Relatório

Instituto Superior de Engenharia de Lisboa

Licenciatura/Mestrado em Engenharia Informática e de Computadores

Programação de Dispositivos Móveis

Verão de 2014/2015

Projeto Final

Grupo 4

António Borba da Silva – 22908

Rui Corveira – 37325

Pedro Cardoso – 37539

1. Abordagem

Para desenvolvimento do projeto, além do cumprimento dos requisitos técnicos do enunciado com recurso criterioso às técnicas apresentadas ao longo do semestre, parte delas já experimentadas nas séries de exercícios realizadas, optámos por incluir mais uma série de tecnologias – essencialmente livrarias – de terceiras partes, o que nos permitiu focar nos problemas a resolver, beneficiando da qualidade e funcionalidade de código produzido profissionalmente por terceiros, poupando tempo nosso em desenvolvimento e incluindo soluções bem pensadas para problemas recorrentes (Ex.: gson, geofences).

Os problemas funcionais centrais que identificamos para resolução foram:

- *A definição da validação para utilização – assegurar mecanismos consistentes de manutenção dos atributos de utilizadores necessários à utilização da aplicação.*
- *O mecanismo de autenticação de utilizadores com recurso aos atributos de utilizadores referidos no ponto anterior.*
- *O mecanismo de publicação das notificações, com respeito pelos requisitos do enunciado.*
- *A receção de notificações de localização e a respetiva apresentação ao utilizador.*

Do ponto de vista técnico, os desafios principais que identificámos foram:

- *A otimização dos algoritmos com vista à parcimónia no consumo de energia dos terminais.*
- *O modo de utilização dos recursos internos dos equipamentos.*
- *A utilização de técnicas que preservem a capacidade de resposta da UI thread.*
- *Os mecanismos de sincronização entre todos os componentes do sistema.*

2. Solução técnica

A. Notificações

Para entrega de notificações de localização pelos terminais de docentes e respetivo consumo pelos alunos subscritores dessas notificações recorremos ao mecanismo de Push Notifications

da plataforma Parse.com.

I. Notificações de localização

01. Channels

Dado que a plataforma Android não tem um mecanismo próprio para este tipo de serviço, a plataforma Parse.com disponibiliza esse serviço com recurso ao serviço GCM[1] da Google.

A plataforma Parse.com disponibiliza diferentes métodos de abordagem ao serviço Push, mas optámos pela utilização de “channels”.

Basicamente cada terminal subscritor declara os “channels” de que ser notificado. Os terminais emissores de notificações publicam notificações identificando os “channels” a que se destinam.

De notar que a emissão e receção de notificações não está ligada a utilizadores mas sim a equipamentos (anónimos também podem enviar/receber notificações), pelo que a gestão de permissões é integralmente delegada no programador.

De notar ainda que um “channel” não tem de ter uma definição prévia, podendo ser qualquer String – ainda que com determinadas características e limitações. A possibilidade de envio de notificações para “channels” e subscrição de um “channel” são portanto inteiramente autónomas, passando a notificação quando o “channel” coincidir. Releva ainda notar que não há qualquer limite anunciado para o número de “channels” subscritos ou para os quais são enviadas notificações.

A metáfora que utilizámos para a utilização do serviço foi a representação da identidade de cada emissor (docente) por uma String construída a partir do seu endereço de email. Sendo os endereços de email dos docentes conhecido à partida por todos os atores, pareceu-nos um bom candidato para essa função de representação de um “channel”.

02. Receção de notificações

O envio da notificação para a plataforma Parse.com é efetuado por uma chamada direta de um método estático ParsePush. Já a receção é mais elaborada a nível de possibilidades de modelação.

O SDK da plataforma Parse.com disponibiliza um Broadcast Receiver da Google para suportar a receção das mensagens no Android.

Através da modelação da informação associada à mensagem recebida é possível condicionar o comportamento desse boradcast receiver.

No caso presente, modelámos a mensagem em JSON, incluindo uma chave “action” a que o broadcast receiver reage enviando invocando um intente para um broadcast receiver que esteja declarado no Manifesto com Intent Filter para o valor dessa chave.

O broadcast receiver da nossa aplicação armazena o conteúdo relevante da notificação numa tabela de um content provider que alimentará uma View de apresentação na MainActivity.

Por outro lado, sempre que a aplicação não esteja visível, o broadcast receiver envia uma

[1] GCM – Google Cloud Messaging

mensagem de notificação para a UI para alertar o aluno subscritor da informação adicional disponível na aplicação. Ao selecionar essa notificação é lançada a Activity que apresenta as mais recentes localizações dos docentes, recebidas nas últimas 12 horas.

03. Sincronização

Tanto as subscrições de “channels” como o envio de notificações são armazenadas internamente num contente provider. Isto assegura que em caso de indisponibilidade ou falha de comunicações, a informação é mantida e sincronizada quando possível.

A sincronização com o serviço Push do Parse.com é assegurada por um SyncAdapter cuja requisição de execução é efectuada no momento da alteração no content provider, através de um pedido requestsync do content provider. O content provider lança um intent para um bound service que responda ao intente-filter android.content.SyncAdapter. Esse serviço limita-se devolver uma instancia de AbstractThreadedSyncAdapter – o nosso SyncAdapter - que sabe efetuar a transformação do conteúdo do provider para as classes de subscrição de “channels” e envio de notificações de localização em formato JSON.

II. Autenticação

Para autenticação utilizamos os mecanismos de AccountManager do Android.

Estes mecanismos suportam a noção de token sobre um login / password autorizados.

O principal desafio na implementação deste mecanismo foi a assegurar a coerência entre a informação no Account Manager e no ParseUser, dado que o Parse.com faz a sua gestão de tokens, pelo que foi necessário mapear o conceito de sessão do AccountManager no conceito de sessão do Parse.com, efectuando login no Parse.com quando o utilizador seleciona uma conta no Account Manager e fazendo logout, quando no Account Manager se invalida o token.

Um ponto relevante na gestão de autenticação é a validação que é efectuada nas tentativas de subscrição ou publicação de notificações que apenas é autorizada a utilizadores que tenham previamente efetuado a validação do email fornecido no login – Os utilizadores apenas conseguem sincronizar subscrições e “channels” ou publicar notificações após validação do email (Por sua vez apenas são autorizados endereços de e-mail “*.isel.pt” e “*.isel.ipl.pt”.

III. Informação de Docentes e Alunos

A par dos mecanismos de autenticação e de notificações já referidos, foi necessário validar o tipo de utilizador para determinar o seu perfil, propriedades e determinar a funcionalidade que lhe é disponibilizada. Para tal, a informação sobre docentes – e também alguma sobre o próprio aluno enquanto utilizador – foi obtida a partir da API disponibilizada pela plataforma Thoth.

O mecanismo de sincronização é em tudo semelhante ao anterior – Content Provider / Account Manager / SyncAdapter, mas agora programados de modo diferente:

- *No Account Manager não há uma conta para acesso, mas apenas um "dummy" para cumprir o suporte ao SyncAdapter.*
- *O SyncAdapter, em vez de ser ativado em função da utilização do Content Provider, é agora lançado periodicamente (diariamente) para receber eventais alterações que possam ocorrer nas listas de Docentes e de Alunos.*
- *O Content Provider é funcionalmente semelhante ao outro de sincronização com o Parse.com, ambora não necessite de alimentar a Viewde Docentes presentes no ISEL, nem a sincronização com a plataforma Parse.com.*

As imagens dos docentes são obtidas nos link fornecidos pelo Thoth (Gravatar) e armazenadas e ficheiro, sendo usadas Handlers e LooperThreads para o efeito, com vista a minimizar o impacto na UI thread.

IV. Leitura de posicionamento

O posicionamento é obtido primariamente pelo provider GPS, e depois pelo provider Network, o GPS não esteja disponível e o MAC address do ponto de acesso conste de uma lista armazenada internamente.

A calendarização da cadência de leituras é calculada em função da distância e da expectativa de tempo de chegada ao ISEL, em função da distância atual e de uma velocidade média de deslocação (90Km/h). Assim, tentamos que sejam evitadas leituras intermédias com pouca probabilidade de serem úteis.

V. Apresentação de posicionamento

Para apresentação das notificações e a par da informação em lista, propusemo-nos efetuar a apresentação num mapa, com suporte no Google Maps.

Começamos por efetuar o levantamento de coordenadas dos edifícios do ISEL e produzir algumas classes que determinam em que edifício (ou no exterior) é que a posição se encontra, mas posteriormente optámos por substituir essas classes pelo Google Fences – uma biblioteca que usa um paradigma de vizinhança de um ponto central, mas que fornece muita funcionalidade adicional, sem grande perda de qualidade de informação, dada a precisão da deteção por GPS.

O mecanismo de atualização das posições no mapa é o mesmo da atualização das posições na View.

Para simularmos o posicionamento de docentes desenvolvemos ainda um pequeno simulador que nos permite selecionar num mapa o posicionamento desejado de um docente de teste e ver essas mudanças a ocorrerem na aplicação, com o push da mensagem e a respetiva receção pelos terminais subscritores dos respetivos "channels".

3. Melhorias identificadas

Ocorreram-nos várias melhorias à aplicação que não efetuamos por falta de tempo, mas

deixamos anotadas as que nos parecem mais relevantes:

- *Dados de configuração no Parse.com*
  - A informação de configuração de posicionamento de edifícios (e qualquer outra) poderia ser mantida numa classe do Parse.com, sendo descarregada pelos terminais após receberem uma notificação para um “channel” “configuration” que todas as aplicações subscreveriam.
- *Validação de utilizadores*
  - Um utilizador com um email validado será sempre um possível utilizador da plataforma. Poderia ser implementado Cloud Code (JavaScript) para em reação a alguma condição reescrever o email de utilizador para que este o tivesse de revalidar.
- *Utilização de CloudCode*
  - As funções de envio de push notifications, bem como de subscrição poderiam ter de chamar funções CloudCode, em vez de acederem diretamente ao serviço para obviar eventuais problemas de segurança. Também a validação do emailVerified poderia ser aí efetuada.
- *Melhorias técnicas*
  - Por falta de tempo e de prática nesse tipo de programação deixámos muito espaço para libertação de processamento da UI thread.
  - O código poderia estar todo melhor estruturado.
  - Este relatório poderia estar mais bem estruturado.

4. Principais dificuldades sentidas

- O IDE não aceita breakpoints em código noutros processos, o que torna dificulta e atrasa bastante o desenvolvimento; É o caso do SyncAdapter.
- A parametrização dos serviços Push no Manifest.xml, em sistemas que usem Flavours no Gradle – como foi o nosso caso – está mal documentada e atrasou bastante o desenvolvimento.
- O IDE exige máquinas com algum músculo. É normal observar ocupação de memória acima de 1GB.