JogginGo!

Relatório Final

Universidade do Porto

Faculdade de Engenharia

FEUP

Mestrado Integrado em Engenharia Informática e Computação

Paradigmas de Programação

Luís Dias - 080509094 - ei08094@fe.up.pt
Luís Gomes - 080509169 - ei08169@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

2 de Junho de 2013

Conteúdo

1 Resumo

Com uma interface limpa e amiga do utilizador, o JogginGo! é uma aplicação Web que permite a gestão de todas as corridas feitas por qualquer utilizador registado. Cada corrida é um tratada como um conjunto de coordenadas GPS (*Global Positioning System*) recolhidas com recurso aos sensores de um dispositivo móvel Android. A cada minuto, intervalo de tempo definido, é recolhida a posição em que o atleta se encontra, e o conjunto de pontos é assim representado visualmente na interface *web*.

2 Introdução

Este projecto insere-se no âmbito da unidade curricular de Paradigmas de Programação, do 4º ano do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto. Tendo em conta o conteúdo programático da unidade curricular, era esperado realizar um projecto que envolvesse mais do que um paradigma de programação e que as diferentes partes desenvolvidas interagissem entre si, de forma a criar um produto único e funcional.

2.1 Objectivo

O objectivo deste trabalho foi criar uma plataforma *mobile* de controlo de corridas de um determinado atleta. Do ponto de vista de utilização, foi necessário criar uma plataforma que permitisse a qualquer utilizador registar-se e visualizar todas suas corridas. Assim, desenvolvemos também um *webservice* que desempenhasse essa tarefa. Para além disso, é o *webservice* o responsável por receber, tratar e armazenar os dados enviados pelo dispositivo móvel.

2.2 Motivação

A motivação principal deste projecto foi o desafio da integração de diferentes linguagens e paradigmas na criação de um produto único e funcional aplicado a um cenário real. Para além da componente académica, o desenvolvimento de um produto útil e enquadrado no mundo actual, com possibilidades de ser usado por qualquer atleta nas suas corridas, foi outra das motivações fulcrais. Achamos que o produto por nós desenvolvido responde às necessidades do público-alvo.

3 Descrição do Sistema Desenvolvido

3.1 Descrição Conceptual

3.1.1 Funcionalidades

Java (Android):

As tarefas disponibilizadas pela componente Java estão relacionadas com a aplicação Android. Esta é a responsável pela ligação com a parte funcionalmente mais importante do sistema, que é a captura de coordenadas durante a corrida de um utilizador. Assim que este termina a corrida, é notificado sobre os percursos que tem por sincronizar, podendo enviar para o webservice apenas alguns deles, ou todos de uma só vez.

- *GPSTracker:* Permite obter as coordenadas actuais do dispositivo móvel do utilizador. É, assim, responsável por comunicar estas alterações à aplicação, com um intervalo definido de 1 minuto, para que estas coordenadas possam ser guardadas para futura sincronização;
- *DatabaseHandler* - Responsável por guardar as coordenadas obtidas e adicioná-las a uma pista, correspondente à corrida actual do utilizador. Estas são guardadas numa base de dados temporária *SQLite*, que guarda os dados da corrida até que o utilizador faz a sincronização com a plataforma ou decide descartá-la.
- *MapWithStats* - Depois de terminar a corrida, o utilizador poderá ver um mapa, recorrendo à *API* do *Google Maps*, com o percurso efectuado. Poderá também ver um conjunto de estatísticas referentes à corrida, como velocidade média, distância percorrida e tempo total.
- *OAuth 2.0* - Para que fosse possível obter os percursos do atleta no dispositivo móvel, assim como fazer a sincronização dos mesmos para o *WebService*, foi necessária a implementação de um módulo *OAuth* que possibilitasse o login na plataforma online, através do *smartphone*.

Ruby on Rails:

A componente Ruby on Rails é a responsável pela plataforma *online* da aplicação. É, no fundo, a interface de comunicação com o utilizador, e é onde este se liga à vertente social da aplicação, onde pode competir de forma amigável pelos melhores tempos.

- *Signup* - Permite ao utilizador registar-se na aplicação para poder usufruir das funcionalidades da aplicação.
- *Login* - O utilizador entra na sua conta para poder ver os seus dados, corridas efectuadas e competir contra outros utilizadores.
- *Logout* - Permite ao utilizador terminar a sessão actual.
- *Test your limits* - Funcionalidade de competição amigável entre utilizadores, em que para determinado percurso o utilizador é desafiado a tentar obter o melhor tempo possível para ficar no *ranking* dos melhores tempos.

- *MyStats* - Permite que o utilizador veja os melhores tempos efectuados por ele, assim como distâncias mais longas e velocidades médias atingidas.
- *AddTrack* - Permite ao utilizador adicionar novas pistas para serem avaliadas como sendo passíveis de ser adicionadas às corridas do *Test your Limits*, para que outros possam competir.

3.1.2 Estrutura do Programa

Este projecto ficou dividido em 2 módulos distintos: *Webservice* e terminal Android. Relativamente ao *Webservive*, e tal como referido ao longo deste documento, é o responsável pela autenticação do utilizador, assim como pelo tratamento de toda a informação recebida do dispositivo móvel. Após receber a informação, o *webservice* analiza-a e guarda-a na base de dados, para mais tarde proceder à sua representação recorrendo a API do Google Maps. De notar que o dispositivo móvel apenas transmite ao *webservice* os percursos que ainda não tenham sido sincronizados.

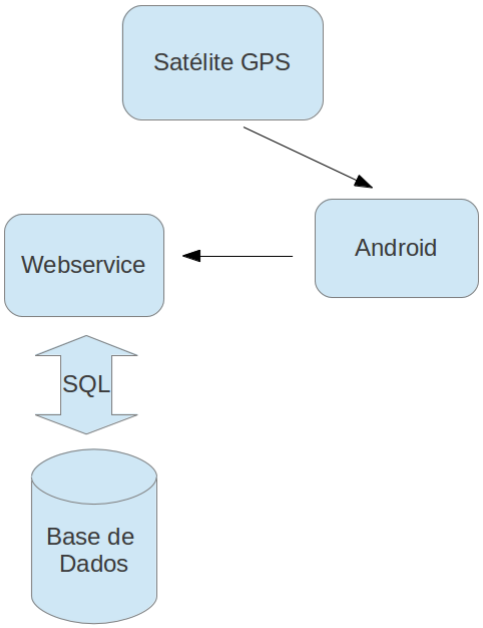

Figura 1: Estrutura do programa

O segundo módulo, ou seja, o terminal Android, trata da recolha das coordenadas GPS e do seu correcto armazenamento, numa base de dados SQLite, até que seja feita a sincronização com o *Webservice*. Para além da recolha, é também da sua responsabilidade a criação de um documento JSON utilizado como base para a comunicação com o terminal web.

3.1.3 Linguagens de Programação

Java:

Para o JogginGo! foi utilizada a linguagem Java para o desenvolvimento da aplicação Android, utilizando o SDK existente para o efeito. É responsável,

assim, pela obtenção de coordenada GPS do dispositivo móvel, pela criação de uma base de dados relacional SQLite e pela comunicação REST com o *webservice*.

A linguagem Java é multi-paradigma, e para este projecto foi utilizada principamente como linguagem mobile e orientada a objectos. O principal factor para a escolha do Java foi o seu excelente comportamento na programação Android, assim como a facilidade de criação de uma base de dados SQLite utilizando objectos. Para além disto, contou muito a familiarização com a linguagem, o que permitiu maior agilidade de desenvolvimento e menos tempo perdido na resolução de problemas.

Ruby:

Para o desenvolvimento do webservice da aplicação, que ao mesmo tempo é a plataforma online, foi utilizado Ruby, principalmente devido à framework Ruby on Rails que permite prototipagem rápida e eficiente. Esta framework tem uma estrutura MVC (Model-View-Controller) que permite facilmente separar a lógica do servidor (back-end) da interface com o utilizador (Front-End). Foi assim utilizada para todo o desenvolvimento de todas as funcionalidades da plataforma online e para criação da base de dados.

A linguagem Ruby também é multi-paradigma, e aqui foram usadas várias vertentes da mesma (orientada a objetos, scripting e reflexiva), tanto através da utilização de classes, do embeber de partes do código dentro de HTML ou Javascript e da utilização e carregamento de plugins (com novas classes) em tempo de execução . Para o desenvolvimento da plataforma foi equacionada a utilização de Python com a framework Django, mas depois de alguma pesquisa foi seleccionado o Ruby (on Rails) devido à facilidade com que se manipula a base de dados, à existência de inúmeros *plugins* que nos podem ajudar a melhorar a plataforma e também porque o Django é mais indicado para uma prototipagem rápida de ferramentas de administração, e menos de componentes com muitos utilizadores (como redes sociais) e muitas funcionalidades distintas.

Javascript:

Foi utilizado Javascript principalmente para uma melhor interacção do utilizador com a plataforma. A sua escolha deveu-se a ser, até ao momento, a única linguagem que permite a realização das suas funcionalidades na web e porque possui a biblioca JQuery que é também a mais utilizada na framework (Ruby on Rails) utilizada para o desenvolvimento da plataforma, que permite alterações ao Domain Objec Model (DOM) das páginas em tempo-real, sem necessidade de fazer novos pedidos ao servidor. Apesar de ser multi-paradigma, foi utilizada como linguagem de *scripting*, já que permite que o seu código esteja embebido noutras linguagens e não requerer qualquer tipo de compilação, sendo o seu código totalmente *client-side*, ou seja, executado na máquina do utilizador.

4 Implementação

Na implementação da aplicação foi necessário proceder a alguns mecanismos que permitissem a comunicação e controlo de dados entre das duas componentes principais, *webservice* e aplicação Android.

Para um corredor, é importante poder manter armazenadas no seu telemóvel

as corridas realizadas até que seja possível a sincronização com a plataforma. Para isso, foi desenvolvido um sistema de autenticação OAuth 2.0 para que esta troca de informação fosse seguro.

Os principais mecanismos de troca de informação são, assim, apresentados abaixo. Para a autenticação, é enviado um pedido para a criação de um *access token* no servidor, a partir da aplicação móvel Android:

```
HttpPost httppost = new HttpPost(OAuth2ClientCredentials.SCOPE+"oauth/token");


List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(2);

nameValuePairs.add(new BasicNameValuePair("client_id", OAuth2ClientCredentials.CLIENT_ID))
nameValuePairs.add(new BasicNameValuePair("client_secret", OAuth2ClientCredentials.CLIENT_
nameValuePairs.add(new BasicNameValuePair("code", code));
//nameValuePairs.add(new BasicNameValuePair("grant_type", "authorization_code"));
httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));


ResponseHandler<String> responseHandler=new BasicResponseHandler();
JSONObject response = new JSONObject(new DefaultHttpClient().execute(httppost, responseHan
```

Isto vai possibilitar a criação do *access token* que permite ao utilizador realizar pedidos que necessitam de autenticação. Na plataforma, o pedido, depois de processado pelo mecanismo de OAuth, devolve as informações pretendidas através do *render* das informações em formato JSON, em Ruby:

```
def show
...
respond_to do |format|
       format.html # index.html.erb
       format.json { render json: @user }
       format.xml { render xml: @user }
     end
...
end
```

Recebendo esta informação, é possível obter os dados do utilizador para futura utilização. Com esta autenticação, torna-se possível obter, por exemplo, as corridas realizadas por um utilizador, assim como fazer a sincronização no servidor de corridas recentemente realizadas.

Começando pela última, com a sincronização de novas corridas utilizando Java com Android:

```
HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost(URL);

...
JSONObject json = new JSONObject();
json.put("user_id", user_id);
```

```
json.put("approved", (t.isApproved() == 1? true:false));
json.put("name", t.getName());
...

response = httpclient.execute(httppost);
if(response.getStatusLine().getStatusCode() == 200){
db.deleteTrack(t);
}
...
```

Do lado do servidor, onde a informação é processada, em Ruby (on Rails):

```
def create

    initial_time = params[:initial_time]
    m=initial_time.split(":")
    year = m[0].to_i
    month = m[1].to_i
    day = m[2].to_i
    hour = m[3].to_i
    minute = m[4].to_i
    second = m[5].to_i
    mil = m[6].to_i

    initial = Time.local(year,month,day,hour,minute,second,mil)

    final_time = params[:final_time]
    m1=final_time.split(":")
    year = m1[0].to_i
    month = m1[1].to_i
    day = m1[2].to_i
    hour = m1[3].to_i
    minute = m1[4].to_i
    second = m1[5].to_i
    mil = m1[6].to_i

    final = Time.local(year,month,day,hour,minute,second,mil)

    delta = final-initial

    c = Time.at(delta).gmtime.strftime('%R:%S:%L')

    @track = Track.new(name:params["name"], city:params["city"],
      country: params["country"], user_id:params["user_id"],
      private: params["private"], approved: params["approved"])
    @user = User.find(params["user_id"])

    @track.points.build(params["points"])
    @track.timings.build(initial_time:initial,final_time:final, global_time:
      c)

    respond_to do |format|
```

```
      if @track.save
        format.html { redirect_to @user, notice: 'Track was successfully created.' }
        format.json { render json: @track, status: :created, location: @track }
      else
        format.html { render action: "new" }
        format.json { render json: @track.errors, status: :unprocessable_entity }
      end
    end
  end
```

Para uma visualização de um perfil, com todas as corridas realizadas pelo utilizador autenticado:

```
HttpClient httpclient = new DefaultHttpClient();
HttpGet request = new HttpGet((String)urls[0]);
            ResponseHandler<String> handler = new BasicResponseHandler();

...

public void onResultReceived(Object... results) {

JSONArray tracks = (JSONArray) results[0];

JSONObject track = tracks.getJSONObject(i);

ids.add(track.getString("id"));
names.add(track.getString("name"));

...
```

O pedido é enviado para a plataforma, que responde com os dados estruturados em JSON:

```
def tracks

@user = !params[:id].nil? ? User.find(params[:id]) : current_user

respond_to do |format|
       format.json { render json: @user.tracks }
     end
end
```

4.1 Ambiente de desenvolvimento

Para os diferentes módulos, foram usadas diferentes tecnologias e ambientes.

O *Webservice* foi desenvolvido utilizando a tecnologia Ruby on Rails, com a ajuda da API do Google Maps, programado no editor de texto Sublime Text,

recorrendo a um servidor local (localhost). Mais tarde, fizemos a migração da plataforma para o sistema de cloud, Heroku.

Relativamente ao terminal móvel, foi desenvolvida em Java (Android), recorrendo ao IDE Eclipse.

Ambos os modelos foram desenvolvidos utilizando computadores portáteis utilizando o sistema operativo Linux (distribuição Ubuntu).

5 Conclusão

As conclusões deste projecto são positivas em vários níveis. O facto de utilizarmos diferentes linguagens de programação e paradigmas para o desenvolvimento de um projecto torna-se um desafio e a maior motivação de desenvolvimento.

O facto de ser uma aplicação com um factor de utilidade real bastante elevado fez com que a satisfação pessoal fosse outro dos factores de grande motivação.

Foi também um bom desafio o desenvolvimento da aplicação Android que fosse capaz de recolher as coordenadas *GPS* ao longo duma corrida, já que não era algo que nenhum dos elementos tivesse experimentado. Para além disso, conseguir ligar estas coordenadas e a um mapa e ao mesmo tempo extrair informações tal como velocidade média, tempo e distâncias percorridas tornou-se bastante positivo.

Para desenvolvimento futuro, seria interessante o desenvolvimento de algumas funcionalidades que desejávamos ter implementado e que permitissem algum destaque face às diferentes aplicações semelhantes existentes no mercado, e ao mesmo tempo adicionar mais funcionalidades da plataforma *online* à aplicação Android e melhorar a sua usabilidade e potencial.

6 Melhoramentos

Há alguns melhoramentos que tínhamos em vista, no caso de ter existido mais tempo para a sua implementação. Dos mais importante, destacaríamos o melhoramento da interface gráfica, que de momento se encontra pouco cuidada pois decidimos dar mais ênfase às funcionalidades do que à usabilidade.

Outro dos melhoramentos que achamos que traría mais valor ao nosso projecto, era a inclusão da funcionalidade "Test your Limits", que permitiria a um qualquer atleta registado, competir num percurso pré-definido pela plataforma.

7 Bibliografia

7.1 URLs

- Storage Options, *http://developer.android.com/guide/topics/data/data-storage.html* , Maio 2013
- Google Developers, *https://developers.google.com/maps/*, Maio 2013
- Latitude and Longitude of a Point, *http://itouchmap.com/latlong.html*, Maio 2013
- GMaps4Rails, *https://github.com/apneadiving/Google-Maps-for-Rails*, Maio 2013
- Stack Overflow, *http://stackoverflow.com*, Maio 2013
- Android GPS, *http://www.androidhive.info/2012/07/android-gps-location-manager-tutorial/*, Maio 2013
- Google Maps Android API V2, *https://developers.google.com/maps/documentation/android/start_the_g* Maio 2013
- OPro Oauth 2.0 API, *https://github.com/opro/opro*

A Apêndice

A.1 Manual de Utilização

Para utilizar o JogginGo! na sua total funcionalidade, o utilizador necessita de uma ligação à Internet e de um *smartphone* Android com conexão GPS/A-GPS e Wi-fi/Ligação de dados.

Na plataforma online, provisoriamente em http://belele.heroku.com, é possível proceder ao registo introduzindo os seus dados pessoais ou fazer *login* com o *username* e *password*[Figura 2].

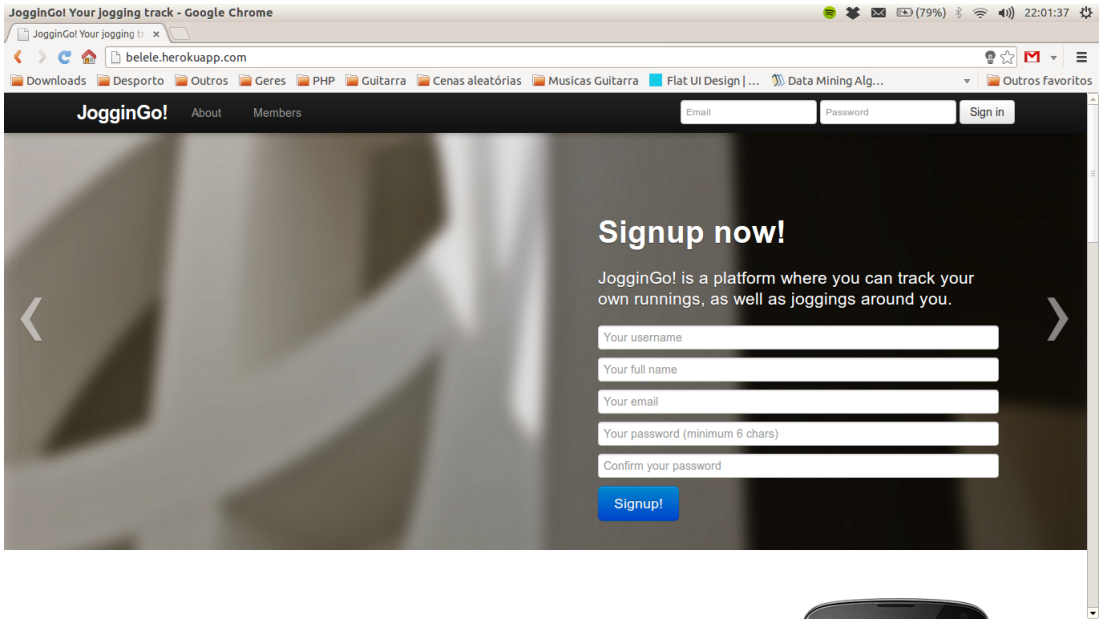

Figura 2: Registo na plataforma

De seguida, é apresentado o perfil do utilizador, com acesso às corridas realizadas e um mapa com o percurso efectuado. É possível, facilmente, alternar entre as corridas realizadas e os tempos efectuados [Figura 3].

Na aplicação Android, o utilizador começa por ter a possibilidade de começar imediatamente o *tracking* de uma corrida [Figura 4]. Também é possivel fazer primeiro login, a partir do botão de opções do telemóvel [Figura 5], e ter acesso às corridas que o utilizador tem sincronizadas com a plataforma [Figura 6].

Começando, é possível ver um ecrã que informa o tempo e distância percorridos [Figura 7].

Depois da corrida, o utilizador é notificado de quantas corridas tem não-sincronizadas, e pode user a notificação para verificar estas corridas [Figura 9]. Pode também ver o mapa da corrida ou realizar uma nova corrida. Caso pretenda, pode também sincronizar os dados com a plataforma ou efectuar o *login* para esse efeito [Figura 8].

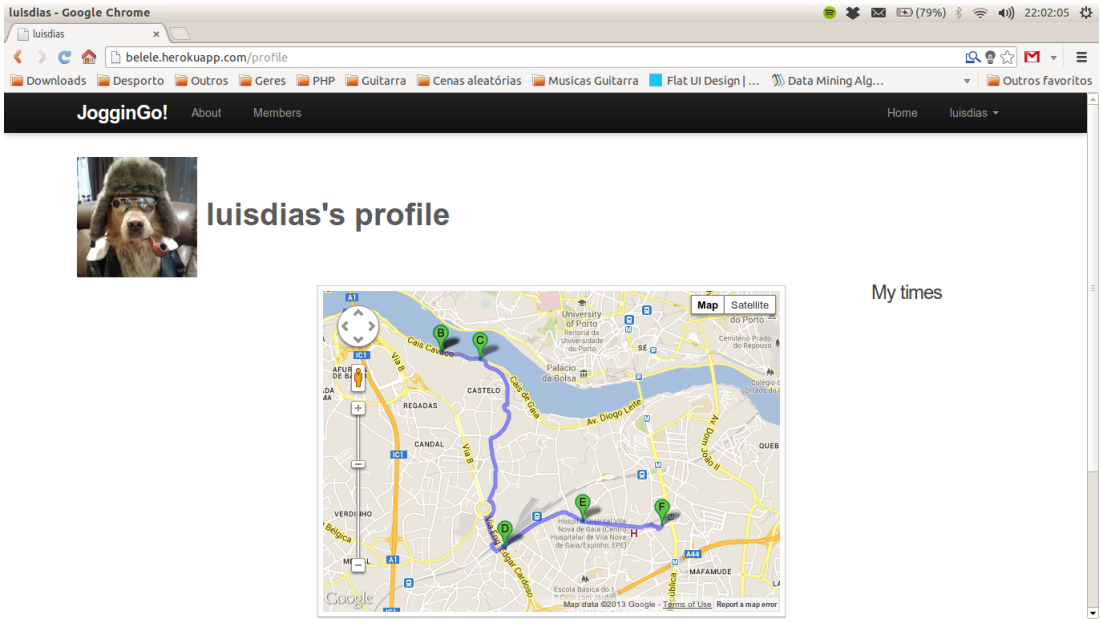

Figura 3: Perfil

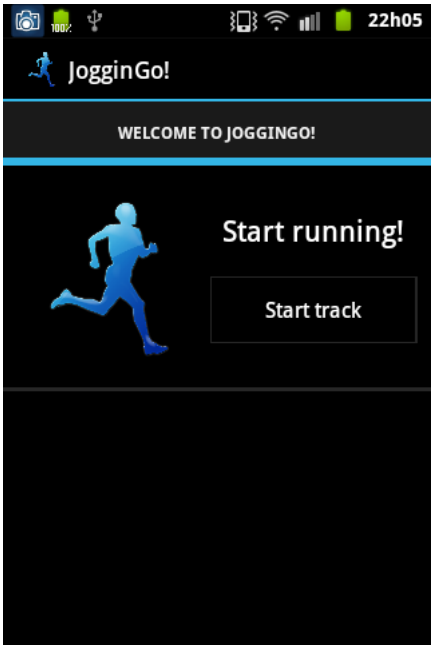

Figura 4: Inicio sem login

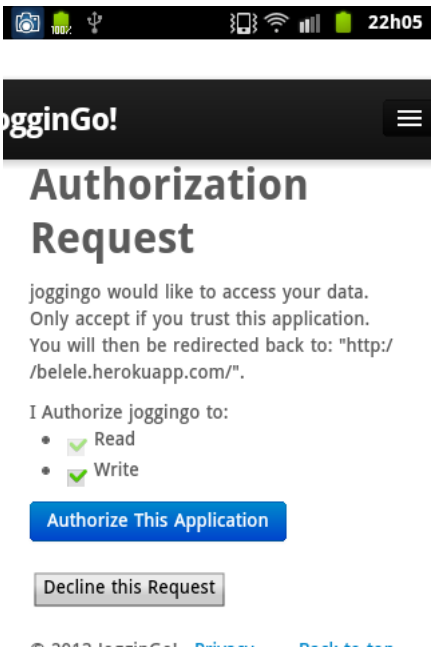

Figura 5: Pedido de autorização da plataforma

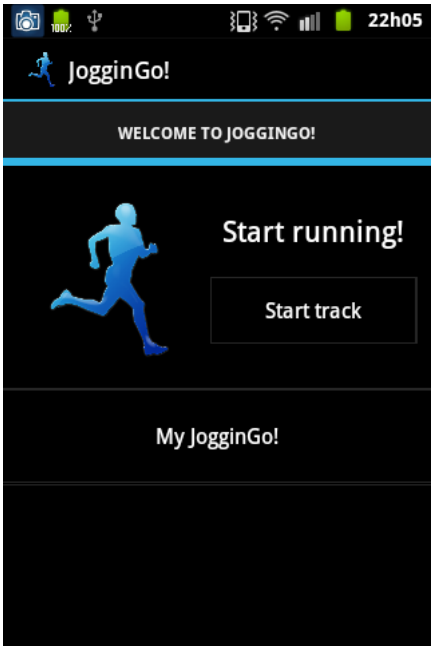

Figura 6: Inicio depois de efectuado o login

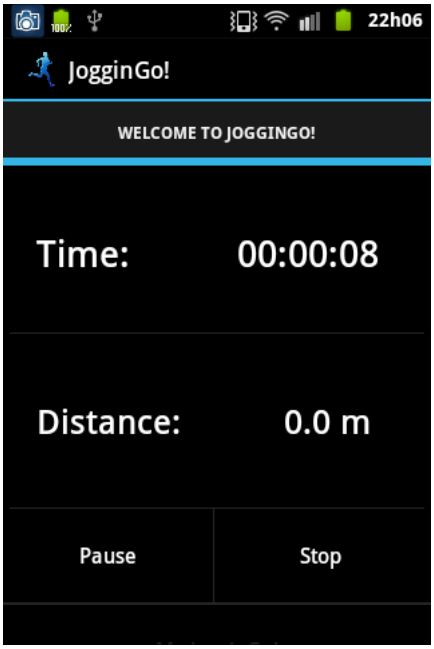

Figura 7: Tempo e distância percorridos

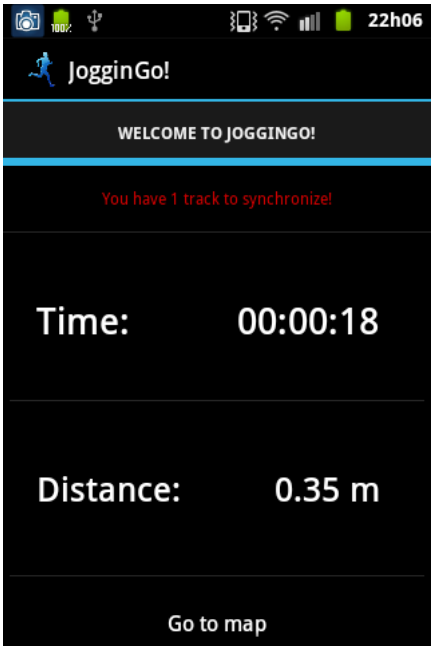

Figura 8: Tempo e distância percorridos

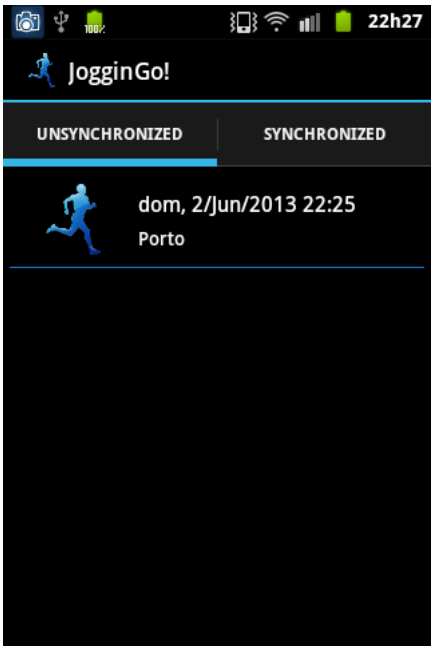

Figura 9: Lista de notificações