta alunos com colisões nos seus horários.
Cenário: Cenário 1
Pré-condição: Diretor de Curso tem de estar autenticado.
Pós-condição: Sistema lista alunos com colisões no horário.

Fluxo Normal:

1. Sistema apresenta lista de alunos com colisões no horário.

Fluxo Alternativo 1: [Não existem alunos com colisões no horário] (passo 1)

1.1 Sistema notifica o diretor de curso que não existem alunos com colisões no horário.

USE CASE: Inserir/Atualizar Aluno (Gestão de Aluno)
Ator: Diretor de Curso.
Descrição: Diretor de Curso insere manualmente um novo aluno no sistema.
Cenário: Cenário 1
Pré-condição: Diretor de Curso está autenticado.
Pós-condição: Um novo aluno é adicionado ao sistema.

Fluxo Normal:

1. Diretor de curso insere número, nome, e-mail, ano, UC's e estatuto do novo aluno.
2. Sistema valida os dados e regista novo aluno.
3. Sistema informa que a inserção foi bem sucedida.

USE CASE: Remover Aluno (Gestão de Aluno)
Ator: Diretor de Curso.
Descrição: Diretor de Curso remove um aluno do sistema.
Cenário: Cenário 1
Pré-condição: Diretor de Curso está autenticado.
Pós-condição: Um aluno é removido do sistema.

Fluxo Normal:

1. Diretor de Curso insere o número do aluno que deseja remover.
2. Sistema verifica que o aluno existe.
3. Sistema remove o aluno dos seus turnos e remove o aluno do sistema.
4. Sistema notifica o ator que a remoção foi bem sucedida.

Fluxo de Exceção 1: [Aluno não existe] (passo 2)

2.1 Sistema Diretor de Curso informa que o aluno não existe.

USE CASE: Editar Turno (Gestão de Horários)
Ator: Diretor de Curso.
Descrição: Diretor de Curso altera dados de um turno.
Cenário: Cenário 1
Pré-condição: Diretor de Curso está autenticado.
Pós-condição: Um turno é atualizado com novos dados.

Fluxo Normal:

1. Diretor de curso insere o turno que deseja alterar.
2. Sistema verifica que o turno existe.
3. Diretor de curso insere os novos dados do turno.
4. Sistema valida os novos dados e atualiza o turno.
5. Sistema notifica o ator que as alterações foram bem sucedidas.

Fluxo Exceção 1: [Turno não existe] (passo 2)

2.1 Sistema informa que o número não existe

USE CASE: Definir Preferências de UC
Ator: Diretor de Curso.
Descrição: Diretor de Curso define as preferências de uma UC.
Cenário: Cenário 1
Pré-condição: Diretor de Curso está autenticado.
Pós-condição: Uma preferência é implementada.

Fluxo Normal:

1. Diretor de Curso seleciona a UC à qual pretende adicionar preferência.
2. Sistema verifica que a UC existe.
3. Diretor de Curso especifica a(s) preferência(s).
4. Sistema aplica a(s) preferência(s).
5. Sistema notifica o ator que as alterações foram bem sucedidas.

Fluxo Exceção 1: [UC não existe] (passo 2)

2.1 Sistema informa ator que UC não existe.

3 Modelação Conceptual

3.1 Diagrama de Classe:

Como modelação inicial do que viria a ser a arquitetura do nosso programa decidimos criar o seguinte diagrama de classe:

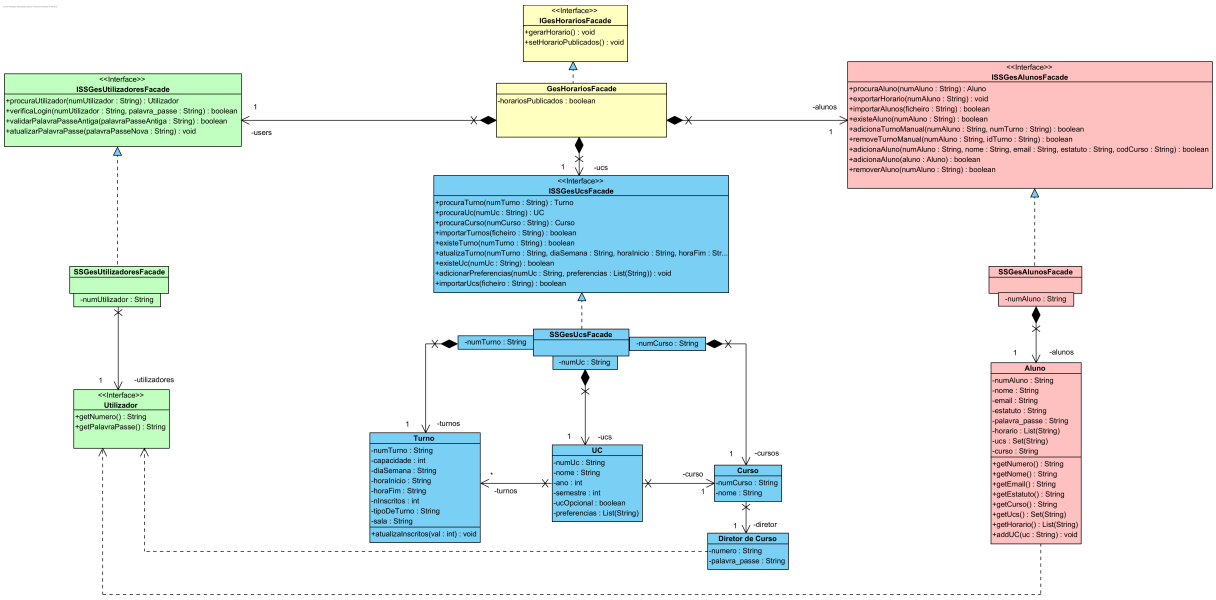


Figura 2: Diagrama de classe

Inicialmente fizemos a modelação **não recorrendo** ao **ORM** de modo a nos focarmos apenas na lógica do problema, o que leva a um design que reflete as necessidades reais impostas pelo mesmo. Decidimos também atribuir cores diferentes àquilo que consideramos vir a ser o nossos diferentes subsistemas. Para identificar estes subsistemas recorremos ao modelo de domínio e a um ficheiro Excel criado por nós, no qual identificamos as responsabilidades de lógica de negócio, definimos API's e identificamos os devidos subsistemas.

Desta forma, as classes a **verde** representam o subsistema de Utilizadores, a **azul** representa o subsistema de Unidades Curriculares (Ucs) e a **vermelho** o subsistema dos Alunos. Todos estes subsistemas são compostos por uma classe Facade e a sua interface, de modo a centralizar as operações desses subsistemas num único ficheiro.

Assim, a amarelo, está a nossa classe **Facade** da lógica de negócio, **GesHorariosFacade**, que trata de executar operações que necessitam de métodos dos vários subsistemas, por exemplo, a geração de horário, necessitará de estar nessa classe Facade, uma vez que necessitará de atribuir turnos aos alunos, classes de subsistemas diferentes.

Podemos ainda verificar que a classe **Aluno** contém um atributo horário, uma lista de **String**, que representa os códigos dos turnos desse aluno, desta forma, caso este queira consultar o seu horário detalhadamente (atributos da classe **Turno**), a classe **GesHorariosFacade** será novamente responsável por fazer esta operação, procurando os turnos dessa lista no subsistema de Unidades Curriculares.

3.1.1 Subsistema de utilizadores (verde)

O subsistema de utilizadores é responsável por gerir as operações dos utilizadores, como por exemplo iniciar sessão (operação verificaLogin), e a sua classe **Facade** contém um Map da interface utilizadores, que irá ser implementada pelos Alunos e Diretor de curso, uma vez que são estes que poderão interagir com o nosso sistema.

3.1.2 Subsistema de Unidades Curriculares(UCs) (azul)

O subsistema de unidades curriculares é responsável por gerir as operações relativas a cada uma das unidades curriculares. A Facade é responsável por guardar em Map's as classes **Turno**, **UC** e **Curso**.

A nossa classe **UC** representa uma unidade curricular e a ela tem atribuídos diversos atributos, os mais relevantes são:

- **turnos** - uma lista da classe **Turno**, que representa precisamente os turnos que existem dessa UC;
- **curso** - uma classe Turno que representa o curso dessa Unidade Curricular.

A classe **Turno** representa os turnos das várias unidades curriculares, e os atributos relevantes são:

- **capacidade** - representa o número máximo de alunos que esse turno terá (capacidade da sala ou número limite do turno), e será útil na parte da geração de horários;
- **diaSemana**, **horaInicio**, **horaFim** - assim como a capacidade, estes atributos serão úteis na geração de horários, uma vez que evitará a sobreposição de turnos num horário de um aluno;
- **nInscritos** - representa o número de inscritos nesse turno;
- **tipoDeTurno** - representa o tipo do turno, PL - Prático-Laboratorial, TP - Teórico-Prático, T-Teórico.

A classe **Curso** representa os cursos que estarão guardados no nosso sistema e cada um destes terá a si associado um Diretor de Curso, que é este quem poderá realizar operações relativas a este curso, como por exemplo gerar horários, editar turnos e inserir alunos.

3.1.3 Subsistema de Alunos (vermelho)

Por fim, temos o subsistema de alunos, onde a sua classe **Facade** é responsável por guardar um Map da classe Aluno que estarão guardados no nosso sistema.

A classe **Aluno** representa os alunos de cada um dos cursos, desta forma, cada um dos alunos será também um utilizador, uma vez que poderá utilizar o sistema para conseguir consultar o seu horário, desta forma, os atributos mais importantes são os seguintes:

- **estatuto** - representa os diferentes estatutos dos alunos, útil na geração de horários;
- **horario** - representa o horário do aluno, guarda uma lista com os números do turnos;
- **ucs** - representa as unidades curriculares em que o aluno se encontra inscrito - guarda uma lista com os números das UC's.
- **curso** - representa o curso em que o aluno está inscrito, guarda o número do curso.

3.2 Diagramas de Sequência:

O diagrama representado na **figura 3** é referente ao **Use Case "Iniciar Sessão"**, pelo que representa o processo necessário para que um utilizador inicie sessão no sistema.

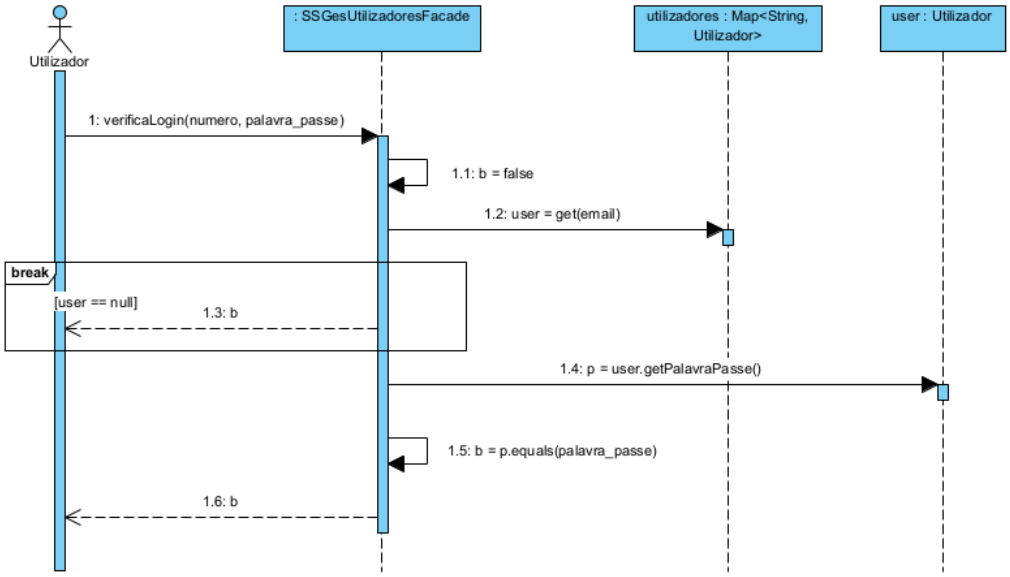


Figura 3: Iniciar Sessão

O diagrama representado na **figura 4** é referente ao **Use Case "Inserir/Atualizar alunos"**, pelo que representa o processo necessário para que o diretor de curso insira manualmente um novo aluno no sistema, ou atualize um já existente.

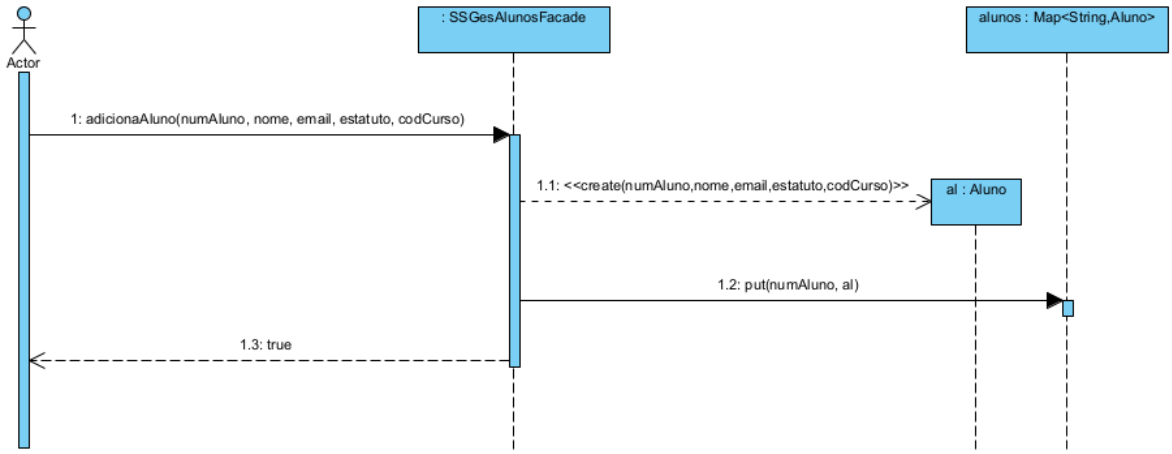


Figura 4: Inserir/Atualizar alunos

O diagrama representado na **figura 5** é referente ao **Use Case "Importar lista de alunos"**, pelo que representa o processo necessário para que o diretor de curso importe, através de um ficheiro ".csv"a lista de alunos inscritos no curso.

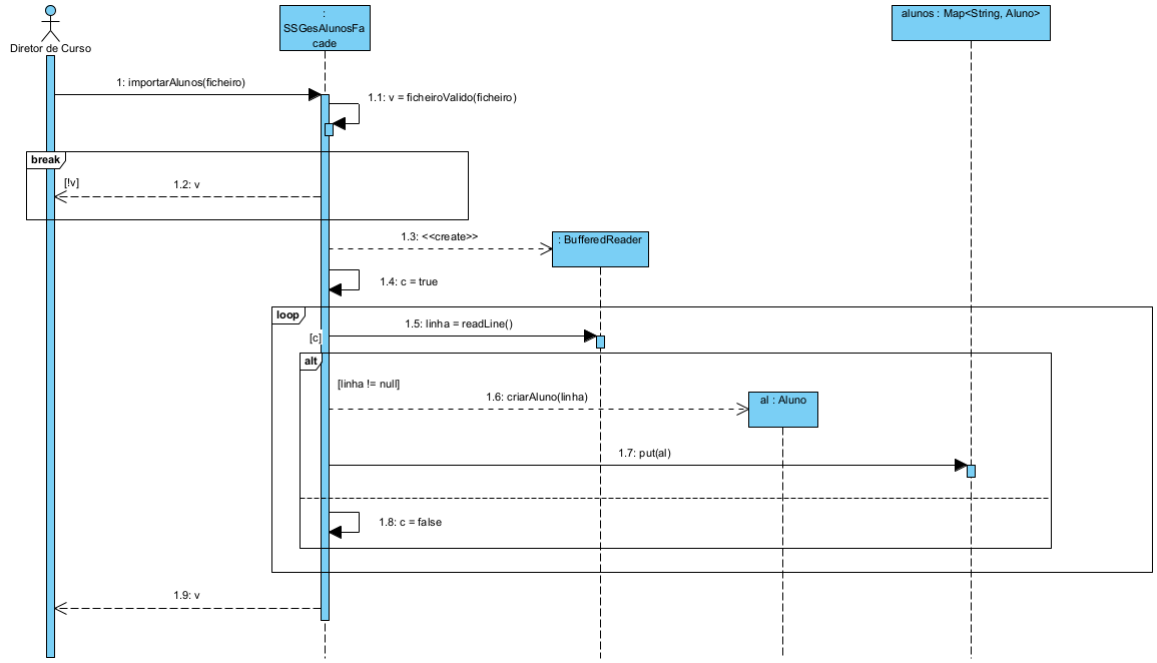


Figura 5: Importar lista de alunos

O diagrama representado na **figura 6** é referente ao **Use Case "Importar lista de turnos"**, pelo que representa o processo necessário para que o diretor de curso importe, através de um ficheiro ".csv"a lista de turnos pertencentes a várias UC's (previamente importadas).

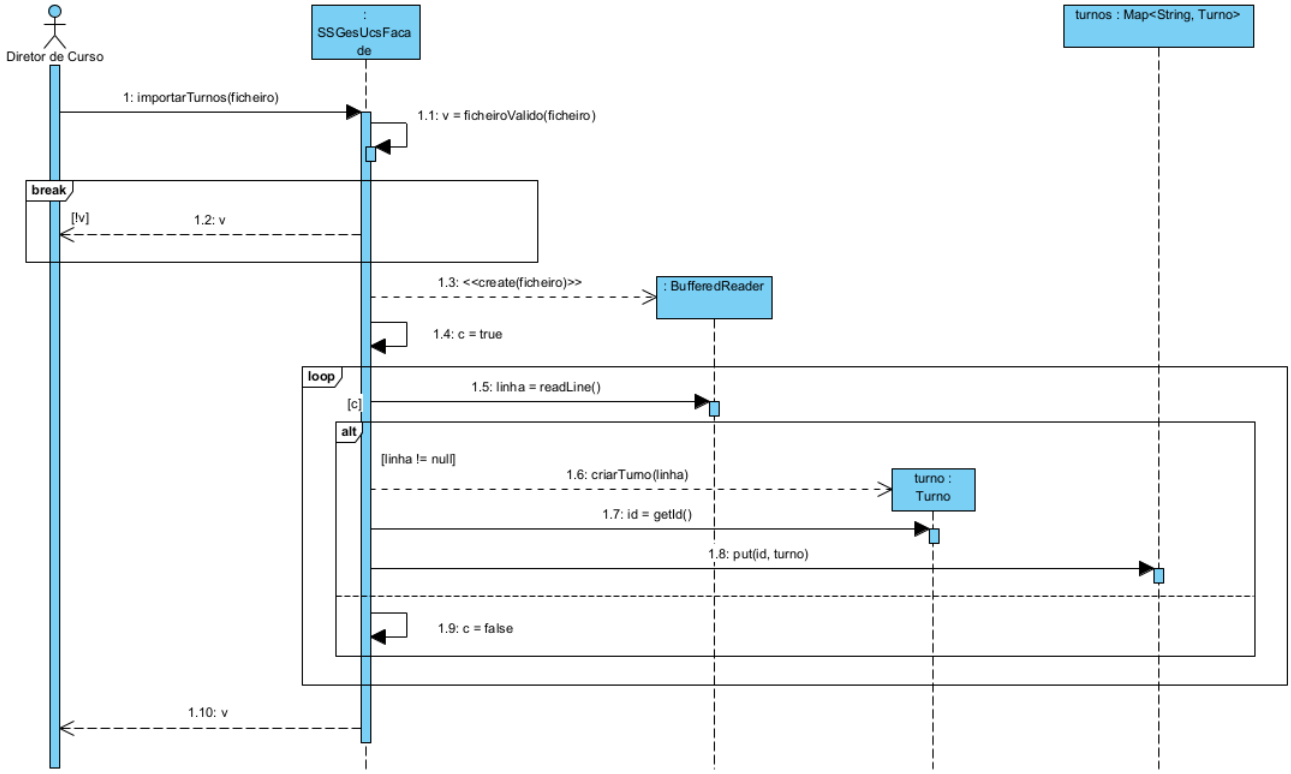


Figura 6: Importar lista de turnos

O diagrama representado na **figura 7** demonstra o processo de verificação da existência de um aluno, pelo que é utilizado em **diversos Use Case**, nomeadamente:

1. Consultar horário de um aluno.
2. Remover aluno.
3. Adicionar turno manualmente a horário.
4. Remover turno manualmente de horário.

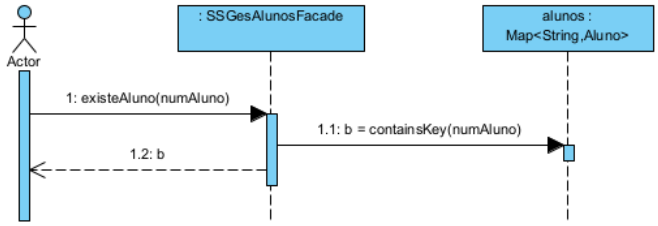


Figura 7: Existe aluno

O diagrama representado na **figura 8** demonstra o processo de verificação da existência de um turno, pelo que é utilizado em **diversos Use Case**, nomeadamente:

1. Editar turno.
2. Adicionar turno manualmente a horário.
3. Remover turno manualmente de horário.

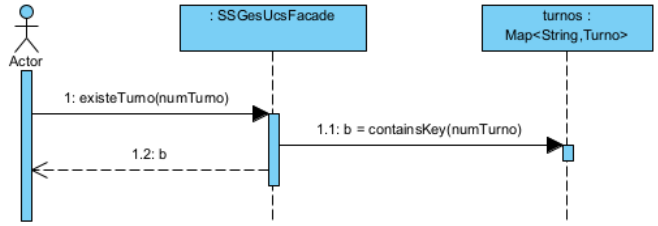


Figura 8: Existe turno

O diagrama representado na **figura 9** é referente ao **Use Case "Adicionar turno manualmente a horário"**, pelo que representa o processo necessário para que o diretor de curso adicione um turno manualmente a um horário já atribuído a um aluno.

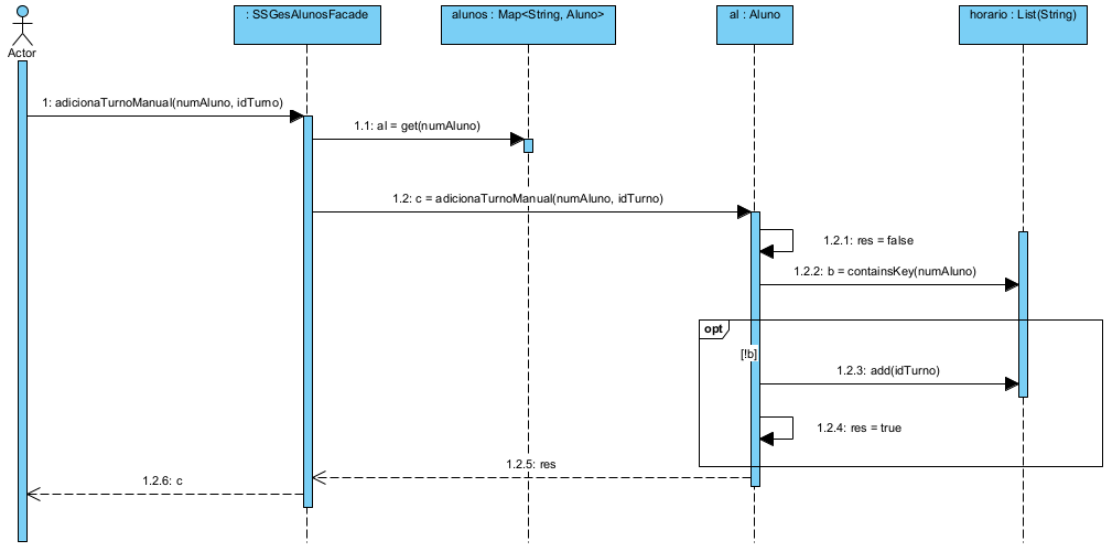


Figura 9: Adicionar turno manualmente a horário

O diagrama representado na **figura 10** é referente ao **Use Case "Remover turno manualmente de horário"**, pelo que representa o processo necessário para que o diretor de curso remova um turno manualmente a um horário já atribuído a um aluno.

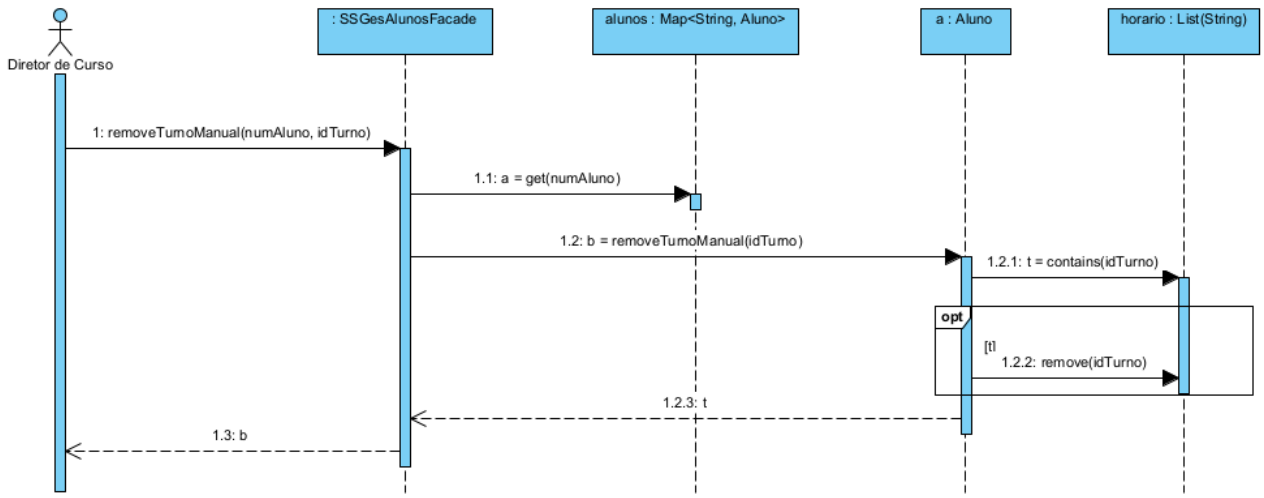


Figura 10: Remover turno manualmente de horário

4 Descrição da solução efetivamente implementada

Ao realizar a transição da fase de modelação concetual para a fase de implementação efetiva, tivemos a necessidade de introduzir **ORM**(**O**bject **R**elational **M**apping) para manter a base de dados independente da lógica de negócios; garantindo melhor escalabilidade e facilitando a manutenção do sistema.

Desta forma, decidimos persistir as classes **Utilizador**, **Turno**, **UC**, **Curso** e **Aluno**, sendo assim estes objetos guardados na nossa base de dados SQL.

4.1 Limitações do nosso sistema

Relativamente à solução implementada, as seguintes funcionalidades foram representadas na modelação lógica mas nesta fase não foram implementadas:

- O nosso sistema só suporta alunos e unidades curriculares dum único curso pelo que, os ficheiros a importar deverão ter em conta esta limitação;
- A geração de horários não tem em conta os semestres, ou seja, os turnos importados para o sistema deverão ser todos dos mesmo semestre;
- Considerámos que a capacidade dos turnos é fornecido no ficheiro a importar, ou seja, qualquer que seja o tipo de turno, a sua capacidade é predefinida no ficheiro;
- Devido ao facto de não termos uma opção de registar utilizador antes de iniciar sessão no sistema, definimos um Diretor de Curso como utilizador base que vai ser registado na base de dados ao iniciar o programa.

Ainda que não tenhamos implementado estas funcionalidades, decidimos na mesma manter, por exemplo, o atributo semestre na classe **UC** para, no futuro, ser possível implementar a geração de horários por semestres. Decidimos também guardar os cursos na nossa base de dados, uma vez que, novamente, caso queiramos no futuro implementar esta funcionalidade, os cursos importados serão automaticamente guardados na base de dados.

4.2 Diagrama de Componentes

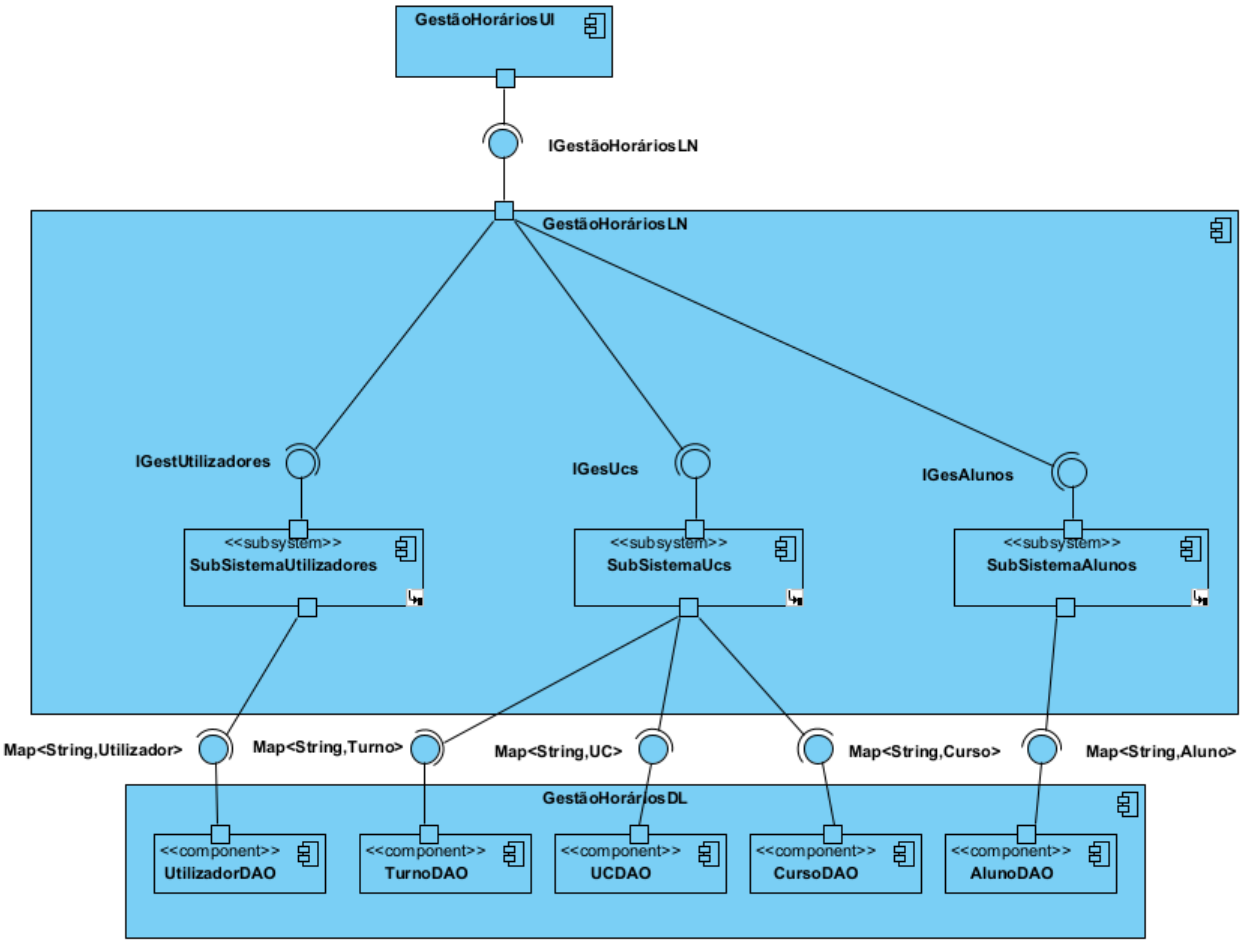


Figura 11: Diagrama de componentes

A imagem acima representa o nosso diagrama de componentes, que foi sendo construído já desde a modelação conceptual, uma vez que já havíamos decidido quais os diferentes subsistemas da nossa lógica de negócios.

O primeiro componente representa a camada UI da nossa aplicação, que permite os nossos utilizadores comunicarem com o nosso sistema.

De seguida temos o componente da lógica de negócios, composto pelos seus vários subsistemas, em que cada um desses é representado pelo seu diagrama de classe, que sofreram alterações (relativas ao diagrama de classe inicial) devido à introdução da persistência de classes.

Por fim, representamos ainda o nosso componente da camada de dados, onde cada um dos subsistemas necessita dos seus correspondentes DAO's, veremos isto em mais detalhe nos diagramas de classe abaixo.

4.3 Diagramas de Classe

Seguem-se os diagramas de classe referentes à **nova implementação**, com os DAO's e com os métodos necessários para o funcionamento correto do nosso sistema:

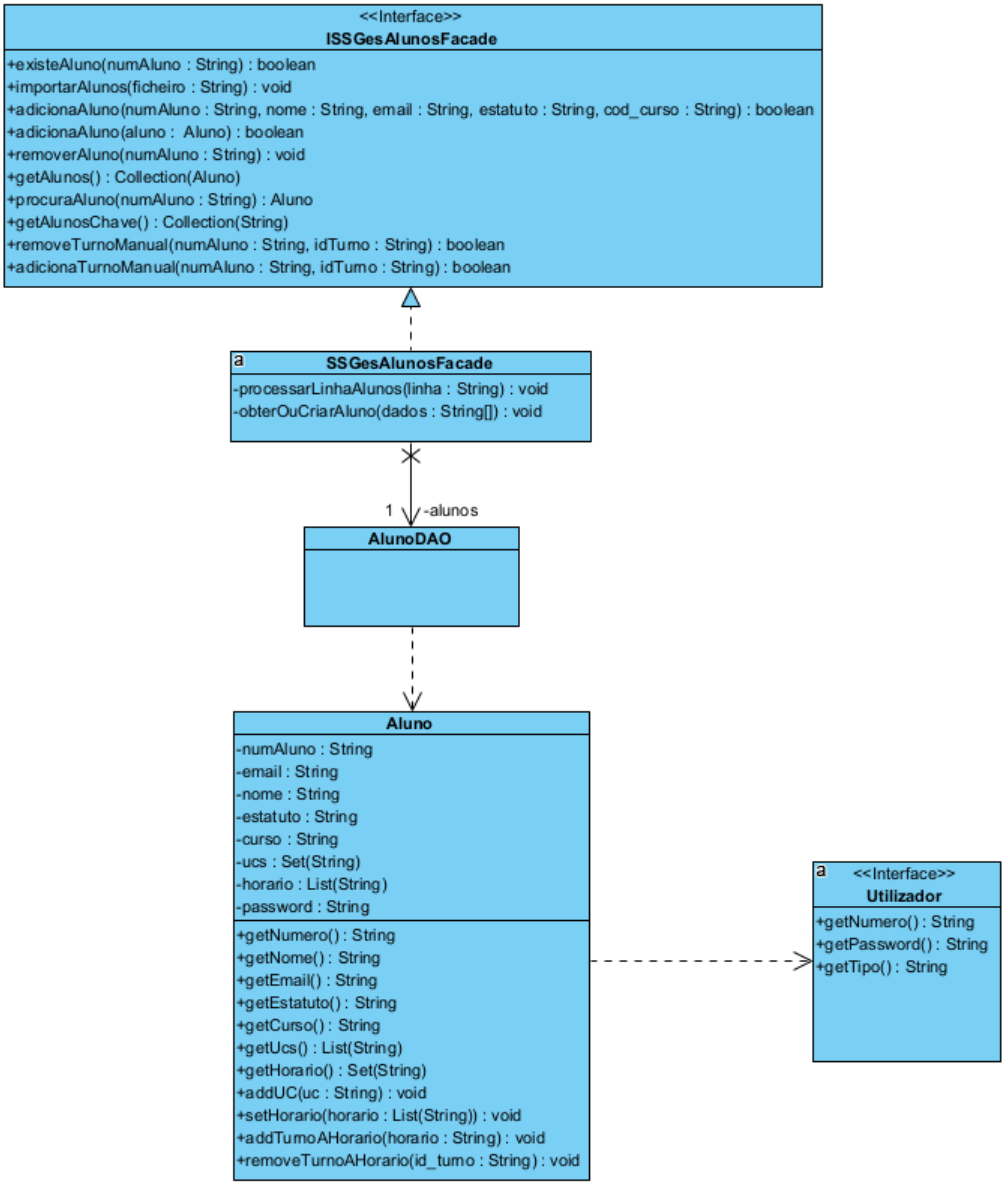


Figura 12: Diagrama de classe do Subsistema de Alunos

Nota: Este diagrama não sofreu alterações em relação à modelação conceptual para além da introdução de "**AlunoDAO**".

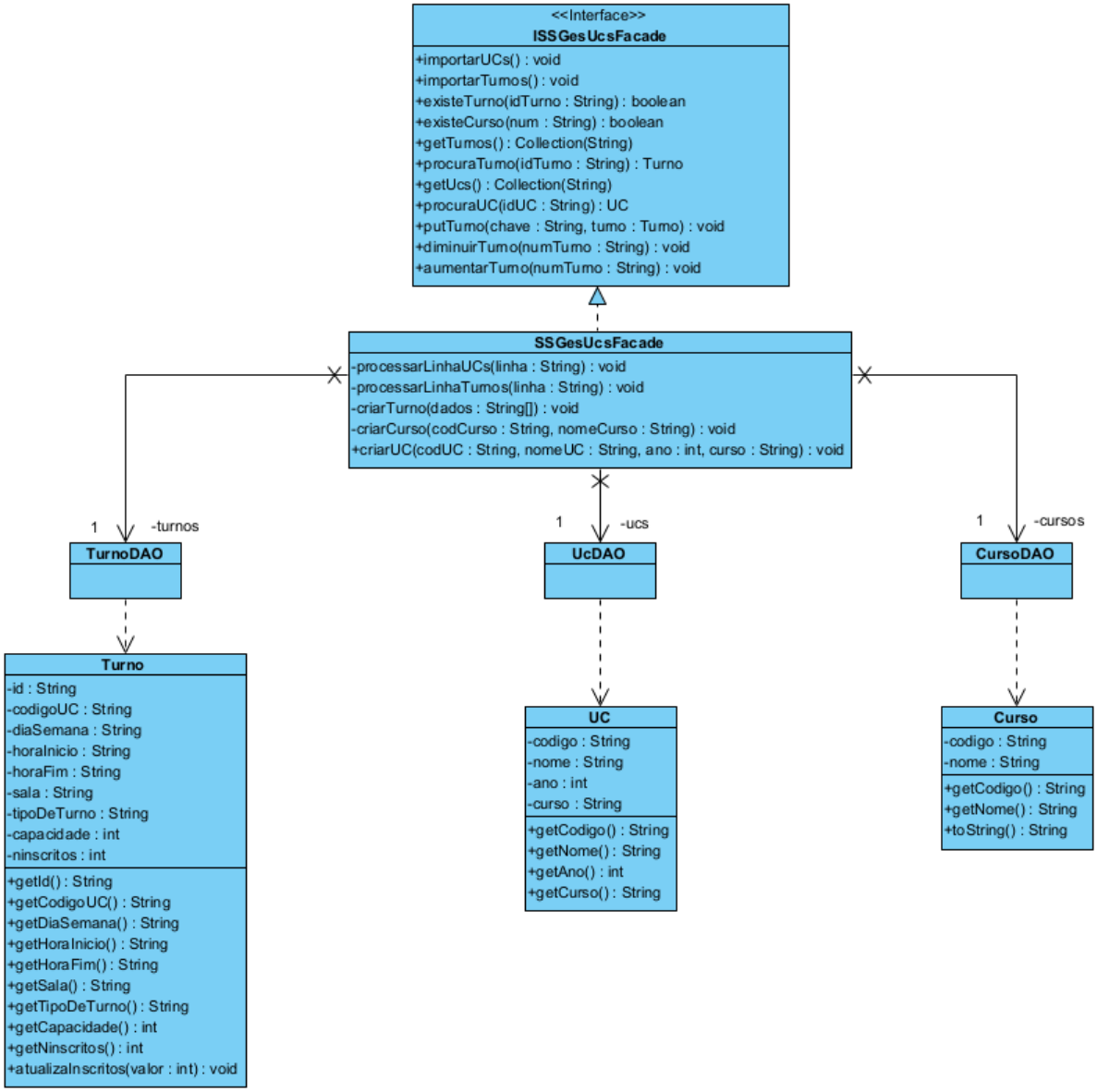


Figura 13: Diagrama de classe do Subsistema de UC‘s

Com a introdução de **DAO**’s a lista de turnos da **UC** deixou de existir, sendo substituída pelo atributo ”codigoUC”, na classe **Turno**, que referência a UC associada. Entre o **Curso** e a **UC** testemunhamos um comportamento similar pelo que na classe **Curso** deixou de existir a lista de UC’s, sendo substituída pelo atributo ”curso”na classe **UC** que referência o respetivo Curso. Desta forma, será a classe **SSGesUcsFacade** que tratará de, por exemplo, o turno procurar a sua UC correspondente com o seu atributo **codigoUC**.

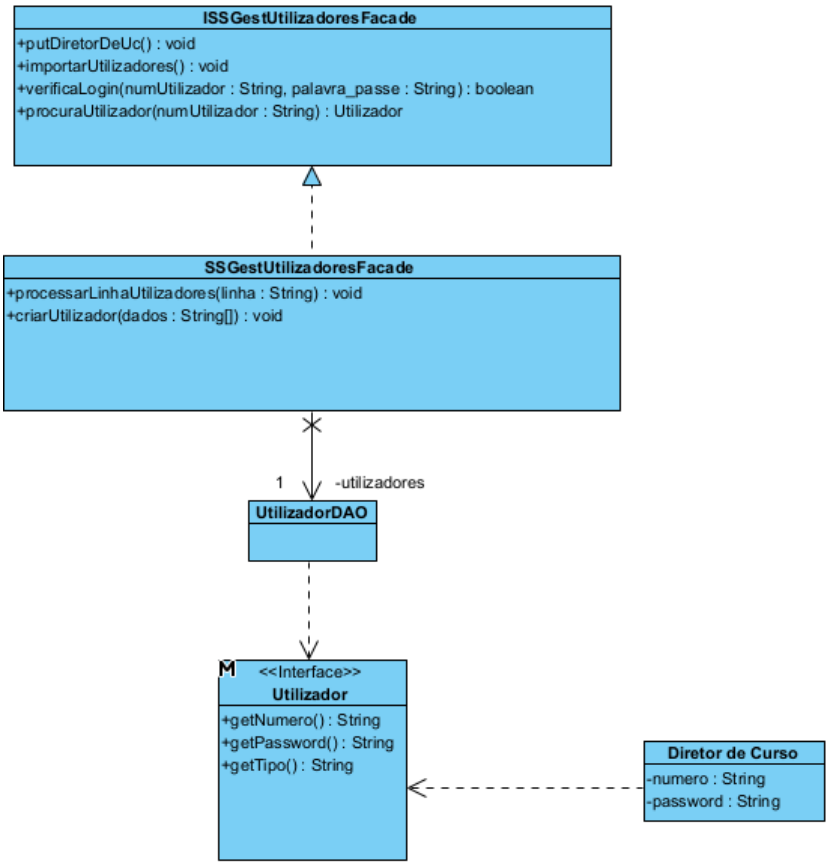


Figura 14: Diagrama de classe do Subsistema de Utilizadores

Decidimos colocar a classe de Diretor de Curso nesta interface, uma vez que a nossa implementação apenas é constituída por um curso, no entanto, numa fase futura teríamos que mudá-lo novamente para o subsistema de UC's, criando uma nova tabela na base de dados para que um curso tenha a si associado um Diretor de Curso.

4.4 Diagramas de Sequência

Após a implementação de ORM decidimos atualizar alguns dos diagramas de sequência anteriores para refletir esta alteração.

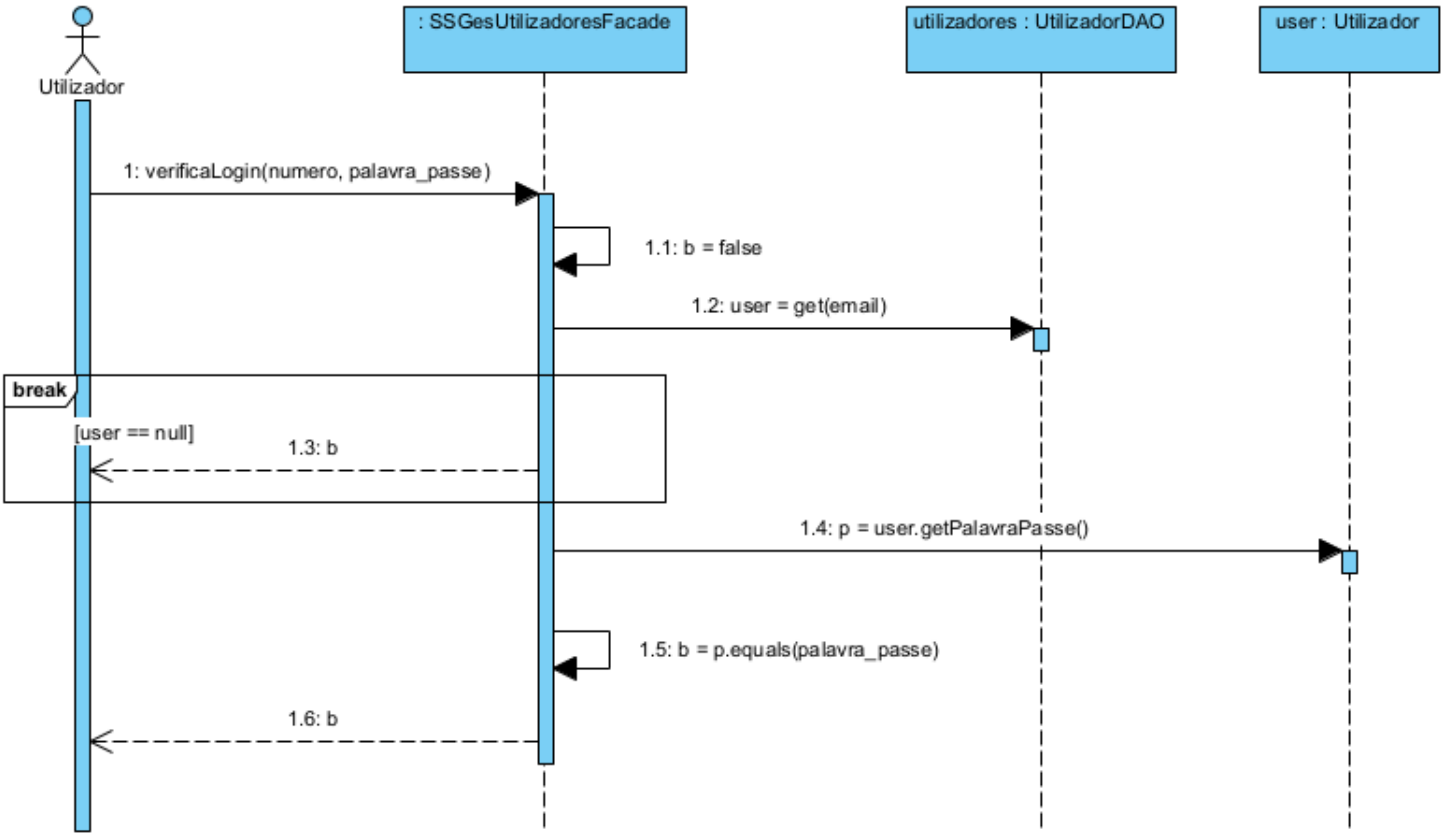


Figura 15: Iniciar Sessão(ORM)

Este diagrama não sofreu alterações para além da substituição do mapa de utilizadores para "**UtilizadorDAO**".

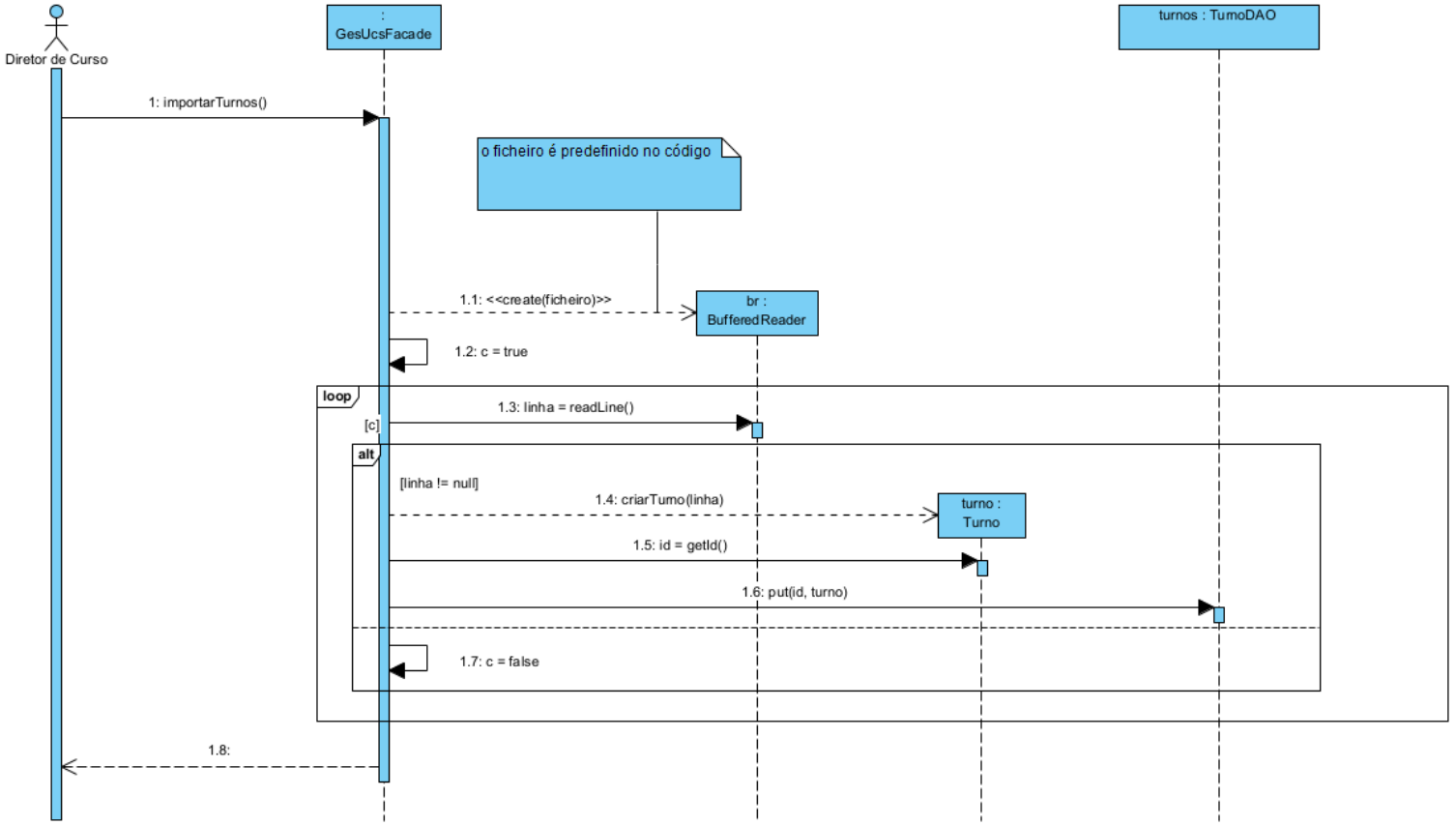


Figura 16: Importar lista de turnos(ORM)

Mais uma vez apenas tivemos de alterar o mapa de turnos pelo "**TurnosDAO**"

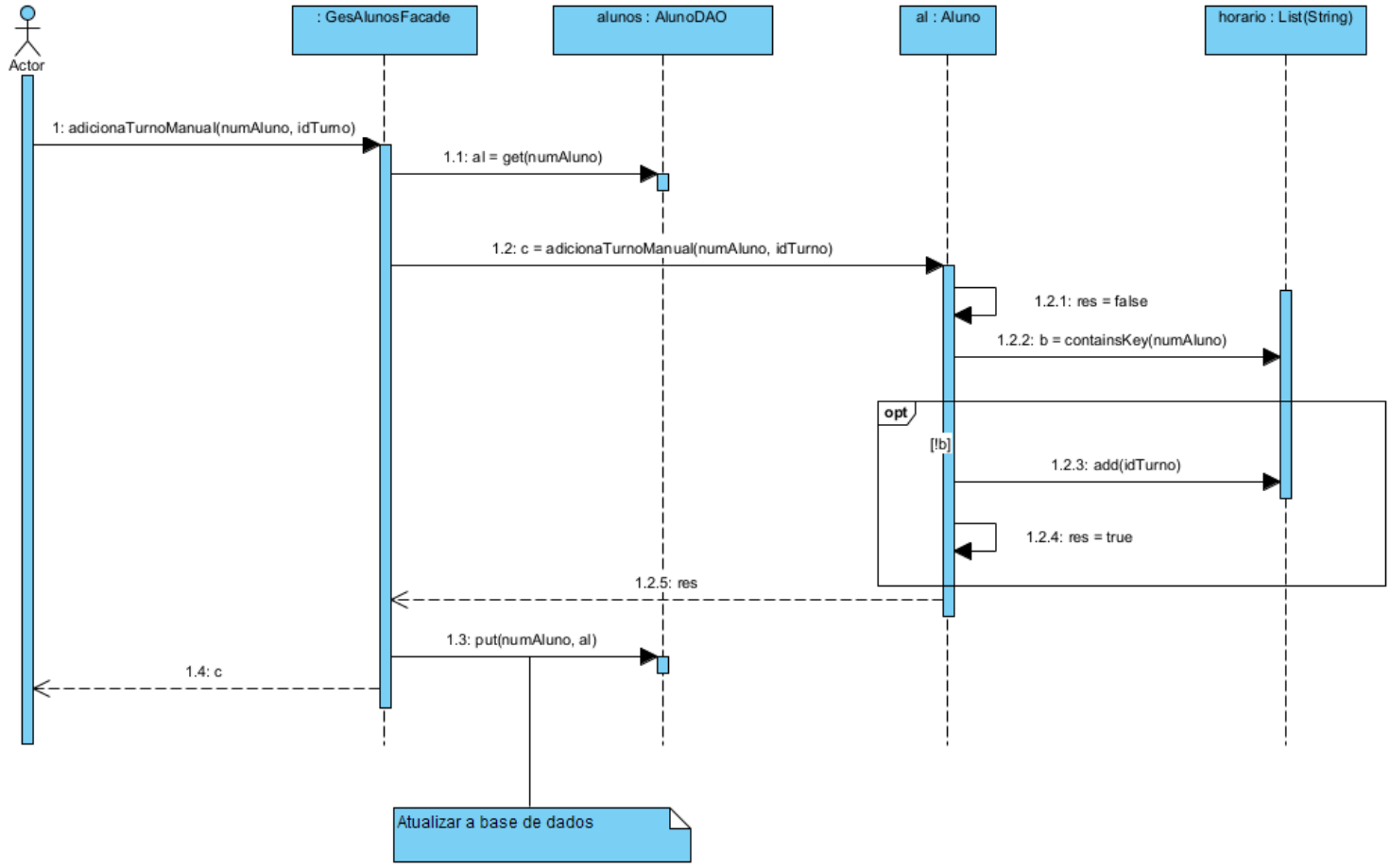


Figura 17: Adicionar turno manualmente a horário(ORM)

Para além de substituir-mos o mapa de alunos pelo "**AlunoDAO**", no final da operação tivemos de adicionar um passo extra para atualizar a base de dados(**1.3**).

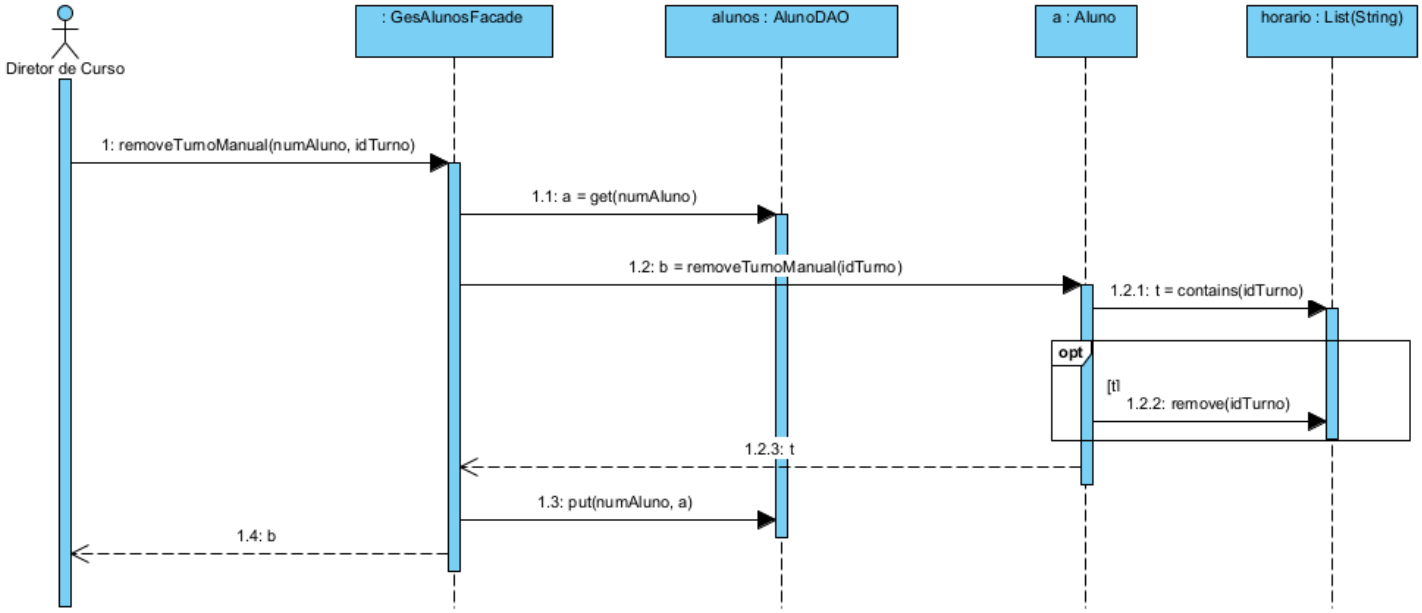


Figura 18: Remover turno manualmente de horário(ORM)

Mais uma vez, para além de substituir-mos o mapa de alunos pelo "**AlunoDAO**", neste diagrama tivemos de atualizar a base de dados(**1.3**).

4.5 Diagrama de Pacotes

Para facilitar a identificação de classes e gestão de subsistemas, fizemos um diagrama de pacotes, responsável por demonstrar as dependências de cada uma das classes dos nossos pacotes.

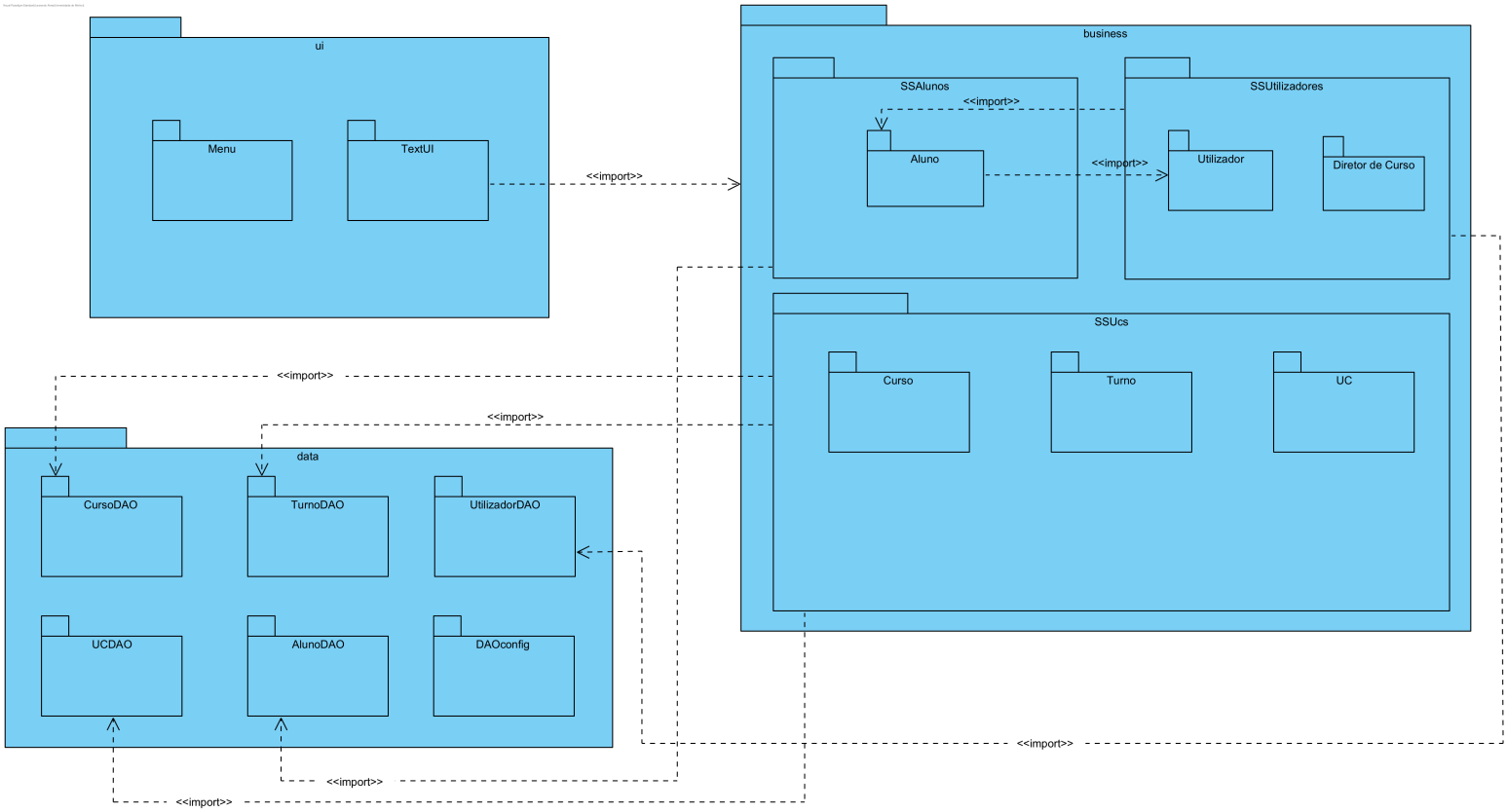


Figura 19: Diagrama de Pacotes

4.6 Manual de Utilização do Sistema

Antes de iniciar o programa vale ressaltar que é obrigatória a inicialização da base de dados correspondente. As linhas de comando que permitem fazê-lo estão registadas no script sql "TrabalhoPraticoDB"da pasta "final". A última linha nesse script serve para apagar a base de dados e está documentada.

Criada a base de dados, podemos iniciar sessão com o registo de Diretor de Curso criado ao iniciar o programa que mencionamos no capítulo 5.1:

```
Bem vindo ao Sistema de Gestão de Horários!
Por favor, inicie sessão:
Número:
duc1
Palavra-passe:
pass
```

Agora que iniciamos sessão no sistema, temos acesso a 4 opções distintas:

1. Operações sobre Alunos
2. Operações sobre Horários
3. Importar Dados
4. Sair

Para aproveitar as operações providenciadas pelo sistema, é recomendado importar dados como primeiro passo. Ao selecionar "Importar Dados", obtemos 3 opções de importação: importar lista de UC's, lista de turnos e alunos; estes devem ser executados nessa ordem para prevenir erros de dependência entre as tabelas na base de dados.

Com os dados importados podemos usar a 4 opção do menu de "Importar Dados"para sair e voltar ao menu principal.

Nas "**Operações sobre Alunos**"temos 3 funções possíveis :

1. Adicionar Aluno

Nesta opção podemos adicionar um aluno ao preencher os campos requisitados; se forem validos o aluno é adicionado.

```
 *** Gestão de Alunos ***
1 - Adicionar Aluno
2 - Consultar Aluno
3 - Listar Alunos
0 - Sair
Opção: 1
Número da novo aluno:
104275
Nome da novo aluno:
Andre
Email da novo aluno:
a104275@uminho.pt
Regime do novo aluno: (enter para sem estatuto)

Codigo de curso do novo aluno:
J3
Aluno adicionado
```

2. Consultar Aluno

Aqui é requisitado o número de um aluno e são impressos os seus dados e turnos atribuídos. Segue-se um exemplo com e sem turnos atribuídos.

```
 *** Gestão de Alunos ***
1 - Adicionar Aluno
2 - Consultar Aluno
3 - Listar Alunos
0 - Sair
Opção: 2
Número a consultar:
200002
Aluno{200002, Pedro Silva, a200002@alunos.uminho.pt, TE, J3, [J301N1]}

Este aluno não tem turnos atribuídos.
```

```
 *** Gestão de Alunos ***
1 - Adicionar Aluno
2 - Consultar Aluno
3 - Listar Alunos
0 - Sair
Opção: 2
Número a consultar:
200060
Aluno{200060, Fernando Sousa, a200060@alunos.uminho.pt, , J3, [J305N3, J304N3]}

ID: 11 | UC: J305N3 | Dia: Sexta | Início: 09.00 | Fim: 11.00 | Tipo: T | Sala: Ed2 0.01 | inscritos: 19 | Capacidade: 100
ID: 13 | UC: J305N3 | Dia: Terça | Início: 18.00 | Fim: 20.00 | Tipo: TP | Sala: Ed1 2.22 | inscritos: 19 | Capacidade: 30
ID: 6 | UC: J304N3 | Dia: Quarta | Início: 14.00 | Fim: 16.00 | Tipo: T | Sala: Ed3 1.01 | inscritos: 34 | Capacidade: 100
ID: 8 | UC: J304N3 | Dia: Terça | Início: 16.00 | Fim: 19.00 | Tipo: TP | Sala: Ed1 1.03 | inscritos: 8 | Capacidade: 30
```

3. Listar Alunos

Finalmente, podemos listar todos os alunos registados na base de dados.

```
 *** Gestão de Alunos ***
1 - Adicionar Aluno
2 - Consultar Aluno
3 - Listar Alunos
0 - Sair
Opção: 3
[Aluno{200104, Luís Fernandes, a200104@alunos.uminho.pt, , J3, [J305N3]}
, Aluno{200050, Roberto Matos, a200050@alunos.uminho.pt, , J3, [J304N3]}
, Aluno{200061, Patrícia Almeida, a200061@alunos.uminho.pt, , J3, [J301N1, J304N2]}
, Aluno{200033, Chris Evans, a200033@alunos.uminho.pt, EINT, J3, [J301N1]}
, Aluno{200105, Fernando Ribeiro, a200105@alunos.uminho.pt, , J3, [J301N1]}
, Aluno{200006, Rui Fernandes, a200006@alunos.uminho.pt, , J3, [J304N3]}
, Aluno{200044, Tomás Marques, a200044@alunos.uminho.pt, , J3, [J301N1]}
, Aluno{200036, João Pinto, a200036@alunos.uminho.pt, , J3, [J301N1]}
, Aluno{200072, Alex Silva, a200072@alunos.uminho.pt, , J3, [J305N3, J304N3]}
, Aluno{200027, João Ferreira, a200027@alunos.uminho.pt, , J3, [J301N1]}
, Aluno{200032, Luís Almeida, a200032@alunos.uminho.pt, , J3, [J304N3]}
, Aluno{200003, Carla Costa, a200003@alunos.uminho.pt, , J3, [J301N1]}
, Aluno{200081, Gabriel Alves, a200081@alunos.uminho.pt, , J3, [J301N1]}
, Aluno{200013, Joana Santos, a200013@alunos.uminho.pt, , J3, [J304N3]}
, Aluno{200093, Mariana Oliveira, a200093@alunos.uminho.pt, , J3, [J301N1]}
, Aluno{200051, Diogo Martins, a200051@alunos.uminho.pt, , J3, [J301N1, J305N3]}
, Aluno{200089, Tiago Matos, a200089@alunos.uminho.pt, , J3, [J301N1]}
, Aluno{200098, Lara Barros, a200098@alunos.uminho.pt, , J3, [J302N1]}
, Aluno{200116, António Pereira, a200116@alunos.uminho.pt, , J3, [J305N3]}
```

Nas "**Operações sobre Horários**"temos, também, 3 operações distintas :

1. **Gerar Horário**

Podemos gerar horários que são atribuídos por todos os alunos. Esta ação é dispendiosa e demora algum tempo a processar devido ao facto de procurar diminuir as colisões.

```
 *** Gestão de Horários ***
1 - Gerar Horários
2 - Atribuição manual de um Aluno
3 - Remoção manual de um Aluno
0 - Sair
Opção: 1
Horarios Gerados!
```

2. **Atribuição manual de um Aluno**

Aqui podemos atribuir alunos a um novo turno de forma manual.

```
 *** Gestão de Horários ***
1 - Gerar Horários
2 - Atribuição manual de um Aluno
3 - Remoção manual de um Aluno
0 - Sair
Opção: 2
Número do aluno:
200060
Identificador do turno:
12
Aluno adicionado ao turno!
```

3. **Remoção manual de um Aluno**

Nesta última opção, possíbilitamos a remoção de um aluno de um turno em que esteja inscrito.

```
 *** Gestão de Horários ***
1 - Gerar Horários
2 - Atribuição manual de um Aluno
3 - Remoção manual de um Aluno
0 - Sair
Opção: 3
Identificador do turno:
12
Número do aluno:
200060
Aluno removido do turno!
```

Após o Diretor de Curso importar a lista de alunos para o sistema, é possível, a esses alunos, iniciar sessão e consultar o seu horário.

```
Bem vindo ao Sistema de Gestão de Horários!
Por favor, inicie sessão:
Número:
200060
Palavra-passe:
pass

 *** Menu ***
1 - Consultar Horário
0 - Sair
Opção: 1
ID: 11 | UC: J305N3 | Dia: Sexta | Início: 09.00 | Fim: 11.00 | Tipo: T | Sala: Ed2 0.01 | inscritos: 19 | Capacidade: 100
ID: 13 | UC: J305N3 | Dia: Terça | Início: 18.00 | Fim: 20.00 | Tipo: TP | Sala: Ed1 2.22 | inscritos: 19 | Capacidade: 30
ID: 6 | UC: J304N3 | Dia: Quarta | Início: 14.00 | Fim: 16.00 | Tipo: T | Sala: Ed3 1.01 | inscritos: 34 | Capacidade: 100
ID: 8 | UC: J304N3 | Dia: Terça | Início: 16.00 | Fim: 19.00 | Tipo: TP | Sala: Ed1 1.03 | inscritos: 8 | Capacidade: 30
```

5 Notas

5.1 Utilizador

Um utilizador refere-se ao utilizador do software de horários, ou seja, ao Diretor de Curso e ao Aluno.

5.2 Consultar Horário e Exportar Horário

O grupo pressupôs que um horário apenas poderá ser consultado ou exportado após a sua publicação.

5.3 Gerar horário

O grupo pressupôs que após a geração de um horário este será automaticamente atribuído ao maior número de alunos possível.

5.4 Alterar horário

O grupo pressupôs que para alterar um turno, é necessário remove-lo do horário e inserir o novo turno manualmente.

5.5 Atribuição manual de turno

O grupo pressupôs que quando um turno é atribuído manualmente, a capacidade da sala não é posta em causa.

6 Conclusão e resultados obtidos

Este trabalho foi essencial para desenvolvermos competências referentes a todo o processo de desenvolvimento de uma aplicação. A nosso ver, achamos que conseguimos atingir os objetivos deste trabalho prático, uma vez que realizámos o trabalho de forma sequencial, desde a elaboração da arquitetura da aplicação, até à implementação do código.

Achamos que existem alguns aspetos que poderiam ser melhorados numa fase futura visto que nem todos os **Use Case** foram implementados e portanto, existem algumas limitações na aplicação relativamente à modelação conceptual, no entanto, a nossa aplicação encontra-se funcional e, com a sua arquitetura, a sua expansão seria relativamente simples.

Relativamente à implementação da aplicação, poderíamos também melhorar a eficiência da função de gerar horários, uma vez que, caso tenhamos uma base de dados com um elevado número de alunos e turnos, o seu tempo de execução será excessivamente alto. Os nossos testes levaram-nos a essa conclusão e, do mesmo modo, notamos que os menus poderiam ser mais intuitivos, com avisos extra, restrições de ações que possam levar a exceções, entre outros.