Relatório de Alto Nível

(19/04/2002 – Versão 4.0)

Gestão de Beneficiários

Universidade do Porto

Faculdade de Engenharia

FEUP

Eduardo Abreu – ei98020@fe.up.pt
Miguel David – ei98019@fe.up.pt
Nuno Ferreira – ei98003@fe.up.pt
Tiago Silva – ei98015@fe.up.pt

Prefácio

Este relatório, sendo já um relatório com algum grau de implementação, ainda não será com certeza a versão final do projecto, uma vez que há menos de uma semana o grupo começou a trabalhar na tecnologia em questão (a plataforma .NET). E com esta, toda a sua nova arquitectura.

Esperamos que as ideias presentes neste relatório não se afastem muito do que o projecto realmente vai ser na fase da implementação.

Sumário

1. Definição da Arquitectura

1.1 Arquitectura lógica

Para caracterizar a arquitectura do sistema, dividi-la-emos, primeiramente em camadas horizontais e, em seguida, em camadas horizontais.

1.1.1 Divisão horizontal

No que toca à divisão horizontal, temos:

- a camada de Interfaces/Componentes Cliente
- a camada de Lógica de Negócio/Componentes Servidor
- a camada de Base de Dados.

Há uma interface presente na aplicação que é a ligação *ODBC* utilizada para estabelecer a ligação entre as camadas de **Lógica de Negócio** e **Base de Dados**.

Podemos também considerar uma interface exterior à aplicação (mencionada como nota no diagrama) que faz a interacção entre a aplicação em discussão e a aplicação do Banco (no que toca a transferências de dinheiro) através de *SOAP*.

A seguir, apresenta-se o diagrama que representa a composição horizontal do sistema.

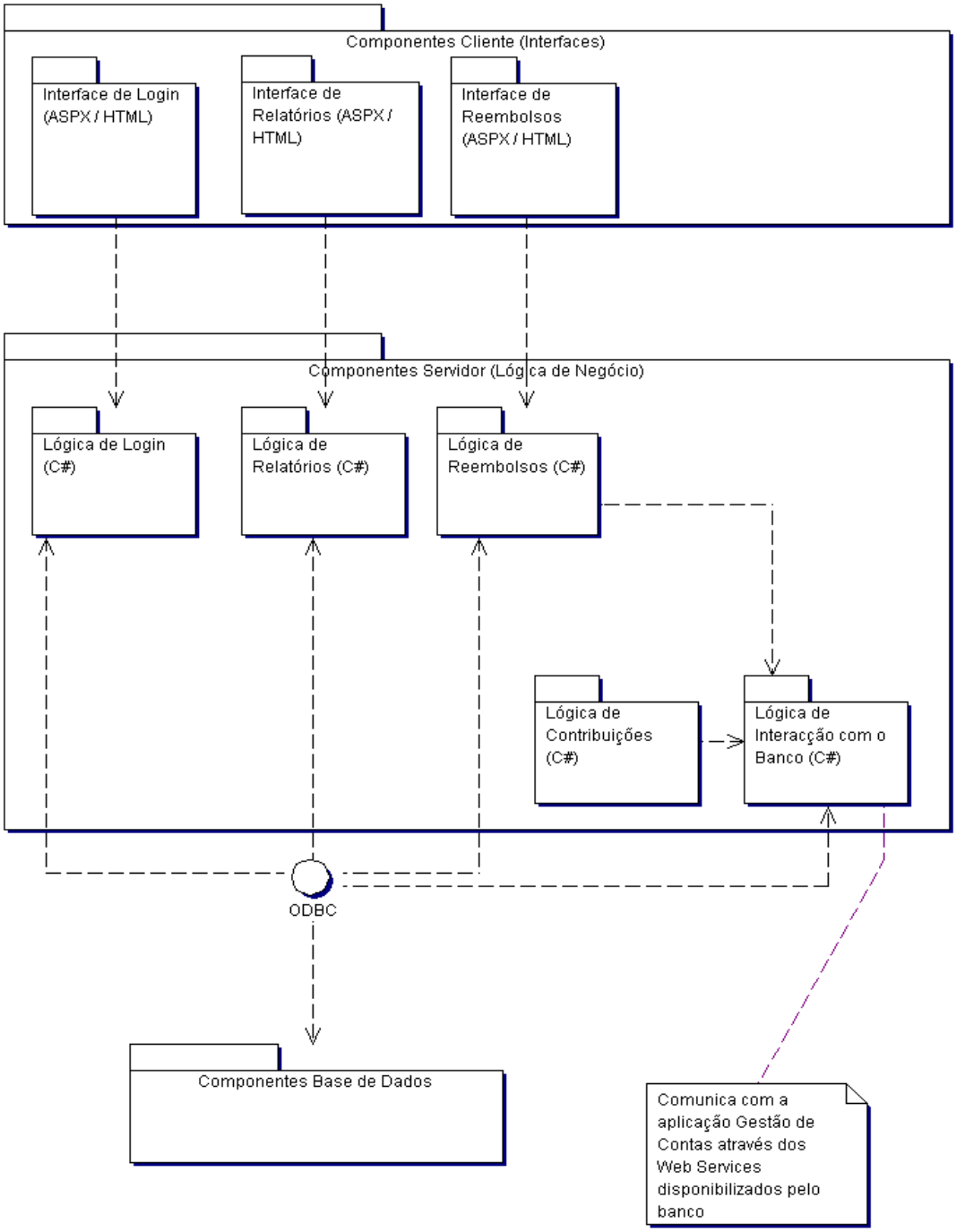


Figura 1 – Arquitectura lógica – divisão horizontal

Camada de Interfaces

A camada de interfaces é a única visível para o utilizador final do sistema. É esta que, juntamente com a camada de Lógica de Negócio, gera dinamicamente e apresenta páginas *html* ao utilizador, sejam elas formulários estáticos, ou resultados dinâmicos de uma conta-corrente de um beneficiário.

Camada de Lógica de Negócio

A lógica de negócio estabelece a ligação entre a camada de interfaces e o sistema de gestão de base de dados (via *ODBC*), tratando os dados de modo a que estes se

encontrem de acordo com as regras de negócio do sistema. É a este nível que a aplicação fornece e acede a funcionalidades de outras aplicações (via *SOAP*).

Camada de Base de Dados

Esta camada é a responsável pela persistência dos dados da aplicação.

1.1.2 Divisão vertical

Decompondo verticalmente a arquitectura do projecto de acordo com as funcionalidades proporcionadas, obtêm-se quatro pacotes lógicos que são apresentados na figura 2. Os pacotes que contêm os casos de uso do sistema incluem já os componentes que tratam da interface com o utilizador e da lógica de negócio.

Login

Este pacote lógico engloba os componentes que permitem que um utilizador se autentique perante o sistema. Engloba os pacotes que geram as interfaces e os pacotes que tratam a lógica de negócio relativa à autenticação.

Relatórios

Este pacote lógico contém componentes que permitem a visualização de relatórios relativos à empresa pagadora de serviços e aos beneficiários do sistema.

Reembolsos

Este pacote lógico, para além de incluir os componentes referentes às interfaces com o utilizador e à lógica de negócio, contém os componentes que tratam da interacção com o banco através de *Web Services*.

Contribuições

O pacote lógico das contribuições é idêntico ao dos reembolsos, incluindo também os componentes que tratam da interacção com o banco através de *Web Services*.

Nesta decomposição vertical foi colocado o pacote **Base de Dados** abaixo de todos os outros pacotes lógicos, visto que este os suporta a todos.

A seguir, apresenta-se o diagrama da decomposição vertical do sistema.

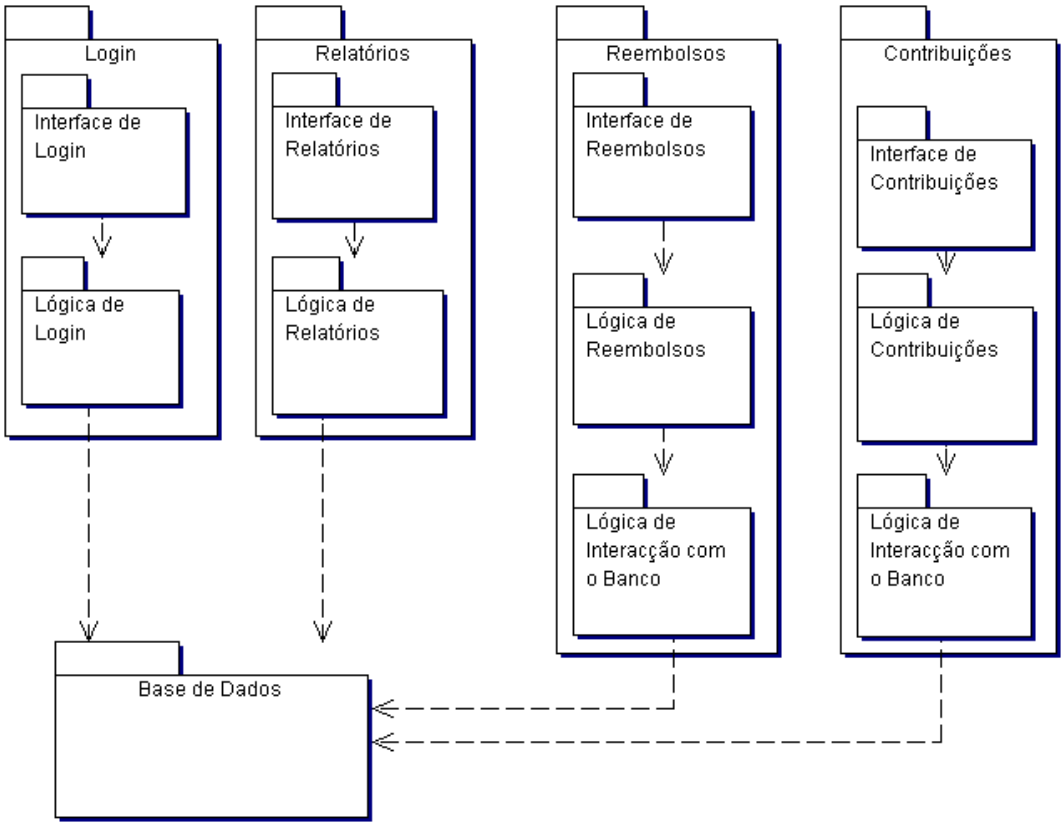


Figura 2 – Arquitectura lógica – divisão vertical

1.2 Mecanismos importantes

1.2.1 Erros de introdução de dados pelo utilizador

Caso o utilizador tenha que introduzir dados no sistema e não o faça ou insira dados inválidos, o sistema indica quais os dados inválidos ou omissos, dando ao utilizador a possibilidade de corrigir os erros.

1.2.2 Resultados de pesquisas vazios

No caso em que os utilizadores pretendam pesquisar a base de dados do sistema (pesquisando beneficiários, por exemplo) e a pesquisa não devolver resultados, o sistema notifica o utilizador do resultado negativo, dando-lhe a possibilidade de repetir a pesquisa (podendo o utilizador alterar os dados da primeira pesquisa).

1.2.3 Mecanismo de passagem de parâmetros entre páginas

Quando há dados que transitam entre páginas (como por exemplo: a inscrição do beneficiário), os dados vão ao servidor onde são tratados[1] e depois enviados de novo para o cliente na forma de uma nova página.

1.3 Arquitectura física de software

O esquema referente à organização dos componentes de *run-time* do sistema encontra-se em anexo.

1.4 Arquitectura física de hardware

A arquitectura física do sistema está representada na figura 3. Um cliente na sua máquina acede através de um *browser* à aplicação (um conjunto de páginas dinâmicas) que são geradas no Servidor *Web* no componente de interfaces. Mas este componente recorre a outro – a Lógica de Negócio – que por sua vez vai buscar os dados a um servidor de base de dados e os trata. O processo também é percorrido em sentido inverso, recorrendo ao servidor do Banco sempre que tal for necessário (como é o caso das contribuições e dos reembolsos).

[1] A plataforma .NET tem funcionalidades que permitem que dados que tenham que transitar entre componentes o façam sem ser por *URL*, *POST* ou *GET*.

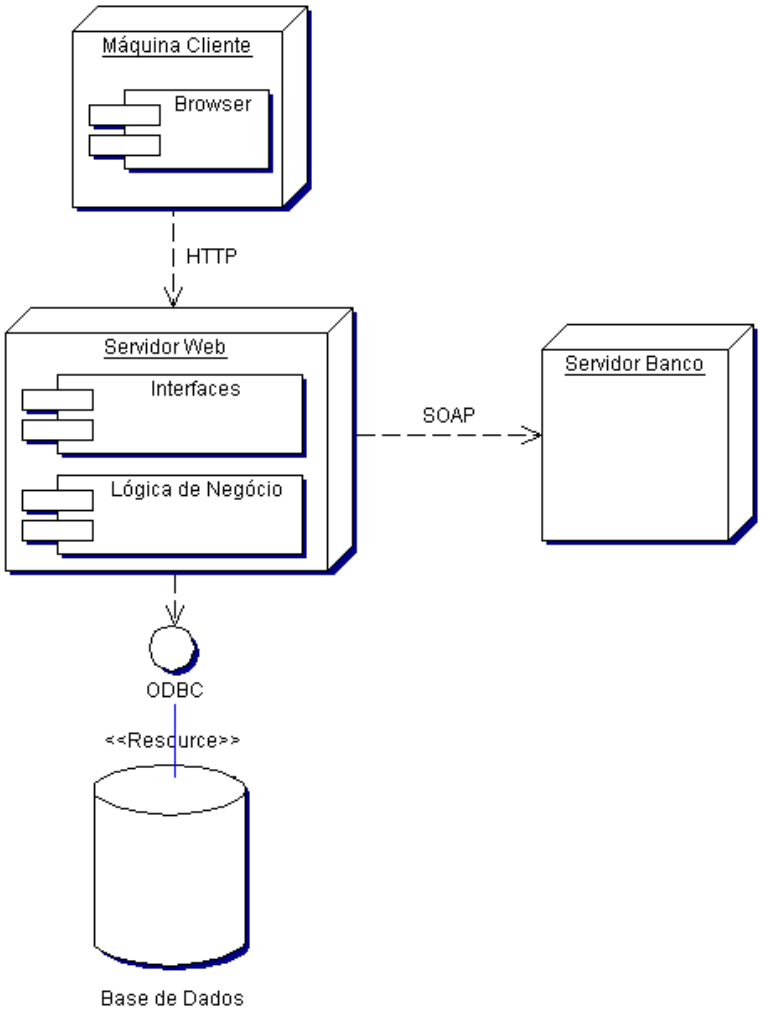


Figura 3 – Diagrama de *deployment* (Arquitectura Física)

1.5 Arquitectura tecnológica

Plataforma de Desenvolvimento

Escolhemos a plataforma *.NET* para o desenvolvimento deste projecto:

1. para experimentar uma plataforma nova
2. porque o mercado de trabalho começa a pedir esse tipo de programação
3. pelas suas características inovadoras

Linguagem de Programação

Escolhemos o *C#* por oposição ao *Visual Basic .NET* para programar porque:

1. esta é uma linguagem orientada aos objectos e à reutilização de componentes
2. a sua sintaxe é muito semelhante à do *Java*, linguagem da *Sun* já nossa conhecida

Outras ferramentas

Para a concretização da análise de requisitos e concepção do projecto foi utilizado o Together. Esta ferramenta já inclui facilidades de implementação, pelo que impõe fortes limitações no que toca a diagramas. Por isso foi também utilizado o MS Visio 2002.

Para o desenho das interfaces com os utilizadores é utilizado o Corel Draw 10.

Linguagem de comunicação entre aplicações

O *XML* não foi escolhido por nós, é uma escolha feita a partir do momento em que se escolhe a plataforma *.NET*, uma vez que esta está embebido na plataforma.

Isto porque o *XML* é cada vez mais uma linguagem universal no mundo dos computadores, e assim podemos programar um *Web Service* em qualquer plataforma e desde que o *output* seja *XML*, qualquer outra plataforma poderá ler essa informação e tratá-la.

Na plataforma *.NET* o *XML* é utilizado não só como meio de comunicação entre *Web Services* como também é o próprio formato para as definições das aplicações como se pode ver em ficheiros *web.config* que são ficheiros para definir, entre outras coisas, as permissões de acesso aos objectos no directório em que esses ficheiros estão colocados.

Protocolo de comunicação entre aplicações

O protocolo utilizado é o *SOAP* (Simple Object Access Protocol) que permite que aplicações troquem informação entre si. Este protocolo também está implícito na plataforma .NET, não foi escolha nossa.

Sistema de Gestão de Base de Dados

O SGBD utilizado é o *Oracle 9i* por oposição ao *SQL Server* da *Microsoft* porque apesar de já haver métodos implementados e optimizados para o *SQL Server* na plataforma *.NET*, queremos que o projecto tenha alguma flexibilidade, de modo a podermos depois portar facilmente a aplicação para qualquer outro SGBD suportado por *ODBC*. Neste caso, vamos também utilizar a tecnologia *ADO.NET*.

Ambiente de programação

Optámos por escolher o *IDE Visual Studio*, em vez de trabalhar no *Together* ou mesmo em editores de texto porque:

1. é um ambiente integrado de desenvolvimento, isto é, permite escrever código, compilá-lo, testá-lo e depurá-lo
2. porque o *Together* é mais orientado à plataforma da *Sun* (*Entreprise Java Beans*)

Web Services

A escolha de *web services* também foi imposta pelos professores da cadeira. Esta tecnologia emergente deve permitir no futuro que as aplicações sejam todas distribuídas e acessíveis em qualquer ponto do mundo desde que se tenha um *browser* e um acesso à internet.

2 Documentação do Protótipo

As funcionalidades implementadas no protótipo são as de autenticação do utilizador e de inscrição de novos beneficiários no sistema.

Foi ainda adicionada ao protótipo uma classe que pretende simular o *Web Service* que o banco deverá fornecer.

2.1 Descrição de componentes de Interface com o Utilizador

Os componentes de interface com o utilizador deste protótipo são relativos às páginas iniciais e à funcionalidade de inscrição de beneficiários. As páginas iniciais são as de menu para os utilizadores (beneficiários e funcionários) e o login do sistema (onde o utilizador se deverá autenticar).

A figura 4 mostra a estrutura dos componentes de interface com o utilizador.

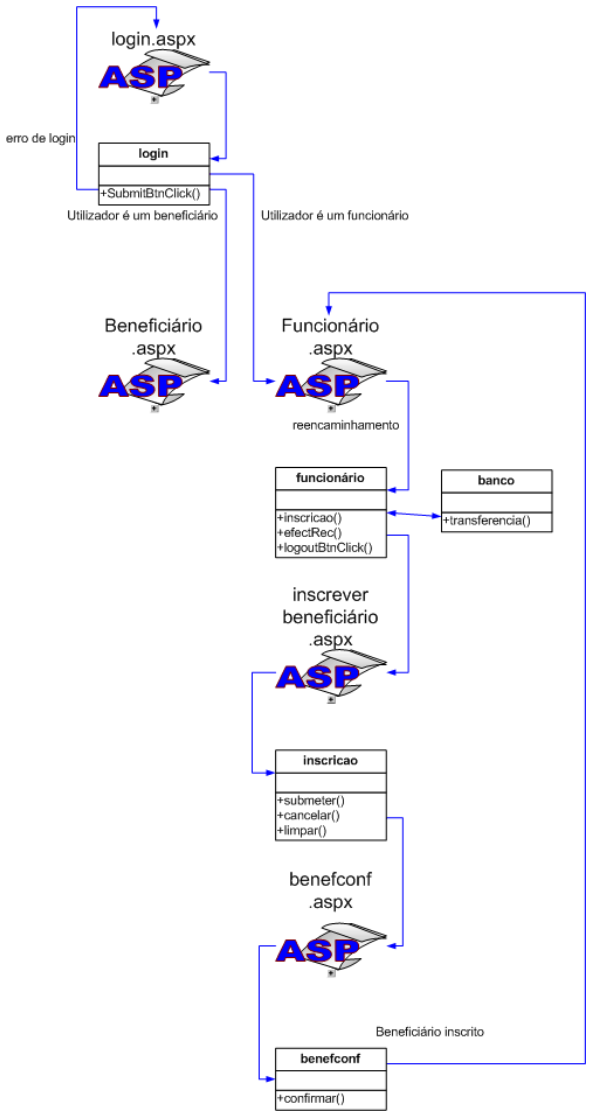


Figura 4 – Diagrama de Componentes do Protótipo

Login.aspx

Este componente é a página de entrada do sistema. Apresenta aos utilizadores o formulário de autenticação e caso o utilizador não seja autorizado a aceder ao sistema, aparece uma mensagem de erro.

No caso de o utilizador ser autenticado pelo sistema, terá acesso às funcionalidades da aplicação de acordo com o seu tipo.

A classe que implementa a lógica de negócio e suporta a implementação desta funcionalidade é a classe **login**.

Beneficiário.aspx

Este componente é o menu principal para os beneficiários.

Funcionário.aspx

Este componente é o menu principal para os utilizadores funcionários. As opções do menu são a inscrição de beneficiários e o processamento de contribuições (esta funcionalidade é simulada, e serve de teste do *Web Service*).

Inscrever beneficiário.aspx

Este componente permite que o utilizador funcionário adicione um novo beneficiário ao sistema através do preenchimento de um formulário onde deverão constar os dados pessoais do beneficiário relevantes ao sistema.

A suportar este componente e a implementar a sua lógica de negócio, existe a classe **inscricao**.

benefconf.aspx

Este componente notifica o utilizador que a inserção do beneficiário foi bem sucedida.

2.2 Descrição de componentes da Lógica de Negócio

Os componentes da lógica de negócio são classes que recebem os dados dos componentes de interface com o utilizador, os validam e executam as devidas operações.

As classes **login.aspx**, **beneficiario.aspx** e **funcionario.aspx** são acompanhadas respectivamente pelas classes **login, inscricao** e **funcionário**.

O componente que simula o *Web Service* a ser disponibilizado pelo banco é a classe **banco**.

login

Este componente de lógica de negócio implementa as regras de autenticação no sistema. Recebe do componente que o usa o: login, password e tipo do utilizador e verifica se este está autorizado a aceder ao sistema.

Se o utilizador for um beneficiário autenticado pelo sistema é reencaminhado para a página menu do beneficiário, se for um funcionário é reencaminhado para a página menu dos funcionários.

funcionário

Este componente trata do reencaminhamento do funcionário para as diferentes funcionalidades. No entanto, não está implementado no protótipo, mas será útil caso se pretenda controlar os acessos e utilização do sistema.

inscricao

Esta classe é responsável pela inserção de novos beneficiários no sistema. Acompanha o componente de interface com o utilizador **inscrever beneficiário.aspx** e recebe deste os dados do novo beneficiário, inserindo-os na base de dados.

benefconf

Esta classe é responsável pelo reencaminhamento do utilizador de volta para o seu menu, após a inserção de um beneficiário. Futuramente, permitirá fazer algum tratamento de erros.

banco

Esta classe implementa um *Web Service* e é utilizada apenas para simular a chamada remota a um serviço de transferência bancária.

2.3 Descrição da Base de Dados

Os componentes presentes na base de dados são as tabelas relacionais onde são mantidos os dados persistentes do sistema. O modelo de classes da base de dados encontra-se em anexo.

3 Planeamento do Projecto

O planeamento do projecto procura equilibrar a distribuição de tarefas entre os membros da equipa de trabalho, tendo em conta que na semana de 29 de Abril a 3 de Maio deverá ser disponibilizada ao cliente parte do sistema funcional.

	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	Interface e lógica de login	5 days?	Mon 22-04-02	Fri 26-04-02		Nuno Ferreira;Eduardo Abreu
2	Interface e lógica de relatórios	5 days?	Mon 22-04-02	Fri 26-04-02		Miguel David;Tiago Silva
3	Entrega de "release" intermédia	5 days?	Mon 29-04-02	Fri 03-05-02		
4	Interface de contribuições	5 days?	Mon 29-04-02	Fri 03-05-02		Nuno Ferreira;Eduardo Abreu
5	Lógica de contribuições	5 days?	Mon 29-04-02	Fri 03-05-02		Miguel David;Tiago Silva
6	Queima das fitas	5 days	Mon 06-05-02	Fri 10-05-02		
7	Interface de reembolsos	5 days?	Mon 13-05-02	Fri 17-05-02		Miguel David;Tiago Silva
8	Lógica de reembolsos	5 days?	Mon 13-05-02	Fri 17-05-02		Nuno Ferreira;Eduardo Abreu
9	Ligação à aplicação do Banco com Web Services	4 days?	Mon 20-05-02	Thu 23-05-02		Nuno Ferreira[50%];Eduardo Abreu[50%]
10	Manual de utilizador	4 days?	Mon 20-05-02	Thu 23-05-02		Nuno Ferreira[50%];Eduardo Abreu[50%]
11	Escrita do relatório de desenvolvimento	4 days?	Mon 20-05-02	Thu 23-05-02		Miguel David;Tiago Silva
12	Entrega do produto final, relatório de desenvolvimento e manual de utilizador	1 day	Fri 24-05-02	Fri 24-05-02		

Figura 5 – Planeamento do projecto

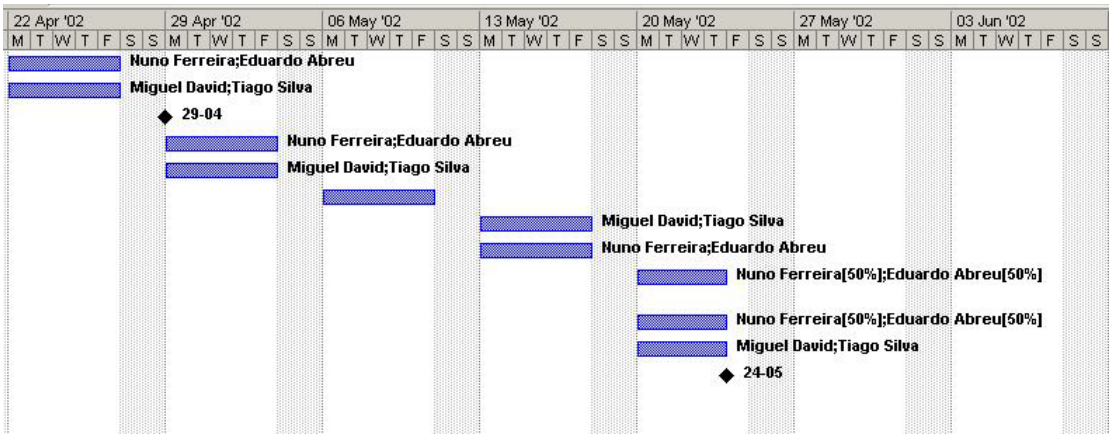


Figura 6 – Planeamento do projecto (diagrama de Gantt)

4 Glossário

ADO.NET – tecnologia nova de acesso a bases de dados feita para a plataforma .NET

ASPX – evolução da linguagem de programação para a *web* ASP para os novos conceitos da plataforma .NET

C# - pelo nome poder-se-ia pensar que era uma evolução do C, mas é mais uma aproximação entre o C^{++} e o Java, eliminando (quase) os apontadores do C^{++}.

Plataforma .NET – plataforma de programação criada pela Microsoft que se baseia em interoperabilidade com o código já existente, integração de linguagens de programação com o CLR (*Common Language Runtime*) em que qualquer linguagem que respeite os princípios do CLR pode criar objectos ou utilizar objectos criados por outras liguagens em *runtime*. Também se baseia no conceito de *Web Service*, em *Garbage Collection* numa linguagem intermédia IL que pode ser compilada *Just-In-Time*.

SOAP – *Simple Object Access Protocol*

Together – Ferramenta CASE que auxilia o projecto e desenvolvimento de *software*.

Visual Studio – Ambiente integrado de programação da Microsoft

Web Services – pedaços de código que podem ser invocados remotamente por HTTP

XML – eXtensible Markup Language, uma meta-linguagem que se está a tornar a língua franca do mundo da programação

5 Bibliografia

Livros

Troelsen, Andrew
C# and the .NET Platform
Apress, 2001

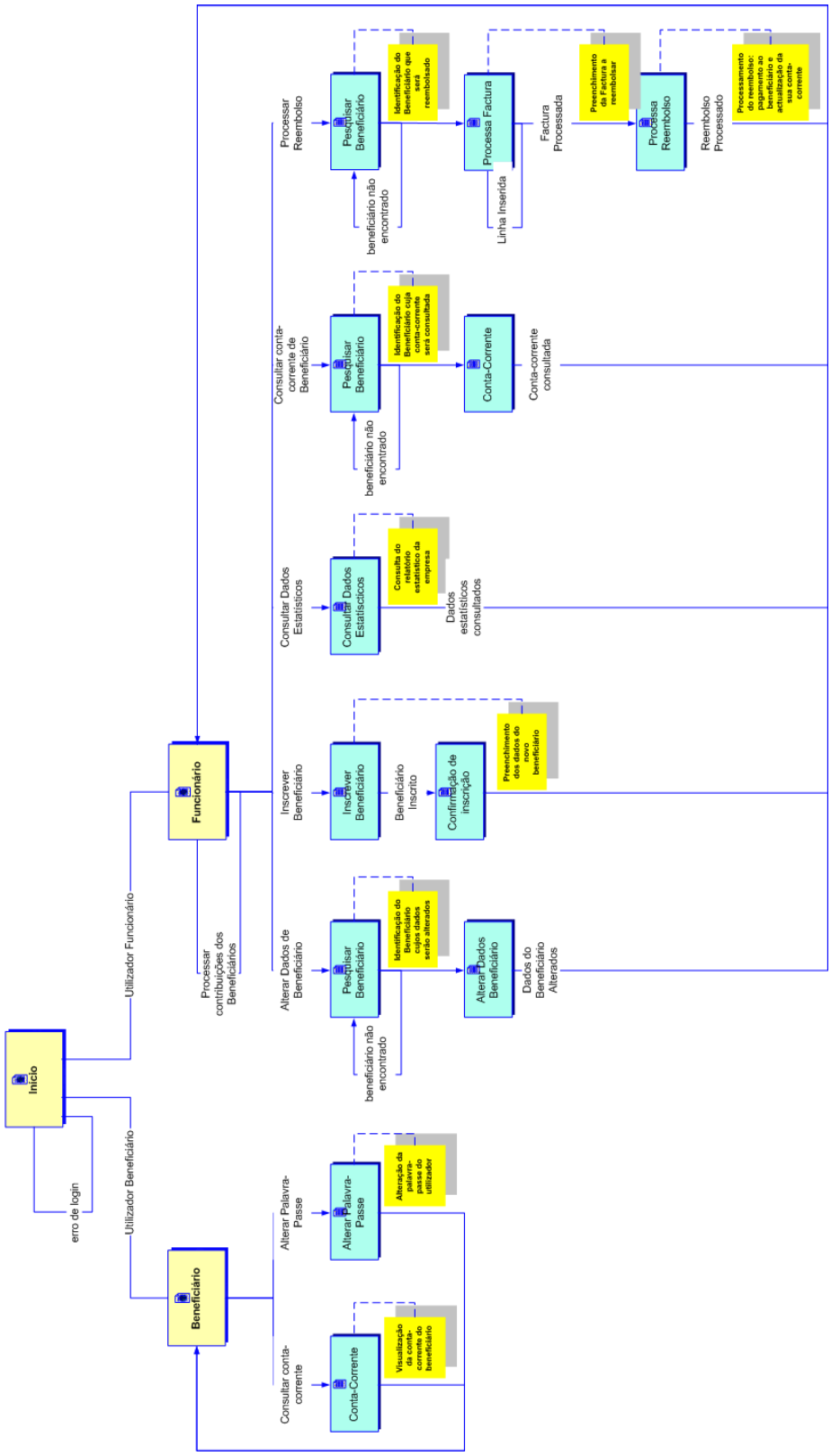
Gunnerson, Eric
A Programmer's Introduction to C#
Apress, 2000

Endereços Web

http://www.codeproject.com/
http://www.mastercsharp.com/
http://www.csharphelp.com/
http://www.cshrp.net/
http://www.hitmill.com/programming/dotNET/csharp.html
http://www.csharp-station.com/
http://samples.godotnet.com/

6 Anexos

6.1 Estrutura física de software



6.2 Modelo de classes da base de dados