



Trabalho final de curso

Do Curso de

Licenciatura em Informática de Gestão (LIG)

Ano Lectivo 2009 / 2010

N.º da Proposta: 27

Título: reengenharia da plataforma Kusco

Professor Orientador:

Prof. Sérgio Guerreiro

Entidade Externa:

COFAC DSI

Alunos:

Manuel Monteiro nº20076165

João Franco nº20075392,

## Sumário

ÍNDICE DE FIGURAS.....	4
ÍNDICE DE FLUXOGRAMAS .....	4
ÍNDICE DE DIAGRAMAS .....	4
AGRADECIMENTOS .....	5
INTRODUÇÃO .....	6
ABSTRACT.....	7
<b>1.....</b>	<b>INTRODUÇÃO AO KUSCO</b>
.....	<b>ERRO! MARCADOR NÃO DEFINIDO.</b>
1.1. Como Aceder a plataforma Kusco.....	<b>Erro! Marcador não definido.</b>
1.2. Instruções de Utilização.....	10
1.3. Consultas .....	11
1.3.1. Horário .....	11
1.3.2. Notas.....	11
1.3.3. Saldo.....	12
1.3.4. Referência Multibanco .....	12
1.3.5. Dados Administrativo .....	12
1.3.6. Cancelamento .....	12
1.4. Alertas.....	13
1.4.1. Alertas de Alteração de Nota .....	<b>Erro! Marcador não definido.</b>
1.4.2. Alerta Pontual: Mensagem para Alunos do Docente .....	13
<b>2.....</b>	<b>ESTADO DA ARTE</b>
.....	<b>14</b>
2.1. Estado da Arte .....	14
2.2. Tecnologias de Suporte .....	14
2.2.1. Java Message Service (JMS).....	<b>Erro! Marcador não definido.</b>
2.2.2. Message-driven Beans.....	16
2.2.3. Plugins .....	17
2.2.4. Colector .....	17
2.2.5. SMSPro Optimus / Outsystems Hub Edition TM .....	18

2.2.5.1. SMSPro Optimus WebService Interface.....	18
2.2.5.2. Outsystems.....	18

### **3..... VISÃO GERAL DO PROJECTO**

..... ERRO! MARCADOR NÃO DEFINIDO.

3.1. Resolução de Problemas .....	23
3.2. Algoritmo.....	<b>Erro! Marcador não definido.</b> 4
3.3. Algoritmo Consulta .....	25
3.3.1. Algoritmo main .....	25
3.3.2. Algoritmo horario.....	27
3.3.1. Algoritmo notas.....	28
3.3.1. Algoritmo saldo.....	30
3.3.1. Algoritmo Referência Multibanco .....	31
3.3.1. Algoritmo Dados Administrativos .....	32
3.4. Base de Dados .....	32
3.4.1. Tabela Aluno .....	34
3.4.2. Tabela Horário.....	34
3.4.3. Tabela Notas.....	35
3.4.4. Tabela Saldo .....	35
3.4.5. Tabela Referência Multibanco .....	36
3.4.6. Tabela Dados Administrativos .....	36
<b>CONCLUSÃO.....</b>	<b>37</b>
<b>BIBLIOGRAFIA .....</b>	<b>38</b>
<b>ANEXO 1 – SCRIPT JAVA.....</b>	<b>39</b>
<b>ANEXO 2 – SCRIPT SQL .....</b>	<b>50</b>

## Índice de figuras

Figura 1.1- Opção “Registrar Canal” .....	8
Figura 1.2- Lista Registo Canais .....	8
Figura 1.3- Janela – Subscrição Canal SMS .....	9
Figura 1.4- sms com código segurança .....	9
Figura 1.5- Validação Registo Canal .....	10
Figura 1.6- Subscrever Alerta .....	10
Figura 2.1- Modelo point-to-point .....	15
Figura 2.2- Modelo publish/subscribe .....	16
Figura 2.3- Aplicação de envio de um enter .....	17
Figura 2.4- Arquitectura do OutSystems Hub Edition .....	19
Figura 3.1- Aplicação Consultas .....	22
Figura 3.2- JDBC .....	23

## Índice de Fluxogramas

Fluxograma 3.1- Algoritmo main.....	26
Fluxograma 3.2- Algoritmo horário .....	27
Fluxograma 3.3- Algoritmo Notas .....	28
Fluxograma 3.4- Algoritmo Saldo .....	30
Fluxograma 3.5- Algoritmo Referência multibanco .....	31
Fluxograma 3.1- Algoritmo Dados Administrativos.....	32

## Índice de Diagramas

Diagrama 3.1- Diagrama da Base de Dados .....	33
Diagrama 3.2- Tabela Aluno .....	34
Diagrama 3.3- Tabela Horário .....	34
Diagrama 3.4- Tabela Notas.....	35
Diagrama 3.5- Tabela Saldo.....	35
Diagrama 3.6- Tabela ref Multibanco .....	36
Diagrama 3.6- Tabela Dados Adminstrativos .....	36

## **Agradecimentos**

Primeiro gostaríamos de dar uma palavra especial de agradecimento ao nosso orientador,

Professor Sérgio Guerreiro, pelo apoio, disponibilidade e compreensão demonstrados ao longo do período de execução deste trabalho.

Gostaríamos ainda de agradecer a ajuda do Eng. Miguel já envolvido no desenvolvimento deste projecto.

Gostaríamos também de agradecer a todos os nossos amigos que de certa forma nos deram o seu apoio e alguma sugestão para o desenvolvimento deste trabalho.

A todos o nosso muito obrigado.

## **Introdução**

*Neste século, o mundo é uma esfera em crescimento, e para que possamos crescer com ele, necessitamos de estar ligado a todo ele.*

*Nisto, aparecem os sistemas móveis, necessários para podermos acompanhar o seu crescimento.*

*Este trabalho de final de curso, enquadra-se no âmbito deste tipo de produtos, uma vez que tem como objectivo permitir o acesso, a partir de uma larga variedade de dispositivos móveis.*

*Mais especificamente, este trabalho foca-se no Kusco - Messaging do Grupo Lusófona.*

*É um serviço da ULHT que permite a partir do telemóvel conseguir obter toda a informação sobre a vida académica, assuntos de tesouraria e secretaria, mudanças de sala, em suma tudo aquilo que interessa.*

*Este serviço foi elaborado pelo departamento de investigação da ULHT, usando uma plataforma da OutSystems Hub Edition™, um serviço gerenciado pela Optimus.*

*Trata-se do SMS PRO, um serviço que permite aos profissionais e empresas comunicar com os seus parceiros de negócio - clientes, fornecedores e colaboradores – de forma simples e rápida, com total flexibilidade e controlo de custos, garantindo assim uma maior flexibilização de utilização.*

*Este serviço foi desenvolvido usando a plataforma OutSystems Hub Edition™. Esta plataforma oferece um ambiente de desenvolvimento visual, extremamente intuitivo e fácil de utilizar, que permite o desenvolvimento rápido de aplicações multi-canal.*

## **Abstract**

*In this century, the world is a sphere in growth, and so we can grow with it, we need to be connected to all of it.*

*Herein, appear the mobile systems, needed to be able to accompany its growth.*

*This work of final course falls within the ambit of this type of products, since it aims to allow access from a wide variety of mobile devices.*

*More specifically, this work focuses on Kusco - Messaging of Lusófona Group.*

*Is a service that allows ULHT from mobile phone to get all the information about academic life, Treasury Affairs and Secretary, changing room, in short everything that matters.*

*This service was prepared by Department of investigation ULHT, using a platform of OutSystems Hub Edition™, a managed service by Optimus.*

*This is the SMS PRO, a service that enables professionals and companies communicate with their business partners – customers, suppliers and employees – simple and quick manner, with total flexibility and cost control, thus guaranteeing greater flexibility of use.*

*This service was developed using the OutSystems Hub Edition™ platform. This platform offers a visual development environment, extremely intuitive and easy to use, which enables the rapid development of multi-channel applications.*

## Introdução ao Kusco

### 1.1. Como Aceder a plataforma Kusco:

Para aceder a este serviço é necessário efectuem o registo no canal SMS antes de subscreverem qualquer um dos serviços acima descritos.

Este registo deve ter em conta os seguintes passos:

1. login no NetP@, na secção dos serviços de Secretaria, seleccionando a opção Registrar Canal.



Figura 0.1 – Opção “Registrar Canal”

2. Na caixa de selecção o aluno deve seleccionar Canal SMS, preencher o campo de texto Número de Telemóvel e clicar em Submeter Registo.

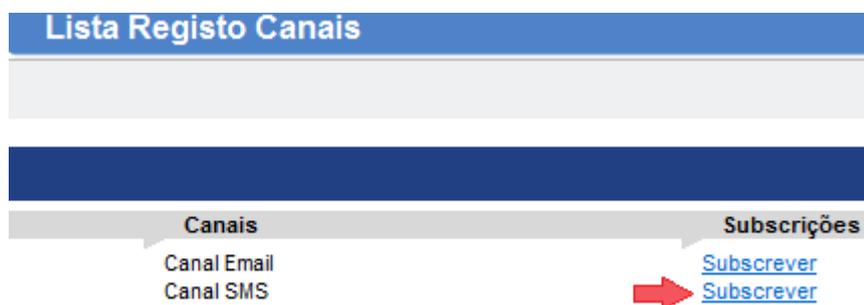


Figura 0.1 – Lista Registo Canais

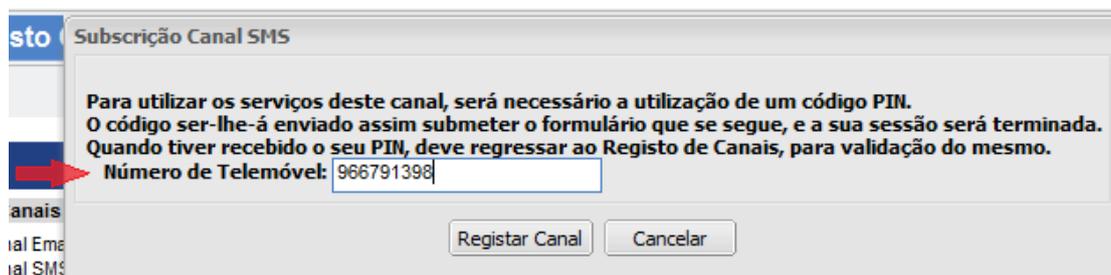


Figura 0.3 – Janela – Subscrição Canal SMS

3. Por questões de segurança, após a submissão do registo será terminada a sessão do NetP@.
4. Alguns minutos depois, o aluno receberá um sms com um código de segurança (PIN) que servirá para validar o registo previamente efectuado.

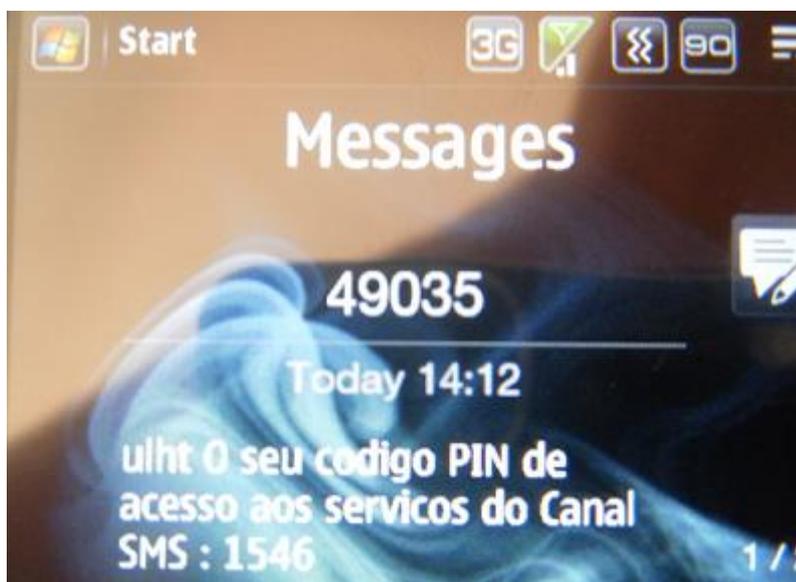


Figura 0.4 – sms com código segurança



Figura 0.5 – Validação Registo Canal

## 1.2. Instruções de Utilização

Após um novo login no NetP@, e acedendo à lista de consultas disponíveis para subscrição, o aluno deverá clicar no link Subscrever Consulta ou Subscrever Alerta correspondente à consulta pretendida.

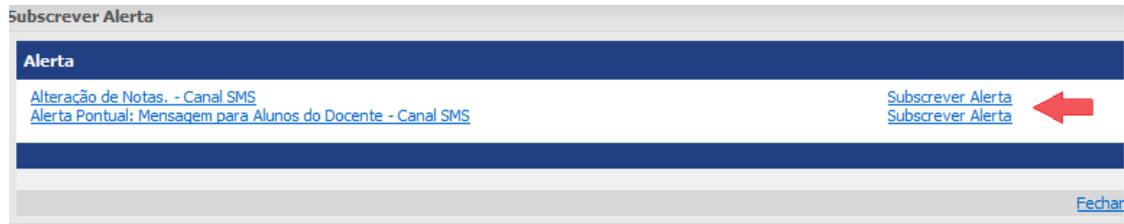


Figura 0.6 – Subscrever Alerta

Este processo deverá ser repetido para cada consulta que deseje subscrever. O número a utilizar para as consultas é o 4902 e o texto a enviar para as consultas deve ser sempre redigido em letras minúsculas.

### 1.3. Consultas

#### 1.3.1. Horário

- No caso de pretender consultar o seu horário, o aluno deverá enviar, para o número 4902, o texto ***ulht h*** seguido de um espaço e de um dos caracteres seguintes:
  - ***s*** - Sábado;
  - ***d*** - Domingo;
  - ***2*** - Segunda-Feira;
  - ***3*** - Terça-Feira;
  - ***4*** - Quarta-Feira;
  - ***5*** - Quinta-Feira;
  - ***6*** - Sexta-Feira.
- Por exemplo, ao enviar o texto ***ulht h 5***, o aluno receberá o seu horário completo de Quinta-Feira da semana corrente. Caso não inclua o código do dia (***ulht h***) o aluno receberá o horário das próximas três aulas, incluindo a que possa estar a decorrer na altura.

#### 1.3.2. Notas

- Relativamente à consulta de nota final, o aluno deve enviar o texto ***ulht n*** seguido de um espaço e da identificação da disciplina pretendida. A identificação da disciplina pode consistir no nome completo da disciplina ou, abreviatura(s) do nome da disciplina ou, do código da disciplina.

### 1.3.3. Saldo

- Para o aluno consultar o saldo de tesouraria deve enviar o texto ***ulht s*** para o **4902**, seguido de um espaço, como por exemplo ***ulht s***. Através desta consulta o aluno receberá o seu saldo de tesouraria, acompanhado do número de aluno.

### 1.3.4. Referência Multibanco

- Através desta Consulta pode ver a sua Referência Multibanco, a pagamento.
- Basta enviar, para o número **4902**, o código ***ulht mb*** no texto da sua mensagem. Por exemplo: ***ulht mb***.

### 1.3.5. Dados Administrativos

- Caso pretenda consultar os seus dados de aluno, deve enviar o texto ***ulht d*** para o **4902**, recebendo em seguida o seu número de aluno, nome, nome do seu curso, código do seu curso e nome de utilizador NetP@.

### 1.3.6. Cancelamento

- Para cancelar qualquer consulta, o utilizador deverá clicar no símbolo **X** correspondente à consulta cuja subscrição deseja anular. Confirmando ou não seguidamente a eliminação da consulta.

## **1.4. Alertas**

### **1.4.1. Alerta de Alteração de Nota**

- Através deste alerta o aluno recebe uma mensagem sempre que uma nota sua seja modificada. A mensagem conterá o nome da disciplina e respectivo código, para que possa efectuar a consulta de nota, se assim o desejar.

### **1.4.2. Alerta Pontual: Mensagem para Alunos do Docente**

- É um alerta que é enviado pelo docente de uma disciplina/turma a todos os seus alunos. O conteúdo do alerta é enviado pelos serviços da universidade a pedido do docente. O conteúdo do alerta é definido pelo docente.

Exemplo: Se o docente se atrasar para uma aula, pode enviar um SMS com essa indicação aos alunos da respectiva turma.

## **2. Estado da Arte**

### **2.1. Estado da Arte**

Este capítulo resulta da investigação de outras aplicações que permitem que o kusco seja uma aplicação segura e eficaz. Neste capítulo são também analisadas as tecnologias de suporte à realização de um sistema desta envergadura, nomeadamente baseada em software livre em java.

### **2.2. Tecnologias de Suporte**

De seguida são apresentadas algumas tecnologias que deram suporte à realização do Kusco; suporte desde a concepção até a implementação.

#### **2.2.1. Java Message Service (JMS)**

JMS é uma parte da plataforma Java, Enterprise Edition e é definido por uma especificação desenvolvida no âmbito do Java Community Process como JSR 914.

As diferentes aplicações de software conseguem comunicar sem que usem um paradigma de mensagens, mudando o formato dos dados da mensagem, onde os dados se podem converter em ficheiro text.

Este paradigma é frequentemente usado em sistemas de negócios (sistemas de informação), são geralmente relacionados aos sistemas de mensagens das empresas, ou Message-Oriented Middleware (MOM).

## JMS API

JMS (Java message service) é uma Java API é uma aplicação que permite, criar, enviar e ler mensagens, foi criada para ser fácil, ampla que permite-se o uso de vários fornecedores de MOM.

JMS API permite a comunicação que é não só flexível, mas como também é confiável e assíncrona.

## Modelos de mensagens JMS

JMS suporta 2 modelos: **point-to-point** (PTP) e **publish/subscribe** (pub-and-sub).

- Modelo de mensagens ponto a ponto [Queque] (fila):
  - É destinado a entrega individual de mensagens através de um canal virtual chamado [Queque]

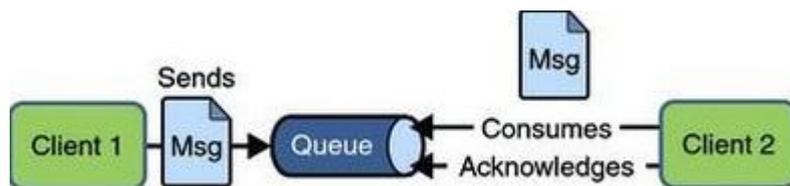


Figura 2.1 – Modelo point-to-point

- Modelo editor/assinante:
  - É destinado a uma transmissão de mensagens de um-para-muitos através de um canal virtual chamado [tópico]

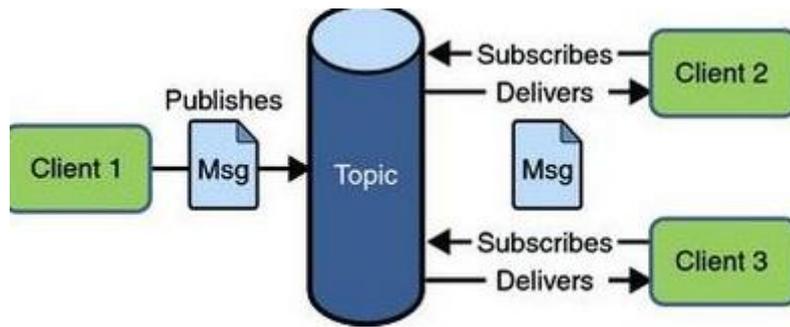


Figura 2.2 – Modelo publish/subscribe

JMS queues (filas) e topic (tópicos) são os destinos JMS e são vinculados no ambiente JNDI e disponibilizados para aplicativos J2EE.

### 2.2.2. Message-driven beans

Message-driven beans podem implementar qualquer tipo de mensagens. Mas usualmente, implementam a tecnologia Java Message Service (JMS).

Uma message-driven bean é um enterprise bean que permite as aplicações Java EE processar mensagens de forma assíncrona. Normalmente, ele actua como um ouvinte de mensagem JMS, que é semelhante a um ouvinte de eventos, excepto que ele recebe mensagens JMS em vez de eventos. As mensagens podem ser enviadas por qualquer componente Java EE (um aplicativo cliente, outra empresa bean ou um componente web) ou por um sistema que não usa a tecnologia Java EE ou aplicação JMS. Message-driven beans podem processar mensagens JMS ou outros tipos de mensagens.

Esta aplicação demonstra como enviar mensagens de uma enterprise bean, neste caso, um bean de sessão em vez de uma aplicação cliente.

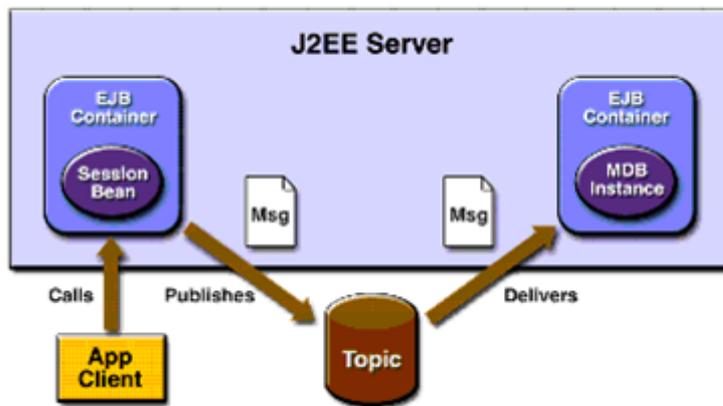


Figura 2.3 – Aplicação de envio de um enter

### 2.2.3. Plugins

Um plugin é um programa de computador usado para adicionar funções a outros programas maiores, provendo alguma funcionalidade especial ou muito específica. Geralmente pequeno e leve, é usado somente sob demanda.

### 2.2.4. Colector

Coletor de lixo (em inglês: garbage collector, ou o acrônimo GC) é um processo usado para a automação do gerenciamento de memória. Com ele é possível recuperar uma área de memória inutilizada por um programa, o que pode evitar problemas de vazamento de memória, resultando no esgotamento da memória livre para alocação.

Esse sistema contrasta com o gerenciamento manual de memória, em que o programador deve especificar explicitamente quando e quais objectos devem ser desalocados e retornados ao sistema. Entretanto, muitos sistemas usam uma combinação das duas abordagens.

Os princípios básicos do colector de lixo são encontrar objectos de um programa que não serão mais acessados no futuro, e desalocar os recursos utilizados por tais objectos. Tornando a deslocação manual de memória desnecessária, e geralmente proibindo tal prática, o coletor de lixo livra o programador de se preocupar com a liberação de recursos já não utilizados, o que pode consumir uma parte significativa do desenvolvimento do software. Também evita que o programador introduza erros no programa devido a má utilização de ponteiros.

## **2.2.5. SMSPro Optimus suportado pela plataforma OutSystems Hub Edition™**

### **2.2.5.1. SMSPro Optimus WebService Interface**

WebService Interface oferece mecanismos com um padrão base simples e aberto de fácil comunicação com a Internet.

Web service technology é baseada em mensagens SOAP (Simple Object Access Protocol), que são codificadas na forma de XML e transportadas por HTTP.

Um Web service interface pode ser visto como uma aplicação universal (API) onde a interface é expressa de forma independente da plataforma.

### **2.2.5.2. OutSystems**

A empresa OutSystems é uma empresa portuguesa e representa o produto “OutSystems Hub Edition”.

Por se tratar mais de uma framework e não de um produto não é tão comportável com os outros sistemas analisados, no entanto, dada a sua importância é justo fazer uma análise mais aprofundada.

Este produto suporta todo o processo de desenvolvimento, distribuição e gestão de aplicações para Internet, Intranet ou mesmo móveis e que ligam utilizadores móveis ou fixos a sistemas empresariais existentes e a base de dados.

Com esta infra-estrutura as empresas fornecedoras de serviços, conseguem reduzir para metade o tempo de distribuição de uma aplicação e reduzem 30% o tempo de desenvolvimento comparado com soluções tradicionais alternativas, segundo números da empresa.

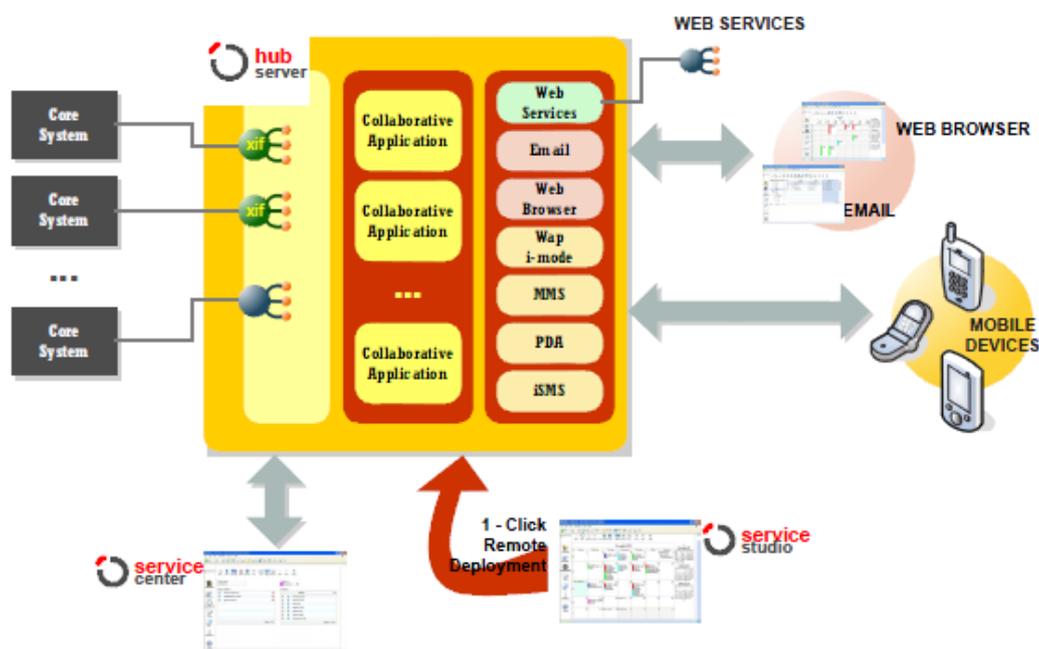


Figura 2.4 – Arquitectura do OutSystems Hub Edition

## OutSystems Hub Edition

Segundo a figura 2.6, o “OutSystems Hub Edition” junta uma ferramenta de desenvolvimento visual, o “Service Studio”, e uma plataforma escalável, o “Servidor Hub”. Este servidor permite ligação com firewalls, servidores de correio electrónico, fornecedores SMS, servidores de autenticação, WAP gateways, entre outros sistemas.

Tem a capacidade de servir ainda várias aplicações com contextos diferentes.

Disponibiliza para gestão das aplicações uma consola de administração, denominada por Service Center.

A plataforma "OutSystems Hub Server" está desenvolvida sobre a framework .NET, fazendo uso intensivo de standards como por exemplo XML, SOAP, e Web services. Dentro da tecnologia wireless suporta ou faz uso de SMS, MMS, WAP, WAP Push, GSM, CDMA (Code-Division Multiple Access) e GPRS.

O "OutSystems Hub Edition" não é um sistema fechado; permite que programadores possam criar e acrescentar novas livrarias ao sistema já existente. Disponibiliza ferramentas para controlar visualmente estas novas aplicações.

### **Integração rápida e fácil com sistemas existentes**

Sistemas existentes podem recombina-se com novos serviços que herdaram as características da plataforma Hub Edition: disponibilidade, eficiência, mecanismos de caching, mecanismos de segurança centralizada. O Hub Edition importa de uma forma nativa e publica Web Services permitindo um modelo de integração rápido e eficiente.

### **Criação de aplicações Intranet, Internet e Móveis**

Uma aplicação pode ser sempre integrada com novos serviços de uma forma simples e eficiente através da ferramenta Service Studio. Rapidamente consegue-se compor visualmente uma interface Web, de correio electrónico ou móveis, um modelo de dados, lógica de negócio, e políticas de segurança, tudo integrado com sistemas externos.

Conforme já referido, constata-se que o “OutSystems Hub Edition” não é um produto final mas sim uma framework de desenvolvimento. Ainda assim, verifica-se que com o tipo de serviços que disponibiliza permite criar sistemas com as mesmas características que o que pretendemos criar.

### **Aplicações comerciais**

Sendo um produto português e natural que a maior parte das suas aplicações seja em empresas portuguesas. Esse facto é visível observando a lista de empresas que utilizam este

Produto. Ainda assim não se confina a empresas nacionais sendo também utilizado em produtos de empresas de outras nacionalidades.

- Telefónica Móviles Corporation - Centro de mensagens.

Um dos produtos do “OutSystems”, o Centro de Mensagens “OutSystems”, foi seleccionado como a plataforma recomendada para as operadoras sem fios do grupo Telefónica Móviles. Um exemplo foi em Espanha cuja companhia passou a suportar todo um conjunto de tecnologias (MMS, WAP Push, etc) no topo das quais serviços inovadores são facilmente configurados. Outras operadoras actuando em mercados emergentes têm à sua disposição um conjunto completo de serviços de mensagens SMS, EMS (Enhanced Messaging Service) , MMS, Wap Push, totalmente configuráveis e parametrizáveis à medida das necessidades locais.

- Optimus - Introdução de novos serviços.

Operadora sem fios em Portugal com 2.5 milhões de subscritores desenvolveu o seu portal WAP Optimus Zone usando o “Outsystems Hub Edition”.

Desde a adopção desta plataforma, muitas aplicações foram desenvolvidas disponibilizando serviços de valor acrescentado aos seus clientes Optimus. Geo-SMS e SMS-Pro são apenas alguns exemplos de serviços.

### 3. Visão Geral do Projecto

Neste projecto vamos mostrar a reengenharia das consultas do Kusco utilizando o Eclipse uma plataforma Java e um servidor mySql. Criamos um programa que permite fazer consultas a partir de um request e um receiver que nos dá a consulta pretendida.

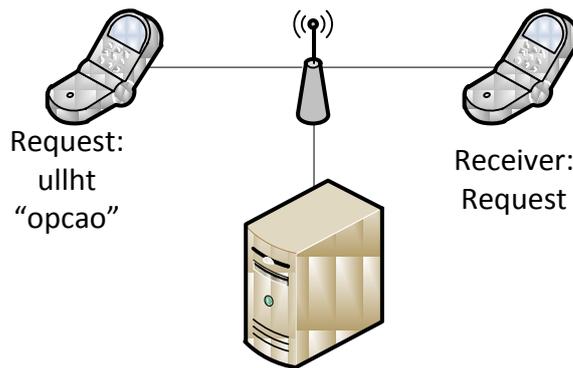


Figura 3.1 – Aplicação Consultas

Neste projecto, vamos utilizar a opção driver MySQL Connector/J, que permite que você desenvolva aplicações Java que interagem com o seu servidor MySQL.

MySQL Connector/J funciona no âmbito da interface Java JDBC, uma API que permite que programas em Java usem servidores de base de dados de uma maneira portátil.

Esta abordagem utiliza uma arquitetura de duas camadas:

- O nível superior é visível para programas de aplicação e apresenta uma interface abstrata para conectar e usar mecanismos da base de dados. A interface da aplicação não depende de detalhes específicos para motores particulares.

- O nível inferior consiste de drivers para os motores individuais de base de dados. Cada driver manipula os detalhes necessários para mapear a interface da aplicação abstracta para operações que compreenderá um mecanismo específico.

A interface JDBC permite aos desenvolvedores escrever aplicações que possam ser usadas com diferentes bases de dados e com um mínimo de esforço de portagem. Depois de instalar-se uma driver para um mecanismo de um determinado servidor, a aplicação JDBC pode comunicar com qualquer servidor desse tipo. Ao usar o MySQL Connector/J, o nosso programa em Java pode aceder ao à minha base de dados MySQL.



Figura 3.2. - JDBC

### ***3.1. Resolução de problemas***

A escrita de um programa de computador tem, normalmente, como objectivo resolver um ou mais problemas. Contrariamente ao que se possa pensar, na maioria das vezes a principal dificuldade não reside na sintaxe ou na semântica da linguagem de programa utilizada, mas sim na procura de soluções para o problema em causa, independentemente da forma ou da linguagem usadas para descrever essa solução.

Assim, o passo fundamental na criação de um programa é a definição do seu algoritmo.

### 3.2. Algoritmo

Um algoritmo pode ser caracterizado como um conjunto finito de instruções bem definidas e não ambíguas, que executadas por uma determinada ordem resolvem um problema.

Assim para conseguir escrever o nosso programa de consultas, é necessário conceber os respectivos algoritmos. Nesta tarefa há que considerar três componentes principais:

- ***Estado inicial***, ou seja, quais os dados da entrada do problema;
- ***Estado final***, ou seja, o que se pretende obter;
- ***Transformação***, ou seja, os passos necessários para transformar o estado inicial no estado final.

Durante a análise do nosso problema será necessário compreendê-lo e dividi-lo nas suas três componentes. O estado inicial e o estado final devem ser identificados claramente. Nesta fase, o objectivo é obter uma panorâmica geral do problema e da transformação necessária, sem ainda procurar definir detalhe como é que este vai ser.

### 3.3. *Algoritmo Consulta*

#### 3.3.1. Algoritmo main

Durante a análise do problema é necessário identificar o estado inicial, o estado final e o processo de transformação.

O **estado inicial** corresponde aos dados de entrada do problema, ou seja, neste caso, as opções necessárias para a consulta:

- Telemovel: número associado ao utilizador;
- Opção: Ulht h (pesquisa do horário).

O **estado final** corresponde aos resultados esperados, ou seja:

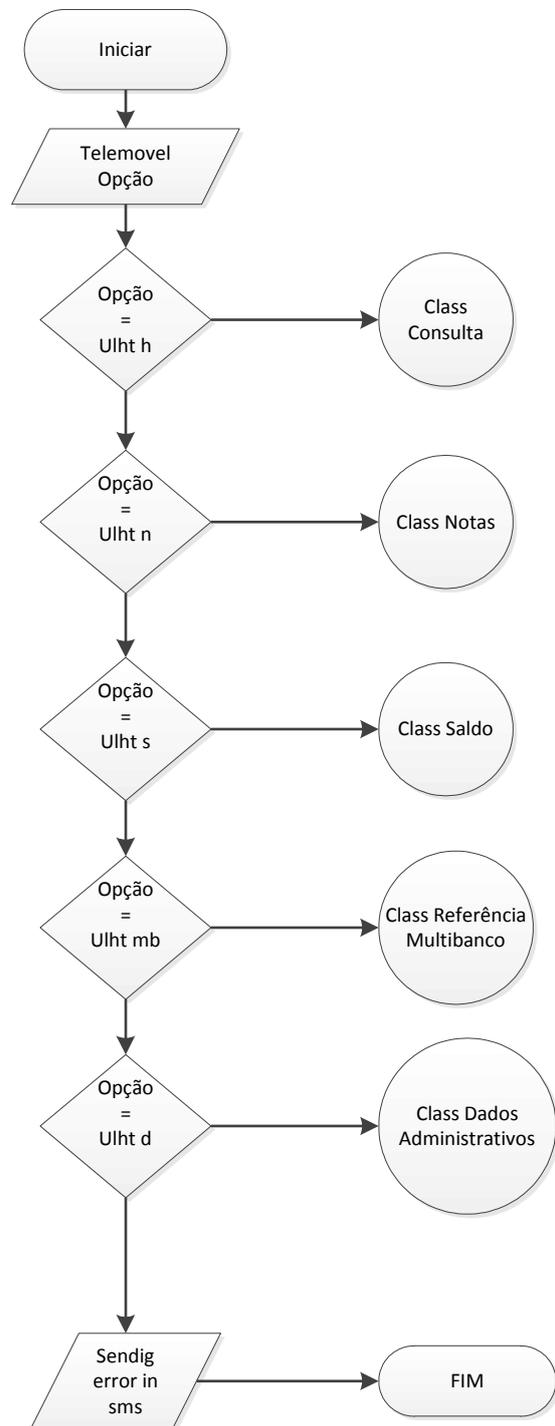
Horário da semana.

O processo de **transformação** é o processo de pesquisa e pode ser descrito genericamente como:

- Pesquisar a opção pretendida com associação do n° de telemóvel do utilizador.

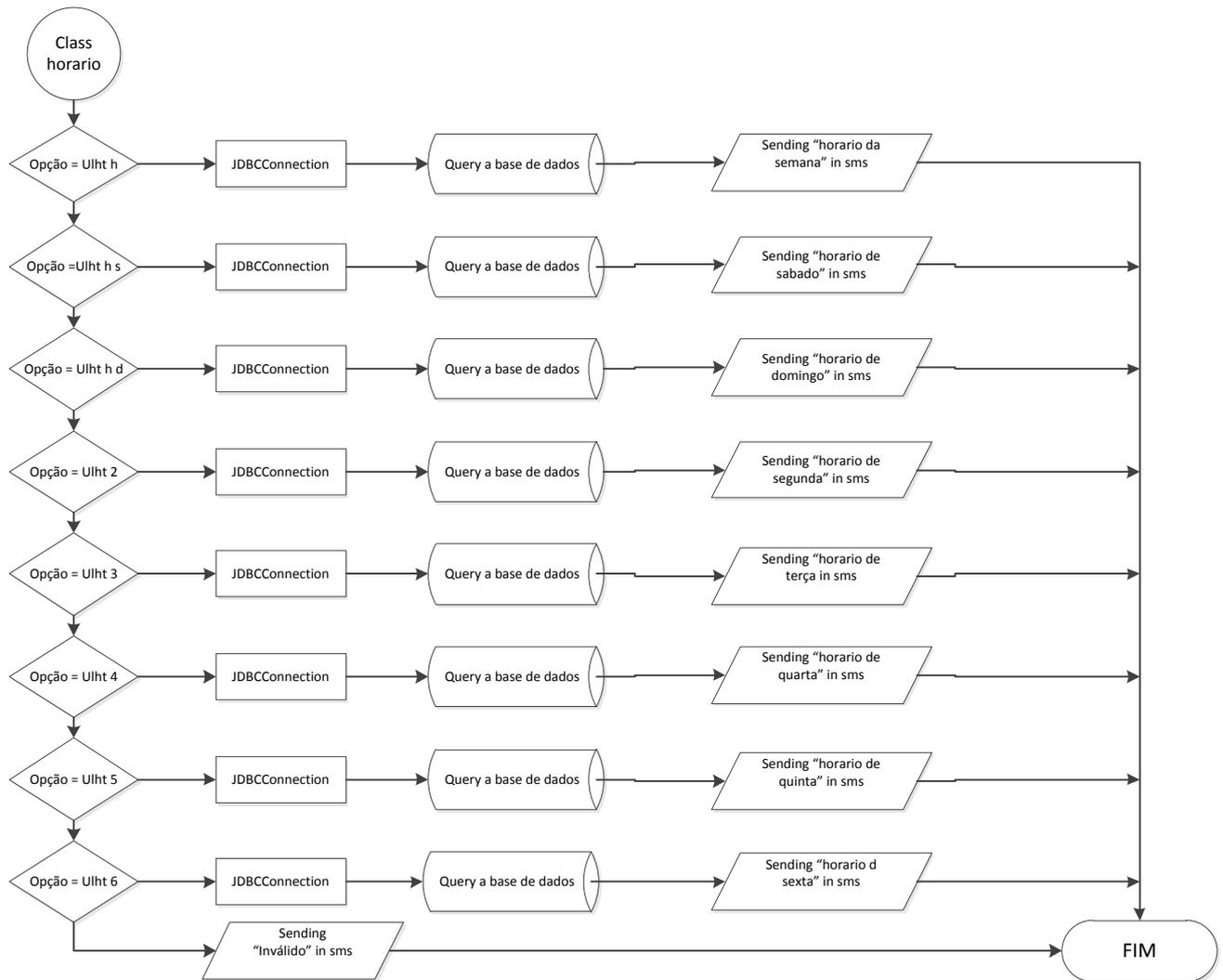
Claro que esta descrição é muito geral e seria suficiente para a maioria das pessoas conseguir resolver o problema. No entanto, não é possível fornecer ao computador instruções como estas e esperar que se resolva o problema. É necessário utilizar instruções mais elementares que possam ser executadas pelo computador. Por exemplo:

- 1- Iniciar
- 2- Ler Telemovel e Opção;
- 3- If (opção = ulht h) -> seguir para classe horario, se não passo 4;
- 4- else if (opção = ulht n) -> seguir para classe notas, se não, passo 5;
- 5- else if (opção = ulht s) -> seguir para classe saldo, se não, passo 6;
- 6- else if (opção = ulht mb) -> seguir para classe Referência Multibanco, se não, passo 7;
- 7- else if (opção = ulht d) -> seguir para classe Dados
- 8- Adm, se não, passo 8;
- 9- Enviar mensagem de “Erro”;
- 10- Fim.



Fluxograma 3.1- Algoritmo main ou consulta.

### 3.3.2. Algoritmo horário



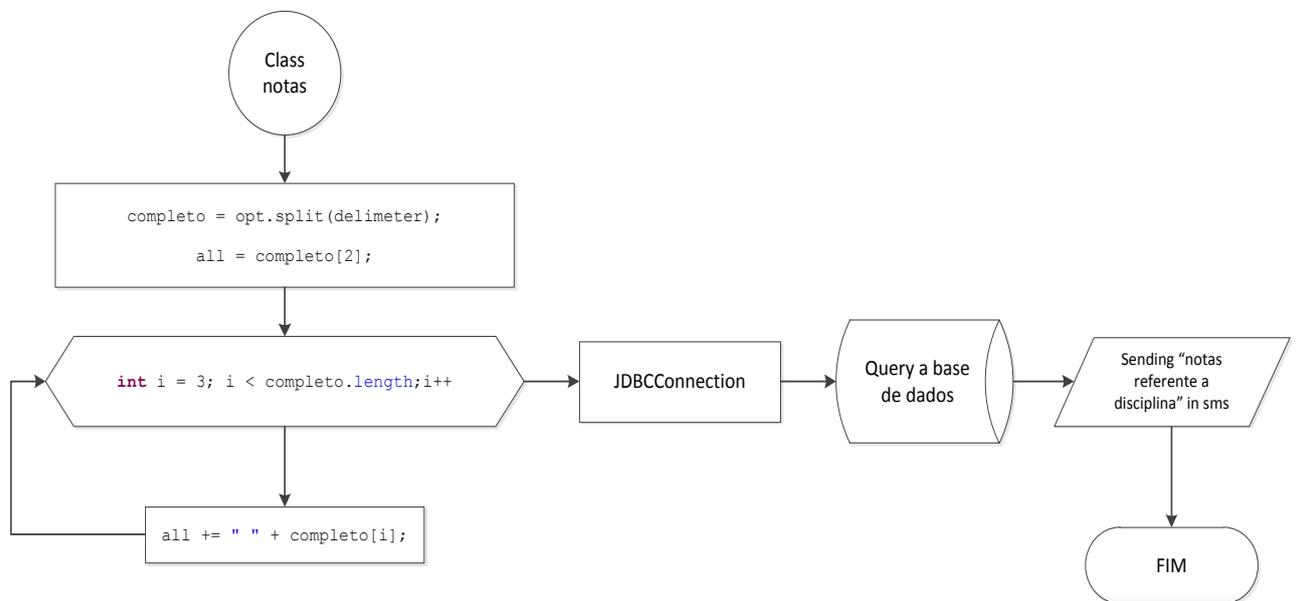
Fluxograma 3.2 – Algoritmo horário

**Algoritmo:**

- 1- Connection to Class horário;
- 2- If (opção = ulht h) -> fazer uma uma query da consulta a base de dados e receber de imediato a resposta dessa consulta, se não, passo 3;
- 3- If (opção = ulht h s) -> fazer uma uma query da consulta a base de dados e receber de imediato a resposta dessa consulta, se não, passo 4;

- 4- If (opção = ulht h d) -> fazer uma query da consulta a base de dados e receber de imediato a resposta dessa consulta, se não, passo 5;
- 5- If (opção = ulht h 2) -> fazer uma query da consulta a base de dados e receber de imediato a resposta dessa consulta, se não, passo 6;
- 6- If (opção = ulht h 3) -> fazer uma query da consulta a base de dados e receber de imediato a resposta dessa consulta, se não, passo 7;
- 7- If (opção = ulht h 4) -> fazer uma query da consulta a base de dados e receber de imediato a resposta dessa consulta, se não, passo 8;
- 8- If (opção = ulht h 5) -> fazer uma query da consulta a base de dados e receber de imediato a resposta dessa consulta, se não, passo 9;
- 9- If (opção = ulht h 6) -> fazer uma query da consulta a base de dados e receber de imediato a resposta dessa consulta, se não, passo 10;
- 10- Enviar mensagem “Inválido”;
- 11- FIM

### 3.3.3. Algoritmo Notas



Fluxograma 3.3 – Algoritmo notas.

**Algoritmo:**

- 1- Connection to Class notas;
- 2- Ler completo e all;
- 3- Se (**int i = 3; i < completo.length; i++**), execute a instrução, (**all += “ + completo[i]**), e volta ao passo 3, quando (**i < completo.length**), não se provar, passo 4;
- 4- Enviar uma query da consulta a base de dados e receber de imediato a resposta dessa consulta numa mensagem;
- 5- FIM.

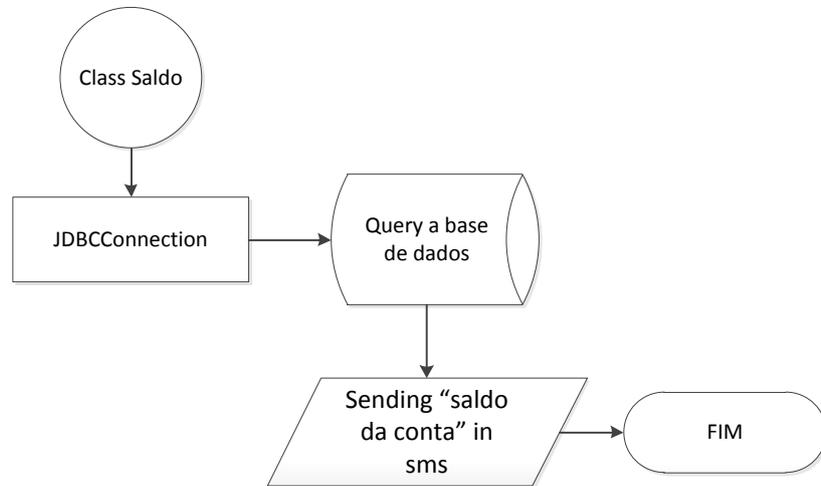
**For**

O funcionamento desta instrução tem três componentes que têm a seguinte função na dinâmica da repetição:

- Inicialização – é executada apenas no início do ciclo. Serve para iniciar a variável “i” que vai controlar o número de vezes que as instruções vão ser repetidas;
- Condição – é calculada antes de cada execução das instruções. Se o seu resultado for verdadeiro, as instruções são executadas, se for falso, o ciclo termina;

Acção – é executada automaticamente em cada iteração após as instruções do ciclo. Serve para calcular o novo valor da variável de controlo (que no nosso caso é a variável “i”).

### 3.3.4. Algoritmo Saldo

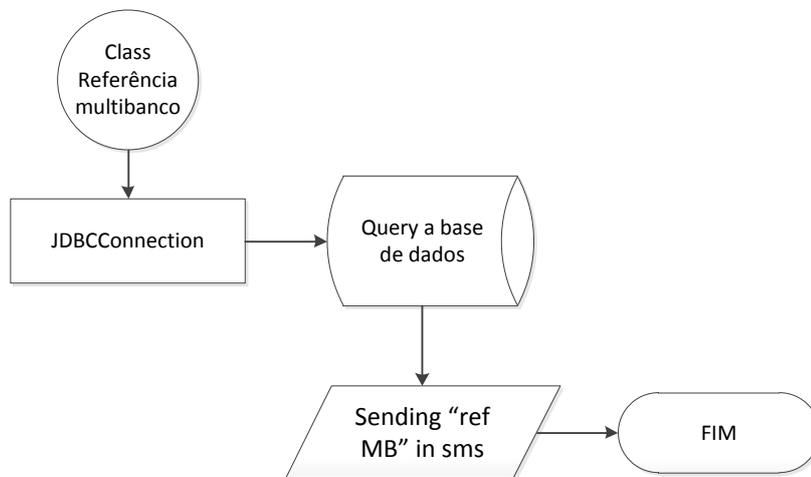


Fluxograma 3.4 – Algoritmo Saldo

***Algoritmo:***

- 1- Connection to Class saldo;
- 2- Enviar uma query da consulta a base de dados e receber de imediato a resposta dessa consulta numa mensagem;
- 3- FIM.

### 3.3.5. Algoritmo Referência Multibanco

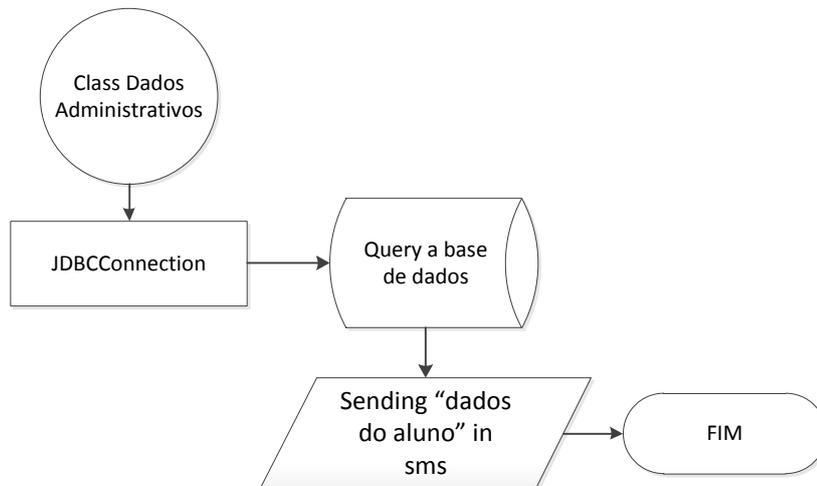


Fluxograma 3.5 – Algoritmo Referência multibanco

**Algoritmo:**

- 1- Connection to Class Referência Multibanco;
- 2- Enviar uma query da consulta a base de dados e receber de imediato a resposta dessa consulta numa mensagem;
- 3- FIM.

### 3.3.6. Algoritmo Dados Administrativos



Fluxograma 3.6 – Algoritmo Dados Administrativos

#### ***Algoritmo:***

- 1- Connection to Class Dados Administrativos;
- 2- Enviar uma query da consulta a base de dados e receber de imediato a resposta dessa consulta numa mensagem;
- 3- FIM.

### 3.4. ***Base de dados***

De uma forma simplista, poderemos dizer que uma base de dados consiste numa colecção de dados estruturados, organizados e armazenados de forma persistente. Com isto, criamos uma base de dados eficiente que pudesse ligar facilmente com o nosso scrip java com uma aplicação JBDC.

A linguagem SQL permite fazer praticamente tudo sobre uma base de dados. No entanto, à linguagem faltam alguns mecanismos comuns na generalidade das linguagens de programação como instruções ou comandos do tipo If-Then-Else, Select-Case, While, For, funções, procedimentos, etc.

Assim e para ultrapassar este problema, decidimos juntar a linguagem SQL com a linguagem java. É por isso que a interface da nossa aplicação seja implementada numa linguagem orientada a objectos e que essa linguagem comunique com a base de dados onde os dados são armazenados através da linguagem SQL.

### Diagrama entidade associação:

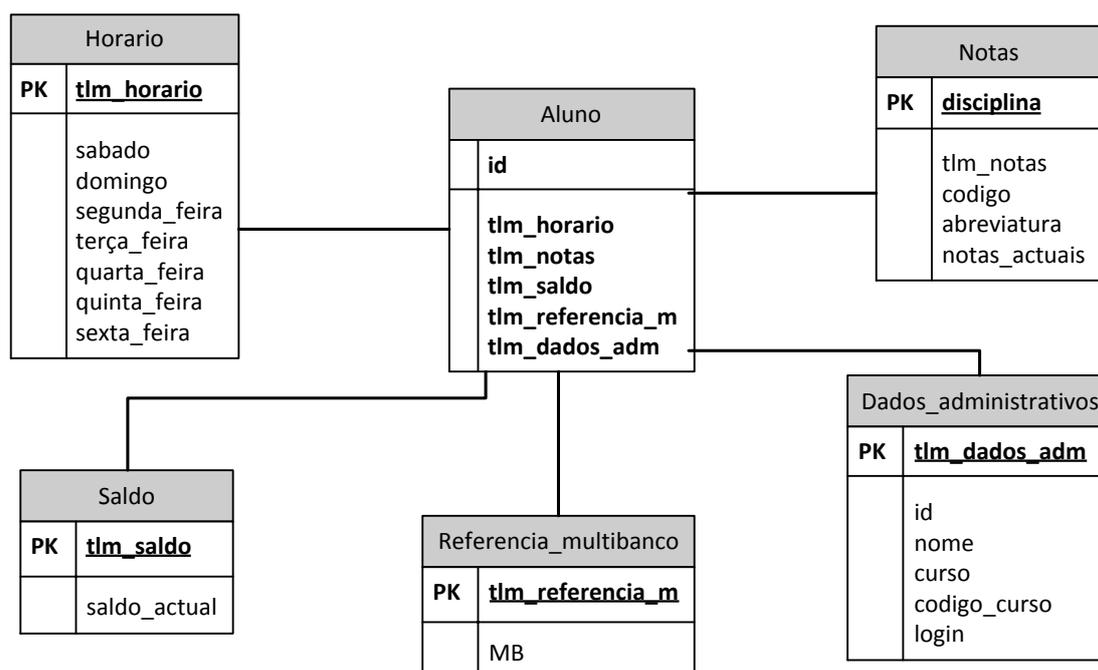


Diagrama 3.1 – Diagrama da Base de Dados

### 3.4.1. Tabela Aluno

Aluno	
id	
	t1m_horario t1m_notas t1m_saldo t1m_referencia_m t1m_dados_adm

Diagrama 3.2 – Tabela Aluno

A tabela coluna é a tabela que vai relacionar-se com as outras tabelas, a existência desta tabela é meramente funcional, ajuda a que não exista resultados ambíguos e ajuda nas relações entre entidades.

### 3.4.2. Tabela Horário

Horario	
PK	<u>t1m_horario</u>
	sabado domingo segunda_feira terça_feira quarta_feira quinta_feira sexta_feira

Diagrama 3.3 – Tabela Horário

A tabela Horário irá usar a sua coluna t1m\_horario para efectuar a pesquisa de horários, consoante se é algum dia da semana que se quer saber ou mesmo a o horário da semana.

### 3.4.3. Tabela Notas

Notas	
<b>PK</b>	<b><u>disciplina</u></b>
	t1m_notas codigo abreviatura notas_actuais

Diagrama 3.4 – Tabela Notas

A tabela Notas esta desenhada de uma forma simples, para que o utilizador quando fizer uma consulta, consiga procurar o a nota desejada pela coluna disciplina, coluna código ou mesmo coluna abreviatura.

### 3.4.4. Tabela Saldo

Saldo	
<b>PK</b>	<b><u>t1m_saldo</u></b>
	saldo_actual

Diagrama 3.5 – Tabela Saldo

A tabela Saldo apenas usa a sua Primary key para fazer a sua pesquisa.

### 3.4.5. Tabela Referência Multibanco

Referencia_multibanco	
PK	<u>tlm_referencia_m</u>
	MB

Diagrama 3.6 – Tabela Ref Multibanco

A tabela Referência Multibanco esta desenhada de uma forma simples, para que o utilizador quando se conectar à base de dados o resultado obtido é o desejado.

### 3.4.6. Tabela Dados Administrativos

Dados_administrativos	
PK	<u>tlm_dados_adm</u>
	id nome curso codigo_curso login

Diagrama 3.7 – Tabela Dados Adm.

A tabela Dados Administrativos apenas com a coluna tlm\_dados\_adm consegue obter as seguintes colunas.

#### **4. Conclusão**

Este projecto foi bastante interessante, principalmente porque permitiu adquirir conhecimentos acerca de várias tecnologias utilizadas nas várias fases de desenvolvimento de um projecto e que poderão ser úteis no futuro ao longo da vida profissional. Ao longo do desenvolvimento deste projecto também adquirimos experiência em nos habituarmos a cumprir prazos, resolver situações imprevistas, planear e dividir bem as nossas tarefas, entre outras coisas.

Deparamo-nos também com os problemas normais que ocorrem no desenvolvimento de um projecto deste tamanho e que ainda não nos tinham acontecido durante o curso. Dificuldades essas de seguir noutras direcções, divergindo para programação de linguagem de um nível mais baixo.

Como já referido, este trabalho foi muito enriquecedor porque para além de ter a componente académica tem também uma componente comercial importante, o que sensibilizou também para outras questões facilidade de utilização, informação útil ao utilizador, etc.

Este projecto levou-nos a perceber que para os utilizadores o importante é a portabilidade e a mobilidade da informação, apenas com uma consulta a partir de um aparelho móvel, conseguimos obter a informação desejada sem nos termos que deslocar a órgãos de foro administrativo, como secretaria e mesmo tesouraria.

De notar que qualquer implementação sobre a aplicação é bastante fácil de efectuar, sendo a própria aplicação acessível em termos de aprendizagem, devido aos vários padrões arquitecturais utilizados e flexibilidade estrutural existente em toda a aplicação.

## Bibliografia

### Livros:

PEREIRA, Alexandre ; POUPA, Carlos – **Como escrever uma tese, monografia ou livro científico: usando o Word**. Lisboa : Sílabo, 2003. ISBN 972-618-290-5. 224p.

MENDES, António; MARCELINO, Maria – Fundamentos de Programação em JAVA 2  
3ª EDIÇÃO, FCA – Editora de Informática.

DAMAS, Luis - SQL Structued Query Languague  
6ª EDIÇÃO, FCA – Editora de Informática.

### Informação Web:

[http://pt.wikipedia.org/wiki/P%C3%A1gina\\_principal](http://pt.wikipedia.org/wiki/P%C3%A1gina_principal) – Enciclopédia Livre

<http://download.oracle.com/javase/tutorial/> - Java Tutorials

<http://www.outsystems.com/> - Out systems

[http://smspro.optimus.pt/app\\_config/\(S\(jrpdmi45veyyrr55t0s1at55\)\)/smspro.aspx](http://smspro.optimus.pt/app_config/(S(jrpdmi45veyyrr55t0s1at55))/smspro.aspx) - Optimus SMS Pro

<http://www.eclipse.org/>

<http://eclipsetutorial.sourceforge.net/>

<http://www.cs.utk.edu/~cs365/examples/datacheck.html> - Java's Scanner

[www.youtube.com](http://www.youtube.com) – Tutorials sobre JDBC

<http://helpdesk.grupolusofona.pt/index.php/servicos/kusco> - Pagina kusco

<http://www.devdaily.com/java/edu/pj/pi010024/> - JDBC – Connecr to a SQL database with JDBC

[http://paginas.fe.up.pt/~ee04245/Relatorios/Relatorio\\_Final\\_Daniel%20Santos\\_ee04245.pdf](http://paginas.fe.up.pt/~ee04245/Relatorios/Relatorio_Final_Daniel%20Santos_ee04245.pdf) – Relatório Final de Curso

<http://paginas.fe.up.pt/~ee00111/relatorio.pdf> - -- Relatório Final de Curso

## Anexo 1 – Script Java – Programa Consultas

### Class Consultas (JAVA)

```
import java.util.Scanner;
import java.io.*;
import java.sql.*;
import javax.swing.*;

public class consultas
{

    public static void main(String args[])
    {

        try
        {

            //Declaracao
            Scanner in = new Scanner(System.in); //Regista o
telemovel do aluno, Opcao de "ulht h, ulht n

            //Ler valores

            System.out.print("numero de telemovel:");
            String telemovel = in.next();
            System.out.print("msg:");

            //Ler valores

            String opcao = new String();
            in.nextLine();
            opcao = in.nextLine();

            //repartir a string em tokens

            String [] completo;

            //completo[0]=variavel ulht... completo[1]=variavel
h

            String delimiter = " ";

            completo = opcao.split(delimiter);

            //INICIO DAS CONSULTAS

            //HORARIOS
            if(completo[0].toString().equalsIgnoreCase("ulht"))
```

```

        if (completo[1].toString().equalsIgnoreCase("h"))
        {
            System.out.print("Enviar mensagem de resposta
para '" + telemovel + "': ");
            horario horarioObject = new horario();
            horarioObject.novaOpcao(opcao, telemovel);
        }
        else
            //NOTAS
            if (opcao.startsWith("ulht n"))
            {
                System.out.print("Enviar mensagem de
resposta para '" + telemovel + "': ");
                notas notasObject = new notas();
                notasObject.n(opcao, telemovel);
            }
            else
                //SALDO
                if (opcao.equalsIgnoreCase("ulht s"))
                {
                    System.out.print("Enviar mensagem
de resposta para '" + telemovel + "': ");
                    saldo saldoObject = new saldo();
                    saldoObject.s(opcao, telemovel);
                }
                else
                    //REFERENCIA MB
                    if (opcao.equalsIgnoreCase("ulht
mb"))
                    {
                        System.out.print("Enviar
mensagem de resposta para '" + telemovel + "': ");
                        referencia_multibanco
referencia_multibancoObject = new referencia_multibanco();

                        referencia_multibancoObject.r(opcao, telemovel);
                    }
                    else
                        //DADOS ADMINISTRATIVOS
                        if
(opcao.equalsIgnoreCase("ulht d"))
                        {
                            System.out.print("Enviar mensagem de resposta para '" +
telemovel + "': ");
                            dados_administrativos
dados_administrativosObject = new dados_administrativos();

                            dados_administrativos.d(opcao, telemovel);
                        }
                        else
                        {

```

```

        System.out.print("Enviar mensagem de resposta para '" +
telemovel + "': Erro ");
                                throw new Exception
());
                                }
        }
        catch(Exception e)
        {
            //Aqui tratas das mensagens sms "ERRO!"
        }
    }
}

```

## Class Horario (JAVA)

```

public class horario {

    public void novaOpcao(String opt, String telemovel)
    {

        try{

            String [] completo;

            String delimiter = " ";

            completo = opt.split(delimiter);

            if (opt.equalsIgnoreCase("ulht h") ){
                //horario (semana)

                new JDBCConnection().runSemana("select * from
Horario where tlm_horario = '" + telemovel + "'");
            }
            else if(opt.equalsIgnoreCase("ulht h s")){
                //horario (sabado)

                new JDBCConnection().runMe("select sabado as
dia from Horario where tlm_horario = '" + telemovel + "'");
            }
            else if(opt.equalsIgnoreCase("ulht h d")){
                //horario (domingo)

                new JDBCConnection().runMe("select domingo as
dia from Horario where tlm_horario = '" + telemovel + "'");
            }
            else if(opt.equalsIgnoreCase("ulht h 2")){
                //horario (segunda)

                new JDBCConnection().runMe("select
segunda_feira as dia from Horario where tlm_horario = '" + telemovel
+ "'");
            }

        }
    }
}

```



```

completo = opt.split(delimiter);

String all = new String();
all = completo[2];
for(int i = 3; i < completo.length;i++ )
{
    all += " " + completo[i];

    /* ulht completo[0], h completo[1], Arquitectura
completo[2], de completo[3], Computadores [4]; */

    //Condição (3 < completo[4])
    //Executa a instrução seguinte (3 < completo[4])
    //Executa a instrução a repetirta a acção (i++) e i
passa para 4.

    //Condição (4 < completo[4])
    //Executa a instrução seguinte (4 < completo[4])
    //Ciclo Termina completo[2] + completo[3] +
completo [4];

    /* ulht completo[0], h completo[1], Sistemas
completo[2], Operativos completo[3]; */

    //Condição (3 < completo[3]), ou seja, é falsa, o
ciclo termina

    //Ciclo Termina completo[2] + completo[3];

}

new JDBCConnection().runMe("select notas_actuais as dia
FROM notas WHERE tlm_notas = '" + telemovel + "' AND (disciplina = '"
+ all.toLowerCase() + "' OR codigo = '" + all.toLowerCase() + "' OR
abreviatura = '" + all.toLowerCase() + "')");

}
catch (Exception e)
{
}
}
}

```

## Class Saldo (JAVA)

```

public class saldo {

    public void s(String opt, String telemovel)
    {
        try {

            String [] completo;

            String delimiter = " ";

            completo = opt.split(delimiter);

```

```

        new JDBCConnection().runMe("select saldo_actual as
dia from saldo where tlm_saldo = '" + telemovel + "'");

    }

    catch (Exception e)
    {
    }
}

```

## Class Referencia Multibanco (JAVA)

```

public class referencia_multibanco {

    public void r(String opt, String telemovel)
    {
        try {

            String [] completo;

            String delimiter = " ";

            completo = opt.split(delimiter);

            new JDBCConnection().runMe("select mb as dia from
referencia_multibanco where tlm_referencia_m = '" + telemovel + "'");

        }

        catch (Exception e)
        {
        }
    }
}

```

## Class Dados Administrativos (JAVA)

```

public class dados_administrativos {

    public static void d(String opt, String telemovel)
    {
        try {

            String [] completo;

            String delimiter = " ";

            completo = opt.split(delimiter);

            new JDBCConnection().runDados("select * from
dados_administrativos where tlm_dados_adm = '" + telemovel + "'");

        }

        catch (Exception e)
        {
        }
    }
}

```

```

    }

    catch (Exception e)
    {
    }
}
}

```

## Class JDBC Connection (JAVA)

```

import java.sql.*;

public class JDBCConnection {

    // OPCA0 3 (CONSULTA DADOS ADMINISTRATIVOS ("select * FROM
    dados_administrativos"))

    public void runDados (String Query)
    throws Exception {

        /* run driverTest method shown below */
        driverTest();

        /* make the connection to the database */
        Connection conMe = makeCon();

        /* now run a select query of the intended database */
        exeQuery3 (conMe, Query);

        /* close the database */
        conMe.close();

    }

    protected void exeQuery3(Connection con, String sqlStatement)
    throws Exception {

        try {
            Statement cs = con.createStatement();
            ResultSet sqls = cs.executeQuery(sqlStatement);

            while (sqls.next()) {
                String id =
                (sqls.getObject("id").toString());
                String nome =
                (sqls.getObject("nome").toString());
                String curso =
                (sqls.getObject("curso").toString());
                String codigo_curso =
                (sqls.getObject("codigo_curso").toString());
                String login =
                (sqls.getObject("login").toString());
            }
        }
    }
}

```

```

        System.out.println(id);
        System.out.println(nome);
        System.out.println(curso);
        System.out.println(codigo_curso);
        System.out.println(login);

    }

    sqls.close();

    } catch (SQLException e) {
        System.out.println ("Error executing sql
statement");
        throw (e);
    }
}

// OPCA0 2 (CONSULTA HORARIO ("select * FROM horario")

public void runSemana (String Query)
throws Exception {

    /* run driverTest method shown below */
    driverTest();

    /* make the connection to the database */
    Connection conMe = makeCon();

    /* now run a select query of the intended database */
    exeQuery2 (conMe, Query);

    /* close the database */
    conMe.close();

}

protected void exeQuery2(Connection con, String sqlStatement)
throws Exception {

    try {
        Statement cs = con.createStatement();
        ResultSet sqls = cs.executeQuery(sqlStatement);

        while (sqls.next()) {
            String id2 =
(sqls.getObject("segunda_feira").toString());
            String id3 =
(sqls.getObject("terça_feira").toString());
            String id4 =
(sqls.getObject("quarta_feira").toString());
            String id5 =
(sqls.getObject("quinta_feira").toString());

```

```

        String id6 =
(sqls.getObject("sexta_feira").toString());
        String ids =
(sqls.getObject("sabado").toString());
        String idd =
(sqls.getObject("domingo").toString());

        System.out.println(id2);
        System.out.println(id3);
        System.out.println(id4);
        System.out.println(id5);
        System.out.println(id6);
        System.out.println(ids);
        System.out.println(idd);

    }

    sqls.close();

    } catch (SQLException e) {
        System.out.println ("Error executing sql
statement");
        throw (e);
    }
}

// OPCA0 INICIAL

public void runMe (String Query)
throws Exception {

    /* run driverTest method shown below */
    driverTest();

    /* make the connection to the database */
    Connection conMe = makeCon();

    /* now run a select query of the intended database */
    exeQuery (conMe, Query);

    /* close the database */
    conMe.close();

}

protected void driverTest () {

    try {

```

```

        Class.forName("org.gjt.mm.mysql.Driver");
        System.out.println("MySQL Driver Found");
    } catch (java.lang.ClassNotFoundException e) {
        System.out.println("MySQL JDBC Driver not found ...
");
    }
}

    protected Connection makeCon (String host, String database,
String user, String password)
    throws Exception {

        String url = "";
        try {
            url = "jdbc:mysql://" + host + ":3306/" + database;
            Connection con = DriverManager.getConnection(url, user,
password);
            System.out.println("Connection established to " + url + "...");
            return con;
        } catch (java.sql.SQLException e) {
            System.out.println("Connection couldn't be established to " +
url);
            throw (e);
        }
    }

    protected Connection makeCon () throws Exception
    {
        String url = "";
        try {
            url = "jdbc:mysql://localhost:3306/database";
            Connection con = DriverManager.getConnection(url,
"manolo", "manolo");
            System.out.println("Connection established to " + url + "...");
            return con;
        } catch (java.sql.SQLException e) {
            System.out.println("Connection couldn't be established to " +
url);
            throw e;
        }
    }

    protected void exeQuery(Connection con, String sqlStatement)
    throws Exception {

        try {
            Statement cs = con.createStatement();
            ResultSet sqls = cs.executeQuery(sqlStatement);

            while (sqls.next()) {
                String id =
(sqls.getObject("dia").toString());

                System.out.println(id);
            }

            sqls.close();

        } catch (SQLException e) {

```

```
        System.out.println ("Error executing sql
statement");
        throw (e);
    }
}

/* this bracket closes the class */
}
```

## Anexo 2 – Script SQL – Base dados Consultas

```
CREATE TABLE Aluno (  
id INTEGER NOT NULL,  
t1m_horario INTEGER NOT NULL,  
t1m_notas INTEGER NOT NULL,  
t1m_saldo INTEGER NOT NULL,  
t1m_referencia_m INTEGER NOT NULL,  
t1m_dados_adm INTEGER NOT NULL,  
PRIMARY KEY(id),  
CONSTRAINT FK_Horario FOREIGN KEY (t1m_horario) REFERENCES  
Horario(t1m_horario),  
CONSTRAINT FK_Notas FOREIGN KEY (t1m_notas) REFERENCES Notas(t1m_notas),  
CONSTRAINT FK_Saldo FOREIGN KEY (t1m_saldo) REFERENCES Saldo(t1m_saldo),  
CONSTRAINT FK_Referencia_multibanco FOREIGN KEY (t1m_referencia_m)  
REFERENCES Referencia_multibanco(t1m_referencia_m),  
CONSTRAINT FK_Dados_administrativos FOREIGN KEY (t1m_dados_adm) REFERENCES  
FK_Dados_administrativos(t1m_horario)  
);
```

```
CREATE TABLE Horario (  
t1m_horario INTEGER NOT NULL,  
sabado VARCHAR(45),  
domingo VARCHAR(30),  
segunda_feira VARCHAR(60),  
terça_feira VARCHAR(60),  
quarta_feira VARCHAR(60),  
quinta_feira VARCHAR(60),  
sexta_feira VARCHAR(60),  
PRIMARY KEY(t1m_horario)  
);
```

```
CREATE TABLE Notas (  

```

```
disciplina VARCHAR(40) NOT NULL,  
t1m_notas INTEGER NOT NULL,  
codigo VARCHAR (40),  
abreviatura VARCHAR (40),  
notas_actuais VARCHAR(10),  
PRIMARY KEY (disciplina)  
);
```

```
CREATE TABLE Saldo (  
t1m_saldo INTEGER NOT NULL,  
saldo_atual DECIMAL(10,2) NOT NULL,  
PRIMARY KEY(t1m_saldo)  
);
```

```
CREATE TABLE Referencia_multibanco (  
t1m_referencia_m INTEGER NOT NULL,  
mb VARCHAR(18) NOT NULL,  
PRIMARY KEY(t1m_referencia_m)  
);
```

```
CREATE TABLE Dados_administrativos (  
t1m_dados_adm INTEGER NOT NULL,  
id INTEGER NOT NULL,  
nome VARCHAR(40) NOT NULL,  
curso VARCHAR(30) NOT NULL,  
codigo_curso INTEGER NOT NULL,  
login VARCHAR(20) NOT NULL,  
PRIMARY KEY(t1m_dados_adm),  
CONSTRAINT FK_Aluno FOREIGN KEY (id) REFERENCES Aluno(id)  
);
```

## Script Intos

```
INSERT INTO aluno VALUES (20076165, 916929360, 916929360, 916929360, 916929360, 916929360);
```

```
INSERT INTO aluno VALUES (2007000, 911234567, 911234567, 911234567, 911234567, 911234567);
```

```
INSERT INTO horario VALUES (916929360, "", "", 'Segunda 09h-11h LP2 11h-13h Inst de Gestao', 'Terça 08h-11h TFC 14-17h Matematica', 'Quarta 09h-11h Código Penal 13-15h Turismo', 'Quinta 08h-11h Marketing 16-18h Biologia', 'Sexta 17h-19h S.O');
```

```
INSERT INTO horario VALUES (911234567, 'Sabado 09h-11h Eng Quimica 11h-13h Futebol', "", "", 'Quarta 09h-11h Fisica', 'Quinta 08h-13h Matematica', 'Sexta 17h-19h Inglês');
```

```
INSERT INTO dados_administrativos VALUES (916929360, 20076165, 'Manuel Monteiro', 'Informática de Gestao', 12, 'a20076165');
```

```
INSERT INTO dados_administrativos VALUES (911234567, 20070000, 'Antonio Monteiro', 'Matematica Aplicada', 322, 'a2007000');
```

```
INSERT INTO notas VALUES ('sistemas operativos', 916929360, '310', 'so', 'Nota: 12');
```

```
INSERT INTO notas VALUES ('arquitectura de computadores', 916929360, '311', 'aq', 'Nota 19');
```

```
INSERT INTO notas VALUES ('dança', 916929360, '312', 'dança', 'Nota: 2');
```

```
INSERT INTO notas VALUES ('trabalho final de curso', 911234567, '322', 'tfc', 'Nota: 14');
```

```
INSERT INTO notas VALUES ('analise e concepção de sistemas', 911234567, '201', 'acs', 'Nota: 13');
```

```
INSERT INTO referencia_multibanco VALUES (916929360, '0010020030056789');
```

```
INSERT INTO referencia_multibanco VALUES (911234567, '0000000000000001');
```

```
INSERT INTO saldo VALUES (916929360, '-333,22');
```

```
INSERT INTO saldo VALUES (911234567, '33,21');
```