|  |
| --- |
| Journal de médiation    Figure 1 : Logo Journal de médiation |

Renaud Grégory

CPNV Ste-Croix

SICA-2A

Imprimé le, mardi 13 mai 2025

Table des matières

[1 Analyse préliminaire 3](#_Toc197956022)

[1.1 Introduction 3](#_Toc197956023)

[1.2 WiseJ.net 3](#_Toc197956024)

[1.2.1 Caractéristiques principales 3](#_Toc197956025)

[1.2.2 Avantages 3](#_Toc197956026)

[1.2.3 Utilisation typique 3](#_Toc197956027)

[1.3 Outils Utilisé 4](#_Toc197956028)

[1.4 Objectifs 4](#_Toc197956029)

[1.5 Planification initiale 5](#_Toc197956030)

[2 Analyse préliminaire 6](#_Toc197956031)

[2.1 Introduction 6](#_Toc197956032)

[2.2 Critères d’ergonomie (Bastien et Scapin) 6](#_Toc197956033)

[2.3 Cryptographie 7](#_Toc197956034)

[2.4 Stratégie de test 8](#_Toc197956035)

[2.5 Risques techniques 8](#_Toc197956036)

[2.6 Planification 8](#_Toc197956037)

[2.7 Dossier de conception 9](#_Toc197956038)

[2.7.1 Maquette 9](#_Toc197956039)

[2.7.2 MCD 14](#_Toc197956040)

[2.7.3 MLD 15](#_Toc197956041)

[3 Réalisation 15](#_Toc197956042)

[3.1 Dossier de réalisation 15](#_Toc197956043)

[3.1.1 Liste des fichiers 15](#_Toc197956044)

[3.1.2 Matériel 17](#_Toc197956045)

[3.1.3 Librairies 17](#_Toc197956046)

[3.2 Description des tests effectués 17](#_Toc197956047)

[3.2.1 Test Fonctionnels 17](#_Toc197956048)

[3.2.2 Test d’interface Utilisateur 39](#_Toc197956049)

[3.3 Erreurs restantes 39](#_Toc197956050)

[3.4 Liste des documents fournis 40](#_Toc197956051)

[4 Conclusions 40](#_Toc197956052)

[5 Annexes 41](#_Toc197956053)

[5.1 Résumé du rapport du TPI / version succincte de la documentation 41](#_Toc197956054)

[5.2 Sources – Bibliographie 42](#_Toc197956055)

[5.3 Table des illustrations 42](#_Toc197956056)

[5.4 Glossaire 43](#_Toc197956057)

[5.5 Journal de travail 46](#_Toc197956058)

[5.6 Manuel d'Installation 46](#_Toc197956059)

[5.7 Manuel d'Utilisation (à faire) 46](#_Toc197956060)

[5.8 Cahier des charges 47](#_Toc197956061)

[5.9 Archive du projet 52](#_Toc197956062)

# Analyse préliminaire

## 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.

## 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.

### Caractéristiques principales

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

### Avantages

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

### 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 Windows Forms avec la puissance du web.

## 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) |
| Doxygen | Logiciel permettant de générer automatiquement de la documentation de code basé sur les entêtes de classe et de méthodes |
| Azure | Site web qui héberge l’application pour la phase de test |

## 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 doit 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 traités comme des entrée erronées (ex : Type de champ)
* L’application est utilisable sur smartphone
* Les données stockées ne sont disponibles qu’à l’utilisateur qui les a insérées

## Planification initiale

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

Une image contenant texte, capture d’écran, logiciel, conception

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.

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 dois 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.

# Analyse préliminaire

## 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) ainsi qu’à la sécurisation des données par le chiffrement.

## Critères d’ergonomie (Bastien et Scapin)

Les critères d’ergonomies 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 utilisés pour évaluer ou concevoir des interfaces utilisateur efficaces 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
  + Menus clairs, Boutons explicites, Messages d’aide
* Charge de travail : L’interface doit minimiser les efforts cognitifs et physiques
  + Remplissage automatique, Groupement logique des champss
* Contrôle explicite : L’utilisateur doit garder le contrôle sur ses actions
  + Confirmation avant suppression, Choix clairs et sans actions cachées
* Adaptabilité : L’interface doit s’adapter aux besoins et préférences des utilisateurs
  + Mode sombre / clair, Interface responsive (mobile / PC)
* Gestion des erreurs : L’interface doit éviter les erreurs et aider à les corriger facilement
  + Messages d’erreur clairs, validation de format (email, code)
* Homogénéité / Cohérence : L’interface doit être cohérente dans tout le système
  + Même design pour tous les boutons, navigation similaire sur tout le site
* Signifiance des Codes et Dénomination : Les icônes, couleurs, labels doivent être compréhensible immédiatement
  + Poubelle = supprimer, Disquette = enregistrer, Aucun jargon technique pour l’utilisateur
* Compatibilité : L’interface doit respecter les habitudes et attentes de l’utilisateur
  + Raccourcis clavier connus (Ctrl + S), Position du bouton « OK » à droite

## 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 illisibles pour toutes personnes non autorisées. 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 :
  + Même clé pour chiffrer et déchiffrer
  + Rapide, mais nécessite de partager la clé secrètement
  + Exemples : AES, DES
* Cryptographie asymétrique
  + Une clé publique pour chiffrer
  + Une clé privée pour déchiffrer
  + Très utilisé pour l’échange sécurisé de données et des signatures numériques
  + Exemples : RSA, ECC
* Fonction de hachage
  + Transforme une donnée en empreinte fixe
  + Fonction irréversible
  + Utile pour stocker des mots de passe, vérifier l’intégrité
  + 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 ajoutée à un mot de passe avant le hachage
* Clé publique / privée : Système de clés utilisé 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 passe ainsi que l’utilisation de certificats pour le HTTPS.

## Stratégie de test

Concernant ma stratégie de test, j’ai décidé de la faire en 2 parties. La première, selon mes tests d’acceptation sur IceScrum (sur PC et sur tablette). Puis la seconde en envoyant l’accès à l’application à différents utilisateurs 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 leur permettre de tester toutes les fonctionnalités, un guide d’utilisation ainsi qu’un Google Form pour permettre les retours de bug ou améliorations.

## Risques techniques

Les risques techniques liés à mon application sont plus au niveau du déploiement. En effet, c’est la première application que je déploie sur un serveur. C’est pourquoi je vais me baser sur la documentation Wisej.net ainsi que la documentation .net Core de Microsoft (malgré le fait que le déploiement n’est pas demandé dans mon cahier des charges).

## Planification

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

## Dossier de conception

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

### Maquette

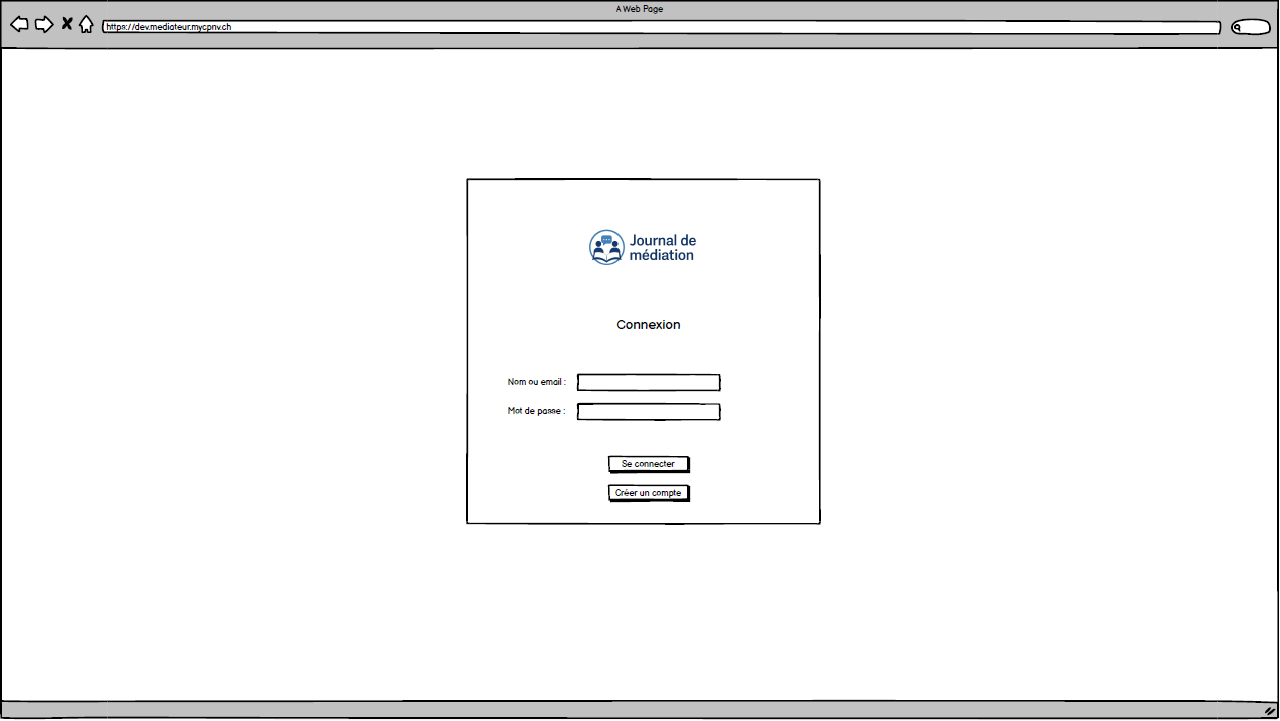


Figure 2 : Page de Login

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

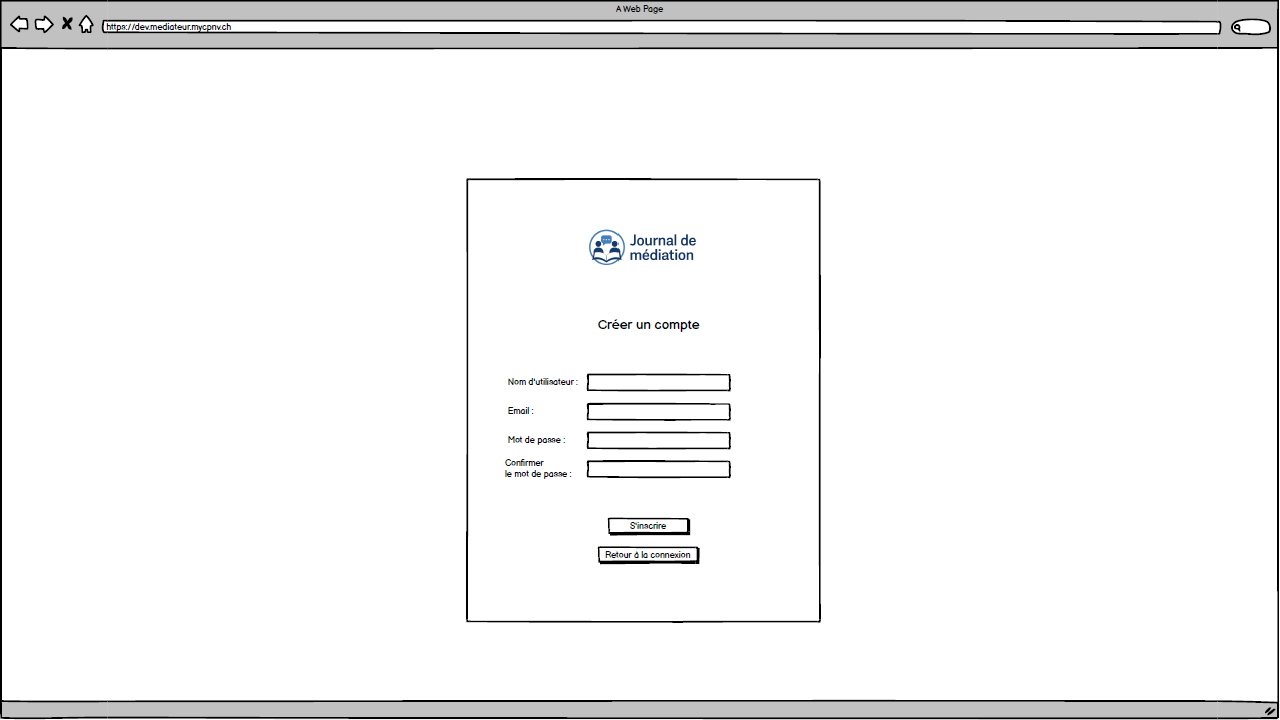


Figure 3 : Page d'enregistrement

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

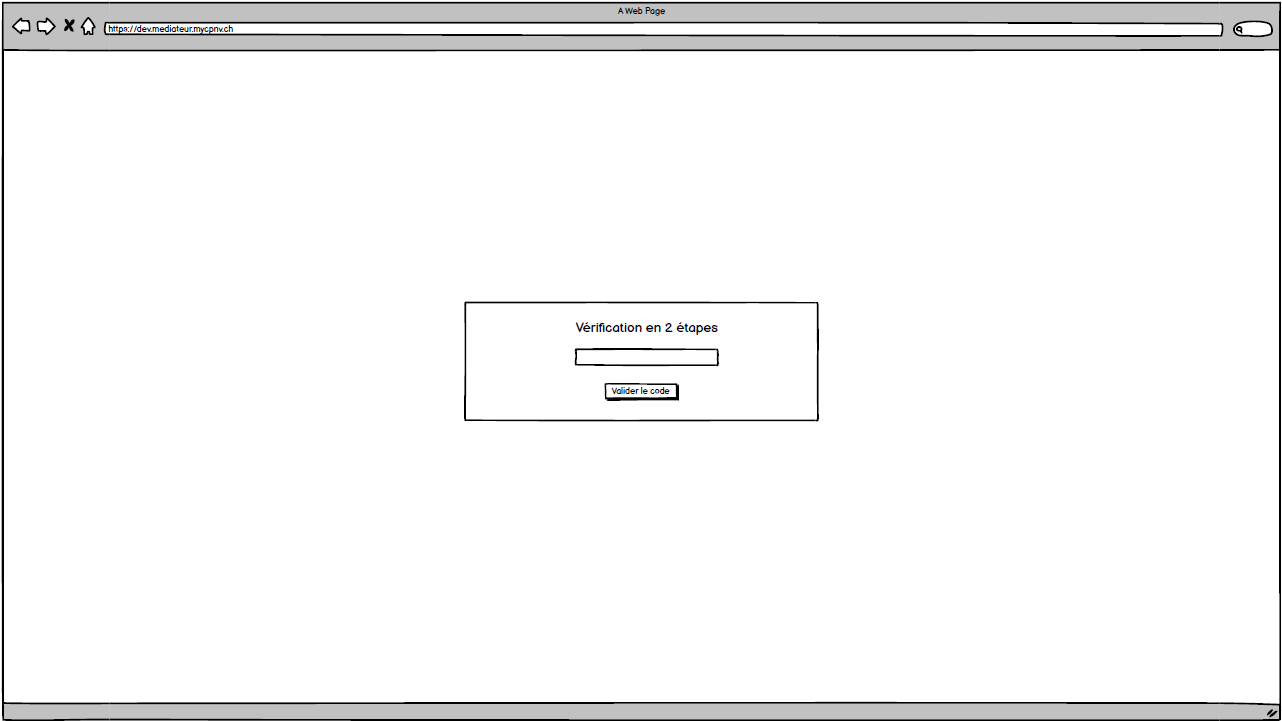


Figure 4 : Page de double authentification

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

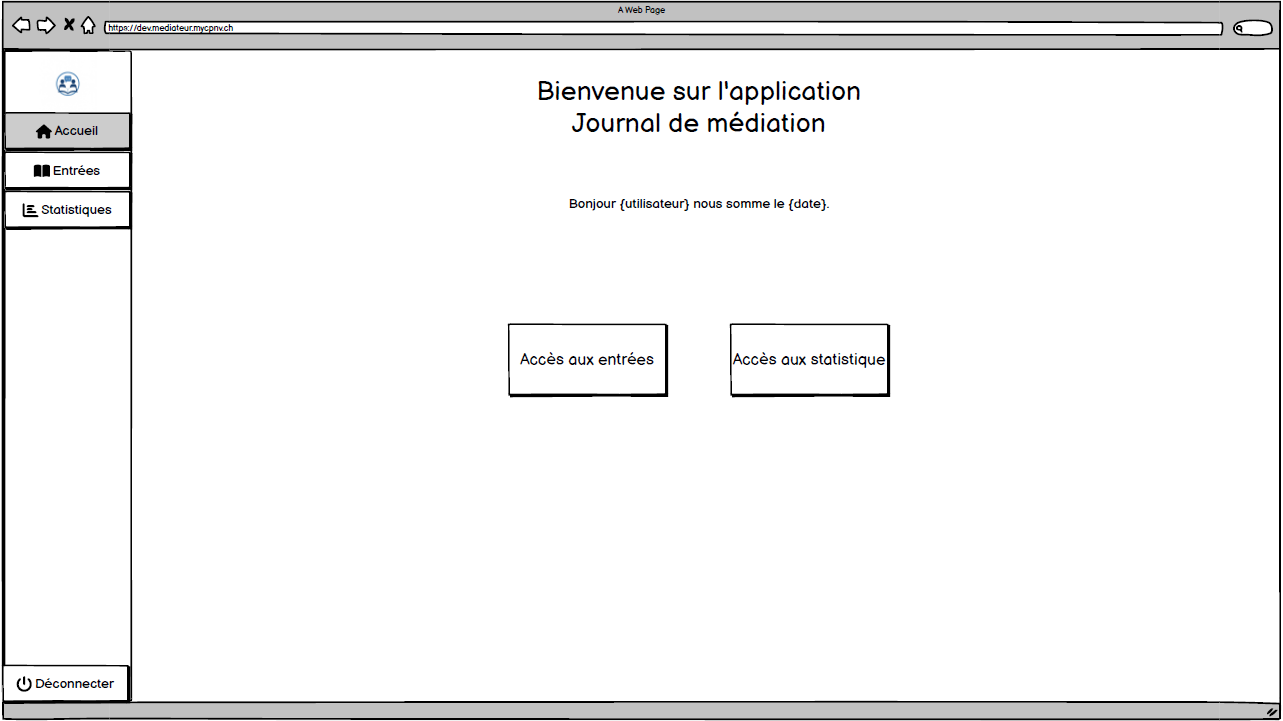


Figure 5 : Page d'accueil

Une page d’accueil qui indique des informations de base

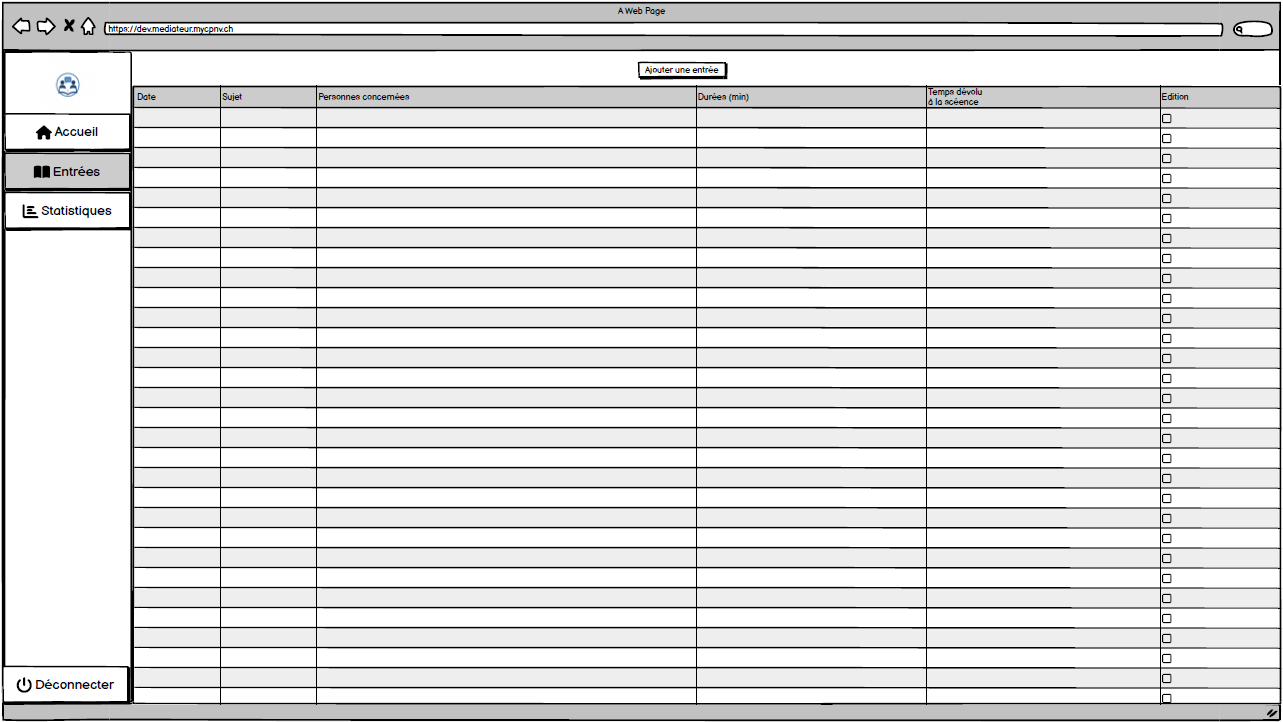


Figure 6 : Page d'affichage des données

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

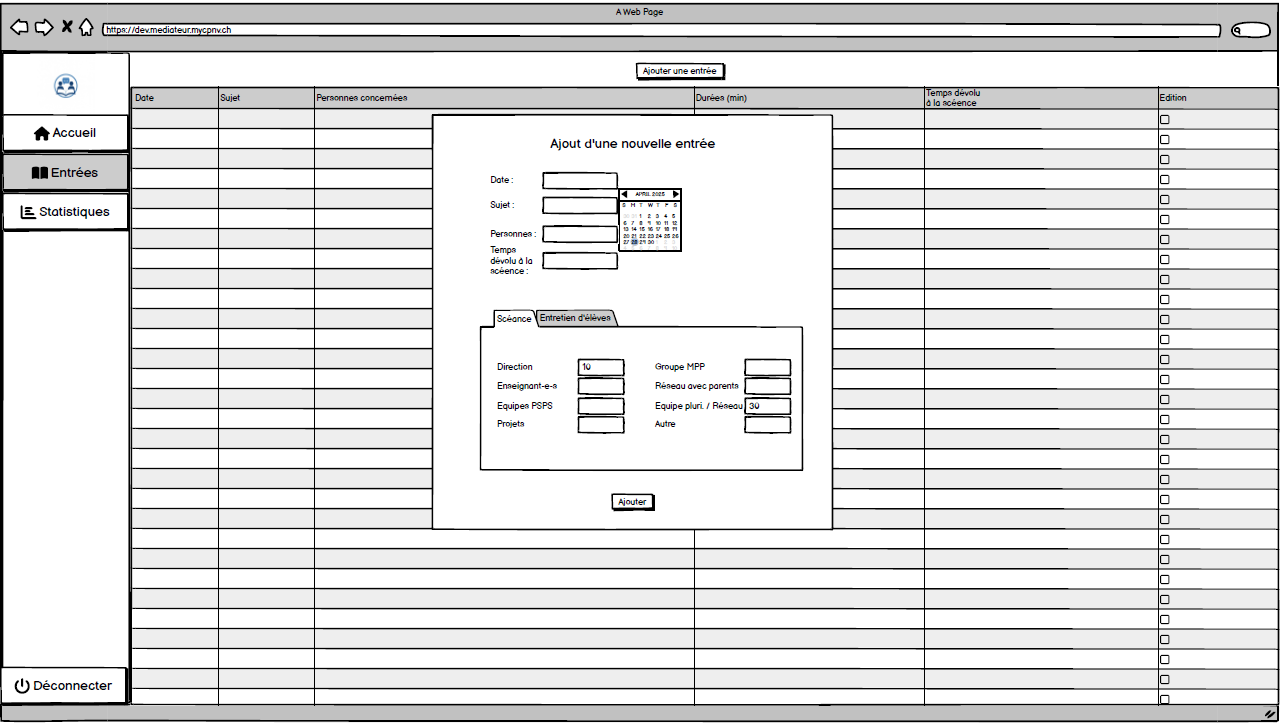


Figure 7 : Formulaire d'ajout de séance

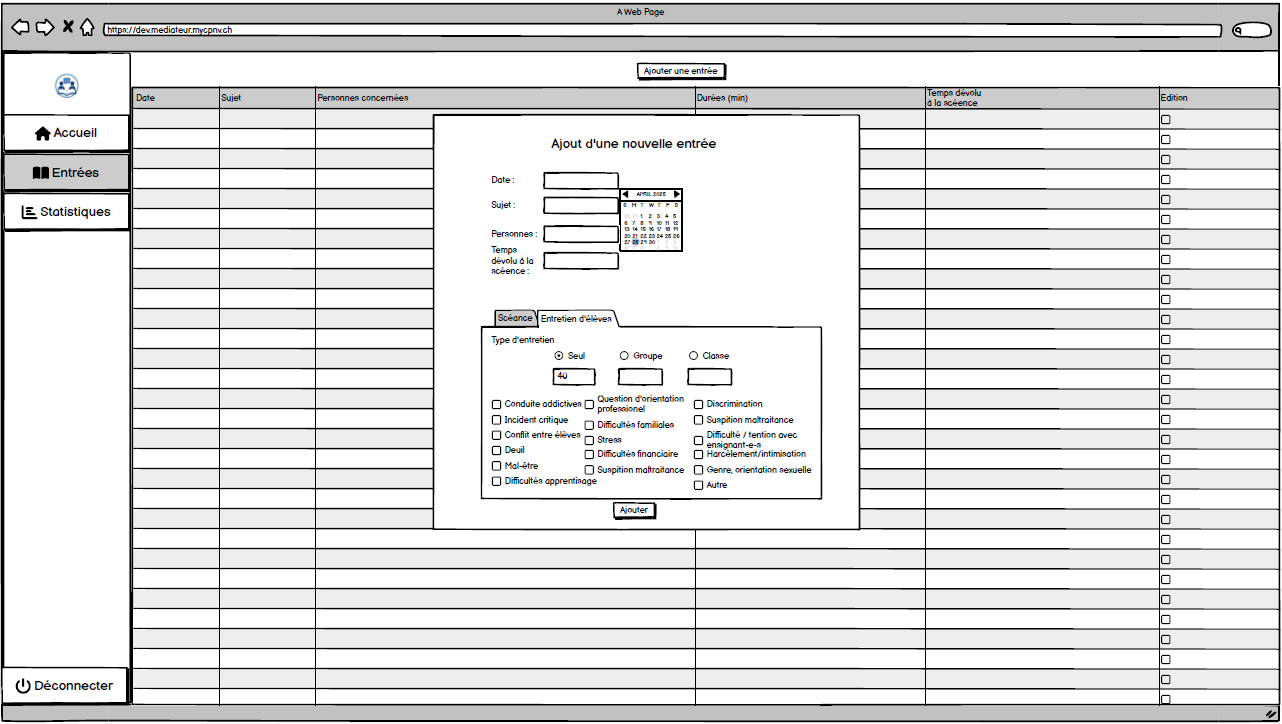


Figure 8 : Formulaire d'ajout d’entretien

L’ajout d’entrée se découpe en 2 parties. Un, la saisie d’entretiens et l’autre, la saisie de séances. Les paramètres de Date, Sujet, Personne concernée et de temps admin sont communs 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 :

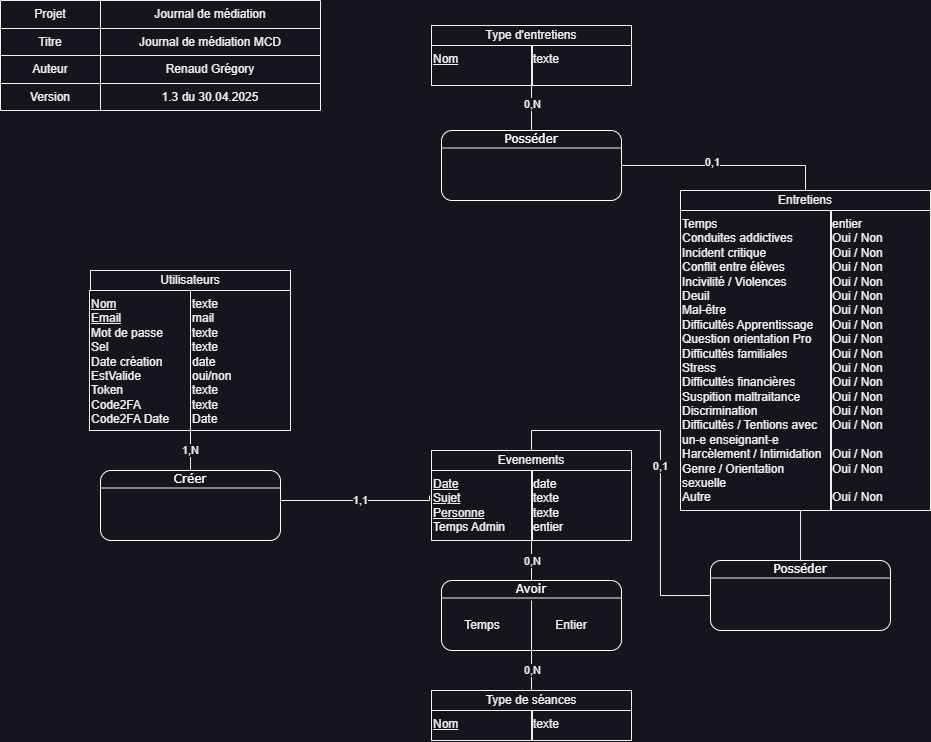
Des boutons de type radio pour choisir la catégorie (Seul, Groupe, Classe) avec un champ pour le temps de l’entretien. Puis une liste de checkbox pour définir les motivations de l’entretien (Conduite addictives, Incident critique, Conflit entre élèves, Incivilités / violences, Deuil, Mal-être, Difficultés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



Figure 9 : Page de statistiques

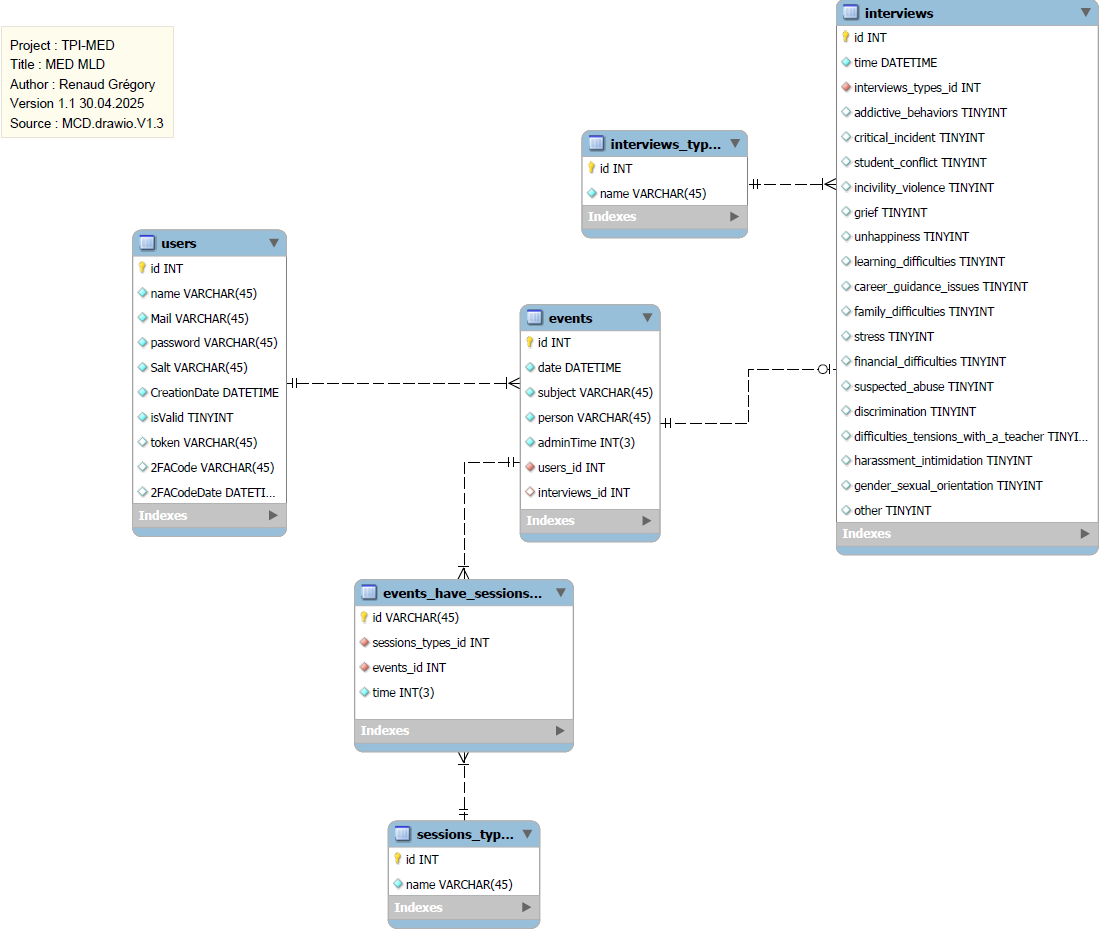
Une page de statistiques liée aux données de l’utilisateur.

### MCD



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

### 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.

# Réalisation

## Dossier de réalisation

### Liste des fichiers

#### Classes

* Database.cs : Fournit des méthodes pour gérer la connexion à la base de données MySQL.
* DateHelper.cs : Fournit des méthodes pour gérer la validation des années scolaires.
* 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’évènement.
* 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.

#### 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.

#### UserControl

Pour un 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.

#### 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 Smtp 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.

### Matériel

Le matériel utilisé est celui du CPNV pour la partie développement (i5-9500 3.00GHz, 16Go RAM, SSD 500Go, Intel UHD Graphics 630) qi 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.

### 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.

### Résultat par rapport à l’analyse

Courant ma phase d’analyse, j’ai dû réaliser plusieurs produit (Maquette, MCD, MLD) pour m’aider et me faciliter le développement de l’application. Durant cette dernière, j’ai dû adapter notamment mon interface aux contraintes de développement et aux contraintes d’ergonomie. Ce qui fait que le produit final possède des variations de disposition par rapport à la maquette initiale.

## Description des tests effectués

### Test Fonctionnels

#### Authentification / Enregistrement

##### Connexion avec un compte valide

Condition de test : Un utilisateur valide entre un bon email ou nom d’utilisateur et mot de passe.

Résultat attendu : Redirection vers la page de 2FA et réception de l’email de 2FA à l’adresse email de l’utilisateur.

Preuve :

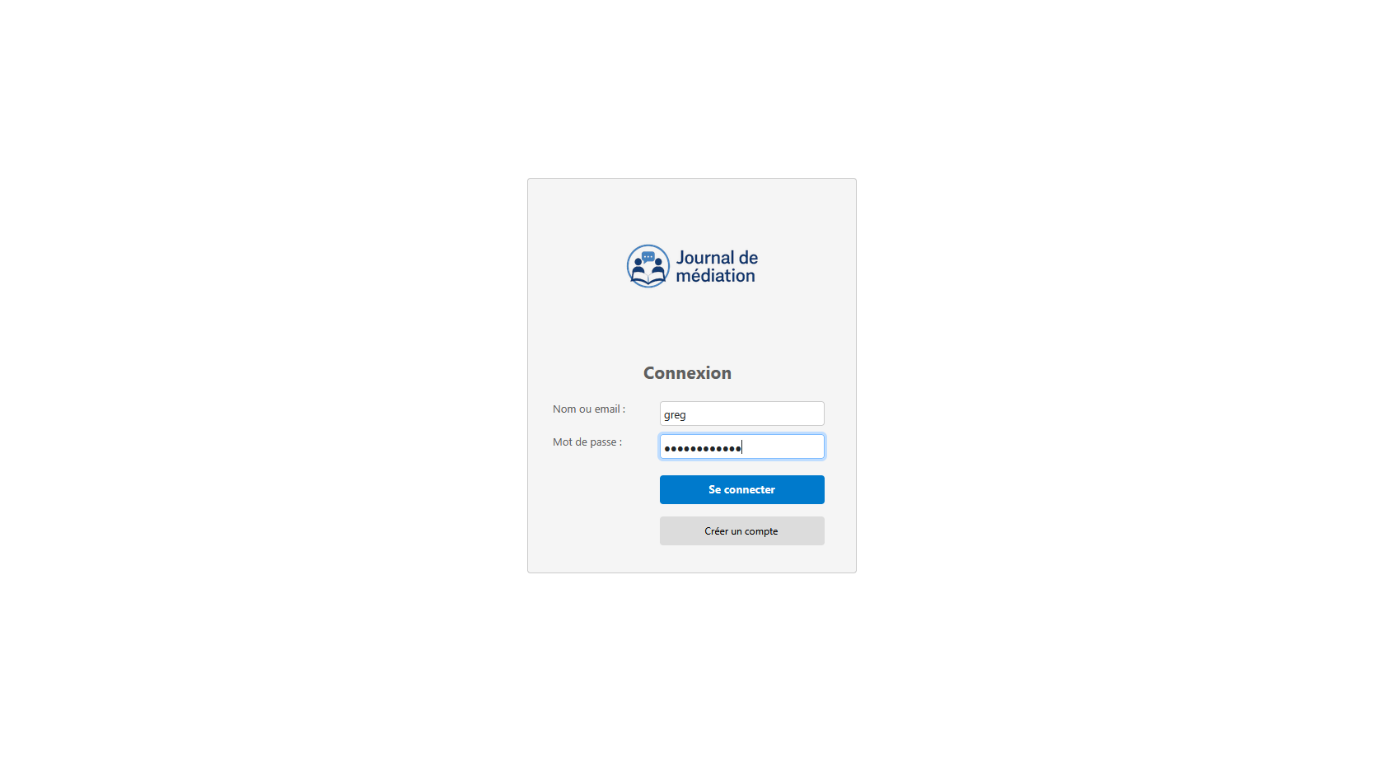


Figure 10 : Connexion avec un compte valide partie 1

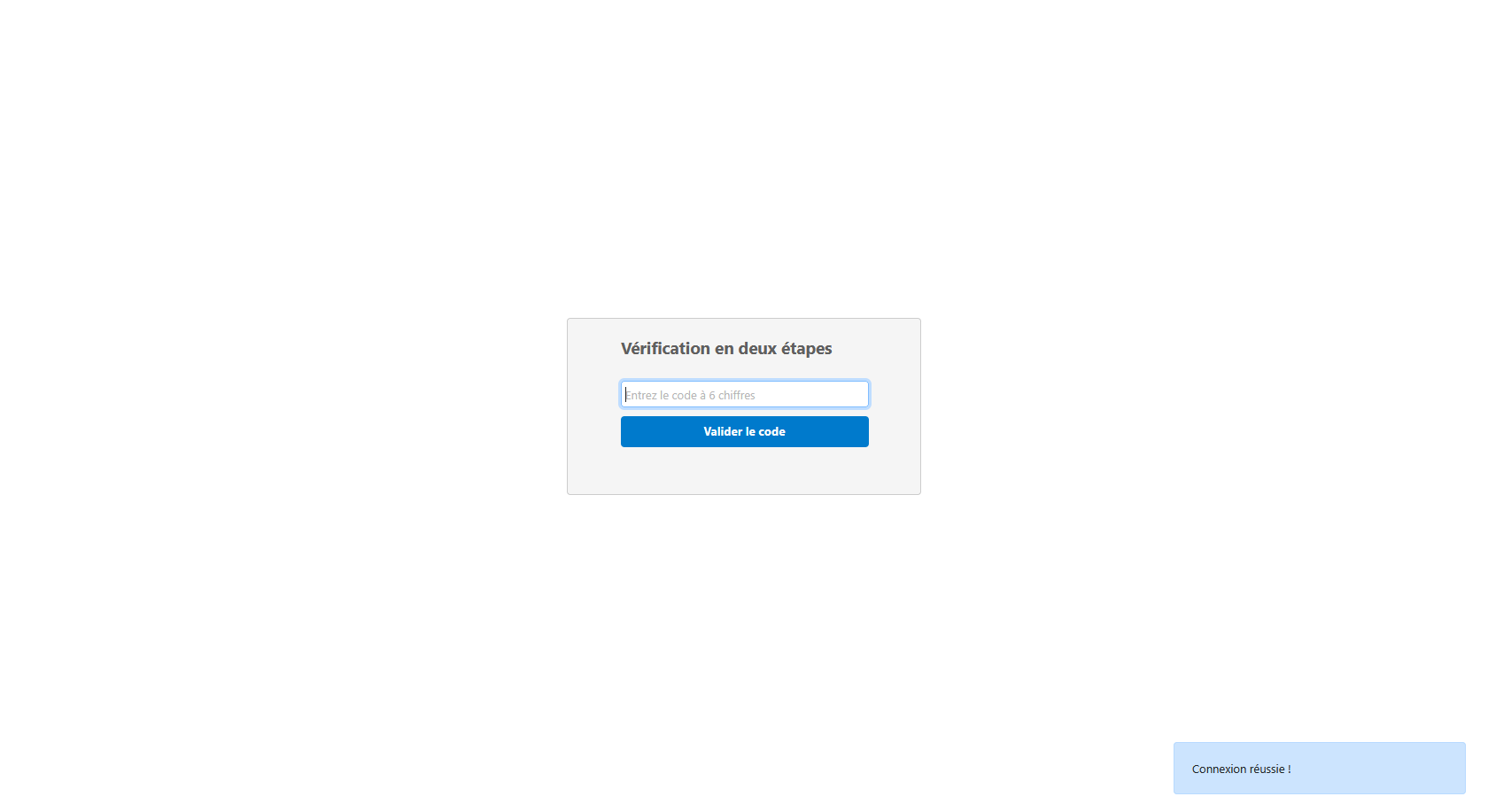


Figure 11 : Connexion avec un compte valide partie 2

##### Validation de la 2FA code valide

Condition de test : L’utilisateur entre le bon code reçu par email.

Résultat attendu : Un message de confirmation que le code est bon s’affiche puis redirige vers la page d’accueil.

Preuve :

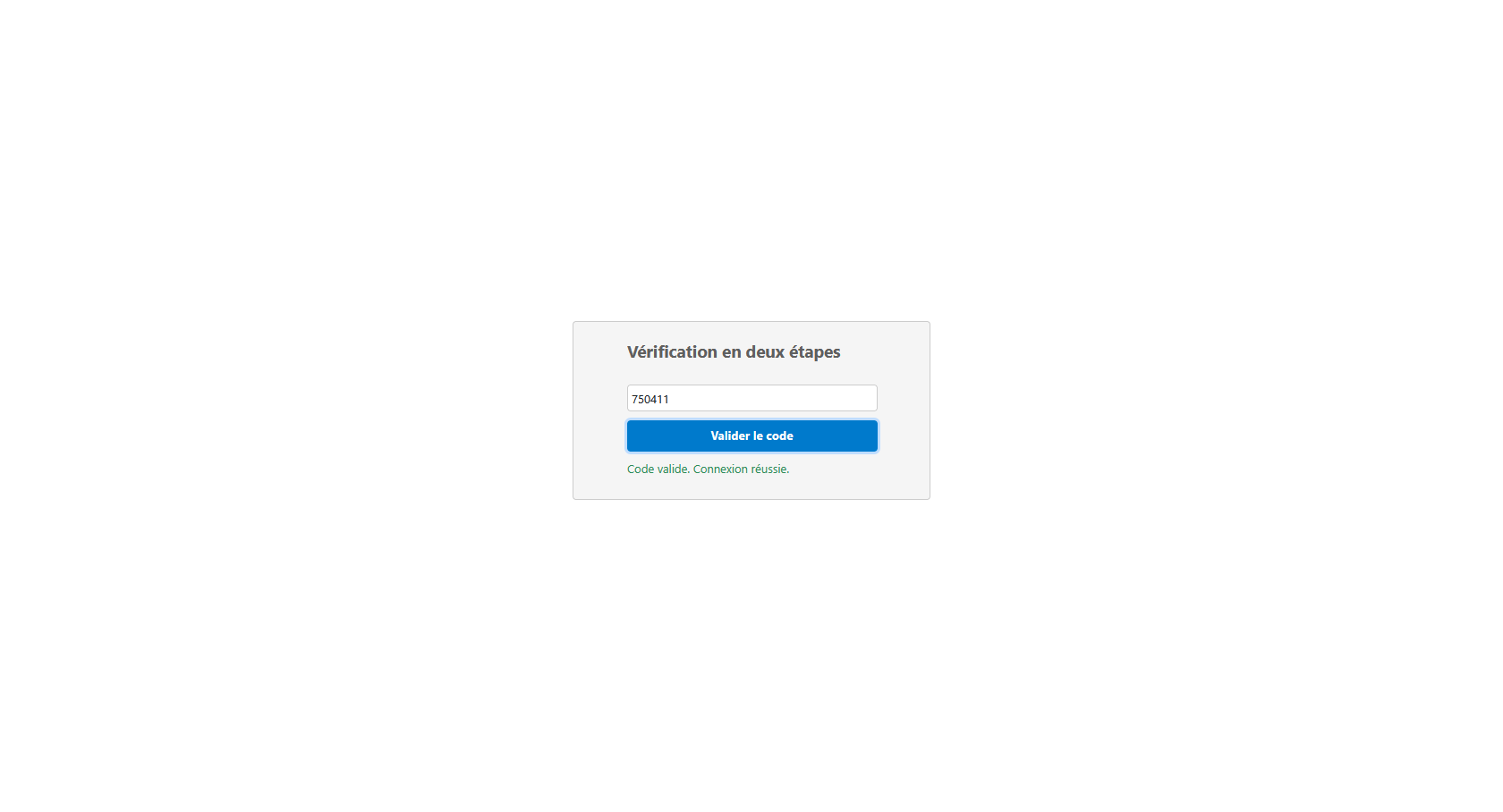


Figure 12 : Validation de la 2FA avec un code valide partie 1

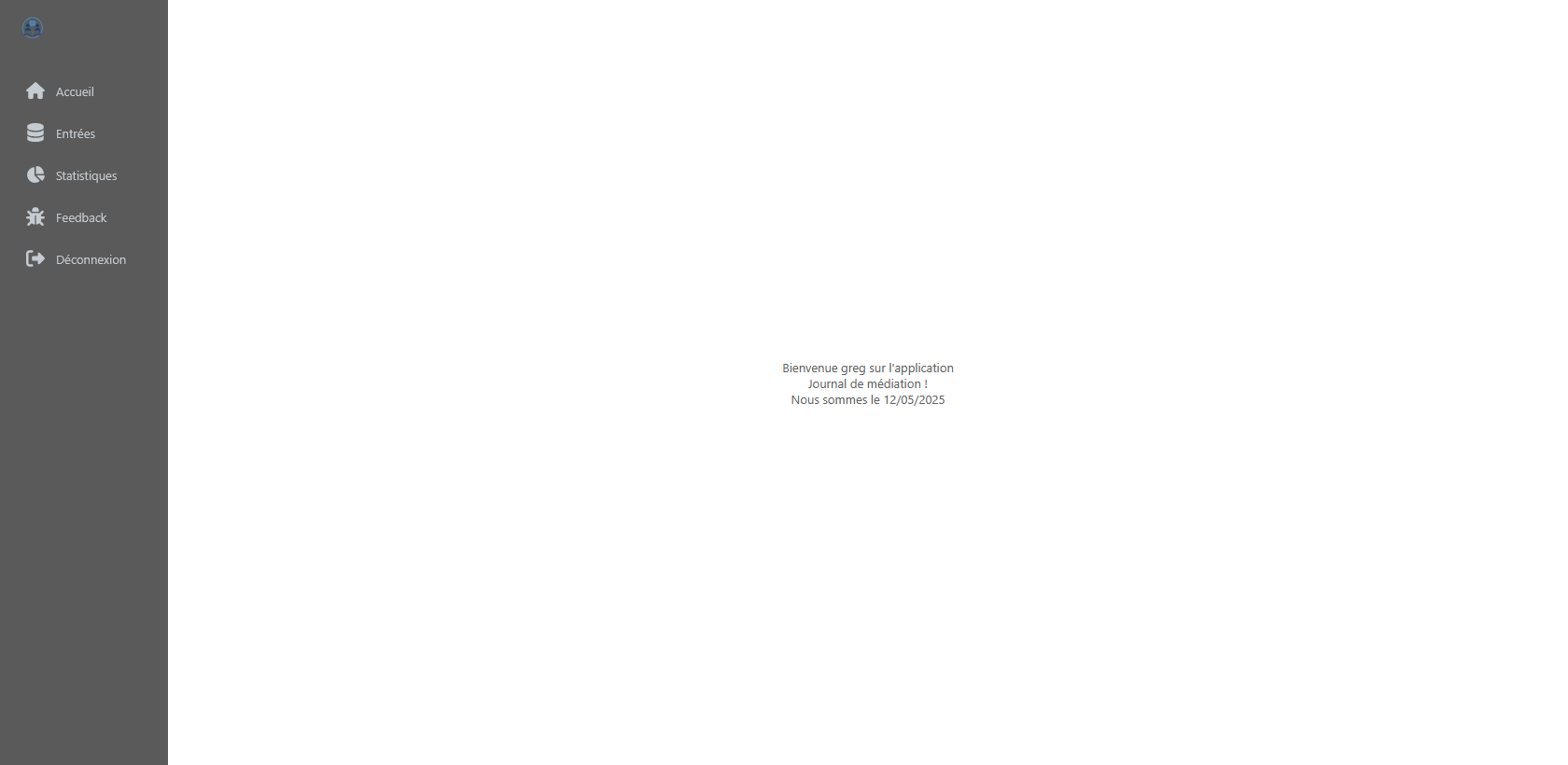


Figure 13 : Validation de la 2FA avec un code valide partie 2

##### Validation de la 2FA code invalide

Condition de test : L’utilisateur entre un code incorrect ou expiré (code valide 15 min)

Résultat attendu : Un message d’erreur « Code invalide ou expiré ».

Preuve :

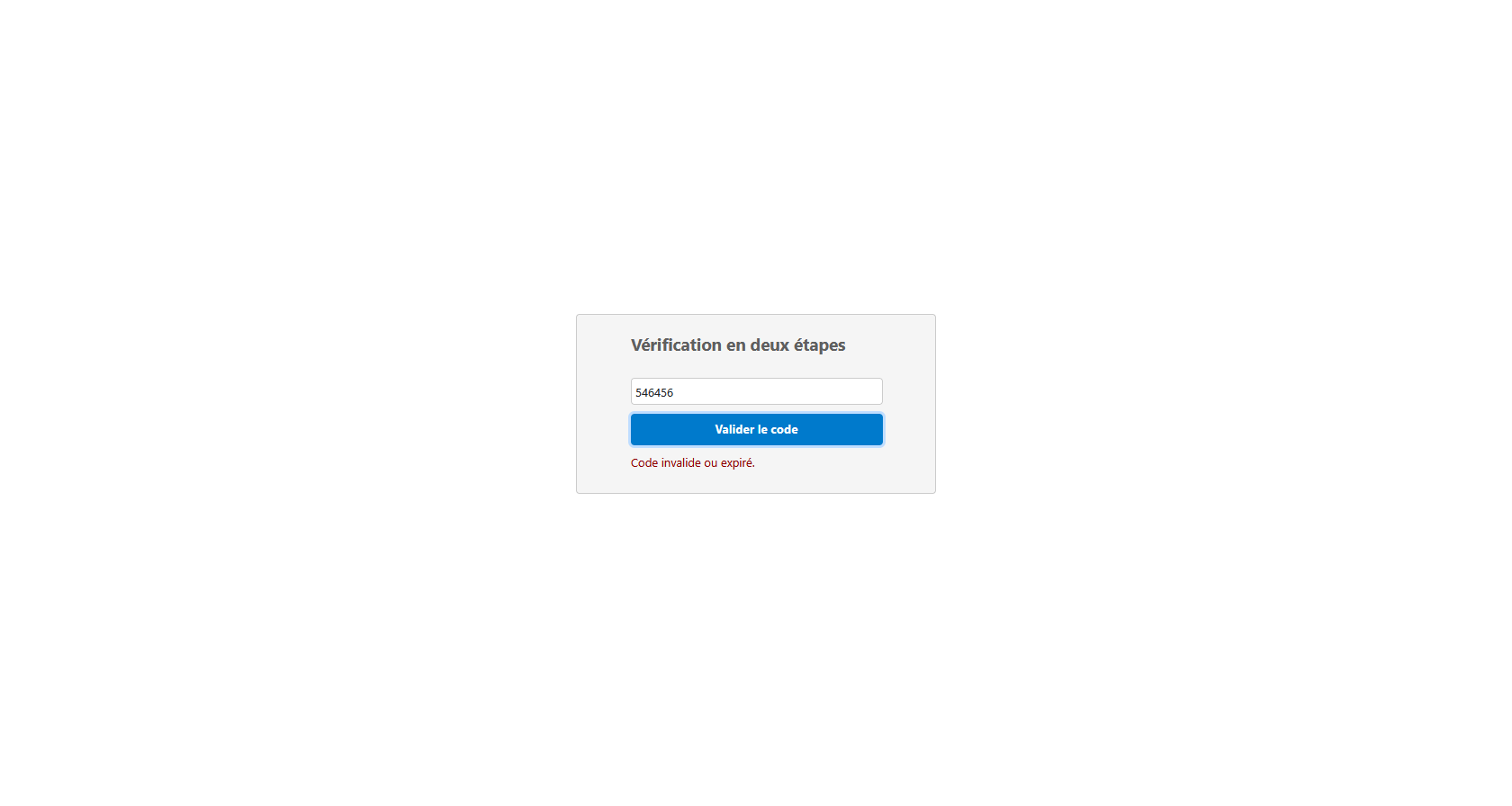


Figure 14 : Validation de la 2FA avec un code invalide

##### Connexion avec un compte invalide (mauvais mot de passe)

Condition de test : L'utilisateur entre un mauvais mot de passe

Résultat attendu : Message d'erreur « Utilisateur ou mot de passe incorrect ».

Preuve :

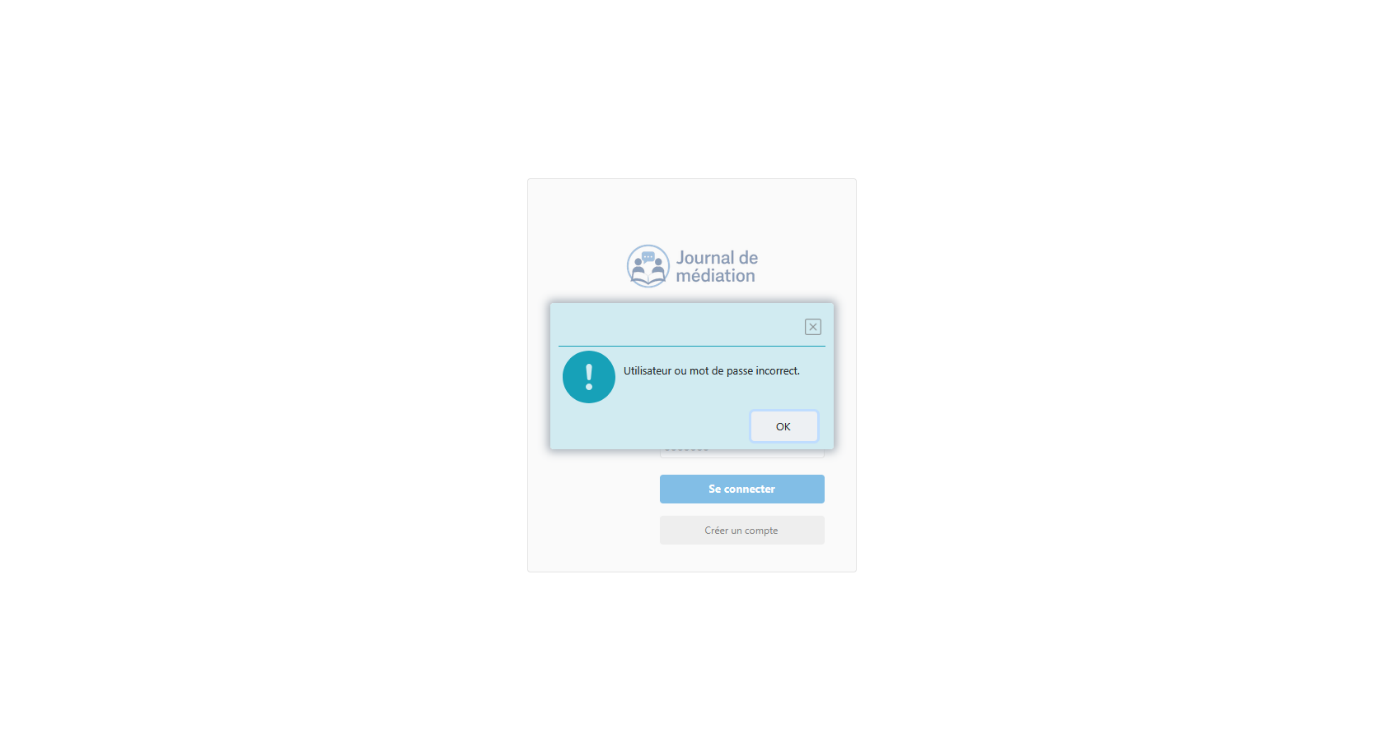


Figure 15 : Connexion avec un compte invalide car un mauvais mot de passe est renseigné

##### Connexion avec un compte invalide (utilisateur inexistant)

Condition de test : L’utilisateur entre une adresse email ou un nom d’utilisateur inexistant.

Résultat attendu : Message d'erreur « Utilisateur ou mot de passe incorrect ».

Preuve :

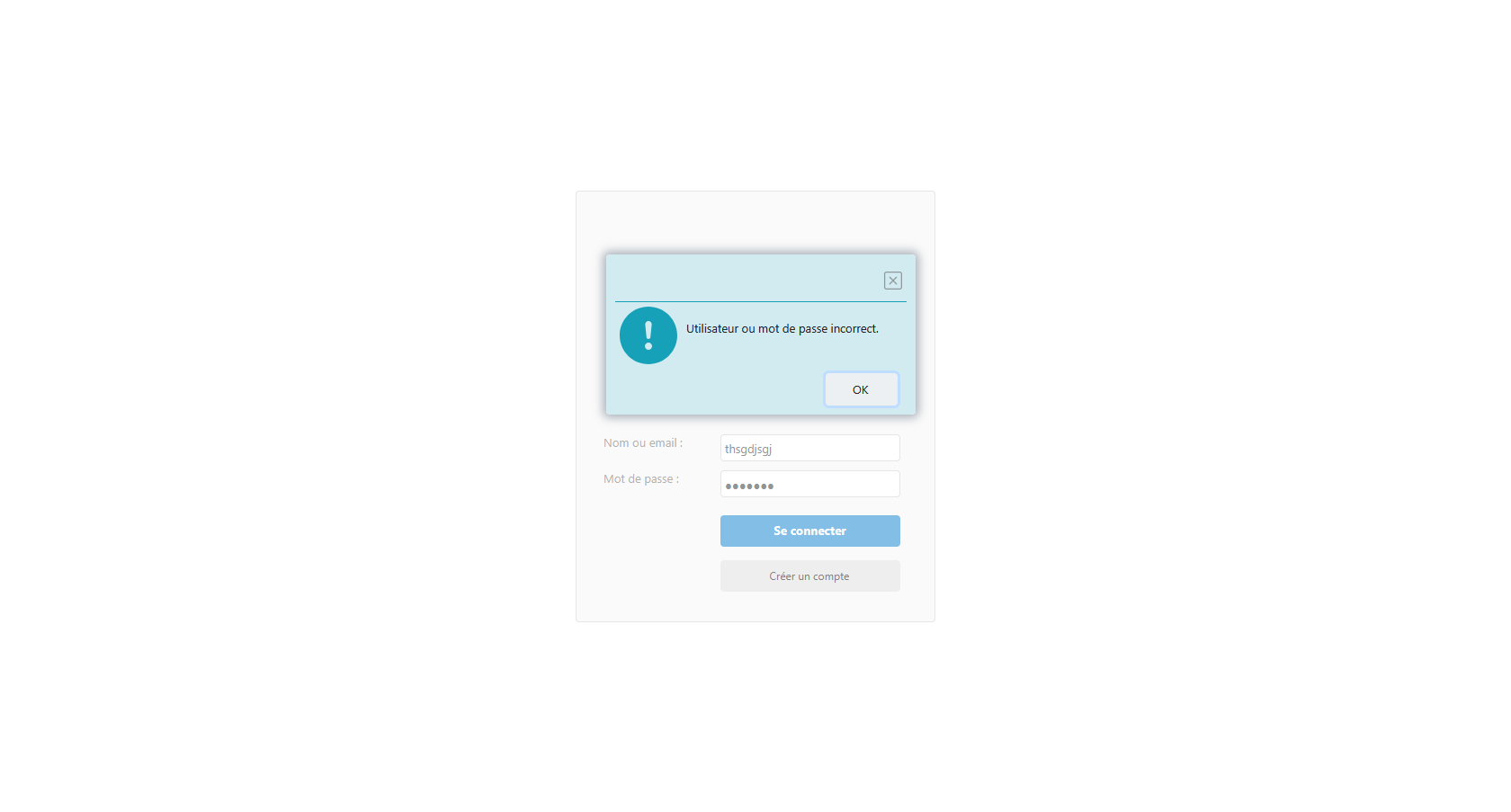


Figure 16 : Connexion avec un compte invalide car l'utilisateur n'existe pas

##### Création d’un compte valide

Condition de test : L'utilisateur entre un nom, email et mot de passe valides.

Résultat attendu : Compte créé + mail envoyé pour validation.

Preuve :

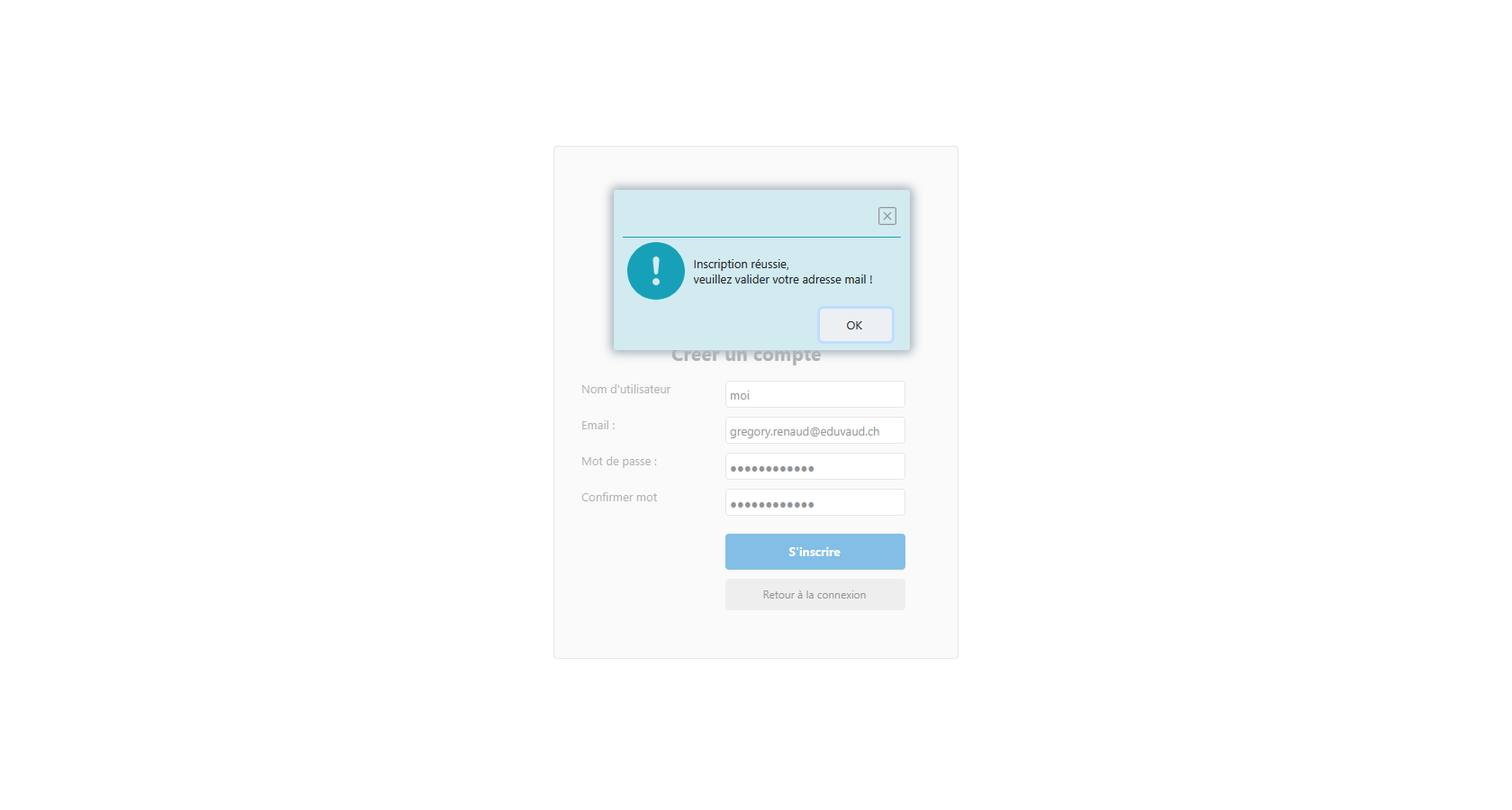


Figure 17 : Création d'un nouvel utilisateur valide

##### Création d’un compte invalide (mauvais domaine)

Condition de test : L’utilisateur tente de créer un compte avec un email n’appartenant pas aux domaines autorisés.

Résultat attendu : Message d’erreur « Veuillez entrer une adresse email valide ».

Preuve :

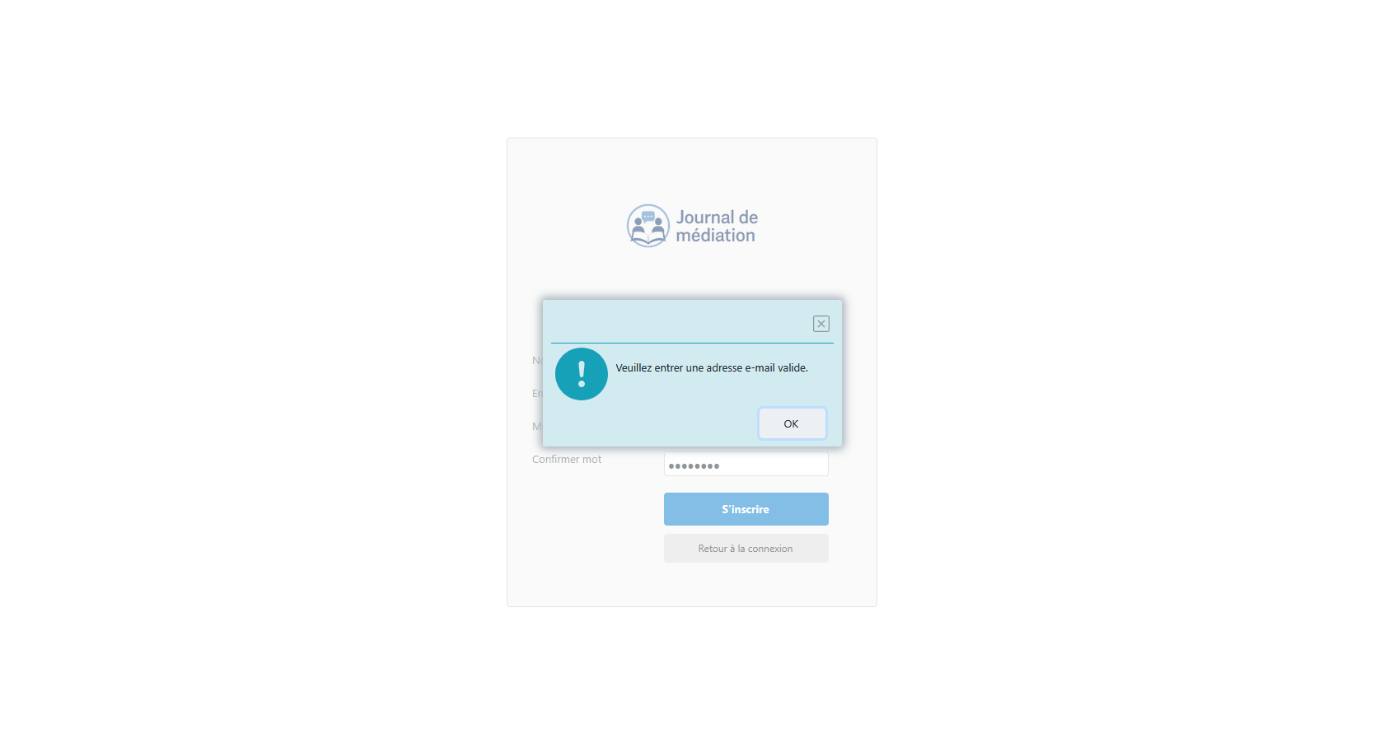


Figure 18 : Création d'un compte invalide car le domaine email n'est pas valide

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

Condition de test : L’utilisateur tente de créer un compte avec une adresse email ou un nom d’utilisateur déjà existant.

Résultat attendu : Message d’erreur « Cet utilisateur existe déjà »

Preuve :

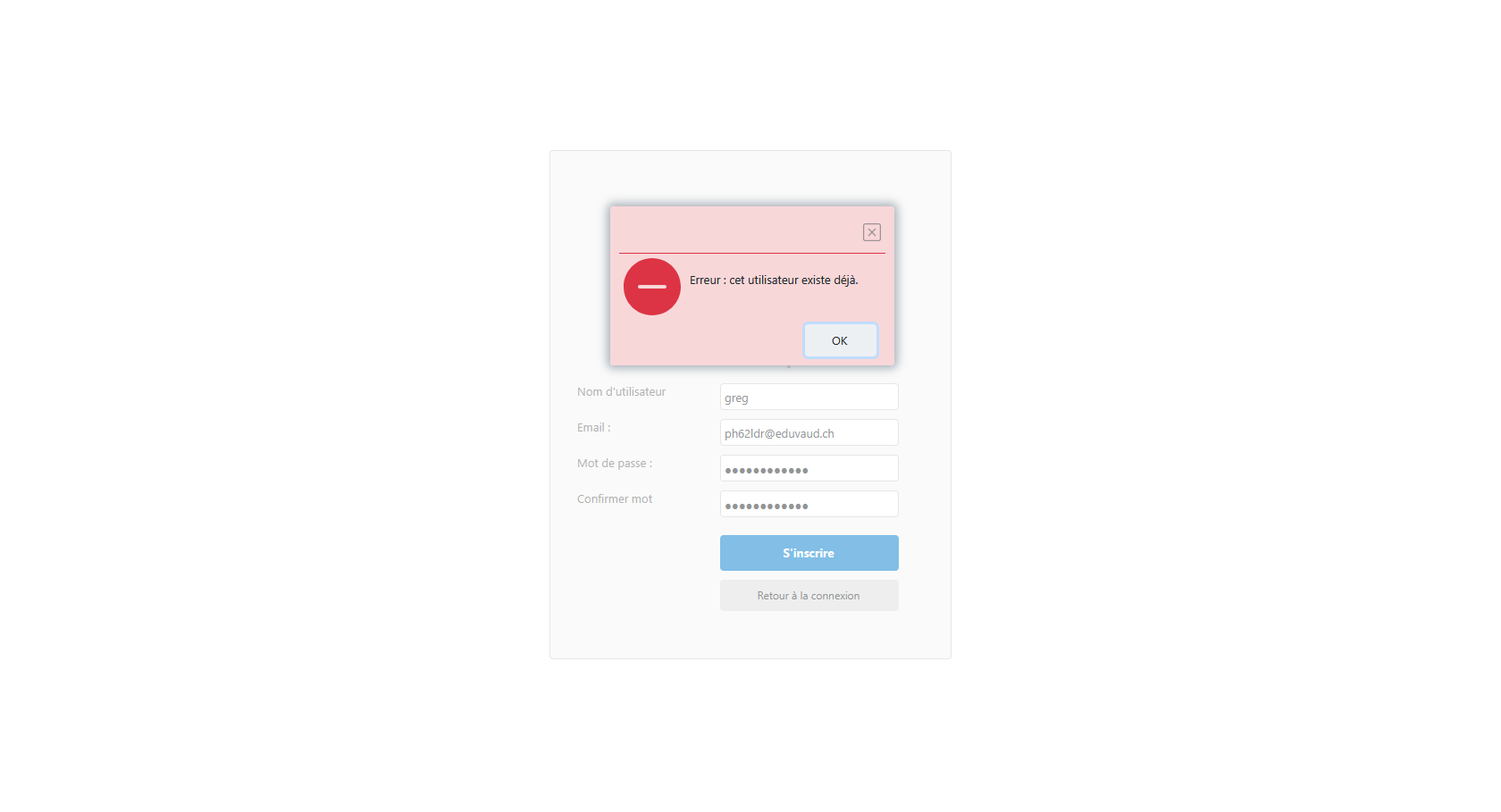


Figure 19 : Création d'un compte invalide car un autre utilisateur possède déjà le même email ou nom

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

Condition de test : L’utilisateur entre un mot de passe trop faible (>= à 8 caractères, 1 majuscule et 1 nombre).

Résultat attendu : Message d’erreur « Le mot de passe doit contenir au minimum 8 caractères, une majuscule et un chiffre ».

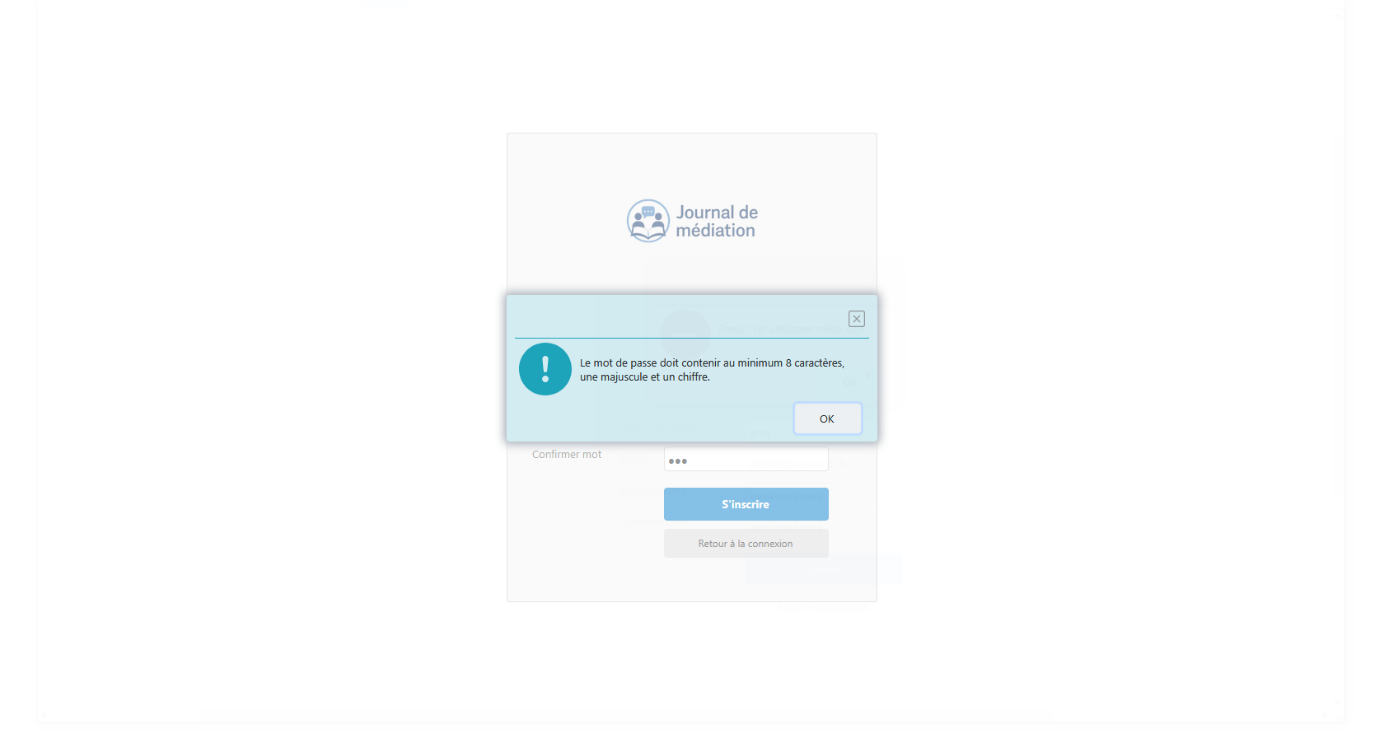
Preuve :

Figure 20 : Création d'un compte invalide car le mot de passe est trop faible

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

Condition de test : L’utilisateur n’entre pas les mêmes mots de passe entre les champs mot de passe et confirmation de mot de passe.

Résultat attendu : Message d’erreur : Message d’erreur « Les mots de passe ne correspondent pas ».

Preuve :

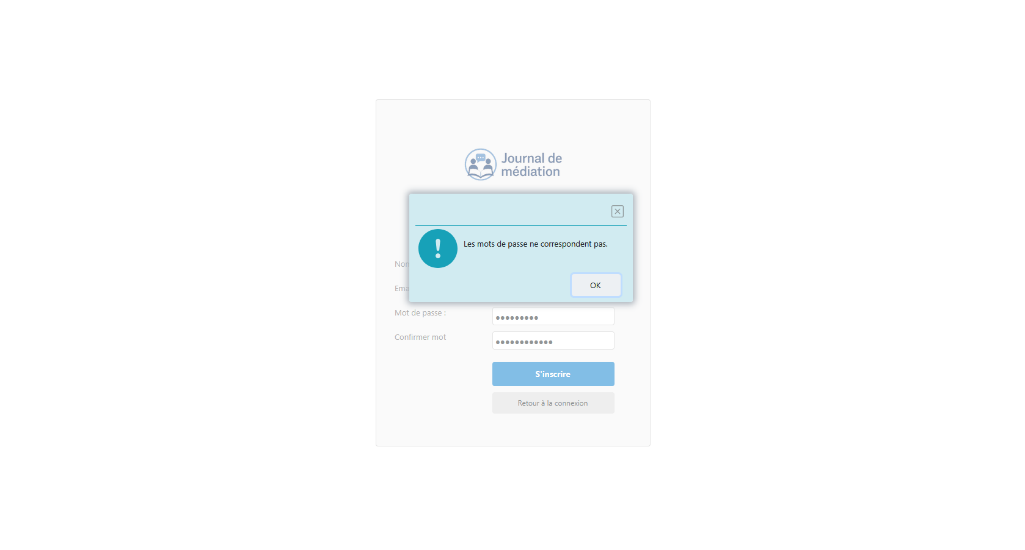


Figure 21 : Création d'un compte invalide car les mots de passe ne correspondent pas entre eux

##### Validation du compte par mail

Condition de test : L’utilisateur clique sur le lien reçu par mail après la création d’un compte.

Résultat attendu : Le lien redirige sur la page de validation du compte. Lorsque l’utilisateur clique sur le bouton, son compte est validé.

Preuve :

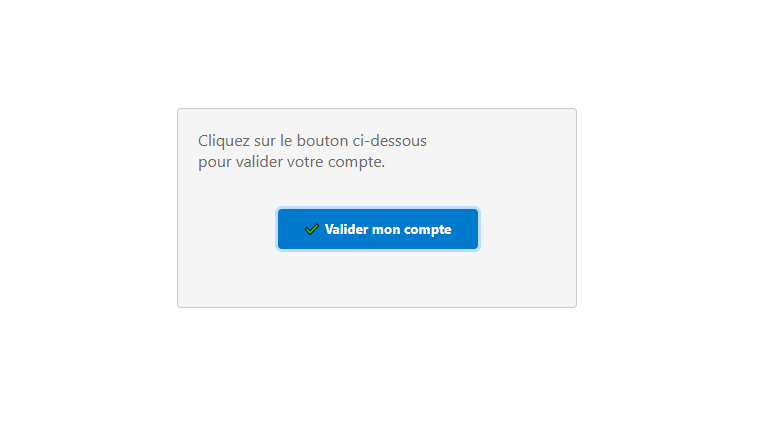


Figure 22 : Validation du compte par mail partie 1

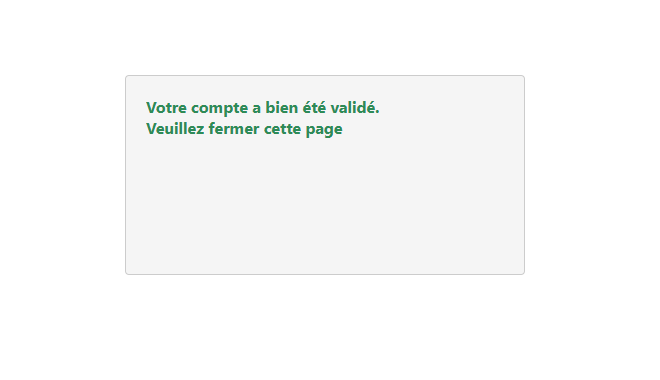


Figure 23 : Validation par mail partie 2

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

Condition de test : L’utilisateur tente de falsifier le token en entrant un token invalide dans l’url.

Résultat attendu : Message d’erreur « Token invalide ou expiré ».

Preuve :

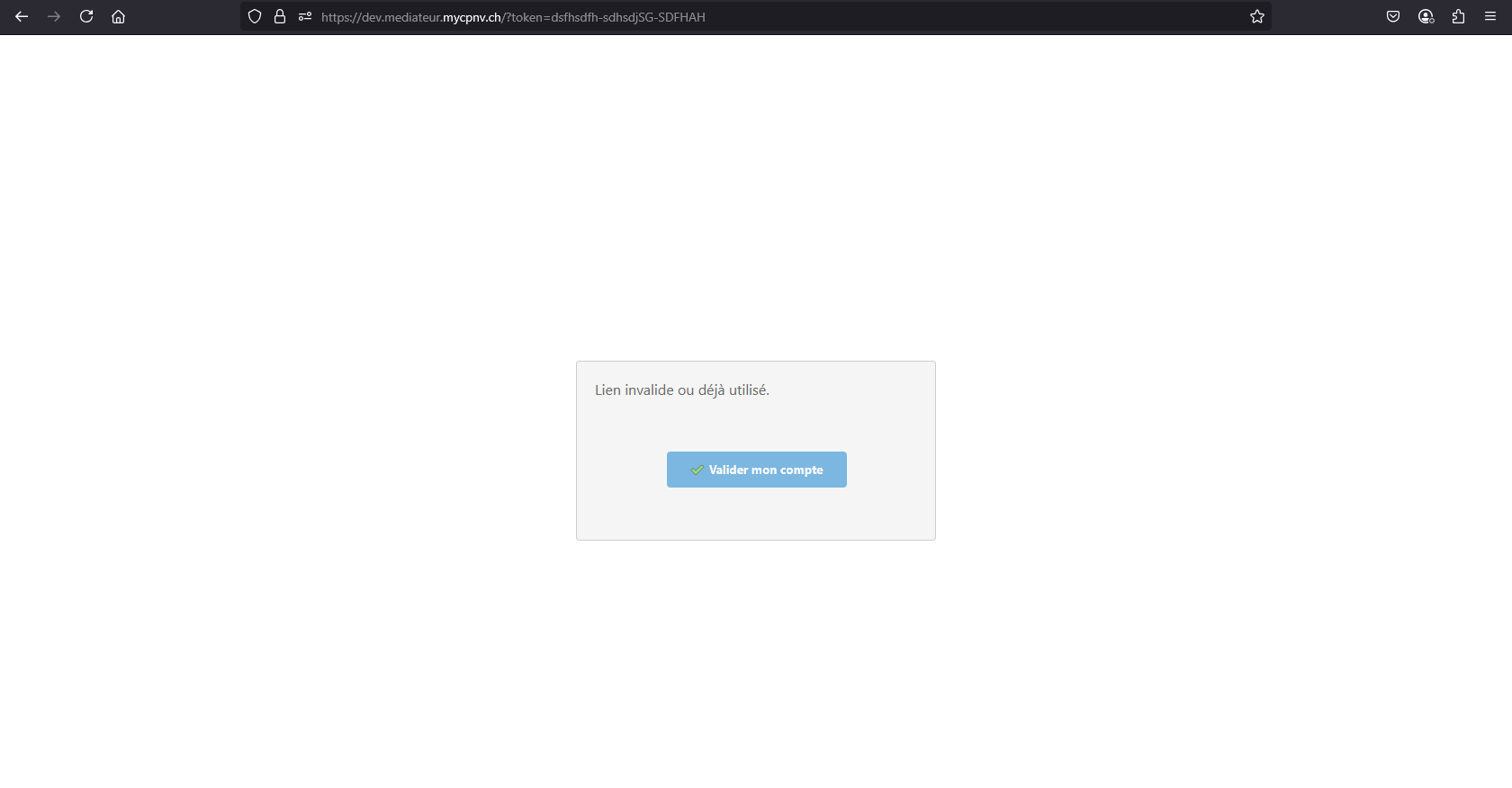


Figure 24 : Validation du compte avec un token invalide ou expiré

##### Déconnexion de l’utilisateur

Condition de test : L’utilisateur est connecté dans l’application et clique sur le bouton de déconnexion.

Résultat attendu : Un message de confirmation apparait et demande si l’utilisateur veut se déconnecter. S’il clique sur oui, il est redirigé sur la page de login. Si non, rien ne se passe.

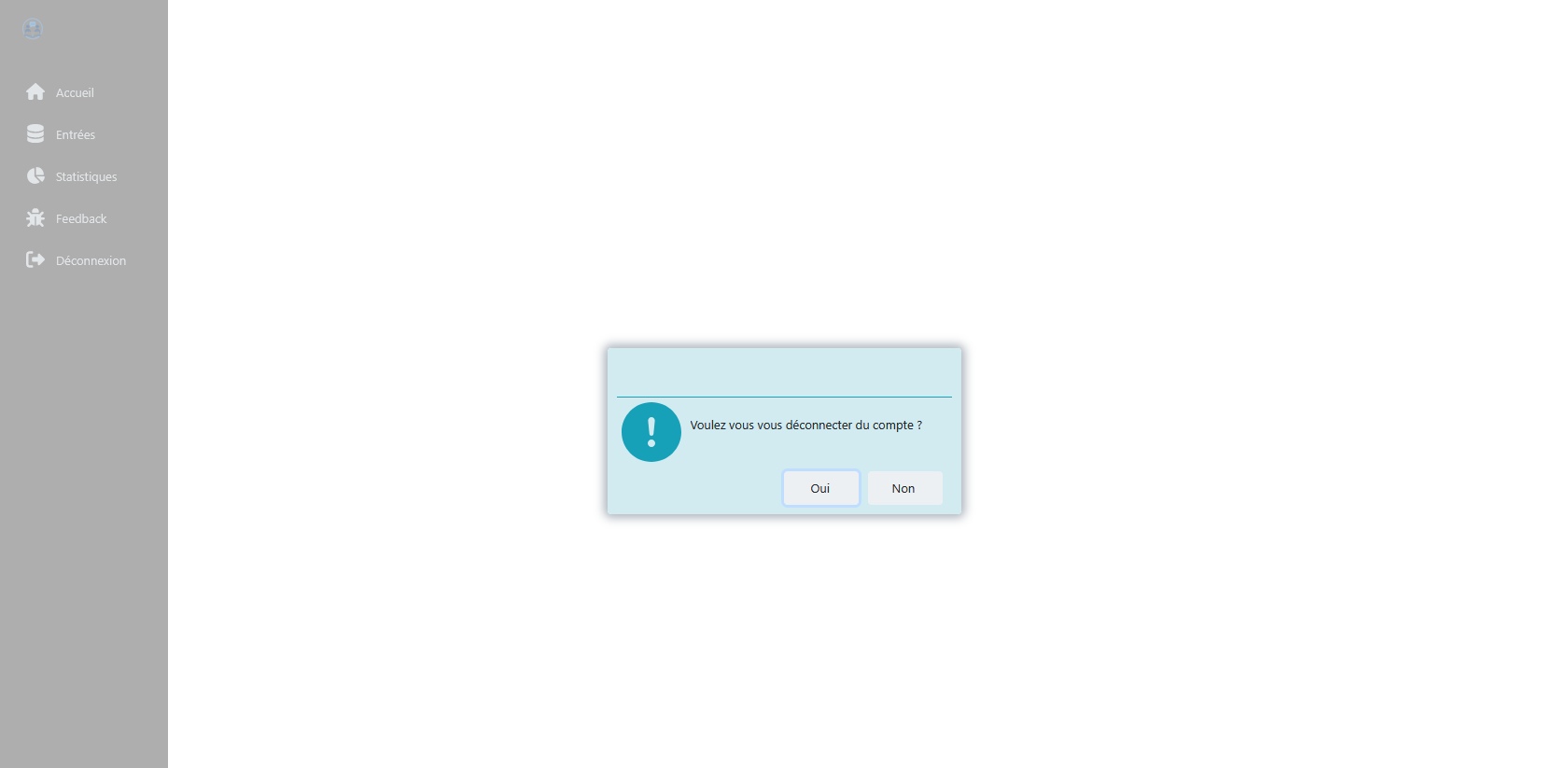
Preuve :

Figure 25 : Déconnexion de l'utilisateur partie 1

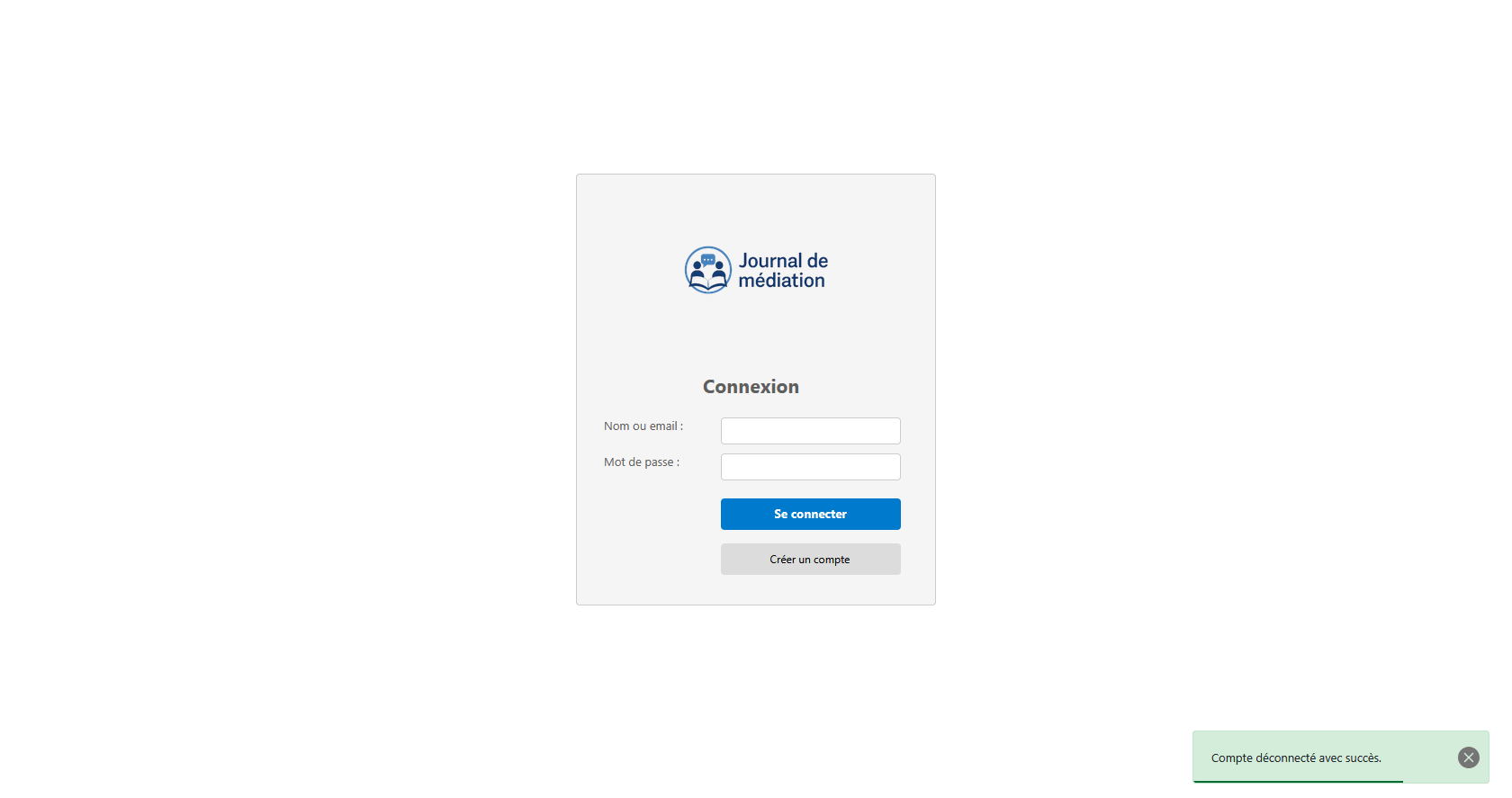


Figure 26 : Déconnexion de l'utilisateur partie 2

#### Affichage des données

##### Chargement correct de la grille

Condition de test : L’utilisateur est connecté et possède au moins 1 donnée entrée dans l’année courante.

Résultat attendu : Les données de l’année courante s’affichent correctement dans la grille.

Preuve :

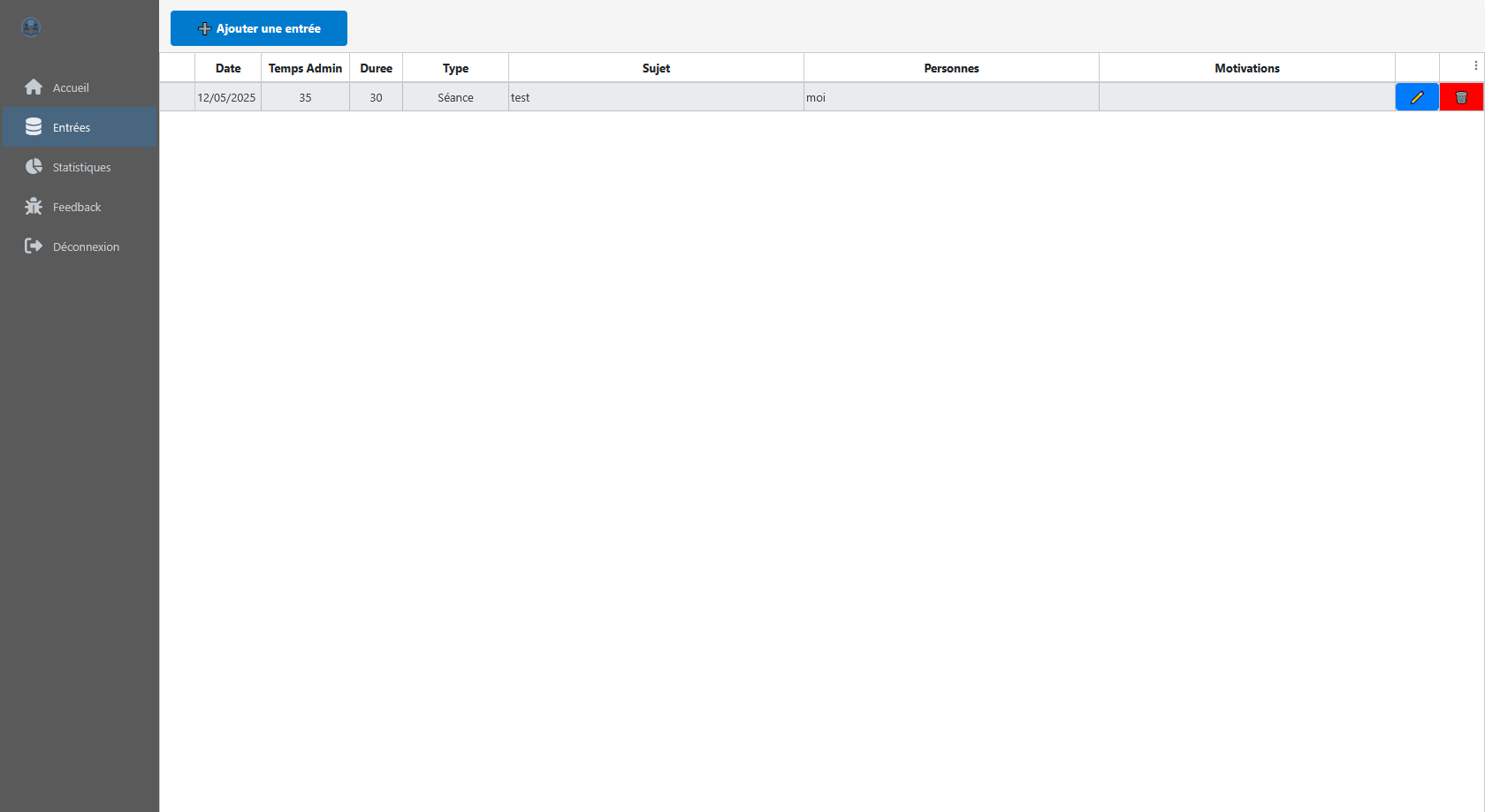


Figure 27 : Chargement correct de la grille de données

##### Suppression d’un évènement

Condition de test : L’utilisateur clique sur le bouton de suppression d’un évènement.

Résultat attendu : Un message de confirmation apparait et si l’utilisateur clique sur oui, l’évènement est supprimé. Sinon, il ne se supprime pas.

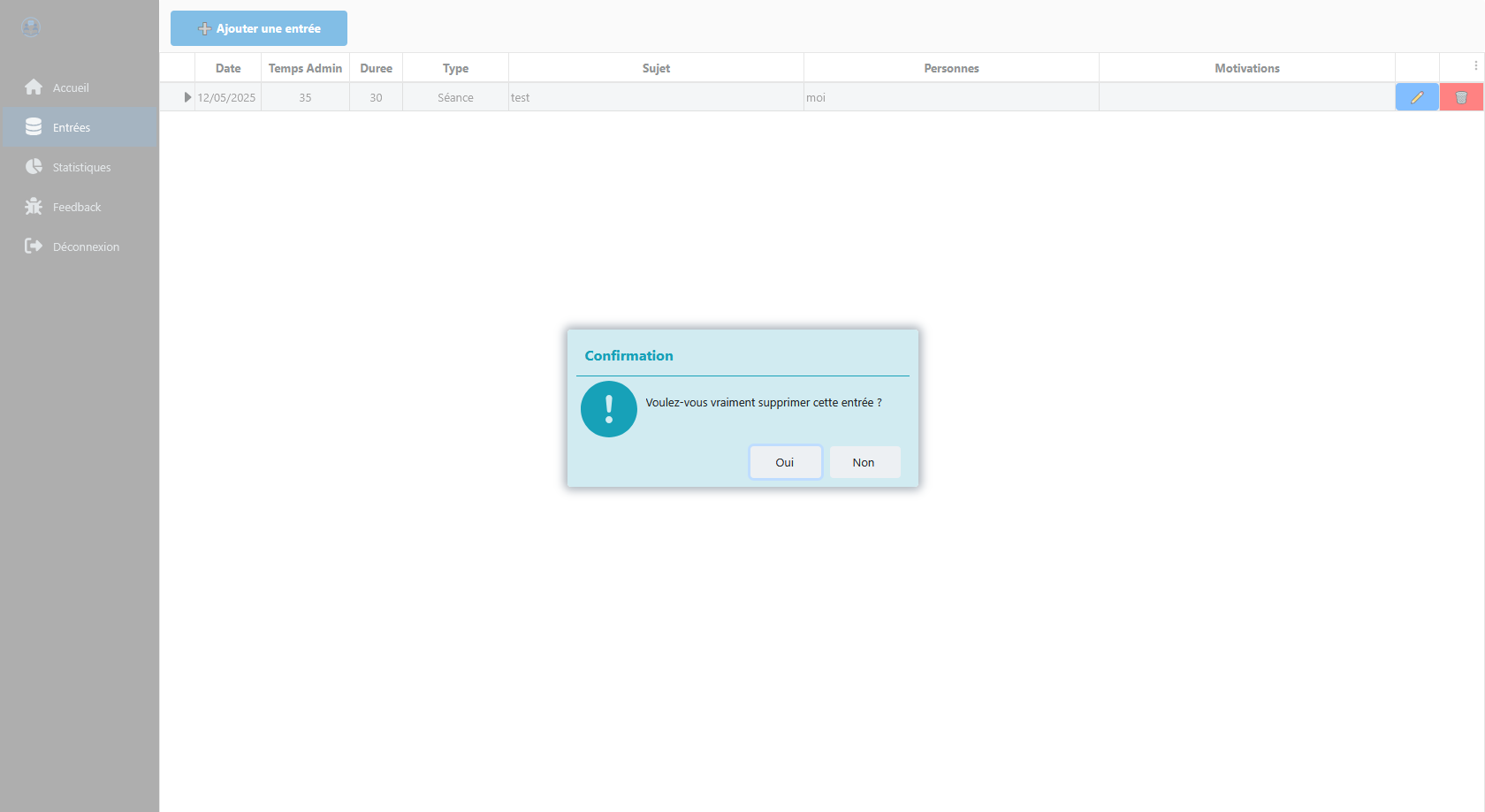
Preuve :

Figure 28 : Suppression d'un évènement partie 1

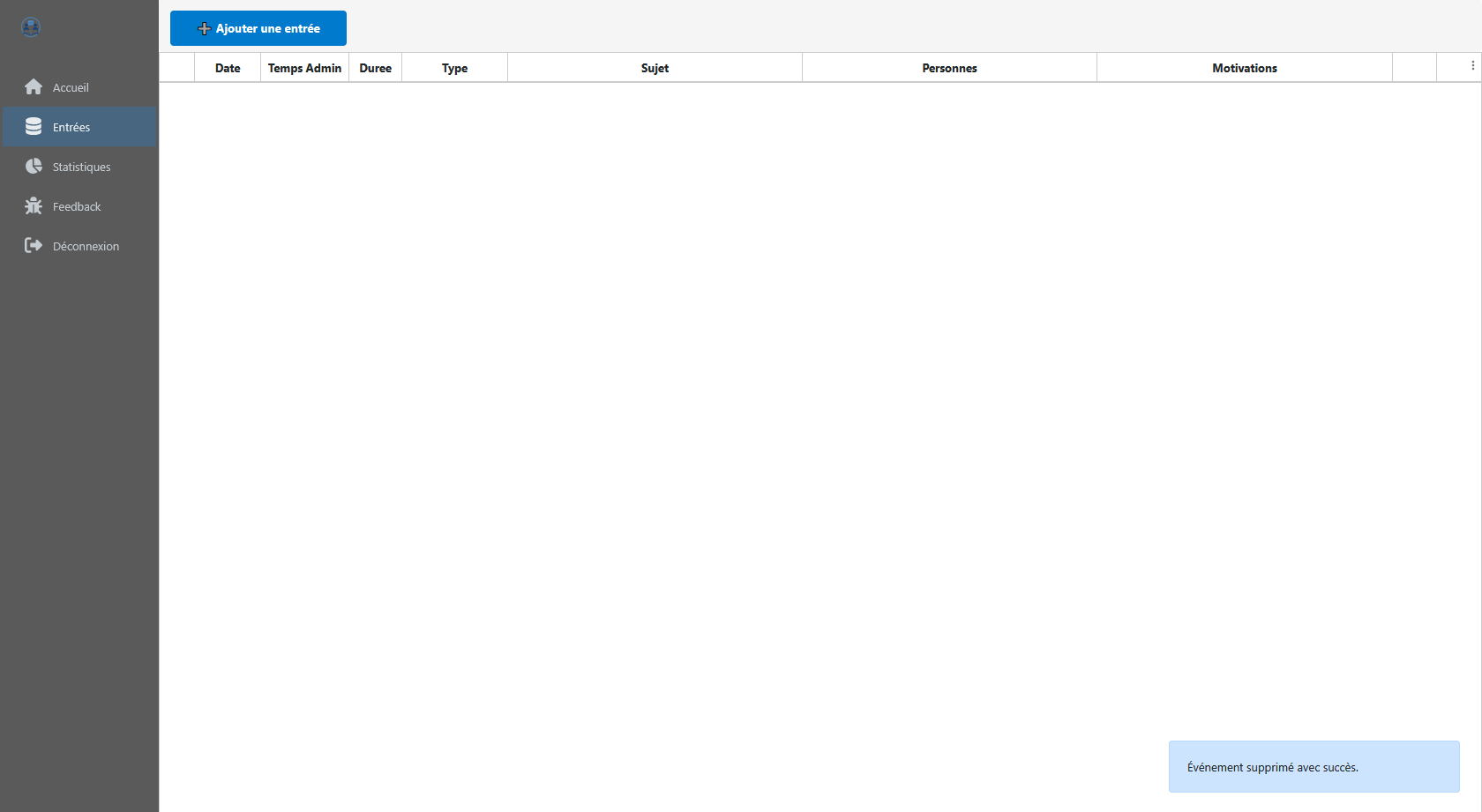


Figure 29 : Suppression d'un évènement partie 2

##### Edition d’un évènement

Condition de test : L’utilisateur clique sur le bouton d’édition d’un évènement.

Résultat attendu : Le formulaire d’édition de l’évènement s’ouvre avec les données préremplies. Une fois éditée, la sauvegarde modifie l’évènement et les nouvelles données sont visible dans la grille.

Preuve :

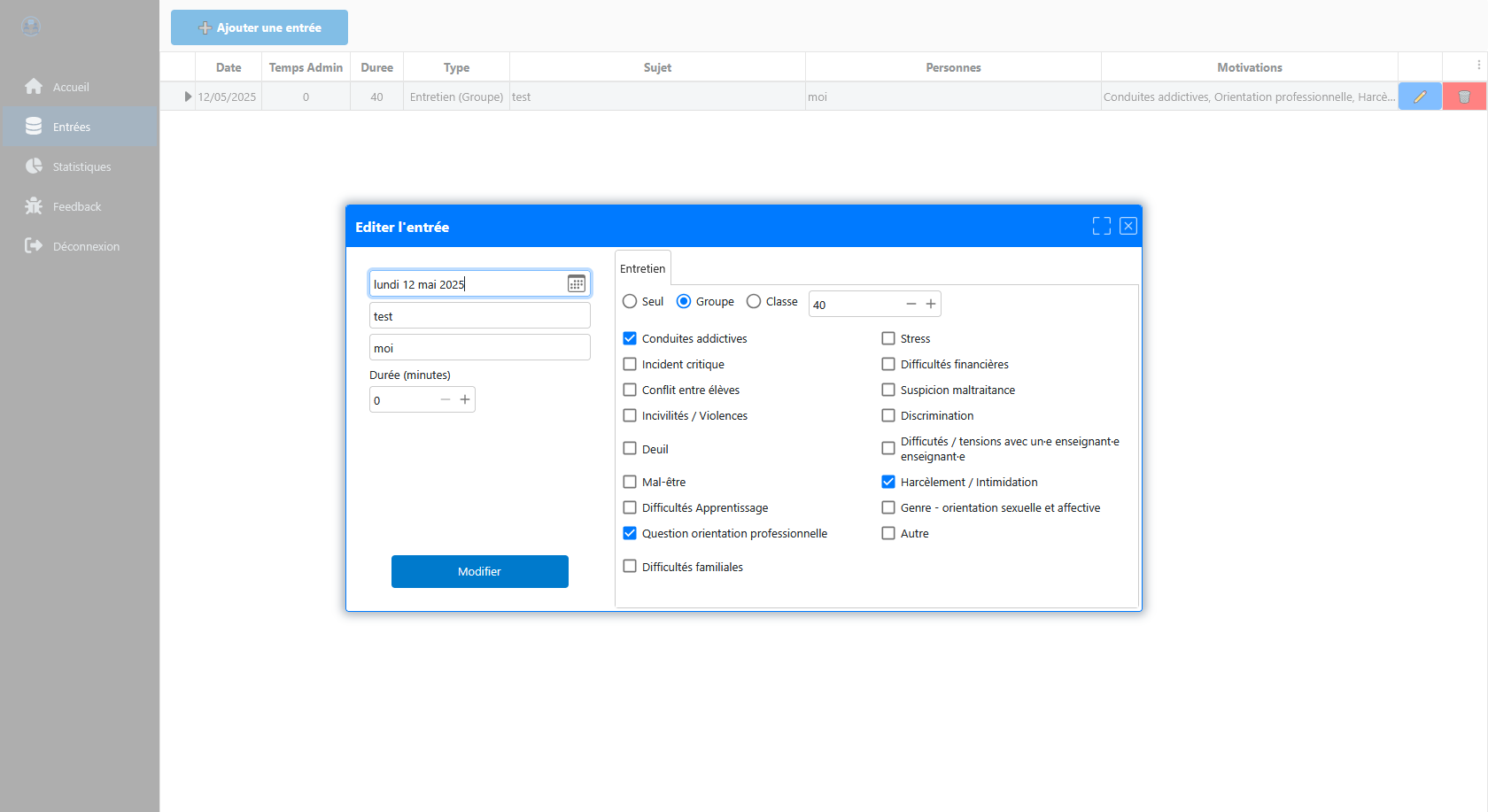


Figure 30 : Edition d'un évènement partie 1

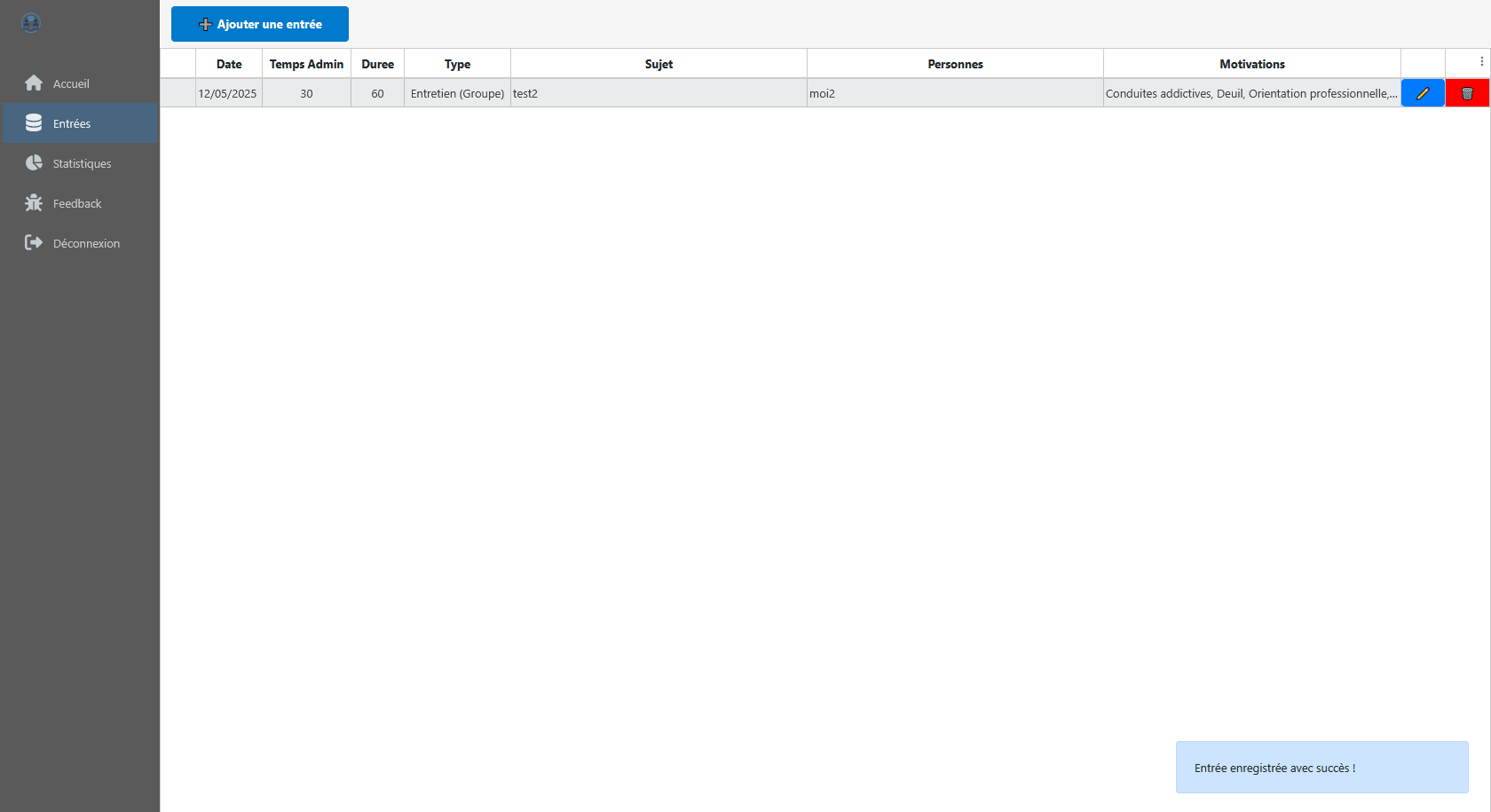


Figure 31 : Edition d'un évènement partie 2

#### Ajout d’évènement

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

Condition de test : L’utilisateur entre une date hors de l’année scolaire en cours (1er août au 31 juillet).

Résultat attendu : Message d’erreur « La date doit être comprise dans l’année scolaire en cours » et remet la sélection de la date au jour courant.

Preuve :

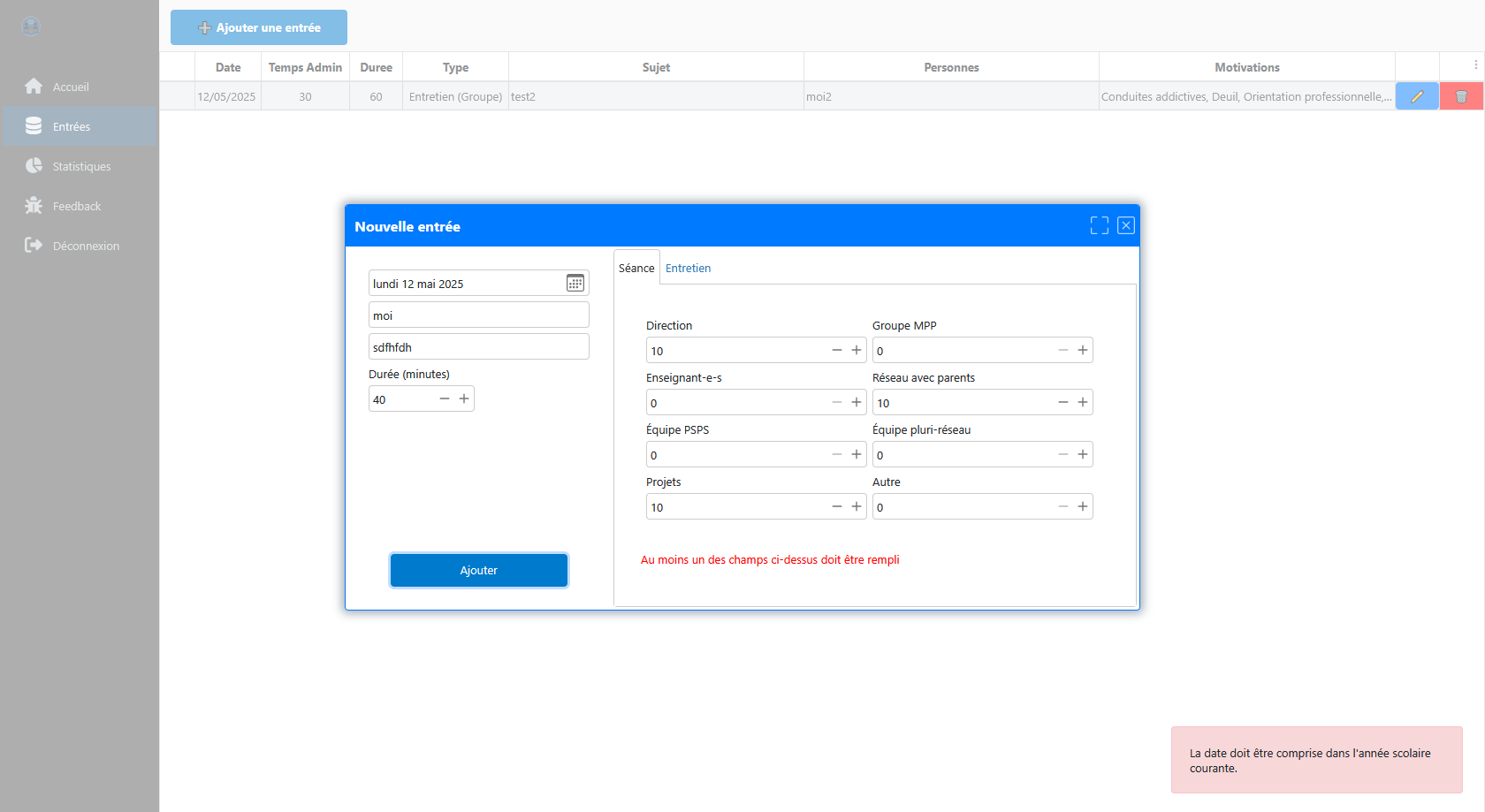


Figure 32 : Ajout d'un évènement invalide car le champ date est hors année scolaire en cours

##### Ajout d’un évènement invalide (sujet vide)

Condition de test : L’utilisateur valide un évènement sans « Sujet ».

Résultat attendu : Message d’erreur « Veuillez entrer un sujet ».

Preuve :

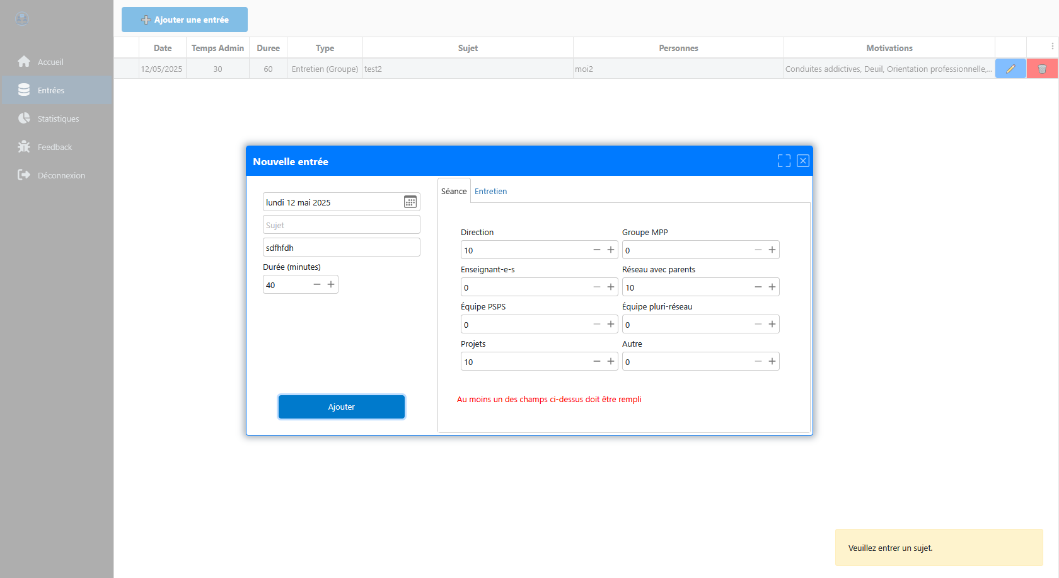


Figure 33 : Ajout d'un évènement invalide car le sujet est vide

##### Ajout d’un évènement invalide (personnes concernées vide)

Condition de test : L’utilisateur valide un évènement sans « Personnes concernées ».

Résultat attendu : Message d’erreur « Veuillez entrer des personnes ».

Preuve :

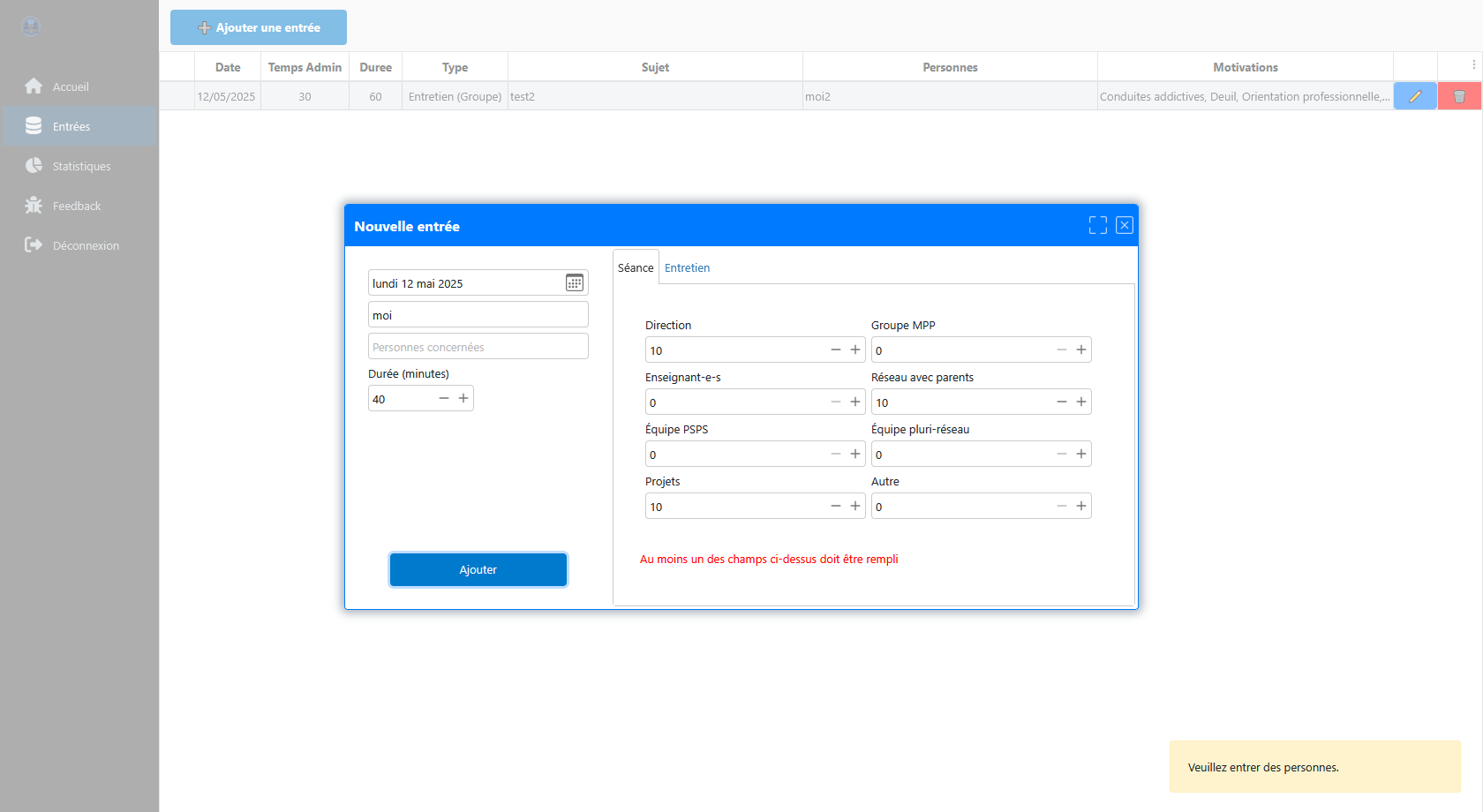


Figure 34 : Ajout d'un évènement invalide car le champ personnes concernées est vide

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

Condition de test : L’utilisateur valide une séance sans aucun temps de séance défini.

Résultat attendu : Message d’erreur « Veuillez enter au moins un temps de séance ».

Preuve :

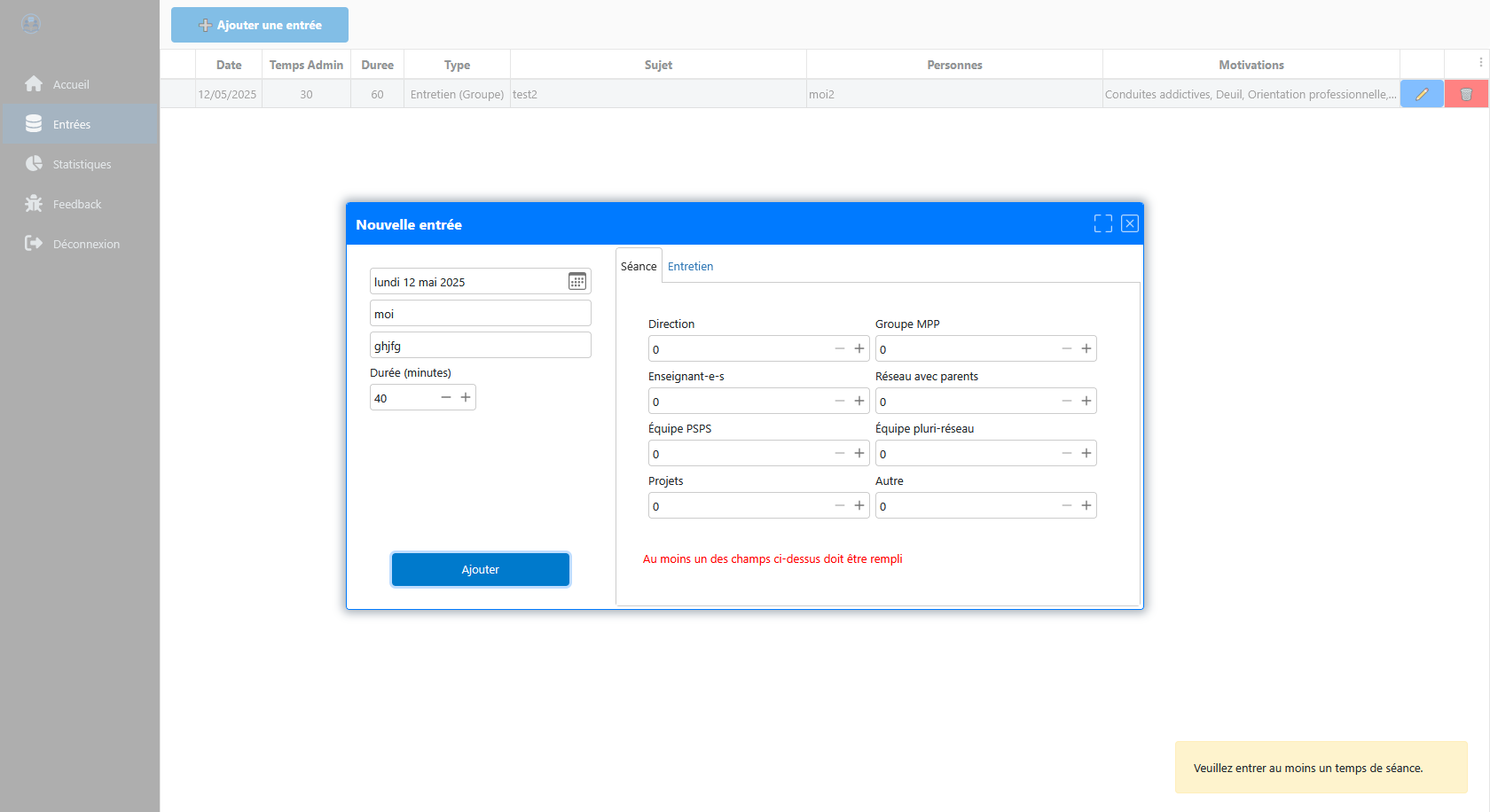


Figure 35 : Ajout d'une séance invalide car aucun champ de temps est remplis

##### Ajout d’une séance valide

Condition de test : L’utilisateur valide une séance avec tous les champs nécessaires remplis.

Résultat attendu : La nouvelle donnée est ajoutée au tableau.

Preuve :

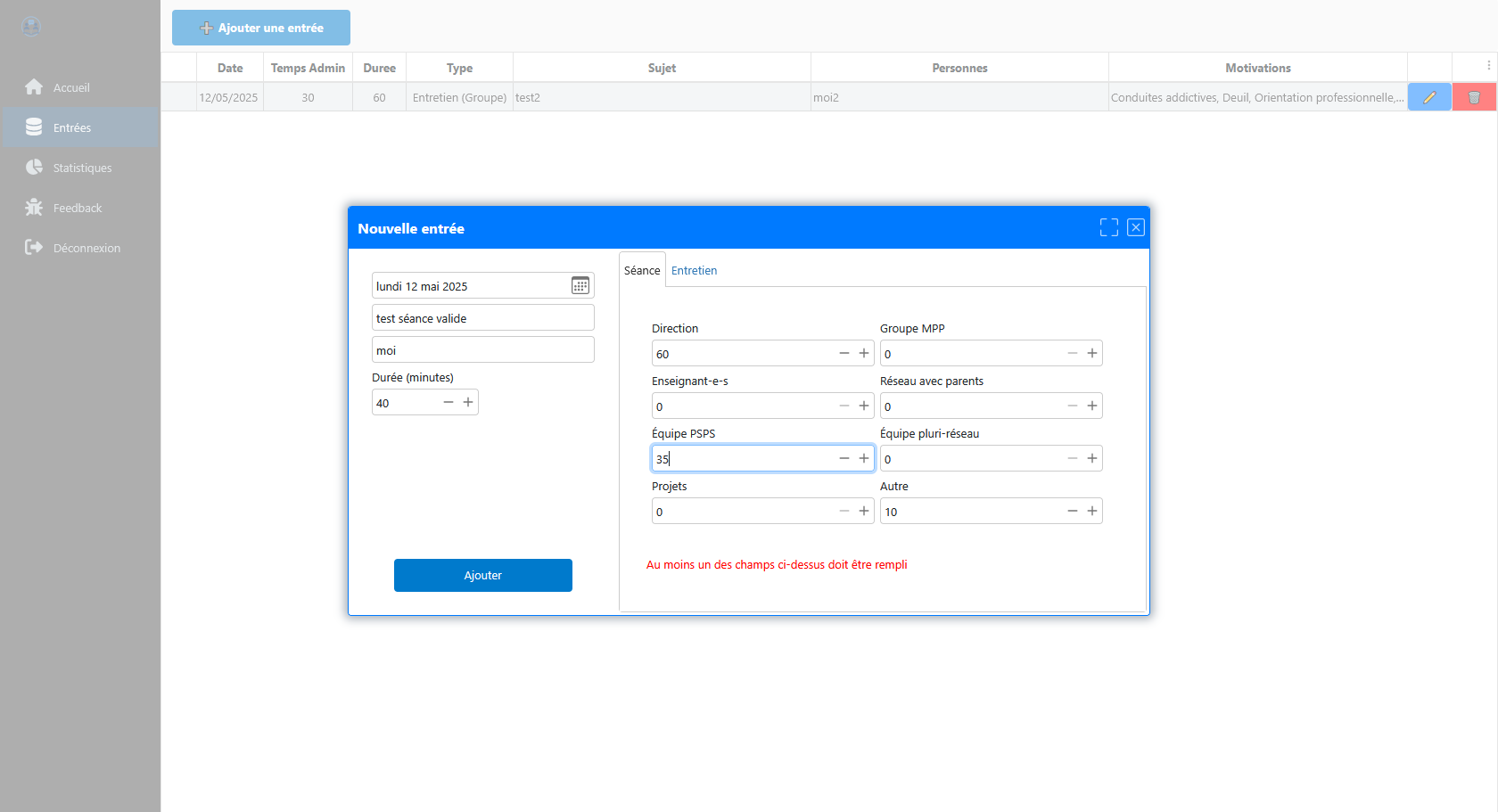


Figure 36 : Ajout d'une séance valide partie 1

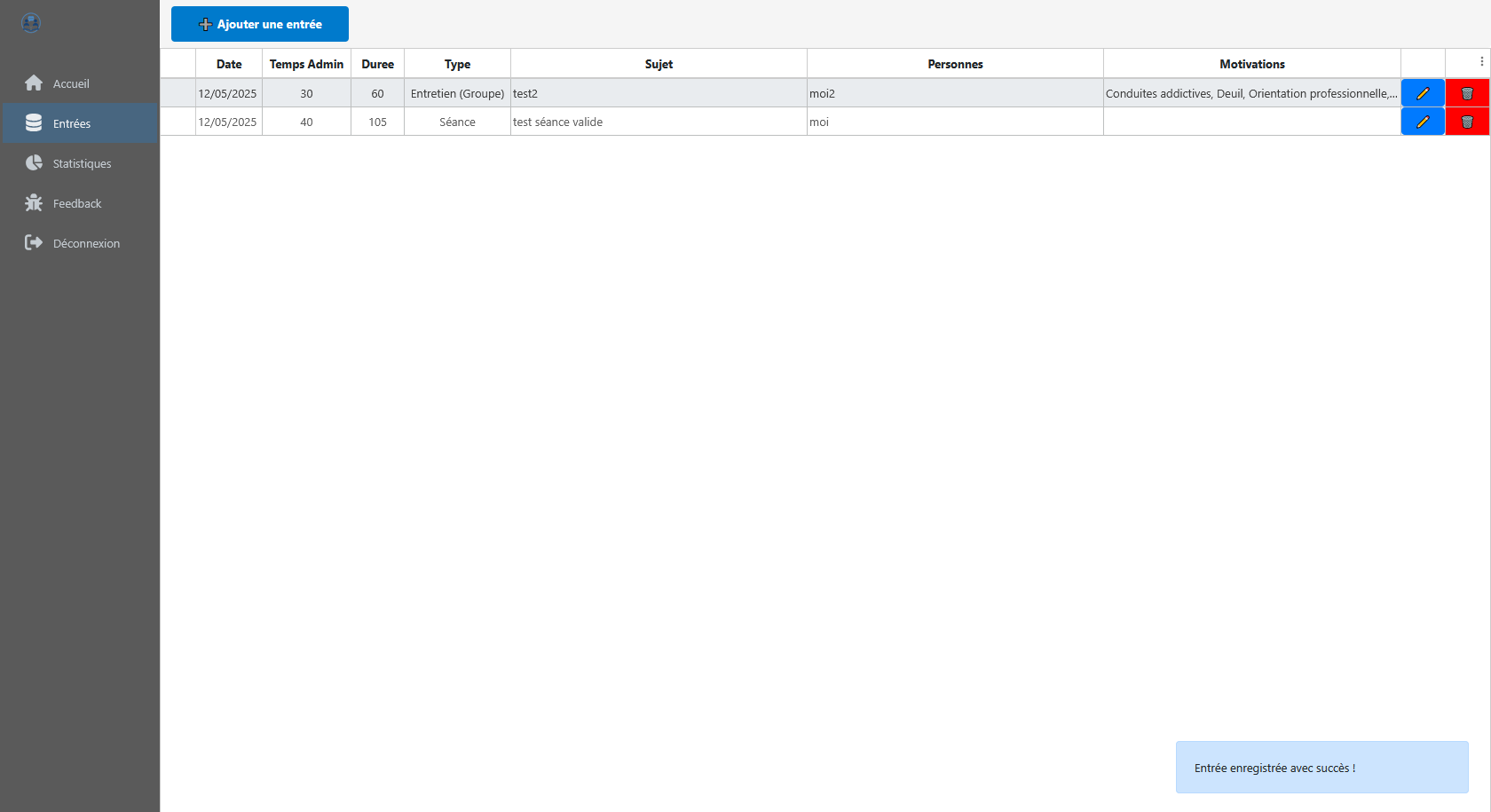


Figure 37 : Ajout d'une séance valide partie 2

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

Condition de test : L’utilisateur valide un entretien sans le type de séance renseignée.

Résultat attendu : Sélection par défaut d’un entretien « Seul ».

Preuve :

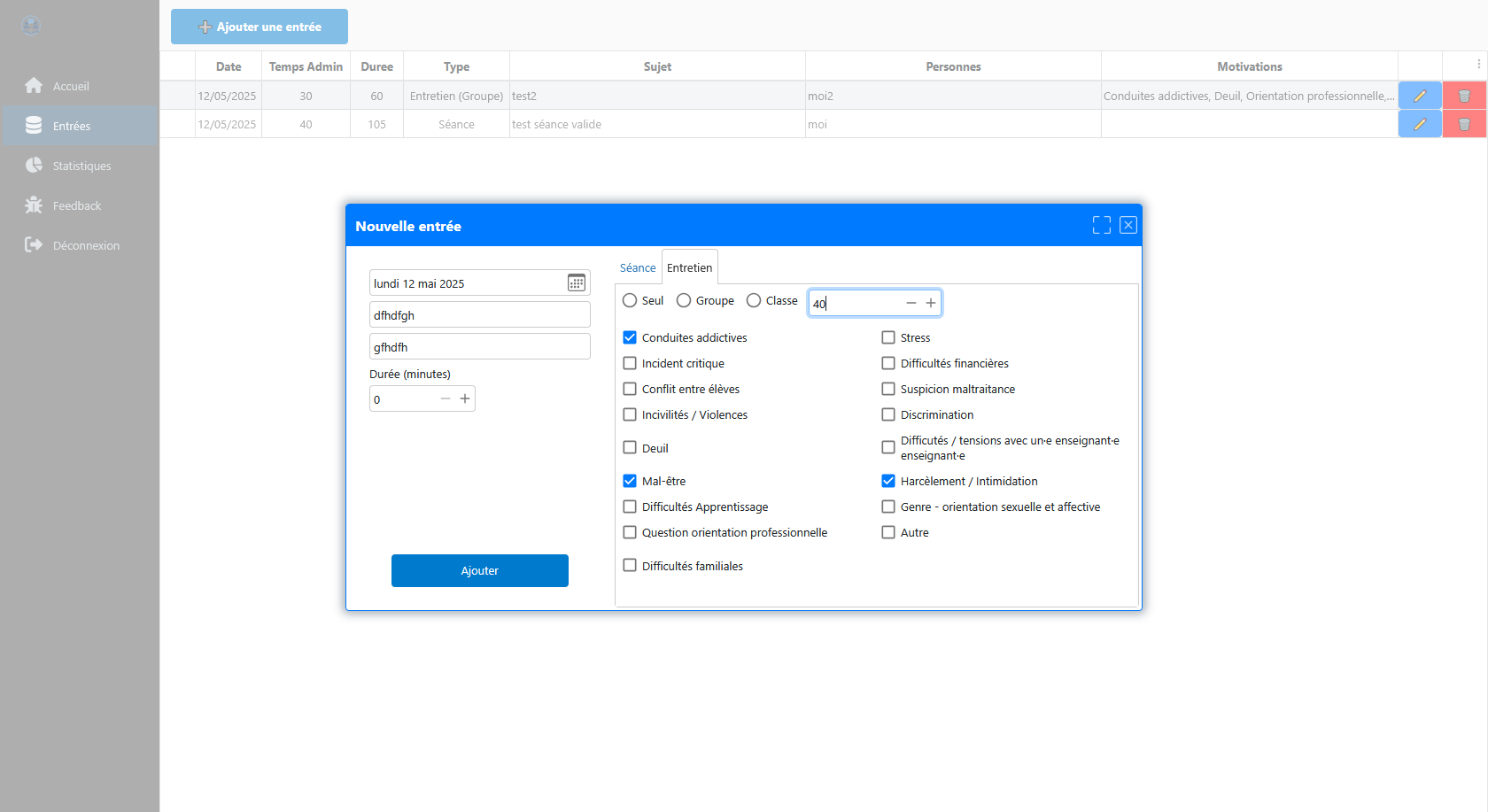


Figure 38 : Ajout d'un entretien invalide car aucun type de séance n'est renseigné partie 1

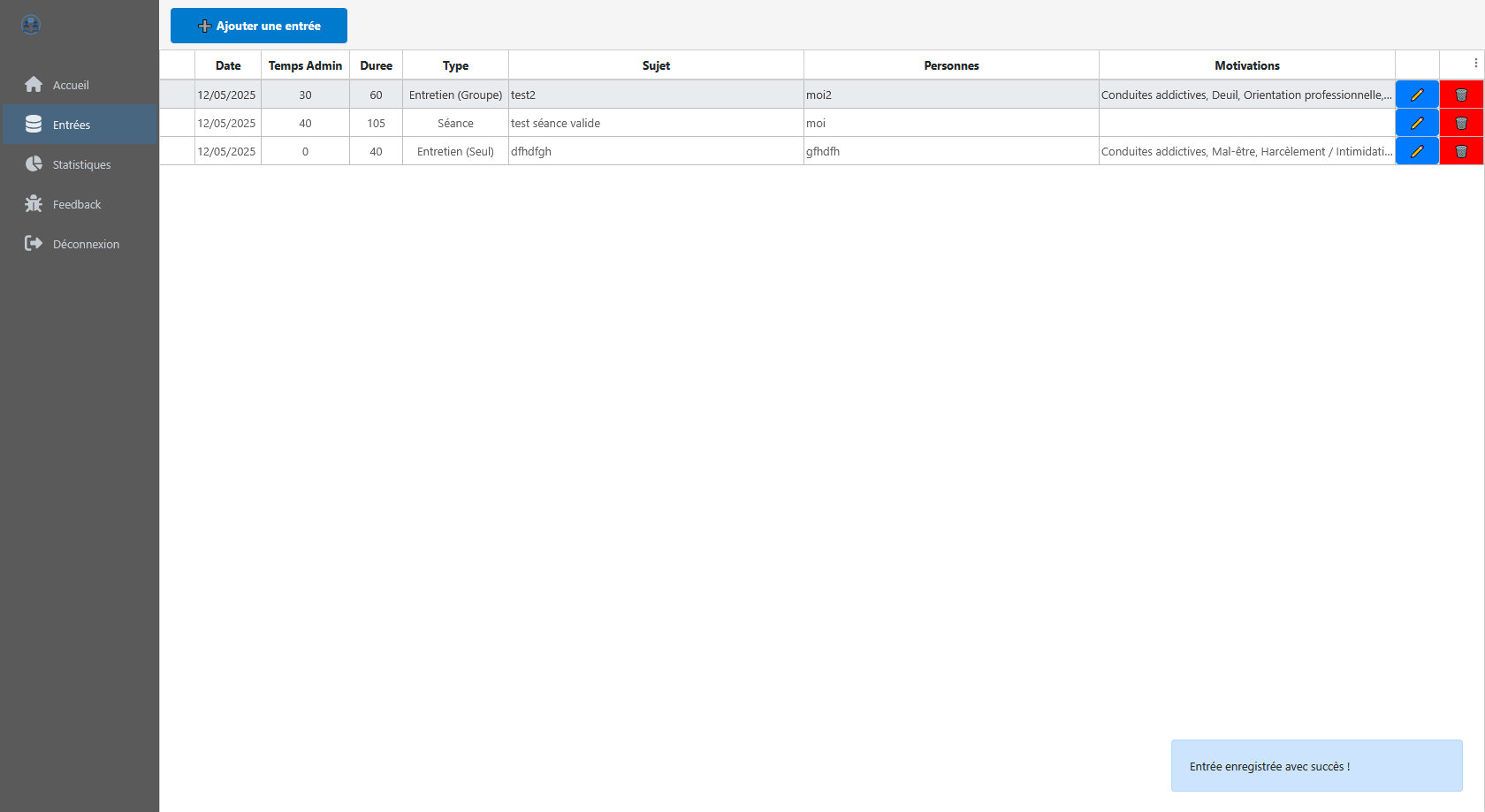


Figure 39 : Ajout d'un entretien invalide car aucun type de séance n'est renseigné partie 2

##### Ajout d’un entretien invalide (aucune motivation renseignée)

Condition de test : L’utilisateur valide un entretien sans motivation renseignée.

Résultat attendu : Message d’erreur « Veuillez sélectionner au moins une motivation ».

Preuve :

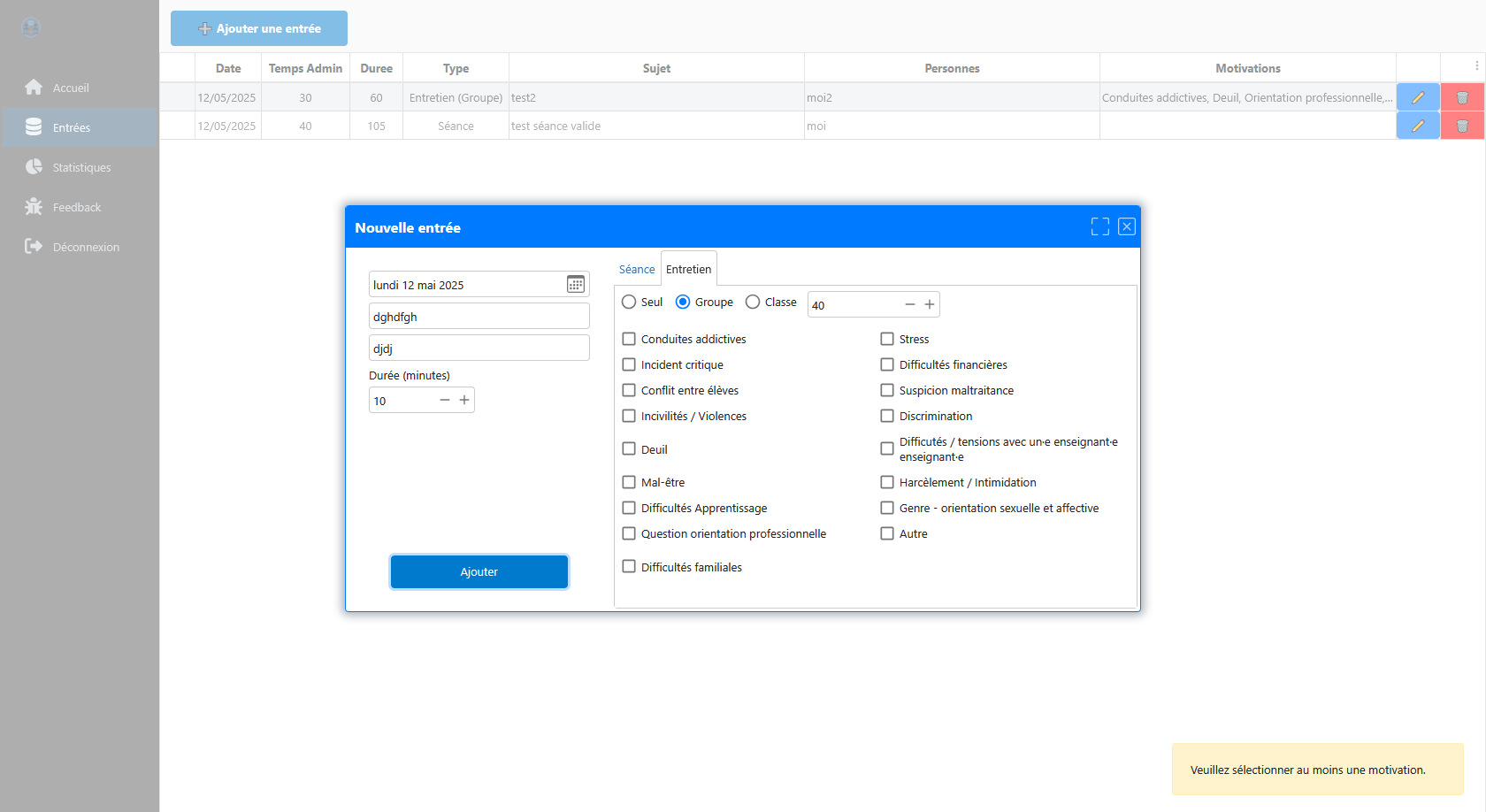


Figure 40 : Ajout d'un entretien invalide car aucune motivation n’est renseignée

##### Ajout d’un entretien valide

Condition de test : L’utilisateur valide un entretien avec tous les champs nécessaires remplis.

Résultat attendu : La nouvelle donnée est ajoutée au tableau.

Preuve :

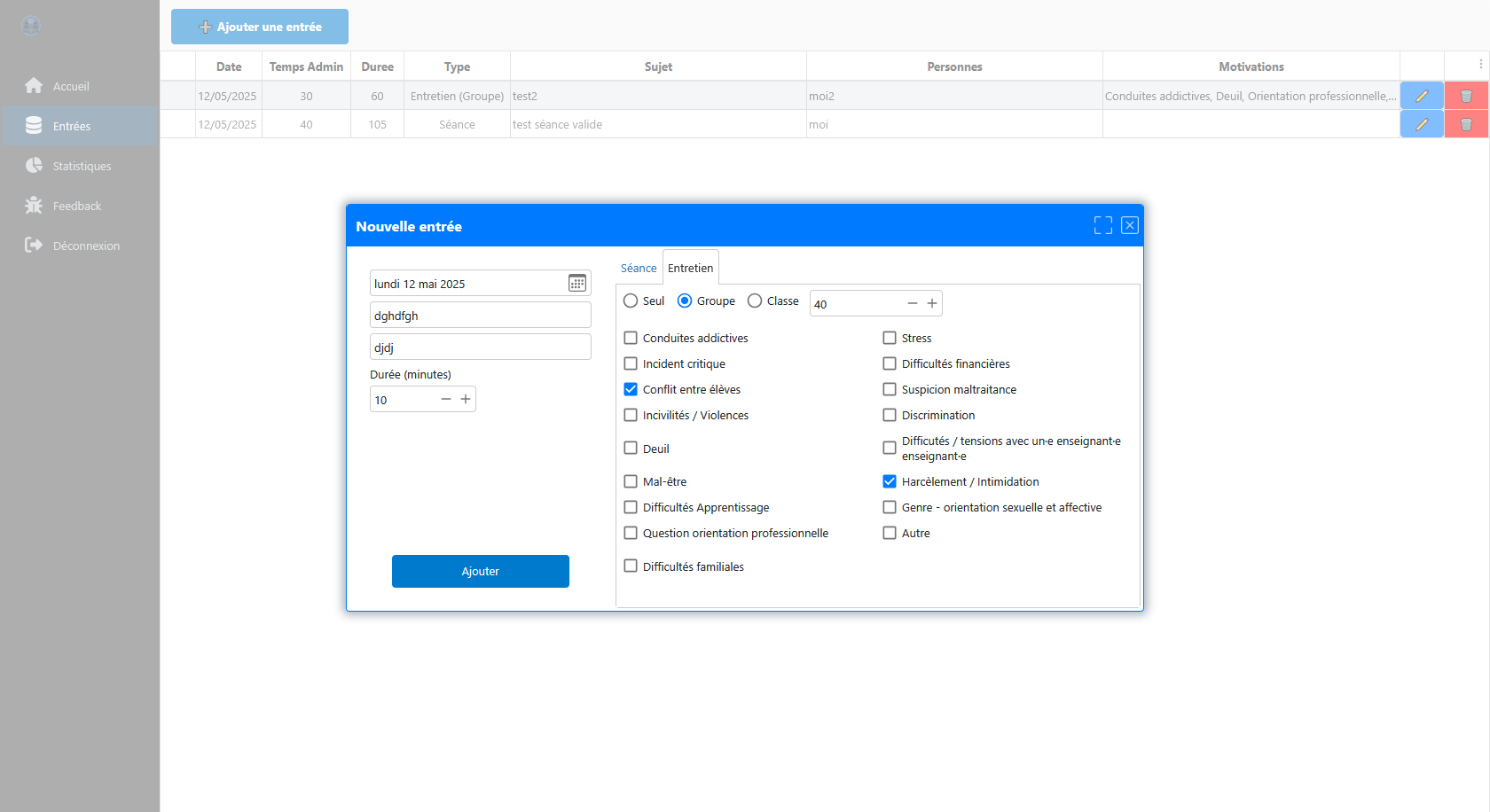


Figure 41 : Ajout d'un entretien valide partie 1

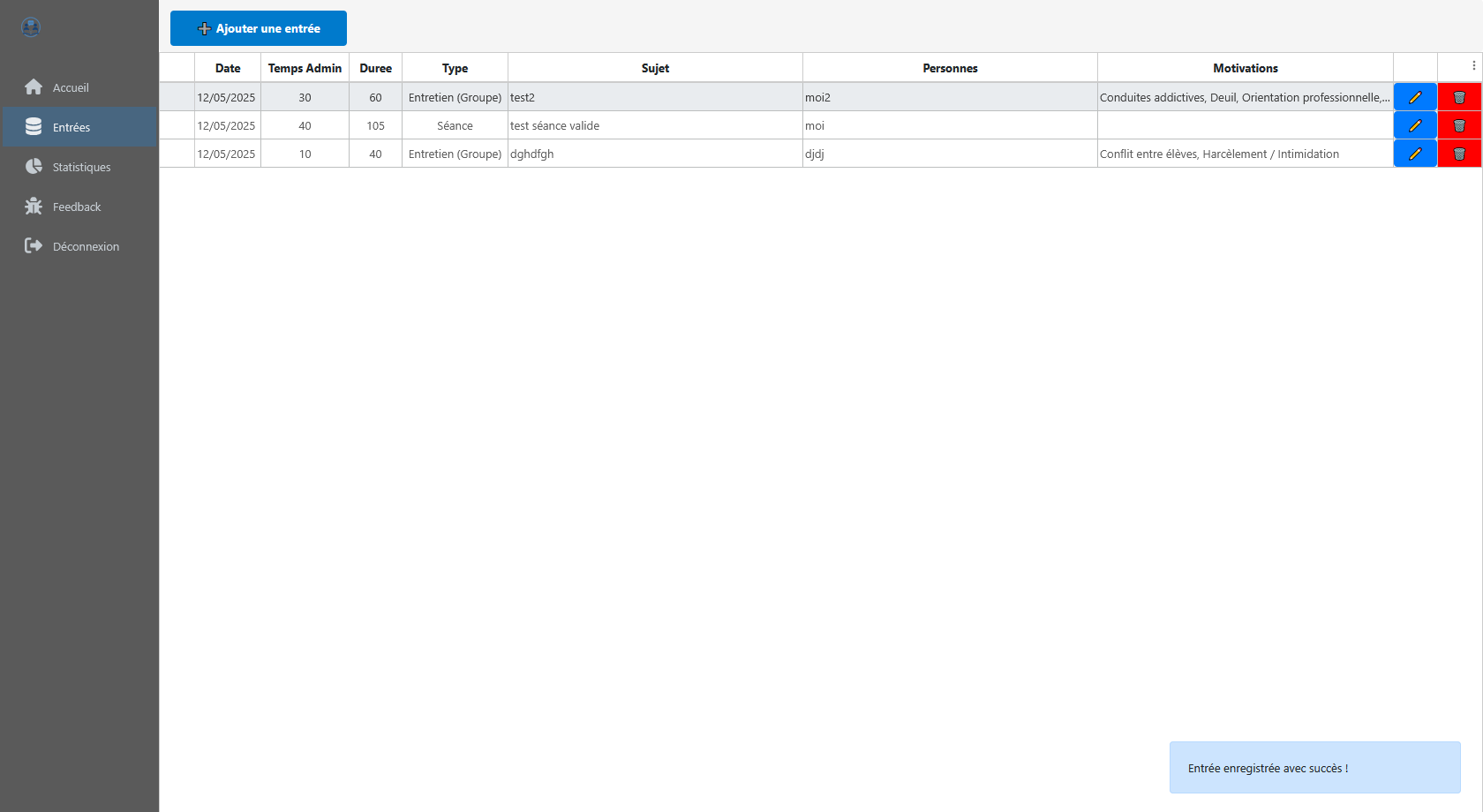


Figure 42 : Ajout d'un entretien valide partie 2

#### Statistiques

##### Affichage du graphique selon l’année définie

Condition de test : L’utilisateur sélectionne une année dans la liste.

Résultat attendu : Les graphiques sont mis à jour dynamiquement pour l’année sélectionnée.

Preuve :

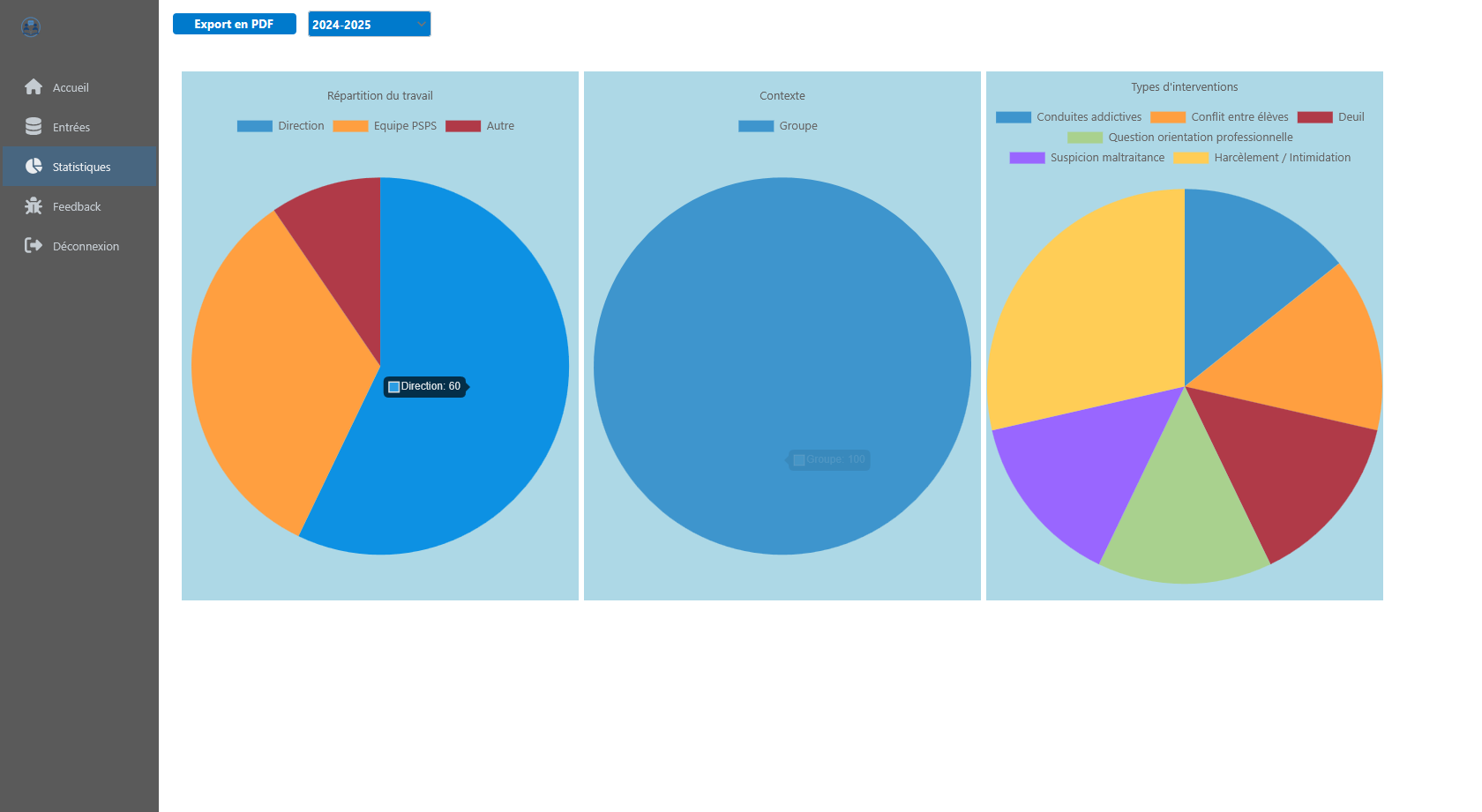


Figure 43 : Affichage des statistiques selon l'année définie

##### Comportement si aucune donnée

Condition de test : L’utilisateur clique sur la page de statistiques sans aucune donnée de sa part.

Résultat attendu : Message d’erreur « Aucune année scolaire disponible ».

Preuve :

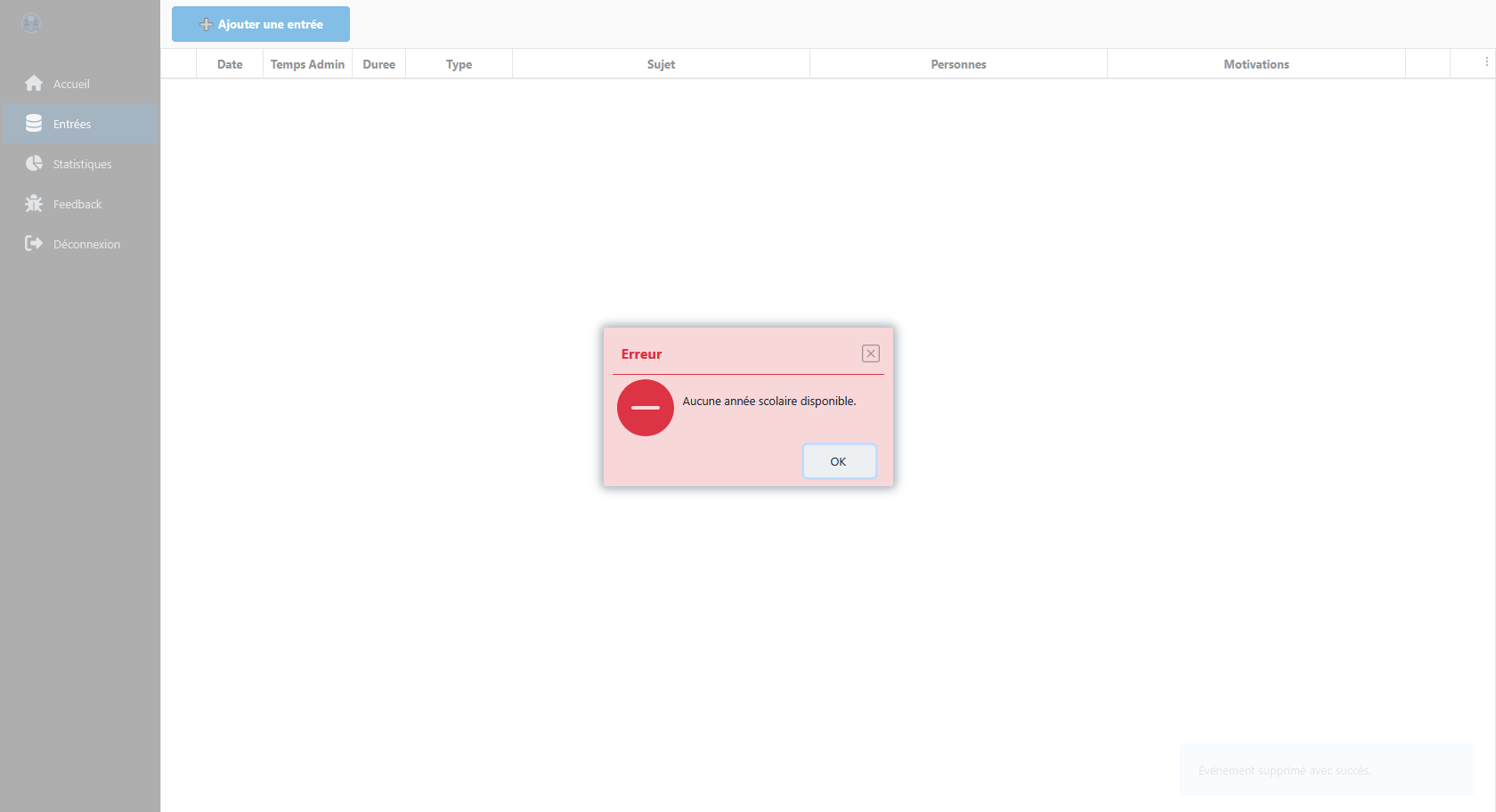


Figure 44 : Message d'erreur si aucune donnée de l'utilisateur n’est trouvée

### Test d’interface Utilisateur

#### Alignement et ergonomie

Condition de test : Vérifier que les boutons et les champs sont alignés et ergonomiques à l’utilisation.

Résultat attendu : L’interface est propre et lisible.

Preuve : Plusieurs enseignants ont testé l’application et valident l’intuitivité et l’utilisation de l’application.

## Erreurs restantes

Le fonctionnement de l’application depuis ordinateur correspond au cahier des charge demandé. Cependant, il reste encore quelques petits bugs pour une utilisation optimale sur mobile. En effet, la navigation des champs ne fonctionne pas car les index de tabulations sont totalement inversés sans raison valable. De plus l’utilisation sur mobile avec un petit écran (téléphone), engendre des problèmes d’affichage aux vues de la quantité de données à afficher, les fenêtres sont trop grandes pour une utilisation idéale sur téléphone.

## Liste des documents fournis

Voici la liste des documents fournis pour la réalisation de ce projet :

* Rapport de projet (ce document)
* Journal de travail
* Code source Git
* Documentation de code Doxygen
* Logo en plusieurs exemplaires (1024x1024, 1024x1024 sans texte)
* Cahier des charges initial
* Maquette
* Sources
* Guide d’utilisation de l’application Journal de médiation

# Conclusions

En conclusion, je suis fier d’avoir pu réaliser ce projet dans les temps, car la charge de travail était conséquente. Finalement, mon projet semble bien réalisé, fonctionnel et prêt pour de la production.

J’ai pu atteindre tous les objectifs fournis dans le cahier des charges avec même quelques ajouts de sécurité comme la validation par mail ou le fait d’accepter uniquement les domaines EduVaud et Edu-Vd pour éviter que n’importe qui puissent créer un compte. Durant la réalisation, j’ai pu me heurter au développement d’interfaces responsives avec WiseJ, ce qui m’a obligé à développer à la main l’interface, pour plus de simplicité, sans utiliser le designer. Les points positifs que j’ai pu retenir de ce projet sont divers. Notamment l’appréciation de l’utilisation de mon application par des testeurs (enseignants qui ne sont pas médiateurs et les médiateurs du CPNV de Ste-Croix). Grâce à ces derniers, j’ai pu résoudre des bugs et des incohérences d’interface.

Certaines améliorations sont possibles dans le futur. Notamment la possibilité d’exporter en PDF les données statistiques de l’année sélectionnée pour faciliter le remplissage du rapport annuel des médiateurs ou encore la possibilité de changer les informations d’identifications de l’utilisateur. Actuellement, cet application (journal de médiation) rend possible la saisie, l’affichage des données d’évènement ainsi que leur statistique, mais il pourrait aussi être mis en lien avec d’autres outils de médiation.

# Annexes

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

Le projet réalisé consiste en la création d'une application web de journal de médiation destinée aux médiateurs scolaires, développée avec le Framework WiseJ.net en C#. L’objectif principal était de moderniser un système ancien basé sur Excel en proposant une solution sécurisée, ergonomique et accessible sur toutes les plateformes (PC, tablette, smartphone). L’application permet aux utilisateurs de s’inscrire, de se connecter via un système d’authentification à deux facteurs (2FA), de saisir des événements de type séance ou entretien, et de visualiser leurs données sous forme de tableau ou de statistiques interactives. Le stockage des données est sécurisé à l’aide d’un système de hachage avec salage, et toutes les interactions suivent les critères d’ergonomie de Bastien et Scapin.

Le projet a été découpé en trois sprints selon la méthode Agile : le premier consacré à l’authentification et à la base du projet, le second à la saisie et à l’affichage des données, et le troisième à l’analyse statistique. De nombreux tests fonctionnels ont été menés sur PC et tablette pour valider les différentes fonctionnalités, avec la participation d’utilisateurs réels (enseignants et médiateurs) via un formulaire de retour.

L’application répond à l’ensemble du cahier des charges initial, avec même des ajouts tels que la validation par email ou la restriction aux domaines autorisés. Malgré quelques limites sur mobile (ergonomie et navigation par tabulation), l’application est stable, fonctionnelle et prête pour une utilisation réelle. Des évolutions futures comme l’export PDF des statistiques ou l’ajout d’outils complémentaires pour les médiateurs sont envisageables. Ce projet a permis de mettre en pratique des compétences techniques en développement .NET, en base de données MySQL, en sécurité informatique et en conception UI, tout en respectant une méthodologie de gestion de projet professionnelle.

## Sources – Bibliographie

Les dates d’utilisation des sources sont référencées dans le journal de travail au fur et à mesures de mon analyse et de mes corrections de bug ou améliorations.

<https://docs.wisej.com/deployment/targets/apache>

<https://wisej.com/>

<https://support.swisscenter.com/Main/frmTicket.aspx?ticketnumber=15A-2F88DAC1-0105&email=gregory.renaud%40eduvaud.ch&h=2B41A2F823B83D3BA58547442888A23A>

<https://www.usabilis.com/criteres-ergonomiques-bastien-et-scapin/>

<https://fr.wikipedia.org/wiki/Cryptographie>

<https://fr.wikipedia.org/wiki/Fonction_de_hachage_cryptographique>

<https://docs.wisej.com/extensions/extensions/navigationbar/api/wisej.web.ext.navigationbar.navigationbar>

<https://docs.wisej.com/api/wisej.web/extenders/wisej.web.tabordermanager>

<https://chatgpt.com/share/68132580-992c-8007-af57-05716a7f78e0>

<https://docs.wisej.com/docs/controls/lists/datagridview>

<https://docs.wisej.com/extensions/extensions/chartjs/api>

De plus, j’ai utilisé ChatGPT ainsi que Github Copilot pour m’aider à la résolution de certaines erreurs ainsi que sur la compréhension et l’utilisation de WiseJ.net.

## Table des illustrations

[Figure 1 : Logo Journal de médiation 1](#_Toc197956063)

[Figure 2 : Page de Login 9](#_Toc197956064)

[Figure 3 : Page d'enregistrement 10](#_Toc197956065)

[Figure 4 : Page de double authentification 10](#_Toc197956066)

[Figure 5 : Page d'accueil 11](#_Toc197956067)

[Figure 6 : Page d'affichage des données 11](#_Toc197956068)

[Figure 7 : Formulaire d'ajout de séance 12](#_Toc197956069)

[Figure 8 : Formulaire d'ajout d’entretien 12](#_Toc197956070)

[Figure 9 : Page de statistiques 13](#_Toc197956071)

[Figure 10 : Connexion avec un compte valide partie 1 18](#_Toc197956072)

[Figure 11 : Connexion avec un compte valide partie 2 18](#_Toc197956073)

[Figure 12 : Validation de la 2FA avec un code valide partie 1 19](#_Toc197956074)

[Figure 13 : Validation de la 2FA avec un code valide partie 2 19](#_Toc197956075)

[Figure 14 : Validation de la 2FA avec un code invalide 20](#_Toc197956076)

[Figure 15 : Connexion avec un compte invalide car un mauvais mot de passe est renseigné 20](#_Toc197956077)

[Figure 16 : Connexion avec un compte invalide car l'utilisateur n'existe pas 21](#_Toc197956078)

[Figure 17 : Création d'un nouvel utilisateur valide 21](#_Toc197956079)

[Figure 18 : Création d'un compte invalide car le domaine email n'est pas valide 22](#_Toc197956080)

[Figure 19 : Création d'un compte invalide car un autre utilisateur possède déjà le même email ou nom 22](#_Toc197956081)

[Figure 20 : Création d'un compte invalide car le mot de passe est trop faible 23](#_Toc197956082)

[Figure 21 : Création d'un compte invalide car les mots de passe ne correspondent pas entre eux 24](#_Toc197956083)

[Figure 22 : Validation du compte par mail partie 1 25](#_Toc197956084)

[Figure 23 : Validation par mail partie 2 25](#_Toc197956085)

[Figure 24 : Validation du compte avec un token invalide ou expiré 26](#_Toc197956086)

[Figure 25 : Déconnexion de l'utilisateur partie 1 27](#_Toc197956087)

[Figure 26 : Déconnexion de l'utilisateur partie 2 27](#_Toc197956088)

[Figure 27 : Chargement correct de la grille de données 28](#_Toc197956089)

[Figure 28 : Suppression d'un évènement partie 1 29](#_Toc197956090)

[Figure 29 : Suppression d'un évènement partie 2 29](#_Toc197956091)

[Figure 30 : Edition d'un évènement partie 1 30](#_Toc197956092)

[Figure 31 : Edition d'un évènement partie 2 30](#_Toc197956093)

[Figure 32 : Ajout d'un évènement invalide car le champ date est hors année scolaire en cours 31](#_Toc197956094)

[Figure 33 : Ajout d'un évènement invalide car le sujet est vide 32](#_Toc197956095)

[Figure 34 : Ajout d'un évènement invalide car le champ personnes concernées est vide 32](#_Toc197956096)

[Figure 35 : Ajout d'une séance invalide car aucun champs de temps est remplis 33](#_Toc197956097)

[Figure 36 : Ajout d'une séance valide partie 1 34](#_Toc197956098)

[Figure 37 : Ajout d'une séance valide partie 2 34](#_Toc197956099)

[Figure 38 : Ajout d'un entretien invalide car aucun type de séance n'est renseigné partie 1 35](#_Toc197956100)

[Figure 39 : Ajout d'un entretien invalide car aucun type de séance n'est renseigné partie 2 35](#_Toc197956101)

[Figure 40 : Ajout d'un entretien invalide car aucunes motivations n'est rensignée 36](#_Toc197956102)

[Figure 41 : Ajout d'un entretien valide partie 1 37](#_Toc197956103)

[Figure 42 : Ajout d'un entretien valide partie 2 37](#_Toc197956104)

[Figure 43 : Affichage des statistiques selon l'année définie 38](#_Toc197956105)

[Figure 44 : Message d'erreur si aucune données de l'utilisateur sont trouvées 39](#_Toc197956106)

## Glossaire

| **Terme** | **Définition** |
| --- | --- |
| **2FA (Authentification à deux facteurs)** | Méthode de sécurité qui demande une deuxième preuve d'identité en plus du mot de passe, par exemple un code envoyé par e-mail. |
| **AES (Advanced Encryption Standard)** | Algorithme de cryptographie symétrique largement utilisé pour sécuriser des données sensibles. |
| **Agile** | Méthodologie de gestion de projet favorisant l’adaptabilité, l’itération rapide et la collaboration continue avec les utilisateurs. |
| **Azure** | Plateforme cloud de Microsoft utilisée pour héberger et déployer des applications web. |
| **Bcrypt** | Algorithme de hachage sécurisé utilisé pour stocker les mots de passe de manière sécurisée avec ajout de "sel". |
| **C#** | Langage de programmation orienté objet développé par Microsoft, utilisé dans les projets .NET. |
| **ChartJS** | Librairie JavaScript (utilisée via WiseJ dans ce projet) permettant de créer des graphiques interactifs. |
| **CRM (Customer Relationship Management)** | Logiciel de gestion de la relation client permettant de suivre les interactions avec les clients, centraliser les informations et améliorer le service client. |
| **Cryptographie** | Science de la sécurisation de l'information, notamment par chiffrement et hachage. |
| **CSS (Cascading Style Sheets)** | Langage de style utilisé pour décrire la présentation visuelle (couleurs, polices, mise en page) des documents HTML. |
| **DES (Data Encryption Standard)** | Ancien algorithme de cryptographie symétrique aujourd’hui considéré comme obsolète à cause de sa faible sécurité. |
| **Doxygen** | Outil de génération de documentation à partir de commentaires dans le code source. |
| **ECC (Elliptic Curve Cryptography)** | Méthode de cryptographie asymétrique basée sur les courbes elliptiques, offrant un bon niveau de sécurité avec des clés plus courtes. |
| **ERP (Enterprise Resource Planning)** | Logiciel de gestion intégrée permettant de gérer les processus internes d’une entreprise. |
| **Git** | Système de gestion de version permettant de suivre les modifications du code source. |
| **GitHub** | Plateforme en ligne pour héberger des projets Git et collaborer à leur développement. |
| **Hash / Hachage** | Transformation d'une donnée en une empreinte fixe, utilisée notamment pour les mots de passe. |
| **HTML (HyperText Markup Language)** | Langage de balisage standard utilisé pour structurer le contenu des pages web. |
| **HTTPS** | Protocole sécurisé de communication web basé sur HTTP et SSL/TLS, utilisé pour chiffrer les échanges. |
| **IDE (Integrated Development Environment)** | Environnement de développement logiciel (ex : Visual Studio). |
| **IceScrum** | Outil de gestion de projet agile basé sur les concepts de Scrum (sprints, stories, etc.). |
| **Interface Responsive** | Interface qui s’adapte automatiquement à différentes tailles d’écrans (smartphone, tablette, PC). |
| **JavaScript** | Langage de programmation côté client utilisé pour rendre les pages web interactives et dynamiques. |
| **JSON** | Format de données léger utilisé pour stocker et échanger des données structurées. |
| **MLD (Modèle Logique de Données)** | Représentation détaillée des tables de base de données et de leurs relations. |
| **MCD (Modèle Conceptuel de Données)** | Schéma décrivant les entités, attributs et relations dans une base de données. |
| **MySQL** | Système de gestion de base de données relationnelle open-source. |
| **NuGet** | Gestionnaire de paquets pour les projets .NET permettant d’ajouter facilement des bibliothèques. |
| **Oracle** | Entreprise informatique connue pour son système de gestion de bases de données (Oracle DB) et son développement de logiciels et outils pour entreprises. |
| **RSA** | Algorithme de cryptographie asymétrique utilisé pour sécuriser les échanges de données. |
| **Salage (Salt)** | Ajout d’une valeur aléatoire à un mot de passe avant son hachage pour augmenter la sécurité. |
| **Scrum** | Méthode agile de gestion de projet basée sur des cycles courts appelés *sprints*. |
| **Seance / Entretien** | Types d’entrées que l’utilisateur peut enregistrer dans l’application. |
| **SHA-256** | Fonction de hachage cryptographique générant une empreinte de 256 bits. |
| **Sprint** | Période de développement courte et fixe (généralement 1 à 4 semaines) dans la méthode Agile, avec des objectifs définis. |
| **SwissCenter** | Hébergeur utilisé pour stocker la base de données et les services associés du projet. |
| **Token** | Chaîne unique utilisée pour valider des opérations comme l’activation de compte ou la réinitialisation de mot de passe. |
| **UI (User Interface)** | Interface utilisateur ; ensemble des éléments visibles avec lesquels un utilisateur interagit dans une application. |
| **UserControl** | Composant graphique réutilisable dans une application WiseJ, affiché dans un Panel. |
| **Validation par mail** | Processus par lequel un utilisateur doit confirmer son adresse email en cliquant sur un lien reçu par mail. |
| **VB.NET (Visual Basic .NET)** | Langage de programmation orienté objet développé par Microsoft, dérivé de Visual Basic, et utilisé dans l’environnement .NET, souvent en alternative à C#. |
| **Visual Studio** | IDE développé par Microsoft utilisé pour développer des applications en C#, .NET, etc. |
| **Windows Form** | Technologie de Microsoft pour créer des interfaces graphiques de bureau (desktop) en .NET, utilisée principalement avec les langages C# ou VB.NET. |
| **WiseJ.net** | Framework web .NET permettant de créer des interfaces web en C# sans avoir besoin de coder en HTML/CSS/JS. |
| **Web.config** | Fichier de configuration d’un projet .NET contenant des paramètres de l’application. |

## Journal de travail

Mon journal de travail étant bien fourni et très grand (plusieurs onglets), il vous est détaillé en annexe et non dans ce document.

## Manuel d'Installation

Concernant l’installation de cette application, j’utilise l’outil de publication de Visual Studio 2022 pour le publier sur Azure courant ma phase de test avec des paramètres spécifiques. Ce qui rend impossible le lancement en local. Pour la phase de production de l’application, un manuel spécifique au déploiement sur la nouvelle infrastructure sera fourni (le déploiement n’est pas dans mon cahier des charges, c’est pourquoi je ne me suis pas encore penché à fond dessus).

## Manuel d'Utilisation

Le manuel d’utilisation est fourni dans un document annexe.

## Cahier des charges

Une image contenant texte, capture d’écran, document, Police

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, conception

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, lettre, Police, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Parallèle, diagramme

Le contenu généré par l’IA peut être incorrect.

## Archive du projet

Les archives du projet sont disponibles, découpées par semaines dans le code source à l’adresse : <https://github.com/mistergriff/TPI-MED/tree/main/Documentation>