

PROJET MODÉLISATION D'UNE BD + SQL SERVEUR



Groupe :

AMROUNE Sofiane

CHALAUX Florian

GALY Pierre

MOREAU Théo

Sommaire

Mise en place du dépôt GitHub	1
Définition et comparatif	2
Utilisation et gestion d'un client Git	3
GLPI	12
Analyse	12
Traitement et configuration de l'application	13
Mise en place du serveur SQL	18
Définition des besoins d'architecture du serveur de base de données	18
Manipulation et paramétrage du serveur de base de données ...	19
Accès et droits	20
Tests	21
Remplissage des données.....	22
Mise en place des stockages	22
Programmation des sauvegardes	23

Mise en place du dépôt GitHub :

1) Définition et comparatif

Aujourd'hui, la taille des projets informatiques est de plus en plus importante. Les équipes de développements comptent, ainsi, de plus en plus de membres. Tout au long de la réalisation des programmes informatiques, de nombreuses personnes "commitent" tous les jours leur code. Avec les systèmes de gestion de source traditionnels tels que CVS ou SVN, il devient de plus en plus compliqué de travailler au quotidien. Les problèmes récurrents sont :

- Les conflits de code lors des commits suite au travail de deux développeurs sur une même portion de code,
- La gestion des différentes versions de code (celle en recette et celle en cours de développement principalement),
- La difficulté de valider l'intégration de son code avec le reste du projet avant le commit.

La cause principale de ces soucis que rencontrent les grosses équipes de développements au quotidien est la centralisation du repository contenant le code source partagé. Pour cette raison, des logiciels de gestion de version distribués ont été créés. Ils gèrent les versions à l'aide d'une architecture des dépôts du code source distribuée. En effet, ces logiciels impliquent que chaque développeur a une copie locale d'un "repository" de référence ou maître avec lequel elle est synchronisée.

À propos de la gestion de version :

Un gestionnaire de version est un système qui enregistre l'évolution d'un fichier ou d'un ensemble de fichiers au cours du temps de manière à ce qu'on puisse rappeler une version antérieure d'un fichier à tout moment.

Il permet de répertorier ces changements et de revenir dessus si besoin est. De plus, il facilite le travail d'équipe en vous avertissant s'il y a des conflits (si deux personnes ont édité un même fichier en même temps).

Voici quelques exemples :

GIT : un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes.

Subversion : un logiciel de versions centralisés. C'est un logiciel distribué sous licence Apache et BSD.

Subversion fonctionne donc sur le mode client-serveur, avec :

Un serveur informatique centralisé et unique où se situent :

- Les fichiers constituant la référence (le « dépôt » ou « référentiel », ou « repository » en anglais),
- Un logiciel serveur Subversion tournant en « tâche de fond » ;
- Des postes clients sur lesquels se trouvent :
- Les fichiers recopiés depuis le serveur, éventuellement modifiés localement depuis,
- un logiciel client, sous forme d'exécutable standalone (ex. : SmartSVN) ou de plug-in (ex. : TortoiseSVN, Eclipse Subversive) permettant la synchronisation, manuelle et/ou automatisée, entre chaque client et le serveur de référence.

Mercurial : un logiciel de gestion de versions décentralisé disponible sur la plupart des systèmes Unix (linux et mac OS X) et Windows. Mercurial est écrit principalement en Python. Il a été créé pour s'utiliser via des lignes de commandes. Toutes les commandes commencent par « hg », en référence au symbole chimique du mercure. Ses principales caractéristiques sont3, entre autres :

- Sa rapidité et sa capacité à gérer les gros projets ;
- Son utilisation sans nécessiter un serveur ;
- Son fonctionnement complètement distribué ;
- Sa robustesse dans la gestion des fichiers ASCII et binaires ;
- Sa gestion avancée des branches et des fusions ;
- Son interface web intégrée.

Comparatif :

Git :

Avantages :

- **Gestion des branches** : possibilité de travailler sur plusieurs projets en parallèle.
Les développeurs peuvent avoir des branches locales, des branches publiques. Les branches locales sont utiles lorsqu'ils travaillent individuellement sur une nouvelle fonctionnalité compliquée qui implique plusieurs fichiers. Pendant leur travail, ils peuvent faire des "pull" et ainsi mettre à jour leur version de développement avec les modifications faites par les collègues. À mesure qu'ils atteignent des objectifs intermédiaires, ils peuvent faire des commit qui resteront locaux. Lorsque la fonctionnalité est terminée, ils peuvent faire un "merge" vers la branche commune de développement (nous on utilise Master), puis "push" et tout le monde aura accès à leur code.
- **L'interface console** :
- **Algorithmes de fusion** : quand un fichier a été modifié par plusieurs personnes en même temps, Git sait s'adapter et choisir un algorithme qui fusionne intelligemment les lignes du fichier qui ont été modifiées. Si par hasard 2 personnes ont modifié en même temps la même ligne (cas rare, mais qui arrive), il y a un conflit et Git laisse des marques dans le fichier pour dire qui a modifié quoi, et vous invite à décider ce que vous gardez.

- **La rapidité** : les mises à jours et données sont fusionnées très rapidement même s'il y a eu de nombreuses modifications.
- **Le Contenu** : contrairement à d'autres logiciels de gestion, Git ne surveille pas les fichiers mais leur contenu.
- **Installation** facile sur toutes les plateformes.

Inconvénients :

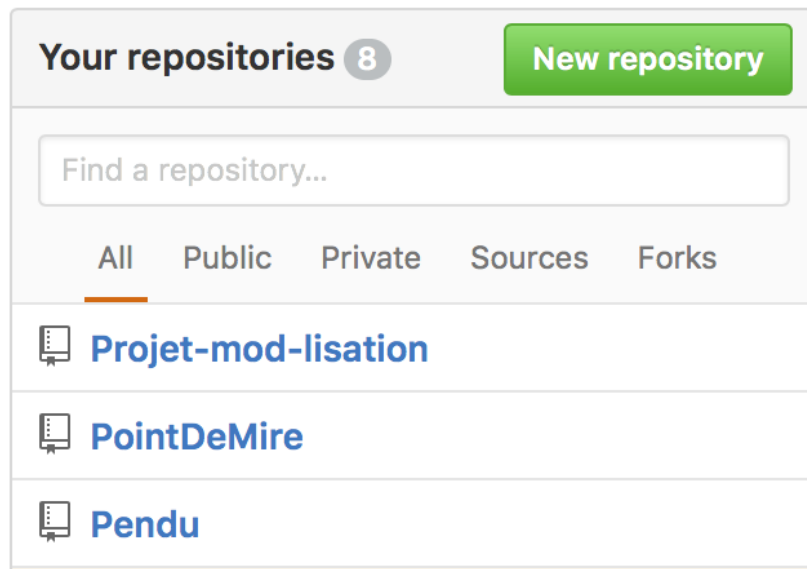
- **La complexité** : Prise en main compliqué et pas facile pour n'importe qui. Logiciel fait par des développeurs pour des développeurs.
- **Le portage sous Windows** : il faut utiliser cygwin. Difficultés de connexions au serveur SSH. Interface console Windows n'est pas trop courante.
- **Très grand nombre de commande** : très complexes et toujours faire attention à ce qu'on fait par peur de perdre ce qu'on a fait.

Avantages :

	Avantages	Inconvénients
GIT	<ul style="list-style-type: none"> - Rapidité - Facilité d'installation - Gestion des branches - Algorithmes de fusion - Interface console - Contenu 	<ul style="list-style-type: none"> - La complexité - Portage sous windows - Retour à la ligne - Très grand nombre de commandes
Mercurial	<ul style="list-style-type: none"> - Facilité d'installation - Multi os 	<ul style="list-style-type: none"> - La complexité - Retour à la ligne
Subversion	<ul style="list-style-type: none"> - Facilité d'utilisation 	<ul style="list-style-type: none"> - Centralisé

2) Utilisation et gestion d'un client Git

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"

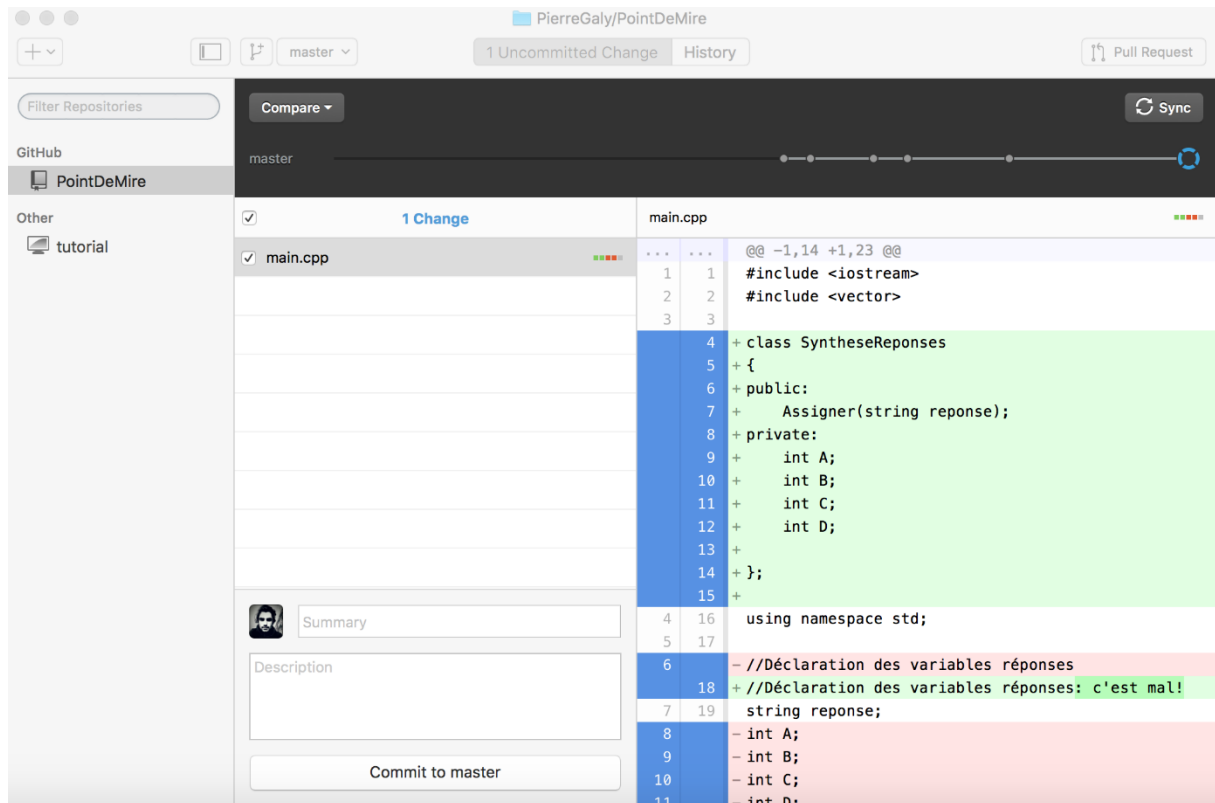
3° Créez un README pour votre dépôt

4° Installer un client git en graphique

Quelques exemples :

- Github desktop
- SourceTree

Github :




5° Client git

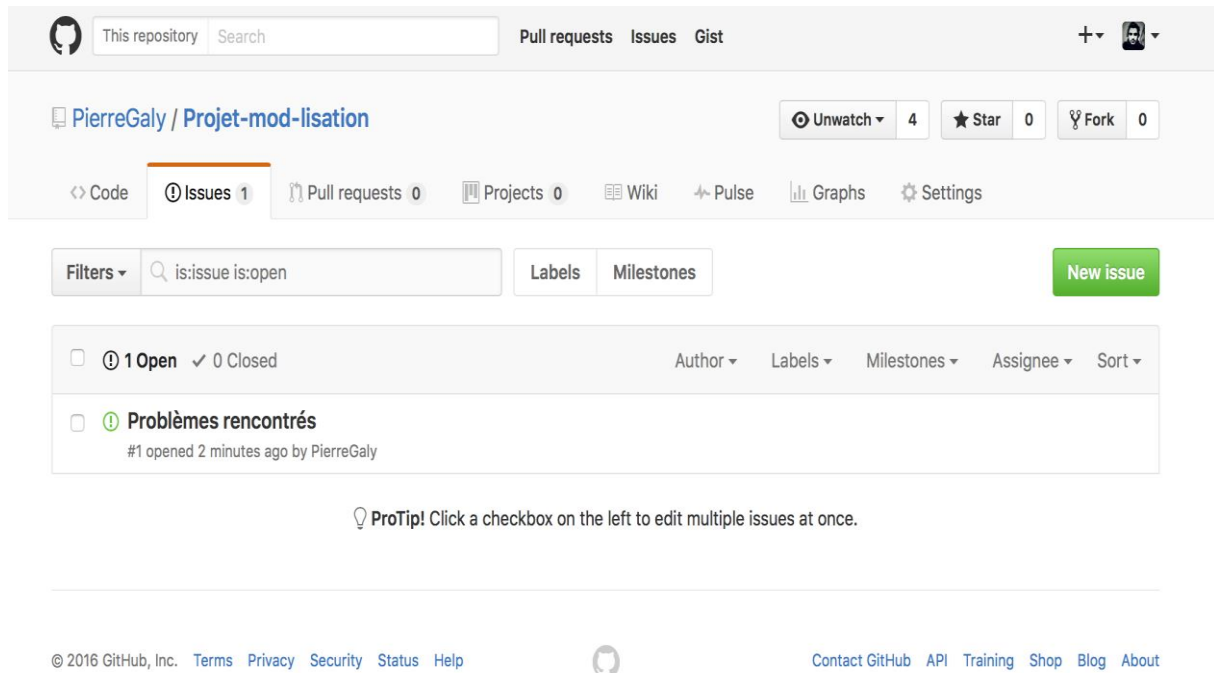
- Sur la gauche, il y a la liste des dépôts que le client suit ; il est possible d'ajouter un dépôt (soit en le clonant, soit en l'attachant localement) en cliquant sur l'icône « + » en haut de la zone.
- Au centre, il y a la zone d'entrée de *commit* qui permet d'entrer un message de validation et de 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 différence 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.

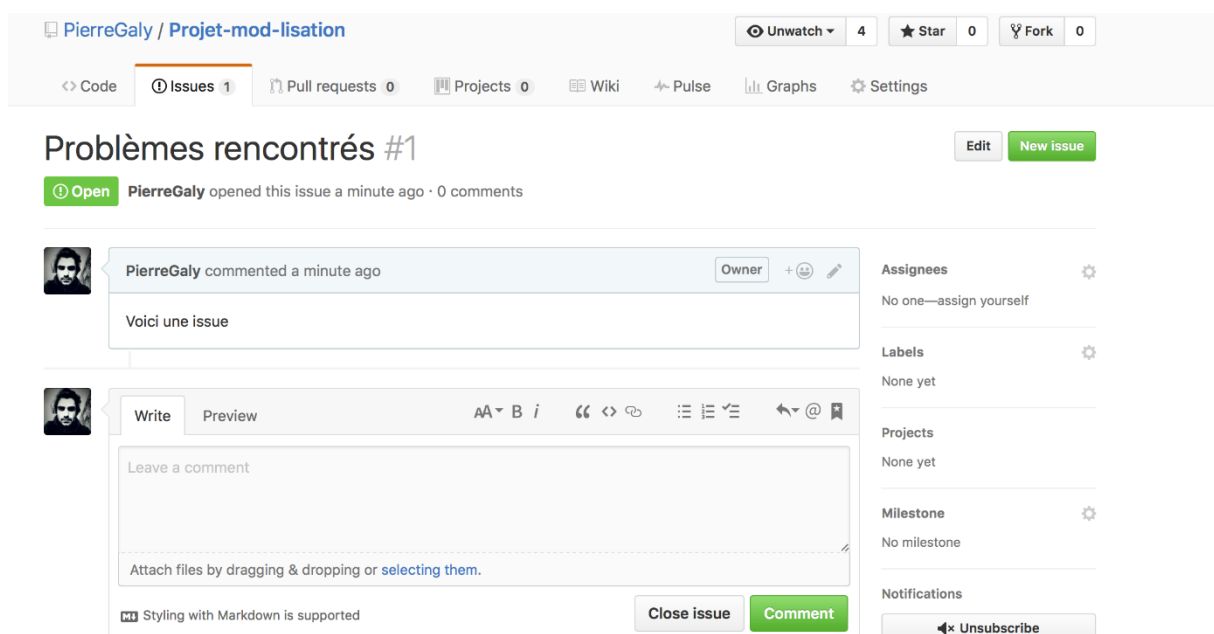
Gestion des issues, des branches, merging de branches :

Les issues :

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



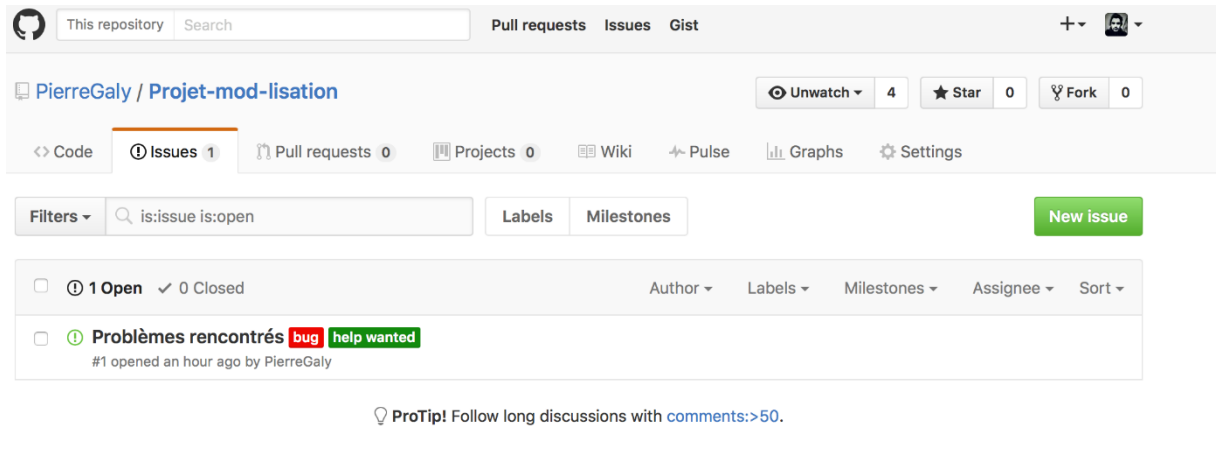
This screenshot shows the GitHub interface for the repository 'PierreGaly / Projet-mod-lisation'. The 'Issues' tab is selected, showing a list of issues. There is one open issue titled 'Problèmes rencontrés' created 2 minutes ago by PierreGaly. The interface includes a search bar, navigation links for Code, Issues, Pull requests, Projects, Wiki, Pulse, Graphs, and Settings. A 'New issue' button is prominently displayed. The footer contains copyright information for GitHub, Inc. and various links for users.



This screenshot shows the detailed view of the issue 'Problèmes rencontrés #1'. The issue is marked as 'Open' and was created by PierreGaly. A comment by PierreGaly is visible, stating 'Voici une issue'. Below the comment is a form to add a new comment, with a 'Comment' button. The right sidebar provides additional information about the issue, including Assignees (No one—assign yourself), Labels (None yet), Projects (None yet), Milestone (No milestone), and Notifications (Unsubscribe).

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 :

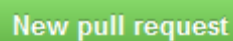
`cd racine_du_depot`

`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 :

A green rectangular button with rounded corners and a white border. The text "New pull request" is written in white, sans-serif font, centered within the button.

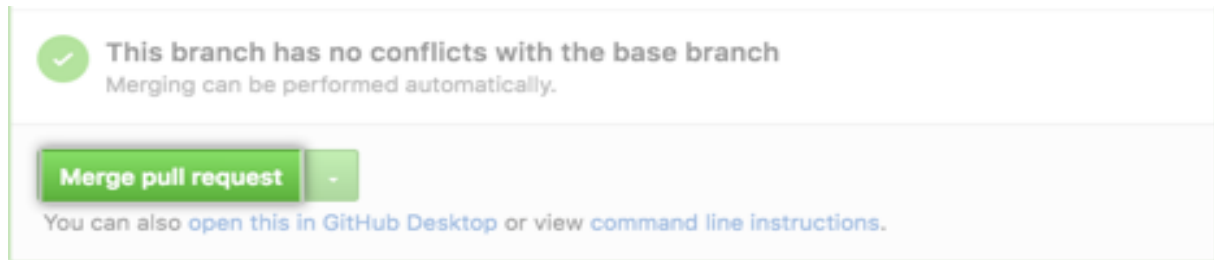
Merging de branches :

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**.

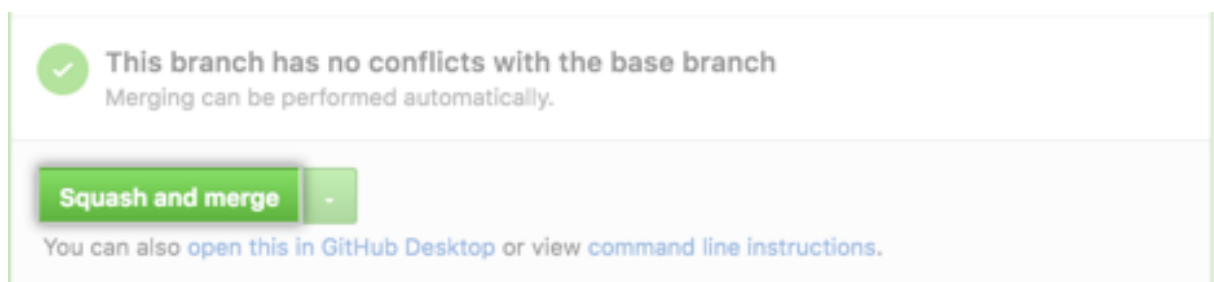


✓ **This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request -

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

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

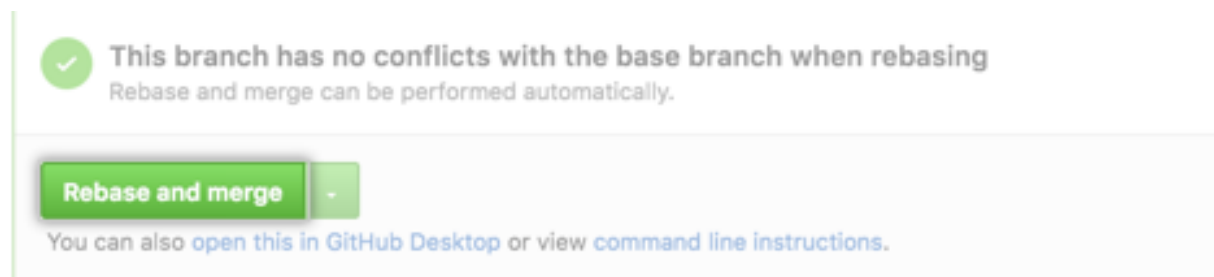


✓ **This branch has no conflicts with the base branch**
Merging can be performed automatically.

Squash and merge -

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

- 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**.



✓ **This branch has no conflicts with the base branch when rebasing**
Rebase and merge can be performed automatically.

Rebase and merge -

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

GLPI :

1) Analyse des besoins et des utilisations de GLPI

GLPI est une solution de Gestion Libre de Parc informatique, elle permet de gérer tous les éléments liés à l'informatique.

GLPI permet la gestion du matériel informatique, la gestion des demandes d'assistance d'utilisateurs ainsi que la gestion des licences.

GLPI est une solution open-source gratuite en application web.

OUAPI : OUAPI est une application web libre pour la gestion de parc informatiques gratuit, mais aussi pour la réalisation de l'inventaire de matériels, de périphériques, de logiciels, etc. Disponible en français, mais disponible uniquement sous windows, il s'agit de la d'un de ses plus gros inconvénient. Le langage de mise en œuvre est PHP,

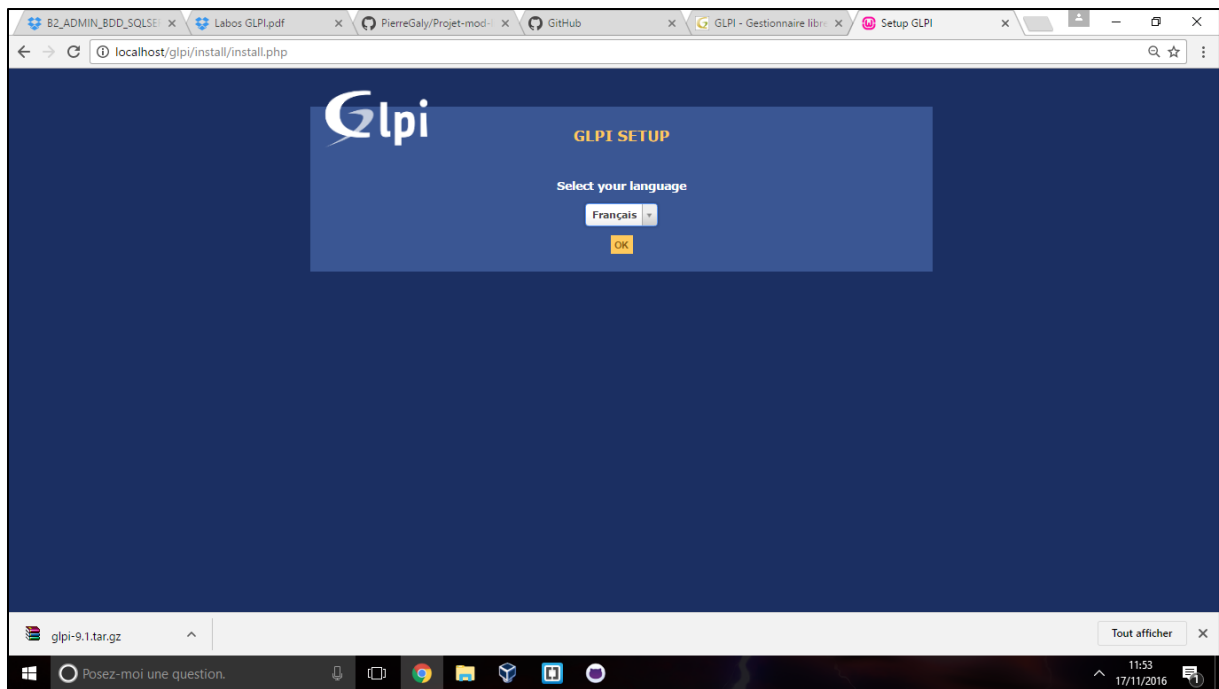
OTRS : OTRS est le premier open-source d'innovation dans les services, il comprend Help Desk (permet aux entreprises d'installer un point de contact privilégié avec ses utilisateurs. Sa mission : aider les clients à résoudre leurs problèmes informatiques). OTRS offre des conseils globaux, de personnalisation et de services de soutien. Disponible sous tous les OS connus il est disponible gratuitement pour un essai d'un mois, puis le prix est fixe à 400euros par mois pour une entreprise de 10 personnes. Le langage mise en œuvre est PER

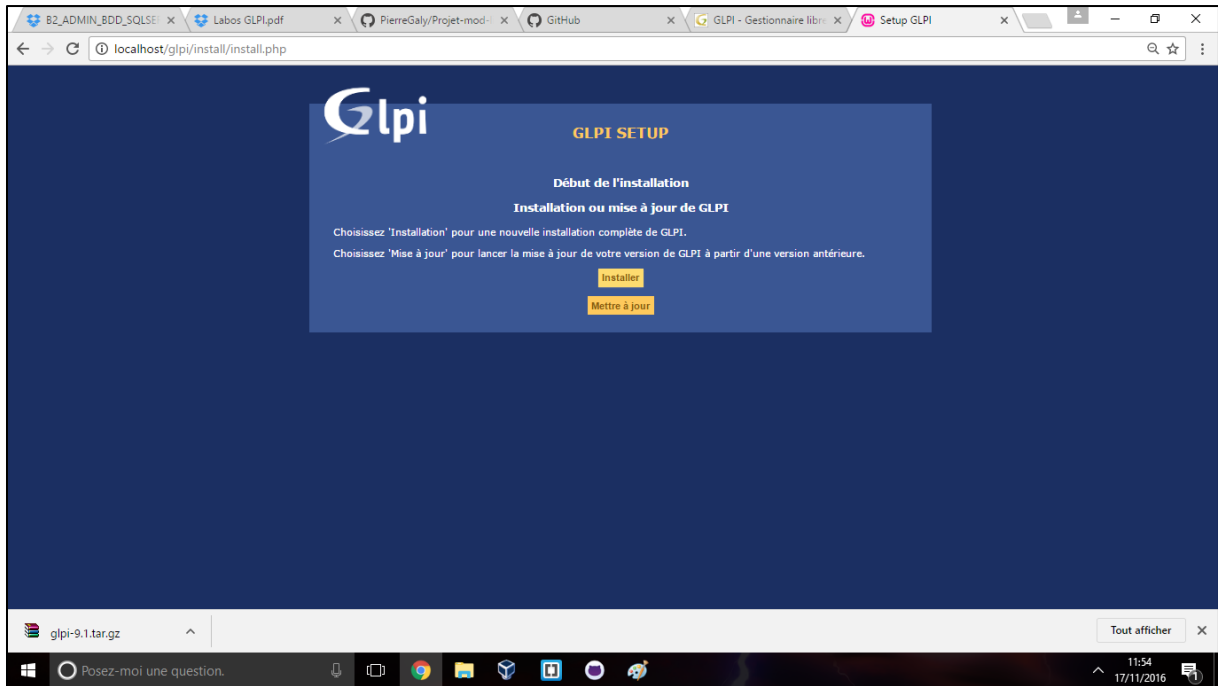
2) Traitement et configuration de l'application :

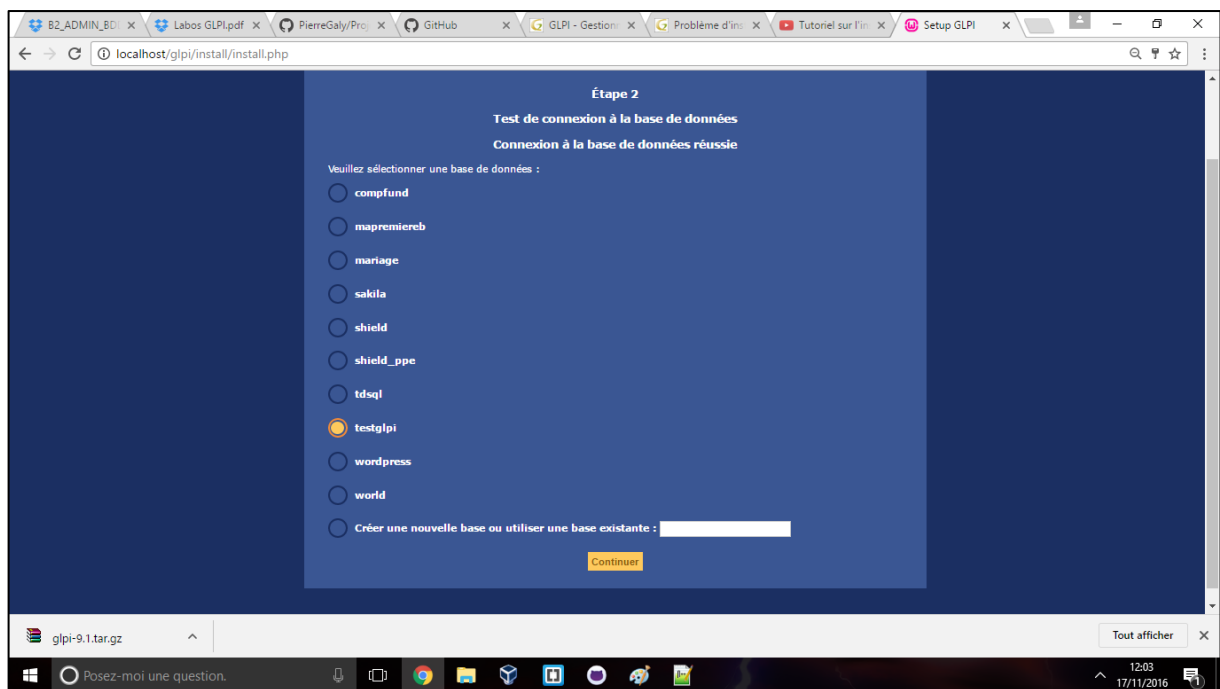
Installation :

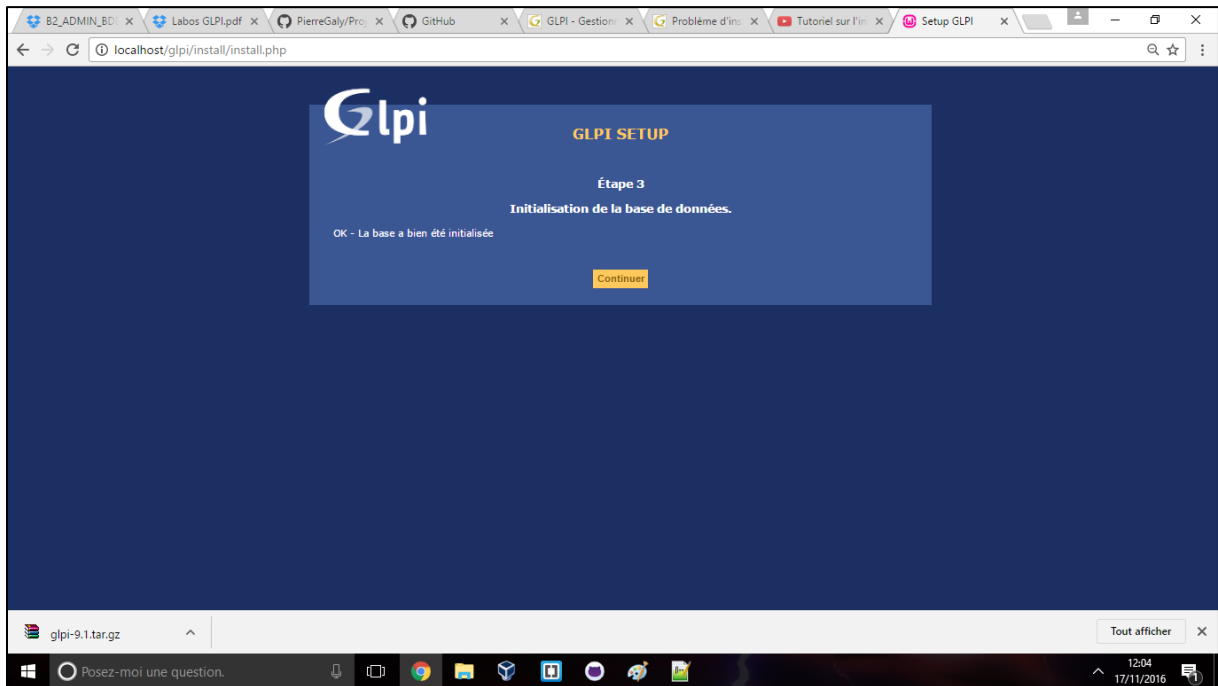
Lien de téléchargement GLPI : <http://glpi-project.org/spip.php?article3>

Installer GLPI sur son serveur WEB









Concurrent de GLPI X achat/vente de m... X Inventaire Inform... X achat/vente de m... X Tutoriel d'installa... X GLPI - Gestionnaire X GLPI - Interface st... X

localhost/glpi/front/central.php#

Rechercher Français ? ⚙️ glpi

Parc Assistance Gestion Outils Administration Configuration

Accueil

Vue personnelle Vue groupe Vue globale Flux RSS Tous

⚠️ Pour des raisons de sécurité, veuillez changer le mot de passe par défaut pour le(s) utilisateur(s) : glpi post-only tech normal

⚠️ Pour des raisons de sécurité, veuillez supprimer le fichier : installinstall.php

Vos tickets en cours (1)

Demandeur	Éléments associés	Description
ID : 1	glpi	Général Problème réseau (0 - 1)

Votre planning

Notes personnelles

Notes publiques

Vos tickets observés (1)

Demandeur	Éléments associés	Description
ID : 1	glpi	Général Problème réseau (0 - 1)

0.152 seconde - 9.42 Mio

GLPI 9.1 Copyright (C) 2015-2016 Teclib - Copyright (C) 2003-2015 INDEPNET Development Team

localhost/glpi/front/central.php#

Posiez-moi une question.

14:42 15/12/2016

Concurrent de GLPI X achat/vente de m... X Inventaire Inform... X achat/vente de m... X Tutoriel d'installa... X GLPI - Gestionnaire X GLPI - Tickets X

localhost/glpi/front/ticket.php

Rechercher Français ? ⚙️ glpi

Parc Assistance Gestion Outils Administration Configuration

Accueil Assistance Tickets

Caractéristiques - Statut est Tous Rechercher

Affichage (nombre d'éléments) 20 Page courante en PDF paysage De 1 à 2 sur 2

Actions

ID	Titre	Statut	Dernière modification	Date d'ouverture	Priorité	Demandeur - Demandeur	Attribué à - Technicien	Catégorie	Temps de résolution
3	Problème réseau	En cours (Attribué)	2016-12-15 15:13	2016-12-15 15:11	Moyenne	glpi	tech		
2	Problème logiciel	Résolu	2016-12-15 15:10	2016-12-15 15:06	Moyenne	glpi	tech		

Actions

Affichage (nombre d'éléments) 20 De 1 à 2 sur 2

0.297 seconde - 25.93 Mio

GLPI 9.1 Copyright (C) 2015-2016 Teclib - Copyright (C) 2003-2015 INDEPNET Development Team

Posiez-moi une question.

15:24 15/12/2016

Mise en place du serveur SQL :

