application Multilingua :

JAVA Android, PHP, SQLite, MySQL, serveur Apache

Documentation technique

Version 1.0

Julien Boulnois

Contenu

Version(s)

03/03/17	1.0

Auteur(s)

Julien Boulnois	1.0	Démarrage

Description du projet et de l'application

La société Multilingua, exploite des salles de formation en langues étrangères (anglais, espagnol, allemand, portugais). Il dispose d'un site en ligne avec des cours et diverses ressources complémentaires à la formation présentielle. Face à l'accroissement de l'usage mobile, il souhaite développer rapidement une présence sur Smartphone.

Conscient qu'un Smartphone ne s'utilise pas de la même façon qu'un site, il souhaite proposer une expérience unique consistant à écouter ou lire une micro-leçon chaque jour, et à valider sa compréhension à travers divers exercices très courts. Les exercices doivent porter sur la leçon du jour mais peuvent aussi porter sur la leçon d'il y a quelques jours, pour vérifier si elle a bien été retenue.

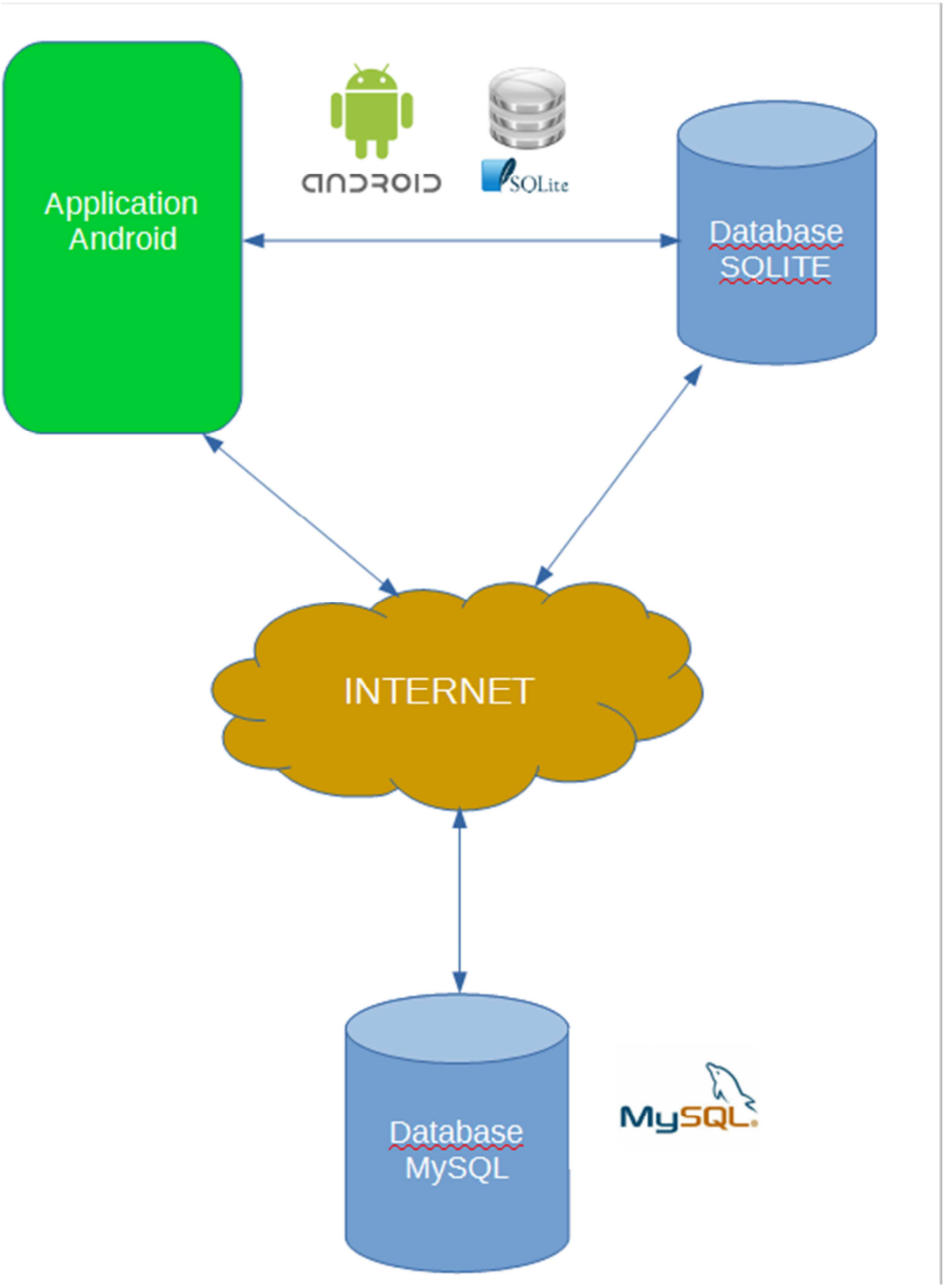
En outre, l'application doit permettre de contacter son responsable de formation chez Multilingua et de voir les prochaines dates de formation présentielles programmées. Un rappel doit être envoyé par l'application 1h avant la formation.

Décomposition du projet

Notre projet se décompose en différentes parties :

- L'application client, qui est une application qui fonctionne sur Android.
- Une base de données embarquée en SQLite permettant de stocker en local certaines données comme l'utilisateur en cours ou certaines dates.
- Une base de données MySQL, qui stocke toutes les données utiles permettant les verifications des utilisateurs, leur lecon du jour, les exercices, les responsables qui s'occupent des « étudiants ».
- Des services Web qui sont hébergés sur un serveur applicatif Apache et qui communiquent en permanence avec le logiciel client, pour assurer de multiples fonctions comme la vérification de l'utilisateur ou la mise à jour des lecons.

Architecture

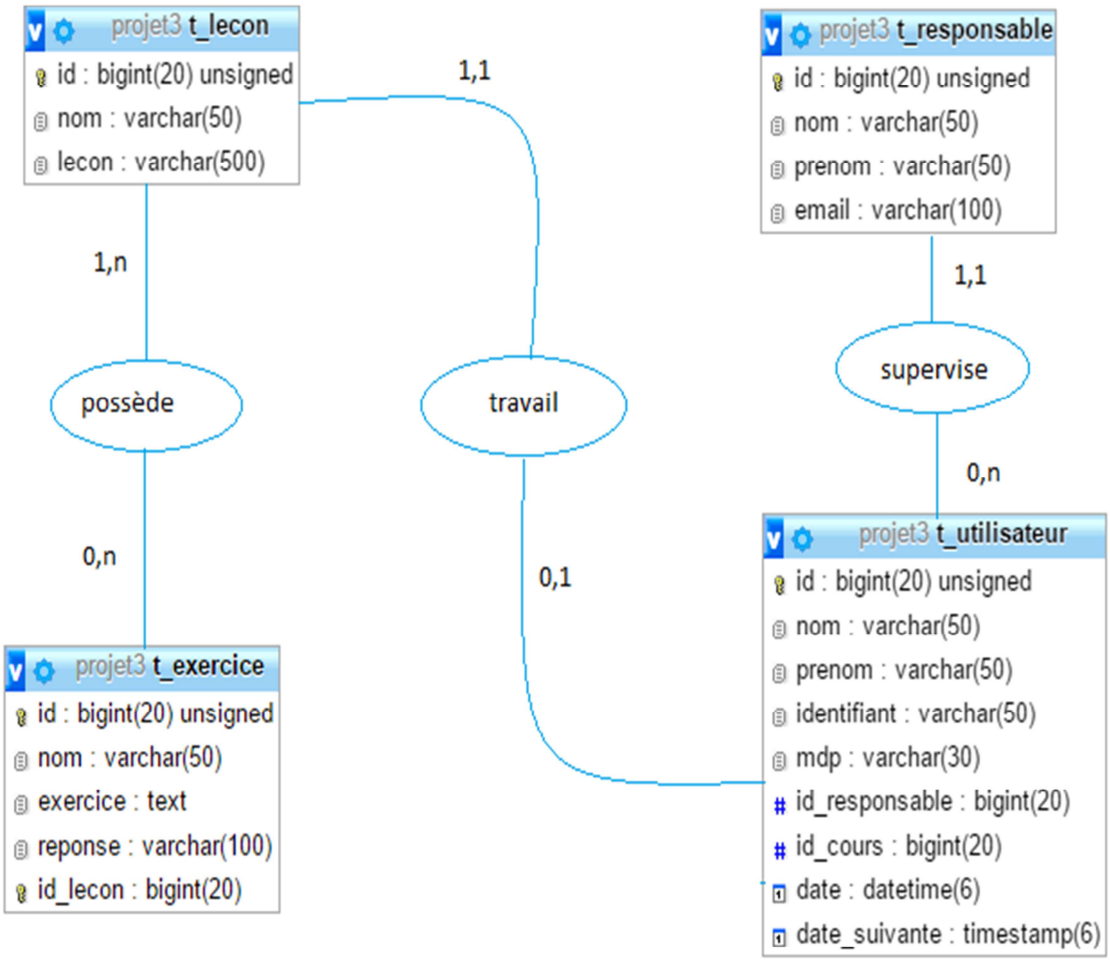


L'application client possède une base de données embarqué et appelle des web services pour effectuer des traitements sur une base MySQL via un serveur Apache.

Technologies utilisées

L'application client est développé en JAVA (Android) et possède une BDD embarqué SQLite. SQLite a été choisi comme SGDB du logiciel client pour sa simplicité. De plus elle est intégrée à Android.
Les WebServices sont codés en PHP et permettent de faire le pont entre la base de données sur internet et l'application client.
La base de données en ligne est une BDD MySQL.
Le serveur utilisé est un serveur Apache.

Modèle de données BDD MySQL



Modèle de données BDD SQLite

2 tables ont étés créés. Une table utilisateur, qui stock le nom de l'utilisateur ainsi que la date à laquelle il a terminé sa dernière leçon du jour, et une table qui stock les dates de RDV, ce qui permet au service créé de checker et vérifier les dates et heures avant de lancer une notification.

Aucun lien n'est fait entre ces 2 tables.

```
// creation de la table
public static final String DB_CREATE_UTILISATEUR = "create table "+
TABLE_UTILISATEUR + " (" +
        ID_UTILISATEUR + " integer primary key autoincrement, " +
NOM_UTILISATEUR + " text not null, " + DATE_LECON + " text not null);";

public static final String DB_CREATE_DATE = "create table "+ TABLE_DATE + "
(" +
        ID_DATE + " integer primary key autoincrement, " +
DATE_RDV_UTILISATEUR + " text not null, "+ DATE_RDV_SUIVANTE_UTILISATEUR +"
text not null);" ;
```

Les services web

5 services web sont nécessaires pour faire fonctionner cette application.

Chacun d'entre eux est créé pour un traitement spécifique :

- Vérifier que l'identifiant de l'utilisateur existe
- Fournir la bonne leçon du jour
- Fournir les exercices correspondant à l'avancée de l'utilisateur
- Trouver le responsable lié à l'utilisateur pour pouvoir le contacter
- Permet de faire un update une fois la leçon du jour terminer

```
<?php
$nom = $_GET["nom"];
try
{
    $bdd = new
PDO('mysql:host=localhost;dbname=projet3;charset=utf8','root','');
    $reponse = $bdd -> query("select t_lecon.nom, t_lecon.lecon,
t_utilisateur.date, t_utilisateur.date_suivante from t_lecon,
t_utilisateur where t_utilisateur.id_cours = t_lecon.id and
                                t_utilisateur.identifiant =
'".$nom."'");
    $output = $reponse -> fetchAll(PDO::FETCH_ASSOC);
}
catch(Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}
    echo (json_encode($output));
```

Le service

Le service a pour but, une fois l'identifiant valider dans l'application, de tourner en arrière-plan et vérifier la date et heure du jour, ainsi que la date et heure du prochain RDV dans l'attente de voir la date arrivée à 1 heure de la date de RDV.
Une fois cette condition validée, une notification apparaît sur le téléphone, alertant l'utilisateur que son prochain rendez-vous approche.

```
private void notifEnvoi ()
{
    NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(this);
    mBuilder.setSmallIcon(R.drawable.flag);
    mBuilder.setContentTitle("Notifications Multilingua");
    mBuilder.setContentText("votre date de RDV approche !");

    NotificationManager notifManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

    notifManager.notify(0, mBuilder.build());
}
```

L'application client

L'identification

Dans cette écran, l'utilisateur entrera son identifiant, puis cliquera sur le bouton permettant de lancer un certain nombre de traitement comme par exemple, vérifier que l'identifiant existe.

L'utilisateur ne pourra pas aller au-delà de cette écran tant que son identifiant n'aura pas été validé.

```
private void Verification ()
{
    int id = Integer.parseInt(idRecu.getText().toString());
    if (id > 0)
    {
        texteValidation.setText("Connexion effectuée avec succès !");
        boutonValidOK.setVisibility(View.VISIBLE);
    }
    else
    {
        texteValidation.setText("Connexion echouée, l'identifiant n'est pas
bon !");
        boutonValidKO.setVisibility(View.VISIBLE);
    }
}
```

L'identifiant est stocké dans la BDD embarquée si il n'y existe pas encore.

Le Menu

Le menu se compose de 3 boutons donnant chacun accès à une partie de l'application :

- La leçon du jour
- Les exercices
- Le contact

Dans ce menu, nous retrouvons aussi les dates des prochains rendez-vous de l'utilisateur.
Ces dates sont stockées dans la base de données MySQL et appelées via le service web correspondant.
Une vérification est également effectuée au moment de cliquer sur le bouton « Leçon » pour vérifier que la leçon du jour n'a pas encore été faites.

Sinon, il n'est plus possible d'y accéder avant le lendemain pour le prochain cours.

```
private void VerifDateLecon ()
{

    Date datetime = new Date();
    String dateJour = new SimpleDateFormat("yyyyMMdd",
Locale.FRANCE).format(datetime);

    int dateDuJour = Integer.parseInt(dateJour);
    String dateLecon = null;

    Cursor c = db.rawQuery("Select "+ CreateBDD.DATE_LECON + " from " +
CreateBDD.TABLE_UTILISATEUR + " where "  +  CreateBDD.NOM_UTILISATEUR + " =
'" + MainActivity.CURRENT_USER +"'",null);

    for(c.moveToFirst(); !c.isAfterLast(); c.moveToNext())
    {
        dateLecon = c.getString(0);
    }

    if (c.getCount() > 0)
    {
        int dateStocker = Integer.parseInt(dateLecon);

        if(dateStocker <= dateDuJour)
        {
            Intent intent = new Intent(Menu.this, viewLecons.class);
            startActivity(intent);
        }
        else
        {
            Toast.makeText(Menu.this, "Vous avez déja visionner votre lecon
du jour", Toast.LENGTH_LONG);
        }
    }
    else
    {
        Intent intent = new Intent(Menu.this, viewLecons.class);
        startActivity(intent);
    }

}
```

La leçon

Pour la leçon du jour, le nom de la leçon, ainsi que la leçon en question est stocké dans la base de données MySQL, et sont récupérés via un service web.
La leçon est un texte qui s'affiche et qui est lu via la synthèse vocal du téléphone.
2 boutons sont également présents :

- Le bouton « Repeat » qui permet de relancer la synthèse vocale
- Le bouton « Terminer » qui va permettre de mettre à jour la leçon sur la base MySQL, et va stocker la date de demain dans la base embarqué dans la table utilisateur.
  L'utilisateur ne pourra plus accéder à l'écran des leçons tant que la date du jour ne sera pas celle inscrite dans la base (ou ultérieur).

  Nous retrouvons également sur cet écran les dates prochaines de RDV toujours stocké dans MySQL et récupérer via un service web.

**Synthèse vocale*

```
public void vocale ()
{
    tts.setLanguage(Locale.FRANCE);
    tts.speak(leconEnCours.getText().toString(), TextToSpeech.QUEUE_FLUSH,
null);
}

@Override
public void onInit(int status)
{
    if (status == TextToSpeech.SUCCESS) {

        int result = tts.setLanguage(Locale.FRANCE);

        if (result == TextToSpeech.LANG_MISSING_DATA
                || result == TextToSpeech.LANG_NOT_SUPPORTED) {
            Log.e("TTS", "This Language is not supported");
        } else {
            vocale();
        }

    } else {
        Log.e("TTS", "Initilization Failed!");
    }
}
```

Les exercices

Les exercices sont stockés et récupérés via le service web, de MySQL vers le client.

L'exercice proposé au client est aléatoire en fonctionne de la leçon ou se trouve l'utilisateur.

Si l'utilisateur est à la leçon 3, alors les exercices récupérer pourront être des exercices allant de la leçon 1 à 3 aléatoirement.

Ce système est généré en SQL dans le web service associé.

```
$reponse = $bdd -> query("select e.exercice, e.reponse from t_exercice e,
t_utilisateur u, t_lecon l
                        where u.identifiant='$nom' and u.id_cours = l.id
and e.id_lecon between 1 and u.id_cours
                        order by ROUND( RAND() * u.id_cours) + 1 LIMIT 1
OFFSET 1");
```

Un bouton « Valider » permet de savoir si la réponse est bonne ou non grâce à un TextView en dessous du bouton.

Contact

Ici, nous pouvons envoyer un mail au responsable lié à l'utilisateur.

Son nom et son adresse mail sont déjà récupérés de MySQL via un web service et inscrit dans des TextView.

L'utilisateur devra noter le sujet du mail ainsi que le message avant de cliquer sur le bouton qui permettra d'ouvrir une application du téléphone pour envoyer le mail.

```
mail email = new mail(sujetMail.getText().toString(),
texteMail.getText().toString());

Intent i = new Intent(Intent.ACTION_SEND);
i.setType("message/rfc822");
i.putExtra(Intent.EXTRA_EMAIL  , new String[]{mail});
i.putExtra(Intent.EXTRA_SUBJECT, email.getSujetMail());
i.putExtra(Intent.EXTRA_TEXT   , email.getCorpsMail());
try
{
    startActivity(Intent.createChooser(i, "Send mail..."));
}
catch (android.content.ActivityNotFoundException ex)
{
    Log.e("Mail :"," erreur d'envoi");
}
```