

# **Journal de médiation**



Table des matières

1	Analyse préliminaire .....	3
1.1	Introduction .....	3
1.2	WiseJ.net.....	3
1.2.1	Caractéristiques principales.....	3
1.2.2	Avantages.....	3
1.2.3	Utilisation typique.....	3
1.3	Outils Utilisé.....	4
1.4	Objectifs.....	4
1.5	Planification initiale .....	5
2	Analyse préliminaire .....	6
2.1	Introduction.....	6
2.2	Critères d'ergonomie (Bastien et Scapin) .....	6
2.3	Cryptographie .....	7
2.4	Stratégie de test.....	8
2.5	Risques techniques .....	8
2.6	Planification .....	8
2.7	Dossier de conception .....	8
2.7.1	Maquette.....	8
2.7.2	MCD .....	13
2.7.3	MLD .....	14
3	Réalisation.....	14
3.1	Dossier de réalisation .....	14
3.1.1	Liste des fichiers .....	14
3.1.2	Matériel.....	16
3.1.3	Librairies .....	16
3.2	Description des tests effectués.....	16
3.2.1	Test Fonctionnels .....	16
3.3	Erreurs restantes .....	17
3.4	Liste des documents fournis .....	17
4	Conclusions .....	17
5	Annexes.....	19
5.1	Résumé du rapport du TPI / version succincte de la documentation .....	19
5.2	Sources – Bibliographie.....	19
5.3	Journal de travail .....	19
5.4	Manuel d'Installation .....	19
5.5	Manuel d'Utilisation.....	19
5.6	Archives du projet.....	19

## 1 Analyse préliminaire

### 1.1 Introduction

Le projet consiste à développer une application Web basé sur WiseJ.net qui permet aux médiateurs de tenir leur journal de médiation de manière simplifiée et centralisée. Cette application met l'accent sur la facilité d'accès notamment par l'utilisation sur différents appareils (smartphone, tablette, ordinateur portable, ordinateur fixe).

Ce projet est destiné aux médiateurs du canton qui souhaitent abandonner leur vieux tableur Excel, avec la possibilité de l'utiliser n'importe où et sur n'importe quelle plateforme.

### 1.2 WiseJ.net

WiseJ.net est un Framework de développement web basé sur .net, conçu pour permettre aux développeurs de créer rapidement des application web interactives, responsive et complexe en utilisant du C# sans avoir besoin de maîtriser HTML, CSS ou JavaScript.

#### 1.2.1 **Caractéristiques principales**

- Développement en C# (ou VB.net)
- Interface graphique similaire à WindowsForm
- Cross-platform (fonctionne sur tous les navigateurs modernes)
- Intégration facile avec MySQL
- Responsive natif pour PC, tablette et mobiles

#### 1.2.2 **Avantages**

- Aucune connaissance approfondie du web nécessaire
- Très rapide à prendre en main pour les développeurs WinForm
- Parfait pour moderniser des applications desktop vers le web
- Beaucoup d'extensions disponible en package NuGet

#### 1.2.3 **Utilisation typique**

- Application métier (ERP, CRM, outil de gestion interne)
- Dashboard interactif
- Logiciel de saisie ou de consultation de données
- Portail utilisateur/admins

Wisej.NET est un framework .NET moderne qui permet de créer des applications web dynamiques avec la même simplicité que des applications desktop, en C#, tout en offrant un rendu fluide, responsive et professionnel dans un navigateur. Il combine le meilleur du monde WinForms avec la puissance du web.

---

### 1.3 Outils Utilisé

Pour mener à bien ce projet, j'aurai besoin de plusieurs outils :

Application	Utilisation
Visual studio 2022	IDE de développement .net
Github desktop	Logiciel desktop pour pousser les différentes versions du projet
Github	Site Web qui héberge les différentes versions du projet
IceScrum	Gestion de projet
SwissCenter	Site Web qui héberge toutes les composantes de l'application (web, BD, mail)
Azure	Site web qui héberge l'application pour la phase de test

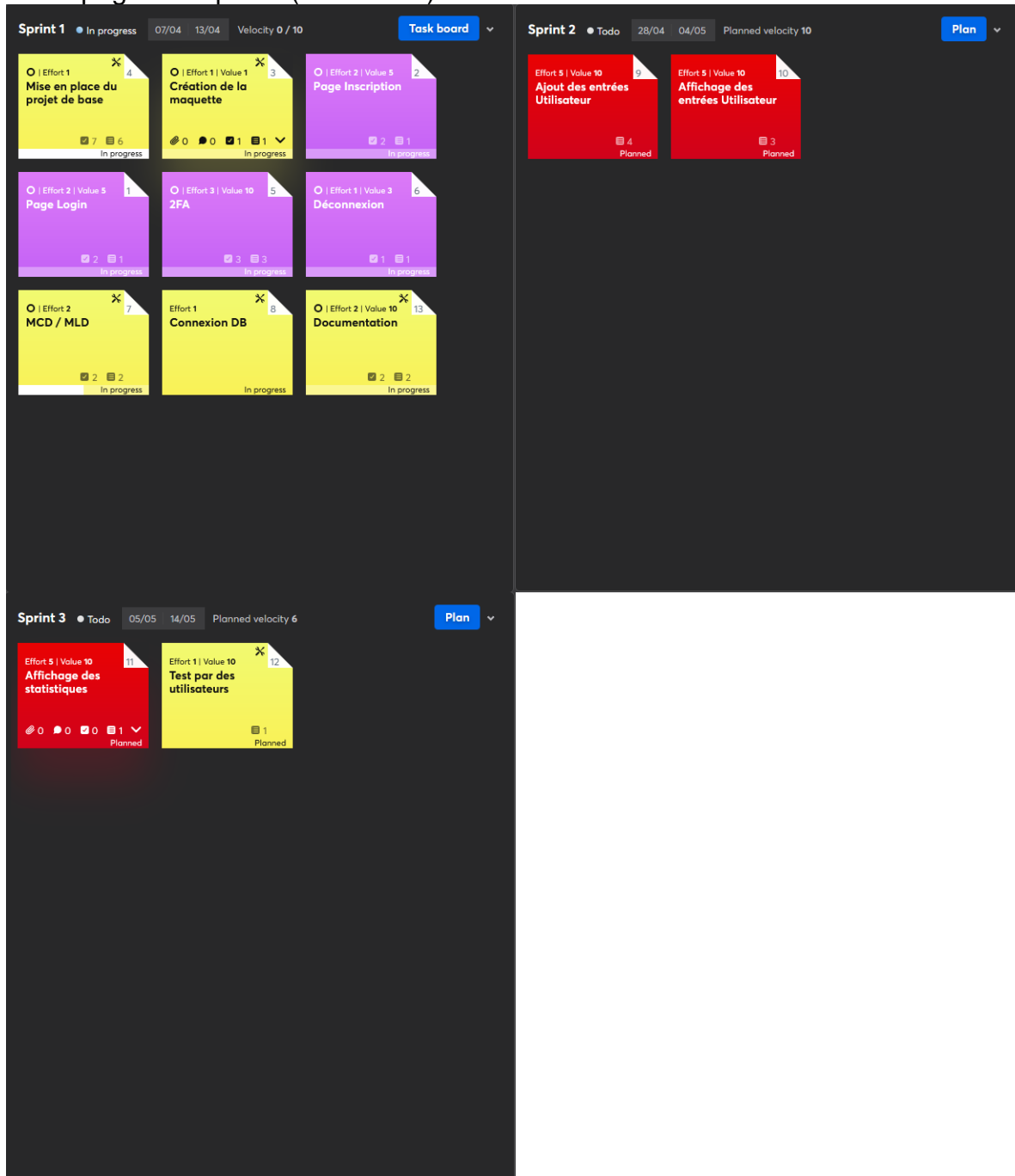
### 1.4 Objectifs

Les objectifs du projet consistent au développement d'une application web qui permet d'entrer des données de rendez-vous des médiateurs. Voici la liste des objectifs défini par le cahier des charges.

- Les données doivent être cryptée
- L'ergonomie de l'application doivent suivre les critères de Bastien et Scapin
- L'application permet de s'enregistrer et de se connecter
- L'application permet d'entrer et de sauver des données
- L'application permet d'afficher des statistiques sur la répartition du temps de travail
- Les cas critiques sont traité comme des entrée erronées (ex :Type de champ)
- L'application est utilisable sur smartphone
- Les données stockées ne sont disponible qu'à l'utilisateur qui les a insérées

## 1.5 Planification initiale

J'ai décidé d'utiliser la méthode agile pour réaliser la planification de ce projet. Voici le découpage des sprints (1 semaine).



Voici les différents sprint goal pour chaque sprint :

Sprint 1 : L'application doit pouvoir gérer l'inscription, la connexion ainsi que toute la partie authentification à 2 facteurs. La base du projet doit être mise en place (IceScrum, repo Git, SwissCenter, base du projet C# avec les implémentations du 1er sprint).

Sprint 2 : L'application doit gérer l'ajout d'entrées utilisateur ainsi que l'affichage de ces entrées dans une liste. Les entrées utilisateur ne doivent être vue uniquement par l'utilisateur concerné.

Sprint 3 : L'application doit pouvoir afficher des statistiques globales des entrées de l'utilisateur. Uniquement les données de cet utilisateur doivent être affichée dans les statistiques.

Durant toute la phase de développement du projet, je vais documenter mon avancement à travers ce document, ainsi qu'à travers mon journal de travail.

## **2 Analyse préliminaire**

### **2.1 Introduction**

Dans cette partie, je vais aborder les différentes analyses et recherches qui m'ont amené à la réalisation du projet. J'ai notamment dû me référer à des critères d'ergonomie (Bastien et Scapin)

### **2.2 Critères d'ergonomie (Bastien et Scapin)**

Les critères d'ergonomiques de Bastien et Scapin sont une référence en ergonomie des interface homme-machine. Ils ont été définis en 1993 et sont encore aujourd'hui largement utilisé pour évaluer ou concevoir des interfaces utilisateur efficace et confortables.

Ils se composent en 8 critères principaux avec leurs exemples respectifs :

- Guidage : L'interface doit guider l'utilisateur dans ce qu'il peut ou doit faire
  - o Menus clairs, Boutons explicites, Messages d'aide
- Charge de travail : l'interface doit minimiser les efforts cognitifs et physique
  - o Remplissage automatique, Groupement logique des champs
- Contrôle explicite : L'utilisateur doit garder le contrôle sur ses actions
  - o Confirmation avant suppression, Choix clairs et sans actions cachées
- Adaptabilité : L'interface doit s'adapter aux besoins et préférences des utilisateurs
  - o Mode sombre / clair, Interface responsive (mobile / PC)
- Gestion des erreurs : L'interface doit éviter les erreurs et aider à les corriger facilement
  - o Messages d'erreur clairs, validation de format (email, code)
- Homogénéité / Cohérence : L'interface doit être cohérente dans tout le système
  - o Même design pour tous les boutons, navigation similaire sur tout le site
- Signifiante des Codes et Dénomination : Les icônes, couleurs, labels doivent être compréhensible immédiatement
  - o Poubelle = supprimer, Disquette = enregistrer, Aucun jargon technique pour l'utilisateur

- Compatibilité : L'interface doit respecter les habitudes et attentes de l'utilisateur
  - o Raccourcis clavier connus (Ctrl + S), Position du bouton « OK » à droite

## 2.3 Cryptographie

Le cryptage des données est fondamental de nos jours pour sécuriser des accès à des données sensibles. La cryptographie est la science qui permet de protéger les informations en les rendant illisible pour tout personne non autorisée. Elle est essentielle en informatique pour garantir la confidentialité, l'intégrité, l'authenticité et parfois la non-répudiation des données.

Les 4 grands objectifs :

- Confidentialité : Seuls les destinataires autorisés peuvent lire l'information
- Intégrité : L'information n'a pas été modifiée durant le transport ou le stockage
- Authentification : On peut confirmer l'identité d'un utilisateur ou d'une source
- Non-répudiation : L'auteur d'un message ne peut pas nier l'avoir envoyé

Types de cryptographie :

- Cryptographie symétrique :
  - o Même clé pour chiffrer et déchiffrer
  - o Rapide, mais nécessite de partager la clé secrètement
  - o Exemples : AES, DES
- Cryptographie asymétrique
  - o Une clé publique pour chiffrer
  - o Une clé privée pour déchiffrer
  - o Très utilisé pour l'échange sécurisé de données et des signatures numériques
  - o Exemples : RSA, ECC
- Fonction de hachage
  - o Transforme une donnée en empreinte fixe
  - o Fonction irréversible
  - o Utile pour stocker des mots de passe, vérifier l'intégrité
  - o Exemple : SHA-256, SHA-3, bcrypt

Quelques concepts clé :

- Chiffrement : Rendre un message illisible sans la clé
- Déchiffrement : Rendre un message lisible avec la clé
- Hachage : Résumer une donnée en empreinte unique
- Signature numérique : Garantir l'identité et l'intégrité d'un message
- Sel : Valeur aléatoire ajouté à un mot de passe avant le hachage
- Clé publique / privée : System de clés utilisées pour chiffrer ou signer

Dans mon application, le hachage des mots de passe avec l'ajout de sel sera utilisé pour chiffrer les mots de passes ainsi que l'utilisation de certificats pour le https.

## **2.4 Stratégie de test**

Concernant ma stratégie de test, j'ai décidé de la faire en 2 parties. La première, en testant via mes tests d'acceptation sur IceScrum (sur PC et sur tablette). Puis la seconde en envoyant l'accès à l'application à différents utilisateur choisis pour me faire un retour concret sur le UI, les bugs ainsi que l'utilisation globale de l'application. Pour les testeurs de l'application, je vais leur fournir un accès libre pour permettre le test de toutes les fonctionnalités, un guide d'utilisation ainsi qu'un google form pour permettre les retours de bug ou améliorations.

## **2.5 Risques techniques**

Les risques techniques liés à mon application est plus du niveau du déploiement. En effet, c'est la première application que je déploie sur un serveur et non juste la faire juste tourner en local. Je vais pour contrer cela me baser sur la documentation Wisej.net ainsi que la documentation .net Core de Microsoft.

## **2.6 Planification**

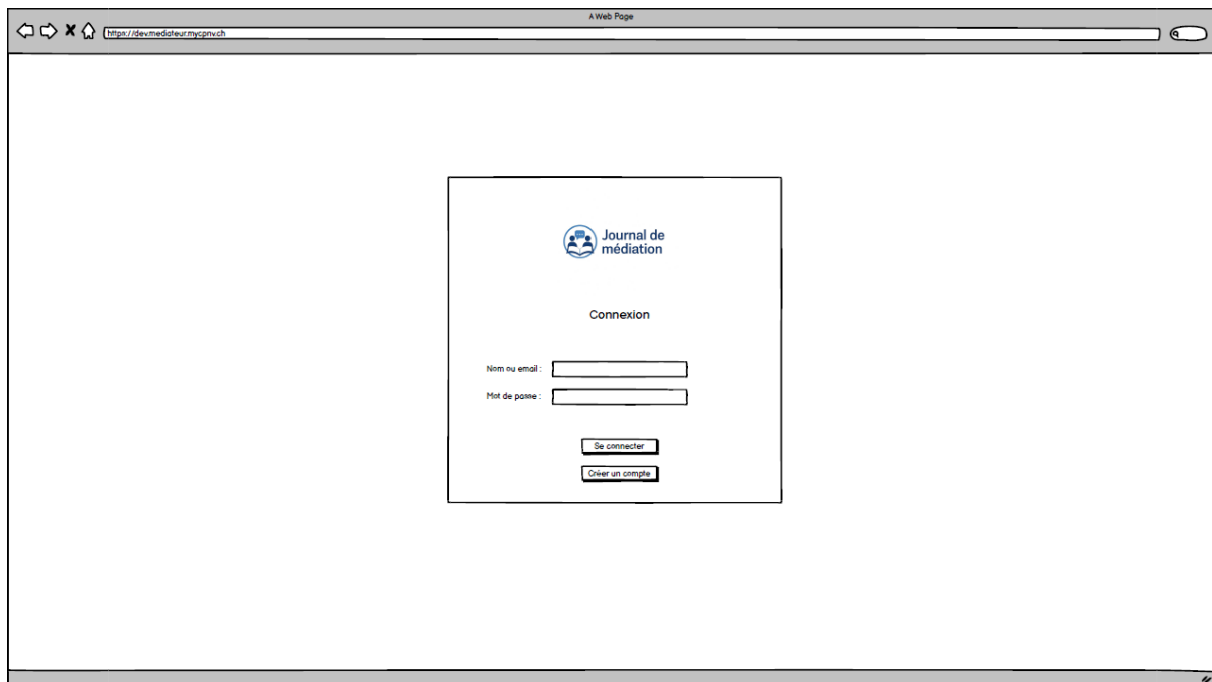
Ma planification initiale n'a pas bougé après analyse mais risque de changer au long du développement vu que j'utilise la méthode Agile. Ainsi, les Stories non finalisées à la fin d'un sprint passent au suivant.

## **2.7 Dossier de conception**

Voici dans cette partie le dossier de conception complet avec la maquette, le MCD et le MLD.

### **2.7.1 Maquette**

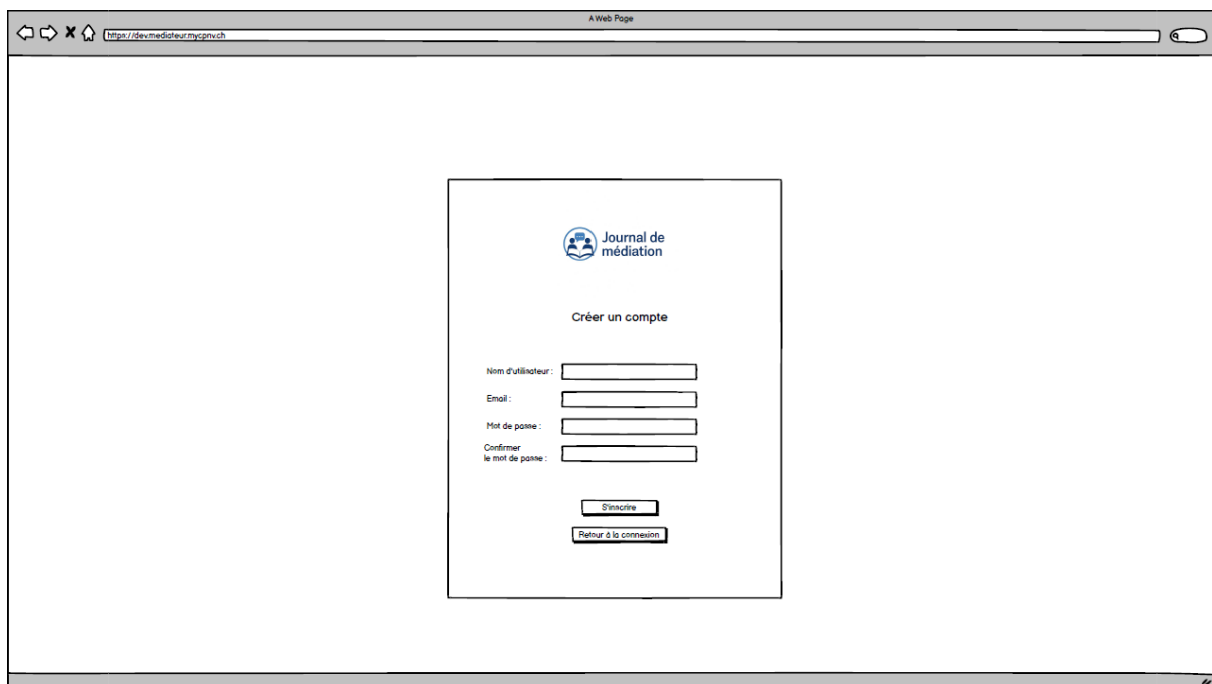




The screenshot shows a web browser window with the address bar displaying "https://dev.mediateur.mycprn.ch". The page content is centered and features the "Journal de médiation" logo at the top. Below the logo, the title "Connexion" is displayed. There are two input fields: "Nom ou email :" and "Mot de passe :". Below these fields are two buttons: "Se connecter" and "Créer un compte".

Figure 1 : Page de Login

Une page de Login avec 2 champs (Utilisateur et mot de passe)



The screenshot shows a web browser window with the address bar displaying "https://dev.mediateur.mycprn.ch". The page content is centered and features the "Journal de médiation" logo at the top. Below the logo, the title "Créer un compte" is displayed. There are four input fields: "Nom d'utilisateur :", "Email :", "Mot de passe :", and "Confirmer le mot de passe :". Below these fields are two buttons: "S'inscrire" and "Retour à la connexion".

Figure 2 : Page d'enregistrement

Une page d'enregistrement qui permet à un nouvel utilisateur de s'inscrire sur l'application.

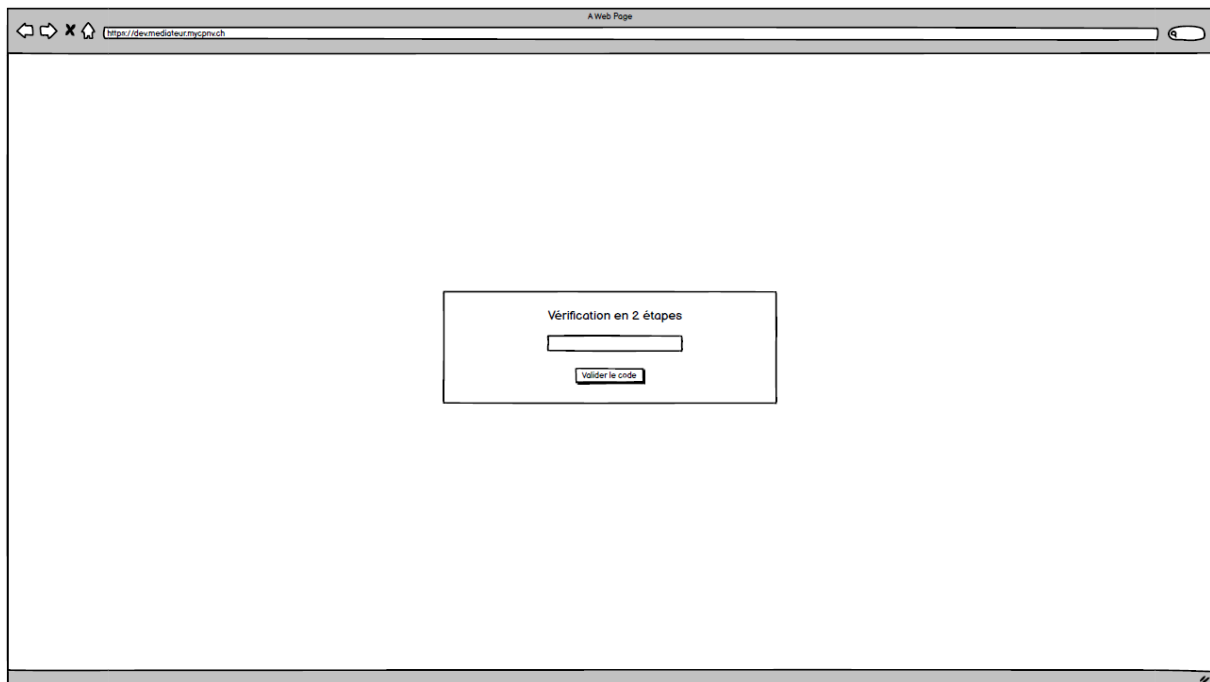


Figure 3 : Page de double authentification

Une page de double authentification qui permet à l'utilisateur de rentrer le code à 6 chiffres reçu par mail.

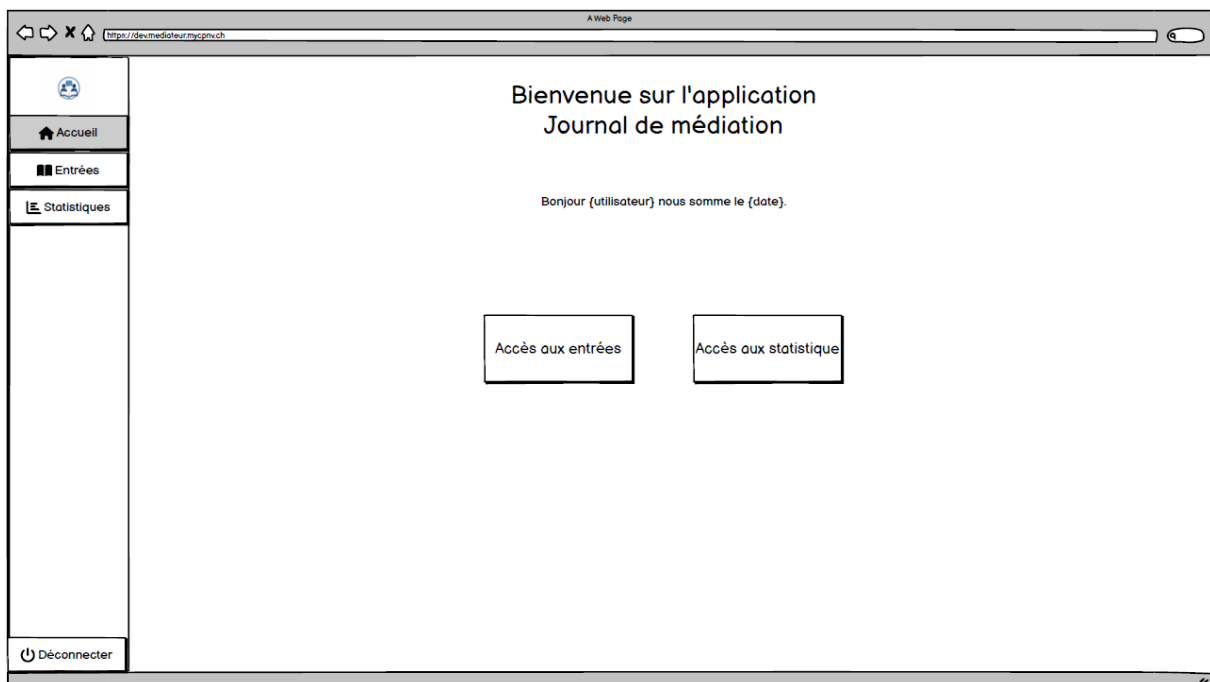


Figure 4 : Page d'accueil

Une page d'accueil qui communique des informations de base

[illegible]

Page d'affichage des données de l'utilisateur. Chaque donnée entrée par un utilisateur sera affichée dans cette grille.

[illegible]

Figure 7 : Formulaire d'ajout d'entretien

L'ajout d'entrée se découpe en 2 parties. Un d'ajout d'entretiens et l'autre d'ajout de séances. Les paramètres de Date, Sujet, Personne concernée et de temps admin sont commun puis d'autres paramètres spécifiques sont en fonction du type choisis.

#### Séance :

- Direction (temps en minutes)
- Enseignant-e-s (temps en minutes)
- Equipe PSPS (temps en minutes)
- Projets (temps en minutes)
- Groupe MPP (temps en minutes)
- Réseau avec parents (temps en minutes)
- Equipe pluri-réseau (temps en minutes)
- Autre (temps en minutes)

#### Entretien :

Des boutons de type radio pour choisir la catégorie (Seul, Groupe, Classe) avec un champ pour le temps de l'entretien. Puis une liste de checkbox pour définit les motivations de l'entretien (Conduite addictives, Incident critique, Conflit entre élèves, Incivilités / violences, Deuil, Mal-être, Difficultés apprentissage, Question d'orientation professionnelles, Difficultés familiales, Stress, Difficultés financières, Suspicion de maltraitements, Discrimination, Difficultés / tentions ave un-e enseignant-e, Harcèlement / Intimidation, Genre – orientation sexuelle et affective, Autre)

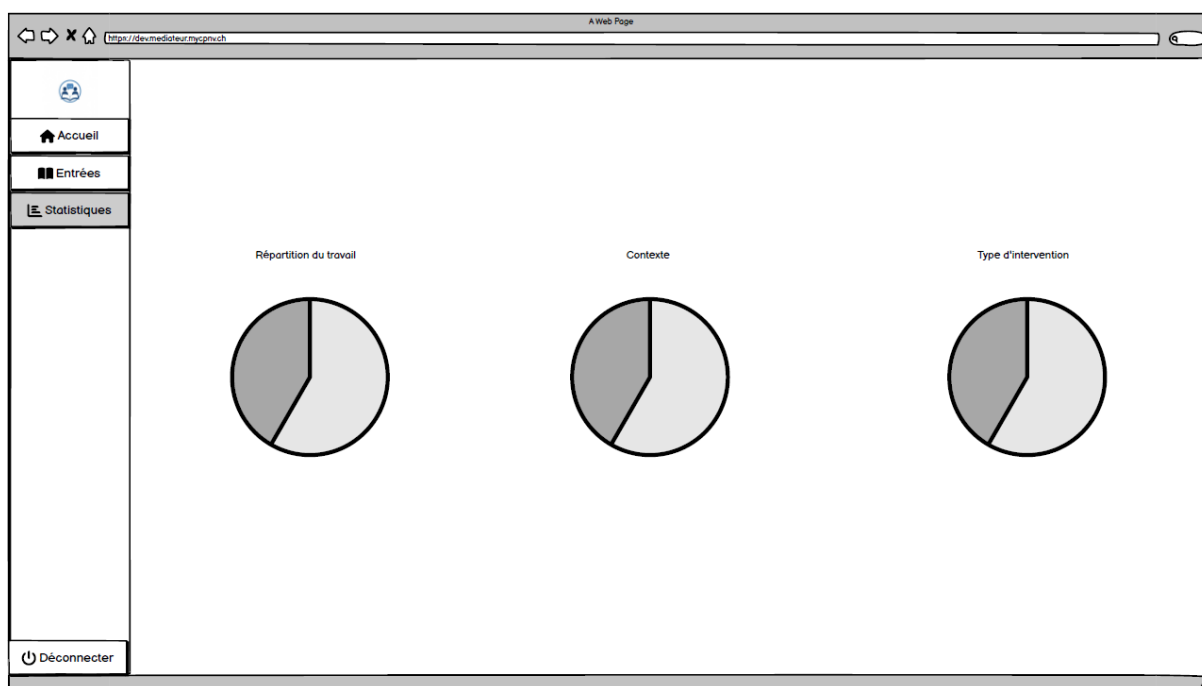
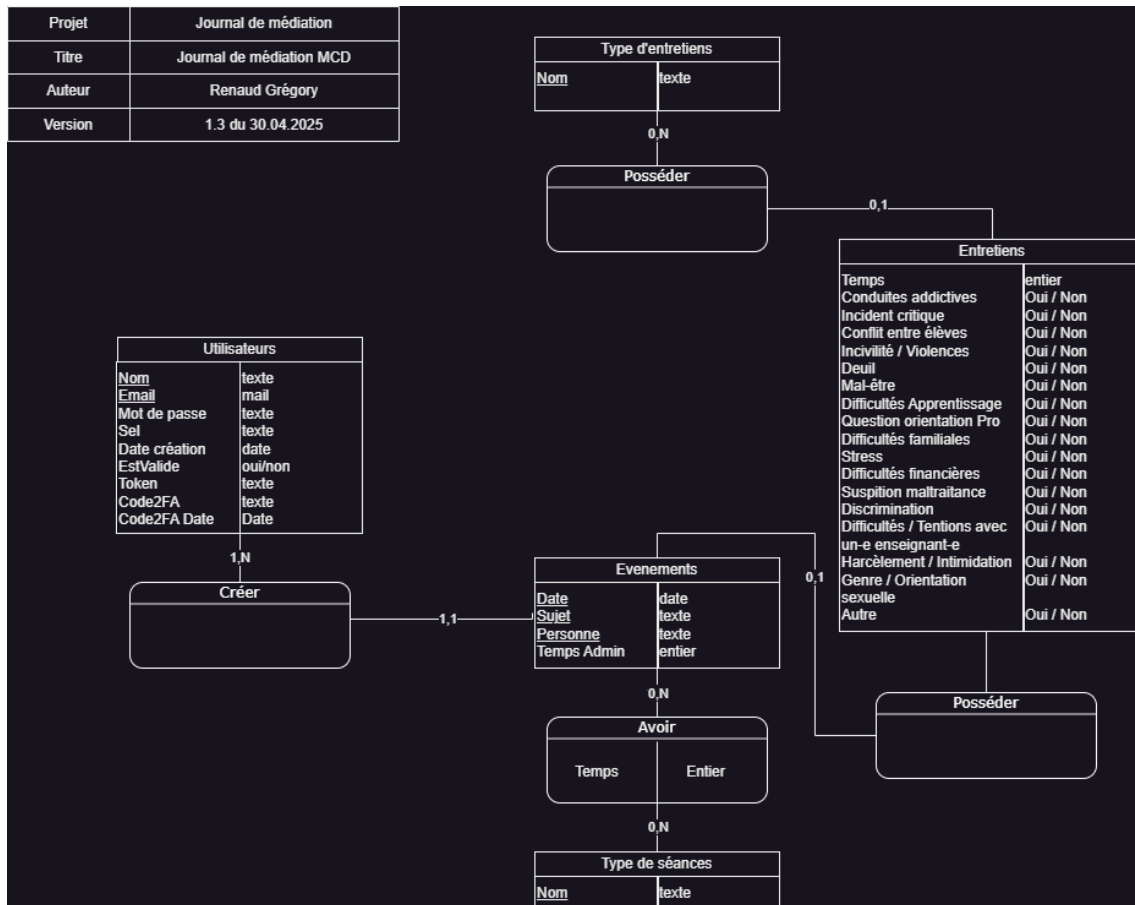


Figure 8 : Page de statistiques

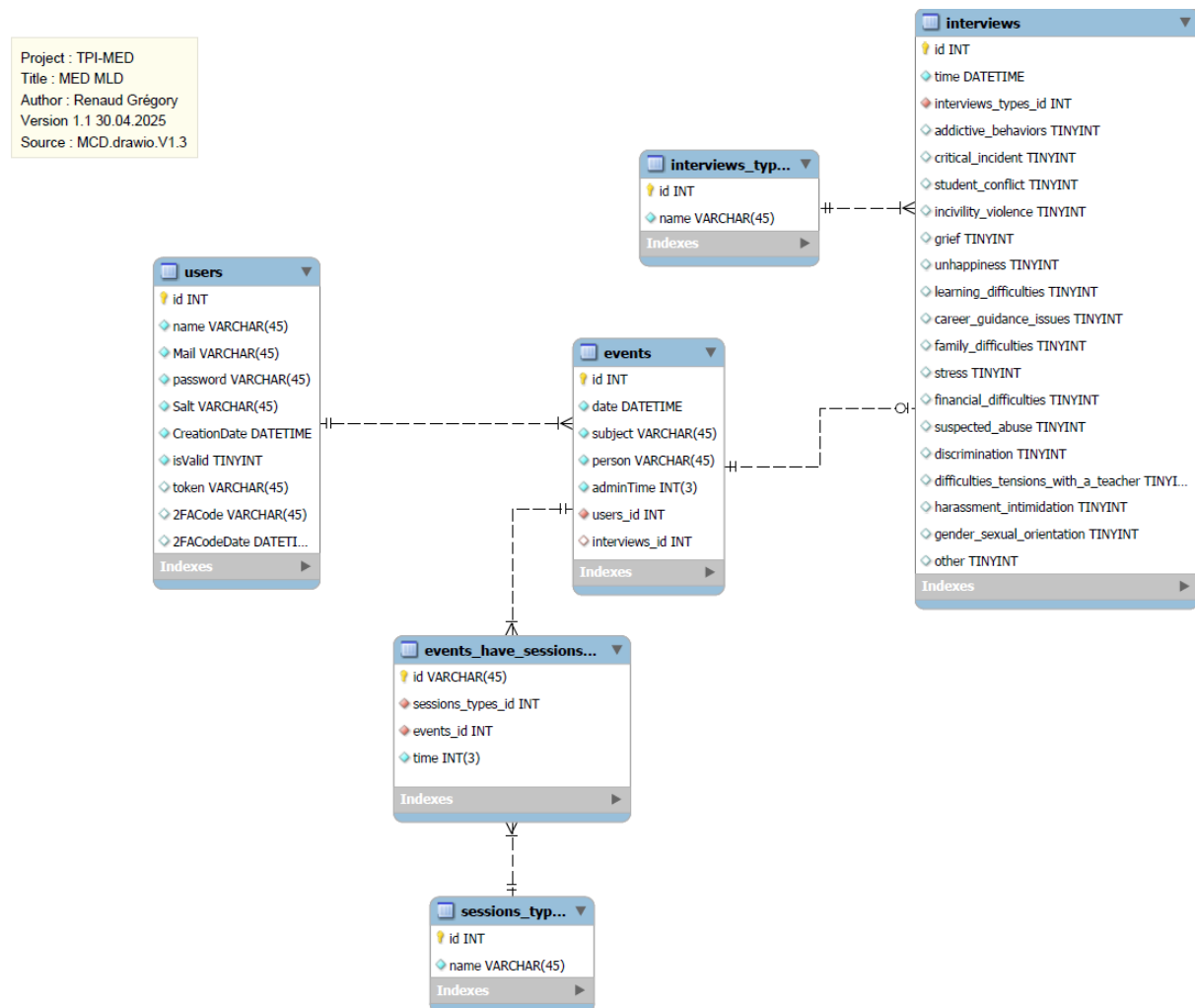
Une page de statistiques liée aux données de l'utilisateur.

## 2.7.2 MCD



Voici le MCD de l'application. Il est composé de 5 tables distinctes (Utilisateurs, Evènements, Type d'entretiens, Entretiens et Type de séances).

### 2.7.3 MLD



Voici la transposition du MCD en MLD. Une nouvelle table est apparue faisant la liaison entre un évènement et un type de session aux vues de la relation N,M qui force l'utilisation d'une table de jointure.

## 3 Réalisation

### 3.1 Dossier de réalisation

#### 3.1.1 Liste des fichiers

##### 3.1.1.1 Classes

- Database.cs : Fournit des méthodes pour gérer la connexion à la base de données MySQL.
- Event.cs : Classe d'objet de type Event.

- 
- EventAffichage.cs : Classe d'objet permettant l'affichage de la datagrid.
  - EventDAO.cs : Classe qui a accès à la DB pour la gestion d'événement.
  - Interview.cs : Classe d'objet de type Interview.
  - InterviewDAO.cs : Classe qui a accès à la DB pour la gestion d'entretiens.
  - MailHelper.cs : Fournit des méthodes utilitaires pour l'envoi de mail, y compris la validation de compte et les codes 2FA.
  - PasswordHelper.cs : Fournit des méthodes utilitaires pour la gestion des mots de passe, y compris la génération de sel et le hachage.
  - Seance.cs : Classe d'objet de type Seance.
  - SeanceDAO .cs : Classe qui a accès à la DB pour la gestion de séances.
  - Utilisateur.cs : Classe d'objet de type Utilisateur.
  - UtilisateurDAO.cs : Classe qui a accès à la DB pour la gestion des utilisateurs.

### 3.1.1.2 Pages

- Code2FAPage.cs + .designer : Représente la page de validation du code à deux facteurs (2FA).
- HomePage.cs + .designer : Représente la page d'accueil de l'application. Elle gère les différents UserControl à afficher à l'aide de la navbar.
- LoginPage.cs + .designer : Représente la page de connexion de l'application avec la logique de connexion.
- RegisterPage.cs + .designer : Représente la page d'inscription de l'application avec la logique d'enregistrement de nouveau compte.
- ValidationPage.cs + .designer : Représente la page de validation de compte utilisateur avec la logique de validation d'email utilisateur.

### 3.1.1.3 UserControl

Pour l'affichage dynamique et plus simplifié, j'ai décidé d'afficher uniquement la page HomePage à l'utilisateur et de modifier le contenu d'un panel par ces fameux UserControl en cliquant sur la navbar.

- AccueilPage.cs + .designer : Représente la fenêtre d'accueil de l'application.
- DataPage.cs + .designer : Représente la fenêtre d'affichage des données ainsi que toute la partie création, modification et suppression de ces derniers.
- NewEntryForm.cs + .designer : Représente le formulaire pour l'ajout de nouvelles données ainsi que pour la modification.
- StatsPage.cs + .designer : représente la fenêtre d'affichage des statistiques de l'utilisateur.

### 3.1.1.4 Configuration

Pour sécuriser mon code, j'ai décidé de compartimenter les données confidentielles.

- Appsettings.Local.json : Fichier contenant les paramètres d'accès à la DB (serveur, database, uid, pwd) et la configuration Smtplib pour l'envoi de mail (host, User, Password).
- Web.config : Fichier de configuration autogénéré de WiseJ qui contient la clé de licence serveur de l'application.

### 3.1.2 Matériel

Le matériel utilisé est celui du CPNV pour la partie développement (i5-9500 3.00GHz, 16Go RAM, SSD 500Go, Intel UHD Graphics 630) qui tourne sous Windows 10 22H2 build 19045.5737. J'utilise Visual Studio 2022 avec .net 9, Wisej.net 3 et C# 7.2.

La database est stockée sur un site SwissCenter fourni par mon chef de projet.

Quant au site web. Il est actuellement hébergé sur des serveur Microsoft Azure fourni par Monsieur Wulliamoz et le CPNV.

### 3.1.3 Librairies

Pour la réalisation de ce projet, j'ai dû utiliser plusieurs librairies externes. La principale est Wisej.net-3 mais j'ai dû aussi utiliser des extensions de Wisej (NavigationBar et ChartJS) ainsi que l'extension MySql.Data de Oracle. Ces 3 extensions sont disponibles à l'aide des packages NuGet.

## 3.2 Description des tests effectués

### 3.2.1 Test Fonctionnels

#### 3.2.1.1 Authentification / Enregistrement

##### 3.2.1.1.1 Connexion avec un compte valide

##### 3.2.1.1.2 Connexion avec un compte invalide

##### 3.2.1.1.3 Création d'un compte valide

##### 3.2.1.1.4 Création d'un compte invalide (mauvais domaine)

##### 3.2.1.1.5 Création d'un compte invalide (même nom d'utilisateur ou email)

##### 3.2.1.1.6 Création d'un compte invalide (mot de passe trop faible)

##### 3.2.1.1.7 Création d'un compte invalide (mot de passe pas correspondant)

##### 3.2.1.1.8 Validation du compte par mail

##### 3.2.1.1.9 Validation du compte par mail (token invalide)

#### 3.2.1.2 Affichage des données

##### 3.2.1.2.1 Chargement correct de la grille

##### 3.2.1.2.2 Suppression d'un évènement



### 3.2.1.2.3 Edition d'un évènement

#### 3.2.1.3 Ajout d'évènement

##### 3.2.1.3.1 Ajout d'un évènement invalide (date hors année scolaire)

##### 3.2.1.3.2 Ajout d'un évènement invalide (sujet vide)

##### 3.2.1.3.3 Ajout d'un évènement invalide (personnes concernées vide)

##### 3.2.1.3.4 Ajout d'une séance invalide (aucun champ de temps rempli)

##### 3.2.1.3.5 Ajout d'une séance valide

##### 3.2.1.3.6 Ajout d'un entretien invalide (pas de type de séance renseigné)

##### 3.2.1.3.7 Ajout d'un entretien invalide (aucunes motivation renseignées)

##### 3.2.1.3.8 Ajout d'un entretien valide

### 3.3 Erreurs restantes

*S'il reste encore des erreurs:*

- *Description détaillée*
- *Conséquences sur l'utilisation du produit*
- *Actions envisagées ou possibles*

### 3.4 Liste des documents fournis

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

- *le rapport de projet*
- *le manuel d'Installation (en annexe)*
- *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
- *autres...*

## 4 Conclusions

*Développez en tous cas les points suivants:*

- *Objectifs atteints / non-atteints*
- *Points positifs / négatifs*

- *Difficultés particulières*
- *Suites possibles pour le projet (évolutions & améliorations)*

## 5 Annexes

### 5.1 Résumé du rapport du TPI / version succincte de la documentation

### 5.2 Sources – Bibliographie

*Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)... Et de toutes les aides externes (noms)*

### 5.3 Journal de travail

Date	Durée	Activité	Remarques

### 5.4 Manuel d'Installation

### 5.5 Manuel d'Utilisation

### 5.6 Archives du projet

*Media, ... dans une fourre en plastique*