**Compte Rendu Général (Semaine du 10/06/2024 au 14/06/2024)**

### **Ce que j’ai vu / appris :**

#### **Métiers :**

1. **Sébastien** :
   * Expérience de 20 ans chez Inetum.
   * Support informatique dans une autre entreprise avant Inetum.
   * Formateur (consultant) et spécialiste en installation tech chez Inetum.
   * A travaillé dans le service d’intégration (développement requis).
   * Promu responsable de service : gestion des commandes, détermination de l’équipe et du matériel, évaluation des délais, accompagnement des clients, résolution de litiges.
2. **Balme Martin** :
   * Développeur full-stack.
   * Stage de 6 mois, alternance de 2 ans (depuis 2020) via le DIIAGE.
   * Salarié depuis 2022 chez Inetum.
   * Missions transverses (sécurité, recherche, API, cartographie).
   * Améliorations et corrections d'anomalies sur le projet GREA.
3. **Richard Duchanoy** :
   * Chef de projet, responsable du service développement, développeur.
   * Parcours académique en BTS, IUT, licence en informatique, master en base de données & IA.
   * Instructeur urbanisme : vérification des dossiers, accompagnement des usagers, rédaction des décisions.

#### **Notions/définitions :**

* **Types valeur/référence nullable** : Variables ou classes pouvant être nulles (même les booléens) caractérisées par un “?”.
* **Classe partielle** : Fractionnement d’une classe pour optimiser son développement et faciliter la collaboration.
* **POO** : Programmation orientée objet avec héritage et polymorphisme.
* **Encapsulation** : Cachage du fonctionnement interne d’un composant pour simplifier l’utilisation via une interface.
* **Classe Abstraite/Interface** : Classe abstraite pour classes similaires, interface pour classes différentes adoptant un contrat commun.
* **Injection de dépendance** : Utilisation des interfaces pour faciliter les tests unitaires et réduire le couplage fort.
* **Région (C#)** : Utilisées pour dissimuler des parties de code non conformes ou non commentées, à utiliser avec précaution.
* **MVC**: est un pattern architectural utilisé pour organiser le code des applications en trois composants principaux : **Model** (Modèle), **View** (Vue) et **Controller** (Contrôleur). Ce modèle de conception permet de séparer les préoccupations, facilitant ainsi la maintenance et l'évolution du code.

### **Ce que j’ai fait :**

1. **Exercice “afficher la liste des clients”** :
   * Utilisation de Razor et @foreach pour afficher les valeurs du modèle dans une vue cshtml.
2. **Exercice “afficher la liste des versions et ajouter une version”** :
   * Réalisation d’une requête POST avec ASP .NET pour enregistrer une nouvelle version (float).
3. **Exercice “effacer une version de la liste des versions”** :
   * Création d’un bouton “supprimer” dans la page index html.
   * Développement de l’action HttpPost DeleteVersion pour vérifier et supprimer une version de la BDD.
   * Création d’un message de confirmation.
4. **Injection de dépendance** :
   * Modification des fichiers dans les dossiers Admin de Cart@ds (Mise à jour des controllers, injection de dépendance sur les classes “Service”).
5. **Documentation et formation** :
   * Poursuite de la documentation sur LINQ, les injections de dépendance, et l’inversion de contrôle.
6. **Réunion avec Christelle** :
   * Compréhension du métier d’instructeur urbanisme.

### **Progrès réalisés :**

* Amélioration de la compréhension des concepts avancés de C# et ASP .NET.
* Réalisation de fonctionnalités avancées (CRUD) dans un environnement ASP .NET.
* Participation active aux réunions et à la compréhension des métiers connexes (instructeur urbanisme).

### **Ressources :**

* [Types valeur Nullable - C# reference | Microsoft Learn](https://learn.microsoft.com/fr-fr/dotnet/csharp/language-reference/builtin-types/nullable-value-types)
* [Types références Nullables - C# reference | Microsoft Learn](https://learn.microsoft.com/fr-fr/dotnet/csharp/language-reference/builtin-types/nullable-reference-types)
* [SelectMany | Linq Examples in C#](https://docs.microsoft.com/fr-fr/dotnet/csharp/programming-guide/concepts/linq/linq-and-strings)
* [Faut-il utiliser les interfaces ou les classes abstraites ? (et comment choisir entre les deux) (dotnetdojo.com)](https://www.dotnetdojo.com/)