Universidade do Minho
Escola de Engenharia

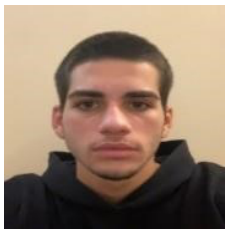
Desenvolvimento de Sistemas de Software

Trabalho Prático – Fase 1

Grupo 44

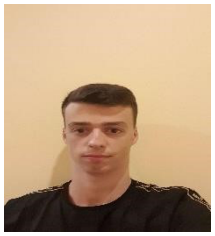
Link GitHub: https://github.com/orgs/LEI-DSS/teams/dss2425-grupo-44

João Pinto
(a104270)



Diogo Silva
(a104183)

Olavo Carreira
(a104526)



Miguel Barrocas
(a104272)

Índice

Introdução *3*

Modelo de Domínio *4*

Modelo de Use Case *5*

1. Criar Conta **6**

2. Login **7**

3. Consultar Horário **8**

4. Gerar Horário **9**

5. Trocar Turno **10**

6. Definir Preferências **11**

7. Alocação Manual **12**

8. Publicar Horário **13**

9. Exportar Horário **14**

10. Importar Lista de Alunos **15**

11. Importar Lista da UC e Horários **16**

Diagrama de Componentes *17*

Diagramas de Classes *18*

Diagrama de Classes SSUtilizadores **18**

Diagrama de Classes SSHorarios **18**

Diagramas de Sequência *19*

existeUtilizador **19**

registaUtilizador **19**

validaPassword **20**

consultaHorario **20**

alocaManual **21**

alocaTodosAlunos **21**

importaAlunos **22**

Implementação do Sistema *22*

Interface *24*

Estrutura da Interface **24**

1. Menu Principal 24
2. Menu Aluno 24
3. Menu Regente 25

Base de Dados do Sistema *27*

Tabelas do Programa **27**

DAO's implementados no Programa *31*

Conclusão e Análise dos resultados obtidos *32*

Introdução

Este relatório foi elaborado no âmbito da unidade curricular de Desenvolvimento de Sistemas de Software, com o objetivo de criar um sistema para auxiliar a gestão da criação de turnos num curso (p.e Licenciatura em Engenharia Informática).

Este projeto propõe a criação de um sistema centralizado que facilite a gestão dos turnos e permita a aplicação de diferentes algoritmos para priorizar alunos com base em critérios como o estatuto ou até regras específicas de cada UC. O sistema visa melhorar a organização e simplificar o processo.

Nesta primeira fase, é-nos solicitado que analisemos o problema de forma a identificar as entidades relevantes e fazer a descrição dos Use Cases do sistema de modo a levantar requisitos funcionais.

Desenvolvemos o Modelo de Domínio, que descreve as entidades do problema e os seus relacionamentos, proporcionando uma visão estática. O Modelo de Use Cases, também foi desenvolvido de forma a mostrar as funcionalidades do sistema e descrever os fluxos de interação.

Modelo de Domínio

Um Modelo de Domínio bem estruturado deve concentrar-se na representação do problema sem abordar soluções para a sua resolução. Com base nisto, o nosso grupo, criou um modelo (figura 1), que tem como objetivo representar de forma clara o problema proposto, abrangendo as entidades envolvidas, assim como os relacionamentos que as ligam. Desta forma, o modelo proporciona uma visão estruturada e detalhada do problema, permitindo uma compreensão completa do cenário antes de se avançar para a próxima fase.

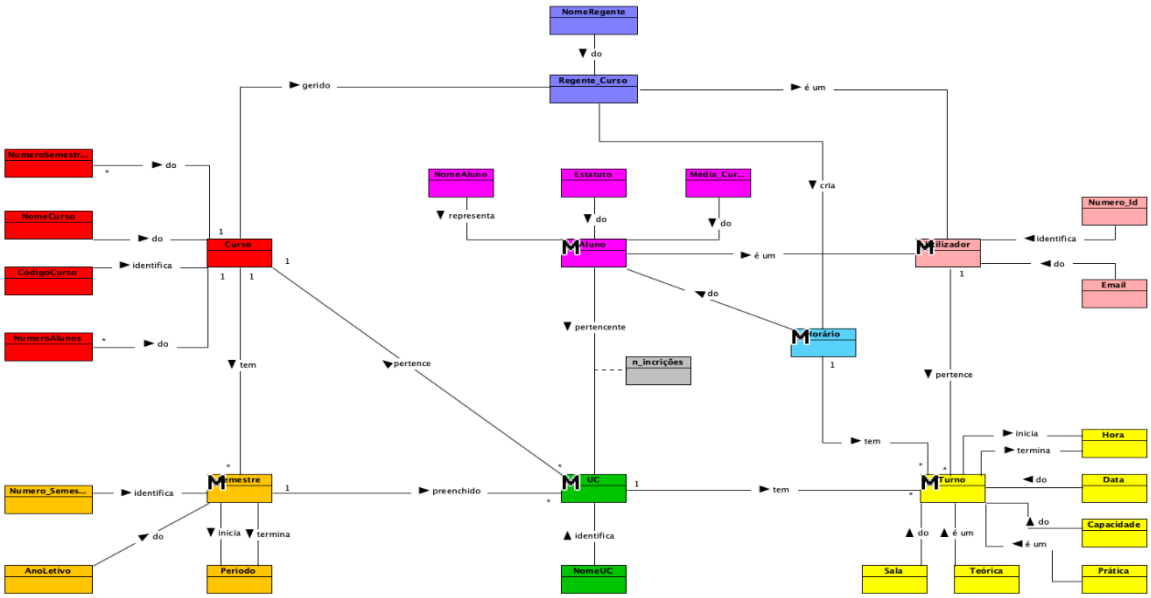


Figura 1 – Modelo de Domínio

Identificamos as seguintes entidades principais:

-> Semestre

-> Curso

-> U.C.

-> Aula

-> Utilizador

As cores no nosso modelo visam categorizar as diferentes entidades principais, de forma a tornar os seus relacionamentos mais fáceis de perceber.

-> Um curso é designado por um nome e identificado pelo seu código. Cada curso tem um número de semestres e um número de alunos.

-> Uma unidade curricular é identificada por um nome e está associada a um ou mais cursos. Cada UC tem aulas num determinado semestre.

-> Um semestre é definido por um número, um ano letivo e um período (que é o período em que um dado semestre se inicia e termina). Cada semestre abrange várias UCs.

-> Um utilizador é identificado por um número de ID e contactado por um email, podendo ser um aluno ou um regente do curso.

-> Uma aula é caracterizada por um hórario específico, um turno, o número de inscritos, a capacidade da sala e o tipo de aula (teórica ou prática). Cada aula está associada a uma UC e a um semestre e é atendida por diversos utilizadores.

Modelo de Use Case

O modelo de Use Case, ao contrário do modelo de domínio, tem como objetivo definir a solução do problema do sistema a desenvolver. Desenvolvemos então onze Use Cases distintos.

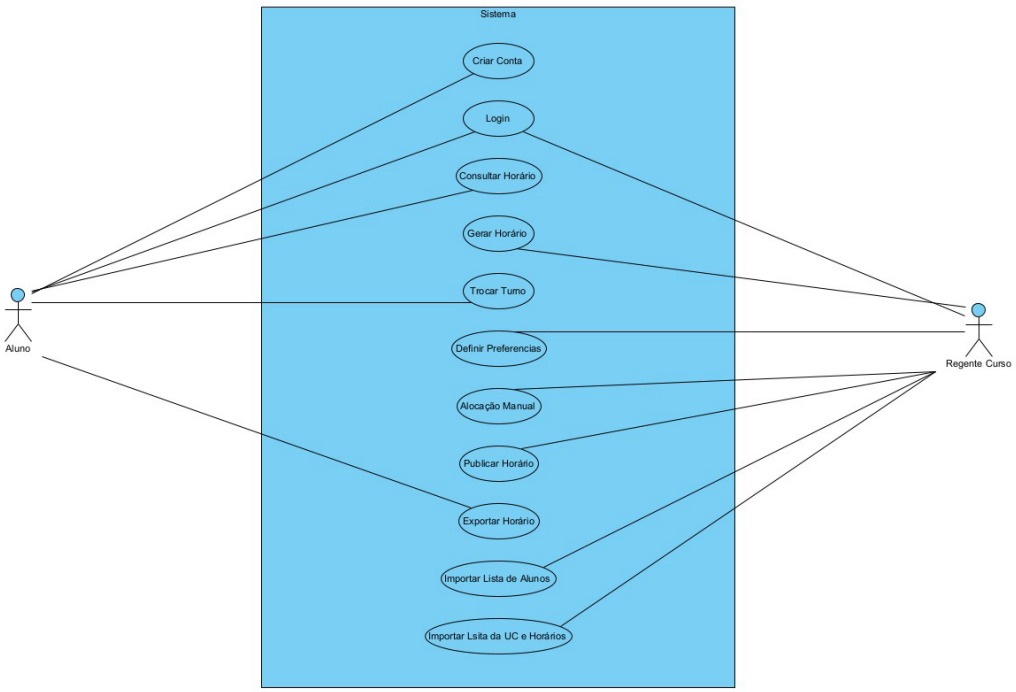


Figura 2 - Modelo de use case

1. Criar Conta

Use Case: Criar Conta
Descrição: Criação de uma conta de utilizador no sistema
Cenários: A Ana cria uma conta como aluna. O João cria uma conta como regente de curso.
Pré-condição: O utilizador não possui uma conta existente
Pós-condição: O utilizador consegue aceder ao sistema com a nova conta

Fluxo normal (1):

1. O sistema solicita o número de utilizador.
2. O utilizador fornece o número.
3. O sistema verifica se o utilizador já existe.
4. O sistema pede uma password.
5. O utilizador fornece a password.
6. O sistema regista o novo utilizador.
7. O sistema confirma a criação da conta e informa o utilizador.

Fluxo alternativo (2) [Número inválido] (passo 2):

2.1 O sistema informa o utilizador que o número que forneceu é inválido.
2.2 Regressa ao passo 1.

Fluxo de exceção (3) [Utilizador já existente] (passo 3):

3.1 O sistema indica que o utilizador já está registado.
3.2 O sistema não regista a nova conta.

2. Login

Use Case: Login
Descrição: Entrada de um utilizador no sistema
Cenários: O Ricardo entra como aluno. O Jorge entra como diretor de curso.
Pré-condição: O utilizador já está registado no sistema **Pós-condição**: O utilizador consegue acesso ao sistema

Fluxo normal (1):

1. O sistema solicita o número do utilizador e a password.
2. O utilizador fornece o número de utilizador e a password.
3. O sistema verifica que o utilizador é válido (existe).
4. O sistema verifica se a password está correta.
5. O sistema concede acesso ao utilizador.

Fluxo de exceção (2) [Utilizador inexistente] (passo 3):

3.1 O sistema indica que o utilizador não existe.
3.2 O sistema não garante acesso ao utilizador.

Fluxo de exceção (3) [Password errada] (passo 4):

4.1 O sistema indica que a password indicada não é válida.
4.2 O sistema não garante acesso ao utilizador.

3. Consultar Horário

Use Case: Consultar Horário
Descrição: O aluno consulta o seu horário
Cenários: O Ricardo consulta o seu horário. A Maria vê o seu horário de aulas.
Pré-condição: O aluno está autenticado.
Pós-condição: O aluno vê o seu horário completo na aplicação

Fluxo normal (1):

1. O sistema apresenta ao utilizador um menu com várias opções, incluindo "Consultar Horário".
2. O aluno seleciona "Consultar Horário".
3. O sistema verifica se o horário já foi gerado
4. O sistema apresenta o horário do aluno.

Fluxo de exceção (2) [Horário não gerado] (passo 3):

3.1 O sistema informa o aluno que o horário ainda não foi gerado.
3.2 O aluno não consegue consultar o horário até o mesmo ser publicado.

4. Gerar Horário

Use Case: Gerar Horário
Descrição: O sistema gera o horário dos alunos, respeitando as preferências das UC e as regras de alocação de turnos.
Cenários: O diretor de curso quer gerar o horário dos alunos inscritos nas UC, respeitando as preferências das UC e considerando alunos com estatuto especial.
Pré-condição: O sistema tem as listas de alunos e UC já importadas, bem como as preferências configuradas.
Pós-condição: O sistema gera o horário, respeitando as restrições e exceções.

Fluxo normal (1):

1. O sistema apresenta ao utilizador um menu com várias opções, incluindo "Gerar Horários".
2. O diretor de curso seleciona a opção"Gerar Horários".
3. O sistema recupera as listas de alunos e UC e as suas preferências.
4. O sistema considera as preferências das UC, como grupos de trabalho, limites de alunos por turno, separação de alunos repetentes, etc.
5. O sistema gera uma primeira alocação de alunos aos turnos, de acordo com as preferências.

Fluxo alternativo (2) [Alunos com Estatuto Especial] (passo 3):

3.1 O sistema identifica os alunos com estatuto especial.
3.2 O sistema permite que esses alunos possam escolher o turno em que querem estar ou sejam automaticamente alocados de acordo com a sua preferência.
3.3 O sistema ignora as restrições das UC para estes alunos, permitindo-lhes acesso a qualquer turno disponível.

Fluxo de exceção (3) [Turno cheio] (passo 4):

4.1 Caso o turno preferido por um aluno com estatuto especial esteja cheio, o sistema oferece ao aluno uma lista de turnos alternativos com vaga.
4.2 O aluno seleciona um novo turno da lista oferecida pelo sistema.

5. Trocar Turno

Use Case: Trocar Turno
Descrição: O aluno solicita a troca de turno
Cenários: A Mariana quer mudar do turno 1 para o turno 2.
Pré-condição: O aluno tem um horário atribuído, está autenticado no sistema e tem turnos com vagas
Pós-condição: O turno é alterado para o aluno, se houver espaço disponível

Fluxo normal (1):

1. O sistema apresenta ao utilizador um menu com várias opções, incluindo "Trocar Horário".
2. O utilizador escolhe a opção de “Trocar Horário”
3. O sistema apresenta todas as UC ao qual o utilizador está inscrito
4. O aluno escolhe a UC à qual deseja trocar de turno
5. O sistema apresenta todas os turnos
6. O aluno escolhe o turno para o qual deseja mudar.
7. O sistema verifica se há vagas no turno desejado.
8. O sistema confirma a troca de turno e atualiza o horário do aluno.

Fluxo alternativo (2) [Turno Cheio] (passo 3):

3.1 O sistema informa o aluno que o turno desejado está cheio.
3.2 O sistema solicita que o aluno escolha outro turno.
3.3 Regressa ao passo 7

6. Definir Preferências

Use Case: Definir Preferências
Descrição: O regente define as preferências de alocação de alunos
Cenários: O professor Manuel configura os alunos repetentes e grupos de trabalho.
Pré-condição: O regente já importou a lista de UC e turnos e está autenticado
Pós-condição: As preferências são guardadas no sistema para futura geração de horários **Fluxo normal (1)**:

1. O sistema apresenta ao utilizador um menu com várias opções, incluindo "Definir Preferências".
2. O regente acede à página de "Definir Preferências".
3. O regente seleciona a UC e os parâmetros de alocação como grupos de trabalho, limites de alunos por turno, separação de alunos repetentes, etc.
4. O sistema guarda as preferências definidas pelo regente.
5. O sistema confirma a operação ao regente.

Fluxo de exceção (2) [Preferências Contraditórias] (passo 3):

3.1 O sistema deteta que as preferências são contraditórias (ex.: aluno está em dois grupos).
3.2 O sistema informa o regente e solicita que resolva os conflitos.

7. Alocação Manual

Use Case: Alocação Manual
Descrição: O regente aloca manualmente alunos que não foram alocados automaticamente **Cenários**: O professor António aloca alunos sem turno atribuído.
Pré-condição: A alocação automática deixou alunos sem turnos atribuídos **Pós-condição**: Os alunos são manualmente alocados aos turnos

Fluxo normal (1):

1. O regente visualiza a lista de alunos sem turno.
2. O regente escolhe os turnos disponíveis e aloca os alunos manualmente.
3. O sistema verifica se a alocação é viável (sem conflitos).
4. O sistema confirma a alocação ao regente.

Fluxo de exceção (2) [Conflito de Horário] (passo 3):

3.1 O sistema deteta que a alocação causa conflitos de horário.
3.2 O sistema informa o regente e solicita que resolva os conflitos antes de continuar.

8. Publicar Horário

Use Case: Publicar Horário
Descrição: O regente publica os horários finais dos alunos **Cenários**: A professora Rita publica os horários para a turma.
Pré-condição: A alocação de todos os alunos foi concluída **Pós-condição**: Os alunos têm acesso ao seu horário final

Fluxo normal (1):

1. O sistema apresenta ao utilizador um menu com várias opções, incluindo "Publicar Horário".
2. O regente seleciona "Publicar Horário".
3. O sistema verifica que todos os alunos têm turnos atribuídos.
4. O regente confirma a publicação.
5. O sistema envia notificações aos alunos e disponibiliza os horários.

Fluxo de exceção (2) [Alunos sem Turno] (passo 2):

2.1 O sistema deteta que há alunos sem turno atribuído.
2.2 O sistema impede a publicação e solicita ao regente que complete a alocação

9. Exportar Horário

Use Case: Exportar Horário
Descrição: O aluno exporta o seu horário para a sua agenda pessoal.
Pré-condição: O aluno tem um horário já publicado e encontra-se autenticado no sistema. **Pós-condição**: O horário é exportado com sucesso para a agenda do aluno.

Fluxo Principal (1):

1. O sistema apresenta ao utilizador um menu com várias opções, incluindo "Exportar Horário".
2. O aluno seleciona a opção "Exportar Horário".
3. O sistema verifica que o aluno já tem o horário
4. O sistema gera o ficheiro do horário e disponibiliza-o para download.
5. O aluno exporta o ficheiro para a agenda

Fluxo de Exceção (2) [Sem horário] (passo 3) :

3.1 O sistema verifica que o aluno ainda não tem horário atribuido

3.2 O aluno não consegue exportar o horário

10. Importar Lista de Alunos

Use Case: Importar lista de alunos
Descrição: O regente de curso importa uma lista de alunos inscritos nas UC para o sistema de gestão de turnos.
Pré-condição: O regente tem o ficheiro com a lista de alunos e está autenticado no sistema

Pós-condição: A lista de alunos é importada com sucesso e os dados ficam disponíveis para utilização.

Fluxo Principal (1) :

1. O sistema apresenta ao utilizador um menu com várias opções, incluindo "Importar Lista de Alunos".
2. O regente de curso acede à opção "Importar Lista de Alunos" no sistema.
3. O sistema solicita que o regente selecione o ficheiro com os dados dos alunos.
4. O regente escolhe o ficheiro correto e carrega-o no sistema.
5. O sistema valida os dados do ficheiro, verificando se tem todas as informações corretas (nomes, números de estudante, etc.).
6. O sistema importa os dados dos alunos e disponibiliza-os para a gestão de turnos.
7. O sistema apresenta uma mensagem de confirmação, informando que a lista foi importada com sucesso.

Fluxo de Exceção (2) [Erro no ficheiro] (passo 5):

5.1 Se o ficheiro tiver erros de formato ou dados incompletos (por exemplo, campos em falta ou formatados incorretamente), o sistema interrompe o processo.

5.2 O sistema informa o regente sobre o erro encontrado e identifica quais campos estão incorretos.

11. Importar Lista da UC e Horários

Use Case: Importar lista da UC e Horários
Descrição: O regente de curso importa uma lista de UC e Horários
Pré-condição: O regente tem o ficheiro com a lista da UC e Horários e está autenticado no sistema

Pós-condição: A lista de UC e horários é importada com sucesso e pode ser usada para a alocação de alunos aos turnos.

Fluxo Principal (1):

1. O sistema apresenta ao utilizador um menu com várias opções, incluindo "Importar Lista da UC e Horários".
2. O regente de curso acede à opção "Importar Lista da UC e Horários " no sistema.
3. O sistema solicita que o regente selecione o ficheiro com os dados da UC e Horários.
4. O regente escolhe o ficheiro correto e carrega-o no sistema.
5. O sistema valida os dados do ficheiro, verificando se tem todas as informações corretas (nomes da UC, horários, turnos, etc.).
6. O sistema importa os dados da UC e Horários e disponibiliza-os para a gestão de turnos.
7. O sistema apresenta uma mensagem de confirmação, informando que a lista foi importada com sucesso.

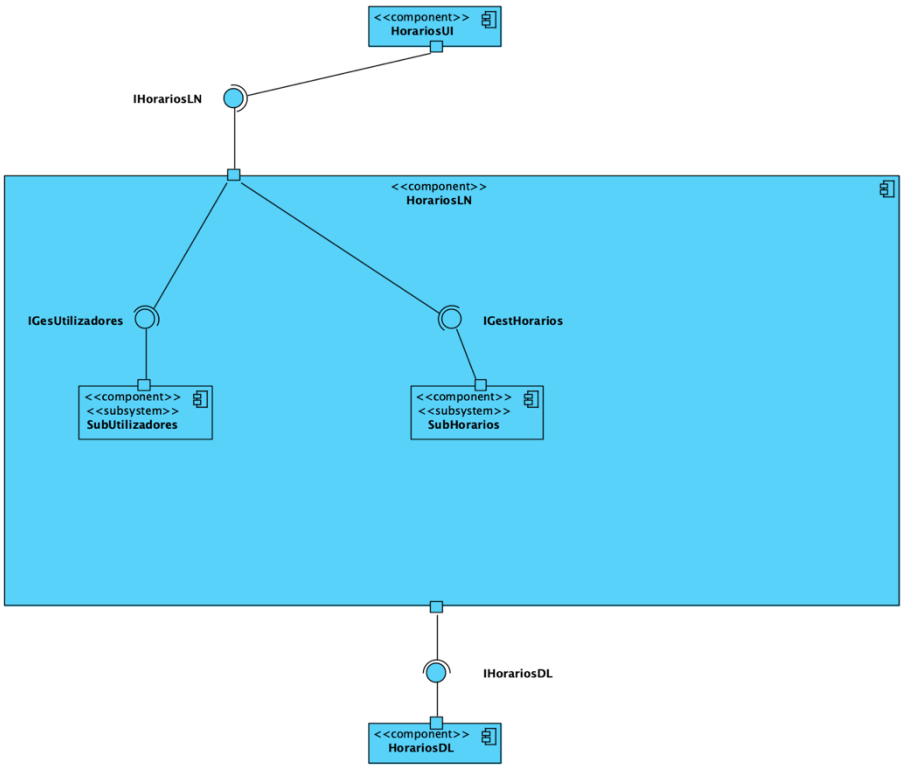
Fluxo de Exceção (2) [Erro no ficheiro] (passo 5):

5.1 Se o ficheiro tiver erros de formato ou dados incompletos (por exemplo, campos em falta ou formatados incorretamente), o sistema interrompe o processo.

5.2 O sistema informa o regente sobre o erro encontrado e identifica quais campos estão incorretos.

Diagrama de Componentes

Os diagramas de componentes têm como objetivo visualizar a organização dos componentes do sistema e os relacionamentos de dependência entre eles. Neste caso, usamos o diagrama de componentes com o principal objetivo de dividir a lógica de negócio do nosso projeto, dividindo as responsabilidades do sistema em dois subsistemas distintos.

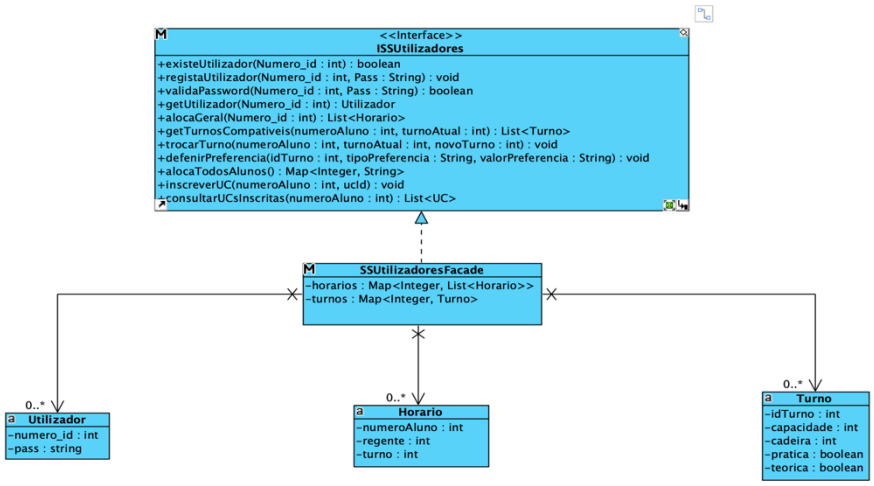


Nota: Decidimos não fazer um diagrama de packages pois achamos que não acrescentaria informação ao nosso trabalho, tendo isto em conta simplesmente assumimos que cada subsistema, representado no diagrama de componentes, corresponde a um package.

Diagramas de Classes

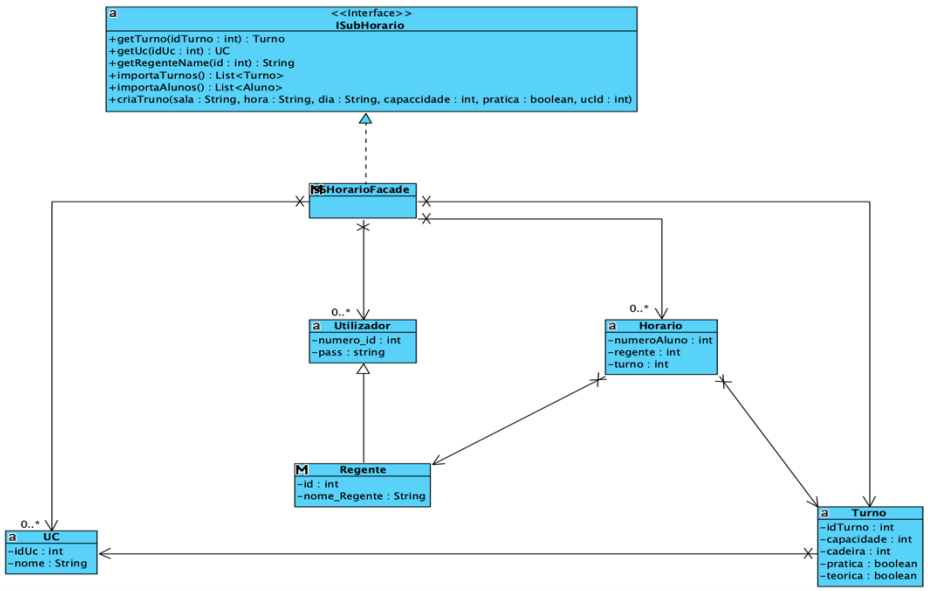
O diagrama de classes tem como objetivo descrever as classes de um sistema e os relacionamentos existentes entre elas. Permite, também, ilustrar a estrutura do sistema, através das operações e atributos das classes. Desta forma, concebemos os seguintes diagramas de classes para cada subsistema do programa:

Diagrama de Classes SSUtilizadores



Este diagrama de classes corresponde ao subsistema de utilizadores e servirá como gestor de todos os utilizadores do sistema (Alunos e Regentes). O subsistema possui um Facade que contém um Map de todos os utilizadores do programa.

Diagrama de Classes SSHorarios

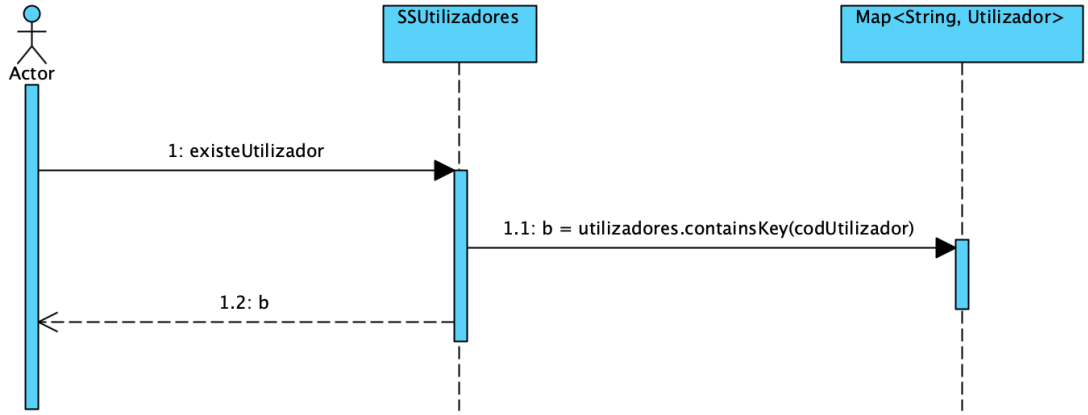


Este diagrama de classes corresponde ao subsistema de horarios e servirá como gestor de todos os horarios do sistema. O subsistema possui um Facade que contém um Map de todos os turnos, uc's e utilizadores que existem.

Diagramas de Sequência

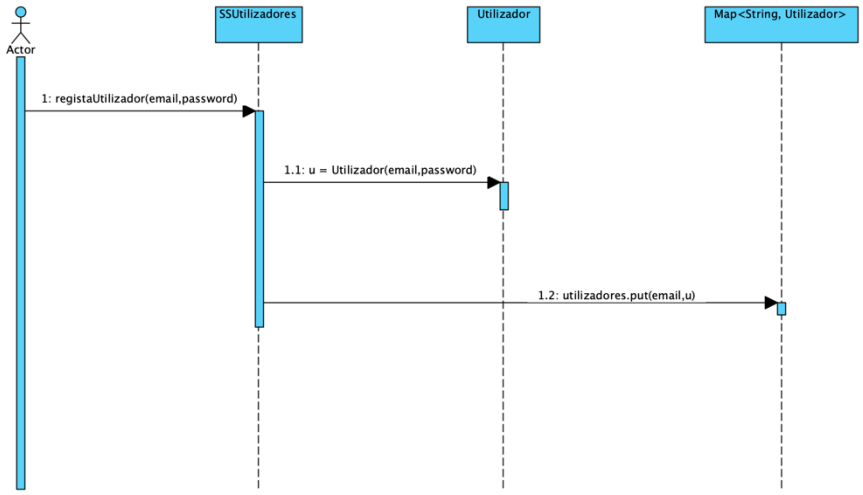
Os diagramas de sequência têm como objetivo focar no ordenamento temporal da troca de mensagens, permitindo visualizar como é que os objetos comunicam entre si.

existeUtilizador



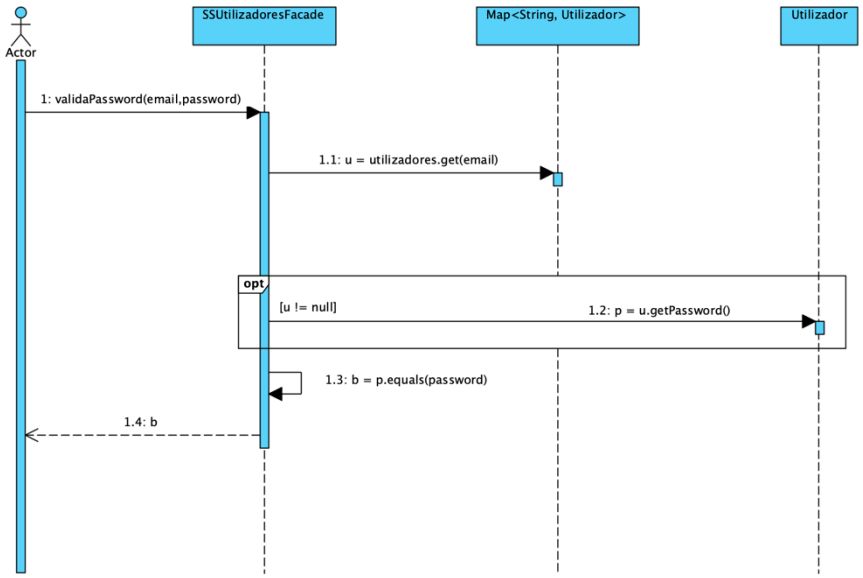
Este diagrama corresponde ao método existeUtilizador que verifica a existência de um utilizador com o username recebido como argumento. Retorna true caso o utilizador exista e false quando não existe.

registaUtilizador



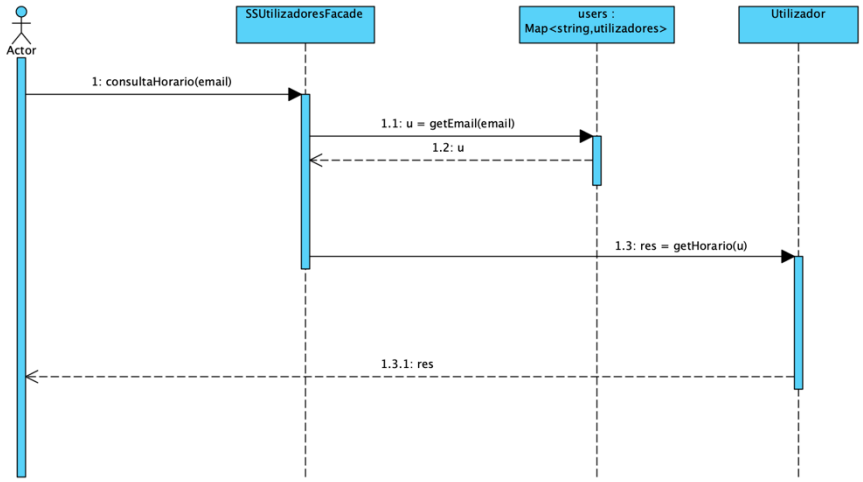
Este diagrama corresponde ao método registaUtilizador que recebe o username e password de um novo utilizador e armazena o no sistema.

validaPassword



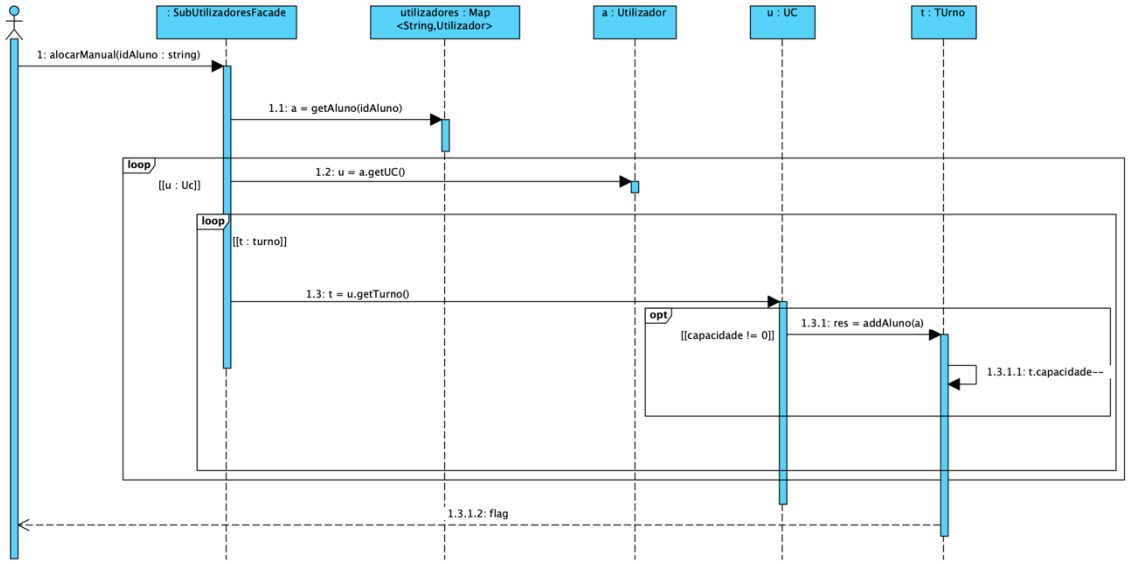
Este diagrama corresponde ao método validaPassword que recebe o username e password de um utilizador e verifica se existe um utilizador com esses dados. Se existir o método devolve true autenticando o utilizador com essa conta senão devolve false e o utilizador não fica autenticado.

consultaHorario



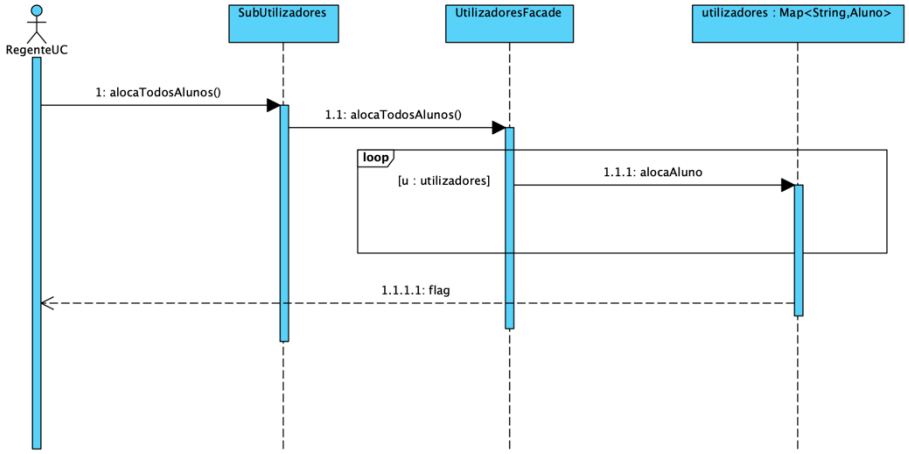
Este diagrama corresponde ao método consultaHorario que recebe o id de um utilizador e devolve um objeto horário desse mesmo utilizador. Assim um aluno consegue consultar o seu horário na paltaforma.

alocaManual



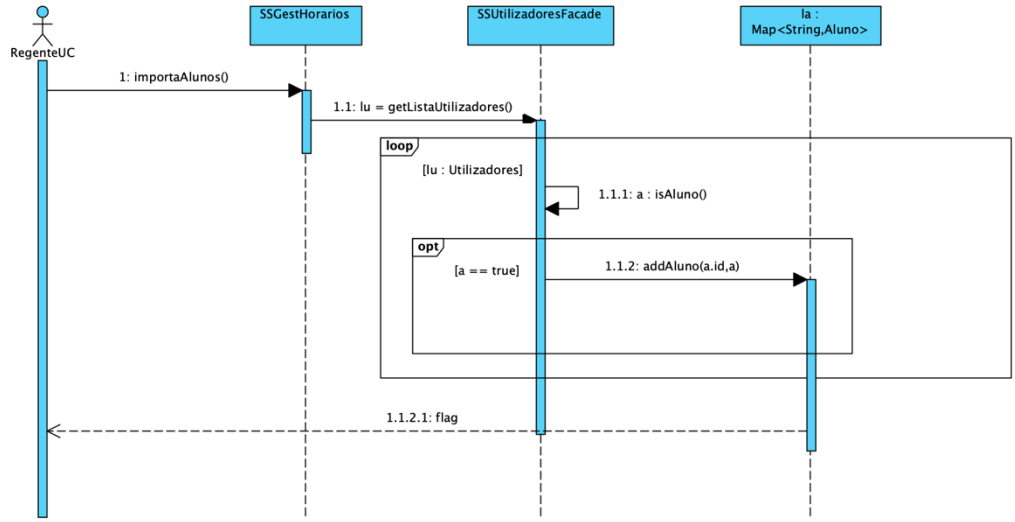
Este diagrama corresponde ao método alocaManual, que consiste em o regente alocar manualmente um aluno a um respetivo turno. Assim ele recebe como argumento o id do aluno e o id do turno, devolve true se o for possível alocar o aluno aquele turno e a capacidade do turno e diminuída, caso não seja possível devolve false.

alocaTodosAlunos



Este diagrama corresponde ao método alocaTodosAlunos, consiste em alocar todos os alunos inscritos no curso nas respetivas UC's de forma automática. Assim ele não precisa de argumentos, vai apenas buscar todos os alunos e todos os turnos que existem e depois percorre os mesmos utilizando uma função auxiliar alocaAuno para ir alocando cada aluno nos seus turnos.

importaAlunos



Este diagrama corresponde ao método importaAlunos que recebe um Map com todos os alunos da base de dados e os coloca numa lista para se usado no resto do sistema

Implementação do Sistema

A implementação do sistema de gestão de horários desenvolveu-se em torno de dois componentes fundamentais: SubHorarios e SubUtilizadores, cada um com responsabilidades específicas e complementares. A SSHorariosFacade funciona como ponto central para operações relacionadas à gestão de horários e turnos.

Seus métodos principais incluem importação de turnos e alunos, criação de novos turnos, consulta de unidades curriculares e identificação de regentes. Esta fachada permite uma abstração eficiente das operações básicas de horário, facilitando a integração entre diferentes componentes do sistema.

Em paralelo, a SSUtilizadoresFacade gerencia todas as interações relacionadas aos usuários, sendo responsável por um conjunto mais amplo de funcionalidades. Suas principais responsabilidades abrangem:

1. Gestão de Utilizadores

- Registo de novos utilizadores (alunos)
- Validação de credenciais
- Verificação de existência de utilizadores
- Autenticação no sistema

2. Operações de Horário

- Consulta de horários individuais
- Troca de turnos
- Identificação de turnos compatíveis
- Alocação manual e automática de alunos

3. Gerenciamento de Preferências

- Definição de preferências de turno
- Verificação de elegibilidade para turnos específicos
- Consideração de estatuto especial e média do curso.

A implementação das classes Aluno e Regente permite uma diferenciação clara de permissões e funcionalidades. A classe Aluno, por exemplo, possui métodos sofisticados como verificaPreferencias() que analisa criteriosamente a elegibilidade do aluno para um determinado turno, considerando:

- Estatuto do aluno
- Média do curso
- Restrições específicas do turno

O método trocarTurno() implementa uma lógica complexa que garante:

- Compatibilidade do novo turno
- Manutenção das restrições de preferência
- Atualização dinâmica da capacidade dos turnos
- Prevenção de conflitos de horário

A alocação de alunos pode ser realizada de duas formas:

- Alocação automática global (alocaTodosAlunos())
- Alocação manual para alunos específicos (alocaManual())

Ambos os métodos implementam verificações rigorosas para manter a integridade do sistema, considerando:

- Disponibilidade de vagas
- Preferências definidas- Restrições de horário
- Estatuto do aluno

A camada de persistência, implementada através de DAOs (Data Access Objects), permite um armazenamento eficiente e seguro dos dados, com classes específicas para:

- AlunoDAO
- RegenteDAO
- HorarioDAO
- TurnoDAO
- UCDAO
- UtilizadoresDAO
- InscricaoUCDAO
- TurnoPreferenciaDAO

Cada DAO implementa operações CRUD (Create, Read, Update, Delete) para sua respetiva entidade, garantindo a consistência dos dados entre a aplicação e o banco de dados. A arquitetura em camadas, com fachadas e DAOs, proporciona:

- Alta modularidade
- Flexibilidade para futuras expansões
- Separação clara de responsabilidades
- Facilidade de manutenção

Pontos críticos da implementação incluem:

- Tratamento de conflitos de horário
- Validação de preferências de turno
- Gestão dinâmica de capacidade
- Suporte a diferentes tipos de utilizadores

O sistema foi projetado para ser robusto, considerando múltiplos cenários e garantindo que as regras de negócio sejam aplicadas consistentemente em todas as operações.

Interface

A interface do sistema de Gestão de Horários foi desenvolvida com o objetivo de proporcionar uma experiência intuitiva e funcional para diferentes tipos de utilizadores, nomeadamente alunos e regentes.

Estrutura da Interface

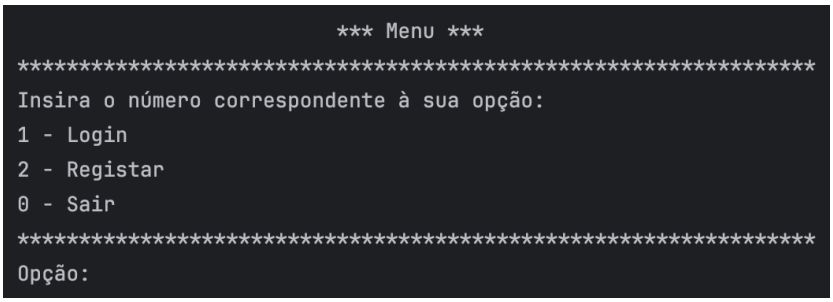
A interface foi implementada na classe GestHorariosUI, que gerencia três menus principais:

1. Menu Principal

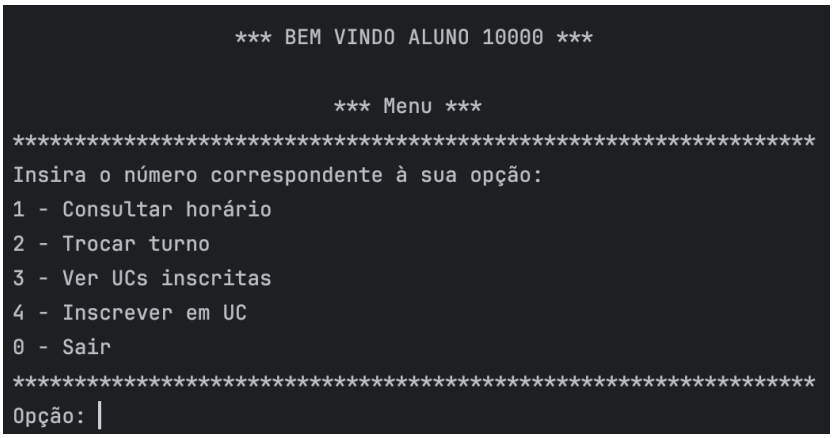
- Login

- Registo de Utilizadores

- Opção para sair do sistema



2. Menu Aluno



O menu do aluno oferece as seguintes opções:

- Consultar horário

- Apresenta o horário atual do aluno

- Mostra detalhes como unidade curricular, dia, hora, sala e tipo de aula (prática ou teórica)

- Trocar turno

- Permite ao aluno alterar o seu turno atual
- Mostra lista de unidades curriculares matriculadas
- Apresenta turnos disponíveis
- Verifica disponibilidade e compatibilidade do turno
- Atualiza o horário do aluno se a mudança for possível

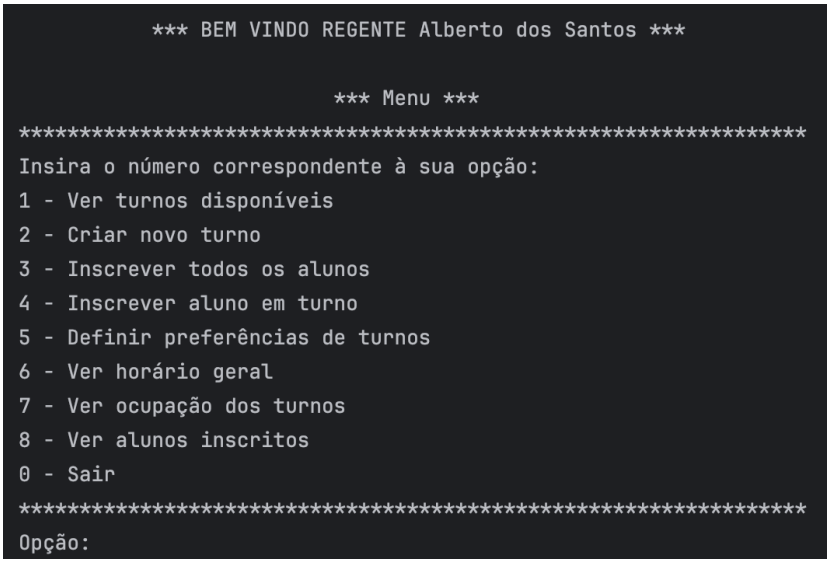
- Ver UCs inscritas

- Lista todas as unidades curriculares em que o aluno está matriculado
- Apresenta ID da disciplina, nome, ano e semestre

- Inscrever em UC

- Mostra unidades curriculares disponíveis
- Permite ao aluno matricular-se em novas disciplinas
- Verifica restrições de matrícula (por exemplo, alunos com estatuto especial limitados a 4 disciplinas)

3. Menu Regente



O menu do regente oferece opções mais avançadas de gestão:

- Ver turnos disponíveis

- Lista todos os turnos existentes
- Mostra detalhes como sala, hora, capacidade, unidade curricular e tipo de turno

- Criar turno

- Permite criar turnos, requer inserção de:
  - Unidade curricular
  - Sala
  - Dia
  - Hora
  - Capacidade
  - Tipo de turno (prático ou teórico)

- Inscrever todos os alunos

- Aloca automaticamente alunos para turnos
- Considera preferências de curso
- Trata alunos com estatuto especial

- Inscrever aluno em turno

- Permite alocação manual de um aluno específico em um turno
- Verifica conflitos e disponibilidade do turno

- Definir preferências de turnos

- Define regras de alocação para turnos
- Define preferências como:
  - Limites de Média dos alunos
  - Separação de alunos repetentes

- Ver horário geral

- Apresenta uma visão abrangente de todos os turnos em todas as unidades curriculares

- Ver ocupação dos turnos

- Mostra vagas disponíveis em cada turno

- Ver alunos inscritos

- Lista todos os alunos no curso
- Mostra detalhes do aluno como ID, nome, média e estatuto

A interface é projetada para ser intuitiva, com navegação clara entre menus e tratamento robusto de erros. Separa funcionalidades com base no tipo de utilizador (aluno ou diretor de curso) e fornece ferramentas abrangentes para gestão de horários.

Base de Dados do Sistema

De forma a conseguirmos armazenas todos os dados relevantes e necessários ao funcionamento do nosso sistema, decidimos implementar uma base de dados, que fosse capaz de guardar toda a informação a fim de sermos, assim, capazes de garantir a persistência de dados do nosso sistema.

Como sistema de gestão de dados, elegemos o MySQL.

Tabelas do Programa

- **Utilizadores**

Numero_id	Pass	Email	isRegente
1	1234	albertoRegente@gmail.com	1
10000	1234	joaopinto@gmail.com	0
11000	1234	diogosilva@gmail.com	0
11100	1234	olavocarreira@gmail.com	0
11110	1234	miguelbarrocas@gmail.com	0
11111	1234	beatriz@gmail.com	0
12000	1234	carlos@gmail.com	0
12100	1234	diana@gmail.com	0
12110	1234	eduardo@gmail.com	0
12111	1234	fernanda@gmail.com	0
13000	1234	gabriel@gmail.com	0
13100	1234	helena@gmail.com	0
13110	1234	igor@gmail.com	0
13111	1234	julia@gmail.com	0
NULL	NULL	NULL	NULL

Contem como chave primáriao Numero_id. Contem os seguintes atributos, Pass (password do utilizador para entrar no site), Email (email do utilizador), isRegente (pode ser 1 ou 0, se 1 é porque é um Regente, se 0 será um aluno). Permite-nos armazenar os dados de todos os utilizadores

- **Regente**

id_Regente	NomeRegente
1	Alberto dos Santos
NULL	NULL

Contem como chave primária o id_Regente. Contem os seguintes atributos, NomeRgente (nome do utilizador regente). Permite-nos armazenar os dados de todos os regentes registados no programa.

- **Alunos**

id	Nome	Estatuto	Ano_Incricao	Media_Curso
10000	Joao Pinto		3	10
11000	Diogo Silva		3	13
11100	Olavo Carreira		3	15
11110	Miguel Barrocas	Especial	3	11
11111	Beatriz Costa		3	12
12000	Carlos Antunes		3	14
12100	Diana Mendes	Especial	3	11
12110	Eduardo Ribeiro		3	13
12111	Fernanda Lima		3	12
13000	Gabriel Nunes		3	14
13100	Helena Rocha	Especial	3	15
13110	Igor Santos		3	13
13111	Julia Martins		3	10
NULL	NULL	NULL	NULL	NULL

Contem como chave primária o id. Contém os seguintes atributos, Nome (nome do aluno), Estatuto (pode ser estatuto especial, ou senão for fica vazio), Ano_Inscricao (ano da licenciatura que o aluno esta inscrito) e Media_Curso (media do aluno no curso). Permite-nos armazenar os dados de todos os alunos registados no programa.

- **Semestre**

Numero_Semestre	Ano_Letivo	Periodo
1	1	07/07/2024 a 12/01/2025
2	1	13/01/2025 a 30/06/2025
3	2	07/07/2025 a 12/01/2026
4	2	13/01/2026 a 30/06/2026
5	3	07/07/2026 a 12/01/2027
6	3	13/01/2027 a 30/06/2027
NULL	NULL	NULL

Contem como chave primária o Numero_Semestre. Contém os seguintes atributos, Ano_letivo (ano letivo do semestre), Periodo (data de inicio a data de fim do semestre). Permite-nos armazenar os dados de todos os semestres registados no programa.

- **Curso**

CodigoCurso	NomeCurso	NumeroSemestre	NumeroAlunos	Regente
1500	LEI	2	170	1
NULL	NULL	NULL	NULL	NULL

Contem como chave primária o CodigoCurso. Contém os seguintes atributos, NomeCurso (corresponde ao nome do curso em questao), NumeroSemestre (número de semestres do curso), NumeroAlunos (numero de vagas do curso), Regente (id do regente do curso). Permite-nos armazenar os dados de todos os cursos registados no programa.

- **UC**

idUC	NomeUC	Ano	Semestre	Curso
1	Calculo de Programas	3	5	1500
2	Comunicações por Computador	3	5	1500
3	DSS	3	5	1500
4	Inteligencia Artifical	3	5	1500
5	Laboratorios Informatica IV	3	5	1500
6	Sistemas Distribuidos	3	5	1500
NULL	NULL	NULL	NULL	NULL

Contem como chave primária o idUC. Contém os seguintes atributos, NumeUC (corresponde ao nome do UC em questao), Ano (ano do curso em que a UC é dada), Sesmtre (numero do semestre em que a UC esta), Curso (código do curso da UC). Permite-nos armazenar os dados de todas as UC's registadas no programa.

- **Turno**

idTurno	Sala	Hora	DataT	Capacidade	Cadeira	Pratica	Teorica
1	2.18	9/11	Segunda	19	4	1	0
2	2.09	9/11	Segunda	19	3	1	0
3	2.09	11/13	Segunda	25	4	1	0
4	1.27	11/13	Segunda	25	4	1	0
5	2.11	11/13	Segunda	19	3	1	0
6	0.11	11/13	Segunda	25	3	1	0
7	0.14	14/16	Segunda	25	4	1	0
8	1.03	14/16	Segunda	25	3	1	0
9	0.08	16/18	Segunda	54	4	0	1
10	2.07	18/20	Segunda	25	4	1	0
11	1.22	9/11	Terça	19	6	1	0
12	2.18	9/11	Terça	25	2	1	0
13	2.11	11/13	Terça	25	4	1	0
14	1.04	11/13	Terça	25	6	1	0
15	2.03	11/13	Terça	25	3	1	0
16	1.21	11/13	Terça	19	2	1	0
17	1.13	11/13	Terça	25	2	1	0
18	0.11	14/15	Terça	19	5	1	0
19	0.11	15/16	Terça	25	5	1	0
20	0.08	16/18	Terça	54	6	0	1
21	1.22	18/20	Terça	19	1	1	0
22	1.27	9/11	Quarta	25	1	1	0
23	2.12	11/13	Quarta	25	2	1	0
24	2.07	11/13	Quarta	25	1	1	0
25	2.10	11/13	Quarta	25	1	1	0
26	2.02	14/15	Quarta	25	5	1	0
27	2.02	15/16	Quarta	25	5	1	0
28	0.08	16/18	Quarta	54	3	0	1
29	0.08	18/20	Quarta	54	5	0	1
30	0.17	09/11	Quinta	25	6	1	0
31	2.01	09/11	Quinta	25	6	1	0
32	2.11	09/11	Quinta	25	2	1	0
33	2.01	11/13	Quinta	25	6	1	0
34	2.11	11/13	Quinta	25	2	1	0
35	0.07	16/18	Quinta	54	2	0	1
36	0.20	16/18	Quinta	55	1	0	1
37	0.07	09/11	Sexta	60	2	0	1
38	0.01	09/11	Sexta	54	1	0	1
39	1.12	11/13	Sexta	25	6	1	0
40	2.14	11/13	Sexta	25	6	1	0
41	2.01	11/13	Sexta	25	2	1	0
42	2.06	14/16	Sexta	25	3	1	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Contem como chace primaria o idTurno. Contem os seguintes atributos, Sala (sala da universidade onde decorrerá a aula), Hora (horas em que a aula vai decorrer inicial/final), DataT (dia da semana), Capacidade (capacidade do turno, quantos alunos pode ter), Cadeira (corresponde ao id da UC para saber de que qual UC é o turno), Pratica/Teorica (o atributo que tiver o 1 correspondente a true, significa que o turno é daquele tipo, ou sej, 1 é true e 0 é false). Permite-nos assim armazenar todos os dados sobre as UC registados no programa.

- **Inscricao_UC**

Aluno	UC
10000	1
11000	1
11100	1
11110	1
11111	1
12000	1
12110	1
12111	1
13000	1
13100	1
13110	1
13111	1
10000	2
11000	2
11100	2
11110	2
11111	2
12000	2
12100	2
12110	2
12111	2
13000	2
13110	2
13111	2
10000	3
11000	3
11100	3
11110	3
11111	3
12000	3
12110	3
12111	3
13000	3
13100	3
13110	3
13111	3
10000	4
11000	4
11100	4
11111	4
12000	4
12100	4
12110	4
12111	4
13000	4
13110	4
13111	4
10000	5
11000	5
11100	5
11110	5
11111	5
12000	5
12110	5
12111	5
13000	5
13100	5
13110	5
13111	5
10000	6
11000	6
11100	6
11110	6
11111	6
12000	6
12100	6
12110	6
12111	6
13000	6
13110	6
13111	6
NULL	NULL

Contem como chave primaria (Aluno, UC), ou seja, uma chave primaria composta, de forma a termos o mesmo aluno inscrito em diferentes cadeiras. Assim aqui não temos outros atributos, permite-nos assim guardar todos os dados sobre as inscrições dos alunos nas UC do curso registadas no programa.

- **Horario**

Aluno	Regente	Turno
10000	1	1
10000	1	5
10000	1	9
10000	1	11
10000	1	16
10000	1	18
10000	1	20
10000	1	21
10000	1	28
10000	1	29
10000	1	35
10000	1	38
11000	1	1
11000	1	5
11000	1	9
11000	1	11
11000	1	16
11000	1	18
11000	1	20
11000	1	21
11000	1	28
11000	1	29
11000	1	35
11000	1	38
11100	1	1
11100	1	5
11100	1	9
11100	1	11
11100	1	16
11100	1	18
11100	1	20
11100	1	21
11100	1	28
11100	1	29
11100	1	35
11100	1	38
11110	1	2
11110	1	11
11110	1	16
11110	1	18
11110	1	20
11110	1	21
11110	1	28
11110	1	29
11110	1	35
11110	1	38
11111	1	1
11111	1	5
11111	1	9
11111	1	11
11111	1	16
11111	1	18
11111	1	20
11111	1	21
11111	1	28
11111	1	29
11111	1	35
11111	1	38
12000	1	1
12000	1	5
12000	1	9
12000	1	11
12000	1	16
12000	1	18
12000	1	20
12000	1	21
12000	1	28
12000	1	29
12000	1	35
12000	1	38
12100	1	1
12100	1	9
12100	1	11
12100	1	16
12100	1	20
12100	1	35
12110	1	1
12110	1	5
12110	1	9
12110	1	11
12110	1	16
12110	1	18
12110	1	20
12110	1	21
12110	1	28
12110	1	29
12110	1	35
12110	1	38
12111	1	1
12111	1	5
12111	1	9
12111	1	11
12111	1	16
12111	1	18
12111	1	20
13000	1	1
13000	1	5
13000	1	9
13000	1	11
13000	1	16
13000	1	18
13000	1	20
13000	1	21
13000	1	28
13000	1	29
13000	1	35
13000	1	38
13100	1	2
13100	1	18
13100	1	21
13100	1	28
13100	1	29
13100	1	36
13110	1	1
13110	1	5
13110	1	9
13110	1	11
13110	1	16
13110	1	18
13110	1	20
13110	1	21
13110	1	28
13110	1	29
13110	1	35
13110	1	38
13111	1	1
13111	1	5
13111	1	9
13111	1	11
13111	1	16
13111	1	18
13111	1	20
13111	1	21
13111	1	28
13111	1	29
13111	1	35
13111	1	38
NULL	NULL	NULL

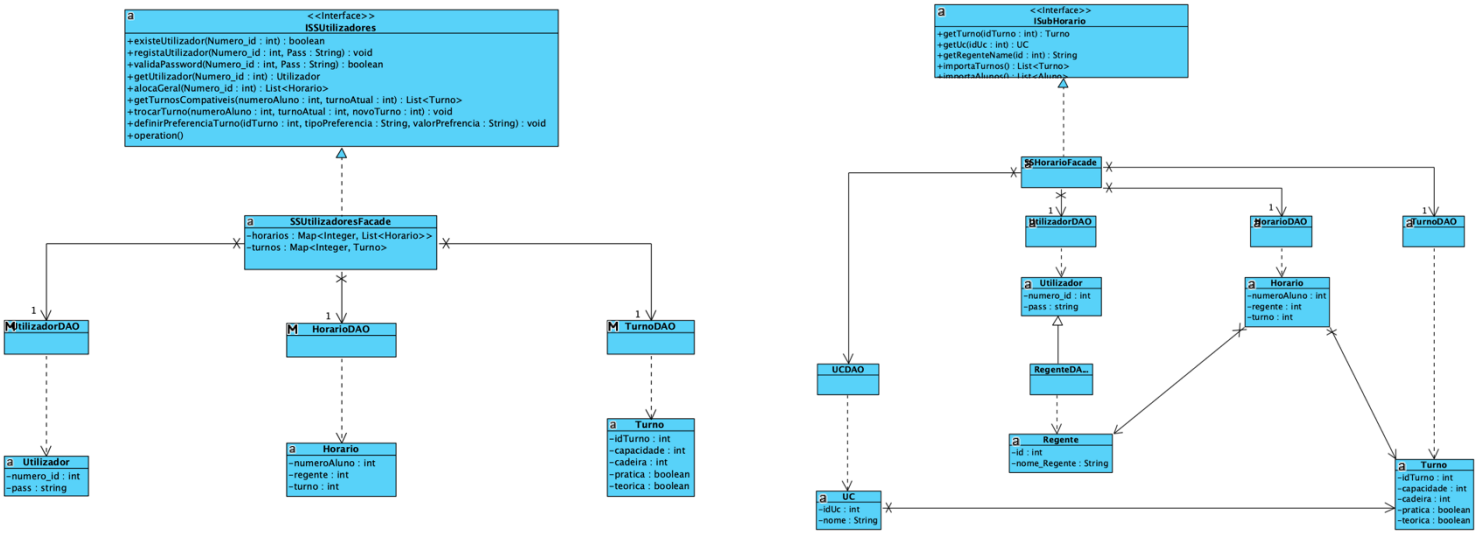
Contem como chave primaria (Aluno, Turno), ou seja, uma chave primaria composta, de forma a termos o mesmo aluno inscrito em diferentes Turnos. Assim aqui temos outro atributo, Regente que remete ao id do Regente do curso em questão, permite-nos assim guardar todos os dados sobre os turnos em que o aluno esta inscrito registados no programa.

DAO’s implementados no Programa

De forma a implementarmos a camada de persistência, procedemos à criação das seguintes classes de DAO’s:

- AlunoDAO
- RegenteDAO
- HorarioDAO
- TurnoDAO
- UCDAO
- UtilizadoresDAO
- InscricaoUCDAO
- TurnoPreferenciaDAO

Estas classes DAOs no nosso programa vão permitir que sejamos capazes de conseguir persistir objetos na base de dados criada, criar objetos a partir da informação que temos na nossa base de dados e encapsular queries SQL.



A inclusão dos DAO’s no sistema implicou as alterações evidenciadas nas figuras acima relativamente à fase anterior. Todos os DAO’s implementam a interface Map para podemos abstrair as classes que os utilizam da sua verdadeira implementação (execução de *queries* SQL na base de dados). Desta maneira, a utilização dos DAO’s torna-se mais simples e assemelha-se à utilização de uma estrutura de dados comum em memória.

Conclusão e Análise dos resultados obtidos

Após a conclusão do desenvolvimento completo do sistema de gestão de horários, podemos fazer uma análise abrangente dos resultados obtidos em todas as fases do projeto.

O Modelo de Domínio desenvolvido inicialmente conseguiu representar de forma clara e eficaz as entidades envolvidas no problema e as relações entre elas. Este modelo serviu como base fundamental para a definição dos requisitos e guiou o desenvolvimento da implementação.

O Modelo de Use Case permitiu descrever as interações e funcionalidades esperadas do sistema, oferecendo uma visão clara de como os utilizadores interagem com ele para realizar diferentes tarefas. Esta definição inicial foi crucial para o desenvolvimento da interface do utilizador, que foi implementada de forma a atender todas as funcionalidades previstas.

A implementação do sistema foi realizada com sucesso, seguindo uma arquitetura bem estruturada com dois subsistemas principais (SSUtilizadores e SSHorarios). O uso do padrão Facade permitiu uma separação clara de responsabilidades e uma interface limpa para acesso às funcionalidades do sistema. A implementação da camada de persistência através dos DAOs proporcionou um armazenamento eficiente e seguro dos dados, garantindo a consistência do sistema.

A interface do utilizador foi desenvolvida de forma intuitiva, com menus específicos para cada tipo de utilizador (aluno e regente), atendendo a todos os requisitos funcionais identificados nos use cases. A implementação modular e a clara separação de responsabilidades facilitam a manutenção e possíveis extensões futuras do sistema.

A base de dados foi projetada de forma a suportar todas as funcionalidades necessárias, com um esquema que reflete adequadamente o modelo de domínio inicial. As tabelas e suas relações permitem o armazenamento eficiente de todas as informações necessárias para o funcionamento do sistema.

Em suma, o projeto atingiu seus objetivos principais, entregando um sistema funcional que atende às necessidades de gestão de horários de forma eficiente e organizada. A arquitetura escolhida e os padrões de design utilizados proporcionaram uma implementação robusta e manutenível, enquanto a interface do utilizador garante uma experiência intuitiva para todos os tipos de utilizadores do sistema.