
Université de Bordeaux 1

Projet de Programmation

# Application CampusUB1

*Auteurs :*
Elyas Ben Hadj Yahia
Ryan Herbert
Paul Maribon-Ferret
Stefan Mijatovic
Tatiana Rocher

*Clients :*
Fabien Baldacci
Arnaud Casteigts

*Encadreur :*
Xavier Blanc

3 février 2013

**Résumé**

Ce document décrit le travail réalisé dans le contexte de notre projet de programmation. Le but de ce projet est de réaliser une application Android permettant d'accéder aux informations pertinentes des différents établissements du campus (Laboratoires, Universités, Ecoles). L'utilisateur choisit les établissements pour lesquels il veut accéder aux informations (dans un premier temps il n'y aura que Bordeaux 1 et LaBRI). L'application permet ensuite d'accéder aux annonces d'événements (en offrant la possibilité des les ajouter à l'agenda) et aux annuaires (en offrant la possibilité d'ajouter aux contacts du téléphone) sélectionnés.
L'idée est donc de rendre les informations liées aux établissements choisis plus faciles d'accès via un smartphone Android, afin de répondre à un besoin en déplacement, en donnant la possibilité d'enregistrer les informations importantes, et de les intégrer aux applications natives du téléphone, pour permettre une utilité de l'application lorsqu'une connexion internet n'est pas disponible.

# Table des matières

# Chapitre 1

# Etude de l'existant

## 1.1 Références

### 1.1.1 Android Application Development

L'ouvrage *Android Application Development* [1] est considéré comme l'une des références majeures dans le domaine du développement des applications sur Android. Il présente notamment l'architecture du système d'opération Android, ainsi que les différentes phases de développement des applications utilisateurs.

### 1.1.2 Site officiel de la SDK Android

Le site officiel de la SDK Android [2] met à disposition des développeurs une panoplie d'outils et de références. Ce site présente ses divers services, notamment les API d'Android et autres services de Google. On y trouve aussi plusieurs domaines d'application utilisés aujourd'hui qu'on peut intégrer dans notre application.

### 1.1.3 Article sur les applications mobiles des universités

Cet article sur les applications mobiles des universités [3] décrit les différentes contraintes liées au développement des applications mobiles d'universités. Etant donné que les étudiants sont de plus en plus équipés de smartphones, il devient intéressant de mettre en place une application mobile qui offre divers services utiles et faciles d'accès.

## 1.2 Applications existantes

### 1.2.1 uMontréal

Cette application propriétaire de l'université de Montréal [4] met à disposition de ses étudiants de nombreux services, tels que des flux d'actualités, un annuaire, un calendrier, et le plan du campus.

### 1.2.2 Plateforme Blackboard

Cette plateforme de développement est utilisée par la majorité des applications campus mobiles aux Etats-Unis [5]. On considère par exemple iStandford [6], application Android qui présente de nombreux services (internes et externes), dans la même philosophie que l'application uMontréal.

### 1.2.3 gReader, lecteur de flux RSS

Ce lecteur de flux RSS [7] offre une interface sobre et pratique pour gérer les abonnements aux flux RSS. Notre développement de l'interface graphique pourra s'inspirer du système des onglets et des toolbars de cette application.

### 1.2.4 Lecteurs de flux RSS open-source

Il existe déjà plusieurs lecteurs de flux RSS open-source, tels que Feedgoal [8] et Android-RSS [9]. les deux étant sous license GNU GPL (v2 et v3, respectivement). On pourra étudier s'il est rentable de reprendre quelques modules, ou de repartir sur notre propre base.

## 1.3 Ressources existantes

### 1.3.1 LDAP

Le protocole LDAP est une ressource potentiellement importante à la conception de notre application, notamment pour la mise en oeuvre de l'annuaire. LDAP signifie Lightweight Directory Access Protocol ; c'est donc un protocole conçu uniquement pour les annuaires, et l'annuaire du LaBRI s'avère être conforme aux normes LDAP. Ce sera donc une des technologies importantes à explorer afin de réaliser une réponse robuste au parsage de

l'annuaire du LaBRI.
Nous nous sommes posés la question de l'accessibilité de l'annuaire venant de l'extérieur, au format LDAP, cependant nous avons trouvé une page d'aide sur un forum du LaBRI[10] qui pourrait donner une solution éventuelle.

### 1.3.2 HTTP/HTML

Une alternative au protocole LDAP serait d'effectuer des requêtes HTTP, afin de parser des pages HTML. Cette méthode pourrait répondre aux besoins de parser les annuaires dont il en est question, si jamais nous ne pouvons nous reposer sur le LDAP. Le problème posé par cette méthode repose dans la robustesse de la solution. C'est-à-dire qu'un changement de l'implémentation des pages (changement/mise à jour des balises, ou même un changement de forme) pourrait rendre l'application inutilisable. L'utilisation de cette méthode demande énormément de travail de généricité du parsage des pages. Ceci pose plusieurs problèmes, notamment sur l'imprévisibilité des changements potentielles dans les pages. Il est impossible de prévoir à 100% les modifications qui pourraient avoir lieu sur les pages, et donc l'application pourrait rapidement devenir inutilisable. Il sera nécessaire de s'inspirer voire de réutiliser certains codes de web crawlers open source afin d'assurer d'avoir une application aussi robuste qu'elle puisse l'être en parsant des pages HTML.

### 1.3.3 Analyse textuelle

Au cours de l'implémentation de cette application, nous pourrions être amenés à effectuer de l'analyse textuelle sur des textes afin d'en extraire les informations dont nous avons besoin, par exemple si une date n'est pas au sein d'une balise facilement repérable/exploitable. Il faudra donc peut-être tenter d'extraire des dates des textes disponibles. Celà pose plusieurs problèmes :

- Le texte peut faire référence à un événement passé, simplement à titre informatif, auquel cas, une date extraite n'aurait pas nécessairement de valeur à l'exploitation avec notre application.
- Le texte peut contenir plusieurs dates, faisant référence à plusieurs événements passés, présents, ou futurs et il sera donc difficile d'exploiter les informations et de découper le texte en plusieurs événements, ou encore d'extraire la date pertinente à l'événement.

- Le texte peut ne contenir aucune date, n'étant simplement qu'une information quelconque.

Les événements inscrits dans les flux RSS du site de Bordeaux 1 possèdent une balise *pubDate*, qui devrait faire référence à la date de publication des articles, mais nous avons remarqué que pour la plupart des articles (notamment pour toutes les soutenances), cette date a été modifiée pour correspondre à la date de la soutenance. Cependant, ce n'est pas toujours le cas pour l'horaire. Certains articles possèdent des horaires qui correspondent à celle de l'événement (comme pour les soutenances, par exemple), mais d'autres ont un horaire qui ne correspond pas avec l'horaire de l'événement. Nous avons donc tiré la conclusion que l'horaire correspond parfois à l'horaire de publication de l'article.
Il sera donc possible de comparer les dates et horaires extraites par analyse textuelle avec les informations contenues dans la balise *pubDate* afin d'augmenter le taux de réussite de la proposition de date au moment de l'ajout dans l'agenda du smartphone.

Code 1.1 – Extrait d'un flux RSS de Bordeaux1 (29/01/2013)

```
<item>
  <title>Trophees du sport</title>
  <link>...</link>
  <description>
       Pour la 3e annee consecutive...
  </description>
  <content:encoded>
    <![CDATA[
      Ils seront recompenses a l occasion des Trophees du sport
      le jeudi 31 janvier 2013 a 18h dans l Atrium...
    ]]>
  </content:encoded>
  <pubDate>Thu, 31 Jan 2013 15:08:00 +0100</pubDate>
</item>
```

Sur cet exemple, l'heure qui apparait dans la balise *pubDate* ne correspond pas à celle de l'événement ; il va donc falloir analyser la description

pour essayer de trouver l'horaire correspondant (ici 18h).

Il existe plusieurs outils utiles à l'implémentation d'une solution à ce problème, notamment JFlex/CUP. Ces outils ressortent pour nous, car leur utilisation nous est déjà familière. Cependant, on peut se poser la question sur la possibilité de les utiliser au sein d'un téléphone Android. Dans le cas contraire, nous pensons également explorer l'utilisation des librairies Android de correction automatique, afin de trouver une solution plus native aux téléphones.
Etant donné ces problèmes, nous devrons surement trouver une approximation au problème, car nous ne pourrons proposer la bonne date à chaque fois qu'un utilisateur souhaite ajouter un élément à son agenda. Il sera donc parfois nécessaire de créer un menu qui s'ouvre automatiquement afin de proposer les dates qui ont été détectées par notre algorithme, et qui donne la possibilité à l'utilisateur de saisir sa propre date (ainsi on réduit de manière conséquente la marge d'erreur sur l'entrée finalisée dans l'agenda).

# Chapitre 2

# L'Application CampusUB1

## 2.1 Aperçu de l'application

### 2.1.1 Actualités

Une des fonctionalités principales de l'application est la récupération, l'affichage et l'enregistrement des événements dans le calendrier de l'utilisateur. Pour cela, nous prévoyons de récuperer les flux RSS les plus utiles de l'université Bordeaux 1[11], notamment les flux disponibles sur la page d'accueil du site de l'université (dont les actualités). En ce qui concerne le site du LaBRI, il n'y a aucun flux RSS et il faudra donc parser directement du HTML.

### 2.1.2 Annuaire

Il sera également possible de consulter les annuaires des établissements concernés, afin d'en extraire les informations désirées et les ajouter aux contacts du smartphone. Afin d'implémenter l'annuaire au sein de notre application Android, il est évident qu'il faut exploiter les annuaires de Bordeaux 1 [12] et du LaBRI. Vu que les accès aux bases de données ne peuvent pas nous être communiqués, nous pensons intégrer l'annuaire à travers des requêtes GET, dans le but de parser les résultats obtenus. Pour le moment ceci semble être la meilleure solution envisageable.

### 2.1.3 Utilitaires

Dans un deuxième temps, nous prévoyons de mettre en place un plan du campus, potentiellement intégré à l'application Maps du smartphone. Finalement, nous parserons les fichiers XML des emplois du temps[13], afin de mettre à disposition une copie locale de l'emploi du temps dans l'agenda du smartphone.

# Chapitre 3

# Analyse des besoins

## 3.1 Besoins fonctionnels

### 3.1.1 Fonctionnalités principales

#### Choix de l'établissement

Etant donné que l'application présentera des services pour deux établissements différents, l'utilisateur doit pouvoir naviguer entre les établissements, s'il souhaite accéder à leurs services et informations [Fig. 6.1].

- Priorité : 5/5
- Faisabilité : 5/5
- Risques : mise à jour des URLs si les adresses des établissements sont modifiées.
- Validation : s'assurer que les sélections et les changements d'établissements s'effectuent correctement.

#### Gestion des actualités

En accédant au menu des actualités, l'utilisateur disposera d'une interface à travers laquelle il recevra le flux des actualités de l'établissement [Fig. 6.2]. L'utilisateur doit pouvoir lire l'annonce de ces événements [Fig. 6.3], et doit pouvoir facilement rajouter cet événement au calendrier de son smartphone (Google Calendar). Cette action redirigera l'utilisateur vers le calendrier, dans lequel l'événement sera créé avec les informations récoltées. L'utilisa-

teur pourra donc modifier, valider ou annuler la procédure.

- Priorité : 5/5
- Faisabilité : 3.5/5
- Difficultés techniques : extraction des informations pertinentes.
- Risques : changement de structure du contenu à parser (flux RSS et pages HTML).
- Validation : assurer l'intégrité des informations extraites.

**Accès à l'annuaire**

L'utilisateur doit pouvoir effectuer facilement les tâches suivantes :

- Rechercher un contact dans l'annuaire par son nom et/ou prénom.
- Visualiser les informations d'un contact.
- Rajouter un contact à la liste des contacts de l'utilisateur (créer un nouveau contact ou éditer existant).
- Envoyer un mail au contact.
- Appeler la personne (si un numéro de téléphone est disponible).

Lors de l'ajout d'un contact, l'utilisateur sera redirigé vers la page de création d'un nouveau contact, dans laquelle les champs pertinents seront pré-remplis, en fonction des informations collectées. L'utilisateur pourra donc modifier, valider ou annuler la procédure.

- Priorité : 5/5
- Faisabilité : 4/5
- Difficultés techniques :
  - Avec LDAP : mise en place du protocole.
  - Sans LDAP : parsage des informations.
- Risques : changement de structure du contenu à parser.
- Validation : assurer l'intégrité des informations extraites.

**Mises à jour**

Notre application pouvant fonctionner hors connexion, les mises à jour de flux RSS étant de type pull, il est important de proposer les mises à jour sur les flux que nous parsons. Par exemple après une longue période sans connexion nous pourrons sauter sur l'occasion de proposer des mises à jour des flux dès que l'utilisateur ouvre l'application et qu'une connexion internet

est disponible. Une autre solution serait de mettre à disposition un bouton *refresh* pour que l'utilisateur puisse effectuer une demande de mise à jour (option préférable pour les personnes ayant un forfait internet limité, ou encore pour l'utilisation à l'étranger). Nous pourrions également proposer une solution intermédiaire, où l'application effectue des mises à jour régulières lorsqu'elle est connectée sur un réseau via le WiFi, mais requiert une demande explicite si le réseau est de type 3G.
L'application devra garder en mémoire les derniers éléments téléchargés pendant un mois après la date de l'événement(durée à revoir éventuellement). Ainsi lorsque l'on n'a pas accès à internet, on pourra accèder à ceux-ci. De plus, lors des mises à jour, on téléchargera uniquement les nouveaux événements. Ceci permettra à l'application d'économiser du temps d'éxécution, et potentiellement du crédit à l'utilisateur.

- Priorité : 5/5
- Faisabilité : 3.5/5
- Risques : mauvaise gestion de la mémoire.
- Difficultés techniques : éviter de télécharger de nouveau des informations déja collectées.
- Validation : vérifier l'intégration des nouvelles informations.

### 3.1.2 Fonctionnalités secondaires

Dans un deuxième temps, nous comptons implémenter quelques optionalités supplémentaires, qui ne rentrent pas dans le cadre du projet initial, mais qu'on estime être assez pratiques et utiles, et permettront de compléter l'application.

#### Emploi du temps

L'utilisateur étudiant doit pouvoir récupérer son emploi de temps du semestre et l'intégrer à Google Calendar. Il doit pouvoir indiquer sa filière, et éventuellement ses cours et ses groupes.

- Priorité : 1.5/5
- Faisabilité : 2.5/5à
- Difficultés techniques : extraction des informations en fonction du choix de l'utilisateur.

- Risques : changement du format de l'emploi du temps.
- Validation : assurer l'intégrité des données extraites.

**Plan du campus**

L'utilisateur pourra naviguer sur le plan du campus en utilisant Google Maps. L'utilisateur disposera des pointeurs vers les bâtiments principaux du campus.

- Priorité : 1/5
- Faisabilité : 2.5/5
- Difficultés techniques : maîtriser l'API de Google Maps.

## 3.2 Besoins non fonctionnels

### 3.2.1 Besoins de performance

L'application CampusUB1 est destinée à toute personne étudiant ou travaillant sur le campus de l'université. Elle devra donc être capable de tourner sur les derniers smartphones, tout en restant compatible avec des smartphones plus anciens et donc moins puissants puisque tout étudiant n'a pas forcément les moyens d'aborder un smartphone récent ayant une forte puissance de calcul et une grande quantité de RAM. CampusUB1 ne sera donc pas gourmande en ressources et pourra exécuter ces services de manière assez rapide même sur les smartphones moins performants.

- Priorité : 3/5
- Faisabilité : 5/5
- Difficultés techniques : bonne gestion de l'espace mémoire et de l'actualisation.
- Risques : implémention complexe et potentiellement longue.
- Validation : vérification que l'utilisation reste assez fluide et ne consomme pas beaucoup de ressources (à préciser un seuil de consommation de RAM).

### 3.2.2 Besoins de fiabilité

Etant donné que les différents services mis à disposition par CampusUB1 à ses clients s'avèrent souvent être d'un besoin professionnel, il est essentiel que les données affichées soient cohérentes et en accord avec les données d'origine. Si un chercheur du LaBRI recherche depuis notre application le numéro de la salle où a lieu sa conférence, il a tout intérêt à récupérer une information correcte. CampusUB1 ne peut tout de même pas garantir l'exactitude de toutes les informations données et ne se tient donc pas responsable d'éventuelles erreurs, sachant que l'application se base sur des algorithmes de reconnaissance de textes qui peuvent eux-mêmes être erronés d'origine. Il est évident que certaines questions de sécurité courantes se posent sachant que l'application nécessite une connection à internet continue, mais vu que CampusUB1 utilise uniquement des données publiques dans les services qu'elle fournit, la question de confidentialité des données ne se pose pas.

- Priorité : 4.5/5
- Faisabilité : 3.5/5
- Difficultés techniques : fiabilité de l'algorithme d'analyse textuelle pour détecter les dates des événements.
- Risques : création d'un événement avec des données erronées.
- Validation : simulation de l'algorithme d'analyse textuelle sur suffisamment de données pour avoir un taux de réussite acceptable (à déterminer).

### 3.2.3 Besoins d'intégrité de données

CampusUB1 doit accéder à certaines ressources personnelles du client pour pouvoir fonctionner, ce qui pose un problème de protection des données. Le calendrier et le répertoire du client sont les ressources auxquelles l'application est susceptible d'accéder, et on devra s'assurer que toute perte éventuelle de données déjà existantes ne soit pas engendrée par une erreur provenant de l'application. L'erreur humaine étant évidemment toujours possible. Toute modification éventuelle de données privées de l'utilisateur se fera donc impérativement avec son avis.

- Priorité : 4.5/5
- Faisabilité : 4.5/5

- Risques : perte des informations existantes sur le smartphone.
- Validation : monitoring du comportement de l'application.

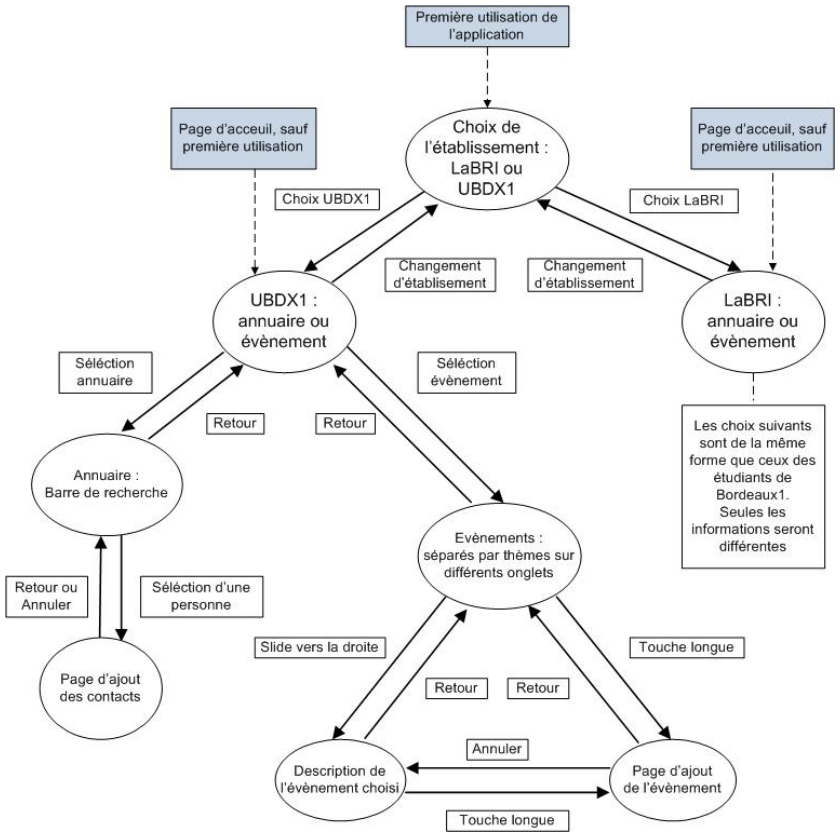
### 3.2.4 Besoins organisationnels

Etant donné que nous travaillerons sur la plateforme Android, le langage Java nous est imposé vu que l'API Android est fournie en Java. L'IDE dans lequel nous avons choisi d'implémenter notre application est Eclipse pour de nombreuses questions pratiques. Les développeurs d'Android disposent d'un plugin officiel pour Eclipse, qui facilite le travail du développeur en lui fournissant une panoplie d'outils tels que les différentes perspectives dont celle de débogage qui nous sera très utile. Ce plugin comporte également un émulateur d'appareils Android, ce qui nous permettra de facilement tester l'application sur une multitude d'appareils virtuels. De plus, l'API Android est très documentée et est maintenue fréquemment.

# Chapitre 4

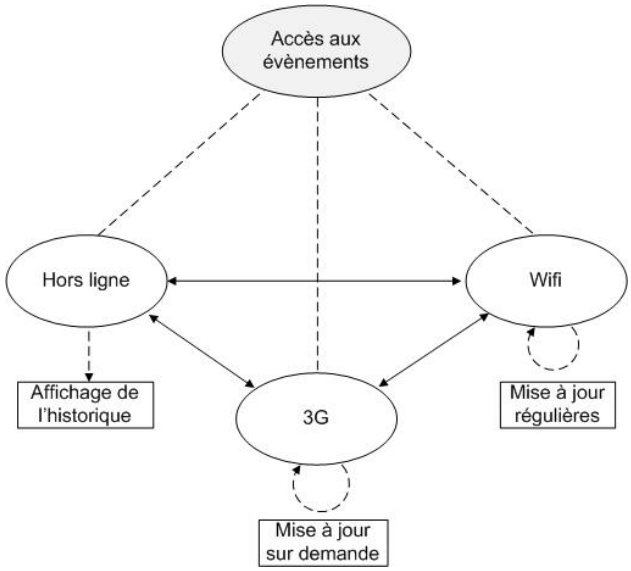
# Diagrammes de fonctionnement

## 4.1 Diagramme des cas d'utilisation



Ce diagramme présente toutes les actions réalisables par l'utilisateur et leurs enchaînements. Lorsque l'utilisateur démarre l'application pour la première fois, il choisit l'établissement auprès duquel il veut s'abonner pour accéder uniquement à ses services. Ce choix est alors retenu, et cette étape est donc omise lors des futurs démarrages de l'application. Une option existera tout de même pour modifier cette sélection si nécessaire. Depuis l'écran d'accueil, l'utilisateur pourra alors accéder aux services de son choix et pourra à tout moment revenir en arrière facilement.

## 4.2 Diagramme des mises à jour




Le diagramme de mises à jour représente les différents modes de connection dans lesquels on peut se trouver lorsqu'on accède aux événements. Il y a trois modes possibles gérés par le téléphone :

- Hors Ligne : pas d'accès à internet.
- 3G : accès au réseau de données mobiles (lent).
- WiFi : accès à un réseau sans-fil (rapide).

## 4.3 Diagramme de Gantt

Le diagramme de Gantt suivant présente une première estimation de la répartition des tâches sur le déroulement du projet.

| Tâches/semaines | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Connexion aux sites Web | S | | | | | | | | | |
| Architecture de l'application | E | E | | | | | | | | |
| Parsage de flux RSS | | T | T | | | | | | | |
| Interaction annuaire (LDAP) | | R | R | | | | | | | |
| Interaction annuaire (HTML) | | | R | R | R | | | | | |
| Interface Graphique | | P | P | P | | | | | | |
| Intégration des données | | | S | S | S | S | | | | |
| Gestion des mises à jour | | | E | E | E | | | | | |
| Analyse textuelle | | | | T | T | T | T | | | |
| Fonction de recherche annuaire | | | | | P | P | | | | |
| Elaboration des tests | R | P | T | S | R | E | T | S | P | E |
| Elaboration du rapport | A | A | A | A | A | A | A | A | A | A |
| Finalisation des tâches | A | A | A | A | A | A | A | A | A | A |

Chaque ligne est destinée à la répartition d'une tâche, et les colonnes représentent chacune une semaine du projet.
Les initiales représentent :

- A : Tout le monde
- E : Elyas Ben Hadj Yahia
- R : Ryan Herbert
- P : Paul Maribon-Ferret
- S : Stefan Mijatovic
- T : Tatiana Rocher

# Chapitre 5

# Tests et Validation

## 5.1 Choix de l'Etablissement

### 5.1.1 Bordeaux 1

Après la sélection de l'établissement Bordeaux 1, l'application devrait rediriger l'utilisateur vers le menu adapté pour la partie Bordeaux 1 [Fig. 6.2]. Au prochain démarrage de l'application, l'utilisateur devrait arriver sur ce menu automatiquement.

### 5.1.2 LaBRI

La sélection de l'établissement LaBRI doit se comporter comme pour le choix de l'établissement Bordeaux 1, mais en redirigeant vers le menu LaBRI [Fig. 6.9].

## 5.2 Evénements

### 5.2.1 Etat Connecté

L'application est connectée à Internet, alors elle télécharge les derniers évènements propres à l'établissement choisit (celles qui n'ont pas encore été téléchargées) et les affiche [Fig. 6.3]. Un petit toast s'affiche pour indiquer la date de la dernière mise à jour.

### 5.2.2 Etat Non Connecté

L'application affiche les événements contenus dans la mémoire du téléphone qui correspondent à l'établissement choisit [Fig. 6.3]. Un toast affiche la date de la dernière mise à jour.

# Chapitre 6

# Exemple d'utilisation

Les images suivantes représentent un premier aperçu de l'interface graphique de l'application. Nous imaginons donc un scénario d'utilisation qui montre le déroulement d'un usage typique.

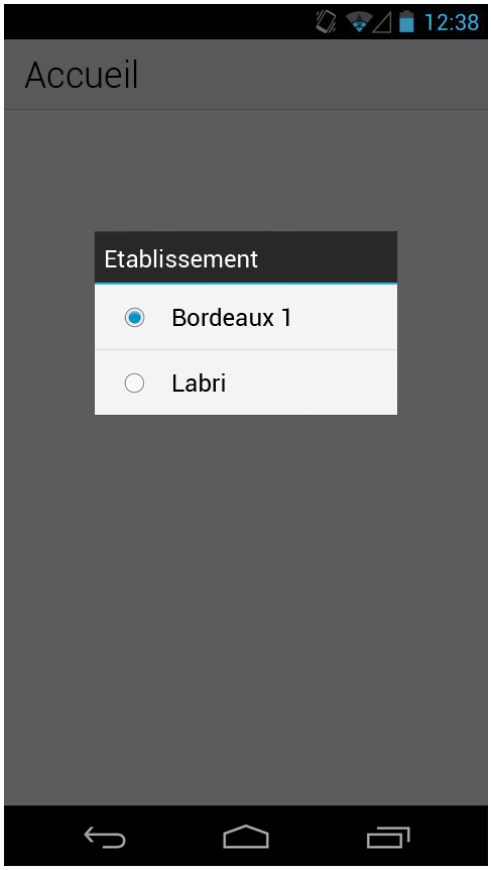



FIGURE 6.1 – Lors du premier lancement de l'application, l'utilisateur doit choisir son établissement. Une fois son choix effectué, il est enregistré et ne sera plus demandé. L'établissement est modifiable via les préférences.

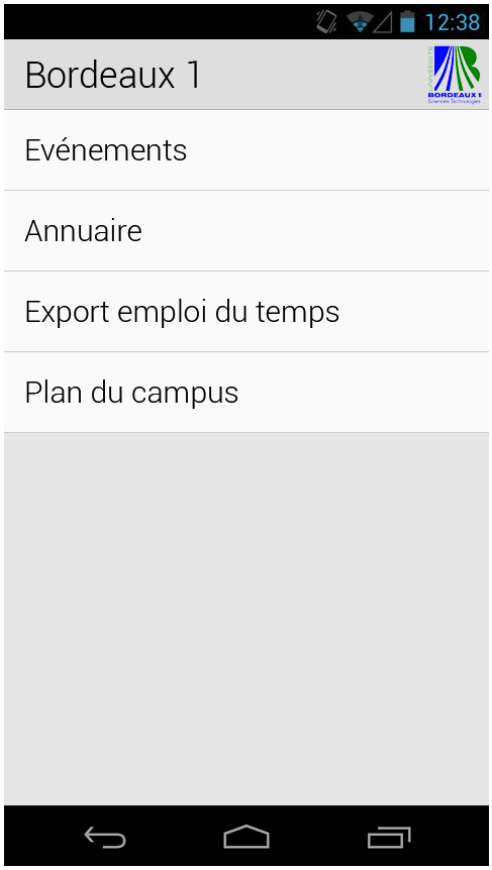



FIGURE 6.2 – Vue d'accueil pour Bordeaux 1


FIGURE 6.3 – Evénements. Pour lire la news et arriver sur l'écran de la figure 6.4, il suffit de toucher la news correspondante.


FIGURE 6.4 – Le bouton "plus" situé en haut à droite permet d'ajouter l'événement au calendrier du smartphone.

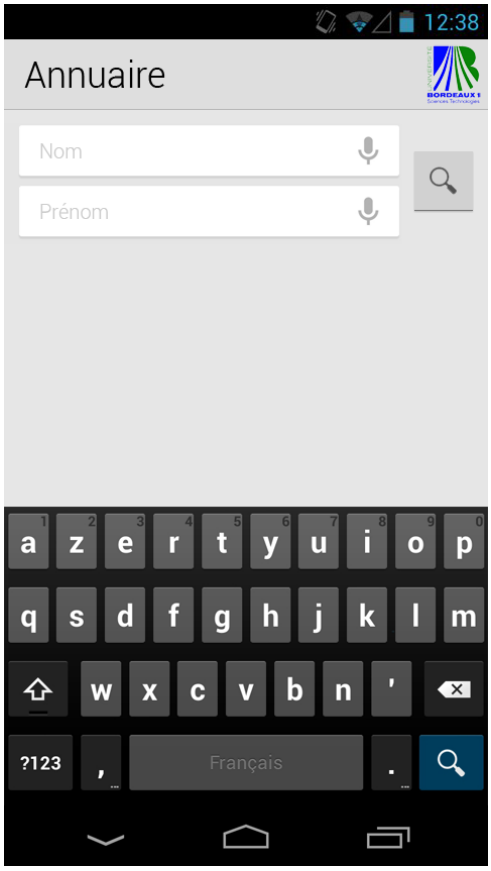



FIGURE 6.5 – Vue de l'annuaire.

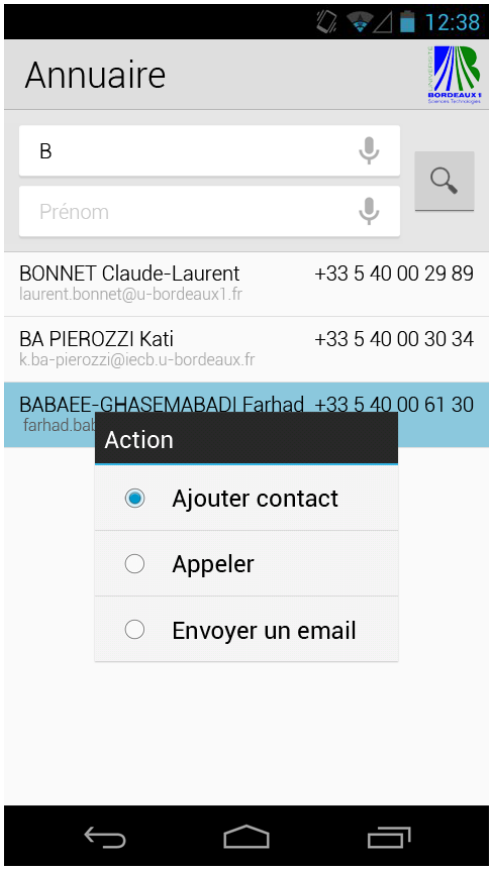



FIGURE 6.6 – Vue des résultats de l'annuaire.

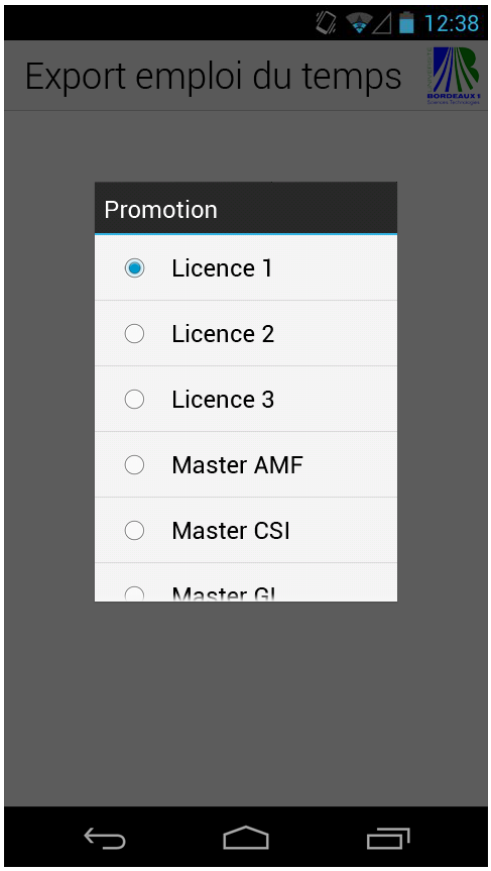



FIGURE 6.7 – Export de l'emploi du temps. L'application demande la promotion seulement la première fois et il faudra aller dans les paramètres pour la modifier ultérieurement.

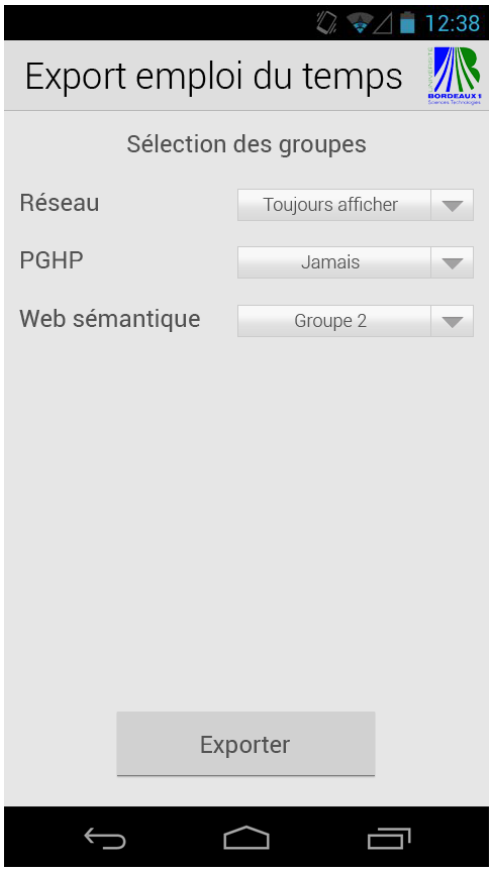



FIGURE 6.8 – Sélection des groupes pour l'export de l'emploi du temps.

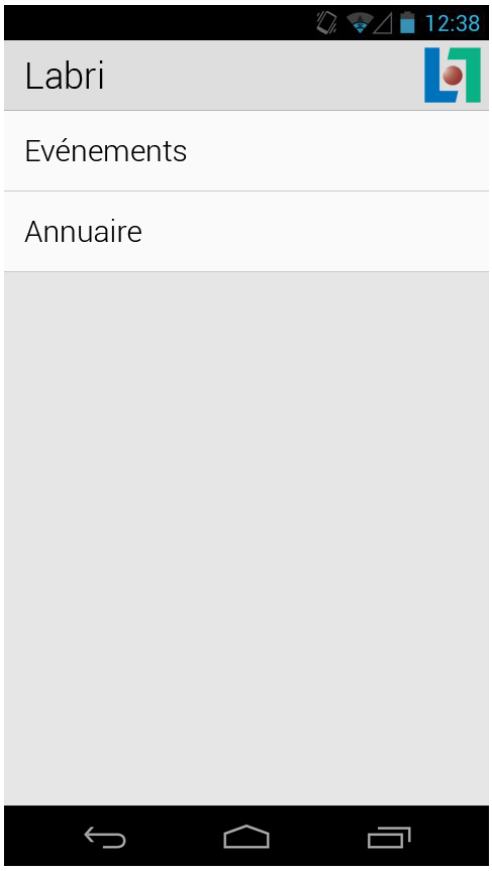



FIGURE 6.9 – Vue d'accueil pour LaBRI

# Bibliographie


[1] Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike. *Android Application Development*. O'REILLY, first edition, 2009.

[2] Google. Android developers. `http://developer.android.com/develop/index.html`.

[3] Kerry Eustice. How to build a university mobile application : best practice and insight, Decembre 2011. `http://www.guardian.co.uk/higher-education-network/2011/dec/08/building-university-mobile-apps`.

[4] DGTIC Université de Montréal. uMontréal, Septembre 2012. `http://www.umontreal.ca/mobile/`.

[5] Blackboard. Blackboard Mobile, 2009. `http://www.blackboard.com/platforms/mobile/overview.aspx`.

[6] Stanford. iStanford, Juillet 2012. `https://play.google.com/store/apps/details?id=com.blackboard.android.central.stanford&hl=en`.

[7] Google. gReader, Septembre 2007. `https://play.google.com/store/apps/details?id=com.noinnion.android.greader.reader&hl=en`.

[8] FeedGoal. `http://code.google.com/p/feedgoal/`.

[9] Android-RSS. `http://code.google.com/p/android-rss/`.

[10] Savel Guilhem. Accès à l'annuaire des comptes mails, Octobre 2012. `https://sos.labri.fr/sos/glpi/front/helpdesk.faq.php?id=78`.

[11] Bordeaux 1. Flux RSS de Bordeaux 1. http://www.u-bordeaux1.fr/pages-speciales/abonnement-aux-flux-rss.html.

[12] Université de Bordeaux 1. Annuaire, 2013. http://www.u-bordeaux1.fr/no-cache/annuaires/annuaire-des-personnes.html.

[13] Celcat.fr. Emplois du temps. http://www.disvu.u-bordeaux1.fr/et/edt_etudiants2/Master/Semestre2/g56017.xml.