# TP1 Versioning et sécurité

1. Si vous n’avez pas de compte GitHub créez-vous un compte
2. Créez un repository pour y déposer le code source de votre site web que vous allez créer. Compléter le ReadMe en indiquant la description du projet ainsi que l’environnement nécessaire
3. Installer Git pour Windows **S’IL N’EST PAS INSTALLE SUR LE POSTE**
4. Configurez votre identité :

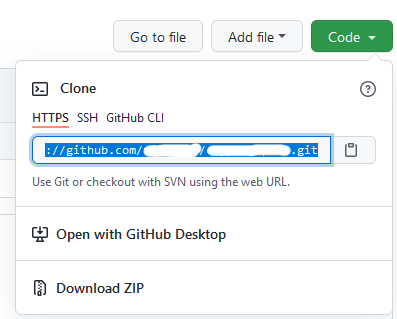
$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

1. Créer un dépôt local avec GIT à l’endroit de votre choix sur le PC. Positionner vous dans le dossier où vous voulez stocker votre projet et tapez :

git init

1. Un dossier caché .git doit avoir été créé dans le dossier choisi
2. Récupérez l’URL du dépôt distant (GitHub)



1. Retournez sur Git et tapez la commande suivante :

git remote add OC https://github.com/SuiteDeVotreLien.git

1. Maintenant vous allez cloner votre dépôt distant vers votre dépôt local avec la commande :

git clone https://github.com/SuiteDeVotreLien.git

Inscription, Authentification et Session

1. Créer dans une nouvelle base de données une table utilisateur. La table utilisateur doit pouvoir contenir :

* Un login
* Un mot de passe
* Une date de création
* Une date d'activation (optionnelle)
* Un Id unique
* Et d'autre champ que vous jugerez nécessaire

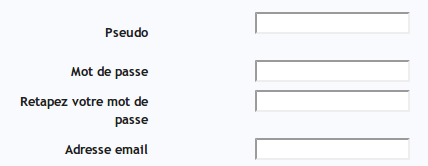
1. Faire un commit de script permettant de générer la BDD
2. Faire un push vers GitHub
3. Vérifiez que le push à bien fonctionné
4. Créer une page web d’authentification qui contient les éléments suivants :

* Lien (S’inscrire) vers la page d’inscription pour les personnes non encore enregistrées.
* Une zone de saisie de login et une zone de saisie de mot de passe pour identifier le  
  visiteur



* Une fois le visiteur est identifié, un message de bienvenue s’affiche avec le nom du  
  visiteur.
* Si le login ne figure pas dans la base, le visiteur est redirigé automatiquement vers la page d’inscription.
* Si le mot de passe saisi est incorrect un message « mot de passe incorrect » s’affiche en rouge

1. Créer une nouvelle branche dans GIT et faire un commit (fichier PHP + le fichier CSS ainsi que l’arborescence)
2. Faire un push vers GitHub
3. Vérifiez que le push à bien fonctionné
4. Créer une page permettant de confirmer que l’utilisateur est bien connecté avec succès
5. Faire un commit sur la branche précédemment créée
6. Fusionnez la branche avec la branche master
7. Faire un push vers GitHub
8. Créer une page d’inscription pour les utilisateurs non encore enregistrés.



* Vérifier que les champs mot de passe, et confirmez mot de passe sont bien  
  identiques.
* Après la validation le visiteur est redirigé vers la page d’authentification.

1. Mettre à jour le projet web dans GitHub
2. Une fois le visiteur est identifié, il sera redirigé vers la page d’accueil. Après la déconnexion le visiteur est redirigé vers la page d’authentification et toute tentative de retourner vers la page d’accueil sans une authentification sera négligée.

**Crypter les mots de passe dans une base de données**  
 Par sécurité, une bonne pratique qui consiste à crypter les mots de passe stockés dans une table de votre base de données. Ainsi, dans le cas où une personne malveillante arriverait à consulter votre table, elle ne pourrait pas voir le mot de passe, mais une suite de caractères dépourvue de sens. Pour hacher un mot de passe, il existe plusieurs fonctions qui se basent sur des algorithmes différents. La bonne fonction à utiliser est  password\_hash, qui va choisir pour vous le meilleur algorithme.

Le hachage :

Le hachage sert principalement à la sécurisation de mots de passe. Lorsque l'utilisateur s'inscrit, vous ne stockez pas son mot de passe mais le hash de celui-ci. Ensuite, lorsqu'il souhaite se connecter, vous faites un hash du mot de passe soumis, et vérifiez qu'il correspond bien à celui enregistré. Ainsi, si un pirate a accès à votre base de données, il ne pourra pas obtenir les mots de passe de vos utilisateurs, qui sont hashés, donc inexploitables.

La particularité du hachage est qu'il fonctionne dans un seul sens : il est impossible de retrouver le mot de passe d'origine une fois qu'il a été haché. De plus, un *hash* (nom donné à la version hachée du mot de passe) est unique.

sha1  ,  md5  ,  sha256  ,  sha512  ... ne sont plus considérées comme des fonctions de hachage sûres aujourd'hui. Même si vous les voyez dans certains (vieux) codes, **ne les utilisez pas**.

Vous pouvez tester de hacher un mot de passe en MD5 ou sha1 et aller sur le site <https://crackstation.net/> pour tenter de le décoder

La bonne fonction à utiliser est  password\_hash, qui va choisir pour vous le meilleur algorithme.

Le salage :

Le salage permet de lutter contre les utilisateurs qui ont souvent tendance à utiliser des mots du langage courant comme mot de passe, ou des prénoms, etc.

L'idée est de concaténer une chaîne de caractère connue au mot de passe, avant de le hasher. L'intérêt est simple : l'introduction d'une chaîne de salage changera tous les hash. Cela veut dire que le pirate doit recalculer ses dictionnaires en incluant votre chaîne de salage, ce qui prend du temps et limite l'intérêt du dictionnaire (rainbow tables).

Pour plus d'efficacité, on utilise le salage dynamique. Celui-ci consiste à déterminer une chaîne de salage différente pour chaque utilisateur. L'intérêt : ici, deux mots de passe identique de deux utilisateurs différents n'auront pas le même hash, car ils ont un salage différent. Ainsi, le pirate doit recalculer tous ses dictionnaires pour chaque utilisateur ce qui empêche tout casse massif des mots de passe que vous stockez. On utilise par exemple le nom d'utilisateur ou l'email de l'utilisateur comme suffixe et/ou comme préfixe.

1. Modifier la page d’inscription pour rendre le mot de passe crypté (utilisation d'un sel dynamique de votre choix en plus de celui présent dans l’algorithme de cryptage).
2. Modifier la page d’authentification pour prendre en charge les modifications effectuées.
3. Mettre en place une gestion des cookies pour permettre à l’utilisateur de rester connecter
4. Mettre à jour le projet web dans GitHub

Pour aller plus loin :

<https://analyse-innovation-solution.fr/publication/fr/hacking/injection-sql-sqli-dorks#h25sksd0zp511qi7p7cionarx1kf41xz>

Tester chez vous si le formulaire est vulnérable. Essayer de corriger la vulnérabilité