Méthode Agile

La méthode Agile est une approche de gestion de projet qui favorise la collaboration, l'adaptation aux changements et les livraisons fréquentes de fonctionnalités. Elle repose sur des cycles courts appelés itérations ou sprints, pendant lesquels une partie du projet est développée, testée et livrée. L'objectif est d'assurer une satisfaction continue des parties prenantes grâce à un produit évolutif.

Gestion de projet

La gestion de projet est l'ensemble des techniques, outils et méthodologies utilisées pour planifier, organiser et exécuter un projet afin d'atteindre des objectifs précis. Elle comprend :

- **La définition des objectifs.**
- **La gestion des ressources (humaines, financières, matérielles).**
- **Le suivi et l'évaluation des progrès:** Des outils comme les diagrammes de Gantt ou des méthodologies comme Agile ou Waterfall sont souvent utilisés.

Kanban :

- Basé sur la visualisation du flux de travail via un tableau (*to-do*, *in progress*, *done*).
- Encourage une amélioration continue et limite le travail en cours (*WIP*).

Scrum est une méthodologie agile qui aide les équipes à développer des produits de manière itérative et incrémentale. Elle repose sur trois piliers : **la transparence, l'inspection et l'adaptation**, et favorise la collaboration, la flexibilité et l'amélioration continue.

Les éléments clés de Scrum :

Équipe Scrum :

- ➔ **Product Owner** : Définit les priorités et maximise la valeur du produit.
- ➔ **Scrum Master** : Facilite le processus et supprime les obstacles.
- ➔ **Équipe de Développement** : Conçoit, développe et livre le produit.

Événements Scrum :

- ➔ **Sprint** : Cycle de développement court (1 à 4 semaines) pour livrer une version utilisable.
- ➔ **Sprint Planning** : Planification des tâches à réaliser dans un sprint.
- ➔ **Daily Scrum** : Réunion quotidienne pour synchroniser l'équipe.
- ➔ **Sprint Review** : Démonstration du travail accompli à la fin du sprint.
- ➔ **Sprint Rétrospective** : Analyse pour améliorer les processus.

Artefacts Scrum :

- ➔ **Product Backlog** : Liste priorisée des fonctionnalités à développer.
- ➔ **Sprint Backlog** : Liste des tâches à accomplir pendant un sprint.
- ➔ **Increment** : Résultat concret et utilisable du sprint.

-> Scrum encourage une approche adaptative où les équipes s'ajustent rapidement aux changements pour maximiser la valeur livrée au client.

- **Transparence** : Toutes les parties prenantes doivent avoir une vision claire des processus et des progrès.
- **Inspection** : Des revues régulières permettent d'évaluer le travail réalisé.
- **Adaptation** :Les ajustements sont faits en fonction des résultats des inspections pour s'adapter aux imprévus

Burn-Down Chart

Le burn-down chart est un graphique qui visualise la progression du sprint en suivant les tâches restantes à accomplir. **Axe X** : Le temps (jours du sprint). **Axe Y** : Le travail restant (nombre de tâches, heures, points d'effort).

Pourquoi utiliser Scrum ?

- Favorise la flexibilité et l'adaptabilité dans des projets complexes.
- Encourager la collaboration entre les parties prenantes.
- Garantit des livraisons fréquentes et des produits de qualité.
- Permet de répondre rapidement aux changements de priorités ou aux imprévus.

La POO est un paradigme de programmation basé sur le concept d'**objets**, qui combinent des données (propriétés) et des comportements (méthodes). Les concepts clés incluent . C'est un outil puissant pour concevoir des logiciels complexes, rendant le code plus clair, modulaire et maintenable.

Principes fondamentaux de la POO :

Encapsulation :

- Les données (attributs) et les fonctions (méthodes) sont regroupées dans une classe, offrant une meilleure modularité.
- Les détails internes sont cachés (protection des données via des niveaux d'accès comme private ou public).

Héritage :

- Une classe peut hériter des attributs et des méthodes d'une autre classe (classe parent).
- Permet la réutilisation de code et facilite la maintenance.

Polymorphisme :

- La même méthode peut se comporter différemment selon le contexte (par exemple, une méthode avec le même nom dans plusieurs classes).
- On distingue deux types :
- Polymorphisme statique (surcharge de méthode).
- Polymorphisme dynamique (redéfinition de la méthode).

Abstraction :

- Se concentre sur les détails essentiels tout en masquant les implémentations complexes.
- Réalisée à l'aide de **classes abstraites** ou d'**interfaces**.

Concepts clés :

- **Classe :** Une structure définissant un plan pour créer des objets (exemple : une classe `Voiture`).
- **Objet :** Une instance concrète d'une classe (exemple : une voiture spécifique).
- **Attributs :** Les propriétés d'un objet (exemple : couleur, modèle).
- **Méthodes :** Les actions ou comportements qu'un objet peut réaliser (exemple : démarrer, freiner).

Avantages de la POO :

- **Réutilisation du code :** Grâce à l'héritage et aux classes modulaires.
- **Facilité de maintenance :** Les modifications sont localisées à des classes spécifiques.
- **Modélisation intuitive :** Proche de la manière dont nous percevons le monde réel.
- **Extensibilité :** Facilite l'ajout de nouvelles fonctionnalités.

API (Application Programming Interface) jouent un rôle essentiel dans le développement moderne en facilitant la communication et l'échange de données entre applications et services.

C'est un ensemble de règles et de protocoles qui permettent à différents logiciels de communiquer entre eux. Elle agit comme une interface, facilitant l'interaction entre une application et d'autres systèmes (services, bases de données, matériels).

Rôles principaux d'une API :

1. **Faciliter l'intégration :**

Les API permettent à différents systèmes de fonctionner ensemble sans dépendre directement de leur structure interne.

2. **Automatisation :**

Les applications peuvent exécuter des actions automatiquement en accédant aux fonctionnalités offertes par une API.

3. **Réutilisation de fonctionnalités :**

Une API expose des fonctionnalités réutilisables, évitant de recréer des fonctions déjà existantes.

Caractéristiques clés :

REST (Representational State Transfer) :

- Basée sur les méthodes HTTP (GET, POST, PUT, DELETE).
- Simple, légère et largement utilisée.

SOAP (Simple Object Access Protocol) :

- Plus complexe, basé sur XML, souvent utilisé pour des intégrations critiques.

GraphQL :

- Offrir des requêtes plus flexibles permet de récupérer uniquement les données nécessaires.

Avantages des API :

- **Interopérabilité :** Connecte des technologies hétérogènes.
- **Économie de temps :** Accès à des services ou données déjà disponibles.
- **Évolutivité :** Les systèmes peuvent évoluer indépendamment grâce à des interfaces standardisées.

`req` **(Request)** : Il représente l'objet de la requête envoyée par le client au serveur. Cet objet contient toutes les informations envoyées par le client lors de l'appel à l'API, telles que :(req.query,req.params,req.body).

`res` **(Response)** : Il représente l'objet de la **réponse** que le serveur va envoyer au client après avoir traité la requête. L'objet `res` permet de définir :

-Le **code de statut** HTTP (par exemple, 200 pour "OK", 404 pour "Not Found", etc.),

-Les **données à renvoyer** au client (souvent avec `res.send()` ou `res.json()`),

-Les **en-têtes** de la réponse, qui peuvent fournir des informations supplémentaires comme le type de contenu ou des informations de cache.

UML est un langage de modélisation graphique utilisé pour représenter les différents aspects d'un système logiciel (architecture, comportement, structure). Il est largement adopté dans le génie logiciel.

Caractéristiques principales :

- Centré sur **l'orienté objet**.
- Utilisé pour analyser, concevoir et documenter des systèmes.
- Basé sur plusieurs types de diagrammes : structurels, comportementaux, interactionnels.

Exemples de diagrammes UML :

1. **Diagramme de classes :** Représente les classes et leurs relations.
2. **Diagramme de cas d'utilisation :** Montre les interactions entre utilisateurs et système.
3. **Diagramme de séquence :** Décrit l'ordre des interactions entre objets.
4. **Diagramme d'activités :** Montre les processus et flux de contrôle.

Avantages :

- Visuel et intuitif pour les équipes techniques.
- Normalisé et adaptable à différents types de projets.
- Utile pour les systèmes complexes.

Merise est une méthode de modélisation utilisée principalement pour les bases de données et les systèmes d'information. Elle est structurée autour de plusieurs niveaux de description.

Modèle Conceptuel de Données (**MCD**) : Représente les entités, relations et attributs.

Modèle Conceptuel de Traitement (**MCT**) : Décrit les traitements indépendamment des technologies. Modèle Physique de Données (**MPD**) : Adaptation aux contraintes d'un SGBD (index, partitions).

-> **UML** est idéal pour les projets orientés objet et pour décrire tous les aspects d'un logiciel.=> **Merise** est mieux adapté pour concevoir et gérer des bases de données et des systèmes d'information structurés.

AJAX (Asynchronous JavaScript and XML). C'est une méthode utilisée pour créer des applications web interactives et dynamiques. Elle permet de communiquer avec un serveur web de manière asynchrone, ce qui signifie que des données peuvent être envoyées et reçues sans recharger la page web entière. Cela améliore l'expérience utilisateur en rendant les applications plus rapides et fluides.

JSON (JavaScript Object Notation). C'est un format léger d'échange de données, facile à lire, et simple à analyser et à générer pour les machines. Il est largement utilisé pour transmettre des données entre un client et un serveur, notamment dans les applications web.

Syntaxe simple : Basé sur des paires clé-valeur, similaire à la notation des objets en JavaScript.

Léger : Conçu pour être minimaliste, ce qui le rend rapide à traiter.

JWT est un moyen simple et sécurisé d'authentifier et d'autoriser les utilisateurs dans des applications web modernes, en permettant de stocker des informations sur le client sans avoir besoin de session côté serveur.

Utilisation :

Authentification : Lorsqu'un utilisateur se connecte, un serveur génère un JWT contenant des informations d'identification et le renvoie au client. Ce jeton est ensuite envoyé avec chaque demande subséquente pour prouver l'identité de l'utilisateur.

Autorisation : Le serveur vérifie que le jeton est valide, que les informations sont correctes et que l'utilisateur a le droit d'effectuer l'action demandée.

Structure

Header : Contient des informations sur le type de jeton (généralement "JWT") et l'algorithme de signature utilisé (par exemple, HMAC SHA256 ou RSA).

Payload : Contient les informations ou les "claims". Les claims sont des données sur l'utilisateur ou l'application, telles que l'identifiant de l'utilisateur, son rôle, la durée de validité du jeton, etc.

Signature : Permet de vérifier que le jeton n'a pas été modifié. Elle est générée en prenant le header et le payload, puis en les signant avec une clé secrète ou une clé publique/privée, selon l'algorithme.

Les tests unitaires sont plus ciblés, tandis que les tests d'intégration vérifient le bon fonctionnement global du système. Les deux types de tests sont complémentaires et utilisés ensemble pour garantir la qualité du code.

La virtualisation permet de séparer les ressources physiques d'un système des environnements logiques qui y sont exécutés. En d'autres termes, elle permet de simuler plusieurs systèmes ou ressources sur une seule machine physique.

=>Virtualisation de serveur : Diviser un serveur physique en plusieurs serveurs virtuels (machines virtuelles ou VM). Chaque VM peut exécuter son propre système d'exploitation.

Asynchrone = Non bloquant, continue l'exécution en attendant la fin d'une tâche.

Synchrone = Bloquant, attend la fin de chaque tâche avant de passer à la suivante.

Threads = Permet l'exécution parallèle de plusieurs tâches dans un programme.

=>

Asynchrone : Se dit d'un processus qui s'exécute indépendamment de l'appelant, permettant à d'autres tâches de continuer pendant que l'opération principale se termine. Exemple : une requête HTTP qui ne bloque pas l'exécution du programme.

Synchrone : Un processus où les tâches sont exécutées dans un ordre séquentiel, bloquant l'exécution du programme jusqu'à la fin de l'opération. Exemple : un appel de fonction qui attend la réponse avant de continuer.

Threads : Ce sont des unités d'exécution légères dans un processus. Un programme peut utiliser plusieurs threads pour effectuer plusieurs tâches en parallèle, ce qui permet d'améliorer la performance, notamment dans les applications multithreadées.

Un microservice est une petite application autonome qui fait une tâche spécifique au sein d'une architecture d'application plus large. Chaque microservice représente une fonction ou un service spécifique de l'application, et chaque service est autonome, avec sa propre logique métier, base de données et API.

Exemple :

Dans une application de commerce électronique, un microservice pourrait gérer les utilisateurs, un autre pourrait gérer les paiements, un autre les commandes, et un autre encore le suivi des livraisons.

Indépendance : Chaque microservice peut être développé, déployé, mis à jour et scalé indépendamment des autres.

Modularité : Chaque microservice gère une tâche ou un domaine spécifique, ce qui facilite la maintenance et l'évolution de l'application.

Communication : Les microservices communiquent entre eux via des API, souvent en utilisant des protocoles comme HTTP/REST ou des messages (par exemple, RabbitMQ, Kafka).

Scalabilité : Il est plus facile de scaler un microservice spécifique sans affecter l'ensemble de l'application.

SQL est adapté aux applications nécessitant une **structure rigide**, une gestion fiable des données, et des transactions complexes.

NoSQL est plus flexible et évolutif, utilisé pour des données **non structurées ou semi-structurées**, et pour des applications nécessitant une **haute disponibilité** et une **scalabilité horizontale**.

Clé-Valeur : Stocke les données sous forme de paires clé-valeur. Exemple : Redis, DynamoDB.

Document : Stocke les données sous forme de documents (souvent JSON). Exemple : MongoDB, CouchDB.

Colonne : Organiser les données en colonnes au lieu de lignes. Exemple : Cassandra, HBase.

Graphe : Utilise des graphes pour les relations entre les données. Exemple : Neo4j.

=>

Modèle relationnel : Les données sont organisées en tables avec des lignes et des colonnes.

Schéma fixe : Les tables doivent avoir une structure bien définie (schéma) avant d'y insérer des données.

Transactions ACID : Garantit la fiabilité des transactions avec les propriétés **ACID** (Atomicité, Cohérence, Isolation, Durabilité).

Langage déclaratif : L'utilisateur spécifie **ce qu'il veut**, pas **comment le faire** (ex. : `SELECT`, `INSERT`, `UPDATE`, `DELETE`).

Exemples de bases de données SQL : MySQL, PostgreSQL, Oracle, SQL Server.

Git est un système de gestion de versions qui permet de suivre les modifications dans un projet, de collaborer avec d'autres développeurs, et de gérer plusieurs versions de code.

GitHub est une plateforme de développement collaboratif qui héberge des dépôts Git, permettant de partager et de collaborer sur des projets de manière efficace.

=>Les commandes Git permettent de gérer les branches, les commits, et de collaborer avec des dépôts distants sur des plateformes comme GitHub.

I/CD signifie **Intégration Continue (CI)** et **Déploiement Continu (CD)**. Ce sont des pratiques utilisées pour rendre le développement et la mise en production du code plus rapides et fiables.

CI - Intégration Continue :

- **But** : Automatiser l'intégration du code des développeurs dans un projet commun.
- **Comment** : Chaque fois qu'un développeur ajoute du code, il est automatiquement testé pour vérifier qu'il fonctionne bien et qu'il n'y a pas de bugs.
- **Avantage** : Cela permet de détecter les erreurs rapidement et d'éviter des conflits entre les différentes versions du code.

CD - Déploiement Continu :

Il y a deux types :

1. **Continuous Delivery (Livraison Continue)** : Le code validé est prêt à être mis en production à tout moment, mais il faut une action manuelle pour le déployer en production.
2. **Continuous Deployment (Déploiement Continu Automatisé)** : Le code validé est automatiquement déployé en production, sans intervention humaine.

En résumé :

- **CI** permet de vérifier que chaque modification de code fonctionne et ne casse rien.
- **CD** permet de rendre ce code rapidement disponible pour être mis en production, de manière automatique ou manuelle.

Avantages :

- **Livraison plus rapide** : Le code arrive plus vite en production.
- **Moins d'erreurs** : Les tests automatiques détectent rapidement les problèmes.
- **Processus plus fiable** : Grâce à l'automatisation, on réduit les risques d'erreurs humaines.