Application Mobile

Présentation

Suite aux demandes des visiteurs, une application Android est en cours de développement. Elle doit permettre aux visiteurs de saisir en direct leurs frais (forfaitisés ou hors forfaits). L'application est une sorte de mémo qui permet d'enregistrer l'information à tout moment. Les visiteurs peuvent ensuite consulter leur mobile pour voir ce qu'ils ont enregistré et ainsi remplir le formulaire sur le site officiel.

Accessibilité

L'application est accessible sur leurs téléphones.

Forme de l'application

Une application android directement accessible sur téléphone.

Il manquait à l'application la partie comptable. Il a été donc à notre charge de l'ajouter. De ce fait, on a dû dans un premier temps modifier la connexion afin de permettre aux comptables de se connecter. Lorsqu'il sera connecté, il sera libre de parcourir la partie comptable du logiciel. A l'aide du menu de la partie comptable, il pourra réaliser deux ses actions :

- La validation des fiches de frais

- Le suivie du paiement des fiches des frais

Langages utilisés

- Java

- xml

Logiciels

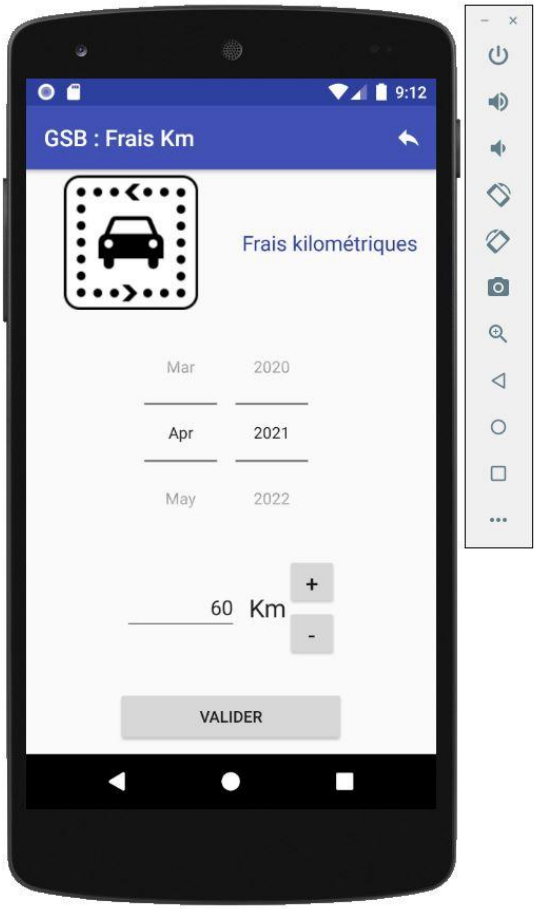
- Android Studio : IDE

- WampServer : simuler le serveur

- PhpMyAdmin : Base de données

Application

Interdiction de saisie directe des quantités



Il s'agissait ici d'empêcher l'apparition du clavier pour empêcher toute saisie manuelle du montant. Seules les commandes + et - devaient servir à modifier la quantité de frais.

Pour cela, l'ajout de la ligne `android:focusableInTouchMode="false"` permet de bloquer le caractère tactile du champ.

Ce code a été ajouté dans la vue correspondante.

```
<EditText
    android:focusableInTouchMode="false"
    android:id="@+id/txtKm"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:width="100dp"
    android:gravity="end|center_vertical"
    android:inputType="number">
    <requestFocus />
</EditText>
```

Enregistrement des autres catégories de frais forfaitisés

Ici, il était nécessaire de dupliquer la vue et l' "Activity" de la page Frais Kilométriques en 3 autres exemplaires pour la nouvelle page frais Nuitées, frais d'Étapes, ainsi que pour la page frais Repas.

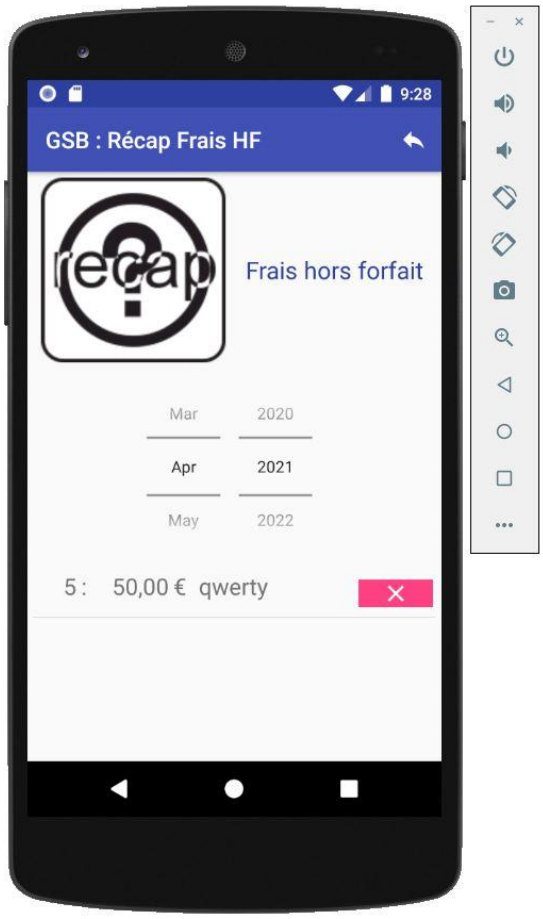
Il fallait par la suite, remplacer chaque champ par les informations correspondantes à la nouvelle page.

Ce fut une étape avec beaucoup de manipulation, mais peu de difficulté, c'est pour cela qu'il est difficile et peu intéressant de la développer ici.

Suppression de frais hors forfait

Suite à l'ajout d'un frais hors forfait dans l'interface dédié, le visiteur peut accéder à son historique la page de récapitulatif prévu à cet effet.

Il fallait permettre la suppression aussi bien visuel que enregistré.



Pour cela il fallait ajouter 3 lignes de codes dans la page FraisHfAdapter :

```
lesFrais.remove(index);
Serializer.serialize(Global.listFraisMois, inflater.getContext());
notifyDataSetChanged();
```

La première ligne permet la suppression de la ligne au niveau de l' "affichage".

La seconde permet la suppression de la ligne dans le Hashtable ou enregistrement.

La dernière notifie l'application d'un changement et produit une actualisation.

Code de la Classe :

```
/**
*Affichage dans la liste
*/
@Override
public View getView(final int index, View convertView, ViewGroup parent) {
  final ViewHolder holder ;
  if (convertView == null) {
     holder = new ViewHolder() ;
     convertView = inflater.inflate(R.layout.layout_liste, parent, false) ;
     holder.txtListJour = convertView.findViewById(R.id.txtListJour);
     holder.txtListMontant = convertView.findViewById(R.id.txtListMontant);
     holder.txtListMotif = convertView.findViewById(R.id.txtListMotif);
     convertView.setTag(holder) ;
  }else{
     holder = (ViewHolder)convertView.getTag();
  }
  holder.txtListJour.setText(String.format(Locale.FRANCE, "%d",
lesFrais.get(index).getJour()));
  holder.txtListMontant.setText(String.format(Locale.FRANCE, "%.2f",
lesFrais.get(index).getMontant())) ;
  holder.txtListMotif.setText(lesFrais.get(index).getMotif()) ;

      holder.cmdSuppHf = convertView.findViewById(R.id.cmdSuppHf);
  holder.cmdSuppHf.setOnClickListener(new View.OnClickListener(){
      public void onClick(View v){
        lesFrais.remove(index);
        Serializer.serialize(Global.listFraisMois, inflater.getContext()) ;
        notifyDataSetChanged();
          }
      });
      return convertView ;
}
```

Synchronisation avec la base de données distante

Tout d'abord, il est nécessaire d'établir le lien avec la base de données.

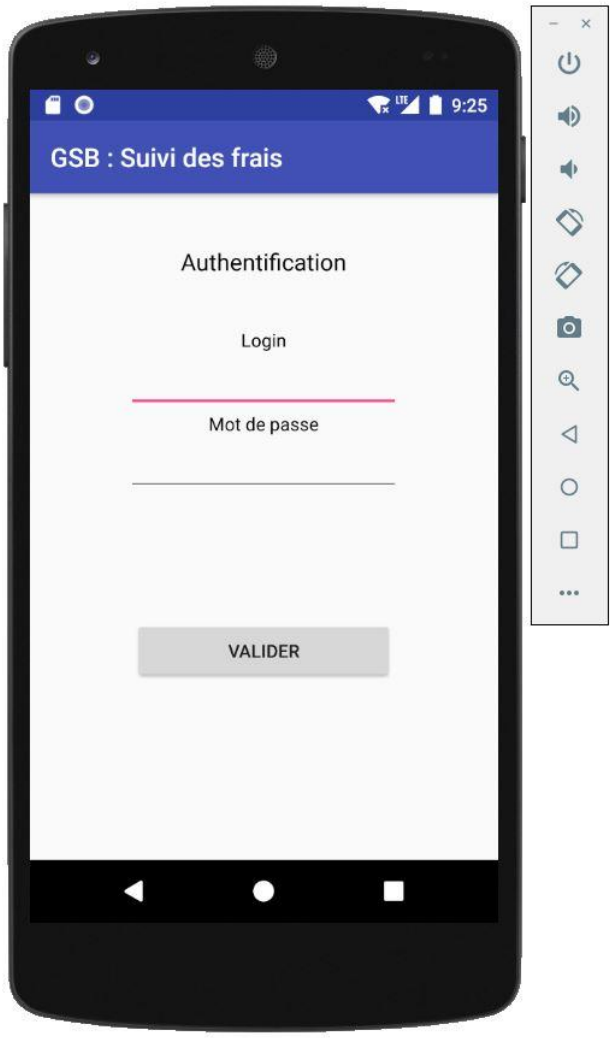
Nous avons alors créé Database.java.

```
public class Database {
   /**
    * se connecter à la base de donne distante
    * @return
    */
   public Connection get_connection(){
       Connection connection=null;
       try{
           Class.forName("com.mysql.cj.jdbc.Driver");

connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/gsb_fra
is","root","root");
       }catch (Exception e) {
           e.printStackTrace();
       }
       return connection;
   }
```

Pour synchroniser ses données à sa propre identité, le visiteur doit s'authentifier.

Pour ce faire, nous avons ajouté une page de connexion au démarrage de l'application.



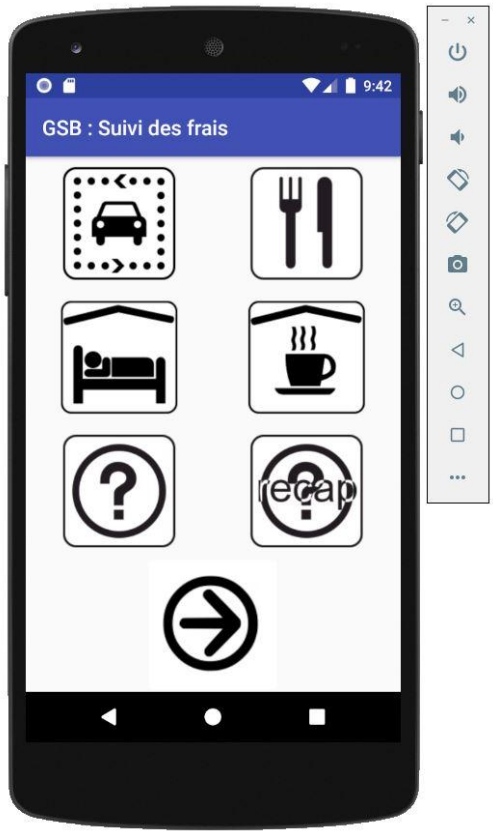
Le visiteur saisit son login ainsi que son mot de passe et voit ses informations vérifiées par la base de données.

```
public void connexion()
{
	Database obj_DB_Connection=new Database();
	Connection connection=obj_DB_Connection.get_connection();
```

```
        PreparedStatement ps=null;
        try {

            String query="SELECT visiteur.login,visiteur.mdp visiteur.id
FROM visiteur WHERE visiteur.login = ? ";
            ps=connection.prepareStatement(query);
            ps.setString(1,pseudo.getText().toString());
            ResultSet rs=ps.executeQuery();
            id_login = rs.getString("id").toString();
            if(rs.getString( "mdp" ).equals(pass.getText().toString()))
            {
                Nextactivity();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```

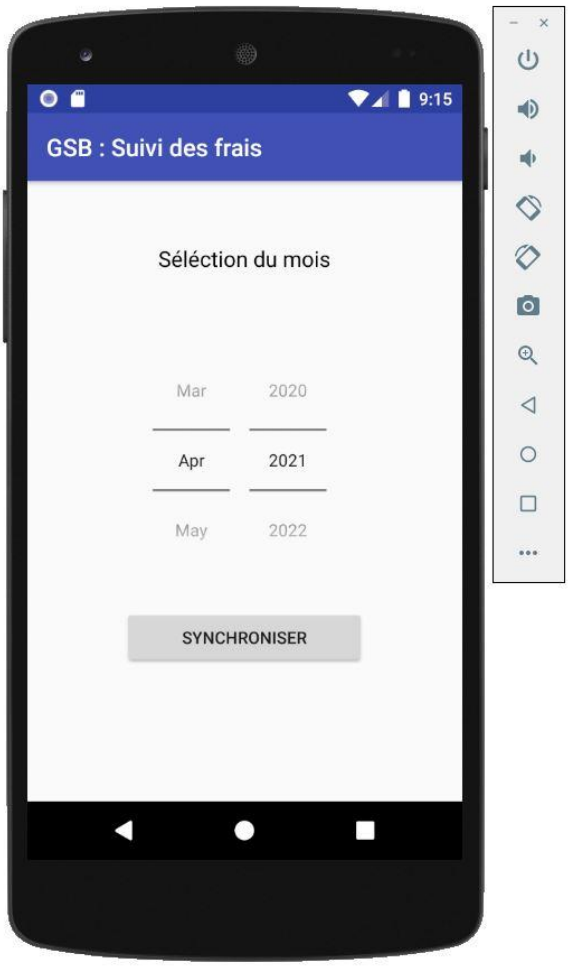
Après avoir saisi tous ses frais, le visiteur à la possibilité de synchroniser ses données grâce au bouton situé en bas de la page.



Pour des raisons techniques, nous avons pris la décision d'offrir une synchronisation par mois. Cela permet à l'utilisateur de synchroniser plus précisément ses informations.

Il a donc été nécessaire de créer une page appelée "activity_sync" dans laquelle se trouve un menu de sélection de période ainsi qu'un bouton de validation de synchronisation.

Cette page est accessible en appuyant sur le dernier bouton de la page d'accueil.



Le choix de la date est ici repris des pages activity précédemment modifiées.

Elle permet de créer une key pour pouvoir par la suite, synchroniser la bonne période.

```
anneesync = ((DatePicker)findViewById(R.id.datSync)).getYear() ;
moissync = ((DatePicker)findViewById(R.id.datSync)).getMonth() + 1 ;
Integer key = anneesync*100+moissync ;
```

Deux synchronisations sont alors à réaliser.

La première concerne les frais forfait du mois sélectionné. La seconde, quant à elle, synchronise les frais hors forfait du mois sélectionné.

- Synchronisation Frais forfait :

```
public void sendforfait(String mois )
{
   Database db = new Database();
   ConnectionActivity cn = new ConnectionActivity();
   String id_login = cn.id_login;
   Integer km ;
   Integer rep;
   Integer nui;
   Integer etp;
   Integer key = anneesync*100+moissync ;

   km = Global.listFraisMois.get(key).getKm();
   rep =Global.listFraisMois.get(key).getRepas();
   nui =Global.listFraisMois.get(key).getNuitee();
   etp =Global.listFraisMois.get(key).getEtape();
   if(!db.verifForfait(id_login,mois))
   {
       db.addfraisForfait(km,rep,nui,etp,id_login,mois);
   }
   else
   {
```

```
        majforfait(mois);
    }
}
```

Lors du clic sur le bouton synchroniser, cette fonction est exécutée si aucun frais forfait n'est présent dans la base de données au mois séléctionné et appelle une seconde fonction contenu dans Database.java :

```
public void addfraisForfait(int km,int rep, int nui, int etp ,String
idvisiteur,String mois )
{
    Database obj_DB_Connection=new Database();
    Connection connection=obj_DB_Connection.get_connection();
    try {
        PreparedStatement pa = connection.prepareStatement("INSERT INTO
fichefrais (idvisiteur,mois,nbjustificatifs,montantvalide,datemodif,idetat)
VALUES (?,?,0,0,now(),'CR');"+ " ");
        PreparedStatement ps = connection.prepareStatement("INSERT INTO
lignefraisforfait (idvisiteur,mois,idfraisforfait,quantite) VALUE(?, ?, ?,
10);"+ " ");
        pa.setString(1,idvisiteur);
        pa.setString(2,mois);
        pa.executeUpdate();
        ps.setString(1,idvisiteur);
        ps.setString(2,mois);
        int p1 = 0;
        for(int i=0 ;i<4 ;i++)
        {
            if(i==0){p1=km;}
            if(i==1){p1=rep;}
            if(i==2){p1=nui;}
            if(i==3){p1=etp;}
            ps.setInt(3,p1);
            ps.executeUpdate();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Si les frais hors forfait sont déjà existant, cette fonction sera alors exécuté :

```
public void majforfait(String mois)
{
   Database db = new Database();
   ConnectionActivity cn = new ConnectionActivity();
   String id_login = cn.id_login;
   Integer km ;
   Integer rep;
   Integer nui;
   Integer etp;
   Integer key = anneesync*100+moissync ;
   km = Global.listFraisMois.get(key).getKm();
   rep =Global.listFraisMois.get(key).getRepas();
   nui =Global.listFraisMois.get(key).getNuitee();
   etp =Global.listFraisMois.get(key).getEtape();
   db.majfraisHDDforfait(km,id_login,mois,"KM");
   db.majfraisHDDforfait(etp,id_login,mois,"ETP");
   db.majfraisHDDforfait(rep,id_login,mois,"REP");
   db.majfraisHDDforfait(nui,id_login,mois,"NUI");
}
```

Cette fonction appelle alors une seconde fonction de Database.java :

```
public void majfraisHDDforfait(int qte ,String idvisiteur,String
mois,String p )
{
   Database obj_DB_Connection=new Database();
   Connection connection=obj_DB_Connection.get_connection();
       try {
```

```
            PreparedStatement ps = connection.prepareStatement("UPDATE
lignefraisforfait SET lignefraisforfait.quantite = ? WHERE
lignefraisforfait.idvisiteur = ? AND lignefraisforfait.mois = ? AND
lignefraisforfait.idfraisforfait = ? ");
            ps.setInt(1,qte);
            ps.setString(2,idvisiteur);
            ps.setString(3,mois);
            for(int i=0; i<4;i++)
            {
                if(i==0)
                {
                    p="KM";
                }
                if(i==1)
                {
                    p="REP";
                }if(i==2)
            {
                p="NUI";
            }if(i==3)
            {
                p="ETP";
            }
                ps.setString(4,p);
                ps.executeUpdate();
            }


        } catch (Exception e) {
            System.out.println(e);
        }
}
```

- Synchronisation Frais hors forfait :

Le clic sur le bouton synchroniser déclenche alors la fonction suivante :

```
public void sendHorsforfait(String anne , String mois )
{
   Database db = new Database();
   ConnectionActivity cn = new ConnectionActivity();
   String id_login = cn.id_login;
   String date;
   k = 0 ;
   Integer key = anneesync*100+moissync ;
   while (!Global.listFraisMois.isEmpty()) {
       joursync =
Global.listFraisMois.get(key).getLesFraisHf().get(k).getJour() ;
       motifsync =
Global.listFraisMois.get(key).getLesFraisHf().get(k).getMotif() ;
       montantsync =
Global.listFraisMois.get(key).getLesFraisHf().get(k).getMontant() ;
       date
=joursync.toString()+"-"+moissync.toString()+"-"+anneesync.toString();

db.addhorforfait(id_login,moissync.toString(),motifsync.toString(),date,mon
tantsync);
       Global.listFraisMois.get(key).getLesFraisHf().remove(k);
       k++;
   }
}
```

Elle récupère toutes les données et les ajoutes à la base de données en appelant la fonction suivante :

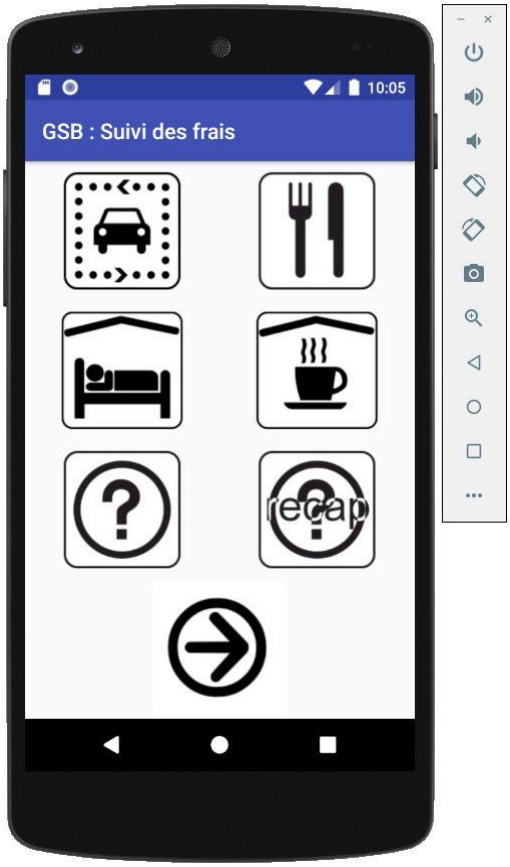
```
public void addhorforfait(String idvisiteur , String mois,String libelle
,String date ,double montant )
{
   Database obj_DB_Connection=new Database();
```

```
   Connection connection=obj_DB_Connection.get_connection();
   try {

       PreparedStatement pa = connection.prepareStatement("INSERT INTO
fichefrais (idvisiteur,mois,nbjustificatifs,montantvalide,datemodif,idetat)
VALUES (?,?,0,?,now(),'CR');"
               + " ");
       PreparedStatement ps = connection.prepareStatement("INSERT INTO
lignefraishorsforfait VALUES (null,?, ?,?, ?, ?);"
               + " ");
       pa.setString(1,idvisiteur);
       pa.setString(2,mois);
       pa.setDouble(3,montant);


       ps.setString(1,idvisiteur);
       ps.setString(2,mois);
       ps.setString(3,libelle);
       ps.setString(4,date);
       ps.setDouble(5,montant);
       pa.executeUpdate();
       ps.executeUpdate();
   } catch (Exception e) {
       e.printStackTrace();
   }
}
```

Après cette synchronisation, le visiteur sera redirigé vers la page d'accueil.



Compétences

- Analyse du cahier des charges d'un service à produire

- Étude des exigences liées à la qualité attendue d'un service

- Conception ou adaptation d'une base de données

- Gestion d'environnements de développement et de test