Projet Modélisation d’une BDD + SQL Serveur

1/ Paramétrage et mise en œuvre d’un dépôt

1° Aller sur Github et créer un nouveau «New Repository ».



2° Remplir l’information sur cette page :

* Nom du dépôt
* Description (en option)
* Public ou Privé (L’option privé est payante)

Et une fois terminé, cliquer sur “Create Repository.”

3° Créez un README pour votre dépôt

4° Installer un client git en graphique

Quelques exemples :

- Github desktop

- SourceTree

Github :



5° Client git

* Sur la gauche, il y a la liste des dépôts que le client suit ; il est possible d’ajouter un dépôt (soit en le clonant, soit en l’attachant localement) en cliquant sur l’icône « + » en haut de la zone.
* Au centre, il y a la zone d’entrée de *commit* qui permet d’entrer un message de validation et de sélectionner les fichiers qui devraient être inclus. Sous Windows, l’historique de validation est affiché directement en dessous ; sous Mac, c’est un onglet séparé.
* À droite, il y a une vue de diff qui montre ce qui a changé dans le répertoire de travail ou les modifications qui ont été incluses dans le *commit* sélectionné.
* La dernière chose à noter est le bouton « Sync » en haut à droite qui est le moyen principal d’interagir via le réseau.

2/ Gestion des issues, des branches, merging de branches.

Les issues

Une fois sur Github, il suffit de cliquer sur l’icône pour avoir accès aux issues.





Les labels

On peut également mettre des labels sur les issues de Github, très utiles pour déterminer visuellement s'il s'agit d'un bug à fixer ou d'une amélioration à effectuer, si le travail concerne le serveur ou l'application Android ...



Les assignations

Il est possible d'assigner chaque issue à un contributeur : cela permet à chacun de déterminer rapidement ce qu'il a à faire, et cela évite que deux personnes travaillent sur la même chose pour se rendre compte à la fin qu'elles ont restitué le même travail et qu'elles se retrouvent avec un conflit à régler.

Les branches

Les branches stables

Une bonne idée est d'utiliser la branche master comme branche la plus stable, pour laquelle le projet fonctionne assurément bien. On pourra utiliser une autre branche comme un "laboratoire" : on y fusionnera les modifications apportées. En cas de gros problème, cela permettra d'avoir toujours une version propre du projet : la branche master.

Les branches de développement

Lorsqu'un contributeur veut résoudre une issue, il va tout d'abord s'assurer qu'il a la dernière version de la branche develop :

*git checkout develop*

*git pull origin develop*

Puis il va créer une nouvelle branche :

*`git checkout -b nouvelle\_branche*

Il pourra alors effectuer toutes les modifications nécessaires avant de les sauvegarder :

*cd racine\_du\_depot*

*git add --all .*

*git commit -m "Message explicatif"*

*git push origin nouvelle\_branche*

Le contributeur peut alors se rendre sur Github pour créer une pull request en comparant nouvelle\_branche à develop :



Merging de branches

1° Sous le nom de référentiel, cliquer sur Pull requests.

2° Dans la liste «Pull requests», cliquez sur la requête d'extraction que vous souhaitez fusionner.

3° Selon les options de fusion activées pour votre référentiel, vous pouvez:

* Fusionnez tous vos commits dans votre branche de base en cliquant sur **Merge pull request**. Si l'option **Merge pull request** n'est pas affichée, cliquez sur le menu déroulant de fusion et sélectionnez **Create a merge commit**.



* Ecrasez votre commit en un commit en cliquant sur le menu déroulant de fusion, en en cliquant sur le bouton Squash et merge.



* Rebasez vos engagements individuellement sur la branche de base en cliquant sur le menu déroulant de fusion, en cliquant sur le bouton **Rebase and merge**.

