



# Mise en Situation

" Pratique des Acquis "

# Plan du cours

FORMOSA

## Ce document est composé des parties suivantes:

1. Les Objectifs du Module
2. La composition des équipes
3. La planning des intervenants
4. Quelques outils utiles
5. Propositions de projets





# Mise en Situation

## Les Objectifs du Module

# Les Objectifs de ce module

Les objectifs de ce module suivent trois axes :

- l'axe organisationnel
- l'axe des compétences
- l'axe de la communication



# Les Objectifs de ce module : Organisationnel

## Organisationnel

- Travailler en équipe
- Organiser votre travail
- Analyser le besoin
- Chiffrer votre projet en Temps (coût)
- Fixer un planning

# Les Objectifs de ce module : Compétences

## Compétences

- Mettre en pratique vos compétences techniques
- Utiliser et lier les compétences acquises dans différents modules
- Apprendre de nouvelles compétences

# Les Objectifs de ce module : Communication

## Communication

- Présenter votre travail
- Présenter votre démarche
- Apprendre à communiquer à un public technique et/ou non technique





# Mise en Situation

## La composition des équipes



# Les Équipes



## Les Équipes

- Elles sont composés par l'équipe pédagogique, suivant les critères suivants :
  - Niveau des étudiants (Groupes équivalent en niveau)
  - Niveau en HTML et rendus
  - Respect des proportions H/F
  - Feeling
- Elle ne sont pas discutables (sauf force majeur)

# Les Équipes



## Équipe #1

BESBAS  
PERDRIX  
PERCIOT  
GONCALVES  
MACHET

## Équipe #2

CAMUS  
PRUVOT  
TARDY-LASSERVE  
CAMPION  
GUENOT

## Équipe #3

COLLION  
ROBERT  
GAMRAT  
VERNET  
RABOIS

## Équipe #4

FREIBURGER  
THEODORE  
OELLERS  
BONANT  
PAVIC  
BOYER

## Équipe #5

HOUNI  
BUISSON  
HOUEVILLE  
TOURRE  
HALLER





# Mise en Situation

## Le planning

# Le planning

	16-10	31-10	5-11	12-11	13-11	14-11	20-11
Matin	GA	E-learning	GA	E-learning	GA	E-learning	Soutenance
Après midi	GA	E-learning	GA	E-learning	GA	E-learning	
	Démarrage et Analyse du Besoin Maquette	Code	Code	Code	Debrief sur les présentations Code	Peaufinage Préparation des présentations	





# Mise en Situation

## Quelques outils utiles

# Quelques outils utiles

## Outils de Gestions de Projets

- Trello
- JIRA (payant)
- Teams
- GitHub





# Mise en Situation

## Proposition de projets

# Les Projets

Nous proposons un ensemble de Projets :

- Il sont facultatifs : vous pouvez proposer vos propres projets qui seront validés
- Vous pouvez partir sur un des projets proposés
- Un projet peut être associé à plusieurs équipes vous êtes alors

**en compétition c'est-à-dire pas de copie de code**



# Les Projets



Les compétences à mettre en œuvre pour un projet :

- (1) Une architecture Client / Serveur
- (2) Coté client : Technologie JS, JQuery
- (3) Coté serveur : Php ou autre à proposer
- (4) Coté Base de données : SQL ou No-SQL



# Projet 1 : Gestion des dépenses



## Sujet 1 : Réaliser un site de gestion de dépenses

**Description :** Dans une famille chaque membre doit pouvoir enregistrer les dépenses qu'il fait sur son compte en banque.

Un membre possède seulement un compte en banque.

L'application propose l'enregistrement des dépenses sur un site web disponible au format "Bureau" / "Mobile". La famille peut créer des budgets/ et des sous-budgets.

Chaque membre de la famille est associé à un ou plusieurs budgets ou sous-budget)

Nous souhaitons pouvoir voir en temps réel l'état des Budget de la famille et avoir un petit bilan à la fin du mois.

Gérer l'inscription et la connexion d'un utilisateur.

**Technologie :** HTML / CSS / JS / PHP / Diagrammes / MySQL

# Projet 2 : Gestion d'opinion publiques



## Sujet 2 : réaliser un site de gestion d'opinions publiques

**Description :** Dans une démarche citoyenne nous souhaitons recenser les opinions publiques sur la société.

Un membre peut participer à ce projet s'il s'enregistre avec les informations de nom, prénom et copie d'une pièce d'identité puis après validation d'un modérateur.

Un membre peut poster des sujets de discussion et aussi donner des avis.

Un modérateur peut supprimer et/ou bannir des sujets

Imaginer des types de sujet simple comme : pour ou contre, des sujets à choix etc.

Gérer l'inscription et la connexion d'un utilisateur.

**Technologie : HTML / CSS / JS / PHP / MySQL**

# Projet 3 : Gestion des Repas



## Sujet 3 : Gestion de repas

**Description :** Nous sommes dans une démarche citoyenne, ici nous voulons gaspiller moins de nourritures.

Ce site Web propose aux utilisateurs de saisir les ingrédients qu'il possède, le site va proposer un ensemble de recettes contenant le plus d'ingrédients possible.

La site à un volet administrateur permettant de mettre en ligne des recettes et des ingrédients

Gérer l'inscription et la connexion d'un utilisateur.

**Technologie : HTML / CSS / JS / PHP / MySQL**



# Projet D : Génération de Diagrammes



## Sujet 4 : génération de memory card.

**Description** : Nous souhaitons proposer a des utilisateurs préalablement enregistrés de générer des jeux de memory card.

Un utilisateur va pouvoir demander à générer un jeu à partir d'un ensemble d'images uploadées et d'un ensemble de cartes souhaitées.

Une url lui sera donnée pour partager son jeu à d'autres.

Il peut posséder plusieurs jeux.

Un modérateur peut se réserver le droit de supprimer des jeux.

Gérer l'inscription et la connexion d'un utilisateur.

**Technologie : HTML / CSS / JS / PHP / MySQL**

# Projet E : Carences Alimentaires



## Sujet 5 - étirement au travail

**Description :** Faire un site web proposant aux utilisateurs enregistrés de réaliser des étirements chronométrés pour la nuque et les épaules.

Le logiciel permet de configurer des journées de travail le nombre de pauses souhaitées.

Une pause est configurable à travers différents exercices.

Gérer l'inscription et la connexion d'un utilisateur.

**Technologie : HTML / CSS / JS / PHP / MySQL**



# Projet F : Gestion des Notes

## Sujet 6 - Boire de l'eau

**Description** : Parce qu'on ne boit pas assez d'eau. Cette application propose d'enregistrer les boissons que l'on boit dans la journée.

Les boissons possèdent des coefficients d'hydratation différentes. Ainsi 100ml d'eau n'hydrate pas pareil que 100ml de soda ou de café.

Nous souhaitons pouvoir calculer les objectifs avec les poids et tailles des utilisateurs et afficher des bilans quotidiens, à la semaine ou au mois.

Gérer l'inscription et la connexion d'un utilisateur.

**Technologie** : HTML / CSS / JS / PHP / MySQL



# Mise en Situation

## Analyse du Besoin et Spécification



## L'Analyse du Besoin et Spécification :

- Énoncer le Besoin
- FRONT : Réaliser des Maquettes
- Modéliser le Besoin (par exemple use case)
- Établir une liste de Web Service (Interface Front et Back) (optionnel)
- BACK : Modéliser les Données (Représentation Objet et Base de données)
- Établir un ensemble de Tâches (BackLog)
- Chiffrer les Taches en terme de durée (Chiffrage)
- Assigner les Tâches aux personnes
- Établir des dates de rendu régulier (Sprint)

## Enoncer le Besoin

- L'application doit pouvoir gérer un ensemble de TODO Listes
- On doit pouvoir enregistrer des nouveaux utilisateurs
- Chaque élément de la liste est marqué fait ou a faire
- On peut supprimer des élément de chaque liste
- Ajouter ou supprimer des listes

## Réaliser des Maquettes



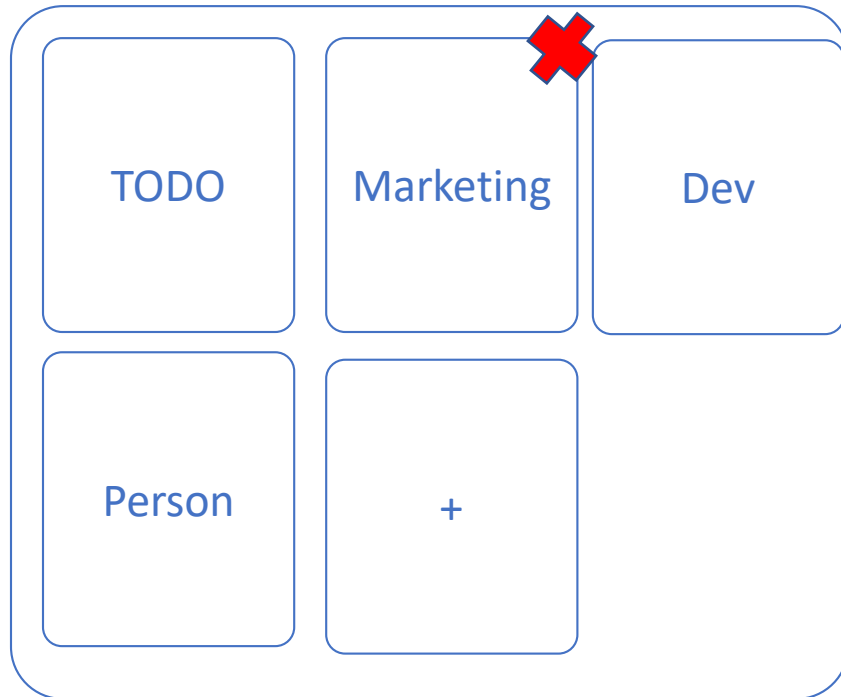
A wireframe of a login form. It consists of a rounded rectangle containing three input fields. The first two are stacked vertically, labeled 'Login' and 'Mdp'. The third is a solid blue button labeled 'Se connecter'.

### Interaction :

- Appuie sur le bouton se connecter
- Si ça marche passage à l'écran 2
- Sinon message d'erreur



## Réaliser des Maquettes



### Interaction :

- Renommer une liste (Double click sur le nom)
- Supprimer une liste (Click sur la croix)
- Ajouter une liste (click sur le +)

## Réaliser des Maquettes

Marketing

☐ Spammer les clients

☐ Faire du Porte à Porte

☒ Pub Métro

### Interaction :

- Ajout d'item
- Suppression d'Item
- Cocher les Item fait
- Enregistrer les modifications
- Revenir aux listes

## Web Services

WS1 :

### **POST : AJOUT DE LISTE**

Paramètre : Nom

Paramètre : Login / Mdp

Retourne : Le modèle / La liste vide

Code : 201 (CREATED)

Code : 400

WS2 :

### **DELETE : SUPPRIMER LIST**

Paramètre : Nom

Paramètre : Login / Mdp

Retourne : Code : 200 (OK)    Code : 400

WS3 :

### **GET : TOUTE LISTE**

Paramètre : -

Paramètre : Login / Mdp

Retourne : Ensemble de Liste



## Modélisation





## BackLog

- |  |                |
|--|----------------|
| - Création du Modèle en Java POJO                | (BACKOFFICE)   |
| - Créer le Service POST d'AJOUT de Liste         | (BACKOFFICE)   |
| - Créer le Service DELETE de Suppression         | (BACKOFFICE)   |
| - Créer le Service De retour de toutes les liste | (BACKOFFICE)   |
| - Créer l'écran 1                                | (FRONT OFFICE) |
| - Créer l'écran 2                                | (FRONT OFFICE) |
| - Créer l'écran 3                                | (FRONT OFFICE) |



## Chiffrage

- Création du Modèle en Java POJO	(BACKOFFICE)	30 min
- Créer le Service POST d'AJOUT de Liste	(BACKOFFICE)	1h
- Créer le Service DELETE de Suppression	(BACKOFFICE)	1h
- Créer le Service De retour de toutes les liste	(BACKOFFICE)	1h
- Créer l'écran 1	(FRONT OFFICE)	4h
- Créer l'écran 2	(FRONT OFFICE)	4h
- Créer l'écran 3	(FRONT OFFICE)	4h





# Répartition des Taches

## Lot 1 – Sprint1    du 18/04 au 30/04

- Création du Modèle en Java POJO	(BACKOFFICE)	30 min	ALEX
- Créer le Service POST d'AJOUT de Liste	(BACKOFFICE)	1h	TIM
- Créer le Service DELETE de Suppression	(BACKOFFICE)	1h	TIM
- Créer le Service De retour de toutes les liste	(BACKOFFICE)	1h	TIM

## Lot 2 – Sprint2    du 30/04 au 30/05

- Créer l'écran 1	(FRONT OFFICE)	4h	ALEX
- Créer l'écran 2	(FRONT OFFICE)	4h	LEA
- Créer l'écran 3	(FRONT OFFICE)	4h	SANDRA



# Introduction à UML

Programmation  
Orientée Objet



## UML

- Modélisation orientée objets
- Couvre toutes les phases d'un projet (analyse des besoins --> déploiement)
- Une synthèse (partielle) de 25 ans de recherche en modélisation

## Modélisation selon trois axes:

- Fonctionnel
- Dynamique
- Statique



# Les Diagrammes Principaux

Les Diagrammes Principaux sont :

1. Diagramme des Cas d'Utilisation
2. Diagramme des Séquences
3. Diagramme des Classes
4. Diagramme d'Activités
5. Diagramme d'Etats

Les diagrammes secondaires sont :

6. Diagramme de communication
7. Diagramme d'Objets
8. Diagramme de Structures Composite
9. Diagramme de Déploiement

# Motivations (génie logiciel)

Les motivations de ce diagramme :

- « Solve the right problem » Résoudre le bon problème
- Analyse des besoins (« requirements »)
  - Déterminer les besoins - ce que le système doit faire
  - Comprendre les besoins
  - Délimiter le système
- centrée sur l'utilisateur
  - Quels sont les besoins du point de vue des utilisateurs
  - Intégrer les points de vue

# Cas d'utilisation (principes)

Le Diagrammes des cas d'utilisation décrit :

- Ce que le système doit faire (comportement souhaité)

Il ne détaille pas sur la manière de réaliser cela :

- Pas de détail de programmation ou de mise en œuvre
- On se détache de la partie pratique

Ce Diagramme est donc un bon outils pour communiquer entre l'utilisateur final et l'équipe de développement



# Cas d'utilisation : interactions utilisateur- système

Ce diagramme décrit en premier l'ensemble des acteurs d'un systèmes

Ainsi que l'ensemble des interaction de l'acteur sur le système, nommé ici "cas d'utilisation"

Les cas d'utilisation se décrivent par dépendance avec d'autres cas d'utilisations

# Éléments du Diagramme : L'acteur



Ce qui existe en dehors du système

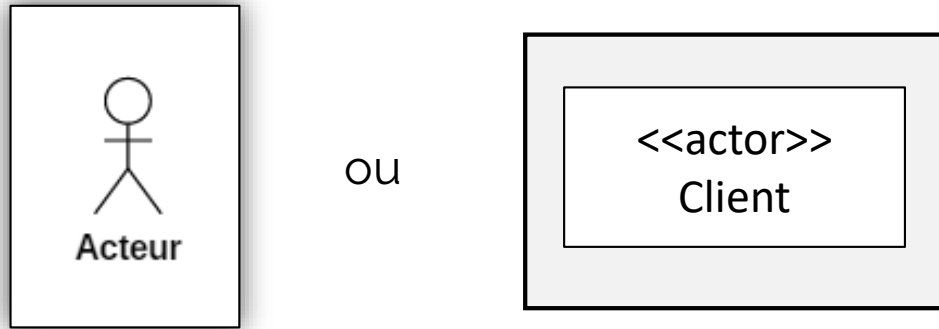
Tout ce qui doit échanger de l'information avec le système personne, machine, organisation, autre ordinateur, autre système

Correspond à un rôle générique que l'utilisateur joue = une manière d'utiliser le système

La même personne (machine, ...) peut jouer plusieurs rôles

# Éléments du Diagramme : L'acteur

**L'acteur, il se représente comme cela :**



Les acteurs sont :

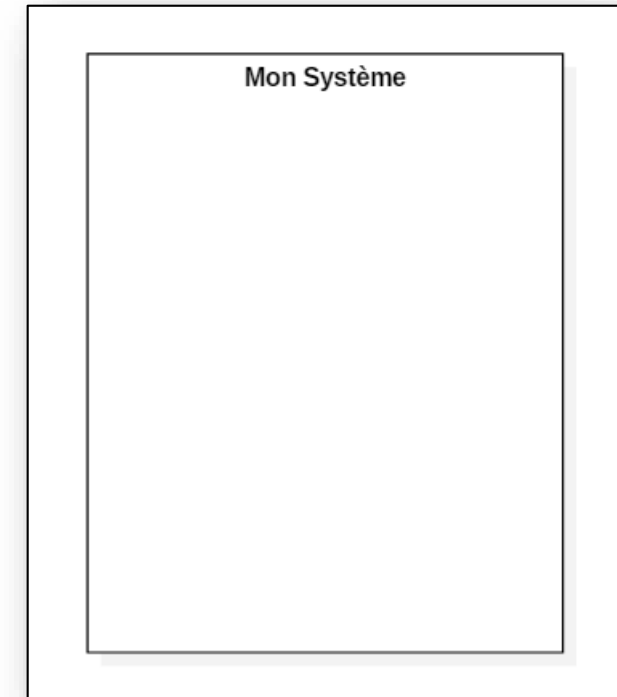
- Les utilisateurs humains directs : administrateur, client, opérateur etc
- Les autres systèmes connexes



# Éléments du Diagramme : Le Système

**Le Système est le système en cours de conception**

Le système est une boîte cette boîte contient des cas d'utilisations

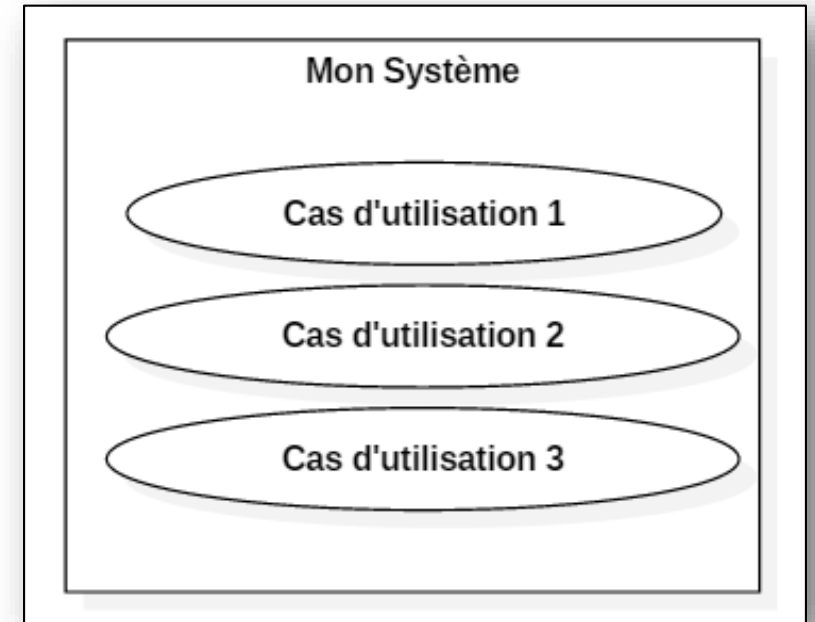


# Éléments du Diagramme : Les cas d'utilisations

## Les Cas d'utilisations

Un cas d'utilisation est un ensemble de séquences d'actions qui sont réalisées par la système.

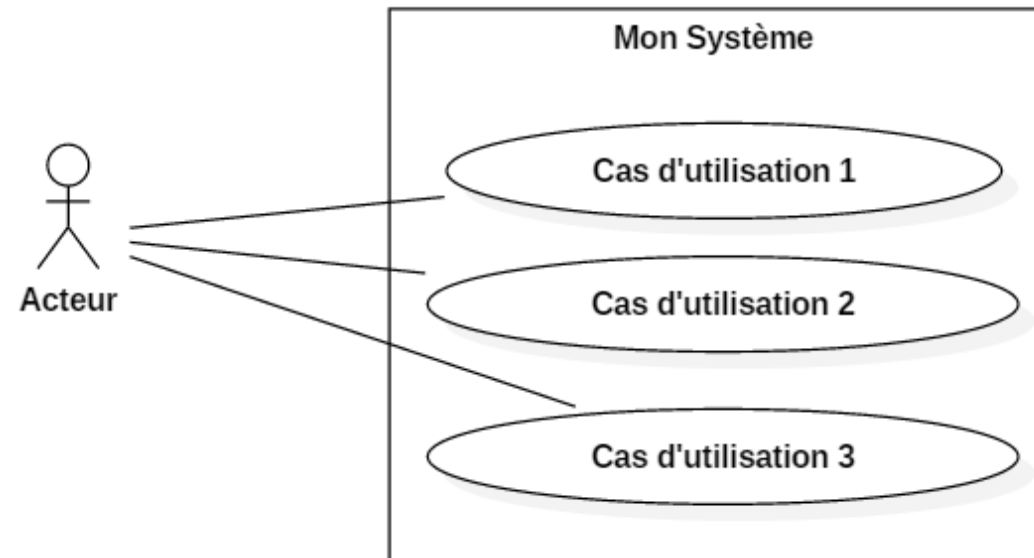
Le cas d'utilisation décrit le comportement du système.



# Éléments du Diagramme : Les cas d'utilisations

## Les Cas d'utilisations

Les cas d'utilisation sont reliées aux acteurs.





# Scénario : GAB (Guichet Automatique de Billets)

## Les Services rendus :

1. Un GAB offre la distribution d'Argent à tout porteur de CB via un lecteur de carte et un distributeur de billets
2. Consultation des soldes de comptes, dépôt d'espèces, de chèque pour les clients porteurs d'une CB de la même banque que le GAB
3. Toutes les transactions sont sécurisés
4. Il faut de temps en temps recharger le distributeur de billets

# Étapes de l'analyse Fonctionnelles

**La démarche va consister en différentes étapes :**

## **Etape 1**

- Identifier les Acteurs
- Réaliser le Diagramme de Contexte Statiques

## **Etape 2**

- Identifier les Cas d'utilisations
- Lier aux acteurs

## **Etape 3**

- Liens entre acteurs

## **Etape 4**

- Liens entre cas d'utilisations

# Etape 1 : Identifier les Acteurs

## L'étape 1 va consister à identifier les acteurs

Considérer l'énoncé phrase par phrase.

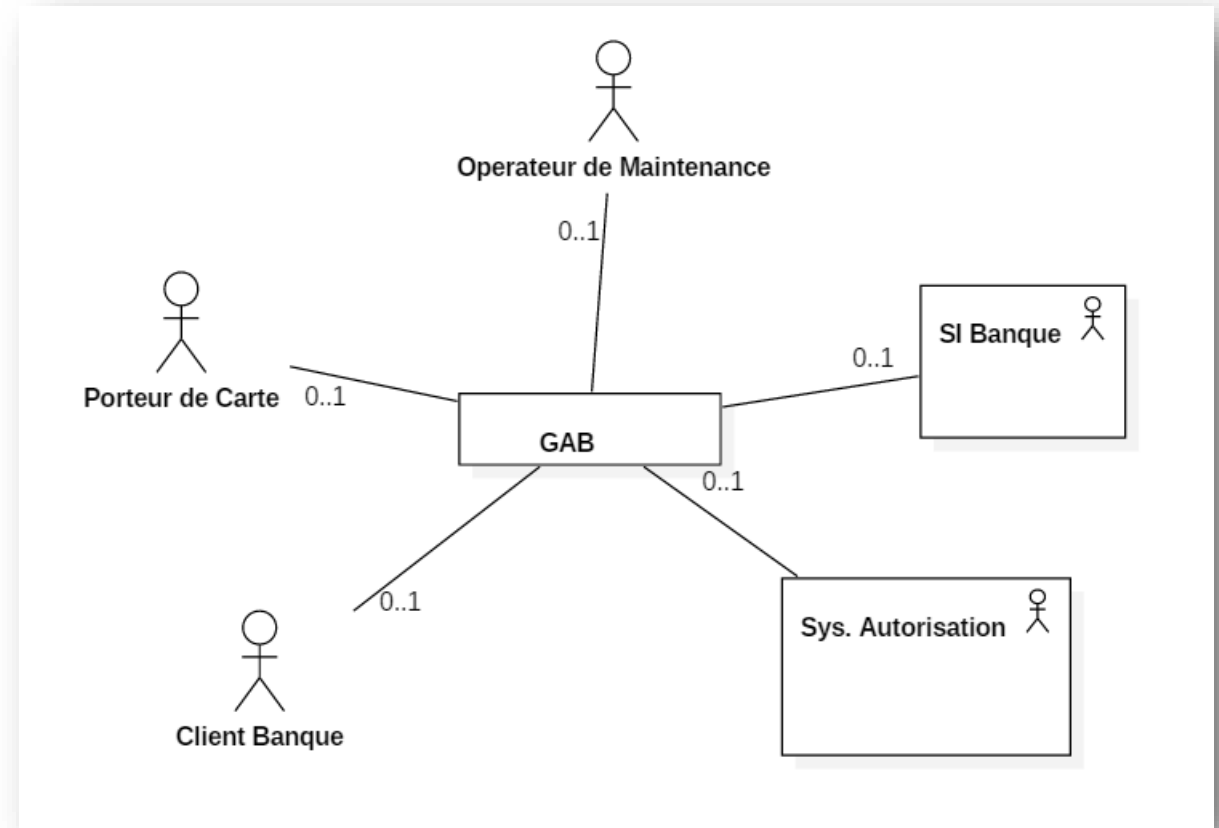
- Porteur de Carte
- Un Client de la banque
- Un Opérateur de maintenance
- Le SI de la banque
- Le Syst. d'autorisation



# Etape 1 : Identifier les Acteurs

## Diagramme de Contexte Statique

Les GAB est un système n'ayant qu'un seul utilisateur en même temps



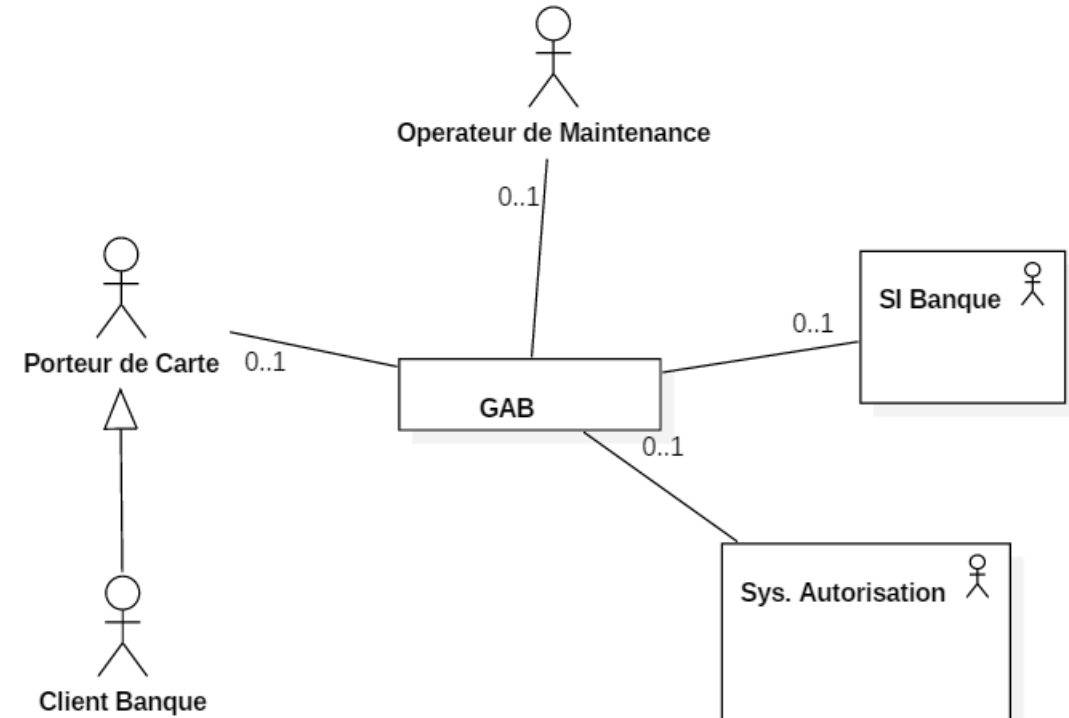
# Etape 1 : Identifier les Acteurs

## Diagramme de Contexte Statique

Nous pouvons représenter la relation entre la porteur de carte et le client de la banque par une relation d'héritage.

On marque cela par une cardinalité 0..1 Signifiant qu'il y a 0 à 1 utilisation du GAB

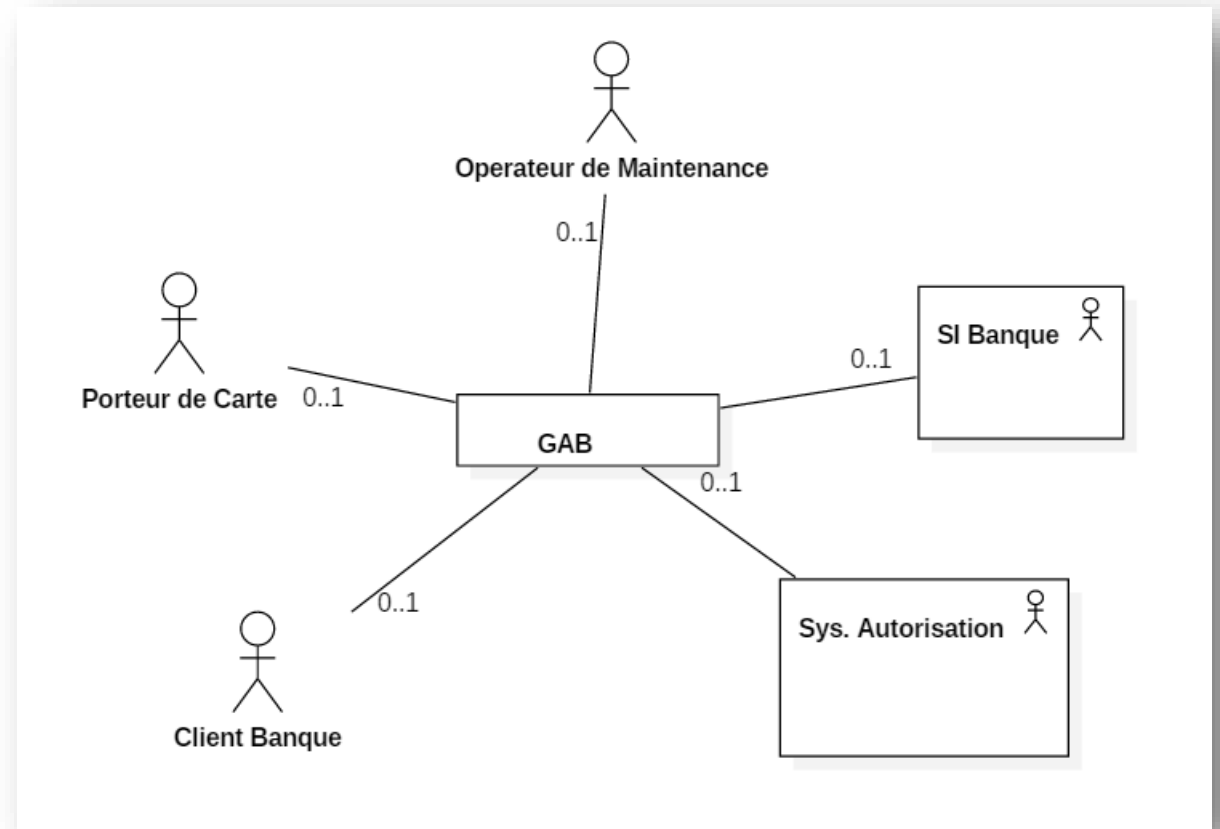
Un Client de Banque est un porteur de carte.



# Etape 2 : Identifier les cas d'utilisation

## Diagramme de Contexte Statique

Les GAB est un système n'ayant qu'un seul utilisateur en même temps





## Etape 2 : Cas d'utilisation rappel

Description d'un ensemble de séquences d'actions, incluant des variantes, qu'un système effectue pour fournir un résultat observable et ayant une valeur pour un acteur.

A partir des acteurs précédemment établit, nous pouvons associer des cas d'utilisations

# Etape 2 : établir les cas d'utilisations

## **Porteur de la carte :**

- Retirer de l'argent

## **Client de la banque :**

- Retirer de l'argent
- Consulter son compte en banque
- Déposer des Espèces
- Déposer des Chèques

## **Opérateur de Maintenance :**

- Recharger le distributeur
- Remplacer les combustibles

**SI Banque** : Rien

**SI Autorisation** : Rien

# Etape 2 : établir les cas d'utilisations

## Porteur de la carte :

- Retirer de l'argent

## Client de la banque :

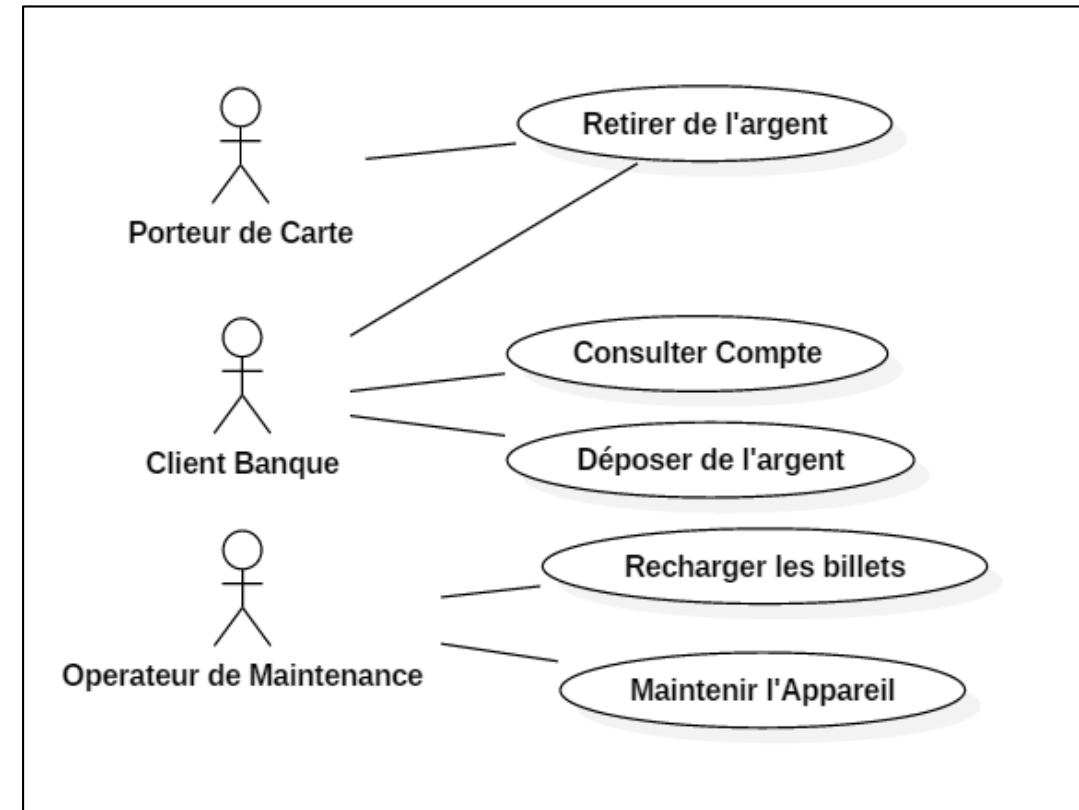
- Retirer de l'argent
- Consulter son compte en banque
- Déposer des Espèces
- Déposer des Chèques

## Opérateur de Maintenance :

- Recharger le distributeur
- Remplacer les combustibles

SI Banque : Rien

SI Autorisation : Rien

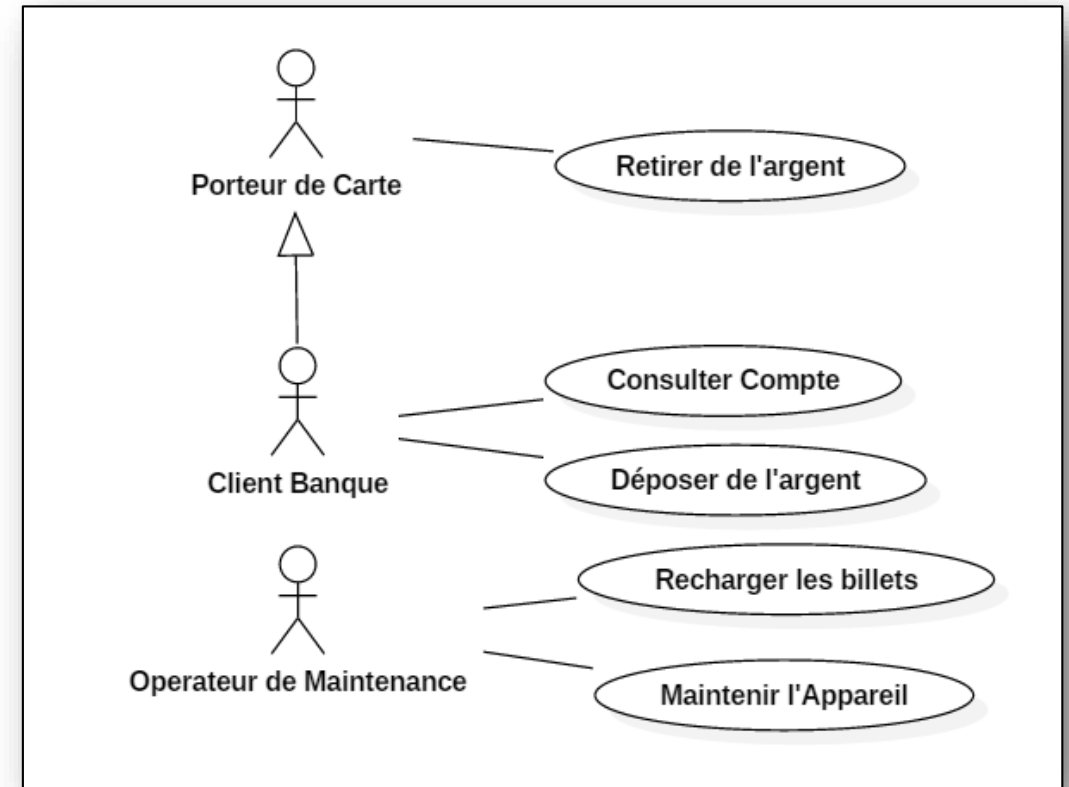


# Etape 3 : Lien entre acteurs

L'héritage entre les acteurs il va permettre de dire qu'un acteur est une spécialisation d'un autre acteur de ce fait il sera associé intrinsèquement aux cas d'utilisations des autres acteurs

On peut traduire ici la relation par la phrase.

"Un Client est un porteur de carte  
il peut donc retirer de l'argent"



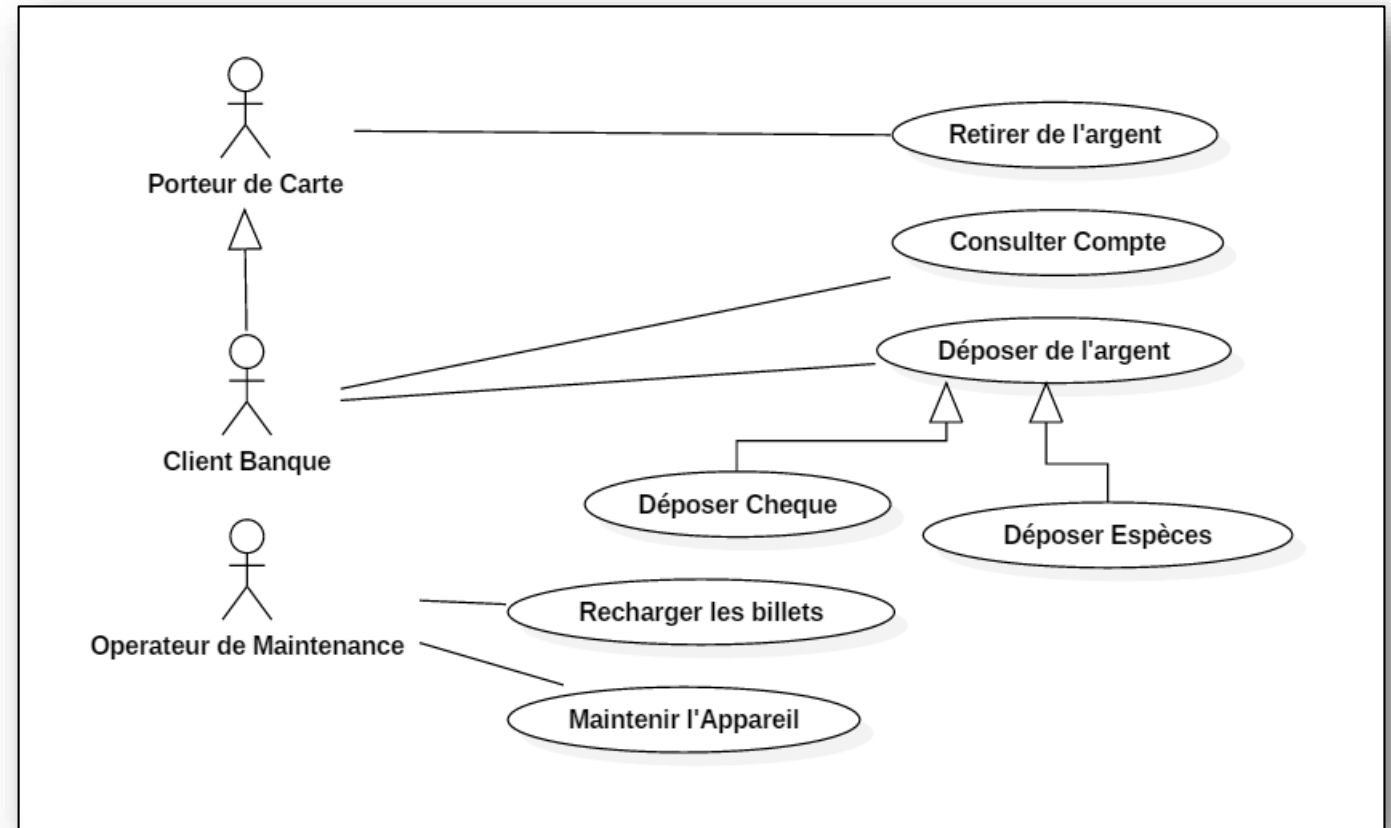


# Etape 4 : Liens entre cas d'utilisations

De même la relation d'héritage peut s'appliquer sur les cas d'utiliser.

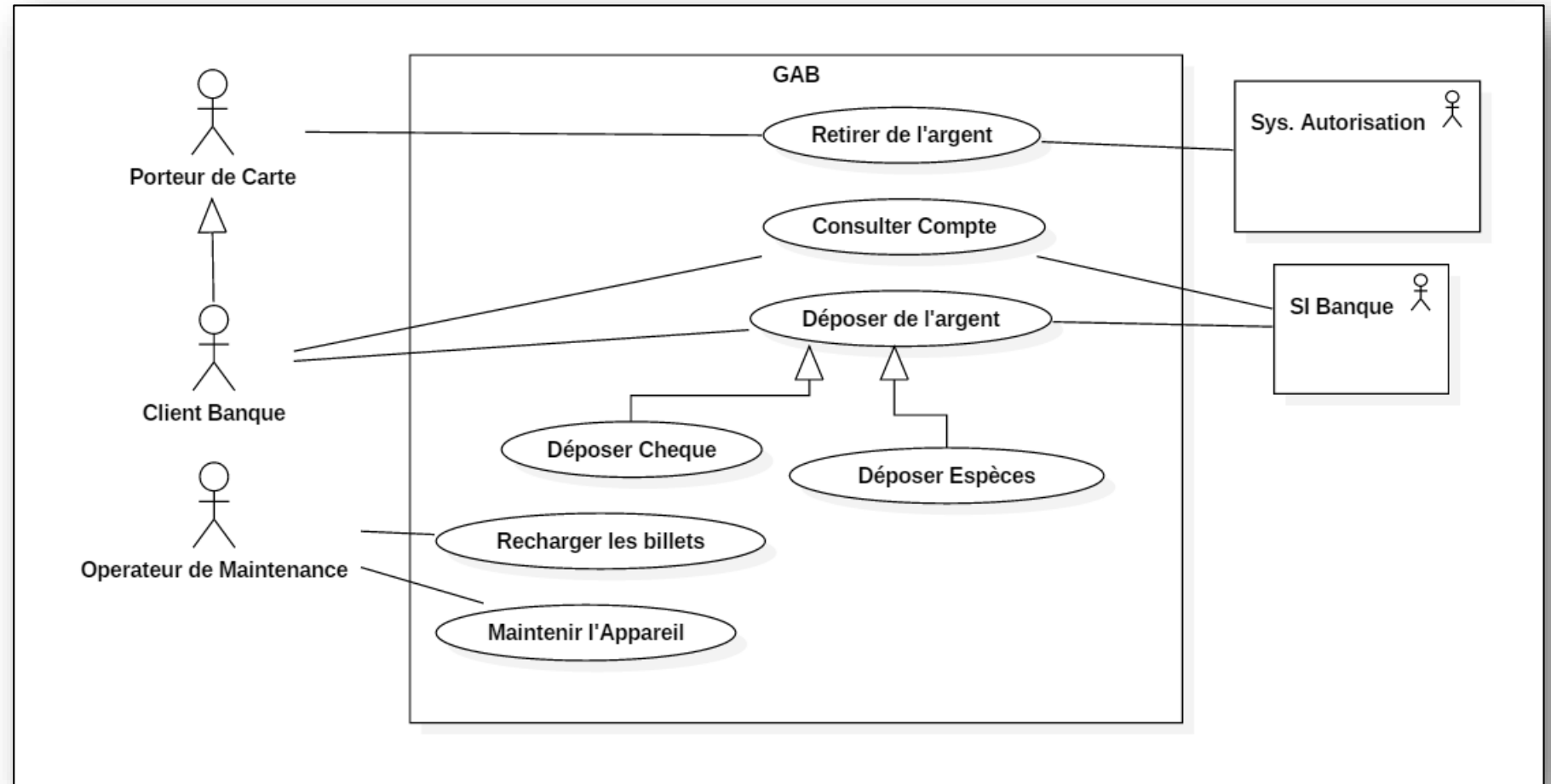
Ainsi déposer l'argent possède deux cas d'utilisation spécialisant ce cas d'utilisation :

- Déposer des espèces
- Déposer du liquide



# Etape 4 : Liens entre cas d'utilisations

Les cas d'utilisation peuvent eux-mêmes faire appel à des systèmes externes précédemment défini

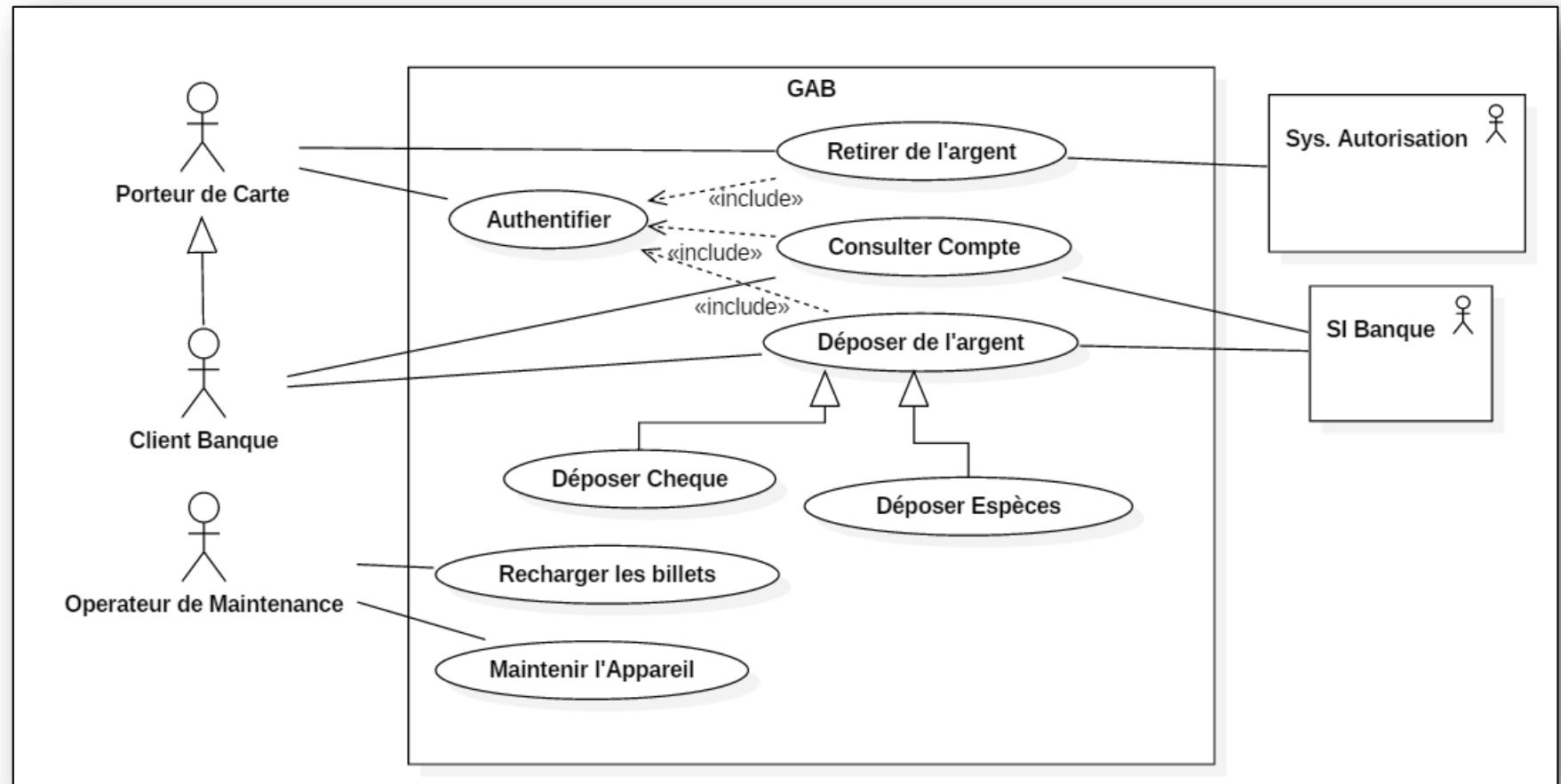


# Etape 4 : Liens entre cas d'utilisations

L'inclusion indique une dépendance entre les cas d'utilisation par exemple :

- Retirer de l'argent
- Consulter le compte
- Déposer de l'argent

Inclus une étape d'authentification



# Les diagrammes de séquences

Les diagrammes de séquences mettent en valeur les échanges de messages (déclenchant des événements) entre acteurs et objets (ou entre objets et objets) de manière chronologique, l'évolution du temps se lisant de haut en bas

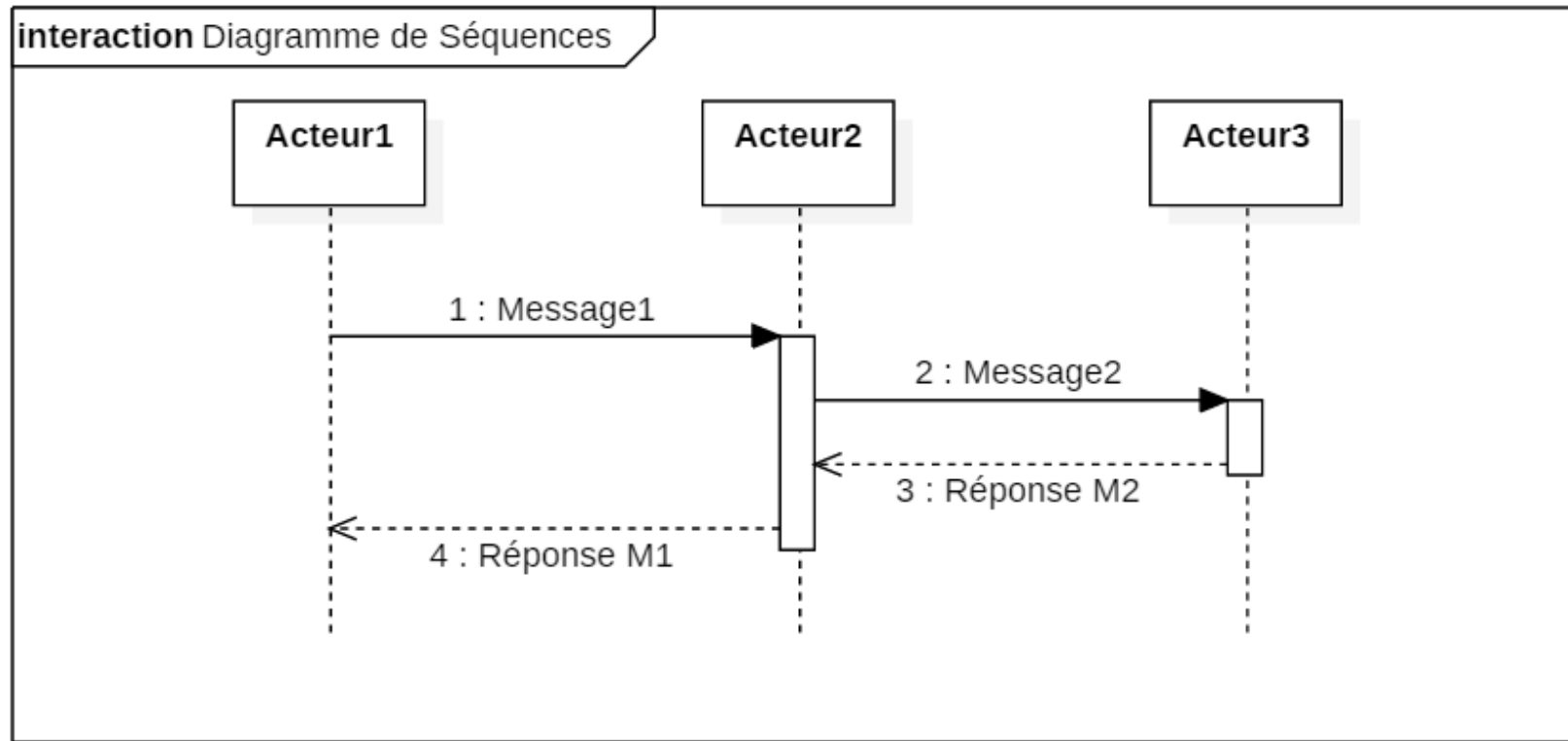
Base: les cas d'utilisation

Mention des objets créés ou détruits lors des exécutions

Spécification des contraintes de temps : durée



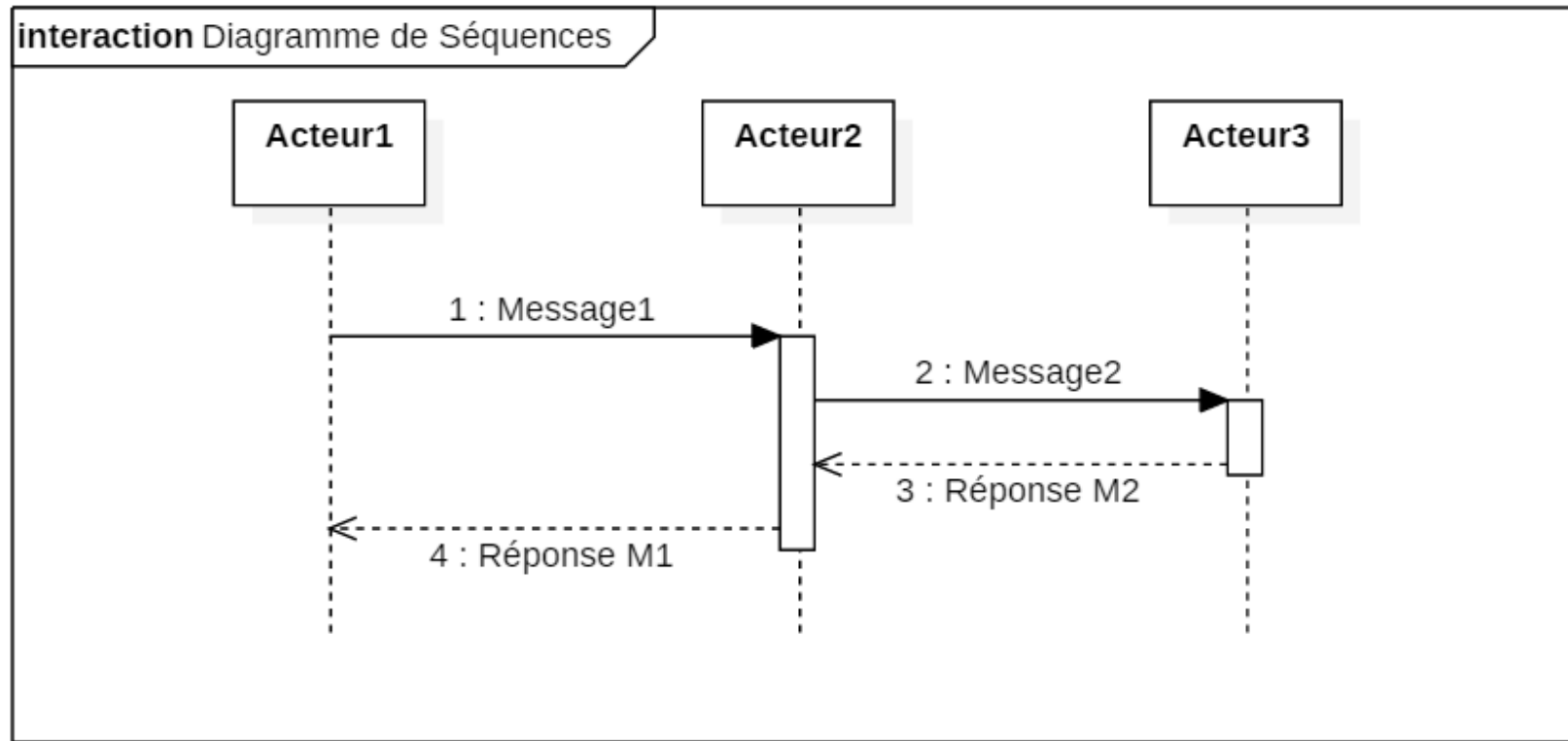
# Exemple de Diagramme de Séquences



# Acteurs d'une interaction

Le Diagramme des séquences peut s'inscrire comme une continuité du diagramme des Cas d'utilisation. Il peut aussi avoir son utilisation propre.

Nous pouvons par exemple créer un diagramme par cas d'utilisation où chaque acteur du diagramme des séquences correspond à un acteur du diagramme des cas d'utilisations.



Le Système est aussi représenté comme un acteur du diagramme des séquences

# Acteurs d'une interaction

Une autre utilisation du diagramme des séquences consiste à utiliser le concept objet, chaque acteurs est une instance de classe, les message sont les appels de méthodes

Ce Diagramme est apprécié pour les appels/les message dans le cadre de système asynchrone.

# Différents Types de Messages

Différents types de messages existent :

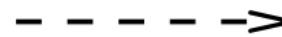
- Les messages synchrones
- Les messages asynchrones
- Les messages de retour



Message synchrone

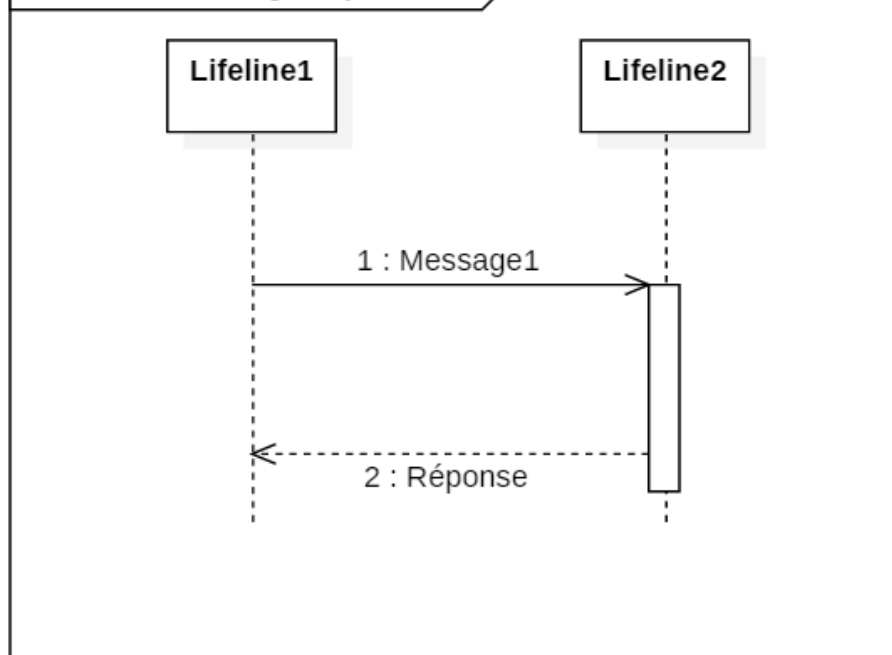


Message asynchrone

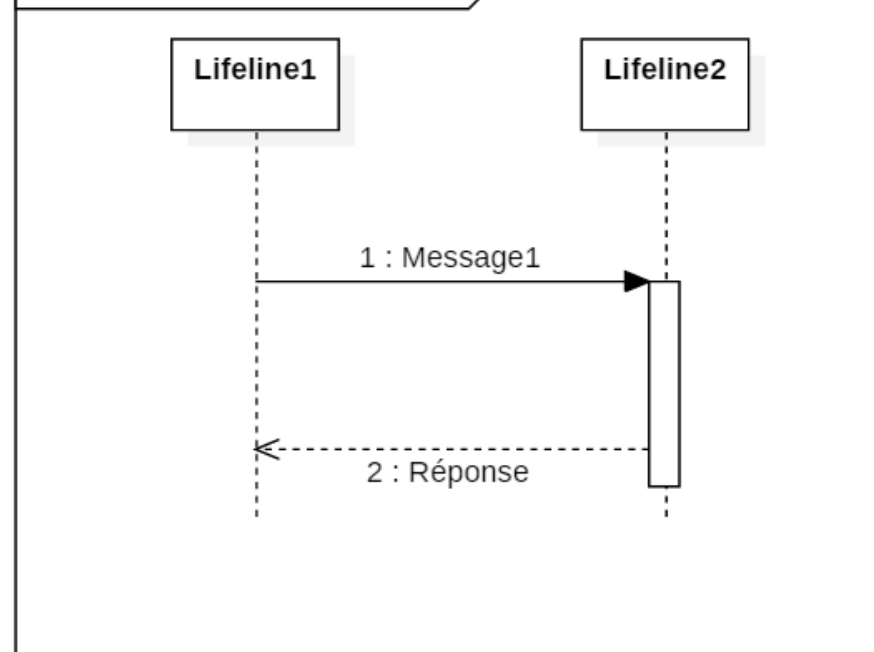


Retour

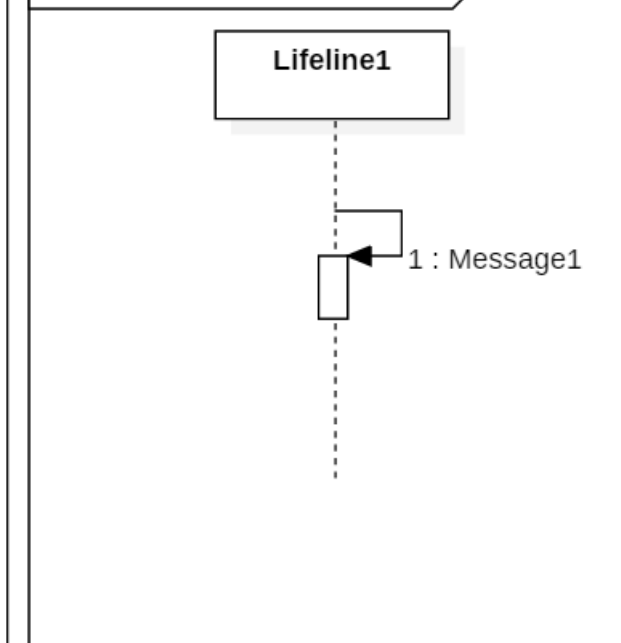
interaction Message Asynchrone



interaction Message Synchrone



interaction Messages Reflexifs

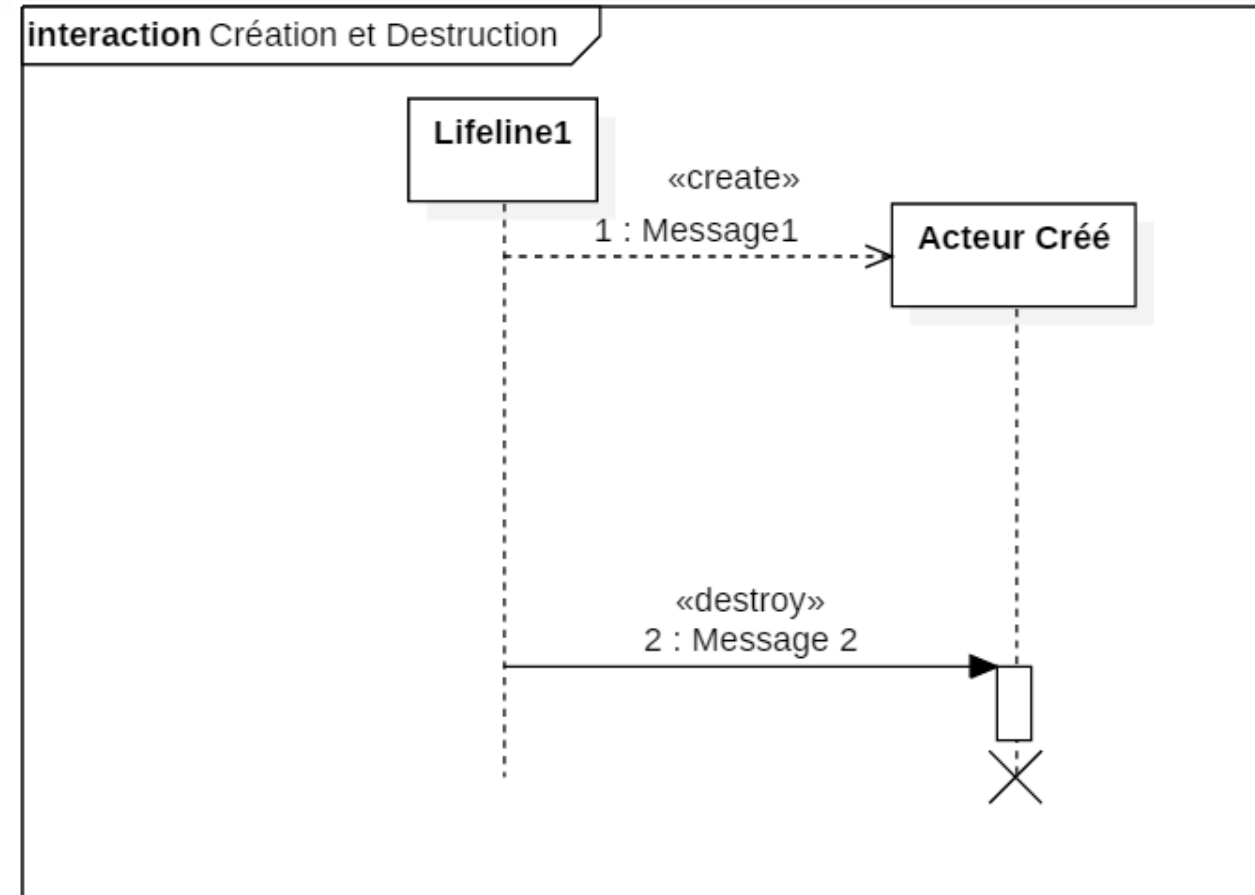




# Différents Types de Messages

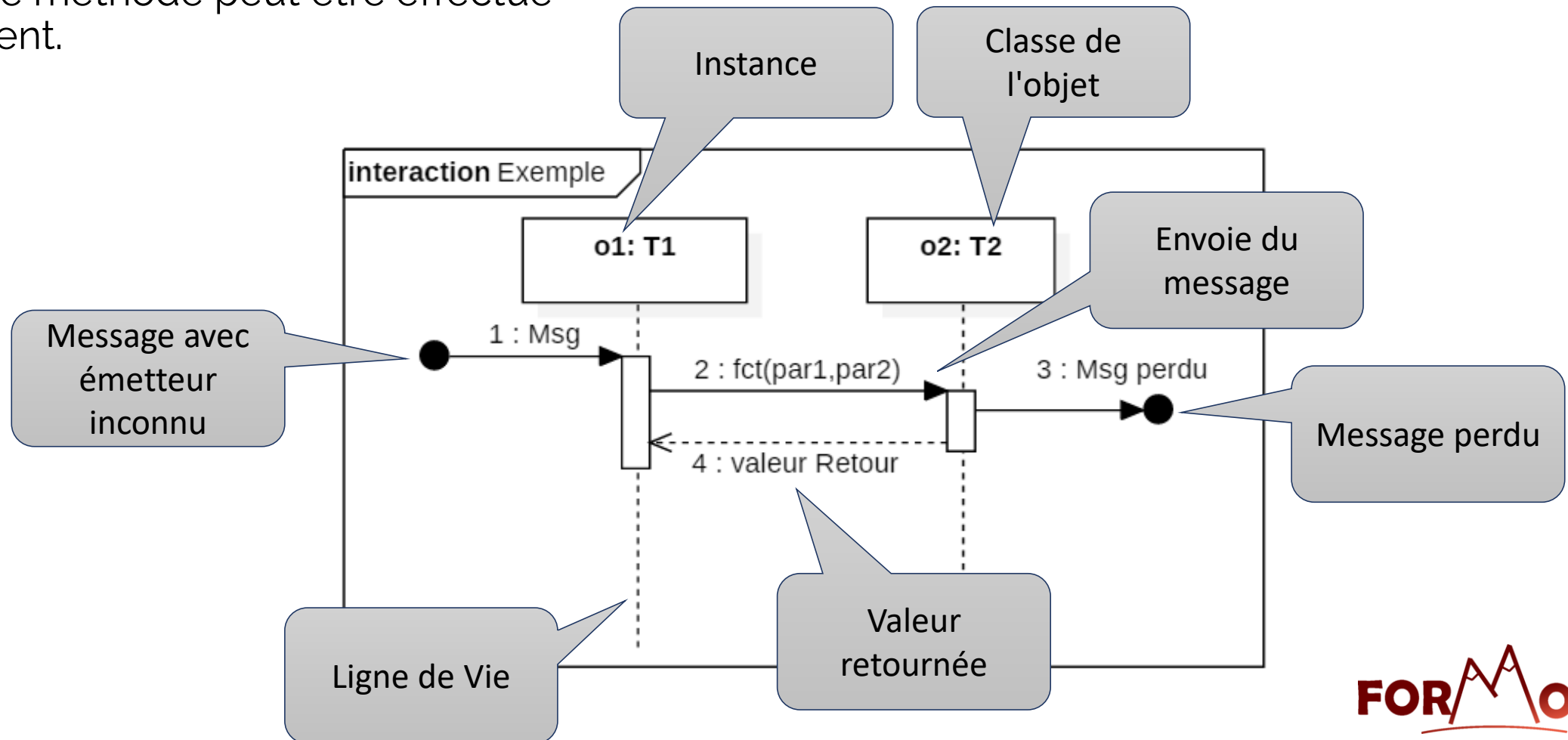
Les messages de création permettent la création d'une Lifeline ou d'un acteur.

Les messages de Destruction permettent la destruction d'une lifeline ou d'un acteur.

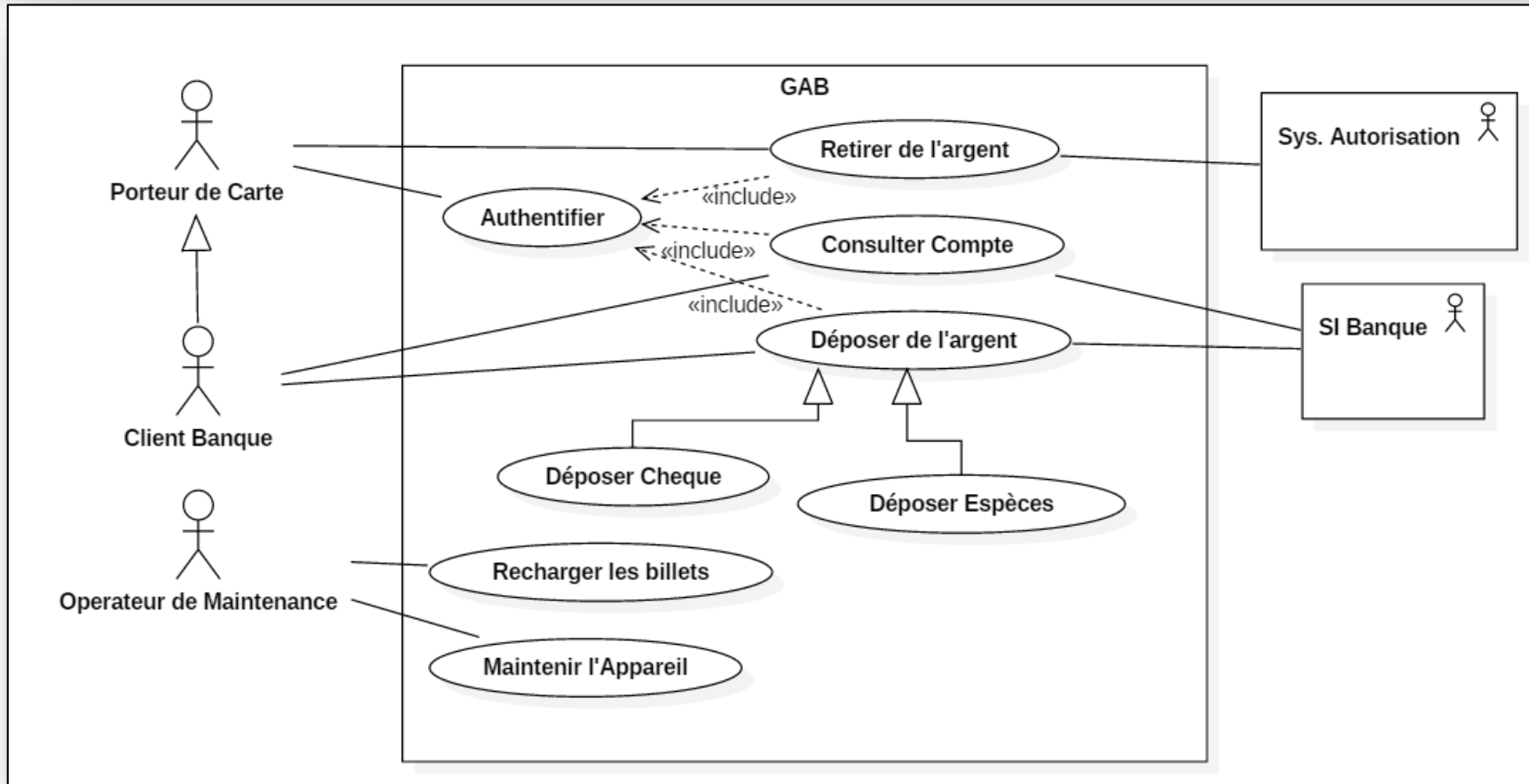


# Diagramme de Séquence et Appel de méthodes

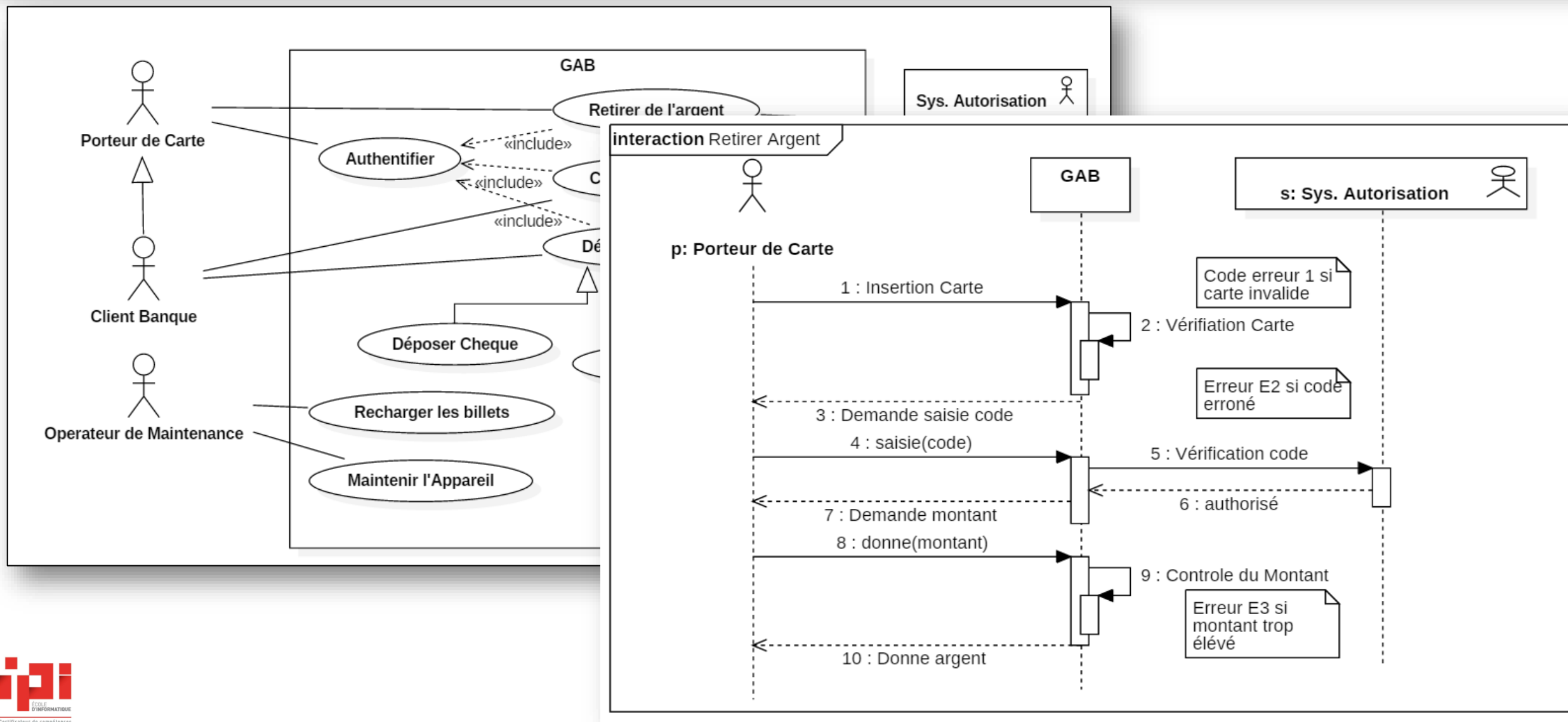
Le lien entre le diagramme de séquence et l'appel de méthode peut être effectué rapidement.



# Lien avec les Cas d'utilisation



# Lien avec les Cas d'utilisation







GIT

" Cycle Développeur "

# Dans ce document

## Le plan de cette formation :

1. Présentation de GIT
2. Démarrage Rapide
3. Les Repository distants
  1. GitHub
4. Les principales commandes

# Introduction

GIT est un gestionnaire de version.

Il permet de :

- Sauvegarder le travail en local et sur un serveur distant
- Revenir à une version précédente de son travail
- Travailler en collaboratif sur un projet

En outre :

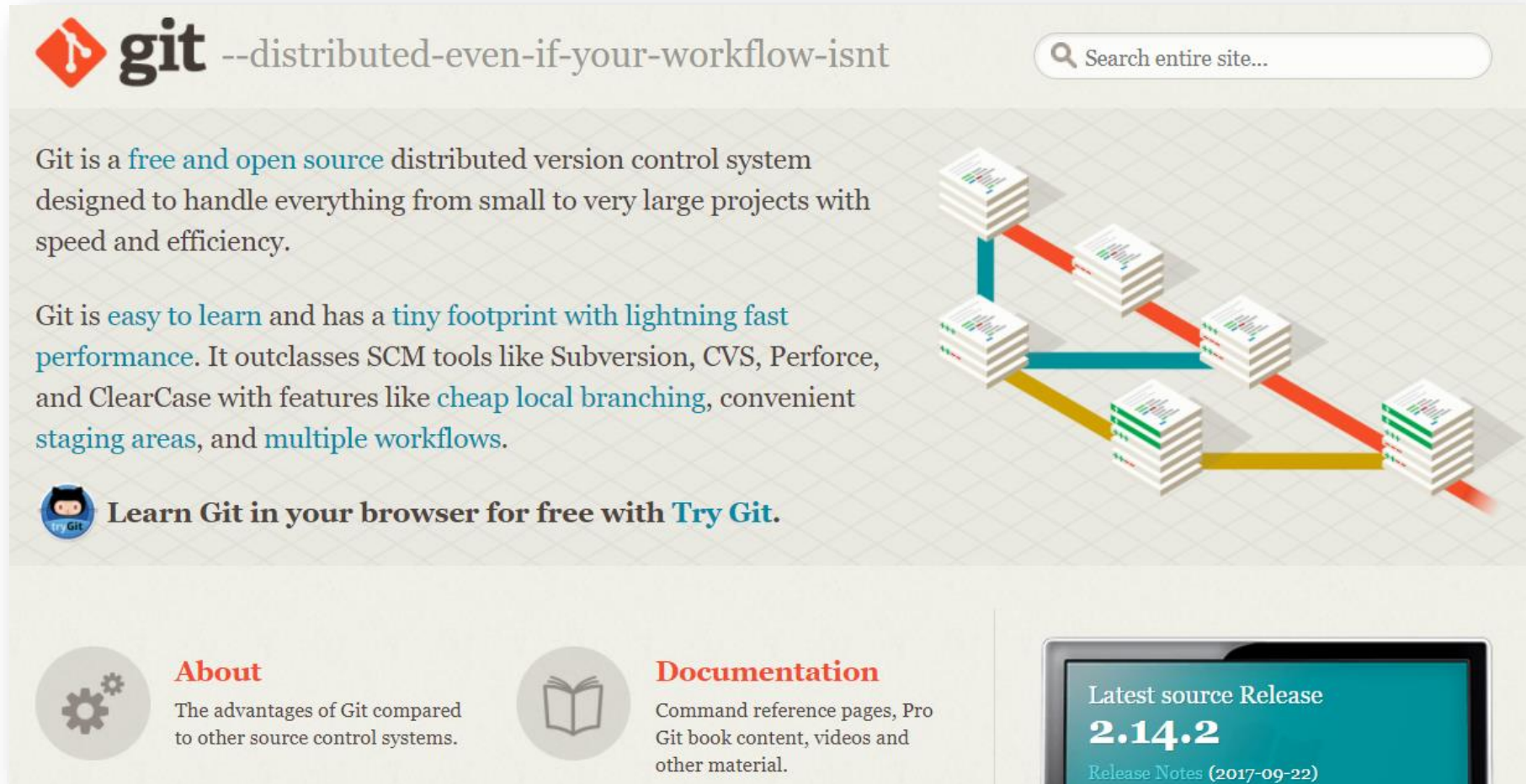
- Il s'interface avec des outils de gestions de projet comme JIRA
- Il s'interface avec des outils d'intégration continue comme JENKINS
- Il participe à l'échange de code source et donc à l'open source



# Télécharger GIT

GIT est téléchargeable sur le site :

<https://git-scm.com/>




The screenshot shows the Git website homepage. At the top left is the Git logo (a red octocat) followed by the text "git --distributed-even-if-your-workflow-isnt". To the right is a search bar with the placeholder text "Search entire site...". Below the header, there is a paragraph describing Git as a "free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency." This is followed by another paragraph stating "Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows." To the right of this text is a diagram illustrating a distributed version control system with multiple stacks of code (represented as books) connected by colored lines (red, blue, yellow) showing branching and merging. Below the text, there is a "Try Git" button with the GitHub logo and the text "Learn Git in your browser for free with Try Git." At the bottom, there are three sections: "About" with a gear icon and text "The advantages of Git compared to other source control systems.", "Documentation" with an open book icon and text "Command reference pages, Pro Git book content, videos and other material.", and a "Latest source Release" section with a tablet icon showing "2.14.2" and "Release Notes (2017-09-22)".

**git** --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

 **Learn Git in your browser for free with Try Git.**

**About**  
The advantages of Git compared to other source control systems.

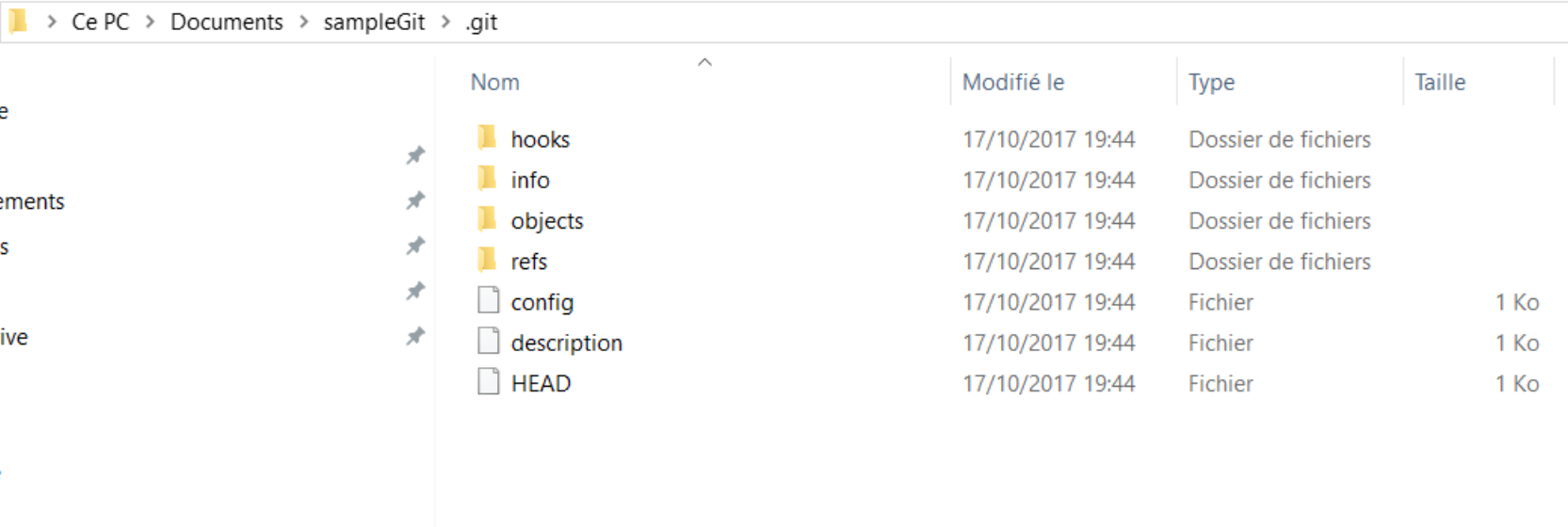
**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Latest source Release**  
**2.14.2**  
Release Notes (2017-09-22)



# Un projet sous GIT

Un projet sous GIT contient toujours un dossier caché nommé .git se dossier contient toutes les modifications d'un projet par l'utilisateur et pas les autres utilisateurs.



Ce PC > Documents > sampleGit > .git				
	Nom	Modifié le	Type	Taille
	hooks	17/10/2017 19:44	Dossier de fichiers	
	info	17/10/2017 19:44	Dossier de fichiers	
	objects	17/10/2017 19:44	Dossier de fichiers	
	refs	17/10/2017 19:44	Dossier de fichiers	
	config	17/10/2017 19:44	Fichier	1 Ko
	description	17/10/2017 19:44	Fichier	1 Ko
	HEAD	17/10/2017 19:44	Fichier	1 Ko

# Un projet sous GIT

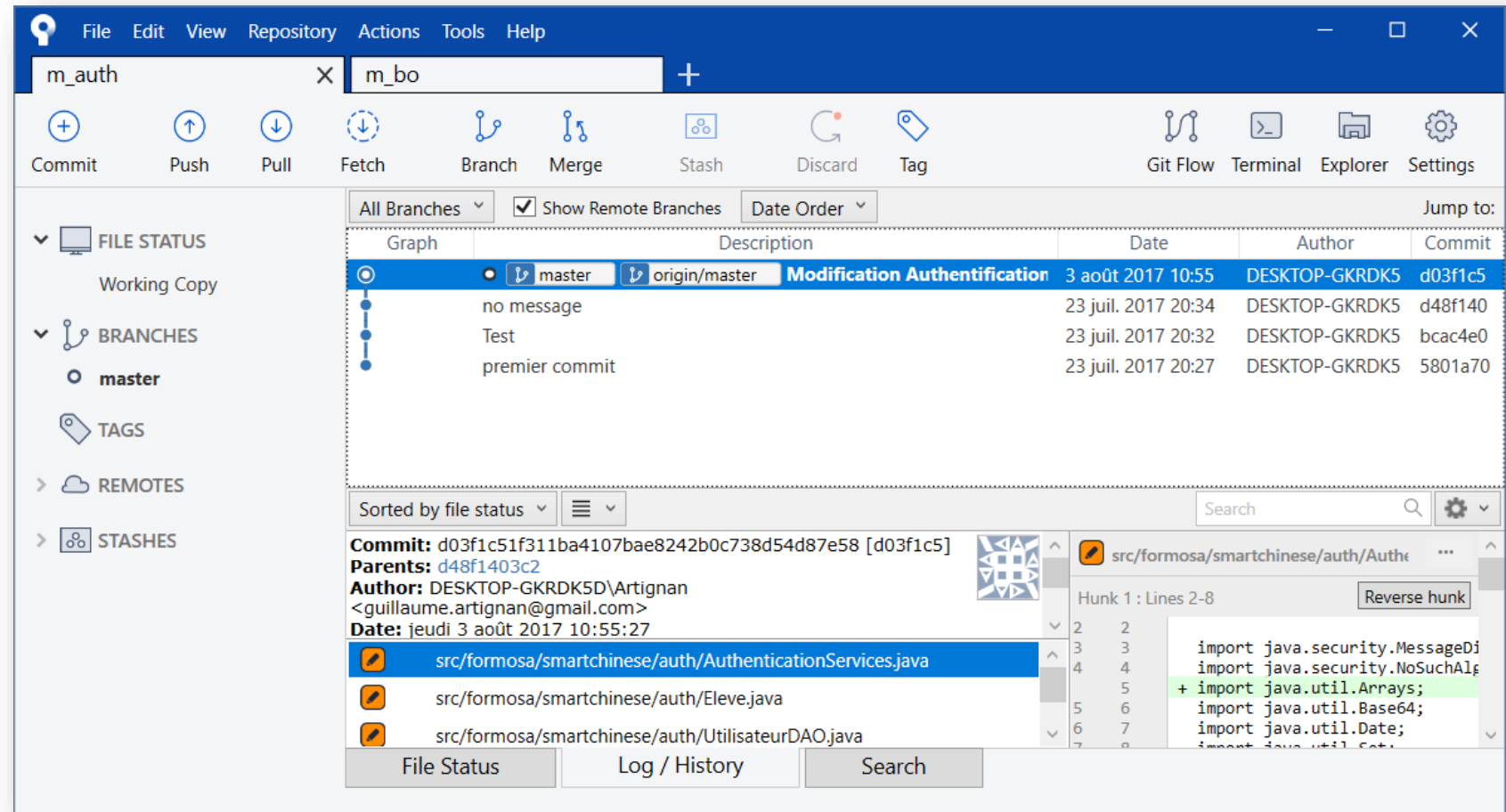
GIT peut fonctionner en ligne de commande mais quelques clients existent rendant l'expérience GIT plus agréable.

## Parmi les client :

Sous unix : SmartGit

Sous windows : SourceTree

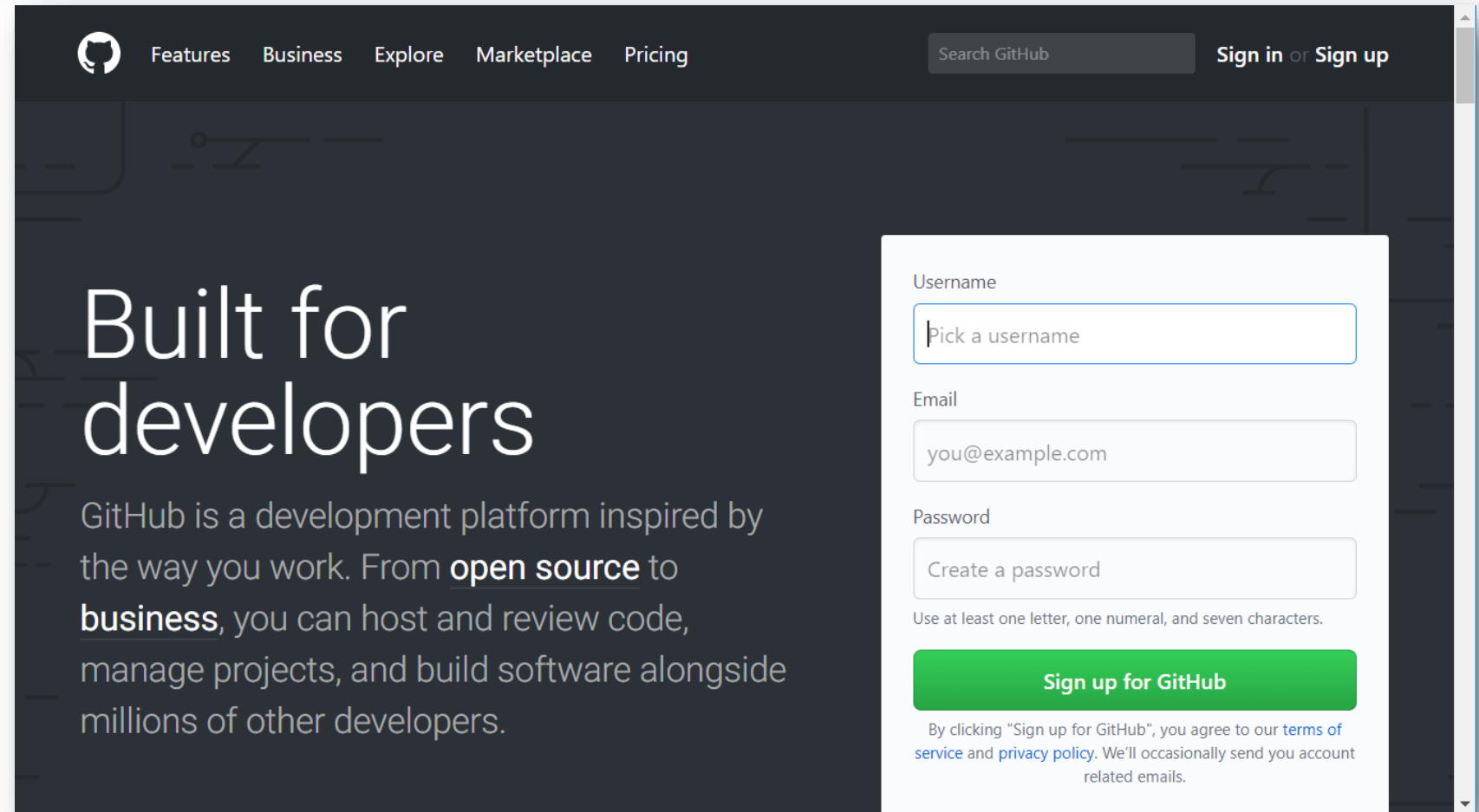
Ces outils donne une interface graphique faisant le lien entre l'utilisateur et les lignes de commandes



# Repository Distant

Il existe différents serveur permettant de stocker les projets, les deux plus connus sont :

- GitHub : Gratuit pour des projets public et payant pour des projets privé
- Bitbucket : Gratuit pour des projets de 5 collaborateurs, payant pour plus.



The screenshot shows the GitHub homepage with a dark theme. The main heading is "Built for developers". Below it, a paragraph describes GitHub as a development platform inspired by the way you work, from open source to business. On the right, there is a sign-up form with fields for Username, Email, and Password. The Username field has a placeholder "Pick a username". The Email field has a placeholder "you@example.com". The Password field has a placeholder "Create a password" and a note "Use at least one letter, one numeral, and seven characters." Below the form is a green button labeled "Sign up for GitHub". At the bottom of the form, there is a disclaimer: "By clicking 'Sign up for GitHub', you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails."

GitHub

Features Business Explore Marketplace Pricing

Search GitHub Sign in or Sign up

## Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside millions of other developers.

Username

Email

Password

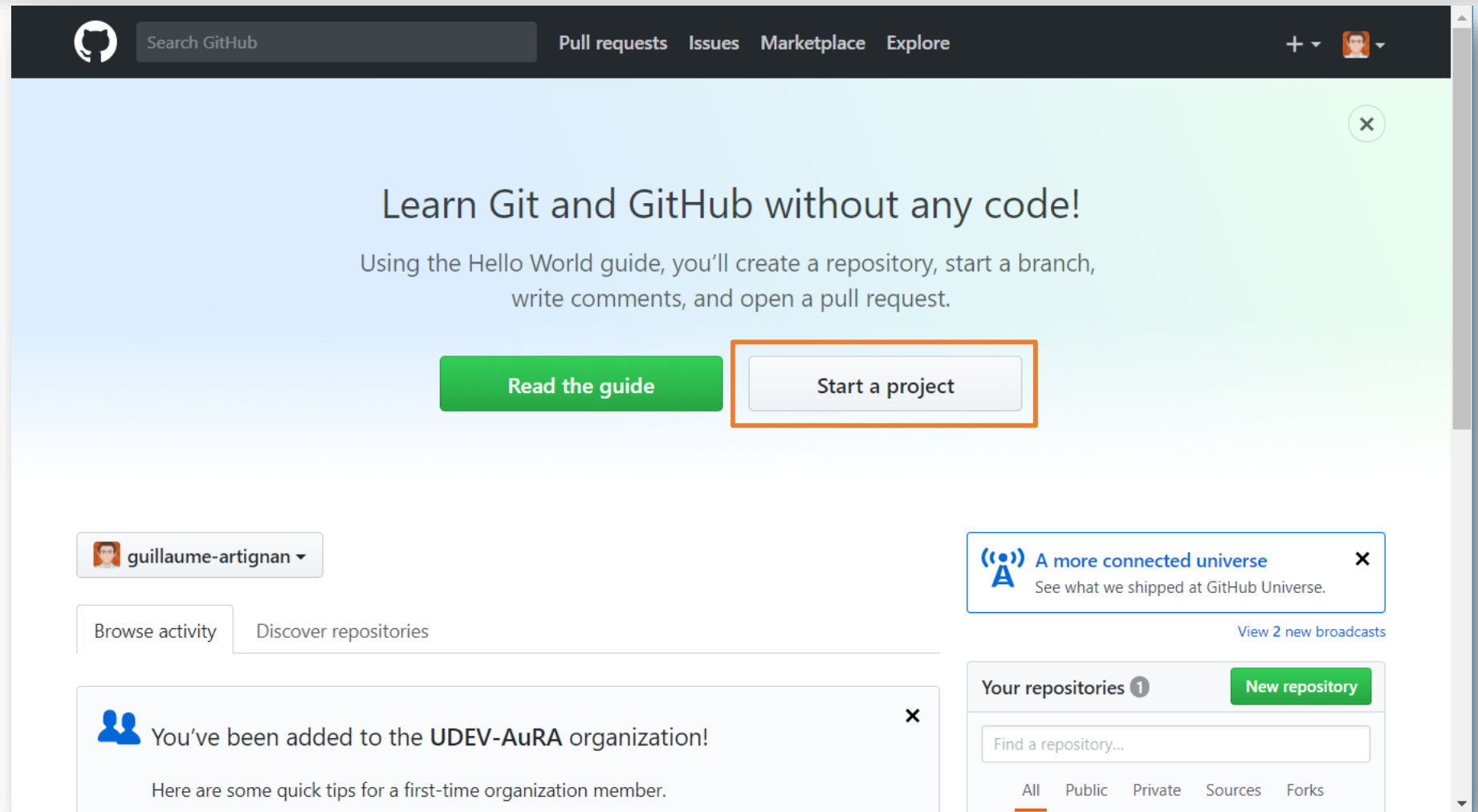
Use at least one letter, one numeral, and seven characters.

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.

# GitHub

GitHub propose la création du projet en ligne puis la récupération de ce projet est possible en utilisant SourceTree.





# Créer un nouveau repository

Pour créer un nouveau repository il suffit de nommer le repository sur GIT HUB

Et cliquer sur le bouton : Create Repository

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



guillaume-artignan ▾

Repository name

projet1



Great repository names are short and memorable. Need inspiration? How about [literate-adventure](#).

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾



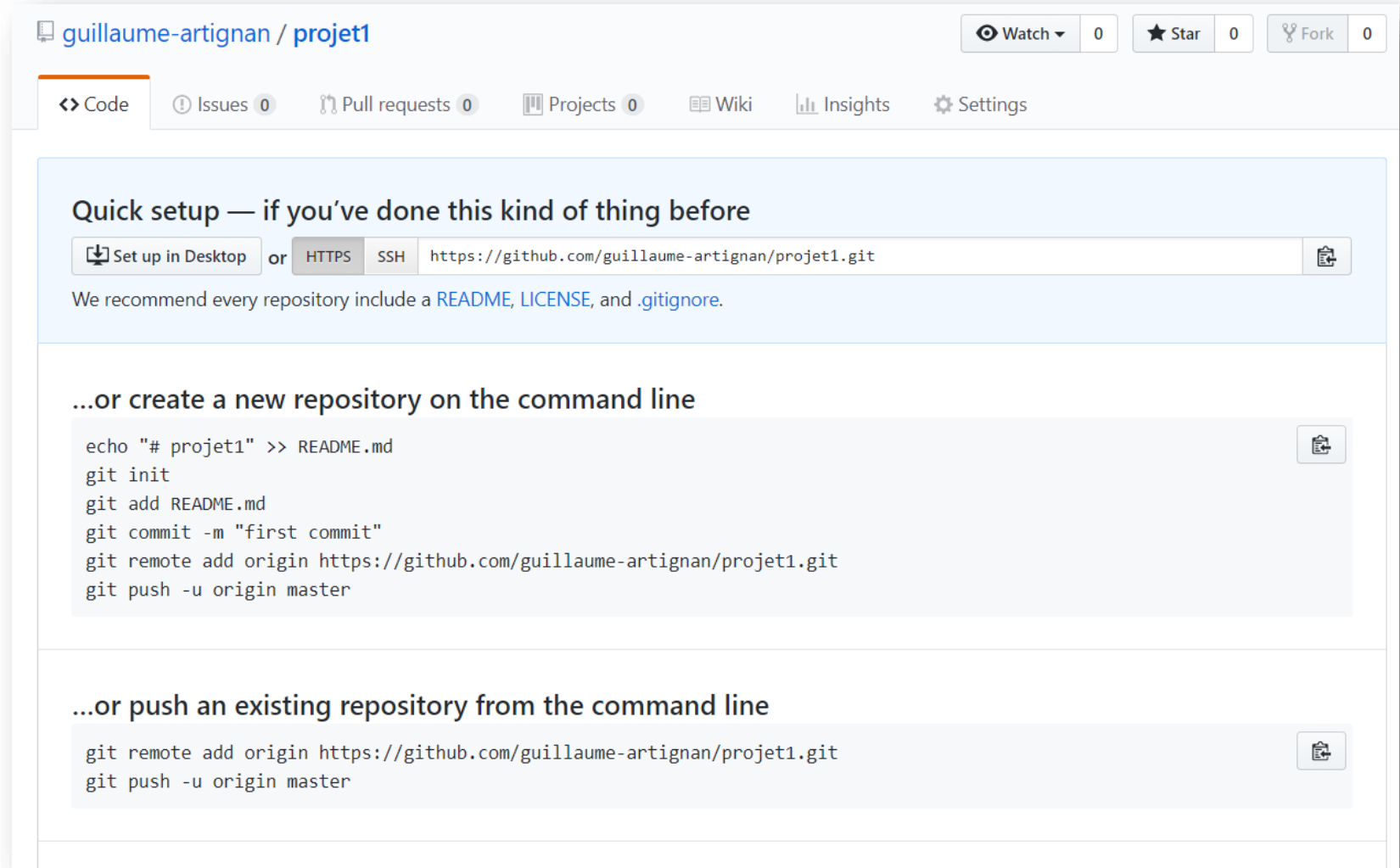
Create repository

# Création du Repository

GIT Hub a alors créé un repository.

Il donne les lignes de commande pour ceux qui souhaite piloter leur projet git avec des lignes de commande

Nous concernant nous n'avons besoin que de l'URL



The screenshot shows the GitHub interface for a repository named 'projet1' by user 'guillaume-artignan'. At the top, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. Below this is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. The main content area is titled 'Quick setup — if you've done this kind of thing before' and provides three options: 'Set up in Desktop', 'HTTPS', and 'SSH'. The 'HTTPS' option is selected, showing the URL 'https://github.com/guillaume-artignan/projet1.git'. Below this, a recommendation states: 'We recommend every repository include a README, LICENSE, and .gitignore.' The next section is titled '...or create a new repository on the command line' and contains a code block with the following commands: 

```
echo "# projet1" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/guillaume-artignan/projet1.git
git push -u origin master
```

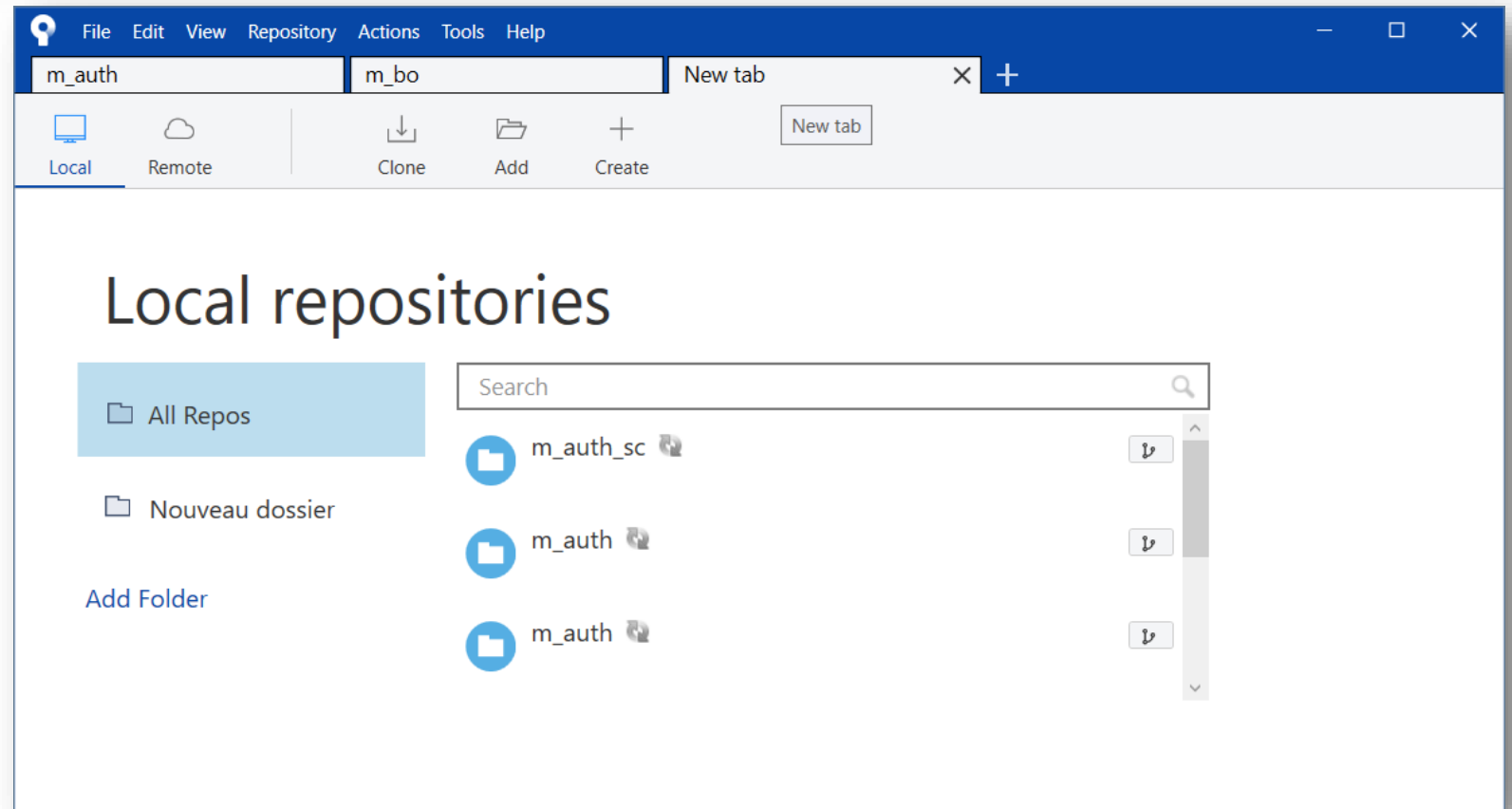
 The final section is titled '...or push an existing repository from the command line' and contains a code block with the following commands: 

```
git remote add origin https://github.com/guillaume-artignan/projet1.git
git push -u origin master
```

# Sous SourceTree

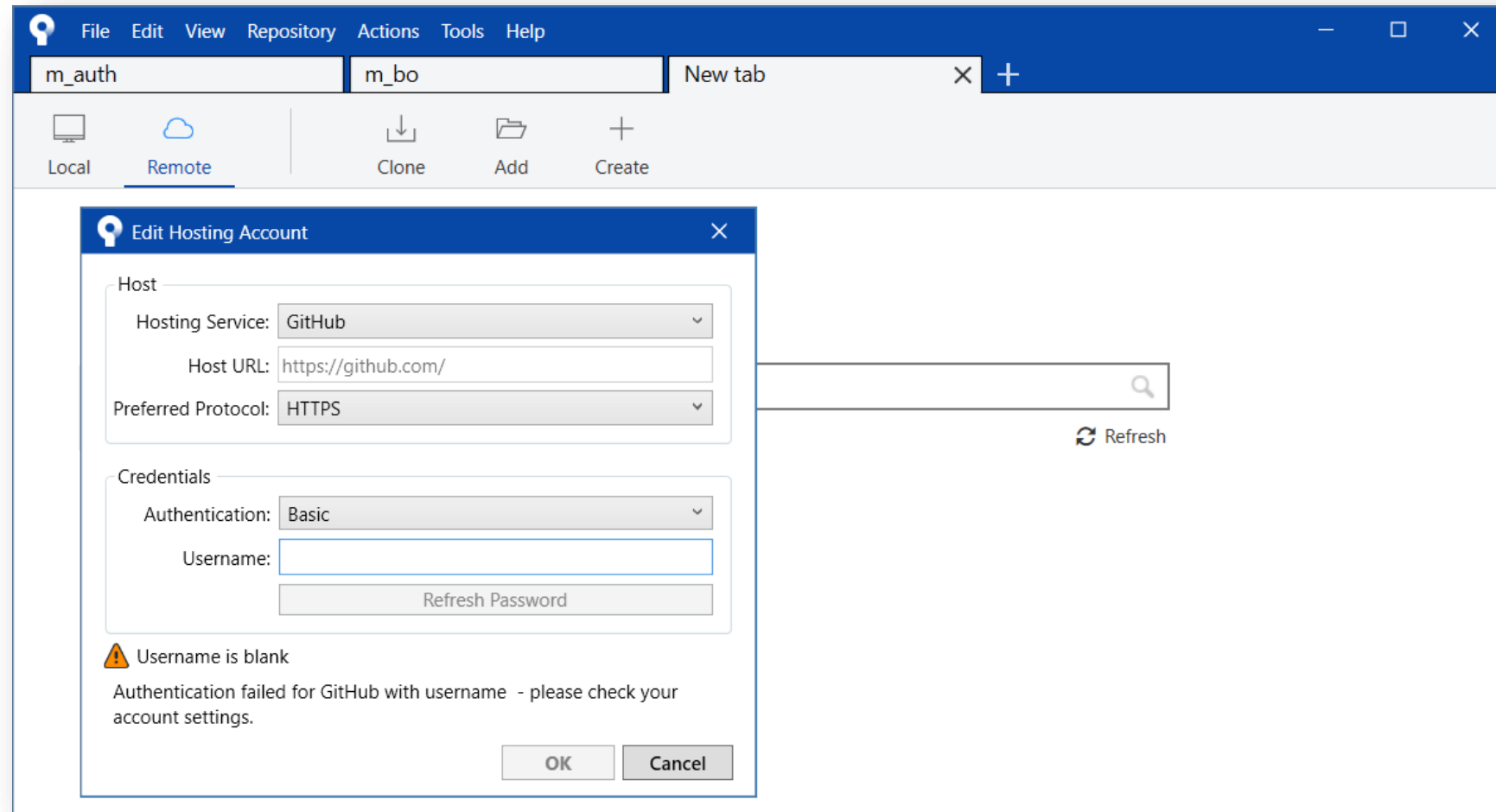
Sous sourceTree nous pouvons ajouter un Repository en cliquant sur l'onglet.

Puis choisir : **Remote**



# Sous SourceTree

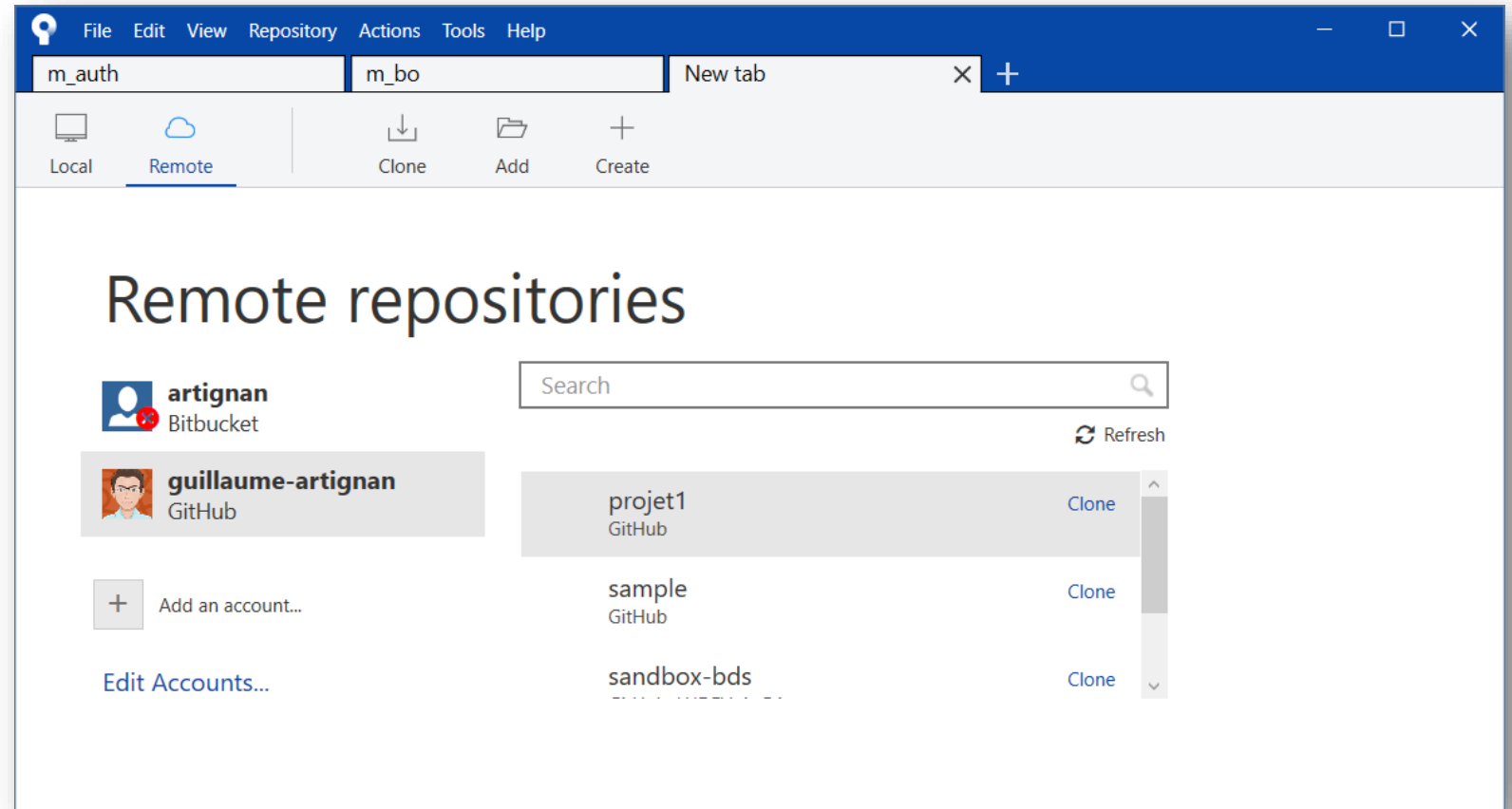
Il est possible de rajouter  
notre Hébergeur de code  
source GitHub





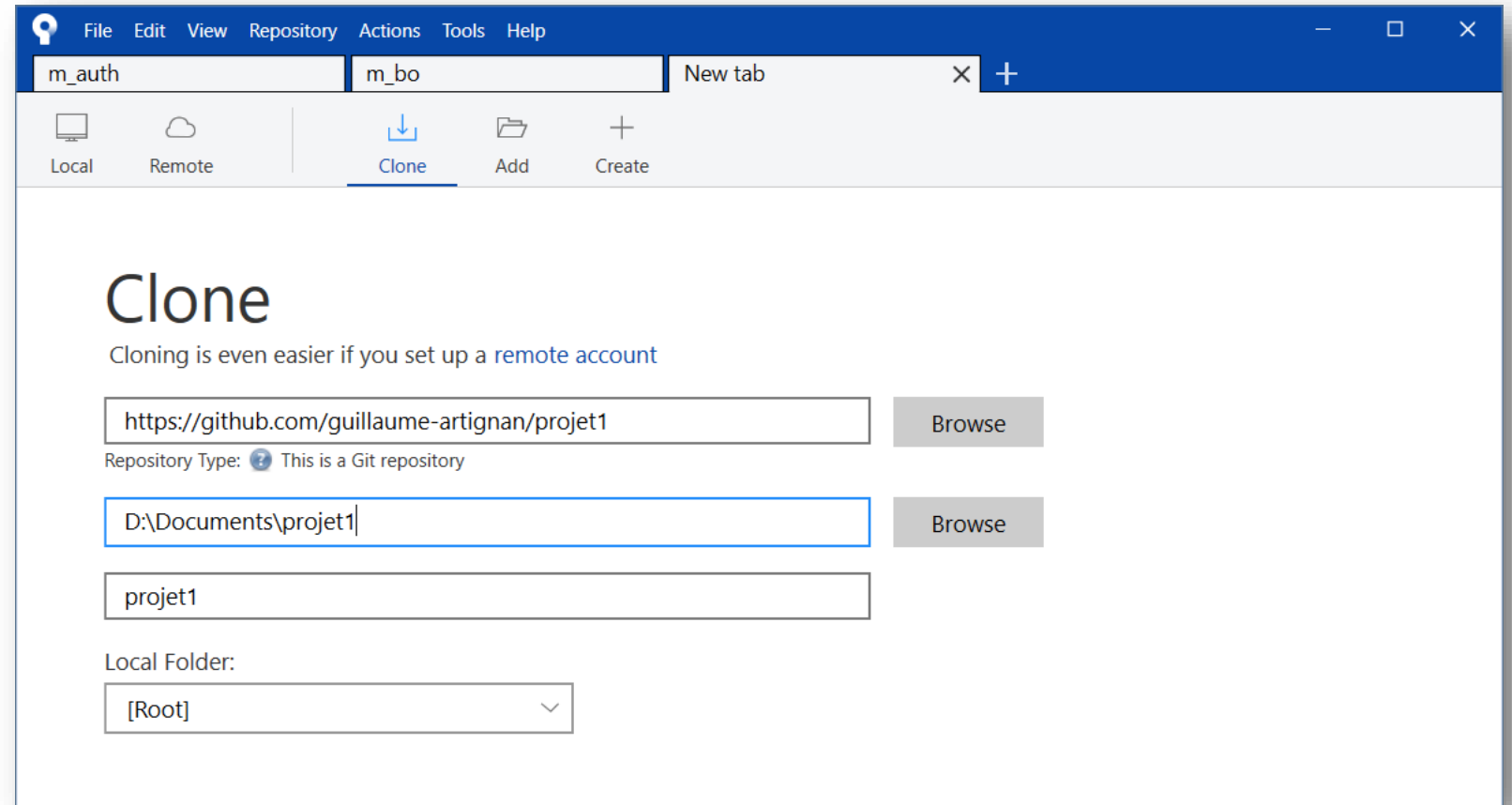
# Selectionner le repository

Une fois connecté vous pouvez sélectionner le repository que vous souhaitez.



# Associer le repository distant à un répertoire

Il suffit alors d'associer le repository distant à un répertoire **vide** du disque dur.

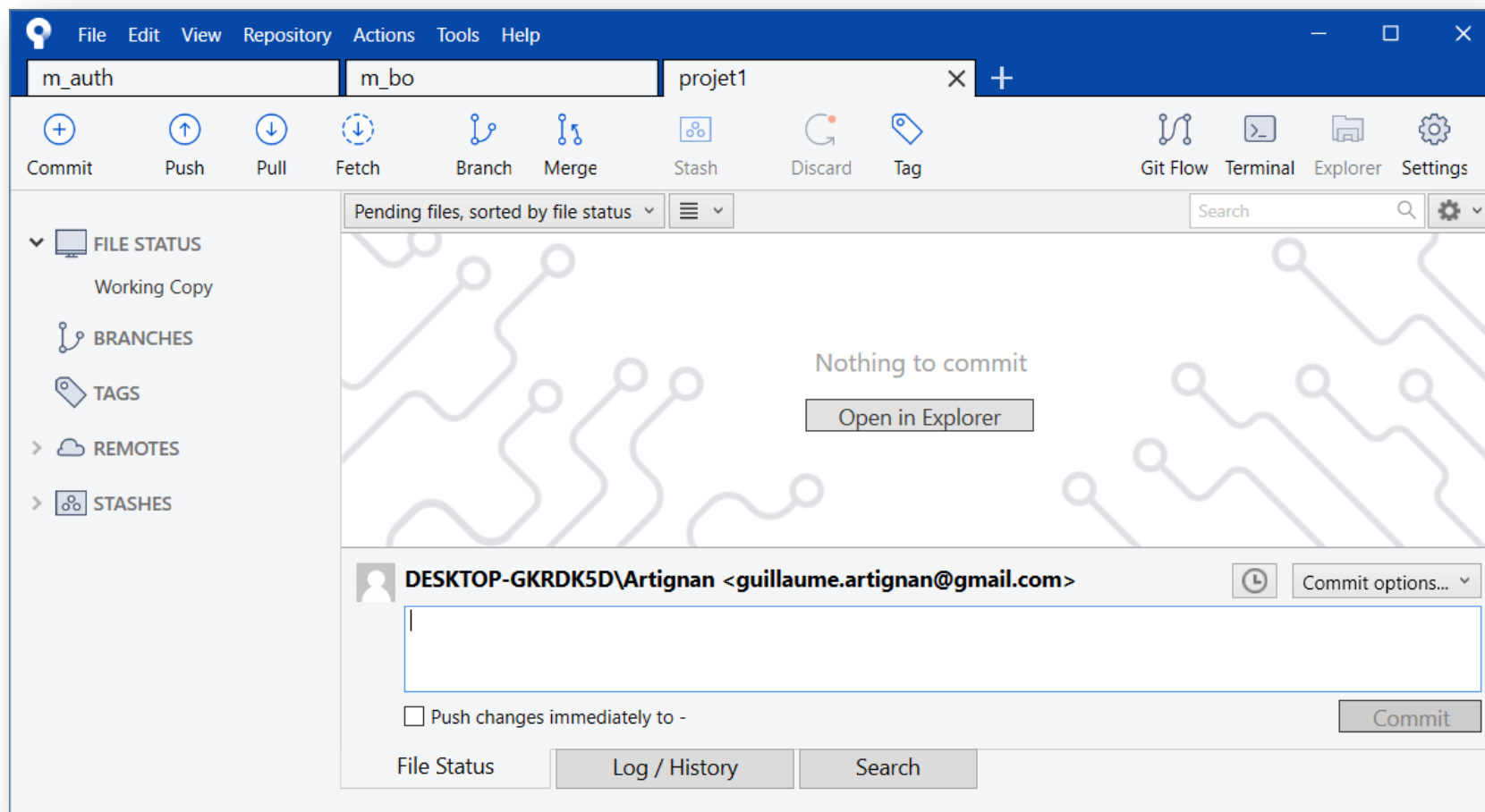


The screenshot shows the 'Clone' dialog box in Visual Studio Code. The window has a blue title bar with the menu 'File Edit View Repository Actions Tools Help'. Below the title bar, there are tabs for 'm\_auth', 'm\_bo', and 'New tab'. The 'Clone' tab is active, showing a toolbar with 'Local', 'Remote', 'Clone', 'Add', and 'Create' buttons. The main content area is titled 'Clone' and includes the text 'Cloning is even easier if you set up a [remote account](#)'. There are three input fields: the first contains 'https://github.com/guillaume-artignan/projet1' with a 'Browse' button to its right; the second contains 'D:\Documents\projet1' with a 'Browse' button to its right; the third contains 'projet1'. Below these fields is a 'Local Folder:' label and a dropdown menu currently showing '[Root]'.

# Félicitation !

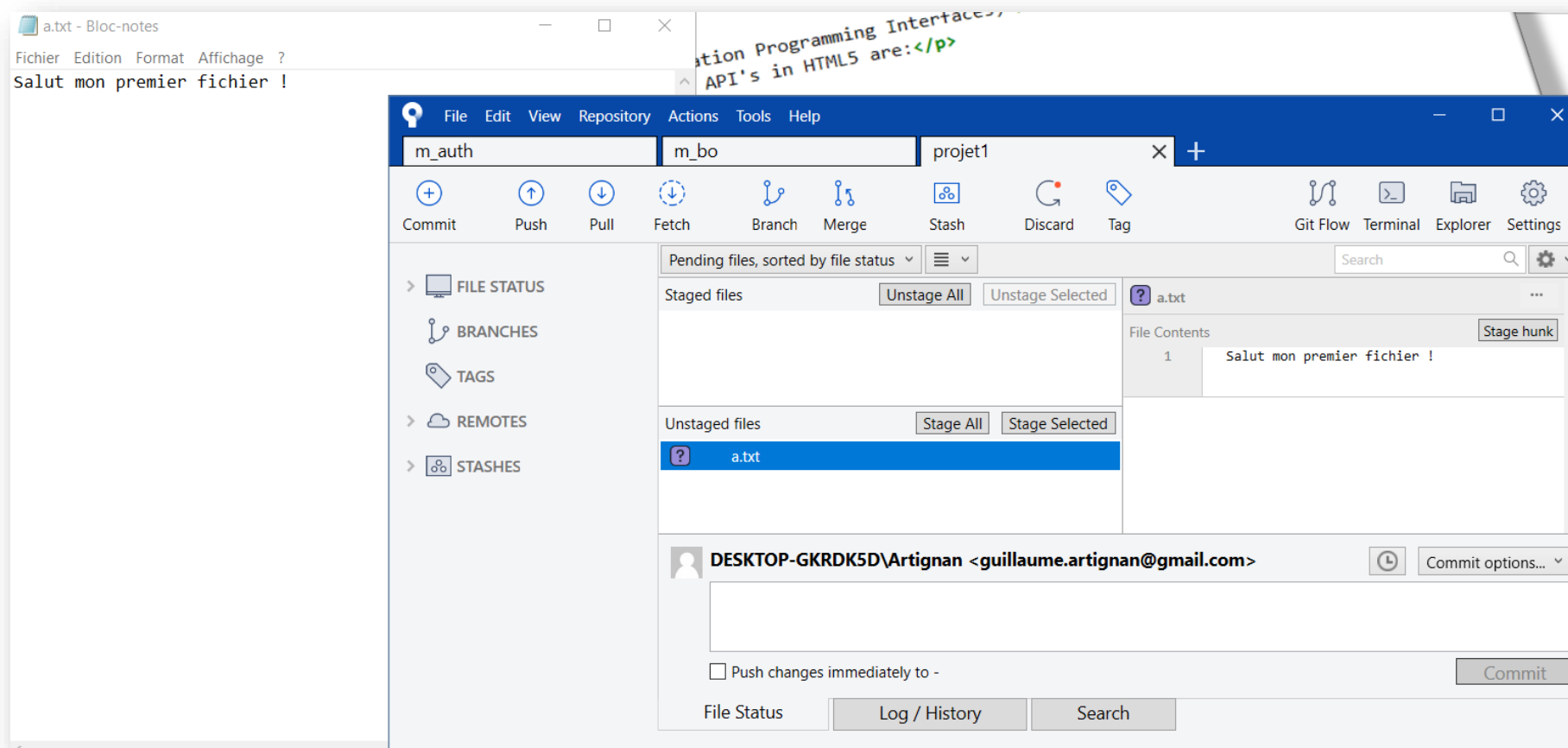
Félicitation vous avez  
associé à un repository  
distant un répertoire sur  
votre disque !

**Tout est prêt à fonctionner !**



# La modification d'un fichier

La création, la modification et la suppression d'un fichier sont directement détaillés par l'outil



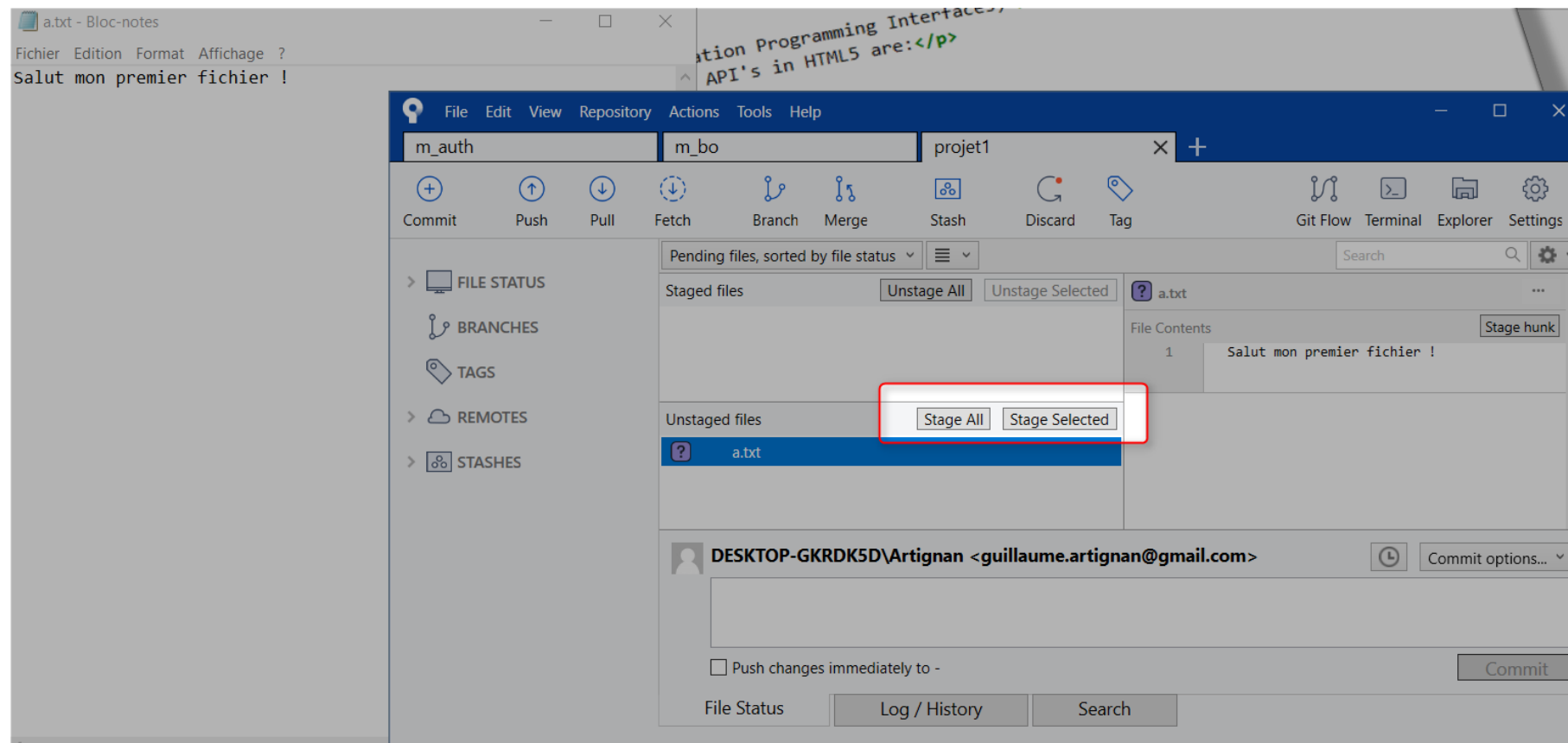


# Etape du Staging

Pour valider/commiter certaines modifications nous devons sélectionner les travaux à valider, cette sélection se nomme "Staging"

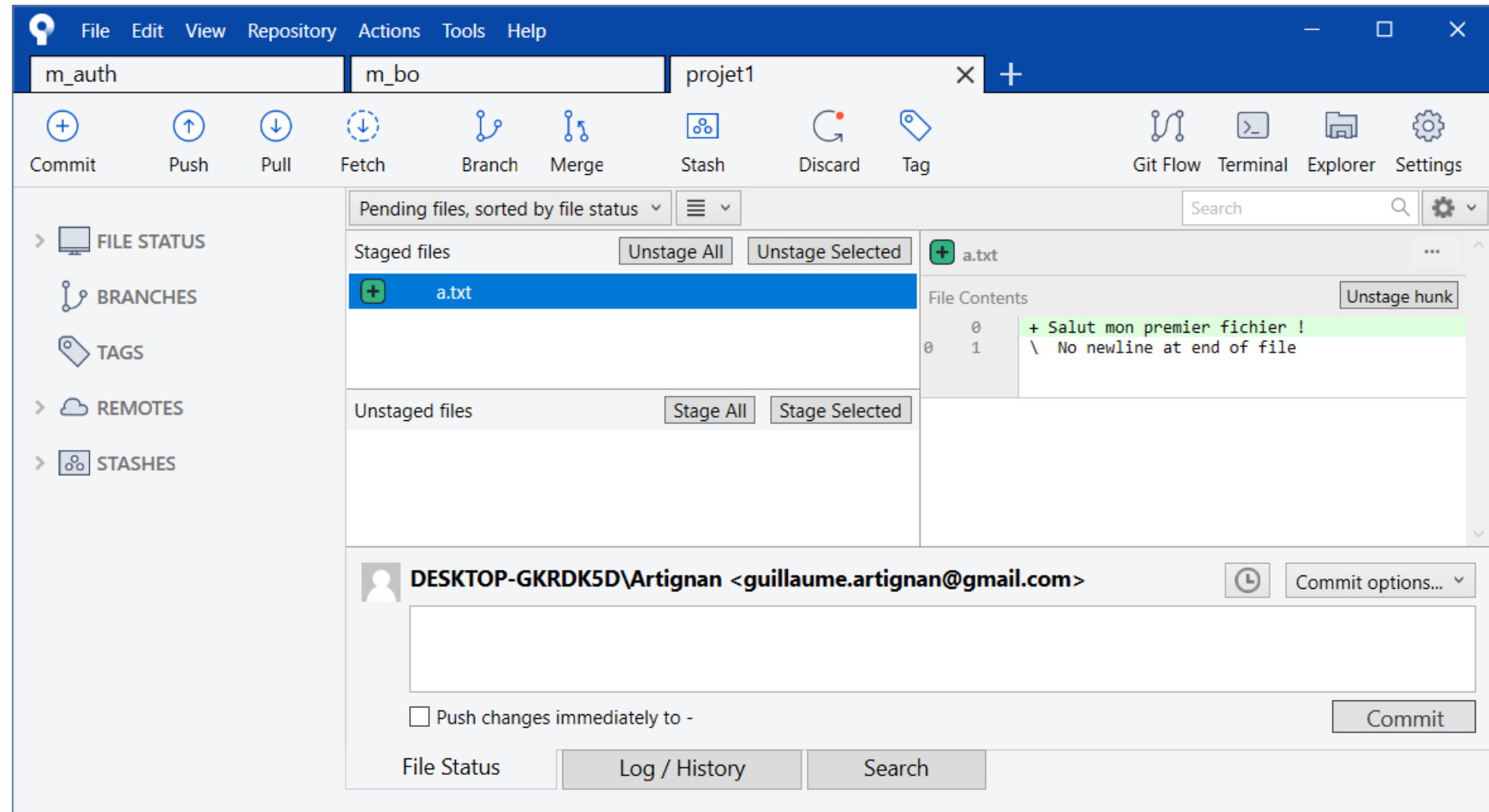
On peut imaginer le staging comme une zone avant la validation.

"Un sas avant l'embarquement"



# Etape du Staging

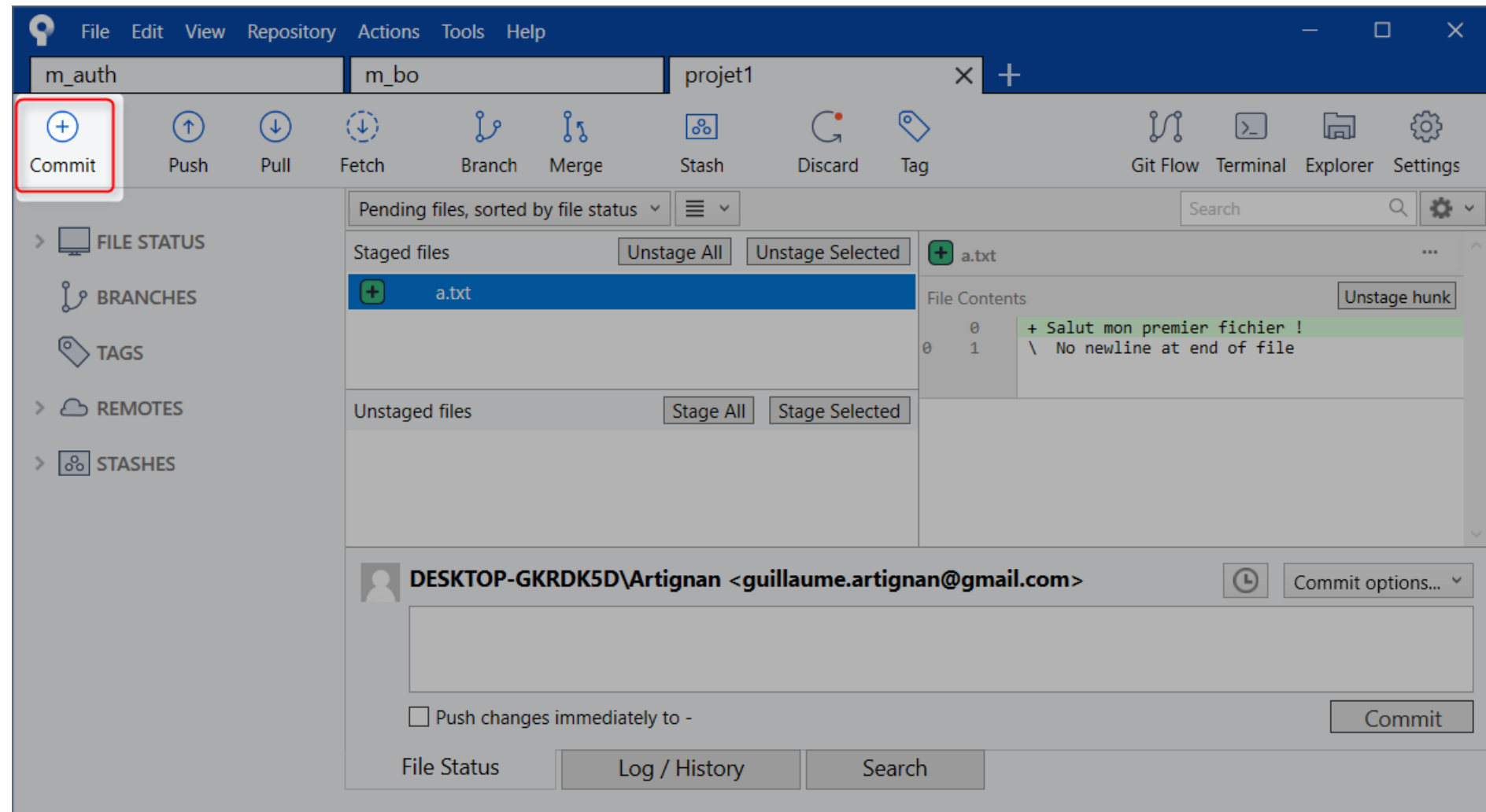
Les modifications sur le fichier a.txt ont été "stagé".



# Etape de la validation / Commit

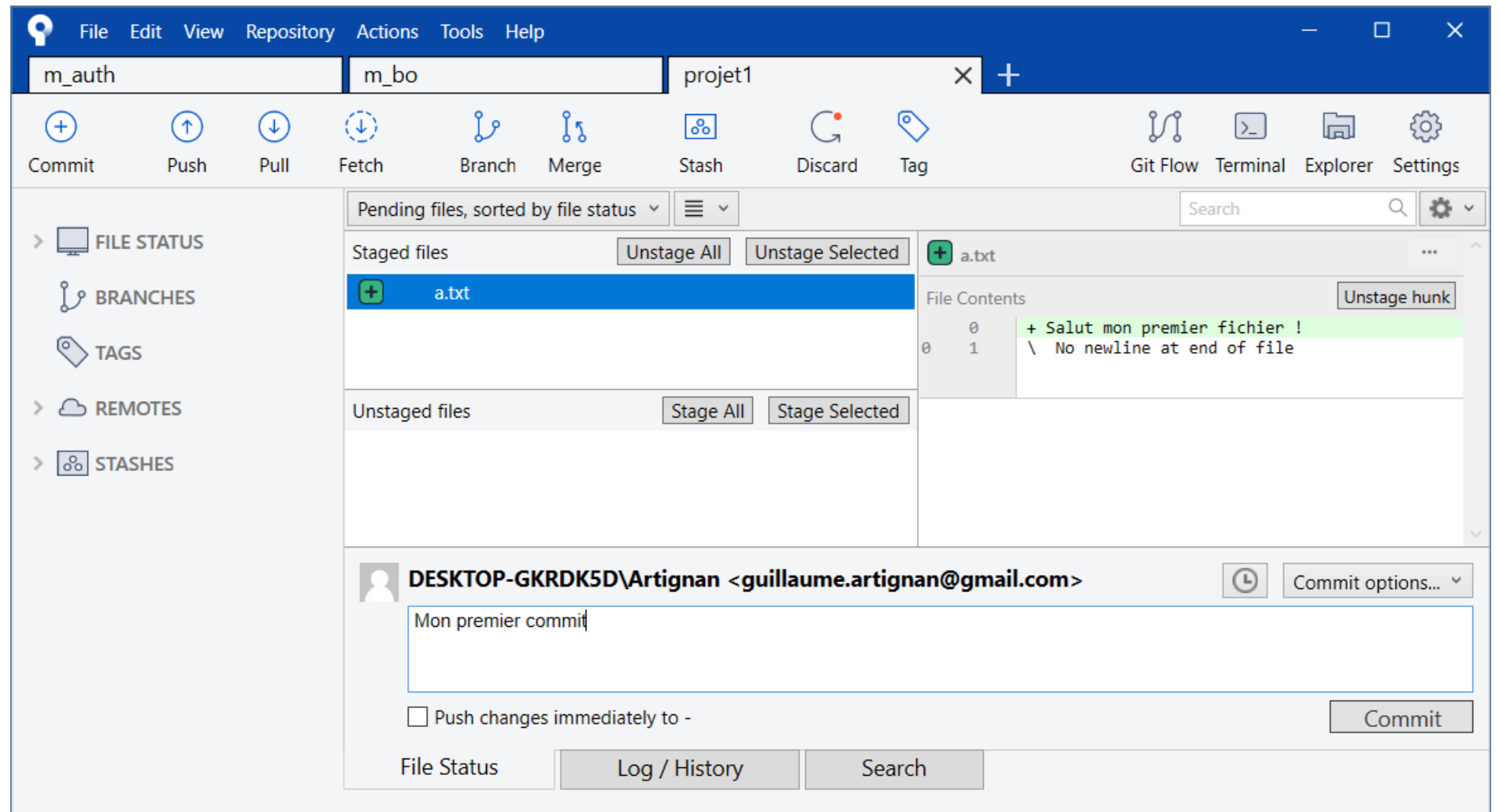
Le commit est l'étape validant une ensemble de modifications sur un projet.

Un commit porte une description.



# Etape de la validation / Commit

En remplissant la zone de description et en "committant" les modification. Celle-ci sont sauvegardées.

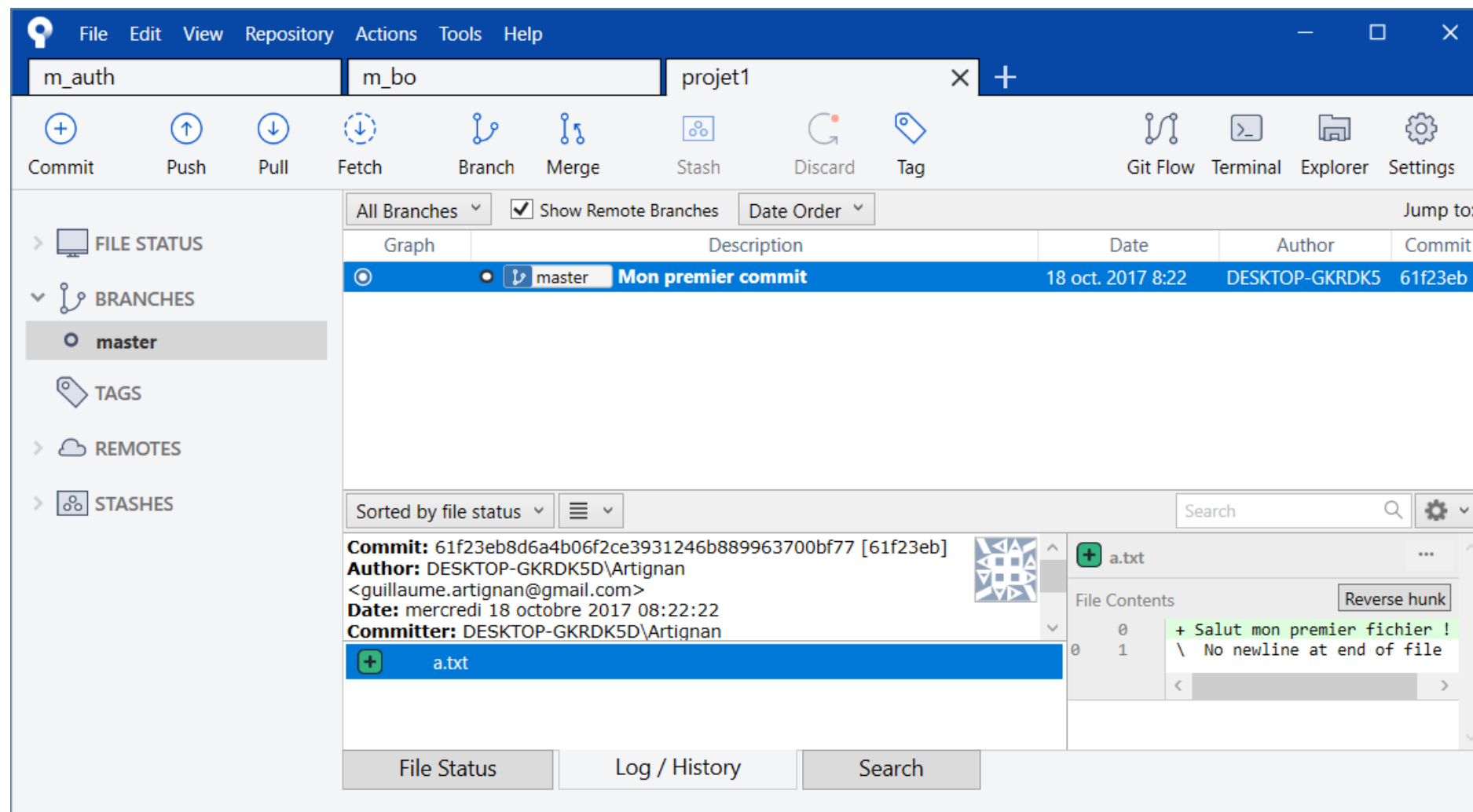




# Etape du commit

Les commit sont représentés par des point reliés par des ligne.

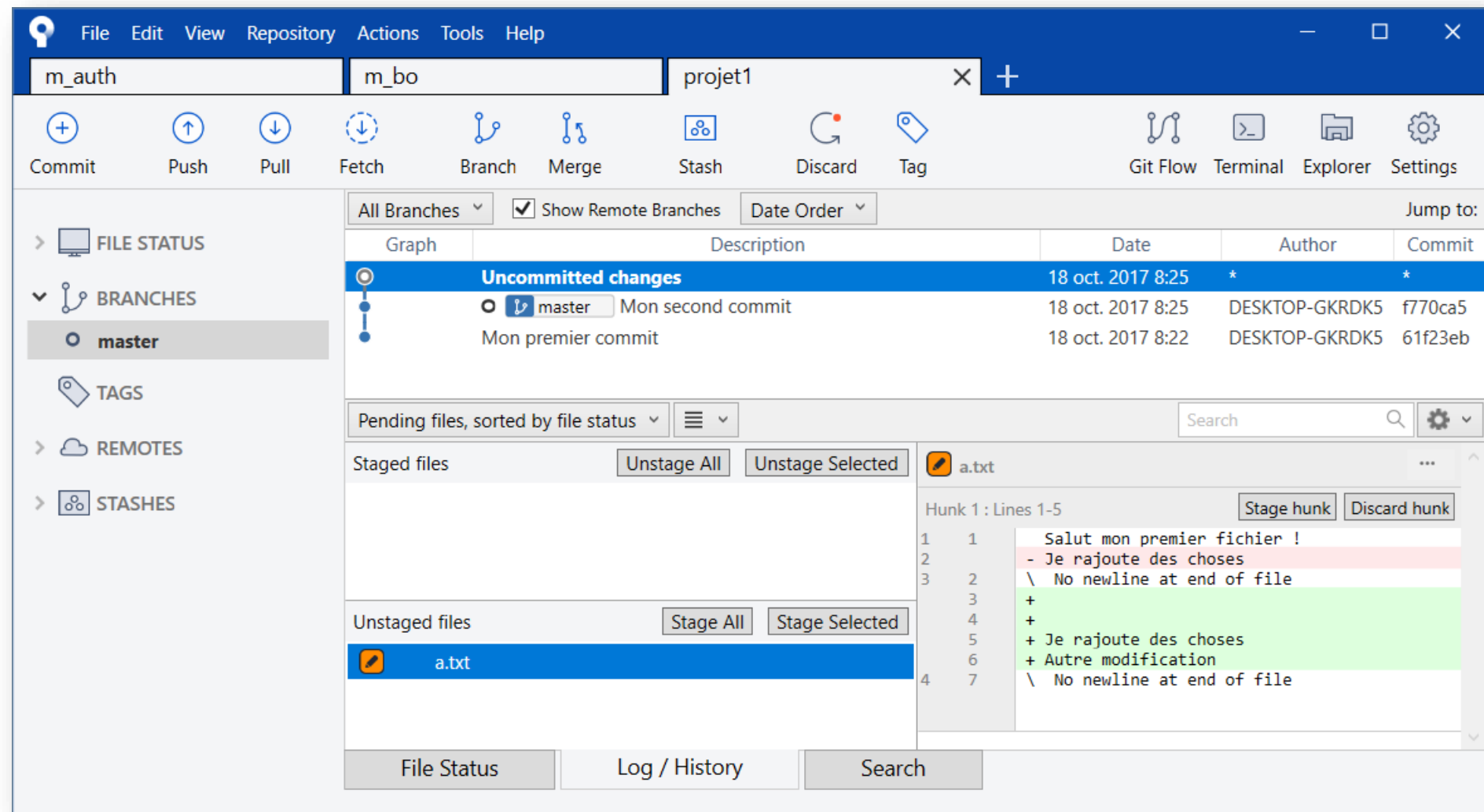
Ci-contre un seul commit a été effectué nous possédons donc qu'un seul point.



# Etape du commit

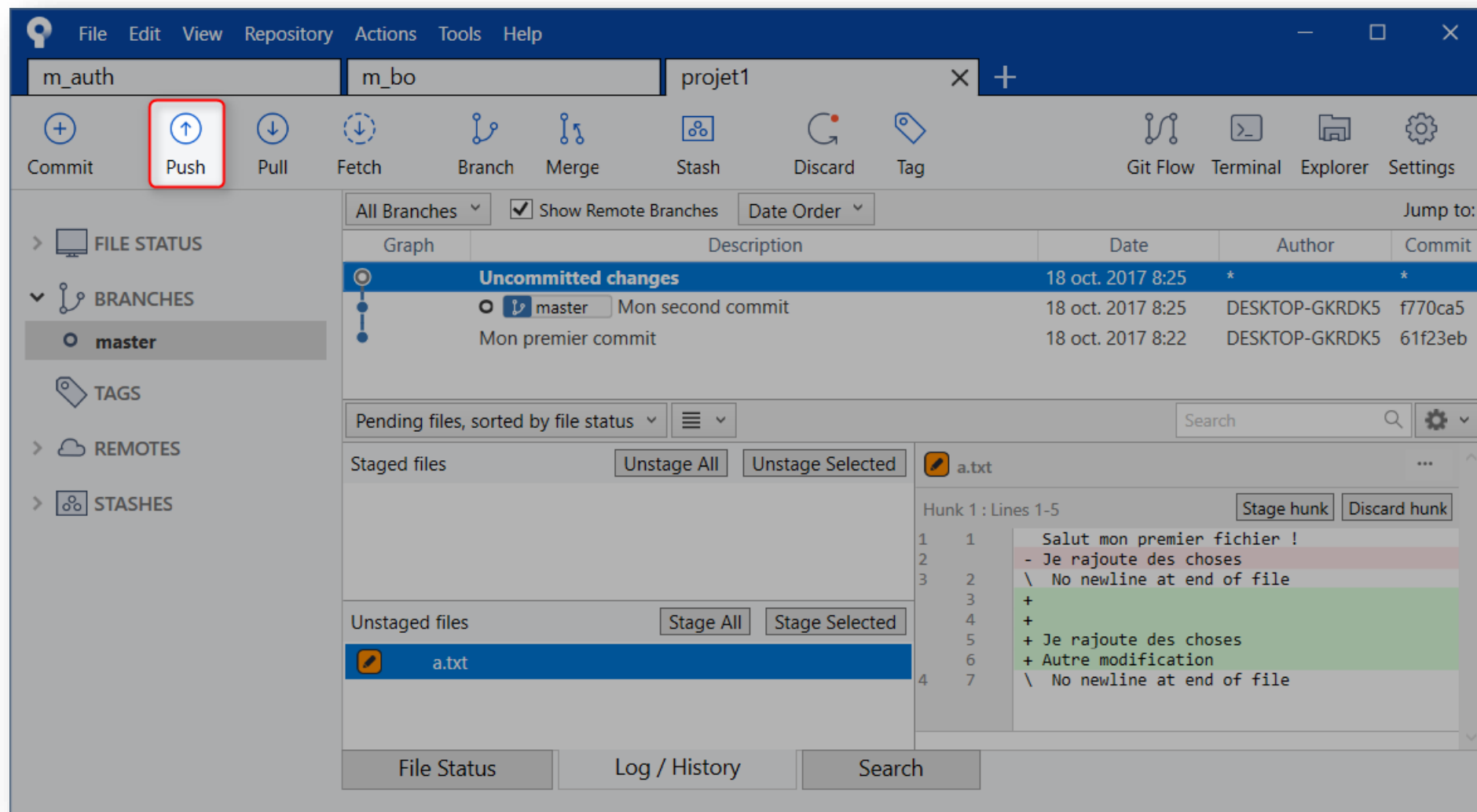
Plusieurs modification  
committées entraine la  
création de ses points

Le label "master"  
représente la  
"branche" principale  
du projet.



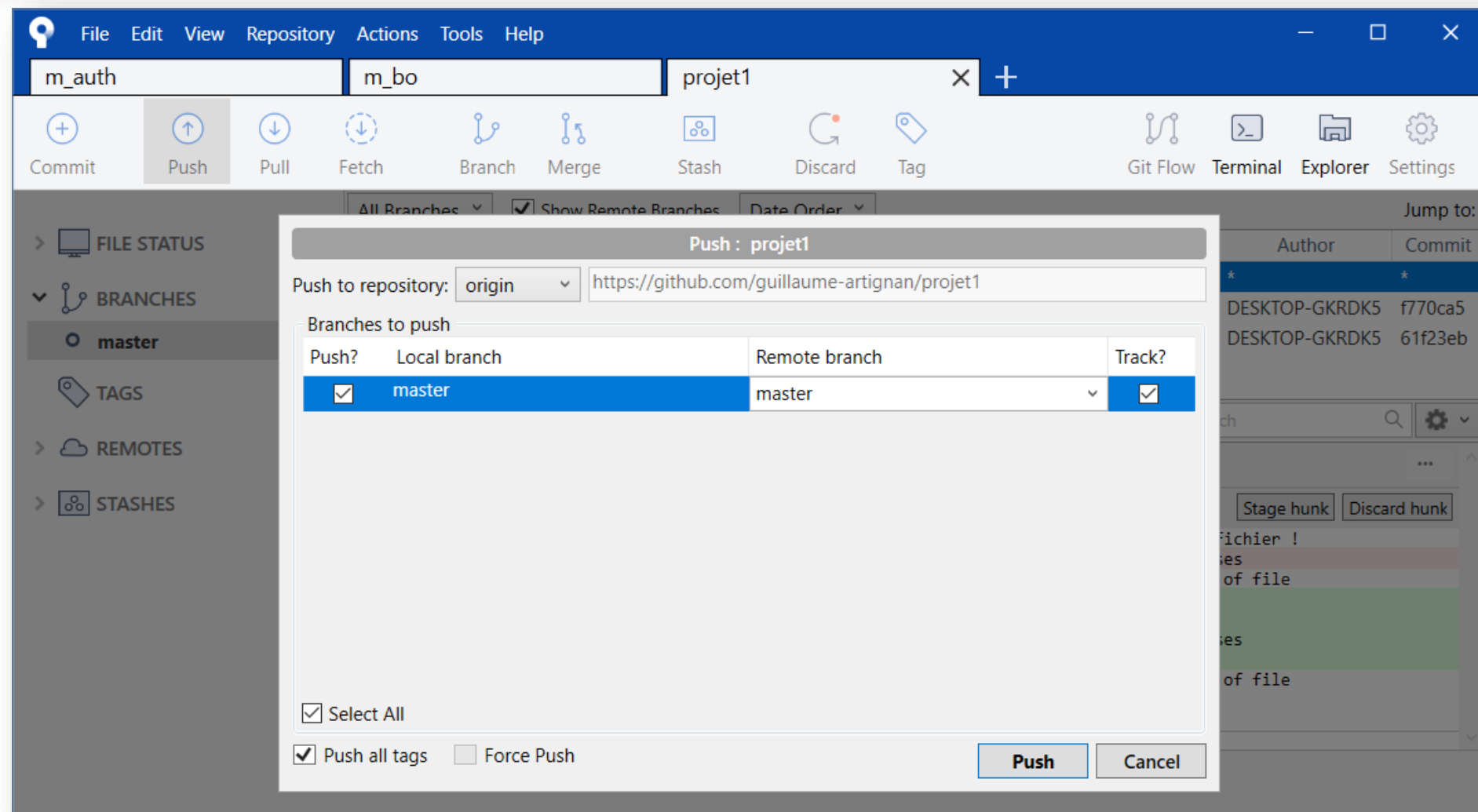
# On push sur le serveur

L'étape du push sur le serveur est une étape importante elle consiste à sauvegarder les modifications sur un serveur distant pour partager ces modifications avec d'autres.



# On push sur le serveur

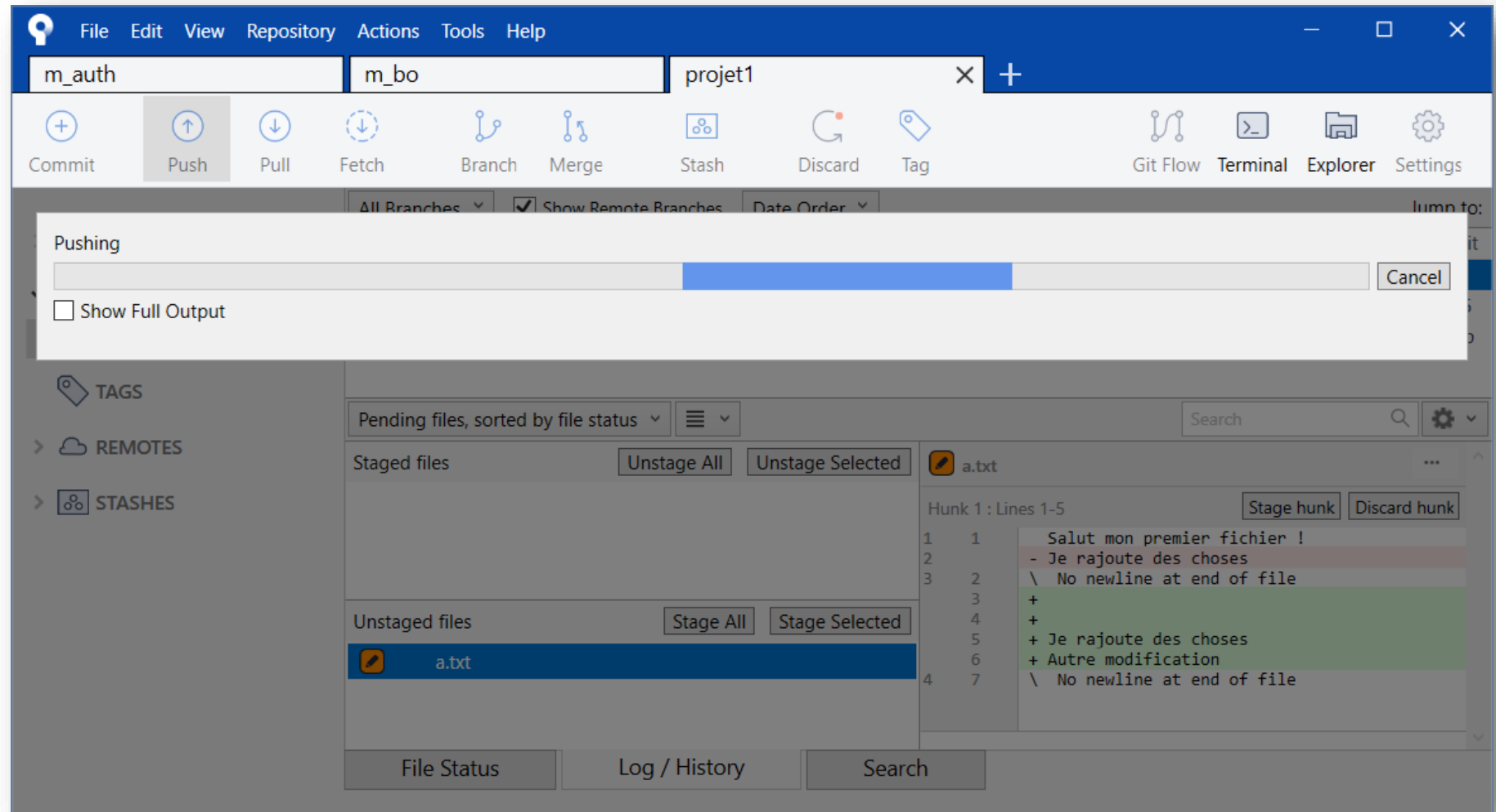
Pour pusher nous devons sélectionner la branche locale que l'on souhaite pusher vers quelle branche distante.





# On push sur le serveur

Le processus prend en général que quelques secondes



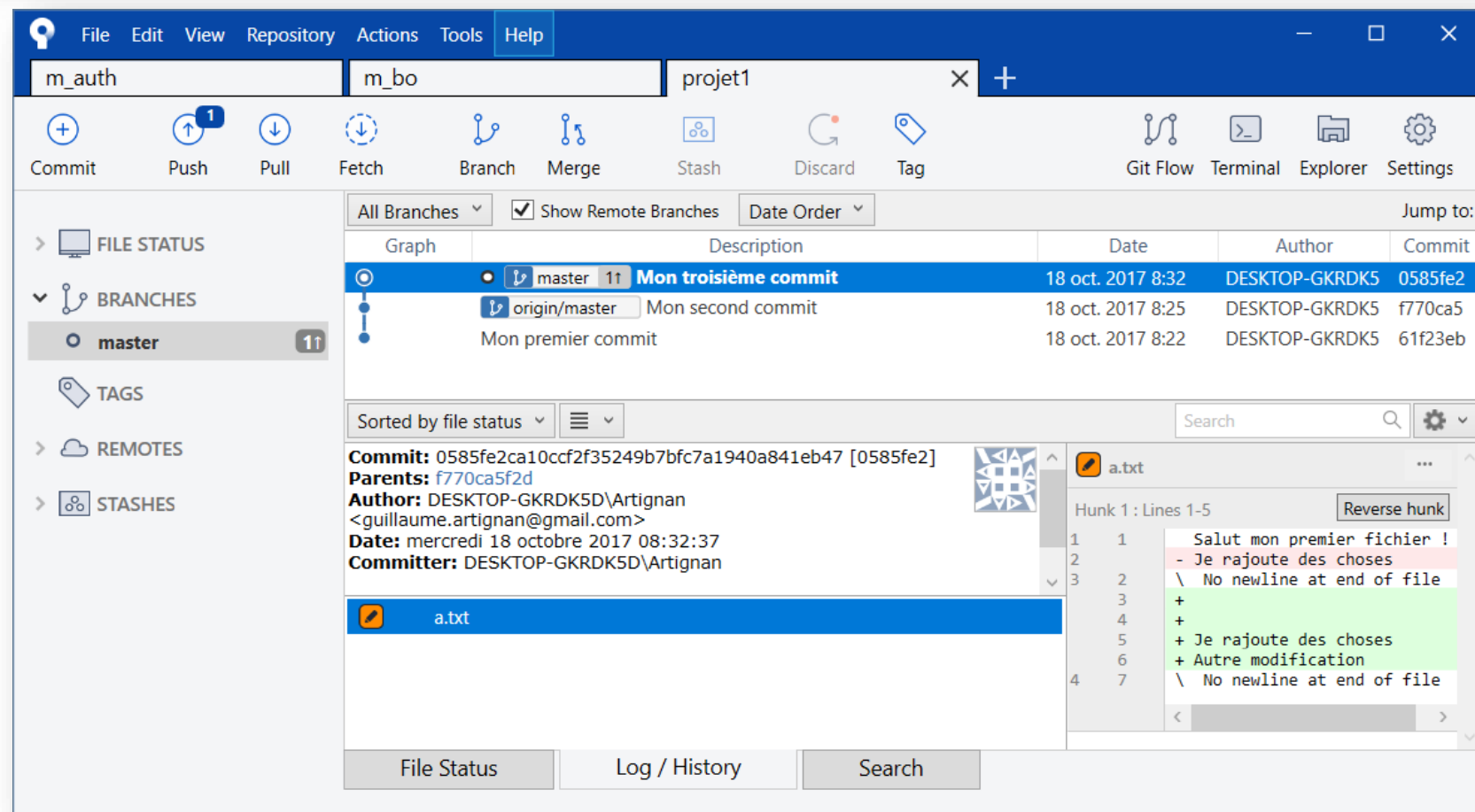
# La branche locale et la branche distante

Après notre push. Si nous committons les modification restantes. Nous voyons apparaître deux branches :

- Master
- Origin/master

La première est la branche locale, la seconde est le branche distante.

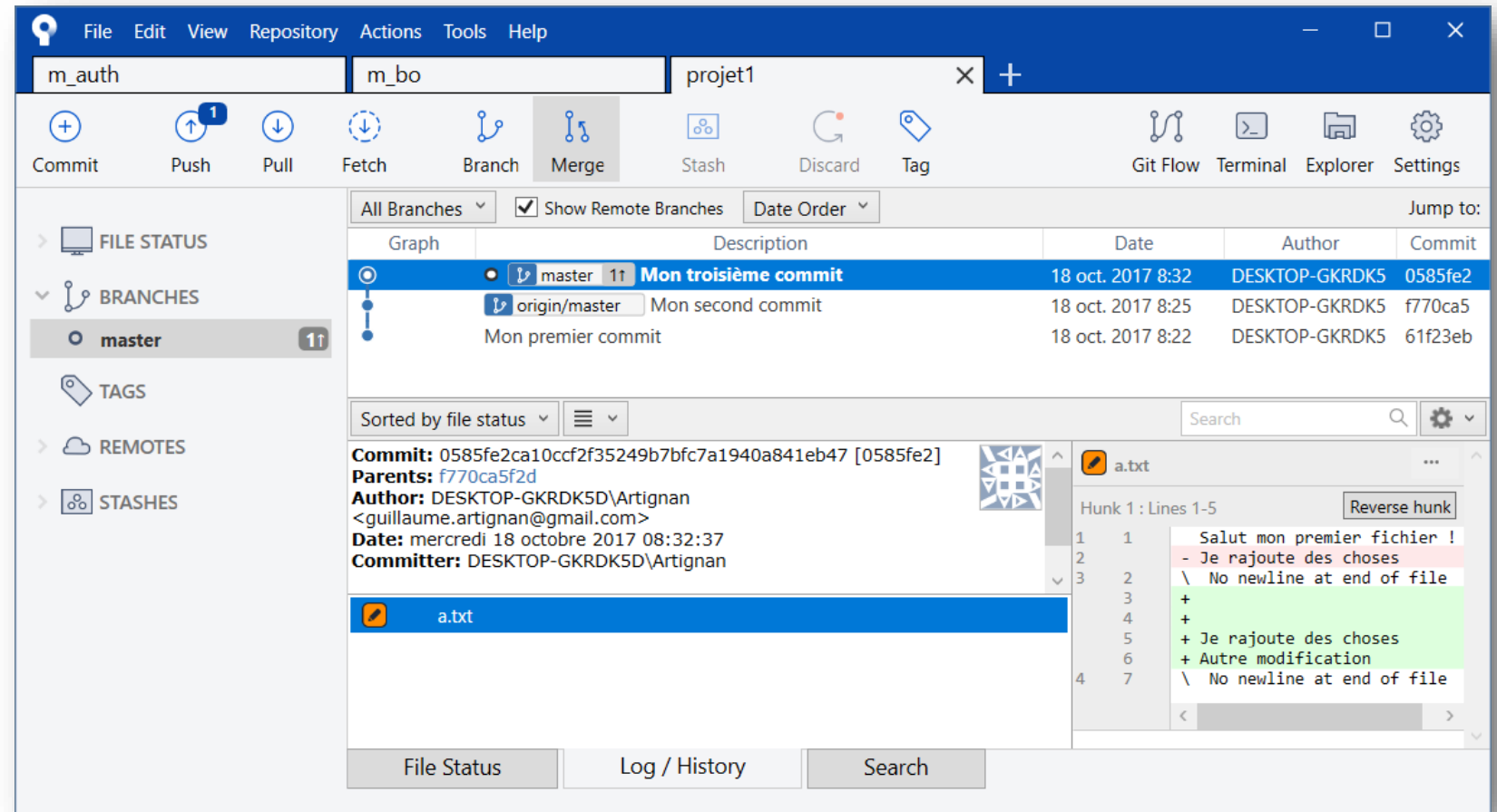
Un petit 1 apparaît sur le bouton push indiquant le décalage entre les deux branches.



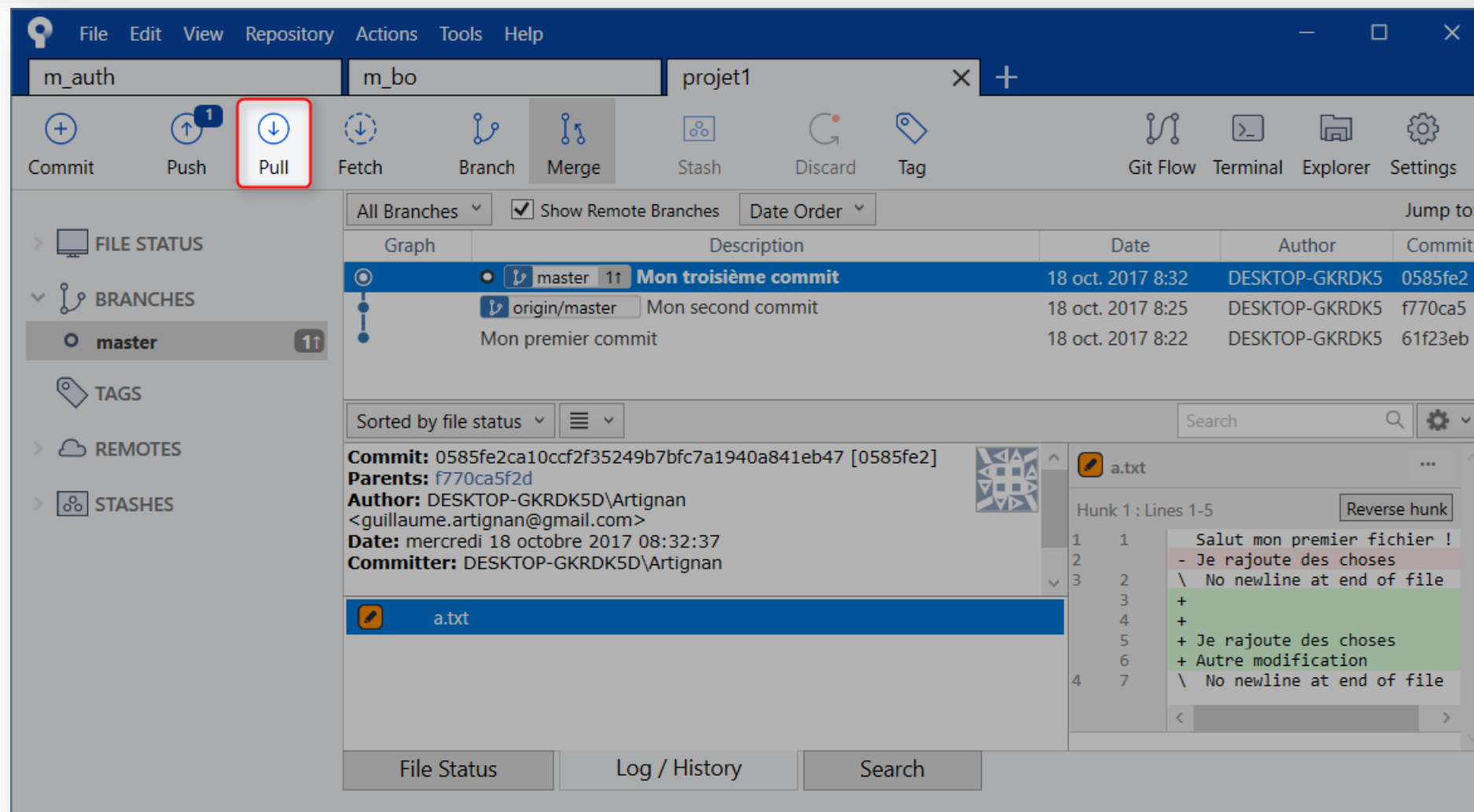
# L'étape du Pull / Récupération

Lorsqu'un commit a déjà été effectué sur le serveur, il est parfois obligatoire de récupérer le travail effectué par les collaborateur.

La fonction Pull va permettre cela.



# L'étape du Pull / Récupération



The screenshot shows the Git GUI application window. The top toolbar contains several icons: Commit, Push, Pull (highlighted with a red box), Fetch, Branch, Merge, Stash, Discard, and Tag. The main area displays the commit history for the 'master' branch. The commit list shows three commits, with the most recent one selected. The commit details for the selected commit (0585fe2) are shown below the list, including the commit message, author, date, and committer. The file status section shows a single file, 'a.txt', which is highlighted. The right pane shows the diff for the selected file, with a hunk of code highlighted in green.

Graph	Description	Date	Author	Commit
●	● master 11 Mon troisième commit	18 oct. 2017 8:32	DESKTOP-GKRDK5	0585fe2
●	● origin/master Mon second commit	18 oct. 2017 8:25	DESKTOP-GKRDK5	f770ca5
●	● Mon premier commit	18 oct. 2017 8:22	DESKTOP-GKRDK5	61f23eb

Sorted by file status

Commit: 0585fe2ca10ccf2f35249b7bfc7a1940a841eb47 [0585fe2]  
Parents: f770ca5f2d  
Author: DESKTOP-GKRDK5\Artignan <guillaume.artignan@gmail.com>  
Date: mercredi 18 octobre 2017 08:32:37  
Committer: DESKTOP-GKRDK5\Artignan

a.txt

Hunk 1 : Lines 1-5

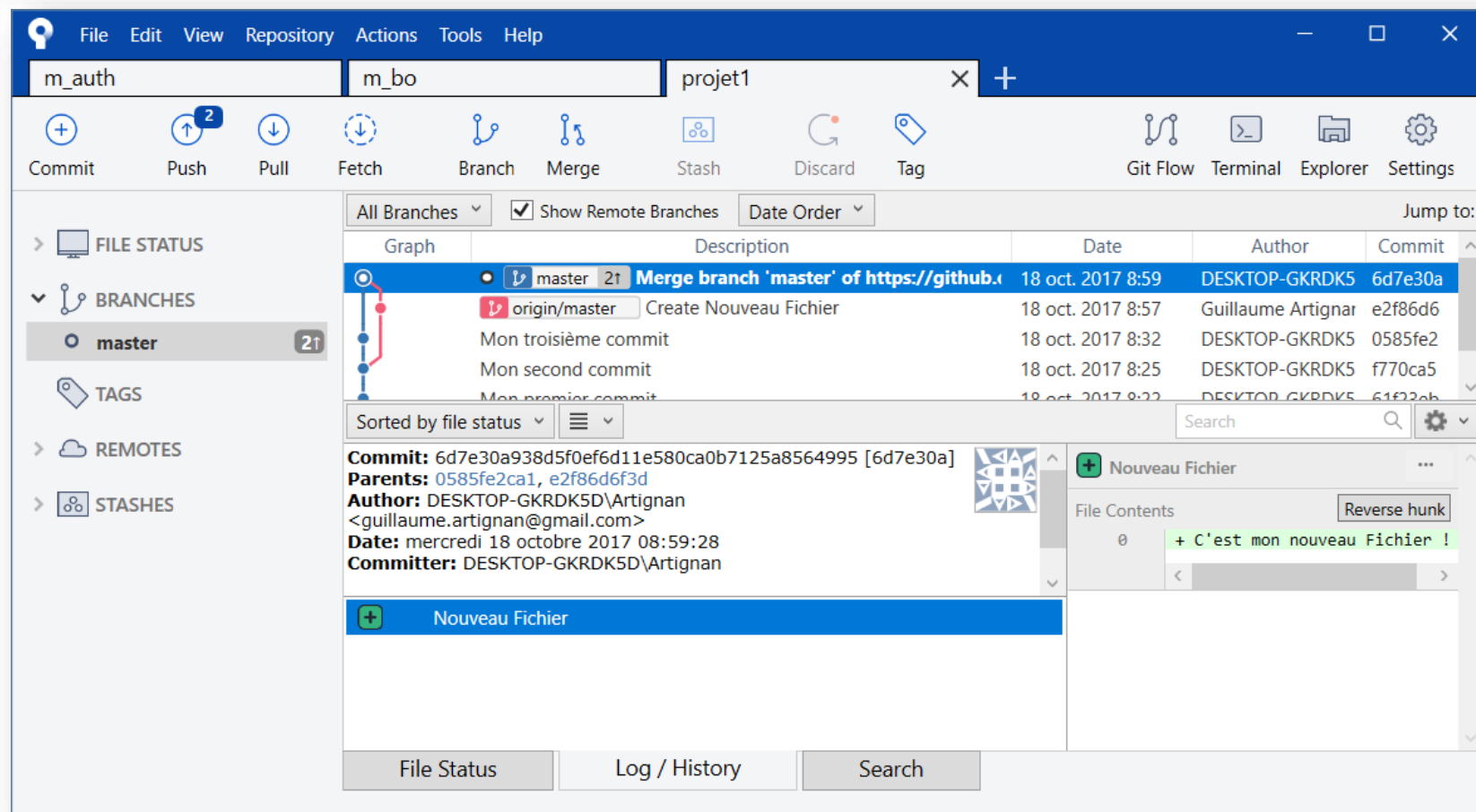
```
1 1 Salut mon premier fichier !  
2 - Je rajoute des choses  
3 \ No newline at end of file  
4  
5 +  
6 + Je rajoute des choses  
7 + Autre modification  
8 \ No newline at end of file
```



# Récupération du code source

Une fois la récupération effectuée.

Un merge est effectué.



# Les Checkout

A tout moment il est possible de revenir sur une version précédente du projet en double cliquant sur un commit du projet.

Nous appelons cela le Checkout.

