Universidade Nova de Lisboa
OMNIS CIVITAS CONTRA SE DIVISA NON STABIT
Faculdade de Ciências e Tecnologia

DEPARTAMENTO DE INFORMÁTICA

Licenciatura em Engenharia Informática

RELATÓRIO DE ESTÁGIO PROFISSIONALIZANTE

Julho de 2012

JOÃO BERNARDO (19424)

ORIENTADOR: CARMEN PIRES MORGADO
COORDENADOR EXTERNO: JOÃO COSTA
INSTITUIÇÃO: INFOSISTEMA – SISTEMAS DE INFORMAÇÃO S.A.

Agradecimentos

Quero expressar o meu enorme agradecimento a todos os que tornaram possível a realização desta positiva experiência que foi a realização do estágio na Infosistema.

Em primeiro lugar, ao meu coordenador de estágio, Eng. João Costa, pela oportunidade que me deu, pela simpatia, profissionalismo e disponibilidade demonstrados desde o primeiro dia de trabalho.

Ao Pedro Ussman e ao Àlvaro Gago pelo companheirismo e pela disponibilidade para me ajudar quando necessário.

Em geral, a todos os colegas da Infosistema, pelo profissionalismo demonstrado e pela enorme simpatia, que facilitaram imenso a minha integração na instituição.

À minha orientadora de estágio, Carmen Morgado, pela paciência e disponibilidade para responder a todas as dúvidas e para me ajudar na elaboração deste relatório.

Por fim, à minha família, namorada e a todos os meus amigos pelo apoio que sempre me têm dado ao longo destes anos.

Resumo

O objectivo deste estágio passava pela migração de uma plataforma (iFlow - www.iflow.pt) de *Business Process Management* (BPM), com processos de *workflow* e gestão documental, disponível em interface web, para tecnologia android. A proposta que me foi feita, no início do estágio, foi que projectasse e desenvolvesse uma aplicação capaz de correr em telemóveis com sistema operativo android, e que chegasse ao final do estágio com uma aplicação que, podendo não estar finalizada em termos gráficos, fosse capaz de cobrir algumas funcionalidades presentes no iFlow. Teria de ser uma aplicação perfeitamente funcional no ponto de vista daquele que é o principal objectivo da plataforma: a consulta e modificação de processos, dando depois o seu devido seguimento.

Para isso, a instituição Infosistema, e em especial, o coordenador Eng. João Costa, comprometeram-se a integrar-me na empresa e a dar-me todo o apoio necessário através de esclarecimentos sobre funcionalidades técnicas da plataforma. Além disso, foram discutidas as direcções a seguir ao longo do desenvolvimento das funcionalidades da aplicação e fornecidos todos os *web services* necessários para o cumprimento do objectivo traçado.

Índice Geral

Índice de Figuras

Índice de Tabelas

1 Introdução

Introdução

Esta secção tem como objectivo fazer uma breve introdução ao estágio e a sua estrutura permaneceu inalterada em relação ao template fornecido para este relatório. A primeira secção descreve o contexto académico em que ocorreu, sendo que na segunda se descreve o contexto científico e tecnológico. Na terceira secção são descritos os objectivos traçados aquando o início do estágio e, por fim, na última secção descreve-se a estrutura deste documento.

1.1 Contexto Académico

Este relatório foi escrito no contexto da disciplina de Estágio Profissionalizante do curso de Licenciatura em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. Trata-se de uma disciplina semestral de 24 créditos, tendo prevista uma carga horária semanal de cerca de 20 horas. Neste estágio a carga horária semanal média foi de aproximadamente 26 horas.

O estágio decorreu durante o período de 2 de Abril de 2012 a 3 de Agosto do mesmo ano e a orientação pedagógica deste projecto esteve a cargo da Professora Carmen Pires Morgado do Departamento de Informática.

O estágio decorreu na instituição *Infosistema - Sistemas Distribuídos S.A.*, situada em Miraflores, concelho de Oeiras. A instituição, usualmente referida como Infosistema, insere-se no grupo das PME (Pequenas e Médias Empresas), e é uma Consultora com serviços de referência em Consultoria, Tecnologia e Outsourcing em Sistemas de Informação.

Figura 1 – Serviços da Infosistema

Os valores da Infosistema são: Criação de Valor e Foco nos Resultados; Integridade, Compentência e Compromisso. Tem, entre outras, parcerias com a Microsoft e a Oracle e visa ajudar as organizações a superar os constantes desafios de negócio, incorporando inovação e talento. Tem como um dos objectivos alcançar um reconhecimento continuado a nível internacional de competência e liderança, antecipando as tendências e necessidades dos seus clientes.

O coordenador de estágio foi o Eng. João Costa, *Manager* da Infosistema, responsável pela àrea de *Technology* desde a sua criação em 2010, tendo eu sido integrado numa equipa de 3 elementos (incluindo o coordenador). Para além do coordenador, também Pedro Ussman, *Senior Consultant Java,* integrado na mesma àrea de *Technology* da Infosistema desde 2011 e Álvaro Gago Martinez, estagiário da Infosistema desde Abril deste ano.

1.2 CONTEXTO TECNOLÓGICO E CIENTÍFICO

A aplicação a desenvolver teria de ser capaz de correr em Android. O Android é um sistema operativo baseado em Linux que pode ser executado em diversos dispositivos, sendo os mais comuns telemóveis, e foi desenvolvido pela Google. Uma vez que é Open Source, todos os dias são lançadas no mercado milhares de aplicações, desenvolvidas por um vasto número de programadores, tornando esta tecnologia líder no mundo dos *smartphones*. Esta aplicação seria uma complemento a ser distribuido juntamente com a plataforma iFlow, descrita mais à frente, acrescentando-lhe valor, tendo em conta o crescimento e a importância da tecnologia android na actualidade. O iFlow permite desenhar e disponibilizar processos de negócio estruturados e suportados na organização, sendo que a aplicação deveria ser capaz de executar uma parte dessas funcionalidades.

1.3 OBJECTIVOS DO ESTÁGIO

A proposta que me foi feita foi o desenvolvimento de uma aplicação que seria uma versão simplificada da interface web da plataforma iFlow. A ideia seria aumentar a mobilidade na intervenção sobre os processos de negócio implementados. Como objectivo principal, no final do estágio, pretende-se que a aplicação esteja totalmente funcional na execução dos passos: consultar um processo, alterar o processo e dar-lhe seguimento. Todos os outros objectivos, como o aspecto gráfico ou a implementação de outras funcionalidades da interface web foram dados como secundários. Num processo, existem

formulários que podem conter elementos diversos e foi-me dito que a aplicação não precisaria de ser capaz de lidar com todos os tipos de elementos, apenas com os mais importantes, ficando assim o gerenciamento de ficheiros como um outro objectivo complementar. Tudo o resto, em relação à aplicação, foi deixado a meu critério tendo eu de decidir, fazendo uso do bom senso, as funcionalidades extra a implementar.

1.4 ESTRUTURA DO DOCUMENTO

Este documento é composto por quatro secções. Na primeira secção, foram mantidas as sub-secções presentes no template diponibilizado, sendo assim feita uma introdução ao estágio, descrevendo o seu contexto académico, tecnológico e científico, os seus objectivos e onde temos uma breve descrição da estrutura do relatório.

A segunda secção descreve as primeiras semanas de estágio e encontra-se dividida em 4 partes: na primeira é apresentada a calendarização do estágio; na segunda é feita a enumeração, com uma breve descrição, das ferramentas de desenvolvimento utilizadas; na terceira parte, que por sua vez tem três sub-secções, é explicado o funcionamento do iFlow assim como as interfaces e funcionalidades disponíveis aos seus utilizadores; e na quarta, introduz-se a tecnologia android, explicando o desenvolvimento de aplicações para esta tecnologia e as suas boas práticas.

Na terceira secção, é descrito o desenvolvimento da aplicação iFlow Mobile assim como as suas várias fases. Encontra-se dividida em 7 sub-secções sendo que as primeiras 6 descrevem funcionalidades da aplicação, enquanto que a última descreve o que foi feito para a finalizar.

A estrutura da secção 4 foi mantida em relação às recomendações, sendo aqui que é feita a apreciação crítica do trabalho realizado, o trabalho futuro e é feita a apreciação global do estágio.

Por fim, na última secção está um anexo, que é um ficheiro XML utilizado como referência base ao longo do estágio.

2 Adaptação à instituição

Adaptação à instituição

Nesta secção, são descritas as duas/três primeiras semanas de trabalho. Numa primeira secção, é apresentada a calendarização do estágio e as alterações que acabaram por acontecer no decorrer do mesmo. Na segunda, são descritas as ferramentas de desenvolvimento que se decidiu utilizar e o nível de API para o qual se decidiu desenvolver a aplicação. Na terceira secção é feita uma descrição da plataforma a migrar (o iFlow) para android: o seu funcionamento, o iFlowEditor e a sua interface web. Nesta secção são descritos os passos importantes na criação de um fluxo com um processo exemplificativo. Na quarta e última secção, apresenta-se a tecnologia android. É descrito, com mais pormenor, como é feito o desenvolvimento de aplicações android, o ambiente de trabalho e as boas práticas sugeridas pela Google para este tipo de desenvolvimento.

2.1 CALENDARIZAÇÃO

Na proposta de estágio apresentada pela instituição aos alunos da cadeira Estágio Profissionalizante, estava presente a calendarização do trabalho previsto para este estágio. (Tabela 2.1)

Semana	Actividades
1	Formação na plataforma Infosistema iFlow (funcional e técnica)
2	Formação e exploração das plataformas de desenvolvimento. Revisão de conhecimentos
3	Instalação dos ambientes de desenvolvimento e deploy (SO, SDKs, SVN, etc)
4-5	Implementação de uma aplicação simples de exemplo. Revisão de conhecimentos
6-10	Construção da aplicação, ainda sem ligação ao serviço da plataforma, com base em especificação funcional a acordar em face do esforço e proficiência do estagiário
11	Revisão e avaliação formal do código realizado pelo coordenador técnico da Infosistema
12-14	Correcção de eventuais erros e implementação das interfaces de ligação aos serviços da plataforma
15-16	Tratamento gráfico e de usabilidade das interfaces de utilizador da

	aplicação. Revisão e controlo de qualidade
17-19	Testes e correcções. Finalização da aplicação. Na semana 18, dará início à elaboração do Relatório de Estágio
20	Revisão formal e análise do código, relatório detalhado sobre lições aprendidas, erros cometidos e aspectos a melhorar pelo coordenador técnico da Infosistema. Conclusão do Relatório de estágio

Tabela 2.1 – Calendarização de estágio

Na prática, e conforme acordado com o coordenador de estágio, tudo acabou por ser algo diferente: apesar do acompanhamento constante dos membros da equipa, acabaram por não acontecer revisões ao código ou revisões de conhecimentos, excepto aquelas feitas por mim mesmo; a primeira semana acabou por ser preenchida pela formação na plataforma iFlow e pela instalação das plataformas de desenvolvimento, tendo as duas semanas seguintes servido para um aprofundar de conhecimentos nas mesmas; as semanas seguintes serviram para o desenvolvimento da aplicação, ainda sem os *web services* necessários, tendo estes sido disponibilizados faseadamente ao longo do estágio; o tratamento gráfico foi algo que foi sendo feito ao longo do desenvolvimento da aplicação; os testes foram feitos no início da penúltima semana de estágio, tendo os erros sido corrigidos nessa mesma semana; e, finalmente, na mesma semana, iniciou-se a produção do relatório de estágio, actividade que se prolongou até ao término do mesmo.

2.2 FERRAMENTAS DE DESENVOLVIMENTO

Numa primeira fase do estágio, foi-me atribuído, como ferramenta de trabalho, um computador portátil com uma instalação do Windows 7 Enterprise e ligação à internet. Assim, na primeira semana, a principal preocupação foi a instalação das ferramentas de desenvolvimento necessárias e a exploração das mesmas. Com o acompanhamento do coordenador de estágio, foi instalado o Eclipse (versão Indigo) para Java como IDE (*Integrated Development Environment*) e foi feito o download do Android SDK (*Software Development Kit*). Foi também necessário instalar o plugin ADT (*Android Development Tools*) para o Eclipse, que permite que rapidamente sejam criados projectos android, fazer debugs, etc.

A todas as versões do android (ex: 1.6, 2.1, 2.3, 3.1, 4.0, etc) são atribuídos nomes (ex: *Cupcake*, *Éclair*, *Honeycomb*, etc), e, nesta fase, foi instalada a plataforma SDK para o Android 4.0 (API 15), conhecido como *Ice Cream Sandwich,* e que a Google declarou como sendo "teóricamente compatível" com

qualquer Smartphone que venha com o Android 2.3 de produção. No entanto, e mais tarde, optou-se por desenvolver a aplicação iFlow–mobile para o Android 2.3.3 (API 10), conhecido como *Gingerbread*, uma vez que, actualmente, é a versão distribuída em cerca de 63% dos aparelhos android([1]). Desta forma, e como as novas versões do android são sempre compatíveis com as aplicações feitas para versões anteriores (e não o contrário), a aplicação desenvolvida no âmbito deste estágio deveria ter compatibilidade para ser utilizada por qualquer aparelho com a versão 2.3.3 ou posterior do android. A Tabela 3 mostra o nível das APIs e os nomes das diferentes versões do android.

API	Versão da plataforma	Nome
1 e 2	Android 1 e 1.1	Base
3	Android 1.5	Cupcake
4	Android 1.6	Donut
5, 6 e 7	Android 2.0, 2.1 e 2.1.x	Eclair
8	Android 2.2.x	Froyo
9 e 10	Android 2.3 e 2.3.x	Gingerbread
11, 12 e 13	Android 3.0.x, 3.1.x e 3.2	Honeycomb
14 e 15	Android 4.0 e 4.0.x	Ice Cream Sandwich
16	Android 4.1 e 4.1.x	Jelly Bean

Tabela 2.2 –Versões do android

Foi-me também fornecida uma versão do iFlow (inclui o editor) que pude instalar e executar em qualquer computador portátil. Esta versão será descrita mais à frente no relatório. Outra das ferramentas utilizadas foi o SoapUI 4.5.0, ferramenta *Open Source* que permite testar os *web services*, e que cria templates de pedidos para um dado web service. Isto permite ver a estutura dos pedidos e das respostas. Alterando os templates dos pedidos com os dados necessários, podemos executar o pedido e observar a respectiva resposta. Esta foi uma ferramenta importante, pois a aplicação iFlow Mobile iria comunicar com um *web service* Axis para a troca de dados. Uma vez que esses dados seriam inseridos numa estrutura XML, também o plugin XML Tools para o editor de texto Notepad++ foi importante na edição, formatação e visualização desses dados.

[1] http://developer.android.com/about/dashboards/index.html

Para a criação de icons, foi utilizado o Android Asset Studio([2]), algo que já vem integrado nos últimos plugins do android para o eclipse. Utilizou-se também o Pixlr([3]), ferramenta necessária para a criação, edição e tratamento dos ficheiros de imagens a utilizar pela aplicação iFlow Mobile.

Por último, o Baretail([4]) foi a escolha para a monotorização dos logs. Com esta ferramenta podemos monotorizar vários logs diferentes e permite uma análise simples dos mesmos através da visualização de diferentes cores para diferentes tipos de mensagens.

2.3 O iFlow

Após a questão das ferramentas de trabalho estar concluída, foi importante começar a perceber o funcionamento da plataforma que se pretendia migrar para tecnologia android. Nesta secção, é feita uma apresentação do iFlow e exemplifica-se com a criação de um processo (bastante simples), descrevendo as suas diferentes fases. Note-se que não serão descritas todas as funcionalidades do iFlow e, nas que serão descritas, evita-se o pormenor, uma vez que, para o cumprimento dos objectivos do estágio, não era necessário esse tipo de detalhe.

O iFlow, tal como referido anteriormente, é uma plataforma de workflow e é acedido, pelos seus utilizadores, através de uma interface web. É uma ferramenta que pretende acelerar os processos de uma instituição, promove a desmaterialização dos mesmos de forma segura e cuja grande vantagem é a sua capacidade de adaptação às necessidades de cada instituição. O Infosistema iFlow, foi desenvolvido a partir de um motor de workflow criado e desenvolvido pela empresa iKnow a partir de 2002 e até 2008. Em 2009, a Infosistema adquiriu as operações da iKnow, incorporando o Infosistema iFlow na sua oferta, tendo sido nesse ano lançada a versão 4.0. Em 2010 foi lançada a versão open source 4.1 e está agora em fase final de desenvolvimento a versão 4.2.

[2] http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html

[3] http://pixlr.com/editor/

[4] http://www.baremetalsoft.com/baretail/

2.3.1 FUNCIONAMENTO

Quando se introduz esta ferramenta numa empresa, o passo mais importante (e também o mais complicado) é a criação dos processos que a instituição necessita. Este passo é feito através do iFlowEditor. Um processo é um conjunto sequencial de acções a realizar com vista a um objectivo comum. No iFlowEditor são utilizados blocos, cada um com a sua função, que são interligados entre si com vista à criação de fluxos. Um fluxo, define-se assim como uma sequência de blocos funcionais organizados para executar as tarefas de um processo. Assim cada bloco corresponde a uma acção e os fluxos irão gerar automaticamente os ecrãs dos utilizadores na interface web. No iFlow, alguns utilizadores podem ser administradores, caso tenham esses privilégios, o que lhes permite criar fluxos como se descreve a seguir. Um administrador cria o fluxo, tal como se apresenta na Figura 2, sendo este um exemplo dos mais básicos.

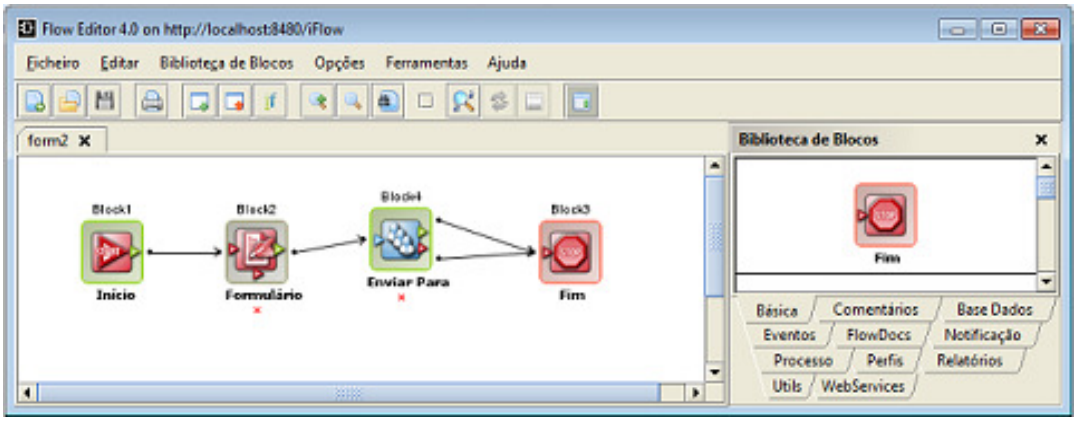

Figura 2 – Fluxo simples

Neste exemplo, com o fluxo descrito, pretende-se a apresentação de um formulário ao utilizador que inicia o processo, sendo depois encaminhado para outros utilizadores.

Quando se guarda um fluxo, o iFlowEditor cria o ficheiro XML correspondente que, quando submetido, fica rapidamente disponível num menu específico para os administradores. Para que fique acessível aos utilizadores, é necessário um último passo: um administrador terá de aceder a esse menu, na interface web, onde pode escolher quais os fluxos que estão online/offline (Figura 3), e terá de alterar o seu estado para online, pois, por defeito, quando criados, os fluxos estão offline.

Figura 3 – Estado dos fluxos

Assim, o fluxo passa a estar disponível aos utilizadores (seleccionados pelo administrador) no menu utilizado para iniciar processos (Secção 2.3.3).

2.3.2 O iFlowEditor

O iFlowEditor é o editor de fluxos do iFLow. Esta é a interface mais complicada de utilizar e requer alguns conhecimentos básicos de programação. Aqui se pretende definir detalhadamente as várias fases de um processo como um conjunto de blocos interligados entre si, através de conectores, no qual os dados do processo circulam e são processados. Este conjunto pode ser visto como uma máquina de estados, onde cada estado é um bloco. Um bloco é a representação de uma funcionalidade que será executada quando o processo passar pelo estado que lhe corresponde. Este é caracterizado por um conjunto de parametrizações, uma funcionalidade e um conjunto de portos (que permitem a conecção com outros blocos). Um bloco tem sempre um porto de entrada, podendo ter vários portos de saída. Geralmente, tem dois portos de saída: porto de saída normal, por onde saí o processo quando a execução do bloco termina como esperada; e um porto de saída de erro, para quando a execução do bloco não é feita correctamente. Existe também o porto de saída de vazia, utilizado em situações que se pretende distinguir como excepções, mas que não são erro. É também importante referir que os blocos são plugins do iFlowEditor, ou seja, podem ser acrescentados blocos, recentemente desenvolvidos, ao iFlowEditor. Hoje em dia, novos blocos vão sendo desenvolvidos e acrescentados conforme as necessidades dos clientes que utilizam o iFlow.

Em seguida, é explicada a criação do fluxo de exemplo na Figura 1. Todo e qualquer fluxo criado no editor começa com o bloco "Início" e termina com o bloco "Fim". Estes são blocos especiais por diferentes razões. Por defeito, o bloco "Início" já lá está, por isso, só temos de colocar o bloco "Fim" arrastando esse bloco da "Biblioteca de Blocos" no menu do lado direito (Figura 3).

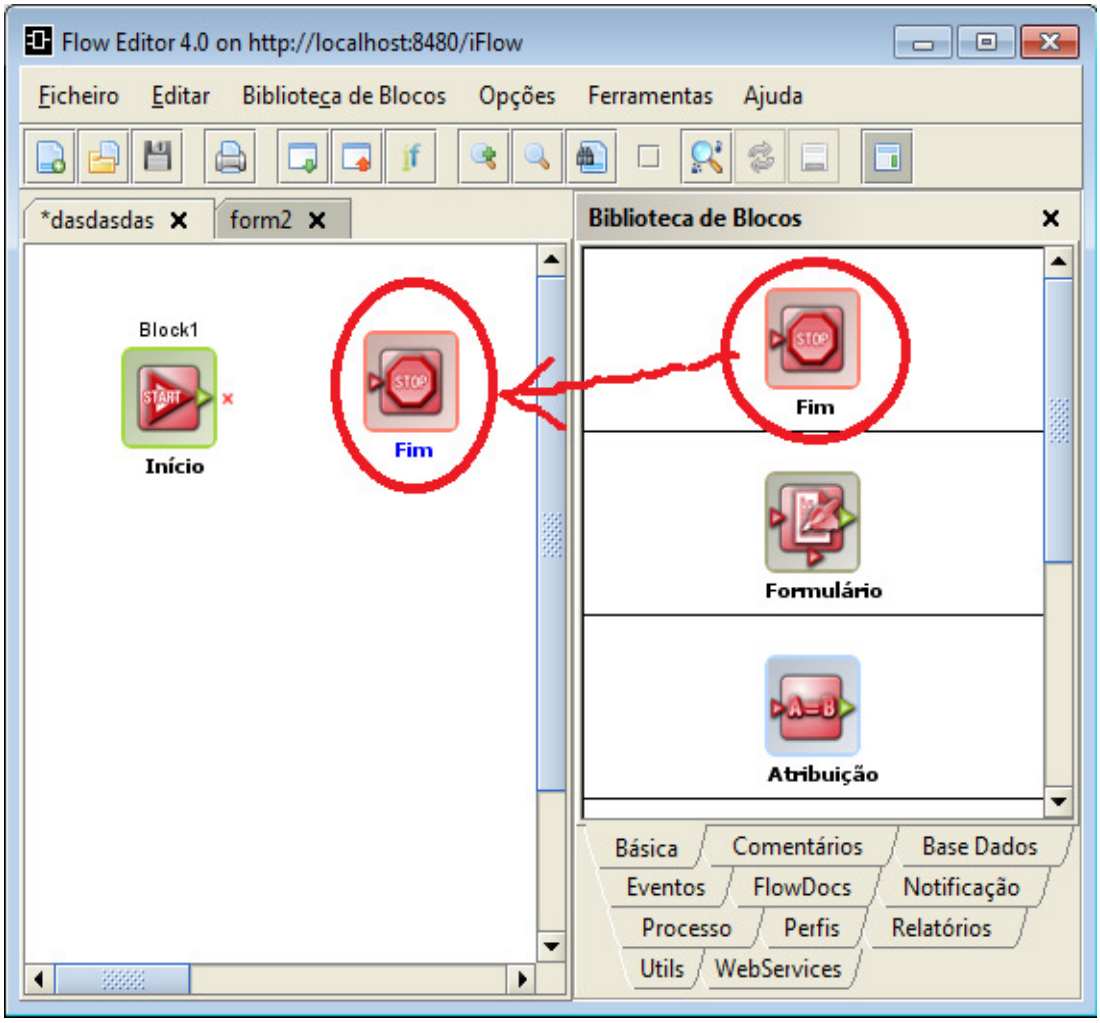

Figura 4 – Arrastar um bloco

Na Biblioteca de Blocos existem separadores de maneira a facilitar a procura dos blocos por tipos. O bloco "Fim" e o bloco "Formulário" são ambos do separador "Básica", sendo que o bloco "Enviar Para" se insere no separador "Perfis". No editor, sempre que fazemos duplo-clique com o rato sobre um bloco, podemos ver os seus atributos. Adicionando o bloco "Formulário" ao fluxo da Figura 4 e editando os seus atributos, podemos adicionar/remover botões ou adicionar/remover campos ao formulário nos botões mais abaixo. Adicionando vários elementos diferentes, ficamos com o seguinte:

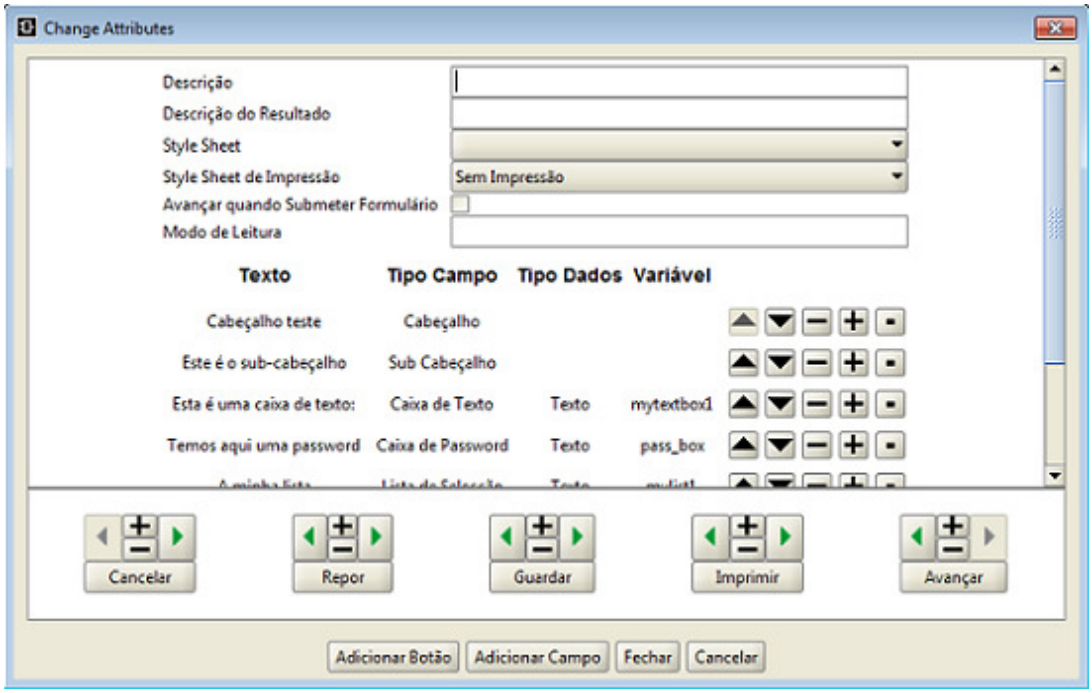

Figura 5 – Atributos do bloco Formulário

De salientar, que aqui foram atribuídos nomes às variáveis nos campos em que essa atribuição era obrigatória. Campos como cabeçalhos, separadores, entre outros, não necessitam variáveis. Depois do formulário feito, é necessário declarar as variáveis. Essa declaração é feita nos atributos do bloco "Início", onde é feita a declaração de todas as variáveis utilizadas pelo fluxo. Existem variáveis que estão lá por defeito e só temos de acrescentar as que pretendemos. Neste fluxo, por exemplo, este é o aspecto dos atributos do bloco "Início":

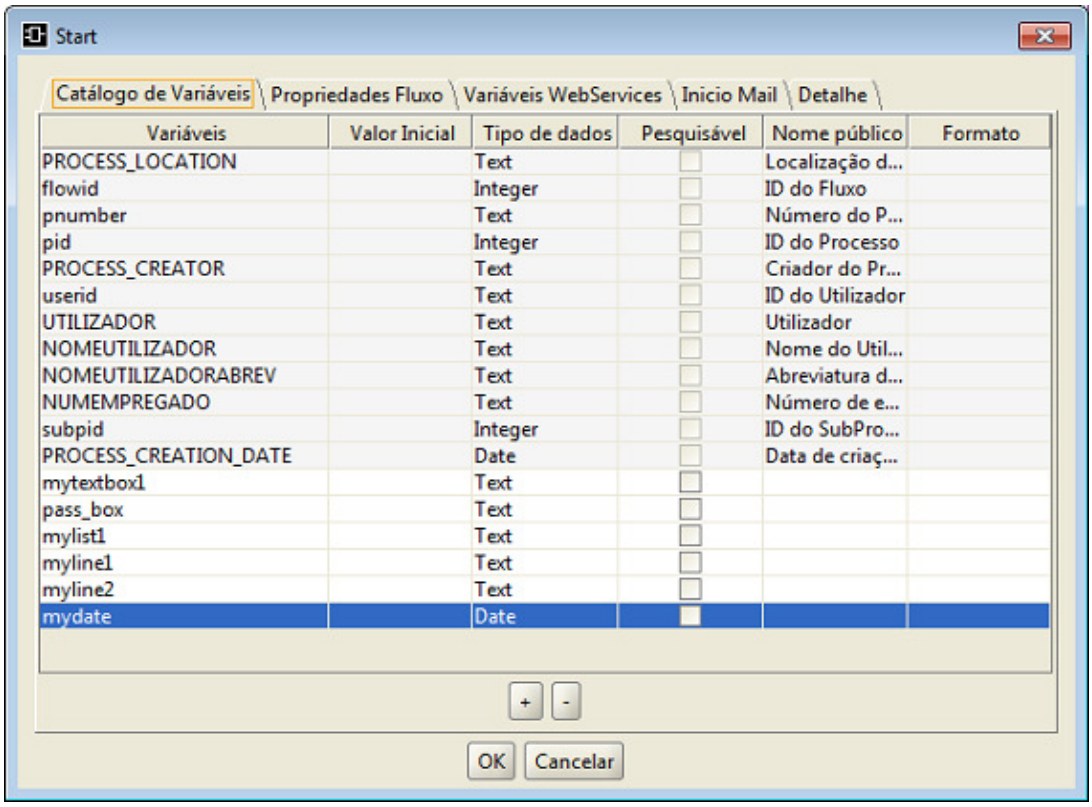

Figura 6– Atributos do bloco Início

Nas linhas brancas temos os atributos que foram acrescentados pelo utilizador, onde foram definidos o seu nome (na coluna "Variáveis"), valor inicial e o seu tipo (na coluna "Tipo de dados"). Para completar o fluxo, define-se também para quem é enviado o formulário, introduzindo o bloco "Enviar para" e editando os seus atributos, podendo ser enviado para um perfil ou um utilizador. Depois de isto feito, é preciso aplicar os conectores. Como se pretende que o fluxo inicie com o formulário e que este seja enviado para um perfil, ligamos os blocos como ilustrado na Figura 1. Com isto feito, basta carregar no botão "Enviar Fluxo" do editor e temos o processo criado e a aguardar que um administrador o coloque online. Na Figura 6, temos alguns dos blocos existentes no iFlowEditor.

Figura 7 – Blocos do iFlow

2.3.3 A INTERFACE WEB

A interface do iFlow é bastante simples de compreender e utilizar, embora tenha alguns pormenores interessantes que não serão descritos. Com mais pormenor, nesta secção, apenas serão descritos os passos que a aplicação android (a desenvolver por mim) deveria ser capaz de fazer.

Sempre que iniciamos o iFlow, esta abre automaticamente uma página de login no *browser* utilizado por defeito. Neste momento, o utilizador introduz as suas credênciais de login (nome de utilizador e password). (Figura 7)

Figura 8 – Página de login iFlow

Logo à partida, quando discutido com o coordenador da instituição, decidiu-se que esta seria uma verificação que teria obrigatoriamente de implementar também na aplicação iFlow Mobile. Em cima, à esquerda, temos várias opções: "iflow.pt", "registe-se", "recuperar senha" e "administrar". Foi decidido que não seria necessário que estas opções estivessem disponíveis na versão mobile, pois, para isso, seria também necessário ter os *web services* correspondentes. Por baixo da caixa de login, temos algum texto com o nome da instituição, bem como a versão do iFlow que está a ser utilizada. Por baixo disso, existe a opção de mudar de idioma e, neste momento, esta plataforma está disponível em 3 línguas distintas: inglês, português e espanhol. Numa primeira fase, será desenvolvida a aplicação apenas em português. Mais tarde, poderia tentar implementar a opção de escolher diferentes idiomas.

Depois de feito o login com os dados correctos, entramos na página principal do iFlow. (Figura 8)

Figura 9 – Página inicial iFlow

Nesta página, temos vários elementos: o logo do iFlow, em cima e à esquerda; uma mensagem de boas-vindas com o nome do utilizador; a data actual, em cima e à direita; o email do utilizador; algumas opções por baixo da data; e os separadores com os seus conteúdos. Os que são realmente importantes no contexto do trabalho do estágio são os separadores: "Painel", "Tarefas" e "Processos". A página do separador "Painel" (que é a inicial) funciona como uma espécie de resumo daquilo que o utilizador tem para fazer. À esquerda pode dar início a um novo processo, estando esses processos separados num menu com estilo *accordion*. A configuração sobre este menu (que processo está em que menu, quais os menus disponíveis, etc) pode ser feita no separador "Admin". À direita, nesta página inicial, o utilizador pode ver as tarefas que estão pendentes, as suas novas mensagens e os seus pedidos de delegações. Para ver os seu conteúdos basta clicar no item, à excepção das mensagens, que são apagadas aquando os cliques. No iFlow, as mensagens funcionam apenas como uma espécie de notificações e entende-se que o utilizador se encontra notificado em relação a uma mensagem quando clica na mesma.

No separador "Tarefas" temos, à direita, a lista das tarefas pendentes e à esquerda um menu de filtragem. Neste menu, podemos filtrar as tarefas pelo fluxo que as originou, pelo número de processo (pid) ou pela data. Na lista de tarefas podemos consultar as tarefas, tal como no separador "Painel". Quando se consulta uma tarefa, aquilo que o utilizador terá de fazer depende sempre do fluxo criado para esse processo e do estado em que se encontra. Ou seja, o utilizador tanto pode ter de preencher um formulário, como dar o processo como concluído, ler uma mensagem, etc, dependendo do que foi definido no fluxo.

Quando o utilizador abre o separador "Processos", tem à esquerda o mesmo menu do separador "Painel" (menu "Iniciar Processo"), podendo iniciar o processo que pretende. Assumindo que um administrador já colocou o processo

do fluxo de exemplo da Figura 1 online e o utilizador inicia esse mesmo processo, temos a página da Figura 9.

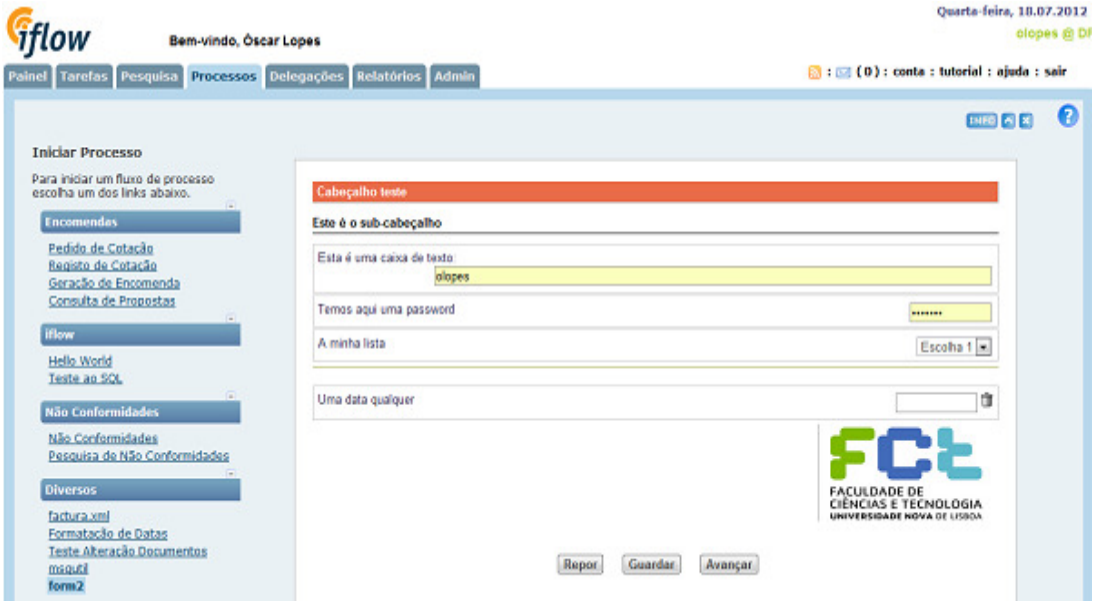

Figura 10 – Iniciar um processo

O utilizador, tal como pretendido, tem então um formulário para preencher com os campos criados no bloco "Formulário" do fluxo e que será reencaminhado ao carregar no botão avançar.

Em relação aos restantes separadores, o separador "Admin" é talvez o mais importante, permitindo alterar permissões de utilizadores, alterar perfis, anular processos, mudar configurações, etc.. No entanto, não será analisado pois não se pretende (pelo menos por enquanto) que um administrador execute essas alterações a partir de um dispositivo android. Os outros separadores (pesquisa, delegações e relatórios) foram definidos como "não-prioritários" à aplicação iFlow Mobile pelo coordenador externo do estágio.

Existem diversas possibilidades de fluxos diferentes no iFlow. Num exemplo concreto, imaginando um fluxo mais complexo em que se pretende criar um relatório sobre uma avaria. Vamos dar o nome de "Verificação de avaria". Um utilizador do Departamento de Vendas observa que há uma avaria num dado equipamento e dá início ao processo. Preenche um formulário com os dados observáveis da avaria e reencaminha para o Departamento de Equipamentos. Aqui, alguém irá receber uma nova tarefa e irá ler os dados do formulário. Depois disto, irá tentar perceber qual a avaria do equipamento e resolvê-la. Quando este tiver feito o seu trabalho, irá voltar a abrir o formulário onde enviará a descrição detalhada do problema e do que foi feito. O utilizador inicial receberá então uma nova tarefa, onde poderá ver a resposta que lhe foi dada e

dar o processo como terminado. E ao fazer isto, é criado um ficheiro em formato PDF que será guardado na base de dados, como uma espécie de relatório de avaria. Este é um exemplo que facilmente poderia ser implementado numa instituição com o iFlow e, o que me foi pedido, no âmbito do estágio, foi que a minha aplicação fosse capaz de executar estes passos do ponto de vista destes dois utilizadores. Deste modo, na aplicação android, o utilizador faria o login com as suas credênciais e consultava as tarefas pendentes. Abria então a tarefa, onde seriam apresentados os dados, alterava esses dados e dava seguimento ao processo, reencaminhando-o. Assim, esta foi, desde sempre, a principal prioridade ao longo do meu trabalho no estágio.

2.4 A TECNOLOGIA ANDROID

Quando se fala de smartphones e tablets, a Google e a Apple são os grandes dominadores deste mercado, e a tecnologia android é considerada por muitos a tecnologia do presente e do futuro. Isto porque, apesar de ambos terem características e pontos fortes diferentes, os aparelhos que utilizam o sistema operativo android são bastante mais baratos e, logo, mais acessíveis ao consumidor comum.

Esta secção destina-se a descrever o ambiente e os princípios do desenvolvimento de aplicações android, porque foi essa a plataforma de desenvolvimento escolhida, bem como, explicar os meus primeiros passos e aplicações experimentais feitas no sentido de melhor conhecer esta tecnologia. No início do estágio, os meus conhecimentos sobre o desenvolvimento de aplicações android eram nulos, mas, por oposição, os conhecimentos adquiridos ao longo do curso ofereciam já uma base sólida sobre programação em linguagem Java e XML, que, na verdade, é o essencial para desenvolver aplicações para este tipo de tecnologia.

2.4.1 AMBIENTE DE DESENVOLVIMENTO

Tal como referido anteriormente, foi utilizado o Eclipse Indigo como IDE, com os devidos plugins e ferramentas instaladas. Com isto, temos a opção de iniciar um novo projecto android, onde basta escolher o nome do projecto e dar um nome a um *package,* que o Eclipse se encarrega de fazer o resto. Uma das pastas criadas pelo Eclipse é a pasta *source* (com o nome "src"), onde está o *package* com o nome que escolhemos, que por sua vez contém uma classe. Esta classe extende a classe *Activity* e, desde logo, é importante perceber o que é uma *activity* (actividade em português) no contexto do android. Uma actividade

funciona como uma espécie de ecrã quando corremos a aplicação. Podemos ter várias actividades a correr ao mesmo tempo, no entanto, a última actividade a ser iniciada é o ecrã que irá estar visível ao utilizador, ou seja, se a aplicação iniciar com uma actividade e nesta houver uma opção que despoleta o começo de outra actividade, então a primeira actividade é parada e a segunda inicia. A Figura 10 mostra o ciclo de vida de uma actividade.

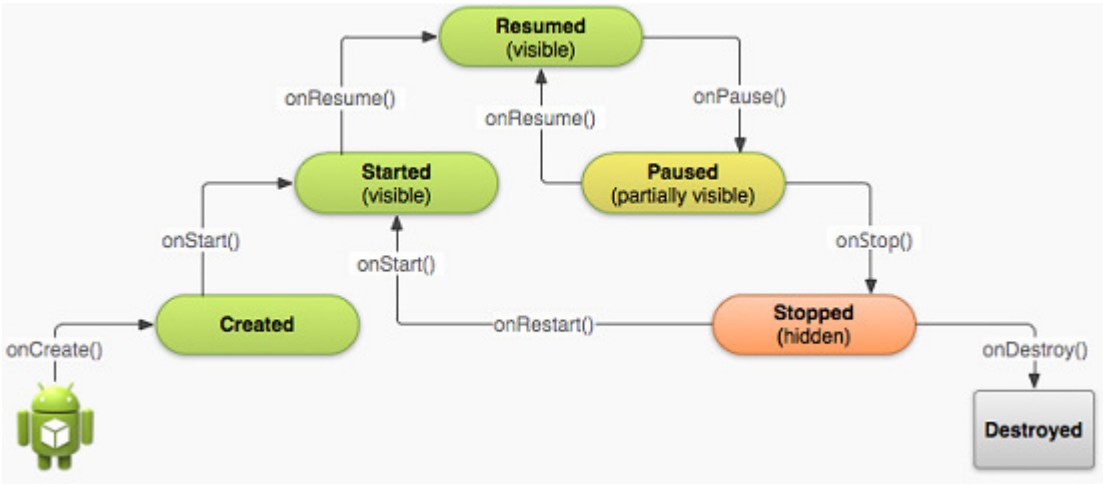

Figura 11 – Ciclo de vida de uma actividade

Neste esquema, que é uma espécie de máquina de estados, podemos observar que métodos são chamados e em que momentos. Assim, podemos observar que, quando começamos uma actividade, o método *onCreate()* é chamado. Este método, em android, funciona como uma espécie de método *main,* este último conhecido por todos os programadores em Java. As actividades são assim classes normais, escritas em linguagem Java, com esta particularidade, sendo que o primeiro código a ser executado está sempre no método *onCreate()*. Uma das primeiras coisas que é preciso definir neste método, é aquilo que o utilizador vai ver no ecrã quando esta actividade inicia. Isto é feito através do método *setContentView(int resLayoutId),* que recebe um inteiro como parâmetro que, por sua vez, é uma referência para um *layout*. Um *layout*, neste contexto, é um ficheiro que contém linguagem XML, ondem se definem exactamente os elementos a apresentar no ecrã. Os *layouts* são sempre colocados numa pasta com o nome "layout", que está dentro da pasta dos *resources* (com o nome "res"). Por defeito, quando se cria um projecto android, é criado também um *layout* com o nome "main.xml", onde existem apenas dois elementos: um *LinearLayout* que contém uma *TextView*. No método *onCreate()* da actividade criada por defeito, está exactamente a linha de código *setContentView(R.layout.main);* que determina que esta actividade, quando criada, vai apresentar no ecrã os elementos presentes no ficheiro .xml chamado "main". A classe R é bastante especial, pois extende a classe Object do Java tendo depois várias *nested classes,* facilitando muito o acesso a todos os

resources. Quando o projecto é criado esta classe é gerada automaticamente, não deve ser modificada e contém constantes que vão ser utilizadas em *build-time* para cada *resource*. A pasta dos *resources* contém várias outras pastas, cada uma com a sua função: várias pastas com o prefixo "drawable" que servem para as imagens nas suas várias resoluções; a pasta "layout" para todos os *layouts*, contendo ficheiros na linguagem XML; e a pasta "values", com ficheiros do mesmo tipo. Para a edição dos elementos presentes nos *layouts*, podemos escrever directamente no ficheiro .xml ou utilizar o *Graphical Layout* (Figura 11) que nos mostra exactamente o que o que utilizador vai ver.

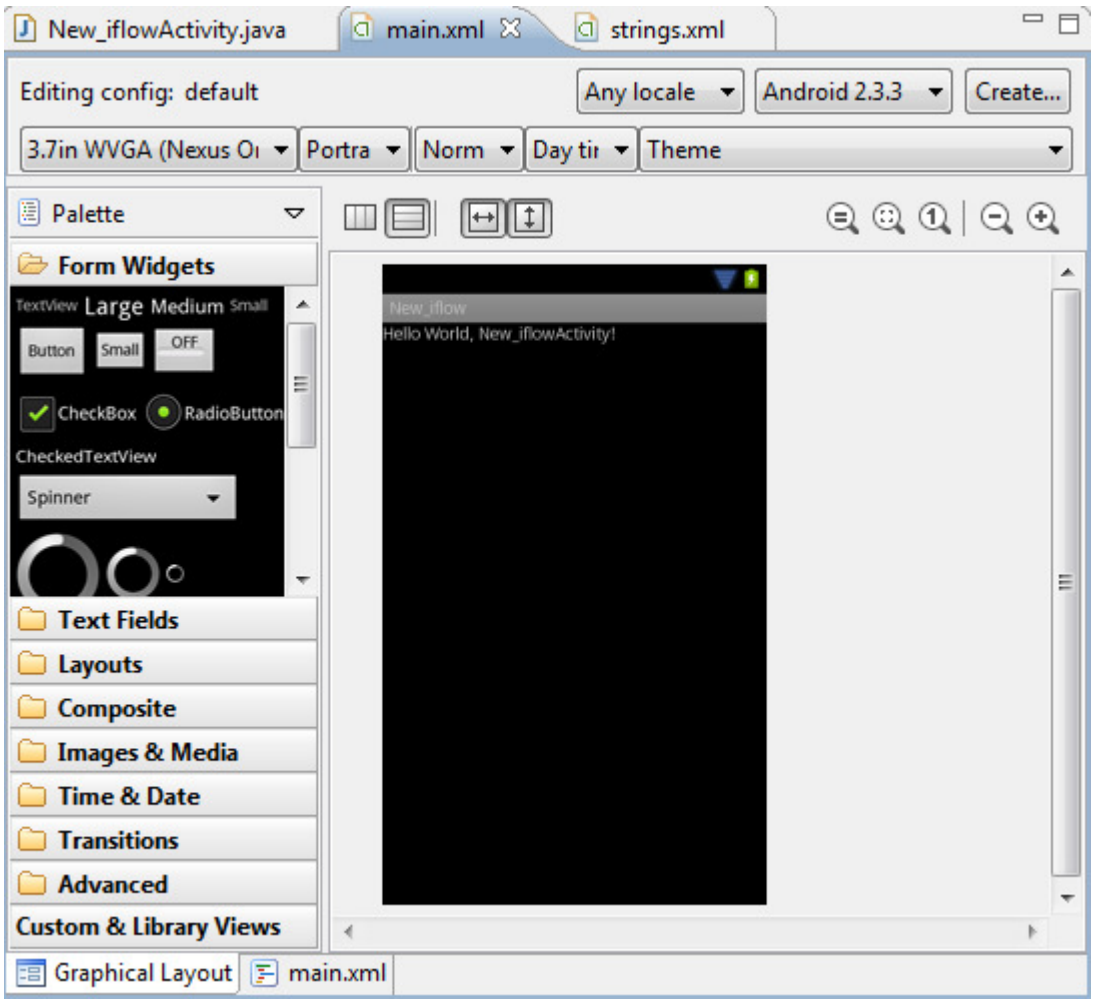

Figura 12 – Graphical Layout

No caso de um novo projecto, a *TextView* apresenta sempre a mensagem "Hello World, " e o nome da actividade, seguido de um ponto de exclamação. Isto aparece porque na *TextView* está definido um atributo da forma *android:text="@string/hello_world"* que é uma referência para uma string que está no ficheiro "strings" da pasta "values". Este ficheiro também é criado automaticamente com duas strings iniciais: "hello_world", já descrita, e "app_name" que será a string com o nome da nossa aplicação. Quando uma aplicação é instalada num dispositivo, este é o nome que será atribuído à

aplicação. Nesta altura, teremos também de compreender o que são views. Uma View, no contexto de android, são exatamente estes elementos a ser apresentados no ecrã, por exemplo, TextView (texto), Button (botões), EditText (caixas de texto), ImageView (imagem), etc. Existem views que, na verdade, só servem para conter outras views, como são os casos de *LinearLayout*, *FrameLayout*, *RelativeLayout* e *TableLayout*. O que mais tarde viria a descobrir, é que é obrigatório, quando se cria uma view, atribuir um valor aos seus atributos *layout_width* e *layout_height,* o que faz sentido, passando este passo a ser a primeira coisa a fazer sempre que crio uma view. Nos exemplos mais à frente, muitas vezes este passo não será mencionado, mas assume-se como feito. O valor destes atributos tanto pode ser um número, existindo várias medidas que podem ser utilizadas (dp, sp, mm, px, pt e in), ou podem ser *match_parent*, *fill_parent* ou *wrap_content*. O *fill_parent* e o *match_parent* fazem exactamente o mesmo, sendo que o *match_parent* apareceu em versões mais recentes do android e servem para que as *views* ocupem o mesmo espaço que a *view* em que estão inseridas. A título de exemplo, se quisermos inserir um botão e quisermos que esse botão ocupe toda a largura do ecrã, mas ocupe o mínimo possível em altura, atribuímos o valor *match_parent* ao atributo *layout_width* e *wrap_content* ao atributo *layout_height.* O botão poderia ser mais pequeno em altura, mas neste caso, é importante usar o *wrap_content* porque normalmente um botão tem um texto e assim garantimos que esse texto irá aparecer por completo. Estes valores só funcionam como queremos se o botão estiver inserido num *LinearLayout* que esteja a ocupar todo o ecrã, pois, se o *LinearLayout* estiver a ocupar apenas metade do ecrã horizontalmente, é exactamente isso que vai acontecer à view *Button*. No *Graphical Layout*, podemos arrastar views do menu à esquerda directamente para o ecrã e, quando fazemos isto, são logo criadas no ficheiro .xml as linhas de código necessárias, ou seja, no caso de uma *TextView*, são automaticamente atribuídos valores aos atributos *layout_width* e *layout_height*, que normalmente são *match_parent* nos dois casos. Estas são as conclusões base de desenvolvimento em android, sendo necessário depois explorar os outros atributos disponíveis para cada view. Pessoalmente, ao longo do estágio, foram poucas as vezes que utilizei o *Graphical Layout*, e preferi sempre escrever eu mesmo os ficheiros .xml, pois, com o atalho de teclado *ctrl+space,* tinha a ajuda do *auto-complete* para escrever os elementos que necessitava.

Para concluir, falta apenas explicar o ficheiro AndroidManifest.xml que se encontra na pasta raiz do projecto. Este ficheiro contém informação essencial que o sistema android necessita para executar o código da aplicação. Entre outras coisas, este ficheiro contém as permissões que a aplicação tem, o nome

do *package* e a versão mínima do SDK que a aplicação vai utilizar, assim como a versão *target.* Também neste ficheiro, temos de declarar todas as actividades que a nossa aplicação vai utilizar. Na raiz deste ficheiro, está o elemento *manifest* que contém um elemento *application.* Este tem normalmente os atributos *icon* e *label*, sendo estas referências para *resources*. É dentro do elemento *application* que colocamos o elemento *activity*, que por sua vez, tem sempre um atributo *name* que é o nome da classe e um elemento *intent-filter* que será abordado mais à frente. Mais uma vez, por defeito, o Eclipse já se encarregou de escrever os dados relativos à actividade que criou com a criação do projecto. Após todos estes elementos do projecto analisados, podemos correr a aplicação. Na primeira vez que se faz isto, é necessário criar um AVD (*Android Virtual Device*), que é, na verdade, um emulador de um aparelho android. Para isso, temos de especificar qual dos SDKs instalados este AVD vai utilizar, dar-lhe um nome e atribuir um valor ao tamanho do cartão de memória (*SD Card*). Com estes passos feitos, podemos ver a nossa aplicação a correr e imaginá-la num aparelho físico (Figura 12).

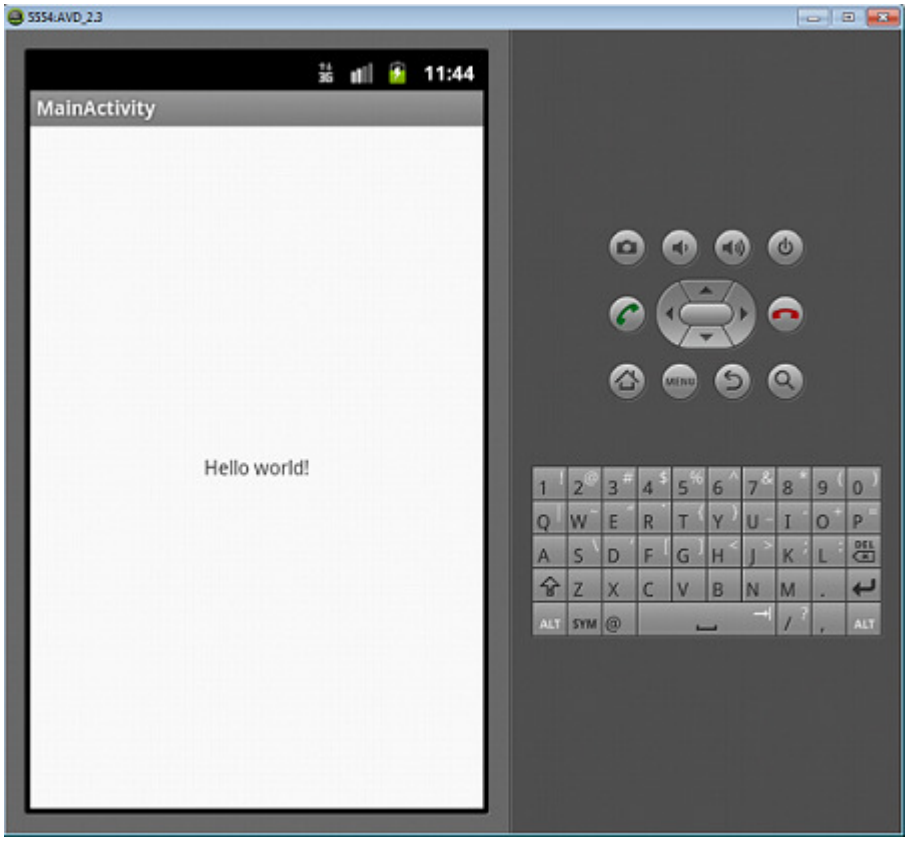

Figura 13 – Hello Android

No caso da Figura 12, é apresentada a aplicação "Hello World" que é criada com um novo projecto em versões mais recentes do eclipse e dos seus plugins para android. Esta versão contém já um tema e alguns ficheiros adicionais, não mencionados anteriormente.

2.4.2 BOAS PRÁTICAS

Ao iniciar este estágio, uma das primeiras coisas que me aconselharam a fazer, foi precisamente informar-me sobre as boas práticas do desenvolvimento de aplicações android. Embora o tenha feito, nem sempre foi possível aplicá-las, por desconhecimento de todos os recursos disponíveis. Ou seja, quando comecei a escrever código, posso ter feito métodos que não foram feitos da melhor maneira possível ou não utilizaram outros métodos que resultariam numa performance melhor, por falta de conhecimento de todas as alternativas para se obter um determinado resultado. Com o passar do tempo e ao ganhar de experiência neste tipo de desenvolvimento, foi possível compreender melhor certos mecânismos e aperfeiçoar a minha metodologia. Apesar de tudo, tentei sempre que o código escrito fosse claro e bem estruturado de modo a ser compreensível para qualquer programador que tenha as bases necessárias. Nesta secção são abordadas as boas práticas para o desenvolvimento em android, sendo a maioria recomendadas pela Google.

Uma coisa que nenhum utilizador gosta que aconteça é que apareça uma caixa de diálogo a dizer que a aplicação não está a responder, perguntando se deseja terminá-la ou esperar (Figura 13).

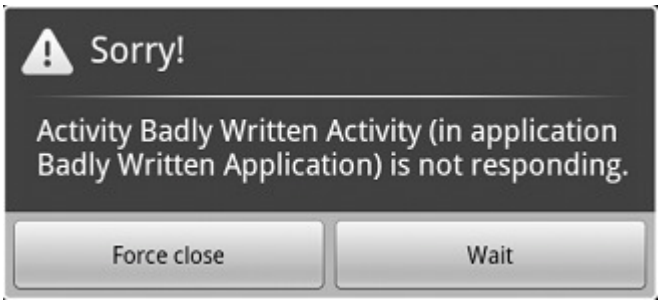

Figura 14 – Erro na aplicação mal escrita

Com isto em mente, existem algumas recomendações para evitar que isso aconteça. Uma aplicação android irá correr num aparelho com memória, poder de computação e bateria bastante limitados, logo, devemos fazer com que cada actividade execute o mínimo de código possível. Deve-se evitar código desnecessário e alocar memória apenas se não puder ser evitado. É extremamente importante que não sejam criados objectos desnecessários e é aconselhado que sejam utilizados métodos *static* e constantes como *static final.* Dever-se-á também informar o utilizador acerca do progresso de uma dada tarefa, pois, o código que está a ser executado não é visível ao utilizador. Como este pode ficar impaciente, se carregar em teclas ou no ecrã, facilmente a indesejada caixa de diálogo aparece. Para evitar isso, utiliza-se normalmente a classe *AsyncTask,* que permite executar uma operação em *background*. A

experiência como utilizador deve sempre ser a prioridade nº1, respeitando as suas expectativas ao navegar ao longo da aplicação. A navegação deve ser intuitiva, e assim, apesar de ser possível, não devemos alterar a função dos botões do aparelho. Por exemplo, o botão *back* deve sempre permitir que o utilizador volte a ecrãs anteriores. Deve-se respeitar as suas preferências e, antes de se transferir informação, é aconselhado perguntar ao utilizador se o deseja fazer. Como o aparelho é limitado, não se deve esperar que o sistema encerre processos. Por isso, quando se inicia uma actividade, deve-se encerrar a anterior, caso não seja precisa. O mesmo se aplica a outros processos. Pelo mesmo motivo, devemos fazer uso das pastas e ficheiros que o sistema tem para strings, imagens, layouts, etc, pois facilita o seu acesso em tempo de execução. Além de tudo isto, deve ter-se em conta que existem diversos diposistivos android, cada um com características diferentes, logo as aplicações desenvolvidas devem ter em conta as diferentes medidas de ecrãs e as suas densidades, tentando suportar a diversidade existente. Este é o resumo das recomendações para este tipo de desenvolvimento, a que se deve acrescentar as recomendações e optimizações para a escrita de código Java.

2.4.3 APLICAÇÕES EXEMPLO

Após os primeiros passos de conhecimento do ambiente de desenvolvimento, foram feitos tutoriais (disponíveis na web) com os quais pudesse ganhar conhecimentos que sabia que iriam ser necessários para desenvolver a aplicação iFlow Mobile. Comecei, então, por uma aplicação muito simples: a aplicação iniciava e aparecia um botão centrado no ecrã que, quando pressionado, fazia iniciar uma nova actividade que utilizava um *layout* diferente, onde aparecia uma imagem a ocupar a totalidade do ecrã. Para alcançar isso, alteramos o conteúdo do *layout* da primeira actividade de maneira a conter apenas um *LinearLayout* com um *Button*. Muito importante no caso dos *LinearLayout,* é colocar o atributo *orientation* com a orientação que queremos dar a esta *view*. Ou seja, se dissermos que a orientação é vertical, todas as *views* dentro do *LinearLayout* vão ficar dispostas verticalmente, uma a seguir à outra, enquanto que se dissermos que é horizontal, as *views* vão ficar lado a lado. Este atributo tem de ser sempre definido numa *view* deste tipo. Aqui introduzimos também o atributo *android:gravity="center_horizontal|center_vertical"*, que coloca todas as views dentro desta centradas no ecrã. Em todos os atributos das views é necessário colocar o prefixo "android:". Na view *Button*, apenas temos de definir o atributo *android:text="@string/start_activity"* e o seu id como *android:id="@+id/myBtn"*. Este último, adiciona um novo id à classe R, ao passo

que, se escrevessemos *android:id="@id/myBtn"* estariamos a fazer referência para um id já existente. Precisamos de atribuir um id a este *Button* para quando quisermos dar-lhe um *listener*, onde vamos especificar o que ele vai fazer. Antes disso, criamos um novo *layout* apenas com um *LinearLayout* e uma *ImageView* cujos *layout_width* e *layout_height* são *match_parent*, assim como no *LinearLayout*. Numa ImageView, temos de especificar qual a imagem que queremos que apareça e fazemos isto com o atributo *src* onde fazemos referência para uma imagem de uma das pastas *drawable*. Assim, temos o seguinte código no *layout* second.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >

   <ImageView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:src="@drawable/pic1" />
</LinearLayout>
```

Isto assumindo que já existe o ficheiro *pic1* numa das pastas drawable. Criamos também uma nova actividade com o nome "SecondActivity" que extende a classe *Activity* e, no método *onCreate()*, escrevemos a linha de código:

```
setContentView(R.layout.second);
```

Depois temos de adicionar esta actividade ao ficheiro AndroidManifest.xml. Aqui, e como queremos que a nossa imagem ocupe o ecrã todo, adicionamos um tema à nossa actividade, ficando com este código:

```
<activity
        android:name=".SecondActivity"
        android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen" >
        <intent-filter>
           <action android:name="com.example.apptest.SECONDACTIVITY" />
           <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
</activity>
```

Por último, falta atribuir um listener ao botão para iniciar esta nova actividade. Para fazer isso, começamos por encontrar o botão pretendido, adicionando na classe MainActivity, no método *onCreate()*, a seguinte linha de código:

```
Button btn = (Button) findViewById(R.id.myBtn);
```

Desta maneira, fazemos o *cast* de View para Button, uma vez que, como já foi explicado, um *Button* também é uma View. E atribuímos-lhe o *listener*:

```
btn.setOnClickListener(new OnClickListener() {
        public void onClick(View arg0) {
                startActivity(new Intent("com.example.apptest.SECONDACTIVITY"));
                finish();
        }
});
```

Neste exemplo vemos uma das utilidades do *intent-filter*: normalmente, às actividades, devemos atribuir-lhes um, e o que foi feito neste caso, foi adicionar uma acção (com um nome) ao *intent* que responsável pelo começo da *SecondActivity*, de maneira que quando criamos um *intent* numa classe, o façamos com uma *string* da acção que inicia a actividade que pretendemos começar. Um *intent*, no contexto android, é uma descrição abstrata de uma operação que queremos executar. Dentro do *intent-filter* temos também de adicionar um elemento *category*, que é, na verdade, o *handler* deste *intent*. Com estes passos feitos, resta acrescentar que, após iniciarmos a actividade, devemos encerrar a anterior com a chamada ao método *finish()* da classe Activity. A Figura 14 mostra o que é visto pelo utilizador quando instala e corre esta aplicação com as suas duas actividades.

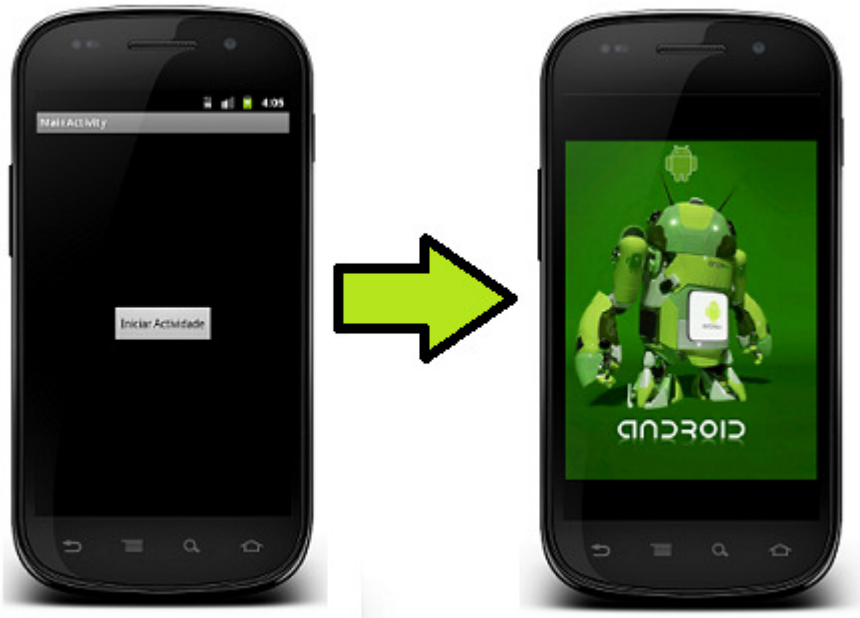

Figura 15 – Primeira aplicação

De notar, a diferença entre o tema normal e o tema *fullscreen,* onde as barras com o nome da aplicação e o estado do telémovel são escondidas. Depois deste exemplo, foram feitas outras experiências no sentido de ficar a conhecer melhor as várias views existentes e os seus atributos. Em particular, estava na altura de começar a pensar como iria implementar em android uma actividade para fazer o login e, depois de isto feito, iniciar uma outra actividade capaz de proporcionar ao utilizador uma experiência de alguma forma parecida com navegação na

interface web do iFlow. Assim, para fazer a actividade do login, sabia que teria de aprender a trabalhar com a *view EditText* e de ser capaz de alterar os seus dados e ler o seu conteúdo. Na verdade, para alterar os dados de uma *EditText*, basta um clique nessa view, esta ganha foco e aparece o teclado android no ecrã para se poder digitar. Para ler os dados, o metodo *getText()* e o método *toString()* encarregam-se de termos o seu conteúdo numa *string*: *aNossaEditTextView.getText().toString()*. Desta forma, para fazer a actividade para o login, necessitava apenas de uma *TextView* com a string "Introduza as suas credênciais", duas *views* do tipo *EditText,* para a introdução do nome de utilizador e da password, e uma *view Button* para fazer o login.

Ao pesquisar e analisar código de algumas aplicações na internet, percebi também que há vários tipos de actividades e uma delas, que iria certamente precisar, é a ListActivity. Uma ListActivity serve precisamente para apesentarmos no ecrã uma lista de elementos de maneira bastante fácil. Para a utilizarmos, em vez de extendermos a classe Activity, extendemos a ListActivity, que por sua vez extende a classe Activity. Esta classe (ListActivity), tem um aspecto pré-definido, no entanto, podemos customizá-lo criando um *layout* com uma ListView e um *layout* para cada elemento da lista. Depois, apenas necessitamos de um adaptador, normalmente uma classe ArrayAdapter<E>, cujo construtor recebe um array com objectos do tipo que pretendemos utilizar, preenchendo assim os elementos da lista com os elementos desse array. É bastante comum ver este tipo de classe definida dentro da classe da actividade onde as queremos utilizar, pois, o normal é que as listas sejam todas diferentes, ou porque recebem objectos diferentes ou porque apresentam elementos diferentes no ecrã. Dentro da classe do adaptador, além de termos de especificar no construtor os elementos que irá utilizar, temos o método *getView(int position, View convertView, ViewGroup parent),* que tem de ser *Override,* onde especificamos o *layout* que iremos utilizar para cada elemento da lista (*row*). Por agora, importa saber que o parâmetro *position* se refere ao índice do elemento na lista e a convertView é a view que vai ser devolvida e que corresponde ao elemento da lista.

Figura 16 – ListActivity

A Figura 15 mostra um exemplo da aplicação de uma ListActivity. Neste exemplo, existe uma classe com o nome Country, uma actividade Main que extende a classe ListActivity e um CountryAdapter que extende a classe ArrayAdapter. Na classe CountryAdapter é aplicado a cada elemento da lista um *layout* com um LinearLayout com orientação horizontal, que por sua vez, contém duas *TextViews* (uma para o nome do país e outra para a sua abreviatura). Depois, para obtermos o resultado da figura, basta ter um layout *main.xml* com uma *ListView*, que utilizamos dentro do método *onCreate()* da actividade Main com a linha de código "*setContentView(R.layout.main);*", criar um novo adaptador e aplicar esse adaptador à nossa ListActivity através da linha "*setListAdapter(adaptador);*". A parte mais complicada deste exemplo, é fazer o *parsing* dos dados de maneira a criarmos objectos do tipo Country e implementar o método *getView* do adaptador onde temos de aplicar o *layout* aos elementos da lista, e alterar o texto das *TextViews* para conterem o nome dos países e suas abreviaturas. Este é um exemplo que demonstra bem algumas das facilidades que as bibliotecas do android oferecem.

Para ajudar qualquer pessoa que queira começar a desenvolver aplicações android, a Google, para além dos conselhos e recomendações que oferece, disponibiliza também centenas de projectos OpenSource, havendo também a possibilidade de iniciar um *Android Sample Project* no eclipse. Estes "Sample Projects" podem ser descarregados da internet (ou, mais facilmente, através do

SDK Manager), o que nos permite ter a acesso a implementações de várias funcionalidades e diferentes tipos de aplicações, podendo verificar o seu funcionamento. Estes projectos são lançados com cada versão, o que permite ficar a conhecer algumas novas classes de cada versão e as suas novas possibilidades.

3 Desenvolvimento da aplicação

Desenvolvimento da aplicação

Esta secção destina-se a explicar o desenvolvimento da aplicação iFlow Mobile, bem como explorar as funcionalidades implementadas. Este foi um trabalho totalmente realizado por mim, sendo que a equipa onde fui inserido, forneceu os *web services* necessários à aplicação e procurou ajudar-me respondendo a dúvidas, indicando como se pretendia que a aplicação funcionasse e o rumo a seguir em cada fase. Esta secção encontra-se assim dividida conforme a várias fases de desenvolvimento, explicando o que foi implementado em cada uma. Ao longo do estágio, nunca houve a obrigação de apesentar com regularidade o trabalho que estava a ser desenvolvido, nem tão pouco houve prazos definidos para cada fase. Houve sim, uma calendarização que acabou por ser completamente flexível, pois, na prática, o desenvolvimento apenas dependeu das dificuldades encontradas e da criação dos *web services* necessários à aplicação. Assim, as reuniões com o coordenador de estágio foram existindo com regularidade, mas dependeram apenas do trabalho realizado, ou seja, cada vez que eram feitos progressos substânciais, o coordenador era informado, discutindo-se depois o que fazer a seguir. Como tal, nem todas as reuniões serão referidas nesta secção, apesar de ter havido um acompanhamento contínuo ao longo do desenvolvimento do iFlow Mobile. No final desta secção encontra-se o estado actual da aplicação, descrevendo-se as funcionalidades implementadas, os testes feitos e as alterações finais.

3.1 TabHost

Tal como referido anteriormente, inicialmente, foi utilizado o SDK para a versão 4.0 do android. Por esta razão, os primeiros testes feitos para implementar os separadores no iFlow Mobile foram feitos com a ajuda a classe ActionBar, algo que teve de ser alterado mais tarde quando se decidiu fazer a aplicação para a versão 2.3.3, pois a classe ActionBar não existia nesta versão. Assim, ao passo que anteriormente tinhamos uma *ActionBar* com *ActionItems* de maneira a simular o comportamento da interface web, tinha agora de o fazer através de um TabHost, existente desde a API 1. O que será descrito nesta secção, será apenas aquilo que foi feito utilizando a classe TabHost. Ao utilizar o *Graphical Layout* para introduzir um *TabHost* num *layout*, ficamos com uma série de linhas de código geradas automáticamente. Dentro do *LinearLayout* habitual, que tem orientação vertical e ocupa todo o ecrã, ficamos com o elemento TabHost, que por sua vez, tem o id "@android:id/tabhost" e também ele ocupa todo o ecrã,

como é susposto. Esta *view* contém duas *views*: um *TabWidget* que contém todas as tabs, com o id "@android:id/tabs" e um *FrameLayout* que serve para mostrar o conteúdo das tabs. O que tem de ser feito de seguida, é criar uma actividade que extende a classe TabActivity, onde vamos criar as nossas tabs. Para isso, é necessário que haja um *layout* para atribuir às tabs e é a partir daqui que as coisas complicam um bocado. O que se pretende é que as tabs contenham uma imagem e por baixo disso o nome da tab. Isto não é mais do que um *LinearLayout*, com orientação vertical, com uma *ImageView* e uma *TextView* e esse é o conteúdo do ficheiro de *layout* para as tabs (tab.xml). A classe LayoutInflater é utilizada para converter o conteúdo de um ficheiro .xml para as views correspondentes. Enquanto que, o método *setContentView* faz isto automaticamente utilizando esta classe, há alturas em que temos de o fazer manualmente, como quando queremos resultados mais complexos do que uma *ListView* simples ou tabs diferentes das que são produzidas por defeito. Como sabia que teria de alterar esse aspecto feio das tabs, comecei a fazer tudo manualmente. Assim, na classe da *TabActivity*, criei um método *createTab* que recebe uma classe, uma string tag, outra string com o nome da tab e um inteiro que é a referência do icon dessa tab para os *resources* e que devolvia um *TabSpec*. Neste método, através do *LayoutInflater*, criamos uma *view* com o *layout* do ficheiro "tab.xml".

```
final View tab = LayoutInflater.from(getTabHost().getContext()).inflate(R.layout.tab,
null);
```

Assumindo que atribuímos um id à *ImageView* e à *TextView* deste ficheiro, no nosso método basta utilizarmos duas vezes o método *findViewById* para ir buscar essas *views* e na *TextView* colocamos a string com o nome da tab, enquanto que na *ImageView* é colocado o icon. Para finalizar, a linha de código:

```
return getTabHost().newTabSpec(tag).setIndicator(view).setContent(intent);
```

faz tudo o que necessitamos. Como esta actividade é uma *TabActivity,* podemos utilizar o método *getTabHost()*, onde criamos o *TabSpec* (que é como um separador), colocamos o seu indicador (o seu aspecto) com a view que criámos com o *LayoutInflater* e o seu conteúdo é o *intent* da actividade que pretendemos iniciar. Para criar este último basta escrever:

```
Intent intent = new Intent().setClass(this, intentClass);
```

onde **this** é o *Context* e intentClass é a classe pretendida. Esta é a altura para se introduzir o conceito de *Context*. Tal como o nome sugere, *Context* é o contexto do estado corrente da aplicação e é utilizado na criação de novos objectos de maneira a que recebam essa informação. Com este método criado, adicionar

uma tab com algumas customizações fica tão simples como escrever as linhas de código no método *onCreate()*:

```
final TabHost tabHost = (TabHost) getTabHost();
tabHost.addTab(createTab(MyTest.class, "tab1", "Tasks", R.drawable.ic_tab_tasks));
```

Era importante, nesta altura, aprender um bocado mais sobre as customizações que poderiam ser feitas num *TabHost*. Assim, coloquei o atributo *gravity* do *LinearLayout* das tabs com o valor "center" de maneira a que a imagem e o texto ficassem centrados. Alterei também o *background* e foram adicionados *paddings* para cada view nesse layout, assim como outras customizações. Após ver um código de exemplo, descobri que era fácil alterar as cores do *background* conforme o estado da tab e o mesmo acontecia para a côr do texto. Para isso, teria de criar uma pasta com o nome "drawable" dentro da pasta "res" onde iria colocar alguns ficheiros .xml para esse efeito: um ficheiro "tab_text_selector.xml" onde teria um elemento *selector* para a côr do texto e outro "tab_bg_selector.xml" onde teria também um elemento *selector* para a côr de fundo das tabs. Além destes dois ficheiros, foram também criados três ficheiros ("tab_selected.xml", "tab_focused.xml" e "tab_normal.xml") na mesma pasta, cada um com um elemento *layer-list* contendo dois itens. Um desses itens é uma *shape* de côr sólida e o outro, que tem margens, contém uma *shape* cuja côr é um *gradient*. Ambas as *shapes* são rectângulos. O modo como isto funciona é o seguinte: no ficheiro de *layout* "tab.xml" colocamos o atributo *background* do *LinearLayout* da raiz com o valor "@layout/tab_background_selector". Neste *selector*, temos:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item
      android:drawable="@drawable/tab_selected"
      android:state_focused="false"
      android:state_pressed="false"
      android:state_selected="true"/>
   <item
      android:drawable="@ drawable /tab_normal"
      android:state_focused="false"
      android:state_pressed="false"
      android:state_selected="false"/>
   <item android:drawable="@ drawable /tab_focused"
      android:state_pressed="true"/>
   <item android:drawable="@ drawable /tab_selected"
      android:state_focused="true"
      android:state_pressed="false"
      android:state_selected="true"/>
</selector>
```

Assim, a côr de fundo das tabs passa a depender do *selector* e este, por sua vez, escolhe um ficheiro .xml para lhes aplicar conforme seu estado. Criando também um *selector* para a côr do texto e fazendo a referência para esse *selector* no atributo *textColor* da *TextView* das tabs, podemos obter o mesmo para o texto.

Neste caso, não é necessário criar os ficheiros com as *layer-lists*, pois, as cores são sólidas e são aplicadas logo no *selector*.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:state_selected="true" android:color="@color/blue"/>
   <item android:state_focused="true" android:color="@color/blue"/>
   <item android:state_pressed="true" android:color="@android:color/white"/>
   <item android:color="@color/dark_blue"/>
</selector>
```

Figura 17 – Estados das tabs

A Figura 16 mostra os diferentes estados das tabs, estando a primeira com o estado seleccionada, a segunda no estado normal e a terceira com o estado *pressed*. Apesar do aspecto feio, estas foram as bases para tentar recriar a navegação da interface web do iFlow na aplicação para android.

3.2 O LOGIN

Após ter feito algumas adaptações ao *TabHost*, comecei a pensar o que seria interessante que a aplicação fizesse quando iniciasse. Entao, e tal como em muitas aplicações do mesmo tipo, a aplicação deveria iniciar com uma imagem *fullscreen* com o logotipo da Infosistema e o nome da aplicação, e, passados cerca de 2 segundos, deveria entrar no ecrã de login. Para implementar este comportamento, decidi criar duas actividades: a primeira com um *layout* muito simples, contendo apenas um *LinearLayout* cujo atributo *background* era uma imagem feita por mim, utilizando o logotipo da instituição e no logotipo do iFlow; e a segunda, para o login, que teria aquelas *views* já descritas, uma *TextView*, duas *EditText* e uma *view Button*. No método *onCreate()* da primeira actividade, a que mais tarde chamei "InicialScreen", iniciava uma *thread* cuja única função era começar uma contagem e no final iniciar a actividade Login e terminar a actividade corrente. Esta actividade manteve-se igual quase até ao final do estágio, altura em que foram acrescentadas algumas verificações à aplicação. A actividade Login, feita tal como se descreveu, ficou assim com o aspecto da Figura 17.

Figura 18 - Login

3.2.1 O PIN

Depois do fim da actividade InicialScreen, teria de se verificar para que actividade a aplicação iria navegar. Aqui foram introduzidos os ficheiros de preferências. Há duas hipóteses quando se utilizam as preferências em android: ou se utiliza o ficheiro de preferências por defeito ou se cria um novo ficheiro com o nome que pretendemos. Eu decidi nesta altura criar um ficheiro "iFlowPrefs" onde inseria um boolean caso já tivesse sido atribuído um pin ao utilizador. Assim, na actividade InicialScreen, verifica-se se o boolean já está nas preferências e se estiver, inicia-se a actividade do Pin, caso contrário, inicia-se a actividade Login seguida da actividade SetPin. As Actividades Pin e SetPin partilharam o mesmo *layout*, contendo um *LinearLayout* com uma *TextView*, uma *GridView* e dois botões: um para voltar atrás e outro para fazer validar o pin. A *TextView* serviria apenas para mostrar os números introduzidos e a *GridView* é apenas uma grelha e, quando se utiliza esta view, temos de criar também um adaptador onde preechemos a grelha com os elementos. Esse adaptador é uma classe que extende a classe BaseAdapter, onde o método *getView* é o mais importante, pois, é onde temos de aplicar um *layout* a cada um dos elementos da grelha, assim como criar os listeners para quando se clica neles.

A actividade SetPin é uma actividade muito especial na aplicação porque é uma actividade com vista a obter um resultado. Ou seja, esta actividade é chamada pela actividade Login da maneira *startActivityForResult("actividadeSetPin"),* e a actividade SetPin, no final, depois do Pin introduzido, retorna esse pin como resultado através do método *setResult("pin", intent)*. Depois disto, a actividade Login, que contém um método *onActivityResult* (que irá executar quando receber o resultado), guarda esse pin nas preferências. De notar que, neste caso, não se terminha a actividade Login e deixa de ser necessário ter o boolean, uma vez que basta verificar se o pin já está nas preferências. Assim, o funcionamento está descrito na Figura 18.

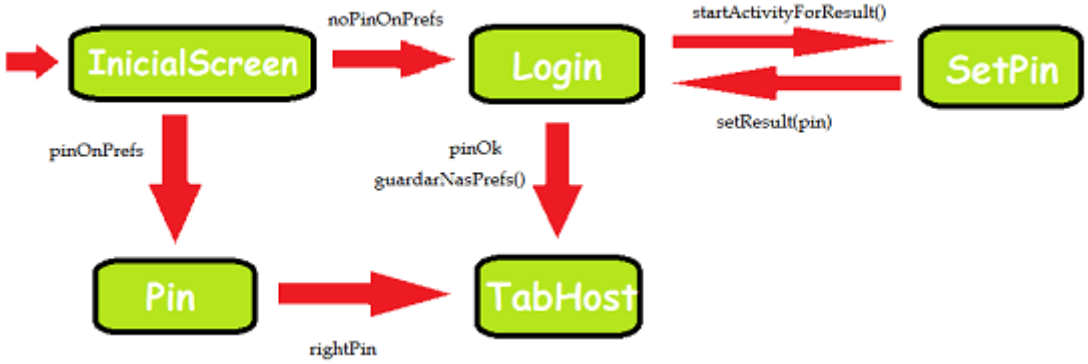

Figura 19 – Actividades de login

A figura não mostra, mas foi introduzida uma característica: quando o utilizador está na actividade Pin e erra o pin 4 vezes, é iniciada a actividade do Login e é apagado o pin guardado nas preferências. Também foi introduzida a opção de mudar de utilizador no menu desta actividade, que faz exactamente o mesmo. Para fazer um menu, é necessário implementar o método *onCreateOptionsMenu*, responsável pela criação do menu, onde lhe atribuimos um *layout* presente na pasta "menu" dos *resources*. Os *layouts* para os menus têm apenas um elemento "menu" contendo vários itens, cada um com os atributos *icon*, *id* e *title*. Os ids são utilizados no método *onOptionsItemSelected* onde colocamos o código a executar conforme o botão "clicado". Assim, foi implementada a forma mais simples da actividade Pin, tendo nesta altura um aspecto muito *default* (Figura 19).

Figura 20 – Actividades Pin e SetPin

3.2.2 ENCRIPTAÇÃO

Foi discutido com a equipa que seria necessário guardar os dados de forma segura no telémovel. Tal como referido anteriormente, teriam de ser guardados o nome de utilizador e a password, o que poderia trazer problemas de segurança. Assim, o que se decidiu fazer foi tentar arranjar algum tipo de encriptação que utilizasse uma string e o pin para encriptar o que precisávamos. Sem entrar em grandes pormenores, esta tarefa acabou por se revelar mais complicada do que aparenta ser. Foi criada uma classe Encryptor, responsável pela encriptação e desencriptação dos dados, que utiliza encriptação AES (Advanced Encription Standard), uma string com os dados, uma string (aleatória) e o pin. Depois de colocar a encriptação a funcionar, houve um problema: não era possível guardar o resultado nas preferências pois os caracteres não eram reconhecidos. Assim, foi necessário codificar os bytes para base 64 (que só contém caracteres reconhecidos pelo android) após a encriptação e descodificá-los antes da desencriptação. A string dos dados ficou na forma *username::password*, pois é pouco provável que tanto o *username*, como a *password* contenham a sequência de caracteres "::". Assim, na desencriptação a string é separada por essa sequência. De salientar, que as preferências têm já alguma segurança de maneira a não serem acessíveis ao "curioso" comum. No entanto, quando alguém consegue aceder ao dados dessas

preferências, é também provável que seja capaz de aceder ao código fonte da aplicação, tornando mais fácil o acesso aos dados encriptados. Por essa razão, o pin nunca é guardado em lado nenhum, servindo como chave para encriptar e desencriptar, guardando-se apenas os dados encriptados. Pelos custos da desencriptação, decidi que, após o login, o nome de utilizador e a password seriam guardados nas preferências como uma espécie de *cookie*, dados que seriam apagados sempre que a aplicação terminasse.

3.3 PARSING DOS DADOS

Por esta altura no estágio, foi-me dito que teria de começar a pensar em como iria fazer o parsing dos ficheiros XML que iria receber do servidor. Comecei por fazer o parsing do ficheiro de um formulário simples (Anexo 1), muito parecido com os dados que iria receber do servidor. Após algumas tentativas e de o ter feito de várias maneiras diferentes, decidi que o melhor seria utilizar bibliotecas para esse efeito. Assim, quando os *web services* me foram disponibilizados, as bibliotecas utilizadas foram a ksoap2-android e a SimpleXML. A biblioteca ksoap2 foi uma importante ajuda na criação dos pedidos HTTP e permitiu também a serialização das respostas de maneira a obter *SoapObjects*. Desta forma, bastava criar as classes dos objectos da resposta e implementar os métodos necessários, que a biblioteca tratava de os serializar. Para isso acontecer, as classes têm de extender a classe KvmSerializable e devem implementar quatro métodos. Em todas as respostas, à excepção de uma, foi utilizada esta biblioteca para a serialização. A única excepção foi um web service que devolvia uma resposta em formato XML, utilizado para a criação dos formulários. Aqui, foi utilizada a biblioteca SimpleXML que, através de anotações, criava também os objectos java correspondentes. Estes parsings foram uma tarefa que foi sendo executada ao longo do estágio, conforme me iam sendo fornecidos novos *web services* para o iFlow Mobile. Foi criada uma classe Client para a criação destes pedidos e, apesar de não se pretender detalhar esta classe, é importante explicar que todos os pedidos são feitos utilizando o nome de utilizador e a password como parâmetros. No entanto, alguns têm também parâmetros adicionais: na consulta de uma tarefa tem de se enviar também o pid, subpid e flowid da mesma, enquanto que, ao mudar um processo, para além de todos estes dados é também necessário enviar uma lista dos elementos a alterar. Quando este último web service foi criado, deparei-me com um problema grave da biblioteca ksoap2: esta biblioteca não permitia enviar nem listas, nem arrays nos parâmetros do pedido. Isto foi um problema que me levou imenso tempo para resolver, mas que acabou por ter uma solução simples o suficiente para não

inviabilizar a utilização da biblioteca. Até à data da entrega deste relatório, foram-me disponibilizados quatro *web services* para o iFlow Mobile:

getUserActivities – responsável por fornecer todas as tarefas de um dado utilizador;

listFlows – responsável por fornecer a lista de todos os processos que o utilizador pode iniciar;

checkFlow – para consulta de uma determinada tarefa;

setProcessData – para alteração dos dados de um formulário.

Ficou também decidido que entretanto se iria criar pelo menos mais um, para ser utilizado no login, uma vez que de momento, o login era feito utilizando o *getUserActivities*, algo que obrigava a receber uma quantidade desnecessária de dados.

3.4 TABS

Nesta secção abordar-se-ão as opções tomadas em relação à navegação dentro da aplicação, ou seja, os seus separadores e o seu conteúdo. À medida que a aplicação foi sendo desenvolvida, entendeu-se que teria de haver pelo menos uma tab para consultar as tarefas, outra para iniciar um processo, e outra para as opções da aplicação, sendo que talvez fosse precisa uma outra tab para as mensagens. Esta última permaneceu quase inalterada no decorrer do estágio, uma vez que, tal como explicado, as mensagens apenas funcionam como uma espécie de notificações e, sendo que não foram criados os *web services* necessários, o conteúdo da tab não chegou a ser alterado. Pessoalmente, pensei também em fazer uma tab Pesquisa de maneira a poder oferecer um comportamento semelhante ao presente no separador Pesquisa da interface web. No entanto, e mais uma vez, como não era prioritário, não me foram fornecidos os *web services* para tal e é uma funcionalidade a implementar no futuro. Foi também pensado que talvez fosse melhor haver uma tab inicial com um resumo das várias tarefas, tal como acontece no iFlow, estando pelo menos agendada a criação de um web service que devolva as últimas dez tarefas do utilizador, havendo ainda uma indefinição em relação ao resto. Tendo isto em conta, na implementação do *TabHost* (Figura 20), decidi atribuir um tamanho às tabs e um *scroll* horizontal, pois, para além de prático para o utilizador, se no futuro forem adicionadas tabs, não será necessário re-implementar o *TabHost*, bastando acrescentar uma linha de código.

Figura 21 – TabHost

Nesta altura, era utilizado o tema *fullscreen*, de maneira a mostrar apenas a aplicação no ecrã, algo que foi alterado mais tarde, passando a mostrar também a barra do estado do telémovel, escondendo apenas a barra do título. Também o título da tab em cima ("As suas tarefas" na figura) foi retirado mais tarde, uma vez que as tabs já continham a informação necessária.

3.4.1 TAREFAS

Tal como mostra a Figura 20, o conteúdo da primeira tab seria então para a consulta de tarefas, estando a actividade implementada por uma *ListActivity*. Enquanto se aguardava o preenchimento da lista, era mostrado um *ProgressDialog*, algo que foi retirado mais recentemente. A cada linha da lista é aplicado um *layout* que contém *LinearLayout* com orientação horizontal. Dentro deste temos uma imagem (icon) e um outro *LinearLayout* com orientação vertical contendo as *TextViews* que mostram os dados da tarefa em cada linha. Foi decidido que os dados a apresentar aqui seriam o id (na figura, "Task X"), o nome do processo ("Assunto da tarefa X") e a data. A introdução do icon "Info" foi uma ideia minha, porque considerei que talvez fosse necessário apresentar mais alguma informação adicional em relação a um processo, aparecendo esta informação numa caixa de diálogo. O utilizador pode ver essa informação e consultar uma tarefa simplemente tocando no ecrã. Nesta fase, foi apresentado o trabalho desenvolvido ao coordenador de estágio, tendo ele ficado agradado

com os rumos que estavam a ser seguidos, uma vez que a maneira como seria apresentada a lista de tarefas tinha sido maioritariamente decida por ele.

3.4.2 INICIAR PROCESSO

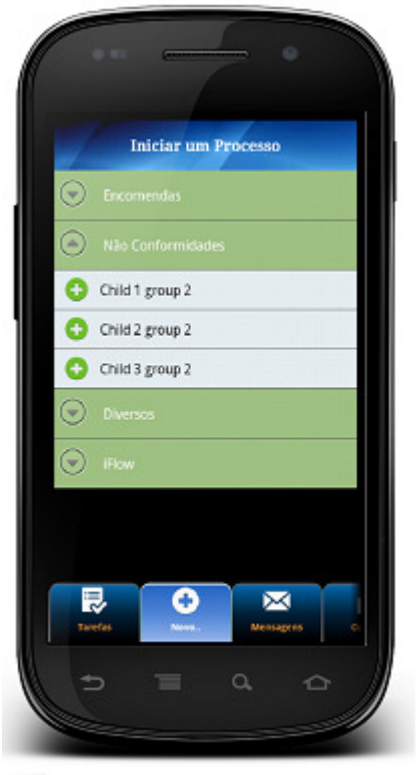

Figura 22 – Tab Iniciar Processo

Na Figura 21, é apresentado o conteúdo da tab "Novo...". A classe desta actividade foi implementada extentendo a classe activity e aplicando um *layout* que, pralém de outras *views*, contém uma *ExpandableListView*, de maneira a recriar o aspecto do menu para iniciar processo na interface web do iFlow. Tal como numa *ListActivity*, foi criado um adaptador para aplicar um *layout* aos elementos dessa *view*, assim como definir o que fazer quando se clica nos mesmos. O *web service* responsável por fornecer os dados para esta actividade só foi completo a algumas semanas do fim do estágio, não havendo, até à data, um web service que permita iniciar um processo a partir de um desses elementos. Isto quer dizer que, por enquanto, na aplicação, quando o utilizador clica num elemento, não é possível iniciar o processo correspondente.

3.4.3 MENSAGENS

Tal como referido anteriormente, o conteúdo da tab Mensagens não foi implementado, tendo sidos experimentados vários layouts diferentes que poderão servir para quando isso acontecer, no futuro.

Figura 23 – Tab Mensagens

Na Figura 22 está o primeiro layout feito para esta tab, assim como um *Toast* que aparece aquando um clique no icon. Os *Toasts* são essas mensagens que aparecem no ecrã para informar o utilizador. Inicialmente, pensou-se em implementar o mesmo comportamento para o icon das tarefas, aparecendo a informação da tarefa, tal como aparece na Figura 22, no entanto, considerou-se ser melhor ter uma caixa de diálogo para o efeito. Este layout foi feito a pensar numa caixa de correio, algo que não é de maneira nenhuma ideal, uma vez que as mensagens do iFlow são um bocado diferentes desse tipo de mensagens.

3.4.4 OPÇÕES

Para a tab Opções, foi pensado criar-se vários itens bem alinhados através de uma GridView em que cada item teria um layout com uma ImageView para o icon e uma TextView para o título. A Figura 23 mostra de onde nasceu a ideia e a sua implementação no iFlow Mobile.

Figura 24 – Tab Opções

Por esta altura, tinha apenas duas opções pensadas: uma para a mudança de utilizador, que deveria ser possível fazer estando dentro da aplicação, e outra para a mudança de temas, dando diferentes cores à aplicação. Mais tarde foram implementadas outras opções.

3.5 CONSULTAR PROCESSO

Esta fase de desenvolvimento foi mais demorada e teve bastantes pormenores interessantes, pois até à altura, todas as *views* tinham sido criadas nos ficheiros .xml de *layouts*, enquanto que agora, teriam de ser criadas dinamicamente dependendo do conteúdo da resposta do *web service*. Depois da implementação do método *checkFlow* (para consulta de uma tarefa) da classe Client e da correcta serialização para objectos java, foi criada uma classe NewForm que extende a classe Activity e onde é aplicado um *layout* que contém um *LinearLayout*, dentro de uma *ScrollView*, sem elementos. Esse *LinearLayout* tem um id que é utilizado na classe para chamar o método *findViewById.* Depois de obtermos essa *view*, podemos adicionar-lhe outras *views* na forma *view.addView(viewXpto)*. Assim, o que teve de ser feito nesta classe foi, para cada campo do formulário, verificar o tipo, criar uma *view* adequada e adicionar essa view ao *LinearLayout*. Não foram implementados todos os tipos de campos existentes no iFlow, apenas os mais importantes: *header*, *subheader*, *textmessage*, *selection*, *textbox*, *arraytable*, *textarea*, *textlabel*, *link*, *separator*, *datecal*, *image* e *file*. Pela maior parte dos nomes se percebe qual é o tipo do campo, à excepção talvez dos campos *selection* que é uma lista de selecção,

arraytable que é uma tabela e *datecal* que é simplesmente uma data que pode ser alterada. O coordenador João Costa falou comigo e decidiu-se que estes eram os tipos base que a aplicação teria de ser capaz de reconhecer e reproduzir. Tudo ficou implementado à excepção do tipo *file*. Nessa reunião com o coordenador, falámos sobre esse tipo e decidimos que neste caso específico, haveria duas formas de introduzir um ficheiro num formulário: carregar o ficheiro a partir do telémovel, em que o utilizador poderia ter algum tipo de navegação nos ficheiros do dispositivo e escolheria o ficheiro a ser carregado; ou carregar o ficheiro via bluetooth. Esta última opção não chegou a ser implementada e foi deixada em "*stand-by*" por não ser uma funcionalidade prioritária. A opção da navegação levou à criação de uma classe própria para o efeito, assim como o tipo *datecal* levou à criação de uma classe CalendarView executada com o tema *Theme.dialog* de maneira a poder-se seleccionar a data pretendida a partir de uma caixa de diálogo. De salientar também que, numa tabela, os tamanhos e os limites variam muito. Este problema foi resolvido colocando uma *ScrollView* horizontal e uma vez que o *LinearLayout* da raiz do *layout* já tem uma *ScrollView* vertical, a tabela pode ser facilmente consultada por completo. A Figura 24 mostra uma consulta de um processo com um formulário muito básico. É aliás, o resultado do parsing e da criação das views necessárias do ficheiro do Anexo 1.

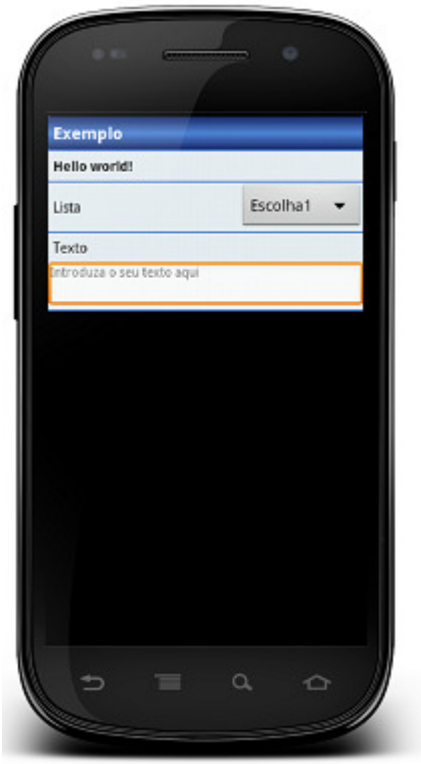

Figura 25 – Consulta de processo

3.6 FUNCIONALIDADES EXTRA

Nesta secção abordam-se todos os extras implementados enquanto aguardava pela disponibilização do último *web service* (*setProcessData* - para alterar os dados de um processo). Nesta fase, foram feitas imensas personalizações até ao ponto em que nenhum dos botões, listas, caixas de diálogo, etc, tem o aspecto do android por defeito, tendo sido tudo personalizado. Presentes nas sub-secções estão os extras que são, na verdade, as possibilidades que o utilizador tem de personalizar a aplicação iFlow Mobile. A primeira sub-secção descreve os ficheiros de preferências guardados e como são manipulados. A segunda sub-secção descreve brevemente a implementação de vários idiomas na aplicação e a terceira a implementação dos temas.

3.6.1 PREFERÊNCIAS

Por esta altura, foi possível disponibilizar uma maior variedade de opções ao utilizador, introduzindo-se aqui as preferências. Uma PreferenceActivity é outro tipo de actividade que facilita imenso o uso de preferências numa aplicação. Sem entrar em muitos detalhes, esta actividade carrega dum ficheiro .xml que tem como elemento base um *PreferenceScreen*, que por sua vez contém preferências, tem um *layout* muito próprio e trata automaticamente de guardar os valores que forem alterados no decorrer da actividade. Para o iFlow Mobile, escolhi utilizar dois ficheiros de preferências. Isto é importante porque há alturas em que é preciso apagar a maior parte das preferências de um dado utilizador, mas pretende-se manter outras. Então decidi manter as preferências que só são apagadas por ordem explícita do utilizador num ficheiro com o nome "iFlowPrefs" e as outras que são apagadas (quando se muda de utilizador, por exemplo) no ficheiro de preferências que o android atribui por defeito a uma aplicação. Uma das verificações importantes, entretanto discutidas com o coordenador, foi que, quando a aplicação iniciava pela primeira vez no dispositivo, ela deveria começar por pedir o URL (*Uniform Resource Locator*) do servidor. Este URL deveria também poder ser alterado na actividade das preferências, algo que acabou por ser implementado.

Figura 26 – Preferências iFlow Mobile

Na actividade das opções foram introduzidos novos itens, entre os quais as preferências. O item Utilizadores abre uma caixa de diálogo que permite mudar de utilizador ou mudar o pin. No caso de se escolher mudar de utilizador, então as preferências (por defeito) são apagadas, voltando aos seus valores por definição. Os itens Option 4 e Sons não foram implementados até à data da entrega do relatório, tendo havido algumas experiências no sentido de introduzir sons na aplicação. Os itens Temas e Idioma serão explicados nas secções seguintes. Nas preferências existem algumas *features* que não foram implementadas: a auto-rotação e notificações. A ideia inicial para a preferência Auto-rotação era poder impedir que o ecrã mudasse a orientação dependendo da posição dos dispositivo, no entanto, quando foram introduzidos outros extras na aplicação, isso deixou de fazer tanto sentido. As notificações seriam para, caso o utilizador recebesse uma nova tarefa enquanto "loggado" na aplicação, receberia uma notificação. Aí poderia clicar na notificação e a actividade utilizada para consultar tarefas seria iniciada com a tarefa que tinha recebido. Existem assim outras preferências:

- **Pasta dos downloads** - onde o utilizador pode escolher a pasta onde serão guardados os ficheiros dos formulários (a implementar quando as funcionalidades dos ficheiros forem introduzidas);

- **Servidor** - onde o utilizador pode alterar o URL do servidor e, ao fazer isto, terá de reiniciar a aplicação;

- **Bluetooth** - uma checkbox que, caso esteja assinalada, permite que o bluetooth se ligue automáticamente em transferências de ficheiros, caso contrário, pergunta ao utilizador se pretende ligar o bluetooth;

- **Login** – activa/desactiva a opção do pin na aplicação. Caso esteja assinalada, irá pedir as credênciais sempre que a aplicação iniciar.

- **Nome** - permite que o utilizador seja cumprimentado nalguns momentos da navegação pelo nome que escolher, ao invés de ser usado o nome de utilizador;

- **Tabs** - permite colocar as tabs em cima ou baixo, no ecrã.

3.6.2 IDIOMAS

Criar uma aplicação com diferentes idiomas é algo bastante fácil de implementar em android. Uma vez que o ficheiro das strings a ser utilizado pela aplicação está dentro da pasta "values" dos *resources*, basta que as strings sejam utilizadas por referência para as strings desse ficheiro e criar várias pastas e vários ficheiros com os diferentes idiomas. Passando a explicar, a aplicação vai utilizar o ficheiro "strings.xml" da pasta values e, para se adicionar a língua inglesa e a espanhola (tal como no iFlow), basta criar duas pastas, cada uma com um ficheiro "strings.xml", com os nomes "values-en" para inglês e "values-es" para o espanhol. Nesses ficheiros estarão as strings na língua correspondente. No caso do iFlow Mobile, selecionando a opção Idioma nas Opções, podemos escolher um desses três idiomas numa caixa de diálogo. Quando o utilizador efectua uma mudança no idioma, a aplicação altera o *Locale* da configuração, indo buscar automaticamente o ficheiro de strings correspondente ao novo idioma. Depois disto, é necessário encerrar a actividade e voltar a iniciar o TabHost de maneira a que as mudanças tenham efeito, algo que é invisível ao utilizador.

Figura 27 – Mudança de idioma

3.6.3 TEMAS

De todas as preferências implementadas, esta foi talvez a mais que deu mais trabalho. No início, foi algo bastante confuso, no entanto tem uma metodologia bastante fácil de aplicar. O que eu, pessoalmente, queria ver implementados eram pelo menos três temas diferentes que pudessem ser utilizados na aplicação. Isto porque, hoje em dia, são muitas vezes estes pormenores que fazem destacar uma aplicação das outras. Os utilizadores dão cada vez mais importância à possibilidade de poderem alterar e personalizar as aplicações conforme o seu gosto. Com isto mente, foram assim implementados mais dois temas para a aplicação: o tema Android e o tema Dark. Isto a juntar ao tema iFlow(tema por defeito) , cujas cores utilizadas são muito parecidas às cores da interface web. Quando comecei a personalizar a aplicação e as suas views, tive a preocupação de tentar que o aspecto fosse parecido ao aspecto do já existente iFlow, embora tenha de ter havido algumas mudanças, pois um formulário visto num browser é muito diferente de um formulário visto num smartphone e as suas cores têm impactos diferentes. Assim, e uma vez que a côr dominante no tema iFlow é o azul, comecei por tentar fazer mais dois temas com cores dominantes verde e vermelho. Assim, a nível de código, foram-lhes dados os nomes Red, Green e Blue. Comecei por criar três ficheiros na pasta "values" dos *resources.* Estes são os ficheiros "attr.xml", "styles.xml" e "themes.xml". A forma como funciona esta implementação é a seguinte: no ficheiro "attr.xml" temos como elemento base "resources" e onde criamos um elemento "attr" com um dado nome e um dado formato na forma `<attr name="dialogBg" format="reference"/>`, definindo assim o atributo "dialogBg" como uma referência. Neste caso, é uma referência utilizada para mudar o fundo de uma caixa de diálogo conforme o tema. No *layout* dessa caixa de diálogo, em vez de ter um atributo `android:background="@xpto/bg"`, existe um atributo *style="?dialogBg"* que é a tal referência para o background. Com isto, é preciso no ficheiro "themes.xml" definir qual o estilo que cada tema vai usar. Ou seja, na aplicação iFlow Mobile, uma vez que no "AndroidManifest.xml" está atribuido o tema `android:theme="@android:style/Theme.Black.NoTitleBar",` no ficheiro "themes.xml" estão os três temas cujo *parent* é esse tema, de maneira a poderem herdar os estilos desse tema, podendo eu personalizar o que quiser. No exemplo do fundo das caixas de diálogo, para o tema Blue tenho um item na forma `<item name="dialogBg">@style/dialogBg_bl</item>`, que é uma referência para um estilo. Desta forma, utilizo a referência attr para explicitar qual o estilo que quero aplicar quando estou a utilizar este tema. E no ficheiro "styles.xml" estará um estilo na forma `<style name="dialogBg_bl"> <item name=`

"android:background"> **@drawable/dialog_bl** </item></style> onde explicito qual o atributo a usar e o recurso *drawable* que lhe quero aplicar. Do mesmo modo, tenho também um estilo, com o nome "dialogBg_rd" para o tema vermelho e "dialogBg_gr" para o tema verde, que utilizam recursos diferentes. Com isto feito, basta no método *onCreate()* do código da classe utilizar o método setTheme(R.style.myTheme_Blue) e automaticamente temos os temas aplicados. Ora, como seria de esperar, a criação destes dois temas veio triplicar a quantidade de alguns recursos, tendo dado muito trabalho a criação de todos os estilos a aplicar às views. No entanto, foi um trabalho que não foi feito por completo uma vez que me foi dito, pelo coordenador do estágio, que a parte gráfica da aplicação seria mais tarde refeita por uma equipa especialista da Infosistema. Depois de algumas experiências, aos olhos do utilizador existem três temas (Dark, Android e iFlow), tendo Android sido o nome dado ao tema Green, iFlow ao Blue e Dark ao Red. Este último é bastante diferente dos outros dois, pois o seu aspecto foi inspirado no tema holo dark do android. Os icons também deveriam mudar conforme o tema, mas isso não foi implementado porque foram utilizados icons bastante básicos para fazer a aplicação. Mais tarde, uma equipa de design irá fazer icons especificamente para o iFlow Mobile, implementando-se a mudança nessa altura. A Figura 27 mostra os três temas pela ordem iFlow, Android e Dark.

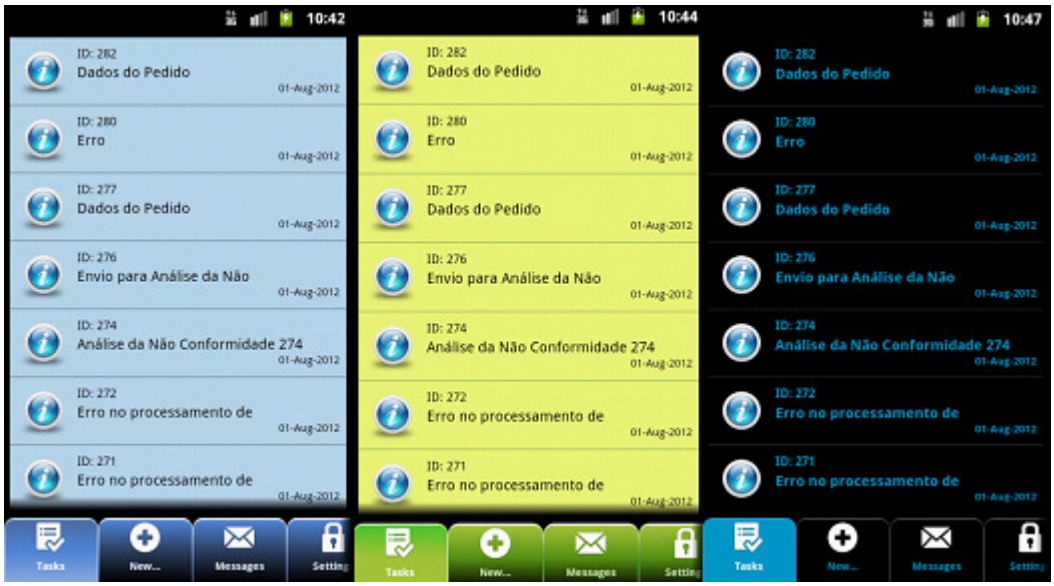

Figura 28 – Temas iFlow Mobile

3.7 O iFlow Mobile

Nesta secção são descritas todas as funcionalidades implementadas actualmente no iFlow Mobile, assim como os testes feitos em dispositivos, as suas consequências e modificações no sentido de melhorar a aplicação.

3.7.1 Testes

A duas semanas do fim do estágio, o iFlow Mobile foi testado pela primeira vez num dispositivo físico. O dispositivo era um Samsung Galaxy Note, que tem um processador Dual Core de 1.4 GHz. Nesta primeira experiência, foi possível observar que o Login da aplicação não era suficientemente rápido, o que é compreensível, pois, foram as primeiras actividades a ser feitas durante o estágio e, na altura, havia ainda uma falta de conhecimento no desenvolvimento deste tipo de tecnologia. Além disso, para efectuar o login, a aplicação teria de receber uma quantidade enorme de dados, algo que também não era ideal. Aqui se decidiu que, mais tarde, seria feito um web service somente para o login. Além disso, foi possível verificar que havia *layouts* que não ficavam com o melhor aspecto (ex: imagens fora do ecrã) quando o dispositivo estava na posição horizontal e que havia botões que não respondiam muito bem aos cliques. Havia também alturas em que era necessário que a aplicação fosse mais rápida do que era nesta fase. Houve funcionalidades (ex: consultar processo, mudar processo, etc) que não foi possível testar, porque a aplicação foi testada utilizando um servidor que não continha ainda alguns dos web services feitos para o iFlow Mobile. Nos momentos em que a aplicação tentava utilizar esses *web services*, a aplicação "rebentava", algo que nunca poderia acontecer.

3.7.2 Alterações finais

Depois da fase de testes, tornou-se necessário fazer algumas alterações de maneira a melhorar a aplicação. Em todas as classes, quando possível, declarou-se as váriaveis como *static final* e os métodos como *static* para melhorar a performance. Especialmente na actividade de login, houve uma melhoria de código de maneira a tornar esta actividade mais rápida. Os botões foram também refeitos, assim como as actividades do Pin e SetPin para que toda a aplicação e os seus botões responda correctamente aos toques no ecrã do dispositivo. Para evitar os erros aquando a mudança de posição do dispositivo, e uma vez que o android oferece essa possibilidade, foram criados *layouts* adicionais para as actividades que ficavam "desformatadas" na posição

horizontal. Por último, e para evitar que a aplicação em caso algum "rebentasse", foram adicionadas verificações que, em caso de erro, fazem aparecer uma caixa de diálogo a dizer que ocorreu um erro e a operação não pode ser concluída.

3.7.3 ESTADO ACTUAL

No presente, o iFlow Mobile, conta com os temas e preferências referidos em secções anteriores, tendo sido refeitos os layouts e estilos da aplicação e adicionadas outras funcionalidades. Nesta secção será demonstrada, através de imagens e comentários, a navegação ao longo da aplicação nos seus diferentes temas e serão explicadas algumas funcionalidades. Será possível distinguir os atributos a que não foram atribuídos estilos e que são comuns a todos os temas.

A aplicação começa com o ecrã inicial em *fullscreen* que dura aproximadamente 2 segundos.

Figura 29 – Navegação, Ecrã inicial

Nesta actividade do ecrã inicial são feitas várias verificações, desde o tema a utilizar, à confirmação que a internet está acessível, entre outras. Caso a internet não esteja acessível, é mostrada uma mensagem indicando esse erro e informando que a aplicação será encerrada. No ecrã login, existe um icon em baixo e à direita que é o icon de ajuda. Quando pressionado mostra uma caixa de diálogo com as instruções a seguir. Este icon existe para todas as actividades, cada um com as suas instruções, antes de se entrar na actividade do TabHost. Caso o utilizador esteja a entrar na aplicação pela primeira vez, aparece-lhe o ecrã da Figura 30.

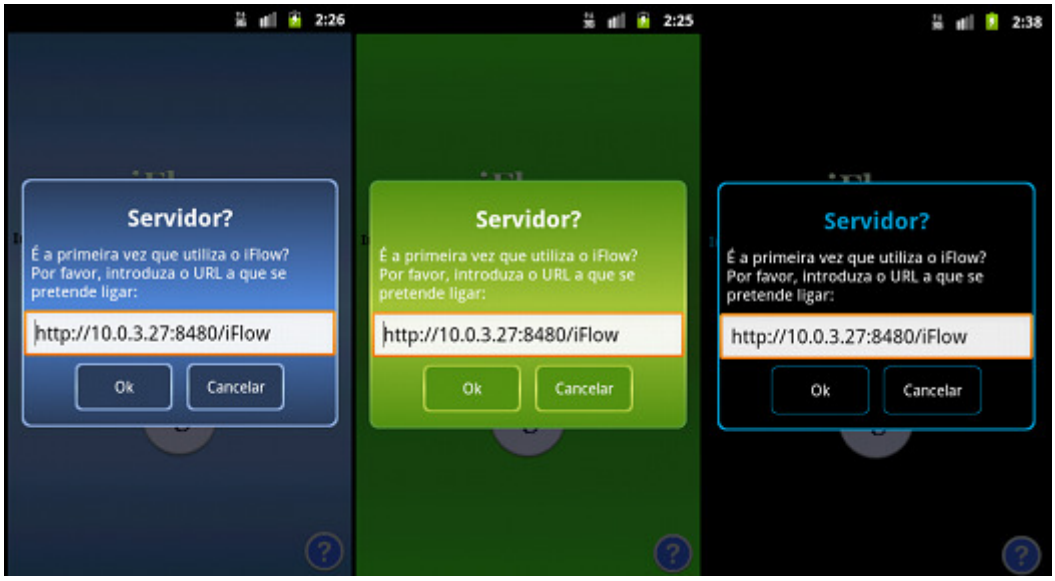

Figura 30 – Navegação, Servidor

Caso contrário, este passo é saltado. Depois do servidor introduzido, inserem-se as credênciais (Figura 31). Esta actividade tem também um menu onde podemos consultar o servidor que está a ser utilizado e mudá-lo caso seja necessário.

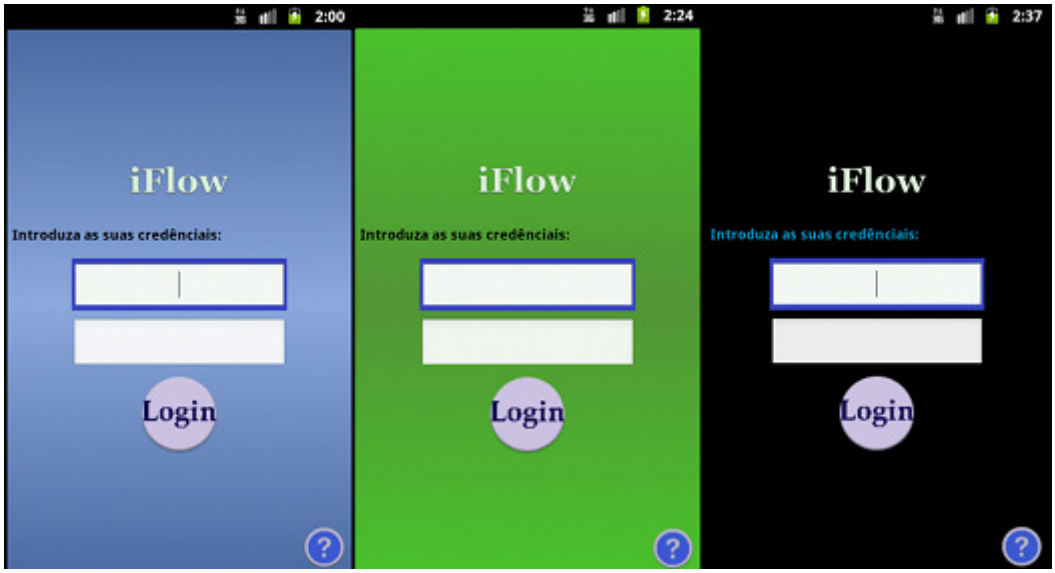

Figura 31 – Navegação, Login

Caso a preferência do pin esteja activada e já tenha sido atríbuido um pin ao utilizador, em vez da actividade do login, temos a actividade do pin (Figura 32). Esta actividade tem o mesmo layout da actividade para atribuir o pin (SetPin), que acontece após a inserção e validação das credênciais.

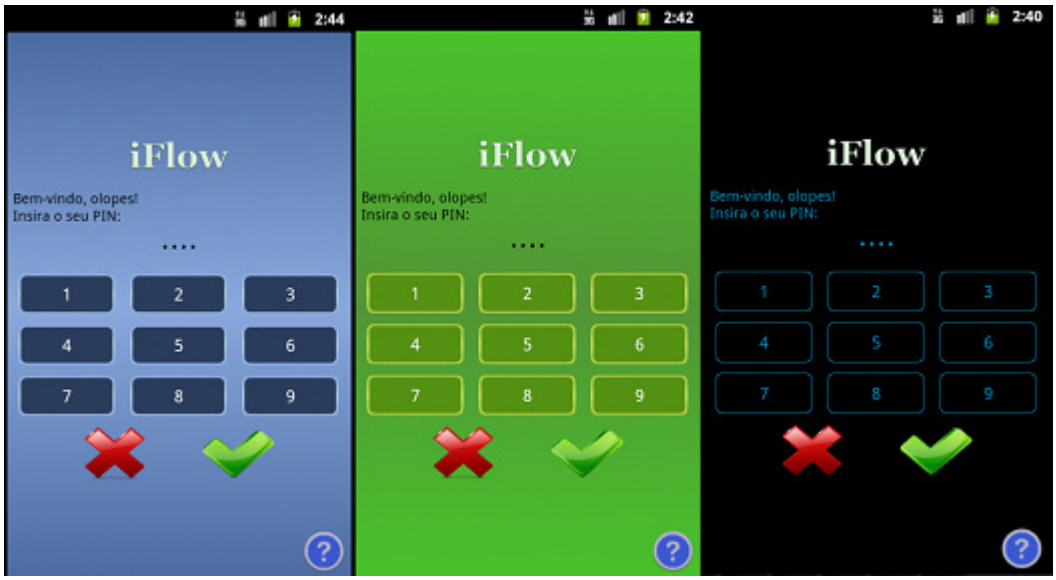

Figura 32 – Navegação, Pin

O nome que aparece na mensagem varia conforme o nome que está na preferência Nome e existe um menu que permite a mudança de utilizador nesta actividade. Depois do pin inserido, e tal como já foi mostrado anteriormente, é então possível consultar as tarefas activas. (Figura 33)

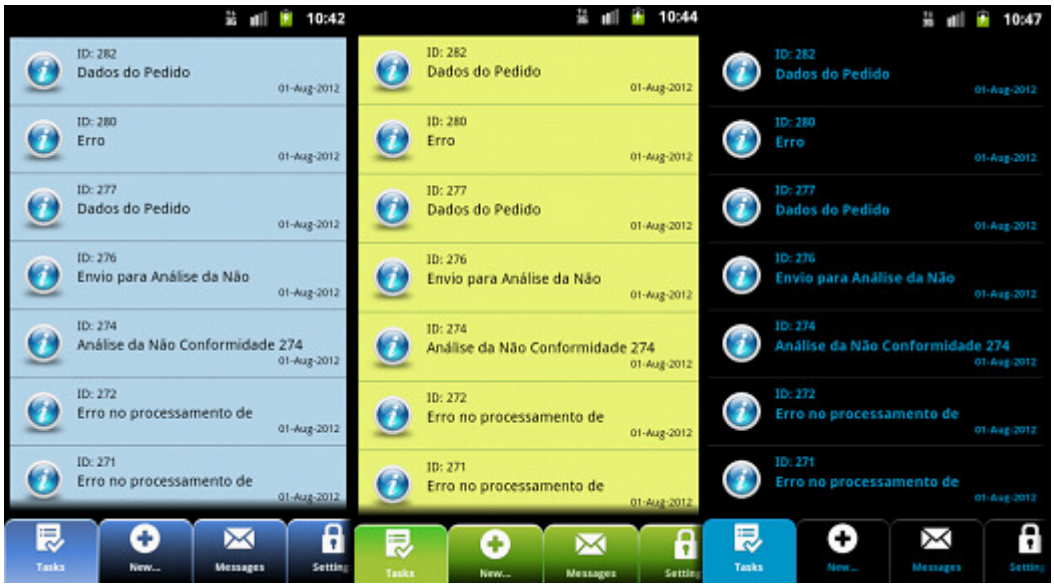

Figura 33 – Navegação, Tarefas

Aqui é apresentado para cada tarefa o seu id, nome e data. Nesta actividade está implementado um filtro de tarefas, no menu, sendo necessário alterar o *web service* de forma a receber uns dados adicionais sobre as tarefas, para que o filtro possa ficar totalmente funcional. No icon das tarefas, podem ser consultados mais dados sobre cada tarefa (Figura 34).

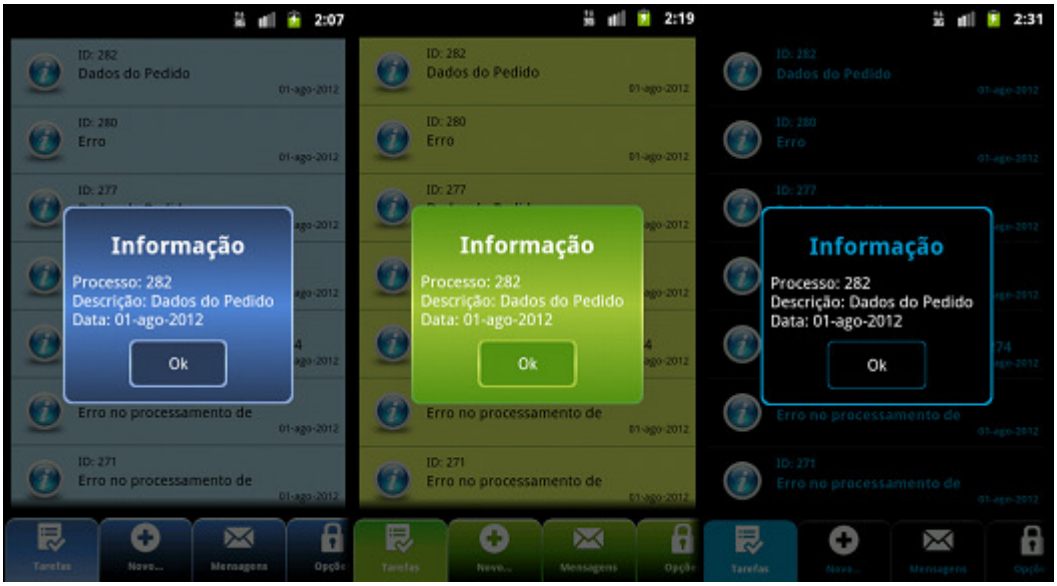

Figura 34 – Navegação, Informação da tarefa

De momento, não mostra nenhum dado adicional, mas o código da aplicação permite que sejam adicionados novos elementos a esta informação de maneira fácil. Ao clicarmos numa tarefa, podem aparecer acontecer várias coisas, mas a mais comum é o preenchimento de um formulário. Na Figura 35, é apresentado um formulário que contém todos os elementos que a aplicação iFlow Mobile suporta neste momento.

Figura 35 – Navegação, Formulário

Este formulário contém uma *view EditText* com uma data que é um elemento do formulário que pode ser alterado. Caso, se clique nessa caixa de texto, aparece o calendário (Figura 36) onde o utilizador pode escolher a nova data, ficando esta com o mesmo formato da data que é substituída.

Figura 36 – Navegação, Calendário

No formulário, está também um elemento do tipo ficheiro onde é possivel inserir um ficheiro. Quando o utilizador carrega na caixa de texto correspondente, é-lhe perguntada qual a fonte do ficheiro que pretende inserir. (Figura 37)

Figura 37 – Navegação, Fonte do ficheiro

Neste momento, a parte do bluetooth ainda não está implementada, enquanto que as outras duas opções permitem navegar no sistema de ficheiros quer da raiz do dispositivo, quer do cartão SD, caso exista (Figura 38). Caso contrário, mostra uma mensagem a dizer que não foi detectado nenhum cartão. Esta é uma verificação que está também implementada para o bluetooth, mostrando uma mensagem de erro no caso do dispositivo não suportar bluetooth.

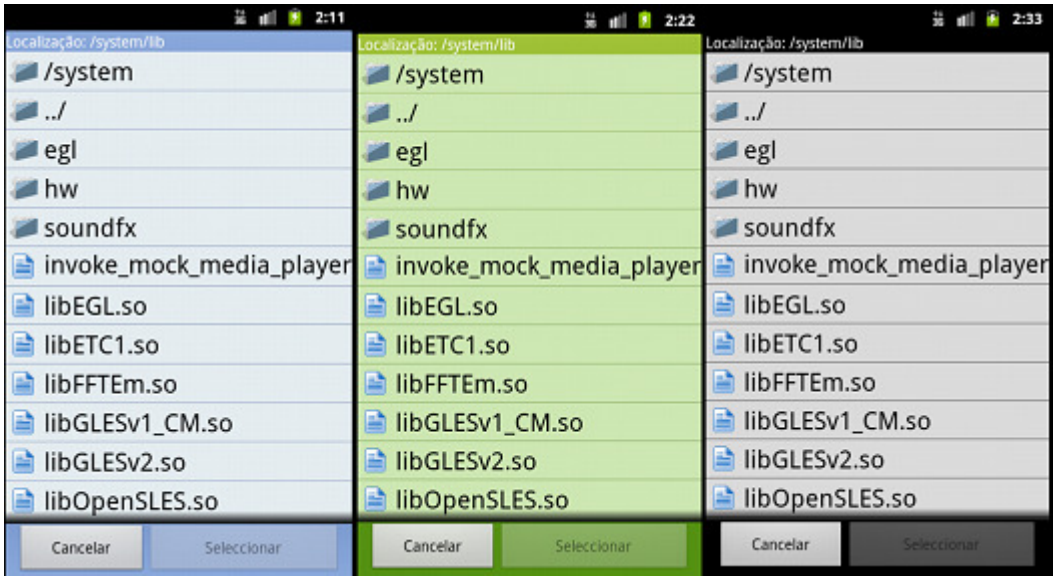

Figura 38 – Navegação, Seleccionar ficheiro

Quando se selecciona um ficheiro, o seu path (caminho) é inserido na caixa de texto, estando ainda por discutir como se irá fazer o *upload* do ficheiro para o formulário. Para concluir a parte dos formulários, já é possível alterar os dados de um formulário, não havendo ainda os *web services* necessários para lhe dar seguimento. A figura seguinte, mostra o aspecto da actividade responsável por iniciar um processo a partir de um fluxo:

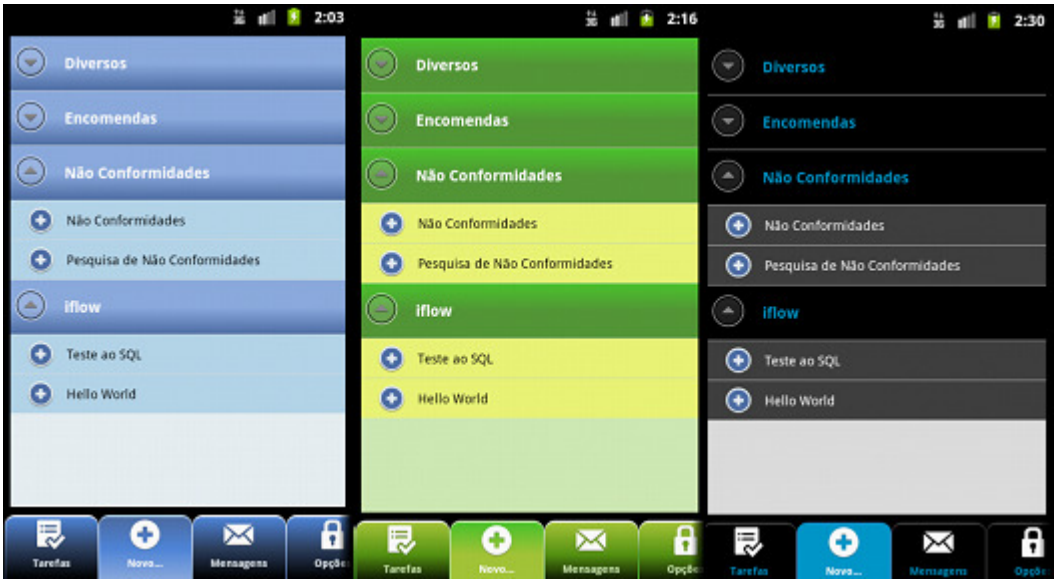

Figura 39 – Navegação, Iniciar processo

Mais uma vez, apenas foi criado o *web service* que retorna os dados necessários para preencher os elementos deste menu, faltando um outro para iniciar o processo a partir daqui.

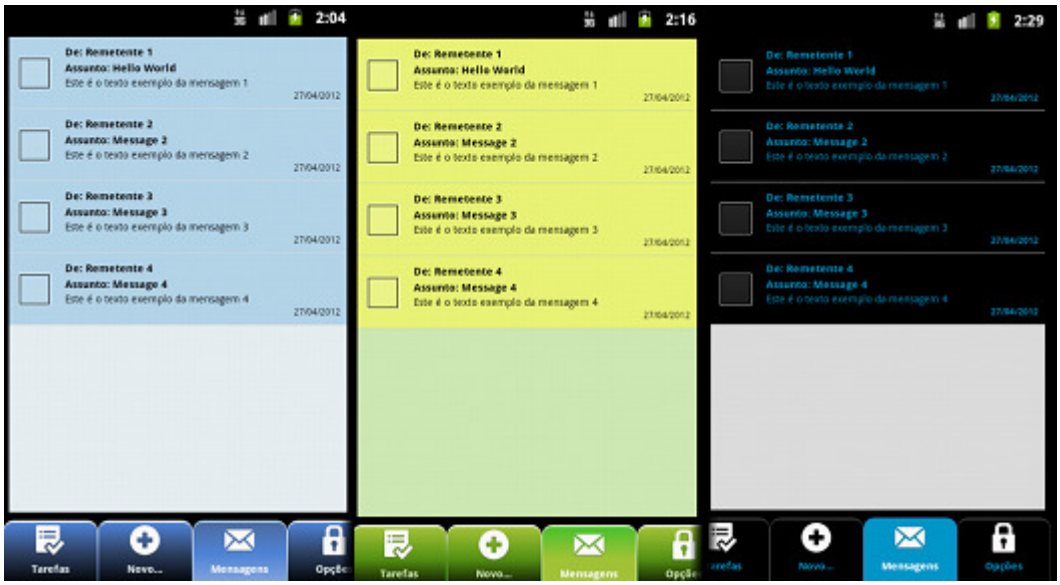

Figura 40 – Navegação, Mensagens

Tal como explicado em secções anteriores, esta actividade ainda não foi refeita, pois são necessários os *web services* que forneçam os dados das mensagens e que as marquem como lidas. Na última tab, temos as tais opções disponíveis ao utilizador (Figura 41)

Figura 41 - Navegação, Opções

Aqui, o botão Preferências navega para a actividade apresentada na secção 3.6.1, sendo que essas preferências se mantêm inalteradas. Esta actividade foi bem explicada na secção 3.6, não havendo até ao momento alterações relevantes. As caixas de diálogo da aplicação são feitas a partir de uma classe produzida para esse efeito. Essa classe permite adicionar à caixa: um título; uma mensagem, que pode ser um texto simples ou uma lista de elementos, como são no caso de se clicar em Idiomas ou em Temas (Figura 42); e botões.

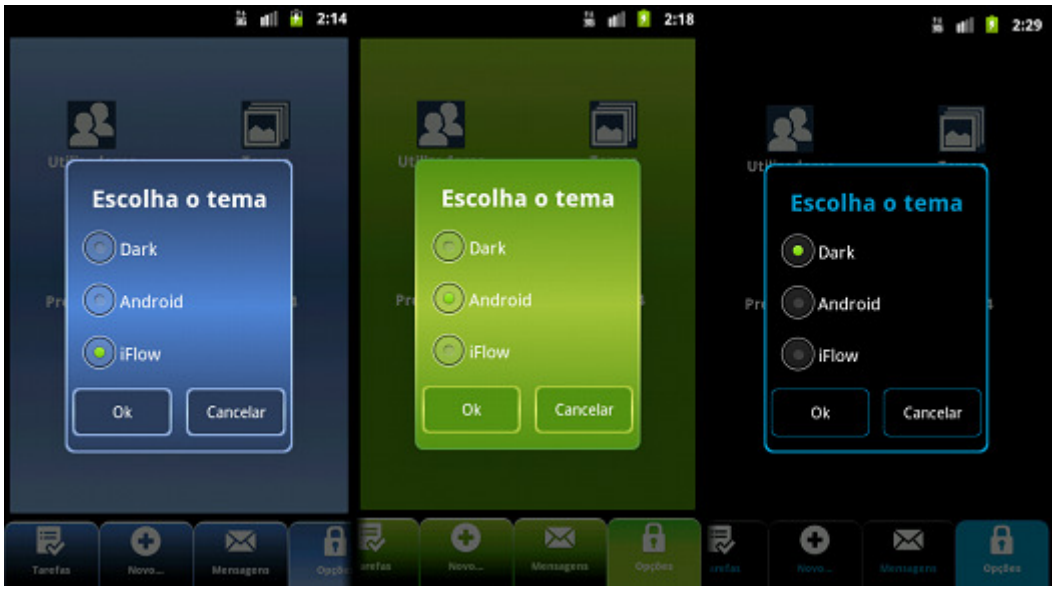

Figura 42 – Navegação, Temas

Quando discutido com o coordenador de estágio, decidiu-se que não deveria ser possível abandonar a aplicação a não ser que o utilizador explicitasse essa vontade. Isto porque, segundo ele, "é muito chato quando uma pessoa que já entrou numa aplicação, está a ver qualquer coisa, sem querer carrega num botão e sai da aplicação. Depois tem de voltar a fazer o login e voltar ao sítio onde estava." Assim, foi implementado que quando o botão *back* ("Voltar") do dispositivo é premido, aparece uma caixa de diálogo que pergunta se realmente quer sair da aplicação. (Figura 43)

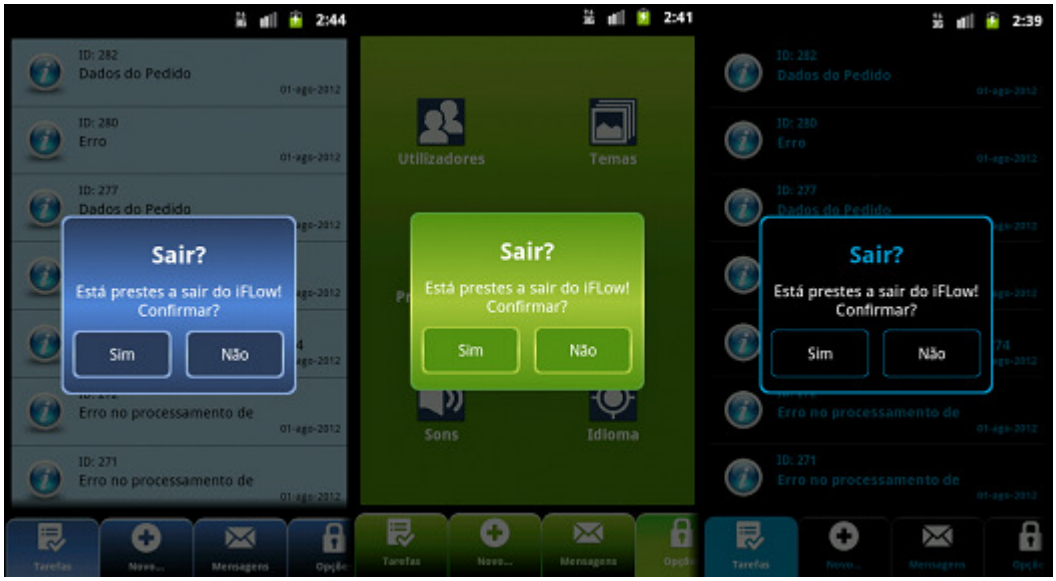

Figura 43 – Navegação, Sair da aplicação

Existem outras *features* da aplicação que estão implementadas em parte, mas não foram referidas. Poderiam ter sido feitas mais personalizações, no entanto, existe ainda alguma indefinição sobre o que se irá fazer a seguir para a

aplicação. De momento, praticamente todas as cores utilizadas são cores sólidas ou gradientes, sendo uma incógnita se no futuro não se irão utilizar ficheiros de imagens como fundos. Nesse caso, estar neste momento a criar mais ficheiros de *layouts* seria completamente desnecessário, pois a utilização de imagens é bastante mais simples, mas tem a desvantagem de aumentar o espaço necessário no disco para a aplicação. Neste momento, e como são utilizadas muito poucas imagens, a aplicação ocupa apenas 1.43MB, um valor bastante baixo para uma aplicação empresarial. Existe também uma classe criada para facilitar a utilização de sons na aplicação. Apesar de não estar a ser usada no momento, torna bastante simples a introdução de um som num dado momento numa actividade.

4 Conclusões

Conclusões

Nesta secção foi mantida a estrutura original do template disponibilizado. Assim, numa primeira secção é feita uma apreciação crítica do trabalho desenvolvido. Na segunda secção, são abordados os planos para trabalhos futuros do aluno na instituição onde foi realizado o estágio e na ultima secção é feita uma apreciação global desse mesmo estágio.

4.1 Apreciação Crítica do Trabalho Desenvolvido

Foi possível lidar com diversas dificuldades no decorrer do trabalho desenvolvido ao longo do estágio: apesar de haver já algumas bases sólidas de programação em linguagem java, não havia previamente qualquer tipo de conhecimentos acerca do desenvolvimento de aplicações para *smartphones*. No local onde foi realizado não havia também nenhum colega com quem fosse possível tirar dúvidas acerca da tecnologia android, no entanto, tal não foi necessário e as dificuldades encontradas foram ultrapassadas com o devido esforço e empenho. No final desta etapa, é com alguma relutância que não posso continuar a trabalhar na melhoria da aplicação desenvolvida até ao seu lançamento, algo que se compreende, uma vez que nunca foi o objectivo de estágio. Tal como referido anteriormente, existem funcionalidades da aplicação que não estão completamente implementadas e outras que gostaria de implementar, porém, a boa estrutura permite que isso seja alcançado de forma relativamente fácil. Para além dos principais objectivos terem sido alcançados, foi-me também possível implementar vários extras na aplicação, algo que me deixa bastante satisfeito com o trabalho que realizei.

4.2 Trabalho Futuro

O trabalho realizado no âmbito deste estágio permite que, no futuro, a aplicação seja distribuída juntamente com a plataforma iFlow. Assim, serão feitas melhorias no aspecto gráfico e implementadas as funcionalidades necessárias para que a aplicação iFlow-mobile seja uma aplicação fácil de utilizar e consiga o propósito da consulta e agilização de processos. O facto de se passar a poder fazê-lo a partir de um *smartphone* será mais uma forma do iFlow acompanhar a evolução tecnológica.

4.3 Apreciação do Estágio

Em relação ao estágio que tive o prazer de realizar na Infosistema, apenas tenho coisas boas a dizer. Tudo foi excelente, desde o ambiente de trabalho,

disponibilidade e profissionalismo dos colegas de trabalho ao resultado final do projecto, bem como a flexibilidade e tolerância demonstrados pelo coordenador João Costa. Fui acolhido na empresa com grande simpatia por todos e não senti, em altura alguma, qualquer tipo de pressão em relação ao trabalho realizado, tendo-me sempre sido demonstrada satisfação pelo mesmo. Tudo isso foram aspectos positivos que ajudaram à motivação e empenho em realizar um bom trabalho, sendo que tive desde sempre, como objectivo pessoal, chegar ao final do estágio com a garantia de satisfação por parte da instituição. Para além de o ter conseguido, foi também muito bom verificar que eu próprio me sinto satisfeito pelo trabalho que realizei. O trabalho realizado ao longo do estágio foi exactamente aquilo que foi proposto pela empresa aos alunos da faculdade, algo que sei que, muitas vezes, não acontece. Chego assim ao término do mesmo com novas competências para realizar desenvolvimento de aplicações para a tecnologia android, a tecnologia "da moda", algo que me abre as portas para muitas empresas pelo seu enorme crescimento nos últimos anos. Para concluir, foi também uma agradável surpresa observar que as bases que o curso de Licenciatura Informática proporcionou são mais do que suficientes para uma fácil adaptação e novas aprendizagens no âmbito de uma instituição.

5 Anexos

5.1 Anexo 1 – Ficheiro XML de um processo

Este ficheiro, tal como referido anteriormente, foi o primeiro ficheiro do qual tive de fazer o *parsing* dos dados. O que importa referir é que tem um elemento "blockdivision", que contém os vários elementos do formulário, seguido dos elementos do tipo botão.

```
<?xml version="1.0" encoding="UTF-8"?>
<form>
   <name>dados</name>
   <action>form.jsp</action>
   <blockdivision>
      <columndivision>
         <field>
            <type>header</type>
            <text>Exemplo</text>
            <even_field>false</even_field>
         </field>
         <field>
            <type>textmessage</type>
            <text>Hello world!</text>
            <cssclass>
            </cssclass>
            <align>left</align>
            <even_field>true</even_field>
         </field>
         <field>
            <type>selection</type>
            <obligatory>false</obligatory>
            <text>Lista</text>
            <variable>list3</variable>
            <onchange_submit>
            </onchange_submit>
            <option>
               <text>Escolha1</text>
               <value>escolha1</value>
               <selected>no</selected>
            </option>
            <option>
               <text>Escolha2</text>
               <value>escolha2</value>
               <selected>no</selected>
            </option>
            <option>
               <text>Escolha3</text>
               <value>escolha3</value>
               <selected>no</selected>
            </option>
            <even_field>false</even_field>
         </field>
         <field>
            <type>textbox</type>
            <text>Texto</text>
            <variable>texto</variable>
            <value>
            </value>
            <size>40</size>
```

```
                    <maxlength>100</maxlength>
                    <obligatory>false</obligatory>
                    <suffix>
                    </suffix>
                    <disabled>false</disabled>
                    <onchange>
                    </onchange>
                    <onblur>
                    </onblur>
                    <onfocus>
                    </onfocus>
                    <even_field>true</even_field>
                  </field>
               </columndivision>
            </blockdivision>
            <button>
               <id>5</id>
               <name>_guardar</name>
               <text>Guardar</text>
               <operation>disableForm();document.dados.op.value=2;</operation>
               <tooltip>Guardar</tooltip>
            </button>
            <button>
               <id>4</id>
               <name>_avancar</name>
               <text>Avançar</text>
               <operation>if(CheckEmptyFields())      {      disableForm();      document.dados.op.value='3';
document.getElementById('_button_clicked_id').value='4'; } else { return false; }</operation>
               <tooltip>Avançar</tooltip>
            </button>
            <hidden>
               <name>0_MAX_ROW</name>
               <value>0</value>
            </hidden>
            <hidden>
               <name>2_MAX_ROW</name>
               <value>0</value>
            </hidden>
            <hidden>
               <name>subpid</name>
               <value>-10</value>
            </hidden>
            <hidden>
               <name>flowid</name>
               <value>80</value>
            </hidden>
            <hidden>
               <name>op</name>
               <value>0</value>
            </hidden>
            <hidden>
               <name>flowExecType</name>
               <value>
               </value>
            </hidden>
            <hidden>
               <name>3_MAX_ROW</name>
               <value>0</value>
            </hidden>
            <hidden>
               <name>1_MAX_ROW</name>
               <value>0</value>
            </hidden>
            <hidden>
               <name>pid</name>
               <value>-10</value>
            </hidden>
            <hidden>
               <name>_serv_field_</name>
               <value>-1</value>
            </hidden>
            <hidden>
               <name>curmid</name>
               <value>0</value>
```

```
    </hidden>
    <hidden>
       <name>_button_clicked_id</name>
       <value>
       </value>
    </hidden>
    <loadingLabel>A processar...</loadingLabel>

</form>
```