Laboratórios de Engenharia de Software

S1 - Configuração do Curso

Relatório de desenvolvimento

versão 1.4

Universidade do Porto

Faculdade de Engenharia

FEUP

Turma 4LEIC3

André Fidalgo Moniz {ei99041@fe.up.pt}

José António Fonseca {ei99032@fe.up.pt}

Mário Filipe Pereira {ei99047@fe.up.pt}

Miguel Flores Sarmento {ei96049@fe.up.pt}

10 de Janeiro de 2003

Conteúdo

Lista de Figuras

Agradecimentos

Agradecemos ao homem invisível. Agradecemos à Sabina por ter aberto sempre a porta.

André, José, Mário e Miguel

1 Introdução

Este relatório tem como principal propósito descrever as diversas fases do desenvolvimento do projecto, a nível conceptual e implementacional. Este documento é acompanhado de diversos diagramas que explicam o funcionamento da aplicação. São também abordadas as alterações efectuadas ao definido nos relatórios anteriores.

A aplicação criada tem como principal objectivo facilitar o trabalho de um Gestor de um curso que necessita de configurar o curso, disciplinas, docentes, alunos e salas. Era também objectivo realizar um *webservice* disponível com a informação sobre um dado plano de estudos.

1.1 Estrutura do relatório

O relatório presente está divido em 6 capítulos e 1 anexo. No primeiro capítulo introduzimos o leitor ao relatório e aos objectivos do projecto. O segundo capítulo faz referência às alterações efectuadas sobre os requisitos especificados, e no terceiro as alterações à arquitectura usada.

Na secção seguinte são descritos os componentes do sistema e sua implementação. É feita uma descrição minuciosa das funções implementadas. Neste mesmo capítulo são também descrito os testes de integração efectuados e o estado de implementação do projecto. Através de diagramas de sequência são também descritos todos os casos de uso implementados.

A documentação de gestão de projecto engloba o quinto capítulo e aborda os tempos gastos pelos executantes ao longo do projecto, finalizando com um comentário sobre o trabalho do grupo.

No final do relatório são tiradas as devidas conclusões. Em anexo colocámos o manual do utilizador que demonstra como se pode usar as funcionalidades do sistema.

2 Revisão da especificação de requisitos

O único tópico que alteramos no relatório de especificação de requisitos foi o modelo de classes de domínio. Os casos uso foram mantidos visto a especificação ter sido validada com sucesso. As alterações foram as seguintes:

- criar classe de categorias de docentes
- apagar relação disciplina com área de conhecimento
- criar classes para matrículas e frequências dos alunos
- foi retirado o campo tópico na classe de conteúdos da disciplina
- criar classe users

2.1 Revisão do modelo de classes do domínio

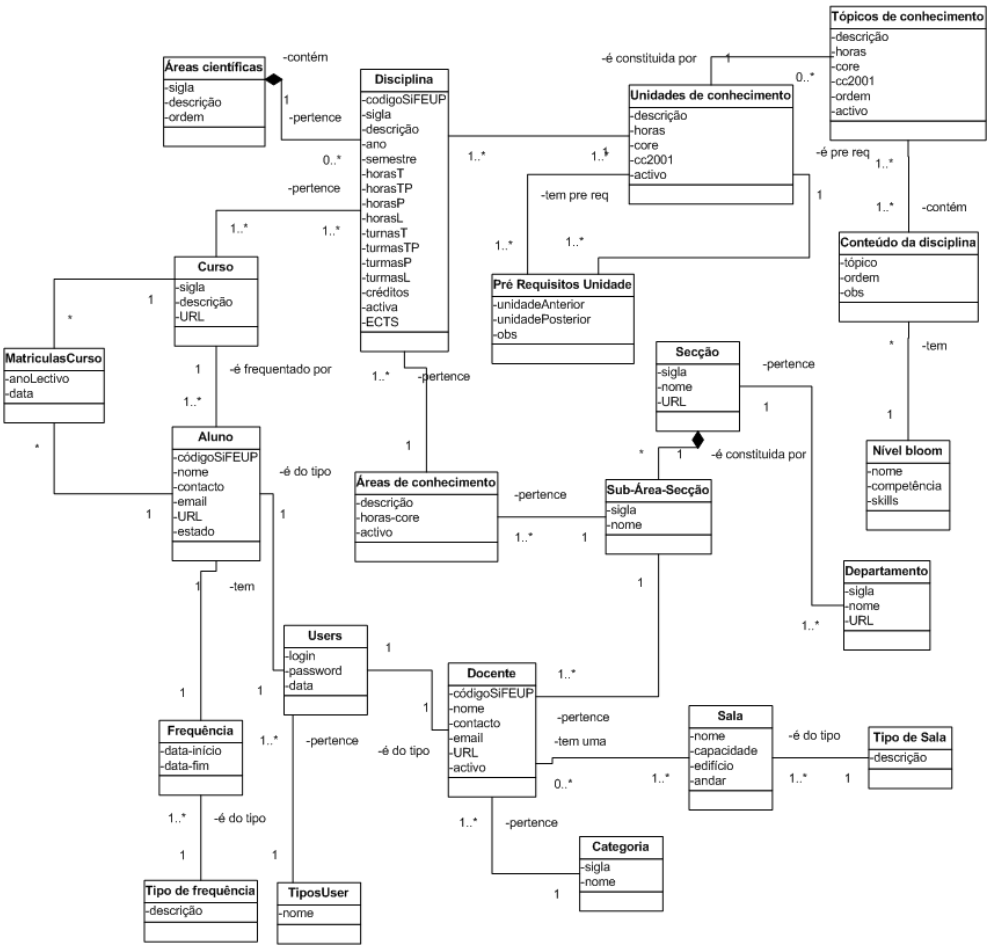

Figura 1: Modelo de classes de domínio

3 Revisão da arquitectura

Na realidade a arquitectura lógica e física foram mantidas mas tivemos que proceder a alterações que serão descritas ao longo do capítulo 4.

As alterações incidiram no número de ficheiros usados na camada de interface e isso foi devido ao enorme número de formulários, consequência do número elevado de casos de uso. Assim foram gerados ficheiros que não estavam especificados. Nas camadas mais baixas não foram efectuadas alterações ao planeado.

4 Relatório de projecto detalhado

4.1 Especificação dos componentes

Todos os componentes da camada da lógica de negócio e da camada de serviço de dados estão disponíveis, pois a nossa opção de implementar todos os componentes como webservices permite essa disponibilidade. Sendo assim, todos os método referentes à camada de lógica de negócio e da camada de serviço de dados que estão descritos na secção seguinte podem ser acedidos fora do nosso sistema. Quanto aos serviços requeridos não foi especificado nenhum deste género.

4.2 Implementação dos componentes

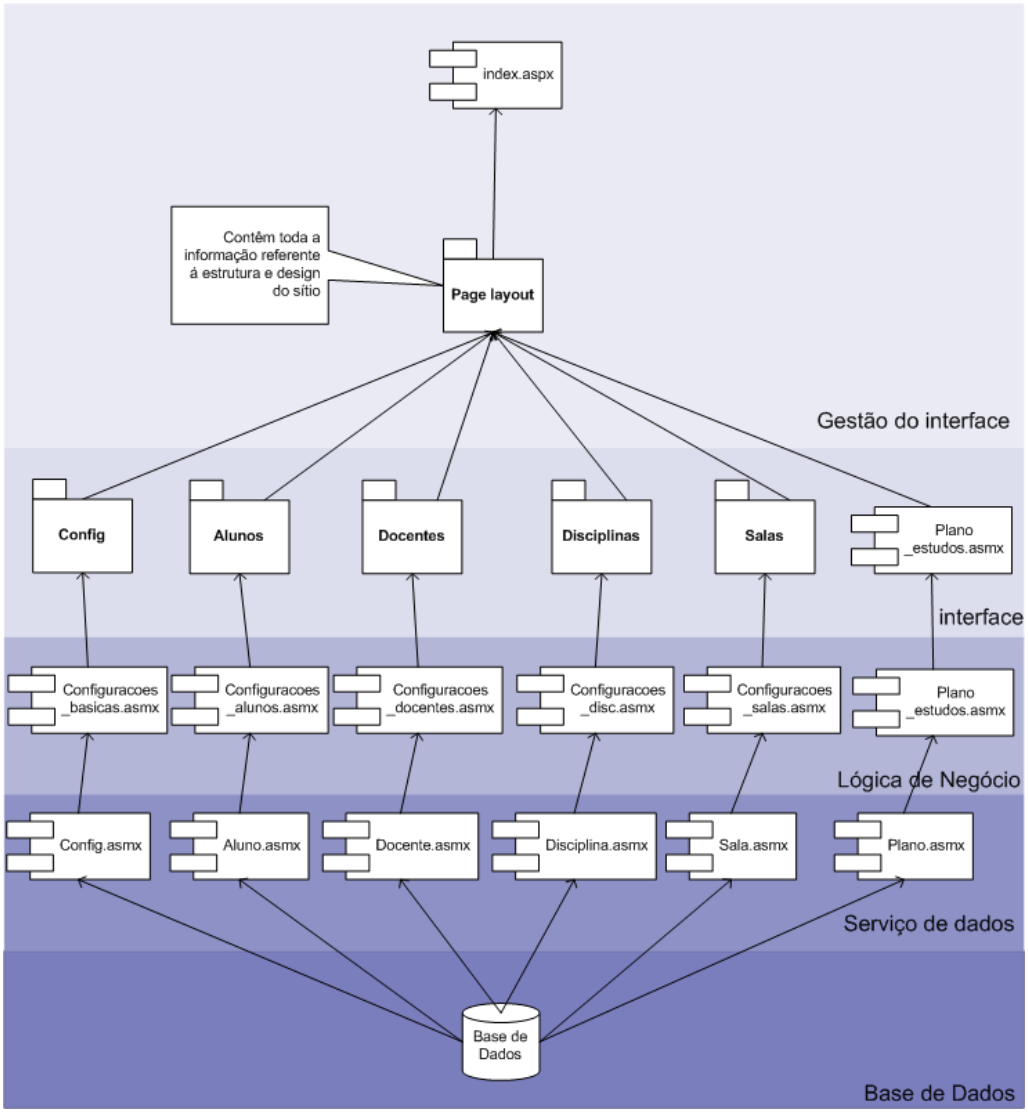

Figura 2: Diagrama de componentes

O diagrama apresentado esquematiza a organização dos vários componentes presentes na nossa arquitectura bem como as camadas em que essa arquitectura [Far01] se divide.

A base de dados, onde é armazenada a informação persistente. O serviço de dados, que tem como principal objectivo encapsular a estrutura da base de dados estabelecendo a ligação com a lógica de negócio. A lógica de negócio, que tem por objectivo processar a informação vinda da camada de interface

ou do serviço de dados. O interface, que está dividido em duas partes bastantes distintas, a Gestão do interface onde são definidos todos os layouts, menus e estilos das páginas (para que o sistema seja completamente uniforme) e o interface propriamente dito onde, basicamente, existem apenas formulários.

4.2.1 Gestão do interface

- **index.aspx**

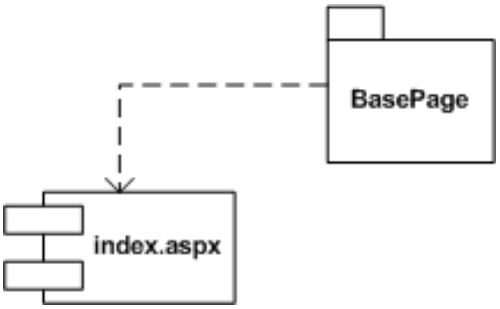

Figura 3: Componente index.aspx

Este componente define a estrutura e os estilos de todo o interface(através da package BasePage (secção 4.2.1)). É o index.aspx que se encarrega de identificar, através do reconhecimento de uma variável enviada pela URL (p.ex: index.aspx?go=salas_registar), e carregar os componentes necessários para a construção da página.

Normalmente, quando se cria uma página dinâmica, definem-se várias coisas como rodapés, cabeçalhos e menus que depois são incluídos em todas as páginas. Esta abordagem, apesar de ser a mais usada, não é a mais correcta pois não permite bastantes coisas, como por exemplo: a construção de layouts dinâmicos e facilmente alteráveis.

A nossa abordagem é exactamente o oposto do normal. Nós temos apenas um ficheiro (neste caso o index.aspx) onde é definido o layout (cabeçalhos, rodapés, menus e estilos) que por sua vez inclui os conteúdos necessários para cada página. Desta forma é possível desenvolver conteúdos (formulários, listagens, etc.) independentes de todo o layout.

- **Package pagelayout**

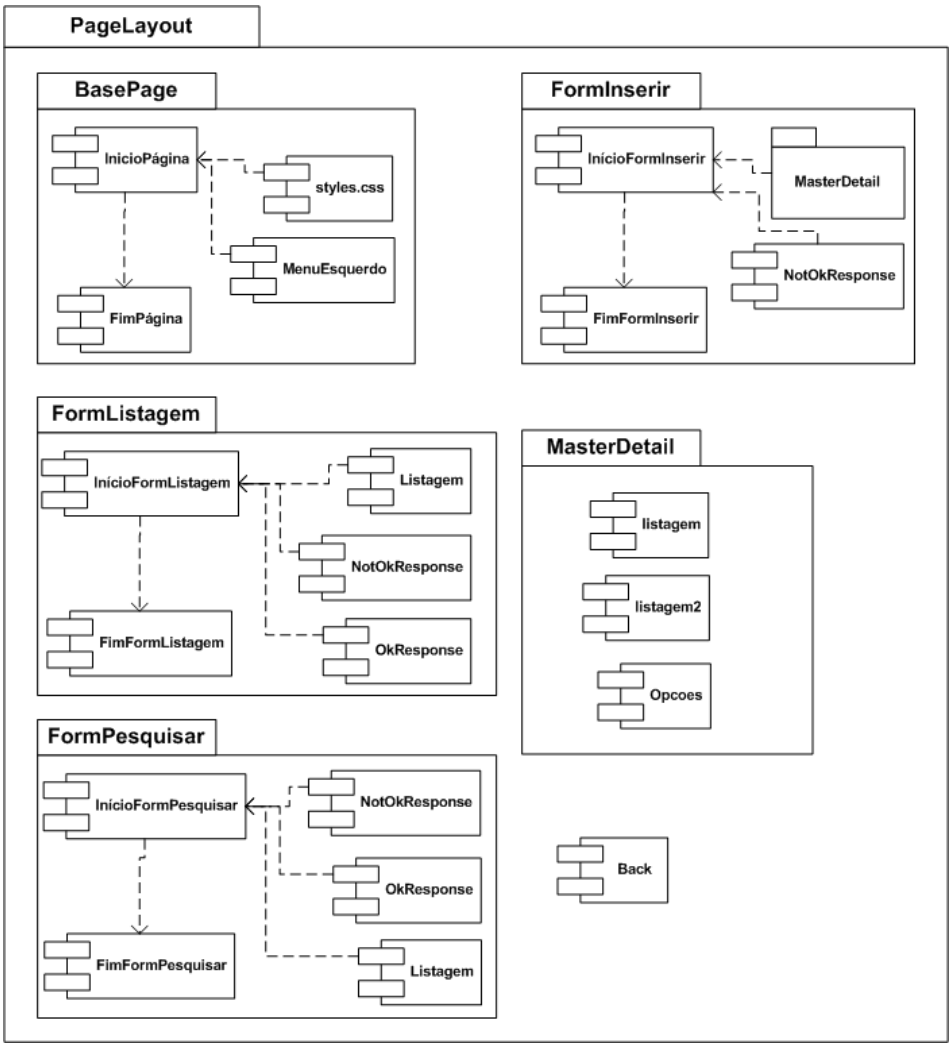

Figura 4: Package pagelayout

Nota: Neste diagrama os componentes com o mesmo nome, apesar de estarem referenciados em packages diferentes, na prática são o mesmo componente. Só estão repetidos para simplificar a visualização do diagrama.

- **BasePage** Esta package define a estutura básica de uma página
  - **InícioPágina**
    Contém o cabeçalho da página base.

* **FimPágina**
  Contém o rodapé da página base.
* **MenuEsquerdo**
  Contém o menu que aparece em todas as páginas.
* **styles**
  Define os estilos para o sistema.

– **FormInserir**

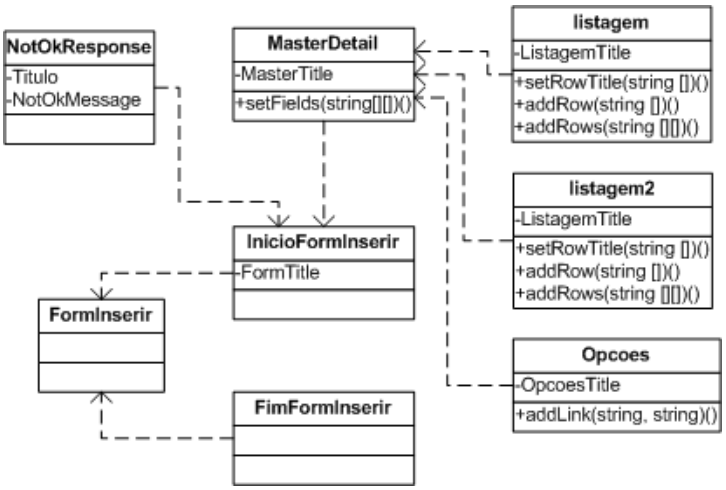

Figura 5: Modelo de classes do FormInserir

Este package define a estrutura básica de um formulário de inserção de dados. Usando esta package, é possível desenvolver formulários de inserção de dados uniformes normalizando assim o aspecto do sistema.

* **InícioFormInserir**
  Define o cabeçalho do formulário.
* **FimFormInserir**
  Define o rodapé do formulário.
* **MasterDetail**
  Esta classe é responsável por visualizar a informação detalhada (este componente é usado em qualquer parte do sistema onde seja necessários visualizar informação detalhada, p.ex: visualizar um docente ou um aluno).
  setFields(string[][]) - Este método recebe uma matriz de strings com a informação detalhada do objecto a visualizar (p.ex: field[0][0] = ”Sigla”, field[0][1]=”LES”, field[1][0]=”Nome”, field[1][1] = ”Laboratórios de Engenharia de Software”)

- **listagem**
  Esta classe serve para fazer listagens e, analogamente ao MasterDetail, é usada em qualquer para do sistema onde seja necessário fazer listagens.
  setRowTitle(string[]) - Define os títulos para as colunas da listagem.
  addRow(string[]) - Adiciona uma linha à listagem.
  addRows(string[][]) - Adiciona muitas linhas de uma vez.

- **listagem2**
  Esta classe é uma réplica da listagem.
- **Opcoes**
  Esta classe serve para associar opções ao MasterDetail (p.ex: Quando se visualiza o MasterDetail de uma disciplina é necessário colocar algumas opções associadas como alterar, apagar, etc)
  addLink(string, string) - Adiciona um link às opções (p.ex: addLink("Alterar disciplina", "index.aspx?go= disciplinas_registar& sifeup=EIC4100").

* **NotOkResponse**
  Esta classe serve para transmitir os avisos de erro. É também usada em qualquer parte do sistema onde seja necessário transmitir erros.

– **FormListagem**

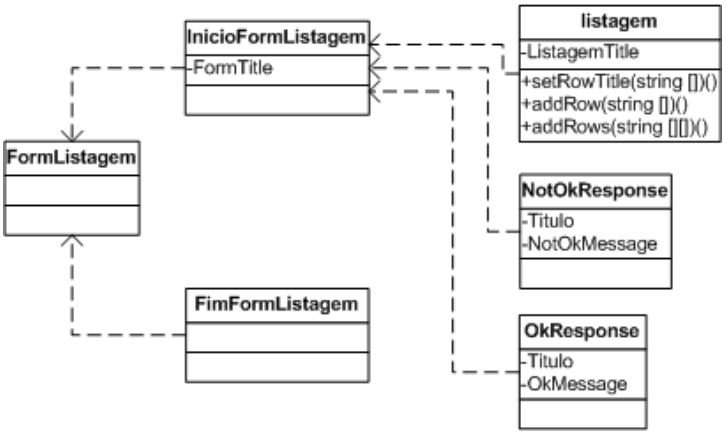

Figura 6: Modelo de classes do FormListagem

Este package define a estrutura de um formulário de inserção de dados com uma listagem associada (p.ex: no caso da inserção dos níveis bloom, não há necessidade de recorrer à visualização do MasterDetail, basta visualizar a listagem com todos os níveis bloom).

* **InícioFormListagem**
  Contém o cabeçalho do FormListagem
* **Listagem**
  Contém à listagem propriamente dita.
* **OkResponse** Este componente serve para transmitir ao utilizador as mensagens de "sucesso" na efectuação de operações (p.ex: "Nível bloom inserido com sucesso").
* **NotOkResponse** Este componente está explicado na descrição do MasterDetail.
* **FimFormListagem** Contém o rodapé do FormListagem

– **FormPesquisar** Este componente é bastante semelhante ao FormListagem. A única diferença reside na disposição dos componentes no layout.

– **MasterDetail** Este componente é está explicado na descrição do FormInserir.

– **Back** Este componente disponibiliza um sistema de navegação (muito simples, apenas com os links retroceder e avançar) nas páginas.

4.2.2 Configurações básicas

- Camada de Interface

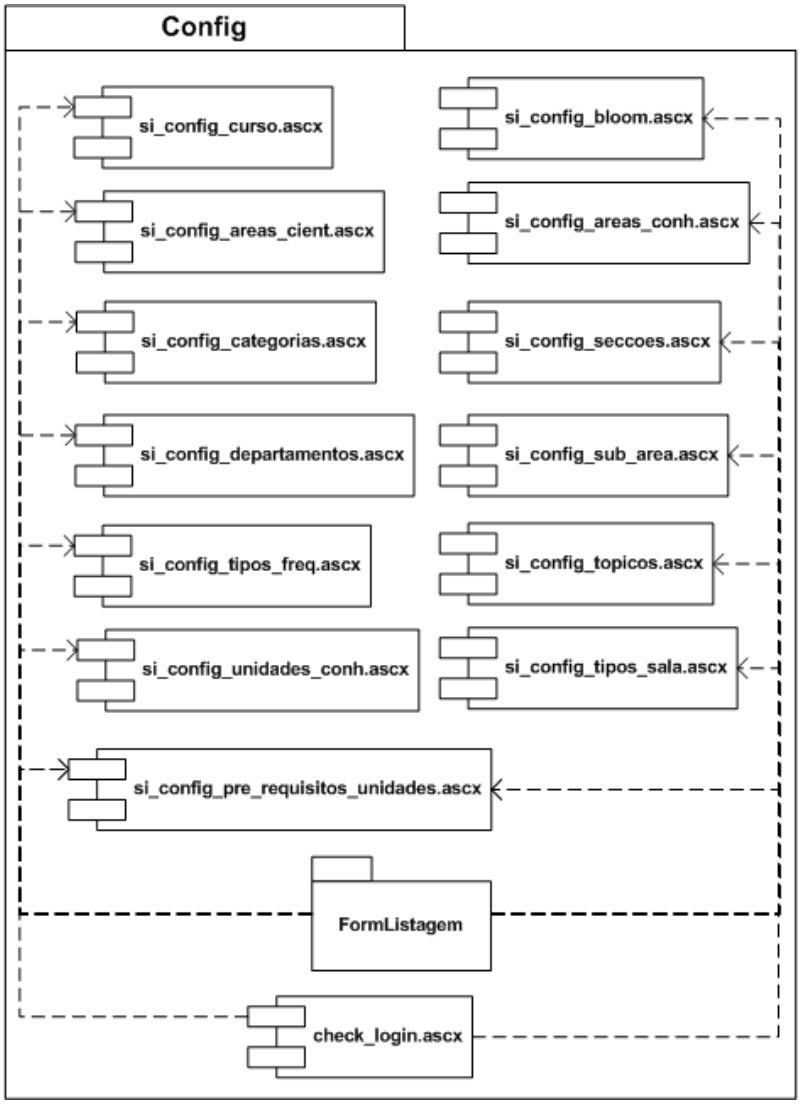

Figura 7: Package Config

Os casos de uso implementados neste componente são os seguintes:

- **areas_cientificas**
- **areas_conhecimento**
- **departamentos**

- **niveis_bloom**
- **pre_requisitos**
- **unidades_conhecimento**
- **topicos_conhecimento**
- **tipos_salas**
- **tipos_frequencia**
- **nivel_bloom**
- **categorias**

Para todos estes casos de uso a interface tem um formulário de inserção, o qual é também usado para a alteração dos dados. Existem então um botão de inserção e um de alteração, os dois associados ao mesmo formulário. Cada um tem também associado um objecto *form* que é utilizado para listagens e para avisos de erro ou de submissão correcta.

De seguida são explicados os métodos que fazem parte da camada de interface. Cada um destes métodos irá ser apresentado de uma forma genérica, pois para todos os casos de uso das Configurações básicas os métodos são semelhantes, sendo modificado só o nome dos métodos. É de facto desnecessário descriminar todos os métodos se são todos semelhantes. Assim para facilitar a compreensão usamos a designação *config* para representar a configuração escolhida no menu.

- **cmd_inserir_Click(object sender, System.EventArgs e)** Este método é inicializado quando é pressionado o botão inserir no formulário de inserção da configuração que selecionamos no menu. Quando é feita esta accão é chamado o método adicionar_config() do componente da lógica de negócio configuracoes_basicas.asmx. Este vai buscar todos os valores que estão nos elementos do formulário.

- **cmd_alterar_Click(object sender, System.EventArgs e)** Este método é inicializado quando é pressionado o botão alterar no formulário de inserção da configuração selecionada. De seguida é executado o método alterar_config() do componente da lógica de negócio configuracoes_basicas.asmx. A própria interface não permite a alteração da sigla de cada uma das configurações por uma questão de organização e de bom senso (são chaves primárias).

– **listar_config()** Este método inicializa os títulos da listagem do objecto *form* e chama o método de listar os dados da componente lógica de negócio configuracoes_basicas.asmx.

Algumas das configurações, nomeadamente areas_cientificas, areas_conhecimento, pre_requisitos, unidades_conhecimento e topicos_conhecimento, tem métodos para preencher as *comboboxes* no formulário da interface. Estes métodos designam-se por preencher_dados_a_preencher().

Em relação a esta camada convém ainda frisar que o *ASP.NET* usa uma técnica muito interessante, o *postBack*. Então quando carregamos num botão de um formulário (adicionar ou alterar) a variável *IsPostBack* estará sempre instânciada com o valor *TRUE*. Nas nossas listagens (todas visto ser um objecto usado por todos os casos de uso) de dados da base de dados, existe sempre um link para apagar e um link para alterar para cada linha da listagem. Ao carregarmos nos links essa variável *IsPostBack* irá ser *FALSE* e assim tratamos dessas acções logo no método Page_Load(), questionando se a página veio de postBack ou não. Repare-se que então para adicionar, apagar, alterar e listar usamos sempre o mesmo ficheiro. Desta forma garantimos robustez na aplicação e um código legível e fiável.

- Camada da lógica de negócio

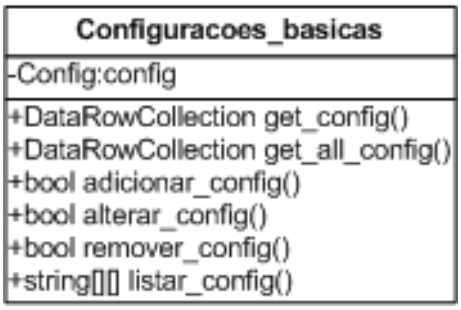

Figura 8: Classe Configuracoes_basicas

Para todos os casos de uso existem métodos que fazem pedidos ao Serviço de dados sobre a configuração em questão. Análogamente à camada de Interface, nesta camada os métodos não diferem muito de caso de uso para caso de uso, simplesmente são alterados os nomes dos métodos da lógica de negócio e dos métodos do Serviço de dados que são chamados. Para isso usàmos novamente a sigla *config* generalizando os métodos criados. Então para um dos casos de uso descritos

anteriormente (secção 4.2.2) foram definidas métodos com os seguintes propósitos:

- **get_config()** Este método retorna os dados relativos a uma determinada configuração básica, sobre a forma de um objecto do tipo *DataRowCollection*. Chama por sua vez um método do *webservice Config.asmx* de forma a aceder aos dados contidos na base de dados.
- **get_all_config()** Este método retorna todas os dados de todos os elementos de uma determinada configuração básica, sobre a forma de um objecto do tipo *DataRowCollection*. Chama por sua vez um método (*Web Method*) do *webservice Config.asmx*, chamado também get_all_config() de forma a aceder aos dados contidos na base de dados.
- **adicionar_config()** Método que recebe todos os dados necessários para a inserção. Verifica primeiro se a sigla já existe e só se não existir é que deixa inserir na base de dados. Insere através do método adicionar_config() do Serviço de dados.
- **alterar_config()** O método de inserção na base de dados verifica primeiro se a sigla já existe e só se não existir é que deixa inserir na base de dados. Insere através do método adicionar_config() do Serviço de dados.
- **remover_config()** Análogamente aos outros métodos executa um método do Serviço de dados que remove os dados da base de dados. O método do Serviço de dados é o remover_config() a.
- **listar_config()** Este método usa o get_all_config() para obter um objecto *DataRowCollection* de forma a poder retornar para a camada de interface um empharray string [][] com os dados. Esta como ja foi referido usa o objecto *listagem* para listar os dados.

Convém referir que o único caso de uso desta package que requeriu algum cuidado especial foi o caso da inserção de pré-requisitos. Isto porque o pré-requisito cria dependência entre as unidades e por isso não convém criar conflitos. O problema não ficou resolvido, mas de qualquer forma não deixamos inserir pré-requisitos quando a unidade é a mesma, nem quando o pré-requisito inverso já existir. Este "bug" conhecido está explicado com melhor detalhe em 4.5.1.

- Camada do serviço de dados

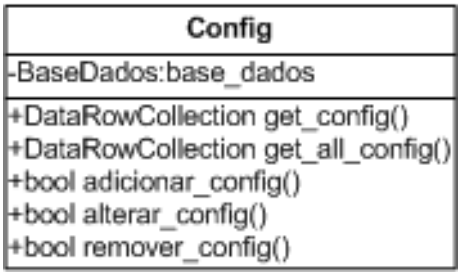

Figura 9: Classe Config

Esta camada é a que recorre à linguagem SQL para aceder à base de dados. Este *Web Service* usa a classe *BaseDados.cs* para aceder à base de dados Oracle. Usando a mesma terminologia das camadas anteriores iremos descrever de uma forma genérica os métodos que são repetidos por todos casos de uso.

- **get_config()** Este método retorna os dados relativos a uma determinada configuração básica, sobre a forma de um objecto do tipo *DataRowCollection*. Chama por sua vez um método da classe *BaseDados* de forma a aceder aos dados contidos na base de dados, que neste caso usa é o *set_sql()*.
- **get_all_config()** Este método retorna todas os dados de todos os elementos de uma determinada configuração básica, sobre a forma de um objecto do tipo *DataRowCollection*. Como o caso de uso anterior executa um *set_sql()* da classe *BaseDados* com o comando SQL que seleciona todos os dados requeridos.
- **adicionar_config()** Método que recebe todos os dados necessários para a inserção vindos da Lógica de Negócio. Depois da verificação da sigla efectuada pela Lógica de negócio este método insere na base de dados através de um comando SQL que é executado como argumento de um método da classe *BaseDados* chamado *exec_sql()*.
- **alterar_config()** Método que recebe todos os dados necessários para a alteração vindos da Lógica de Negócio. Através de um comando SQL que é executado como argumento de um método da classe *BaseDados* chamado *exec_sql()* faz o *UPDATE* da configuração selecionada.
- **remover_config()** Análogamente aos outros métodos executa um comando sql, que remove os dados, e que é argumento do método *exec_sql()*.

4.2.3 Alunos

- **Camada de interface**

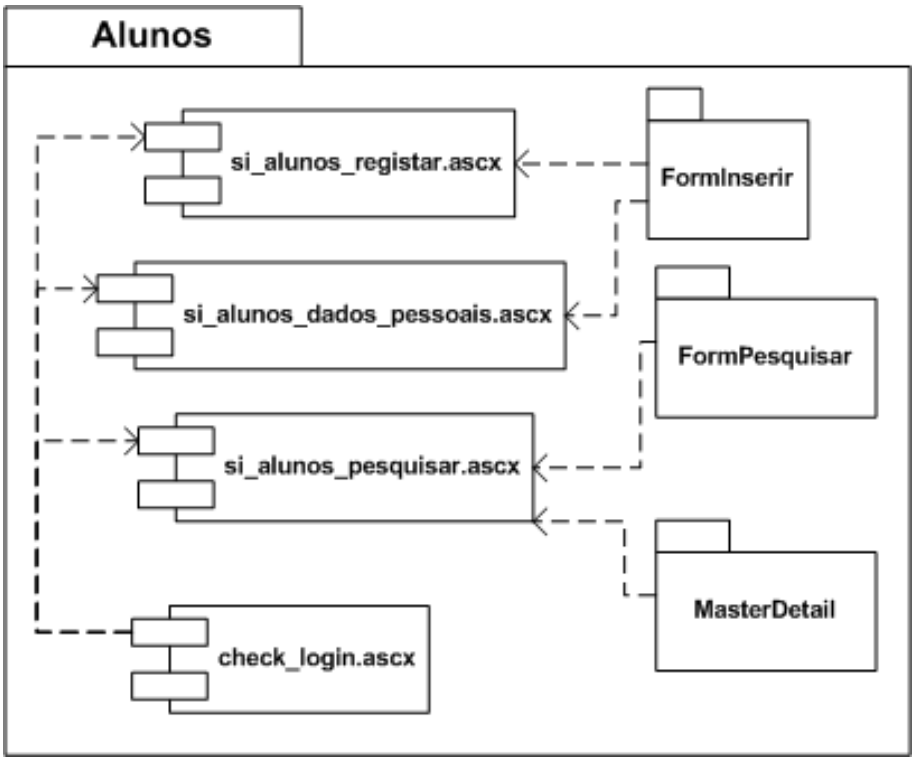

Figura 10: Package Alunos

Todos os componentes da secção da configuração dos alunos seguem praticamente a mesma arquitectura dos descritos anteriormente nas outras secções, visto serem baseados também em inserções, alterações, pesquisas e remoções.

- **si_alunos_registar**
  Este componente possui dois métodos que podem ser chamados dependendo do "contexto". Ao carregar a página, são analisados os valores vindos por *Get* e a variável *IsPostBack*. Dependendo destes, é visualizado um botão ou outro.
  - **cmd_inserir_Click()**
    Este método é executado quando é pressionado o botão de inserir. Envia os dados do formulário para o método `adicionar_aluno()` da lógica de negócio e recebe o resultado

da operação de inserção. Em função do resultado, mostra a mensagem de erro ou de sucesso respectiva.

* **cmd_alterar_Click()**
  Este método é relativo ao botão de alterar. Envia os dados do formulário para o método `alterar_aluno()` da lógica de negócio e recebe o resultado da operação de alteração. Dependendo do resultado, mostra a mensagem de erro ou de sucesso respectiva.

– **si_alunos_pesquisar**

* **cmd_pesquisar_Click()**
  Este método é executado quando é pressionado o botão de pesquisar. É chamado o método `pesquisar_alunos()`, da lógica de negócio e é devolvido um *array de strings* com o resultado, dependendo dos dados inseridos para pesquisa. É então feita a listagem dos resultados.
* **alterar**
  Embora não existam métodos para apagar ou ver mais informações de um aluno, quando é efectuada uma pesquisa, existe a possibilidade de, através de um *link*, efectuar estas operações. A opção tomada é verificada no método `Page_Load()` de forma análoga à descrita na parte de registar. Para apagar um aluno, é chamado o método `remover_aluno()`. Para ver mais informações, é chamado o método `listar_aluno()` e é feita a listagem. Ambos os métodos pertencem à camada da lógica de negócio.

– **si_alunos_matricular**

* **listar_matriculas(string codigo)**
  Este método lista as matriculas do aluno.

– **dados_pessoais**
  Este componente é praticamente igual à funcionalidade de alterar descrita anteriormente, com a particularidade de apenas dizer respeito ao aluno que está a utilizar o sistema.

- **Camada da lógica de negócio**

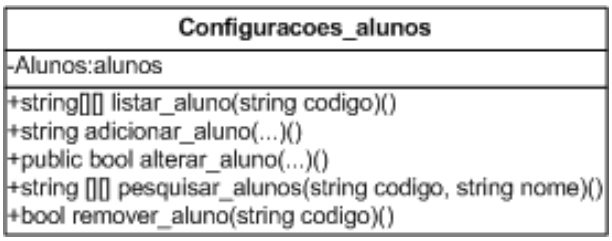

Figura 11: Classe Configuracoes_alunos

  - **Configuracoes_alunos**

    * **listar_aluno()**
      Este método recebe um código de aluno, chama o `get_aluno()` do serviço de dados, "transforma" a *DataRowCollection* recebida numa *matriz de strings* e retorna-a.
    * **adicionar_aluno()**
      Recebe os dados a adicionar, verifica se já existe o aluno, invocando `get_aluno()` e também se o aluno já existe na tabela *Users* através do `get_user()`. De seguida tenta inserir o aluno, chamando o método `adicionar_aluno()`, e retorna uma *string* dependendo do resultado. Os métodos chamados são referentes à camada do serviço de dados.
    * **alterar_aluno()**
      Recebe os dados a alterar, invoca o método `alterar_aluno()`, do serviço de dados, e retorna o valor lógico referente ao sucesso da operação de alteração.
    * **pesquisar_aluno()**
      Recebe os dados a pesquisar, chama o método `get_alunos()` do serviço de dados, e retorna os dados na estrutura de uma *matriz de strings*.
    * **remover_aluno()**
      Recebe o código do aluno, invoca o método `remover_aluno` do serviço de dados e retorna o valor lógico referente ao sucesso da operação de eliminação.

- **Camada do serviço de dados**

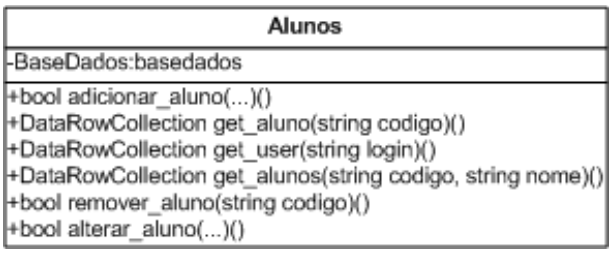

Figura 12: Classe Alunos

  - **Alunos**

    * **adicionar_aluno()**
      Recebe os dados do aluno, insere-o nas tabelas *Users* e *Alunos* enviando o código *SQL* correspondente através método `exec_sql()` da camada da base de dados. Retorna o valor lógico referente ao sucesso da operação de inserção.
    * **get_user()**
      Recebe o login do utilizador, chama o método `set_sql()` da base de dados com a interrogação *SQL* correspondente e retorna o resultado na forma de uma *DataRowCollection*.
    * **get_aluno()**
      É análogo ao método anterior, diferindo apenas na tabela interrogada.
    * **remover_aluno()**
      Recebe o código do aluno a remover, envia o código *SQL* correspondente através método `exec_sql()` da camada da base de dados e retorna o valor lógico referente ao sucesso da operação de eliminação.
    * **alterar_aluno()**
      Recebe os dados a alterar, envia o código *SQL* correspondente através do método `exec_sql()` da camada da base de dados e retorna o valor lógico referente ao sucesso da operação de alteração.

4.2.4 Docentes

- **Camada de interface**

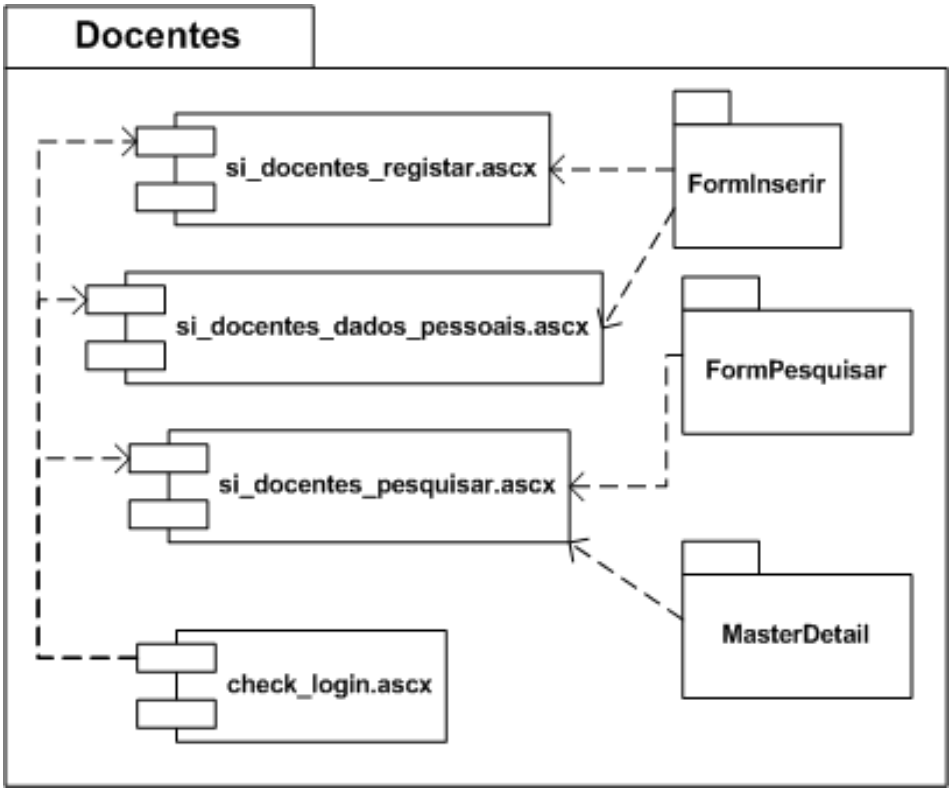

Figura 13: Package Docentes

– **si_docentes_registar.asmx** Este componente contém os métodos que permitem ao utilizador criar um novo docente. O componente possui a interface que contém os campos e os botões para a inserção dos dados. Também faz a visualização dos dados inseridos e das mensagens de erro que possam surgir. Este componente possui os seguintes métodos:

∗ **cmd_inserir_Click(object sender, System.EventArgs e)** Este método é inicializado quando é pressionado o botão inserir no *form* registar. Quando é feita esta acção é chamado o método `registar_docentes()` do componente configuracoes_docentes.asmx, que receberá os dados inseridos no formulário. Se esta operação for bem sucedida, é feita uma listagem dos dados através do método `get_master_detail()`

da camada da lógica de negócio. Se a inserção não for bem sucedida, é enviada para a interface uma mensagem com a descrição do erro.

  - **Page_Load(object sender, System.EventArgs e)** Este método é inicializado quando é pedida uma nova visualização. É feito um controle ao tipo de utilizador que quer aceder à página, aos objectos e campos que devem ou não ser visualizados. Aqui é controlada o pedido de referente à visualização de mais dados referentes a um dado docente e à eliminação dos dados de um dado docente, numa listagem.
  - **cmd_alterar_Click(object sender, System.EventArgs e)** Muito semelhante ao `cmd_inserir_Click()`, este método difere no facto de que chama o método `alterar_aluno()` do configuracoes_docentes.asmx e que efectua realmente a alteração do docente.

- **si_docentes_pesquisar.asmx** Este componente permite procurar os dados de docentes que estejam na base de dados. Preenchendo o campo nome, ou o campo código do formulário existente, é possível fazer uma pesquisa à base de dados. São também visualizadas as mensagens de erro que possam existir durante a pesquisa. Na listagem é possível alterar, eliminar ou obter todas as informações que existem na base de dados sobre o utilizador, já que na listagem só é disponibilizada a informação referente aos três primeiros campos da tabela na base de dados.
  - **pesquisar_button_Click(object sender, System.EventArgs e)** Após pressionar o botão da pesquisa o método é inicalizado e é feita uma pesquisa, chamando o método `listar_docentes()` da camada de lógica de negócio. Se conseguir encontrar valores para a pesquisa é chamado uma função que lista os resultados da pesquisa (função essa que existe no package Form-Pesquisar), caso contrário, é envidad uma mensagem de erro através do objecto not_ok_response.
  - **Page_Load(object sender, System.EventArgs e)** Este método é muito semelhante ao existente no componente anterior, excepto que contém o código que trata a chamada referente à alteração dos dados do docente.
- **si_docentes_dados_pessoais.asmx** Este método é muito semelhante ao anterior, excepto no facto de que o docente apenas pode alterar os dados a que tem acesso.

- **Camada da lógica de negócio**

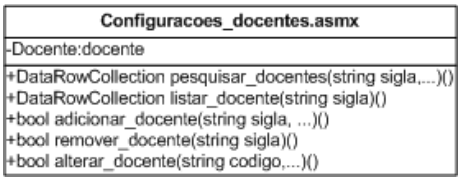

Figura 14: classe correspondente à lógica de negócio dos docentes

  - **configurações_docentes.asmx** Este componente comporta toda a lógica de negócio referente ao docente. Os métodos retornam os valores pedidos ou as mensagens referentes ao sucesso dos pedidos.
    * listar_docente(string sigla) Este método constrói uma matriz na qual preenche as linhas com os dados dos docentes que têm a sigla igual à pedida. Em cada coluna encontra-se um campo da tabela. As duas ultimas colunas da matriz referem-se à alteração, remoção ou visualização de mais dados referentes aos dados.
    * pesquisar_docentes(string sigla) Método que faz a listagem de todos os dados do docente.
    * adicionar_docente(string codigo,...) Método utilizado pela camada da Lógica de Negócio para fazer o registo dos dados na base de dados. Este método recebe variáveis cujos valores irão ser introduzidas na base de dados. A camada da Lógica de Negócio, através deste método, verifica se já existem docentes com código ou login igual ao que está a ser fornecido pela camada de interface, testando igualmente se realmente existe a sala que o utilizador está a fornecer. Retorna uma mensagem de erro caso algum destes casos aconteça. Se tudo correr bem, os dados serão passados à camada de serviço de dados, sendo fornecida uma mensagem de erro caso algo anormal aconteça.
    * remover_docentes(string sigla) Este é um método muito simples que apenas passa a sigla do docente a remover à camada de serviço de dados.
    * alterar_docente(string codigo,...) Este método é exactamente igual ao `registar_docentes()`, excepto no facto de que chama o método `alterar_docente()` da camada de serviço de dados, em vez do método `adicionar_docente()` da camada de serviço de dados.

- **Camada do serviço de dados**

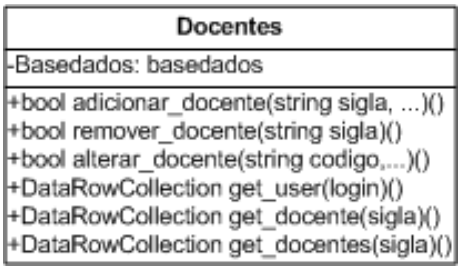

Figura 15: classe correspondente à lógica de negócio dos docentes

  - get_docente(String sigla) Este método recebe a sigla do docente, chama o método `set_sql()` da base de dados com a interrogação *SQL* correspondente e guarda os valores que esta retorna num DataRowCollection e por sua vez retorna esta estrutura.
  - get_all_docentes() Chama o método `set_sql()` da base de dados com a interrogação *SQL* correspondente (sem restrições) e guarda os valores que esta retorna num DataRowCollection e por sua vez retorna esta estrutura.
  - remover_docente(string sigla) Metodo que envia o código *SQL* correspondente através método `exec_sql()` da camada da base de dados e retorna um booleano. TRUE se conseguiu correr a string e FALSE em caso constrário.
  - registar_docentes(string codigo, string sigla,...) Método que recebe os dados do docente, insere-o nas tabelas *Users* e *Alunos* enviando o código *SQL* correspondente através método `exec_sql()` da camada da base de dados e retorna um booleano. TRUE se conseguiu correr a string e FALSE em caso constrário.
  - alterar_docente(string codigo, string sigla, ...) recebe os dados a alterar, envia o código *SQL* correspondente através do método `exec_sql()` e retorna um booleano. TRUE se conseguiu correr a string e FALSE em caso contrário.

4.2.5 Disciplinas

- **Camada de interface**

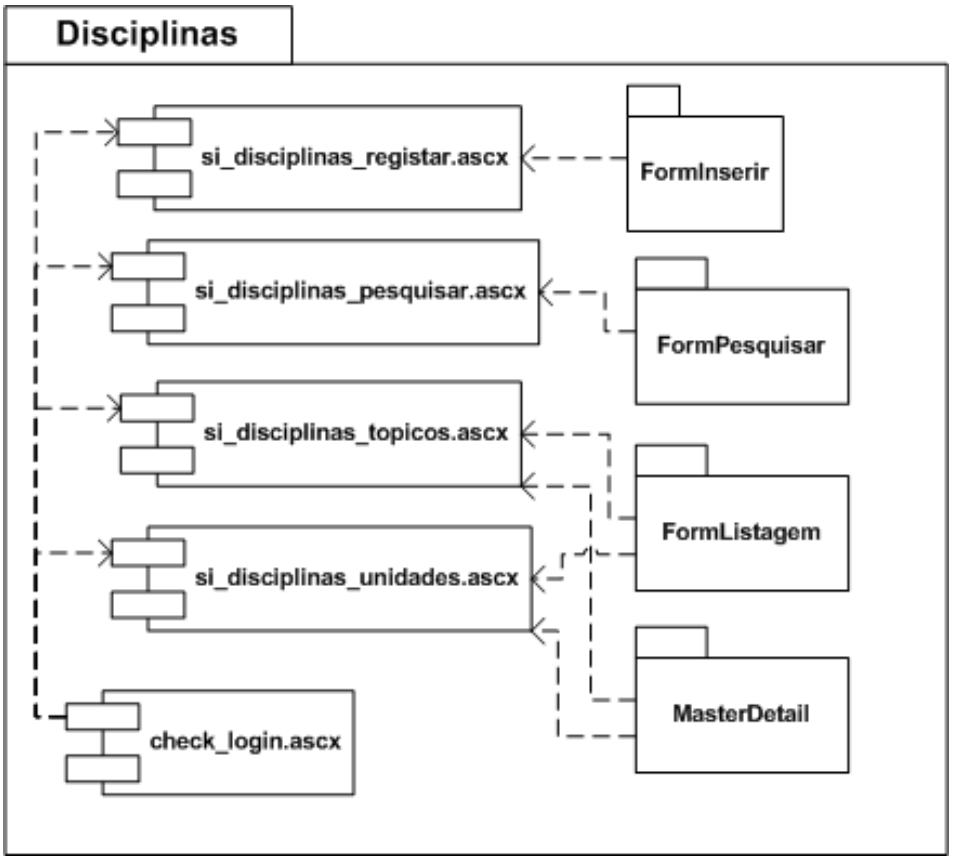

Figura 16: Package Disciplinas

Como se pode ver pelo diagrama, todos os componentes usam o check_login. Esse componente é responsável por verificar se o utilizador é válido (caso não o seja, invoca o sitema de login).

- **si_disciplinas_registar.ascx**
  Este componente é responsável pelo formulário de inserção e alteração de disciplinas. Segue-se uma breve descricao dos métodos deste componente.
  - **void Page_Load(object sender, System.EventArgs e)**
    Inicializa a página construindo o formulário. Este método é também responsável por reconhecer se a situação, isto é, se

é uma situação de inserção de uma nova disciplina, ou se é uma situação de alteração de disciplina, e tomar inicializar o formulário dependendo da situação.

* **void cmd_inserir_Click(object sender, System.EventArgs e))**
  Invoca o método **adicionar_disciplina(...)** da lógica de negócio e caso a lógica de negócio retorne falso, este método activa o objecto **NotOkResponse** para transmitir uma mensagem de erro. Caso a lógica de negócio retorne verdadeira, este método chama o método **show_master_detail** descrito no próximo item.
* **void show_master_detail(string sifeup)**
  Este método inicializa o objecto **MasterDetail** e comunica-lhe a informação a visualizar.
* **void cmd_alterar_Click(object sender, System.EventArgs e)**
  Este método é semelhante ao **cmd_inserir_Click(object sender, System.EventArgs e)** com a diferença de em vez de chamar a função **adicionar_disciplina(...)** chama a função **alterar_disciplina(...)**.

– **si_disciplinas_pesquisar.ascx**
  Este componente serve para visualizar informação sobre as disciplinas.

* **void Page_Load(object sender, System.EventArgs e)**
  Este método é responsável por inicializar a página (listagem, visualização de detalhes de disciplinas, etc).
* **void listar_unidades(string sifeup)**
  Lista as unidades de conhecimento de uma disciplina. Para isso recorre ao método **get_unidades()** da lógica de negócio.
* **void listar_topicos(string sifeup)**
  Este método é semelhante ao anterior, mas invoca da lógica de negócio o método **get_tópicos()**

– **si_disciplinas_topicos.ascx**

* **void Page_Load(object sender, System.EventArgs e)**
  Este método encarrega-se de inicializar a página (formulário, listagem, etc).
* **void cmd_gravar_Click(object sender, System.EventArgs e)** Este método invoca **adicionar_topico()** da lógica de negó-

cio e em função do retorno da lógica de negócio, activa ou não a mensagem de erro.

  * **void listar_topicos(string sifeup)** Este método faz a listagem dos tópicos já atribuídos à disciplina que se está a visualizar. Para isso, recorrer ao método **get_topicos()** da lógica de negócio.

- **si_disciplinas_unidades.ascx**
  Este componente é semelhante ao componente anterior, mas em vez de tratar a informação dos tópicos de conhecimento trata a informação das unidades de conhecimento.

- **Camada da lógica de negócio**
  Esta classe trata de toda a lógica de negócio relacionada com as disci-

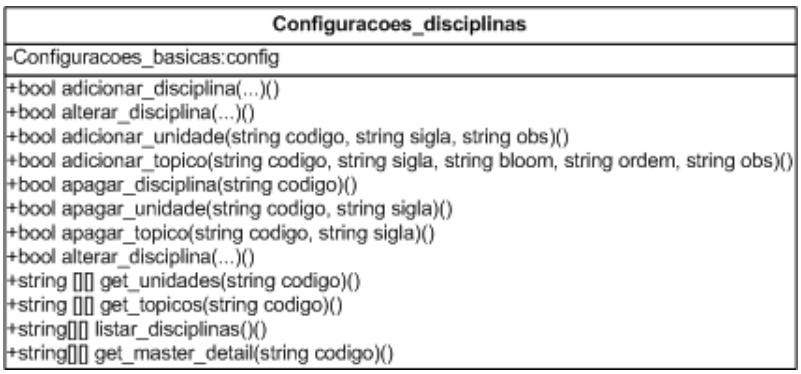

Figura 17: Classe Configuracoes_disciplinas

plinas. Segue-se então uma breve descrição dos métodos desta classe.

  - **bool adicionar_disciplina(...[1])**
    Este método invoca o serviço de dados a fim de inserir uma disciplina na base de dados.
  - **bool alterar_disciplina(...)**
    Este método é semelhante ao anterior.
  - **bool adicionar_unidade(string codigo, string sigla, string obs)**
    Este método invoca o serviço de dados a fim de atribuir uma unidade de conhecimento identificada pelo argumento "sigla" a uma disciplina identificada pelo "codigo" do sifeup.
  - **bool adicionar_topico(string codigo, string sigla, string bloom, string ordem, string obs)**
    Este método é semelhante ao anterior, com a diferença de que para os tópicos é necessário enviar também um nível "bloom" e uma "ordem".
  - **bool apagar_disciplina(string codigo)**
    Invoca do serviço de dados a função que elimina da base de dados uma disciplina.
  - **bool apagar_unidade(string codigo, string sigla)**
    Apaga um unidade de conhecimento atribuida a uma disciplina. Para isso, este método invoca o serviço de dados enviando-lhe o código da unidade e o código da disciplina de conhecimento.

[1]Não especificamos os argumentos desta função por serem muitos (dezanove).

- **bool apagar_topico(string codigo, string sigla)**
  Este método é semelhante ao anterior, mas funciona para os tópicos de conhecimento.
- **bool get_unidades(string codigo)**
  Retorna todas as unidades de conhecimento de uma determinada disciplina (identificada pelo código).
- **bool get_topicos(string codigo)**
  Retorna todos os tópicos de conhecimento atribuidos a uma disciplina (identificada pelo código).
- **bool listar_disciplinas()**
  Devolve uma listagem (ordenada) das várias disciplinas na base de dados.
- **bool get_master_detail(string codigo)**
  Retorna os detalhes de uma disciplina (identificada pelo código).

- **Camada do serviço de dados**

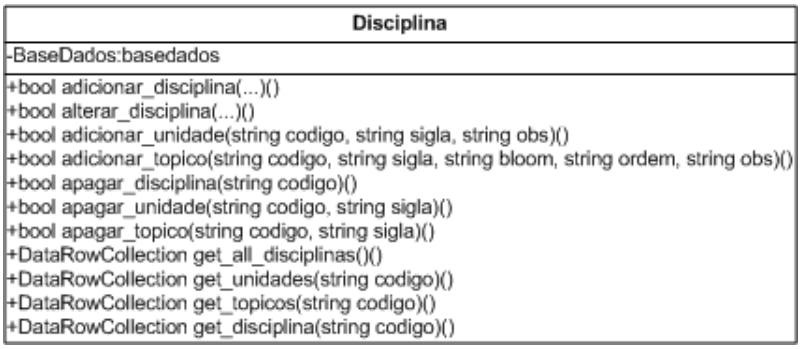

Figura 18: Classe Disciplina

Esta classe efectua o serviço de dados para as disciplinas, isto é, interpreta pedidos vindos da lógica de negócio e comunica (através de SQL) com a base de dados por forma a realizar esses pedidos.

- **bool adicionar_disciplina(..)**
  Este método recolhe os dados vindos da lógica de negócio e efectua a comunicação com a base de dados por forma a adicionar uma nova disciplina ao sistema.
- **bool alterar_disciplina(...)**
  Analogamente ao método anterior, também este recolhe os dados da lógica de negócio e comunica com a base de dados, com a diferença que este método, ao invés de inserir uma nova disciplina, altera uma disciplina já existente.
- **bool adicionar_unidade(string codigo, string sigla, string obs)**
  Atribui a uma determinada disciplina uma nova unidade de conhecimento.
- **bool adicionar_topico(string codigo, string sigla, string bloom, string ordem, string obs)**
  Atribui a uma disciplina um tópico de conhecimento
- **bool apagar_disciplina(string codigo)**
  Recebe da lógica de negócio o código da disciplina a apagar, e efectua a comunicação com a base de dados por forma a realizar o pedido.
- **bool apagar_unidade(string codigo, string sigla)**
  Este método elimina uma atribuição de uma determinada unidade a uma disciplina.

- **bool apagar_topico(string codigo, string sigla)**
  Este método funciona de forma semelhante ao anterior.
- **DataRowCollection get_all_disciplinas()**
  Retorna uma colecção de de registos contendo a informação de todas as disciplinas.
- **DataRowCollection get_unidades(string codigo)**
  Envia para a lógica de negócio uma colecção de registos contendo as várias unidades de conhecimento atribuidas a uma determinada disciplina identificada pelo código.
- **DataRowCollection get_topicos(string codigo)**
  Semelhante a método anterior, mas funciona para os tópicos de conhecimento.
- **DataRowCollection get_disciplina(string codigo)**
  Retorna para a lógica de negócio um registo com as informações completas de uma disciplina.

4.2.6 Salas

Este será o componente mais simples, pois apenas trabalha com uma tabela. Semelhante aos componentes anteriores, possui operações de inserção, alteração, pesquisa e remoção.

- **Camada de interface**

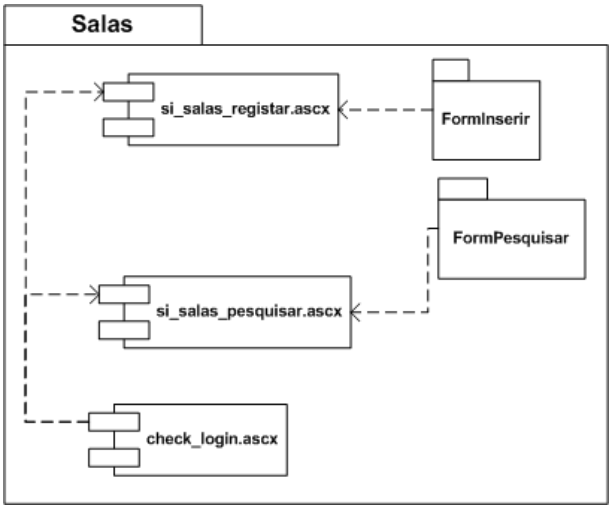

Figura 19: Package Salas

  - **si_salas_registar()**
    Este componente permite inserir e alterar a informação existente sobre as salas e Utiliza o master detail para controlar a página. Segue-se uma descrição dos métodos mais importantes:
    * cmdInserir_Click()
      Chama o método `adicionar_sala()` da camada da lógica de negócio quando é pressionado o botão Inserir. caso o método retorne erro, é enviada uma mensagem para o utilizador. Caso contrário é feita a listagem dos dados insiridos com um aviso.
    * cmdAlterar_Click()
      Chama o método `alterar_sala()` da camada da lógica de negócio quando é pressionado o botão Alterar. caso o método retorne erro, é enviada uma mensagem de erro para o utilizador. Caso contrário é feita a listagem dos dados alterados.
    * Page_Load()
      Este método faz a listagem das várias salas segundo o formato

das páginas anteriores, bem como a verificação da variável IsPostBack.

– **si_salas_pesquisar()**
Este componente controla e faz a listagem das pesquisas do dados referentes as salas. segue-se uma descrição dos seus métodos.

* cmd_pesquisar_Click()
Este método chama o método `pesquisar_salas()` da camada de lógica de negócio. caso o método retorne valores, estes são listados. Caso contrário, é dada uma mensagem de erro.
* Page_Load()
Faz a verificação da variável Get.

• **Camada da lógica de negócio**

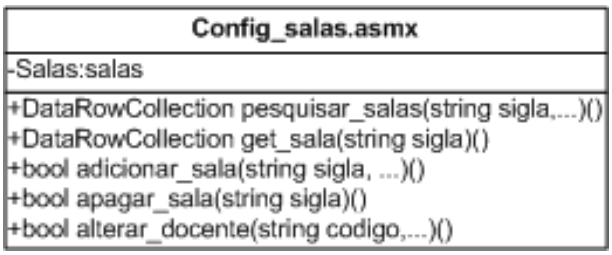

Figura 20: Classe Configuracoes_salas

– **Configuracoes_salas**

* **adicionar_sala()**
Este método chama o método `adicionar_sala()` da camada de serviço de dados, preenchendo com os valores recebidos pela camada de sistema. Retorna o valor de retorno do método adicionar_sala da camada de serviço de dados.
* **alterar_sala()**
Este método chama o método `alterar_sala()` da camada de serviço de dados, preenchendo com os valores recebidos pela camada de sistema. Retorna o valor de retorno do método `alterar_sala()` da camada de serviço de dados.
* **apagar_sala()**
Este método chama o método `apagar_sala()` da camada de serviço de dados. Retorna o valor de retorno do método `apagar_sala()` da camada de serviço de dados.

* **pesquisar_salas()** Chama o método `pesquisar_sala()` da camada de serviço de dados e insere os resultados numa matriz, retornando essa mesma matriz.
* **get_master_detail()** Este método guarda os valores de uma determinada sala - obtidos através do método `get_sala()` - numa matriz, retornando essa mesma matriz.

- **Camada do serviço de dados**

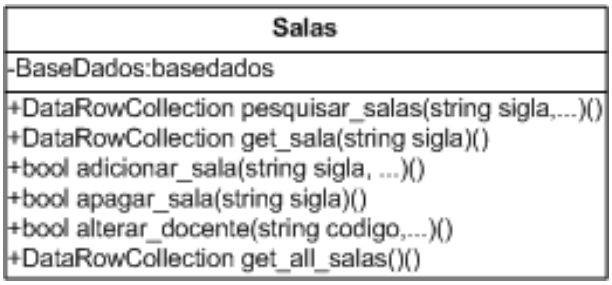

Figura 21: Classe Salas

– **Salas**

* **adicionar_sala()**
  Recebe os dados da sala, insere-os na tabela *Salas*, enviando o código *SQL* correspondente através método `exec_sql()` da camada da base de dados. Retorna o valor lógico referente ao sucesso da operação de inserção.
* **alterar_sala()**
  Recebe os dados a alterar, envia o código *SQL* correspondente através do método `exec_sql()` da camada da base de dados e retorna o valor lógico referente ao sucesso da operação de alteração.
* **get_sala()**
  Recebe a sigla da sala, chama o método `set_sql()` da base de dados com a interrogação *SQL* correspondente e retorna o resultado na forma de uma *DataRowCollection*.
* **get_all_salas()**
  Semelhante ao método anterior, retorna todas as salas existentes.

* **remover_sala()**
  Recebe a sigla da sala a remover, envia o código *SQL* correspondente através método `exec_sql()` da camada da base de dados e retorna o valor lógico referente ao sucesso da operação de eliminação.

4.2.7 Plano de estudos

- **Camada de interface**

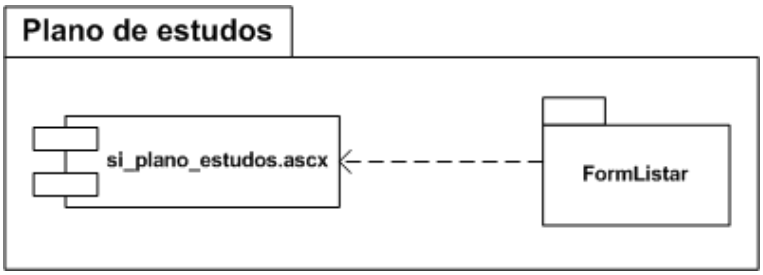

Figura 22: Package Plano de estudos

Esta camada utiliza o *WebService* do plano de estudos para obter uma listagem do plano de estudos do curso.
A listagem é visualizada através do uso do componente `FormListagem`.

- **Camada da lógica de negócio**

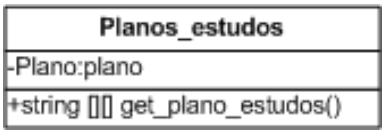

Figura 23: Classe Plano_estudos

  - **Plano_estudos**

    * **get_plano_estudos(string curso)**
      Este método chama o `get_plano(curso)` do serviço de dados, "transforma" a *DataRowCollection* recebida numa *matriz de strings* e retorna-a.

- **Camada do serviço de dados**

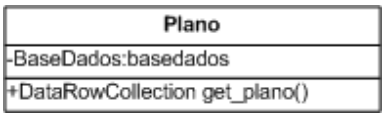

Figura 24: Classe Plano

  - **Plano**

    * **get_plano(string curso)**
      Este método faz a seguinte interrogação ao serviço de dados e retorna uma *DataRowCollection* com o resultado.
      "`SELECT anocurso, nome, areacientifica, semestre, horast, horastp, horasp, creditos, ects, optativa FROM disciplinas WHERE curso='"+curso+"' GROUP BY anocurso, semestre, nome, areacientifica, horast, horastp, horasp, creditos, ects, optativa ORDER anocurso, semestre`"

4.3 Realização de casos de utilização

Foram implementados **todos** os casos de uso descritos na especificação de requisitos.

4.3.1 Configurações Básicas

Nas configurações básicas, todos as inserções, alterções e remoções na base de dados são da responsabilidade do Gestor. Consequentemente estes dados estão protegidos pelo sistema de autentificação e todos os seguintes casos de uso verificam que tipo de utilizador está presente na sessão. A listagem está disponível para todos os utilizadores do sistema.

- Configurar departamentos

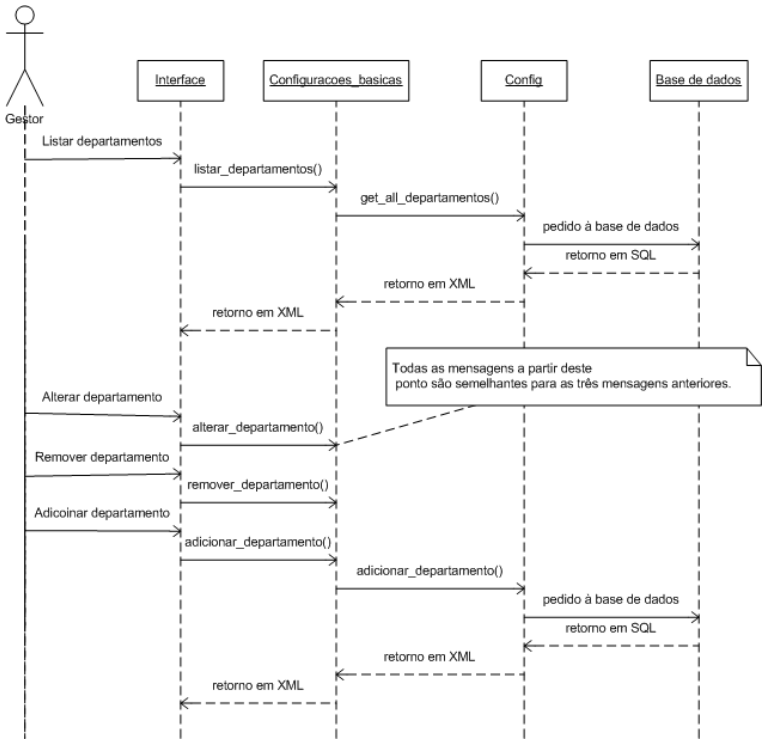

Figura 25: Diagrama de sequência relativo à configuração de departamentos

Este caso de uso é constituído por 4 funções primordiais: Adicionar, alterar, apagar e listar departamentos. A lógica de negócio é responsável por pedir a lista de todos os departamentos ao Serviço de Dados e assim listar todos os departamentos na interface.

- Configurar áreas científicas
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (ver 4.3.1) Devido à extensão dos nossos casos de uso, optámos por desenhar só os diagramas de sequência que acharmos extremamente necessários, ou seja que forem diferentes do configurar departamentos. Assim, a diferença entre o caso de uso da configuração dos departamentos e este caso de uso é só no nome das funções.

- Configurar cursos
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar áreas de conhecimento
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar tópicos de conhecimento
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar unidades de conhecimento
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar pré-requisitos de unidades
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar secções
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar sub-área de secções
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar tipos de salas
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar níveis de bloom
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar categorias dos docentes
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

- Configurar tipos de frequência
  A comunicação é efectuada exactamente da mesma forma que no caso configurar departamentos. (secção 4.3.1)

4.3.2 Docentes

Os registos e valores referentes aos docentes apenas podem ser alterados pelos gestores, excepto no que diz respeito aos dados pessoais. Sendo assim é feita uma verificação ao utilizador caso este queira aceder a uma destas areas de acesso restrito.

- Configurar docente

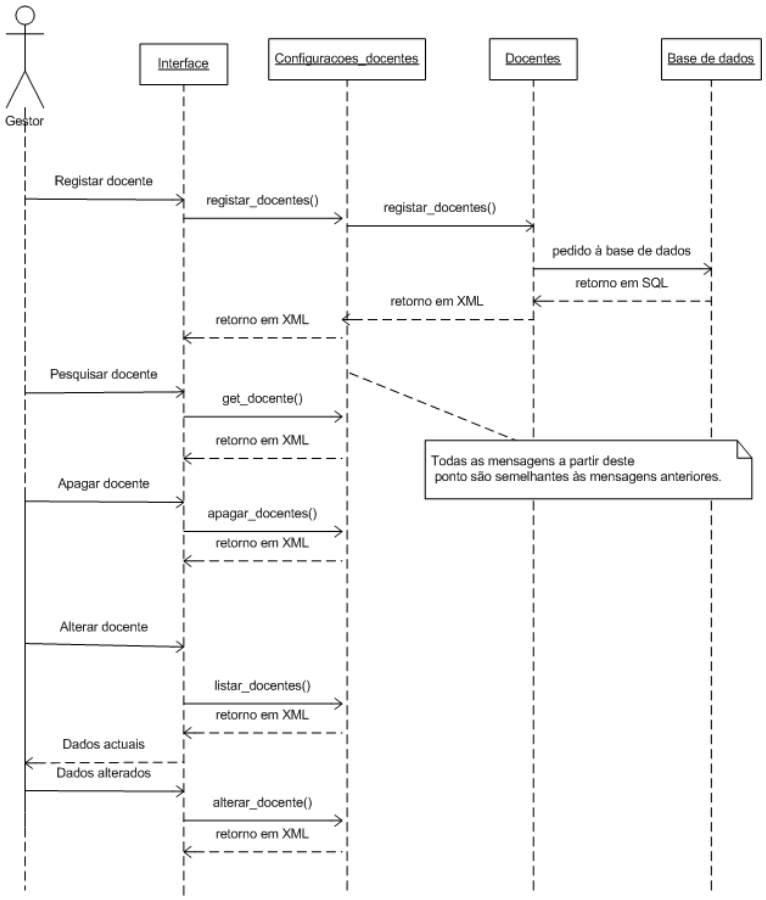

Figura 26: Diagrama de sequência relativo à configuração dos docentes

- Registar docente
  Ao preencher os dados e adicionar o docente, o utilizador recebe uma lista com os dados do docente. O utilizador pode então pedir para voltar a alterar os dados do docente ou então elimina-lo.

- Visualizar docente
  A visualização do docente é feita depois de uma pesquisa pela sigla e/ou pelo nome. É dada então uma lista com alguns dos dados correspondentes aos docentes que na base de dados correspondam aos parâmetros especificados. Nessa lista é dada a hipótese de podermos alterar os dados, apagar ou ainda visualizar em mais pormenor os dados do docente, podendo ai alterar os dados ou apaga-los.

- Configurar dados pessoais do docente

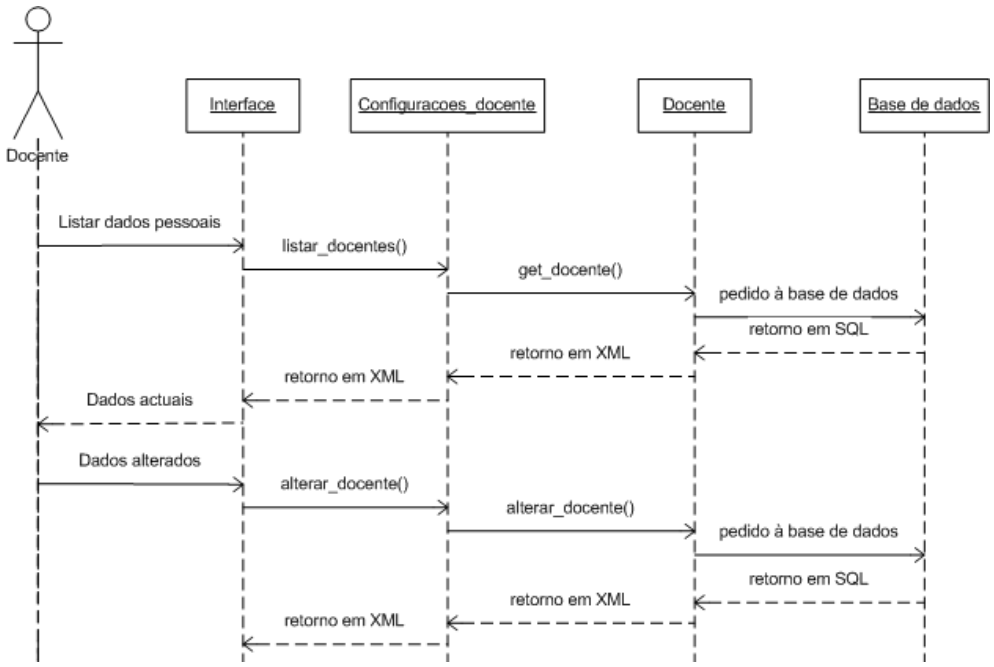

Figura 27: Diagrama de sequência relativo aos dados pessoais dos docentes

Aqui o docente pode configurar os seus dados pessoais. Assim, o docente ao registar-se, pede uma listagem dos seus dados e se o desejar, pode alterar os dados para os quais tem permissões para isso.

4.3.3 Alunos

A configuração dos alunos, também necessita da autenticação do utilizador, visto só o Gestor ter este tipo de privilégio. No entanto, existe uma funcionalidade da configuração dos alunos que diz respeito ao próprio aluno. Esta é a configuração dos seus dados pessoais.

- Configurar alunos

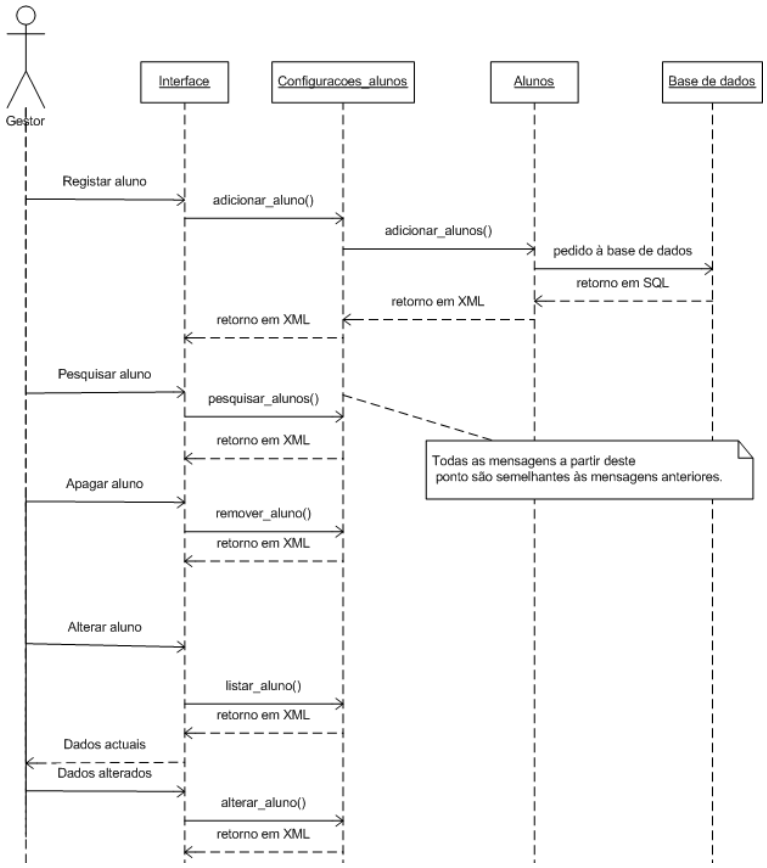

Figura 28: Diagrama de sequência relativo à configuração dos alunos

O processo de execução de todos estes métodos é basicamente o mesmo. A interface comunica com o *WebService* da lógica de negócio que, por sua vez, comunica com o do serviço de dados, o qual tem acesso à base de dados.

- Registar aluno
  Os dados do formulário de registo do aluno são enviados para a lógica de negócio, daí para o serviço de dados, onde é executada a inserção do aluno na base de dados. Além da sua inserção na tabela `Alunos`, também é criado um registo na tabela `Users` com o tipo de utilizador, para posteriormente este poder ser autenticado com um aluno pelo sistema.

- Visualizar aluno
  É efectuada uma pesquisa e, seguindo o mesmo método de comunicação com a base de dados descrito anteriormente, é feita uma listagem dos resultados, com a opção de ver mais opções, alterar ou apagar cada um dos alunos pertencentes ao resultado.

- Configurar dados pessoais

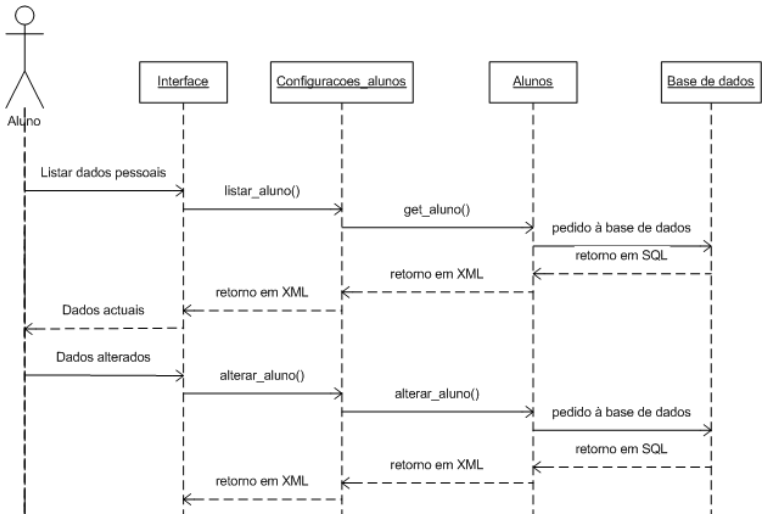

Figura 29: Diagrama de sequência relativo à configuração dos dados pessoais do aluno

O aluno tem a possibilidade de visualizar e alterar os seus dados pessoais, sendo o processo de comunicação semelhante ao anterior.

4.3.4 Salas

Resolvemos não fazer uma descrição detalhada destes casos, visto serem bastante semelhantes aos anteriores.

4.3.5 Disciplinas

Resolvemos não fazer uma descrição detalhada destes casos, visto serem bastante semelhantes aos anteriores.

4.3.6 Plano de estudos

- Visualizar plano de estudos

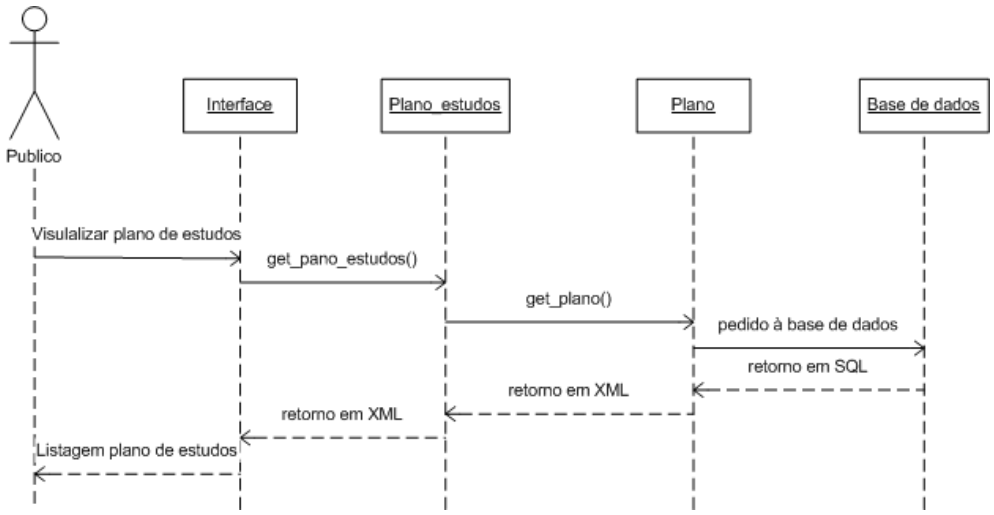

Figura 30: Diagrama de sequência relativo à visualização do plano de estudos

A visualização do plano de estudos é o nosso *WebService publico*. No entanto, além de retornarmos por *XML* o plano de estudos, também temos a possibilidade de fazer a sua visualização através do nosso sitema.
O processo de comunicação entre as camadas é idêntico aos descritos anteriormente.

4.4 Informação sobre testes de integração

Efectuámos testes de integração com apenas um dos outros grupos (S4). Estes testes foram a integração da interface, embora não tenham sido, na sua totalidade, efectuados com sucesso.
O nosso sistema como já foi referido (4.2.1) é bastante modular, sendo muito fácil aplicar novos *layouts* ao site todo em pouco tempo. Desta forma o "encaixe" do nosso sistema noutro site será de integração bastante rápida e facilitada.

4.5 Estado da implementação

Finalizado o projecto, realizámos com sucesso todos os requisitos, mínimos e não mínimos, por nós especificados ([MFPS02]).

4.5.1 "Bugs" conhecidos

Gostariamos de referir, um dos requisitos, que apesar de implementado não restringe a inserção de ciclos fechados de pré-requisitos de unidades. Ou seja, uma unidade não pode ser pré-requisito de outra, e ao mesmo tempo essa mesma ser pré-requisito da outra, como indicado na figura seguinte (4.5.1).

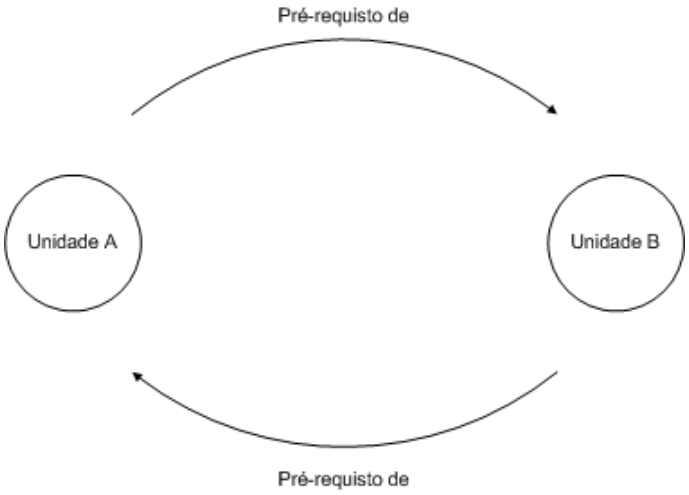

Figura 31: Exemplo de pré-requisito

Visto esta restrição ser simples de implementar, através do objecto de acesso à base de dados, e de consultas *SQL*, restringimos a inserção deste tipo de pré-requisitos. Mas por outro lado poderiam surgir ciclos enormes fechados de unidades (4.5.1), e assim tornou-se impossível a implementação desta restrição.

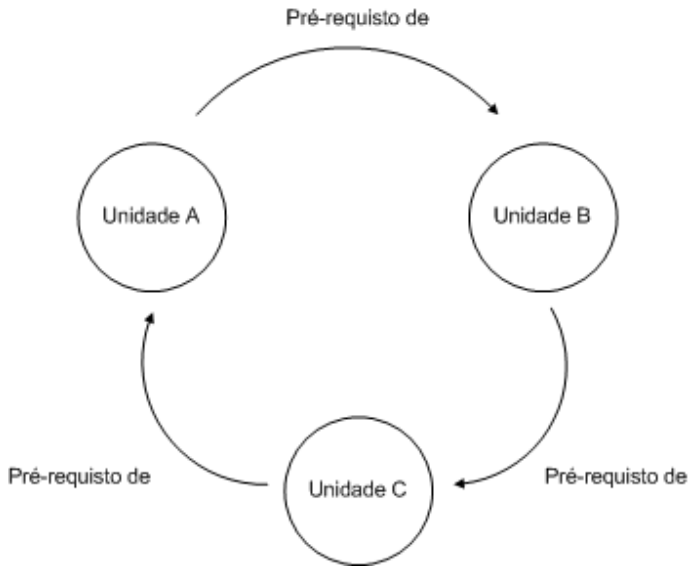

Figura 32: Exemplo de um ciclo fechado de várias unidades

5 Documentação de gestão de projecto

Nesta secção é apresentada uma estimativa do tempo gasto por cada elemento do grupo durante o desenvolvimento dos diferentes componentes do trabalho.

5.1 Tempo gasto em cada módulo

- **Interface e seus objectos**
  Decidimos fazer uma estruturação inicial dos objectos que iriam ser utilizados pelos vários módulos da nossa aplicação. Estes foram criados numa fase inicial do projecto, de forma a acelerar o desenvolvimento do mesmo numa fase posterior.
  A interface foi sendo melhorada à medida que o trabalho foi decorrendo.

Tarefa	Responsável	Duração (horas)
Interface e seus objectos	**José Fonseca**	**Total: 15**
Page Layout		6
Fomulários		4
Mensagens de erro		2
Master detail		1
Outros objectos		2

- **Configurações básicas**

Tarefa	Responsável	Duração (horas)
Configurações básicas	**André Moniz**	**Total: 25**
Áreas científicas		2
Áreas de conhecimento		1
Unidades de conhecimento		2
Categorias de docentes		2
Criar curso		2
Calendário escolar		2
Tópicos de conhecimento		1
Departamentos		2
Pré-requisitos		3
Tipos de frequência		1
Tipos de sala		3
Níveis bloom		1
Secções		1
Sub-área de secção		2

- **Configuração das Disciplinas**

Tarefa	Responsável	Duração (horas)
Disciplinas	**José Fonseca**	**Total: 17**
Criar disciplina		3
Visualizar disciplina		12
Alterar disciplina		2

- **Configuração dos Docentes**

Tarefa	Responsável	Duração (horas)
Docentes	**Miguel Sarmento**	**Total: 23**
Registar docente		16
Pesquisar docente		4
Alterar dados pessoais		2

- **Configuração dos alunos**

Tarefa	Responsável	Duração (horas)
Alunos	**Mário Pereira**	**Total: 19**
Registar aluno		9
Pesquisar aluno		7
Alterar dados pessoais		3

- **Configuração das salas**

Tarefa	Responsável	Duração (horas)
Salas	**José Fonseca**	**Total: 4**
Registar sala		3
Pesquisar sala		1

- **Redação do relatório**
  Não foram discriminados os tempos para cada um dos autores do relatório, visto este ter sido feito em paralelo. Indicamos apenas o tempo total gasto na realização do relatório.

Tarefa	Responsável	Duração (horas)
Relatório de desenvolvimento	**Todos**	**Total: 175**

5.2 Total de tempo gasto por cada elemento do grupo

Dado que os módulos foram desenvolvidos paralelamente, não existe uma grande distinção no tempo gasto entre os elementos do grupo.

Elemento	Trabalho total (horas)
André Moniz	69
José Fonseca	79
Mário Pereira	69
Miguel Sarmento	67

5.3 Comentários sobre o funcionamento do grupo

O grupo funcionou bastante bem. Isto foi devido não só a termos uma boa relação pessoal uns com os outros, mas principalmente, ao facto de ter sido efectuada, numa fase inicial, uma análise e estruturação de todo o trabalho. Como consequência, o desenvolvimento decorreu de uma forma organizada, o que nos permitiu criar os diferentes módulos paralelamente.

6 Conclusão

Concluído o projecto, apercebemo-nos da importância do relatório de desenvolvimento. Uma análise deste relatório permite obter informação sobre todo o processo de desenvolvimento e o estado final do produto.
Foi feita uma revisão na especificação de requisitos com o objectivo de estes estarem de acordo com o produto final. Além disso, foram especificados todos os aspectos referentes à implementação da arquitectura e os seus componentes.
Outro aspecto importante deste relatório é a documentação de gestão do projecto, onde estão descritos detalhadamente os aspectos referentes à distribuição do trabalho entre os membros do grupo e o funcionamento do mesmo.
Em relação ao projecto, concluimos que a escolha desta arquitectura, a qual verificamos ser bastante versátil e reutilizavel, foi um factor crucial para o cumprimento de todos os requisitos impostos inicialmente. A criação de componentes dinâmicos permitiu-nos uma grande reutilização dos mesmos, tendo acelerando o processo de desenvolvimento em módulos com funcionalidades idênticas.
Pensamos que o cumprimento dos requisitos é bastante importante no desenvolvimento de um projecto pois, só assim é possivel satisfazer o cliente e, consequentemente, dar à equipa uma imagem de profissionalismo e competência.

A Manual de Utilizador

A.1 Página de entrada

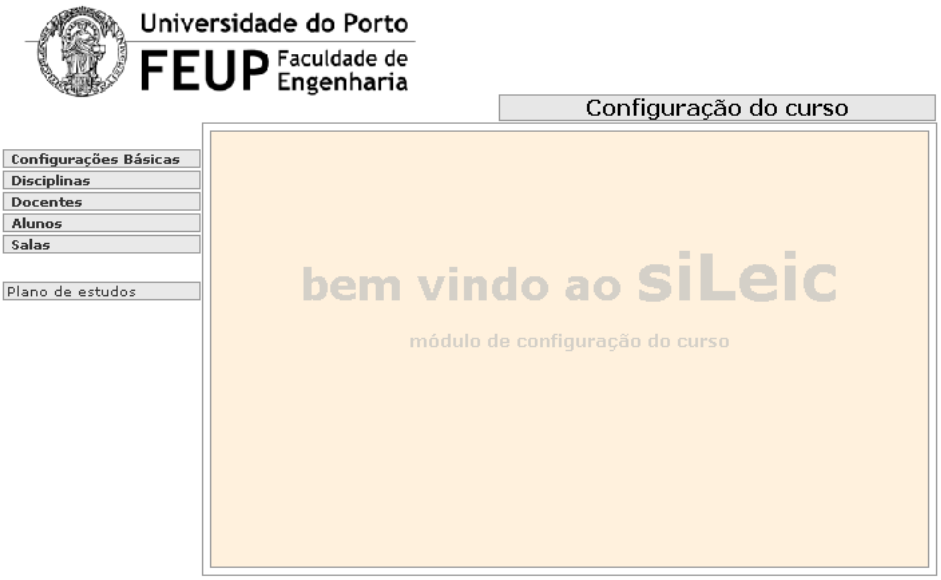

Figura 33: Página de entrada no site de configuração

Esta é a página inicial do produto. Daqui podemos aceder ás secções principais da configuração do curso, bem como ao plano de estudos. Para aceder a qualquer uma das opções, basta seleccionar com o rato em qualquer uma delas.

A.2 Autentificação

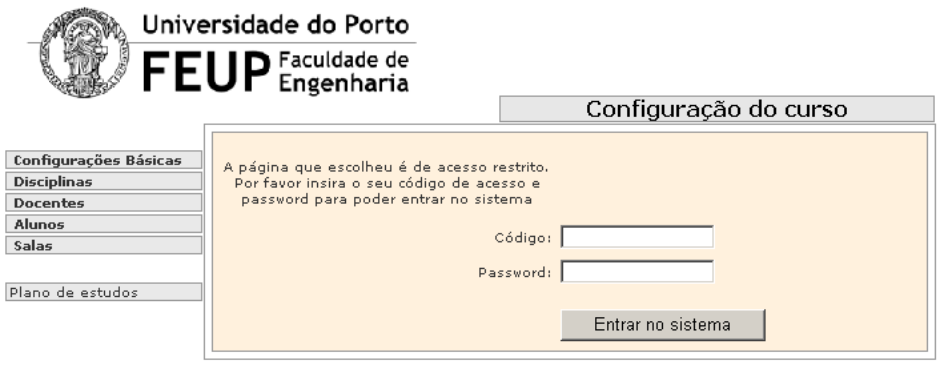

Figura 34: Página de autentificação

Esta página surge quando é necessário verificar se o utilizador tem autorização para aceder à página que pretende. Para isso é necessário que possua um código e uma password. Para então aceder à área restrita, é necessário preencher os campos com os valores respectivos e carregar no botão entrar no sistema. Caso seja aceite a informação que forneceu, é disponibilizada a pagina que pretende. Caso contrário é novamente pedida a informação.

A.3 Página de configuração básicas do curso

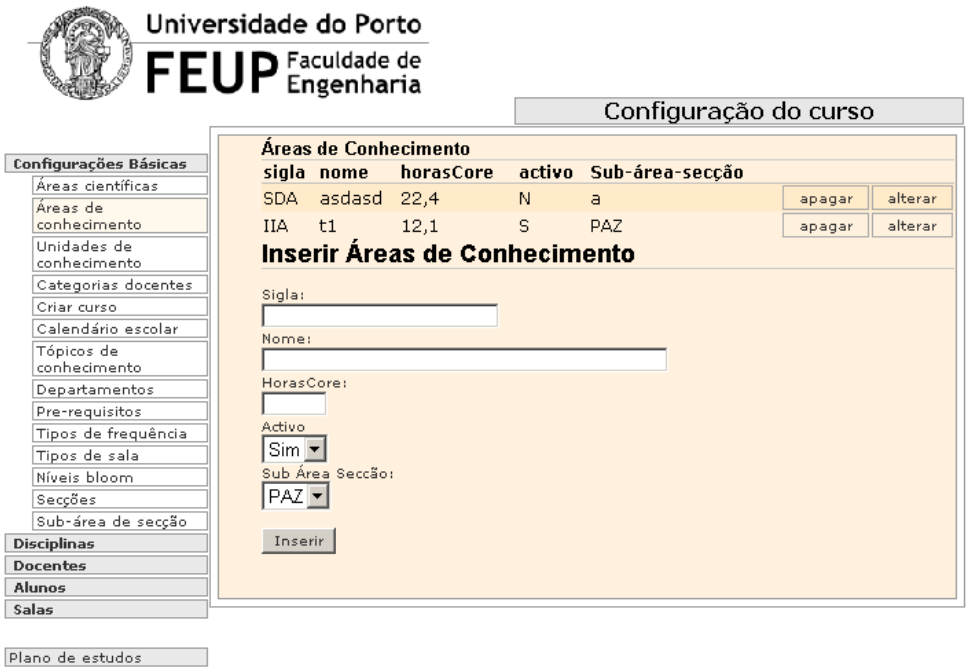

Figura 35: Página de inserção das áreas de conhecimento

Nesta página são inseridos os dados referentes ás áreas de conhecimento. É feita uma listagem das áreas já existentes. A partir desta listagem podemos alterar ou apagar os dados já inseridos. Abaixo da listagem encontram-se as caixas de texto e as combobox referentes aos campos a preencher para uma nova área de conhecimento. Do lado esquerdo surge o sub-menu de selecção, referente às configurações básicas. Ao seleccionar cada uma destas secções do sub-menu, é disponibilizada a página correspondente à secção.

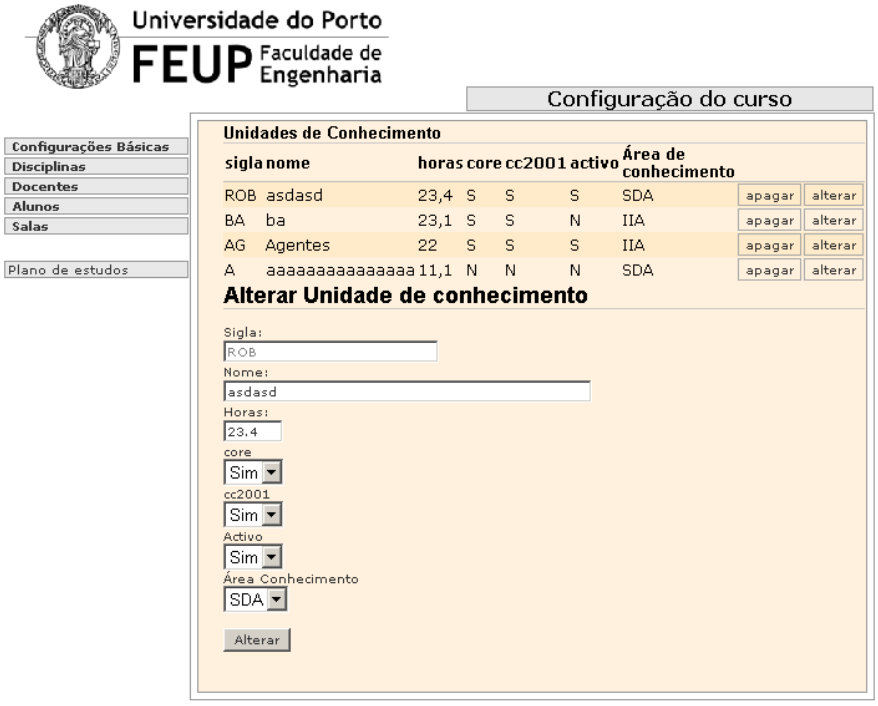

Figura 36: Página de alteração das áreas de conhecimento

Nesta secção são alterados os dados referentes às áreas de conhecimento. Quando somos confrontados com a opção de alterar, os campos são automáticamente preenchidos com os valores actuais. Ao fazer o pedido de alteração os valores a inserir são os que se encontram nos campos.

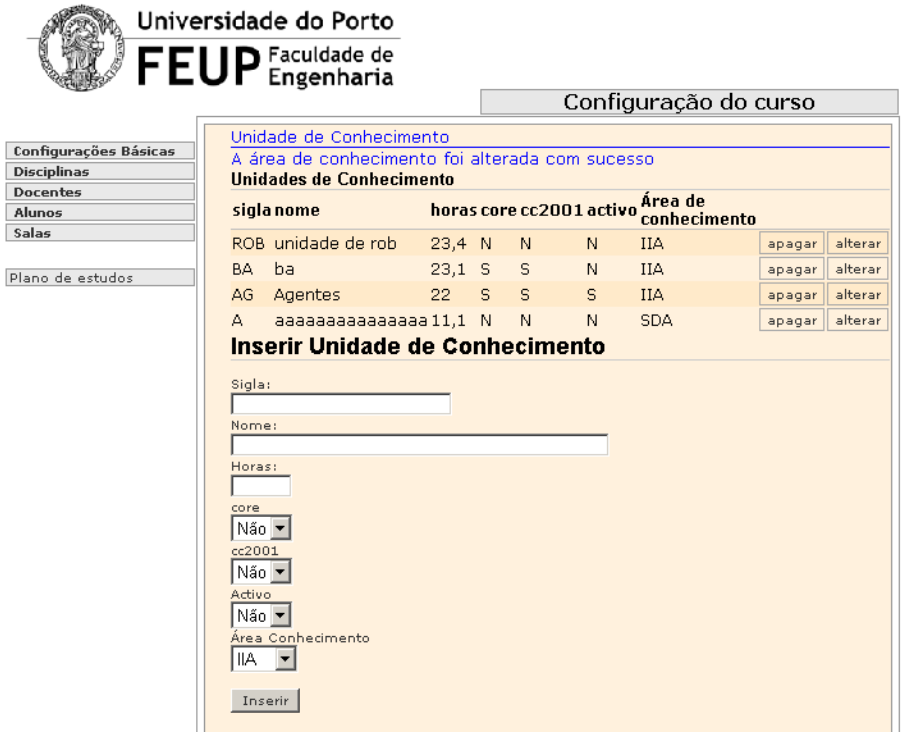

Figura 37: Página de alteração das áreas de conhecimento

Caso os dados sejam alterados com sucesso é dado um aviso e feita uma nova listagem.

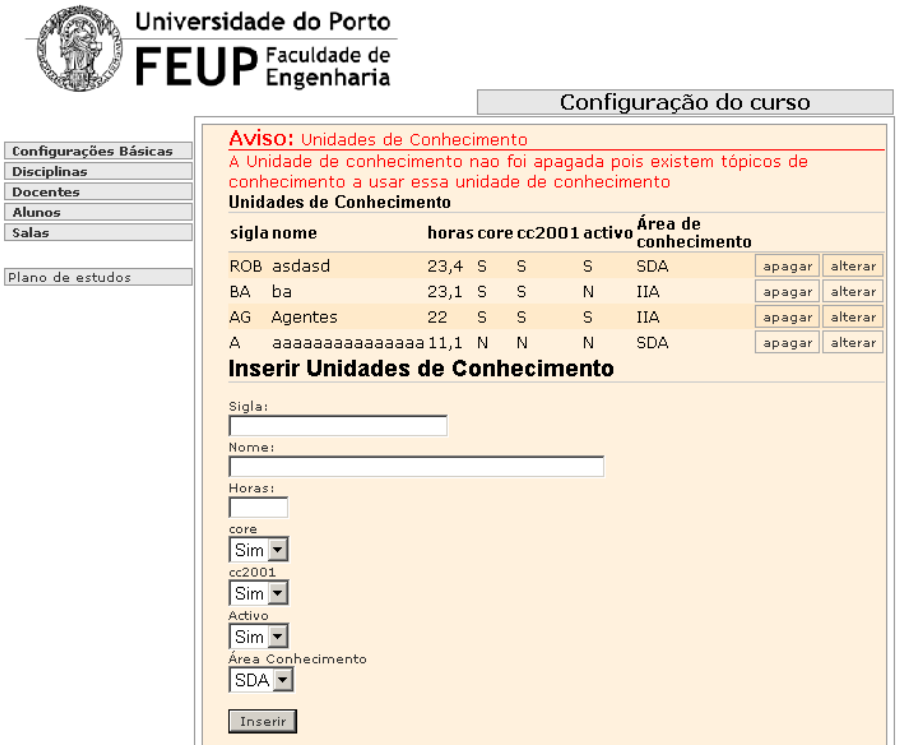

Figura 38: Página de alteração das áreas de conhecimento

Este é um exemplo de uma mensagem de erro. Como neste caso existem dependências entre os dados, não foi possível remover a unidade de conhecimento. É necessário verificar as depêndencias e elimina-las antes de apagar a unidade de conhecimento. Os dados mantêm-se inalterados.

A.4 Página de configuração de disciplinas

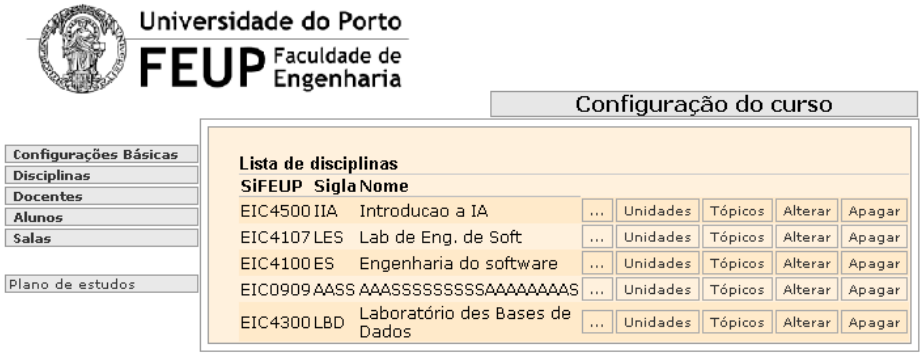

Figura 39: Resultado da pesquisa de disciplinas

Para além das habituais opções de alterar e apagar, aqui podemos alterar quais os tópicos que estão associados a uma dada disciplina através do botão Tópicos. Para alterar quais as unidades de conhecimento que estão ligados a uma dada disciplina, basta seleccionar o botão Unidades. Podemos ainda visualizar todos os dados referntes a uma dada disciplina através do botão '...', já que a disciplina por si só possui demasiados campos para que possamos fazer uma listagem com todos esses campos.

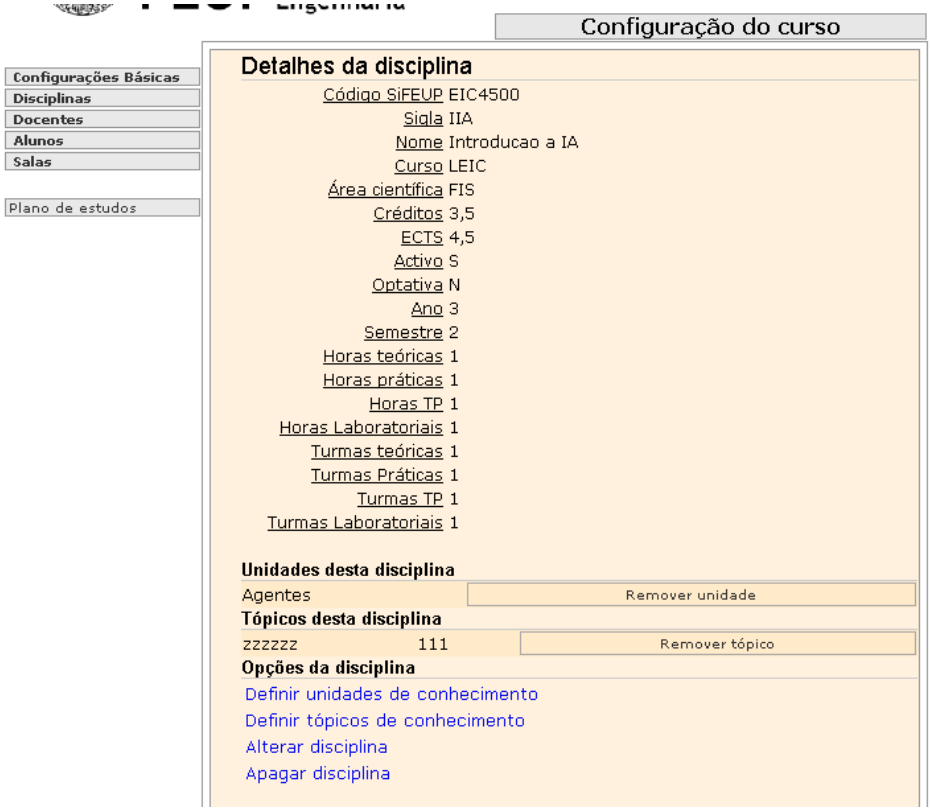

Figura 40: Visualização de uma disciplina

Este é um exemplo de uma listagem completa dos dados referentes a uma disciplina, bem como uma listagem dos tópicos e das unidades referenciadas a esta disciplina. As opções da disciplina (Alterar disciplina, Apagar disciplina, Definir unidades de conhecimento e Definir tópicos de conhecimento) estão disponíveis através de hiperligações no fundo da página. Para remover uma unidade associada a esta disciplina é necessário seleccionar o botão Remover unidade referente a essa unidade. Para remover um tópico é necessário seleccionar o botão Remover tópico referente ao tópico que se quer remover.

Figura 41: Alteração de uma disciplina

Caso se opte por alterar uma disciplina, estes são os campos a preencher. Para alterar o valor das caixas de activação, basta selecciona-las com o rato.

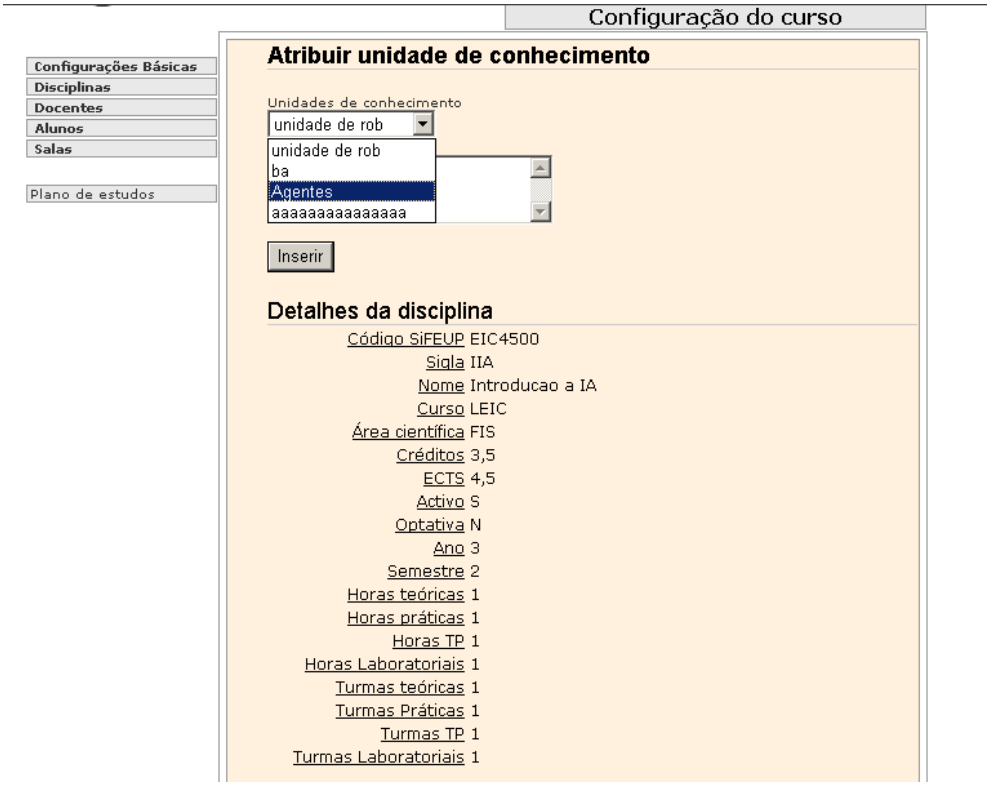

Figura 42: Configurações de unidades de uma disciplina

Se optarmos por definir a unidade de conhecimento surge acima dos dados da disciplina os campos que aparecem na figura. Para associar à disciplina a unidade seleccionada na combobox, seleccionar o botão Inserir.

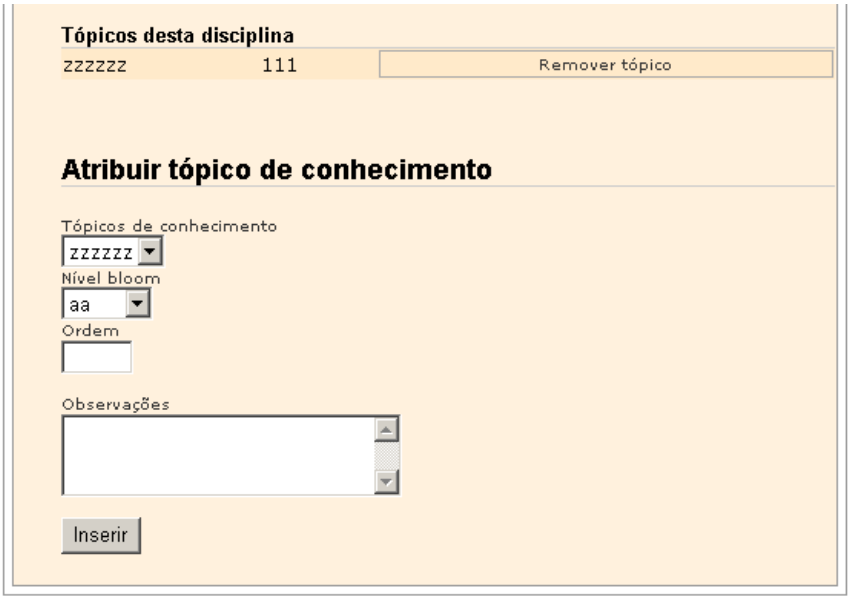

Figura 43: Configurações de tópicos de uma disciplina

Ao adicionar um tópico de conhecimento é necessário definir qual o tópico que se deseja adicionar, que nível bloom se pretende e qual a ordem. Se o utilizador desejar, poderá acrescentar algumas observações. Para associar o tópico de conhecimento à disciplina, seleccionar o botão Inserir.

A.5 Página de configuração de docentes

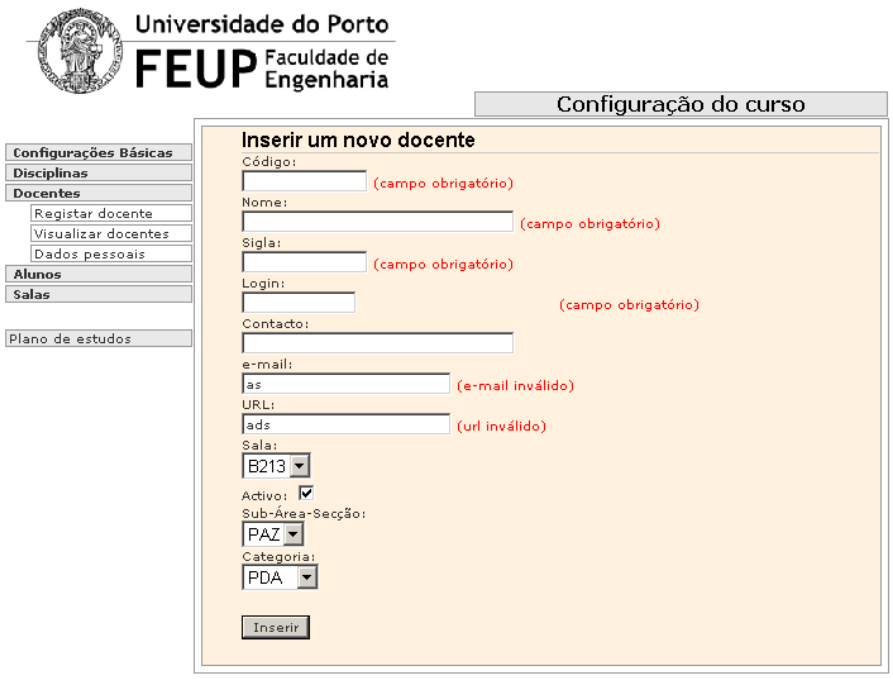

Figura 44: Erros na inserção de um docente

Para inserir um novo docente, é necessário preencher certos campos. Na figura Pode-se verificar quais os campos de preenchimento obrigatório. O e-mail tem que corresponder ao formato xxxx@xxxx e o URL o formato http://xxxxx.

Figura 45: Resultado da pesquisa de docentes

A pesquisa dos docentes pode ser feita por Código e/ou nome. É então feita uma pesquisa com base nesses valores e fornecida uma listagem dos docentes encontrados que cumpram os requisitos da pesquisa. É dado inicio a essa pesquisa através do botão Pesquisar. A listagem inicialmente encontra-se vazia e só após uma primeira pesquisa é feita uma listagem. Através dos botões associados aos docentes podemos alterar os dados do docente, apagar o docente ou visualizar todos os dados do docente.

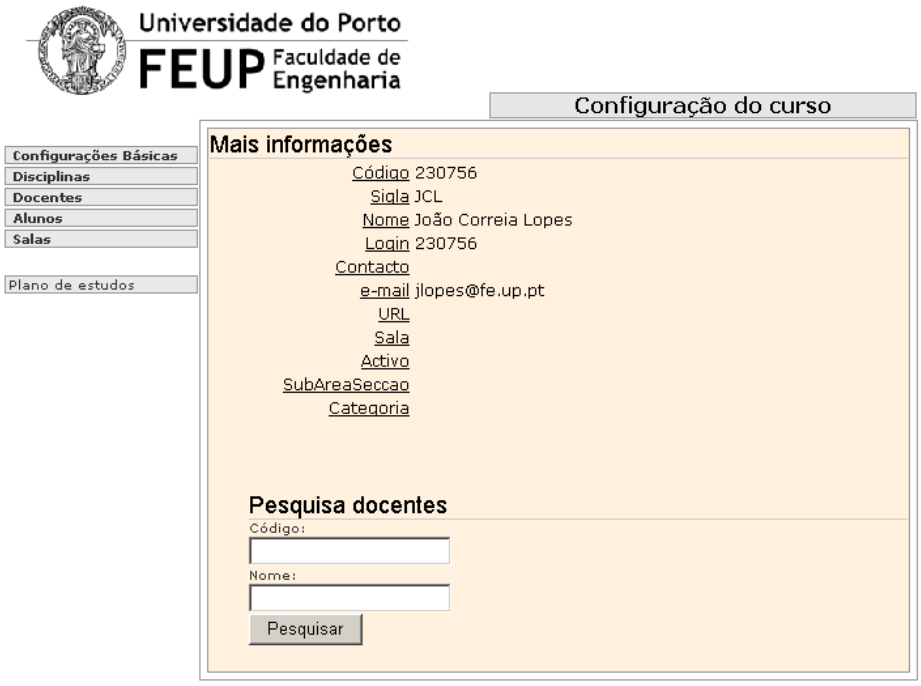

Figura 46: Visualização de dados de um docente

Neste caso podemos fazer uma nova pesquisa, preenchendo os campos necessários.

A.6 Página de configuração de alunos

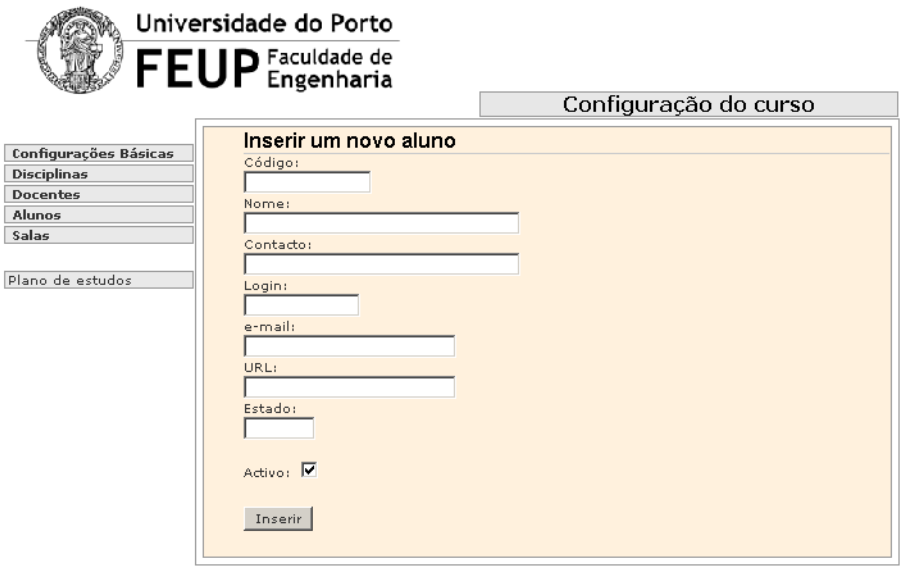

Figura 47: Registo de um aluno

Para registar uma aluno, basta preencher os campos (os campos obrigatórios são os mesmos que existem na página inserir docente) e seleccionar o botão Inserir. Para alterar o valor das caixas de activação, basta seleccionalas com o rato.

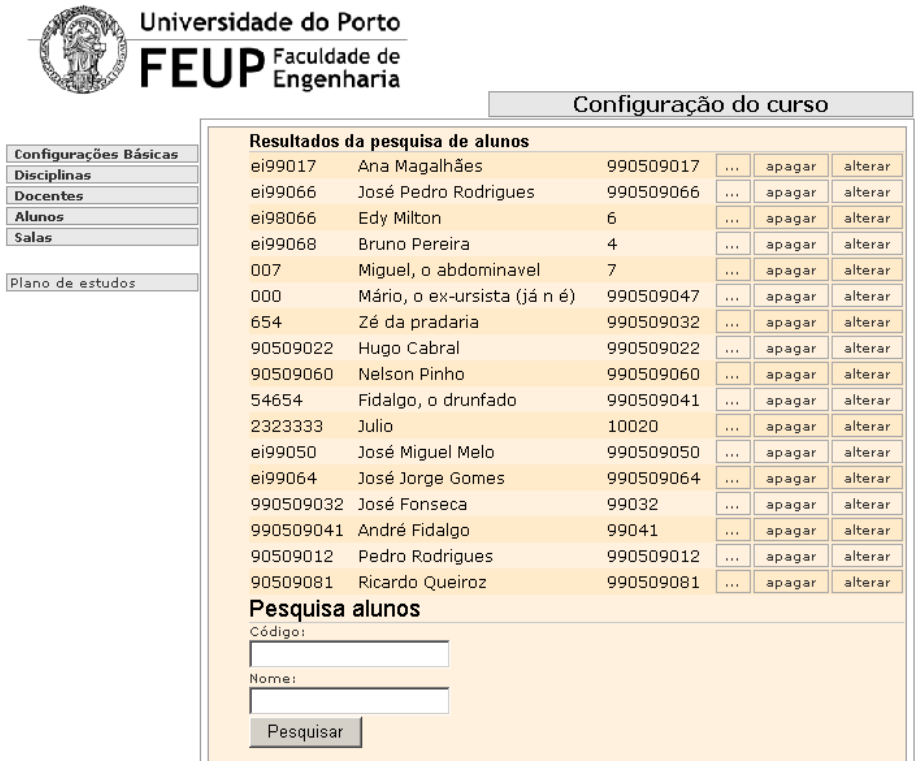

Figura 48: Resultado da pesquisa de alunos

A pesquisa de alunos é em tudo semelhante à pesquisa de docentes.

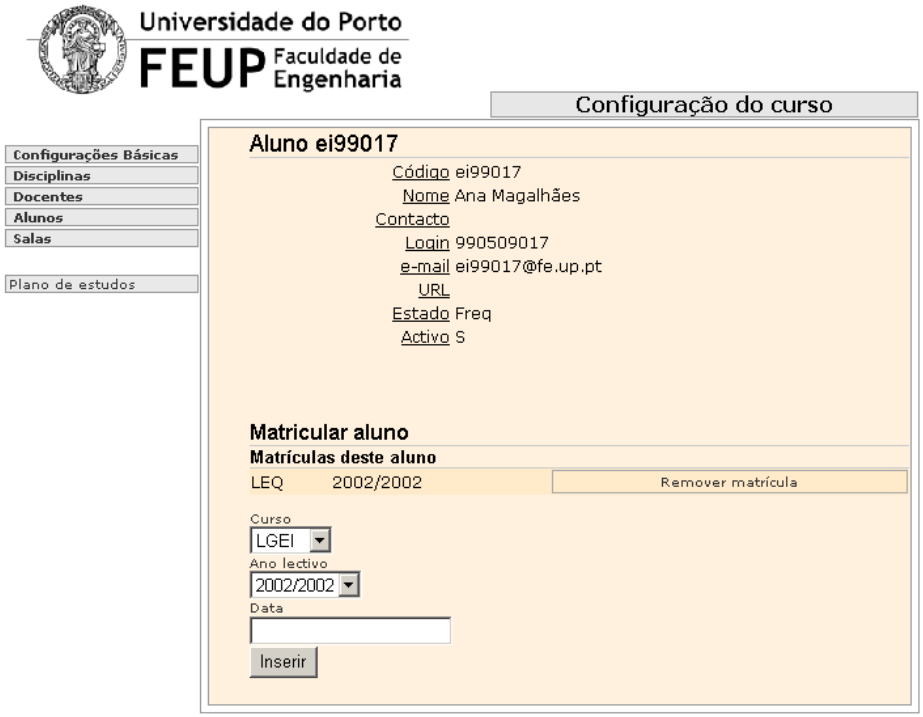

Figura 49: Visualização de dados de um aluno

Ao visualizar os dados dos alunos, é dado qual o curso em que este se encontra matriculado. Para remover essa matrícula, seleccionar o botão Remover matrícula.

A.7 Página de configuração das salas

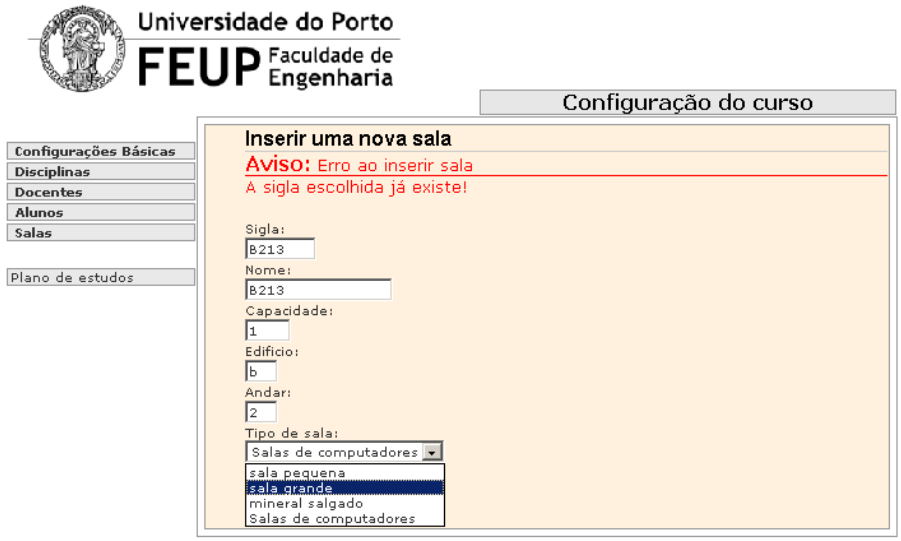

Figura 50: Registo de uma sala

Para registar uma sala, basta preencher os campos da página. Neste caso verificou-se um erro, pois a sigla da sala que se pretendia inserir já existia.

Figura 51: Resultado da pesquisa de salas

Para pesquisar um dada sala podemos procurar ou por sigla ou por nome. No que diz respeito às opções da listagem das salas podemos apenas alterar os dados da sala ou apagar a sala.

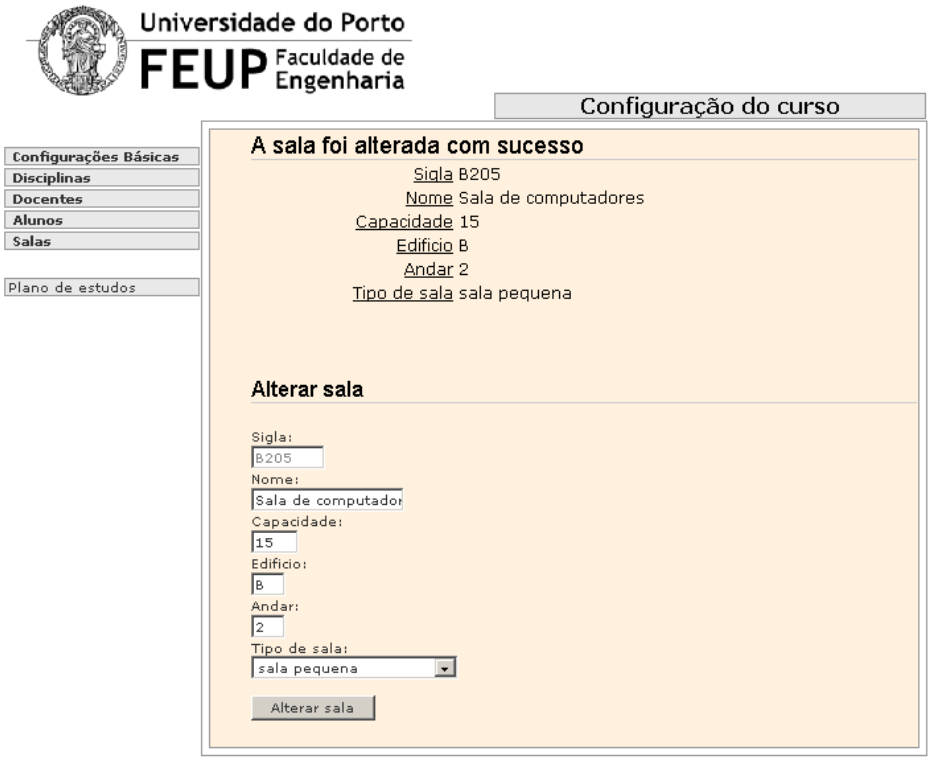

Figura 52: Alteração de dados de uma sala

Ao inserir uma sala, podemos alterar os dados da sala inserida, alterando os dados nas caixas de texto e na combobox e seleccionando o botão Alterar sala.

A.8 Página de visualização do plano de estudos

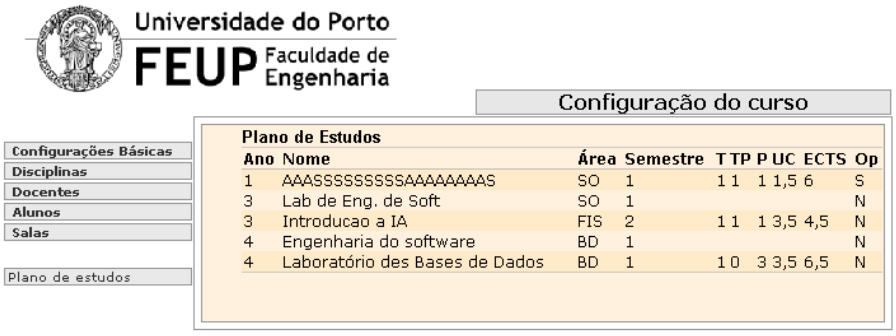

Figura 53: Plano de estudos

Se optarmos por visualizar o plano de estudos, é feita uma listagem das disciplinas sem qualquer opção de alteração.

Referências

[Cor02] Microsoft Corporation. Microsoft .net. http://www.microsoft.com/net, 2002.

[Far01] João Pascoal Faria. Slides sobre uml. http://www.fe.up.pt/ jpf/teach/ES/UML/UML.zip, 2001.

[JCL02] João Pascoal Faria João Correia Lopes. Sítio de engenharia de software. http://www.fe.up.pt/ jpf/teach/ES/index.html, 2002.

[Lop02] João Correia Lopes. Sítio de laboratório de engenharia de software. http://www.fe.up.pt/ jlopes/teach/les.html, 2002.

[MFPS02] André Moniz, José Fonseca, Mário Pereira, and Miguel Sarmento. Relatório de especificação de requisitos. http://www.fe.up.pt/ ei99041/LES/er.pdf, 2002.