

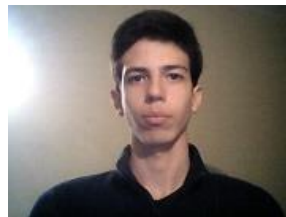


Universidade do Minho
Escola de Engenharia

Trabalho Prático

Desenvolvimento de Sistemas de Software

Alunos:



André Filipe Ferreira de Mira Vieira A78322



Eduardo Gil Ribeiro da Rocha A77048



Ricardo André Araújo Neves A78764

Data:

30 de dezembro de 2017

Índice

1. Introdução	3
2. Objetivos do Trabalho Prático	3
3. Modelo de Domínio	4
3.1. Especificação	4
3.2. Diagrama.....	5
4. Modelo de Use Cases	6
4.1. Diagrama.....	6
4.2 Especificação dos Use Cases.....	7
5. Diagrama de Classes	12
6. Base de Dados.....	13
7. Diagramas de Sequência	14
8. Implementação do Programa.....	15
9. Imagens da Interface Desenvolvida.....	16
10. Como Instalar	18
11. Aspetos a Melhorar.....	18
12. Conclusão.....	19

1. Introdução

Este trabalho prevê a produção de um **sistema de gestão de turnos práticos** de um curso, no âmbito da disciplina de **Desenvolvimento de Sistemas de Software**. Este trabalho de grupo serve para praticar a utilização de **tecnologias UML**, assim como desenvolvimento de sistemas que se encontram divididos em três camadas, assim como foi lecionado nas aulas da disciplina: **Camada de Apresentação, Camada de Negócios e Camada de Dados**.

Logo, o objetivo é ganhar as capacidades necessárias para produzir um sistema de software complexo de forma eficiente e profissionalmente adequada, praticando através da produção de um sistema de tamanho reduzido.

2. Objetivos do Trabalho Prático

Este projeto prático teve como objetivo a criação de um programa que capacitasse a **gerência de turnos práticos** de uma Universidade. O ponto de partida consensual entre os elementos do grupo foi de que todo o programa tinha de ser simples, ter uma interface fácil de utilizar e, principalmente, eficaz.

Com o objetivo principal delineado, o grupo utilizou várias ferramentas que nos possibilitou a criação desta plataforma.

Em primeiro lugar, para o desenvolvimento, tanto dos Modelos de Use Cases, de Domínio e de Classes, como dos vários Diagramas de Sequência das funcionalidades do programa, o grupo tomou partido do **Visual Paradigm**, que foi recomendado pelos docentes da cadeira de Desenvolvimento de Sistemas de Software.

Uma vez criados todos os diagramas, foi utilizada a plataforma **MySQLWorkbench**, na qual criamos a base de dados que, no momento, achamos mais indicada para o suporte ao nosso programa final.

Por fim, o **NetBeans IDE**, uma ferramenta de **Integrated Development Environment (IDE)**, foi o que nos permitiu criar todo o esqueleto do nosso programa, incluindo desenhar a interface gráfica do mesmo, em JavaSwing.

3. Modelo de Domínio

3.1. Especificação

Um **Estatuto Especial** é um atributo conferido a um aluno que lhe proporciona uma série de novos direitos, como participar nos Exames de Época Especial, entre outros. Estes estatutos são: **Trabalhador Estudante**, **Estudante Militar** ou **Atleta de Alto Rendimento**.

Um **Aluno** é o utilizador do serviço, onde pode verificar e trocar os turnos das suas Unidades Curriculares, entre outras funcionalidades, que iremos mencionar mais abaixo.

Uma **Unidade Curricular** é uma disciplina à qual os alunos estão inscritos. Esta é gerida por um **Docente Responsável**, outro potencial utilizador do sistema. Este pode alterar a composição dos turnos da UC que gere.

Um **Docente** é o terceiro e último utilizador da aplicação. Está associado a uma ou mais Unidades Curriculares, e pode registar as faltas dos alunos, se necessário.

Um **Turno** é um grupo criado para organizar os alunos, de modo a não haver conflito de horários. Pode ser **Teórico** ou **Prático**. Se, por algum motivo, um aluno quiser trocar um dado turno prático com outro colega, pode fazê-lo, recorrendo a este programa. Um turno tem um número máximo de alunos, que, por nenhum meio, pode ser ultrapassado.

A **Sala** é onde um turno de uma UC é lecionado, por um docente. À sala está associada uma dada **Capacidade**, ou seja, a quantidade de alunos que podem estar a assistir á aula. Se, por uma razão especial, um aluno de outro turno quiser assistir á aula de um outro turno, apenas será lhe concedida permissão se a capacidade da sala não estiver esgotada.

O **Registo de Faltas** contém todos os registos dos alunos e a quantidade de vezes que não compareceram a uma aula. É gerido pelo docente de cada turno.

O **Registo de Trocas** contém todas as trocas de turnos realizados pelos alunos numa Unidade Curricular. É gerido pelo docente responsável.

3.2. Diagrama

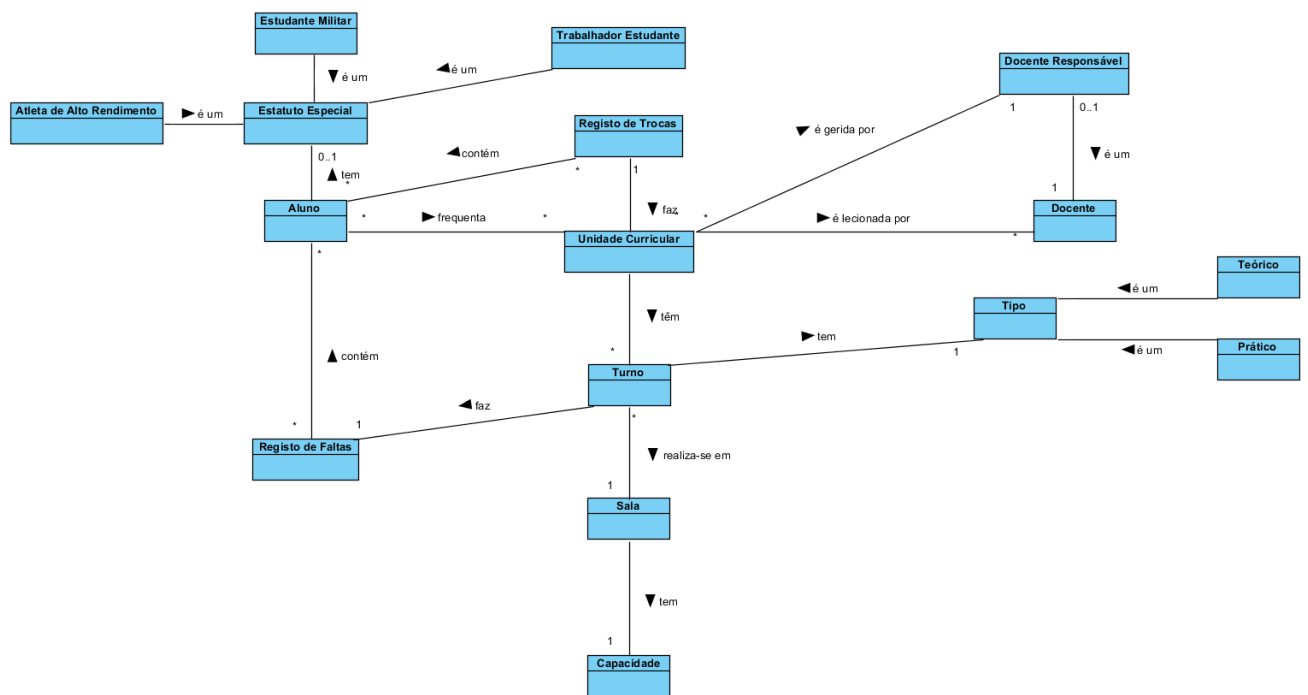


Figura 1. Modelo de Domínio



Figura 2. Logotipo da ferramenta Visual Paradigm

4. Modelo de Use Cases

4.1. Diagrama



Figura 3. Modelo de Use Cases

Neste **Modelo de Use Cases**, podemos verificar quais as funcionalidades do programa desenvolvido, tais como, efetuar o Login, registar uma nova conta, inscrever-se numa UC, entre outros processos.

De cada lado do modelo, verificamos a existência de dois utilizadores distintos, o **Aluno** e o **Docente**. Estes, naturalmente, só conseguem aceder a funcionalidades que lhes são proporcionadas. Como exemplo, o Aluno não consegue verificar quais os alunos de um dado turno, bem como o Docente não pode trocar o turno a que foi atribuído.

4.2 Especificação dos Use Cases

Efetuar Login

Info Use Case Notes Flow of Events Details Requirements Diagrams Test Plan References Description

Main_converted

Super Use Case		
Author	André Vieira, Eduardo Rocha, Ricardo Neves	
Date	19/out/2017 16:25:38	
Brief Description		
Preconditions		
Post-conditions	Utilizador é autenticado	
Flow of Events		Actor Input
	1	Utilizador introduz o seu nome e password
	2	
	3	
	4	
		System Response
		Sistema procura se essas credenciais se encontram na base de dados
		Sistema encontra essas credenciais
		Sistema autentica o utilizador e leva-o para um novo ecrã
Exceção 1 [sistema não encontra as credenciais] (passo 3)		Actor Input
	1	
		System Response
		Sistema informa que não encontrou as credenciais

Figura 4. Especificação do Use Case “Efetuar Login”

Verificar Informação de UC

Info Use Case Notes Flow of Events Details Requirements Diagrams Test Plan References Description

Main

Super Use Case		
Author	André Vieira, Eduardo Rocha, Ricardo Rocha	
Date	8/nov/2017 19:56:49	
Brief Description		
Preconditions	Utilizador autenticado	
Post-conditions		
Flow of Events		Actor Input
	1	
	2	Utilizador escolhe uma UC
	3	
	4	
		System Response
		Sistema fornece lista de UCs
		Sistema verifica a informação associada a essa UC
		Sistema mostra lista de turnos inscritos na UC
Exception 1 (Utilizador não tem UCs atribuídas) (passo 1)		Actor Input
	1	
		System Response
		Sistema indica que utilizador não está inscrito a nenhuma UC

Figura 5. Especificação do Use Case “Verificar Informação de UC”

Verificar docentes da UC

Info
Use Case Notes
Flow of Events
Details
Requirements
Diagrams
Test Plan
References
Description

20th Century Font
8

Main

Super Use Case		
Author	André Vieira, Eduardo Rocha, Ricardo Neves	
Date	8/nov/2017 18:58:36	
Brief Description		
Preconditions	Utilizador está autenticado	
Post-conditions		
Flow of Events		Actor Input
		System Response
	1	Sistema mostra lista de UCs a que utilizador está inscrito
	2	Utilizador escolhe UC
	3	Sistema verifica lista de docentes de UC
4	Sistema mostra lista de docentes ao utilizador	
Exception 1 (utilizador não está inscrito a qualquer UC) (passo 1)		Actor Input
		System Response
	1	Sistema indica que utilizador não está inscrito a nenhuma UC

Figura 6. Especificação do Use Case “Verificar docentes da UC”

Efetuar registo

Info
Use Case Notes
Flow of Events
Details
Requirements
Diagrams
Test Plan
References
Description

20th Century Font
8

Main

Super Use Case		
Author	André Vieira, Eduardo Rocha, Ricardo Neves	
Date	8/nov/2017 19:12:27	
Brief Description		
Preconditions		
Post-conditions	Utilizador fica registado e autenticado no sistema	
Flow of Events		Actor Input
		System Response
	1	Utilizador indica nome, email e palavra-passe
	2	Sistema verifica se dados são válidos
	3	Sistema regista utilizador
Exception 1 (dados inválidos) (passo 2)		Actor Input
		System Response
	1	Sistema informa que dados introduzidos são inválidos

Figura 7. Especificação do Use Case “Efetuar registo”

Trocar turno

Info
Use Case Notes
Flow of Events
Details
Requirements
Diagrams
Test Plan
References
Description

20th Century Font
8

Main

Super Use Case			
Author	Ricardo		
Date	8/nov/2017 20:09:47		
Brief Description			
Preconditions			
Post-conditions			
Flow of Events		Actor Input	System Response
	1		Sistema mostra lista de UCs a que utilizador está inscrito
	2	Utilizador escolhe UC	
	3	Utilizador indica aluno com que quer trocar	
	4		Sistema verifica se troca é válida
Exception 1 (utilizador não está inscrito a UCs) [passo 1]		Actor Input	System Response
	1		Sistema indica que utilizador não está inscrito a nenhuma UC
Exception 2 (troca inválida) [passo 4]		Actor Input	System Response
	1		Sistema informa utilizador que troca é inválida

Figura 8. Especificação do Use Case “Trocar turno”

Registar faltas

Info
Use Case Notes
Flow of Events
Details
Requirements
Diagrams
Test Plan
References
Description

20th Century Font
8

Main

Super Use Case			
Author	André Vieira, Eduardo Rocha, Ricardo Neves		
Date	8/nov/2017 20:23:35		
Brief Description			
Preconditions	Utilizador está autenticado		
Post-conditions			
Flow of Events		Actor Input	System Response
	1		Sistema mostra lista de UCs e turnos a que o utilizador está associado
	2	Utilizador escolhe turno	
	3		Sistema mostra lista de alunos inscritos ao turno
	4	Utilizador escolhe aluno	
	5	Utilizador indica número de faltas	
	6		Sistema atualiza número de faltas do aluno
Exception 1 (utilizador não está inscrito a UCs) [passo 1]		Actor Input	System Response
	1		Sistema indica que utilizador não está inscrito a nenhuma UC

Figura 9. Especificação do Use Case “Registar faltas”

Inscrever na UC

Info
Use Case Notes
Flow of Events
Details
Requirements
Diagrams
Test Plan
References
Description

20th Century Font
8

Main

Super Use Case		
Author	André Vieira, Eduardo Rocha, Ricardo Neves	
Date	8/nov/2017 21:57:33	
Brief Description		
Preconditions	Utilizador autenticado	
Post-conditions	Utilizador está inscrito na UC	
Flow of Events		Actor Input
		System Response
	1	Sistema apresenta todas as UCs
	2	Utilizador escolhe uma UC
	3	Sistema verifica se inscrição é válida
	4	Sistema realiza inscrição
Exception 1 (inscrição inválida) [passo 3]		Actor Input
		System Response
1	Sistema informa que utilizador está inscrito	
1	Sistema informa que inscrição não se pode realizar	

Figura 10. Especificação do Use Case “Inscrever na UC”

Verificar alunos de um turno

Info
Use Case Notes
Flow of Events
Details
Requirements
Diagrams
Test Plan
References
Description

20th Century Font
8

Main

Super Use Case		
Author	André Vieira, Eduardo Rocha, Ricardo Neves	
Date	8/nov/2017 22:16:16	
Brief Description		
Preconditions	Utilizador autenticado	
Post-conditions		
Flow of Events		Actor Input
		System Response
	1	Sistema imprime lista de turnos a que o utilizador está associado
	2	Utilizador escolhe turno
	3	Sistema imprime lista de alunos pertencentes a turno
Exception 1 (utilizador não está associado a UCs) [passo 1]		Actor Input
		System Response
1	Sistema informa que utilizador não está associado a nenhuma UC	

Figura 11. Especificação do Use Case “Verificar alunos de um turno”

Verificar registo de faltas

Info Use Case Notes Flow of Events Details Requirements Diagrams Test Plan References Description

20th Century Font 8

Main

Super Use Case		
Author	Ricardo	
Date	8/nov/2017 22:08:28	
Brief Description		
Preconditions		
Post-conditions		
Flow of Events		Actor Input
		System Response
	1	Sistema imprime lista de UCs a que utilizador está associado
	2	Utilizador escolhe UC
	3	Sistema verifica se existe registo de faltas
4	Sistema mostra registo de faltas	
Exception 1 (utilizador não está associado a UCs) [passo 1]	Actor Input	System Response
	1	Sistema informa que utilizador não está associado a nenhuma UC
Exception 2 (não existe registo de faltas) [passo 3]	Actor Input	System Response
	1	Sistema informa que não existe registo de faltas

Figura 12. Especificação do Use Case “Verificar registo de faltas”

Verificar registo de trocas

Info Use Case Notes Flow of Events Details Requirements Diagrams Test Plan References Description

20th Century Font 8

Main

Super Use Case		
Author	André Vieira, Eduardo Rocha, Ricardo Neves	
Date	8/nov/2017 22:03:37	
Brief Description		
Preconditions	Utilizador autenticado	
Post-conditions		
Flow of Events		Actor Input
		System Response
	1	Sistema mostra lista de UCs
	2	Utilizador seleciona UC
	3	Sistema verifica se existe registo de trocas para UC
4	Sistema mostra registo de trocas	
Exception 1 (utilizador não está associado a UCs) [passo 1]	Actor Input	System Response
	1	Sistema informa que utilizador não está associado a nenhuma UC
Exception 2 (não existe registo de trocas) [passo 2]	Actor Input	System Response
	1	Sistema informa que não existe registo de trocas

Figura 13. Especificação do Use Case “Verificar registo de trocas”

5. Diagrama de Classes

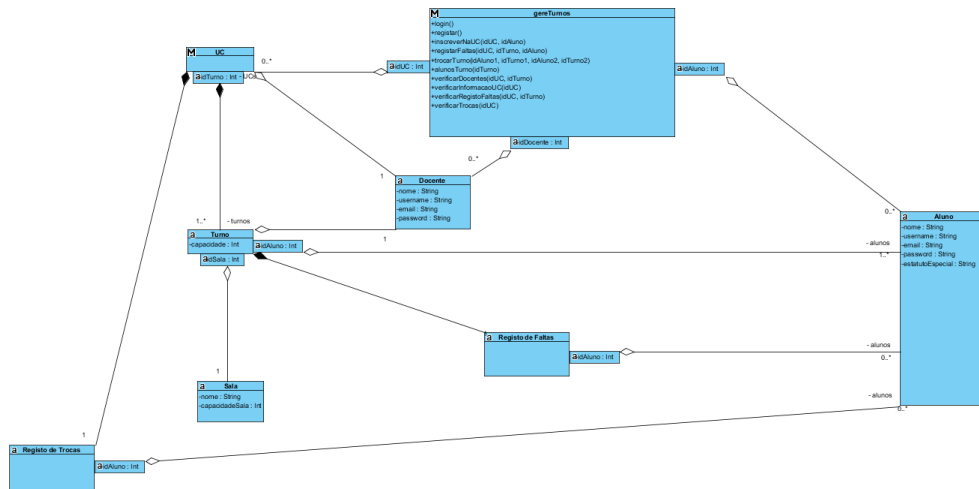


Figura 14. Diagrama de Classes sem DAOs

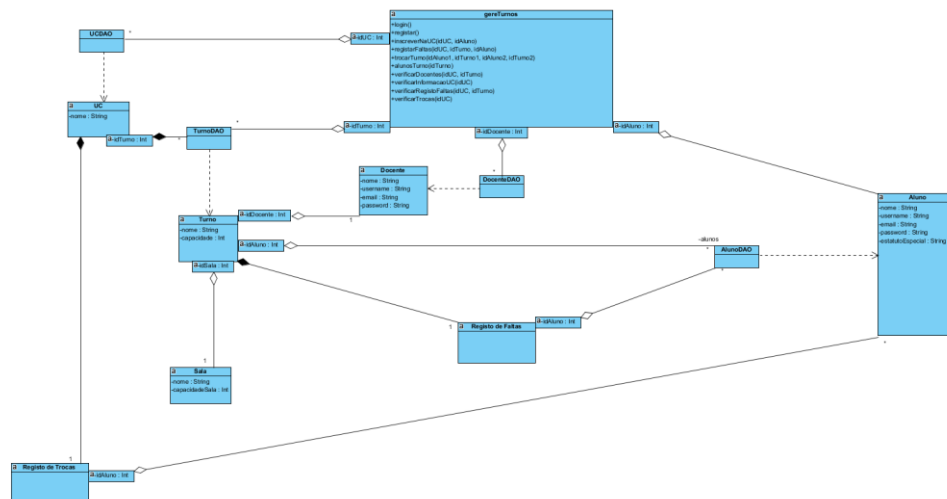


Figura 15. Diagrama de Classes com os diferentes DAOs

Como podemos observar, estão aqui disponíveis os dois **Diagramas de Classes** realizados pelo grupo de trabalho.

O primeiro Diagrama não contém qualquer tipo de **Data Access Objects** (DAOs). Sem estes DAOS, a informação ficaria apenas **alocada em memória**, não sendo possível voltar a aceder à mesma, uma vez terminada a execução do programa.

Depois de tomar conhecimento destes DAOs nas aulas de DSS, o grupo reuniu-se e decidiu que o melhor caminho a seguir era utilizar este mecanismo que guarda toda a informação depositada em uma **base de dados**.

6. Base de Dados

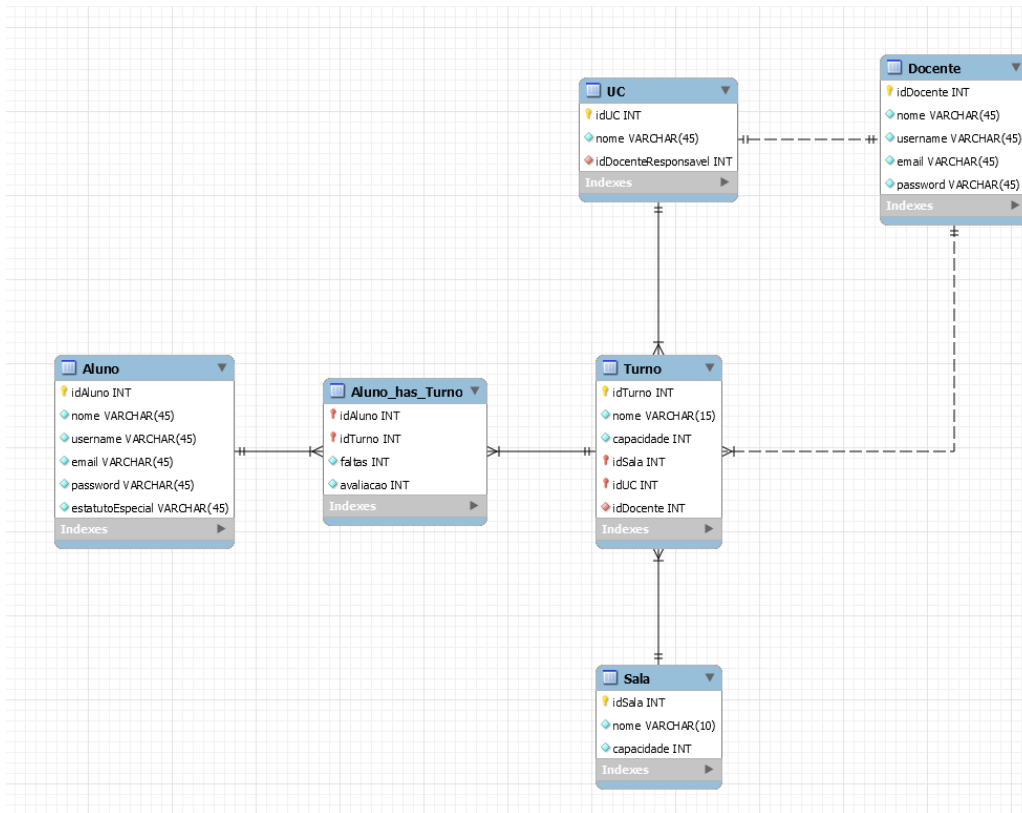


Figura 16. Esquema lógico da base de dados utilizada

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- Schema dsdb

-- Schema dsdb
CREATE SCHEMA IF NOT EXISTS 'dsdb' DEFAULT CHARACTER SET utf8 ;
USE 'dsdb' ;

-- Table 'dsdb'.'Docente'
CREATE TABLE IF NOT EXISTS 'dsdb'.'Docente' (
  'idDocente' INT NOT NULL AUTO_INCREMENT,
  'nome' VARCHAR(45) NOT NULL,
  'username' VARCHAR(45) NOT NULL,
  'email' VARCHAR(45) NOT NULL,
  'password' VARCHAR(45) NOT NULL,
  PRIMARY KEY ('idDocente'))
ENGINE = InnoDB;

-- Table 'dsdb'.'UC'
CREATE TABLE IF NOT EXISTS 'dsdb'.'UC' (
  'idUC' INT NOT NULL AUTO_INCREMENT,
  'nome' VARCHAR(45) NOT NULL,
  'idDocenteResponsavel' INT NOT NULL,
  PRIMARY KEY ('idUC'),
  INDEX 'fk_UC_Docente1_idx' ('idDocenteResponsavel' ASC),
  CONSTRAINT 'fk_UC_Docente1'
    FOREIGN KEY ('idDocenteResponsavel')
      REFERENCES 'dsdb'.'Docente' ('idDocente')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- Table 'dsdb'.'Aluno'
CREATE TABLE IF NOT EXISTS 'dsdb'.'Aluno' (
  'idAluno' INT NOT NULL AUTO_INCREMENT,
  'nome' VARCHAR(45) NOT NULL,
  'username' VARCHAR(45) NOT NULL,
  'email' VARCHAR(45) NOT NULL,
  'password' VARCHAR(45) NOT NULL,
  'estatutoEspecial' VARCHAR(45) NOT NULL,
  PRIMARY KEY ('idAluno'))
ENGINE = InnoDB;

-- Table 'dsdb'.'Sala'
CREATE TABLE IF NOT EXISTS 'dsdb'.'Sala' (
  'idSala' INT NOT NULL AUTO_INCREMENT,
  'nome' VARCHAR(10) NOT NULL,
  'capacidade' INT NOT NULL,
  PRIMARY KEY ('idSala'))
ENGINE = InnoDB;

-- Table 'dsdb'.'Turno'
CREATE TABLE IF NOT EXISTS 'dsdb'.'Turno' (
  'idTurno' INT NOT NULL AUTO_INCREMENT,
  'nome' VARCHAR(15) NOT NULL,
  'capacidade' INT NOT NULL,
  'idSala' INT NOT NULL,
  'idUC' INT NOT NULL,
  'idDocente' INT NOT NULL,
  PRIMARY KEY ('idTurno'),
  INDEX 'fk_Turno_Sala1_idx' ('idSala' ASC),
  INDEX 'fk_Turno_Docente1_idx' ('idDocente' ASC),
  CONSTRAINT 'fk_Turno_Sala1'
    FOREIGN KEY ('idSala')
      REFERENCES 'dsdb'.'Sala' ('idSala')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT 'fk_Turno_UC1'
    FOREIGN KEY ('idUC')
      REFERENCES 'dsdb'.'UC' ('idUC')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT 'fk_Turno_Docente1'
    FOREIGN KEY ('idDocente')
      REFERENCES 'dsdb'.'Docente' ('idDocente')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- Table 'dsdb'.'Aluno_has_Turno'
CREATE TABLE IF NOT EXISTS 'dsdb'.'Aluno_has_Turno' (
  'idAluno' INT NOT NULL,
  'idTurno' INT NOT NULL,
  'faltas' INT NOT NULL,
  'avaliacao' INT NOT NULL,
  PRIMARY KEY ('idAluno', 'idTurno'),
  INDEX 'fk_Aluno_has_Turno_Aluno1_idx' ('idAluno' ASC),
  INDEX 'fk_Aluno_has_Turno_Turno1_idx' ('idTurno' ASC),
  CONSTRAINT 'fk_Aluno_has_Turno_Aluno1'
    FOREIGN KEY ('idAluno')
      REFERENCES 'dsdb'.'Aluno' ('idAluno')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT 'fk_Aluno_has_Turno_Turno1'
    FOREIGN KEY ('idTurno')
      REFERENCES 'dsdb'.'Turno' ('idTurno')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Figura 17. Script de criação da base de dados acima

Como já foi referido no ponto 2 deste mesmo relatório, a ferramenta utilizada para o **desenvolvimento da base de dados** que suportaria o nosso programa final foi o **MySQLWorkbench**, devido a ser relativamente fácil de utilizar e bastante intuitiva.

De acordo com o esquema lógico fornecido, todas as entidades estão relacionadas entre si. Assim, usando **Java Database Connectivity** (JDBC), podemos aceder a toda informação presente da base de dados muito rapidamente, executando certas **Querys**. Para além disto, se for necessário atualizar alguma entrada, é só escrever e executar um **Update**.

Alguns dos **requisitos** iniciais desta base de dados, que podem ser compreendidos, se analisar o esquema lógico cuidadosamente, foram que:

- ✓ Cada **Entidade** apresenta uma chave primária, que é sempre dada pelo seu **Identifier** (ID);
- ✓ Uma **UC** apenas pode ter um **docente responsável**;
- ✓ Um **Turno** apenas é lecionado por um **Docente**, apesar de um **Docente** poder lecionar vários **Turnos** (possivelmente, de **Unidades Curriculares distintas**).



Figura 18. Logotipo da ferramenta MySQLWorkbench

7. Diagramas de Sequência

Disponíveis no **Anexo** presente do final deste Relatório.

8. Implementação do Programa

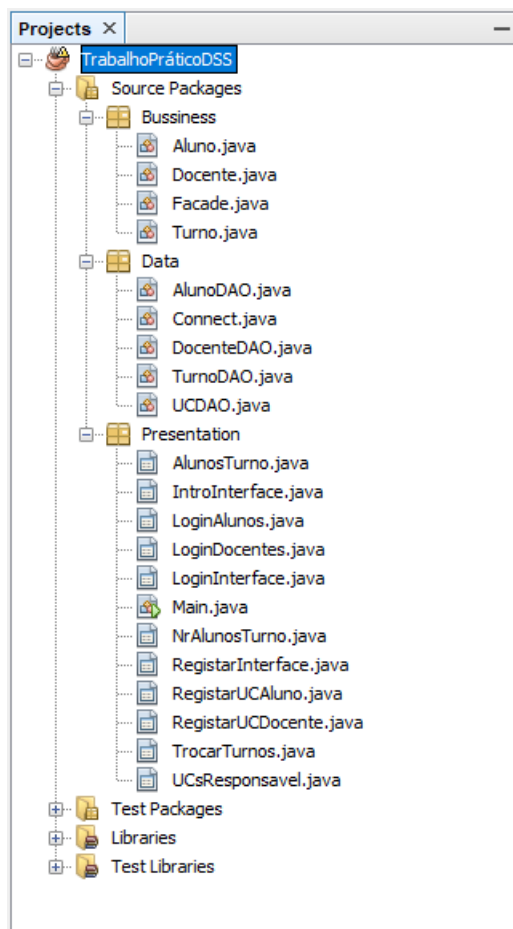


Figura 19. Classes desenvolvidas

Para uma melhor organização das classes do nosso sistema, foram criados **três Packages**: **Presentation** (onde se encontram as Interfaces do programa), **Business** (onde se encontra a **Facade**, que conecta este Package com o Presentation Package) e o **Data Package** (onde se encontra as classes que vão buscar as várias informações presentes na base de dados).



Figura 20. Logotipo da ferramenta NetBeans

9. Imagens da Interface Desenvolvida

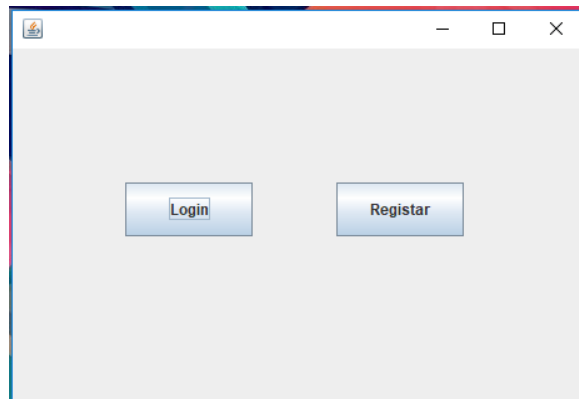


Figura 21. Interface para efetuar Login ou registrar no sistema

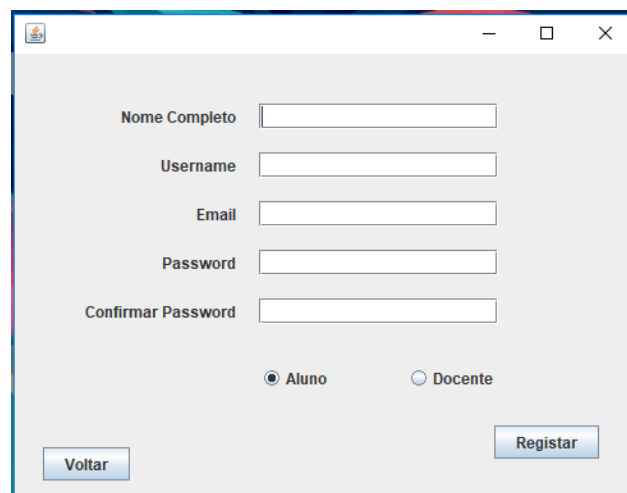
A screenshot of a web application window for account registration. The window has a light gray background and a standard title bar. The form contains the following elements: five text input fields labeled 'Nome Completo', 'Username', 'Email', 'Password', and 'Confirmar Password' stacked vertically; two radio buttons below the password fields, labeled 'Aluno' (which is selected) and 'Docente'; and two buttons at the bottom, 'Voltar' on the left and 'Registrar' on the right.

Figura 22. Interface para registrar uma nova conta

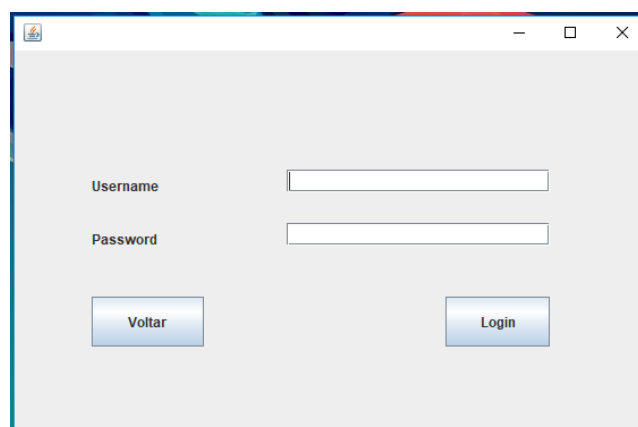
A screenshot of a web application window for login. The window has a light gray background and a standard title bar. The form contains two text input fields labeled 'Username' and 'Password' stacked vertically. Below the password field, there are two buttons: 'Voltar' on the left and 'Login' on the right.

Figura 23. Interface para efetuar Login

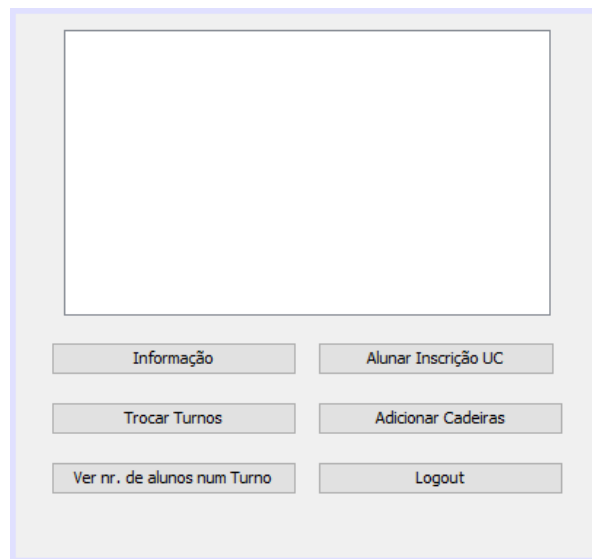


Figura 24. Interface disponível após um Aluno ter entrado no sistema

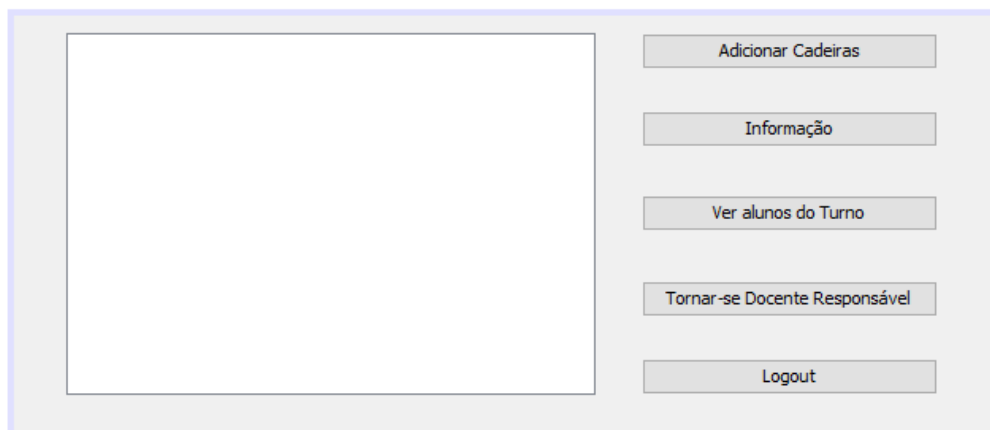


Figura 25. Interface disponível após um Docente ter entrado no sistema

10. Como Instalar

1. Instalar a ferramenta **MySQLWorkbench**.
2. Executar **Scripts de Criação** e de **Povoamento** da base de dados.
3. Instalar a ferramenta **NetBeans IDE**.
4. Alterar, na classe **Connect**, as **credenciais de utilizador**, de modo a conectar-se à base de dados.
5. Run.

11. Aspetos a Melhorar

Como grande parte de todos os projetos realizados, tanto por um simples programador que está a aprender os princípios da programação, como por uma empresa de reconhecimento Mundial, o nosso não é exceção.

Assim, sendo que não devemos apresentar os aspetos positivos, há certas melhorias que poderíamos ter realizado para tornar o nosso programa ainda mais produtivo e eficaz.

Por exemplo, os aspetos que deveríamos ter implementado / melhorado no nosso código são:

- O Docente Responsável pela Unidade Curricular ver o Registo de Trocas da mesma;
- Dar a capacidade aos Docentes para trocar de turnos, ou mesmo, anular a sua inscrição ao mesmo.

Para além destes aspetos, apesar de ser secundário, gostaríamos de tornar a interface um pouco mais apelativa e interativa.

12. Conclusão

A produção de **Diagramas de Classe** tornou a organização do projeto mais fácil e a produção de **Diagramas de Sequências** facilitou a realização dos métodos do programa.

Apesar de o trabalho realizado ter espaço para melhorias, entendendo que um dos objetivos deste trabalho era entender os materiais lecionados na Unidade Curricular, assim como os seus usos na vida profissional de um Engenheiro Informático, podemos dizer que saímos do trabalho com entendimento das utilidades das **tecnologias UML**.

O trabalho ajudou-nos também a entender melhor como se deve organizar um programa e como se deve integrar uma base de dados num Sistema de Software.

Anexos (Diagramas de Sequência)

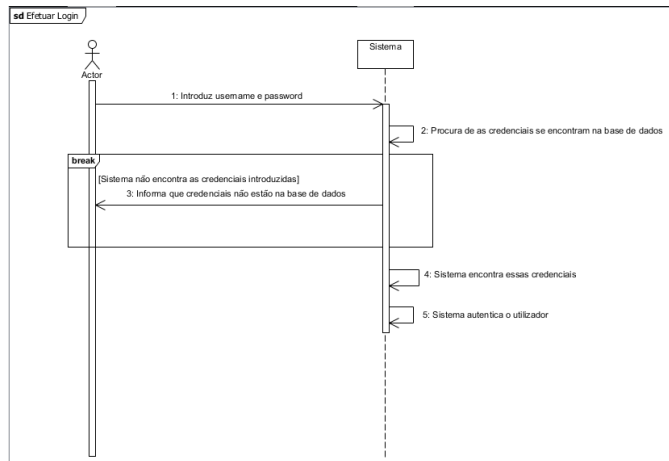


Figura 26.

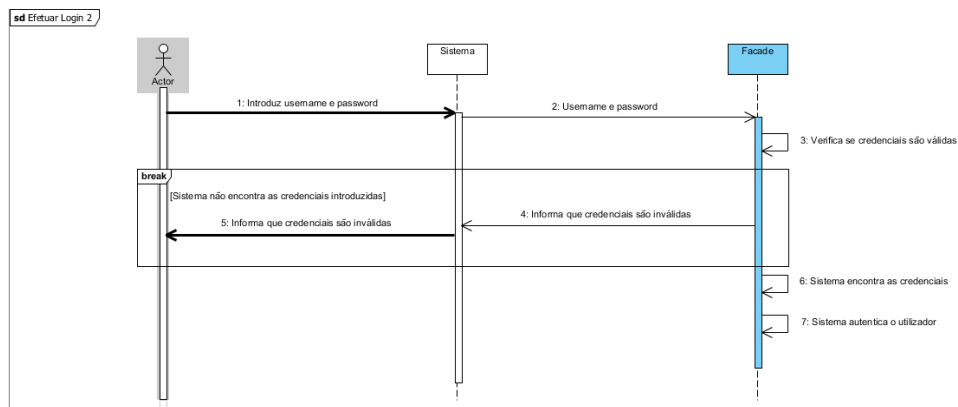


Figura 27.

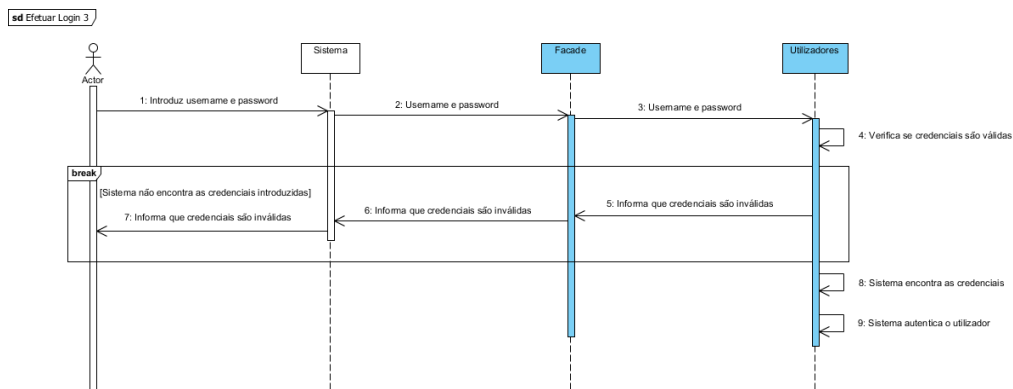


Figura 28.

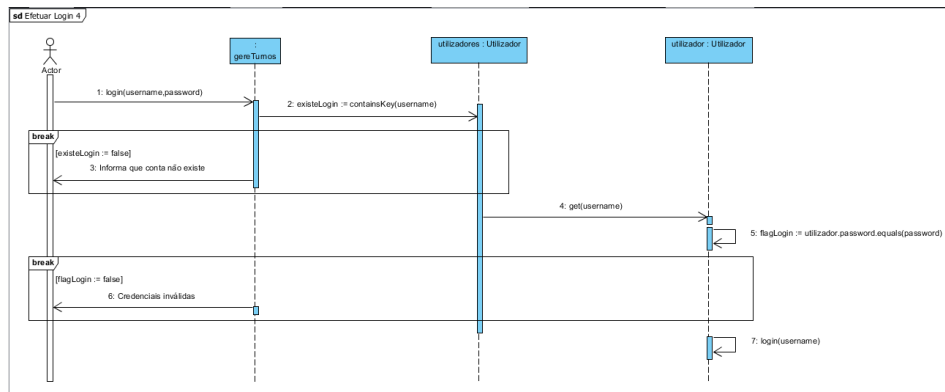


Figura 29.

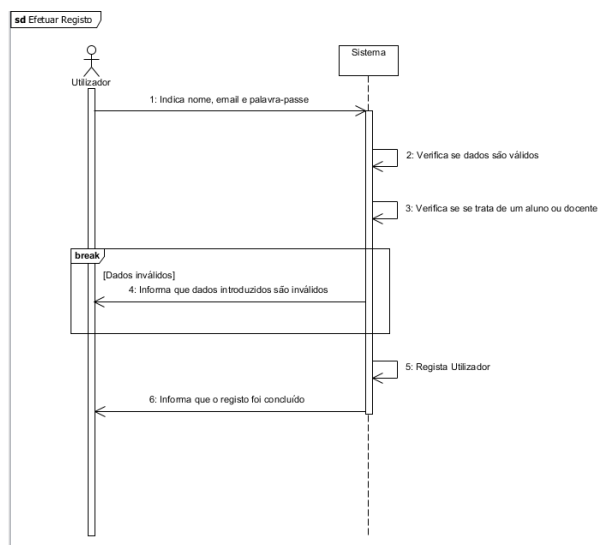


Figura 30.

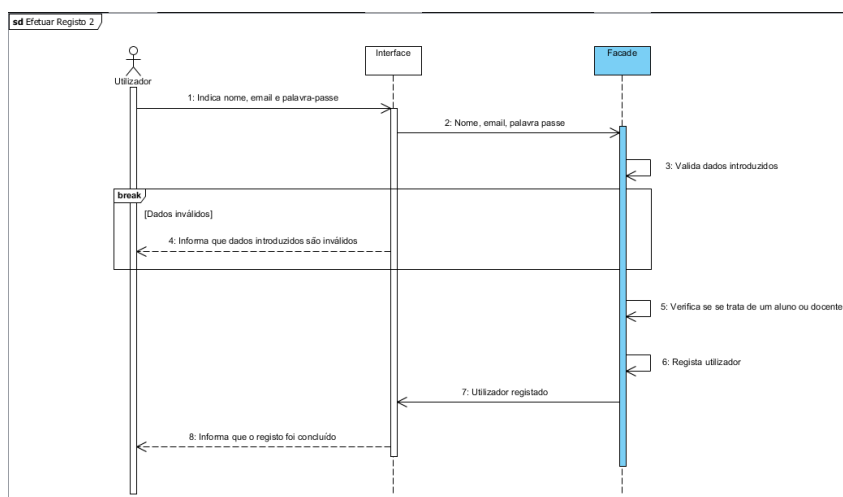


Figura 31.

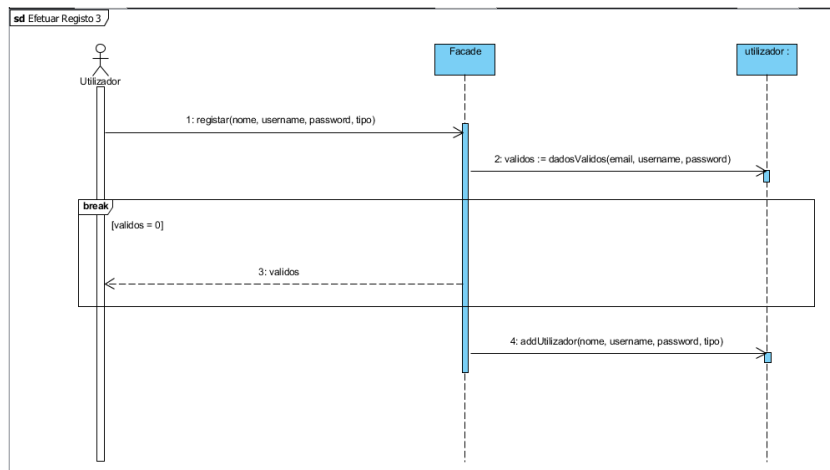


Figura 32.

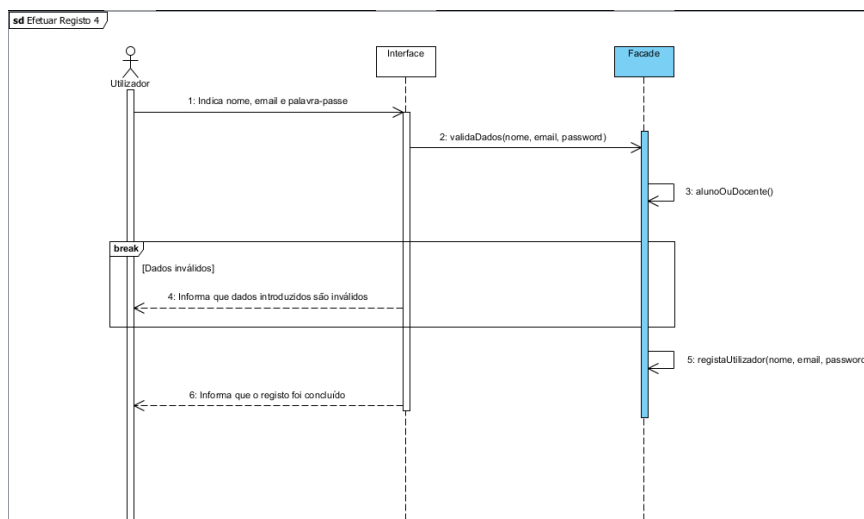


Figura 33.

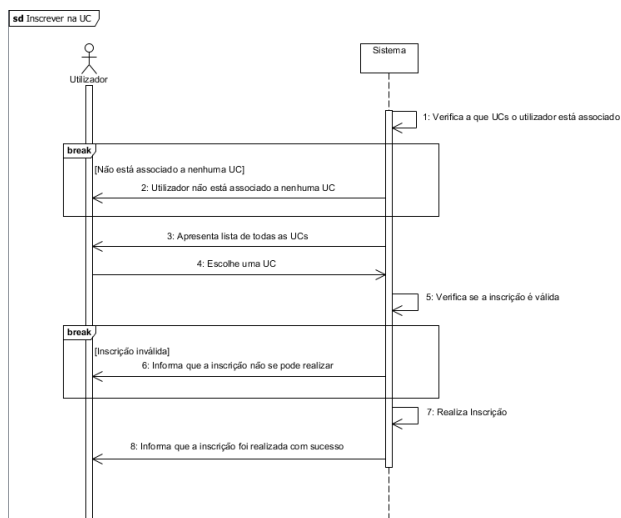


Figura 34.

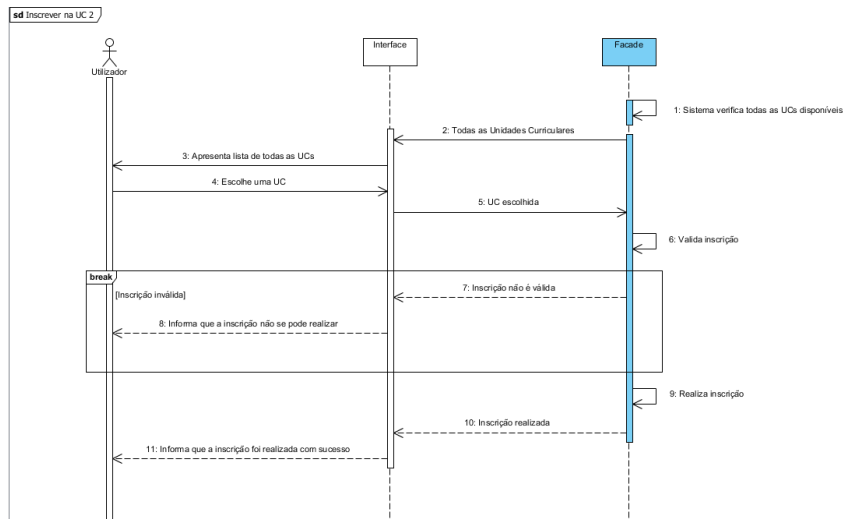


Figura 35.

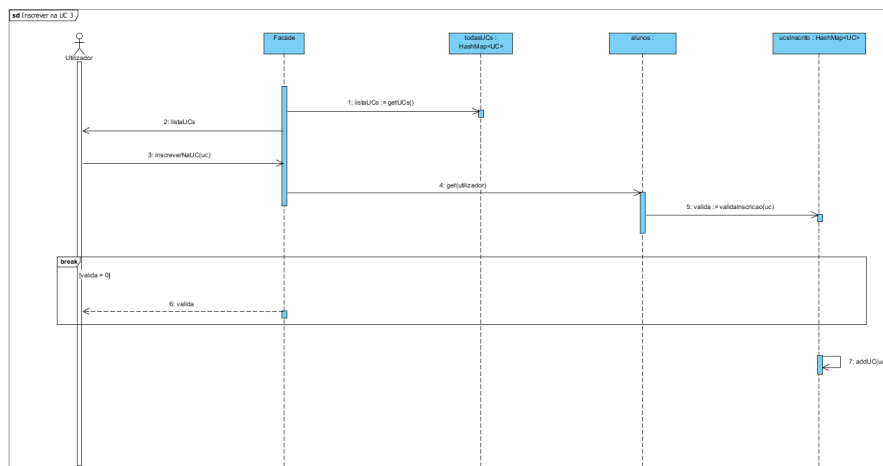


Figura 36.

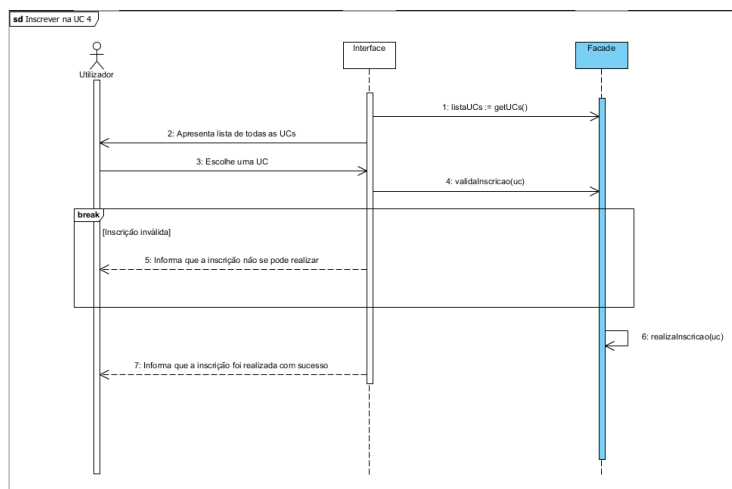


Figura 37.

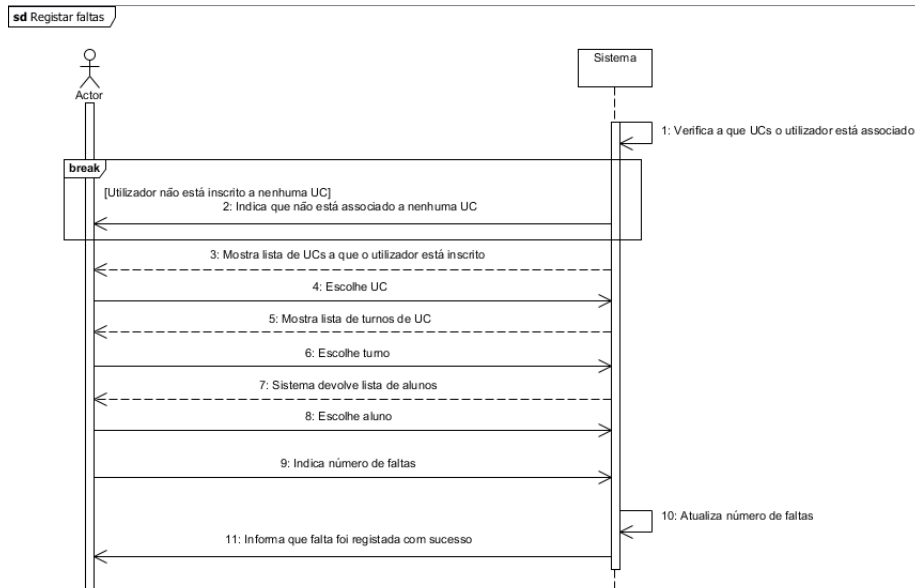


Figura 38.

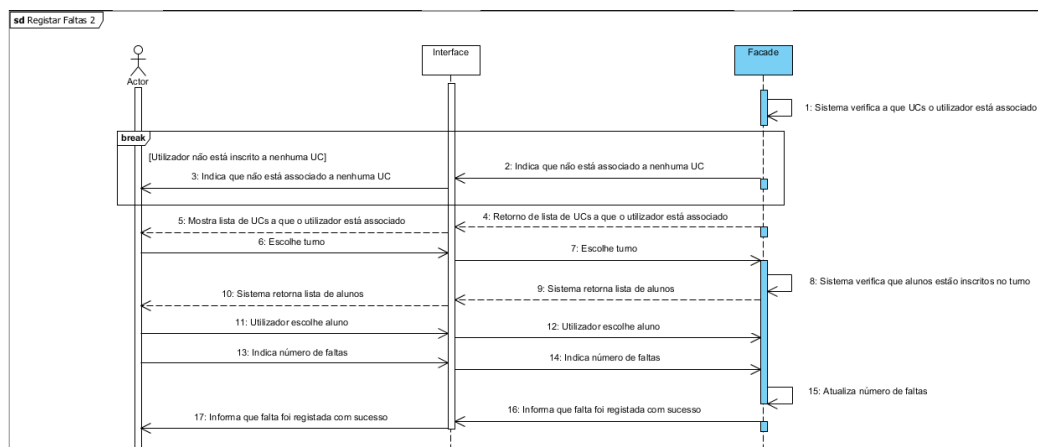


Figura 39.

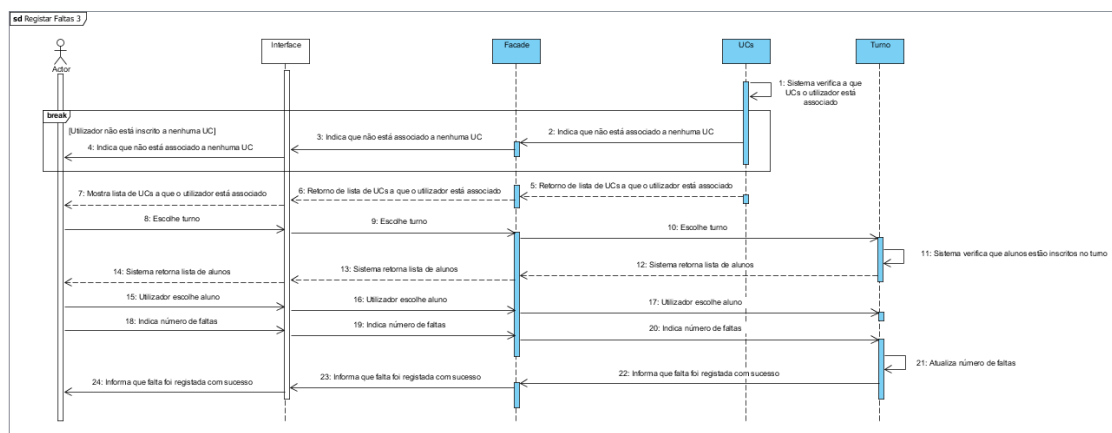


Figura 40.

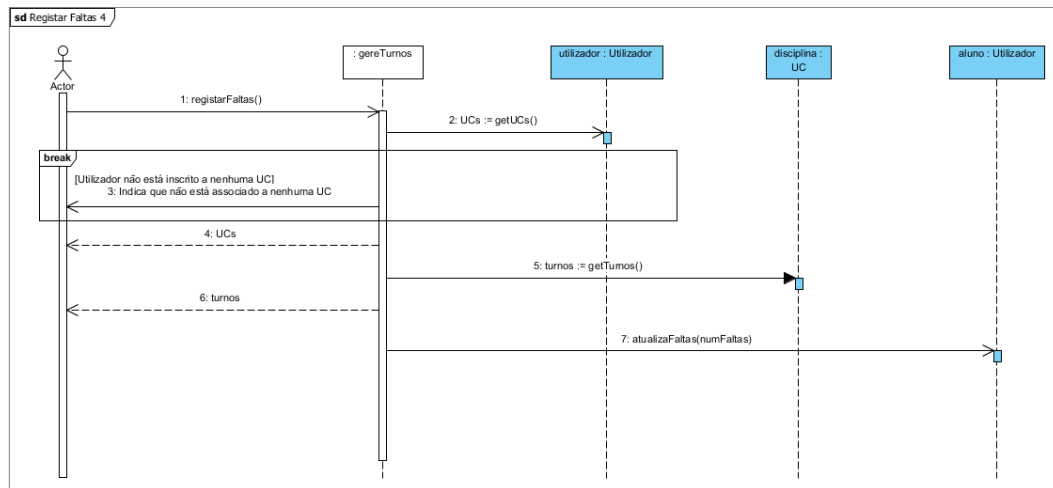


Figura 41.

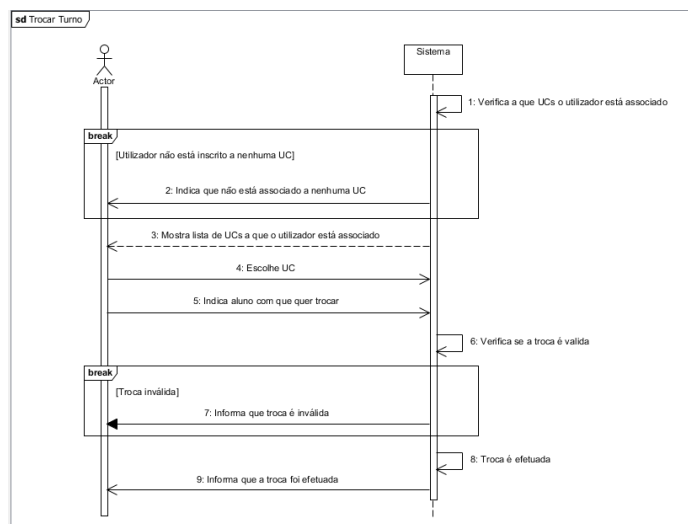


Figura 42.

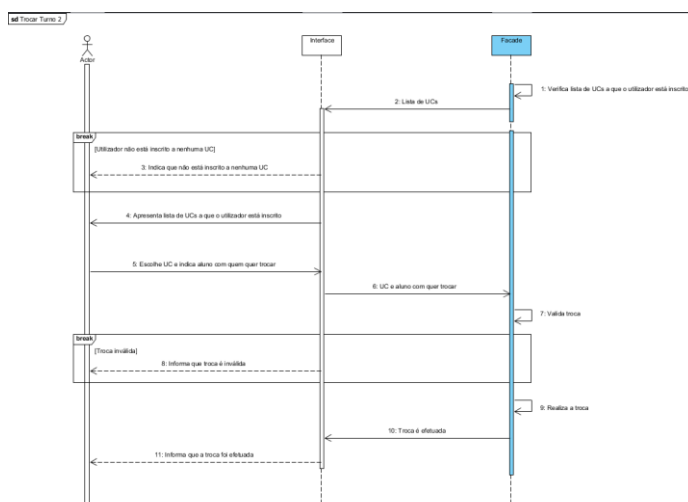


Figura 43.

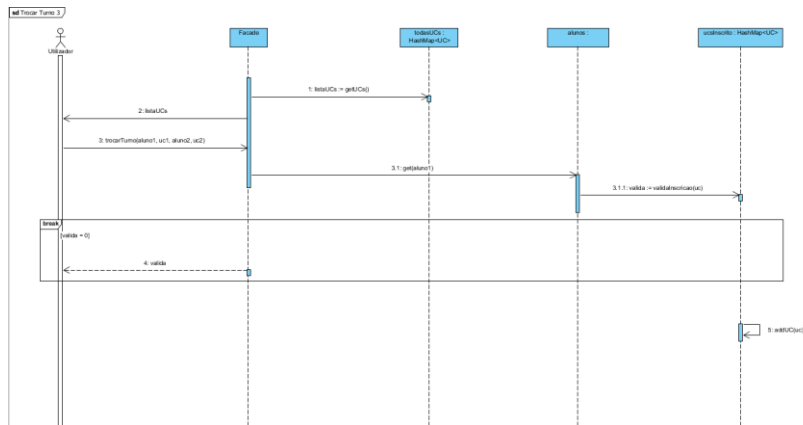


Figura 44.

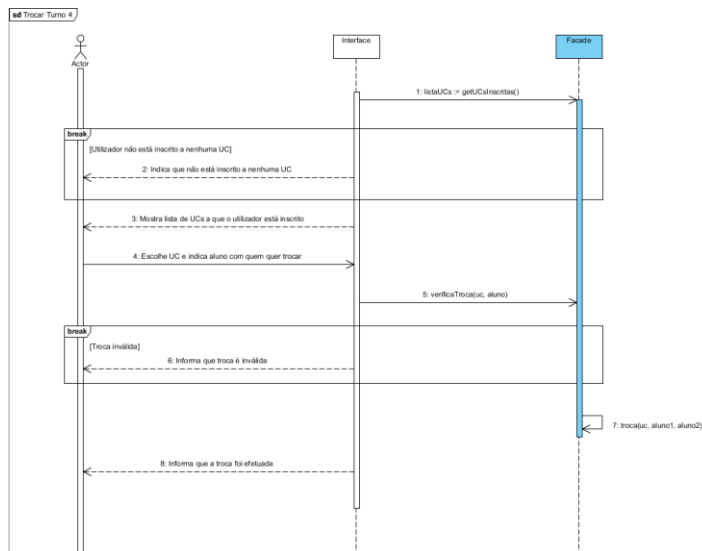


Figura 45.

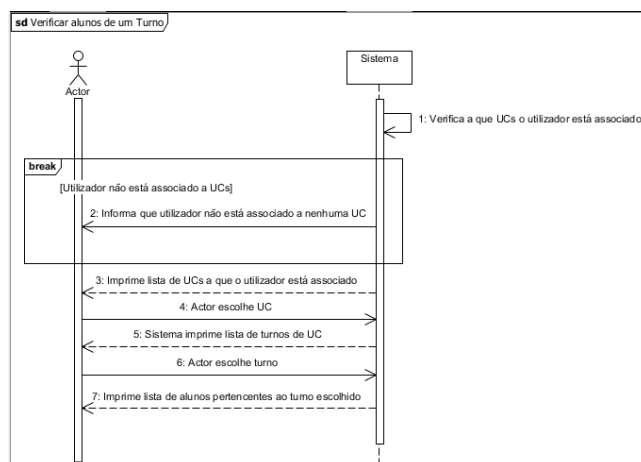


Figura 46.

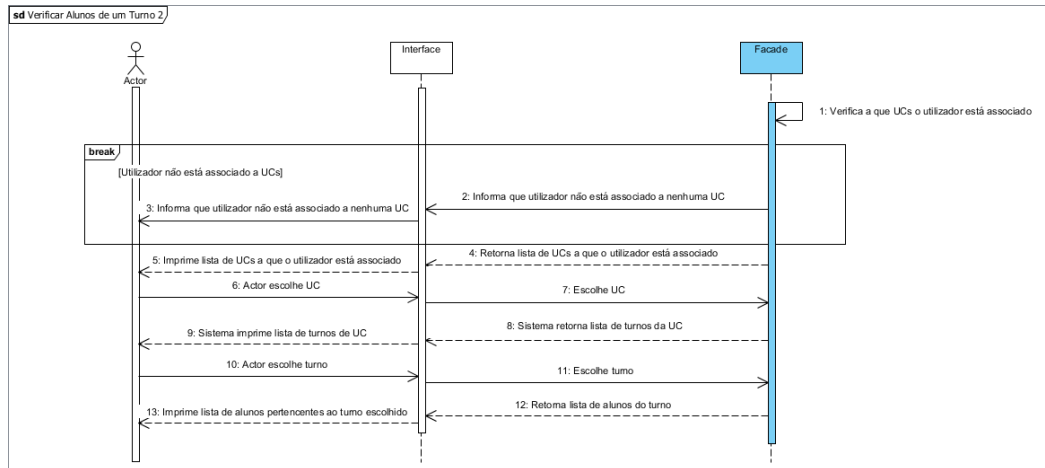


Figura 47.

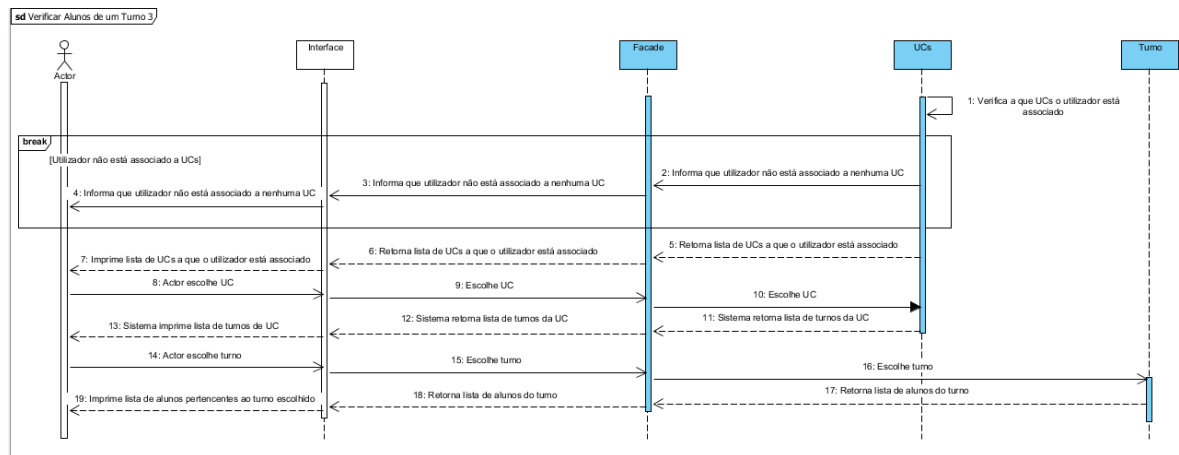


Figura 48.

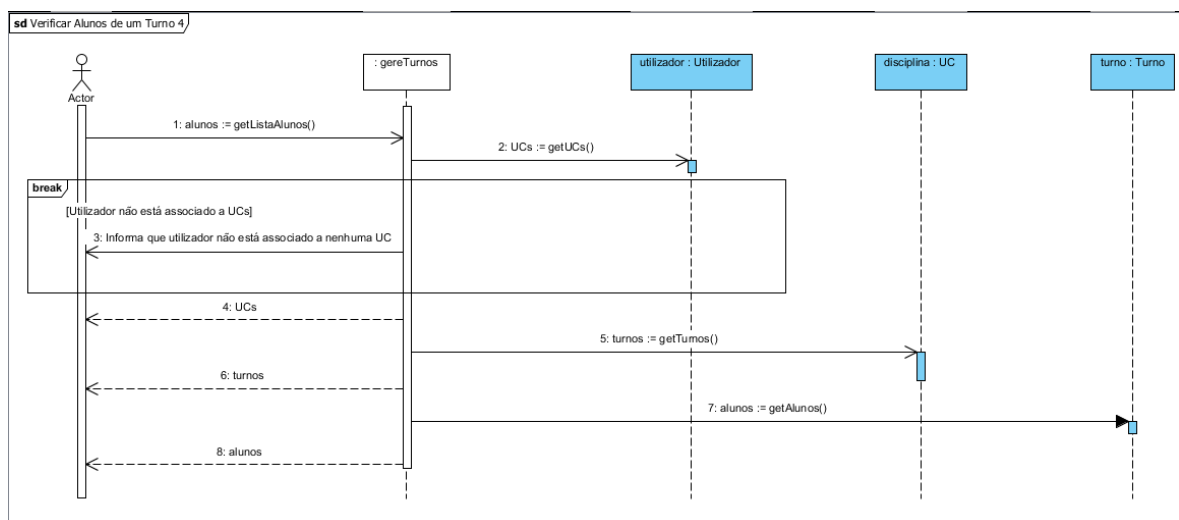


Figura 49.

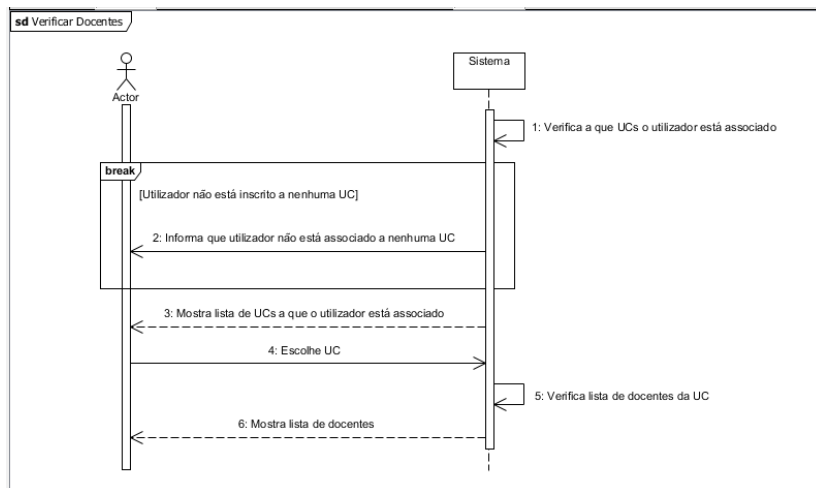


Figura 50.

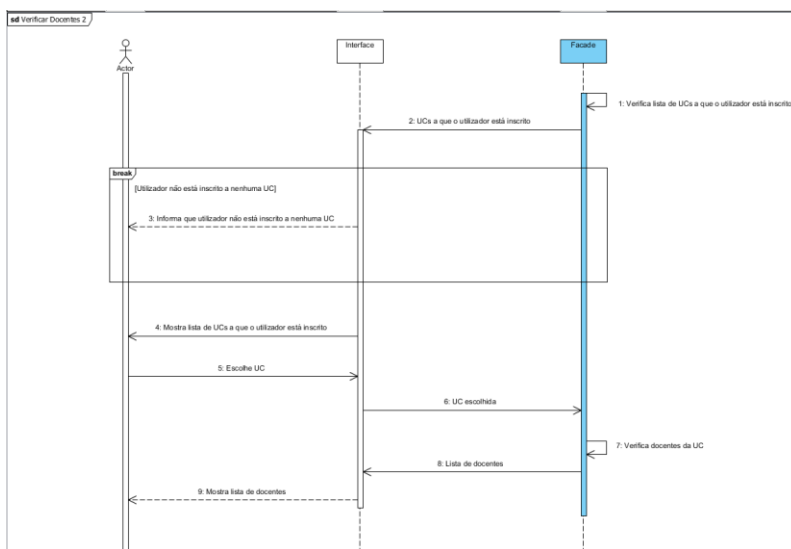


Figura 51.

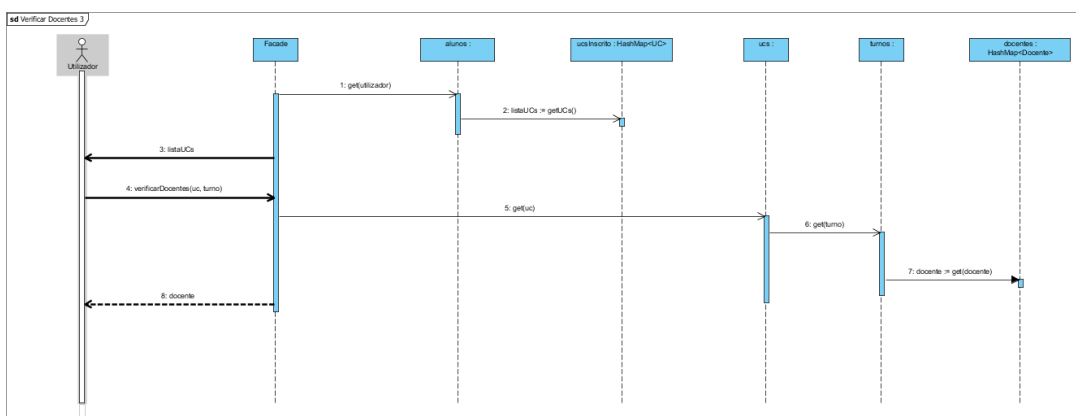


Figura 52.

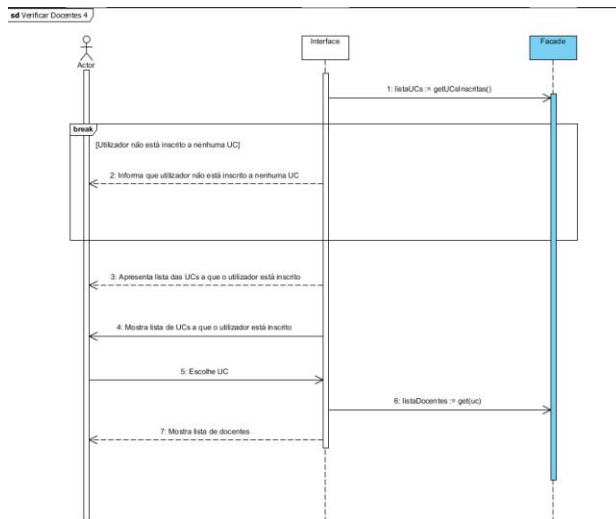


Figura 53.

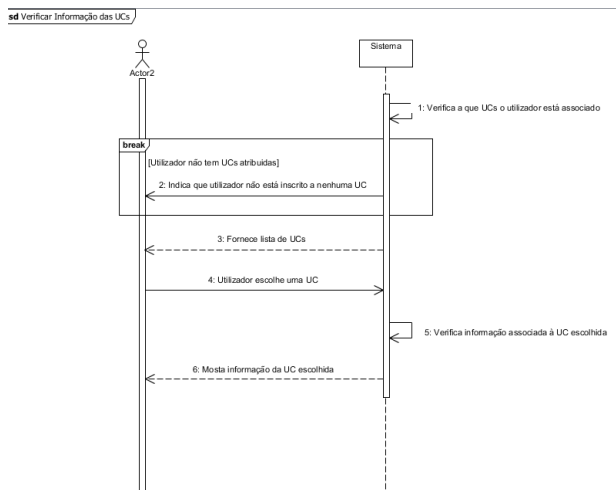


Figura 54.

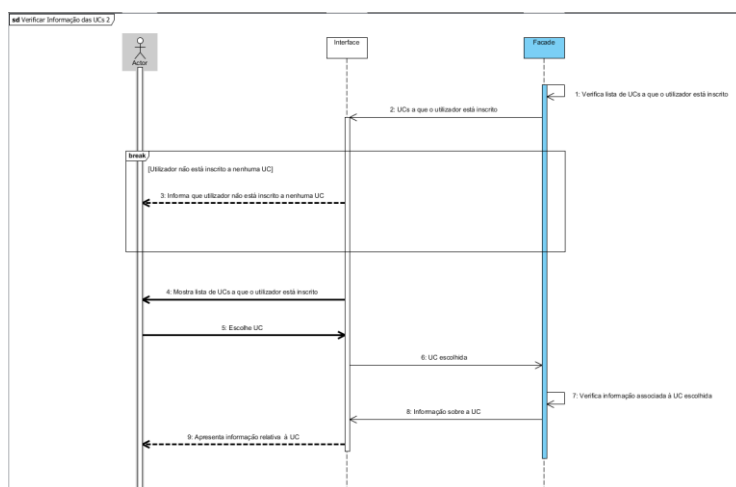


Figura 55.

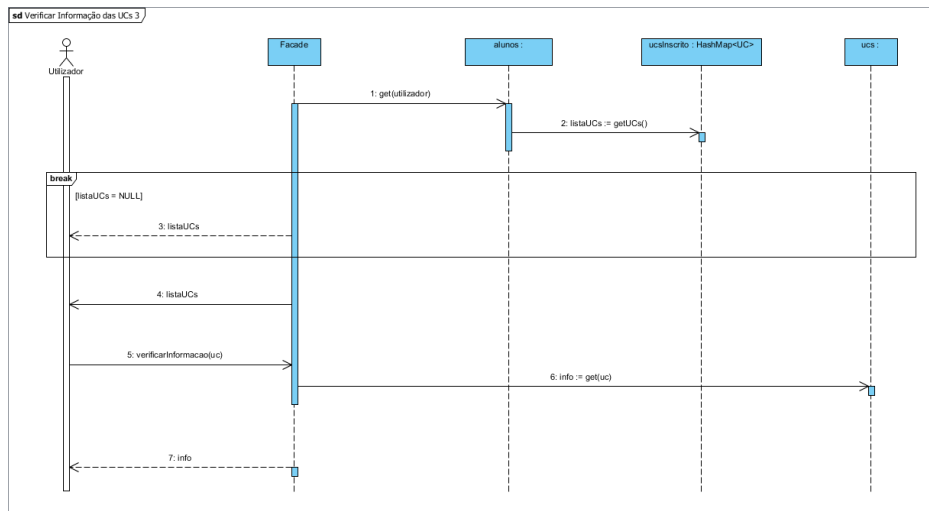


Figura 56.

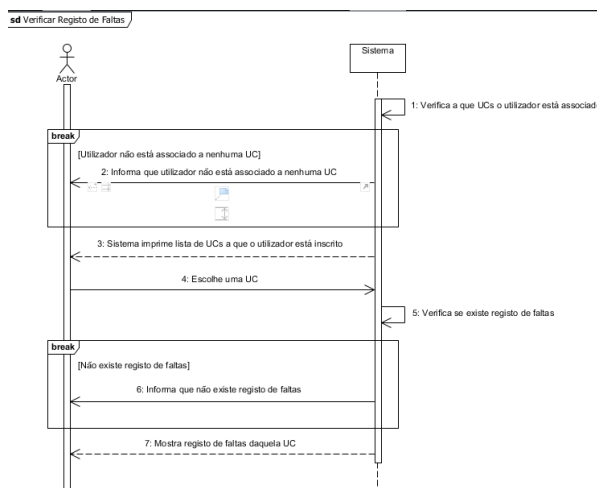


Figura 57.

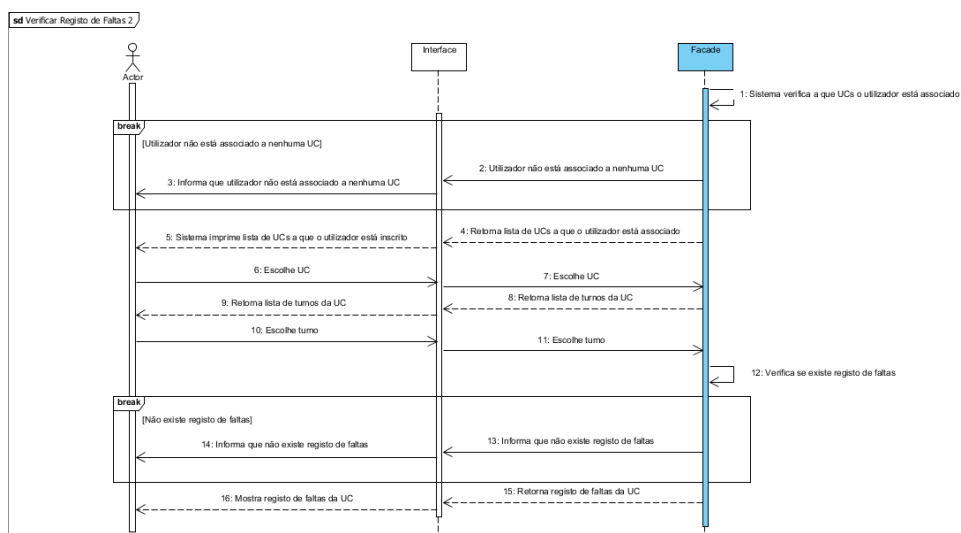


Figura 58.



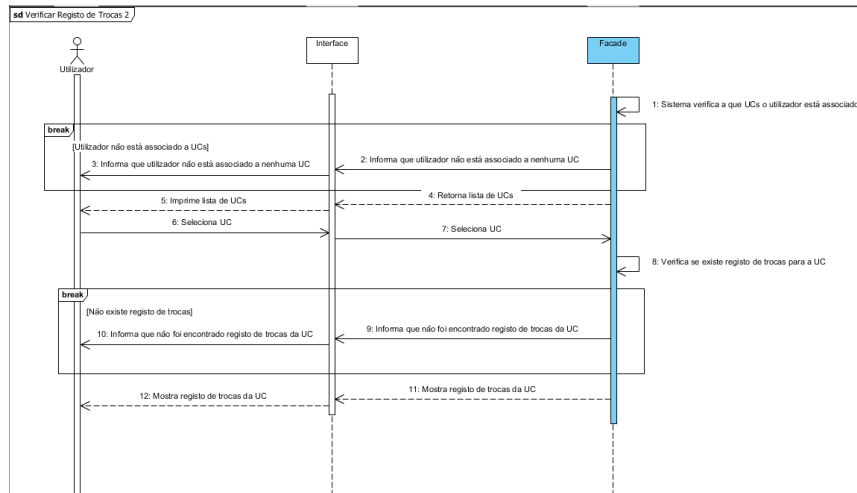


Figura 62.

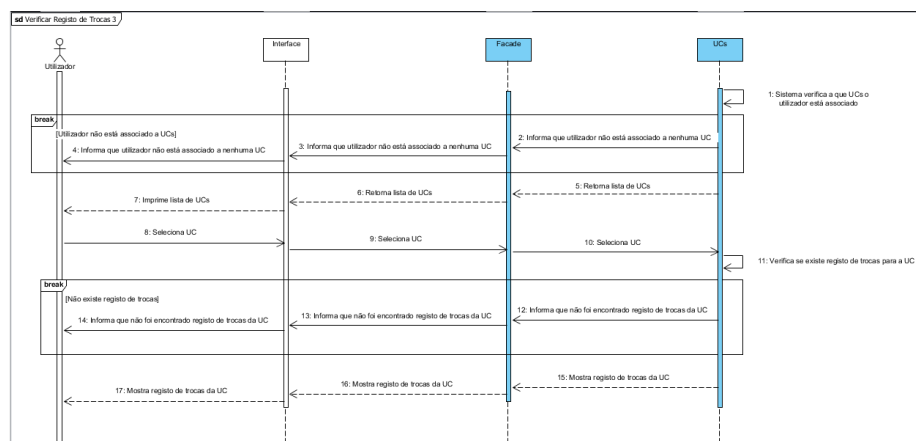


Figura 63.

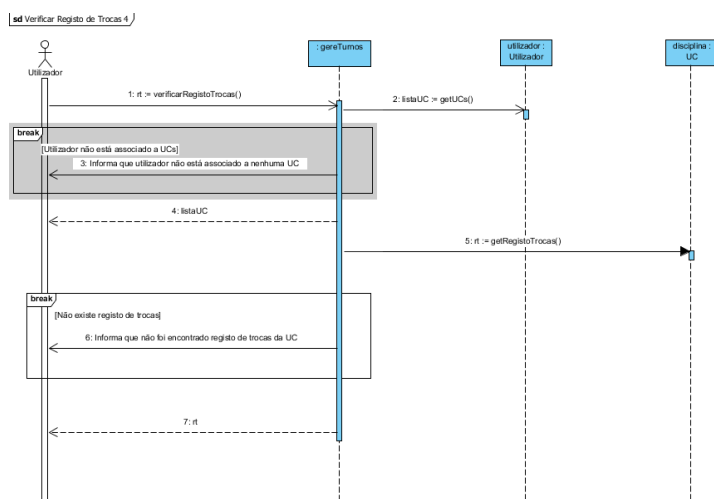


Figura 64.