|  |
| --- |
| Versioning et gestion de versions décentralisé |

**BC03.** Élaborer, intégrer et optimiser les architectures de systèmes d'information et garantir l'industrialisation du développement

**Volume en jours :** 4 jours

**Volume en heures :** 28 heures

# Compétences liées :

**C21.** Assurer la gestion et le suivi des révisions de code source à l’aide d'un logiciel de gestion de versions comme Git afin de garantir la traçabilité des changements et la coordination des versions au sein de l’équipe de développement

**C22.** Définir et appliquer des procédures de fusion de code et de résolution de conflits à l'aide de plateformes telles que GitHub ou GitLab pour faciliter le travail collaboratif, la révision de code et l'intégration continue (CI).

# Objectifs :

Comprendre les concepts fondamentaux et l'importance du versionning dans le développement logiciel. Expliquer le fonctionnement et les avantages des systèmes de versionning décentralisés, ainsi que les stratégies de gestion associées. Maîtriser l'utilisation avancée du versionning dans des environnements collaboratifs.

# Pré-requis :

Connaissance de base en développement logiciel et compréhension des concepts de gestion de projets. Expérience préalable avec des outils de versionnage simple ou centralisé.

# Plan détaillé du module :

## **Chapitre 1 : Concepts de base du versionning**

* Définition du versionning
* Importance du versionning dans le développement logiciel
* Les différents types de versionning
* Les principaux outils de versionning
* Bonnes pratiques en matière de versionning

## **Chapitre 2 : Systèmes de versionning décentralisés**

* Fonctionnement d'un système de versionning décentralisé
* Avantages et inconvénients des systèmes de versionning décentralisés
* Exemples de systèmes de versionning décentralisés (Git, Mercurial)
* Utilisation avancée des fonctionnalités de versionning décentralisé
* Stratégies de gestion de branches et de fusions

## **Chapitre 3 : Utilisation avancée du versionning**

* Gestion des conflits lors de la fusion de branches
* Utilisation des tags pour marquer des versions stables
* Intégration continue et déploiement continu avec le versionning
* Bonnes pratiques pour la gestion des branches et des versions
* Utilisation de hooks pour automatiser des tâches liées au versionning

## **Chapitre 4 : Gestion des versions dans un environnement collaboratif**

* Stratégies de travail collaboratif avec le versionning
* Gestion des droits d'accès aux dépôts de versionning
* Utilisation de branches partagées pour des fonctionnalités en cours de développement
* Utilisation de sous-modules pour gérer des dépendances entre différents projets
* Bonnes pratiques pour la contribution et la revue de code en utilisant le versionning

# Exercices du module :

Exercice 1 :

Titre : Création d'un dépôt Git et premiers commits

Énoncé : Créez un nouveau dépôt Git sur votre ordinateur et ajoutez-y un fichier README.md. Ensuite, effectuez trois commits successifs en modifiant à chaque fois le contenu du fichier. Assurez-vous d'indiquer un message clair et descriptif pour chaque commit.

Exercice 2 :

Titre : Gestion avancée des branches avec Git

Énoncé : Créez une nouvelle branche dans votre dépôt Git et apportez-y des modifications. En parallèle, effectuez des modifications sur la branche principale (master). Ensuite, fusionnez la branche que vous avez créée avec la branche principale en gérant les éventuels conflits. Assurez-vous de documenter les étapes que vous avez suivies.

Exercice 3 :

Titre : Utilisation des tags pour marquer des versions stables

Énoncé : Identifiez une version stable de votre projet et créez un tag pour la marquer. Ensuite, effectuez des modifications dans le code et créez une nouvelle version stable. Créez un nouveau tag pour marquer cette version et expliquez la différence entre les deux tags.

Exercice 4 :

Titre : Travail collaboratif avec Git

Énoncé : Travaillez avec un ou plusieurs collègues pour simuler un environnement de développement collaboratif. Créez des branches partagées pour travailler sur des fonctionnalités spécifiques et effectuez des fusions régulières pour vous assurer de l'intégration continue du code. Documentez les étapes de collaboration et les actions effectuées.

Exercice 5 :

Titre : Intégration de hooks pour automatiser des tâches avec Git

Énoncé : Créez un hook pre-commit dans votre dépôt Git pour effectuer une vérification automatique du code avant chaque commit. Paramétrez ce hook pour qu'il vérifie la présence de certaines erreurs de syntaxe ou de formatage. Effectuez des commits en introduisant délibérément des erreurs pour observer le fonctionnement du hook. Documentez les résultats obtenus et les actions correctives éventuellement nécessaires.

# Bibliographie :

Mercurial : 'Git par la pratique' de Sylvain Peyronnet, 'La gestion de configuration avec Git' de Bénédicte Alloing, 'Versioning et gestion des sources' de Rémy Bégin

# Méthodes pédagogiques :

Utilisation de supports (ppt, etc.), Autre