Concepts de Conception Logicielle (MVC-MLD-CDC-UML)

1. MVC (Modèle-Vue-Contrôleur)

Objectif

Le design pattern MVC est utilisé pour séparer les préoccupations dans une application. Cela permet de rendre le code plus modulaire, testable et maintenable.

Composants

- **Modèle** : Gère les données de l'application et la logique métier. Il interagit avec la base de données et représente les objets que l'application manipule.
- **Vue** : Représente l'interface utilisateur de l'application. Elle affiche les données du modèle et envoie les actions de l'utilisateur au contrôleur.
- **Contrôleur** : Intercepte les actions de l'utilisateur, traite les entrées (par exemple, les formulaires), interagit avec le modèle pour récupérer ou modifier des données, et met à jour la vue.

Exemple de Code

```
// Exemple d'implémentation simple en Node.js

// Modèle (Model)
class User {
  constructor(name) {
    this.name = name;
  }
}

// Vue (View)
class UserView {
  render(user) {
    console.log(`User Name: ${user.name}`);
  }
}

// Contrôleur (Controller)
class UserController {
  constructor(user, view) {
    this.user = user;
    this.view = view;
  }

  setUserName(name) {
    this.user.name = name;
    this.view.render(this.user); // Met à jour la vue avec le nouveau nom
  }
```


created with craft

```
}

// Utilisation
const user = new User("Alice");
const userView = new UserView();
const userController = new UserController(user, userView);

userController.setUserName("Bob"); // Affiche "User Name: Bob"
```

2. MCD (Modèle Conceptuel de Données)

Objectif

Le MCD est une représentation abstraite des données d'une application qui décrit les entités et leurs relations. Il est utilisé pour la conception de bases de données et constitue la première étape avant de passer au MLD (Modèle Logique de Données).

Caractéristiques

- **Entités** : Correspondent aux objets ou concepts (par exemple, Utilisateur, Produit).
- **Attributs** : Propriétés des entités (par exemple, nom, prix).
- **Relations** : Connexions entre les entités, souvent décrites par des cardinalités (par exemple, un Utilisateur peut avoir plusieurs Produits).

Exemple

```
Entités :
1. Utilisateur
   - id (PK)
   - nom
   - email

2. Produit
   - id (PK)
   - nom
   - prix

Relations :
- Un Utilisateur peut acheter plusieurs Produits (1,N)
- Un Produit peut être acheté par plusieurs Utilisateurs (N,N)
```

3. MLD (Modèle Logique de Données)

Objectif

created with craft

Le MLD est une représentation abstraite des données d'une application, indépendamment de la façon dont ces données seront physiquement stockées. Il sert de pont entre le modèle conceptuel et le modèle physique.

Caractéristiques

- **Entités** : Correspondent à des objets ou des concepts dans l'application (par exemple, Utilisateur, Produit).
- **Attributs** : Propriétés ou caractéristiques des entités (par exemple, nom, prix).
- **Relations** : Connexions entre les entités (par exemple, un Utilisateur peut avoir plusieurs Produits).

Exemple

```
Utilisateur
- id (PK)
- nom
- email

Produit
- id (PK)
- nom
- prix

Relation : Un Utilisateur peut acheter plusieurs Produits
```

4. CDC (Cahier des Charges)

Objectif

Le cahier des charges est un document qui définit les besoins, les exigences et les contraintes d'un projet. Il sert de guide tout au long du développement.

Contenu

- **Introduction** : Présentation du projet et de son contexte.
- **Objectifs** : Ce que le projet doit accomplir.
- **Fonctionnalités** : Détails des fonctionnalités attendues (ex. authentification, gestion des utilisateurs).
- **Contraintes** : Limitations techniques, réglementaires ou de temps.

Exemple

created with craft

```
Cahier des Charges pour une Application de Gestion de Produits

1. Introduction
   - Ce projet consiste à développer une application web pour gérer les produits
d'un e-commerce.

2. Objectifs
   - Permettre aux utilisateurs de créer, lire, mettre à jour et supprimer
des produits.

3. Fonctionnalités
   - Authentification des utilisateurs.
   - Gestion des produits (CRUD).
   - Recherche de produits par catégorie.

4. Contraintes
   - L'application doit être développée en utilisant React et Node.js.
   - Déploiement prévu sur AWS.
```

5. UML (Langage de Modélisation Unifié)

Objectif

UML est un langage de modélisation standardisé utilisé pour spécifier, visualiser, construire et documenter les artefacts d'un système.

Diagrammes UML

- **Diagramme de classes** : Montre la structure statique du système en représentant les classes, leurs attributs et leurs relations.
- **Diagramme de séquence** : Montre l'interaction entre les objets en représentant les messages échangés.
- **Diagramme d'activités** : Modélise le flux de travail ou les processus d'affaires.

Exemple de Diagramme de Classes

```
+------------------+
|      Utilisateur |
+------------------+
| - id: int        |
| - nom: string    |
| - email: string  |
+------------------+
| + créer()        |
| + lire()         |
| + mettreÀJour()  |
| + supprimer()    |
```


created with craft

```
+------------------+
```

Conclusion

Cette fiche couvre les concepts de MVC, MCD, MLD, CDC, et UML, qui sont tous essentiels pour la conception et le développement d'applications logicielles. Chacun de ces concepts joue un rôle crucial dans la planification, la modélisation et l'organisation des projets, rendant le développement plus efficace et structuré.

created with craft