Desenvolvimento de Sistemas de *Software*

Sistema de Gestão de Turnos

Grupo 16

David Kramer (A77849)
Joaquim Simões (A77653)
José Menezes (A79187)
Rafael Costa (A61799)

Dezembro 2017




Índice

1 Introdução

Este trabalho tem como objetivo o desenvolvimento de uma aplicação de gestão de turnos no contexto de um curso universitário. No início de cada ano letivo existe sempre uma atribuição aleatória de diferentes horários aos alunos, mas nem sempre estes correspondem com aquilo que os alunos desejam. Esta aplicação trata de tentar satisfazer os alunos permitindo que estes efetuem trocas de turnos entre eles, dando prioridade aos alunos com estatuto especial. Adicionalmente, também permite aos professores responsáveis das unidades curriculares fazerem gestão dos respetivos turnos, segundo a administração da direção de curso.

Este relatório pretende apresentar todos os aspetos da aplicação: o seu âmbito, requisitos, diagramas e respetivas explicações, aspetos funcionais e estruturação do código, e fornecer uma análise crítica da mesma.

2 Requisitos

Esta aplicação é orientada para alunos, professores regentes das unidades curriculares, e direção de curso. Como tal, os requisitos são diferentes para cada um deles. De seguida são enumerados os requisitos, para cada uma das entidades mencionadas, que se espera que sejam cumpridos pelo Sistema de Gestão de Turnos.

2.1 Aluno

Relativamente a um aluno o sistema deve suportar dois tipos desta entidade: um aluno de estatuto normal e um aluno de estatuto trabalhador estudante. O segundo tipo apresenta privilégios relativamente às trocas de turnos. De seguida são enumeradas as funcionalidades que o sistema deve suportar no que toca à entidade Aluno:

1. Registar-se em unidades curriculares.
2. Consultar as unidades curriculares inscritas e seus respetivos turnos atribuídos.
3. Solicitar uma troca de turno (no caso do aluno de estatuto normal).
4. Efetuar uma troca de turno direta, caso seja possível (no caso do aluno de estatuto trabalhador estudante).
5. Cancelar um pedido de uma troca de turno.
6. Verificar o estado de uma troca que entrou em fila de espera.

2.2 Docente

1. Consultar as unidades curriculares que leciona e verificar os seus respetivos turnos.
2. Definir a capacidade dos turnos das unidades curriculares que leciona.
3. Remover alunos dos seus turno.
4. Efetuar trocas de turnos a alunos.
5. Adicionar alunos aos seus turnos.
6. Marcar faltas aos aluno dos seus turnos.
7. Visualizar histórico dos seus turnos.

2.3 Direção de curso

1. Configurar o sistema fornecendo um ficheiro contendo a informação das unidades curriculares de um curso, respetivo horário e turnos, lista de professores pertencentes a esse curso e respetivos alunos inscritos nesse curso (este ficheiro deve estar num formato válido conhecido pelo sistema).
2. Informar o sistema que deve proceder ao início da atribuição de turnos aos alunos.
3. Informar o sistema do início das aulas, sendo a partir desse momento apenas possível aos professores responsáveis alterar a composição dos turnos.

2.4 Requisitos do sistema

1. Deve permitir que cada uma das entidades descritas acima possa registar-se e efetuar *login*.
2. Ser capaz de extrair a informação necessária do ficheiro de configuração fornecido pela direção de curso.
3. Atribuir horários (num modo aleatório) aos alunos. Os horários devem ser robustos e flexíveis. evitando que os alunos tenham aulas sobrepostas.
4. Gerir os pedidos de trocas de turnos através de filas de espera.
5. Conceder aos alunos com o estatuto de trabalhador estudante prioridade aos alunos de estatuto normal no que diz respeita às trocas de turnos.
6. Impedir que os alunos efetuem ou solicitem trocas entre si, depois de aulas já terem começado. Além destes requisitos que apresentam funcionalidades do sistema, espera-se que outros requisitos não funcionais sejam cumpridos. Espera-se que a *User Interface* da aplicação seja intuitiva, que o tempo de resposta das diferentes funcionalidades do sistema aos

diversos tipos de clientes sejam o menor possível. Finalmente, os dados armazenados pela aplicação devem ocupar o menor espaço de memória possível.

3 Modelo de Domínio

No Modelo de Domínio apresentamos as Entidades que compõem o problema e os seus Relacionamentos. No contexto do sistema desenvolvido tomamos como Entidades tanto os utilizadores do sistema, nomeadamente Alunos e Professores, como também as partes vitais do mesmo, por exemplo os Turnos e Horários - e ainda algumas entidades funcionais do programa como Filas de Espera.

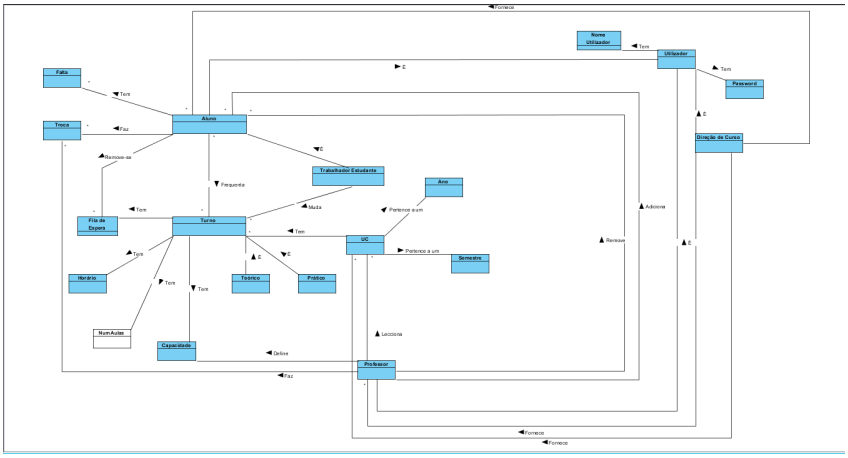

Figura 1: Modelo de Domínio

4 Diagrama de Classes

Definir um diagrama de classes é a base de qualquer sistema Orientado a Objetos. Através deste tipo de diagrama obtemos informações relativas a todas as classes de um sistema, como as suas variáveis,funções e a forma como se associam a outras classes.

De modo a desenvolver o diagramas de classes dividimos as *packages* existentes relativas a este projeto criando dois modelos: *Business* e *Data*. Dentro de cada um destes apresentamos todas as variáveis e funções relativas às classes pertencentes a cada *package*, tendo sempre em consideração a visibilidade de cada atributo e função assim como o tipo relação existente entre classes. Entre estes tipo de relações, foram usadas as seguintes: associação, dependência e generalização.

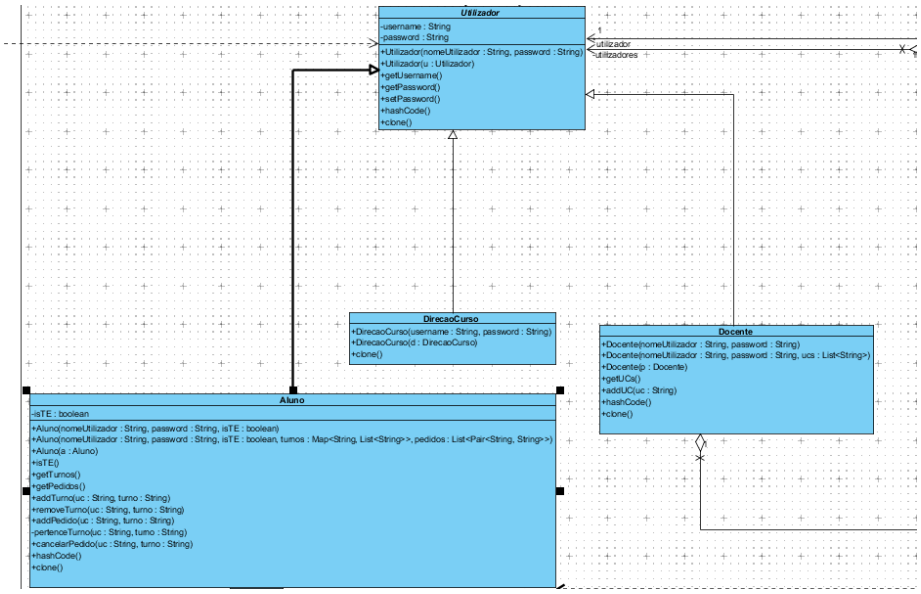

Figura 2: Generalização

Como é possível ver pela a imagem, as classes Aluno, DirecaoCurso e Docente são classes mais específicas (subclasses) da classe mais geral Utilizador. Assim, estas três subclasses são relacionadas com a classe Utilizador através de uma generalização.

De forma a demonstrar como usamos os restos das relações entres classes e visibilidade dos atributos/funções iremos analisar a classe Turno da *package Business*:

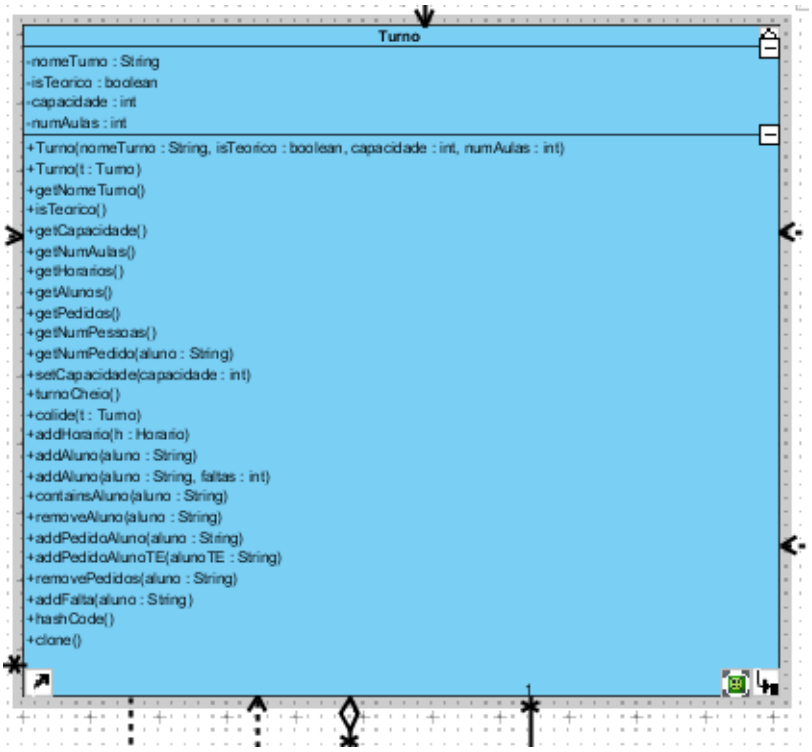

Figura 3: Classe Turno

Quanto à visibilidade, verificamos que atributos como nomeTurno ou numAulas (por exemplo) têm um nível de acesso privado. Relativamente às associações verificamos que para representar o atributo horários usamos uma relação do tipo associação com multiplicidade "*", uma vez que se trata um turno tem vários horários. Finalmente, o uso da dependência pode ser justificado, por exemplo, pela função "addHorario", que recebe uma variável do tipo horário, criando assim uma dependência entre a classe Turno e a classe Horário.

5 Diagramas de *Use Case*

Os diagramas de *use case* definem as interações entre atores e o sistema. Neste caso os atores serão os utilizadores, ou seja, Professores, Alunos e a Direção de Curso. No contexto do problema criamos dois diagramas de *use case*, um deles sendo o diagrama de *use case* geral que apresenta todas as interações entre atores e o sistema, e outro mais específico que detalha uma interação mais complexa, a Gestão de Turnos por parte de um Docente.

5.1 Diagrama *Use Case* geral

Neste diagrama, como mencionado acima, apresentamos todas a interações entre os atores e o sistema.

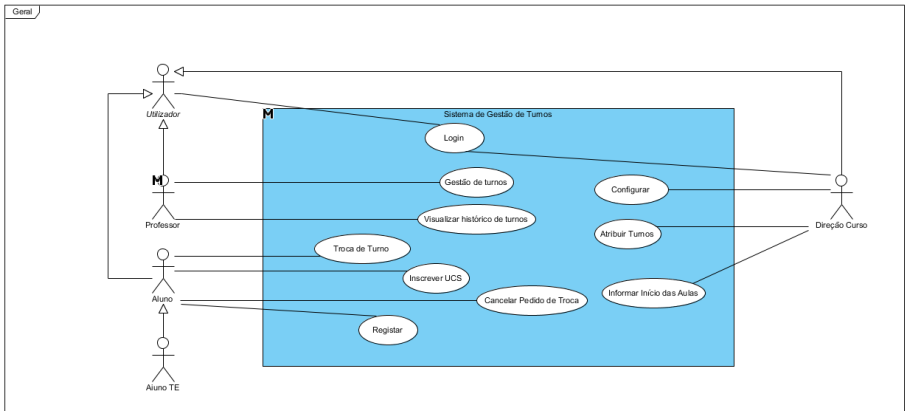

Figura 4: Diagrama de *Use Case* geral

5.1.1 *Use Case Login*

É pedido ao utilizador os seus dados de *login* começando pelo nome de utilizador e de seguida a *password*, caso estes sejam válidos é-lhe informado que o *login* foi efetuado com sucesso, caso contrário informa que os dados são inválidos.

5.1.2 *Use Case* Registo

Para o utilizador se registar é necessário que este insira os dados (nome de utilizador e *password*) que pretende utilizar para posteriormente efetuar o *login*. Após este primeiro passo o sistema valida os dados e informa que este foi registado com sucesso (caso seja o caso). Pode porém acontecer que os dados já tenham sido usados por outro utilizador procedendo ao cancelamento da operação. É possível também que os dados inseridos sejam inválidos procedendo também ao cancelamento da operação.

5.1.3 *Use Case* Troca de Turno Normal

Numa primeira etapa é mostrado ao aluno a lista de turnos a que este está inscrito. Após a seleção do turno que pretende mudar é-lhe mostrado de novo uma lista dos turnos existentes referente à UC que este tenciona trocar. Depois da seleção do turno pretendido o sistema verifica se existem eventuais candidatos em fila de espera que coincidam com a troca pretendida. Caso se verifique, é eliminado o aluno do turno de destino da sua posição na fila de espera do turno de origem. De seguida, é efetuada a troca entre os dois alunos.

No entanto, caso não exista nenhum aluno em fila de espera que coincida com a troca pedida este é colocado em fila de espera do turno destino e é de seguida informado do sucedido.

5.1.4 *Use Case* Inscrever UCs

Inicialmente é mostrado ao aluno a lista de UCs a que se pode inscrever, este seleciona as pretendidas. De seguida, o sistema inscreve-o nas UCs selecionadas e informa-o que a inscrição foi efetuada com sucesso.

5.1.5 *Use Case* Cancelar Pedido de Troca

O sistema mostra inicialmente a lista de pedidos de troca realizados pelo aluno, este seleciona um dos pedidos e indica que o pretende cancelar. O sistema elimina então o pedido e informa o aluno. Existe no entanto o caso em que o aluno desiste de cancelar o pedido após a seleção do mesmo sendo que o sistema indica nesse caso que o processo foi cancelado.

5.1.6 *Use Case* Configurar

Num primeiro passo a direção de curso carrega o ficheiro com a configuração pretendida (Lista de alunos, UCS, etc) o sistema então é configurado e informa que esta operação foi realizada com sucesso (caso seja o caso). É possível, no entanto, que o ficheiro não esteja no formato correto sendo que o sistema informa o utilizador procedendo ao cancelamento da operação.

5.1.7 *Use Case* Atribuir Turnos

Neste *Use Case* o utilizador indica que pretende fazer a atribuição dos turnos e de seguida o sistema procede à distribuição dos alunos nos diferentes turnos indicando posteriormente que o procedimento foi efetuado com sucesso.

5.1.8 *Use Case* Informar Início das Aulas

Após o utilizador indicar que as aulas já iniciaram o sistema procede ao cancelamento e ao impedimento de quaisquer trocas de turnos informando de seguida que o pedido foi efetuado com sucesso.

5.2 Diagrama *Use Case* Gestão de Turnos

Consideramos que a "Gestão de Turnos" necessitaria de um diagrama *Use Case* próprio tendo em conta a sua complexidade de funções.

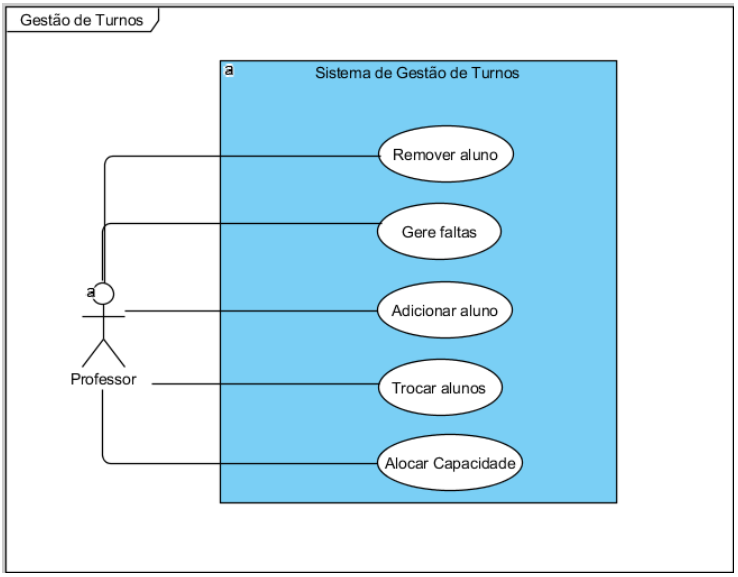

Figura 5: Diagrama de *Use Case* Gestão de Turnos

5.2.1 *Use Case* Remover aluno

Neste *Use Case* o sistema começa por apresentar ao utilizador uma lista com todas as UCs lecionadas pelo mesmo. Após a seleção da mesma e do turno pretendido é-lhe apresentada uma lista com todos os alunos inscritos no turno em questão. O utilizador seleciona depois o aluno que pretende remover, procedendo o sistema à sua eliminação. No final, informa que a operação foi realizada com sucesso.

5.2.2 *Use Case* Adicionar aluno

Para adicionar um aluno a um determinado turno de uma UC, é mostrado ao utilizador uma lista com as UCs que este leciona, de seguida e após a seleção da UC pretendida, é pedido que este escolha um aluno de uma lista de alunos inscritos na UC. Seguidamente, são apresentados os turnos que não estejam cheios para que o professor escolha o turno destino. Cabe a partir deste passo ao sistema proceder à remoção do aluno do seu turno antigo, adiciona-lo ao seu novo turno, remove-lo da fila de espera da UC em causa e, por fim, informar que o aluno foi adicionado com sucesso.

5.2.3 *Use Case* Trocar alunos

Mais uma vez é apresentado ao Professor a lista de UCs que este leciona. Após a seleção da UC pretendida é pedido ao professor que selecione de uma lista de alunos inscritos os dois alunos que este pretende trocar. A partir daqui o sistema começa por verificar os turnos dos alunos em questão, removendo o

primeiro aluno do seu turno de origem e de eventuais filas de espera que este estivesse inserido. O mesmo processo é efetuado para o segundo aluno. Os dois alunos são então inseridos nos seus novos turnos e o sistema informa o utilizador que a troca foi realizada com sucesso.

5.2.4 *Use Case* Alocar Capacidade

Após o utilizador selecionar a UC e o turno pretendido é-lhe pedido que este indique a nova capacidade desse turno. O sistema verifica então o valor indicado e, caso seja válido, procede à sua alteração informando-o que o processo foi realizado com sucesso. Caso o valor indicado não seja válido, este indica que o valor introduzido é inválido procedendo ao cancelamento do processo.

6 Diagramas de Sequência de Subsistemas

Os Diagramas de Sequência de Subsistemas surgem após a introdução de subsistemas do projeto aos "Diagramas de Sequência (*Use Case*)" sendo que estes nos permitem visualizar de forma mais aprofundada o funcionamento dos diferentes processos tendo em conta os subsistemas adicionados (*BusinessFacade*, Docente, UC, Turno, Aluno, *DataFacade*, UCDAO, AlunoDAO). Apresentamos, de seguida, os diagramas que consideramos mais importantes, entre os quais "TrocaTurnoDSS"e "RemoverAlunoDSS".

6.1 Diagrama de Sequência TrocaTurnoDSS

Este diagrama demonstra a operação de troca ou pedido de troca de turno por um aluno. Inicialmente temos a *UI* a pedir ao subsistema *BusinessFacade* as UCS às quais o aluno está inscrito. De seguida é pedido que este escolha a UC e o turno de destino. A partir deste momento avança para o *BusinessFacade* de modo a recolher a lista de alunos inscritos à UC e de seguida para a "Gestão de Turnos" onde vai ser realizada a troca. Inicialmente, verifica-se se o aluno é ou não trabalhador estudante. Caso seja e o turno não esteja cheio a troca é efetuada imediatamente. Caso contrário verifica se existe algum aluno em fila de espera para o turno origem que pertença ao turno de destino (do aluno que requisitou a troca) e, caso isto se verifique, procede-se à troca de turno (a função "EfetuarTroca" efetua tanto a troca de turno como a eliminação do aluno de eventuais filas de espera a que este pertença). No último caso (ninguém em fila de espera), se o aluno for "Trabalhador Estudante" é adicionado à fila de espera correspondente ao turno destino na primeira posição, caso contrário é adicionado no fim e, finalmente, é atualizada a base de dados. Por fim, caso tenha sido efetuada a troca, é informado o utilizador que esta foi efetuada com sucesso. Caso contrário é informado que foi colocado em fila de espera.

6.2 Diagrama de Sequência RemoverAlunoDSS

Em primeiro lugar, de modo a apresentar a lista de turnos de todas as UCS relacionadas com o Docente, a *UI* pede todas as UCs e, de seguida, todos os turnos a elas associados. Após a seleção do turno pretendido obtêm-se os alunos deste, sendo apresentados ao docente. Após a escolha do aluno que se pretende remover este é eliminado do turno e de seguida decorre a remoção do turno em causa dos turnos do aluno, assim como a atualização da base de dados. Na etapa final é então informado o utilizador que a remoção foi realizada com sucesso.

6.3 Diagrama de EfetuarTroca

Começamos por adicionar o aluno ao turno de destino e de seguida por removê-lo do turno de origem. Numa segunda etapa é necessário remover o aluno de quaisquer filas de espera que estivesse inserido. De modo a atualizar os dados do aluno é necessário remover o antigo turno da lista de turnos do aluno e adicionar o novo turno à lista. Finalmente, atualiza-se a base de dados (no *dataFacade*) procedendo mais uma vez à remoção do turno antigo, adição do novo turno e remoção de quaisquer pedidos existentes referentes à UC em questão.

6.4 Diagrama de CalcTurnos

O cálculo de turnos para os diferentes alunos inscritos é um processo bastante complexo e, por isso, foi dividido em vários sub-diagramas. A saber: CalcTurnos, CalcTurnosTeo, CalcTurnosPraticos, GetUCsOrdenadas, FiltrarAlunosInscritos, GetTurnosAluno e TrocarAluno. De notar que a solução gerada não é a solução ótima, ou seja, não é a solução com menor número de colisões entre horários dos turnos inscritos dos alunos. Para se chegar a essa solução deveriam-se gerar todas as soluções possíveis e destas selecionar a melhor (obviamente utilizando *backtracking* como otimização). No entanto a solução proposta é bastante mais eficiente e atinge a meta estipulada (permite que não haja colisões de horários no mesmo ano).

Começa-se por ordenar as diferentes unidades curriculares por ordem descrescente de ano. Isto vai permitir que se inscrevam primeiro os alunos de anos superiores e permitir que ao menos as unidades curriculares dos seus anos não possuam colisões entre si, antes de os inscrever a unidades curriculares a que reprovaram. Depois disso, os alunos são inscritos nos turnos teóricos das unidades curriculares. No caso de uma unidade curricular possuir mais que um turno teórico, os alunos são distribuídos equitativamente entre esses turnos.

Finalmente inicia-se o processo de inscrição dos turnos práticos. Inicialmente ordenam-se as unidades curriculares com menor número de turnos práticos. Isto porque estas não têm tanta versatilidade de horários como unidades com muitos turnos práticos. Ao se inscrever um aluno num turno pode acontecer um dos seguintes casos:

- O turno não está cheio e não há colisões nos horários do aluno - o aluno é inscrito nesse turno.
- O turno está cheio e não há colisões nos horários do aluno - tenta-se trocar algum dos alunos inscritos nesse turno para outro turno.
- Os horários do aluno não coincidem com nenhum turno - inscreve-se o aluno num turno que não esteja cheio.

7 Diagrama de Máquinas de Estado

Os diagramas de Máquinas de Estado permitem modelar o comportamento do sistema, representando todos os estados que o sistema atravessa em resposta aos eventos que podem ocorrer. No caso deste sistema, todas as alterações de estado do sistema são derivadas de ações efetuadas pelo utilizador. Assim sendo, modelamos vários diagramas de modo a englobar todas as mudanças de estado que o sistema atravessa, correspondentes às várias ações que o utilizador pode realizar. Começamos por modelar o diagrama de ME (Máquinas de Estado) de Registo, visto ser a primeira ação comum a todos os utilizadores e sem a qual nenhum outro estado do sistema pode ser atingido. Depois modelamos o *Login*, um processo que toma vários estados conforme o diferente tipo de utilizador - aluno, professor ou direção de curso. Depois modelamos diagramas ME para os estados possíveis a cada utilizador.

No caso da Direção de Curso, o respetivo diagrama de ME engloba apenas um estado, e todos os eventos associados (correspondentes às ações possíveis por parte do utilizador) podem apenas alterar o estado no caso de *logout* - de resto o programa permanece no mesmo estado - isto é, na única página associada à Direção de Curso.

O diagrama ME relativo ao Professor engloba vários estados e tem associado os diagramas ME de Gestão de Turnos e de Marcar Faltas, visto serem eventos compostos por diferentes estados.

O diagrama de ME Aluno também engloba vários estados, relacionados aos vários eventos que podem advir das ações do utilizador: Este pode-se inscrever em UCs, ou escolher trocar de turno - o processo de troca de turno tem o seu próprio diagrama ME, visto ser composto por vários estados.

8 Diagrama de Atividades

O Diagrama de Atividades apresenta-se ainda como outra forma de modelar o comportamento do sistema. Os elementos principais deste diagrama são as ações e os dados que são transmitidos entre estas, formando o fluxo do sistema, desde o seu início ao seu término. No caso do presente Sistema de Gestão de Turnos, dividimos o sistema em três partições, que correspondem às três entidades capazes de realizar ações: Aluno, Professor e Direção de Curso.

O fluxo do sistema começa com a sua configuração que é realizada pela Direção de Curso com o carregamento do ficheiro de configuração, isto é, o ficheiro que contém todas as informações necessárias para o sistema configurar a sua base de dados, possibilitando a partir daí as ações subsequentes pela parte dos utilizadores. Aquando do carregamento do ficheiro de configuração os alunos podem-se inscrever nas respetivas UCs e os professores podem alocar a capacidade dos turnos - estes utilizadores estão limitados a estas ações até que a Direção de Curso despolete a atribuição de horários (feita pelo sistema). A partir daí os alunos podem efetuar as ações de pedidos de troca de turno e os professores podem, para além de alocar a capacidade dos turnos, adicionar, remover, e trocar alunos entre turnos, tal como visualizar o histórico de cada turno.

A partir do momento em que a Direção de Curso realiza a ação de indicar o início das aulas (ao que nos referimos no diagrama, por simplicidade, como *Iniciar Aulas*), os alunos são impossibilitados de realizar mais trocas; os professores podem efetuar, para além das ações anteriormente possíveis, a marcação de faltas. O diagrama termina com a ação simbólica correspondente ao final do Semestre - em termos práticos isto significa que um novo ficheiro de configuração será carregado pela Direção de Curso, recomeçando o ciclo do sistema.

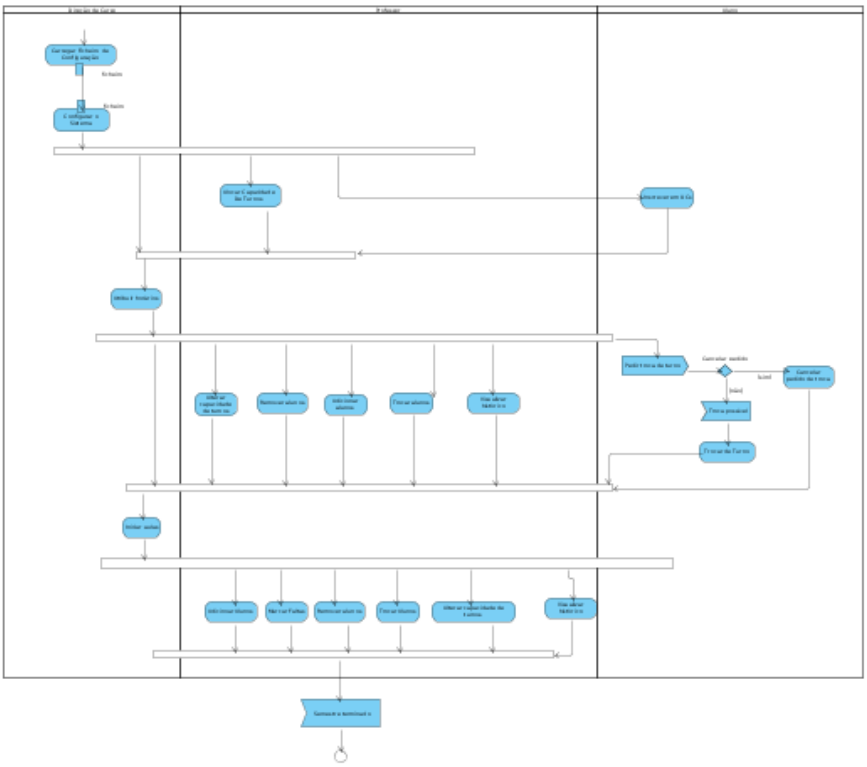

Figura 6: Diagrama de Atividades

9 Diagrama de *Package*

Os diagramas de *package* são úteis para identificar as dependências entre um grande número de classes e assim entre *packages*. Assim, uma vez que entres as três camadas existentes (*Presentation*, *Data* e *Business*) existem duas *façades* (*DataFacade* e *BusinessFacade*), as relações entre *packages* são apenas representas por acessos da *presentation* à *BusinessFacade* e da *BusinessFacade* à *DataFacade*. Como consequência, a *package Presentation* acaba por ser dependente da *Business* que por sua vez depende da *package Data*.

10 Diagrama de Instalação

Com o desenvolvimento do diagrama de instalação identificamos os diferentes componentes necessários para a instalação da aplicação. Como a aplicação permite a leitura e o *parsing* de ficheiros em formato *JSON* fornecidos pela direção de curso para a configuração do sistema, deve ser utilizado o ficheiro

jar json-simple. Foi fornecido um ficheiro "init.json" que contém alguns dados relativos a algumas unidades curriculares, docentes e alunos.

Por outro lado a aplicação usa como recurso à persistência de dados uma base de dados em *MySQL* através da ferramenta *JDBC*. Assim, o projeto fornece dois *scripts*, a saber: o *script* de criação da base de dados e o *script* de populamento da mesma. Recomedamos a execução do *script* de populamento ao invés do ficheiro de configuração *json* fornecido, para uma melhor experiência com a aplicação. O *script* de populamento apresenta uma amostra significativa com unidades curriculares e turnos extraídos do portal académico. Foram recolhidos todos os turnos e horários dos três primeiros anos de Engenharia Informática do primeiro semestre do presente ano letivo. O *script* apresenta a lista dos diferentes docentes (de um modo não realista). Os nomes de utilizador dos docentes apresentam o seguinte formato (pA0X) em que "A" corresponde ao ano do curso e "X" a um número figurativo do docente. Finalmente, são fornecidos trezentos alunos (cem por cada ano) e que já estão inscitos às diferentes unidades curriculares. Para facilitar a utilização da aplicação, todos os utilizadores possuem a *password* "pass".

11 Camadas Aplicacionais

A nossa aplicação está dividida em três camadas, a saber *Presentation layer*, *Business layer* e *Data layer*. As camadas seguem uma hierarquia em que cada camada apenas acede a camada de hierarquia inferior. Neste caso temos que a *Presentation* apenas acede a *Business* e, por sua vez, a *Business* apenas acede a *Data*. De notar que os diferentes acessos são efetuados através de *façades*. Assim sendo, as camada de *Business* e *Data* apresentam as suas próprias *façades*. Nas categorias seguintes são descritas, com detalhe, estas três camadas.

11.1 *Presentation Layer*

Esta camada é responsável pela interação entre o utilizador e a aplicação propriamente dita. Como a aplicação suporta três diferentes tipos de utilizador (Direção de Curso, Docente e Aluno) foram desenvolvidas diferentes páginas para cada um destes tipos. Além das páginas mencionadas, foi criada mais uma página (*Home*) responsável pelo *login*\registo de um Utilizador. Optou-se por limitar ao máximo o número de diferentes páginas da aplicação, já que um elevado número de páginas distintas pode fazer com que uma aplicação se torne pouco intuitiva.

Esta camada é também responsável por eventuais leituras e escritas de ficheiros. A aplicação usa como recurso um ficheiro de configuração ("config.txt") de modo a poder verificar se as aulas já começaram ou não. Por outro lado, a aplicação encontra-se preparada para efetuar a leitura e o *parsing* de um ficheiro de configuração fornecido pela direção de curso de modo a iniciar o sistema com a informação relativa aos horários, turnos e diferentes unidades curriculares,

bem como a lista de docentes e alunos envolvidos no curso. Este último ficheiro deverá estar no formato *JSON* (no projeto é fornecido um ficheiro "init.json" de teste com algumas unidades curriculares, docentes e alunos).

Os diferentes pedidos efetuados pelo utilizador são enviados à *Business Façade* e, depois de esta os processar, são ilustrados ao utilizador com recurso às diferentes janelas desenvolvidas em *Swing*.

11.2 *Business Layer*

A *Business Layer* trata de toda a lógica e cálculos envolvidos na aplicação e responde aos pedidos efetuados pela *Presentation Layer*. É esta camada que determina os diferentes turnos atribuídos aos alunos inscritos e que trata de todas as trocas de turnos efetudas por alunos ou docentes.

Sempre que a camada é inicializada ela carrega todos os dados, de uma forma não persistente, relativos às diferentes unidades curriculares e respetivos turnos, assim como de todos os utilizadores envolvidos na aplicação. Sempre que é efetuado um pedido pela *Presentation*, esta camada efetua todos os cálculos necessários e determina se é necessário atualizar alguma informação resultante do pedido. Caso seja necessário, esta camada envia um pedido à *Data Layer* de modo a atualizar os dados armazenados na base de dados.

11.3 *Data Layer*

Esta última camada é responsável pela persistência dos dados produzidos pela aplicação. Os dados são guardados numa base de dados *MySQL* através da ferramenta *JDBC*. Esta camada é acedida através da *Data Façade* que recebe pedidos provenientes da *Business Façade*. Sempre que recebe um pedido esta camada usa o *Data Access Object* apropriado de modo a persistir a informação na base de dados. A seguinte figura ilustra o modelo lógico desenvolvido para a criação da base de dados da aplicação.

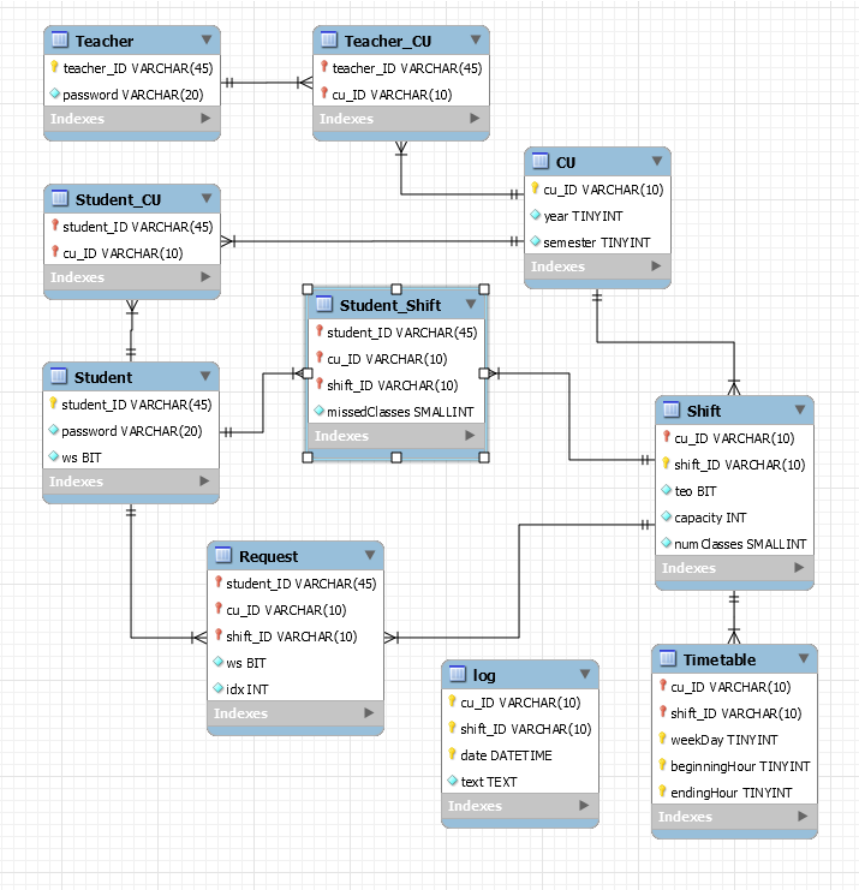

Figura 7: Modelo lógico da base de dados

12 Conclusão e análise crítica

Após a realização da primeira fase do projeto, onde elaboramos o Modelo de Domínio do Sistema de Gestão de Turnos, o seu diagrama de Use Case e respetivas especificações, realizamos os restantes diagramas com o objetivo de facilitar o desenvolvimento do software em si.

Relativamente aos diagramas, todos estes tiveram um papel importante. Tornaram a organização e robustez do código melhores e permitiram ter uma visão geral do projeto que não teríamos sem a realização dos mesmos. Consideramos que os diagramas foram bem realizados, no entanto entendemos que alguns estejam abertos a discussão uma vez que a ambiguidade foi algo com que nos deparámos na sua realização.

Quanto ao código, para o desenvolvimento deste usamos também o conhecimento adquirido sobre o conceito de aplicações Multi-Camada (tal como

nos diagramas) o que permitiu tornar a comunicação entre cada camada bastante simples e intuitiva, assim como a melhorar a organização do código. Deste modo, consideramos que a organização do código é bastante intuitiva, um aspeto bastante importante no desenvolvimento de software, quer pela apresentação gráfica do código, quer pela facilidade em alterar ou adicionar funcionalidades ao código.

Este trabalho preenche com sucesso e ultrapassa todos os objetivos mínimos da primeira e da segunda fase, sendo que nem todas as funcionalidades descritas no enunciado estão presentes no software (funcionalidade de os alunos poderem assistir a turno, por exemplo). No entanto, apresentamos funcionalidades como a consulta de um estado de um pedido de troca por parte do aluno, que não estão presentes no enunciado.

Embora tenhamos cumprido com sucesso os requisitos impostos deparamonos com algumas dificuldades no desenvolvimento deste projeto. O desenvolvimento de uma boa *user interface*, com o menor número de páginas envolvidas não resultou no produto esperado em que consistia em apenas uma página *responsive*. Tivemos bastantes dificuldades neste aspeto devido à pouca experiência no que toca ao uso da ferramenta *Swing* e também devido ao fator de tempo. Também devido ao fator de tempo não foi possível resolvermos algumas questões de robustez da aplicação, nomeadamente a melhoração do registo dos diferentes tipos de utilizadores, tratamento de capacidades de turnos que nunca poderiam suportar todos os alunos inscritos e carregamento de apenas a informação necessária à aplicação e não de toda a informação disponível.