Relatório de Desenvolvimento

(25/05/2002 – Versão 1.0)

Gestão de Beneficiários – P7

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

Índice

Introdução

Este documento descreve a implementação da aplicação de Gestão de Beneficiários. Na primeira fase do relatório serão descritos os componentes do sistema. Na segunda fase é referida a implementação dos casos de uso.

As arquitecturas deste sistema, devido à escolha tecnológica (plataforma tecnológica .NET), não são aquilo que esperaríamos de uma aplicação *Web*. A nossa experiência fazia-nos contar com uma arquitectura em 3 camadas claramente distintas (o que se passou em trabalhos passados com os JAVA BEANS). Os JAVA BEANS permitiam separar nitidamente as camadas de interface com o utilizador e de lógica de negócio. Com a plataforma eleita neste trabalho, não existe uma clara separação entre estas camadas, o que, apesar de não dificultar a implementação, torna a especificação e descrição da arquitectura e mesmo descrição da implementação mais difíceis de clarificar.

Descrição dos Componentes

É possível distinguir dois tipos de componentes que correm a nível do servidor *Web* (lógica de negócio, interfaces gráficas e *Web Services*):

- Componentes sem interacção com o utilizador: os componentes que implementam os *Web Services*. Estes componentes constituem classes cujos métodos são chamados por aplicações externas ao sistema via SOAP e XML. Interagem apenas com as aplicações externas e com a base de dados.
- Componentes com interacção com o utilizador: cada componente que interaja com os utilizadores é, na verdade composto por dois:
  - Componente Gráfico: o componente que mostra os elementos de interface com o utilizador. Estes componentes são páginas *ASP*, que se encontram definidos em ficheiros com extensão “aspx”.
  - Componente de Lógica de Negócio: o componente que trata da inicialização do componente gráfico, gere a interacção com o utilizador e com outros componentes. Estes componentes encontram-se definidos em ficheiros com extensão “aspx.cs”.

  São estes os componentes que dão ao sistema uma arquitectura difícil de definir. Embora sejam componentes perfeitamente distintos (definidos em ficheiros diferentes e com funcionalidades e responsabilidades diferentes), quando há um componente gráfico em carregamento, há um componente de Lógica de Negócio em execução.

Os dados do sistema são mantidos numa base de dados relacional.

alterar

Estes componentes permitem a alteração de dados pessoais de um beneficiário. Participam nos casos de uso Inscrever Beneficiário e Alterar dados de Beneficiário.

Componente Gráfico - alterar.aspx

A componente gráfica é constituída por um formulário já preenchido com os dados do utilizador. Possui os botões **Submeter** e **Cancelar**. O botão **Submeter** chama o método `submeter()` que efectua as alterações. O botão **Cancelar** chama o método `cancelar()` que redirecciona para o componente `funcionario.aspx`.

Os componentes do formulário devem estar sempre devidamente preenchidos. Para garantir a validade do preenchimento do formulário, são utilizadas as funcionalidades *ASP*:

- `RequiredFieldValidator`: verifica se um campo se encontra preenchido. Actua sobre os campos **Nome**, **Morada** e **Nib**.
- `RegularExpressionValidator`: verifica se o conteúdo de um campo tem o formato esperado. Actua sobre os campos do **Código Postal** (**cod1** tem que conter 4 números e **cod2** tem que conter 3 números), **Telefone** que tem que possuir 9 números e **Data de Nascimento** que deve respeitar o formato de data “dd/mm/aaaa".

Componente de Lógica de Negócio - alterar.aspx.cs

Componente *code behind* que apoia o componente `alterar.aspx` e implementa as regras de negócio. Implementa os métodos:

- `Page_Load()`: executado sempre que o componente `funcionario.aspx` é carregado. Começa por verificar a sessão (se o utilizador se encontra devidamente autenticado). Obtém o código do beneficiário cujos dados deverão ser alterados a partir da sessão. Obtém os dados pessoais da base de dados e preenche o formulário de alteração.
- `submeter()`: chamado sempre que é pressionado o botão **Submeter**, converte os dados do formulário para o formato conhecido pela base de dados e efectua a alteração.
- `cancelar()`: chamado quando é pressionado o botão **Cancelar**, redirecciona para o `funcionário.aspx`.

alterarpwd

Estes componentes permitem a alteração da palavra passe de um beneficiário e participam no caso de uso Alterar Palavra Passe.

Componente Gráfico - alterarpwd.aspx

O componente gráfico é constituído por um formulário que pede para escrever a palavra passe antiga e a nova palavra passe por duas vezes. Contém os campos **Password Antiga**, **Nova Password** e **Reescrever Password**. Contém os botões **Submeter**, que chama o método `SubmitBtnClick()` e permite efectuar a alteração e **Cancelar** que chama o método `Cancelar()` e faz regressar ao menu de beneficiários sem alterar a palavra passe.

Para alterar a palavra passe, todos os campos têm que ser preenchidos e o campo **Reescrever Password** tem que ter conteúdo igual ao do campo **Nova Password**. Assim a funcionalidade *ASP* `RequiredFieldValidator` (que verifica se determinado campo se encontra preenchido) actua sobre todos os campos e a funcionalidade `CompareValidator` (que verifica se um campo está preenchido com o mesmo conteúdo que outro campo) actua sobre **Reescrever Password**.

Componente de Lógica de Negócio - alterarpwd.aspx.cs

Este é o componente *code behind* que apoia o componente `alterarpwd.aspx` e implementa as regras de negócio. Implementa os métodos:

- `Page_Load()`: executado sempre que o componente `funcionario.aspx` é carregado. Começa por verificar na sessão se o utilizador se encontra devidamente autenticado como beneficiário.
- `SubmitBtnClick`(): chamado sempre que é pressionado o botão **Submeter**. Antes de submeter a nova palavra passe, verifica se os campos **Nova Password** e **Reescrever Password** são iguais e se o campo **Password Antiga** é igual à sua actual palavra passe.
- `cancelar()`: chamado quando é pressionado o botão **Cancelar**, redirecciona para o componente `beneficiario.aspx`.

beneconf

Estes componentes apresentam os dados do beneficiário que foi inserido para confirmação e possível alteração. Participam nos casos de uso Inscrever Beneficiário e Alterar Dados de Beneficiário.

Componente Gráfico - beneconf.aspx

A componente gráfica é constituída por um conjunto de etiquetas que são preenchidas com os dados dos beneficiários e os botões **confirmar** e **Corrigir**.

Componente de Lógica de Negócio - beneconf.aspx.cs

Este é o componente *code behind* que apoia o componente `beneconf.aspx` e implementa as regras de negócio. Implementa os métodos:

- `Page_Load()`: executado sempre que o componente `funcionario.aspx` é carregado. Começa por verificar na sessão se o utilizador se encontra devidamente autenticado como funcionário. Vai buscar os dados do beneficiário inserido, copiando a instância da classe `inscricao` (definida no componente `inscricao.aspx.cs`) que estava activa na página anterior.
- `corrigir()`: chamado sempre que é pressionado o botão **corrigir**, redirecciona para o componente `alterar.aspx`.
- `confirmar()`: chamado quando é pressionado o botão **confirmar**, redirecciona para o `funcionario.aspx`.

beneficiario

Estes componentes apresentam o menu para os beneficiários. É o ponto de partida para os casos de uso que podem ser utilizados pelos beneficiários (Consultar Conta Corrente do Beneficiário e Alterar Palavra Passe).

Componente Gráfico - beneficiario.aspx

Este componente apresenta o conjunto de botões que permitem que um beneficiário consulte a sua conta corrente, altere a sua palavra passe ou abandone a aplicação. Contém, assim, os seguintes botões:

- **Alterar Password**: este botão é o que permite alterar a palavra passe e chama o método `altPassword()`.
- **Conta Corrente**: este botão permite que o utilizador visualize a sua conta corrente com o pagador de serviços e chama o método `contaCorrente()`.
- **Logout**: este botão permite abandonar a aplicação e chama o método `logout()`.

Componente de Lógica de Negócio - beneficiario.aspx.cs

Este componente apoia o componente `beneficiario.aspx` e implementa as regras de negócio. São implementados os seguintes métodos:

- `Page_Load()`: executado sempre que o componente beneficiario.aspx é carregado. Começa por verificar na sessão se o utilizador se encontra devidamente autenticado como beneficiário.
- `altPassword()`: chamado sempre que é pressionado o botão **Alterar Password**. Redirecciona para o componente `alterarpwd.aspx`.
- `contaCorrente()`: chamado quando é pressionado o botão **Conta Corrente**. Redirecciona para o componente `contacorrente.aspx`.

confproc

Estes componentes mostram ao utilizador o resultado de uma pesquisa de beneficiários e permitem um seja seleccionado. Participam no caso de uso Identifica Beneficiário.

Componente Gráfico - confproc.aspx

Este componente apresenta os resultados de pesquisas a beneficiários. Os resultados são apresentados de uma forma tabular com o número que os identifica no sistema e o nome. Cada beneficiário encontrado é acompanhado por uma ligação que permite que se efectuem operações sobre ele (as operações podem ser o processamento de facturas (reembolsos), alteração de dados pessoais e visualização da conta corrente). Possui ainda o botão **Voltar** que permite regressar ao menu dos funcionários.

Componente de Lógica de Negócio - confproc.aspx.cs

Este componente apoia o componente `beneficiario.aspx` e implementa as regras de negócio. São implementados os seguintes métodos:

- `Page_Load()`: executado sempre que o componente `beneficiario.aspx` é carregado. Começa por verificar na sessão se o utilizador se encontra devidamente autenticado como beneficiário. Neste método é efectuada a pesquisa dos beneficiários. Vai buscar os dados à página anterior (por cópia da instância da classe procura definida no componente `procura.aspx.cs`) e faz a pergunta à base de dados. Coloca os resultados na tabela do componente `confproc.aspx`.
- `voltar()`: chamado quando é pressionado o botão **Voltar**, redirecciona para o componente `funcionario.aspx`.
- `Del()`: chamado quando é seleccionado um beneficiário. Obtém a identificação do beneficiário escolhido, colocando-a na sessão, e, consoante o pedido que estiver na sessão, efectua o seguinte redireccionamento:
  - "alterar", redirecciona para o componente `alterar.aspx`;
  - "factura", redirecciona para o componente `identfactura.aspx`;
  - "conta", redirecciona para o componente `contacorrente.aspx`;

contacorrente

Estes são os componentes que mostram a conta corrente de um beneficiário. Participam no caso de uso Consultar Conta Corrente de Beneficiário.

Componente Gráfico - contacorrente.aspx

Este componente mostra todos os movimentos efectuados entre o beneficiário e o pagador de serviços. Possui duas tabelas para apresentar os proveitos e os custos do beneficiário e o botão **Voltar** que faz regressar ao menu de funcionários (se o utilizador for um funcionário) ou beneficiários (se o utilizador for um beneficiário).

Componente de Lógica de Negócio - contacorrente.aspx.cs

Este é o componente que implementa a lógica de negócio e apoia o componente `contacorrente.aspx`.

Este componente apoia o componente `beneficiario.aspx` e implementa as regras de negócio. São implementados os seguintes métodos:

- `Page_Load()`: este método é chamado sempre que é carregado o componente `contacorrente.aspx`. Começa por verificar na sessão se o utilizador se encontra autenticado.
  Para apresentar os custos e para os proveitos são criadas estruturas do tipo DataTable, para onde são passados os resultados da pesquisa na base de dados. As DataTable são ligadas a outras estruturas presentes no componente contacorrente.aspx (DataGrid) que apresentam os resultados na forma tabular.
- `voltar()`: este método é chamado quando é pressionado o botão **Voltar**. Redirecciona para o componente `funcionario.aspx` se a sessão indicar que se trata de um utilizador funcionário ou para o componente `beneficiario.aspx` se a sessão indicar que se trata de um utilizador beneficiário.

contribuicoes

Estes componentes implementam o processamento das contribuições dos beneficiários. São os que efectuam as transferências bancárias das contas dos beneficiários para a conta do pagador de serviços. Participam no caso de uso Receber Contribuição.

Componente Gráfico - contribuicoes.aspx

Este componente apresenta um botão – **Iniciar** – que dá início ao processamento das contribuições e um botão – **Voltar** – que permite regressar ao menu dos funcionários.

Componente de Lógica de Negócio - contribuicoes.aspx.cs

Este componente processa os pagamentos das contribuições. Implementa os métodos:

- `Page_Load()`: este método é chamado sempre que é carregado o componente `contribuicoes.aspx`. Verifica se o utilizador se encontra autenticado e verifica se, no presente mês, já foram efectuadas as contribuições. Se já tiverem sido efectuadas, desliga o botão Iniciar e emite a devida mensagem.
- `voltar()`: este método é chamado quando se pressiona o botão **Voltar** e redirecciona para o componente `funcionario.aspx`.
- `iniciar()`: este é o método chamado quando foi pressionado o botão **Iniciar**. Começa por obter os NIBs dos beneficiários. Para cada beneficiário, chama *Web Service* do banco para transferência bancária e regista a transferência.

erro

Estes são os componentes que lidam com erros de interacção com o utilizador.

Componente Gráfico - erro.aspx

Este componente apresenta a mensagem indicando qual o erro que ocorreu e tem o botão **Voltar** que faz regressar ao início da aplicação (o utilizador terá que se autenticar de novo).

Componente de Lógica de Negócio - erro.aspx.cs

Este é o componente que corre por trás do componente erro.aspx. Implementa dos métodos:

- `Page_Load()`: este método é chamado quando é carregado o componente `erro.aspx`. Vai buscar à sessão o erro que ocorreu e indica o erro ao componente `erro.aspx`.
- `voltar()`: este método é chamado quando se pressiona o botão **Voltar**. Este método limpa a sessão (o utilizador deixa de estar autenticado) e remete para o início (componente `index.htm`)

funcionário

Estes componentes implementam o menu dos funcionários. O componente gráfico é constituído por botões que dão acesso às funcionalidades dos funcionários. O componente de regras de negócio (*code behind*) responde às acções do utilizador. São o ponto de partida para os casos de uso utilizados pelos funcionários (Inscrever Beneficiário, Alterar Dados de Beneficiário, Processar Reembolsos, Receber Contribuição, Consultar Conta Corrente do Beneficiário e Relatórios).

Componente Gráfico - funcionario.aspx

Componente gráfico para interacção com o utilizador. Possui os botões:

- **Inscrição**: chama o método `inscricao()`; permite a inscrição de novos beneficiários.
- **Alterar Dados**: chama o método `altDados()`; permite alterar dados pessoais de beneficiários.
- **Conta Corrente**: chama o método `ContCorrente()`; permite consultar a conta corrente de um beneficiário.
- **Processar Factura**: chama o método `processarFactura()`; permite processar uma factura de um beneficiário para proceder ao reembolso.
- **Relatórios**: chama o método `relatorio()`; permite visualizar o relatório de contas do pagador de serviços.
- **Efectuar Recebimentos**: chama o método `contribuicoes()`; permite efectuar os recebimentos (contribuições dos beneficiários).
- **Logout**: chama o método `logouBtnClick()`; permite abandonar a aplicação.

Componente de Lógica de Negócio - funcionário.aspx.cs

Componente *code behind* que apoia o componente `funcionario.aspx` e implementa as regras de negócio. Implementa os métodos:

- `Page_Load()`: executado sempre que o componente `funcionario.aspx` é carregado. Neste caso, verifica na sessão se o utilizador se encontra devidamente autenticado.
- `inscricao()`: chamado quando o utilizador pressiona o botão **Inscricao** do componente `funcionario.aspx`. Redirecciona para o componente `inscricao.aspx`.
- `logoutBtnClick()`: executado quando é pressionado o botão **Logout**. Limpa a sessão e redirecciona para o componente `login.aspx`.
- `altDados()`: é chamado quando é pressionado o botão **Alterar Dados**. escreve na sessão “alterar" para indicar ao componente `procura.aspx` que se pretende alterar dados a beneficiários. Redirecciona depois para o componente `procura.aspx`.
- `ContCorrente()`: é chamado quando é pressionado o botão **Conta Corrente**. Escreve na sessão "conta" para indicar ao componente `procura.aspx` o que se pretende. É para este último componente que este método redirecciona.
- `processarFactura()`: é chamado quando é pressionado o botão **Processar Factura**. Escreve na sessão "factura" para indicar ao

componente `procura.aspx` que se trata do processamento de facturas e remete para este último componente.

- `relatorio()`: chamado quando se pressiona o botão **Relatorios**. Redirecciona para o componente `relatorios.aspx`.
- `contribuicoes()`: chamado quando o utilizador pressiona o botão **Efectuar Recebimentos**. Redirecciona para o componente `contribuicoes.aspx`.

identfactura

Estes componentes são os que preenchem os dados identificadores das facturas a reembolsar. Permitem que o utilizador indique a entidade emissora da factura, o número da factura e a data. Participam no caso de uso Processar Reembolsos.

Componente Gráfico - identfactura.aspx

Este componente apresenta um formulário com os campos Entidade, Número e Data. É verificado o preenchimento do formulário com as seguintes funcionalidades ASP:

- `RequiredFieldValidator`: verifica se um campo se encontra preenchido. Todos os campos são verificados.
- `RangeValidator`: verifica se a data da factura é válida; actua sobre o campo Data.
- `RegularExpressionValidator`: verifica se o conteúdo de um campo respeita um formato. Actua sobre a data para forçá-la a ser no formato "dd/mm/aaaa".

Tem ainda os botões:

- **Submeter**: permite continuar o processamento da factura.
- **Cancelar**: cancela o processamento da factura e regressa ao menu de funcionários.

Componente de Lógica de Negócio - identfactura.aspx.cs

Este componente implementa a lógica de negócio para o componente `indentfactura.aspx`. Para tal, implementa os métodos:

`Page_Load()`: este método é chamado sempre que é carregado o componente `identfactura.aspx`. Valida a sessão, verificando se o utilizador se encontra autenticado como funcionário e obtém a data actual para poder validar a data da factura inserida pelo utilizador.

`SubmitBtnClick()`: chamado quando se pressiona o botão **Submeter**. Verifica a validade da factura (se já existe uma factura com a mesma identificação). Se for uma factura válida, coloca a sua identificação na sessão e redirecciona para o componente `procfactura.aspx`. Se a factura não for válida, emite a mensagem indicando que a factura já foi processada.

`cancelar()`: chamado quando é pressionado o botão Cancelar, redirecciona para o componente `funcionario.aspx`.

inscricao

Estes são os componentes que permitem a inscrição de beneficiários no sistema de pagamento de serviços. Participam no caso de uso Inscrição de Beneficiários.

Componente Gráfico - inscricao.aspx

Este é o componente que gera o formulário onde deverão constar os dados pessoais do beneficiário. Contém os campos:

- **Nome**;
- **Morada**;
- **Código Postal**: este campo é, na verdade, dois campos (um para cada componente do código postal);
- **Telefone**;
- **Data de Nascimento**;
- **Sexo**;
- **Estado Civil**;
- **NIB**;

A validade do conteúdo dos campos do formulário tem que ser verificado. Para tal, são usadas as funcionalidades ASP:

- `requiredfieldvalidator`: verifica se um campo se encontra preenchido. Visto que todos os campos são de preenchimento obrigatório, esta funcionalidade actua sobre todos os campos.
- `RegularExpressionValidator`: esta funcionalidade permite verificar se o conteúdo de um campo respeita determinado formato. Actua sobre os campos:
  - Código Postal: actua sobre os dois campos que compõem este campo. Verifica se o primeiro campo tem 4 números e se o segundo tem 3 números.
  - Telefone: verifica se este campo se encontra preenchido por 9 números;
  - Data: verifica se a data inserida se encontra no formato "dd/mm/aaaa".
- `Rangevalidator`: verifica se o conteúdo de um campo se encontra dentro de um determinado intervalo. Actua sobre o campo Data para verificar se esta é válida.

Possui os botões:

- **Submeter**: permite a inscrição do beneficiário. Chama o método `submeter()`.
- **Cancelar**: cancela a inscrição, regressando ao menu dos funcionários. Chama o método `cancelar()`.
- **Limpar**: limpa os campos do formulário.

Componente de Lógica de Negócio - inscricao.aspx.cs

Componente *code behind* que apoia o componente `inscricao.aspx` e implementa as regras de negócio. Implementa os métodos:

`Page_Load()`: executado sempre que o componente `inscricao.aspx` é carregado. Verifica na sessão se o utilizador se encontra devidamente autenticado

como funcionário e obtém a data actual para poder validar a data de nascimento introduzida pelo utilizador.

`submeter()`: este método é chamado quando é pressionado o botão **Submeter**. Converte os dados inseridos no formato aceite pela base de dados e insere-os.

`cancelar()`: chamado quando é pressionado o botão **Cancelar**, redirecciona para o componente `funcionario.aspx`.

`limpar()`: chamado quando o utilizador pressiona o botão **Limpar**, limpa o formulário.

login

Estes componentes permitem a autenticação dos utilizadores.

Componente Gráfico - login.aspx

Este componente apresenta ao utilizador dois campos onde deverão ser preenchidos o nome do utilizador e a sua palavra passe e o tipo de utilizador (beneficiário ou funcionário).

Componente de Lógica de Negócio - login.aspx.cs

Componente *code behind* que apoia o componente `login.aspx` e implementa as regras de negócio. Implementa os métodos:

`Page_Load()`: este método é chamado sempre que é carregado o componente `login.aspx`. Neste caso, este método não faz nada, visto que não há inicializações a fazer.

`SubmitBtnClick()`: este método é chamado sempre que é pressionado o botão **Submeter**. Verifica se a palavra-passe introduzida é a do utilizador identificado. Se se tratar de um beneficiário devidamente autenticado, escreve na sessão o tipo do utilizador e redirecciona para o componente `beneficiario.aspx`. Se se tratar de um funcionário, escreve na sessão o tipo do utilizador e redirecciona para o componente `funcionario.aspx`.

procfactura

Estes componentes permitem o processamento de uma factura para reembolso. Participam no caso de uso Processar Reembolsos.

Componente Gráfico - procfactura.aspx

Este componente apresenta o cabeçalho da factura (com a sua identificação), o conteúdo introduzido e um formulário que permite adicionar linhas. Quando é introduzida uma nova linha (através do preenchimento do formulário e pressão do botão **Adicionar**), a página é recarregada com o conteúdo da factura actualizado.

Componente de Lógica de Negócio - procfactura.aspx.cs

Componente code behind que apoia o componente procfactura.aspx e implementa as regras de negócio. Implementa os métodos:

- `Page_Load()`: este método é chamado sempre que é carregado o componente `procfactura.aspx`. Começa por criar uma tabela que conterá as linhas que o utilizador vai inserindo e coloca-a na sessão. Esta tabela fica ligada à tabela presente no componente `procfactura.aspx`. Coloca também a variável que conterá o desconto total na sessão. Vai ainda buscar os serviços presentes na base de dados. O botão **Confirmar** começa desligado, pois no início, a factura está vazia.
- `adicionar()`: este método é chamado sempre que é pressionado o botão **Adicionar**. Para a linha inserida, determina o reembolso e insere-a na tabela que se encontra na sessão. Se se tratar da primeira linha inserida, o botão **Confirmar** é ligado.
- `cancelar()`: chamado quando é pressionado o botão **Cancelar**, limpa a sessão e redirecciona para o componente `funcionario.aspx`.
- `confirm()`: este método é chamado quando se pressiona o botão **Confirmar**. Este é o método que insere a factura e as linhas na base de dados. Começa por efectuar a transferência bancária chamando o *Web Service* disponibilizado pelo banco. Depois de efectuado o pagamento do reembolso, insere a factura. Depois de inserida a factura, insere as linhas constantes na factura.
- `Del()`: este método é chamado quando é escolhida uma linha para ser eliminada. Quando retira a linha da tabela que está na sessão, recalcula o reembolso.

procura

Estes componentes implementam a funcionalidade que permite pesquisar beneficiários. As pesquisas são efectuadas para processar facturas (efectuar reembolsos), consultar a conta corrente dos beneficiários e para alterar dados pessoais de beneficiários. Participam no caso de uso Identifica Beneficiário.

Componente Gráfico - procura.aspx

Este componente mostra ao utilizador um formulário de pesquisa de beneficiários. Apresenta um campo a ser preenchido com o identificador do beneficiário e um campo a ser preenchido com o nome do beneficiário. Apresenta ainda o botão **Submeter** que chama o método `SubmitBtnClick()`. Este botão permite efectuar a pesquisa.

Componente de Lógica de Negócio - procura.aspx.cs

Componente *code behind* que apoia o componente `procura.aspx` e implementa as regras de negócio. Implementa os métodos:

- `Page_Load()`: executado sempre que o componente `funcionario.aspx` é carregado. Efectua a verificação da sessão (se o utilizador se encontra devidamente autenticado).
- `SubmitBtnClick()`: este método é chamado quando é pressionado o botão **Submeter** do componente `procura.aspx`. Coloca na sessão os dados para a pesquisa.

relatorios

Estes componentes permitem a visualização dos relatórios de contas do pagador de serviços para o mês corrente.

Componente Gráfico - relatorios.aspx

Este componente mostra os proveitos numa tabela e os custos noutra tabela. Para cada linha de custo ou proveito mostra a descrição.

Apresenta o botão Voltar que faz regressar ao menu dos funcionários.

Componente de Lógica de Negócio - relatorios.aspx.cs

Componente *code behind* que apoia o componente `relatorios.aspx` e implementa as regras de negócio. Implementa os métodos:

- `Page_Load()`: chamado sempre que é carregado o componente `relatorios.aspx`. Primeiro, consulta a sessão para confirmar se o utilizador já se encontra autenticado. Processa a data actual de modo a obter os custos e proveitos do mês corrente.
  Os custos do pagador de serviços são mostrados por serviço. Para isso, este método obtém da base de dados:
  - `data` da tabela `Facturas`: usa esta coluna para escolher os reembolsos do mês corrente;
  - `valor` da tabela `Linhas`: usa este valor para calcular o reembolso e determinar os custos;
  - `designação` da tabela `Servicos`: é utilizada esta coluna para a descrição dos custos.
  - `percentagem`, `plaffond` e `franquia` da tabela `Serviços`: estes campos são usados para calcular os custos.

  A relação obtida da base de dados vem ordenada pelas colunas `designação`, `data` e `valor`.
  Os dados obtidos da base de dados são formatados para serem apresentados e inseridos numa tabela ligada à tabela de custos do componente `relatorios.aspx`.
  Os proveitos do pagador de serviços provêm das contribuições dos beneficiários. Para obter as contribuições, este método obtém da base de dados a soma das contribuições efectuadas no mês corrente. O total das contribuições obtido é colocado numa tabela ligada à tabela de proveitos do componente gráfico.
- `voltar()`: este método é chamado quando é pressionado o botão **Voltar** e redirecciona para o componente funcionario.aspx.

Despesa

Este componente implementa o *Web Service* que a aplicação disponibiliza ao gestor de acordos. Neste componente define-se a classe `Despesa` que contém o método

```
public bool lancarDespesaMedianteAcordo(Entidade benef,
                                        Entidade prestador,
                                        long numFactura,
                                        ArtigoSaude artigo,
                                        string data,
                                        decimal valor).
```

Este método constitui o *Web Service* disponibilizado. Aceita como argumentos

- o beneficiário (`benef` da classe `Entidade`): a identificação do beneficiário;
- o prestador (`prestador` da classe `Entidade`): a identificação da entidade que prestou o serviço ao beneficiário e emitiu a factura;
- o número da factura (`numFactura` do tipo `Long`): o número da factura emitida;
- o serviço de saúde da linha (`artigo` da classe `ArtigoSaude`): o serviço de saúde a que a linha recebida se refere;
- a data da factura (`data` do tipo `string`);
- o valor da linha (`valor` do tipo `decimal`): a quantia paga pelo pagador de serviços à entidade prestadora.

Este *Web Service* implementa o caso de uso Lançamento de Despesa. Recebe da aplicação que o chama, as linhas da factura e insere-as no sistema.

A recepção das linhas da factura é feita uma de cada vez. Quando é recebida uma linha, é verificado se a respectiva factura já existe. Se é a primeira linha recebida, é criada a respectiva factura e, só depois, inserida a linha. Se já não é a primeira linha, é identificada a respectiva factura a que pertencerá a linha.

Notifica

Este componente implementa o caso de uso Notificação de Despesa. Este caso de uso foi uma imposição dos docentes. O *Web Service* está implementado através do método

```
public int notificarTransferencia(string NIBContaOrigem,
                                   string NIBContaDestino,
                                   decimal valor,
                                   string data,
                                   string descricao).
```

Este método faz parte da classe `Notificacao` e aceita como argumentos:

- o NIB da conta de origem (NIBContaOrigem do tipo string);
- o NIB da conta de destino (NIBContaDestino do tipo string);
- o valor a transferir (valor do tipo decimal);
- a data da transferência (data do tipo string);
- a descrição da transferência (descricao do tipo string).

O que este método faz é registar as transferências na base de dados. Quando o NIBContaDestino é o NIB do pagador de serviços, é identificado o beneficiário a quem corresponde o NIBContaOrigem e regista a transferência na tabela Notificação como um depósito. Quando o NIBContaDestino não é a do pagador de serviços, é identificado o beneficiário a quem pertence o NIB e é registada a transferência na tabela Notificação como um levantamento.

Base de Dados

A seguir, apresenta-se o diagrama de classes da base de dados. Para mais detalhes, consultar os DDLs em anexo.

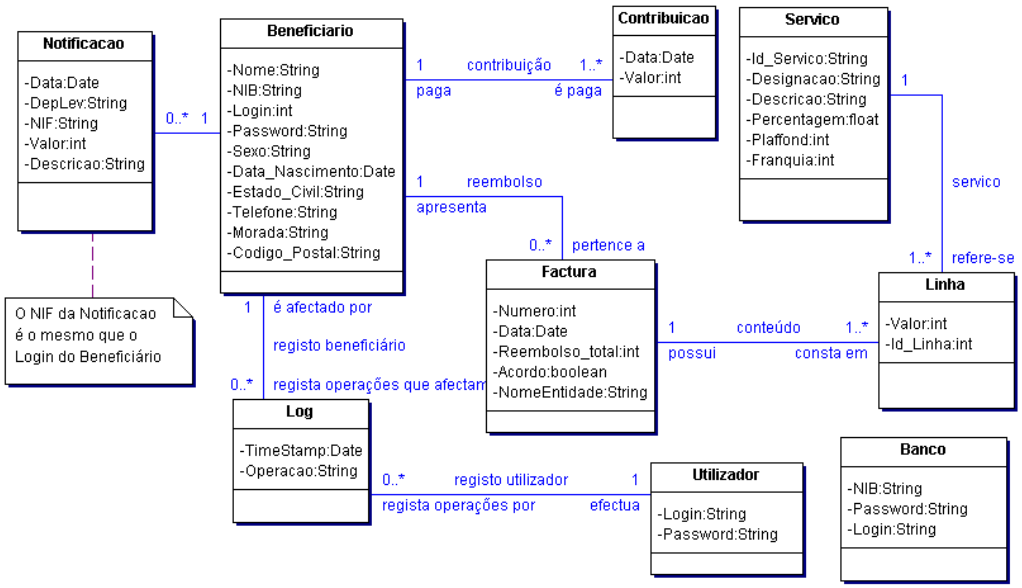

Figura 1. Diagrama de classes da base de dados

Banco

Esta classe é mapeada para a base de dados na relação Bancos. Mantém dados importantes para a interacção com os bancos.

Beneficiario

Esta classe corresponde à relação Beneficiarios e mantém os dados importantes para a identificação dos beneficiários. A chave da relação Beneficiários é a coluna Login que corresponde ao NIF (número de identificação fiscal)

Contribuicao

A relação Contribuicoes corresponde à classe Contribuicao e mantém um registo das contribuições pagas pelos beneficiários.

Factura

Esta classe corresponde à tabela Facturas e guarda as facturas reembolsadas.

Linha

Esta classe é mapeada para a base de dados na relação Linhas. Mantém as linhas (conteúdo) das facturas reembolsadas.

Log

Esta classe corresponde à tabela Logs e mantém o registo da actividade do sistema.

Notificacao

Esta classe corresponde à tabela Notificacoes e regista as notificações de transferências bancárias.

Servico

A tabela Servicos corresponde à classe Servico e guarda as informações relevantes para os serviços descontados pelo pagador de serviços. Mantém as informações relativas ao plaffond (máximo reembolsado por um serviço), à franquia (mínimo custo a partir do qual o beneficiário tem direito a desconto) e à percentagem (do custo a que o beneficiário tem direito de ser reembolsado).

Utilizador

Esta classe é mapeada para a base de dados na relação Utilizadores e mantém a informação sobre os funcionários do pagador de serviços que podem utilizar o sistema.

Realização de Casos de Uso

Foram implementados todos os casos de uso presentes na análise de requisitos.

Alterar Dados de Beneficiário

A alteração de dados de beneficiários é feita por funcionários do pagador de serviços, pelo que devem estar autenticados. Este caso de uso inclui o caso de uso Identificar Beneficiário. Quando se encontra identificado o beneficiário, o componente `alterar.aspx` mostra os dados do beneficiário e permite alterá-los.

Inscrever Beneficiário

A inscrição de novos beneficiários tem que ser feita por funcionários. Este caso de uso tem como ponto de partida o menu dos funcionários e participam os componentes:

- `inscricao` (com o componente gráfico e de lógica de negócio): disponibiliza o formulário onde devem ser introduzidos os dados do novo beneficiário. Quando é submetido o novo beneficiário, é carregado o componente beneconf (com o componente gráfico e de lógica de negócio).
- `beneconf` (com o componente gráfico e de lógica de negócio): mostra os dados do novo beneficiário para confirmação.

Processar Reembolsos

O processamento de reembolsos é usado pelo funcionário. Depois de encontrado o beneficiário a ser reembolsado, o utilizador deverá identificar a factura, preenchê-la e submetê-la. O sistema encarrega-se de efectuar a transferência do reembolso. Participam neste caso de uso os componentes:

- `identfactura` (com o componente gráfico e de lógica de negócio): recebe os dados identificadores da factura a ser processada. Depois de identificada a factura, remete para o componente `procfactura`. Envia os dados identificadores da factura aos componentes que lhe seguem na sequência através da sessão.
- `procfactura` (com o componente gráfico e de lógica de negócio): permitem o preenchimento da factura. Quando a factura é submetida, é calculado e efectuado o reembolso (através da invocação do *Web Services* do Banco) e registadas a factura e a acção.

Apresenta-se a seguir o diagrama de sequência para este caso de uso:

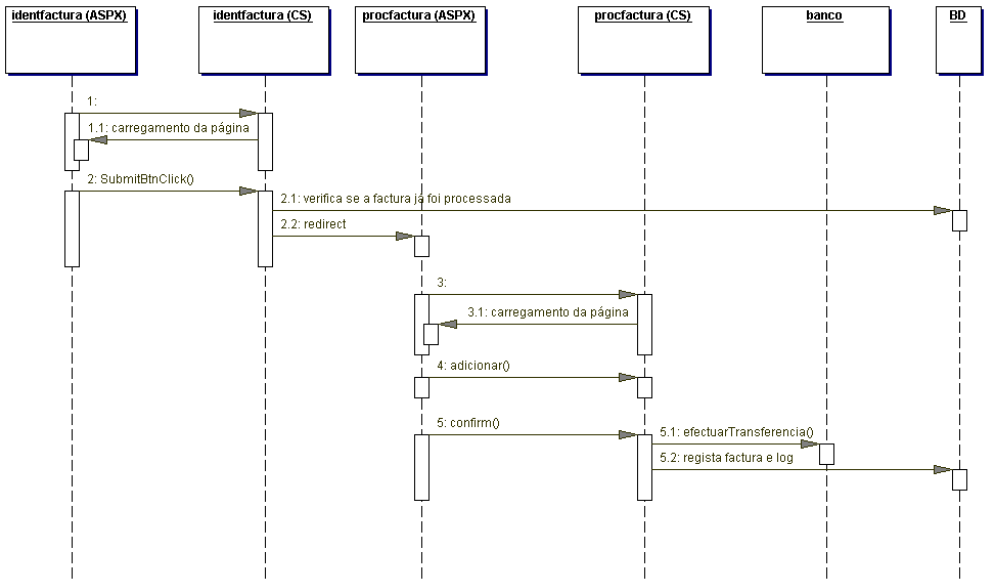

Figura 2. Diagrama de sequência para o caso de uso Processar Reembolsos

Identificar Beneficiário

Este caso de uso existe em outros casos de uso que necessitam que seja conhecido um beneficiário (Alterar Dados de Beneficiário, Processar Reembolsos e Consultar Conta Corrente). Dois componentes participam neste caso de uso:

- `procura` (com o componente gráfico e de lógica de negócio): fornece um formulário que deve ser preenchido com os dados para a pesquisa. Este componente envia, via sessão, os dados da pesquisa para o componente `confproc`.
- `confproc` (com o componente gráfico e de lógica de negócio): este componente recebe do anterior os dados da pesquisa e efectua-a na base de dados. Mostra os resultados da pesquisa, possibilitando a escolha de uma das ocorrências.

Receber Contribuição

O recebimento de contribuições só pode ser utilizado pelos funcionários. Neste caso de uso participa o componente `contribuicoes` (com o componente gráfico e de lógica de negócio). Este componente verifica se as contribuições do mês corrente ainda não foram efectuadas. Se puderem ser feitas, efectuam as necessárias transferências, utilizando o *Web Service* disponibilizado pelo banco.

No diagrama de sequência que se apresenta, as mensagens 2.2 e 2.3 são efectuadas para cada beneficiário.

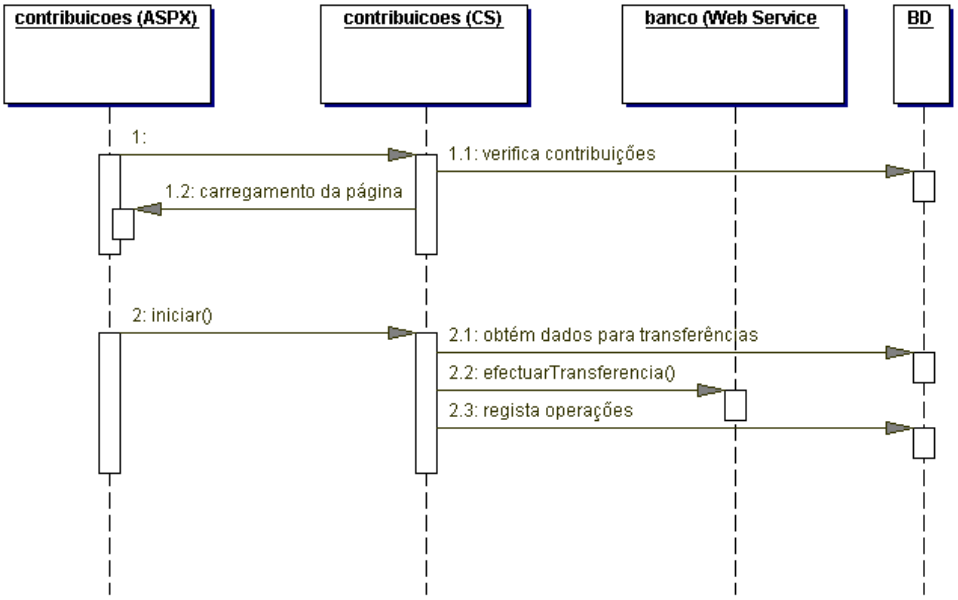

Figura 3. Diagrama de sequência para o caso de uso Processar Reembolsos

Notificação de Transferência

Este caso de uso é a implementação do *Web Service* disponibilizado aos bancos para notificações de transferências. Quando é chamado, regista a notificação recebida na base de dados.

Consultar Conta Corrente do Beneficiário

Este caso de uso pode ser utilizado por beneficiários e funcionários.

Quando o utilizador é um beneficiário, dispensa-se a identificação de beneficiários. Se o utilizador for um funcionário, é necessário proceder à identificação do beneficiário cuja conta corrente será consultada. Participa neste caso de uso o componente `contacorrente` (com o componente gráfico e de lógica de negócio) que mostra os custos e perdas para o beneficiário.

Relatórios

Este caso de uso só pode ser utilizado por funcionários. Permite a consulta de resultados contabilísticos do pagador de serviços.

Notificação de Despesa

A notificação de despesas é uma funcionalidade que deverá ser utilizada pelo gestor de acordos para notificar o pagador de serviços que um beneficiário beneficiou de um pagamento de serviços. Participa neste caso de uso o componente `Notificacao` que implementa o *Web Service* necessário.

Alterar Palavra Passe

Este caso de uso é utilizado pelo beneficiário para alterar a sua palavra passe que lhe dá acesso ao sistema. Participa neste caso de uso o componente `altpwd` (com o componente gráfico e de lógica de negócio).

Testes

Testes aos Componentes

Os componentes foram sendo desenvolvidos dentro da implementação dos casos de uso. Os casos de uso em desenvolvimento foram sendo testados. A validação dos componentes dá-se quando são validados os casos de uso.

Os componentes que implementam os *Web Services* disponibilizados foram testados e validados recorrendo a aplicações-cliente simuladas.

Testes de Integração

Os testes reais de integração da aplicação serão realizados na semana de 27 de Maio e seguintes.

Foram testados os *Web Services* `consultarSaldo`, `efectuarTransferencia` e `notificarTransferencia` com o Banco, que deram resultados positivos.

Não foi testado o *Web Service* `lancarDespesaMedianteAcordo`, devido à indisponibilidade do grupo de Gestão de Acordos.

Conclusões

O trabalho foi completado de uma forma bastante satisfatória, tomando em conta que se tratava de uma tecnologia nova (a plataforma .NET), e de um projecto global muito ambicioso de integração de várias aplicações.

Houve, devido à novidade do projecto, bastante atraso especialmente a nível da especificação de requisitos, o que fez com que a maior parte do projecto fosse feita na sua parte final, só a partir do momento da sua implementação concreta. Isto levou a algumas alterações significativas a nível, por exemplo, da base de dados, já bastante tarde no projecto.

No entanto o grupo de trabalho respondeu bastante bem e o projecto está completo.

Anexos

DDLs

```
CREATE TABLE bancos
 (
  login                     VARCHAR2(20) NOT NULL,
  password                  VARCHAR2(20) NOT NULL,
  nib                       VARCHAR2(24) NOT NULL
 )
 PCTFREE    10
 PCTUSED    40
 INITRANS   1
 MAXTRANS   255
/


ALTER TABLE bancos
 ADD CHECK ("LOGIN" IS NOT NULL)
/

ALTER TABLE bancos
 ADD CHECK ("PASSWORD" IS NOT NULL)
/

ALTER TABLE bancos
 ADD CHECK ("NIB" IS NOT NULL)
/




CREATE TABLE beneficiarios
 (
  nome                      VARCHAR2(240) NOT NULL,
  nib                       VARCHAR2(24) NOT NULL,
  login                     NUMBER NOT NULL,
  password                  VARCHAR2(20) NOT NULL,
  sexo                      VARCHAR2(1) NOT NULL,
  data_nascimento           DATE NOT NULL,
  estado_civil              VARCHAR2(1) NOT NULL,
  telefone                  NUMBER(9) NOT NULL,
  morada                    VARCHAR2(240) NOT NULL,
  codigo_postal             VARCHAR2(8) NOT NULL
 )
 PCTFREE    10
 PCTUSED    40
 INITRANS   1
 MAXTRANS   255
/

ALTER TABLE beneficiarios
 ADD CONSTRAINT beneficiar_pk PRIMARY KEY (login)
/

ALTER TABLE beneficiarios
 ADD CHECK ("NOME" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("NIB" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("LOGIN" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("PASSWORD" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("SEXO" IS NOT NULL)
```

```
/

ALTER TABLE beneficiarios
 ADD CHECK ("DATA_NASCIMENTO" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("ESTADO_CIVIL" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("TELEFONE" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("MORADA" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("CODIGO_POSTAL" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("NOME" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("NIB" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("LOGIN" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("PASSWORD" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("SEXO" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("DATA_NASCIMENTO" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("ESTADO_CIVIL" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("TELEFONE" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("MORADA" IS NOT NULL)
/

ALTER TABLE beneficiarios
 ADD CHECK ("CODIGO_POSTAL" IS NOT NULL)
/




CREATE TABLE contribuicoes
 (
  data                       DATE NOT NULL,
  valor                      NUMBER NOT NULL,
  beneficiar_login           NUMBER NOT NULL
 )
 PCTFREE    10
 PCTUSED    40
 INITRANS   1
 MAXTRANS   255
/
```

```
ALTER TABLE contribuicoes
 ADD CONSTRAINT contrib_pk PRIMARY KEY (data,beneficiar_login)
/

ALTER TABLE contribuicoes
 ADD CONSTRAINT contrib_beneficiar_fk FOREIGN KEY (beneficiar_login)
      REFERENCES BENEFICIARIOS(login)
/

ALTER TABLE contribuicoes
 ADD CHECK ("DATA" IS NOT NULL)
/

ALTER TABLE contribuicoes
 ADD CHECK ("VALOR" IS NOT NULL)
/

ALTER TABLE contribuicoes
 ADD CHECK ("BENEFICIAR_LOGIN" IS NOT NULL)
/

ALTER TABLE contribuicoes
 ADD CHECK ("DATA" IS NOT NULL)
/

ALTER TABLE contribuicoes
 ADD CHECK ("VALOR" IS NOT NULL)
/

ALTER TABLE contribuicoes
 ADD CHECK ("BENEFICIAR_LOGIN" IS NOT NULL)
/




CREATE TABLE facturas
 (
  nomeentidade               VARCHAR2(255) NOT NULL,
  numero                     NUMBER NOT NULL,
  acordo                     VARCHAR2(3),
  data                       DATE NOT NULL,
  reembolso_total            NUMBER NOT NULL,
  beneficiar_login           NUMBER NOT NULL
 )
 PCTFREE    10
 PCTUSED    40
 INITRANS   1
 MAXTRANS   255
/


ALTER TABLE facturas
 ADD CONSTRAINT factura_pk PRIMARY KEY (nomeentidade,numero)
/

ALTER TABLE facturas
 ADD CONSTRAINT factura_beneficiar_fk FOREIGN KEY (beneficiar_login)
      REFERENCES BENEFICIARIOS(login)
/

ALTER TABLE facturas
 ADD CHECK ("NUMERO" IS NOT NULL)
/

ALTER TABLE facturas
 ADD CHECK ("DATA" IS NOT NULL)
/

ALTER TABLE facturas
 ADD CHECK ("REEMBOLSO_TOTAL" IS NOT NULL)
/

ALTER TABLE facturas
```

```
 ADD CHECK ("BENEFICIAR_LOGIN" IS NOT NULL)
/

ALTER TABLE facturas
 ADD CHECK ("NUMERO" IS NOT NULL)
/

ALTER TABLE facturas
 ADD CHECK ("DATA" IS NOT NULL)
/

ALTER TABLE facturas
 ADD CHECK ("REEMBOLSO_TOTAL" IS NOT NULL)
/

ALTER TABLE facturas
 ADD CHECK ("BENEFICIAR_LOGIN" IS NOT NULL)
/




CREATE TABLE linhas
 (
  id_linha                   NUMBER NOT NULL,
  valor                      NUMBER NOT NULL,
  factura_nomeentidade       VARCHAR2(255) NOT NULL,
  factura_numero             NUMBER NOT NULL,
  servico_id_servico         VARCHAR2(4)
 )
 PCTFREE    10
 PCTUSED    40
 INITRANS   1
 MAXTRANS   255
/


ALTER TABLE linhas
 ADD CONSTRAINT linha_pk PRIMARY KEY
(id_linha,factura_nomeentidade,factura_numero)
/

ALTER TABLE linhas
 ADD CONSTRAINT linha_factura_fk FOREIGN KEY
(factura_nomeentidade,factura_numero)
      REFERENCES FACTURAS(nomeentidade,numero)
/

ALTER TABLE linhas
 ADD CONSTRAINT linha_servico_fk FOREIGN KEY (servico_id_servico)
      REFERENCES SERVICOS(id_servico)
/

ALTER TABLE linhas
 ADD CHECK ("ID_LINHA" IS NOT NULL)
/

ALTER TABLE linhas
 ADD CHECK ("VALOR" IS NOT NULL)
/

ALTER TABLE linhas
 ADD CHECK ("FACTURA_NOMEENTIDADE" IS NOT NULL)
/

ALTER TABLE linhas
 ADD CHECK ("FACTURA_NUMERO" IS NOT NULL)
/

ALTER TABLE linhas
 ADD CHECK ("ID_LINHA" IS NOT NULL)
/

ALTER TABLE linhas
 ADD CHECK ("VALOR" IS NOT NULL)
/
```

```
ALTER TABLE linhas
 ADD CHECK ("FACTURA_NOMEENTIDADE" IS NOT NULL)
/

ALTER TABLE linhas
 ADD CHECK ("FACTURA_NUMERO" IS NOT NULL)
/




CREATE TABLE logs
 (
  beneficiarios_login        NUMBER NOT NULL,
  utilizadores_login         VARCHAR2(20) NOT NULL,
  operacao                   VARCHAR2(20) NOT NULL,
  data                       DATE NOT NULL
 )
 PCTFREE    10
 PCTUSED    40
 INITRANS   1
 MAXTRANS   255
/


ALTER TABLE logs
 ADD CONSTRAINT bancos_pk PRIMARY KEY (data, beneficiarios_login)
/

ALTER TABLE logs
 ADD CONSTRAINT logs_beneficiarios_fk FOREIGN KEY (beneficiarios_login)
      REFERENCES BENEFICIARIOS(login)
/

ALTER TABLE logs
 ADD CONSTRAINT logs_utilizadores_fk FOREIGN KEY (utilizadores_login)
      REFERENCES UTILIZADORES(login)
/

ALTER TABLE logs
 ADD CHECK ("BENEFICIARIOS_LOGIN" IS NOT NULL)
/

ALTER TABLE logs
 ADD CHECK ("UTILIZADORES_LOGIN" IS NOT NULL)
/

ALTER TABLE logs
 ADD CHECK ("OPERACAO" IS NOT NULL)
/

ALTER TABLE logs
 ADD CHECK ("DATA" IS NOT NULL)
/



CREATE TABLE notificacoes
 (
  data_transf                DATE NOT NULL,
  deplev                     VARCHAR2(1) NOT NULL,
  nif                        NUMBER NOT NULL,
  valor                      NUMBER NOT NULL,
  descricao                  VARCHAR2(240)
 )
/

ALTER TABLE notificacoes
 ADD CONSTRAINT notifica_pk PRIMARY KEY (data_transf)
/

ALTER TABLE notificacoes
 ADD CONSTRAINT notifica_benef_fk FOREIGN KEY (nif)
      REFERENCES BENEFICIARIOS(login)
/
```

```
CREATE TABLE servicos
 (
  designacao                VARCHAR2(120) NOT NULL,
  descricao                 VARCHAR2(240) NOT NULL,
  percentagem               NUMBER NOT NULL,
  plaffond                  NUMBER NOT NULL,
  franquia                  NUMBER NOT NULL,
  id_servico                VARCHAR2(4) NOT NULL
 )
 PCTFREE    10
 PCTUSED    40
 INITRANS   1
 MAXTRANS   255
/


ALTER TABLE servicos
 ADD CONSTRAINT servico_pk PRIMARY KEY (id_servico)
/

ALTER TABLE servicos
 ADD CHECK ("DESIGNACAO" IS NOT NULL)
/

ALTER TABLE servicos
 ADD CHECK ("DESCRICAO" IS NOT NULL)
/

ALTER TABLE servicos
 ADD CHECK ("PERCENTAGEM" IS NOT NULL)
/

ALTER TABLE servicos
 ADD CHECK ("PLAFFOND" IS NOT NULL)
/

ALTER TABLE servicos
 ADD CHECK ("FRANQUIA" IS NOT NULL)
/

ALTER TABLE servicos
 ADD CHECK ("DESIGNACAO" IS NOT NULL)
/

ALTER TABLE servicos
 ADD CHECK ("DESCRICAO" IS NOT NULL)
/

ALTER TABLE servicos
 ADD CHECK ("PERCENTAGEM" IS NOT NULL)
/

ALTER TABLE servicos
 ADD CHECK ("PLAFFOND" IS NOT NULL)
/

ALTER TABLE servicos
 ADD CHECK ("FRANQUIA" IS NOT NULL)
/




CREATE TABLE utilizadores
 (
  login                     VARCHAR2(20) NOT NULL,
  password                  VARCHAR2(20) NOT NULL
 )
 PCTFREE    10
 PCTUSED    40
 INITRANS   1
 MAXTRANS   255
```

```
/


ALTER TABLE utilizadores
 ADD CONSTRAINT users_pk PRIMARY KEY (login)
/

ALTER TABLE utilizadores
 ADD CHECK ("LOGIN" IS NOT NULL)
/

ALTER TABLE utilizadores
 ADD CHECK ("PASSWORD" IS NOT NULL)
/

ALTER TABLE utilizadores
 ADD CHECK ("LOGIN" IS NOT NULL)
/

ALTER TABLE utilizadores
 ADD CHECK ("PASSWORD" IS NOT NULL)
/
```