Laboratórios de Engenharia de Software

S1 - Configuração do Curso
Relatório de Desenho de Alto Nível

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}

14 de Novembro de 2002

Conteúdo

Lista de Figuras

Agradecimentos

Agradecemos aos nossos amigos, que compartilharam
connosco a longa espera pela configuração (ou não) do IIS.
À Joana Nunes pela paciência e sentido de humor...

André, José, Mário e Miguel

1 Introdução

Este relatório de desenho de alto nível tem como objectivo principal descrever a arquitectura lógica e física do sistema a desenvolver. Portanto iremos esclarecer as questões relativas à arquitectura usada, assim como aos *web-services* idealizados. Dentro deste relatório encapsulamos também o planeamento do projecto e a documentação do protótipo.

1.1 Estrutura do trabalho

Este relatório está estruturado em 5 capítulos. O primeiro capítulo é a introdução do trabalho. No segundo capítulo encontra-se a informação relativa à arquitectura, onde definimos a arquitectura lógica, física e tecnológica, para além dos mecanismos importantes que devem existir na arquitectura do módulo. No que diz respeito à arquitectura lógica, definimo-la com a ajuda de diagramas UML (de pacotes e objectos) para uma melhor compreensão da estrutura. É de referir que decidimos por uma arquitectura lógica em 4 camadas, onde incluímos uma camada de lógica de negócio. Devido à estruturação do trabalho, achamos que esta arquitectura é a mais indicada para uma futura implementação deste módulo num sistema. A arquitectura física foi definida por vários diagramas. No diagrama informal, foi definido a localização física de cada um dos componentes lógicos. No diagrama de distribuição foi feita a transição dos pacotes do modelo lógico para os nós da arquitectura física. Foi também incluído um diagrama em que são definidas as ligações entre os vários componentes existentes nas várias camadas do módulo.

Na secção da base de dados foram definidas as tabelas da base de dados referente ao módulo a desenvolver, para isso apresentamos um esquema relacional.

O terceiro capítulo refere-se ao protótipo apresentado. Para a representação do protótipo foi escolhido o caso de uso das configurações dos níveis de Bloom, pois é um caso de uso bastante simples o que facilita a compreensão do protótipo. É feita a especificação do caso de uso, a implementação nas camadas da aplicação, os testes a que este protótipo foi sujeito, exemplos de utilização e diagramas relativos aos casos de utilização. No capítulo do planeamento, estão definidas as iterações (a sua descrição, duração, elementos do grupo que participaram e *timing*). Está também incluído um diagrama de Gantt, com os nomes dos participantes da iteração e calendarização.

No último capitúlo temos a conclusão do trabalho efectuado.

2 Arquitectura

2.1 Arquitectura lógica

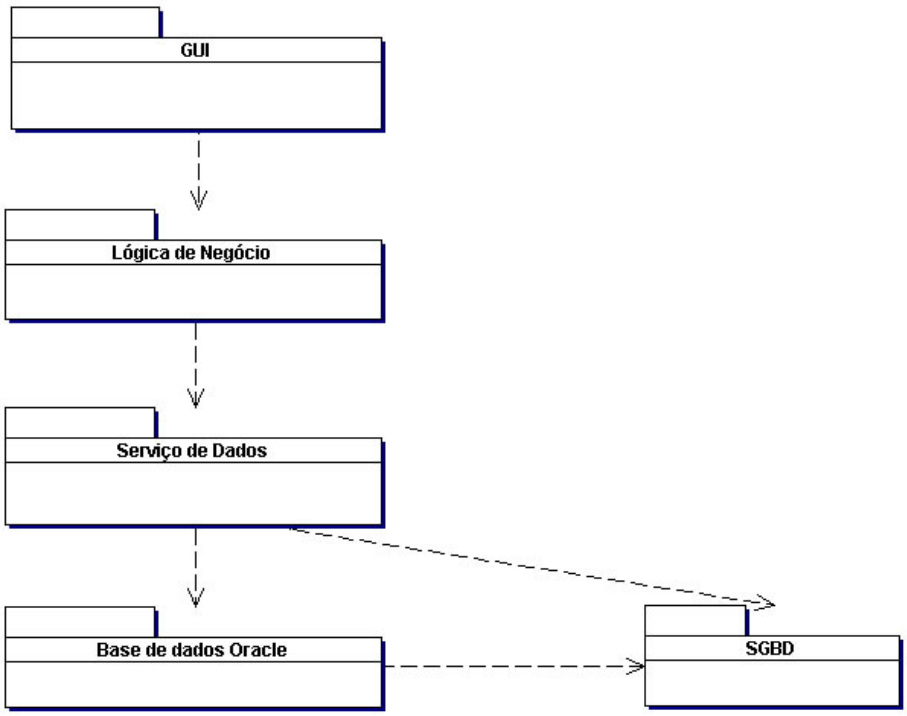

Figura 1: Arquitectura lógica

Este modelo de pacotes tem como objectivo principal distinguir as quatro camadas nas quais vamos dividir a nossa aplicação, que visam um único e principal objectivo: a interoperabilidade. Esse é o mesmo objectivo que está na origem do aparecimento dos webservices.

A estruturação de aplicações tem vindo a evoluir ao longo dos tempos. Se antigamente eram pensados dois módulos, o cliente e o servidor, hoje em dia as coisas são bastante diferentes, tendo em visa as novas tecnologias, como os webservices. Nesta primeira fase a interface gráfica (ou a linha de comandos) ficava do lado do cliente, enquanto que o servidor tratava de toda a lógica de negócio e do acesso à base de dados. Esta arquitectura era tida como eficaz, mas com o desenvolvimento rápido e quase sufocante

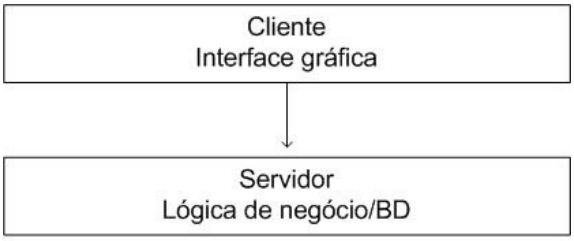

Figura 2: Modelo clássico

da computação dos dias de hoje, a introdução de novas plataformas para desenvolvimento usando programação por objectos, permitiram introduzir uma maior abstracção entre os componentes das aplicações.

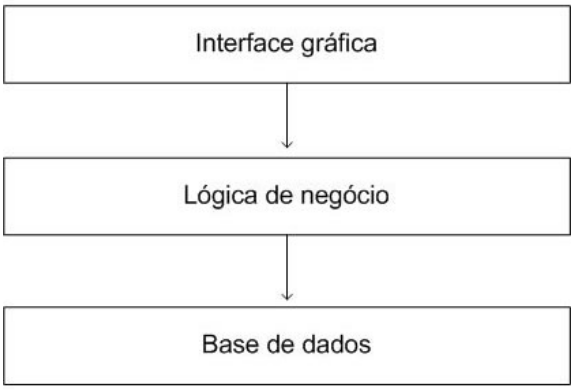

Figura 3: Separação da lógica de negócio

Neste segunda fase separou-se a lógica de negócio da base de dados, deixando de estar dependente do SGBD usado. Assim a lógica de negócio pode ser implementada numa linguagem com muito mais flexibilidade do que as linguagens que estão disponíveis nos SGBD's, estas muito mais direccionadas às consultas e outros tipos de operações sobre a base de dados.

Esta último diagrama mostra como vamos implementar a nossa aplicação, com quatro camadas. A última camada apresentada é o *Serviço de dados* que vai servir de interface à base de dados da aplicação. A camada de base de dados do modelo anterior corresponde geralmente a procedimentos na própria base de dados, que serão invocados pela lógica de negócio usando por exemplo PL/SQL.

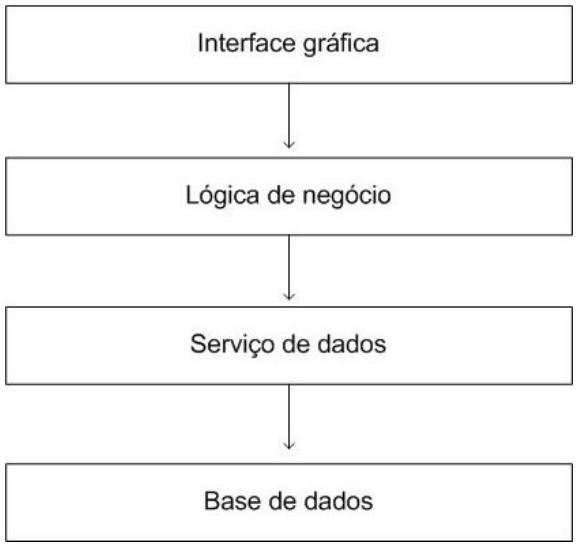

Figura 4: Arquitectura usada

Este passo torna-se então de extrema relevância pois vai dar à nossa base de dados o estatudo de interoperável. Assim conseguimos uma abstracção total por parte da lógica de negócio em relação ao SGBD usado. Imaginemos que o siLEIC queria disponibilizar os seus dados a outro sistema de informação. Poderia simplesmente mostrar o esquema da base de dados e dar-lhes o acesso devido. Não seria mais seguro e correcto disponibilizar uma documentação sobre como usar a interface que a camada de serviços de dados irá fornecer? Sim, obviamente que sim. Sem dúvida que isto mostra as fortes potencialidades dos webservices.

Temos assim que os objectivos para esta arquitectura são os seguintes:

- permitir alterações em qualquer uma das camadas sem que isso interfira nas outras;
- retirar da lógica de negócio a necessidade de conhecer a estrutura da base de dados, oferencendo *webservices*, que quando invocados acedem directamente à informação contida na base de dados
- permitir que os todas as camadas estejam alojadas em máquinas distintas e implementadas em linguagens distintas.

Sabemos porém que isto torna as comunicações bastante mais lentas, mas visto isto ser um trabalho puramente académico em que queremos ao máximo

explorar os limites dos webservices, optámos por esta arquitectura. Interessa também estudar as dificuldades que surgem na implementação de webservices quer ao nível real da interoperabilidade oferecido pelas normas SOAP e todas as outras que estão relacionadas. Concluindo, algo deste género apenas se justificaria no caso de se necessitar de realizar uma aplicação completamente distribuída.

2.1.1 Descrição das camadas

GUI - Interface gráfica

A interface gráfica com o utilizador trata dos aspectos de visualização de informação contida na base de dados. Todos os pedidos de informação são efectuados à *Lógica de Negócio*. Estando este módulo separado dos outros torna-se mais fácil actualiza-lo. Mais uma vez, qualquer mudança neste componente da arquitectura não influencia nenhum dos outros. Assim, a evidenciar ainda mais as potencialidades dos *webservices*, o sistema permite a criação de várias interfaces diferentes com o utilizador, de modo simples e consistente: interface Web, interface Wap, interfaces para *laptop*'s, etc. Isto porque se todos os componentes da Lógica de Negócio são disponibilizados como web services, o seu acesso é idêntico independentemente da interface que esteja definida. Neste projecto só irá ser implementada uma interface web.

Lógica de Negócio

Segundo esta estrutura a lógica de negócio não necessita de saber nada sobre a base de dados. Nem a forma como está estruturada, nem tanto a forma como as consultas são optimizadas são necessárias para implementar a lógica de negócio. Apenas precisa de ter conhecimento dos *webservices* disponibilizados pelo Serviço de dados. Esta separação ainda frisa mais a interoperabilidade pois podemos, por exemplo, aceder a bases de dados distintas sem sequer necessitarmos de ter conhecimento disso. Esta camada é responsável por criar *webservices* que representam os casos de uso definidos anteriormente no relatório de especificação de requisitos. A Lógica de negócio é também responsável pelo identificação e tratamento de erros na configuração de um curso.

Serviço de dados

Esta camada serve para acedermos à base de dados, e por isso necessita de conhecer muito bem a estrutura e as potencialidades da base de dados. Será assim a imagem da base de dados do ponto de vista das outras camadas. É através dos webservices disponibilizados por esta camada que as duas de cima (ou outras entidades exteriores ao projecto) comunicam com a base de dados. Esta camada irá estar sempre muito associada á camada da base de

dados, embora possam funcionar em sistemas distintos, mas isto não garante que o serviço de dados não conheça a 100 % a camada inferior.

Basicamente o serviço de dados vai fornecer objectos que espelham entidades da base de dados. Ou seja irão ser criados objectos que guardam informação da base de dados, e estes objectos por sua vez serão chamados pelos *webservices* desta camada. É de frisar que os *webservices* vão retornar dados e não objectos. A camada superior (Lógica de Negócio) só recebe dados que vai enviar por sua vez à camada de Interface.

Base de dados

Como base da aplicação temos a base de dados. Como é sabido, aqui estarão armazendos todos os dados relativos à configuração da LEIC. À camada superior, são oferecidas capacidades de execução de consultas e actualizações da mesma informação. As potencialidades disponíveis dependerão sempre de qual o sistema de gestão de bases de dados escolhido, sendo que a utilização de funcionalidades específicas do SGBD para melhoria de performance ou facilitação do desenvolvimento é totalmente independente e não afecta de forma alguma a implementação da lógica de negócio e da interface gráfica.

2.2 Mecanismos importantes

Verificação Ao serem inseridos os tópicos das disciplinas, é feita a verificação dos pré-requisitos destes tópicos, pois existe uma hierarquia na qual têm que estar ordenados. Ao inserir um dado tópico numa dada disciplina, é testado se todos os tópicos que são pré-requisitos deste tópico a ser inserido já foram leccionados em disciplinas anteriores. Esta verificação é feita baseando-se no semestre da disciplina a que cada tópico pré-requesito está associado, e no semestre da disciplina a que queremos associar um tópico.

Tratamento de erros Os dados ao serem inseridos na base de dados são verificados pela camada da lógica de negócio. Esta verificação trata de inconsistências nos dados que possam ocorrer no preenchimento do formulário pelo utilizador, sendo feito um aviso pela interface a este.

Autenticação Havendo partes do serviço que não estão disponíveis ao público, é feito um pedido de verificação de autenticação de cada vez que utilizador queira aceder a cada uma dessas secções. Essa verificação é feita a partir de um webservice, que comunica com o respectivo módulo que trata da autenticação.

Concorrência Neste módulo e apesar deste ser um aspecto de vital importância para qualquer sistema de base de dados, não cremos que haja problemas de concorrência, pois todos os dados são inseridos apenas por um utilizador que é o administrador. Caso a autenticação seja bem feita para que seja apenas este a inserir os dados, não haverá problemas. As visualizações serão feitas através de ficheiros .pdf, que serão reconstruídos a cada actualização dos tópicos ou do plano de estudos.

Ligação à base de dados A ligação à base de dados é feita através da camada de persistência, que por sua vez está ligada à camada do servidor aplicacional. Dado que vamos utilizar a plataforma .NET, esta ligação com a base de dados é feita com o ADO.NET que utiliza o XML para uma transferência eficiente dos dados.

Encriptação A encriptação é feita pelo próprio .NET, encriptando o envelope SOAP, com mecanismos existentes na plataforma .NET. É de extrema importância que estes mecanismos sejam implementados pois o SOAP por si só não tem mecanismos de encriptação.

2.3 Arquitectura física

2.3.1 Diagrama informal de distribução

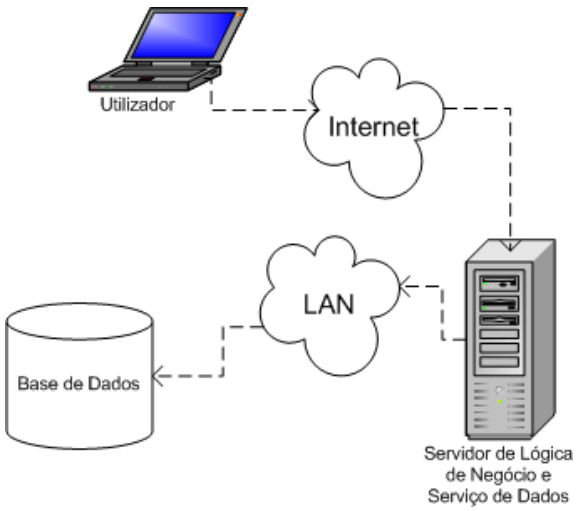

Figura 5: Diagrama informal de distribução

Este diagrama representa informalmente a arquitectura[Far01] física do sistema. Como se pode ver, existem dois servidores. Um que serve a base de dados e outro que serve a aplicação. Esta separação em termos físicos é muito vantajosa, apesar de, à partida parecer que não. À primeira vista parece mais lógico ter a aplicação e a base de dados no mesmo servidor. Estando na mesma máquina, em princípo o acesso à base de dados, por parte da aplicação, seria mais rápido. Mas neste caso, estamos a falar de um servidor web. Os servidores web, além de terem de executar a aplicação, têm de fazer muitas outras coisas (p.ex: controlar os acessos e segurança) e necessitam de uma configuração específica e optimizada. Tendo isso em conta é mais vantajoso, para o funcionamento do sistema, ter um servidor só a servir a aplicação (completamente optimizado para o efeito), e ter um servidor (que normalmente é integrado na mesma rede local que o servidor web) só a servir base de dados.

2.3.2 Diagrama de distribução

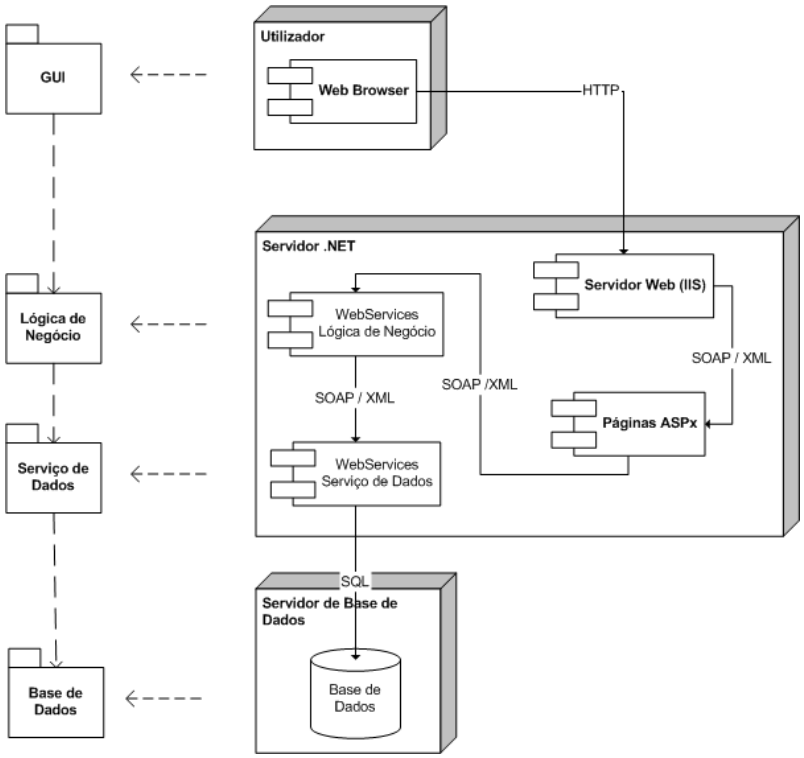

Figura 6: Diagrama de distribução

O diagrama de distribuição representa a estrutura física[Far01] de todo o sistema. Como se pode ver (e como já foi referido no diagrama anterior) o sistema têm ao seu dispôr dois servidores, um para a Base de Dados e outro para a Lógica de Negócio e Serviço de Dados.

- **Utilizador**
  Este nó representa o computador do utilizador do sistema. O utilizador recorre a um WebBrowser que faz os pedidos ao servidor web.

- **Servidor .NET**
  Este nó integra o Servidor Web, as Páginas ASPx e os WebServices que efectuam a lógica de negócio e o serviço de dados. O Servidor Web trata os pedidos HTTP (vindos do cliente) e invoca as respectivas Páginas ASPx. Por sua vez, as Páginas ASPx invocam os WebServices necessários para que seja efectuada a lógica de negócio. Os WebServices que efectuam a lógica de negócio comunicam com os WebServices do serviço de dados para obterem informação da base de dados. Neste

ponto pode surgir uma certa polémica - porquê o uso de WebServices a comunicarem com outros WebServices quando se sabe que os WebServices são mais lentos que os acessos directos? Ao usarmos WebServices para comunicar com a base de dados, estamos a fazer duas coisas muito importante: estamos a encapsular completamente a base de dados (o que é muito importante para a segurança e uniformização do sistema) e estamos a disponibilizar um sistema de acesso (a outros módulos ou entidades) à nossa informação sem revelar a estrutura da base de dados. O facto separarmos a Lógica de Negócio do Serviço de Dados possibilita-nos também disponibilizar a nossa lógica de negócio a outros módulos (p.ex: a um módulo que esteja responsável pela interface do siLEIC).

- **Servidor de Base de dados**
  Este nó contêm apenas o servidor de base de dados Oracle. O facto de termos este servidor separado do servidor da aplicação permite-nos contruir uma máquina muito mais optimizada - como é sabido, as configurações, a nível de software e hardware, para um servidor de base de dados são muito diferentes das de um servidor web - que se reflete num serviço mais eficiente.

2.3.3 Diagrama de componentes

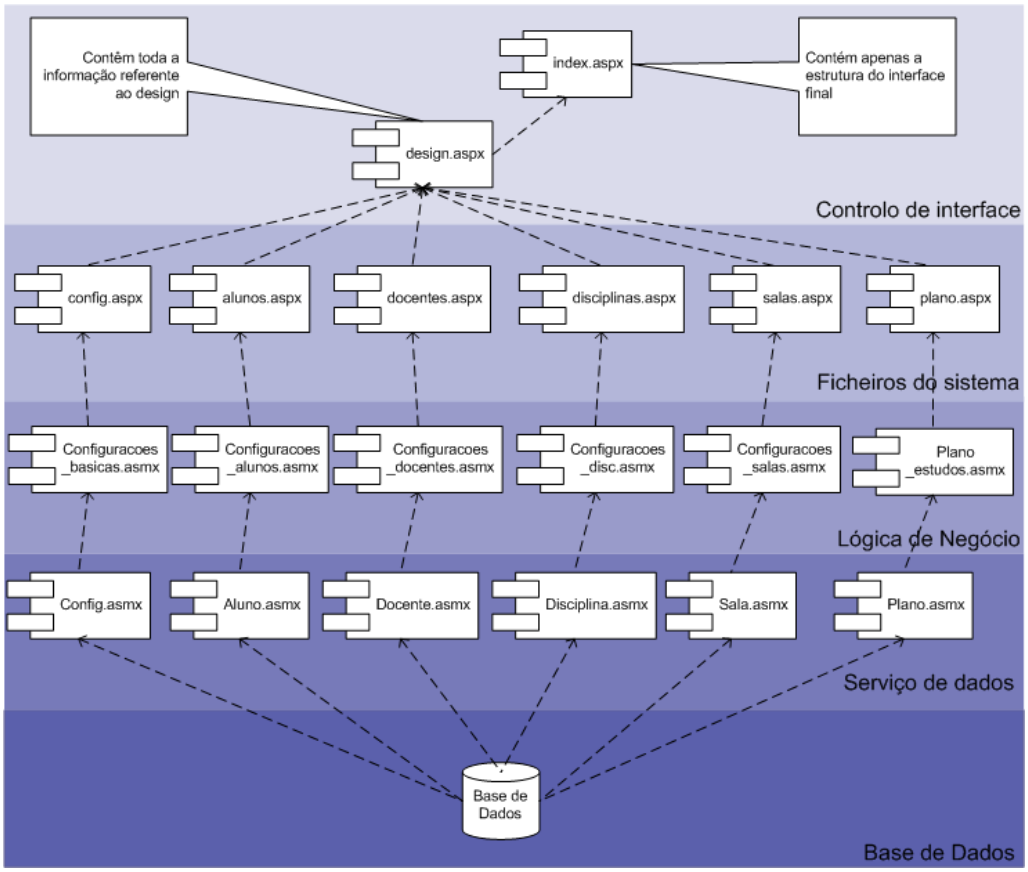

Figura 7: Diagrama de componentes

Este diagrama identifica as componentes físicas[Far01] que constituem o sistema. Segue-se uma breve descrição das várias camadas.

A Base de Dados é a camada de presistência onde toda a informação é guardada. Esta camada só é acedida pelo Serviço de Dados. Desta forma toda a estrutura da base de dados é encapsulada.

O Serviço de Dados é constituído por seis WebServices que se encarregam de fazer a ligação entre a base de dados e a Lógica de Negócio. Cada WebService está responsável por ”gerir” um certa parte da base de dados (p.ex: o Config.asmx é responsável por recolher, da base de dados, as informações que dizem respeito às configurações básicas). Desta forma é possível encapsular toda a base de dados mantendo sempre a informação perfeitamente acessível.

A Lógica de Negócio, à semelhança do Serviço de Dados, é constituída por seis WebServices diferentes. Cada um deles é responsável pela lógica de negó-

cio que lhe está associada (p.ex: o Configuracoes_salas.asmx é responsável por toda a lógica de negócio que diz respeito ao tratamento de informações relativas às salas). Com esta organização, é possível identifcar claramente os vários subsistemas do nosso módulo. Esta divisão, apesar de simplicista torna o sistema mais portável e robusto.

Os Ficheiros do Sistema, analogamente às outras camadas, são divididos em seis ficheiros (um para cada subsistema). Cada um destes ficheiros é responsável por invocar os WebServices da lógica de negócio respectiva. É também da sua reponsabilidade construir um interface básico (não formatado) para a lógica de negócio.

O Controlo de Interface é constituido por dois ficheiros. O ficheiro "index.aspx" que contêm a estrutura do interface (p.ex: a definição da posição dos menus e conteúdos) a partir do qual são incluidos todos os outros ficheiros. O outro ficheiro, o "design.aspx" que contêm todas as definições de design é reponsável por fazer a formatação de todo o interface (p.ex: define as cores dos links, controla a colocação de imagens, as cores de fundo, o tipo de menus, etc). Tendo todo o sistema que controla o design do sítio centrado num único ficheiro, é possível uma manutenção do design muito mais facilitada (p.ex: alterar as cores dos menus na época Natalícia ou colocar um coelhinho nos cantos das imagens na altura da Páscoa). Como todos os ficheiros são incluidos a partir do ficheiro "index.aspx" os elementos do grupo podem trabalhar sem terem de se preocupar com a estrutura do sítio ou com o design e vice-versa, ou seja a equipa responsável pelo desenvolvimento do design pode trabalhar sem ter de se preocupar com os conteúdos.

2.3.4 Base de dados

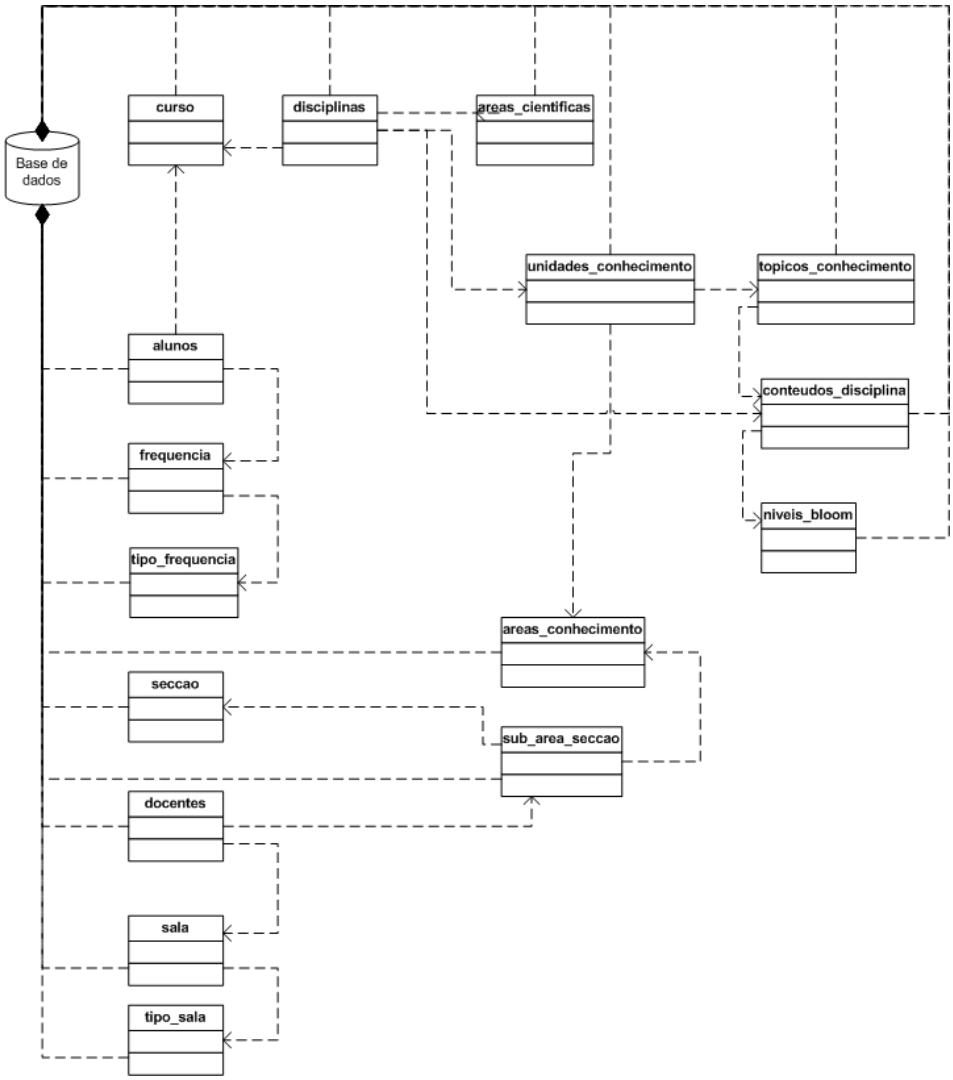

Figura 8: Estrutura de dependências da base de dados

O diagrama[Far01] apresentado esquematiza as várias tabelas (e quais as suas dependências) em que a base de dados se divide. Segue-se uma breve descrição de cada tabela:

- **curso**
  Contem informação sobre o curso.
- **disciplinas**
  Regista a informação sobre as disciplinas.

- **areas_cientificas**
  Guarda a informação sobre as áreas cientificas de um curso.
- **unidades_conhecimento**
  Regista informação sobre as unidades de conhecimento de uma área de conhecimento.
- **topicos_conhecimento**
  Guarda a informação sobre os tópicos de conhecimento de uma unidade.
- **conteudos_disciplina**
  Guarda os conteudos de uma disciplina e associa os tópicos de conhecimento às disciplinas.
- **niveis_bloom**
  Contem informação acerca dos níveis bloom.
- **areas_conhecimento**
  Regista as várias áreas de conhecimento.
- **alunos**
  Contem informação sobre os alunos.
- **frequencia**
  Regista a frequencia do aluno.
- **tipo_frequencia**
  Contêm os vários tipos de frequencias que existem.
- **docentes**
  Regista a informação sobre os docentes.
- **seccao**
  Contem informação sobre as secções.
- **sub_area_seccao**
  Relaciona os docentes com as áreas de conhecimento e as séccções
- **sala**
  Contêm informação sobre as salas.
- **tipo_sala**
  Contêm informação acerca dos tipos de salas.

2.3.5 Esquema relacional

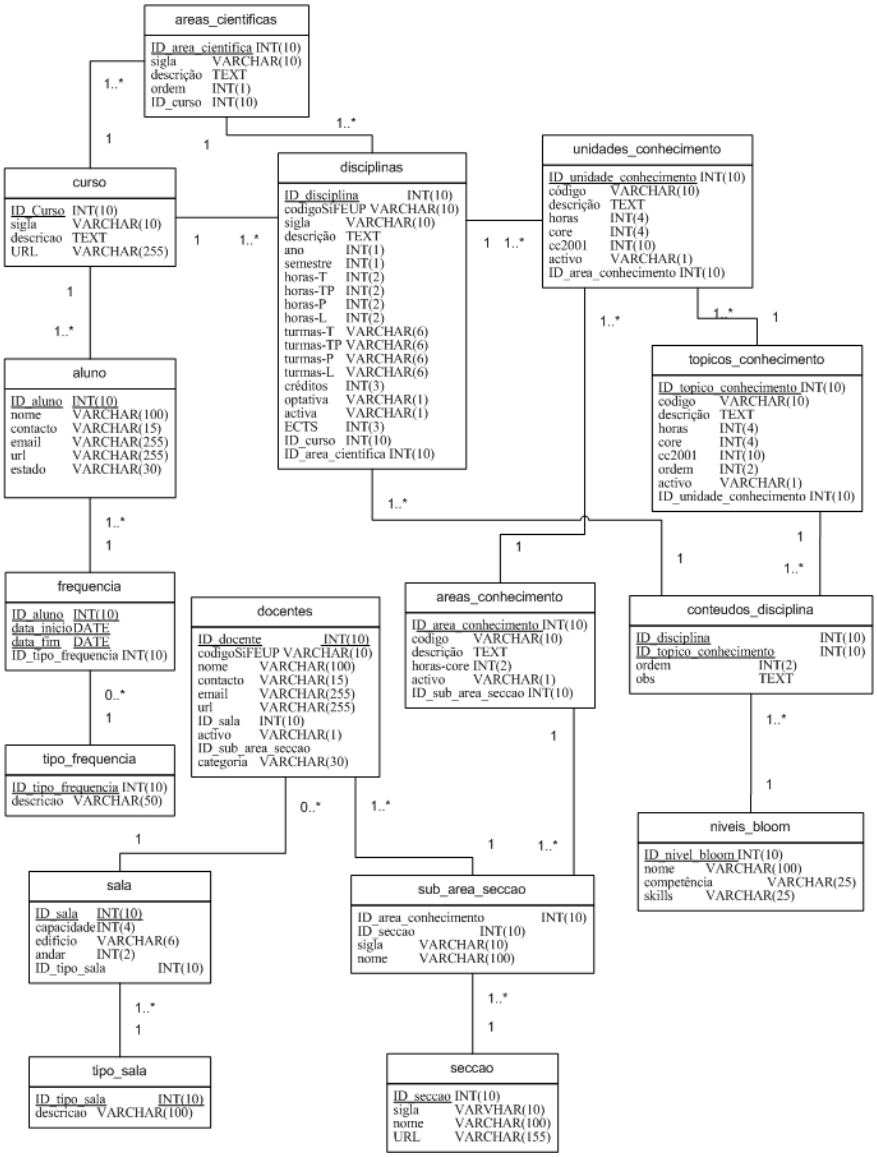

Figura 9: Esquema relacional da base de dados

Este diagrama esquematiza e estabelece as relações entre as várias tabelas que constituem a nossa base de dados.

2.4 Arquitectura tecnológica

2.4.1 Aplicações

- Edit plus - Edição de texto da documentação. Para este programa existe um ficheiro de *tags* para o desenvolvimento de documentos em latex, para além de possuir um serviço de ftp.
- Microsoft Visual Studio .NET - ferramenta de desenvolvimento de software baseado na plataforma Microsoft .NET. Esta aplicação tem uma ajuda muito completa (e complexa) e com wizards muito úteis e funcionais, automatizando variados processos.

2.4.2 Tecnologias

No que diz respeito ás tecnologias utilizadas no desenvolvimento do módulo, remetemos para o relatório de tecnologias já apresentado. Neste apresentam-se as tecnologias ASP.NET, ADO.NET, C#, Visual Basic .NET, html e css e a sua utilização neste módulo.

3 Protótipo

3.1 Introdução

Neste tópico do relatório vamos demonstrar como irá funcionar a nossa arquitectura, para isso iremos implementar um caso de uso escolhido por nós. A interface irá disponibilizar um menu com diversas opcções de configuração do curso, das quais só esta estará a funcionar para esta fase. Efectivamente só podemos selecionar um caso de uso do módulo das configurações básicas porque sem estas configurações iniciais o resto dos casos de uso que delas dependem não iriam funcionar.

Selecionamos então um caso de uso mais básico possível para ser mais simples a implementação das comunicações entre as quatro camadas. O caso de uso escolhido foi então a **configuração dos níveis de bloom**. Basicamente todos estes casos de uso, que efectuam configurações do curso (ou tecnicamente falando, preencheem tabelas de dados estáticos do curso), vão funcionar da mesma forma que o caso aqui demonstrado.

Apesar de termos toda a estrutura do protótipo pensada e planeada, não nos foi possível efectuar a sua implementação pois a faculdade não disponibilizou os recursos necessários (O IIS que foi instalado nos computadores não funciona pois não executa ficheiros "aspx" e "asmx").

3.2 Especificação

Neste caso de uso o único actor é o Gestor. Caso já existam níveis bloom definidos, irá ser apresentada ao Gestor uma listagem das características de todos os níveis bloom existentes. Cada nível bloom tem a opção de ser editado ou apagado. É também possivel inserir novos níveis bloom, sendo necessário nesse caso preencher um formulário com 3 campos, nomeadamente: o nome, competência e skills. A imagem seguinte mostra como irá funcionar a listagem e o formulário de inserção.

3.3 Implementação

Em relação ao processo de autenticação, a única verificação que teremos que fazer é ao nível da sessão. Basicamente temos que perguntar se existe uma variável de sessão que identifica um Gestor. Se não existir vai ser invocado um *webservice* do grupo da autenticação.

Na **lógica de negócio** existem 6 *webservices* com diversos *webmethods*. Para esta fase iremos implementar o *webservice* Configuracoes_basicas e os *webmethods* get_lista_niveis_bloom(), adicionar_bloom(), alterar_bloom() e remover_bloom(). Este *webservice* irá ainda conter outros metódos para configurar um curso que serão implementados numa fase posterior. Pode parecer que os métodos dos *webservices* da lógica de negócio simplesmente invocam métodos iguais aos métodos dos *webservices* do serviço de dados. De facto, é isso que efectivamente acontece neste caso de uso, mas não será igual para o resto dos casos de uso.

3.3.1 Lógica de negócio

Descrição dos web methods do webservice Configuracoes_basicas.asmx:

get_lista_niveis_bloom()

Este método irá invocar um *webservice* do serviço de dados para obter informação dos níveis de bloom. Assim terá acesso a uma listagem de todos os níveis de bloom contidos na base de dados.

adicionar_bloom(string nome, string competencia, string skills)

Este método invoca um método do serviço de dados para inserir um nivel de bloom. Recebe como argumentos os campos necessários para definir o nível de bloom.

alterar_bloom(string nome, string competencia, string skills)

Este método invoca um método do serviço de dados para alterar os dados de um nivel de bloom.

remover_bloom(int id_bloom)

Este método invoca um método do serviço de dados para remover um nivel de bloom.

3.3.2 Serviço de dados

Descrição dos web methods do webservice Config.asmx:

get_nivel_bloom(int id_bloom)

Este método acede à base de dados e retira a informação relativa a um dado nível Bloom.

alterar_nivel_bloom(string nome,string competencia,string skills ,string funcao)

Este método permite inserir, alterar ou remover o nivel bloom definido pelo nome.

3.4 Testes

Não foi possível efectuarmos o módulo de testes devido à inexistência de um servidor IIS na faculdade capaz de correr *webservices*.

3.5 Exemplos de utilização

Pelo mesmo motivo do tópico anterior não é possível apresentarmos exemplos de utilização.

3.6 Desenho detalhado relativo ao caso de utilização

3.6.1 Objectos da lógica de negócio

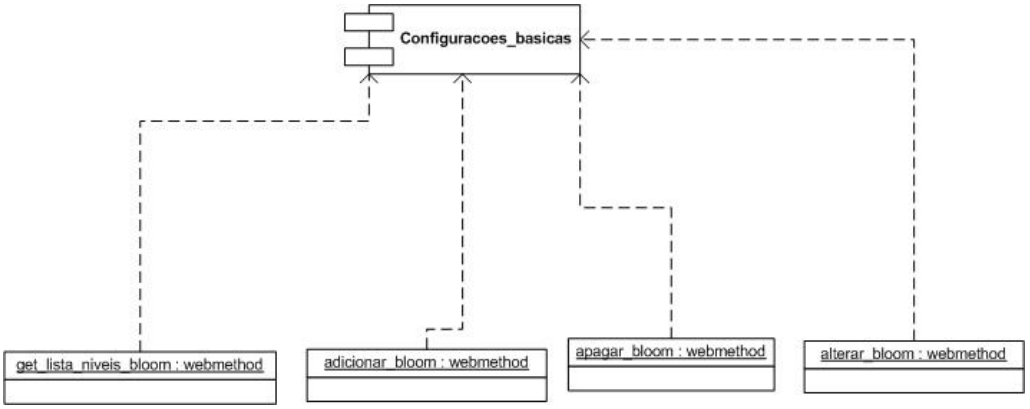

Figura 10: Objectos da lógica de negócio

Esta imagem representa a ligação que existe entre o *webservice* e os *webmethods* da camada da lógica de negócio. Estes métodos ao serem chamados, comunicam com a camada de serviço de dados, nomeadamente com *webmethod* referente aos níveis Bloom. Após receberem o resultado do pedido, este é enviado para o webservice, para que este o possa enviar à interface. As comunicações feitas com a interface e com a camada de serviços de dados são feitas por XML, enquanto as comunicações entre os *webmethods* e os *webservices* são feitas por ASP.

3.6.2 Objectos da serviços de dados

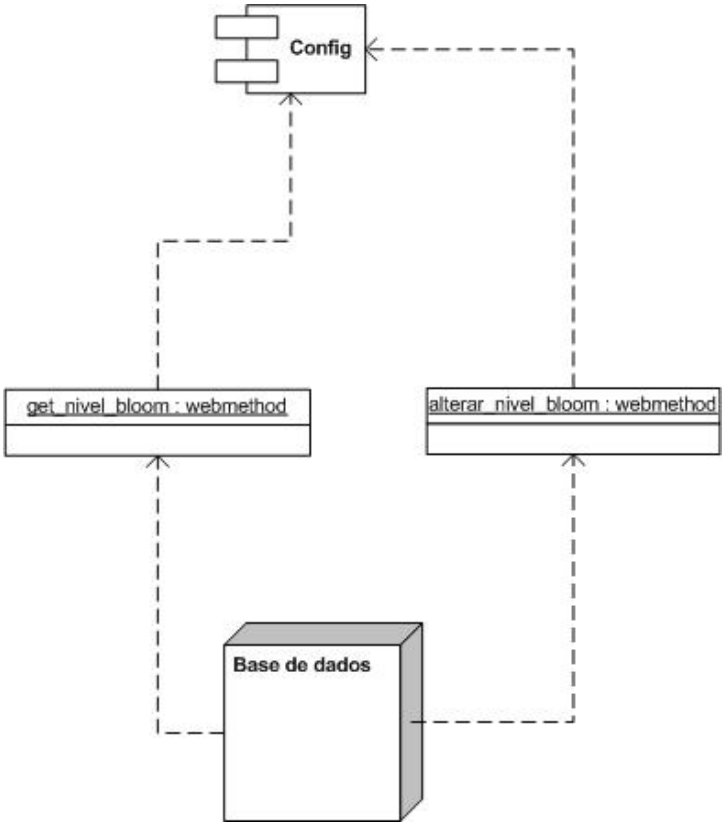

Figura 11: Objectos da serviços de dados

O esquema aqui apresentado demonstra como se processam os pedidos entre a camada de servicos de dados e a base de dados. Através destes métodos é possível realizar todos os pedidos feitos pelo *webmethod* da camada de lógica de negócio. A comunicação feita entre os *webmethods* e a base de dados é feita por SQL, enquanto comunicação feita com a camada de lógica de negócio é feita por XML. Por sua vez, a comunicação entre os *webmethods* e os *webservices* são feitas por ASP.

3.6.3 Objectos da base de dados

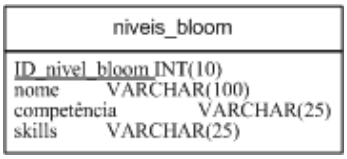

Figura 12: Objectos da base de dados

O unico objecto da base de dados utilizado por este protótipo é a tabela referente aos níveis Bloom.

3.6.4 Diagrama de actividades

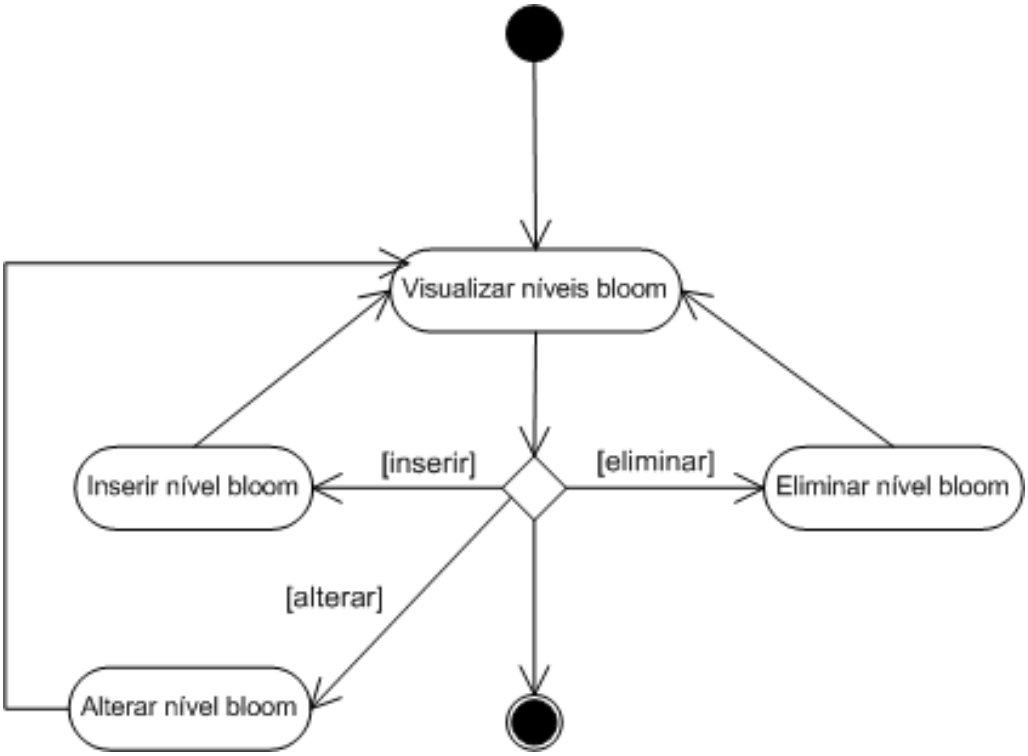

Figura 13: Diagrama de Actividades

3.6.5 Diagrama de sequência

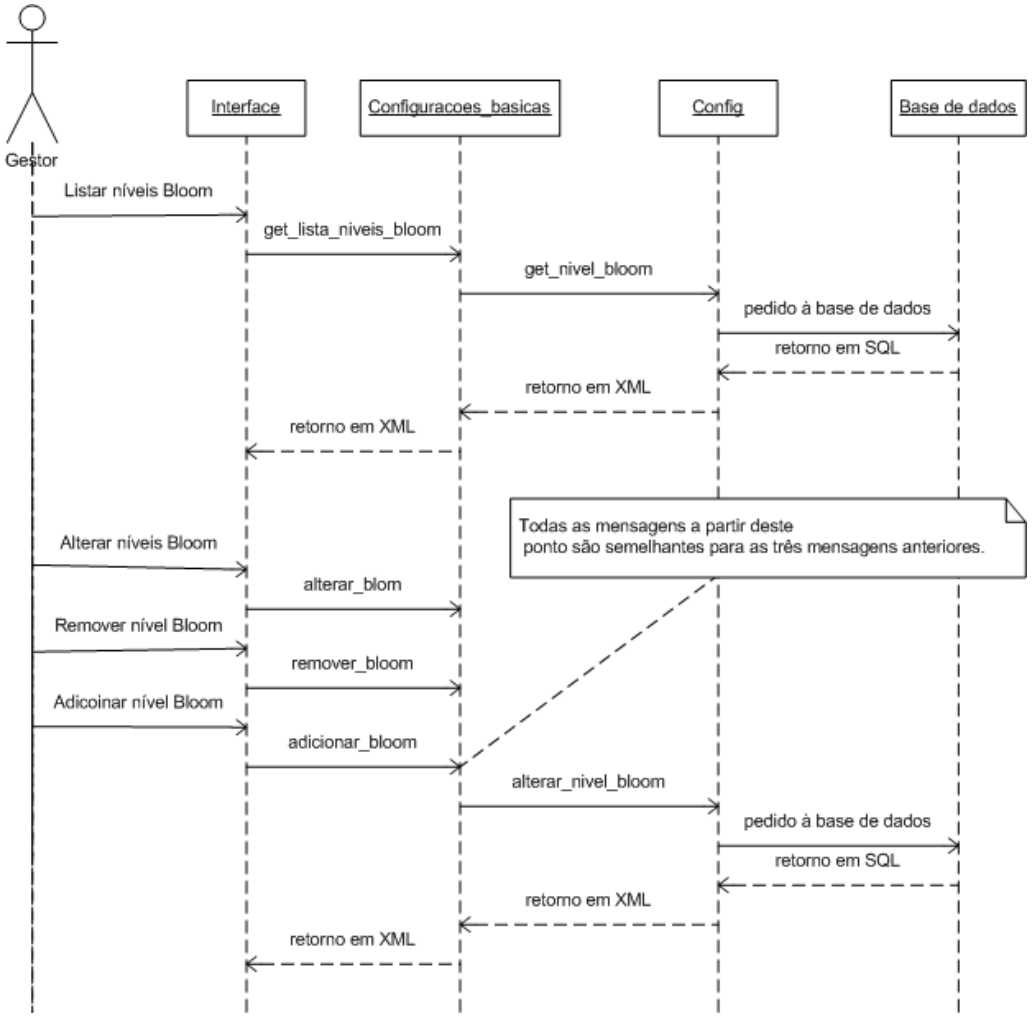

Figura 14: Diagrama de sequência

4 Planeamento

O planeamento deste projecto está mapeado por tarefas e pessoa associada à tarefa.

Assim teremos uma fase de arranque, em que estaremos os quatro a trabalhar na mesma tarefa, de forma a ambiemtarmo-nos por inteiro à tecnologia usada. Nesta fase inicial iremos então executar a tarefa da qual a criação dos restantes *webservices* está pendente: o *webservice* Configuracoes_basicas. Para além deste *webservice*, em paralelo iremos implementar um objecto para desenhar formulários que vai automatizar muito o controlo de erros, e o design de formulários. Para além desse objecto vamos criar outro que encapsula o acesso à base de dados.

Só depois destas duas tarefas serem terminadas podemos então dividir tarefas pelos quatro membros do grupo, de forma a optimizar ao máximo o trabalho. A primeira fase passa também por uma aprendizagem de utilização do sistema. Esta informação está descrita mais aprofundadamente no subcapítulo seguinte.

4.1 Diagrama de Gantt

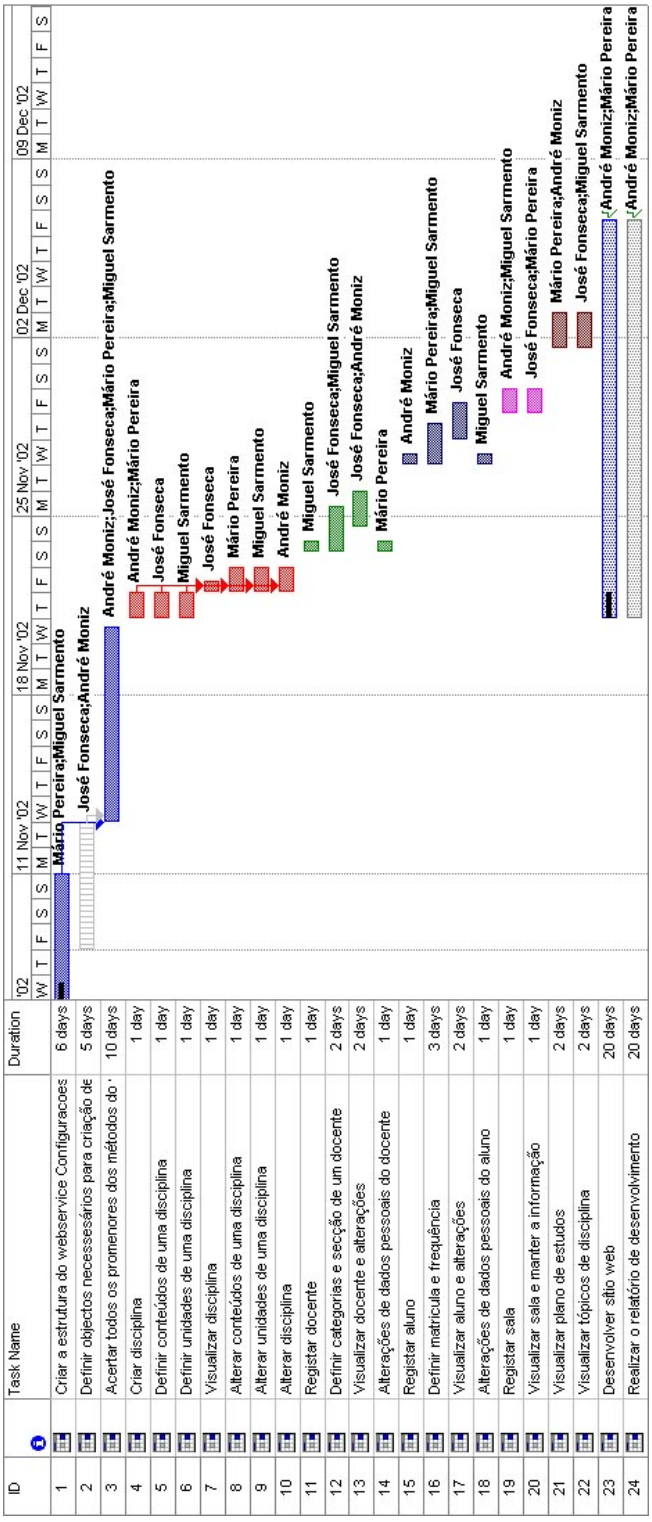

Figura 15: Diagrama de Gantt

4.2 Descrição das tarefas

4.2.1 Definir objecto necessesário para criação de formulários

Recursos:André Moniz e José Fonseca
Data início:08-11-2002
Data fim:12-11-2002
Completo:10%
Descrição:

A criação do objecto relacionado com a criação de formulários vai ser um dos processos chave na criação da apliacação. A partir do momento que este objecto estiver completo será muito fácil e intuitivo criar formulários e chamar *webservices* da lógica de negócio. O tratamento de erros fica também ao cargo deste objecto. Sem a criação deste objecto não podemos começar as tarefas de implementação dos outros *webservices*. Implementar um objecto para encapsular o acesso à base de dados.

4.2.2 Criar a estrutura do webservice Configuracoes_basicas

Recursos:Mário Pereira e Miguel Sarmento
Data início:06-11-2002
Data fim:10-11-2002
Completo:10%
Descrição:

Esta tarefa é primordial para o projecto, pois é uma fase de aprendizagem e de ambientação dos executantes. Inclui criar todos os *webmethods* deste *webservice*. Assim podemos dividir esta tarefa em outras pequenas tarefas que passam por realizar todos os casos de uso deste grupo em termo físicos sob a forma de funções (*webmethods*). O procedimento é relativamente semelhante para todos os casos de uso:

- criar o formulário de inserção
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados
- implementar o *webmethod* no serviço de dados para aceder à informação na base de dados

4.2.3 Acertar todos os promenores dos métodos do webservice Configuracoes_basicas

Recursos:Mário Pereira, Miguel Sarmento, André Moniz, José Fonseca
Data início:12-11-2002
Data fim:20-11-2002
Descrição:

Esta tarefa implica que todos trabalhemos em conjunto para nos ajudarmos nas dúvidas relativas ao objecto para criação de formulários, e também para implementarmos todos os *webmethods* deste *webservice*. Esta é uma fase de cooperação e em que todos vão trabalhar em todas as camadas para nas fases posteriores todos serem capazes de implementar um procedimento dentro da nossa arquitectura de cima para baixo e de baixo para cima.

4.2.4 Criar disciplina

Recursos:Mário Pereira e André Moniz
Data início:21-11-2002
Data fim:21-11-2002
Descrição:

- criar o formulário de inserção com os campos correspondentes.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados.
- implementar o *webmethod* no serviço de dados para aceder à informação na base de dados sobre as disciplinas.

4.2.5 Definir conteúdos de uma disciplina

Recursos:José Fonseca
Data início:21-11-2002
Data fim:21-11-2002
Descrição:

- criar o formulário de inserção.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados.
- implementar o *webmethod* no serviço de dados para aceder à informação na base de dados.

4.2.6 Definir unidades de uma disciplina

Recursos:Miguel Sarmento
Data início:21-11-2002
Data fim:21-11-2002
Descrição:

- criar o formulário de inserção
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados.
- implementar o *webmethod* no serviço de dados para aceder à informação na base de dados.

Teremos especial atenção para os pre-requisitos entre as unidades.

4.2.7 Visualizar disciplina

Recursos:José Fonseca
Data início:21-11-2002
Data fim:22-11-2002
Descrição:

- criar a página de visualização de todos os dados de uma disciplina.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai buscar toda a informação sobre uma disciplina.
- implementar o *webmethod* no serviço de dados para aceder à informação na base de dados sobre a disciplina. Este webmethod está no ficheiro disciplina.asmx e é como se fosse um objecto que espelha a entidade disciplina. Para toda a informação relativa a uma disciplina existirão métodos para retornar essa informação.

4.2.8 Alterar conteúdos de uma disciplina

Recursos:Mário Pereira
Data início:22-11-2002
Data fim:22-11-2002
Descrição:

- criar o formulário de alteração de contéudos de uma disciplina.

- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai buscar toda a informação sobre os conteúdos de uma disciplina.
- usar o já criado *webmethod* no serviço de dados para aceder à informação na base de dados sobre os conteúdos de uma disciplina. Implementar o webmethod para alterar os conteúdos.

4.2.9 Alterar unidades de uma disciplina

Recursos:Miguel Sarmento
Data início:22-11-2002
Data fim:22-11-2002
Descrição:

- criar o formulário de alteração de unidades de uma disciplina.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai buscar toda a informação sobre as unidades de uma disciplina.
- usar o já criado *webmethod* no serviço de dados para aceder à informação na base de dados sobre as unidades de uma disciplina. Implementar o webmethod para alterar as unidades. Mais uma vez ter em conta os pré-requesitos entre as unidades aquando da alteração.

4.2.10 Alterar disciplina

Recursos:André Moniz
Data início:22-11-2002
Data fim:22-11-2002
Descrição:

- criar o formulário de alteração de dados de uma disciplina.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai alterar a informação sobre uma disciplina.
- usar o já criado *webmethod* no serviço de dados para aceder à informação na base de dados sobre uma disciplina.

4.2.11 Registar docente

Recursos:Miguel Sarmento
Data início:23-11-2002
Data fim:23-11-2002
Descrição:

- criar o formulário de inserção de dados de um docente.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai inserir os dados de um docente.
- implementar o *webmethod* no serviço de dados para inserir os dados pessoais de um docente.

4.2.12 Definir categorias e secção de um docente

Recursos:José Fonseca, Miguel Sarmento
Data início:23-11-2002
Data fim:25-11-2002
Descrição:

- criar o formulário de inserção da categoria e da secção de um docente.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai inserir a categoria e a secção de um docente.
- implementar o *webmethod* no serviço de dados para inserir a secção e a categoria de um docente.

4.2.13 Visualizar docente e alterações

Recursos:José Fonseca, André Moniz
Data início:24-11-2002
Data fim:25-11-2002
Descrição:

- criar a página de visualização da informação de um docente.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai buscar os dados do docente á base de dados.

- implementar o *webmethod* no serviço de dados para obter toda a informação de um docente no ficheiro. docente.asmx.

4.2.14 Alterações de dados pessoais do docente

Recursos:Mário Pereira
Data início:23-11-2002
Data fim:23-11-2002
Descrição:

- criar o formulário de alteração de dados de um docente.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai alterar os dados de um docente.
- implementar o *webmethod* no serviço de dados para alterar os dados pessoais de um docente.

4.2.15 Registar aluno

Recursos:André Moniz
Data início:26-11-2002
Data fim:27-11-2002
Descrição:

- criar o formulário de inserção de dados de um aluno.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai inserir os dados de um aluno.
- implementar o *webmethod* no serviço de dados para inserir os dados pessoais de um aluno.

4.2.16 Definir matricula e frequência de um aluno

Recursos:José Fonseca, Miguel Sarmento
Data início:26-11-2002
Data fim:28-11-2002
Descrição:

- criar o formulário de inserção da categoria e da secção de um docente.

- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai inserir a categoria e a secção de um docente.
- implementar o *webmethod* no serviço de dados para inserir a secção e a categoria de um docente.

4.2.17 Visualizar aluno e alterações de frequência e matricula

Recursos:José Fonseca, André Moniz
Data início:28-11-2002
Data fim:29-11-2002
Descrição:

- criar a página de visualização da informação de um aluno.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai buscar os dados do aluno á base de dados. Implementar métodos para alterar a frequência de um aluno
- implementar o *webmethod* no serviço de dados para obter toda a informação de um aluno no ficheiro aluno.asmx. Também implementar métodos para alterar a frequência de um aluno.

4.2.18 Alterações de dados pessoais do aluno

Recursos:Mário Pereira
Data início:26-11-2002
Data fim:27-11-2002
Descrição:

- criar o formulário de alteração de dados de um aluno.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai alterar os dados de um aluno.
- implementar o *webmethod* no serviço de dados para alterar os dados pessoais de um aluno.

4.2.19 Registar sala

Recursos:André Moniz, Miguel Sarmento
Data início:29-11-2002
Data fim:29-11-2002
Descrição:

- criar o formulário de inserção de uma sala.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai inserir os dados de uma determinada sala.
- implementar o *webmethod* no serviço de dados para inserir os dados pessoais de uma sala.

4.2.20 Visualizar sala e manter a informação

Recursos:José Fonseca, Mário Pereira
Data início:29-11-2002
Data fim:29-11-2002
Descrição:

- criar a página de visualização da informação de uma sala.
- implementar o *webmethod* na lógica de negócio que invoca um *webmethod* do *webservice* do serviço de dados que vai buscar os dados da sala á base de dados.
- implementar o *webmethod* no serviço de dados para obter toda a informação de uma sala no ficheiro sala.asmx.

4.2.21 Visualizar plano de estudos

Recursos:André Moniz, Mário Pereira
Data início:01-12-2002
Data fim:02-12-2002
Descrição:

- criar a página de visualização do plano de estudos.
- implementar o *webmethod* na lógica de negócio que retorna toda a informação sobre o plano de estudos.
- implementar o *webmethod* no serviço de dados para obter toda a informação do plano de estudos no ficheiro plano_estudos.asmx.

4.2.22 Visualizar tópicos de disciplina

Recursos:José Fonseca, Miguel Sarmento
Data início:01-12-2002
Data fim:02-12-2002
Descrição:

- criar a página de visualização dos tópicos de uma disciplina.
- implementar o *webmethod* na lógica de negócio que retorna toda a informação sobre os tópicos de uma dada disciplina.
- implementar o *webmethod* no serviço de dados para obter toda a informação dos tópicos das disciplinas no ficheiro plano_estudos.asmx.

4.2.23 Desenvolver sítio web

Recursos:André Moniz, Mário Pereira, José Fonseca, Miguel Sarmento
Data início:21-11-2002
Data fim:06-12-2002
Descrição:

Esta é uma tarefa que se vai desenvolvendo à medida que se vai executando as outras tarefas. Inclui criar o sitio web com todos os links necessários e com a camada de design sobre a de funcionalidade.

4.2.24 Realizar o relatório de desenvolvimento

Recursos:André Moniz, Mário Pereira, José Fonseca, Miguel Sarmento
Data início:21-11-2002
Data fim:06-12-2002
Descrição:

Realizar o relatório de desenvolvimento à medida que se vai realizando as outra tarefas de implementação.

5 Conclusão

Todos os projectos necessitam de uma boa estruturação e planeamento. Essa estruturação e planeamento começa na definição dos requisitos do sistema (apresentados no Relatório de Especificação de Requisitos) e culmina na estruturação da arquitectura física/lógica e planeamento das tarefas.

Julgamos ser de extrema importância na nossa formação enquanto engenheiros realizarmos uma análise deste tipo de um projecto. Qualquer aplicação necessita de uma análise de requisitos, mas pensar na arquitectura e projectar o planeamento torna-se fundamental para prever e antecipar todos os problemas que possam surgir ao longo da implementação. Estruturando arquitecturas e definindo tarefas para a implementação conseguem-se aplicações robustas (de fácil manutenção e actualização) mantendo a elegância da solução.

Portanto consideramos muito proveitoso a realização deste tipo de trabalho a nível académico, pois muito nos vai ajudar no futuro próximo.

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.