

Rapport Projet Développement mobile

Étienne Baillif, Fabrice Foudrin L3 informatique

1^{er} mai 2021

Résumé

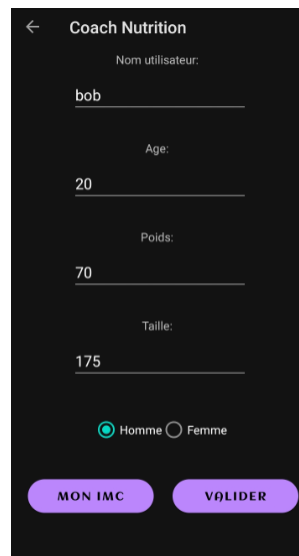
Dans ce rapport nous allons vous décrire notre application dans sa globalité ainsi que les difficultés que nous avons rencontrées lors de sa conception.

1 Introduction

Dans le cadre de l'UE développement mobile, nous avons un mois pour développer en binôme une application mobile sur les systèmes android et IOS dans le but de mettre n'oeuvre ce que nous avons vue pendant les heures de cours et d'acquérir une certaine autonomie d'apprentissage en programmation mobile. Premièrement nous allons effectuer une description générale de l'application puis nous allons décrire l'architecture du code, enfin on va parler des difficultés que l'on a rencontré lors de la conception de l'application ainsi qu'une conclusion de notre rapport.

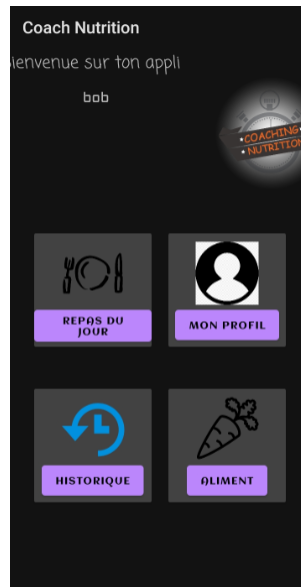
2 Description générale de l'application

Notre projet est une application de coach nutritionnel, elle permet à l'utilisateur d'avoir un suivi des aliments qu'il consomme par jour ainsi que de l'informer sur son état de santé actuelle via un calcul d'indice de masse grasseuse (IMG). Notre application se repose sur cinq activity. Lors du premier lancement de l'application, l'écran d'accueil habituel ne s'affichera pas, mais vous trouverez à la place l'écran "Mon profil". Il vous demande de remplir quelques champs :

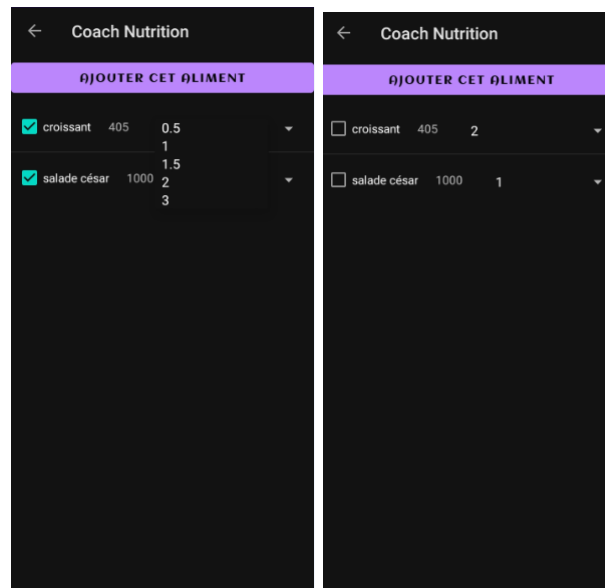


Vous pouvez maintenant confirmer vos informations vous auriez plus tard la possibilité de les modifier

Ecran principal : sur l'écran principal il y a 4 boutons qui sont disponibles. Dans l'ordre :



1. Le bouton de renseignement de repas "Repas du jour" :



Il vous amène sur l'écran de sélection de nourriture, et vous permet de sélectionner ce que vous avez mangé durant votre repas. Vous cochez la nourriture correspondante, et vous indiquez également la quantité consommée, par défaut, vous en mangez une portion.

2. Le bouton du Profil "Mon Profil" :

Coach Nutrition

Nom utilisateur:

bob

Age:

20

Poids:

70

Taille:

175

☒ Homme ☐ Femme

MON IMC VÁLIDER

-11.3: IMG trop faible

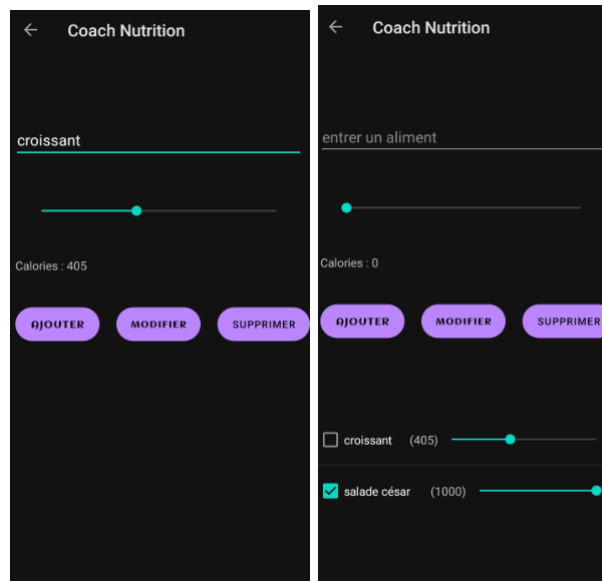
Vous trouverez ici les renseignements fournis au lancement de l'application et vous avez la possibilité de les modifier. De plus cette activity contient un bouton "mon img" qui permet de calculer l'indice de masse graisseuse selon les informations saisies dans "mon Profil"

3. Le bouton de suivi alimentaire "Historique" :

Coach Nutrition	
Thu Apr 29 16:41:28 GMT+04:00 2021	1405.0
SUPPRIMER	
Thu Apr 29 16:41:39 GMT+04:00 2021	1000.0
SUPPRIMER	

Vous pouvez ici voir vos repas que vous avez consommés présentés avec la Date et l'heure correspondante et le nombre total de calories du repas.

4. Le bouton pour ajouter dans la base de données des aliments "Les Aliments" :

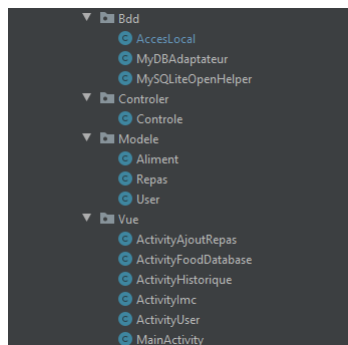


Ici, vous pouvez ajouter un aliment dans la base de données, ou modifier les aliments présents. Vous avez besoin du nom de l'aliment et des calories pour une portion qui vous correspond. Par exemple, si vous consommez 100g de pates, vous devriez entrer le nombre de calories correspondant. Vous appuyez ensuite sur Ajouter pour l'enregistrer dans l'application. Pour modifier un aliment, faites glisser la seekbar de calories. Vous pouvez modifier plusieurs éléments à la fois. Appuyez ensuite sur Modifier. Enfin, vous pouvez supprimer un ou plusieurs éléments en cochant les aliments désirés et en appuyant sur "Supprimer".

L'application doit assurer la persistance des données, pour cela nous avons utilisé une base de données local basé sur SQLite. Nous décrirons le code dans la partie suivante.

3 Architecture du code

Nous avons construit notre application en suivant le modele modele vue controller (MVC).



Le fonctionnement général de l'appli se base sur ce pattern, il permet d'avoir une meilleure organisation dans notre code grâce à une répartition de la logique du code en 3 parties.

Premièrement le package modèle qui contient les données manipulées par le programme et la partie logique (calcul). Ensuite le package vue qui affiche les données qu'il a récupérées auprès du modèle.

Enfin le package contrôleur qui va faire le lien entre les modele et la vue, et permet l'accès aux différentes méthodes de la base de données.

Nous avons décidé ajouté un package bdd pour inclure les classes concernant la base de données, qui contient les méthodes créer, lire, modifier supprimer (CRUD) et aussi la classe MySqlConnectionHelper qui hérite de SQLiteOpenHelper qui va permettre de créer la base de données. Nous avons le dossier res qui va contenir les layouts pour les interfaces graphiques ainsi que les images que nous avons utilisées.

Nous allons vous décrire plus précisément ce que contiennent les packages :

1. La classe Controle :

Elle contient les méthodes permettant d'instancier des objets du package modele et de préparer leurs insertions dans la base de données en utilisant les méthodes de la classe AccesLocal (décrite ci-dessous). Elle est la particularité de pouvoir être instancié uniquement par la méthode getInstance() qui est un pattern singleton (une seule instantiation). Controle est la classe centrale de l'application, elle permet de faire le lien entre la classe de base de données accesLocal ainsi que les différentes activity du package vue . Toutes ces devront utiliser getInstance() pour avoir acces aux méthodes CRUD (Create Read Update Delete).

```
public static final Controle getInstance(Context contexte) {
    if (Controle.instance == null) { //si null alors créer instance

        Controle.instance = new Controle(); //n'est pas accessible de l'extérieur
        accesLocal = new AccesLocal(contexte); //accède à la bdd
    }
    return Controle.instance;
}

public void creerAliment(String nom, int nbcalories){
    int id = 0;
    aliment = new Aliment(id,nom, nbcalories);
    accesLocal.ajoutAliment(aliment);
}
```

La classe controle ne contient pas la logique de l'application comme l'indique le pattern mvc, elle est déportée dans le modele et dans notre cas dans la classe accesLocal qui permet d'effectuer toutes les opérations. La méthode ci-dessus permet par exemple d'instancier un Aliment et de l'insérer dans la base de données par la méthode ajoutAliment(Aliment) faisant appel à un objet de la classe AccesLocal qui a été initialisé dans la classe Controle.

```
public List<Aliment> loadAliment(){
    List<Aliment> allAliment = accesLocal.getAllAliments();
    return allAliment;
}
```

2. Package Modele :

Il contient les classes Aliment , Repas, User. Ce sont des classes basiques avec un constructeur, des propriétés ainsi que les getters et setters pour avoir accès aux méthodes de l'extérieur. La classe Repas a une méthode getTotalCalories() qui effectue la somme des calories en récupérant les paramètres nécessaires par l'un de ses constructeurs, de plus la classe User possède une méthode calculImg() qui calcule l'indice de masse grasseuse de l'utilisateur.

```
public double getTotalCalories() {

    int premierId = lesAliments.get(0).getId();
    for (Aliment unAliment : lesAliments){
        if(premierId == unAliment.getId()){
            totalCalories = unAliment.getCalories()*selectQte;
        }else{
            totalCalories = totalCalories + (unAliment.getCalories() * selectQte);
        };
    }
    return totalCalories;
}

private void calculIMG(){
    //taille en metre
    float tailleM = ((float) taille/100);
    this.img = (float)((1.2 * poids / (tailleM*tailleM)) + (0.23*age) - (10.83*sexe) - 5.4);
}
```

3. Package Vue : Ce package inclut les classes qui héritent de AppCompatActivity, elles servent à faire le lien entre les objets l'interface graphique en XML et le code java. MainActivity est la page de lancement de l'application, cette classe va instancier une première fois un contrôle pour avoir accès à verifuserexistant() qui va vérifier si un utilisateur est déjà enregistré dans la base, elle fonctionne de cette manière : s'il n'y en a pas on redirige grace à la classe Intent vers la page qui permet la création d'un user, sinon on lance la page actuelle c'est-à-dire la page d'accueil.

```
if (controle.verifUserExistant() == false) {
    Intent intent = new Intent(this, ActivityUser.class);
    startActivity(intent);
} else if(controle.verifUserExistant() == true) {...}
```

La logique qu'on a utilisée pour concevoir ces classes est pratiquement la même. On déclare les propriétés avec une ArrayList<ObjetConcerné> qui va contenir les éléments à afficher dans une liste, un objet Controle, des méthodes qui utilisent la classe Controle pour charger des éléments de la base de données dans une liste ainsi qu'une classe ArrayAdapter pour afficher les données dans une liste et pour l'écoute des événements sur les checkbox et les spinners.

```
public void loadAllFood() {
    alimentsDispo = new ArrayList < Aliment > (controle.loadAliment());
    //affichage des aliments dans la listView
    final ActivityAjoutRepas.Foo;
    lesRepasDisponibles.setAdapter(adapter);dCustomAdapter2 adapter = new ActivityAjoutRepas
}

}
```

4. Package base de données

Ce package contient une classe MySQLOpenHelper qui hérite de la classe abstraite SQLiteOpenHelper qui permet de créer et gérer une base locale. Lorsqu'elle sera instanciée, la méthode onCreate() est appelée pour créer les tables de la base via des requêtes SQL qu'on a enregistrées dans des constantes.

```
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    sqLiteDatabase.execSQL(CREATE_TABLE_USER);
    sqLiteDatabase.execSQL(CREATE_TABLE_FOOD);
    sqLiteDatabase.execSQL(CREATE_TABLE_REPAS);
}
```

Ensuite on a créé classe Acceslocal qui elle va instancié un objet de MySQLLiteOpenHelper pour effectuer l'insertion, suppression, modification et la récupération des données.

```
private User cursorToUser(Cursor curseur) {
    int IdUser = getLastID("user", "idUser");
    String nom = curseur.getString(1);
    . . .
    User user = new User(IdUser, nom, age, poids, taille, sexe);
    return user;
}

public List<User> getAllUser() {
    bd = accesBD.getWritableDatabase();
    List<User> lesprofils = new ArrayList<User>();
    Cursor curseur = bd.query(MySQLiteOpenHelper.TABLE_USER,
        allUserColumns, null, null, null, null, null);
    curseur.moveToFirst();
    while (!curseur.isAfterLast()) {
        User unuser = cursorToUser(curseur);
        lesprofils.add(unuser);
        curseur.moveToNext();
    }
    curseur.close();
    return lesprofils;
}
```

Dans le code précédent qui retourne la liste des utilisateurs, on ajoute dans une liste des objets du type User qu'on va créer par la méthode cursorToUser(). Cette méthode prend en paramètre un Cursor qui permet de parcourir la table. CursorToUser() retourne un objet User et initialise ses propriétés avec les champs correspondant dans la table Utilisateur. Nous avons aussi utilisé des requêtes SQL lorsqu'on voulait supprimer ou modifier un utilisateur par son identifiant par exemple.

```
public void deleteRepas(int id) {
    bd = accesBD.getWritableDatabase();
    String req = "Delete FROM repas where idRepas =" + id;
    bd.execSQL(req);
}
```

4 Difficultés

De première vue l'application est tout à fait fonctionnelle , mais nous sommes passé par plusieurs stades ou nous avons dû faire face à des problèmes et en voici une partie :

- La mise en place du projet en commun sur github(soucis avec le push/pull au master ainsi qu'aux différentes branches)
- La mise en place d'une architecture MVC
- Certaines fonctions (exemple : celle qui récupère chaque aliment mangé dans un repas)
- La contrainte/gestion du temps.
- Adaptation de chaque view afin qu'elle s'adapte a chaque appareil (en mode portrait et paysage).
- Récupération de chaque aliment qui est inclus dans un Repas
- Développement de la version IOS

Notre application est plus au moins bien structuré grace au pattern MVC qui nous permet d'appliquer le principe de séparation des responsabilités, en l'occurrence celles du traitement de l'information et de sa mise en forme.

5 Conclusion

Tout d'abord en développement android où l'on a appris à gérer la persistance des données, la gestion d'une liste et la partie conception graphique via les contraintes et le code XML. Ensuite ce projet nous a permis de découvrir le versioning avec git, qui est un outil indispensable lors d'un projet collaboratif. Enfin il nous a permis d'acquérir une certaine autonomie en utilisant des ressources sur internet tel que la documentation android. En plus d'être un projet pédagogique il est aussi ludique et nous a donné beaucoup de liberté dans le code et dans la conception.