Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática
Programação Centrada na Rede

Projecto da Disciplina - Fases de Desenho e Implementação

SISTEMA DE GESTÃO DE MOMENTOS DE AVALIAÇÃO

Ano lectivo: 2012/2013 / 2º Semestre / 3º ano

Aluno: Pedro Moreira N.10015
Docente: Luís Bruno
26 de Setembro de 2013

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Objectivos

Este relatório insere-se na avaliação da disciplina "Programação Centrada na Rede", leccionada no 2º semestre do 3º ano da licenciatura em Engenharia Informática.
O trabalho surge na sequência do projecto interdisciplinar de Engenharia de Software, Bases de Dados I e II, Hipermédia e Acessibilidade e, por último, Programação Centrada na Rede.
O tema é o desenvolvimento de uma aplicação que permita gerir os momentos de avaliação das diferentes disciplinas dos cursos da Estig. Nele pretende-se apresentar um resumo da análise, fase de desenho e implementação da aplicação.

1.2 Problema

Actualmente, cada docente marca as suas diferentes avaliações de forma individual, podendo ou não enviar essa informação ao Coordenador de Curso. Este, para ter noção da distribuição de carga dos momentos de avaliação ao longo de um semestre tem que, manualmente, registar essa informação, que poderá ser alterada posteriomente, sem que ele receba qualquer notificação. Por outro lado não existe um repositório que centralize todos os momentos de avaliação realizados nas disciplinas ao longo do semestre e que seja transparente para os seus diferentes beneficiários.

1.3 Solução

Neste projecto pretende-se a criação de uma aplicação na linguagem PHP, com suporte de mySQL, que centralize o registo de todos os momentos de avaliação mantendo os respectivos intervenientes actualizados acerca das alterações às mesmas.
Por forma a evitar uma carga excessiva aos alunos, as avaliações terão obrigatóriariamente que ser analisadas pelo coordenador do curso em causa.
O presente projecto contemplará a implementação de cinco casos de uso, um específico para cada um dos três actores e dois que serão comuns a todos eles.

Capítulo 2

Descrição Geral

2.1 Perspectiva

Este trabalho irá resultar numa aplicação que possibilite aos intervenientes poderem efectuar operações sobre os momentos de avaliação da ESTIG num repositório comum que mantenha os respectivos intervenientes informados das eventuais alterações aos registos associados ao seu perfil.

2.1.1 Interfaces

Sistema

O sistema deverá funcionar como repositório para todos os momentos de avaliação dos cursos da Estig.

Utilizador

A interface de utilizador deverá ser compatível com os principais browsers e respeitar as principais regras de usabilidade e acessibilidade.

Software

O projecto será implementado na linguagem PHP com ligação a base de dados mySQL. Serão utilizados os APIś jQuery e Smarty.

2.1.2 Funções

O sistema implementado irá suportar as seguintes funções:

- **Efectuar Login**
- **Listar Avaliações**
- **Validar / Cancelar Validação de Momentos de Avaliação**

- **Inscrever / Cancelar Inscrições em Momentos de Avaliação**

- **Marcar / Cancelar Marcação de Novos Momentos de Avaliação**

2.1.3 Características do Utilizador

Os utilizadores da aplicação serão alunos, docentes e coordenadores dos cursos leccionados na ESTIG. A sua utilização esporádica e em periodos de tempo espaçados faz com que em termos de usabilidade se tenha especial atenção à facilidade de aprendizagem no uso das interfaces.

Capítulo 3

Diagrama de Casos de Uso

Como é visivel na Figura 3.1, para este trabalho, foram definidos 5 casos de uso, sendo um para cada tipo de actores e dois que, apesar de terem características específicas para cada tipo de actor são comuns aos três.
Todos os tipos de utilizador podem efectuar login/logout, bem como listar as suas avaliações por datas e visualizar os detalhes das mesmas num dia escolhido.
O docente pode também marcar ou cancelar uma avaliação de uma disciplina por ele leccionada, enquanto que o aluno pode inscrever-se ou cancelar a inscrição numa avaliação de uma disciplina em que esteja matriculado. Por fim o coordenador pode validar ou cancelar a validação de uma avaliação de um curso que coordene.

3.1 EFECTUAR LOGIN

3.1.1 Descrição

Todos os utilizadores do sistema podem efectuar login ou logout do sistema.

3.1.2 Actores

User

3.1.3 Cenário Principal

Login

Figura 6.2

O Utilizador:
1. Ao aceder ao site o sistema verifica se o utilizador está logado.
2. Caso o utilizador esteja logado:
2.1 O sistema consulta os cursos associados ao utilizador, agrupados pelo papel desempenhado no mesmo
2.2 O sistema consulta as disciplinas associadas ao utilizador para cada pepel desempenhado e curso
2.3 O sistema consulta as avaliações associadas a cada disciplina do utilizador

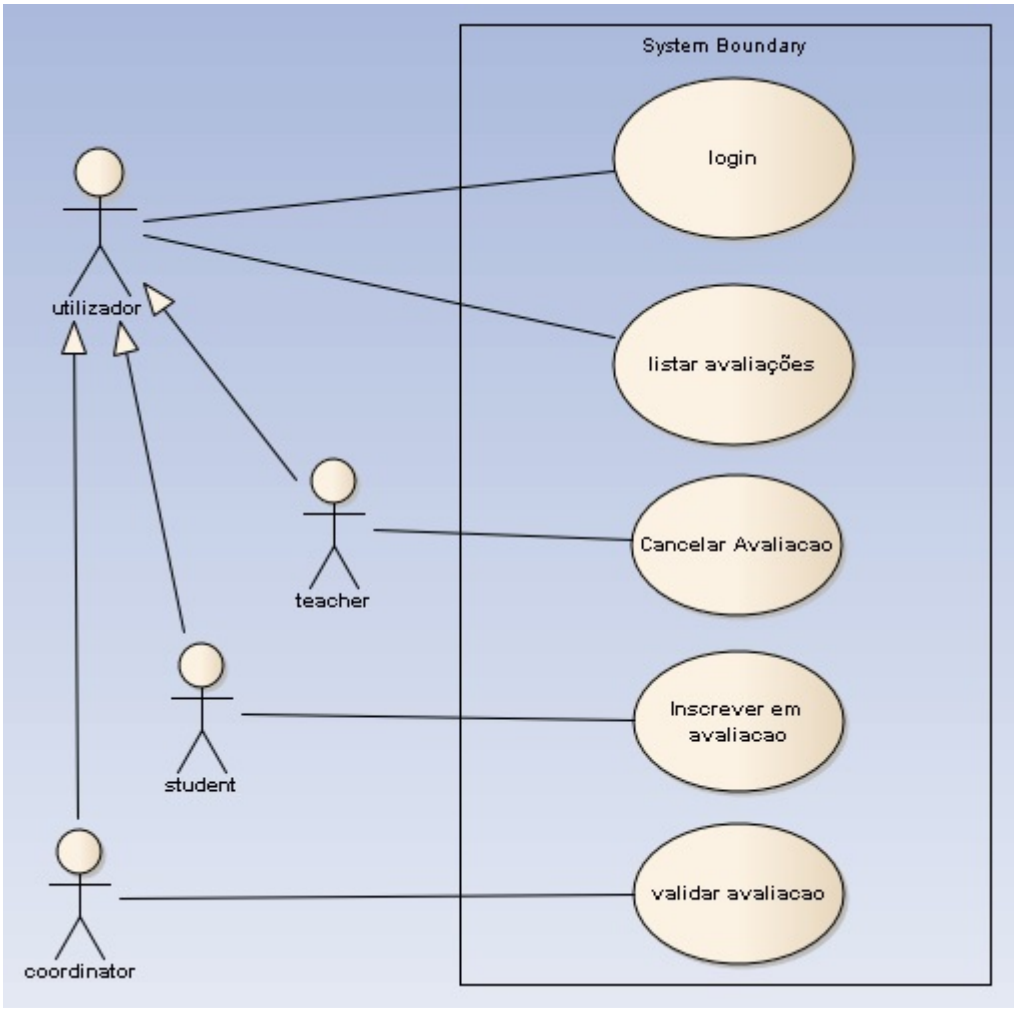

Figura 3.1: Diagrama: Casos de Uso

3.1.4 Extensões ou Variações

Logout

Figura 6.3

O Utilizador:

1. Escolhe efectuar logout.
2. O sistema limpa os dados do utilizador na sessão actual.
3. O sistema limpa os cursos associados ao utilizador como coordinator.
4. O sistema limpa os cursos associados ao utilizador como teacher.
5. O sistema limpa os cursos associados ao utilizador como student.

3.2 LISTAR AVALIAÇÕES

3.2.1 Descrição

A cada acesso ao sistema, o mesmo consulta as avaliações associadas ao utilizador para semre apresentadas no calendário no intervalo de datas definido, bem como os detalhes das avaliações do dia seleccionado.

3.2.2 Pré-condições

O utilizador deverá estar logado e apenas terá acesso a avaliações a ele associadas como student, teacher ou coordinator.

3.2.3 Actores

User

3.2.4 Cenário Principal

Listar avaliações entre datas

Figura 6.4 O user:

1. Acede ao sistema, já logado.
2. O sistema obtem a lista de cursos a ele associados, passando pelos mesmo passos já descritos no login.
3. O sistema obtem o intervalo de datas definido.
4. O sistema obtem a lista de avaliações definidas no intervalo de datas.
5. É apresentado ao utilizador a lista de avaliações no calendário.

3.2.5 Extensões ou Variações

Listar detalhes de avaliações

Figura 6.5

O user:

1. Ao escolher um dia no calendário.
2. O sistema verifica o dia escolhido.
3. O sistema obtem a lista de avaliações associadas ao user para esse dia.
4. O sistema apresenta ao utilizador os detalhes das avaliações.

3.3 VALIDAR AVALIAÇÕES

3.3.1 Descrição

O Coordenador de Curso pode validar ou cancelar a validação das avaliações das disciplinas do curso a ele associado.

3.3.2 Pré-condições

O coordenador deverá estar logado e apenas terá acesso a avaliações de disciplinas do seu curso.

3.3.3 Actores

Coordenador

3.3.4 Cenário Principal

Validar Avaliação

Figura 6.10 O coordenador:

1. Ao visualizar a avaliação, escolhe validar.
2. O sistema valida a valiação.
3. O sistema actualiza o calendario
4. O sistema apresenta o calendario

3.3.5 Extensões ou Variações

Cancelar Validação

Figura 6.11

O Coordenador:

1. Ao visualizar uma disciplina, escolhe cancelar a sua validação.
2. O sistema cancela a validação
3. O sistema actualiza o calendário
4. O sistema apresenta o calendário

3.4 MARCAR AVALIAÇÃO

3.4.1 Descrição

O docente pode criar, alterar e cancelar momentos de avaliação para as disciplinas em que seja docente.

3.4.2 Pré-condições

Apenas é possível criar momentos de avaliação para as disciplinas em que o docente esteja associado.

3.4.3 Actores

Docente

3.4.4 Cenário Principal

Figura 6.8

Novo Momento de Avaliação

O Docente:

1. Escolhe criar um novo momento de avaliação.
2. Caso tenha já preenchido o formulário:

2.1 O sistema cria a nova avaliação

2.2 O sistema actualiza o calendário

2.3 O sistema apresenta o calendário

3. Caso não tenha preenchido o formulário:

3.1 O sistema obtem a lista de disciplinas leccionadas pelo docente

3.2 O sistema apresenta o formulário para que o docente crie a nova avaliação para uma das disciplinas por ele leccionadas

3.4.5 Extensões ou Variações

Figura 6.9

Cancelar Momento de Avaliação

O Docente:

1. Escolhe o dia em que está marcada a Avaliação.
2. O sistema apresenta os detalhes da avaliação.
3. O Docente escolhe cancelar avaliação.
4. O sistema cancela a avaliação.
5. O sistema actualiza o calendário
6. O sistema apresenta o calendário actualizado

3.5 INSCREVER EM AVALIAÇÃO

3.5.1 Descrição

O Aluno pode inscrever-se ou cancelar a inscriçõa nas avaliações das disciplinas a ele associado.

3.5.2 Pré-condições

O aluno deverá estar logado e apenas terá acesso a avaliações das disciplinas onde está matriculado.

3.5.3 Actores

Aluno

3.5.4 Cenário Principal

Inscrever em Avaliação

Figura 6.6 O aluno:

1. Ao visualizar a avaliação, escolhe inscrever-se
2. O sistema inscreve o aluno

3. O sistema actualiza o calendario
4. O sistema apresenta o calendario

3.5.5 Extensões ou Variações

Cancelar Inscrição

Figura 6.7

O aluno:

1. Ao visualizar uma disciplina, escolhe cancelar a sua inscrição
2. O sistema cancela a inscrição
3. O sistema actualiza o calendário
4. O sistema apresenta o calendário

Capítulo 4

Modelação de Interfaces

4.1 Efectuar Login

Figura 4.1: Interface: Login

4.2 Inscrever em Momento de Avaliação

Figura 4.2: Interface: Efectuar ou Cancelar Inscrição em Momento de Avaliação

4.3 Marcar e Cancelar Momento de Avaliação

Figura 4.3: Interface: Marcar Novo Momento de Avaliação

4.4 Validar e Cancelar Validação de Momentos de Avaliação

Figura 4.4: Interface: Validar ou Cancelar Validação de Momento de Avaliação

Capítulo 5

Modelação da Base de Dados

Resumidamente a base de dados está representada da seguinte forma:
TABELAS

- **curso** - Representa os cursos leccionados
- **ano_lectivo** - Representa um ano lectivo
- **utilizador** - todos os utilizadores ficam registados nesta tabela
- **curso_ano** - Representa um curso leccionado num determinado ano, nesta tabela fica representado o coordenador do curso nesse ano associando o id do respectivo user na tabela utilizador
- **semestre** - Representa um semestre de um determinado ano lectivo
- **disciplina** - Representa uma disciplina
- **disciplina_semestre** - Faz referencia a uma disciplina leccionada num determinado semestre
- **Docente** - Representa um ou mais utilizadores designados como docentes de uma determinada disciplina
- **aluno** - Representa uma matrícula de um utilizador como aluno num determinado curso
- **matricula_disciplina** - Representa a matricula numa disciplina de um determinado aluno
- **avaliacao** - Representa uma avaliacao marcada para uma disciplina leccionada num determinado semestre
- **avaliacao_datas_alt** - Nesta tabela ficam registadas as datas alternati vas escolhidas pelo docente para que o coordenador possa ter a possibilidade de trocar caso aconteça um peso demasiado excessivo nas avaliações para os alunos.
- **avaliacao_aluno** - Representa a inscrição de um aluno numa avaliação e é onde fica registada a sua nota

STORED PROCEDURES

- **get_coordinator_evaluations(user_num, num_discipline)** - Obter lista de avaliações com o perfil de coordenador de um determinado utilizador

```
DROP PROCEDURE IF EXISTS `get_coordinator_evaluations`$$
CREATE PROCEDURE `get_coordinator_evaluations`(IN user_num INT, num_discipline
    INT)
begin
SELECT
  av.num_avaliacao as id, av.tipo_avaliacao as type,
  av.peso as weight, av.data_avaliacao as date, av.activada as validated
  FROM avaliacao as av

  RIGHT JOIN (
    SELECT a.num_avaliacao
    FROM curso_ano as ca
    LEFT JOIN disciplina d
      ON d.num_curso = ca.num_curso
    RIGHT JOIN avaliacao a
      ON a.num_disciplina = d.num_disciplina
    WHERE ca.num_coordenador = user_num
  ) AS x
  ON  x.num_avaliacao = av.num_avaliacao

  WHERE av.num_disciplina = num_discipline;
end$$
```

- **get_teacher_evaluations(user_num, num_discipline)** - Obter lista de avaliações com o perfil de docente de um determinado utilizador

```
DROP PROCEDURE IF EXISTS `get_teacher_evaluations`$$
CREATE PROCEDURE `get_teacher_evaluations`(IN user_num INT, num_discipline INT)
begin
SELECT
  av.num_avaliacao as id, av.tipo_avaliacao as type,
  av.peso as weight, av.data_avaliacao as date, av.activada as validated
  FROM avaliacao as av
  RIGHT JOIN (
    SELECT a.num_avaliacao
    FROM docente d
    LEFT JOIN avaliacao a
      ON a.num_disciplina = d.num_disciplina
      AND a.num_semestre = d.num_semestre
    WHERE d.num_utilizador = user_num
    AND a.num_disciplina = num_discipline
  ) AS x
  ON  x.num_avaliacao = av.num_avaliacao
  WHERE av.num_disciplina = num_discipline;
end$$
```

- **get_student_evaluations(user_num, num_discipline)** - Obter lista de avaliações com

o perfil de aluno de um determinado utilizador

```
DROP PROCEDURE IF EXISTS `get_student_evaluations`$$
CREATE PROCEDURE `get_student_evaluations`(IN user_num INT, num_discipline INT)
begin
SELECT
  av.num_avaliacao as id, av.tipo_avaliacao as type,
  av.peso as weight, av.data_avaliacao as date, x.validated
  FROM avaliacao as av
  LEFT JOIN (
    SELECT a.num_avaliacao, 1 as validated
    FROM avaliacao_aluno a
    WHERE a.num_utilizador = user_num
    AND a.num_disciplina = num_discipline
  ) AS x
  ON  x.num_avaliacao = av.num_avaliacao
  WHERE av.num_disciplina = num_discipline
  AND av.activada = 1;
end$$
```

- **get_cursos_user(user_num)** - Obter lista de cursos a que utilizador esteja associado

```
DROP PROCEDURE IF EXISTS `get_cursos_user`$$
CREATE PROCEDURE `get_cursos_user`(IN user_num INT)
begin
SELECT c.num_curso as id, c.nome_curso as name, a.titulo_ano as year, x.role
  FROM curso_ano as ca
  RIGHT JOIN (
    SELECT a.num_curso, 'student' as role    FROM aluno a
    WHERE a.num_utilizador = user_num
    UNION (
      SELECT d.num_curso, 'teacher' as role
      FROM docente dc
        LEFT JOIN disciplina d
        ON d.num_disciplina = dc.num_disciplina
        WHERE dc.num_utilizador = user_num
    )
    UNION (
      SELECT c.num_curso, 'coordinator' as role
      FROM curso_ano as c
      WHERE c.num_coordenador = user_num
    )
      ) as x
    ON x.num_curso = ca.num_curso
      LEFT JOIN ano_lectivo a
      on a.num_ano = ca.num_ano
LEFT JOIN curso c
ON c.num_curso = ca.num_curso;
end$$
```

- **get_coordinator_disciplines(user_num, num_course)** - Obter lista de disciplinas com o perfil de coordenador de um determinado utilizador

```
DROP PROCEDURE IF EXISTS `get_coordinator_disciplines`$$
CREATE PROCEDURE `get_coordinator_disciplines`(IN user_num INT, num_course INT)
begin
SELECT d.num_disciplina as id, d.descricao as description, d.titulo as title
  FROM disciplina d
  RIGHT JOIN curso_ano ca
  ON ca.num_curso = d.num_curso
  AND ca.num_coordenador = user_num
  WHERE d.num_curso = num_course;
end$$
```

- **get_teacher_disciplines(user_num, num_course)** - Obter lista de disciplinas com o perfil de docente de um determinado utilizador

```
DROP PROCEDURE IF EXISTS `get_teacher_disciplines`$$
CREATE PROCEDURE `get_teacher_disciplines`(IN user_num INT, num_course INT)
begin
SELECT d.num_disciplina as id, d.descricao as description, d.titulo as title
  FROM disciplina d
  RIGHT JOIN docente dc
  ON dc.num_disciplina = d.num_disciplina
  AND dc.num_utilizador = user_num
  WHERE d.num_curso = num_course;
end$$
```

- **get_student_disciplines(user_num, num_course)** - Obter lista de disciplinas com o perfil de aluno de um determinado utilizador

```
DROP PROCEDURE IF EXISTS `get_student_disciplines`$$
CREATE PROCEDURE `get_student_disciplines`(IN user_num INT, num_course INT)
begin
SELECT d.num_disciplina as id, d.descricao as description, d.titulo as title
  FROM disciplina d
  RIGHT JOIN matricula_disciplina md
  ON md.num_disciplina = d.num_disciplina
  AND md.num_utilizador = user_num
  WHERE d.num_curso = num_course;
end$$
```

- **add_evaluation(user_num, discipline_id, date, weight, classroom, type, observations)** - Adicionar uma avaliação a uma determinada disciplina

```
DROP PROCEDURE IF EXISTS `add_evaluation`$$
CREATE PROCEDURE `add_evaluation`(IN user_num INT(11), discipline_id INT(11),
    date DATE, weight INT(11), classroom VARCHAR(20), type VARCHAR(100),
    observations VARCHAR(1000))
begin
```

```
if (SELECT 1 FROM docente
  where num_utilizador = user_num
  AND num_disciplina = discipline_id) = 1 THEN
begin
INSERT INTO avaliacao (num_disciplina, num_semestre, data_avaliacao, peso, sala,
    tipo_avaliacao, observacoes, activada) values (discipline_id, 1, date,
    weight, classroom, type, observations, 0);
end;
end if;
end$$
```

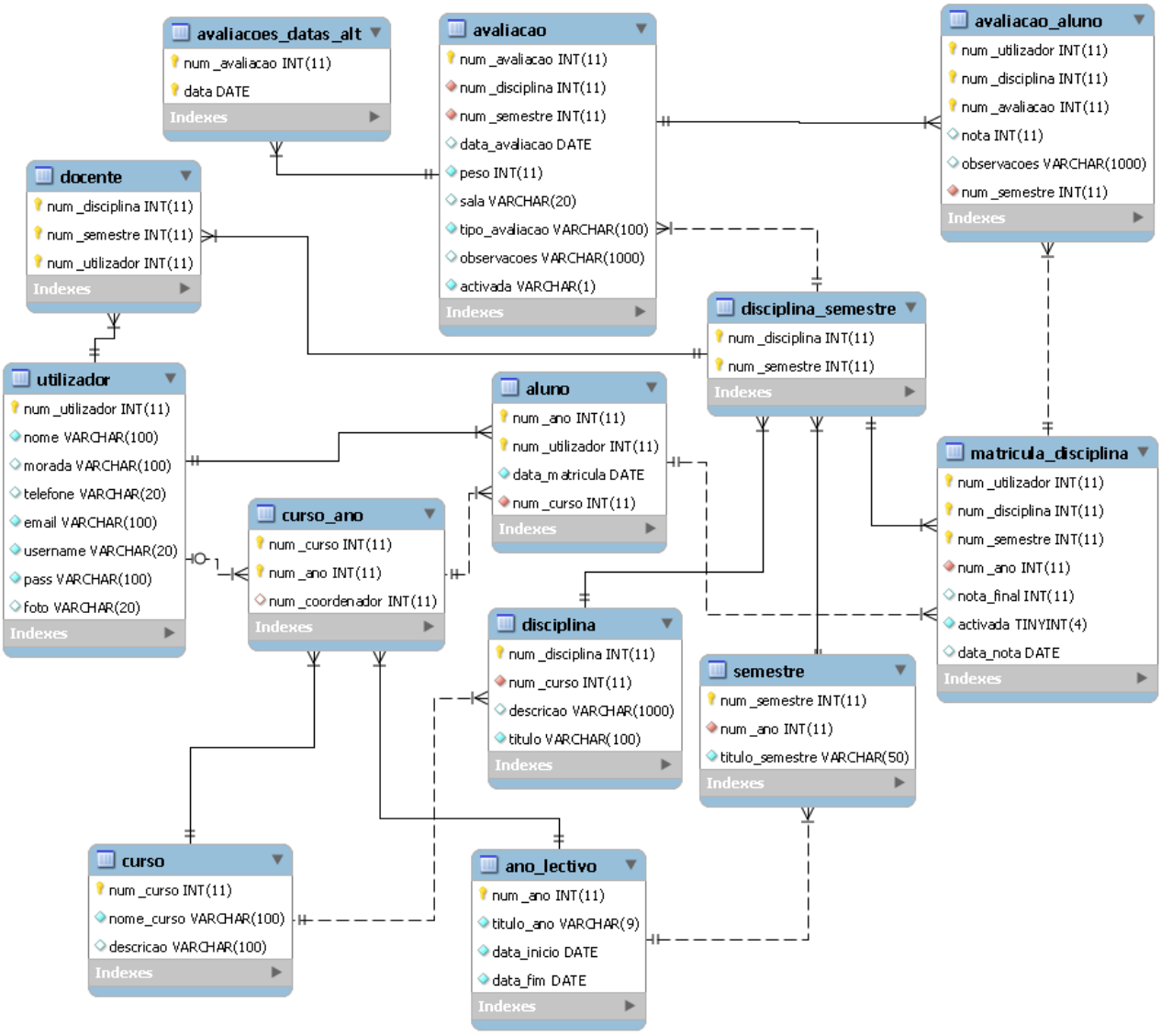

Figura 5.1: Base de Dados: Modelo Físico

Capítulo 6

Modelação UML

6.1 Diagrama de Classes

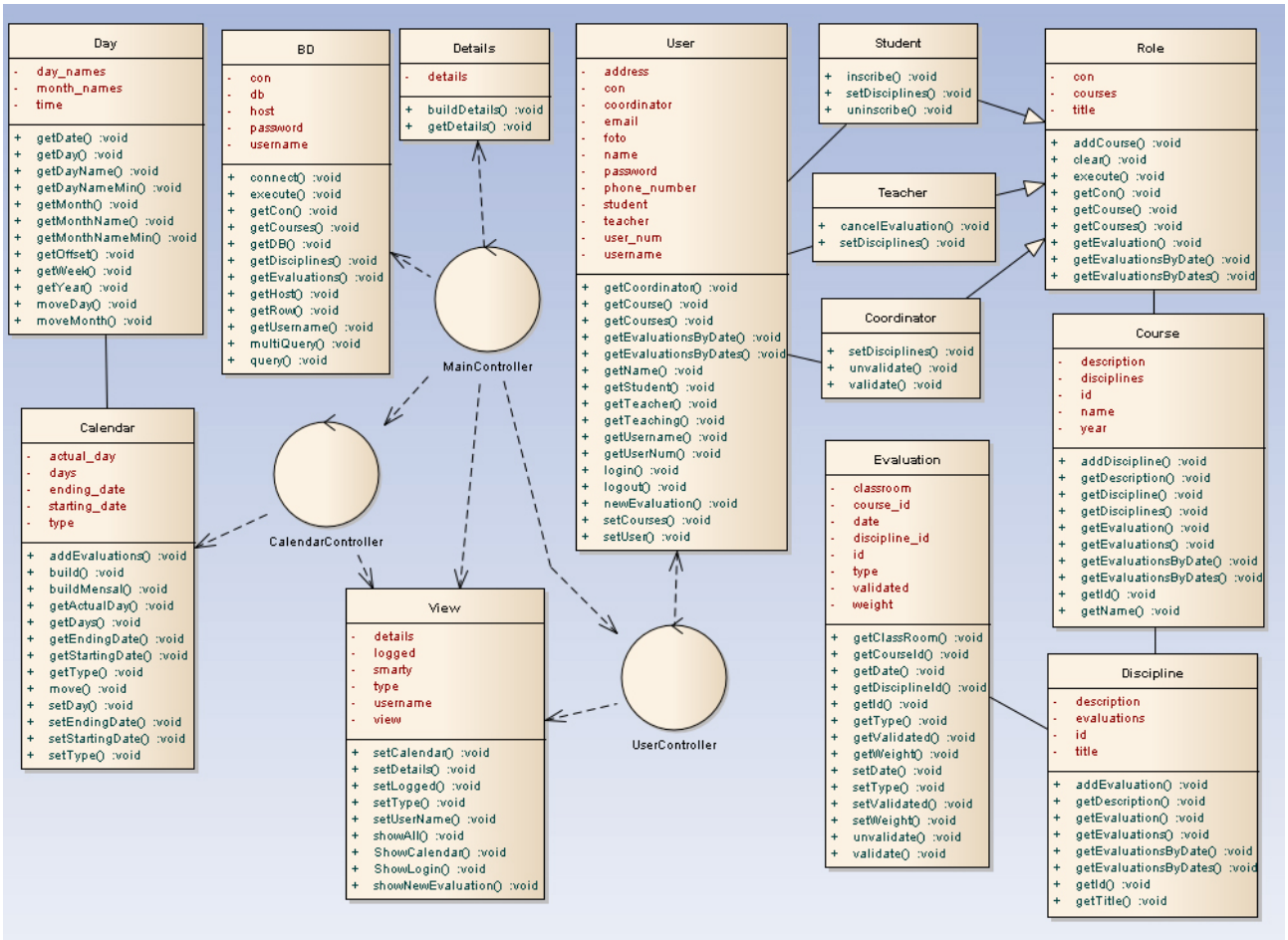

Figura 6.1: UML: Diagrama de Classes

6.2 Diagramas de Sequência

6.2.1 Efectuar Login

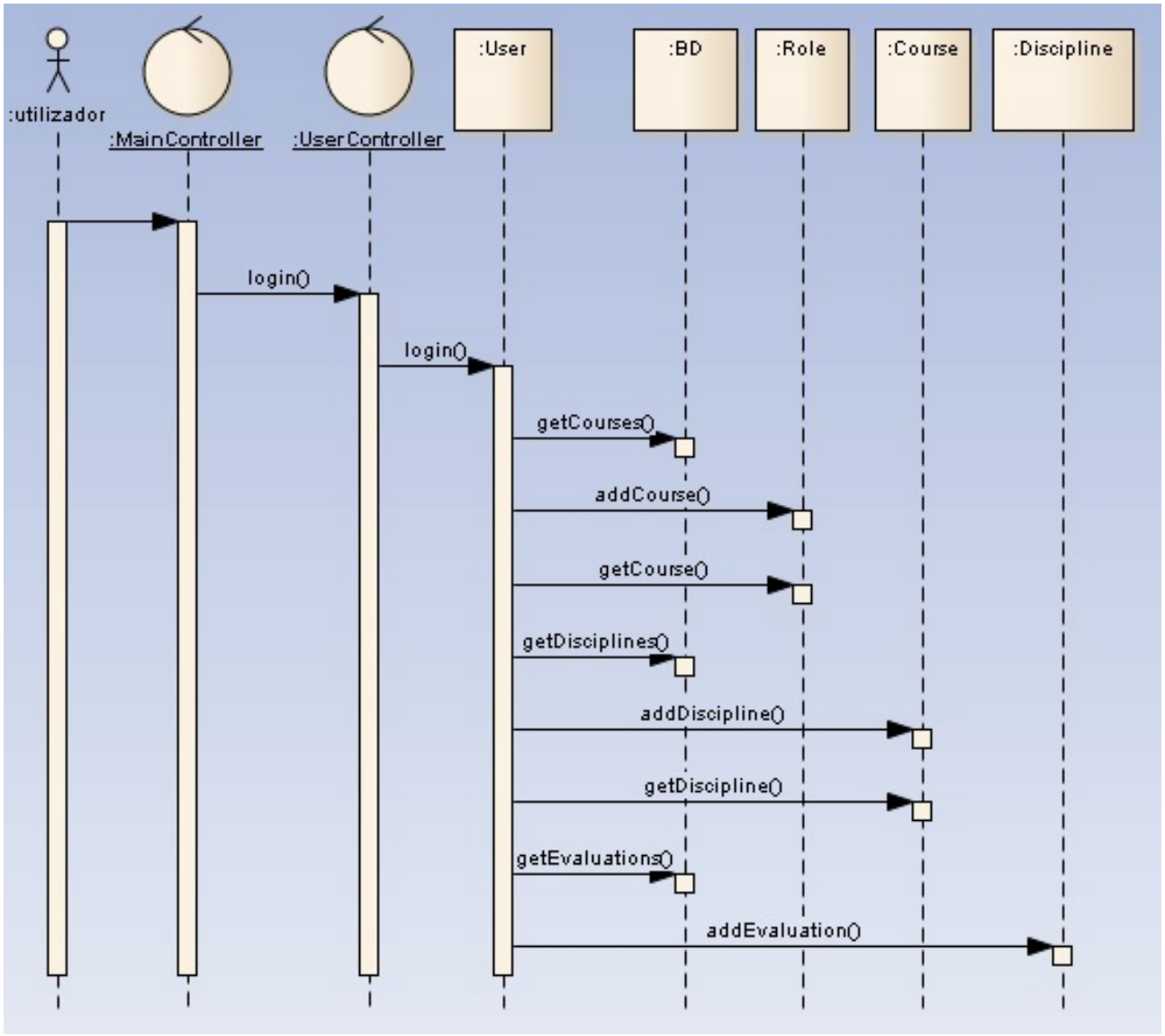

Figura 6.2: Diagrama de Sequência: Login

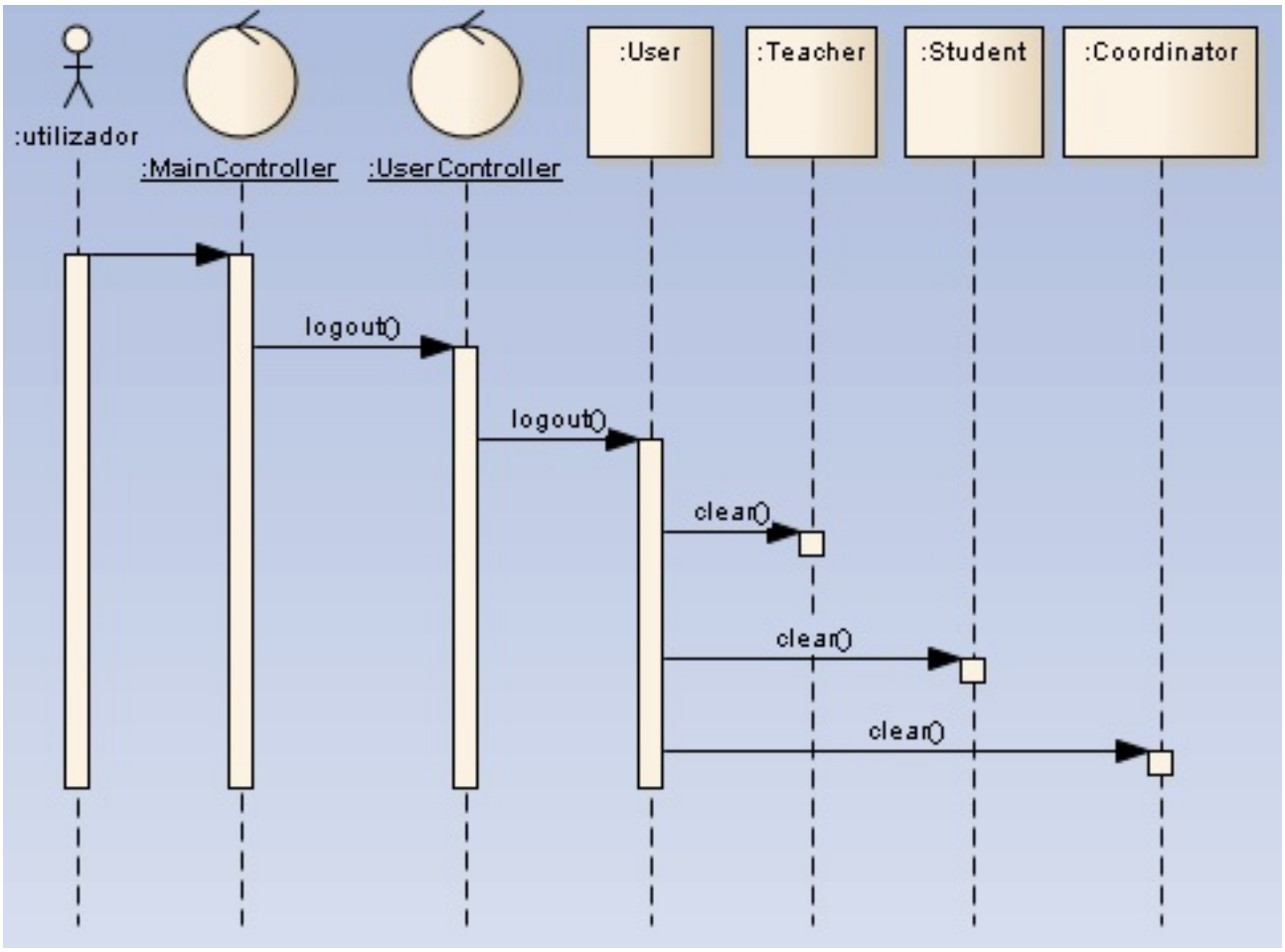

Figura 6.3: Diagrama de Sequência: Logout

6.2.2 Listar Avaliações

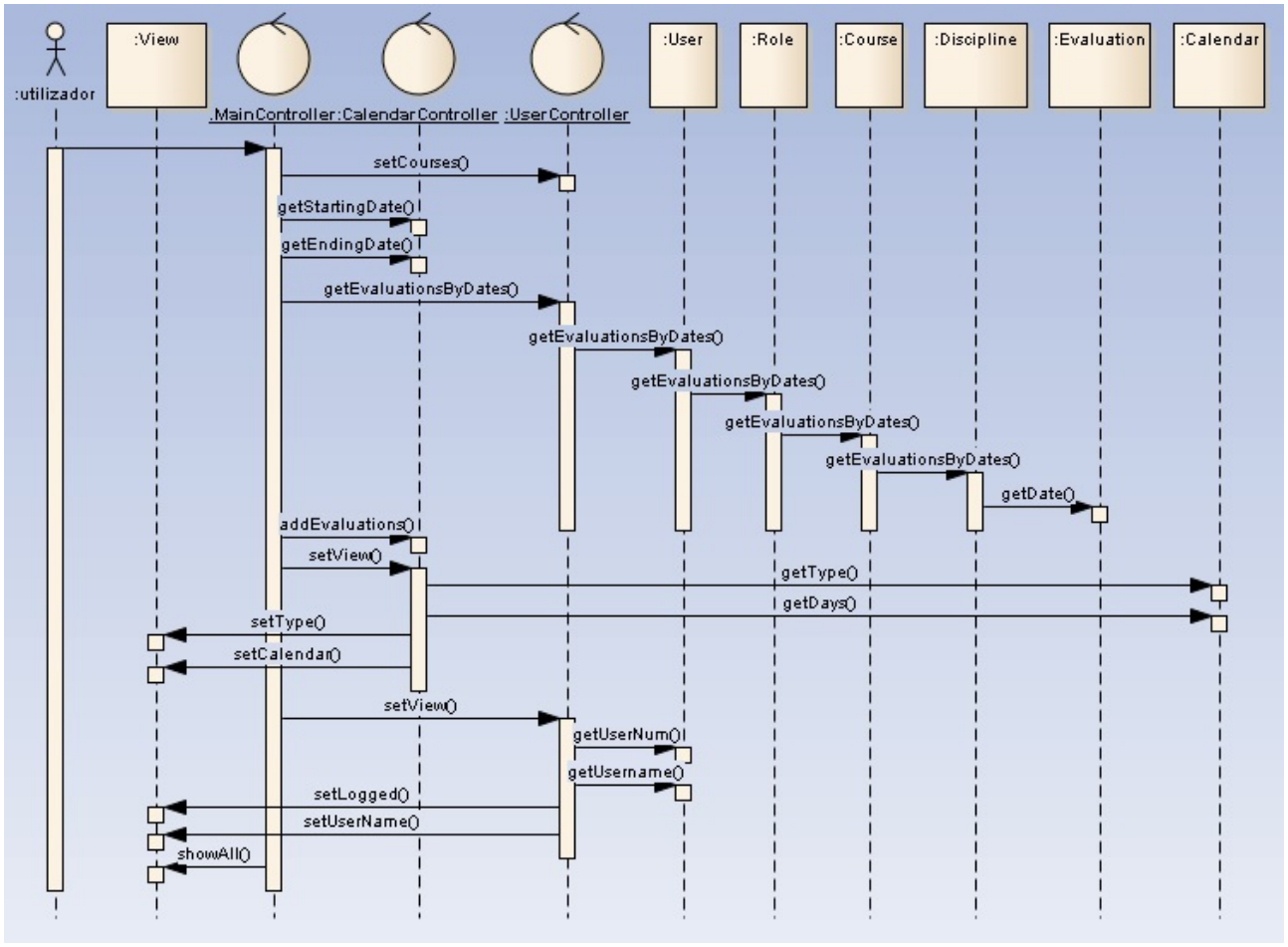

Figura 6.4: Diagrama de Sequência: Listar Avaliações Entre Datas

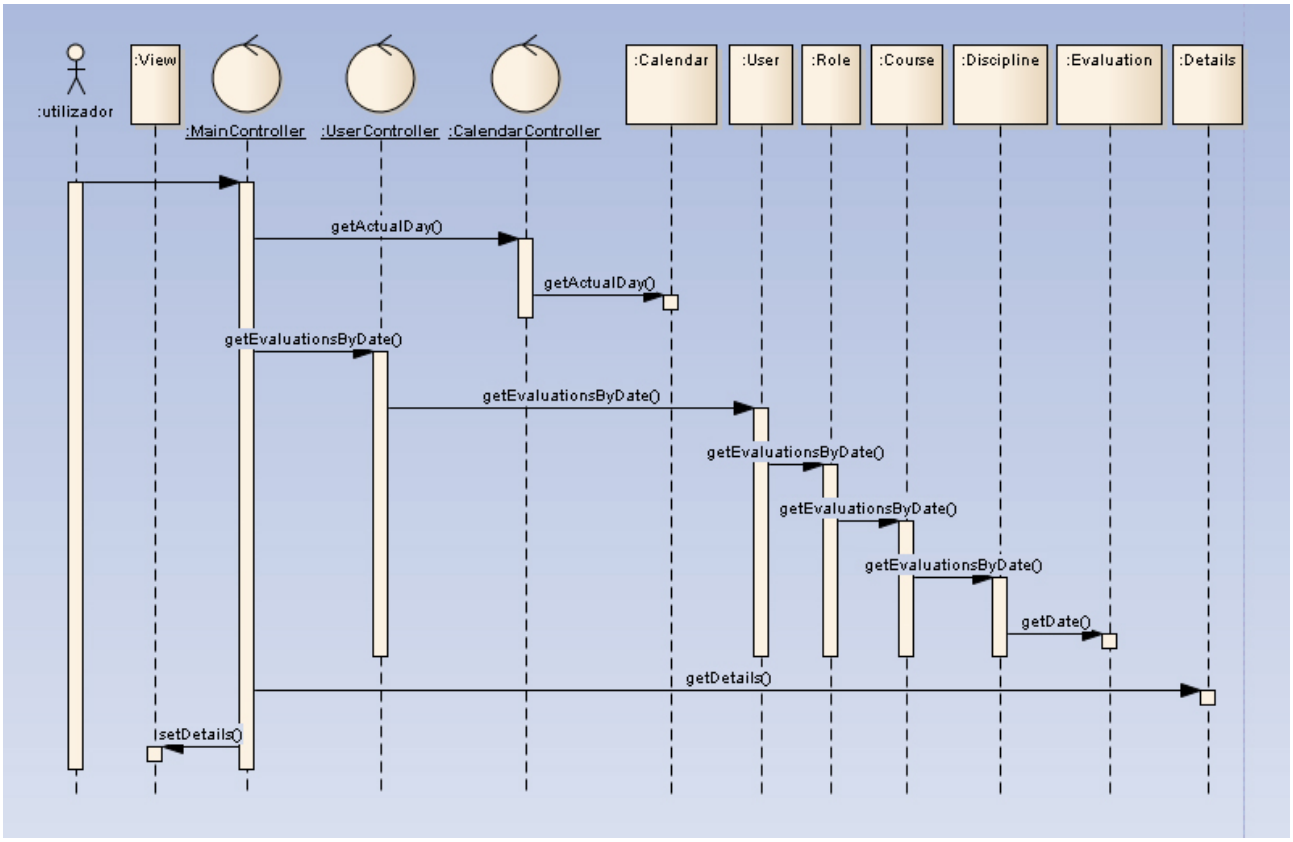

Figura 6.5: Diagrama de Sequência: Listar Detalhes de Avaliação

6.2.3 Inscrição em Avaliação

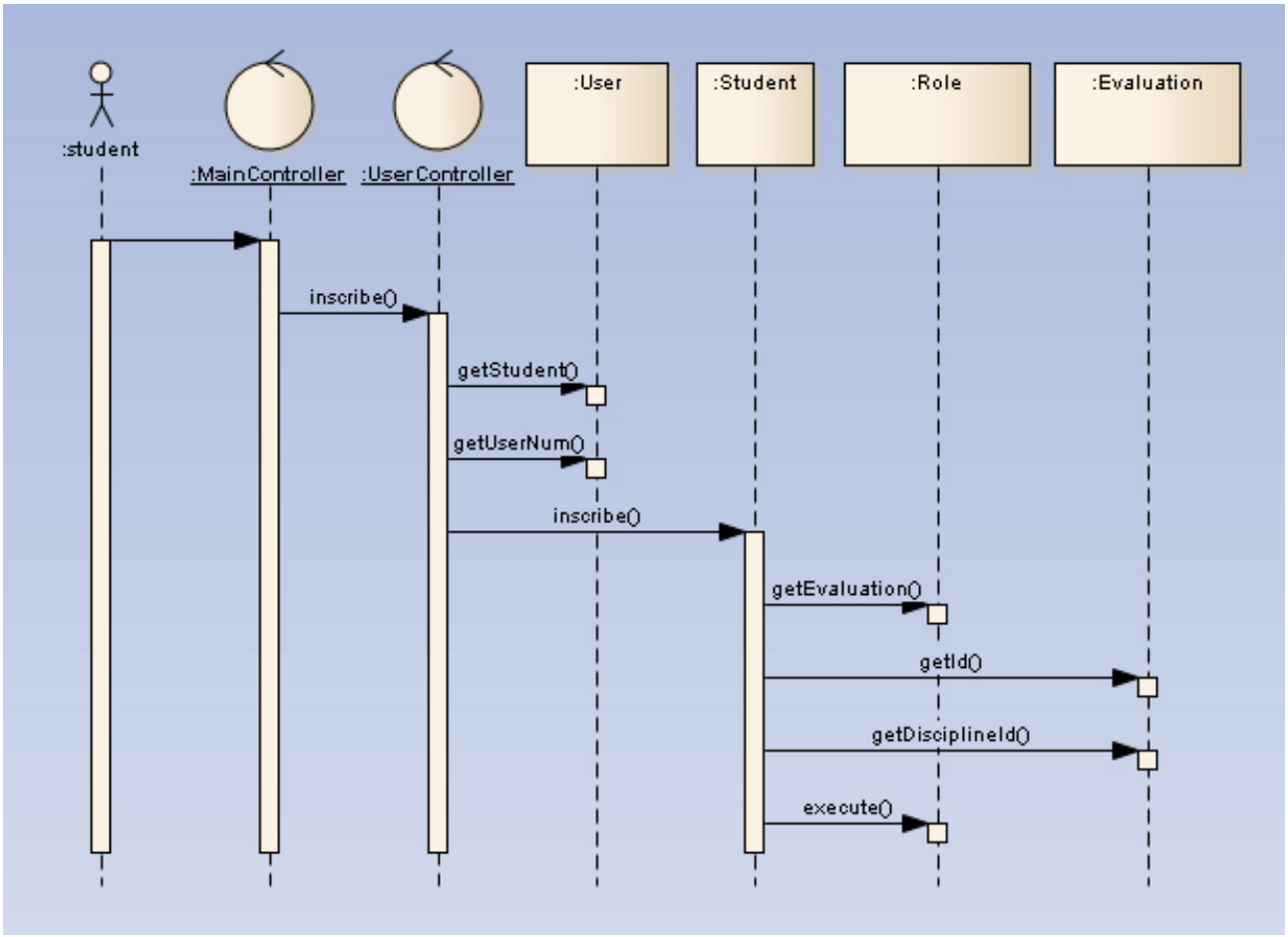

Figura 6.6: Diagrama de Sequência: Efectuar Inscrição em Momento de Avaliação

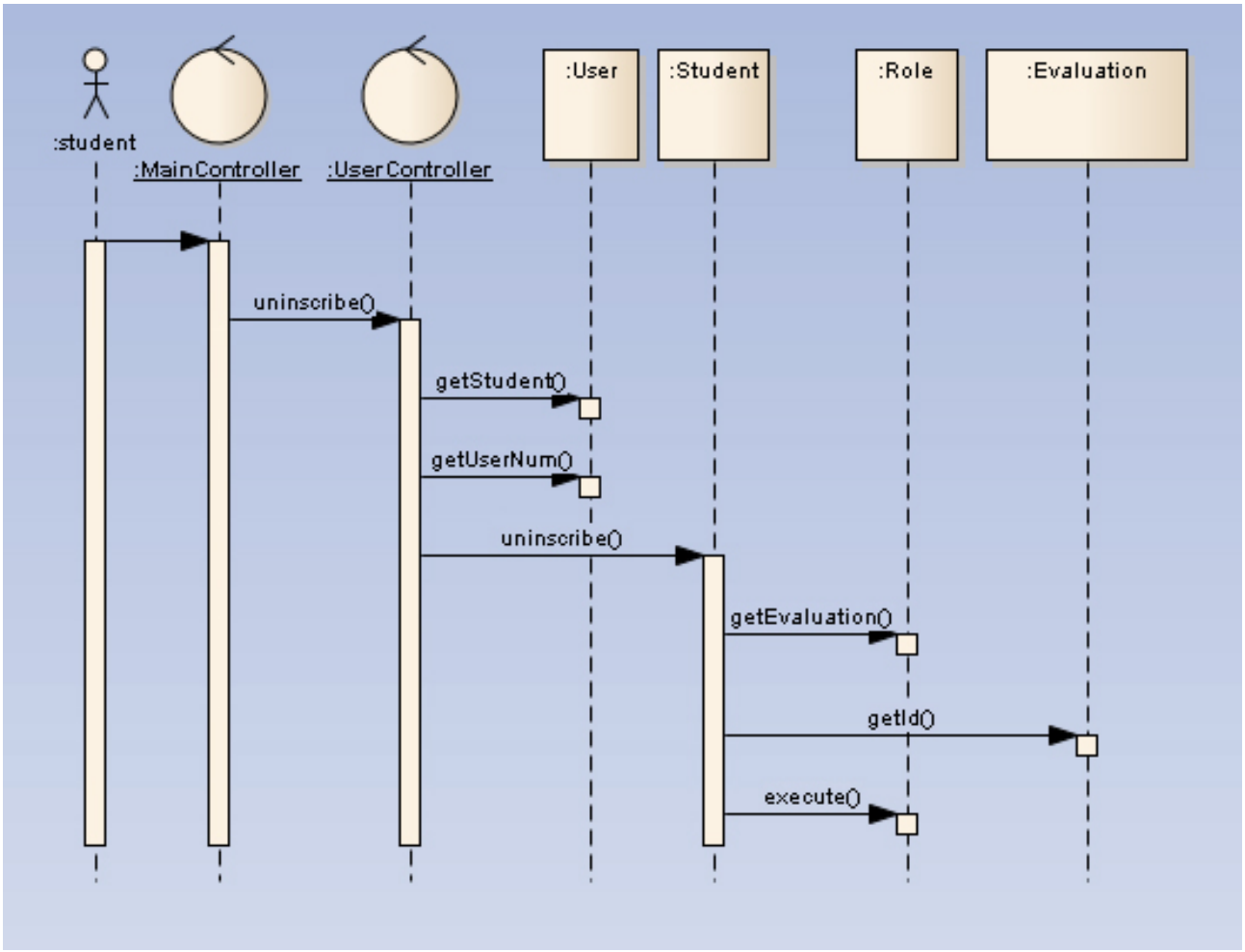

Figura 6.7: Diagrama de Sequência: Cancelar Inscrição de Momento de Avaliação

6.2.4 Marcar Avaliação

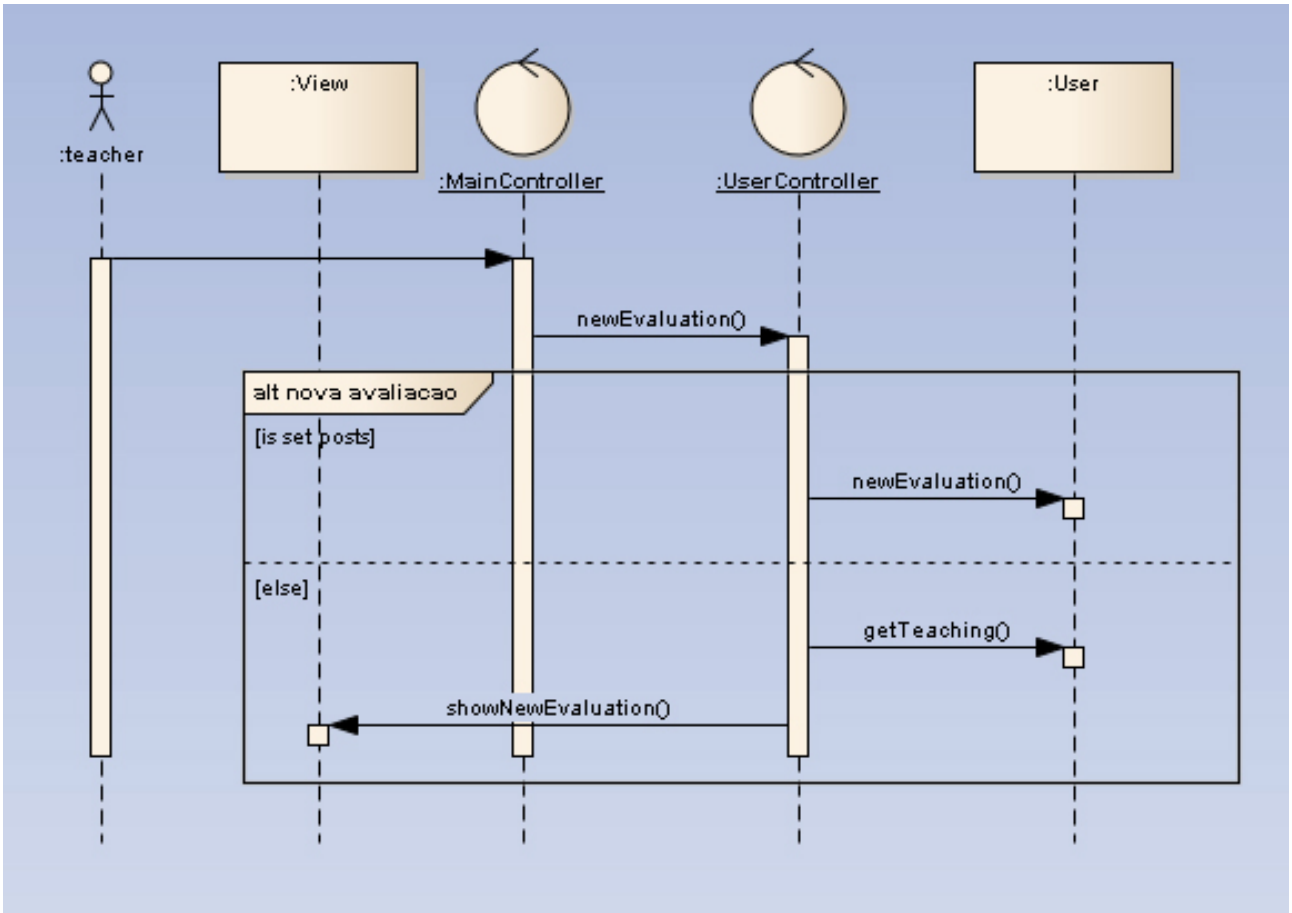

Figura 6.8: Diagrama de Sequência: Novo Momento de Avaliação

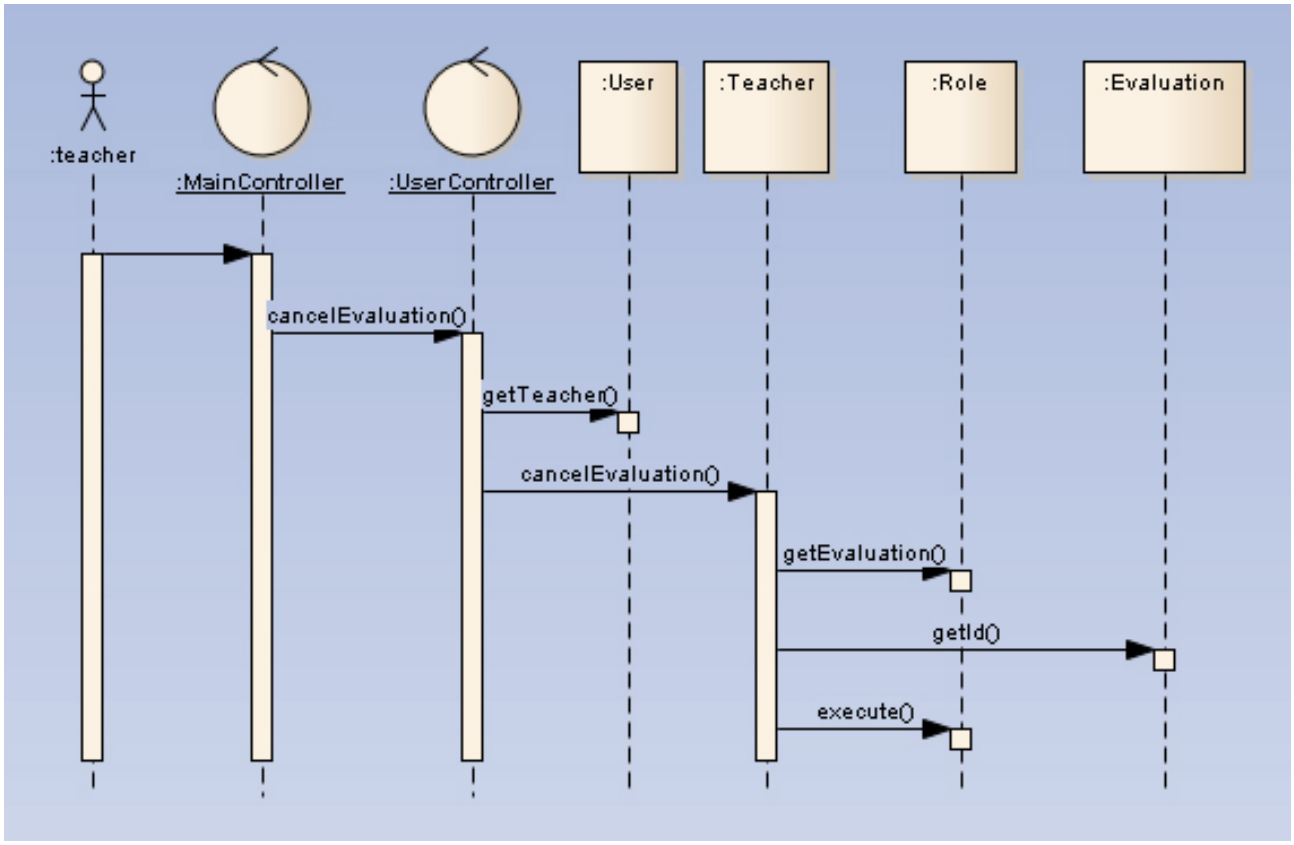

Figura 6.9: Diagrama de Sequência: Cancelar Momento de Avaliação

6.2.5 Validar Avaliações

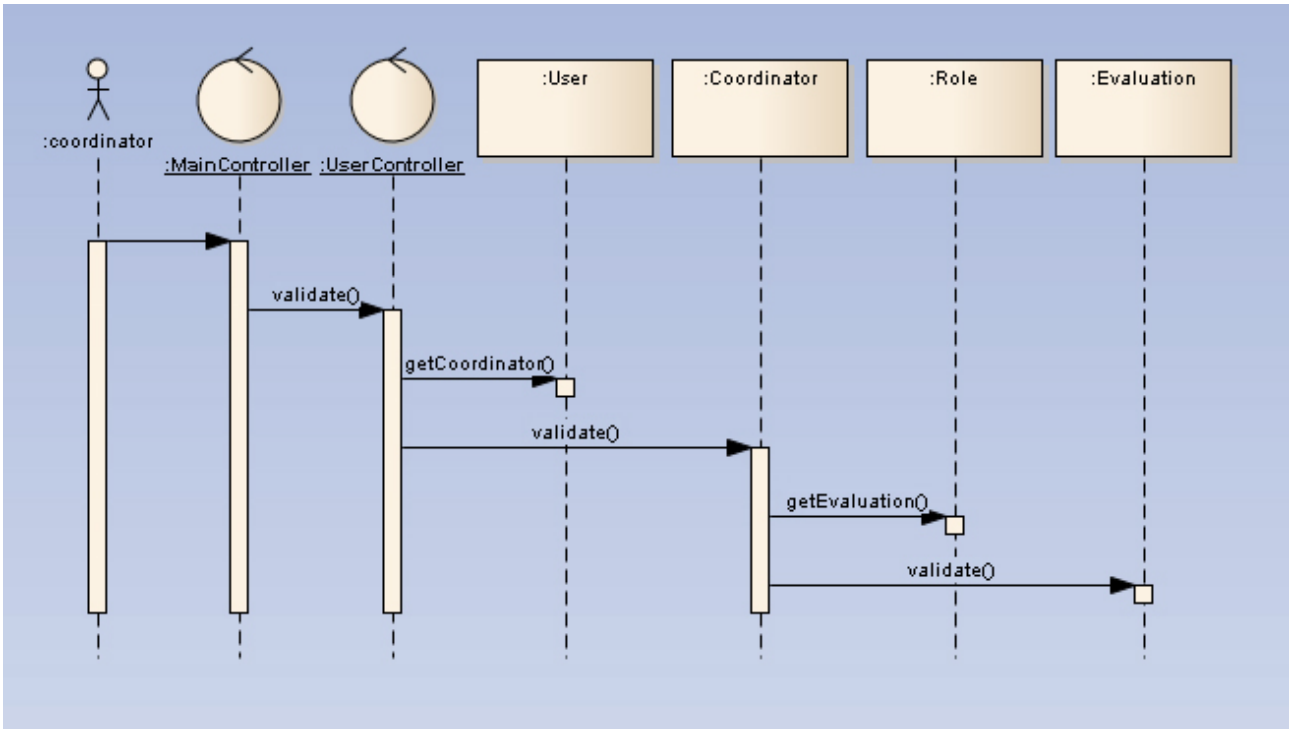

Figura 6.10: Diagrama de Sequência: Validar Avaliação

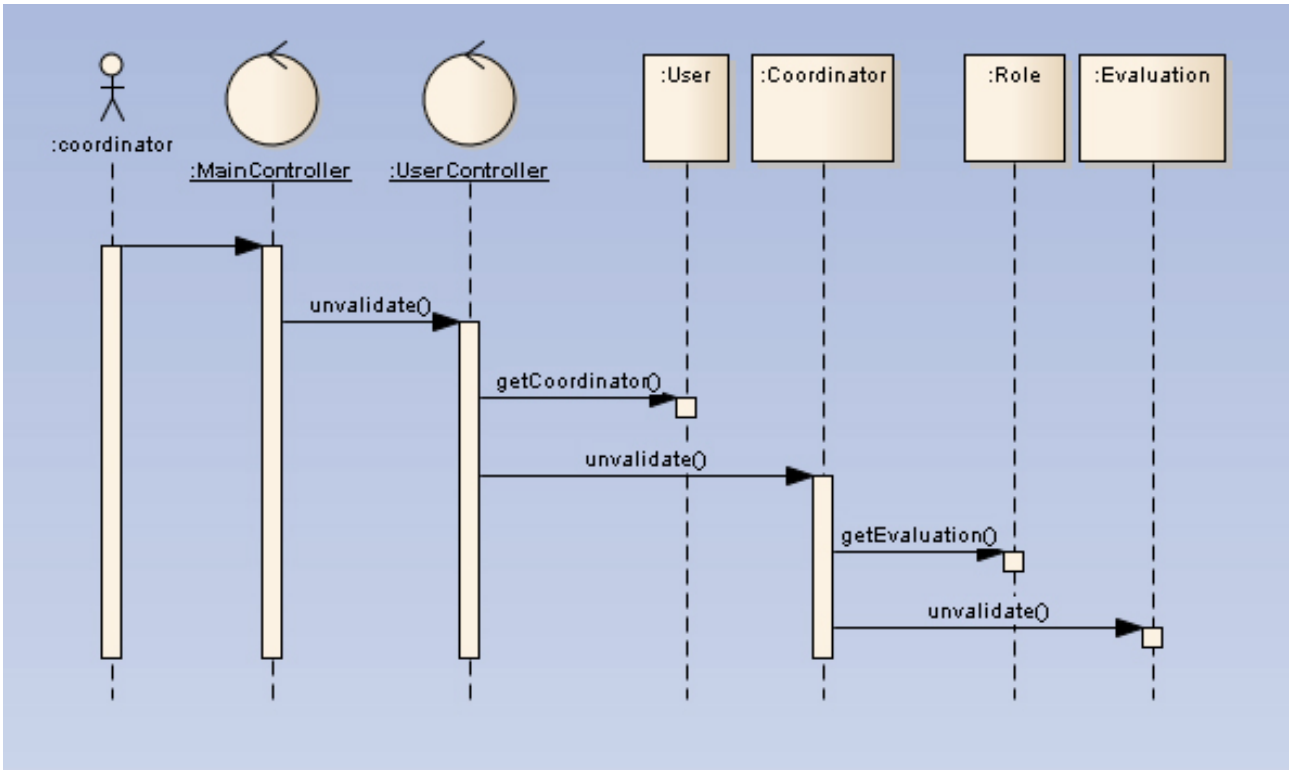

Figura 6.11: Diagrama de Sequência: Cancelar Validação de Momento de Avaliação

Capítulo 7

Implementação

7.1 Introdução

Enquadramento geral do projecto, objectivos a atingir e identificação das tarefas que foram objecto de implementação.

7.2 Decisões Globais de Implementação

7.2.1 Tecnologias Utilizadas

Para este projecto foi escolhida a linguagem de programação PHP com ligação a base de dados MySQL, é utilizado o principio de programação orientada a objectos com um modelo de arquitectura Model-View-Controller (MVC).
Para a ligação entre o PHP e o MySQL é utilizada a extenção MySQLi. Para a criação das páginas a apresentar ao utilizador foi implementado o sistema de gestão de templates para PHP - Smarty (versão 3.1). E também o API de Javascript - jQuery (versão jquery-1.3.1).

7.2.2 Armazenamento dos Dados

Os dados utilizados pela aplicação são armazenados na base de dados local (mySQL) e quando um utilizador acede ao site é criada uma sessão em PHP que guarda, temporariamente, os dados necessários para manter o utilizador logado bem como as datas escolhidas (inicial, final e actual).

7.2.3 Hierarquia de Ficheiros

Os ficheiros com os controladores estão colocados na directoria *"controller"*, os do modelo na directoria *"model"*, da vista na directoria *"view"*. As folhas de estilo ("CSS") dentro da directoria *"css"*, os elementos gráficos são colocados dentro de *"imagens"*. Todos os ficheiros com as bibliotecas javascript ou funções criadas para este site ficam dentro da directoria *"js"*. Por fim os ficheiros utilizados pelo Smarty para apresentação de html ficam dentro da directoria *"templates"*.
As classes utilizadas no site estão fisicamente localizadas em ficheiros de extensão PHP com o mesmo nome da respectiva classe.

7.3 Decisões de Implementação Específicas

7.3.1 Ligação à Base de Dados

O controlador principal do site está no ficheiro `MainController.php`, no seu construtor é criado um objecto que servirá de ligação à base de dados. esse objecto é da classe de modelo `BD`. Esta classe aceita no seu construtor o endereço do servidor onde a base de dados está armazenada, o nome de utilizador, password e o nome da base de dados. Com esses dados o sistema efectua a ligação.
Quando a execução do código termina, esta classe fecha a conexão com a base de dados através do seu método `__destruct`.
A classe para ligação à base de dados contém vários métodos dos quais se podem aqui destacar o método `multiQuery` que permite obter resultados que uma chamada de uma stored procedure que retorne mais que uma linha. Para consultas "normais"existem os métodos `query` que retorna os resultados num array, `execute` para executar querys que não retornem resultados e o `getRow` que apenas retorna uma linha.

7.3.2 Funcionamento do controlador principal

O controlador principal `MainController` começa por criar uma sessão `PHP` no servidor, criar os objectos para a ligação à base de dados, vista e dois objectos para controladores "secundários": `UserController` e `CalendarController`.
De seguida executa a acção pretendida pelo utilizador através do método `actions`, este método verifica no array `$_GET` as variáveis recebidas sendo que o primeiro será a classe a utilizar e os seguintes serão os métodos com o respectivo parâmetro.
Por exemplo, para um acesso com `?Calendar&setDay=20131023`, é executado o método `setDay(20131023)` da classe `CalendarController`.

7.3.3 Login

Sempre que o utilizador acede ao site é criado um objecto da classe `UserController` que recebe no seu construtor o objectos da ligação à base de dados e da vista. É então criado o `user` da classe de modelo `User` e executado o método `login`. Este método verifica se foi submetido o formulário de login, sendo que o tenha sido executa o método `login` do `user` com o username e password recebidos como parâmetros, caso não tenha sido submetido o formulário o método é chamado sem parâmetros.
Os objectos da classe `User` contêm três objectos das classes de modelo `Coordinator`, `Teacher` e `Student`, o seu construtor executa também o método da própria classe `login`. Este método, verifica se recebe parâmetros, caso não tenha recebido, executa o método `setUser`, caso tenha recebido, executa um query para confirmar a existência de um utilizador com os dados recolhidos.
Após estar definido, ou não, o utilizador é executado o método `setCourses`.
Este método começa por chamar o método `getCourses` do objecto da classe `BD` que por sua vez executa a stored procedure `get_cursos_user` que retorna a lista de cursos a que o utilizador está associado, para cada um deles será adicionado ao objecto correspondente ao papel do utilizador em questão o respectivo curso.
Para cada curso é obtida a lista de disciplinas que está associada ao utilizador com o papel do

mesmo. A partir da lista obtida é adicionada ao curso a mesma.
Por fim, para cada disciplina obtida sao consultadas as avaliações da mesma, associadas ao utilizador e são adicionadas ao objecto da Disciplina criado.

7.3.4 Logout

Quando um utilizador efectua o logout, todas as variáveis do objecto da classe modelo `User` e executa o método `clear` de cada uma das classes dos papéis possiveis. Estas classes herdam da classe `Role` que contem os métodos comuns a todas. Este método esvazia o array com os cursos.

7.3.5 Listar Avaliações

O sistema após ter feito as tarefas de verificar que utilizador está logado e quais os cursos, disciplinas e avaliações a ele estão associados, executa as acções solicitadas pelo método `GET`. Terminadas estas tarefas o `MainController` executa de novo o método `setCourses` no `UserController` que por sua vez delega a execução no `User`.
Após isso solicita ao controlador `CalendarController` quais as datas, inicial e final, definidas e solicita ao `UserController` as avaliações contidas nesse intervalo de datas. O controlador solicita as avaliações no intervalo ao `Usert` que por sua vez solicita a cada um dos `Role`, estes obtêm a lista a partir de cada um dos `Course`, que interrogam as suas `Discipline` que por sua vez retornam todas as `Evaluation` cuja data esteja dentro do intervalo pretendido.
Esses dados são passados à `View` que apresenta o resultado ao utilizador com auxílio do template engine *Smarty*.

7.3.6 Ver Detalhes de Avaliação

Os detalhes das avaliações são apresentados ao ser seleccionado um dia no calendário. Sempre que o utilizador acede ao site, o `CalendarController` cria um novo objecto da classe de modelo `Calendar`.
No seu construtor é de imediato definida a data inicial, data final e o dia actual, que poderá ser o dia em que o acesso é feito ao sistema ou dia escolhido no calendário. Sempre que o código deixa de ser executado no servidor o método `__destruct` desta classe armazena em variáveis da sessão actual as datas. Essas variáveis serão sempre utilizadas a cada acesso para a definição das mesmas, caso não haja interacção do utilizador a esse nível.
Com a data actual adquirida o `MainController` solicita ao `UserController` as avaliações associadas ao `User` para esse dia.
Obtida a lista, a mesma é passada à classe de modelo `Details`, que organiza a informação por forma a ser facilmente utilizada pelo Smarty que apresenta os detalhes das mesmas, apresentando os comandos possiveis a cada uma de acordo com o `Role` associado ao `User`.

7.3.7 Marcar Avaliação

A marcação de uma avaliação é uma acção que é encaminhada para o `UserController` e chamado o método `newEvaluation`.
Neste método é verificado se foi submetido o formulário para a inserção da nova avaliação, em caso afirmativo o controlador adiciona a avaliação ao `User` através do método `newEvaluation` que

por sua vez executa a stored procedure `add_evaluation`, como já foi explicado anteriormente a stored procedure verifica se o utilizador tem a permissão para criar tal avaliação.
Caso não tenha sido submetido o formulário, o controlador questiona à classe `User` quais as disciplinas que o `User` tem como `Role: Teacher` através do método `getTeaching` da classe `User`. A variável `teaching` é definida no `login`, ou seja logo quando o `User` é criado em que question a classe `BD` quais as disciplinas em que o `User` é `Teacher`. Sempre que esta variável contém alguma disciplina o menu do site apresenta o item *"nova avaliação"*.
Obtidos os dados é apresentado o formulário ao utilizador com uma caixa de selecção onde são apresentadas as disciplinas onde o utilizador é docente.

7.3.8 Cancelar Avaliação

Ao cancelar uma avaliação o `UserController` solicita o objecto `Teacher` ao `User` e executa nesse objecto o método `cancelEvaluation`, desta forma apenas poderá ser cancelada uma avaliação em que o `User` é `Teacher`.

7.3.9 Validar e Cancelar Validação de Avaliação

A validação da avaliação é feita de forma semelhante à anterior, mas deta forma o controlador solicita ao `User` o objecto `Coordinator` e executa o seu método `validate` ou `unvalidate` conforme a situação.

7.3.10 Inscrever e Cancelar Inscrição em Avaliação

A inscrição numa avaliação é feita com o `UserController` a solicitar ao `User` o objecto `Student` e executando o seu método `inscribe` ou `uninscribe` conforme a acção pretendida.

Capítulo 8

Conclusão

O trabalho apresentado neste relatório, resultou num site onde é possivel gerir momentos de avaliação de cursos leccionados no IPBeja. É possivel um utilizador seguir com o mesmo registo ao longo do tempo, podendo assumir os papéis de Aluno, Docente e/ou Coordenador de Curso. É possivel ao utilizador visualizar as avaliações relacionadas com os papéis por ele desempenhados e, mais especificamente, enquanto Coordenador de Curso: validar ou cancelar a validação de uma avaliação, enquanto Docente: marcar ou cancelar a marcação de uma avaliação e enquanto Aluno: inscrever-se ou cancelar a inscrição num momento de avaliação.
Para gerar as página html foi utilizado o motor de templates `smarty` que torna mais simples e de melhor compreensão essas tarefas.
Por uma questão de segurança, com o intuito de evitar erros de programação ou acessos indevidos, foram criados stored procedures para controlar o acesso ou alterações aos momentos de avaliação.
Como trabalho futuro deverá ser implementada uma forma de criação de um login à base de dados para cada utilizador para que se possa limitar o acesso directo às tabelas e o mesmo seja apenas feito através de stored procedures onde será utilizado como referência para as permissões de acesso aos dados o utilizador logado ao invés da disponibilização do id de utilizador como está feito actualmente.

Apêndice A

Apêndices

A.1 index.php

```
<?php
#error_reporting(1);
#error_reporting(E_ALL);
#ini_set('display_errors', '1');

require_once("controller/MainController.php");
$avaliacoes = new MainController();
?>
```

A.2 controller/MainController.php

```
<?php
define( "HOST",  "localhost"  );
define( "USERNAME", "moreira_aval"   );
define( "PASSWORD", "2000Santarem"   );
define( "DATABASE", "moreira__avaliacoes" );

require_once("controller/CalendarController.php");
require_once("controller/UserController.php");
require_once("model/BD.php");
require_once("model/Details.php");
require_once("view/View.php");

class MainController {
  private $user;
  private $con;
  private $calendar;
  private $details;

  function __construct(){
    session_start();
```

```
  $this->con     = new BD(HOST, USERNAME, PASSWORD, DATABASE);

  $this->view    = new View();
  $this->user    = new UserController($this->con, $this->view);
  $this->calendar  = new CalendarController($this->con, $this->view);

  $this->actions();

  $this->user->setCourses();

  $start = $this->calendar->getStartingDate();
  $end = $this->calendar->getEndingDate();
  $evaluations = $this->user->getEvaluationsByDates($start, $end);

  $this->calendar->addEvaluations($evaluations);

  $actual_day = $this->calendar->getActualDay();
  $evaluations = $this->user->getEvaluationsByDate($actual_day);

  $this->details = new Details($evaluations);

  $this->setView();
  #$this->teste();


}

private function actions(){
  if(count($_GET) > 0){
    $i = 0;
    foreach($_GET as $key => $value){
      # primeiro GET, se definido e o construtor a chamar
      if($i == 0){
        if(class_exists($key."Controller")){
          $controller = $key."Controller";
          $object = strtolower($key);
        }
      }

      # segundo GET, se definido, e o metodo a chamar
      if($i == 1){
        if(method_exists($controller, $key)){
          $this->$object->$key($value);
        }
      }
      $i++;
    }
  }
}

private function setView(){
```

```
    $this->user->setView();
    $this->calendar->setView();
    $this->view->setDetails($this->details->getDetails());
    $this->view->setMenu();
    $this->view->showAll();
  }
}
?>
```

A.3 controller/UserController.php

```
<?php
require_once("model/User.php");

require_once("view/View.php");

class UserController{

  private $user;
  private $view;
  private $con;

  function __construct($con, $view){
    $this->con     = $con;
    $this->view    = $view;
    $this->user    = new User($this->con);
    $this->login();
  }

  public function login(){
    if(isset($_POST['form_utilizador'])){
      $username = $_POST['form_utilizador'];
      $password = $_POST['form_password'];
      $this->user->login($username, $password);
    } else {
      $this->user->login();
    }
  }

  public function setCourses(){ $this->user->setCourses();}

  public function validate($evaluation){
    if(isset($_GET['validate'])){
      $coordinator = $this->user->getCoordinator();
      $coordinator->validate($evaluation);
    }
  }

  public function unvalidate($evaluation){
```

```
  if(isset($_GET['unvalidate'])){
    $coordinator = $this->user->getCoordinator();
    $coordinator->unvalidate($evaluation);
  }
}

public function inscribe($evaluation){
  if(isset($_GET['inscribe'])){
    $student = $this->user->getStudent();
    $id = $this->user->getUserNum();
    $student->inscribe($id, $evaluation);
  }
}

public function uninscribe($evaluation){
  if(isset($_GET['uninscribe'])){
    $student = $this->user->getStudent();
    $id = $this->user->getUserNum();
    $student->uninscribe($id, $evaluation);
  }
}

public function cancelEvaluation($evaluation){
  if(isset($_GET['cancelEvaluation'])){
    $teacher = $this->user->getTeacher();
    $teacher->cancelEvaluation($evaluation);
  }
}

public function newEvaluation(){
  if(isset($_POST['evaluation'])){
    $array[0] = $_POST['disciplina'];
    $array[1] = $_POST['data'];
    $array[2] = $_POST['peso'];
    $array[3] = $_POST['sala'];
    $array[4] = $_POST['tipo'];
    $array[5] = $_POST['observacoes'];
    $this->user->newEvaluation($array);
  } else {
    $disciplines = $this->user->getTeaching();
    $this->view->showNewEvaluation($disciplines);
  }
}

public function setView(){
  $this->view->setLogged($this->user->getUserNum() != NULL);
  $this->view->setUserName($this->user->getName());
  $this->view->setTeacher(count($this->user->getTeaching()) > 0);
}

public function logout()  { $this->user->logout();    }
```

```
  public function getEvaluationsByDate($date){
    return $this->user->getEvaluationsByDate($date);
  }

  public function getEvaluationsByDates($start, $end){
    return $this->user->getEvaluationsByDates($start, $end);
  }
}
?>
```

A.4 controller/CalendarController.php

```
<?php
require_once("model/Calendar.php");

class CalendarController{
  private $view;
  private $calendar;
  private $con;

  function __construct($con, $view){
    $this->con   = $con;
    $this->view  = $view;
    $this->calendar = new Calendar();
  }

  public function move($i){
    $this->calendar->move($i);
  }

  public function setView(){
    $this->view->setType($this->getType());
    $this->view->setCalendar($this->getDays());
    $date = $this->calendar->getActualDay();
    $this->view->setDate(array(date("d/m/Y", $date), $date));
  }

  public function setDay($day){
    $this->calendar->setDay($day);
    $this->calendar->build();
  }

  public function addEvaluations($evaluations){
    $this->calendar->addEvaluations($evaluations);
  }

  public function getDays()   { return $this->calendar->getDays();   }
  public function getStartingDate() { return $this->calendar->getStartingDate(); }
```

```
  public function getEndingDate()  { return $this->calendar->getEndingDate();  }
  public function getActualDay()  { return $this->calendar->getActualDay();  }
  public function getType()   { return $this->calendar->getType();   }
}
```

A.5 model/BD.php

```
<?php
class BD{
  private $host;
  private $db;
  private $username;
  private $password;
  private $con;

  /*
   * construtor, define as variaveis e chama o metodo para a conexao
   */
  function __construct($host, $username, $password, $db){
    $this->host      = $host;
    $this->db      = $db;
    $this->username    = $username;
    $this->password    = $password;
    $this->connect();

  }

  public function getHost()   { return $this->host;   }
  public function getDB()    { return $this->db;    }
  public function getUsername()  { return $this->username;  }
  public function getCon()   { return $this->con;    }

  /*
   * Ligacao a base de dados
   */
  public function connect(){
    $this->con = new mysqli($this->host, $this->username, $this->password, $this->db);
    if (mysqli_connect_errno()) {
      printf("Connect failed: %s\n", mysqli_connect_error());
    }
  }

  public function isTeacher($user_num){
    $query = sprintf("
      select d.num_disciplina, d.titulo
      from docente as t
      left join disciplina as d
      on d.num_disciplina = t.num_disciplina
      where num_utilizador = %d", $user_num);
```

```
  $result = $this->query($query);

  return $result;
}

public function getEvaluations($role, $user_num, $discipline){
  $value = array();
  switch($role){
    case "coordinator":
      $query = sprintf("call get_coordinator_evaluations( %d, %d );", $user_num,
          $discipline);
      $value = $this->multiQuery($query);
      break;
    case "teacher":
      $query = sprintf("call get_teacher_evaluations( %d, %d );", $user_num,
          $discipline);
      $value = $this->multiQuery($query);
      break;
    case "student":
      $query = sprintf("call get_student_evaluations( %d, %d );", $user_num,
          $discipline);
      $value = $this->multiQuery($query);
      break;
    default:
      break;
  }
  return $value;
}

public function getCourses($user_num){
  $query = sprintf("call get_cursos_user( %d )", $user_num);
  return $this->multiQuery($query);
}

public function getDisciplines($role, $user_num, $course_id){
  $value = array();
  switch($role){
    case "coordinator":
      $query = sprintf("call get_coordinator_disciplines( %d, %d )", $user_num,
          $course_id);
      $value = $this->multiQuery($query);
      break;
    case "teacher":
      $query = sprintf("call get_teacher_disciplines( %d, %d )", $user_num,
          $course_id);
      $value = $this->multiQuery($query);
      break;
    case "student":
      $query = sprintf("call get_student_disciplines( %d, %d )", $user_num,
          $course_id);
      $value = $this->multiQuery($query);
```

```
      break;
    default:
      break;
  }
  return $value;
}

public function multiQuery($query){
  $results = array();
  if ($this->con->multi_query($query)) {
      do {
          if ($result = $this->con->use_result()) {
              while ($row = $result->fetch_row()) {
                  array_push($results, $row);
              }
              $result->close();
          }
      } while ($this->con->next_result());
  }
  return $results;
}

/*
 * efectuar uma consulta que se supoe retornar resultados
 */
public function query($query){
  $results = array();
  $temp = $this->con->query($query) or die($this->con->error.__LINE__);

  if($temp->num_rows > 0) {
    while($row = $temp->fetch_assoc()) {
      array_push($results, $row);
    }
  }
  return $results;
}

public function getRow($query){
  $result = array();
  $temp = $this->con->query($query) or die($this->con->error.__LINE__);

  if($temp->num_rows > 0) {
    $row = $temp->fetch_assoc();
    foreach($row as $key => $value){
      $result[$key] = $value;
    }
  }
  return $result;
}

/*
```

```
   * executar comandos mysql que nao retornem resultados (insert, delete, update,
       etc...)
   */
  public function execute($query){
    $this->con->query($query);
  }

  /*
   * desligar a conexao a bd
   */
  function __destruct(){
    mysqli_close($this->con);
  }
}
?>
```

A.6 model/Calendar.php

```
<?php
require_once("Day.php");

define("MONTHS", 6);
define("DEFAULT_TYPE", "monthly");

class Calendar{
  private $type;
  private $starting_date;
  private $days;
  private $ending_date;
  private $actual_day;

  function __construct($type = "", $starting_date = ""){
    $this->days = array();
    $this->setType($type);
    $this->setStartingDate($starting_date);
    $this->setEndingDate();
    $this->setDay();
    $this->build();
  }

  public function getDays()   { return $this->days;   }
  public function getStartingDate() { return $this->starting_date; }
  public function getEndingDate()  { return $this->ending_date; }
  public function getActualDay()  { return $this->actual_day;  }
  public function getType()   { return $this->type;   }

  public function setDay($day = ""){
    if($day == ""){
      if(isset($_SESSION['actual_day'])){
```

```
      $this->actual_day = $_SESSION['actual_day'];
    } else {
      $this->actual_day = strtotime(date("Ymd"));
    }
  } else {
    $this->actual_day = strtotime($day);
  }
}

public function setType($type){
  if($type == ""){
    if(isset($_SESSION['type'])){
      $type = $_SESSION['type'];
    } else {
      $type = DEFAULT_TYPE;
    }
  }
  $this->type = $type;
  $this->setStartingDate();
  $this->setEndingDate();
}

public function move($i){
  $date = $this->starting_date;
  if($this->type == "monthly"){
    $date = mktime(0,0,0, date("m", $date) + $i, 1, date("Y", $date));
  } else {
    $date = mktime(0,0,0, date("m", $date), date("d", $date) + $i, date("Y", $date));
  }
  $this->setStartingDate($date);
  $this->setEndingDate();
  $this->build();
}

private function setStartingDate($day = ""){
  if($day == ""){
    if(isset($_SESSION['starting_date'])){
      $day = $_SESSION['starting_date'];
    } else {
      $day = strtotime(date("Ymd"));
    }
  }
  if($this->getType() == "monthly"){
    $this->starting_date = mktime(0, 0, 0, date("m", $day), 1, date("Y", $day));
  } else {
    $offset = date("w", $day);
    $this->starting_date = mktime(0, 0, 0, date("m", $day), date("d", $day) -
        $offset, date("Y", $day));
  }
  $_SESSION['starting_date'] = $this->starting_date;
}
```

```
private function setEndingDate(){
  $start = $this->starting_date;
  if($this->type == "monthly"){
    $this->ending_date = mktime(0, 0, 0, date("m", $start) + MONTHS, date("t",
        $start), date("Y", $start));
  } else {
    $this->ending_date = mktime(0,0,0, date("m", $start), date("d", $start) + 6,
        date("Y", $start));
  }
}

public function build(){
  $this->days = array();
  #if($this->type == "monthly"){
    $this->vista_actual = 1;
    $this->buildMensal();
  #} else {
  # $this->vista_actual = 0;
  # $this->buildSemanal();
  #}
}

public function addEvaluations($evaluations){
  foreach($evaluations as $role => $years){
    foreach($years as $year => $days){
      if(array_key_exists($year, $this->days)){

        foreach($days as $day => $evals){
          foreach($evals as $e){
            $evaluation = $e['evaluation'];
            if(!array_key_exists($day, $this->days[$year]['avaliacoes'])){
              $this->days[$year]['avaliacoes'][$day] = array();
              if($evaluation->getValidated() == 0){
                $this->days[$year]['class'][$day] = "dia-mes cor-erro";
              } else {
                $this->days[$year]['class'][$day] = "dia-mes cor-ok";
              }
            } else {
              $this->days[$year]['class'][$day] = "dia-mes cor-erro";
            }
            array_push($this->days[$year]['avaliacoes'][$day], $e);
          }
        }
      }
    }
  }
}

private function buildMensal(){
  $this->vista_selected = 0;
```

```
    $month = date("m", $this->starting_date);
    $year = date("Y", $this->starting_date);
    $day = new Day(mktime(0, 0, 0, $month, 1, $year));

    for($i = 0; $i < MONTHS; $i++){
      $id = $day->getYear().$day->getMonth();
      $this->days[$id]['mes']['num'] = $month;
      $this->days[$id]['mes']['titulo'] = $day->getMonthName();
      $this->days[$id]['ano'] = $day->getYear();
      for($j = 0; $j < $day->getOffset(); $j++){
        $this->days[$id]['dias'][$j] = " ";
        $this->days[$id]['nomes'][$j] = "";
      }

      do{
        $class = "dia_mes";
        $id = $day->getYear().$day->getMonth();
        $d = $day->getDay();
        $this->days[$id]['dias'][$j] = $day->getDay();
        $this->days[$id]['data'][$j] = $day->getYear().$day->getMonth().$day->getDay();
        $this->days[$id]['class'][$d] = $day->getDate() == $this->actual_day ?
            "dia_seleccionado" : $class;
        $this->days[$id]['nomes'][$j++] = sprintf("%s", $day);
        $this->days[$id]['avaliacoes']  = array();
        $day = $day->moveDay(1);
      } while($day->getMonth() == $month);

      for(;$j < 42; $j++){
        $this->days[$id]['dias'][$j] = " ";
        $this->days[$id]['nomes'][$j] = "";
      }

      $month = $day->getMonth();
    }
  }

  function __destruct(){
    $_SESSION['starting_date'] = $this->starting_date;
    $_SESSION['ending_date'] = $this->ending_date;
    $_SESSION['actual_day']  = $this->actual_day;
    $_SESSION['type']   = $this->type;
  }
}
?>
```

A.7 model/Coordinator.php

```
<?php
require_once ("Course.php");
```

```
require_once ("Role.php");

class Coordinator extends Role {

  function __construct($con) {
    parent::__construct($con, "coordinator");
  }

  public function setDisciplines() {
    parent::setDisciplines(parent::$con -> getCoordinatorDisciplines($id));
  }

  public function validate($id) {
    $evaluation = parent::getEvaluation($id);
    if ($evaluation) {
      $evaluation -> validate(parent::getCon());
    }
  }

  public function unvalidate($id) {
    $evaluation = parent::getEvaluation($id);
    if ($evaluation) {
      $evaluation -> unvalidate(parent::getCon());
    }
  }

}
?>
```

A.8 model/Course.php

```
<?php
require_once("model/Discipline.php");
class Course{
  private $id;
  private $name;
  private $description;
  private $year;
  private $disciplines;

  function __construct($course){
    $this->id    = $course[0];
    $this->name    = $course[1];
    #$this->description = $course['description'];
    $this->year    = $course[2];
    $this->disciplines = array();
  }

  public function addDiscipline($discipline){
```

```
    $this->disciplines[$discipline[0]] = new Discipline($discipline);
  }

  public function getEvaluations($discipline = ""){
    $evaluations = array();
    if($discipline != ""){
      $evaluations = $this->disciplines[$discipline]->getEvaluations();
    } else {
      $evaluations = array();
      foreach($this->disciplines as $discipline){
        $d = $discipline->getId();
        if(!array_key_exists($d, $evaluations)){
          $evaluations[$d] = array(
            "discipline" => $discipline->getTitle(),
            "evaluations" => array()
          );
        }
        array_push($evaluations[$d]["evaluations"], $discipline->getEvaluations());
      }
    }
    return $evaluations;
  }

  public function getEvaluationsByDate($date) {
    $result = array();
    foreach($this->disciplines as $d){
      $evaluations = $d->getEvaluationsByDate($date);
      foreach($evaluations as $ym => $days){
        if(!array_key_exists($ym, $result)){
          $result[$ym] = array();
        }
        foreach($days as $day => $evs){
          if(!array_key_exists($day, $result[$ym])){
            $result[$ym][$day] = array();
          }
          foreach($evs as $e){
            array_push(
              $result[$ym][$day],
              array(
                "course"   => $this->getName(),
                "discipline" => $d->getTitle(),
                "evaluation" => $e
              )
            );
          }
        }
      }
    }
    return $result;
  }
```

```
  public function getEvaluationsByDates($start, $end) {
    $result = array();
    foreach($this->disciplines as $d){

      $evaluations = $d->getEvaluationsByDates($start, $end);
      foreach($evaluations as $ym => $days){
        if(!array_key_exists($ym, $result)){
          $result[$ym] = array();
        }
        foreach($days as $day => $evs){
          if(!array_key_exists($day, $result[$ym])){
            $result[$ym][$day] = array();
          }
          foreach($evs as $e){
            array_push(
              $result[$ym][$day],
              array(
                "course"   => $this->getName(),
                "discipline" => $d->getTitle(),
                "evaluation" => $e
              )
            );
          }
        }
      }
    }
    return $result;
  }

  public function getEvaluation($id){
    $evaluation = NULL;
    foreach($this->disciplines as $discipline){
      $evaluation = $discipline->getEvaluation($id);
      if($evaluation != NULL){
        return $evaluation;
      }
    }
    return $evaluation;
  }

  public function getId()        { return $this->id;      }
  public function getName()      { return $this->name;    }
  public function getDescription()   { return $this->description;  }
  public function getYear()      { return $this->year;    }
  public function getDiscipline($id)   { return $this->disciplines[$id]; }
  public function getDisciplines()   { return $this->disciplines;  }
}
?>
```

A.9 model/Day.php

```
<?php
class Day{
  private $time;
  private static $day_names;
  private static $month_names;

  function __construct($time){
    $this->time = $time;
    self::$day_names = array(
      array("full" => "Domingo",     "min" => "DOM"),
      array("full" => "Segunda-feira",  "min" => "SEG"),
      array("full" => "Ter&ccedil;a-feira", "min" => "TER"),
      array("full" => "Quarta-feira",   "min" => "QUA"),
      array("full" => "Quinta-feira",   "min" => "QUI"),
      array("full" => "Sexta-feira",    "min" => "SEX"),
      array("full" => "S&aacute;bado",   "min" => "SAB")
    );
    self::$month_names = array(
      NULL,
      array("full" => "Janeiro",     "min" => "JAN"),
      array("full" => "Feveiro",     "min" => "FEV"),
      array("full" => "Mar&ccedil;o",   "min" => "MAR"),
      array("full" => "Abril",     "min" => "ABR"),
      array("full" => "Maio",      "min" => "MAI"),
      array("full" => "Junho",     "min" => "JUN"),
      array("full" => "Julho",     "min" => "JUL"),
      array("full" => "Agosto",    "min" => "AGO"),
      array("full" => "Setembro",    "min" => "SET"),
      array("full" => "Outubro",     "min" => "OUT"),
      array("full" => "Novembro",    "min" => "NOV"),
      array("full" => "Dezembro",    "min" => "DEZ")
    );
  }

  public function moveDay($i){
    return new Day(
      mktime(
        date("H", $this->time),
        date("i", $this->time),
        date("s", $this->time),
        date("m", $this->time),
        date("d", $this->time) + $i,
        date("Y", $this->time)
      )
    );

  }
```

```
  public function moveMonth($i){
    return new Day(
      mktime(
        date("H", $this->time),
        date("i", $this->time),
        date("s", $this->time),
        date("m", $this->time) + $i,
        date("d", $this->time),
        date("Y", $this->time)
      )
    );
  }

  public function getDate()  { return $this->time;    }
  public function getDay()  { return date("d", $this->time); }
  public function getMonth()  { return date("m", $this->time); }
  public function getYear()  { return date("Y", $this->time); }
  public function getWeek()  { return date("W", $this->time); }
  public function getOffset()  { return date("w", $this->time); }

  public function __toString(){
    return sprintf("%s, %d de %s de %d",
      $this->getDayName(),
      $this->getDay(),
      $this->getMonthName(),
      $this->getYear()
    );
  }

  public function getDayName(){
    return self::$day_names[date("w", $this->time)]["full"];
  }

  public function getDayNameMin(){
    return self::$day_names[date("w", $this->time)]["min"];
  }

  public function getMonthName(){
    return self::$month_names[date("n", $this->time)]["full"];
  }

  public function getMonthNameMin(){
    return self::$month_names[date("n", $this->time)]["min"];
  }
}
?>
```

A.10 model/Details.php

```php
<?php
class Details{
  private $details;

  function __construct($evaluations){
    $this->details = array();
    $this->buildDetails($evaluations);
  }

  private function buildDetails($evaluations){
    $this->details = array();
    foreach($evaluations as $role => $months){
      foreach($months as $days){
        foreach($days as $evs){
          foreach($evs as $id => $e){
            $evaluation = $e['evaluation'];
            array_push(
              $this->details,
              array(
                "role"     => $role,
                "course"   => $e['course'],
                "discipline" => $e['discipline'],
                "date"     => date("d/m/Y", $evaluation->getDate()),
                "type"     => $evaluation->getType(),
                "weight"   => $evaluation->getWeight(),
                "classroom"  => $evaluation->getClassRoom(),
                "id"     => $evaluation->getID(),
                "validated"  => $evaluation->getValidated()
              )
            );
          }
        }
      }
    }
  }

  public function getDetails(){ return $this->details; }
}
?>
```

A.11 model/Evaluation.php

```php
<?php
class Evaluation{
  private $id;
  private $type;
  private $weight;
  private $date;
```

```
  private $validated;
  private $classroom;
  private $course_id;
  private $discipline_id;

  function __construct($evaluation){
    $this->id      = $evaluation[0];
    $this->type    = $evaluation[1];
    $this->weight  = $evaluation[2];
    $this->date    = strtotime($evaluation[3]);
    $this->validated  = $evaluation[4];
    $this->classroom  = NULL;
    $this->course_id  = $evaluation[5];
    $this->discipline_id = $evaluation[6];
  }

  public function validate($con)  { $this->setValidated($con, 1); }
  public function unvalidate($con) { $this->setValidated($con, 0); }

  private function setValidated($con, $value){
    $this->validated = $value;
    $query = sprintf("UPDATE avaliacao SET activada = %d WHERE num_avaliacao = %d",
        $value, $this->id);
    $con->execute($query);
  }

  public function setType($type)    { $this->type = $type;  }
  public function setWeight($weight)   { $this->weight = $weight; }
  public function setDate($date)    { $this->date = $date;  }

  public function getId()    { return $this->id;    }
  public function getType()   { return $this->type;   }
  public function getDate()   { return $this->date;   }
  public function getWeight()   { return $this->weight;   }
  public function getValidated()  { return $this->validated;  }
  public function getClassRoom()  { return $this->classroom;  }
  public function getCourseId()  { return $this->course_id;  }
  public function getDisciplineId() { return $this->discipline_id; }
}
?>
```

A.12 model/Role.php

```
<?php
abstract class Role{
  private $con;
  private $title;
  private $courses;
```

```
function __construct($con, $title){
  $this->con = $con;
  $this->title = $title;
  $this->courses = array();
}

public function clear(){
  $this->courses = array();
}

public function addCourse($course){
  $this->courses[$course[0]] = new Course($course);
}

public function getCon(){
  return $this->con;
}

public function getEvaluation($id){
  $evaluation = NULL;
  foreach($this->courses as $course){
    $evaluation = $course->getEvaluation($id);
    if($evaluation != NULL){
      return $evaluation;
    }
  }
  return $evaluation;
}

public function execute($query){
  $this->con->execute($query);
}

public function getCourse($id) { return $this->courses[$id]; }
public function getCourses() { return $this->courses;  }

public function getEvaluationsByDate($date){
  $result = array();
  foreach($this->courses as $course){
    $evaluations = $course->getEvaluationsByDate($date);
    foreach($evaluations as $ym => $days){
      if(!array_key_exists($ym, $result)){
        $result[$ym] = array();
      }
      foreach($days as $day => $evs){
        if(!array_key_exists($day, $result[$ym])){
          $result[$ym][$day] = array();
        }
        foreach($evs as $e){
          array_push(
            $result[$ym][$day],
```

```
            array(
              "course"   => $e['course'],
              "discipline" => $e['discipline'],
              "evaluation" => $e['evaluation']
            )
          );
        }
      }
    }
  }
  return $result;
}

public function getEvaluationsByDates($start, $end){
  $result = array();
  foreach($this->courses as $course){
    $evaluations = $course->getEvaluationsByDates($start, $end);
    foreach($evaluations as $ym => $days){
      if(!array_key_exists($ym, $result)){
        $result[$ym] = array();
      }
      foreach($days as $day => $evs){
        if(!array_key_exists($day, $result[$ym])){
          $result[$ym][$day] = array();
        }
        foreach($evs as $e){
          array_push(
            $result[$ym][$day],
            array(
              "course"   => $e['course'],
              "discipline" => $e['discipline'],
              "evaluation" => $e['evaluation']
            )
          );
        }
      }
    }
  }
  return $result;
}
}
?>
```

A.13 model/Student.php

```
<?php
require_once("Course.php");
require_once("Role.php");
```

```
class Student extends Role{

  function __construct($con){
    parent::__construct($con, "student");
  }

  public function setDisciplines(){
    parent::setDisciplines(parent::$con->getStudentDisciplines($id));
  }

  public function inscribe($user_num, $evaluation){
    $evaluation = parent::getEvaluation($evaluation);
    if($evaluation){
      $query = sprintf("
        INSERT INTO avaliacao_aluno
        SET num_avaliacao = %d,
        num_utilizador = %d,
        num_disciplina = %d,
        num_semestre = 1",
        $evaluation->getId(),
        $user_num,
        $evaluation->getDisciplineId()
      );
      parent::execute($query);
    }
  }

  public function uninscribe($user_num, $evaluation){
    $evaluation = parent::getEvaluation($evaluation);
    if($evaluation){
      $query = sprintf("
        DELETE FROM avaliacao_aluno
        WHERE num_avaliacao = %d
        AND num_utilizador = %d",
        $evaluation->getId(), $user_num
      );
      parent::execute($query);
    }
  }
}
?>
```

A.14 model/Teacher.php

```
<?php
require_once("Course.php");
require_once("Role.php");

class Teacher extends Role{
```

```
  function __construct($con){
    parent::__construct($con, "teacher");
  }

  public function setDisciplines(){
    parent::setDisciplines(parent::$con->getTeacherDisciplines($id));
  }

  public function cancelEvaluation($evaluation){
    $evaluation = parent::getEvaluation($evaluation);
    if($evaluation){
      $query = sprintf("
        DELETE FROM avaliacao
        WHERE num_avaliacao = %d
        ",
        $evaluation->getId()
      );
      parent::execute($query);
    }
  }
}
?>
```

A.15 model/User.php

```
<?php
require_once("model/Coordinator.php");
require_once("model/Teacher.php");
require_once("model/Student.php");

class User{
  private $con;
  private $user_num;
  private $name;
  #private $address;
  #private $phone_number;
  #private $email;
  private $username;
  private $password;
  #private $foto;
  private $coordinator;
  private $teacher;
  private $student;
  private $teaching;

  function __construct($con){
    $this->con     = $con;
    $this->coordinator = new Coordinator($this->con);
```

```
  $this->teacher  = new Teacher($this->con);
  $this->student  = new Student($this->con);
  $this->login();
}

private function setUser(){
  if(isset($_SESSION['username'])){
    $query = sprintf("SELECT * FROM utilizador WHERE username = '%s' AND pass =
        '%s'",
      $_SESSION['username'], $_SESSION['password']);
    $result = $this->con->getRow($query);
    if(count($result) > 0){
      $this->user_num   = $result['num_utilizador'];
      $this->name     = $result['nome'];
      #$this->address   = $result['morada'];
      #$this->phone_number = $result['telefone'];
      #$this->email    = $result['email'];
      $this->username   = $result['username'];
      #$this->foto     = $result['foto'];
      $this->password   = $result['pass'];
      $this->teaching   = $this->con->isTeacher($this->user_num);
    } else {
      $this->logout();
    }
  } else {
    $this->logout();
  }
}

public function newEvaluation($array){
  # array(discipline_id, date, weight, calssroom, type, observations)
  # user_num INT(11), discipline_id, date, weight, classroom, type, observations )

  $query = sprintf("call add_evaluation('%d', '%s', '%s', '%s', '%s', '%s', '%s')",
    $this->user_num, $array[0], date("Ymd", $array[1]),
    $array[2], $array[3], $array[4], $array[5]
  );
  $this->con->execute($query);
}

public function login($username = "", $password = ""){
  if($username == "" || $password == ""){
    $this->setUser();
  } else {
    $query = sprintf("SELECT * FROM utilizador WHERE username = '%s' AND pass =
        '%s'",
      $username, md5($password));
    $result = $this->con->getRow($query);
    if(count($result) > 0){
      $this->user_num   = $result['num_utilizador'];
      $this->name     = $result['nome'];
```

```
      $this->address    = $result['morada'];
      $this->phone_number  = $result['telefone'];
      $this->email    = $result['email'];
      $this->username   = $result['username'];
      $this->foto     = $result['foto'];
      $this->password   = $result['pass'];
      $this->teaching   = $this->con->isTeacher($this->user_num);
    }
  }
  $this->setCourses();
}

public function setCourses(){
  $user_num = $this->getUserNum();
  // obter todos os cursos associados ao user_num

  /* course returned array fields:
   * 0 => id
   * 1 => name
   * 2 => year
   * 3 => role
   */
  $courses = $this->con->getCourses($user_num);
  foreach($courses as $course){
    $role = $course[3];
    // criar o curso no role respectivo
    $this->$role->addCourse($course);
    // obter as disciplinas associadas ao curso e user_num

    /* discipline returned array fields:
     * 0 => id
     * 1 => description
     * 2 => title
     */
    $disciplines = $this->con->getDisciplines($course[3], $user_num, $course[0]);

    $c = $this->$role->getCourse($course[0]);
    foreach($disciplines as $discipline){
      // criar a disciplina no curso do respectivo role do user_num
      $c->addDiscipline($discipline);

      /* evaluation returned array fields:
       * 0 => id
       * 1 => type
       * 2 => weight
       * 3 => date
       * 4 => validated
       */
      $evaluations = $this->con->getEvaluations($role, $user_num, $discipline[0]);
      $d = $c->getDiscipline($discipline[0]);
      foreach($evaluations as $e){
```

```
        $e[5] = $c->getId();
        $e[6] = $d->getId();
        $d->addEvaluation($e);
      }
    }
  }
}

public function logout(){
  $this->user_num  = NULL;
  $this->name    = NULL;
  $this->address  = NULL;
  $this->phone_number = NULL;
  $this->email   = NULL;
  $this->username  = NULL;
  $this->foto    = NULL;
  $this->password  = NULL;
  $this->teaching  = array();
  $this->teacher->clear();
  $this->coordinator->clear();
  $this->student->clear();
}

public function getUserNum()  { return $this->user_num;  }
public function getName()   { return $this->name;   }
public function getUsername()  { return $this->username;  }
public function getCoordinator() { return $this->coordinator; }
public function getTeacher()  { return $this->teacher;  }
public function getStudent()  { return $this->student;  }
public function getTeaching()  { return $this->teaching;  }

public function getCourses()
{
  $courses = array( "coordinator" => $this->coordinator,
           "teacher"  => $this->teacher,
           "student"  => $this->student );
  return $courses;
}

public function getCourse($id){
  $courses = array( "coordinator" => $this->coordinator->getCourse($id),
           "teacher"  => $this->teacher->getCourse($id),
           "student"  => $this->student->getCourse($id) );
  return $courses;
}

public function getEvaluationsByDate($date){
  $coordinator = $this->coordinator->getEvaluationsByDate($date);
  $teacher   = $this->teacher->getEvaluationsByDate($date);
  $student   = $this->student->getEvaluationsByDate($date);
```

```
    return array("coordinator" => $coordinator, "teacher" => $teacher, "student" =>
        $student);
  }

  public function getEvaluationsByDates($start, $end){
    $coordinator = $this->coordinator->getEvaluationsByDates($start, $end);
    $teacher = $this->teacher->getEvaluationsByDates($start, $end);
    $student = $this->student->getEvaluationsByDates($start, $end);

    return array("coordinator" => $coordinator, "teacher" => $teacher, "student" =>
        $student);
  }

  function __destruct(){
    $_SESSION['username'] = $this->username;
    $_SESSION['password'] = $this->password;
  }
}
?>
```

A.16 view/View.php

```
<?php
define('SMARTY_DIR', 'smarty/libs/');
require_once(SMARTY_DIR . 'Smarty.class.php');

class View{
  private $smarty;
  private $logged;
  private $username;
  private $type;
  private $details;
  private $is_teacher;
  private $menu;

  function __construct(){
    $this->smarty = new Smarty();
    $this->smarty->assign('template_dir', 'templates/');
    $this->smarty->assign('compile_dir', 'templates_c/');
    $this->smarty->assign('configs_dir', 'configs/');
    $this->smarty->assign('cache_dir', 'cache/');
    $this->type = "mensal.tpl";
    $this->menu = "";
    $this->form = "";
  }

  public function showAll(){
    $this->smarty->assign("form", $this->form);
    $this->smarty->display("index.tpl");
```

```
}

public function showCalendar(){
  $this->smarty->display($this->tipe);
}

public function showLogin(){
  $this->smarty->display("identificacao.tpl");
}

public function setLogged($value){
   $this->logged = $value;
   if($value == TRUE){
    $this->smarty->assign("login_action", "?User&logout");
    $this->smarty->assign("titulo_botao_login", "logout");
   } else {
    $this->smarty->assign("login_action", "#");
    $this->smarty->assign("titulo_botao_login", "login");
   }
}

public function setTeacher($value){
  $this->is_teacher = $value;
}

public function setDetails($details){
  $this->details = "";
  foreach($details as $e){
    $this->details = sprintf(
      '%s
        {assign "date" "%s"}
        {assign "discipline" "%s"}
        {assign "type" "%s"}
        {assign "weight" "%s"}
        {assign "classroom" "%s"}
        {assign "id" "%s"}
        {assign "validated" "%s"}
        {include file="%s_details.tpl"}
        <hr>
      ',
      $this->details,
      $e['date'],
      $e['discipline'],
      $e['type'],
      $e['weight'],
      $e['classroom'],
      $e['id'],
      $e['validated'],
      $e['role']
    );
  }
```

```
    $this->smarty->assign("details", $this->details);
  }

  public function setMenu(){
    if($this->is_teacher == TRUE){
      $this->menu = sprintf(
        '%s
          {include file="teacher_menu.tpl"}
        ',
        $this->menu
      );
    }
    $this->smarty->assign("menu", $this->menu);
  }

  public function setUserName($name){
    $this->username = $name;
    $this->smarty->assign("username", $name);
  }

  public function setDate($date){
    $this->smarty->assign("data", $date);
  }

  public function setCalendar($calendar){
    $this->smarty->assign("calendario", $calendar);
  }

  public function showNewEvaluation($disciplines){
    $this->smarty->assign("disciplinas", $disciplines);
    $this->form = '{include file="new_evaluation.tpl"}';
  }

  public function setType($value){
    $this->tipo = $value.".tpl";
    $this->smarty->assign("tipo_calendario", $this->tipo);
  }
}
```

A.17 css/simple.css

```
@charset "utf-8";
body{
  font-family:Arial, Helvetica, sans-serif;
  margin-left:0px;
  margin-right:0px;
}
div#ajuda, div#ajuda_login, div#ajuda_site{
  position:absolute;
```

```
  top:0px;
  left:0px;
  padding:10px;
}
div#painel_login, div#avisos, #detalhes_avaliacao{
  display:none;
}
div#titulo_site{
  margin:auto;
  text-align:center;
}
div#painel_login, div#avisos, div#detalhes_avaliacao{
  font-size:12px;
  text-align:center;
}
div#b_login{
  position: absolute;
  top: 0px;
  right: 0px;
  padding:10px;
}
table#forms{
  margin-left: auto;
  margin-right: auto;
  padding:5px;
  border: 1px solid black;
}
table#forms td.titulos{
  text-align:right;
  font-weight:bold;
}
table#forms td.valores{
  text-align:left;
}
/* MENU */
div#menu{
  background-color:#000;
  color:#FFFFFF;
  width:100%;
  overflow: auto;
  display:block;
}
div#menu_itens{
  float:left;
  overflow:auto;
  padding:10px;
  color:#FFFFFF;
}

div#menu_itens a { color:#FFFFFF;}
div#menu_item{
```

```
  color:white;
  font-weight:bold;
  font-size:14px;
  padding: 20px;
  display:inline-block;
}
div#alertas{
  padding-top:10px;
  padding-bottom:10px;
  text-align:center;
  width:100%;
}
div#calendario{
  font-family:Arial, Helvetica, sans-serif;
  font-size:10px;
  float:left;
  width:555px;
}
div#vistas{
  clear:both;
  margin-bottom:10px;
  margin-top:10px;
  overflow:auto;
}
div#prev{
  width:30px;
  height:100%;
  float:left;
  margin-right:5px;
  cursor:pointer;
}
div#next{
  width:30px;
  height:100%;
  float:left;
  margin-left:5px;
  cursor:pointer;
}
div#identificacao{
  float:right;
  padding:10px;
  font-size:70%;
  overflow:auto;
  color:#FFFFFF;

}
div#conteudo{
  margin:auto;
  text-align:center;
  width:910px;
  clear:both;
```

```
  overflow:auto;
}
div#menu_item:hover{
  background-color:#666;
}
div#mes{
  float:left;
  padding:2px;
}
.d0, .d1, .d2, .d3, .d4{
  float:left;
  width:10px;
  height:10px;
}
.d0{background-color:#EEE;}
.d1{background-color:blue;}
.d2{background-color:green;}
.d3{background-color:GoldenRod;}
.d4{background-color:red;}
.titulo_semana_seleccionado{
  background-color:#0CF;
  cursor:pointer;
}
div#detalhes{
  width:250px;
  float:right;
  height:100%;
  text-align:left;
  border:#000 1px solid;
  padding:5px;
}
table#semana{
  border:#000 1px solid;
  width:100%;
}
.bold{
  font-weight:bold;
}
.titulo_semana{
  border:#000 1px solid;
  cursor:pointer;
}
.ano_um  { background-color:#EEE; height:75px; border:#000 1px solid; }
.ano_dois { background-color:#CCC; height:75px; border:#000 1px solid; }
.ano_tres { background-color:#AAA; height:75px; border:#000 1px solid; }
.calendario_titulo_mes{
  background-color:#000;
  color:#FFF;
  text-transform:uppercase;
  font-weight:bold;
}
```

```
.calendario_titulo_dias{
  background-color:#CCC;
}
.mes{
  border:#999 1px solid;
  width:180px;
}
.calendario_dias{
  background-color:#EEE;
}
.dia_seleccionado{
  background-color:#0CF;
  cursor:pointer;
}
.dia_mes{
  background-color:#EEE;
  cursor:pointer;
}
.cor-neutro{
  background-color:#EEE;
}
.cor-info{
  background-color: blue;
  color:white;
}
.cor-ok{
  background-color: green;
  color:white;
}
.cor-erro{
  background-color:GoldenRod;
  color:white;
}
.cor-alerta{
  background-color: red;
  color:white;
}

.datas{
  font-size:70%;
}
.disciplina{
  font-size:90%;
  font-weight:bold;
}
table.detalhes{
  width:100%;
}
table.detalhes tr td{
  text-align:center;
  border:#000 1px solid;
```

```
  font-size:60%;
}
#validar, #cancelar, #inscrever, #detalhes_aval, #eliminar_avaliacao, #inscrever,
    #cancelarInscricao{
  width:45%;
  font-size:60%;
}
#validar{
  background-color:#F00;
  color:#FFF;
}


#data, #sala{
  width:25%;
  font-size:60%;
}
ul{
  text-align:left;
}
/* Overlay */
#simplemodal-overlay {background-color:#000; cursor:wait;}

/* Container */
#simplemodal-container {height:200px; width:300px; color: #000; background-color:
    #FFF; border:4px solid #444; padding:12px;}

#simplemodal-container .simplemodal-data {padding:8px;}

#simplemodal-container code {background:#141414; border-left:3px solid #65B43D;
    color: #000; display:block; font-size:12px; margin-bottom:12px; padding:4px 6px
    6px;}

#simplemodal-container a {color:#000;}

#simplemodal-container a.modalCloseImg {background:url(../imagens/x.png) no-repeat;
    width:25px; height:29px; display:inline; z-index:3200; position:absolute;
    top:-15px; right:-16px; cursor:pointer;}

#simplemodal-container h3 {color: #000;}
```

A.18 scripts/codes.js

```
$(document).ready(function() {

  $('#semestre').click(function(){
    window.location.href='?Calendario&tipo=mensal';
  });
```

```
  $('#semana').click(function(){
    window.location.href='?Calendario&tipo=semanal';
  });

  $('#login').click(function() {
    $("#painel_login").modal({overlayClose:true});
  });

});

function validarAvaliacao(num_avaliacao){
  window.location.href='?User&validate='+num_avaliacao;
}

function cancelarValidacao(num_avaliacao){
  window.location.href='?User&unvalidate='+num_avaliacao;
}

function cancelarAvaliacao(num_avaliacao){
  window.location.href='?User&cancelEvaluation='+num_avaliacao;
}

function inscrever(num_avaliacao){
  window.location.href='?User&inscribe='+num_avaliacao;
}

function cancelarInscricao(num_avaliacao){
  window.location.href='?User&uninscribe='+num_avaliacao;
}
```

A.19 templates/b_login.tpl

```
<a href="{$login_action}" title="{$titulo_botao_login}" id="{$titulo_botao_login}" >
       {$titulo_botao_login}
     </a>
```

A.20 templates/coordinator_details.tpl

```
<span class="datas">{$date}</span>
<br>
<span class="disciplina">{$discipline}</span>
<br>
<span class="texto">{$type}</span>
<table class="detalhes">
  <tbody>
    <tr>
```

```
      <td>peso da avalia&ccedil;&atilde;o</td>
      <td>sala(s)</td>
    </tr>
    <tr>
      <td>{$weight}</td>
      <td>{$classroom}</td>
    </tr>
  </tbody>
</table>
{if $validated == 0}
<input id="validar" value="validar avalia&ccedil;&atilde;o"
    onclick="validarAvaliacao({$id});" type="button">
{else}
<input id="cancelar" value="cancelar valida&ccedil;&atilde;o"
    onclick="cancelarValidacao({$id});" type="button">
{/if}
<input id="data" value="alterar data" type="button"><input id="sala" value="alterar
    sala" type="button">
```

A.21 templates/detalhes_avaliacao.tpl

```
    <div id="ajuda">ajuda</div>
    <h1>DETALHES DA AVALIA&Ccedil;&Atilde;O</h1>
        <div id="detalhes_disciplina"></div>
        <div id="detalhes_tipo"></div>
        <div id="detalhes_data"></div>

        <table>
          <tr>
              <td>
                  <span class="bold">Descri&ccedil;&atilde;o</span><br />
              <div id="detalhes_descricao"></div>
                </td>
                <td>
                  <span class="bold">Restri&ccedil;&otilde;es</span><br />
              <div id="detalhes_restricoes"></div>
                </td>
            </tr>
        </table>
        <div id="detalhes_salas"></div>
        <div id="detalhes_pesos"></div>
  <input type="button" value="fechar" class="simplemodal-close" />
```

A.22 templates/detalhes.tpl

```
detalhes do dia
```

A.23 templates/identificacao.tpl

```
{$username}
```

A.24 templates/index.tpl

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title></title>
<link href="css/simple.css" rel="stylesheet" type="text/css" />
<script type='text/javascript' src='scripts/jquery.js'></script>
<script type="text/javascript" src="scripts/jquery-1.3.1.min.js"></script>
<script type='text/javascript' src='scripts/jquery.simplemodal.js'></script>
<script type="text/javascript" src="scripts/codes.js"></script>
</head>
<body>
  <div id="cabecalho">
      <div id="ajuda_site">
        <a href="#">Ajuda</a>
      </div>
        <div id="b_login">
          {include file="b_login.tpl"}
        </div>

      <div id="titulo_site">
          <img src="/pcr/imagens/logotipo.png"
              alt="Logotipo "
                title="Logotipo" />
          <h1>
            GEST&Atilde;O DE AVALIA&Ccedil;&Otilde;ES</h1>
        </div>

    </div>
    <!-- fim do cabecalho -->

    <div id="menu">
      <div id="menu_itens">
        {eval $menu}
      </div>
      <div id="identificacao">
        {include file="identificacao.tpl"}
      </div>
    </div>

    <div id="conteudo">
```

```
    <div id="forms">
    {eval $form}
  </div>
      <div id="vistas">
          {include file="vistas.tpl"}
        </div>
      <div id="prev">
      <a href="?Calendar&move=-1">
        <img src="imagens/prev.jpg" title="Retroceder" alt="seta a esquerda" />
      </a>
    </div>
    <div id="calendario">
         {include file=$tipo_calendario}
    </div>
    <div id="next">
      <a href="?Calendar&move=1">
        <img src="imagens/next.jpg" title="Avancar" alt="seta a direita" />
      </a>
    </div>

        <div id="detalhes">
      {eval $details}
    </div>
    </div>

  <div id="avisos"></div>

  <div id="painel_login">
    <div id="ajuda_login">ajuda</div>
    <h1>AUTENTICA&Ccedil;&Atilde;O</h1>
        <form action="?User&login" method="post"
          enctype="multipart/form-data" id="autenticacao" name="autenticacao">
        <label for="form_utilizador">UTILIZADOR:</label>
        <input type="text" id="form_utilizador" name="form_utilizador" />
        <br />
        <label for="form_password">PASSWORD:</label>
        <input type="password" id="form_password" name="form_password" />
        <br />
        <div id="lgn"><input type="submit" id="ok" name="ok" value="ok"/></div>
        <input type="reset" value="cancel" class="simplemodal-close" />

    <br />recuperar password
        </form>
  </div>

  <div id="detalhes_avaliacao">
      {* include file="detalhes_avaliacao.tpl" *}
    </div>
</body>
</html>
```

A.25 templates/menu.tpl

```
{$menu}
```

A.26 templates/monthly.tpl

```
{foreach from=$calendario item=vars}
<div id="mes">
<table class="mes">
<tr class="calendario_titulo_mes">
  <td colspan="7">{$vars['mes']['titulo']}, {$vars['ano']}</td>
</tr>
<tr class="calendario_titulo_dias">
  <td>d</td>
  <td>s</td>
  <td>t</td>
  <td>q</td>
  <td>q</td>
  <td>s</td>
  <td>s</td>
</tr>

{assign var=i value=0}
  {foreach from=$vars['dias'] key=key_dia item=dia}
     {if $i eq 7 }
       </tr>
       {assign var=i value=0}
     {/if}
     {if $i eq 0}
     <tr>
    {/if}
      <td class="calendario_dias">
          <div class="{if array_key_exists($dia,
              $vars['class'])}{$vars['class'][$dia]}{/if}">
            {if $dia > 0}
              <a
                href="?Calendar&setDay={$vars['data'][$key_dia]}"
                title="{$vars['nomes'][$key_dia]}"
              >
                {$dia}
              </a>
            {else}
               
            {/if}
          </div>
      </td>

      {$i = $i+1}
```

```
    {/foreach}
</table>
</div>
{/foreach}
```

A.27 templates/new_evaluation.tpl

```
<hr>
<form method="post" action="" name="nova_avaliacao" id="nova_avaliacao"
    enctype="multipart-form/data" >
<table id="forms">
  <thead>
    <tr>
      <th colspan="2"><h3>INSERIR NOVA AVALIA&Ccedil;&Atilde;O</h3></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td class="titulos">Tipo de avalia&ccedil;&atilde;o:</td>
      <td class="valores"><input type="text" name="tipo"></td>
    </tr>
    <tr>
      <td class="titulos">Data:</td>
      <td class="valores">
        {$data[0]}
        <input type="hidden" name="data" value="{$data[1]}">
      </td>
    </tr>
    <tr>
      <td class="titulos">Peso:</td>
      <td class="valores"><input type="text" name="peso"></td>
    </tr>
    <tr>
      <td class="titulos">Sala:</td>
      <td class="valores"><input type="text" name="sala"></td>
    </tr>
    <tr>
      <td class="titulos">Observa&ccedil;&otilde;es:</td>
      <td class="valores"><textarea name="observacoes"></textarea></td>
    </tr>
    <tr>
      <td class="titulos">Disciplina:</td>
      <td class="valores">
        <select name="disciplina">
          <option value="">Escolha uma disciplina</option>
          {foreach from=$disciplinas item=d}
              <option value="{$d.num_disciplina}">{$d.titulo}</option>
          {/foreach}
        </select>
```

```
      </td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <th colspan="2">
        <input type="hidden" name="evaluation" value="nova">
        <input type="submit" value="inserir">
        <input type="reset" value="limpar">
      </th>
    </tr>
  </tfoot>
</table>
</form>
<hr>
```

A.28 templates/painel_login.tpl

```
<div id="ajuda_login">ajuda</div>
    <h1>AUTENTICA&Ccedil;&Atilde;O</h1>
        <form action="" method="post"
          enctype="text/plain" id="autenticacao" name="autenticacao">
        <label for="form_utilizador">UTILIZADOR:</label>
        <input type="text" id="form_utilizador" name="form_utilizador" value="" />
        <br />
        <label for="form_password">PASSWORD:</label>
        <input type="password" id="form_password" name="form_password" value="" />
        <br />
        <div id="lgn"><input type="submit" id="ok" name="ok" value="ok"/></div>
        <input type="reset" value="cancel" class="simplemodal-close" />

    <br />recuperar password
        </form>
```

A.29 templates/rodape.tpl

```
</div>
</body>
</html>
```

A.30 templates/student_details.tpl

```
<span class="datas">{$date}</span>
<br>
```

```
<span class="disciplina">{$discipline}</span>
<br>
<span class="texto">{$type}</span>
<table class="detalhes">
  <tbody>
    <tr>
      <td>peso da avalia&ccedil;&atilde;o</td>
      <td>sala(s)</td>
    </tr>
    <tr>
      <td>{$weight}</td>
      <td>{$classroom}</td>
    </tr>
  </tbody>
</table>
{if $validated == 0}
<input id="validar" value="inscrever" onclick="inscrever({$id});" type="button">
{else}
<input id="cancelar" value="cancelar inscri&ccedil;&atilde;o"
    onclick="cancelarInscricao({$id});" type="button">
{/if}
```

A.31 templates/teacher_details.tpl

```
<span class="datas">{$date}</span>
<br>
<span class="disciplina">{$discipline}</span>
<br>
<span class="texto">{$type}</span>
<table class="detalhes">
  <tbody>
    <tr>
      <td>peso da avalia&ccedil;&atilde;o</td>
      <td>sala(s)</td>
    </tr>
    <tr>
      <td>{$weight}</td>
      <td>{$classroom}</td>
    </tr>
  </tbody>
</table>
<input id="cancelar" value="cancelar avalia&ccedil;&atilde;o"
    onclick="cancelarAvaliacao({$id});" type="button">
<input id="data" value="alterar data" type="button"><input id="sala" value="alterar
    sala" type="button">
```

A.32 templates/teacher_menu.tpl

```
<a href="?User&newEvaluation">nova avalia&ccedil;&atilde;o</a>
```