

Aprahamian Kaïl

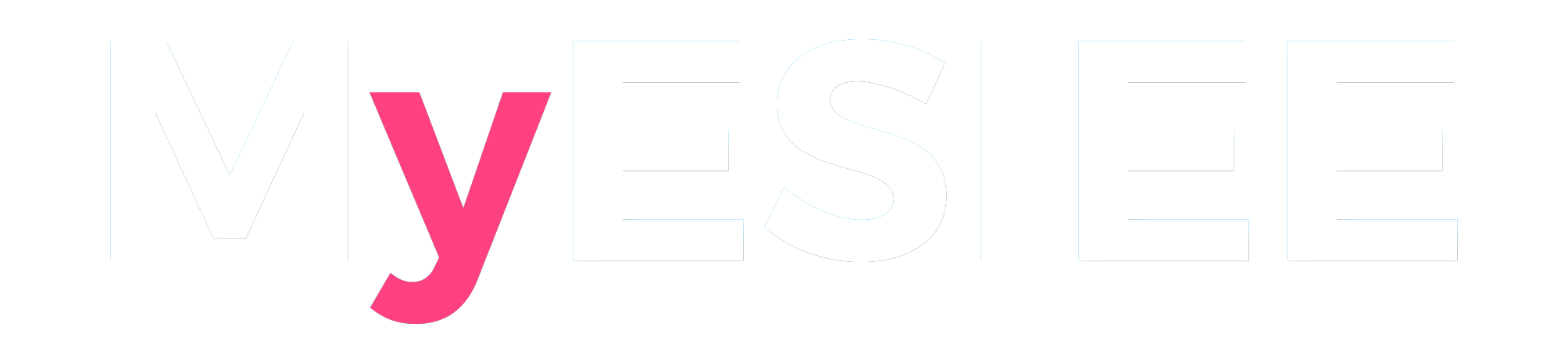
Dupont Léo

Kuhn Johann

Houacine Mehdi

Projet de troisième année

Mai – juin 2015



[Introduction 3](#_Toc422532208)

[Base de données 4](#_Toc422532209)

[Structure de la BDD 4](#_Toc422532210)

[Récupération des données et remplissage de la BDD 4](#_Toc422532211)

[Synchronisation de la base de données avec Android 5](#_Toc422532212)

[Mise à jour des données 5](#_Toc422532213)

[Synchronisation des données 5](#_Toc422532214)

[Version de la base de données 6](#_Toc422532215)

[Aurion 6](#_Toc422532216)

# Introduction

# Base de données

## Structure de la BDD

*Cette partie a été réalisée par Johann Kuhn*

La base de données est composée de quatre tables :

## Récupération des données et remplissage de la BDD

Les données des tables salle et prof proviennent en très grande partie de documents récupérés auprès du service de la planification de l’école. Il s’agit de documents Excel regroupant des informations sur toutes les salles de l’école (épi, étage, numéro de bureau et service rattaché entre autre, mais surtout le type de salle, la surface occupée et le nom de l’enseignant à qui appartient ce bureau s’il s’agit d’un bureau).

Cependant, ces documents n’étaient pas suffisamment complets pour notre usage (présence d’un projecteur, d’une imprimante, type de tableau notamment y étaient absents) et ces documents étaient assez anciens : ils n’avaient pas été mis à jours depuis novembre 2014.

Cette partie a été réalisée par Mehdi Houacine

Nous avons choisi de tenir à jour notre propre document Excel avec ces informations manquantes sur les salles que nous sommes allé vérifier par nous-mêmes (notamment Johann) dans presque toutes les salles de l’école sur une durée approximative d’une semaine.

Suite à cela, j’ai réalisé en Python une extraction de ces données (sur les salles et sur les enseignants) en les convertissant sous un format de données standard exploitable, le .csv. Une fois toutes ces données stockées dans une structure Python, je les ai stockés dans notre base de données grâce à une bibliothèque Python permettant des interactions avec une base de données MySQL : mysql-connector. Voici des exemples de code assez simples provenant de la documentation de cette libraire montrant comment créer une table en Python dans une base de données par le biais d’instruction SQL standard : <http://dev.mysql.com/doc/connector-python/en/connector-python-example-ddl.html>.

Pour les enseignants, la table prof est constituée d’un numéro de bureau provenant des fichiers Excels cités plus tôt, leur nom provient d’une liste de noms extraites de ces documents Excel et d’ADE par le biais de sa Web API, et leur adresse mail est générée par un simple script Python selon l’expression suivante : les sept premières lettres du nom de famille suivies de la première lettre du prénom ou bien "nom.prénom" si la taille du nom est inférieure à 7 lettres puis "@esiee.fr".

Nous sommes conscients au vu de l’ancienneté de ces données qu’une minorité d’entre elles est obsolète, pour cela nous avons donc prévu un module de participation sur l’application et le site Web afin que les étudiants qui l’utilisent puisse nous signaler les données erronées ou manquantes. Ces appréciations sont consignées dans une table de la base de données jusqu’à ce que nous les corrigions.

## Synchronisation de la base de données avec Android

Cette partie a été réalisée par Léo Dupont

L’application Android utilise une copie de notre base de données principale afin de limiter la taille des requêtes HTTP et de proposer des prédictions pour les noms des salles et des profs (voir la partie Android).

### Mise à jour des données

Afin de garder synchrones la base de données SQLite Android et la base de données MySQL du serveur, j’ai suivi ce tutoriel dans les grandes lignes : <http://programmerguru.com/android-tutorial/how-to-sync-remote-mysql-db-to-sqlite-on-android/> bien qu’il ait nécessité beaucoup de changement afin de l’adapter au projet. De plus, nous avons ici deux tables à synchroniser et non une seule.

De plus, contrairement au tutoriel, nous ne pouvons pas stocker dans la base de données du serveur une information indiquant sur telle ligne a été synchronisée sur Android puisque notre application est destinée à être utilisée par plusieurs appareils (voir la partie suivante concernant la résolution de ce problème).

J’ai donc créé des scripts PHP permettant d’obtenir toutes les données des tables salle et prof au format JSON, en particulier le script getData.php qui est utilisé par l’application : <https://mvx2.esiee.fr/mysql_sync/getdata.php?table=salle> (pour récupérer la table salle).

### Synchronisation des données

Afin de savoir si le contenu de la base de données d’un appareil Android correspond au contenu de la base de données du serveur, j’ai écrit un script PHP permettant d’obtenir un hash en SHA-256 de tout le contenu des tables salle et prof. De cette façon, il est facile de comparer le contenu de deux bases de données sans faire transiter un grand nombre de données par internet. Ce script est stocké dans le fichier bdd.php et accessible par cette URL :

<https://mvx2.esiee.fr/api/bdd.php?func=getHashVersion>

Le hash obtenu est stocké dans la mémoire du mobile à chaque fois qu’une mise à jour est effectuée. Il suffit alors de comparer le hash de version stocké dans l’appareil avec le hash de version obtenu par cette requête.

### Version de la base de données

Afin de pouvoir afficher dans l’écran “À propos” de l’application la date de dernière mise à jour de la base de données principale, j’ai ajouté la fonction getLastUpdate au script bdd.php, accessible par cette URL :

<https://mvx2.esiee.fr/api/bdd.php?func=getLastUpdate>

Ce script se contente de récupérer la valeur de la clé db\_last\_update stockée dans la table infos.

# Aurion

# ADE

Pour déterminer si une salle est libre ou non, nous avons choisi de nous baser sur l’emploi du temps de l’ESIEE, lequel est géré par le logiciel ADE.

## La Web API d’ADE

Contrairement à Aurion, ADE propose une API Web étoffée, permettant d’obtenir n’importe quelle information disponible sur cette plateforme relativement facilement. Nous avons obtenu la documentation de cette API grâce à M. Bruno Rougier.

Cette API fonctionne par requêtes HTTP de type GET et fournit des réponses au format XML. Les requêtes sont de cette forme : <https://planif.esiee.fr:8443/jsp/webapi?function=xxx>. Il s’agit à chaque fois de préciser la bonne fonction et les paramètres qui l’accompagnent. En général, il est nécessaire d’effectuer plusieurs requêtes d’affilée pour obtenir l’information désirée. Voici, par exemple, comment obtenir la liste des cours utilisant la salle 5201V le 19/06/2015 :

|  |
| --- |
| 1. Connexion à une session. Afin de récupérer un sessionID :   ?function=connect&login=lecteur1&password=   1. Choix du projet ADE. L’ESIEE crée un nouveau projet par année, le projet de l’année 2014-2015 est le numéro 4 :   ?sessionId=14e0dae2bb6&function=setProject&projectId=4   1. Récupération du resourceID lié à la salle 5201V :   ?sessionId=14e0dae2bb6&function=getResources&name=5201V&detail=0   1. Récupération des événements planifiés le 19/06/2015 utilisant la ressource 659 :   ?sessionId=14e0dae2bb6&function=getEvents&resources=659&detail=0&date=06/19/2015  Ceci nous retourne une liste d’éléments XML représentant chacun un cours planifié et ayant, entre autres, deux attributs intéressants de la forme : endHour="10:00" startHour="08:30".   1. Déconnexion de la session :   ?sessionId=14e0dae2bb6&function=disconnect |

## Notre API personnalisée

Comme nous le voyons, cette API permet de récupérer les informations souhaitées mais nécessite beaucoup de requêtes et d’interprétations des réponses XML. De plus, il n’est pas possible de rechercher les salles par des caractéristiques telles que la présence d’un projecteur ou d’une imprimante. Il nous a donc fallu créer notre propre API capable d’accepter les paramètres dont nous avons besoin (comme les caractéristiques d’une salle), de coupler les informations de notre base de données avec celles d’ADE, d’effectuer un algorithme pour déterminer les disponibilités d’une salle donnée, et de nous renvoyer la réponse dans un format clair et léger.

Pour cela j’ai créé une classe PHP ADE munie de toutes les fonctions nécessaires pour effectuer des requêtes vers notre base de données et vers ADE, ainsi que de choisir les salles concernées par nos critères de recherche. Cette classe permet d’obtenir une liste de salles avec leurs disponibilités au format JSON, mais également de générer une image de l’emploi du temps d’une salle ou d’un professeur à une date donnée (cette image est générée par la Web API d’ADE grâce à la fonction imageET). Voici le contenu du fichier readme.md que j’ai écrit pour accompagner cette API :

### Utilisation de la fonction rechSalle

Cette fonction permet d'obtenir une liste de salles répondant à certains critères optionnels ainsi que leur disponibilité.

La réponse au format JSON est de cette forme : [{"5004":"45"}, ...], où "5004" est le nom de la salle et "43" la disponibilité.

#### Format de l'URL et critères de recherche

Une requête peut être de la forme :

<https://mvx2.esiee.fr/api/ade.php?func=rechSalle&nom=5004&type=it&taille=m&projecteur=0&tableau=1&imprimante=0>

Voici la liste des paramètres possibles :

* func=rechSalle : pour utiliser la fonction de recherche de salles (seul paramètre obligatoire).
* nom : le nom complet de la salle en BDD. Si au moins un des paramètres epi ou etage est spécifié, le paramètre nom ne sera pas pris en compte.
* type : le type de salle recherchée (it, elec ou banal).
* taille : Peut prendre les valeurs S, M ou L (majuscule ou minuscule). Correspond à la taille de la salle, respectivement petite, moyenne et grande.
* projecteur : la présence d'un projecteur (0 : non, 1 : oui).
* tableau : la présence de tableau(x) (0 : aucun, 1 : blanc, 2 : noir, 3 : les deux).
* imprimante : la présence d'une imprimante (0 : non, 1 : oui).
* epi : l'épi de la salle (correspond au premier chiffre des noms des salles).
* etage : l'étage de la salle (correspond au deuxième chiffre des noms des salles).

#### Format de la disponibilité d'une salle

La disponibilité d'une salle peut prendre ces valeurs :

- "-1" si la salle n'est pas disponible actuellement.

- "0" si la salle est disponible jusqu'à la fin de la journée.

- un autre entier correspondant au nombre de minutes durant lesquelles la salle est encore libre. Par exemple, si à 14h15, une salle a une disponibilité de 45, cela signifie qu'elle est actuellement libre mais qu'elle sera occupée à 15h00.

### Utilisation de la fonction dispoSalle

Cette fonction permet d'obtenir une image au format GIF de l'emploi du temps d'une salle à un jour donné.

#### Format de l'URL et paramètres

Une requête peut être de la forme :

<https://mvx2.esiee.fr/api/ade.php?func=dispoSalle&nom=5004&date=06/18/2015>

Les paramètres func=dispoSalle et nom sont obligatoires. Le format du nom est le même que pour la fonction rechSalle.

Le paramètre date correspond à la date du jour souhaité au format américain "mm/jj/aaaa" (exemple : 06/18/2015 pour le 18 juin 2015). S'il est omis, la date d'aujourd'hui sera utilisée.

Les paramètres largeur et hauteur correspondent aux dimensions en pixels de l'image à générer.

### Utilisation de la fonction dispoProf

Cette fonction permet d'obtenir une image au format GIF de l'emploi du temps d'un professeur à un jour donné. Cette fonction s’utilise comme la fonction dispoSalle, avec func=dispoProf et le paramètre nom faisant référence au nom d’un professeur enregistré en base de données.

## Conclusion

Grâce à cette API personnalisée, basée elle-même sur la Web API d’ADE, nous pouvons obtenir facilement les informations nécessaires à notre projet à partir de la version Web comme de l’application Android avec une simple URL. De plus, cela garantit que les informations affichées sur le site Web et sur l’application seront les mêmes et qu’une modification du script PHP influera sur les deux versions du produit.