|  |  |  |  |
| --- | --- | --- | --- |
| **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. Ca 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). | | |  |
| Visa: | | |  |