Annexes

Table des matières

1. Cahier des charges

Travail pratique individuel (TPI)

« Formation Informaticien »

Cahier des charges

Nom du candidat :	**Murat Bayrakci**
Supérieur professionnel :	**Plinio Sacchetti**
Experts :	**Principal : Nikles Jérôme Auditeur : Rota Renato**
Sujet à traiter :	**Hypercool reader** **Lecteur d'horaire de classe**
Durée du travail :	**Du 13.03.2017 au 11.05.2017** **80 heures = 110 périodes**
Lieu de travail :	**CPLN-ET Salle B106B (et B114A)**

« Le travail du candidat reste propriété du maître d'apprentissage »

	Lu et approuvé le :	Signature:
Le candidat	13.03.2017	
Le supérieur professionnel	07.03.2017	
L'expert principal	07.03.2017	

Procédure :

Le candidat réalise un travail personnel sur la base d'un cahier des charges reçu le premier jour de l'épreuve. Le cahier des charges est approuvé par les experts désignés. Il est en outre présenté, commenté et discuté avec l'élève. Par sa signature, l'élève accepte le travail proposé. L'élève a connaissance de la grille d'évaluation avant de débuter le travail. Il est entièrement responsable de la sécurité de ses données. En cas de graves problèmes, le candidat avertit l'enseignant responsable au

plus vite. A la fin du temps de travail imparti, le candidat remet son rapport de travail en 3 exemplaires originaux papiers et 3 exemplaires sur support informatique.

L'ensemble du projet sera documenté par :

- Une page de titre
- Une table des matières
- Une introduction décrivant le sujet
- Un planning du projet
- Un journal de travail, mis à jour quotidiennement, qui décrira les diverses étapes et activités liées au projet :
  - date et nombre d'heures de travail consacrées au projet (pour chaque jour)
  - travaux effectués (on évitera une énumération trop détaillée et une consignation « minute par minute ») problèmes rencontrés
  - toute aide extérieure reçue
  - des événements particuliers comme les modifications de l'énoncé du travail (discuté en accord avec le supérieur professionnel), les interruptions dans le travail, les imprévus, etc.
- Une documentation de développement permettant d'effectuer la maintenance du projet
- Une documentation utilisateur permettant d'utiliser le projet
- Une conclusion
- Une bibliographie
- Des annexes (cahier des charges, documents de références, etc.).

Sujet :
Hypercool Reader – l'horaire en ligne.

Environnement de développement :
1 à 2 ordinateurs (desktop, laptop) ; éventuellement un appareil mobile Android personnel.

Contexte :
Les horaires pour chaque classe sont disponibles en ligne via le lien « horaires.cpln.ch », cependant la lisibilité via un appareil mobile n'est pas satisfaisante. Une application Android doit être développée pour avoir une meilleure lecture de l'horaire.

Exigences fonctionnelles :
L'utilisateur de l'application devra saisir le nom de classe (ressource de type nom : 3M3I2) dont il veut obtenir l'horaire du jour. L'application affichera les données en les récupérant depuis le lien http://devinter.cpln.ch/pdf/hypercool; les données renvoyées sont au format JSON.

L'utilisateur doit pouvoir naviguer d'un jour à l'autre au moyen de boutons. Prévoir également le passage aux semaines précédente et suivante.

Il est possible d'indiquer une date via un sélecteur de date. Au lancement la date du jour sera sélectionnée par défaut, ainsi que le nom de la dernière classe consultée.

Exigences de l'interface utilisateur :
L'application sera native pour un système Android.

Des contrôles doivent être effectués pour éviter des erreurs de saisie. Des messages clairs d'erreurs doivent être mise en place.

Les noms des classes déjà saisies seront stockés en interne (persistance) pour permettre de les sélectionner sans les ressaisir.

Les éléments de l'écran doivent s'adapter aux dimensions de l'appareil mobile.

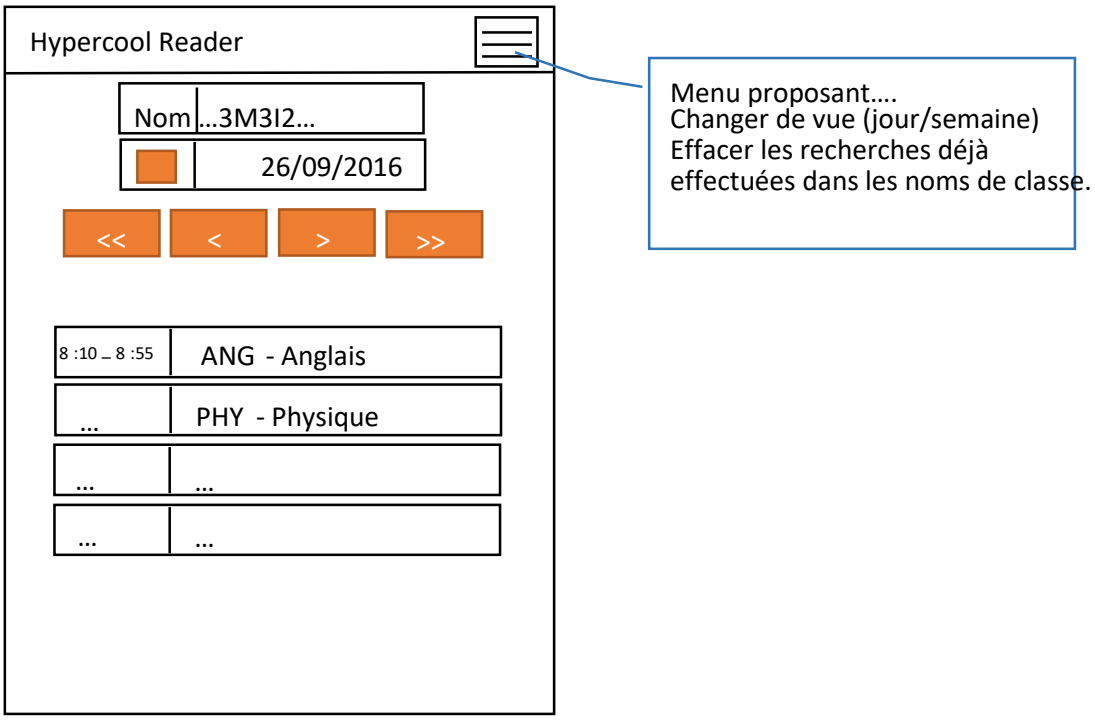
Si le temps le permet :

- On pourra naviguer d'un jour à l'autre d'un glissement de doigt
- Stocker dans la base de données locale l'horaire complet d'une classe et permettre une consultation hors ligne. A tout moment on peut recharger cet horaire.
- Gestion de la rotation (portrait/paysage)

Exigences de qualité :
Seront livrés :

- Un mode d'emploi utilisateur des utilisations fonctionnelles
- Un mode d'emploi d'installation de l'application
- Un fichier APK en plus des sources

Annexes :
Exemple d'interface : possibilité d'en proposer une autre.



Exempledu mode «vuejour».

Accès aux données via Hypercool : http://devinter.cpln.ch/pdf/hypercool

1. Récupère le code de la classe :
   http://devinter.cpln.ch/pdf/hypercool/controler.php?action=ressource&nom=3M3I2 Retourne un JSON : {"4354" : {"nom" : "3M3I2", "code" : "ET"}}

2. Récupère tout l'horaire selon un code de classe :
   http://devinter.cpln.ch/pdf/hypercool/controler.php?action=horaire&ident=4354&sub=all Retourne un JSON (Liste) :
   [{"codeMatiere" : "ICT-153", "libelle" : "….", …..},

   {"codeMatiere" : "ICT-153", "libelle" : "….", ….. } ,

   {"codeMatiere" : "ICT-153", "libelle" : "….", ….. } ,

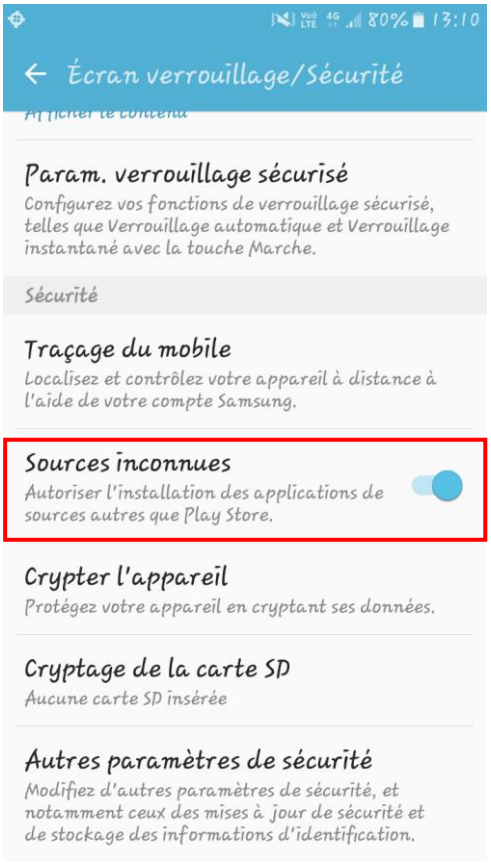
   …

   ]

3. Récupère l'horaire pour une date :
   http://devinter.cpln.ch/pdf/hypercool/controler.php?action=horaire&ident=4354&sub=date&date=23-52017
   Retourne un JSON (Liste) : au format similaire au point 2.

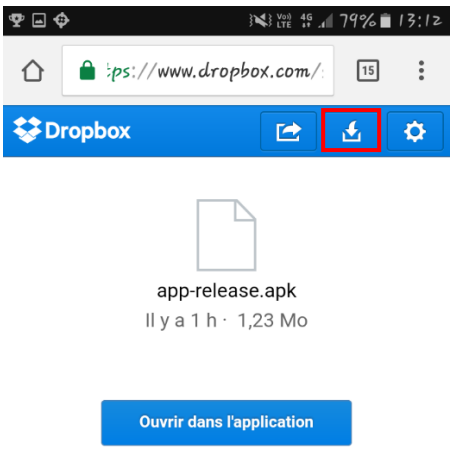
2. Guide d'installation de l'application

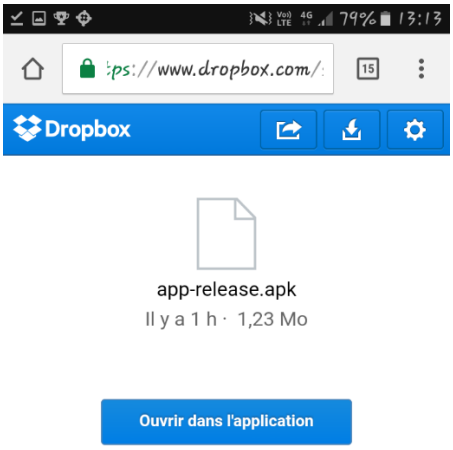
Tout d'abord il faut autoriser le téléphone à installer des applications ne provenant pas de Play Store.
Pour effectuer ceci, rendez-vous dans les paramètres puis « Sécurité ».
Activez l'option « Sources inconnues »

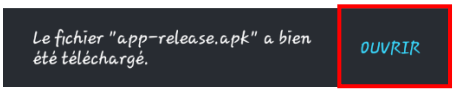


Ouvrez un navigateur et tapez ce lien dans la barre de recherche :
https://www.dropbox.com/s/4pow1i5fgp7of0k/app-release.apk?dl=0

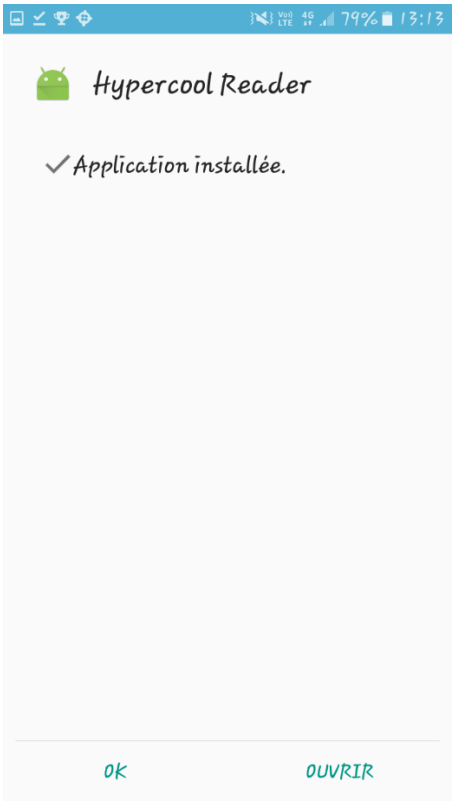
Appuyez ensuite sur l'icône du bouton télécharger en haut à droite.







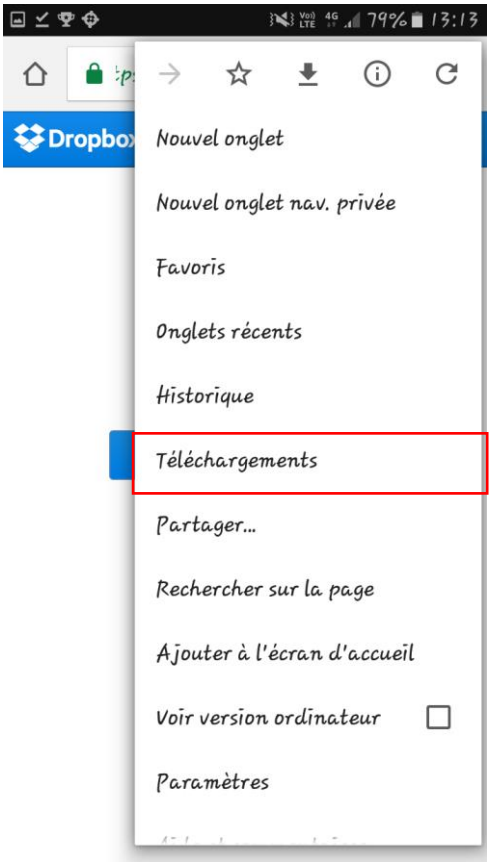
Appuyez sur ouvrir, puis installer.

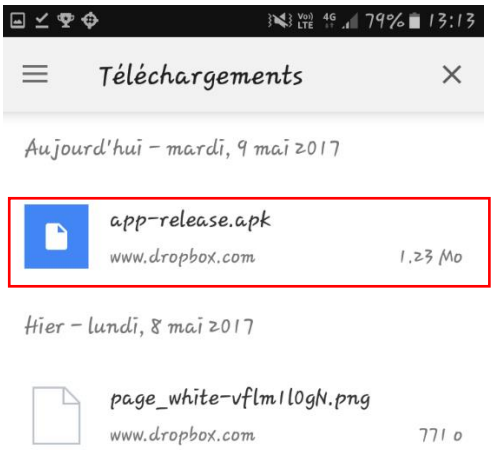


Si vous n’avez pas eu le temps d’appuyer sur « ouvrir », vous pouvez trouver le fichier téléchargé dans les téléchargements du navigateur.

Exemple avec Google Chrome :

Appuyez sur le menu, puis « Téléchargements ». Cliquez alors sur le fichier téléchargé.

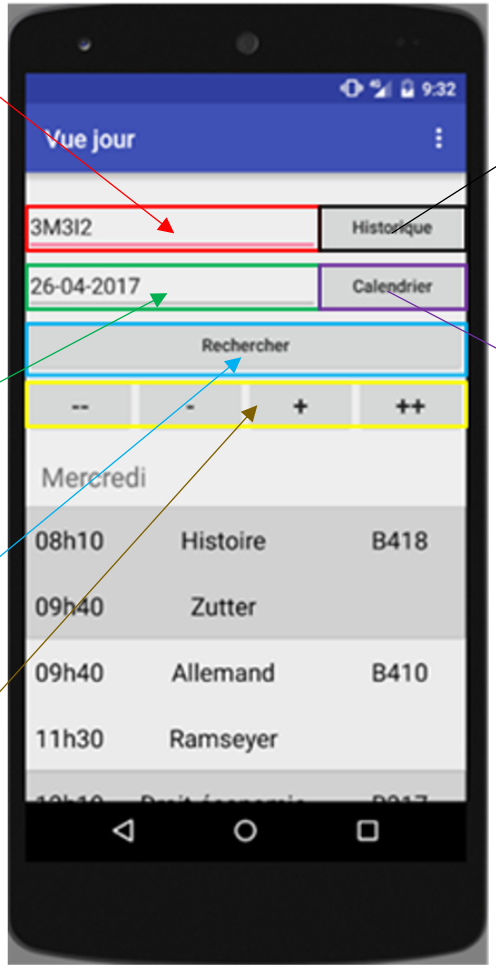


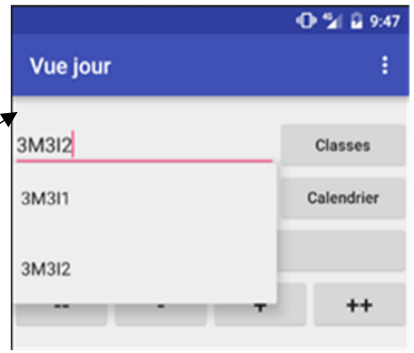


3. Guide d'utilisation de l'application

Guide Utilisateur Hypercool Reader

Vue Jour





Champ classe :
Tapez le nom de la classe que vous voulez rechercher. Une liste déroulante vous aidera à choisir parmi la liste des classes disponibles.

Bouton Historique :
Lors de l'appui sur ce bouton, une liste déroulante va apparaitre avec la liste des classes déjà recherchées. Appuyez de nouveau sur ce bouton (Dorénavant « Classes ») pour afficher à nouveau la liste de toutes les classes disponibles.

Champ date :
Tapez la date que vous voulez pour afficher l'horaire. Un appui sur le bouton calendrier vous permet De faire apparaitre un calendrier pour vous aider à choisir la date.

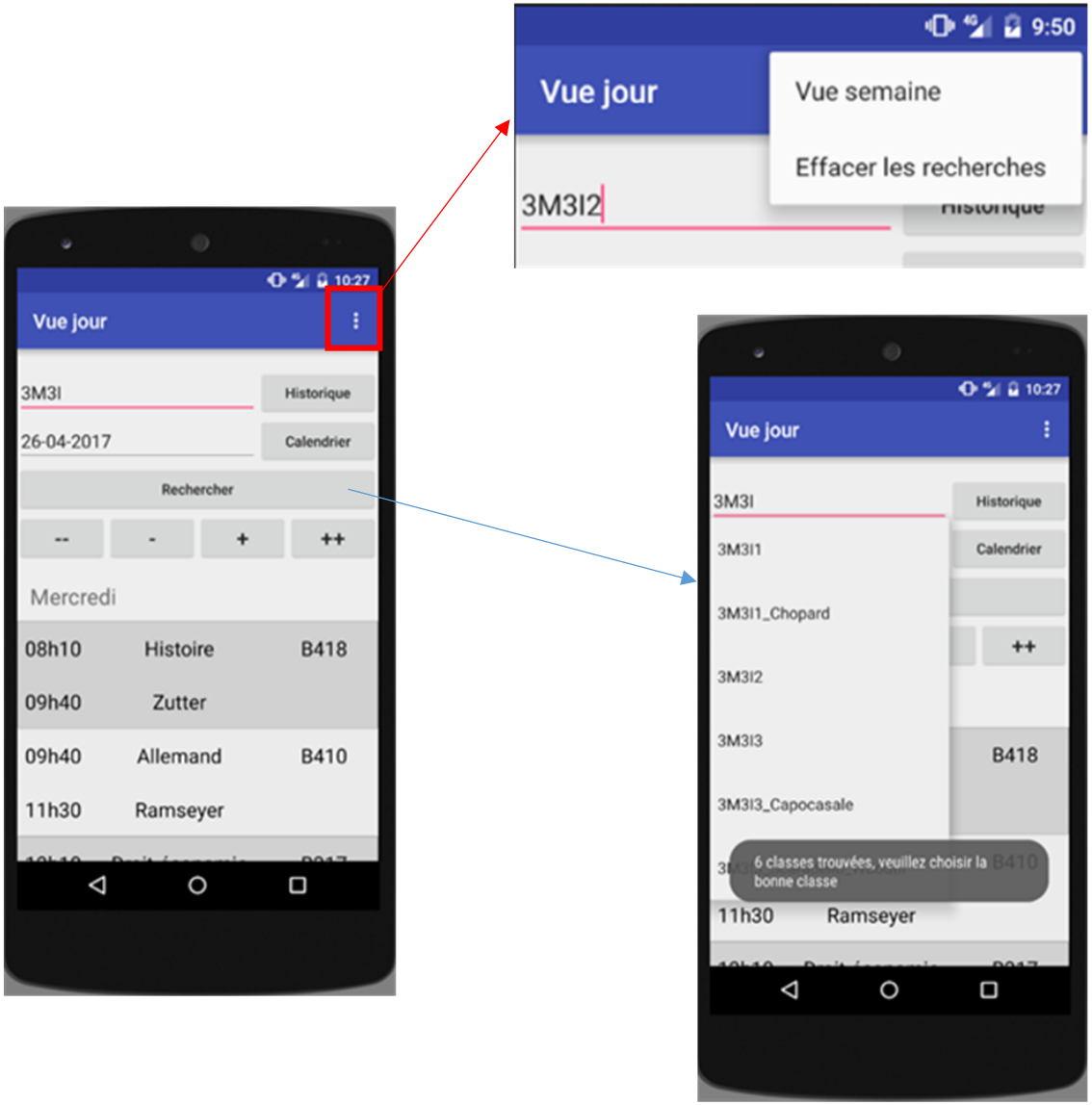
Bouton Rechercher :
Une fois la classe et le date entrée, appuyez sur ce bouton pour afficher l'horaire.

Boutons Avancer/Reculer date :
Les boutons «--» et «-» permettent de reculer la date d'une semaine ou d'un jour. Les boutons «++» et «+» font avancer la date.

Menu :
Appuyez sur le bouton menu, en haut à droite pour afficher les deux options. La première vous permet de changer de mode d'affichage et la deuxième, de vider l'historique des recherches.

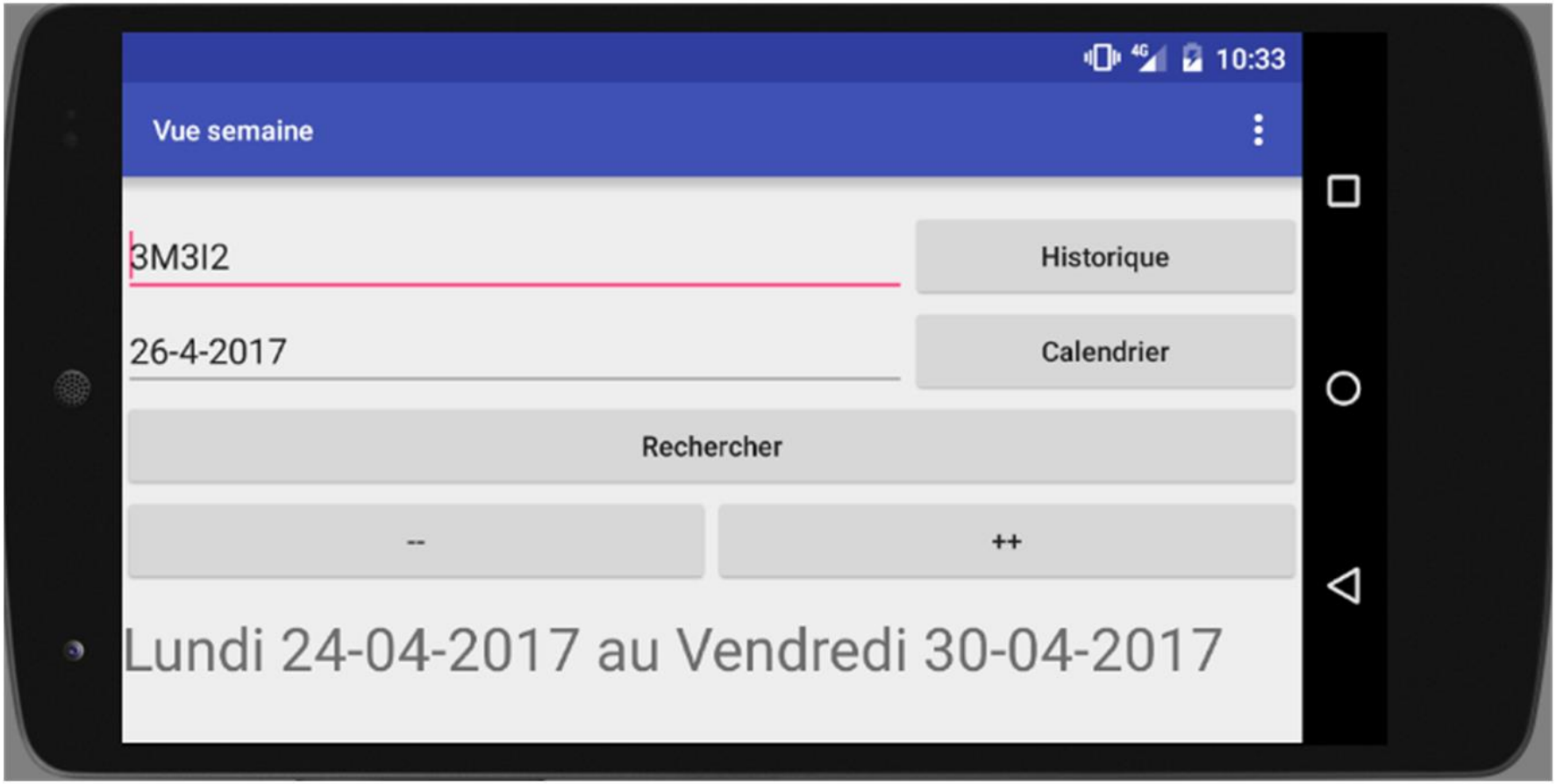
Balayer l'écran :
Pour avancer ou reculer l'horaire d'un jour, vous pouvez également balayer l'écran de gauche à droite et inversement.

Choix plusieurs classes:
Lorsque vous entrez le nom de certaines classes (par exemple 3m3i), il se peut que plusieurs choix s'offrent à vous. Dans ce cas-là, choisissez la classe désirée parmi la liste puis appuyer sur Rechercher.



Vue Semaine

Dans la vue semaine, les boutons agissent de la même façon que dans la vue Jour. La seule différence, est qu'il n'y a plus de bouton pour avancer/reculer de jour, et le balayage de l'écran augmente/diminue la date d'une semaine.



4. Journal de travail

Nom : Bayrakci	**Prénom : Murat**	**Classe : 3M3I2**
Semaine N° :	**Du : 13.03.2017**	**Au : 15.05.2017**

Genre de travail	Durée
Lundi 13.03.2017 : (4 périodes)	
Lecture du cahier des charges	1
Mise en place du Trello, d'une liste de fonctionnalités et définition des priorités des tâches	1
Création du Planning (Répartition des tâches)	1
Création du Planning (Mise en place des périodes nécessaires aux tâches et dépendances)	1
Mardi: 14.03.2017 (9 périodes)	
Création du Schéma de la première vue (sur papier)	1
Re-Création du Schéma de la première vue car l'espace était mal utilisé (sur papier)	1
Création du Schéma de la seconde vue (sur papier)	1
Création du protocole de test initial	1
Fin du planning. Ajout des schémas et du planning dans Trello sous « Documentation »	1
Création du schéma réseau	1
Lecture et Prise en main du site Hypercool	1
Tests de différentes requêtes sur le site Hypercool	1
Analyse des données JSON reçues et rédaction de quelques questions à poser au client	1
Lundi : 20.03.2017 (4 périodes)	
Changement des libellés des cartes du tableau « liste des fonctionnalités ». Création du diagramme de flux de l'activité principal.	1

Création du layout (simple) de la première activité	1
Création du code pour prendre l'id de la classe, puis utiliser cet id pour faire une seconde requête qui va renvoyer l'horaire Brut de la classe	1
Création du code pour placer toutes les informations d'une branche dans un tableau. Pas encore fini.	1
Mardi : 21.03.2017 (9 périodes)	
Relecture du code crée précédemment et simplification de quelques fonctions. Suite de la mise en place de données reçues dans des tableaux.	1
Mise en place des données dans les tableaux finie. Début du tri pour avoir l'horaire chronologique.	1
Tri des tableaux mis en attente. Recherche de documentation pour la création des layouts dynamiques pour afficher les données.	1
Création de plusieurs fichiers xml dans /drawable pour le layout dynamique.	1
Tentative de création du layout dynamique. Vu que je n'arrivais pas, je décide de passer à la suite pour ne pas perdre de temps.	1
Création du menu pour changer d'activité et création des boutons permettant de changer de jour/semaine.	1
Résolution d'un bug qui faisait crash l'application quand aucunes données n'étaient reçues (ajout de try/catch). Fin des boutons permettant de changer jour/semaine, désormais fonctionnels.	1
Mise en place de la date actuelle dès le lancement de l'application.	1
Exécution de différents tests pour la création du layout dynamique (ajout d'un layout depuis le fichier java, ajout de plusieurs textview avec une boucle…). Décision de passer les données des arrays aux arraylists pour pouvoir les trier plus facilement.	1
Lundi 27.03.2017 (4 périodes)	
Démonstration du programme actuel au professeur. Lors de la vérification de mon code par M. Sacchetti, celui-ci m'a donné de nombreux conseils à mettre en place dans le code.	1

Ajouts des sources dans git. Changement dans le code, dorénavant, les données json reçues sont placées dans des arrayslists (array avant). Cela va être plus simple pour trier les données. Mise en place des conseils donnés par M. Sacchetti dont : Utilisation du fichier strings.xml, passage de certaines variables de int en bool, création d'une nouvelle class pour l'asyncstask.	1
Création du use case (sur papier) et suite du tri des données des tableaux cette fois ci avec des arraylists.	1
Fin du tri et ajout de divers try/catch au Jsonobject pour éviter le crash de l'application quand aucune donnée n'était reçue.	1
Mardi 28.03.2017 (4 périodes)	
A la suite des recherches effectuées les jours précédents, j'ai pu commencer la création du layout dynamique. Tout d'abord je créé les fonctions pour créer un textview, un layout horizontal, un layout vertical en fonction du nombre de branches reçues	1
Utilisation des fonctions créées précédemment pour créer le layout. Dorénavant fonctionnel. Ajout d'un scrollview en xml car l'affichage prenaient plus de place que la taille verticale de l'écran.	1
Recherche sur l'utilisation de l'autocompleteTextview et sur l'écriture ligne par ligne dans un fichier texte. Début de la mise en place de l'historique.	1
Création des 3 fonctions : Ecriture texte, Lecture texte et suppression des doublons. Mise en commun des fonctions pour ainsi avoir l'écriture dans le fichier fonctionnelle. Affinement de la version démo.	1
Jeudi 30.03.2017 (5 périodes)	
Création de la seconde activité, du layout et mise en place des requêtes avec des dates choisies arbitrairement.	1
Traitement des données JSON reçues pour les mettre dans des arraylists. Le code est pratiquement le même que la première activité, mais cette fois-ci, je fais une boucle avec le nombre de jour.	1
Début de la fonction permettant de mettre dans un tableau les dates du lundi à dimanche en fonction de la date, qui est dans le champ date.	1
Fin de la fonction et présentation et discussion avec l'expert.	1

Implémentation de la fonction DateEnSemaine() dans l'onCreate, et modification des requêtes qui utilisent dorénavant les dates de la semaine (fournies par le tableau des dates de la semaine).	1
Mardi 18.04.2017 (9 périodes)	
Changement des id de quelques views dans la seconde activité, car les mêmes id dans les deux activités créaient des interférences. Création des cases du tableau pour les périodes dans le xml.	1
Récupération des views pour les mettre dans des linearlayouts et textview 2 dimensions (avec findviewbyid). Ajout du switch avec comme case les heures de débuts (par exemple 8h10, 8h55…). Selon le cas, j'envoie à la fonction RemplissageCases() le numéro du layout en question et la date du début.	1
Création de la fonction RemplissageCases() qui va s'occuper de traiter les données (nom de la branche, salle…) et les placer dans la case correspondante dans le layout.	1
Amélioration de la fonction RemplissageCases() qui va maintenant détecter le nombre de périodes et ainsi colorier plusieurs cases.	1
Une fois l'affichage un peu prêt fonctionnelle, création du tableau de couleur et de l'algorithme pour attribuer la même couleur aux mêmes branches.	1
Amélioration esthétique : maintenant, si la fonction détecte plusieurs périodes, alors le texte va être centré. Si la branche dure plus d'une période, alors je vais également afficher le nom du professeur concerné.	1
Lorsque la branche dure une seule période, pour ne pas prendre trop de place, le code matière va être affiché à la place du libellé. Si la branche dure plusieurs périodes, alors je vais couper les libelles de plus de 20 caractères et les afficher sur plusieurs lignes. Car sans cela, les libellés trop longs faisaient décaler tout le tableau.	1
Petit problème technique survenu : Hypercool et Horaires.cpln ne répondaient plus du tout et je ne recevais donc plus aucunes données. Je décide de passer à la suite et puis revenir lorsque tout devient à nouveau fonctionnel. Création du menu pour pouvoir changer de vue dans l'activité semaine. Création des deux boutons permettant de modifier la date d'une semaine (très semblable à la vue jour)	1
Recherche d'informations pour pouvoir utiliser les données JSON lorsque l'on ne connait pas l'id.	1
Jeudi 20.04.2017 (4 périodes)	

Mise en place de l'autocompletetextview qui va afficher toutes les classes reçues. Pour cela j'utilise un Iterator, comme ça je peux récupérer l'id des classes (que je ne connais pas). Envoi du mail à M. Ferrari pour savoir s'il était possible de faire une requête nous renvoyant uniquement les ressources de type classe.	1
Changement de la fonction TraitementId() dans le deux activités. Avant pour avoir l'id de la classe, étant donné que je ne connaissais pas le nom du jsonarray, je l'obtenais en faisant un substring. Ce n'était pas très propre comme manière de faire. Maintenant j'ai changé et je travaille avec l'iterator. (Même principe que le point précédent.)	1
Recherche d'informations pour pouvoir afficher un calendrier lorsque l'on appuie sur un bouton. Début de la mise en place du datePicker.	1
Fin de la mise en place du datePicker dans les deux activités. Dorénavant fonctionnelle.	1
Vendredi 21.04.2017 (4 périodes)	
Mise à niveau de l'historique car dorénavant la liste de classe utilise l'autocompletetextview. Je vais alors créer un spinner (liste déroulante) avec la liste des classes déjà recherchées.	1
La liste marche un peu près sauf qu'il y a un petit bug qui est présent. Si on clique sur le bouton, lors du choix d'un item du spinner, le string de l'item sélectionné ne se met pas dans le champ classe. J'ai vu que c'était l'item qui était en preview dans le spinner qui ne répondait pas. Les autres choix fonctionnaient.	1
Essayer de résoudre le bug.	1
Je n'ai pas réussi à trouver l'origine du bug, je décide alors de passer à la suite pour ne pas perdre de temps.	1
Lundi 24.04.2017 (0 période)	
Prof absent	0
Mardi 25.04.2017 (7 périodes à cause de la conférence)	

J'ai découvert que l'utilisation du .get() bloquait l'écran. Je décide alors d'essayer une nouvelle méthode pour faire passer les données reçues de l'Asynctask à l'activité principale. Recherche de documentation pour pouvoir passer les outputs de l'asynctask dans l'activité principale. Création des fonctions pour récupérer les outputs.	1
Mise en place des requêtes dans les deux activités. Ajout d'une barre de chargement	1
Test des requêtes cette fois-ci de manière asynchrone. Petit problème car ça fonctionne uniquement après 2 clics sur le bouton rechercher. Tentative de résolution du bug.	1
Résolution du bug précédent. Le problème était que je lançais au même temps la requête pour prendre l'id d'une classe et la requête qui renvoie l'horaire d'une classe en fonction de l'id. Effectivement ça posait problème car la seconde requête n'avait pas encore l'id. J'ai donc décidé de lancer la 2eme requête après la première.	1
Correction de quelques erreurs : Les classes contenant un espace ne renvoyaient aucunes données. Pour régler cela, je remplace les espaces par %20 lors de mise en variable. Dorénavant, je mets le contenu de la variable qui contient la classe en majuscule. Cela permet de ne pas avoir de doublons dans l'historique. (3m3i2 et 3M3I2)	1
Conférence	2
Ajout de messages d'erreurs lorsque le champ date/classe est vide ou erroné, ou que l'utilisateur n'a pas de connexion internet. Ajout d'une couleur différente pour les cases de midi.	1
Mise en place du balayage de l'écran. Lorsque l'on balaye l'écran de droite à gauche (et inversement), on passe au jour suivant/précédent (activité jour) ou à la semaine suivante/précédente (activité semaine)	1
Jeudi 27.04.2017 (4 périodes)	
Début de la mise en place d'une nouvelle manière pour afficher l'historique. Avant je voulais faire apparaitre une liste déroulante lorsque l'utilisateur cliquait sur le bouton historique. Je n'avais pas totalement réussi, je vais donc, à l'appui du bouton historique, changer la source de l'autocompletetextview du champ classe pour que celui-ci affiche l'historique.	1

Démonstration au professeur. Création d'un second menu pour afficher « Vue Semaine » et « Vue Jour » alors qu'avant on affichait « Changer de Vue ». Discussion avec le professeur et annulation de la fonctionnalité pour lancer la recherche lorsque les deux champs étaient remplis.	1
Fin de la mise en place de la nouvelle manière pour afficher l'historique. Lorsqu'une classe n'a pas de cours, j'affiche dorénavant un message informatif.	1
Début de la gestion du cas lorsque plusieurs classes sont renvoyées (« 3m3i » renvoie « 3m3i2 », « 3m3i1 » et « 3m3i3 ».	1
Vendredi 28.04.2017 : (4 périodes)	
Fin de la gestion lorsque plusieurs classes étaient reçues. Création du fichier pour enregistrer la dernière classe recherchée.	1
Blocage des boutons avancer/reculer jour/semaine lorsque l'utilisateur doit choisir parmi la liste de classes retournées. Lors du choix de la classe, je masque le clavier pour inciter l'utilisateur à choisir parmi la liste.	1
Ajout d'une ligne dans le manifest pour gérer la rotation de l'écran et ainsi ne pas faire boguer l'affichage. Petit problème néanmoins : la date revient à la date du jour actuelle lors du changement d'orientation. A régler si le temps le permet à la fin.	1
Correction de l'écriture des classes dans le fichier historique. Avant les boutons pour avancer/reculer de jours enregistraient à chaque clic le contenu du champ classe. Ça posait problème car si l'utilisateur recherchait une classe (ex : 3m3i2) puis décidait de changer le contenu du champ classe (ex 3m3iaaaa) sans appuyer sur le bouton rechercher, les boutons + - enregistraient quand même la classe erronée. Maintenant ces boutons n'enregistrent plus dans le fichier, mais uniquement le bouton rechercher le fait.	1
Mardi : 02.05.2017 (9 périodes)	
Gestion des périodes aux heures non conventionnelles : Création de la boucle permettant d'envoyer l'heure de début et le numéro de la case à la fonction RemplissageCases2().	1
Création de la fonction RemplissageCases2() pour gérer les périodes commençant à des heures non conventionnelles.	1
Création du use case sur Visio.	1
Création du protocole de test.	1

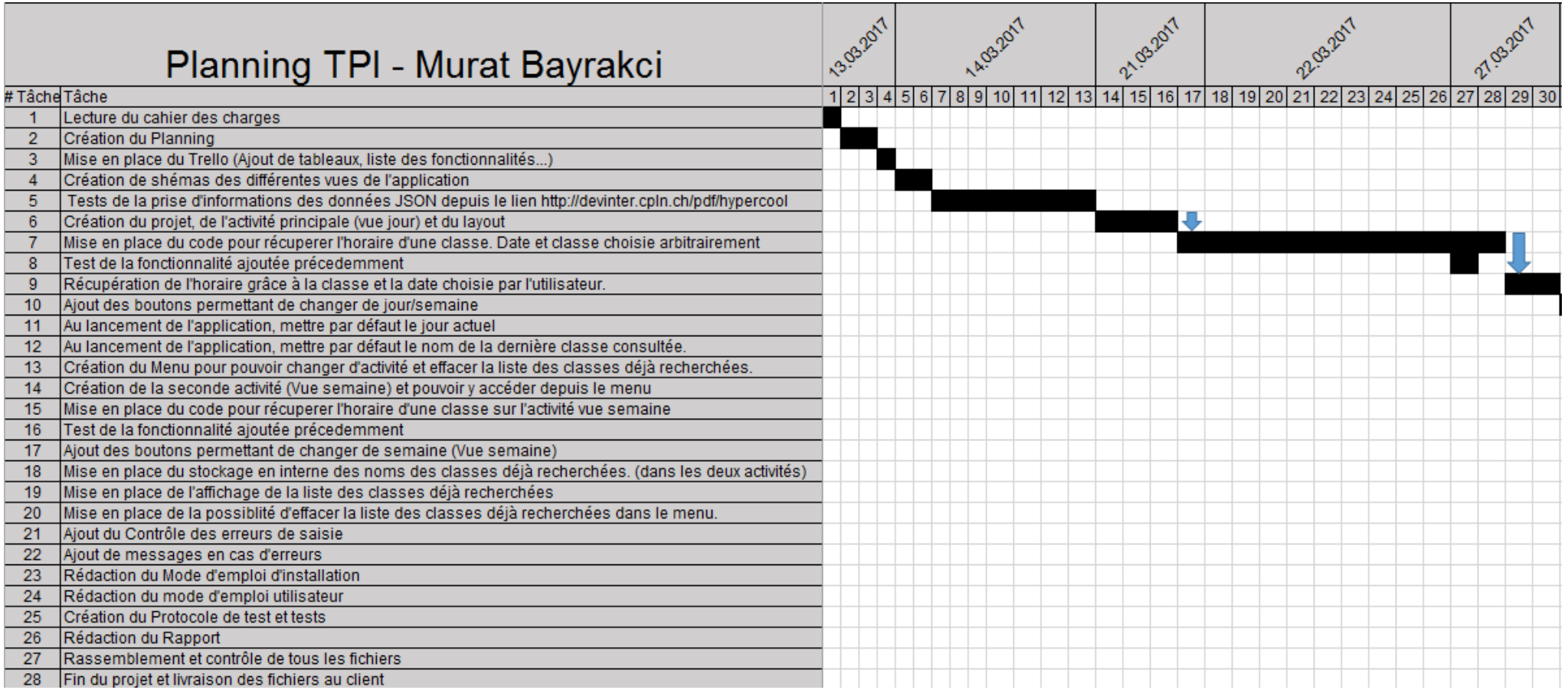
Test de l'application en fonction du protocole de test.	1
Création du guide utilisateur.	1
Début du rapport : Rédaction de l'introduction	1
Démonstration et discussion avec l'expert.	1
Rédaction de l'explication détaillée du projet.	1
Jeudi : 04.05.2017 (9 périodes)	
Suite du rapport : Rédaction des conventions de nommage et ajout des schémas architecture du système et du use case	1
Création des diagrammes de flux de différentes fonctions	1
Rédaction du use case « Affichage de l'horaire du jour lors du démarrage »	1
Suite de la rédaction du use case précédent	1
Rédaction du use case « Affichage de l'horaire d'un jour avec classe et date choisie par l'utilisateur », début du use case « Affichage de l'horaire de la semaine directement au passage à la vue semaine ».	1
Fin du use case précédent et début de la rédaction du use case « Affichage de l'horaire d'une semaine avec classe et date choisie par l'utilisateur »	1
Rédaction du use case « Modifier la date avec des boutons »	1
Rédaction des use case « Modifier la date en balayant l'écran » et « Sélectionner la date avec un calendrier »	1
Rédaction des use case « Basculer entre la vue semaine et la vue jour » et « Afficher l'historique des recherches	1
Vendredi : 05.05.2017 (4 périodes)	
Rédaction des use case « Effacer l'historique des recherches » et « stocker les données reçues dans une base de données »	1
Rédaction des use case « Gestion de la rotation de l'écran » et « Application s'adaptant à la taille de l'écran »	1
Rédaction de la conclusion du rapport	1
Rédaction des points à améliorer	1

Lundi : 08.05.2017 (4 périodes)	
Rédaction du chapitre « Bugs rencontrés et solutions »	1
Ajout de la liste de liens utilisés dans le chapitre « Références »	1
Ajout de quelques tests dans le protocole de test. + tests	1
Recherche de documentation pour obtenir l' .apk de l'application	1
Mardi 09.05.2017 (9 périodes)	
Création du guide d'installation de l'application.	1
Simplification du code de la vue jour (suppression des parties inutiles, …)	1
Simplification du code de la vue semaine.	1
Relecture du rapport pour enlever les petites erreurs.	1
Relecture du rapport pour enlever les petites erreurs.	1
Relecture du rapport pour enlever les petites erreurs.	1
Création du fichier annexe.	1
Mise à jour du planning final.	1
Vérification de la mise en page du rapport (numérotation, espacement…).	1
Jeudi : 11.05.2017 (3 périodes)	
Rassemblement de tous les fichiers.	1
Vérification de tous les fichiers	1
Livraison des fichiers au professeur.	1
Bilan	
Total : 109 périodes (deux périodes ont été perdues lors de la conférence et l'une d'elle n'a pas pu être rattrapée).	

5. Planning

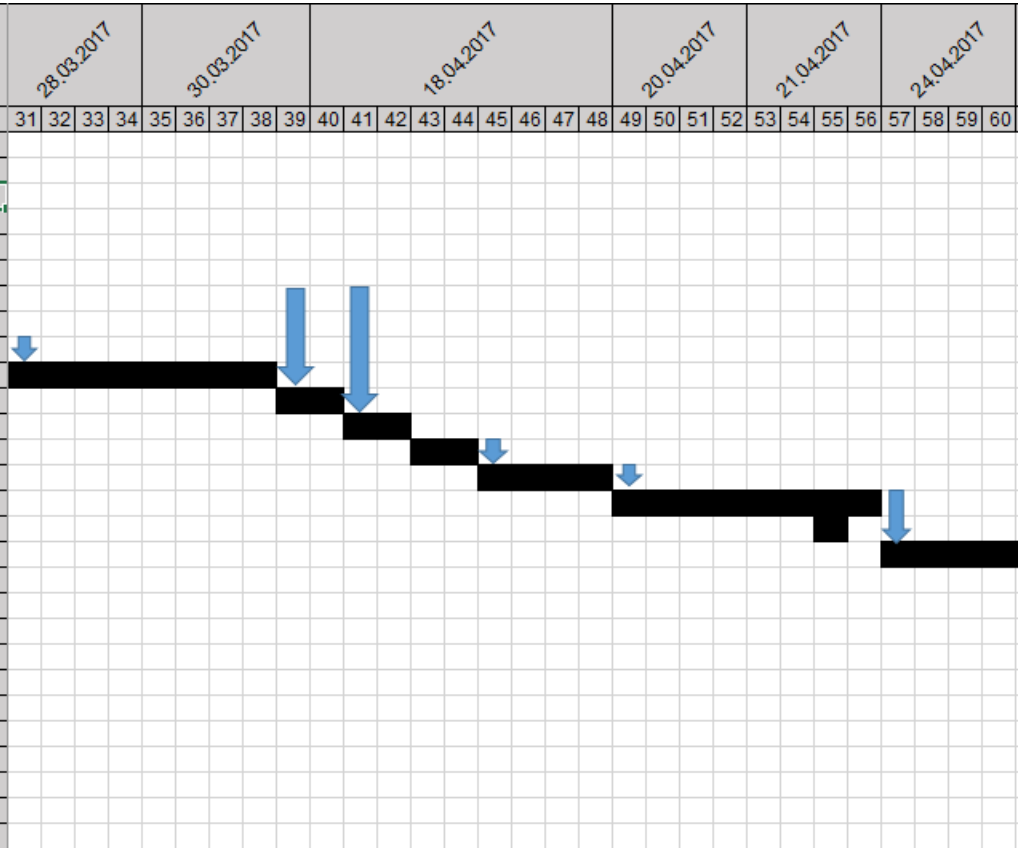
Un fichier excel est disponible pour plus de lisibilité.

Planning initial



Planning TPI - Murat Bayrakci		13.03.2017				14.03.2017								21.03.2017				22.03.2017									27.03.2017			
# Tâche	Tâche	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	Lecture du cahier des charges																														
2	Création du Planning																														
3	Mise en place du Trello (Ajout de tableaux, liste des fonctionnalités...)																														
4	Création de shémas des différentes vues de l'application																														
5	Tests de la prise d'informations des données JSON depuis le lien http://devinter.cpln.ch/pdf/hypercool																														
6	Création du projet, de l'activité principale (vue jour) et du layout																														
7	Mise en place du code pour récuperer l'horaire d'une classe. Date et classe choisie arbitrairement																														
8	Test de la fonctionnalité ajoutée précedemment																														
9	Récupération de l'horaire grâce à la classe et la date choisie par l'utilisateur.																														
10	Ajout des boutons permettant de changer de jour/semaine																														
11	Au lancement de l'application, mettre par défaut le jour actuel																														
12	Au lancement de l'application, mettre par défaut le nom de la dernière classe consultée.																														
13	Création du Menu pour pouvoir changer d'activité et effacer la liste des classes déjà recherchées.																														
14	Création de la seconde activité (Vue semaine) et pouvoir y accéder depuis le menu																														
15	Mise en place du code pour récuperer l'horaire d'une classe sur l'activité vue semaine																														
16	Test de la fonctionnalité ajoutée précedemment																														
17	Ajout des boutons permettant de changer de semaine (Vue semaine)																														
18	Mise en place du stockage en interne des noms des classes déjà recherchées. (dans les deux activités)																														
19	Mise en place de l'affichage de la liste des classes déjà recherchées																														
20	Mise en place de la possiblité d'effacer la liste des classes déjà recherchées dans le menu.																														
21	Ajout du Contrôle des erreurs de saisie																														
22	Ajout de messages en cas d'erreurs																														
23	Rédaction du Mode d'emploi d'installation																														
24	Rédaction du mode d'emploi utilisateur																														
25	Création du Protocole de test et tests																														
26	Rédaction du Rapport																														
27	Rassemblement et contrôle de tous les fichiers																														
28	Fin du projet et livraison des fichiers au client																														

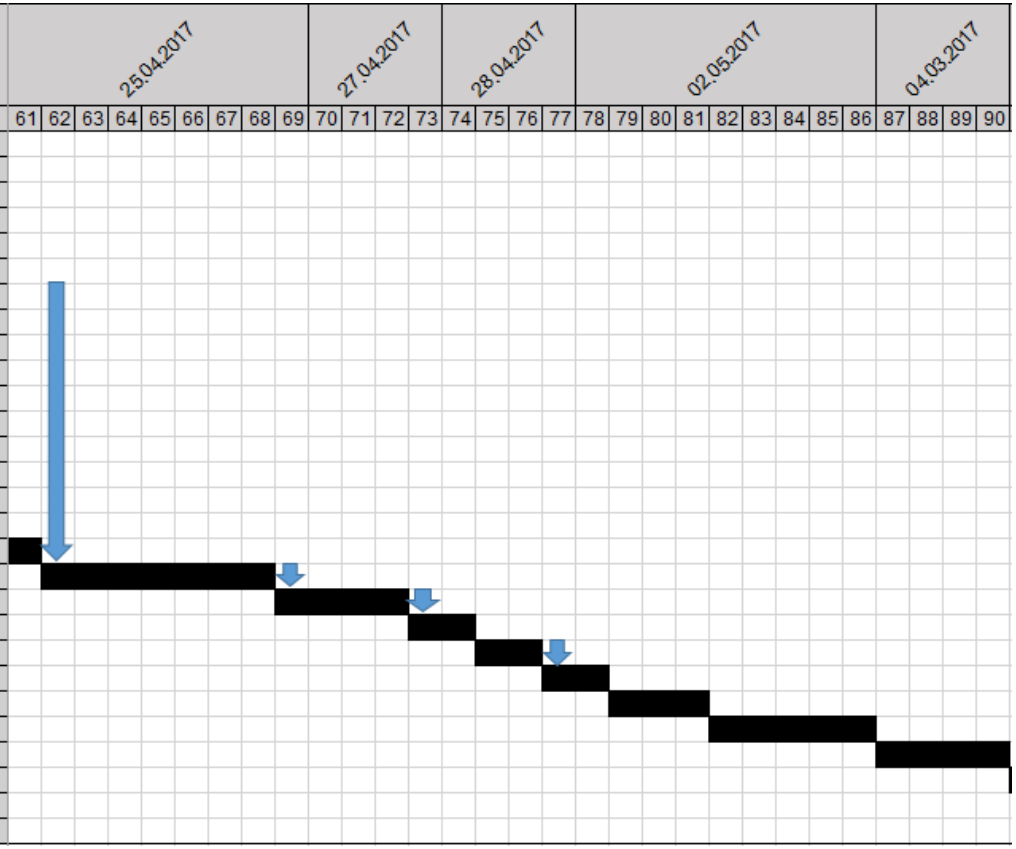
Planning TPI - Murat Bayrakci



# Tâche	Tâche
1	Lecture du cahier des charges
2	Création du Planning
3	Mise en place du Trello (Ajout de tableaux, liste des fonctionnalités...)
4	Création de shémas des différentes vues de l'application
5	Tests de la prise d'informations des données JSON depuis le lien http://devinter.cpln.ch/pdf/hypercool
6	Création du projet, de l'activité principale (vue jour) et du layout
7	Mise en place du code pour récuperer l'horaire d'une classe. Date et classe choisie arbitrairement
8	Test de la fonctionnalité ajoutée précedemment
9	Récupération de l'horaire grâce à la classe et la date choisie par l'utilisateur.
10	Ajout des boutons permettant de changer de jour/semaine
11	Au lancement de l'application, mettre par défaut le jour actuel
12	Au lancement de l'application, mettre par défaut le nom de la dernière classe consultée.
13	Création du Menu pour pouvoir changer d'activité et effacer la liste des classes déjà recherchées.
14	Création de la seconde activité (Vue semaine) et pouvoir y accéder depuis le menu
15	Mise en place du code pour récuperer l'horaire d'une classe sur l'activité vue semaine
16	Test de la fonctionnalité ajoutée précedemment
17	Ajout des boutons permettant de changer de semaine (Vue semaine)
18	Mise en place du stockage en interne des noms des classes déjà recherchées. (dans les deux activités)
19	Mise en place de l'affichage de la liste des classes déjà recherchées
20	Mise en place de la possiblité d'effacer la liste des classes déjà recherchées dans le menu.
21	Ajout du Contrôle des erreurs de saisie
22	Ajout de messages en cas d'erreurs
23	Rédaction du Mode d'emploi d'installation
24	Rédaction du mode d'emploi utilisateur
25	Création du Protocole de test et tests
26	Rédaction du Rapport
27	Rassemblement et contrôle de tous les fichiers
28	Fin du projet et livraison des fichiers au client

Planning TPI - Murat Bayrakci

# Tâche	Tâche
1	Lecture du cahier des charges
2	Création du Planning
3	Mise en place du Trello (Ajout de tableaux, liste des fonctionnalités...)
4	Création de shémas des différentes vues de l'application
5	Tests de la prise d'informations des données JSON depuis le lien http://devinter.cpln.ch/pdf/hypercool
6	Création du projet, de l'activité principale (vue jour) et du layout
7	Mise en place du code pour récuperer l'horaire d'une classe. Date et classe choisie arbitrairement
8	Test de la fonctionnalité ajoutée précedemment
9	Récupération de l'horaire grâce à la classe et la date choisie par l'utilisateur.
10	Ajout des boutons permettant de changer de jour/semaine
11	Au lancement de l'application, mettre par défaut le jour actuel
12	Au lancement de l'application, mettre par défaut le nom de la dernière classe consultée.
13	Création du Menu pour pouvoir changer d'activité et effacer la liste des classes déjà recherchées.
14	Création de la seconde activité (Vue semaine) et pouvoir y accéder depuis le menu
15	Mise en place du code pour récuperer l'horaire d'une classe sur l'activité vue semaine
16	Test de la fonctionnalité ajoutée précedemment
17	Ajout des boutons permettant de changer de semaine (Vue semaine)
18	Mise en place du stockage en interne des noms des classes déjà recherchées. (dans les deux activités)
19	Mise en place de l'affichage de la liste des classes déjà recherchées
20	Mise en place de la possiblité d'effacer la liste des classes déjà recherchées dans le menu.
21	Ajout du Contrôle des erreurs de saisie
22	Ajout de messages en cas d'erreurs
23	Rédaction du Mode d'emploi d'installation
24	Rédaction du mode d'emploi utilisateur
25	Création du Protocole de test et tests
26	Rédaction du Rapport
27	Rassemblement et contrôle de tous les fichiers
28	Fin du projet et livraison des fichiers au client



Planning TPI - Murat Bayrakci		05.03.2017				08.03.2017				09.03.2017										11.03.2017			Date
# Tâche	Tâche	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	Période
1	Lecture du cahier des charges																					
2	Création du Planning																					
3	Mise en place du Trello (Ajout de tableaux, liste des fonctionnalités...)																					
4	Création de shémas des différentes vues de l'application																					
5	Tests de la prise d'informations des données JSON depuis le lien http://devinter.cpln.ch/pdf/hypercool																					
6	Création du projet, de l'activité principale (vue jour) et du layout																					
7	Mise en place du code pour récuperer l'horaire d'une classe. Date et classe choisie arbitrairement																					
8	Test de la fonctionnalité ajoutée précedemment																					
9	Récupération de l'horaire grâce à la classe et la date choisie par l'utilisateur.																					
10	Ajout des boutons permettant de changer de jour/semaine																					
11	Au lancement de l'application, mettre par défaut le jour actuel																					
12	Au lancement de l'application, mettre par défaut le nom de la dernière classe consultée.																					
13	Création du Menu pour pouvoir changer d'activité et effacer la liste des classes déjà recherchées.																					
14	Création de la seconde activité (Vue semaine) et pouvoir y accéder depuis le menu																					
15	Mise en place du code pour récuperer l'horaire d'une classe sur l'activité vue semaine																					
16	Test de la fonctionnalité ajoutée précedemment																					
17	Ajout des boutons permettant de changer de semaine (Vue semaine)																					
18	Mise en place du stockage en interne des noms des classes déjà recherchées. (dans les deux activités)																					
19	Mise en place de l'affichage de la liste des classes déjà recherchées																					
20	Mise en place de la possiblité d'effacer la liste des classes déjà recherchées dans le menu.																					
21	Ajout du Contrôle des erreurs de saisie																					
22	Ajout de messages en cas d'erreurs																					
23	Rédaction du Mode d'emploi d'installation																					
24	Rédaction du mode d'emploi utilisateur																					
25	Création du Protocole de test et tests																					
26	Rédaction du Rapport																					
27	Rassemblement et contrôle de tous les fichiers																					
28	Fin du projet et livraison des fichiers au client																					

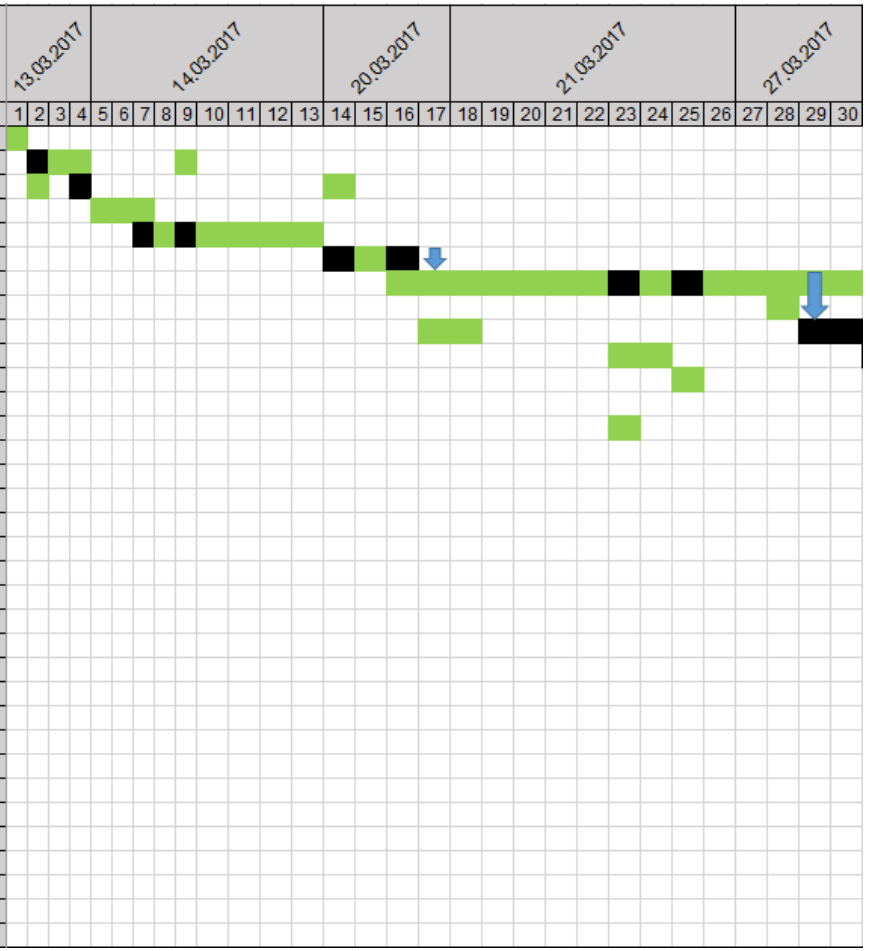
Planning final

Le total est de 109 périodes au lieu de 110. Une conférence de deux périodes a eu lieu le 25.04.2017. J'ai pu rattraper uniquement une période.

Légende	
	Tâche réalisée
	Tache initiale

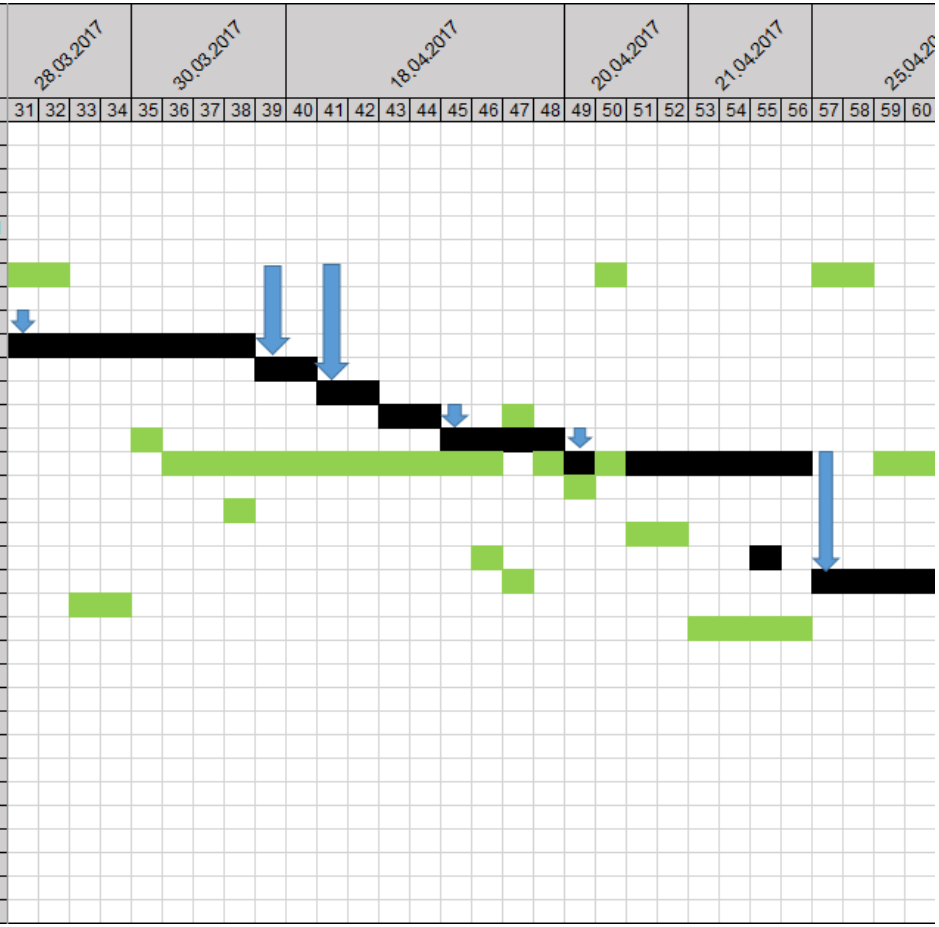
Planning final TPI - Murat Bayrakci

# Tâche	Tâche
1	Lecture du cahier des charges
2	Création du Planning
3	Mise en place du Trello (Ajout de tableaux, liste des fonctionnalités...)
4	Création dse shémas des différentes vues de l'application et du use case
5	Tests de la prise d'informations des données JSON depuis le lien http://devinter.cpln.ch/pdf/hypercool + Création protocole de test intital
6	Création du projet, de l'activité principale (vue jour) et du layout
7	Mise en place du code pour récuperer l'horaire d'une classe. Date et classe choisie arbitrairement
8	Test de la fonctionnalité ajoutée précedemment
9	Récupération de l'horaire grâce à la classe et la date choisie par l'utilisateur.
10	Ajout des boutons permettant de changer de jour/semaine
11	Au lancement de l'application, mettre par défaut le jour actuel
12	Au lancement de l'application, mettre par défaut le nom de la dernière classe consultée.
13	Création du Menu pour pouvoir changer d'activité et effacer la liste des classes déjà recherchées.
14	Création de la seconde activité (Vue semaine) et pouvoir y accéder depuis le menu
15	Mise en place du code pour récuperer l'horaire d'une classe sur l'activité vue semaine
16	Récupération de la liste de toutes les classes disponibles et affichage comme suggestion dans le champ classe
17	Présentation et discussion avec l'expert.
18	Apparition du calendrier lors de l'appui sur le bouton calendrier
19	Test de la fonctionnalité ajoutée précedemment
20	Ajout des boutons permettant de changer de semaine (Vue semaine)
21	Mise en place du stockage en interne des noms des classes déjà recherchées. (dans les deux activités)
22	Mise en place de l'affichage de la liste des classes déjà recherchées
23	Gestion lorsque plusieurs classes sont renvoyées.
24	Mise en place de la possiblité d'effacer la liste des classes déjà recherchées dans le menu.
25	Ajout du balayage de l'écran
26	Ajout du contrôle des erreurs de saisies
27	Ajout de messages en cas d'erreurs
28	Rédaction du Mode d'emploi d'installation
29	Rédaction du mode d'emploi utilisateur
30	Création du fichier annexe
31	Création du Protocole de test et tests
32	Rédaction du Rapport
33	Rassemblement et contrôle de tous les fichiers
34	Fin du projet et livraison des fichiers au client



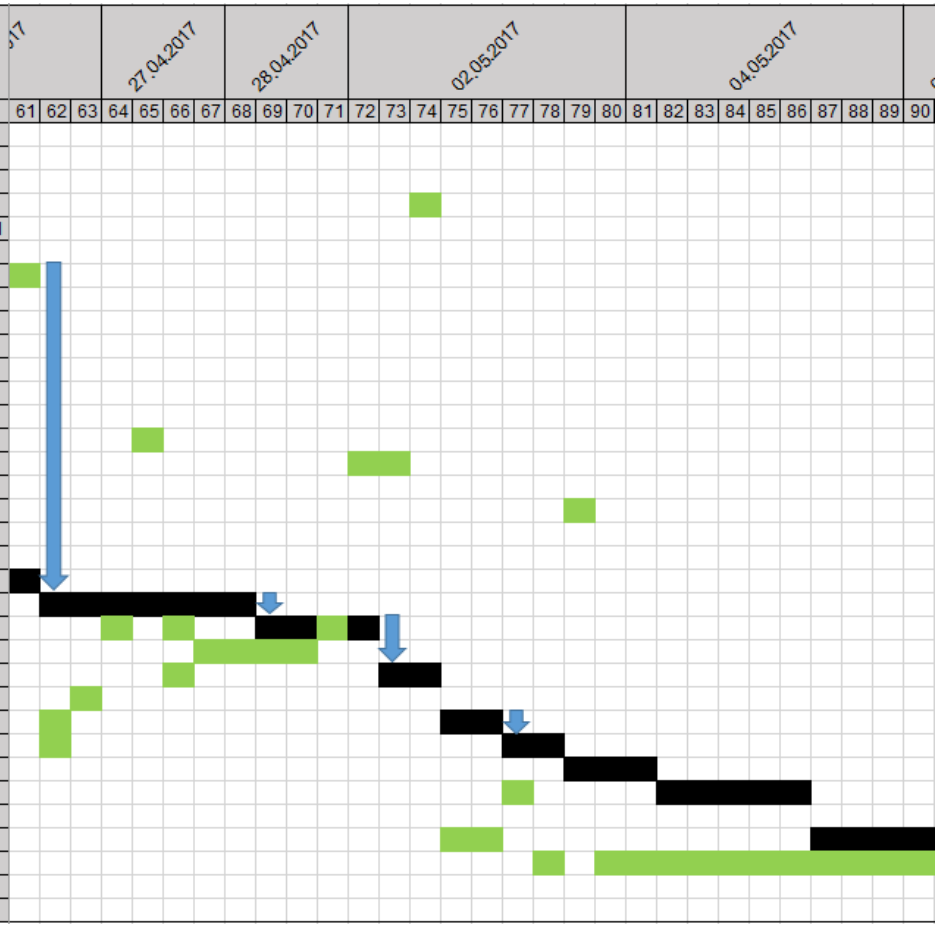
Planning final TPI - Murat Bayrakci

# Tâche	Tâche
1	Lecture du cahier des charges
2	Création du Planning
3	Mise en place du Trello (Ajout de tableaux, liste des fonctionnalités...)
4	Création dse shémas des différentes vues de l'application et du use case
5	Tests de la prise d'informations des données JSON depuis le lien http://devinter.cpln.ch/pdf/hypercool + Création protocole de test intital
6	Création du projet, de l'activité principale (vue jour) et du layout
7	Mise en place du code pour récuperer l'horaire d'une classe. Date et classe choisie arbitrairement
8	Test de la fonctionnalité ajoutée précedemment
9	Récupération de l'horaire grâce à la classe et la date choisie par l'utilisateur.
10	Ajout des boutons permettant de changer de jour/semaine
11	Au lancement de l'application, mettre par défaut le jour actuel
12	Au lancement de l'application, mettre par défaut le nom de la dernière classe consultée.
13	Création du Menu pour pouvoir changer d'activité et effacer la liste des classes déjà recherchées.
14	Création de la seconde activité (Vue semaine) et pouvoir y accéder depuis le menu
15	Mise en place du code pour récuperer l'horaire d'une classe sur l'activité vue semaine
16	Récupération de la liste de toutes les classes disponibles et affichage comme suggestion dans le champ classe
17	Présentation et discussion avec l'expert.
18	Apparition du calendrier lors de l'appui sur le bouton calendrier
19	Test de la fonctionnalité ajoutée précedemment
20	Ajout des boutons permettant de changer de semaine (Vue semaine)
21	Mise en place du stockage en interne des noms des classes déjà recherchées. (dans les deux activités)
22	Mise en place de l'affichage de la liste des classes déjà recherchées
23	Gestion lorsque plusieurs classes sont renvoyées.
24	Mise en place de la possiblité d'effacer la liste des classes déjà recherchées dans le menu.
25	Ajout du balayage de l'écran
26	Ajout du contrôle des erreurs de saisies
27	Ajout de messages en cas d'erreurs
28	Rédaction du Mode d'emploi d'installation
29	Rédaction du mode d'emploi utilisateur
30	Création du fichier annexe
31	Création du Protocole de test et tests
32	Rédaction du Rapport
33	Rassemblement et contrôle de tous les fichiers
34	Fin du projet et livraison des fichiers au client



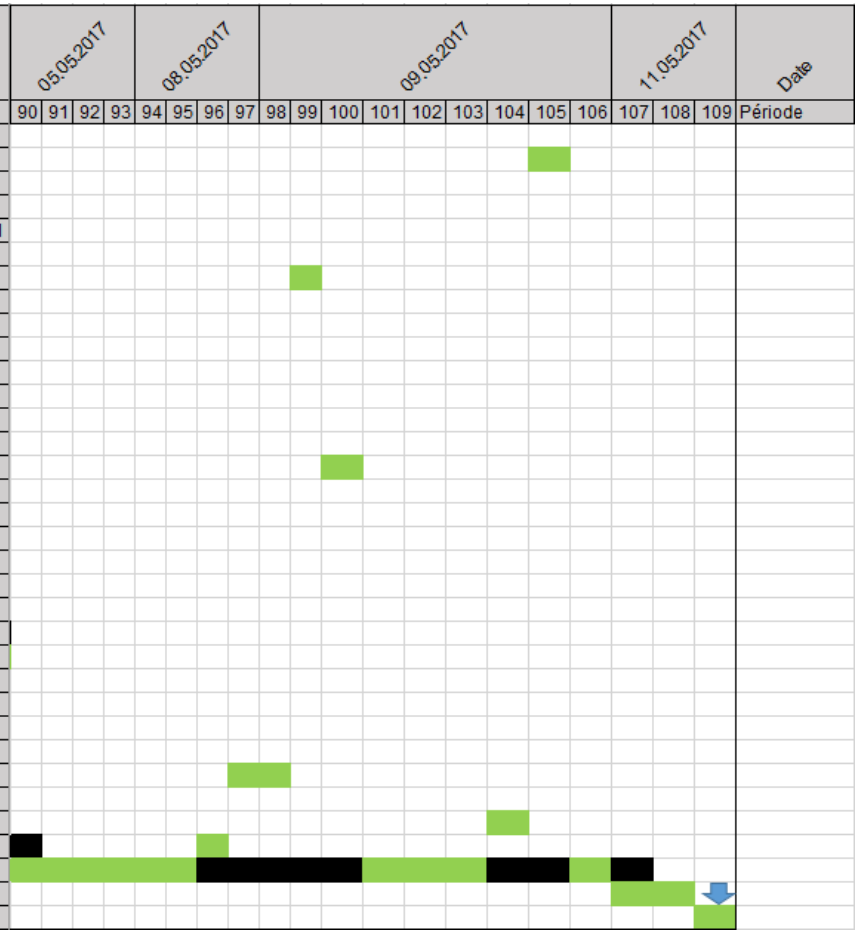
Planning final TPI - Murat Bayrakci

# Tâche	Tâche
1	Lecture du cahier des charges
2	Création du Planning
3	Mise en place du Trello (Ajout de tableaux, liste des fonctionnalités...)
4	Création dse shémas des différentes vues de l'application et du use case
5	Tests de la prise d'informations des données JSON depuis le lien http://devinter.cpln.ch/pdf/hypercool + Création protocole de test intital
6	Création du projet, de l'activité principale (vue jour) et du layout
7	Mise en place du code pour récuperer l'horaire d'une classe. Date et classe choisie arbitrairement
8	Test de la fonctionnalité ajoutée précedemment
9	Récupération de l'horaire grâce à la classe et la date choisie par l'utilisateur.
10	Ajout des boutons permettant de changer de jour/semaine
11	Au lancement de l'application, mettre par défaut le jour actuel
12	Au lancement de l'application, mettre par défaut le nom de la dernière classe consultée.
13	Création du Menu pour pouvoir changer d'activité et effacer la liste des classes déjà recherchées.
14	Création de la seconde activité (Vue semaine) et pouvoir y accéder depuis le menu
15	Mise en place du code pour récuperer l'horaire d'une classe sur l'activité vue semaine
16	Récupération de la liste de toutes les classes disponibles et affichage comme suggestion dans le champ classe
17	Présentation et discussion avec l'expert.
18	Apparition du calendrier lors de l'appui sur le bouton calendrier
19	Test de la fonctionnalité ajoutée précedemment
20	Ajout des boutons permettant de changer de semaine (Vue semaine)
21	Mise en place du stockage en interne des noms des classes déjà recherchées. (dans les deux activités)
22	Mise en place de l'affichage de la liste des classes déjà recherchées
23	Gestion lorsque plusieurs classes sont renvoyées.
24	Mise en place de la possiblité d'effacer la liste des classes déjà recherchées dans le menu.
25	Ajout du balayage de l'écran
26	Ajout du contrôle des erreurs de saisies
27	Ajout de messages en cas d'erreurs
28	Rédaction du Mode d'emploi d'installation
29	Rédaction du mode d'emploi utilisateur
30	Création du fichier annexe
31	Création du Protocole de test et tests
32	Rédaction du Rapport
33	Rassemblement et contrôle de tous les fichiers
34	Fin du projet et livraison des fichiers au client



Planning final TPI - Murat Bayrakci

# Tâche	Tâche
1	Lecture du cahier des charges
2	Création du Planning
3	Mise en place du Trello (Ajout de tableaux, liste des fonctionnalités...)
4	Création dse shémas des différentes vues de l'application et du use case
5	Tests de la prise d'informations des données JSON depuis le lien http://devinter.cpln.ch/pdf/hypercool + Création protocole de test intital
6	Création du projet, de l'activité principale (vue jour) et du layout
7	Mise en place du code pour récuperer l'horaire d'une classe. Date et classe choisie arbitrairement
8	Test de la fonctionnalité ajoutée précedemment
9	Récupération de l'horaire grâce à la classe et la date choisie par l'utilisateur.
10	Ajout des boutons permettant de changer de jour/semaine
11	Au lancement de l'application, mettre par défaut le jour actuel
12	Au lancement de l'application, mettre par défaut le nom de la dernière classe consultée.
13	Création du Menu pour pouvoir changer d'activité et effacer la liste des classes déjà recherchées.
14	Création de la seconde activité (Vue semaine) et pouvoir y accéder depuis le menu
15	Mise en place du code pour récuperer l'horaire d'une classe sur l'activité vue semaine
16	Récupération de la liste de toutes les classes disponibles et affichage comme suggestion dans le champ classe
17	Présentation et discussion avec l'expert.
18	Apparition du calendrier lors de l'appui sur le bouton calendrier
19	Test de la fonctionnalité ajoutée précedemment
20	Ajout des boutons permettant de changer de semaine (Vue semaine)
21	Mise en place du stockage en interne des noms des classes déjà recherchées. (dans les deux activités)
22	Mise en place de l'affichage de la liste des classes déjà recherchées
23	Gestion lorsque plusieurs classes sont renvoyées.
24	Mise en place de la possiblité d'effacer la liste des classes déjà recherchées dans le menu.
25	Ajout du balayage de l'écran
26	Ajout du contrôle des erreurs de saisies
27	Ajout de messages en cas d'erreurs
28	Rédaction du Mode d'emploi d'installation
29	Rédaction du mode d'emploi utilisateur
30	Création du fichier annexe
31	Création du Protocole de test et tests
32	Rédaction du Rapport
33	Rassemblement et contrôle de tous les fichiers
34	Fin du projet et livraison des fichiers au client



6. Protocole de test

Activité Vue jour

Au lancement de l'application

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Insérer automatiquement le nom de la dernière classe recherchée dans le champ classe.	Lancer l'application	Dernière classe recherchée dans le champ classe.	OK
2.	Insérer automatiquement la date actuelle dans le champ date.	Lancer l'application	Date actuelle dans le champ date.	OK
3.	Masquer le clavier pour ne pas entraver la visibilité de l'horaire	Lancer l'application	Le clavier n'apparait pas	OK
4.	Si la classe présente dans les champs classe renvoie plusieurs id (3m3i renvoie 3m3i2, 3m3i1, 3m3i3), ne pas demander de choisir la bonne classe.	Lancer l'application, taper 3m3i3 dans le champ classe puis rechercher. Dans la liste déroulante, choisir 3m3i3, puis appuyer à nouveau sur le bouton rechercher. Quitter l'application et la relancer.	Liste déroulante n'apparait pas.	OK
5.	Lancer automatiquement la recherche	Lancer l'application	Affichage de l'horaire de la classe correspondante.	OK
6.	Si l'utilisateur n'a pas de connexion internet, afficher un message d'erreur qui permet d'accéder aux paramètres	Lancer l'application en mode avion	Message d'erreur avec 2 boutons. La première permettant de continuer vers l'application et la seconde qui redirige vers les paramètres du téléphone.	OK
7.	Afficher le jour de la semaine	Lancer l'application	« Lundi » à « Dimanche » en fonction du jour actuel	OK
8.	Si Hypercool n'envoie plus aucunes données, avertir l'utilisateur.	Effectuer une recherche lorsque Hypercool est hors-ligne.	Message informant à l'utilisateur que l'horaire n'est actuellement pas disponible.	Pas pu tester

Gestion et affichage des erreurs de saisie

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Ne pas lancer la recherche et afficher un message d'erreur lorsque l'utilisateur tente de lancer la requête avec une classe vide.	Ne rien mettre dans le champ classe, puis appuyer sur le bouton rechercher.	Message d'erreur informant que le champ classe est vide. Requête non lancée.	OK

2.	Ne pas lancer la recherche et afficher un message d'erreur lorsque l'utilisateur tente de lancer la requête avec une date vide.	Ne rien mettre dans le champ date, puis appuyer sur le bouton rechercher.	Message d'erreur informant que le champ date est vide. Requête non lancée.	OK
3.	Ne pas lancer la recherche et afficher un message d'erreur lorsque l'utilisateur tente de lancer la requête avec une classe inexistante.	Insérer « Batman » dans le champ classe, puis appuyer sur le bouton rechercher.	Message d'erreur informant que la classe entrée n'existe pas. Requête non lancée.	OK
4.	Ne pas lancer la recherche et afficher un message d'erreur lorsque l'utilisateur tente de lancer la requête avec une date incorrecte	Insérer « Batman » dans le champ date, puis appuyer sur le bouton rechercher.	Message d'erreur informant que la date entrée est incorrecte. Requête non lancée.	OK
5.	Si Hypercool n'envoie plus aucunes données, avertir l'utilisateur.	Effectuer une recherche lorsque Hypercool est hors-ligne.	Message informant à l'utilisateur que l'horaire n'est actuellement pas disponible.	Pas pu tester
6.	Si champ date/classe est vide/erroné, rendre inactif les boutons --, -, +, ++ et le balayage de l'écran.	Entrer une classe/date vide/erroné, lancer la recherche puis appuyer sur les boutons --, -, +, ++. Essayer de balayer l'écran	Boutons et balayage ne changent pas la date.	OK
7.	Si champ classe est vide/erroné, ne pas mettre le contenu du champ dans l'historique.	Insérer « 3m3i2fff » et « » dans le champ classe puis appuyer sur le bouton rechercher. Appuyer sur le bouton historique pour vérifier que ce que l'on vient d'entrer n'y figure pas.	Classe fausse ou vide pas ajouté à l'historique.	OK

Interaction avec l'application

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Lors de l'appui sur le bouton calendrier, afficher un calendrier avec comme date par défaut, le jour actuel	Clic sur le bouton calendrier	Affichage du calendrier avec la date du jour actuelle par défaut.	OK
2.	Après avoir choisi la date avec le calendrier, mettre la date dans le champ date.	Clic sur le bouton calendrier, choisir une date différente du jour actuel.	Changement du champ date avec la date choisie.	OK
3.	Lors de l'appui sur le bouton « -- », enlever une semaine à la date présente dans le champ date et lancer la recherche.	Clic sur le bouton « -- »	Nouvel affichage avec cette fois ci, une semaine en moins.	OK
4.	Lors de l'appui sur le bouton « - », enlever un jour à la date présente dans le champ date et lancer la recherche.	Clic sur le bouton « - »	Nouvel affichage avec cette fois ci, un jour en moins.	OK

5.	Lors de l'appui sur le bouton « + », ajouter un jour à la date présente dans le champ date et lancer la recherche.	Clic sur le bouton « + »	Nouvel affichage avec cette fois ci, un jour en plus.	OK
6.	Lors de l'appui sur le bouton « ++ », ajouter une semaine à la date présente dans le champ date et lancer la recherche.	Clic sur le bouton « ++ »	Nouvel affichage avec cette fois ci, une semaine en plus.	OK
7.	Lors de l'appui sur le menu, proposer 2 choix: Effacer l'historique et Vue semaine.	Clic sur le menu.	Apparition des 2 choix	OK
8.	Depuis le menu, lancer l'activité vue semaine en cas de clic sur « Vue Semaine »	Clic sur le menu, puis choisir « Vue semaine »	Lancement de la vue semaine	OK
9.	Ne pas laisser l'utilisateur accéder à la vue semaine lors de l'appui sur « Vue Semaine » si celui-ci n'a pas de connexion internet.	Activer le mode avion, clic sur le menu, puis choisir Vue semaine.	Message d'erreur avec 2 boutons. La première permettant de continuer vers l'application et la seconde qui redirige vers les paramètres du téléphone.	OK
10.	Depuis le menu, effacer les recherches déjà effectuées en cas de clic sur « Effacer les recherches »	Clic sur le menu, puis choisir « Effacer les recherches ». Clic sur le bouton Historique	L'historique ne s'affiche pas car il est vide.	OK
11.	Balayer l'écran de gauche à droite pour reculer d'un jour.	Balayer l'écran de gauche à droite.	La date diminue d'un jour et lance la requête.	OK
12.	Balayer l'écran de droite à gauche pour augmenter d'un jour.	Balayer l'écran de droite à gauche.	La date augmente d'un jour et lance la requête.	OK
13.	Lorsque l'on est en mode liste classe, proposer toutes les classes disponibles pendant que l'utilisateur écrit dans le champ classe.	Lancer l'application, taper une classe dans le champ classe.	Apparition d'une liste en dessous du champ.	OK
14.	Lors de l'appui sur le bouton « Historique », changement du texte du bouton en « Classes » et afficher un message qui montre que l'on est en mode historique.	Appuyer sur le bouton Historique.	Message informant que l'utilisateur est passé en mode historique. Changement du texte du bouton, dorénavant « Classes ».	OK
15.	Lors de l'appui sur le bouton « Historique », afficher la liste déroulante avec les classes déjà recherchées, et masquer le clavier la première fois.	Appuyer sur le bouton Historique.	Affichage de la liste déroulant avec les classes déjà recherchées. Clavier non disponible la première fois.	OK
16.	Lors de l'appui sur le bouton « Classes », changement du texte du bouton en	Appuyer sur le bouton Classes.	Message informant que l'utilisateur est passé en mode affichage de toutes les classes.	OK

	« Historique » et affichage d'un message qui montre que l'on est en mode liste classes.		Changement du texte du bouton, devenant « Historique ».	
17.	L'application s'adapte à la taille de l'écran.	Lancer l'application avec plusieurs tailles d'écran différentes.	Adaptation de l'application pour que celui-ci reste agréable à utiliser.	OK

Requête

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Lors de l'appui du bouton rechercher, lancer la requête (champ date et classe juste et non vide). Affichage de l'horaire correct.	Insérer « 3m3i2 » dans le champ classe et cliquer sur le bouton rechercher.	Affichage de l'horaire de la classe correspondante.	OK
2.	Lors de l'appui du bouton rechercher, afficher une icône de chargement.	Insérer « 3m3i2 » dans le champ classe et cliquer sur le bouton rechercher.	Affichage d'une icône de chargement avant l'affichage de l'horaire.	OK

Classes renvoyant plusieurs id (ex : 3m3i renvoie 3m3i2, 3m3i3, 3m3i1)

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Afficher un message montrant qu'il y a plusieurs classes retournées et le nombre.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher.	Apparition d'un message informatif avec le nombre de classes retournées.	OK
2.	Masquer le clavier et ouvrir la liste déroulante contenant les classes retournées.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher.	Clavier non disponible et apparition de la liste déroulante avec les classes retournées.	OK
3.	Bloquer les boutons --, -, +, ++, et le balayage de l'écran.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher. Ne rien choisir dans la liste déroulante et appuyer sur les boutons --, -, +, ++ et essayer de balayer l'écran.	Boutons et balayages inactifs durant le choix de la bonne classe.	OK
4.	Après que l'utilisateur ait choisi la classe via la liste déroulante, afficher correctement l'horaire de cette classe.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher. Choisir une classe dans la liste déroulante et clic sur rechercher.	Affichage correct de l'horaire de la classe choisie.	OK

5.	Après que l'utilisateur ait choisi la classe via la liste déroulante, afficher le clavier la prochaine fois qu'il appuie sur le champ classe.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher. Choisir une classe dans la liste déroulante et clic sur rechercher. Appuyer sur le champ classe.	Apparition du clavier.	OK
6.	Après que l'utilisateur ait choisi la bonne classe via la liste déroulante, puis appuyé sur Rechercher, ne pas afficher la liste des classes retournées lorsqu'il appuie sur le bouton ++, +, - , --.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher. Choisir une classe dans la liste déroulante et clic sur rechercher. Appuyer sur les boutons ++, +, -, --	Changement de la date sans afficher la liste des classes retournées.	OK

Affichage

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Si la classe n'a pas de cours pour la date recherchée, avertir l'utilisateur	Insérer « 3m3i2 » dans le champ classe, Choisir une date qui tombe sur un dimanche et appuyer sur le bouton rechercher.	Apparition d'un message qui signale qu'il n'y a pas de cours.	OK
2.	Affichage d'un « / » à l'endroit où devait être le nom du professeur, si celui-ci n'existe pas.	Insérer « 3m3i2 » dans le champ classe, Choisir la date « 20-04-2017 » et appuyer sur le bouton rechercher.	/ à la place du nom du professeur	OK
3.	Couleurs du tableau alternés pour une meilleure visibilité.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher.	Couleurs du tableau alternés.	OK
4.	Affichage correct lorsqu'une classe à un espace dans le nom	Insérer « 1BPC AFP » dans le champ classe et appuyer sur le bouton rechercher.	Affichage correct et non pas du vide.	OK
5.	Lors du basculement en mode paysage/portrait, l'affichage reste correct.	Cliquer sur le bouton pour tourner le téléphone.	Affichage correct.	OK
6.	Lors du basculement en mode paysage/portrait, le contenu des champs classe et date sont les mêmes qu'avant d'avoir basculer.	Taper 3m3i2 et le 03-05-2017 dans les champs puis cliquer sur le bouton pour tourner le téléphone.	3m3i2 et 03-05-2017 dans les champs	KO

Activité Vue semaine

Lors de l'ouverture de l'activité

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Insérer automatiquement le nom de la dernière classe recherchée dans le champ classe.	Lancer l'activité	Dernière classe recherchée dans le champ classe.	OK
2.	Insérer automatiquement la date actuelle dans le champ date.	Lancer l'activité	Date actuelle dans le champ date.	OK
3.	Masquer le clavier pour ne pas entraver la visibilité de l'horaire	Lancer l'activité	Le clavier n'apparait pas	OK
4.	Si la classe présente dans le champ classe renvoie plusieurs id (3m3i renvoie 3m3i2, 3m3i1, 3m3i3), ne pas demander de choisir la bonne classe.	Lancer l'activité, taper 3m3i3 dans le champ classe puis rechercher. Dans la liste déroulante, choisir 3m3i3, puis appuyer à nouveau sur le bouton rechercher. Quitter l'application et la relancer.	Liste déroulante n'apparait pas.	OK
5.	Lancer automatiquement la recherche	Lancer l'activité	Affichage de l'horaire de la classe correspondante.	OK
6.	Afficher les dates de la semaine recherchée	Lancer l'activité	« Lundi 24.04.2017 à Vendredi 30.04.2017 » en fonction du jour actuel	OK

Gestion et affichage des erreurs de saisie

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Ne pas lancer la recherche et afficher un message d'erreur lorsque l'utilisateur tente de lancer la requête avec une classe vide.	Ne rien mettre dans le champ classe, puis appuyer sur le bouton rechercher.	Message d'erreur informant que le champ classe est vide. Requête non lancée.	OK
2.	Ne pas lancer la recherche et afficher un message d'erreur lorsque l'utilisateur tente de lancer la requête avec une date vide.	Ne rien mettre dans le champ date, puis appuyer sur le bouton rechercher.	Message d'erreur informant que le champ date est vide. Requête non lancée.	OK
3.	Ne pas lancer la recherche et afficher un message d'erreur lorsque l'utilisateur tente de lancer la requête avec une classe inexistante.	Insérer « Batman » dans le champ classe, puis appuyer sur le bouton rechercher.	Message d'erreur informant que la classe entrée n'existe pas. Requête non lancée.	OK

4.	Ne pas lancer la recherche et afficher un message d'erreur lorsque l'utilisateur tente de lancer la requête avec une date incorrecte	Insérer « Batman » dans le champ date, puis appuyer sur le bouton rechercher.	Message d'erreur informant que la date entrée est incorrecte. Requête non lancée.	OK
5.	Si Hypercool n'envoie plus aucunes données, avertir l'utilisateur.	Effectuer une recherche lorsque Hypercool est hors-ligne.	Message informant à l'utilisateur que l'horaire n'est actuellement pas disponible.	Pas pu tester
6.	Si champ date/classe est vide/erroné, rendre inactif le boutons --, -, +, ++ et le balayage de l'écran.	Entrer une classe/date vide/erroné, lancer la recherche puis appuyer sur les boutons --, -, +, ++.	Boutons ne changent pas la date.	OK
7.	Si champ classe est vide/erroné, ne pas mettre le contenu du champ dans l'historique.	Insérer « 3m3i2fff » et « » dans le champ classe puis appuyer sur le bouton rechercher. Appuyer sur le bouton historique pour vérifier que ce que l'on vient d'entrer n'y figure pas.	Classe fausse ou vide pas ajoutée à l'historique.	OK

Interaction avec l'application

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Lors de l'appui sur le bouton calendrier, afficher un calendrier avec comme date par défaut, le jour actuel	Clic sur le bouton calendrier	Affichage du calendrier avec la date du jour actuel par défaut.	OK
2.	Après avoir choisi la date avec le calendrier, mettre la date dans le champ date.	Clic sur le bouton calendrier, choisir une date différente du jour actuel.	Changement du champ date avec la date choisie.	OK
3.	Lors de l'appui sur le bouton « -- », enlever une semaine à la date présente dans le champ date et lancer la recherche.	Clic sur le bouton « --«	Nouvel affichage avec cette fois ci, une semaine en moins.	OK
4.	Lors de l'appui sur le bouton « ++ », ajouter une semaine à la date présente dans le champ date et lancer la recherche.	Clic sur le bouton « ++ »	Nouvel affichage avec cette fois ci, une semaine en plus.	OK
5.	Lors de l'appui sur le menu, proposer 2 choix : Effacer l'historique et Vue jour.	Clic sur le menu.	Apparition des 2 choix	OK
6.	Depuis le menu, lancer l'activité vue jour en cas de clic sur « Vue jour »	Clic sur le menu, puis choisir « Vue jour »	Lancement de la vue jour	OK
7.	Depuis le menu, effacer les recherches déjà effectuées en cas de clic sur « Effacer les recherches »	Clic sur le menu, puis choisir « Effacer les recherches ». Appuyer sur le bouton historique	L'historique ne s'affiche pas car il est vide.	OK

8.	Balayer l'écran de gauche à droite pour reculer d'une semaine.	Balayer l'écran de gauche à droite.	La date diminue d'une semaine et lance la requête.	OK
9.	Balayer l'écran de droite à gauche pour augmenter d'une semaine.	Balayer l'écran de droite à gauche.	La date augmente d'une semaine et lance la requête.	OK
10.	Lorsque l'on est en mode liste classe, proposer toutes les classes disponibles pendant que l'utilisateur écrit dans le champ classe.	Lancer l'activité, taper une classe dans le champ classe.	Apparition d'une liste en dessous du champ.	OK
11.	Lors de l'appui sur le bouton « Historique », changement du texte du bouton en « Classes » et afficher un message qui montre que l'on est en mode historique.	Appuyer sur le bouton Historique.	Message informant que l'utilisateur est passé en mode historique. Changement du texte du bouton, dorénavant « Classes ».	OK
12.	Lors de l'appui sur le bouton « Historique », afficher la liste déroulante avec les classes déjà recherchées, et masquer le clavier la première fois.	Appuyer sur le bouton Historique.	Affichage de la liste déroulant avec les classes déjà recherchées. Clavier non disponible la première fois.	OK
13.	Lors de l'appui sur le bouton « Classes », changement du texte du bouton en « Historique » et afficher un message qui montre que l'on est en mode liste classe.	Appuyer sur le bouton Classes.	Message informant que l'utilisateur est passé en mode affichage de toutes les classes. Changement du texte du bouton, devenant « Historique ».	OK
14.	L'application s'adapte à la taille de l'écran.	Lancer l'activité avec plusieurs tailles d'écrans différentes.	Adaptation de l'application pour que celui-ci reste agréable à utiliser.	OK

Requête

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Lors de l'appui du bouton rechercher, lancer la requête (champ date et classe juste et non vide). Affichage de l'horaire correct.	Insérer « 3m3i2 » dans le champ classe et cliquer sur le bouton rechercher.	Affichage de l'horaire de la classe correspondante.	OK
2.	Lors de l'appui du bouton rechercher, afficher une icone de chargement.	Insérer « 3m3i2 » dans le champ classe et cliquer sur le bouton rechercher.	Affichage d'une icone de chargement avant l'affichage de l'horaire.	OK

Classes renvoyant plusieurs id (p.ex 3m3i renvoie 3m3i2, 3m3i3, 3m3i1)

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Afficher un message montrant qu'il y a plusieurs classes retournées et le nombre.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher.	Apparition d'un message informatif avec le nombre de classes retournées.	OK
2.	Masquer le clavier et ouvrir la liste déroulante contenant les classes retournées.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher.	Clavier non disponible et apparition de la liste déroulante avec les classes retournées.	OK
3.	Bloquer les boutons --, ++, et le balayage de l'écran.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher. Ne rien choisir dans la liste déroulante et appuyer sur les boutons --, ++ et essayer de balayer l'écran.	Boutons inactifs durant le choix de la bonne classe.	OK
4.	Après que l'utilisateur ait choisi la classe via la liste déroulante, afficher correctement l'horaire de cette classe.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher. Choisir une classe dans la liste déroulante et clic sur rechercher.	Affichage correct de l'horaire de la classe choisie.	OK
5.	Après que l'utilisateur ait choisi la classe via la liste déroulante, afficher le clavier la prochaine fois qu'il appuie sur le champ classe.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher. Choisir une classe dans la liste déroulante et clic sur rechercher. Appuyer sur le champ classe.	Apparition du clavier.	OK
6.	Après que l'utilisateur ait choisi la bonne classe via la liste déroulante, puis appuyé sur Rechercher, ne pas afficher la liste des classes retournées lorsqu'il appuie sur le bouton ++, +, -, --.	Insérer « 3m3i1 » dans le champ classe et cliquer sur le bouton rechercher. Choisir une classe dans la liste déroulante et clic sur rechercher. Appuyer sur les boutons ++, +, -, --	Changement de la date sans afficher la liste des classes retournées.	OK

Affichage

N°	Description	Actions à tester	Résultats attendus	Statut
1.	Affichage d'un « / » à l'endroit où devait être le nom du professeur, si celui-ci n'existe pas.	Insérer « 3m3i2 » dans le champ classe, Choisir la date « 20-04-2017 » et appuyer sur le bouton rechercher.	/ à la place du nom du professeur	OK

2.	Affichage correct lorsqu'une classe à un espace dans le nom	Insérer « 1BPC AFP » dans le champ classe et appuyer sur le bouton rechercher.	Affichage correct et non pas du vide.	OK
3.	Les branches doivent être colorées.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Les branches sont colorées.	OK
4.	Lorsque la branche dure plus d'une période, étendre la couleur sur le nombre de période.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Nombre de cases coloriées varient en fonction du nombre de période.	OK
5.	Les mêmes branches doivent être de même couleur.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Les branches de même libellés ont la même couleur.	OK
6.	Les cases entre 12h15 et 13h10 doivent être d'une couleur différente.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Cases entre 12h15 et 13h10 de couleur grises.	OK
7.	Tout à gauche de l'écran, avoir l'heure des périodes, jusqu'à 23h00	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Heure des périodes à côté des cases.	OK
8.	Lorsque que l'on affiche le libelle d'une branche, avoir la première lettre en majuscule et les suivantes en minuscules.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Première lettre toujours en majuscule suivie de lettres en minuscules.	OK
9.	Affichage des classes ayant des périodes décalées. (Ex. commence à 8h30 au lieu de 8h10)	Insérer « 1HOPV » dans le champ classe et appuyer sur le bouton rechercher	Coloriage des périodes entre les cases.	OK
10.	Coloriage de plusieurs cases avec les classes ayant plusieurs périodes décalées.	Insérer « 1ESDA 2016-2018 » dans le champ classe et appuyer sur le bouton rechercher.	Coloriage étendu sur plusieurs périodes.	KO
11.	Lorsqu'une branche dure une seule période, afficher le code matière au lieu du libellé.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Affichage de « MATH » au lieu de « Mathématique » si la branche dure une période.	OK
12.	Lorsqu'une branche dure plus d'une période, afficher le libellé de la branche.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Affichage de « Mathématique » au lieu de « MATH » si la branche du plus d'une période.	OK

13.	Lorsque le libelle d'une branche est long, et la branche dure plus d'une période, afficher les 2 parties du libelle sur 2 textviews.	Insérer « MPESPA » dans le champ classe et appuyer sur le bouton rechercher	Les libelles longs sont découpés en 2 et les deux parties sont affichées l'un au-dessus de l'autre.	OK
14.	Lorsqu'une branche dure plus d'une période, afficher également le nom du professeur en dessous de la salle.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Affichage du nom en dessous de la salle si plus d'une période.	OK
15.	Lorsqu'une branche dure plus de deux périodes, centrer le texte verticalement.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Texte centré quand il y a plus de deux périodes.	OK
16.	Les branches de natures différentes doivent être séparées par des bordures.	Insérer « 3m3i2 » dans le champ classe et appuyer sur le bouton rechercher	Bordure qui apparait entre les branches.	KO

7. Code source

MainActivity.java

```
package ch.cpln.bayrakcimu.tpi_horaire;
import android.app.Activity;
import android.app.DatePickerDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.drawable.ShapeDrawable;
import android.graphics.drawable.shapes.RectShape;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.util.TypedValue;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.view.inputmethod.InputMethodManager;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.Writer;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Collections;
import java.util.Date;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Locale;


public class MainActivity extends AppCompatActivity implements AsyncReponse {

    String strId="";     // Variable qui va contenir l'id de la classe.
    String strContenuClasse ="";
    String strContenuDate="";
    ArrayList<String> AlistLibelleToutesClasses = new ArrayList<String>();   // Arraylist de string
qui va contenir le nom de toutes les classes existantes.

    // Ces deux variables vont être utilisées pour mettre correctement la couleur et la largeur des
textviews lors de l'affichage.
```

```
    int iCouleurTableau = 0;
    Boolean bLargeurChampsHeures=true;

    Boolean bPremiereOuverture = true;  // Cette variable servira à savoir si c'est la première fois
que l'on ouvre l'application.
    Boolean bEditTexteVide = false;
    Boolean bInternetOk = true;    // True si l'utilisateur à accès à internet.
    Boolean bAlternateur=true;     // Pour alterner la source de l'autocompletetextview.
    Boolean bAfficherContenuActv = false;   // si au clic, on doit afficher le contenu de l'actv.

    // La dernière classe recherchée va normalement se mettre dans le champ classe lors du onCreate.
Mais si le fichier qui stocke la dernière classe
    //n'existe pas encore, alors on va mettre le contenu de cette variable.
    String strClasseParDefaut="3M3I2";

    // Arraylist qui va contenir les différentes classes lorsque la requête nous renvoie plusieurs
classes.
    ArrayList<String> AlistPlusieursClasses = new ArrayList<String>();

    int iCptNombreClasse = 0;                       // Le nombre de classes recues, quand plusieurs
sont renvoyées.
    boolean bPlusieursClasseValidees = true;        // Si il faut afficher la liste des classes
retournées (quand plusieurs id reçu).
    boolean bAppelDepuisBoutonPlusMoins=false;   // Pour savoir si la requête à été lancée depuis
les boutons +, ++ , - , --.

    int iChoixTraitementOutput=0;   // Pour permettre d'aiguiller les données reçues des requêtes.
    String strDerniereClasseRecherchee= "";
    ArrayList<String> AlistHistorique = new ArrayList<String>();  // Variable qui va contenir le nom
des classes déjà recherchées.

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Mise en place du datepicker
        Button btnCalendrier = (Button) findViewById(R.id.BtnCalendrier);
        btnCalendrier.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Calendar cDateActuelle = Calendar.getInstance();
                int iAnneeActuelle = cDateActuelle.get(Calendar.YEAR);
                int iMoisActuel = cDateActuelle.get(Calendar.MONTH);
                int iJourActuel = cDateActuelle.get(Calendar.DAY_OF_MONTH);

                final DatePickerDialog datePickerDialog = new DatePickerDialog(MainActivity.this,
new DatePickerDialog.OnDateSetListener() {
                    public void onDateSet(DatePicker datepicker, int iAnneeChoisie, int iMoisChoisi,
int iJourChoisi) {

                        int iJour = iJourChoisi;
                        int iMois = iMoisChoisi;
                        int iAnnee = iAnneeChoisie;

                        String strDateComplete = String.valueOf(iJour) + "-" + String.valueOf(iMois
+ 1) + "-" + String.valueOf(iAnnee);
                        EditText etDate = (EditText) findViewById(R.id.EtDate);
                        etDate.setText(strDateComplete);

                    }
                }, iAnneeActuelle, iMoisActuel, iJourActuel);
                datePickerDialog.setTitle("Veuillez Choisir la date");
                datePickerDialog.show();
            }

        });

        strDerniereClasseRecherchee = getDerniereClasseRecherchee();

        // On va masquer le clavier lors de la première execution.
        getWindow().setSoftInputMode(
                WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN
        );
```

```
        //Test de la connectivité
        bInternetOk = EstConnecte();

        if (bInternetOk) {
            ContenuEditText();
            TraitementDate(strContenuDate, 0);  // Traitement de la date avec 0 jour, pour afficher
le jour de la semaine
            Requete();
        } else {
            AfficheErreurInternet();
        }

        //Détection du balayage horizontal de l'écran.
        // Le code à l'intérieur est le même que les boutons + et -.

        LinearLayout llPrincipal = (LinearLayout) findViewById(R.id.LlPricipal);
        llPrincipal.setOnTouchListener(new SwipeListener(MainActivity.this) {
            public void onSwipeRight() {

                if (!bPlusieursClasseValidees) {
                    if (!bEditTexteVide) {
                        bPlusieursClasseValidees = true;
                        bAppelDepuisBoutonPlusMoins = true;
                        iChoixTraitementOutput = 0;
                        TraitementDate(strContenuDate, -1);
                        Requete();
                    }
                }
            }
            public void onSwipeLeft() {

                if (!bPlusieursClasseValidees) {
                    if (!bEditTexteVide) {
                        bPlusieursClasseValidees = true;
                        bAppelDepuisBoutonPlusMoins = true;
                        iChoixTraitementOutput = 0;
                        TraitementDate(strContenuDate, 1);
                        Requete();
                    }
                }
            }
        });

        // Si l'utilisateur vient de cliquer sur le bouton pour afficher lhistorique, alors on va
afficher le contenu de l'actv.
        final AutoCompleteTextView actvClasse = (AutoCompleteTextView)
findViewById(R.id.ActvClasse);
        actvClasse.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (bAfficherContenuActv) {
                    actvClasse.showDropDown();
                    CacherClavier(getBaseContext(), actvClasse);
                }
                bAfficherContenuActv = false;

                if (bPlusieursClasseValidees) {
                    CacherClavier(getBaseContext(), actvClasse);
                    bAfficherContenuActv = true;  // Ici on remet a true sinon quand on clique, on
ne peut plus taper dans le champs.
                }

            }
        });

        Button btnPlusJour = (Button) findViewById(R.id.BtnPlusJour);
        btnPlusJour.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!bPlusieursClasseValidees) {
                    if (!bEditTexteVide) {
                        bPlusieursClasseValidees = true;
                        bAppelDepuisBoutonPlusMoins = true;
                        iChoixTraitementOutput = 0;
                        TraitementDate(strContenuDate, 1);
                        Requete();
```

```
            }
        }
    }
});
Button btnPlusSemaine = (Button) findViewById(R.id.BtnPlusSemaine);
btnPlusSemaine.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (!bPlusieursClasseValidees) {
            if (!bEditTexteVide) {
                bPlusieursClasseValidees = true;
                bAppelDepuisBoutonPlusMoins = true;
                iChoixTraitementOutput = 0;
                TraitementDate(strContenuDate, 7);
                Requete();
            }
        }
    }
});
Button btnMoinsJour = (Button) findViewById(R.id.BtnMoinsJour);
btnMoinsJour.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (!bPlusieursClasseValidees) {
            if (!bEditTexteVide) {
                bPlusieursClasseValidees = true;
                bAppelDepuisBoutonPlusMoins = true;
                iChoixTraitementOutput = 0;
                TraitementDate(strContenuDate, -1);
                Requete();
            }
        }
    }
});
Button btnMoinsSemaine = (Button) findViewById(R.id.BtnMoinsSemaine);
btnMoinsSemaine.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        if (!bPlusieursClasseValidees) {
            if (!bEditTexteVide) {
                bPlusieursClasseValidees = true;
                bAppelDepuisBoutonPlusMoins = true;
                iChoixTraitementOutput = 0;
                TraitementDate(strContenuDate, -7);
                Requete();
            }
        }
    }
});

Button btnRechercher = (Button) findViewById(R.id.BtnRechercher);
btnRechercher.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        bInternetOk = EstConnecte();
        if (bInternetOk) {
            bEditTexteVide = false;
            bAppelDepuisBoutonPlusMoins = false;
            iChoixTraitementOutput = 0;
            ContenuEditText();
            if (!bEditTexteVide) {
                TraitementDate(strContenuDate, 0);
                Requete();
            }
        } else {
            AfficheErreurInternet();
        }


    }
});

final Button btnAlternateur = (Button) findViewById(R.id.BtnAlternateur);
btnAlternateur.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {

                // En fonction de la variable, Changemenet du texte du bouton, et modification de la
source de l'actv.
                if (bAlternateur) {
                    MiseEnPlaceActv(AlistHistorique);
                    Toast.makeText(MainActivity.this, "Affichage de l'historique",
Toast.LENGTH_SHORT).show();
                    bAfficherContenuActv = true;
                    btnAlternateur.setText("Classes");
                    actvClasse.performClick();   // pour afficher la liste de l'histo.
                } else {
                    MiseEnPlaceActv(AlistLibelleToutesClasses);
                    Toast.makeText(MainActivity.this, "Affichage de toutes les classes",
Toast.LENGTH_SHORT).show();
                    btnAlternateur.setText("Historique");
                }
                bAlternateur = !bAlternateur;
            }
        });

    }

    // Fonction pour masquer le clavier.
    public static void CacherClavier(Context context, View view) {
        InputMethodManager inputMethodManager = (InputMethodManager)
context.getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(view.getWindowToken(),
InputMethodManager.HIDE_IMPLICIT_ONLY);
        inputMethodManager.hideSoftInputFromWindow(view.getApplicationWindowToken(), 0);
    }

  // Si l'utilisateur ne dispose pas de connection internet, un message d'erreur s'affiche.
    public void AfficheErreurInternet() {
        AlertDialog.Builder Alerte = new AlertDialog.Builder(MainActivity.this);
        Alerte.setCancelable(false);
        Alerte.setTitle("Pas de connexion internet");
        Alerte.setMessage("Votre appareil n'est actuellement pas connecté à internet. Veuillez
vérifier votre connexion. ");
        Alerte.setPositiveButton("Paramètres", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Intent intent = new Intent(android.provider.Settings.ACTION_SETTINGS);
                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            }
        })
                .setNegativeButton("Continuer", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                    }
                });
        final AlertDialog Message = Alerte.create();
        Message.show();
    }

    //Fonction qui va recevoir les outputs des requetes.
   public void RetourOutput(String output) {

       // En fonction de iChoixTraitementOutput, on va aiguiller le resultat.
       if (iChoixTraitementOutput == 0) {
           strId = TraitementId(output);   // strid va contenir l'id reçu : ex 4343
           if (iCptNombreClasse > 1 && !bPlusieursClasseValidees) {       // si plusieurs classes
ont été trouvées, alors on affiche la liste.
               Toast.makeText(MainActivity.this, iCptNombreClasse + " classes trouvées, veuillez
choisir la bonne classe", Toast.LENGTH_LONG).show();
               bAfficherContenuActv = true;
               MiseEnPlaceActv(AlistPlusieursClasses);
               bPlusieursClasseValidees = true;
               AutoCompleteTextView actvClasse = (AutoCompleteTextView)
findViewById(R.id.ActvClasse);
               actvClasse.performClick();

           } else {
               bAfficherContenuActv=false;  // Ca va permettre de reafficher le clavier une fois
qu'on a choisi la bonne classe, sinon on n'avait pas de clavier.
```

```
                bPlusieursClasseValidees = false;
                // Test pour voir si la classe n'a pas été trouvée.
                if ("".equals(strId) || strId == null) {
                    Toast.makeText(MainActivity.this, "Classe non trouvée",
Toast.LENGTH_SHORT).show();
                    bEditTexteVide = true;
                } else {
                    // Si id correct, alors on execute la requête suivante.
                    AsyncAffichageHoraire asyncAffichageHoraire1 = new
AsyncAffichageHoraire(MainActivity.this, 0);
                    asyncAffichageHoraire1.delegate = (AsyncReponse) this;

asyncAffichageHoraire1.execute("http://devinter.cpln.ch/pdf/hypercool/controler.php?action=horaire&i
dent=" + strId + "&sub=date&date=" + strContenuDate);
                    // Si appel depuis un bouton +, ++, -, --, alors on n'écrit pas dans le fichier,
car la classe est déjà écrite.
                    if (!bAppelDepuisBoutonPlusMoins) {
                        EcritureFichier();
                    }
                    bAppelDepuisBoutonPlusMoins = false;
                    LectureFichier();
                    if (!bAlternateur) {
                        MiseEnPlaceActv(AlistHistorique);
                    } else {
                        MiseEnPlaceActv(AlistLibelleToutesClasses);
                    }
                }
            }
        }

        // Traitement du flux reçu de la requête 2, on va mettre le nom des classes dans une
arraylists.
        if (iChoixTraitementOutput == 1) {
            TraitementToutesClasses(output);
        }
        if (iChoixTraitementOutput == 2) {
            // Si on ne recoit rien, on va afficher un message,
            if (!"[]".equals(output)) {
                AffichageHoraire(output);
            } else {
                LinearLayout llGeneral = (LinearLayout) findViewById(R.id.LlGeneral);
               // on vide l'affichage.
                llGeneral.removeAllViews();
                TextView tvJourDeLaSemaine = (TextView) findViewById(R.id.TvJourDeLaSemaine);
                tvJourDeLaSemaine.append(" - Pas de cours");
            }
        }
        iChoixTraitementOutput++;
    }


    // fonction pour savoir si l'utilisateur a internet
    public boolean EstConnecte(){
        ConnectivityManager cm =
(ConnectivityManager)getBaseContext().getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = cm.getActiveNetworkInfo();
        boolean bInternetOk = networkInfo != null && networkInfo.isConnectedOrConnecting();
        return  bInternetOk;
    }


    // Fonction pour mettre en place et mettre à jour l'autocompleteTextview
    public void MiseEnPlaceActv(ArrayList<String> Alist){
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line, Alist);
        final AutoCompleteTextView actvClasse = (AutoCompleteTextView)findViewById(R.id.ActvClasse);
        actvClasse.setThreshold(0);
        actvClasse.setAdapter(adapter);
    }


    // Ecriture de la classe dans le fichier.
    public void EcritureFichier() {

        AutoCompleteTextView actvClasse = (AutoCompleteTextView) findViewById(R.id.ActvClasse);
        String strChampsClasse = actvClasse.getText().toString();
```

```
        // On va mettre en majuscule pour éviter la redondance (3m3i2 et 3M3I2)
        strChampsClasse = strChampsClasse.toUpperCase();
        File fChemin = getBaseContext().getFilesDir();
        File fFichier = new File(fChemin, "storage.txt");
        Writer writer;
        // Si c'est vide, on ne met pas dans le fichier. Utile par exemple lors du changement
d'orientation si le champ était vide.
        if (!"".equals(strChampsClasse)) {
            try {
                writer = new BufferedWriter(new FileWriter(fFichier, true));
                writer.append(strChampsClasse + "\n");
                writer.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        // Ecriture dans le fichier dernière classe
        File fFichierDerniereClasse = new File(fChemin, "DerniereClasse.txt");

        if (!"".equals(strChampsClasse)) {
            try {
                PrintWriter pw = new PrintWriter(fFichierDerniereClasse);
                pw.close();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            Writer writer2;
            try {
                writer2 = new BufferedWriter(new FileWriter(fFichierDerniereClasse, true));
                writer2.append(strChampsClasse);
                writer2.close();
            } catch (IOException e) {
            }
        }
    }

    // Lecture du fichier et ajout des lignes dans l'arraylist
    public void LectureFichier(){
        File fChemin = getBaseContext().getFilesDir();
        File fFichier = new File(fChemin, "storage.txt");
        String strligne ="";
        try{
            BufferedReader input = new BufferedReader(new FileReader(fFichier));
            while ((strligne = input.readLine()) != null) {
                int i= 0;
                AlistHistorique.add(i, strligne);
                i++;
            }
            input.close();
        }catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        SuppressionDoublons();
    }

    //Fonction pour récupérer la dernière classe recherchée
    public String getDerniereClasseRecherchee() {

        File fChemin = getBaseContext().getFilesDir();
        String strligne = "";
        File fFichierDerniereClasse = new File(fChemin, "DerniereClasse.txt");
        try {
            BufferedReader input = new BufferedReader(new FileReader(fFichierDerniereClasse));
            while ((strligne = input.readLine()) != null) {
                strDerniereClasseRecherchee = strligne;
            }
            input.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
        return strDerniereClasseRecherchee;
    }


    //Fonction pour supprimmer les doublons de lignes dans l'arraylist contenant l'historique..
    public void SuppressionDoublons(){
        HashSet<String> hashSet = new HashSet<String>();
        hashSet.addAll(AlistHistorique);
        AlistHistorique.clear();
        AlistHistorique.addAll(hashSet);
    }

    // Ajout de jour / semaine à la date.
    public void TraitementDate(String strContenudate, int iJourAModifier) {

        try {
            SimpleDateFormat Formater = new SimpleDateFormat("dd-MM-yyyy");
            Date ObjetDate = Formater.parse(strContenudate);
            Calendar calendrier = Calendar.getInstance();
            calendrier.setTime(ObjetDate);
            calendrier.add(Calendar.DATE, iJourAModifier);
            strContenuDate = Formater.format(calendrier.getTime());
            EditText EtDate = (EditText) findViewById(R.id.EtDate);
            EtDate.setText(strContenuDate);

            // Affichage du jour de la semaine
            String strNomduJour = calendrier.getDisplayName(Calendar.DAY_OF_WEEK, Calendar.LONG,
Locale.getDefault());
            switch (strNomduJour) {
                case "Monday":
                    strNomduJour = "Lundi";
                    break;
                case "Tuesday":
                    strNomduJour = "Mardi";
                    break;
                case "Wednesday":
                    strNomduJour = "Mercredi";
                    break;
                case "Thursday":
                    strNomduJour = "Jeudi";
                    break;
                case "Friday":
                    strNomduJour = "Vendredi";
                    break;
                case "Saturday":
                    strNomduJour = "Samedi";
                    break;
                case "Sunday":
                    strNomduJour = "Dimanche";
                    break;
            }
            TextView tvJourDeLaSemaine = (TextView) findViewById(R.id.TvJourDeLaSemaine);
            tvJourDeLaSemaine.setText(strNomduJour);
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }

    // Fonction pour créer le menu
    public boolean onCreateOptionsMenu(Menu menu){
        getMenuInflater().inflate(R.menu.menu, menu);
        return true;
    }

    // Lorseque l'on clique sur un bouton du menu, on lance la fonction correspondante.
    public boolean onOptionsItemSelected(MenuItem item){
        switch (item.getItemId()){
            case R.id.action_changer_vue:
                VueSemaine();
                return true;
            case R.id.action_effacer:
                EffacerRecherches();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

```
//Si l'utilisateur n'a pas de connexion internet, on ne lui laisse pas accéder à la vue semaine.
    public void VueSemaine() {

        bInternetOk = EstConnecte();
        if (bInternetOk) {
            Intent intent = new Intent(this, Activite_VueSemaine.class);
            startActivity(intent);
        } else {
            AfficheErreurInternet();
        }
    }

    // Création d'un nouveau fichier historique, et vidage de l'arraylist.
    public void EffacerRecherches() {

        File fChemin = getBaseContext().getFilesDir();
        File fFichier = new File(fChemin, "storage.txt");
        try {
            PrintWriter pw = new PrintWriter(fFichier);
            pw.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        AlistHistorique.clear();
        LectureFichier();
        MiseEnPlaceActv(AlistLibelleToutesClasses);
        bAlternateur = true;
        Button btnAlternateur = (Button) findViewById(R.id.BtnAlternateur);
        String strTexteAlternateur = btnAlternateur.getText().toString();
        if ("Classes".equals(strTexteAlternateur)) {
            btnAlternateur.setText("Historique");
        }
    }

    // Cette fonction va mettre le contenu des champs dans des variables.
    public void ContenuEditText() {

        AutoCompleteTextView actvClasse = (AutoCompleteTextView) findViewById(R.id.ActvClasse);
        if (bPremiereOuverture) {
            if ("".equals(strDerniereClasseRecherchee)) {
                actvClasse.setText(strClasseParDefaut);
            } else {
                actvClasse.setText(strDerniereClasseRecherchee);
            }
        }
        strContenuClasse = actvClasse.getText().toString();
        if ("".equals(strContenuClasse)) {
            bEditTexteVide = true;
            Toast.makeText(MainActivity.this, "Veuillez entrer une classe",
Toast.LENGTH_SHORT).show();
        } else {
            // On remplace les espaces par %20 sinon la classe n'est pas valide.
            strContenuClasse = strContenuClasse.replace(" ", "%20");
        }
        SimpleDateFormat Formater = new SimpleDateFormat("dd-MM-yyyy");
        Calendar calendar = Calendar.getInstance();
        String strJourActuel = Formater.format(calendar.getTime());
        EditText etDate = (EditText) findViewById(R.id.EtDate);
        strContenuDate = etDate.getText().toString();
        if (bPremiereOuverture) {
            strContenuDate = strJourActuel;
        }
        if ("".equals(strContenuDate)) {
            bEditTexteVide = true;
            Toast.makeText(MainActivity.this, "Veuillez entrer une date",
Toast.LENGTH_SHORT).show();
        } else {
            etDate.setText(strContenuDate);
        }
        bPremiereOuverture = false;
    }

    // Lancement de 2 requêtes, d'abord celle pour avoir l'id de la classe, puis celle pour remplir
```

```
l'arraylist contenant les noms des classes.
    public void Requete() {
        AsyncAffichageHoraire asyncAffichageHoraire0 = new AsyncAffichageHoraire(MainActivity.this,
1);
        asyncAffichageHoraire0.delegate = (AsyncReponse) this;

asyncAffichageHoraire0.execute("http://devinter.cpln.ch/pdf/hypercool/controler.php?action=ressource
&nom=" + strContenuClasse);

        AsyncAffichageHoraire asyncAffichageHoraire2 = new AsyncAffichageHoraire(MainActivity.this,
1);
        asyncAffichageHoraire2.delegate = (AsyncReponse) this;

asyncAffichageHoraire2.execute("http://devinter.cpln.ch/pdf/hypercool/controler.php?action=ressource
&nom=");
    }

    public void TraitementToutesClasses(String strOutputlisteclasse) {
        JSONObject reader = null;
        String strCode = "";
        AlistLibelleToutesClasses.clear();

        // On va récupérer les noms des toutes les classes existantes.
        try {
            reader = new JSONObject(strOutputlisteclasse);
            Iterator iterator = reader.keys();
            for (int iCpt = 0; iCpt < 329; iCpt++) {
                strCode = (String) iterator.next();
                JSONObject jsonObject = reader.getJSONObject(strCode);
                AlistLibelleToutesClasses.add(iCpt, jsonObject.getString("nom"));
            }
        } catch (JSONException e) {
            e.printStackTrace();
            // Si cette requete ne répond pas, ca signifie que hypercool n'est pas disponible.
            Toast.makeText(MainActivity.this, "Hypercool hors-ligne, impossible de récupérer les
données.", Toast.LENGTH_SHORT).show();
        }
        HashSet<String> hashSet = new HashSet<String>();
        hashSet.addAll(AlistLibelleToutesClasses);
        AlistLibelleToutesClasses.clear();
        AlistLibelleToutesClasses.addAll(hashSet);

    }

    public void AffichageHoraire(String strOutput) {

// Création des arrays et arrayslist pour stocker les données reçues.
        String[] ArrayHeureDebut = new String[100];
        String[] ArrayHeureFin = new String[100];
        String[] ArrayHeureDebutComplet = new String[100];
        String[] ArrayHeureFinComplet = new String[100];
        String[] ArrayLibelle = new String[100];
        String[] ArrayProfesseur = new String[100];
        String[] ArraySalle = new String[100];
        Integer[] ArrayCalculHeure = new Integer[100];


        ArrayList<Integer> AListCalcul = new ArrayList<Integer>();
        ArrayList<String> AlistHeureDebutComplet = new ArrayList<String>();
        ArrayList<String> AlistHeureFinComplet = new ArrayList<String>();
        ArrayList<String> AlistLibelle = new ArrayList<String>();
        ArrayList<String> AlistProfesseur = new ArrayList<String>();
        ArrayList<String> AlistArraySalle = new ArrayList<String>();
        ArrayList<Integer> AlistCalculHeure = new ArrayList<Integer>();

        int i = 0;
        int iHeureDebut = 0;
        int iHeureFin = 0;

        try {
            JSONArray jsonArrayGeneral = new JSONArray(strOutput);
            for (i = 0; i < jsonArrayGeneral.length(); i++) {

                JSONObject jsonObjectGeneral = jsonArrayGeneral.getJSONObject(i);
                ArrayHeureDebut[i] = jsonObjectGeneral.getString("heureDebut");
                ArrayHeureFin[i] = jsonObjectGeneral.getString("heureFin");
```

```
                ArrayLibelle[i] = jsonObjectGeneral.getString("libelle");
                try {
                    JSONArray jsonArrayProf = jsonObjectGeneral.getJSONArray("professeur");
                    ArrayProfesseur[i] = jsonArrayProf.getString(0);

                } catch (JSONException e) {
                    ArrayProfesseur[i] = "/";
                }
                try {
                    JSONArray jsonArraySalle = jsonObjectGeneral.getJSONArray("salle");
                    ArraySalle[i] = jsonArraySalle.getString(0);

                } catch (JSONException e) {
                    ArraySalle[i] = "/-";
                }
                try {
                    ArrayProfesseur[i] = ArrayProfesseur[i].substring(0,
ArrayProfesseur[i].indexOf(" "));

                } catch (Exception e) {
                }
                try {
                    ArraySalle[i] = ArraySalle[i].substring(0, ArraySalle[i].indexOf("-"));
                } catch (Exception e) {

                }
                ArrayHeureDebutComplet[i] = ArrayHeureDebut[i];
                ArrayHeureFinComplet[i] = ArrayHeureFin[i];
                ArrayHeureDebut[i] = ArrayHeureDebut[i].substring(0, ArrayHeureDebut[i].length() -
3);
                ArrayHeureFin[i] = ArrayHeureFin[i].substring(0, ArrayHeureFin[i].length() - 3);

                // Ajout des données présentes dans les array aux arraylists.
                AlistHeureDebutComplet.add(i, ArrayHeureDebut[i]);
                AlistHeureFinComplet.add(i, ArrayHeureDebut[i]);
                AlistProfesseur.add(i, ArrayProfesseur[i]);
                AlistArraySalle.add(i, ArraySalle[i]);
                AlistLibelle.add(i, ArrayLibelle[i]);

                try {
                    iHeureDebut = Integer.parseInt(ArrayHeureDebut[i]);
                    iHeureFin = Integer.parseInt(ArrayHeureFin[i]);
                    ArrayCalculHeure[i] = iHeureDebut + iHeureFin;
                    AlistCalculHeure.add(i, ArrayCalculHeure[i]);
                    AListCalcul.add(i, iHeureDebut + iHeureFin);

                } catch (NumberFormatException nfe) {
                }
            }

        } catch (JSONException e) {
            e.printStackTrace();
        }

        // Tri et basculement en fonction du bon ordre des données.
        JSONArray jsonArrayGeneral = null;
        try {
            jsonArrayGeneral = new JSONArray(strOutput);
            Collections.sort(AListCalcul);    // Tri du calcul càd HeureDebut + HeureFin.
            for (int iBranche = 0; iBranche < jsonArrayGeneral.length(); iBranche++) {
                for (int iCpt = 0; iCpt < jsonArrayGeneral.length(); iCpt++) {
                    if (AlistCalculHeure.get(iCpt).equals(AListCalcul.get(iBranche))) {
                        AlistLibelle.add(iBranche, ArrayLibelle[iCpt]);
                        AlistHeureDebutComplet.add(iBranche, ArrayHeureDebutComplet[iCpt]);
                        AlistHeureFinComplet.add(iBranche, ArrayHeureFinComplet[iCpt]);
                        AlistProfesseur.add(iBranche, ArrayProfesseur[iCpt]);
                        AlistArraySalle.add(iBranche, ArraySalle[iCpt]);
                    }
                }
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }

        //Création des layouts necessaires à l'affichage.
        LinearLayout llGeneral = (LinearLayout) findViewById(R.id.LlGeneral);
```

```
        LinearLayout llHoriz = new LinearLayout(getBaseContext());
        LinearLayout llVert1 = new LinearLayout(getBaseContext());
        LinearLayout llVert2 = new LinearLayout(getBaseContext());
        LinearLayout llVert3 = new LinearLayout(getBaseContext());
        iCouleurTableau = 0;
        if ("{\"error\":\"la date n'est pas valide\"}".equals(strOutput)) {
            Toast.makeText(MainActivity.this, "Veuillez entrer une date valide",
Toast.LENGTH_SHORT).show();
            bEditTexteVide = true;
        } else {

            // Creation de l'affichage.
            llGeneral.removeAllViews();
            for (int iBranche = 0; iBranche < jsonArrayGeneral.length(); iBranche++) {
                llHoriz = CreationLayoutHoriz();
                llVert1 = CreationLayoutVertic();
                llVert2 = CreationLayoutVertic();
                llVert3 = CreationLayoutVertic();
                iCouleurTableau++;
                TextView tv1 = CreationTextView(AlistHeureDebutComplet.get(iBranche));
                TextView tv2 = CreationTextView(AlistHeureFinComplet.get(iBranche));
                bLargeurChampsHeures = false;
                TextView tv3 = CreationTextView(AlistLibelle.get(iBranche));
                TextView tv4 = CreationTextView(AlistProfesseur.get(iBranche));
                TextView tv5 = CreationTextView(AlistArraySalle.get(iBranche));
                TextView tv6 = CreationTextView(" ");
                bLargeurChampsHeures = true;
                llGeneral.addView(llHoriz);
                llHoriz.addView(llVert1);
                llHoriz.addView(llVert2);
                llHoriz.addView(llVert3);
                llVert1.addView(tv1);
                llVert1.addView(tv2);
                llVert2.addView(tv3);
                llVert2.addView(tv4);
                llVert3.addView(tv5);
                llVert3.addView(tv6);
            }
        }
    }

    // Création du TextView
    public  TextView CreationTextView (String strTexte) {

        DisplayMetrics displayMetrics = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
        int iHauteurEcran = displayMetrics.heightPixels;
        int iLargeurEcran = displayMetrics.widthPixels;
        TextView tv = new TextView(getBaseContext());
        tv.setText(strTexte);
        if (bLargeurChampsHeures) {
            tv.setWidth(iLargeurEcran / 5);
        } else {
            tv.setWidth(iLargeurEcran / 2);
        }
        tv.setHeight(iHauteurEcran / 11);
        tv.setTextSize(TypedValue.COMPLEX_UNIT_SP, 20);
        tv.setGravity(Gravity.CENTER);
        tv.setTextColor(Color.BLACK);
        return tv;
    }


    public  LinearLayout CreationLayoutHoriz() {
        LinearLayout ll = new LinearLayout(getBaseContext());
        ll.setOrientation(LinearLayout.HORIZONTAL);
        ll.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.WRAP_CONTENT));
        ShapeDrawable sd = new ShapeDrawable();
        sd.setShape(new RectShape());
        sd.getPaint().setStrokeWidth(5);
        sd.getPaint().setStyle(Paint.Style.STROKE);
        ll.setBackground(sd);
        return ll;
    }
```

```
    public LinearLayout CreationLayoutVertic() {

        LinearLayout ll = new LinearLayout(getBaseContext());
        ll.setOrientation(LinearLayout.VERTICAL);
        ll.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT));
        ShapeDrawable sd = new ShapeDrawable();
        if ((iCouleurTableau % 2) == 1) {
            sd.getPaint().setColor(Color.argb(235, 235, 235, 235));
        } else {
            sd.getPaint().setColor(Color.argb(201, 201, 201, 201));
        }
        ll.setBackground(sd);
        return ll;
    }


    // Récupération de l'id uniquement du flux json. iCptNombreclasse prend le nombre de classes
reçues.
    public String TraitementId(String strOutput) {
        AlistPlusieursClasses.clear();
        iCptNombreClasse = 0;
        JSONObject reader = null;
        String[] arrayCode = new String[1000];
        try {
            reader = new JSONObject(strOutput);
            Iterator iteratorObj = reader.keys();
            while (iteratorObj.hasNext()) {
                arrayCode[iCptNombreClasse] = (String) iteratorObj.next();
                JSONObject jsonObject = reader.getJSONObject(arrayCode[iCptNombreClasse]);
                AlistPlusieursClasses.add(iCptNombreClasse, jsonObject.getString("nom"));
                iCptNombreClasse++;
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return arrayCode[0];
    }

}
```

Activite_VueSemaine.java

```
package ch.cpln.bayrakcimu.tpi_horaire;
import android.app.Activity;
import android.app.DatePickerDialog;
import android.content.Context;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.TypedValue;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.view.inputmethod.InputMethodManager;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedReader;
```

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.Writer;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
import java.util.Date;
import java.util.HashSet;
import java.util.Iterator;

public class Activite_VueSemaine extends AppCompatActivity implements AsyncReponse {

    // Déclaration des variables globales
    String strId="";
    String[] Outputs = new String[10];   // Tableau qui va contenir les outputs de la requete 3
    ArrayList<String> AlistLibelleToutesClasses = new ArrayList<String>();

    String strContenuClasse ="";
    String strContenuDate="";
    int iChoixTraitementOutput=0;

    Boolean bPremiereOuverture = true;
    Boolean bEditTexteVide = false;
    Boolean bDateIncorrecte = false;
    Boolean bAlternateur = true;
    Boolean bAfficherContenuActv = false;

    ArrayList<String> AlistHistorique = new ArrayList<String>();
    ArrayList<String> AlistDateSemaine = new ArrayList<String>();  // ArrayList qui va contenir les
dates de la semaine actuelle

    String[] libelleBranche = new String[100];  // Variable qui va être utilisée pour mettre la même
couleur aux mêmes branches,
    int iNouvelleBranche=0;
    int iNumCouleur=0;
    boolean bAppelDepuisBoutonPlusMoins=false;

    int iCptNombreClasse = 0;
    boolean bPlusieursClasseValidees = true;
    ArrayList<String> AlistPlusieursClasses = new ArrayList<String>();
    String strDerniereClasseRecherchee= "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_activite__vue_semaine);

//Mise en place du datepicker
        Button btnCalendrier2 = (Button)findViewById(R.id.BtnCalendrier2);
        btnCalendrier2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Calendar cDateActuelle=Calendar.getInstance();
                int iAnneeActuelle = cDateActuelle.get(Calendar.YEAR);
                int iMoisActuel=cDateActuelle.get(Calendar.MONTH);
                int iJourActuel=cDateActuelle.get(Calendar.DAY_OF_MONTH);

                final DatePickerDialog mDatePicker=new DatePickerDialog(Activite_VueSemaine.this,
new DatePickerDialog.OnDateSetListener() {
                    public void onDateSet(DatePicker datepicker, int AnneeChoisie, int MoisChoisi,
int JourChoisi) {

                        int  iJour = JourChoisi;
                        int iMois = MoisChoisi;
                        int iAnnee = AnneeChoisie;

                        String strDateComplet = String.valueOf(iJour) + "-" + String.valueOf(iMois +
```

```
1) + "-" + String.valueOf(iAnnee);
                        EditText etDate2 = (EditText)findViewById(R.id.EtDate2);
                        etDate2.setText(strDateComplet);

                    }
                },iAnneeActuelle, iMoisActuel, iJourActuel);
                mDatePicker.setTitle("Choisir la date");
                mDatePicker.show();
            }

        });

//Obligation de passer en orientation paysage
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

        // on n'affiche pas le clavier lors de la première fois.
        getWindow().setSoftInputMode(
                WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN
        );

        strDerniereClasseRecherchee = getDerniereClasseRecherchee();

        // Si dans le pire des cas strDerniereClasseRecherchee et vide (meme si ca ne devrait pas
arriver) alors on met une classe arbitrairement pour ne pas faire crash l'application.
        if(strDerniereClasseRecherchee==null || "".equals(strDerniereClasseRecherchee)) {
            strDerniereClasseRecherchee = "3m3i2";
        }
        ContenuEditText();
        DateEnSemaine(strContenuDate);
        Requete();

        Button btnPlusSemaine2 = (Button)findViewById(R.id.BtnPlusSemaine2);
        btnPlusSemaine2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(!bPlusieursClasseValidees) {
                    if (!bEditTexteVide) {
                        bPlusieursClasseValidees = true;
                        bAppelDepuisBoutonPlusMoins = true;
                        iChoixTraitementOutput = 0;
                        if (!bDateIncorrecte) {
                            TraitementDate(strContenuDate, 7);
                            DateEnSemaine(strContenuDate);
                            Requete();
                        }
                    }
                }
            }
        });

        Button btnMoinsSemaine2 = (Button)findViewById(R.id.BtnMoinsSemaine2);
        btnMoinsSemaine2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(!bPlusieursClasseValidees) {
                    if (!bEditTexteVide) {
                        bPlusieursClasseValidees = true;
                        bAppelDepuisBoutonPlusMoins = true;
                        iChoixTraitementOutput = 0;
                        if (!bDateIncorrecte) {
                            TraitementDate(strContenuDate, -7);
                            DateEnSemaine(strContenuDate);
                            Requete();
                        }
                    }
                }
            }
        });

        Button btnRechercher2 = (Button)findViewById(R.id.BtnRechercher2);
        btnRechercher2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                bDateIncorrecte=false;
                bEditTexteVide=false;
                bAppelDepuisBoutonPlusMoins=false;
```

```
                iChoixTraitementOutput=0;
                ContenuEditText();
                if(!bEditTexteVide) {
                    DateEnSemaine(strContenuDate);
                    if(!bDateIncorrecte){
                    Requete();
                    }
                }

            }
        });

        final AutoCompleteTextView actvClasse2 =
(AutoCompleteTextView)findViewById(R.id.ActvClasse2);
        actvClasse2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(bAfficherContenuActv){
                    actvClasse2.showDropDown();
                    CacherClavier(getBaseContext(), actvClasse2);
                }
                bAfficherContenuActv=false;
                if(bPlusieursClasseValidees) {
                    CacherClavier(getBaseContext(), actvClasse2);
                    bAfficherContenuActv=true;

                }
            }
        });

        final Button btnAlternateur2 = (Button)findViewById(R.id.BtnAlternateur2);
        btnAlternateur2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(bAlternateur){

                    MiseEnPlaceActv(AlistHistorique);
                    Toast.makeText(Activite_VueSemaine.this,"Affichage Historique",
Toast.LENGTH_SHORT).show();
                    bAfficherContenuActv = true;
                    btnAlternateur2.setText("Classes");
                    actvClasse2.performClick();

                }else{
                    MiseEnPlaceActv(AlistLibelleToutesClasses);
                    Toast.makeText(Activite_VueSemaine.this,"Affichage toutes les classes",
Toast.LENGTH_SHORT).show();
                    btnAlternateur2.setText("Historique");
                }
                bAlternateur = !bAlternateur;
            }
        });

        // Mise en place du swipe, code pareil que les bouton --, ++
        LinearLayout llPrincipal2 = (LinearLayout)findViewById(R.id.LlPricipal2);
        llPrincipal2.setOnTouchListener(new SwipeListener(Activite_VueSemaine.this) {

            public void onSwipeRight() {
                if(!bPlusieursClasseValidees) {
                    if (!bEditTexteVide) {
                        bPlusieursClasseValidees = true;
                        bAppelDepuisBoutonPlusMoins = true;
                        iChoixTraitementOutput = 0;
                        if (!bDateIncorrecte) {
                            TraitementDate(strContenuDate, -7);
                            DateEnSemaine(strContenuDate);
                            Requete();
                        }
                    }
                }

            }
            public void onSwipeLeft() {
                if(!bPlusieursClasseValidees){
                    if(!bEditTexteVide){
                        bPlusieursClasseValidees=true;
```

```
                        bAppelDepuisBoutonPlusMoins=true;
                        iChoixTraitementOutput=0;
                        if(!bDateIncorrecte){
                            TraitementDate(strContenuDate, 7);
                            DateEnSemaine(strContenuDate);
                            Requete();
                        }
                    }
                }
            }

        });

    }

// Fonction pour cacher le clavier
    public static void CacherClavier(Context context, View view) {
        InputMethodManager inputMethodManager = (InputMethodManager)
context.getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(view.getWindowToken(),
InputMethodManager.HIDE_IMPLICIT_ONLY);
        inputMethodManager.hideSoftInputFromWindow(view.getApplicationWindowToken(), 0);
    }

    public void RetourOutput(String strOutput){

        //avec iChoixTraitementOutput, on va aiguiller le resultat.
        if(iChoixTraitementOutput==0){
            strId = TraitementId(strOutput);
            if(iCptNombreClasse>1 && !bPlusieursClasseValidees) {
                Toast.makeText(Activite_VueSemaine.this, iCptNombreClasse + " classes trouvées,
veuillez choisir la bonne classe", Toast.LENGTH_LONG).show();
                bAfficherContenuActv = true;
                MiseEnPlaceActv(AlistPlusieursClasses);
                bPlusieursClasseValidees = true;
                AutoCompleteTextView actvClasse2 = (AutoCompleteTextView)
findViewById(R.id.ActvClasse2);
                actvClasse2.performClick();
            }
            else {
                bAfficherContenuActv=false;  // permet de remettre le clavier.
                bPlusieursClasseValidees = false;
                if ("".equals(strId) || strId == null) {
                    Toast.makeText(Activite_VueSemaine.this, "Classe non trouvée",
Toast.LENGTH_SHORT).show();
                    bEditTexteVide = true;
                } else {
                    // Si tout est bon, lancement 5 requêtes.
                    AsyncAffichageHoraire[] asyncGeneral = new AsyncAffichageHoraire[8];
                    for (int i = 0; i < 5; i++) {
                        asyncGeneral[i] = new AsyncAffichageHoraire(Activite_VueSemaine.this, 0);
                        asyncGeneral[i].delegate = (AsyncReponse) this;

asyncGeneral[i].execute("http://devinter.cpln.ch/pdf/hypercool/controler.php?action=horaire&ident="
+ strId + "&sub=date&date=" + AlistDateSemaine.get(i));
                    }
                    if (!bAppelDepuisBoutonPlusMoins) {
                        EcritureFichier();
                    }
                    bAppelDepuisBoutonPlusMoins = false;
                    LectureFichier();
                    if (!bAlternateur) {
                        MiseEnPlaceActv(AlistHistorique);
                    } else {
                        MiseEnPlaceActv(AlistLibelleToutesClasses);
                    }
                }
            }
        }
        if(iChoixTraitementOutput==1){
            TraitementToutesClasses(strOutput);
        }
        // A chaque fois que les données de l'horaire arrivent, on met dans des tableaux
        if(iChoixTraitementOutput==2 ||iChoixTraitementOutput==3 ||iChoixTraitementOutput==4
||iChoixTraitementOutput==5 ||iChoixTraitementOutput==6){
            Outputs[iChoixTraitementOutput-2] = strOutput;
```

```
        }
        // Une fois que toutes les requêtes ont été effectuées, on fait appel à la fonction
AffichageHoraire.
        if(iChoixTraitementOutput==6) {
            AffichageHoraire(Outputs[0], Outputs[1], Outputs[2], Outputs[3], Outputs[4]);
            TextView tvJourDeLaSemaine2 = (TextView)findViewById(R.id.TvJourDeLaSemaine2);
            if(!bDateIncorrecte) {
                tvJourDeLaSemaine2.setText("Lundi " + AlistDateSemaine.get(0) + " au Vendredi " +
AlistDateSemaine.get(6));
            }
        }
        iChoixTraitementOutput++;
    }

    // Fonction pour mettre en place et mettre à jour l'actv.
    public void MiseEnPlaceActv(ArrayList<String> Alist){
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line, Alist);
        AutoCompleteTextView actvClasse2 = (AutoCompleteTextView)
                findViewById(R.id.ActvClasse2);
        actvClasse2.setThreshold(0);
        actvClasse2.setAdapter(adapter);
    }

    public void TraitementDate(String strContenudate, int iJourAModifier){

        // Fonction permettant d'ajouter des semaines à la date actuelle
        Date ObjetDate = null;
        try {
            SimpleDateFormat Formater = new SimpleDateFormat("dd-MM-yyyy");
            ObjetDate = Formater.parse(strContenudate);
            Calendar calendrier = Calendar.getInstance();
            calendrier.setTime(ObjetDate);
            calendrier.add(Calendar.DATE,iJourAModifier);
            strContenuDate = Formater.format(calendrier.getTime());
            EditText etDate = (EditText)findViewById(R.id.EtDate2);
            etDate.setText(strContenuDate);
        } catch (ParseException e) {
            e.printStackTrace();
        }

    }

    public void ContenuEditText(){

        // Obtention des contenus des edittexts pour les mettre dans des variables.
        AutoCompleteTextView actvClasse2 =(AutoCompleteTextView)findViewById(R.id.ActvClasse2);
        if(bPremiereOuverture){
            actvClasse2.setText(strDerniereClasseRecherchee);
        }
        strContenuClasse = actvClasse2.getText().toString();
        if("".equals(strContenuClasse)) {
            bEditTexteVide = true;
            Toast.makeText(Activite_VueSemaine.this, "Veuillez entrer une classe",
Toast.LENGTH_LONG).show();
        }else {
            // remplacement des espaces par %20 sinon il ne reconnaissait pas la classe.
            strContenuClasse = strContenuClasse.replace(" ", "%20");
        }

        SimpleDateFormat Formater = new SimpleDateFormat("dd-MM-yyyy");
        Calendar calendar = Calendar.getInstance();
        String strJourActuel = Formater.format(calendar.getTime());
        EditText etDate2 = (EditText)findViewById(R.id.EtDate2);
        strContenuDate = etDate2.getText().toString();

        //Si c'est la première fois qu'on fait la requete ( ouverture de l'application) alors on
mets le jour actuel dans la variable contenuDate.
        if(bPremiereOuverture) {
            strContenuDate = strJourActuel;
        }
        if("".equals(strContenuDate)){
            bEditTexteVide=true;
            Toast.makeText(Activite_VueSemaine.this,"Veuillez entrer une date",
Toast.LENGTH_LONG).show();
        }else {
```

```
            etDate2.setText(strContenuDate);
        }
        bPremiereOuverture = false;
    }

    public void Requete(){

        AsyncAffichageHoraire asyncAffichageHoraire2 = new
AsyncAffichageHoraire(Activite_VueSemaine.this, 1);
        asyncAffichageHoraire2.delegate = (AsyncReponse) this;

asyncAffichageHoraire2.execute("http://devinter.cpln.ch/pdf/hypercool/controler.php?action=ressource
&nom=" + strContenuClasse);

        AsyncAffichageHoraire asyncAffichageHoraire = new
AsyncAffichageHoraire(Activite VueSemaine.this, 1);
        asyncAffichageHoraire.delegate = (AsyncReponse) this;

asyncAffichageHoraire.execute("http://devinter.cpln.ch/pdf/hypercool/controler.php?action=ressource&
nom=");

    }
    public void TraitementToutesClasses(String strOutputlisteclasse){

        JSONObject reader = null;
        String strCode ="";
        AlistLibelleToutesClasses.clear();
        try {
            reader = new JSONObject(strOutputlisteclasse);
            Iterator iterator = reader.keys();
            for(int i=0;i<329;i++) {
                strCode = (String) iterator.next();
                JSONObject jsonObject = reader.getJSONObject(strCode);
                AlistLibelleToutesClasses.add(i,jsonObject.getString("nom"));
            }
        } catch (JSONException e) {
            e.printStackTrace();
            // Si cette requête ne renvoie rien, alors hypercool est hors ligne.
            Toast.makeText(Activite_VueSemaine.this,"Hypercool hors-ligne, impossible de récupérer
les données.", Toast.LENGTH_SHORT).show();
        }

        HashSet<String> hashSet = new HashSet<String>();
        hashSet.addAll(AlistLibelleToutesClasses);
        AlistLibelleToutesClasses.clear();
        AlistLibelleToutesClasses.addAll(hashSet);
    }

    public void AffichageHoraire(String strOutputJ1, String strOutputJ2, String strOutputJ3, String
strOutputJ4, String strOutputJ5) {

        //Création des arraylists qui vont contenir toutes les données par thème (prof, salle...)

        ArrayList<String>[] alistGeneralProf = new ArrayList[10];
        for(int i = 0;i<5;i++){
            alistGeneralProf[i] = new ArrayList<>();
        }
        ArrayList<String>[] alistGeneralSalle = new ArrayList[10];
        for(int i= 0;i<5;i++){
            alistGeneralSalle[i] = new ArrayList<>();
        }
        ArrayList<String>[] alistGeneralCodeMatiere = new ArrayList[10];
        for(int i=0;i<5;i++){
            alistGeneralCodeMatiere[i] = new ArrayList<>();
        }
        ArrayList<String>[] alistGeneralLibelle = new ArrayList[10];
        for(int i=0;i<5;i++){
            alistGeneralLibelle[i] = new ArrayList<>();
        }
        ArrayList<String>[] alistGeneralHeureDebutComplet = new ArrayList[10];
        for(int i=0;i<5;i++){
            alistGeneralHeureDebutComplet[i] = new ArrayList<>();
        }
        ArrayList<String>[] alistGeneralHeureFinComplet = new ArrayList[10];
        for(int i=0;i<5;i++){
            alistGeneralHeureFinComplet[i] = new ArrayList<>();
```

```
        }
        ArrayList<Integer>[] alistCalcul = new ArrayList[10];
        for(int i=0;i<5;i++){
            alistCalcul[i] = new ArrayList<>();
        }
        ArrayList<Integer>[] alistCalculHeure = new ArrayList[10];
        for(int i=0; i<5;i++){
            alistCalculHeure[i] = new ArrayList<>();
        }

        // Création du tableau de couleur pour les branches.
        int[] arrayCouleurs = new int[100];
        arrayCouleurs[0] = Color.rgb(240,236,103);
        arrayCouleurs[1] = Color.rgb(153,255,255);
        arrayCouleurs[2] = Color.rgb(204,255,204);
        arrayCouleurs[3] = Color.rgb(255,204,255);
        arrayCouleurs[4] = Color.rgb(255,102,178);
        arrayCouleurs[5] = Color.rgb(200,59,245);
        arrayCouleurs[6] = Color.rgb(255,51,123);
        arrayCouleurs[7] = Color.rgb(255,100,200);
        arrayCouleurs[8] = Color.rgb(222,76,51);
        arrayCouleurs[9] = Color.rgb(210,234,51);
        arrayCouleurs[10] = Color.rgb(10,145,51);
        arrayCouleurs[11] = Color.rgb(255,51,54);
        arrayCouleurs[12] = Color.rgb(255,51,151);
        arrayCouleurs[13] = Color.rgb(100,51,251);
        arrayCouleurs[14] = Color.rgb(255,21,71);
        arrayCouleurs[15] = Color.rgb(143,51,51);
        arrayCouleurs[16] = Color.rgb(56,51,51);
        arrayCouleurs[17] = Color.rgb(125,51,51);
        arrayCouleurs[18] = Color.rgb(255,65,510);
        arrayCouleurs[19] = Color.rgb(200,123,45);
        arrayCouleurs[20] = Color.rgb(0,200,43);
        arrayCouleurs[21] = Color.rgb(213,255,0);

        //tableau des données des 5 jours.
        String[] output = {strOutputJ1,strOutputJ2,strOutputJ3,strOutputJ4,strOutputJ5};

        // Création des arrays car plus simple d'utilisation.
        String[][] arrayLibelle = new String[100][100];
        String[][] arrayProf = new String[100][100];
        String[][] arrayHeureDebut = new String[100][100];
        String[][] arrayHeureFin = new String[100][100];
        String[][] arraySalle = new String[100][100];
        String[][] arrayHeureDebutComplet = new String[100][100];
        String[][] arrayHeureFinComplet = new String[100][100];
        Integer[][] arrayCalcul = new Integer[100][100];
        String[][] arrayCodeMatiere = new String[100][100];

        int iHeureDebut = 0;
        int iHeureFin = 0;

        // Le nombre de branches reçues pour une journée
        Integer[] NombreInfoRecu = new Integer[100];

        if("{\"error\":\"la date n'est pas valide\"}".equals(strOutputJ1)){
            Toast.makeText(Activite_VueSemaine.this,"Veuillez entrer une date valide",
Toast.LENGTH_SHORT).show();
            bDateIncorrecte=true;
        }else {
            try {

                // 2 fonction pour mettre le contenu des branches de tous les jours.
                for (int iJour = 0; iJour < 5; iJour++) {

                    JSONArray jsonArraygeneral2 = new JSONArray(output[iJour]);
                    for (int iNbBranche = 0; iNbBranche < jsonArraygeneral2.length(); iNbBranche++)
{

                        JSONArray jsonArrayGeneral = new JSONArray(output[iJour]);
                        JSONObject jsonObjectGeneral = jsonArrayGeneral.getJSONObject(iNbBranche);
                        arrayLibelle[iJour][iNbBranche] = jsonObjectGeneral.getString("libelle");
                        arrayLibelle[iJour][iNbBranche] =
arrayLibelle[iJour][iNbBranche].substring(0, 1).toUpperCase() +
arrayLibelle[iJour][iNbBranche].substring(1).toLowerCase();
                        arrayCodeMatiere[iJour][iNbBranche] =
```

```
jsonObjectGeneral.getString("codeMatiere");
                        arrayHeureDebut[iJour][iNbBranche] =
jsonObjectGeneral.getString("heureDebut");
                        arrayHeureFin[iJour][iNbBranche] = jsonObjectGeneral.getString("heureFin");

                        // Si pas de prof, alors on met un / sinon ca fait crash.
                        try {
                            JSONArray jsonArrayProf = jsonObjectGeneral.getJSONArray("professeur");
                            arrayProf[iJour][iNbBranche] = jsonArrayProf.getString(0);
                        } catch (JSONException e) {
                            arrayProf[iJour][iNbBranche] = "/";
                        }
                        try {
                            JSONArray jsonArraySalle = jsonObjectGeneral.getJSONArray("salle");
                            arraySalle[iJour][iNbBranche] = jsonArraySalle.getString(0);
                        } catch (JSONException e) {
                            arraySalle[iJour][iNbBranche] = "/-";
                        }
                        // On parse les profs pour avoir uniquement le nom.
                        try {
                            arrayProf[iJour][iNbBranche] = arrayProf[iJour][iNbBranche].substring(0,
arrayProf[iJour][iNbBranche].indexOf(" "));
                        } catch (Exception e) {

                        }
                        // On parse la salle car le code contient des nombres inutiles (B106B-
sdasda)
                        try {
                            arraySalle[iJour][iNbBranche] =
arraySalle[iJour][iNbBranche].substring(0, arraySalle[iJour][iNbBranche].indexOf("-"));
                        } catch (Exception e) {

                        }
                        arrayHeureDebutComplet[iJour][iNbBranche] =
arrayHeureDebut[iJour][iNbBranche];
                        arrayHeureFinComplet[iJour][iNbBranche] = arrayHeureFin[iJour][iNbBranche];
                        arrayHeureDebut[iJour][iNbBranche] =
arrayHeureDebut[iJour][iNbBranche].substring(0, arrayHeureDebut[iJour][iNbBranche].length() - 3);
                        arrayHeureFin[iJour][iNbBranche] =
arrayHeureFin[iJour][iNbBranche].substring(0, arrayHeureFin[iJour][iNbBranche].length() - 3);
                        try {
                            iHeureDebut = Integer.parseInt(arrayHeureDebut[iJour][iNbBranche]);
                            iHeureFin = Integer.parseInt(arrayHeureFin[iJour][iNbBranche]);
                            arrayCalcul[iJour][iNbBranche] = iHeureDebut + iHeureFin;
                        } catch (NumberFormatException e) {

                        }
                        // Passage des données depuis les arrays aux arraylists.

                        alistGeneralProf[iJour].add(iNbBranche, arrayProf[iJour][iNbBranche]);
                        alistGeneralSalle[iJour].add(iNbBranche, arraySalle[iJour][iNbBranche]);
                        alistGeneralLibelle[iJour].add(iNbBranche, arrayLibelle[iJour][iNbBranche]);
                        alistGeneralHeureDebutComplet[iJour].add(iNbBranche,
arrayHeureDebut[iJour][iNbBranche]);
                        alistGeneralHeureFinComplet[iJour].add(iNbBranche,
arrayHeureFinComplet[iJour][iNbBranche]);
                        alistCalcul[iJour].add(iNbBranche, arrayCalcul[iJour][iNbBranche]);
                        alistCalculHeure[iJour].add(iNbBranche, arrayCalcul[iJour][iNbBranche]);
                        alistGeneralCodeMatiere[iJour].add(iNbBranche,
arrayCodeMatiere[iJour][iNbBranche]);

                        NombreInfoRecu[iJour] = iNbBranche + 1;

                    }
                }

                // Si on ne recoit aucunes données, alors on met à 0
                for (int i = 0; i < 5; i++) {
                    if (NombreInfoRecu[i] == null || NombreInfoRecu[i].equals("null")) {
                        NombreInfoRecu[i] = 0;
                    }
                }

                // Tri des arraylists en fonction du calcul.
                for (int i = 0; i < 5; i++) {
                    Collections.sort(alistCalcul[i]);
```

```
                }

                // Basculement des places en fonction du calcul,

                for (int iJour = 0; iJour < 5; iJour++) {
                    for (int i2 = 0; i2 < NombreInfoRecu[iJour]; i2++) {
                        for (int i = 0; i < NombreInfoRecu[iJour]; i++) {
                            if (alistCalculHeure[iJour].get(i).equals(alistCalcul[iJour].get(i2))) {
                                alistGeneralLibelle[iJour].add(i2, arrayLibelle[iJour][i]);
                                alistGeneralHeureDebutComplet[iJour].add(i2,
arrayHeureDebutComplet[iJour][i]);
                                alistGeneralHeureFinComplet[iJour].add(i2,
arrayHeureFinComplet[iJour][i]);
                                alistGeneralProf[iJour].add(i2, arrayProf[iJour][i]);
                                alistGeneralSalle[iJour].add(i2, arraySalle[iJour][i]);
                                alistGeneralCodeMatiere[iJour].add(i2, arrayCodeMatiere[iJour][i]);
                            }
                        }
                    }
                }

                // Tableau de toutes les périodes pour pouvoir les utiliser pour savoir le nombre de
période.
                String[] arrayPeriodes = {"07h25", "08h10", "08h55", "09h40", "10h45", "11h30",
"12h15", "13h10", "13h55", "14h55", "15h40", "16h25", "17h30", "18h15", "19h00", "20h00", "20h45",
"21h30", "22h15"};
                String[] arrayJourSemaine = {"Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi"};

                // On va mettre les layouts du xml dans le ll[]
                final LinearLayout[][] llGeneral = new LinearLayout[100][100];

                for (int iJour = 0; iJour < 5; iJour++) {
                    for (int iPeriode = 0; iPeriode < 19; iPeriode++) {
                        String Idtv = "ll_" + arrayJourSemaine[iJour] + "_" +
arrayPeriodes[iPeriode];
                        int iResID = getResources().getIdentifier(Idtv, "id", getPackageName());
                        llGeneral[iJour][iPeriode] = (LinearLayout) findViewById(iResID);
                    }
                }

                TextView[][] tvGeneralLibelle = new TextView[200][200];
                for (int iJour = 0; iJour < 5; iJour++) {
                    for (int iPeriode = 0; iPeriode < 19; iPeriode++) {
                        String Idtv = "tv_" + arrayJourSemaine[iJour] + "_" +
arrayPeriodes[iPeriode] + "_libelle";
                        int iResID = getResources().getIdentifier(Idtv, "id", getPackageName());
                        tvGeneralLibelle[iJour][iPeriode] = (TextView) findViewById(iResID);
                        tvGeneralLibelle[iJour][iPeriode].setTextSize(TypedValue.COMPLEX_UNIT_DIP,
13);
                    }
                }
                TextView[][] tvGeneralSalle = new TextView[200][200];

                for (int iJour = 0; iJour < 5; iJour++) {
                    for (int iPeriode = 0; iPeriode < 19; iPeriode++) {
                        String Idtv = "tv_" + arrayJourSemaine[iJour] + "_" +
arrayPeriodes[iPeriode] + "_salle";
                        int iResID = getResources().getIdentifier(Idtv, "id", getPackageName());
                        tvGeneralSalle[iJour][iPeriode] = (TextView) findViewById(iResID);
                        tvGeneralSalle[iJour][iPeriode].setTextSize(TypedValue.COMPLEX_UNIT_DIP,
13);
                    }
                }

                // Vidage de toutes les cases.
                for (int iJour = 0; iJour < 5; iJour++) {
                    for (int iPeriode = 0; iPeriode < 19; iPeriode++) {
                        llGeneral[iJour][iPeriode].setBackgroundColor(Color.TRANSPARENT);
                        llGeneral[iJour][iPeriode].setBackgroundResource(R.drawable.border);
                        tvGeneralLibelle[iJour][iPeriode].setBackgroundColor(Color.TRANSPARENT);
                        tvGeneralSalle[iJour][iPeriode].setBackgroundColor(Color.TRANSPARENT);
                        tvGeneralLibelle[iJour][iPeriode].setText("");
                        tvGeneralSalle[iJour][iPeriode].setText("");
                    }
                }
```

```
                // Couleur midi a cause de la pause.
                for (int i = 0; i < 5; i++) {

llGeneral[i][6].setBackground(getResources().getDrawable(R.drawable.border_midi));
                }

                // Selon l'heure début on va envoyer le numero du layout à la fonction
RemplissageCases.
                for (int iJour = 0; iJour < 5; iJour++) {
                    for (int iBranche = 0; iBranche < NombreInfoRecu[iJour]; iBranche++) {
                        for (int iCpt = 0; iCpt < 19; iCpt++) {
                            if
(alistGeneralHeureDebutComplet[iJour].get(iBranche).equals(arrayPeriodes[iCpt])) {
                                int iNumeroll = iCpt;
                                String strHeureDebut = arrayPeriodes[iCpt];
                                RemplissageCases(iJour, iBranche, iNumeroll, strHeureDebut,
llGeneral, arrayPeriodes, alistGeneralHeureFinComplet, tvGeneralLibelle, tvGeneralSalle,
alistGeneralLibelle, alistGeneralSalle, alistGeneralProf, alistGeneralCodeMatiere, arrayCouleurs);
                            }
                        }
                    }
                }

            //traitement des périodes qui commencent à des heures non conventionelles.
                String[][] arrayPeriodeEntre = new String[100][100];

                arrayPeriodeEntre[0][0] = "07h30";
                arrayPeriodeEntre[0][1] = "07h35";
                arrayPeriodeEntre[0][2] = "07h40";
                arrayPeriodeEntre[0][3] = "07h45";
                arrayPeriodeEntre[0][4] = "07h50";
                arrayPeriodeEntre[0][5] = "07h55";
                arrayPeriodeEntre[0][6] = "08h00";
                arrayPeriodeEntre[0][7] = "08h05";

                arrayPeriodeEntre[1][0] = "08h15";
                arrayPeriodeEntre[1][1] = "08h20";
                arrayPeriodeEntre[1][2] = "08h25";
                arrayPeriodeEntre[1][3] = "08h30";
                arrayPeriodeEntre[1][4] = "08h35";
                arrayPeriodeEntre[1][5] = "08h40";
                arrayPeriodeEntre[1][6] = "08h45";
                arrayPeriodeEntre[1][7] = "08h50";

                arrayPeriodeEntre[2][0] = "09h00";
                arrayPeriodeEntre[2][1] = "09h05";
                arrayPeriodeEntre[2][2] = "09h10";
                arrayPeriodeEntre[2][3] = "09h15";
                arrayPeriodeEntre[2][4] = "09h20";
                arrayPeriodeEntre[2][5] = "09h25";
                arrayPeriodeEntre[2][6] = "09h30";
                arrayPeriodeEntre[2][7] = "09h35";

                arrayPeriodeEntre[3][0] = "09h45";
                arrayPeriodeEntre[3][1] = "09h50";
                arrayPeriodeEntre[3][2] = "09h55";
                arrayPeriodeEntre[3][3] = "10h00";
                arrayPeriodeEntre[3][4] = "10h05";
                arrayPeriodeEntre[3][5] = "10h10";
                arrayPeriodeEntre[3][6] = "10h15";
                arrayPeriodeEntre[3][7] = "10h20";

                arrayPeriodeEntre[4][0] = "10h50";
                arrayPeriodeEntre[4][1] = "10h55";
                arrayPeriodeEntre[4][2] = "11h00";
                arrayPeriodeEntre[4][3] = "11h05";
                arrayPeriodeEntre[4][4] = "11h10";
                arrayPeriodeEntre[4][5] = "11h15";
                arrayPeriodeEntre[4][6] = "11h20";
                arrayPeriodeEntre[4][7] = "11h25";

                arrayPeriodeEntre[5][0] = "11h35";
                arrayPeriodeEntre[5][1] = "11h40";
                arrayPeriodeEntre[5][2] = "11h45";
                arrayPeriodeEntre[5][3] = "11h50";
                arrayPeriodeEntre[5][4] = "11h55";
```

```
arrayPeriodeEntre[5][5] = "12h00";
arrayPeriodeEntre[5][6] = "12h05";
arrayPeriodeEntre[5][7] = "12h10";

arrayPeriodeEntre[6][0] = "12h20";
arrayPeriodeEntre[6][1] = "12h25";
arrayPeriodeEntre[6][2] = "12h30";
arrayPeriodeEntre[6][3] = "12h35";
arrayPeriodeEntre[6][4] = "12h40";
arrayPeriodeEntre[6][5] = "12h45";
arrayPeriodeEntre[6][6] = "12h50";
arrayPeriodeEntre[6][7] = "12h55";

arrayPeriodeEntre[7][0] = "13h15";
arrayPeriodeEntre[7][1] = "13h20";
arrayPeriodeEntre[7][2] = "13h25";
arrayPeriodeEntre[7][3] = "13h30";
arrayPeriodeEntre[7][4] = "13h35";
arrayPeriodeEntre[7][5] = "13h40";
arrayPeriodeEntre[7][6] = "13h45";
arrayPeriodeEntre[7][7] = "13h50";

arrayPeriodeEntre[8][0] = "14h00";
arrayPeriodeEntre[8][1] = "14h05";
arrayPeriodeEntre[8][2] = "14h10";
arrayPeriodeEntre[8][3] = "14h15";
arrayPeriodeEntre[8][4] = "14h20";
arrayPeriodeEntre[8][5] = "14h25";
arrayPeriodeEntre[8][6] = "14h30";
arrayPeriodeEntre[8][7] = "14h35";

arrayPeriodeEntre[9][0] = "15h00";
arrayPeriodeEntre[9][1] = "15h05";
arrayPeriodeEntre[9][2] = "15h10";
arrayPeriodeEntre[9][3] = "15h15";
arrayPeriodeEntre[9][4] = "15h20";
arrayPeriodeEntre[9][5] = "15h25";
arrayPeriodeEntre[9][6] = "15h30";
arrayPeriodeEntre[9][7] = "15h35";

arrayPeriodeEntre[10][0] = "15h45";
arrayPeriodeEntre[10][1] = "15h50";
arrayPeriodeEntre[10][2] = "15h55";
arrayPeriodeEntre[10][3] = "16h00";
arrayPeriodeEntre[10][4] = "16h05";
arrayPeriodeEntre[10][5] = "16h10";
arrayPeriodeEntre[10][6] = "16h15";
arrayPeriodeEntre[10][7] = "16h20";

arrayPeriodeEntre[11][0] = "16h30";
arrayPeriodeEntre[11][1] = "16h35";
arrayPeriodeEntre[11][2] = "16h40";
arrayPeriodeEntre[11][3] = "16h45";
arrayPeriodeEntre[11][4] = "16h50";
arrayPeriodeEntre[11][5] = "16h55";
arrayPeriodeEntre[11][6] = "17h00";
arrayPeriodeEntre[11][7] = "17h05";

arrayPeriodeEntre[12][0] = "17h35";
arrayPeriodeEntre[12][1] = "17h40";
arrayPeriodeEntre[12][2] = "17h45";
arrayPeriodeEntre[12][3] = "17h50";
arrayPeriodeEntre[12][4] = "17h55";
arrayPeriodeEntre[12][5] = "18h00";
arrayPeriodeEntre[12][6] = "18h05";
arrayPeriodeEntre[12][7] = "18h10";

arrayPeriodeEntre[13][0] = "18h20";
arrayPeriodeEntre[13][1] = "18h25";
arrayPeriodeEntre[13][2] = "18h30";
arrayPeriodeEntre[13][3] = "18h35";
arrayPeriodeEntre[13][4] = "18h40";
arrayPeriodeEntre[13][5] = "18h45";
arrayPeriodeEntre[13][6] = "18h50";
arrayPeriodeEntre[13][7] = "18h55";
```

```
                // on va tester si l'heure de début est la même, tout d'abord les variations
08h15... puis la layout, 09h00, puis le jour.
                for(int iJour=0;iJour<5;iJour++) {
                    for (int iBranche = 0; iBranche < NombreInfoRecu[iJour]; iBranche++) {
                        for (int iPeriode = 0; iPeriode < 19; iPeriode++) {   // le nombre de
périodes au total.
                            for (int i = 0; i < 8; i++) { // 8 = le nombre de la seconde dimension
du tableau.

if(alistGeneralHeureDebutComplet[iJour].get(iBranche).equals(arrayPeriodeEntre[iPeriode][i])){
                                    int iNumeroll = iPeriode;
                                    RemplissageCases2(iJour, iBranche,
iNumeroll,tvGeneralLibelle,tvGeneralSalle,alistGeneralLibelle,alistGeneralSalle,alistGeneralCodeMati
ere,arrayCouleurs);
                                }
                            }
                        }
                    }
                }

                iNouvelleBranche = 0;
                iNumCouleur = 0;
                libelleBranche = new String[libelleBranche.length];
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }

    public void RemplissageCases(int iJour, int iBranche, int iNuml1, String strHeureDebut,
LinearLayout[][] llGeneral, String[] arrayPeriode, ArrayList<String>[] AlistHeureFin,
                                 TextView[][] tvLibelle, TextView[][] tvSalle, ArrayList<String>[]
AlistLibelle, ArrayList<String>[] AlistSalle, ArrayList<String>[] AlistProf,
                                 ArrayList<String>[] AlistCodeMatiere, int[] arrayCouleurs) {

        // Si même branche, alors même couleur

        if (Arrays.asList(libelleBranche).contains(AlistLibelle[iJour].get(iBranche))) {
            iNumCouleur = Arrays.asList(libelleBranche).indexOf(AlistLibelle[iJour].get(iBranche));
        } else {
            libelleBranche[iNouvelleBranche] = AlistLibelle[iJour].get(iBranche);
            iNouvelleBranche++;
            iNumCouleur = Arrays.asList(libelleBranche).indexOf(AlistLibelle[iJour].get(iBranche));
        }

        int iNumeroll = iNuml1;
        int iNombreCharMax = 15;    // APrès ce nombre de caractère, on va coupé le libelle en 2
parties.
        llGeneral[iJour][iNumeroll].setBackgroundColor(arrayCouleurs[iNumCouleur]);
        int iIndex1 = 0;
        int iIndex2 = 0;
        int iIndexFinal = 0;    // Va être égal au nombre de périodes
        iIndex1 = Arrays.asList(arrayPeriode).indexOf(strHeureDebut);
        String strHeureFin = AlistHeureFin[iJour].get(iBranche);
        iIndex2 = Arrays.asList(arrayPeriode).indexOf(strHeureFin);
        iIndexFinal = iIndex2 - iIndex1;
        tvLibelle[iJour][iNumeroll].setText(AlistCodeMatiere[iJour].get(iBranche));
        tvSalle[iJour][iNumeroll].setText(AlistSalle[iJour].get(iBranche));
        int iLength = AlistLibelle[iJour].get(iBranche).length();
        String strPartie1 = "";
        String strPartie2 = "";
        if (iLength > iNombreCharMax) {
            strPartie1 = AlistLibelle[iJour].get(iBranche).substring(0, iNombreCharMax);
            strPartie2 = AlistLibelle[iJour].get(iBranche).substring(iNombreCharMax);
        }
        //Coloriage des cases en fonction du nombre de période.
        for (int i = 1; i < iIndexFinal; i++) {
            llGeneral[iJour][iNumeroll + i].setBackgroundColor(arrayCouleurs[iNumCouleur]);

        }

        //traitement au cas par cas
        if (iIndexFinal == 2) {
            if (iLength > iNombreCharMax) {

                tvLibelle[iJour][iNumeroll].setText(strPartie1 + "-");
```

```
                tvSalle[iJour][iNumeroll].setText(strPartie2);
                tvLibelle[iJour][iNumeroll + 1].setText(AlistSalle[iJour].get(iBranche));
                tvSalle[iJour][iNumeroll + 1].setText(AlistProf[iJour].get(iBranche));
            } else {
                tvLibelle[iJour][iNumeroll].setText(AlistLibelle[iJour].get(iBranche));
                tvLibelle[iJour][iNumeroll + 1].setText(AlistProf[iJour].get(iBranche));
            }
        }
        if (iIndexFinal == 3 || iIndexFinal == 4) {
            if (iLength > iNombreCharMax) {
                tvLibelle[iJour][iNumeroll].setText("");
                tvSalle[iJour][iNumeroll].setText("");
                tvSalle[iJour][iNumeroll].setText(strPartie1 + "-");
                tvLibelle[iJour][iNumeroll + 1].setText(strPartie2);
                tvSalle[iJour][iNumeroll + 1].setText(AlistSalle[iJour].get(iBranche));
                tvLibelle[iJour][iNumeroll + 2].setText(AlistProf[iJour].get(iBranche));
            } else {
                tvLibelle[iJour][iNumeroll].setText("");
                tvSalle[iJour][iNumeroll].setText("");
                tvLibelle[iJour][iNumeroll + 1].setText(AlistLibelle[iJour].get(iBranche));
                tvSalle[iJour][iNumeroll + 1].setText(AlistSalle[iJour].get(iBranche));
                tvLibelle[iJour][iNumeroll + 2].setText(AlistProf[iJour].get(iBranche));
            }
        }
        if (iIndexFinal == 5) {
            if (iLength > iNombreCharMax) {
                tvLibelle[iJour][iNumeroll].setText("");
                tvSalle[iJour][iNumeroll].setText("");
                tvSalle[iJour][iNumeroll + 1].setText(strPartie1 + "-");
                tvLibelle[iJour][iNumeroll + 2].setText(strPartie2);
                tvSalle[iJour][iNumeroll + 2].setText(AlistSalle[iJour].get(iBranche));
                tvLibelle[iJour][iNumeroll + 3].setText(AlistProf[iJour].get(iBranche));
            } else {
                tvLibelle[iJour][iNumeroll].setText("");
                tvSalle[iJour][iNumeroll].setText("");
                tvLibelle[iJour][iNumeroll + 2].setText(AlistLibelle[iJour].get(iBranche));
                tvSalle[iJour][iNumeroll + 2].setText(AlistSalle[iJour].get(iBranche));
                tvLibelle[iJour][iNumeroll + 3].setText(AlistProf[iJour].get(iBranche));
            }
        }
    }

//Fonction pour les périodes aux heures non conventionnelles.
    public void RemplissageCases2(int iJour, int iBranche, int iNumll,
                                  TextView[][] tvLibelle, TextView[][] tvSalle, ArrayList<String>[]
AlistLibelle, ArrayList<String>[] AlistSalle,
                                  ArrayList<String>[] AlistCodeMatiere, int[] arrayCouleurs) {

        if (Arrays.asList(libelleBranche).contains(AlistLibelle[iJour].get(iBranche))) {
            iNumCouleur = Arrays.asList(libelleBranche).indexOf(AlistLibelle[iJour].get(iBranche));
        } else {
            libelleBranche[iNouvelleBranche] = AlistLibelle[iJour].get(iBranche);
            iNouvelleBranche++;
            iNumCouleur = Arrays.asList(libelleBranche).indexOf(AlistLibelle[iJour].get(iBranche));
        }

        tvSalle[iJour][iNumll].setText(AlistCodeMatiere[iJour].get(iBranche));
        tvLibelle[iJour][iNumll+1].setText(AlistSalle[iJour].get(iBranche));
        tvSalle[iJour][iNumll].setBackgroundColor(arrayCouleurs[iNumCouleur]);
        tvLibelle[iJour][iNumll+1].setBackgroundColor(arrayCouleurs[iNumCouleur]);
    }


    // Récupération de l'id

    public String TraitementId(String strOutput) {

        AlistPlusieursClasses.clear();
        iCptNombreClasse = 0;
        JSONObject reader = null;
        String[] arrayCode = new String[1000];
        try {
            reader = new JSONObject(strOutput);
            Iterator iteratorObj = reader.keys();
            while (iteratorObj.hasNext()) {
```

```
                arrayCode[iCptNombreClasse] = (String) iteratorObj.next();
                JSONObject jsonObject = reader.getJSONObject(arrayCode[iCptNombreClasse]);
                AlistPlusieursClasses.add(iCptNombreClasse, jsonObject.getString("nom"));
                iCptNombreClasse++;
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return arrayCode[0];
    }

    // Fonction pour avoir toutes les dates de la semaine en fonction de la date envoyée.

    public void DateEnSemaine(String strDateatraiter) {

        SimpleDateFormat Formater = new SimpleDateFormat("dd-MM-yyyy");
        Date ObjetDate = null;
        try {
            ObjetDate = Formater.parse(strDateatraiter);

        } catch (ParseException e) {
            e.printStackTrace();
            bDateIncorrecte = true;
            Toast.makeText(Activite_VueSemaine.this, "Veuillez entrer une date valide",
Toast.LENGTH_LONG).show();
        }
        Calendar calendrier = Calendar.getInstance();
        try {
            calendrier.setTime(ObjetDate);
        } catch (Exception e) {
        }
        calendrier.add(Calendar.DAY_OF_MONTH, Calendar.MONDAY -
calendrier.get(Calendar.DAY_OF_WEEK));
        String strDatelundi = Formater.format(calendrier.getTime());

        //Remplissage du tableau avec les 7 dates
        for (int i = 0; i < 7; i++) {
            AlistDateSemaine.add(i, strDatelundi);
            calendrier.add(Calendar.DATE, 1);
            strDatelundi = Formater.format(calendrier.getTime());
        }
    }

    public boolean onCreateOptionsMenu(Menu menu){
        getMenuInflater().inflate(R.menu.menu2, menu);
        return true;
    }
    public boolean onOptionsItemSelected(MenuItem item){
        switch (item.getItemId()){
            case R.id.action_changer_vue:
                VueJour();
                return true;
            case R.id.action_effacer:
                EffacerRecherches();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
    public void VueJour(){

        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
    public void EffacerRecherches() {

        File fChemin = getBaseContext().getFilesDir();
        File fFichier = new File(fChemin, "storage.txt");
        try {
            PrintWriter pw = new PrintWriter(fFichier);
            pw.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        AlistHistorique.clear();
        LectureFichier();
```

```
        MiseEnPlaceActv(AlistLibelleToutesClasses);
        bAlternateur = true;
        Button btnAlternateur2 = (Button) findViewById(R.id.BtnAlternateur2);
        String strTexteAlternateur = btnAlternateur2.getText().toString();
        if ("Classes".equals(strTexteAlternateur)) {
            btnAlternateur2.setText("Historique");
        }

    }
    public void EcritureFichier() {
        AutoCompleteTextView actvClasse2 = (AutoCompleteTextView) findViewById(R.id.ActvClasse2);
        String strChampsClasse2 = actvClasse2.getText().toString();
        strChampsClasse2 = strChampsClasse2.toUpperCase();
        File fChemin = getBaseContext().getFilesDir();
        File fFichier = new File(fChemin, "storage.txt");
        Writer writer;

        if (!"".equals(strChampsClasse2)) {
            try {
                writer = new BufferedWriter(new FileWriter(fFichier, true));
                writer.append(strChampsClasse2 + "\n");
                writer.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        //Fichier derniere classe.
        File fFichierDerniereClasse = new File(fChemin, "DerniereClasse.txt");
        if (!"".equals(strChampsClasse2)) {
            try {
                PrintWriter pw = new PrintWriter(fFichierDerniereClasse);
                pw.close();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            Writer writer2;

            try {
                writer2 = new BufferedWriter(new FileWriter(fFichierDerniereClasse, true));
                writer2.append(strChampsClasse2);
                writer2.close();
            } catch (IOException e) {
            }
        }
    }


    public String getDerniereClasseRecherchee() {

        File fChemin = getBaseContext().getFilesDir();
        String strLigne = "";
        File fFichierDerniereClasse = new File(fChemin, "DerniereClasse.txt");
        try {
            BufferedReader input = new BufferedReader(new FileReader(fFichierDerniereClasse));
            while ((strLigne = input.readLine()) != null) {
                strDerniereClasseRecherchee = strLigne;
            }
            input.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return strDerniereClasseRecherchee;
    }

    public void LectureFichier() {

        File fChemin = getBaseContext().getFilesDir();
        File fFichier = new File(fChemin, "storage.txt");
        String strLigne = "";
        try {
            BufferedReader input = new BufferedReader(new FileReader(fFichier));
            while ((strLigne = input.readLine()) != null) {
                int i = 0;
```

```
                AlistHistorique.add(i, strLigne);
                i++;
            }
            input.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        SuppressionDoublons();
    }
    public void SuppressionDoublons(){
        HashSet<String> hashSet = new HashSet<String>();
        hashSet.addAll(AlistHistorique);
        AlistHistorique.clear();
        AlistHistorique.addAll(hashSet);
    }
}
```

SwipeListener.java

```
package ch.cpln.bayrakcimu.tpi_horaire;

import android.content.Context;
import android.view.GestureDetector;
import android.view.GestureDetector.SimpleOnGestureListener;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;

public class SwipeListener implements OnTouchListener {

    private final GestureDetector gestureDetector;

    public SwipeListener (Context ctx){
        gestureDetector = new GestureDetector(ctx, new GestureListener());
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        return gestureDetector.onTouchEvent(event);
    }
    private final class GestureListener extends SimpleOnGestureListener {
        private static final int SWIPE_THRESHOLD = 100;
        private static final int SWIPE_VELOCITY_THRESHOLD = 100;
        @Override
        public boolean onDown(MotionEvent e) {
            return true;
        }
        @Override
        public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
            boolean result = false;
            try {
                float diffY = e2.getY() - e1.getY();
                float diffX = e2.getX() - e1.getX();
                if (Math.abs(diffX) > Math.abs(diffY)) {
                    if (Math.abs(diffX) > SWIPE_THRESHOLD && Math.abs(velocityX) >
SWIPE_VELOCITY_THRESHOLD) {
                        if (diffX > 0) {
                            onSwipeRight();
                        } else {
                            onSwipeLeft();
                        }
                        result = true;
                    }
                }
            } catch (Exception exception) {
                exception.printStackTrace();
            }
            return result;
        }
    }
    public void onSwipeRight() {
```

```
    }
    public void onSwipeLeft() {
    }
}
```

AsyncAffichageHoraire.java

```
package ch.cpln.bayrakcimu.tpi_horaire;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;
import android.widget.TextView;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

/**
 * Created by BayrakciMu on 27.03.2017.
 */
public class AsyncAffichageHoraire extends AsyncTask<String, String, String> {
    HttpURLConnection conn;
    URL url = null;

    public AsyncReponse delegate = null;
    ProgressDialog dialog;


    Activity activite;
    public int iNum;
    public AsyncAffichageHoraire(Activity a, int iNumrecu)
    {
        this.activite = a;
        dialog = new ProgressDialog(activite);
        iNum = iNumrecu;
    }

    public void onPreExecute() {
        if(iNum==0){
            this.dialog.setMessage("Réception des données...");
            this.dialog.setCancelable(false);
            this.dialog.show();
        }
    }



    @Override
    protected String doInBackground(String... params) {


        try {
            url = new URL(params[0]);

        } catch (MalformedURLException e) {
            e.printStackTrace();
            return "exception1";
        }

        try {
            conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("POST");
            conn.setDoInput(true);
            conn.setDoOutput(true);
            conn.connect();
```

```
        } catch (IOException e) {
            e.printStackTrace();
            return "exception2";
        }

        try {
            int iCode_reponse = conn.getResponseCode();

            if (iCode_reponse == HttpURLConnection.HTTP_OK) {
                InputStream input = conn.getInputStream();
                BufferedReader reader = new BufferedReader(new InputStreamReader(input));
                StringBuilder result = new StringBuilder();
                String strLigne;


                while ((strLigne = reader.readLine()) != null) {
                    result.append(strLigne);
                }
                return (result.toString());


            } else {
                return ("unsuccessful");
            }
        } catch (IOException e) {
            e.printStackTrace();
            return "exception3";
        } finally {
            conn.disconnect();
        }

    }


    @Override
    protected void onPostExecute(String result) {

            dialog.dismiss();
            delegate.RetourOutput(result);

    }

}
```

AsyncReponse.java

```
package ch.cpln.bayrakcimu.tpi_horaire;

/**
 * Created by BayrakciMu on 25.04.2017.
 */
public interface AsyncReponse {
    void RetourOutput(String output);
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<ScrollView android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="ch.cpln.bayrakcimu.tpi_horaire.MainActivity"
    android:orientation="vertical"
    android:id="@+id/LlPricipal">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:paddingTop="20dp"
    android:weightSum="3">
    <AutoCompleteTextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:id="@+id/ActvClasse"
        android:layout_weight="2"/>
    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:id="@+id/BtnAlternateur"
        android:text="@string/phistorique"
        android:textAllCaps="false"
        android:layout_weight="1"
        />
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:weightSum="3">
    <EditText
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:id="@+id/EtDate"
        android:layout_weight="2"
        />
    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:id="@+id/BtnCalendrier"
        android:text="@string/pcalendrier"
        android:textAllCaps="false"
        android:layout_gravity="center"
        android:layout_weight="1"
        />
</LinearLayout>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:id="@+id/BtnRechercher"
        android:textAllCaps="false"
        android:text="@string/prechercher"
        />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:weightSum="12">
        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/BtnMoinsSemaine"
            android:layout_weight="3"
            android:text="@string/dateMoinsMoins"
            android:textSize="22dp"/>
        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/BtnMoinsJour"
            android:layout_weight="3"
            android:text="@string/dateMoins"
            android:textSize="22dp"/>
        <Button
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/BtnPlusJour"
            android:layout_weight="3"
            android:text="@string/datePlus"
            android:textSize="22dp"/>
        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/BtnPlusSemaine"
            android:layout_weight="3"
            android:text="@string/datePlusPlus"
            android:textSize="22dp"/>
    </LinearLayout>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/TvTest"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/TvJourDeLaSemaine"
        android:paddingBottom="10dp"
        android:paddingLeft="13dp"
        android:textSize="22dp"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:weightSum="12"
        android:orientation="vertical"
        android:id="@+id/LlGeneral">
    </LinearLayout>
</LinearLayout>
</ScrollView>
```

activity_activite_vue_semaine.xml :

Vu la longueur du fichier, les parties similaires sont remplacées par des «… ».

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/LlPricipal2"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="ch.cpln.bayrakcimu.tpi_horaire.Activite_VueSemaine"
     android:orientation="vertical">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:weightSum="3"
   >
    <AutoCompleteTextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:id="@+id/ActvClasse2"
        android:text="3m3i2"
        android:layout_weight="2"
        />
    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:id="@+id/BtnAlternateur2"
        android:text="@string/phistorique"
        android:textAllCaps="false"
```

```
        android:layout_weight="1"
        />
</LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:weightSum="3">
        <EditText
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:id="@+id/EtDate2"
            android:text="30-03-2017"
            android:layout_weight="2"
            />
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:id="@+id/BtnCalendrier2"
            android:text="@string/pcalendrier"
            android:textAllCaps="false"
            android:layout_weight="1"
            />
    </LinearLayout>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:id="@+id/BtnRechercher2"
        android:text="@string/prechercher"
        android:textAllCaps="false"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:weightSum="2">
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:id="@+id/BtnMoinsSemaine2"
            android:text="@string/dateMoinsMoins"/>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:id="@+id/BtnPlusSemaine2"
            android:text="@string/datePlusPlus"/>
    </LinearLayout>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/TvJourDeLaSemaine2"
        android:textSize="30dp"
        android:paddingTop="10dp"/>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:weightSum="32">
    <LinearLayout
        android:paddingTop="25dp"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_weight="2"
        >
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="center"
            >
            <TextView
                android:layout_width="match_parent"
```

```
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="@string/p07h25"
                android:textSize="12dp"
                />
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="center"
            >
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="@string/p08h10"
                android:textSize="12dp"
                />
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"/>
        </LinearLayout>
```

…

seul le contenu du texte change, jusqu'à 23h00

…

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center"
        >
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="@string/p22h15"
            android:textSize="12dp"
            />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center"
        >
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="@string/p23h00"
            android:textSize="12dp"
            />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"/>
    </LinearLayout>
</LinearLayout>
<LinearLayout
    android:layout_width="0dp"
```

```
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_weight="6"
    android:paddingTop="35dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@drawable/border"
        android:id="@+id/ll_Lundi_07h25">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Lundi_07h25_libelle"
            android:gravity="center"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Lundi_07h25_salle"
            android:gravity="center"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@drawable/border"
        android:id="@+id/ll_Lundi_08h10">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Lundi_08h10_libelle"
            android:gravity="center"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Lundi_08h10_salle"
            android:gravity="center"/>
    </LinearLayout>
```

…

Seul l'id change, jusqu'à 22h15

…

```
  <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@drawable/border"
        android:id="@+id/ll_Lundi_21h30">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Lundi_21h30_libelle"
            android:gravity="center"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Lundi_21h30_salle"
            android:gravity="center"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@drawable/border"
        android:id="@+id/ll_Lundi_22h15">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Lundi_22h15_libelle"
            android:gravity="center"/>
        <TextView
            android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:id="@+id/tv_Lundi_22h15_salle"
            android:gravity="center"/>
    </LinearLayout>
</LinearLayout>
<LinearLayout
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_weight="6"
    android:paddingTop="35dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@drawable/border"
        android:id="@+id/ll_Mardi_07h25">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Mardi_07h25_libelle"
            android:gravity="center"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Mardi_07h25_salle"
            android:gravity="center"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@drawable/border"
        android:id="@+id/ll_Mardi_08h10">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Mardi_08h10_libelle"
            android:gravity="center"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tv_Mardi_08h10_salle"
            android:gravity="center"/>
    </LinearLayout>
```

…

Pareil pour le mardi, mercredi, jeudi et vendredi, seul l’id des views change.

…

```
<LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                android:background="@drawable/border"
                android:id="@+id/ll_Vendredi_21h30">
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:id="@+id/tv_Vendredi_21h30_libelle"
                    android:gravity="center"/>
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:id="@+id/tv_Vendredi_21h30_salle"
                    android:gravity="center"/>
            </LinearLayout>
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                android:background="@drawable/border"
                android:id="@+id/ll_Vendredi_22h15">
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
```

```
                    android:id="@+id/tv_Vendredi_22h15_libelle"
                    android:gravity="center"/>
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:id="@+id/tv_Vendredi_22h15_salle"
                    android:gravity="center"/>
            </LinearLayout>
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
</ScrollView>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ch.cpln.bayrakcimu.tpi_horaire">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Hypercool Reader"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:configChanges="orientation|keyboardHidden|screenSize"
        >
        <activity android:name=".MainActivity"   android:label="@string/TitrevueJour">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Activite_VueSemaine"
android:label="@string/TitrevueSemaine"></activity>
    </application>

</manifest>
```

Border.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle" >
    <solid android:color="@android:color/white" />
    <stroke android:width="1px" android:color="#000000"/>
</shape>
```

Border_midi.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle" >
    <solid android:color="@android:color/darker_gray" />
    <stroke android:width="1px" android:color="#000000"/>
</shape>
```

Menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">


    <item
        android:id="@+id/action_changer_vue"
        android:title="Vue semaine"
        app:showAsAction="never" />
```

```
    <item
        android:id="@+id/action_effacer"
        android:title="Effacer les recherches"
        app:showAsAction="never" />

</menu>
```

Menu2.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">


    <item
        android:id="@+id/action_changer_vue"
        android:title="Vue jour"
        app:showAsAction="never" />

    <item
        android:id="@+id/action_effacer"
        android:title="Effacer les recherches"
        app:showAsAction="never" />

</menu>
```

Colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Strings.xml

```
<resources>
    <string name="app_name">TPI_horaire</string>
    <string name="datePlus">+</string>
    <string name="datePlusPlus">++</string>
    <string name="dateMoins"> -</string>
    <string name="dateMoinsMoins">--</string>
    <string name="p07h25"> 07h25</string>
    <string name="p08h10"> 08h10</string>
    <string name="p08h55"> 08h55</string>
    <string name="p09h40"> 09h40</string>
    <string name="p10h45"> 10h45</string>
    <string name="p11h30"> 11h30</string>
    <string name="p12h15"> 12h15</string>
    <string name="p13h10"> 13h10</string>
    <string name="p13h55"> 13h55</string>
    <string name="p14h55"> 14h55</string>
    <string name="p15h40"> 15h40</string>
    <string name="p16h25"> 16h25</string>
    <string name="p17h30"> 17h30</string>
    <string name="p18h15"> 18h15</string>
    <string name="p19h00">19h00</string>
    <string name="p20h00"> 20h00</string>
    <string name="p20h45"> 20h45</string>
    <string name="p21h30"> 21h30</string>
    <string name="p22h15"> 22h15</string>
    <string name="p23h00"> 23h00</string>
    <string name="TitrevueJour">Vue jour</string>
    <string name="TitrevueSemaine">Vue semaine</string>
    <string name="prechercher">Rechercher</string>
    <string name="phistorique">Historique</string>
```

```
    <string name="pcalendrier">Calendrier</string>
</resources>
```

Styles.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```