# TP Git - Github

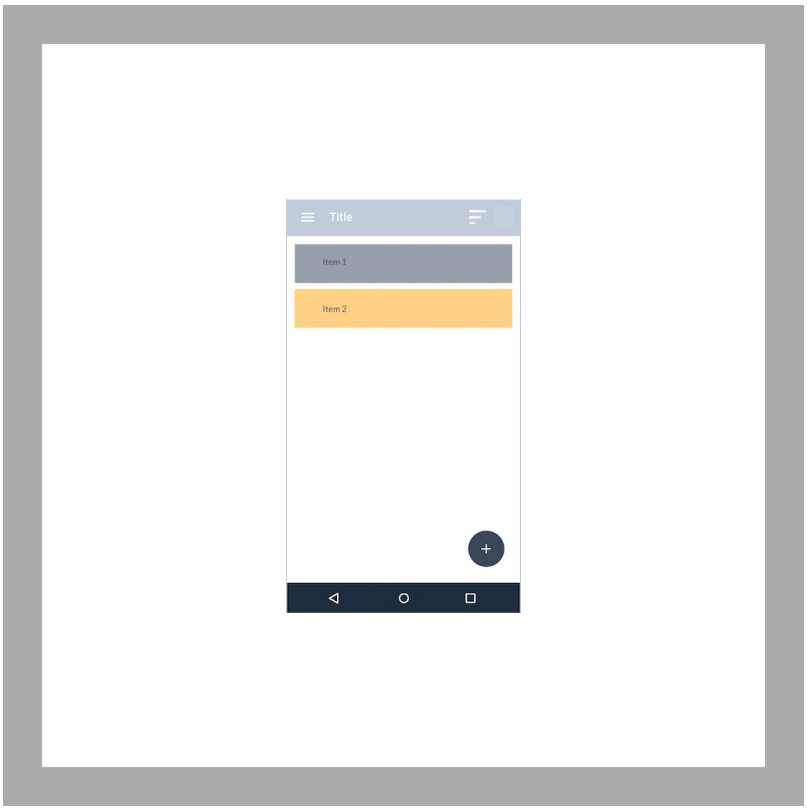
## Liens utiles

Git cheat sheet
Oh shit git !?!

## Sujet

Créer une application qui contient au moins :

- 1 RecyclerView
- 1 NavigationDrawer
- 1 FloatingButton
- 1 Module de recherche




Le sujet de l'application : Carnet de voyages Utilisez un maximum de commandes git pour vous familiariser avec l'outil.

# Pré requis

Si git n'est pas installé sur votre machine, il faut le faire :
Installation
Configuration

# Getting started (30m)

## 1. Créer votre compte Github

Tuto
N'oubliez pas d'ajouter votre clé SSH ! (cf diapo de cours)

## 2. Désigner un CHEF DE PROJET et un LEAD DEVELOPER dans chaque équipe ainsi que deux EXPERTS TECHNIQUES

- Le CHEF DE PROJET crée le dépôt du TP sur GitHub et ajoute les personnes de l'équipe + lrizand@amiltone.com
- Le LEAD DEV :
    - crée un projet avec une MainActivity dans Android Studio.
    - initialise git pour le projet créé (penser au .gitignore)
- Les EXPERTS TECHNIQUES : trouvent un nom, définissent une ergonomie consignée sous forme de schéma détaillant la navigation complète de l'application. Ce schéma sera intégré au projet dans une branche "documentation".

# Développements (1h)

## 3. Chaque personne de l'équipe a une partie bien précise à développer :

- un EXPERT TECHNIQUE se chargera du NavigationDrawer
- un autre EXPERT TECHNIQUE se chargera du module de tri. L'utilisateur doit pouvoir trier la liste par nom et par date.
- le LEAD DEVELOPER gère le bouton d'ajout d'éléments
- Le CHEF DE PROJET gère la liste des éléments

Le développement comprend :

- L'ajout de l'élément graphique sur la page d'accueil
- L'action sur cet élément et l'ouverture de la page suivante
    - Floating button > ouverture d'une Dialog de création d'objet avec plusieurs champs à remplir
    - NavigationDrawer > affichage du menu
    - Module de tri : proposer un tri par nom ou par date et mette à jour la liste en fonction de l'option sélectionée
    - Tap sur un élément de la RecyclerView : affichage du détail de l'élément sélectionné

Le code n'est pas le plus important pour la séance d'aujourd'hui. Vous devez aller vite pour vous confronter à de nombreuses commandes git (et aux problèmes qui vont avec !) Pour cette partie, chaque personne crée sa propre branche et travaille uniquement sur celle-ci.

*N.B.*
*Pour cette partie, vous ne devez utiliser que les commandes locales (sauf pour cloner le repo créé par le CHEF DE PROJET). Attention au nommage des branches et aux messages de commit, tout doit être clair et respecter les bonnes pratiques listées dans le cours.*

Cheklist des commandes à tester :

- git clone
- git diff
- git add
- git commit
- git checkout
- git branch
- git reset
- git revert

# Mise en commun (30 minutes)

Vous allez maintenant mettre votre travail en commun. Pour cela, tour à tour: Le LEAD DEV commence et
1. Reste sur sa branche, merge master dans sa branche de développement, résout les conflits s'il y en a
2. Se place ensuite dans master, puis merge sa branhce de développement sur master
Ensuite, c'est au tour du CHEF DE PROJET de faire le même travail, puis aux EXPERTS TECHNIQUES.

*N.B.
Dans cette partie vous allez mettre en pratique les notions de GitHub Flow, de résolution de conflit, de travail collaboratif. Avant tout merge, n'oubliez pas de récupérer le travail fait précédemment ...

Cheklist des commandes à tester :

- git remote
- git fetch
- git pull
- git push
- git merge > à tester absolument
- git rebase > à tester absolument
- git reset
- git revert