

---

# Rapport de Projet

Réalisé par :

MAYLIN Enzo, BAVOILLOT Léanne, CAPO Mathys, DE MURCIA Enzo, JOST Victor

Projet Encadré Par :

M. GASQUET Malo

Pour L'obtention du BUT Informatique

Année universitaire 2023-2024

---

# Remerciements

---

Nous tenons sincèrement à remercier toutes les personnes qui ont permis à ce projet d'exister.

Nous voudrions d'abord remercier M. Gasquet, M. Lebreton ainsi que M. Nadal pour la rédaction des travaux dirigés de Complément Web, ainsi que M. Poupet et M. Rosenfeld pour les travaux dirigés de JavaScript. Ces travaux nous ont permis une bonne compréhension des technologies que nous avons utilisées dans ce projet.

Nous remercions à nouveau M. Gasquet pour sa disponibilité quant aux questions que nous avons, et son expertise sur le projet.

Et un autre merci aux professeurs qui ont travaillé sur l'élaboration du Trello-Trollé, et qui ont pris le temps de le rendre ludique et amusant pour nous, étudiants.

Tous les autres enseignants qui nous ont aidé à comprendre les concepts adjacents, tels qu'avec les modules de *qualité au-delà du relationnel*, *qualité de développement*, *management des systèmes d'information*, *gestion de projet agile*, ou encore *communication interne* ont également droit à nos sincères remerciements.

Et nous finirons par remercier l'IUT Montpellier-Sète pour la qualité de son enseignement.

---

# Résumé

---

## Français

Dans le cadre de notre apprentissage des technologies du web, nos professeurs de l'IUT Montpellier-Sète ont développé une application web qu'ils ont appelé Trello-Trollé dont le code était fonctionnel, mais volontairement améliorable, dans le but que nous, étudiants, apprenions à réusiner ce type de site web en quelque chose de plus adapté aux technologies d'aujourd'hui.

Par équipe de 5 étudiants, nous avons donc effectué ce réusinage de la façon la plus efficace possible. Afin de faire cela au mieux, nous nous sommes appuyés sur nos précédentes compétences acquises lors des travaux dirigés réalisés en cours. Les technologies utilisées sont : *Twig*, *Routage par chemins*, *API REST*, *Promises de JavaScript*. Nous les détaillerons plus tard dans ce rapport.

## English

As part of our web technology learning, our professors at IUT Montpellier-Sète developed a web application they named Trello-Trollé, whose code was functional but deliberately improvable. The purpose was for us, the students, to learn how to refactor this type of website into something more suited to today's technologies.

In teams of 5 students, we carried out this refactoring in the most efficient way possible. To do this effectively, we relied on our previous skills acquired during the supervised work done in class. The technologies used are: *Twig*, *routing by paths*, *REST API*, *JavaScript Promises*. We will detail these later in this report.

---

# Table des matières

---

Français	2
English	2
<b>ANALYSE</b>	<b>8</b>
<b>1. Base de données</b>	<b>9</b>
Démarche	9
1.1. Anomalies	10
1.2. Incohérences	11
1.3. Dépendances fonctionnelles	12
1.4. Choix de la clé primaire	13
1.5. Normalisation de la base de données	14
<b>2. Architecture de l'application</b>	<b>17</b>
Démarche	17
2.1. CRUD	17
2.2. DRY1	18
2.3. SOLID	19
2.4. API REST	20
2.5. KISS	20
<b>3. Faille de sécurité</b>	<b>21</b>
Démarche	21
3.1. Faille XSS (stored)	21
3.2. Accès aux comptes par oubli de mot de passe	23
3.3. Accès au mot de passe via un ordinateur partagé	25
3.4. Mauvaise redirection	28
3.5. Mot de passe en clair dans la base de données	29
<b>4. Problèmes d'utilisation du site</b>	<b>30</b>
Démarche	30
4.1. Problème sur l'affichage du login	30
4.2. Obligation de remplir tous les champs	32
4.3. Affichage des pages	33
<b>5. Analyse ergonomique</b>	<b>34</b>
Démarche	34
5.1. L'accueil	34
5.2. Inscription	35
5.3. Connexion	35
5.4. J'ai oublié mon mot de passe	36
5.5. Gestion des tableaux	36
5.6. Édition des tableaux	37

5.7. Gestion du compte	38
<b>6. Conception</b>	<b>39</b>
6.1. Entités du domaine	39
6.2. Processus et interactions	39
6.3. Choix technologiques	41
6.4. Suppression des failles de sécurité	42
<b>7. Réalisation</b>	<b>44</b>
7.1. Architecture	44
7.2. Routing	45
<b>8. Validation, résultats et perspectives</b>	<b>46</b>
8.1. Validation	46
8.2. Résultats	47
<b>9. Organisation du projet</b>	<b>50</b>
9.1. Outils de communication	50
9.2. Méthode de développement	51
9.3. Retour critique	51
9.4. Bilan critique	52
<b>10. Conclusion</b>	<b>53</b>
<b>11. Les sources</b>	<b>54</b>

---

# Table des figures

---

ANALYSE.....	8
Figure 1 : Message Flash d'une anomalie de suppression.....	10
Figure 2 : Anomalie de modification dans la Base de Donnée.....	10
Figure 3 : Incohérence de nom/prénom dans la Base de Donnée.....	11
Figure 4 : Incohérence de titre de colonne dans la Base de Donnée.....	11
Figure 5 : Schéma de la nouvelle Base de Donnée.....	16
Figure 6 : Code répété dans le ControleurUtilisateur.....	18
Figure 8 : Code pour l'exploitation de faille XSS.....	22
Figure 9 : Exemple d'insertion de code JavaScript.....	22
Figure 10 : Message de la faille XSS.....	22
Figure 11 : Affichage du mot de passe oublié.....	23
Figure 12 : Code pour récupérer les mots de passe avec l'email.....	24
Figure 13 : Mot de passe stocké dans les cookies.....	25
Figure 14 : Mot de passe prés rempli après une connexion.....	25
Figure 15 : Affichage de la mise à jour du profil.....	26
Figure 16 : Code du stockage du mot de passe dans les cookies.....	27
Figure 17 : Code de création de la Base de donnée.....	29
Figure 18 : Login de l'utilisateur mal affiché.....	30
Figure 19 : Code de l'erreur de l'affichage du login.....	31
Figure 20 : Champs mettre à jour du compte.....	32
Figure 21 : Code du champ du login.....	32
Figure 22 : Page liste des tableaux.....	33
Figure 23 : Nouveau modèle E/A.....	39
Figure 24 : Trigger avant_suppression_utilisateur.....	40
Figure 25 : Diagramme de cas d'utilisation d'un tableau.....	41
Figure 26 :Correction échappement de caractères.....	43
Figure 27 : Schéma MVCS2.....	44
Figure 28 La couverture des tests.....	46

---

# Glossaire

---

<sup>1</sup>**DRY** : Don't Repeat Yourself, principe de développement pour favoriser la lisibilité et la qualité du code.

<sup>2</sup>**MVCS** : modèle d'architecture qui répartit les responsabilités en quatre couches : Modèle, Vue, Contrôleur, Services.

<sup>3</sup>**KISS** : keep it simple, stupid ( garde ça simple, idiot ).

<sup>4</sup>**Composer** : gestionnaire de dépendances PHP qui facilite l'intégration et la gestion de bibliothèques tierces.

---

# Introduction

---

Dans le cadre de la gestion de projets agiles, il est souvent difficile de répartir correctement les tâches entre tous les membres d'un projet. Dans notre monde où la covid a mis en lumière l'intérêt du télétravail, il s'avère maintenant vital pour une entreprise de rendre ses systèmes utilisables à distance. En ce sens, la problématique se pose : comment mettre en commun l'état des tâches du projet quand les membres n'ont pas (ou peu) de contacts physiques ? C'est là que vient l'idée d'un système informatique de gestion des tâches. Un système visuel dans lequel est reproduit à l'identique les tableaux des grandes entreprises, où sont rassemblés la performance, la flexibilité, la confidentialité et le confort visuel.

Le projet a pour ambition la reproduction informatique d'un tableau, comportant des colonnes, avec des cartes dans lequel interviennent des participants. Un tableau a un nom, et possède une "liste blanche" comprenant les membres pouvant éditer / lire le tableau. Chaque colonne possède également un nom, et comprend une ou plusieurs cartes avec un titre, une description et une couleur. Les utilisateurs ayant les droits d'édition pourront créer, supprimer, éditer et déplacer les cartes à travers les différentes colonnes à l'aide d'un "drag and drop" afin de s'organiser au mieux. Ils pourront également créer, supprimer, éditer et classer des colonnes, ainsi que créer et éditer des tableaux avec leurs membres (inviter, exclure...).

Dans ce rapport orienté analyse de l'existant, nous détaillerons dans un premier temps l'analyse de la Base de Données (analyse du choix, les anomalies, etc.), puis l'architecture de l'application en s'appuyant sur les principes de qualité de code (DRY\*, SOLID\*, etc.). Une fois cela fait, nous continuerons avec les failles de sécurité que nous avons répertoriées dans l'application, illustrée avec des captures d'écran de cas concrets, et nous terminerons avec une analyse ergonomique l'application. Nous pourrons ainsi, dans un avenir (très) proche, poursuivre notre activité sur le réusinage de l'application.



# ANALYSE

---

# 1. Base de données

---

## Démarche

Nous cherchons dans cette partie à rendre notre Base de Données optimisée, c'est-à-dire faciliter le besoin que l'on a à travers son utilisation. Dans notre cas, nous cherchons à faciliter l'accessibilité et la mise à jour des données, c'est ainsi que nous tendons vers la *normalisation*. Normaliser, c'est faire en sorte que nos tables respectent trois principes essentiels : Rendre les données atomiques (autrement dit n'avoir qu'une seule donnée par case), faire des liaisons minimales entre chacune, puis regrouper les données directes entre elles.

D'abord, nous faisons un travail d'analyse afin de pointer les anomalies. Une anomalie, c'est un aspect de la Base de Données qui n'est pas souhaité en fonction du besoin que l'on cherche à combler. Pour cela, nous nous basons sur notre expérience en s'appuyant sur l'analyse implicite de l'existant, ainsi que sur ce que nous avons vu en cours. Nous en déduisons donc où, sur le site, nous devons chercher pour trouver des erreurs (ou incohérences) sur lesquelles s'appuyer.

Une fois les anomalies (et incohérences) trouvées, nous sommes en mesure de justifier une nouvelle forme de Base de Données, ce que nous cherchons donc à mettre en place. Le but est de reprendre le travail fait par le développeur dans l'optique de savoir ce qui aurait dû être fait. Voici notre démarche :

1. Analyser l'existant

C'est-à-dire répertorier tous les schémas, les scripts d'insertion, les formats de données, etc.

2. Répertorier les liaisons entre les données (ce que l'on appelle *Dépendances Fonctionnelles*)

Nous nous basons pour cela sur notre seule logique, appuyée par l'utilisation de ces données dans le code.

3. Normaliser les tables

Nous utilisons pour cela le théorème de Casey-Delobel vu en cours.

4. Déterminer la clé primaire

À l'aide des précédentes DF, nous pouvons déterminer celles qui sont les plus importantes (celles qui donnent toutes les autres).

## 1.1. Anomalies

La base de données actuelle se compose d'une unique table regroupant l'ensemble des données, incluant celles des utilisateurs, des tableaux, des colonnes et des cartes.

Une redondance significative est observée dans la table `app_db`, notamment au niveau des attributs liés à un tableau (`codeTableau`, `participants`, etc...) et aux informations des utilisateurs (`nom`, `prénom`, `email`, `mdp`, `mdphaché`). Ces anomalies vont créer des problèmes lors des actions de suppressions, insertions et modifications de données.

Exemple de problème de **suppression** :

Un utilisateur est dans l'obligation d'avoir à minima une carte, à cause du choix de la clé primaire. Ce choix amène au message d'erreur de la figure 1.

Vous ne pouvez pas supprimer cette carte car cela entrainera la suppression du compte du propriétaire du tableau

Figure 1 : Message Flash d'une anomalie de suppression

**Modification :**

Si un utilisateur modifie son nom ou son prénom, il faudra modifier plusieurs lignes (si on ne modifie pas toutes les lignes concernées, on va encore avoir des incohérences). Si toutes les lignes ne sont pas modifiées on a les incohérences de la figure 2.

	login	nom	prenom
1	MaylinEnzo4@	Maylin	Enzo
2	capom	capo	mathys
3	capom	jean	neimar
4	capom	paul	paul

Figure 2 : Anomalie de modification dans la Base de Donnée

**Insertion :**

Si un utilisateur souhaite créer cinq colonnes dans son tableau, mais qu'il ne possède que quatre cartes, il lui sera impossible d'enregistrer cette information sans créer une nouvelle carte.

## 1.2. Incohérences

Un même login peut posséder des noms et des prénoms différents. Dans la figure 3, l'utilisateur de login "MaylinEnzo4@" possède trois noms différents et quatre prénoms différents.

	login	nom	prenom
13	MaylinEnzo4@	oui	non
14	MaylinEnzo4@	Maylin	EnzoModifié
15	MaylinEnzo4@	Maylin	Enzo
16	MaylinEnzo4@	Neymar	Jean

Figure 3 : Incohérence de nom/prénom dans la Base de Donnée

Il est possible qu'une colonne présente plusieurs titres différents. La figure 4 nous montre que la colonne d'identifiant 32 est associée aux titres "titre 1" et "titre 2". Or, dans le contexte de Trello-trollé, cette configuration de deux titres pour une même colonne est incohérente.

idcolonne	titrecolonne
32	titre 2
32	titre 1

Figure 4 : Incohérence de titre de colonne dans la Base de Donnée

Cette base de données comporte de la redondance, des incohérences et des anomalies en raison de son manque de normalisation. Dans ce contexte, une base de données normalisée serait préférable étant donné la fréquence élevée des opérations de modification et de suppression de données, qui sont souvent problématiques pour les bases de données non normalisées.

## 1.3. Dépendances fonctionnelles

Pour ne pas avoir toutes ces anomalies de mise à jour, il faut travailler sur un schéma "sain", sans redondance soit un schéma normalisé.

Pour cela on décompose en utilisant Casey-Delobel. Pour être sûr de ne perdre aucune relation, en réduisant la redondance et garantir l'intégrité des données. (La contrepartie de cette normalisation est qu'on a maintenant plusieurs relations et que les requêtes vont devoir comporter des jointures. Ces requêtes seront donc plus complexes à écrire et plus longues à exécuter. Cependant on s'assure qu'il n'y ait plus aucune anomalies).

Soit  $R$  la relation trouvée à partir de la table `app_db`.

$R = \{ \text{login, email, nom, prenom, mdpHache, mdp, idTableau, codeTableau, titreTableau, participants, idColonne, titreColonne, idCarte, titreCarte, descriptifCarte, couleurCarte} \}$

Soient `Participants` et `Affectés`, les relations trouvées par extraction des documents JSON.

`Participants = { login, prenom, nom, email, mdp, mdphache }`

`Affectes = { login, prenom, nom, email, mdp, mdphache }`

À l'aide de nos connaissances du projets Trello-Trollé, nous avons trouvé par déduction les dépendances fonctionnelles suivantes :

Soient  $F$ ,  $F\_Participants$  et  $F\_Affectes$  les ensembles des dépendances fonctionnelles des relations  $R$ , `Participants` et `Affectes`.

$F = \{$   
     $\text{login} \rightarrow \text{email, nom, prenom, mdpHache, mdp} ;$   
     $\text{idTableau} \rightarrow \text{codeTableau, titreTableau} ;$   
     $\text{idColonne} \rightarrow \text{titreColonne, idTableau} ;$   
     $\text{idCarte} \rightarrow \text{titreCarte, descriptifCarte, couleurCarte, idColonne} ;$   
     $\text{login, idTableau} \rightarrow \text{participant} ;$   
     $\text{login, idCarte} \rightarrow \text{affecte}$   
 $\}$

$F\_Participants = \{ \text{login} \rightarrow \text{prenom, nom, email, mdp, mdphache} \}$

$F\_Affectes = \{ \text{login} \rightarrow \text{prenom, nom, email, mdp, mdphache} \}$

## 1.4. Choix de la clé primaire

La clé primaire assure l'unicité des lignes et facilite les références entre les tables. Elle garantit l'intégrité référentielle et optimise les performances. Le choix de la clé primaire est important pour conserver les données.

Soit  $F^+$ , la fermeture transitive de  $F$ .

```
F+ = {  
    login → email, nom, prenom, mdpHache, mdp  
    idTableau → codeTableau, titreTableau, email, nom, prenom, mdpHache, mdp  
    idColonne → titreColonne, idTableau, codeTableau, titreTableau  
    idCarte → titreCarte, descriptifCarte, couleurCarte, idColonne, titreColonne,  
              idTableau, codeTableau, titreTableau  
    login, idTableau → participant  
    login, idColonne → participant  
    login, idCarte → affecte, participant  
}
```

On peut remarquer que les attributs `idCarte` et `login` sont les seuls attributs de  $R$  qui ne sont jamais à droite des DF. On peut en déduire qu'ils seront forcément dans les clés.

Or, `idCarte, login` est une clé, car par transitivité, on retrouve tous les attributs de  $R$ . On peut en conclure que `idCarte, login` est la seule clé de  $R$ . C'est donc un choix cohérent d'avoir choisi `idCarte, login` comme clé primaire de la table `app_db`.

## 1.5. Normalisation de la base de données

Il est clair que les relations Participants et Affectés sont normalisées jusqu'à la troisième forme normale (3FN) car la clé primaire est correcte, les dépendances fonctionnelles sont élémentaires et directes. Cependant, la relation R présente des anomalies de normalisation. Nous allons maintenant la décomposer afin de remédier à ces problèmes.

$R^{1FN} = \{ \underline{\text{login}}, \text{email}, \text{nom}, \text{prenom}, \text{mdpHache}, \text{mdp}, \text{idTableau}, \text{codeTableau}, \text{titreTableau}, \text{participants}, \text{idColonne}, \text{titreColonne}, \underline{\text{idCarte}}, \text{titreCarte}, \text{descriptifCarte}, \text{couleurCarte}, \text{affecte} \}$

• **login** → **email, nom, prenom, mdpHache, mdp** •

→  $R1^{3FN} = \{ \underline{\text{login}}, \text{email}, \text{nom}, \text{prenom}, \text{mdpHache}, \text{mdp} \}$

→  $R2^{1FN} = \{ \underline{\text{login}}, \text{idTableau}, \text{codeTableau}, \text{titreTableau}, \text{participant}, \text{idColonne}, \text{titreColonne}, \underline{\text{idCarte}}, \text{titreCarte}, \text{descriptifCarte}, \text{couleurCarte}, \text{affecte} \}$

R2

• **login, idTableau** → **participant** •

→  $R21^{3FN} = \{ \underline{\text{login}}, \underline{\text{idTableau}}, \text{participant} \}$

→  $R22^{1FN} = \{ \underline{\text{login}}, \text{idTableau}, \text{codeTableau}, \text{titreTableau}, \text{idColonne}, \text{titreColonne}, \underline{\text{idCarte}}, \text{titreCarte}, \text{descriptifCarte}, \text{couleurCarte}, \text{affecte} \}$

R22

• **idTableau** → **codeTableau, titreTableau** •

→  $R221^{3FN} = \{ \underline{\text{idTableau}}, \text{codeTableau}, \text{titreTableau} \}$

→  $R222^{1FN} = \{ \underline{\text{login}}, \text{idTableau}, \text{idColonne}, \text{titreColonne}, \underline{\text{idCarte}}, \text{titreCarte}, \text{descriptifCarte}, \text{couleurCarte}, \text{affecte} \}$

R222

• **idColonne** → **titreColonne, idTableau** •

→  $R2221^{3FN} = \{ \underline{\text{idColonne}}, \text{titreColonne}, \text{idTableau} \}$

→  $R2222^{1FN} = \{ \underline{\text{login}}, \text{idColonne}, \underline{\text{idCarte}}, \text{titreCarte}, \text{descriptifCarte}, \text{couleurCarte}, \text{affecte} \}$

R2222

• **idCarte** → **titreCarte, descriptifCarte, couleurCarte, idColonne** •

→  $R22221^{3FN} = \{ \underline{\text{idCarte}}, \text{titreCarte}, \text{descriptifCarte}, \text{couleurCarte}, \text{idColonne} \}$

→  $R22222^{3FN} = \{ \underline{\text{login}}, \underline{\text{idCarte}}, \text{affecte} \}$

D'après le théorème de Casey-Delobel, voici la décomposition de la relation R :

- $R1^{3FN} = \{ \underline{\text{login}}, \text{email}, \text{nom}, \text{prenom}, \text{mdpHache}, \text{mdp} \}$
- $R21^{3FN} = \{ \underline{\text{login}}, \underline{\text{idTableau}}, \text{participant} \}$
- $R221^{3FN} = \{ \underline{\text{idTableau}}, \text{codeTableau}, \text{titreTableau} \}$
- $R2221^{3FN} = \{ \underline{\text{idColonne}}, \text{titreColonne}, \text{idTableau} \}$
- $R22221^{3FN} = \{ \underline{\text{idCarte}}, \text{titreCarte}, \text{descriptifCarte}, \text{couleurCarte}, \text{idColonne} \}$
- $R22222^{3FN} = \{ \underline{\text{login}}, \underline{\text{idCarte}}, \text{affecte} \}$

**Avec cette décomposition il n'y a pas de perte de données.**

- $F1 = \{ \text{login} \rightarrow \text{email}, \text{nom}, \text{prenom}, \text{mdpHache}, \text{mdp} \}$
- $F21 = \{ \text{login}, \underline{\text{idTableau}} \rightarrow \text{participant} \}$
- $F221 = \{ \underline{\text{idTableau}} \rightarrow \text{codeTableau}, \text{titreTableau} \}$
- $F2221 = \{ \underline{\text{idColonne}} \rightarrow \text{titreColonne}, \text{idTableau} \}$
- $F22221 = \{ \underline{\text{idCarte}} \rightarrow \text{titreCarte}, \text{descriptifCarte}, \text{couleurCarte}, \text{idColonne} \}$
- $F22222 = \{ \text{login}, \underline{\text{idCarte}} \rightarrow \text{affecte} \}$

On remarque que  $(F1 \cup F2 \cup F3 \cup F4 \cup F5 \cup F6)$  est égal à F.

Donc :  $(F1 \cup F2 \cup F3 \cup F4 \cup F5 \cup F6)^* = F^*$

**Nous pouvons voir que la décomposition est sans perte de dépendances fonctionnelles.**

Il est observé que les relations Participants, Affectes et R1 partagent la même clé, les mêmes dépendances fonctionnelles, ainsi que les mêmes attributs. Par conséquent, il est possible de supprimer les relations Participants et Affectes qui sont des doublons de R1.

Cela signifie que les attributs affecte et "participant" redirigent vers la relation R1 avec la clé "login". Ces attributs peuvent être remplacés par "login".

Par conséquent, nous avons :

- $R22222 = \{ \underline{\text{login}}, \underline{\text{idCarte}}, \text{login} \}$
- $F22222 = \{ \text{login}, \underline{\text{idCarte}} \rightarrow \text{login} \}$
- $R21 = \{ \underline{\text{login}}, \underline{\text{idTableau}}, \text{login} \}$
- $F21 = \{ \text{login}, \underline{\text{idTableau}} \rightarrow \text{participant} \}$



En utilisant la réflexivité, la redondance de l'attribut "login" peut être supprimée, aboutissant à :

R22222 = { login, idCarte }

R21 = { login, idTableau }

Jusqu'à présent, nous n'avons pas accordé d'attention particulière aux noms des relations, nous avons simplement opté pour des noms simples. Afin de clarifier cette situation, nous procédons à une étape de renommage.

Nous obtenons avec ces relations, le schéma de la figure 5.

Utilisateurs = { login, email, nom, prenom, mdpHache, mdp }

Participants = { login, idTableau }

Tableaux = { idTableau, codeTableau, titreTableau }

Colonnes = { idColonne, titreColonne, idTableau }

Cartes = { idCarte, titreCarte, descriptifCarte, couleurCarte, idColonne }

Affectes = { login, idCarte }

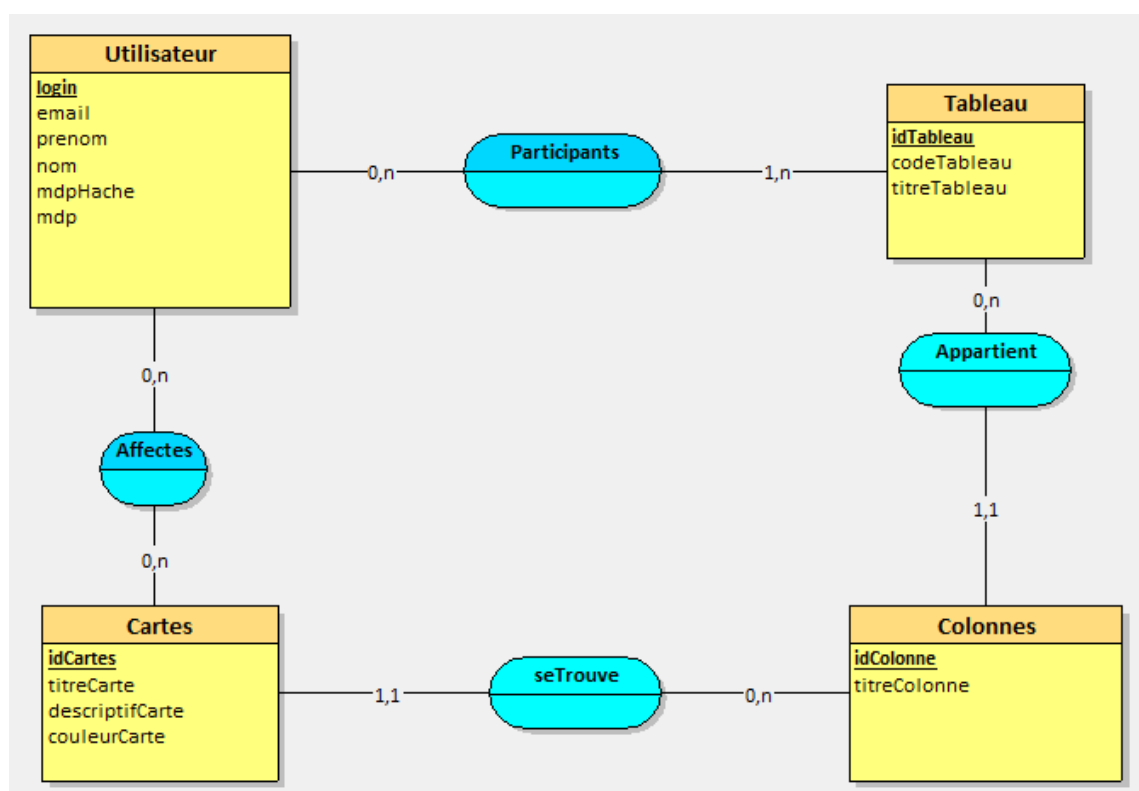


Figure 5 : Schéma de la nouvelle Base de Données

## 2. Architecture de l'application

---

### Démarche

L'architecture de l'application est la partie qui suit la base de données. Cette fois nous analysons le code en lui-même, en termes de qualité basé sur les principes populaires de programmation qualitative. Dans le cadre de services tels que Trello-Trollé, la qualité de code est impérative. En effet, si notre code est trop gourmand en termes de ressources, nous pouvons très vite surcharger nos serveurs à mesure que les clients augmentent, et faire ainsi grimper les coûts des machines et d'entretien. En ce sens, un code de mauvaise qualité est un code qui ne favorise pas le maintien et l'évolution de notre application. Ce qui est problématique car nous perdrons des clients si notre application n'évolue pas dans le temps. Donc, en somme, il vaut peut-être mieux donner un peu plus de temps à la programmation du site web pour gagner en qualité, que de tenter de réduire tant bien que mal le temps de développement en rendant un travail difficilement améliorable.

Pour ce faire, nous avons décidé d'employer une méthode qui nous paraissait simple : répertorier les principaux principes de qualité de code, et pour chacun d'eux, analyser en quoi notre application y répond ou n'y répond pas. Ainsi, dans la future phase de conception, nous saurons directement quels sont les points à améliorer, ce qui nous permettra de gagner grandement en efficacité.

### 2.1. CRUD

Trello-trolle est un outil de gestion de projet en ligne. Grâce à cette application Web, un utilisateur est capable de : créer, lire, modifier, supprimer ses tableaux, ajouter/supprimer une colonne/carte ; les quatre opérations de base pour le stockage d'information dans la base de données.

## 2.2. DRY<sup>1</sup>

DRY<sup>1</sup> (Don't Repeat Yourself) est un principe essentiel en programmation qui vise à éviter les redondances inutiles. En ne le respectant pas, notre application actuelle présente des duplications de code qui peuvent rendre la compréhension et la maintenance plus complexes. En cas de nécessité de modification, celle-ci doit être effectuée en plusieurs endroits, augmentant ainsi le risque d'erreurs.

Par exemple, dans le `ControleurUtilisateur` dans les fonctions `mettreAJour` et `supprimer`, nous avons la figure 6 qui est répété :

```
$carteRepository = new CarteRepository();
$cartes =
$carteRepository->recupererCartesUtilisateur($login);
foreach ($cartes as $carte) {
    $participants = $carte->getAffectationsCarte();
    $participants = array_filter($participants,
        function ($u) use ($login) {
            return $u->getLogin() !== $login;
        }
    );
    $participants[] = $utilisateur;
    $carte->setAffectationsCarte($participants);
    $carteRepository->mettreAJour($carte);
}

$tableauRepository = new TableauRepository();
$tableaux =
$tableauRepository->recupererTableauxParticipeUtilisateur($login);
foreach ($tableaux as $tableau) {
    $participants = $tableau->getParticipants();
    $participants = array_filter($participants,
        function ($u) use ($login) {
            return $u->getLogin() !== $login;
        }
    );
    $participants[] = $utilisateur;
    $tableau->setParticipants($participants);
    $tableauRepository->mettreAJour($tableau);
}
```

Figure 6 : Code répété dans le `ControleurUtilisateur`

## 2.3. SOLID

### 1.1. S

*Le principe S (Single Responsibility) n'est pas respecté.*

Le site respecte actuellement le model MVC, hors sans la couche service les contrôleurs ont trop de responsabilité et le code ne respecte plus le principe de Single Responsibility. Il est donc nécessaire de passer sur un modèle MVCS<sup>2</sup>.

### 1.2. O

*Le principe O (Open / Close) n'est pas respecté.*

En regardant la class Carte, si différents types de cartes sont venus à être implémentés, il faudra modifier l'ensemble des méthodes associées. KISS

### 1.3. L

*Le principe L (Liskov Substitution) n'est pas respecté.*

Étant donné qu'aucune interface n'est implémentée, le principe de substitution de Liskov ne peut pas être appliqué au code.

### 1.4. I

*Le principe I (Interface Segregation) n'est pas respecté.*

Étant donné qu'aucune interface n'est implémentée, le principe de ségrégation des interfaces ne peut pas être appliqué au code.

### 1.5. D

*Le principe D (Dependency Inversion) n'est pas respecté.*

Chaque contrôleur présente une forte dépendance vis-à-vis des repositories, avec chacun d'entre eux faisant appel à 3 ou 4 repositories. Il est donc nécessaire de mettre en place des interfaces pour pouvoir injecter les dépendances.

## 2.4. API REST

Les API permettent la communication entre les composants d'une application et avec d'autres développeurs via des requêtes et des réponses standardisées. Les services Web RESTful (qui satisfont les contraintes REST) suivent les principes fondamentaux suivants :

1. Adoption d'une convention de nommage pour les URI des ressources.
2. Utilisation des verbes HTTP pour les opérations sur les ressources.
3. Emploi des codes de réponse HTTP pour indiquer le succès ou l'échec d'une requête.
4. Échange de données au format JSON ou XML.
5. Être sans état, ce qui signifie que chaque requête est traitée indépendamment des précédentes.
6. Offrir une découverte de service, fournissant des URL pour les actions liées aux ressources.

## 2.5. KISS

Dans la class `src/Modele/DataObject/HTTP/Cookie.php`, nous trouvons une méthode `fun` qui a plusieurs défauts. D'abord, cette fonction est récursive et elle s'appelle à l'infini. Puis, si nous arrivons à arriver aux sous fonction, elles n'aboutissent à rien. Elles ne sont pas implémentées. Le principe KISS<sup>3</sup> n'est pas respectée avec cette difficulté.

## 3. Faille de sécurité

---

### Démarche

La sécurité d'un site web est primordiale pour la survie de l'économie engendrée par ce dernier. Si un site n'est pas sécurisé, des pirates informatiques auront alors champ libre pour le hacker. Les risques sont nombreux : fuite de données (mot de passe, nom, prénom, carte bancaire, etc.), injection de script malveillant dans les machines clientes, et beaucoup d'autres. Aussi, en termes légaux, le site se doit de respecter certaines normes imposées. En cas de non-respect de ces dernières, la société possédant le site web est exposée à des amendes pouvant grandement nuire à leur activité. C'est pourquoi nous prenons le temps de faire des tests de sécurité (aussi appelé *pentest* pour *penetration testing*).

Au sein de notre formation, nous n'avons jamais véritablement eu de cours de sécurité. Seuls des procédés anti-hack nous ont été communiqués, et nous avons eu informellement des retours de sécurité informatique dans notre précédente SAÉ. Notre démarche a donc été simple : d'abord, nous testons ce que nos évaluateurs testaient sur nos projets pour nous faire remarquer des failles de sécurité (jusqu'ici, tout se fait sur l'interface). Ensuite, nous parcourons le code pour répertorier où est-ce qu'il serait intéressant de creuser, et, dès qu'une faille est trouvée, nous l'exploitons jusqu'à parvenir à avoir des éléments suffisants sur lesquels nous appuyer pour notre analyse. À chaque partie, vous retrouverez donc une faille associée avec son analyse.

### 3.1. Faille XSS (stored)

Il existe plusieurs vulnérabilités XSS sur le site, qui peuvent être exploitées à travers différents champs de texte, tels que le champ de login / nom / prénom lors de la création d'un compte ou encore via le nom d'un tableau. Cependant, la méthode la plus efficace consiste à passer par le nom d'une colonne. Dans la suite, nous examinerons comment exploiter cette faille à travers ce champ spécifique.

Cette faille est rendue possible, car il n'y a pas d'appel à "`htmlspecialchars()`" pour afficher certaines informations dans la vue tableau.php. Le code de la figure 8 le démontre.

```
<div class="colonne">
  <div class="titre icons_menu">
    <span><?= $colonnes[$i]->getTitreColonne() ?></span>
```

Figure 8 : Code pour l'exploitation de faille XSS

De ce fait, si on met une balise HTML en nom de colonne, elle sera interprétée par le navigateur de chaque membre du tableau lorsqu'ils se rendront dessus. On peut donc mettre une balise `<script></script>` pour exécuter du code JavaScript malveillant. Voici un exemple d'utilisation:

1. On met une balise script en nom de colonne comme sur la figure 9.



Figure 9 : Exemple d'insertion de code JavaScript

(il est également possible de passer par d'autres balises, par exemple la balise "``" aura le même effet)

2. Lorsque n'importe quel membre charge le tableau, le code js est exécuté et il y a donc notre alerte qui s'affiche qui est celle de la figure 10.



Figure 10 : Message de la faille XSS

Comme mentionné précédemment, il est possible d'exploiter une vulnérabilité XSS à divers endroits du code (à chaque champ de texte sans `htmlspecialchars()` qui affiche une valeur de la base de données entrée par un utilisateur), et pour chacun d'eux, la manipulation nécessaire pour que le navigateur interprète la balise est différente.

Les conséquences d'une faille XSS peuvent varier, allant de simples messages s'affichant à l'ouverture d'un tableau, comme illustré précédemment, à des impacts plus graves. Un attaquant peut non seulement bloquer l'accès au tableau pour d'autres membres, mais aussi exécuter du code JavaScript sur l'ordinateur de la victime, permettant ainsi le vol de cookies, de jetons de session voire même l'installation de logiciels malveillants sur l'ordinateur ciblé.

Pour remédier à cette faille, il est recommandé d'encadrer chaque champ de texte affichant une valeur de la base de données entrée par un utilisateur avec `htmlspecialchars()` ou d'utiliser une solution comme Twig.

## 3.2. Accès aux comptes par oubli de mot de passe

Il existe une faille de sécurité majeure, grâce à elle n'importe qui peut récupérer le mot de passe de n'importe quel compte directement sur le site en disposant uniquement de l'email associé au compte. Il suffit de cliquer sur "[Login et/ou mot de passe oubliés ?](#)" sur la page de connexion et de renseigner l'email. Si par exemple, on renseigne l'email "[lbvl@orange.fr](mailto:lbvl@orange.fr)" (l'email associé au compte "leannebvl"), on obtient les informations de la figure 11 :

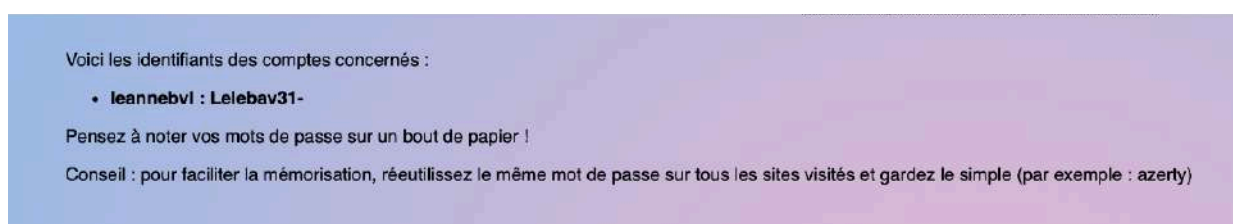


Figure 11 : Affichage du mot de passe oublié

On peut donc avoir accès au compte "leannebvl" en ne possédant que son email, ce qui est très problématique, cela signifie que tout monde peut avoir accès au compte de n'importe qui.

La figure 12 montre le code à l'origine de cette faille.



```

public static function recupererCompte(): void {
    if(ConnexionUtilisateur::estConnecte()) {
        ControleurTableau::redirection("utilisateur",
            "afficherListeMesTableaux");
    }
    if (!ControleurUtilisateur::issetAndNotNull(["email"])) {
        MessageFlash::ajouter("warning", "Adresse email
            manquante");
        ControleurUtilisateur::redirection("utilisateur",
            "afficherFormulaireConnexion");
    }
    $repository = new UtilisateurRepository();
    $utilisateurs = $repository->recupererUtilisateursParEmail(
        $_REQUEST["email"]);
    if(empty($utilisateurs)) {
        MessageFlash::ajouter("warning", "Aucun compte associé
            à cette adresse email");
        ControleurUtilisateur::redirection("utilisateur",
            "afficherFormulaireConnexion");
    }
    ControleurUtilisateur::afficherVue('vueGenerale.php', [
        "pagetitle" => "Récupérer mon compte",
        "cheminVueBody"=>"utilisateur/resultatResetCompte.php",
        "utilisateurs" => $utilisateurs
    ]);
}

```

Figure 12 : Code pour récupérer les mots de passe avec l'email

Cette fonction se trouve dans le ControleurUtilisateur à la ligne 354, c'est elle qui est appelée lorsque l'on clique sur "[Login et/ou mot de passe oubliés ?](#)". On peut voir que dans un premier temps, elle fait plusieurs vérifications, ce qui est une bonne chose (bien que ces vérifications ne devraient pas être dans la couche Contrôleur, mais dans la couche Service). Cependant, il n'y a aucune vérification que l'email appartient bel et bien à la personne qui la renseigne. Un meilleur système serait d'envoyer un email avec un lien unique de réinitialisation de mot de passe (il n'est normalement pas possible d'afficher le mot de passe, cela est rendu possible par le stockage en claire dans la base de donnée de ce dernier, cependant il ne doit apparaître que haché et il n'est pas possible de le "dé-haché").

### 3.3. Accès au mot de passe via un ordinateur partagé

Une autre faille de sécurité est présente dès la création d'un compte, tous les utilisateurs sont concernés et ne peuvent pas y remédier. En effet, à la création du compte, les cookies enregistrent le mot de passe et l'identifiant choisis par l'utilisateur. C'est alors qu'une fois arrivé sur la page du formulaire de connexion, les champs de l'identifiant et du mot de passe de l'utilisateur seront déjà pré-remplis.

Cette image, figure 13, nous montre bien que le mot de passe est enregistré en clair dans les cookies, et donc peut être accessible via des vols de cookies.

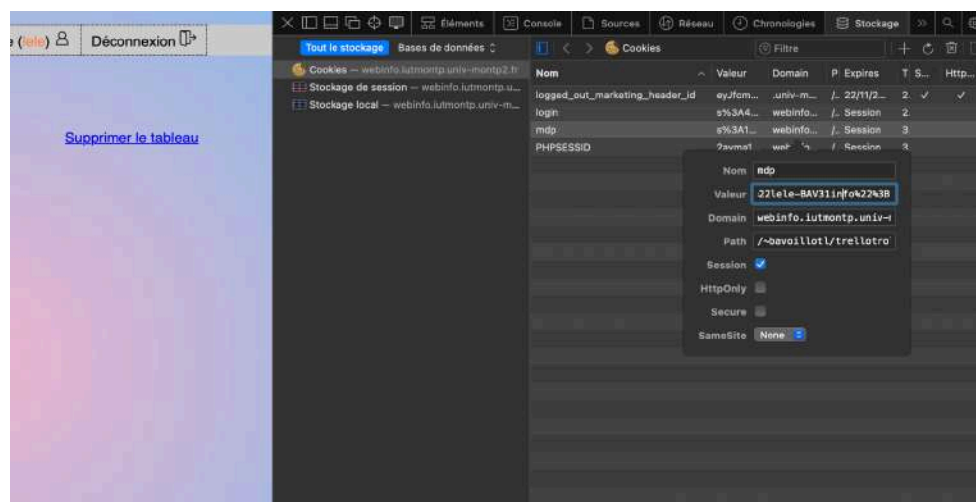


Figure 13 : Mot de passe stocké dans les cookies

Comme on peut le voir sur la figure 14, sans même enregistrer le mot de passe dans un gestionnaire exprès, les champs sont déjà pré-remplis.

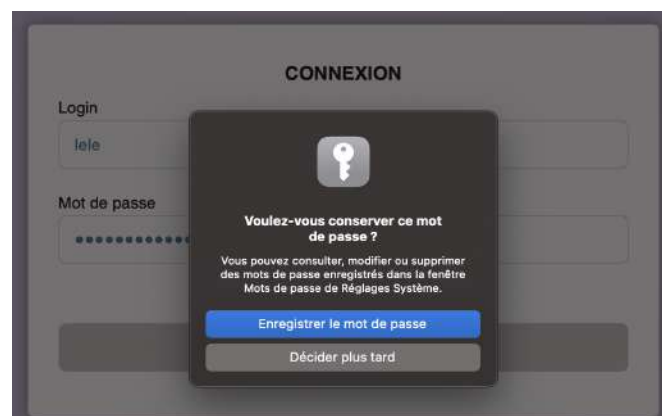


Figure 14 : Mot de passe prés remplit après une connexion

Stocker des informations sensibles comme les mots de passe dans des cookies expose ces informations à un risque accru de violation de la confidentialité. De plus, l'utilisateur n'a pas le contrôle sur les informations stockées dans son navigateur. Cela signifie que si l'utilisateur se connecte à partir d'un ordinateur partagé ou public, ses informations d'identification pourraient être accessibles à d'autres personnes qui utilisent le même navigateur.

Lorsqu'on clique sur "Mettre à jour le compte", la page du formulaire est donc déjà pré-rempli comme sur la figure 15 :



**MISE À JOUR DU PROFIL**

Login\*  
lele

Prenom\*  
bel

Nom\*  
lele

Email\*  
lele@gmail.com

Ancien mot de passe\*  
.....

Nouveau mot de passe\*  
6 à 50 caractères, au moins une minuscule, une majuscule et un caractère spécial

Figure 15 : Affichage de la mise à jour du profil

Si l'utilisateur souhaite modifier ses informations, une fois arrivé sur la page du formulaire de modification, l'ancien mot de passe est déjà pré rempli, il n'y a donc aucune vérification pour s'assurer que l'utilisateur connaît effectivement l'ancien mot de passe.

Cette faille est dû au code de la Figure 16

```
if ($succesSauvegarde) {  
    Cookie::enregistrer("login", $_REQUEST["login"]);  
    Cookie::enregistrer("mdp", $_REQUEST["mdp"]);  
    [...]  
}  
  
Cookie::enregistrer("mdp", $_REQUEST["mdp"]);  
  
ConnexionUtilisateur::connecter($utilisateur->getLogin());  
Cookie::enregistrer("login", $_REQUEST["login"]);  
Cookie::enregistrer("mdp", $_REQUEST["mdp"]);  
MessageFlash::ajouter("success", "Connexion effectuée.");
```

Figure 16 : Code du stockage du mot de passe dans les cookies

Ce code se trouve dans la classe `ControleurUtilisateur`, dans la fonction “créerDepuisFormulaire”, “connecté”, et “mettre à jour”. Une fois l’inscription réussie, alors les cookies enregistrent ce que l’utilisateur vient de rentrer pour le login et le mot de passe, mais aussi lors de la modification de son compte ainsi que la connexion.

L’enregistrement du mot de passe en clair se fait aussi dans la base de données. Une faille supplémentaire peut être exploitée. Si la base de données venait à se faire pirater, les hackers auraient accès à l’ensemble des comptes.

Pour éviter cette faille, il suffit de supprimer cette fonctionnalité où les cookies enregistrent le mot de passe, il y a d’autres gestionnaires de mot de passe très sécurisés qui s’en occupent.

## 3.4. Mauvaise redirection

Lorsqu'un utilisateur malveillant tente de supprimer une carte d'un tableau pour lequel il n'a pas les droits, le site affiche le message "Vous n'avez pas de droits d'édition sur ce tableau". Cependant, il redirige ensuite l'utilisateur vers le tableau en question en mode lecture seule.

Cette vulnérabilité est présente sur toutes les actions d'édition sur un tableau, telles que, `afficherFormulaireCreationColonne`, `supprimerColonne`, `creerCarte`, etc. Elle est présente dix fois dans le code, à chaque fois Cette approche facilite la découverte du code du tableau en testant différents identifiants de carte (`idCarte`) pour retrouver le `codeTableau` correspondant. Normalement, retrouver le `codeTableau` par force brute devrait être une tâche ardue, mais cette redirection permet de contourner cette difficulté. Par exemple, en essayant le lien suivant :

`http://trelotrolle-code-de-base/web/controleurFrontal.php?action=supprimerCarte&controleur=carte&idCarte=10`, le message "Vous n'avez pas de droits d'édition sur ce tableau" s'affiche, mais l'utilisateur est redirigé vers un lien semblable à celui-ci :

`http://trelotrolle-code-de-base/web/controleurFrontal.php?action=afficherTableau&controleur=tableau&codeTableau=e1c2acba25d1e204a9ade9767c9c17aa9e7c2d5ef6ec5c040a63b770192e0145`.

Ainsi, il devient aisé pour un script de tester les identifiants de carte de 1 à 10 000 et de stocker les URL vers lesquelles il est redirigé, permettant ainsi d'accéder à de nombreux tableaux auxquels l'utilisateur n'a en temps normal pas accès.

## 3.5. Mot de passe en clair dans la base de données

Il semble y avoir un attribut motDePasse dans la BD, en plus d'un attribut mdpHache ce qui est inquiétant niveau sécurité. Dans la BD, figure 17, nous avons deux attributs concernant le mot de passe. Il y a mdphache qui correspond aux mots de passe haché et mdp qui est le mot de passe en clair de l'utilisateur.

```
CREATE TABLE app_db (  
  login VARCHAR(30),  
  nom VARCHAR(30),  
  prenom VARCHAR(30),  
  email VARCHAR(255),  
  mdphache VARCHAR(255),  
  mdp VARCHAR(50),  
  idtableau INT,  
  codetableau VARCHAR(255),  
  titretableau VARCHAR(50),  
  participants jsonb,  
  idcolonne INT,  
  titrecolonne VARCHAR(50),  
  idcarte INT,  
  titrecarte VARCHAR(50),  
  descriptifcarte TEXT,  
  couleurcarte VARCHAR(7),  
  affectationscarte jsonb,  
  CONSTRAINT pk_db PRIMARY KEY (login, idcarte));
```

Figure 17 : Code de création de la Base de donnée

Stocker les mots de passe en clair dans une base de données est une faille de sécurité majeure et une violation flagrante des principes du RGPD (Règlement Général sur la Protection des Données). En effet, cette pratique expose les utilisateurs à des risques significatifs en matière de sécurité et de confidentialité des données. Le RGPD exige que les données personnelles soient traitées de manière sécurisée et protégées contre tout accès, divulgation ou altération non autorisés. Étant donné le mot de passe déjà haché, la seule chose à faire pour éviter cette faille est d'enlever l'attribut "mdp" dans la base de données.

## 4. Problèmes d'utilisation du site

---

### Démarche

Les problèmes d'utilisation du site sont les particularités que les développeurs ont oublié lors de sa création, ou bien mal pensé. Cela peut être aussi un problème de mise à jour sociétal aussi, comme avec les lois (RGPD), ou bien encore un changement de mode. En tout cas, le défaut, important ou non, doit être réparé. Nous devons donc d'abord les répertorier avant de se lancer tête baissée, car dans le cas contraire nous risquerions d'en manquer (trop concentré sur le code, moins sur l'interface).

En termes de démarche, et ce paragraphe sera aussi valable pour l'ergonomie, nous tentons plein de choses. Tout comme avec les failles, nous avons en tant que développeur déjà la puce à l'oreille concernant les points qu'il faut regarder. Une fois l'interface explorée, nous regardons le code pour trouver d'autres pistes qui nous ont échappé sur l'interface, et nous les exploitons pour en tirer des défaillances concrètes.

### 4.1. Problème sur l'affichage du login

Dans le menu de navigation, une fois connecté, on peut apercevoir que le login n'est pas toujours identique à celui choisi par l'utilisateur. En effet, les caractères spéciaux ne se convertissent pas en entités HTML.

Sur la figure 18, le login "MaylinEnzo4@" a été remplacé par "MaylinEnzo4%40". Le caractère "@" est devenu "%40" :



Figure 18 : Login de l'utilisateur mal affiché

Dans la figure 19, il y a le code qui comporte l'erreur :

```
$loginHTML =  
htmlspecialchars(ConnexionUtilisateur::getLoginUtilisateurConn  
ecte());  
$loginURL =  
rawurlencode(ConnexionUtilisateur::getLoginUtilisateurConnecte  
());?>  
<li>  
    <a  
href="controleurFrontal.php?action=afficherListeMesTableaux&co  
ntroleur=tableau">Mes tableaux</a>  
</li>  
<li>  
    <a  
href="controleurFrontal.php?action=afficherDetail&controleur=u  
tilisateur&login=<?= rawurlencode($loginURL) ?>">  
        Mon compte (<span><?= $loginURL ?></span>)   
    </a>  
</li>
```

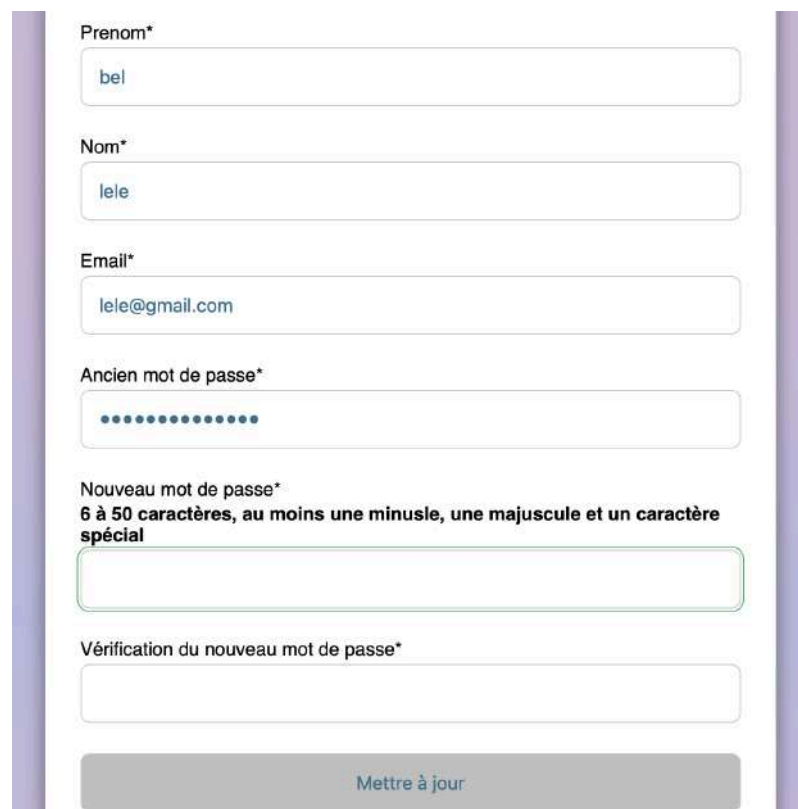
Figure 19 : Code de l'erreur de l'affichage du login

Observons le code de l'erreur situé dans la vue générale. Au lieu d'utiliser \$loginURL dans la balise <span>, il faudrait utiliser \$loginHTML. Cela garantira que le login de l'utilisateur est correctement échappé pour être affiché dans le HTML, évitant ainsi les problèmes de sécurité potentiels liés à l'injection de code malveillant (Vulnérabilités XSS) et aussi éviter un rendu inattendu dans le navigateur.



## 4.2. Obligation de remplir tous les champs

Une fois dans la page "Mon compte" et en cliquant sur "Mettre à jour le compte", l'utilisateur est redirigé vers la page de modification du profil. Cependant, on peut voir sur la figure 20 qu'il est indispensable de remplir tous les champs, y compris "Nouveau mot de passe" et "Vérification du nouveau mot de passe"; ce qui peut être une perte de temps pour l'utilisateur (et donc potentiellement une perte de clientèle) qui ne souhaite pas modifier son mot de passe, mais un autre champ.



The screenshot shows a web form for updating a user profile. It contains the following fields and elements:

- Prenom\***: A text input field containing the value "bel".
- Nom\***: A text input field containing the value "lele".
- Email\***: A text input field containing the value "lele@gmail.com".
- Ancien mot de passe\***: A password input field represented by a series of dots.
- Nouveau mot de passe\***: A text input field. Above it, a note specifies: "6 à 50 caractères, au moins une minuscule, une majuscule et un caractère spécial".
- Vérification du nouveau mot de passe\***: A text input field for confirming the new password.
- Mettre à jour**: A grey button at the bottom of the form.

Figure 20 : Champs mettre à jour du compte

Toujours dans cette page, il y a le login de l'utilisateur qui a la même apparence que tous les autres champs modifiables.

Cependant, il est impossible à l'utilisateur de le modifier. Une confusion est possible comme le montre la figure 21.

```
<input type="text" value="lele" id=" login_id" readonly>
```

Figure 21 : Code du champ du login

## 4.3. Affichage des pages

Sur la page “Liste des tableaux” nous avons la possibilité d’en créer autant que l’utilisateur le souhaite. Une fois en avoir créé un certain nombre nous pouvons voir qu’ils s’empilent tous. Il serait plus simple pour un utilisateur d’avoir ses tableaux sur plusieurs pages pour éviter qu’il se perde dans une page infinie. La figure 22 l’illustre bien.

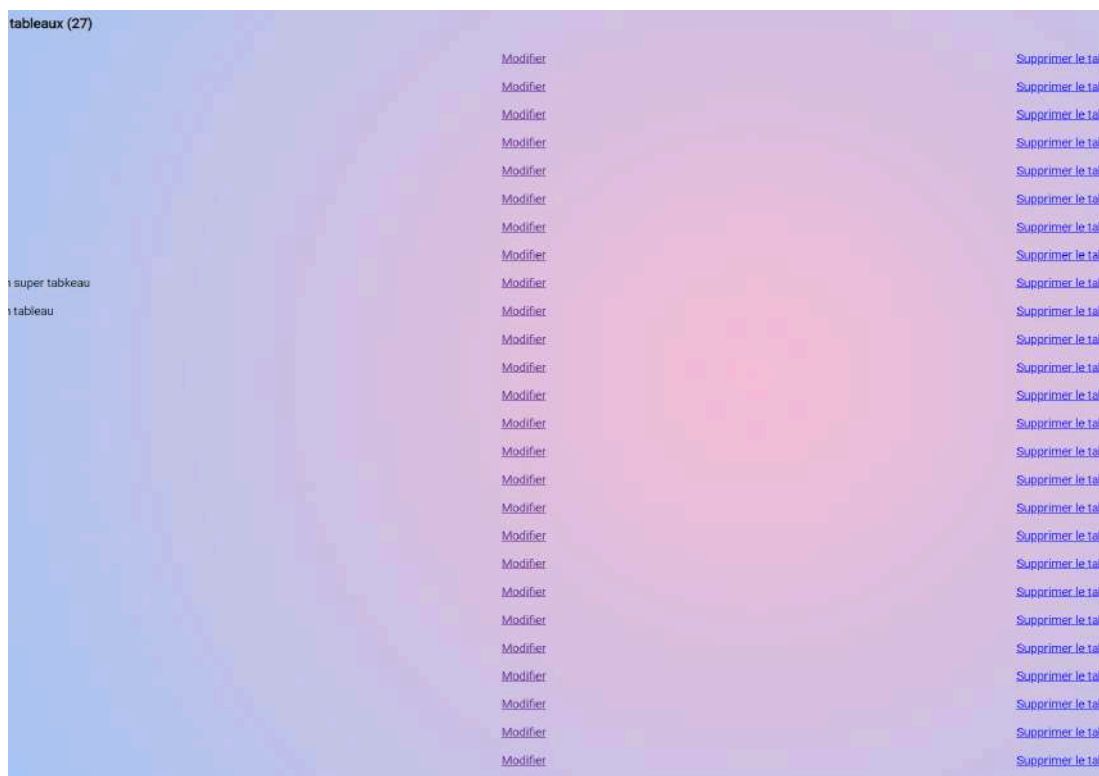


Figure 22 : Page liste des tableaux

## 5. Analyse ergonomique

---

### Démarche

Dans un environnement informatique dans lequel la concurrence en termes de sites est rude, l'importance de l'ergonomie devient centrale. En termes d'ergonomie, nous incluons dans cette partie tout ce qui concerne la satisfaction utilisateur. Notre analyse ne comprend pour l'instant que l'existant, et non la solution. Nous avons divisé cette analyse en autant de pages qu'il y a dans le site (accueil, inscription, connexion, "j'ai oublié mon mdp", gestion tableaux, édition tableaux et "mon compte"). Nous rappelons que notre but est de faire rester l'utilisateur un maximum sur le site, c'est donc sur cela que nous nous baserons pour déterminer les bons et mauvais points de l'ergonomie. Nous estimons également que notre cible est souvent amenée à faire de la gestion de projet, en entreprise (en équipe donc) en télétravail (voir introduction du doc). Idéalement, nous attendons également une telle satisfaction client qu'il parle autour de lui de notre site pour nous faire de la pub, nous utiliserons également ce point pour appuyer nos arguments.

### 5.1. L'accueil

Du point de vue de l'ergonomie, l'accueil a beaucoup d'espaces vides ainsi que du texte plus que des images. Le tout a un fond dégradé clair, qui donne un bon contraste avec la couleur noire du texte, de la navigation et du copyright. Cependant, pour que l'utilisateur. Cependant pour inciter les utilisateurs à faire de la publicité pour l'entreprise, il aurait été judicieux d'ajouter autant que possible le nom/logo, pour faciliter l'expansion de l'entreprise. De plus, nous ne trouvons pas non plus de titre / logo dans l'onglet en haut de la page, ce qui aurait été bien pour s'aligner avec le marché et retrouver facilement l'application web quand l'utilisateur a beaucoup d'onglets ouverts. Il faut tout de même noter que la navigation est intuitive, l'utilisateur sait directement à quoi s'attendre avec chaque bouton, et n'est pas surchargé d'information. L'utilisateur est même guidé avec des liens hypertexte, ce qui est un très bon point. Pour finir, la page d'accueil est responsive, ce qui facilite beaucoup les choses si l'on veut accéder au site depuis le téléphone. Seule la brutalité du texte peut déranger, il aurait été pertinent de rendre le tout un peu plus enfantin pour faire un effet de simplicité

proche de nos meilleurs concurrents (formes simples avec carrés, disques, ovales, etc. plutôt que lignes droites noires).

## 5.2. Inscription

L'inscription est l'une des pages les plus importantes pour nous. Si l'utilisateur franchit le pas en s'inscrivant sur le site, il y aura de bien meilleures chances qu'il y revienne par la suite. La page d'inscription a un très bon visuel, avec des formes simples (rectangles) centrées, avec une écriture bien contrastée, un éclairage des sections quand la souris passe dessus, et même guidé avec des exemples. Les titres des sections sont composés d'une '\*' quand l'information est obligatoire, ce qui est également un bon point, car nous nous alignons sur la concurrence. De plus, le bouton d'inscription change de couleur quand nous appuyons dessus, ce qui permet d'être renseigné sur la validité de notre opération. Quand un mot de passe est renseigné, celui-ci est caché, ce qui rassure l'utilisateur. Le fait que l'indication sur le mot de passe soit en gras de la même taille que le reste du texte donne de l'importance à cette partie ; cependant cela peut perturber l'utilisateur, car cette indication est visuellement plus importante que les titres des sections, il aurait donc fallu adapter la taille du texte pour faire un véritable effet d'indication. Notons que la page n'est pas du tout accessible depuis un téléphone.

## 5.3. Connexion

La page de connexion est tout aussi importante, parce que c'est elle qui permet à notre utilisateur d'accéder aux principales fonctionnalités du site. Tout comme la page d'inscription, elle n'est pas adaptée pour fonctionner visuellement sur téléphone. Cependant, elle comporte tous les mêmes avantages que la page d'inscription en termes d'ergonomie. Après s'être inscrit, il y a même une pré-complétion de nos identifiants sur la page, ce qui est excellent pour la simplicité du site. Il est également important de noter que l'utilisateur est informé de la validité de ses actions sur le site à l'aide de messages flashes colorés, ce qui est aussi un vrai plus.

## 5.4. J'ai oublié mon mot de passe

L'oubli de mot de passe est régulier chez les utilisateurs, et il est donc important de pouvoir le changer facilement et en toute simplicité. La page respecte ce principe : elle est simple et facile. L'utilisateur est même conseillé sur le choix de son mot de passe. C'est un aspect très positif du site. Cependant (je rappelle que nous parlons d'ergonomie), la confiance en termes de sécurité peut très vite s'évaporer si l'utilisateur se rend compte que son compte peut facilement être piraté si l'on connaît son adresse email, ou s'il se rend compte que les conseils vont complètement à l'encontre des principes de véritable sécurité. Mais si le conseil est pris comme une blague, cela peut aussi avoir un effet positif sur la proximité de l'utilisateur avec le site.

## 5.5. Gestion des tableaux

La gestion des tableaux se fait une fois l'utilisateur connecté, elle permet de naviguer entre les différents tableaux de l'utilisateur et est donc central en termes de simplicité. Sur cette page, la nouvelle navigation est très intuitive, tout comme la précédente, et présente les mêmes avantages / défauts que l'accueil du site. Il est précisé le nombre de tableaux, et pour chaque ligne, nous retrouvons la configuration d'un tableau (titre, modifier, supprimer). Tout cela se fait avec des liens hypertextes, ce qui permet la navigation certes, mais qui peut très vite devenir insupportable visuellement. Si nous possédons un grand nombre de tableaux, il devient en effet très difficile de garder l'œil sur la bonne ligne, et nous risquons par exemple de supprimer un tableau qu'on ne voulait pas (d'autant plus qu'il n'y a pas de demande de confirmation avant suppression). Dans le cas de beaucoup de tableaux, le bouton d'ajout se retrouve alors tout en bas de la liste, ce qui peut apporter de la frustration si l'on doit scroller trop longtemps. Il est aussi important de noter que la page est responsive même si ce n'est pas pensé idéalement.

## 5.6. Édition des tableaux

Parmi toutes les pages du site, c'est celle-ci qui fera la différence avec les concurrents. C'est elle qui facilitera la tâche à l'utilisateur, et c'est donc sur elle que l'utilisateur fondera sa satisfaction principale. La page est bien réussie. Elle est très visuelle bien qu'elle manque cruellement de couleurs. Les formes sont simples, la structure est vite compréhensible (administration à gauche et édition à droite). La gestion des membres est facile, avec un lien hypertexte qui redirige vers une page visuelle également. Toutes les sections éditables ont des boutons 'crayon' pour éditer, et un bouton 'x' pour supprimer, ce qui est très bien en termes de compréhension. De plus, des exemples sont pré-crées (cartes) pour permettre à l'utilisateur d'expérimenter. Pour continuer positivement, quand trop de colonnes sont créées, la section est défilable horizontalement bien que cela soit peu intuitif. La création de cartes est très simple, avec l'ombre des cartes qui apporte un confort visuel. Tout comme avec les initiales des membres affectés à une carte. Pour les points négatifs, la liste est longue. Commençons d'abord par le responsive qui ne permet pas du tout d'y accéder sur téléphone, puis les lignes noires à l'intérieur des cartes pour séparer le titre de la description, et des membres assignés qui portent à confusion visuellement, car l'on pourrait croire qu'il s'agit de plusieurs cartes différentes (les lignes ne servent à rien en fait). Les boutons sont visuels, mais trop petits. Il peut arriver qu'on supprime la carte sans le vouloir, alors qu'on voulait seulement la modifier (d'autant qu'il n'y a pas de confirmation avant suppression). Le tout est peu visuel, il serait bien que le bouton réagisse au passage de la souris (pour situer un peu mieux sur quel bouton on appuie). En termes fonctionnels, le fait de supprimer une carte supprime aussi la colonne, ce qui peut être gênant si la colonne avait pour ambition d'être utilisée à répétition. En ce même sens, la modification de la colonne sur laquelle se situent les cartes est peu pratique, même si cela n'a que peu d'importance. Pour finir, il est étrange de pouvoir modifier les informations de son compte via le tableau, cela porte atteinte à la cohérence du site et peut ainsi créer un effet négatif.

## 5.7. Gestion du compte

La gestion du compte est la partie la moins importante, mais gagne tout de même des points dans la satisfaction utilisateur. Contrairement aux pages d'inscription et de connexion, elle ne dispose pas d'arrière-plan qui met un contraste clair entre les sections et le background. De plus, les boutons sont d'une couleur différente aux sections décrites précédemment, ce qui donne de l'incohérence. Ces mêmes boutons ne sont pas tout à fait simples d'utilisation puisqu'il faut appuyer sur le texte pour lancer l'action. Cela peut porter à confusion, car si l'utilisateur clique et qu'il ne se passe rien, il pourrait croire qu'il n'y a pas accès. De plus, l'utilisation des textfields pour afficher les informations n'est pas non plus pertinente, puisque l'on est amené à croire que la modification se fait ici même. La mise à jour du profil, quant à elle, reprend l'interface de la connexion / inscription, et n'a donc rien de changeant par rapport à ces dernières. Pour finir, la gestion du compte comporte toutes les informations modifiables (sauf le login), et respecte ainsi le RGPD.

En somme, le site comporte de bons avantages techniques (guides, autocomplétion, etc.) mais a quelques lacunes visuelles qui pourraient être améliorées avec de la vulgarisation. Parfois, le site a quelques incohérences qui peuvent perturber l'utilisateur (par exemple l'édition du compte depuis le tableau) mais cela reste tout de même minime. L'expérience utilisateur est donc correcte, mais demande une restructuration pour un résultat encore plus optimal.

# 6. Conception

## 6.1. Entités du domaine

Pour mieux comprendre le fonctionnement et la structure de notre système logiciel, il est essentiel de définir et d'identifier les principales entités du domaine qui interagissent au sein de l'application web.

- **Utilisateur** : Représente les utilisateurs du site web.
- **Carte** : Représente une carte (unité d'information) affichée sur le site.
- **Colonne** : Représente une colonne dans un tableau.
- **Tableau** : Représente une collection de cartes organisées en colonnes.

## 6.2. Processus et interactions

Les quatre entités du domaine possèdent des interactions comme on peut voir dans la figure 23. Un utilisateur peut être un participant ou le propriétaire du tableau. Son rôle détermine les droits (CRUD) qu'il possède sur les autres entités du domaine. Plusieurs utilisateurs peuvent être affectés à une carte.

Aussi, un tableau possède plusieurs colonnes qui possèdent chacune plusieurs colonnes. La figure 25 permet une visualisation caricaturée des droits de l'utilisateur sur un tableau dont il a le code.

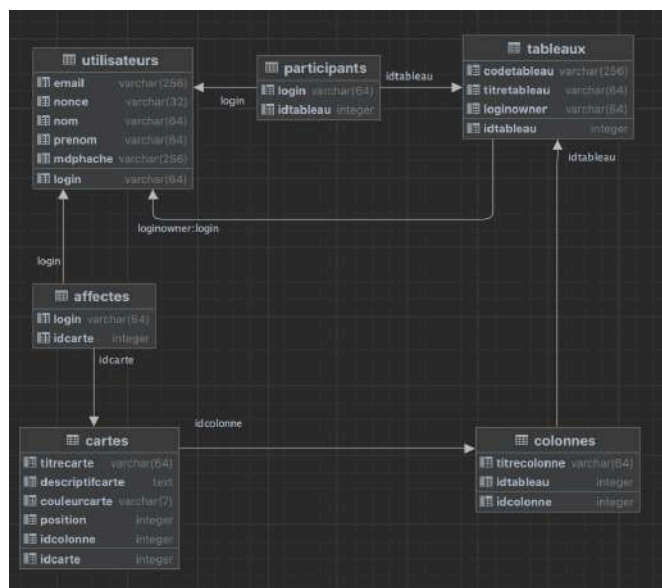


Figure 23 : Nouveau modèle E/A



Ces différentes interactions ne posent pas un problème lors de leur lecture ou de leur ajout, mais en posent un pour leur suppression. Des triggers ont donc été mis en place.

Par exemple, le trigger `avant_suppression_utilisateur` (voir figure 24), supprime de la base de données toutes les interactions d'un utilisateur avec les autres entités du domaine, avant la suppression de l'utilisateur.

```
-- Création de la fonction du trigger
supprimer_tableaux_participants_affectes_avant_utilisateur --
CREATE OR REPLACE FUNCTION
supprimer_tableaux_participants_affectes_avant_utilisateur()
  RETURNS TRIGGER AS $$
BEGIN
  -- Supprimer les tableaux associés à l'utilisateur qui va
  être supprimé
  DELETE FROM Tableaux
  WHERE loginOwner = OLD.login;
  -- Supprimer les participants associées à l'utilisateur qui
  va être supprimé
  DELETE FROM Participants
  WHERE login = OLD.login;
  -- Supprimer les affectes associées à l'utilisateur qui va
  être supprimé
  DELETE FROM Affectes
  WHERE login = OLD.login;

  RETURN OLD;
END;
$$ LANGUAGE plpgsql;
-- Création du trigger avant_suppression_utilisateur
CREATE TRIGGER avant_suppression_utilisateur
  BEFORE DELETE ON Utilisateurs
  FOR EACH ROW
EXECUTE FUNCTION
supprimer_tableaux_participants_affectes_avant_utilisateur();
```

Figure 24 : Trigger `avant_suppression_utilisateur`

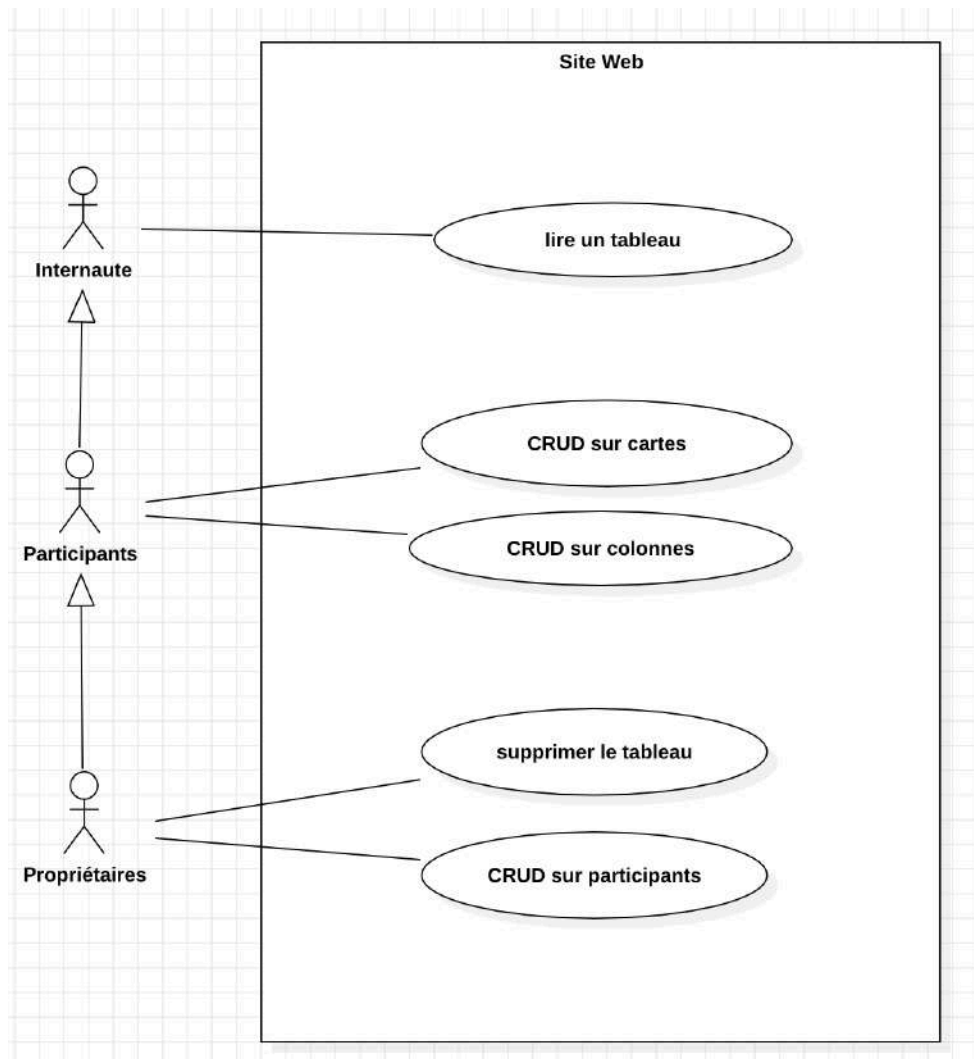


Figure 25 : Diagramme de cas d'utilisation d'un tableau

## 6.3. Choix technologiques

Pour le développement de notre système de gestion de tableaux, nous avons opté pour une combinaison de technologies robustes et éprouvées. Le backend de l'application est développé en PHP, un langage reconnu pour sa robustesse et sa fiabilité dans le domaine du développement web. Quant au frontend, nous utilisons JavaScript, notamment avec une API REST qui permet une communication client / serveur beaucoup plus appréciable (requêtes plus courtes, quantité d'informations en transit moins importante, interface davantage interactive). C'était également pour nous l'occasion de mettre en pratique nos compétences nouvelles acquises en cours de développement web. La gestion des données est assurée par PostgreSQL, une base de données relationnelle reconnue pour sa performance, sa fiabilité et sa capacité à gérer des données complexes. Cette solution offre une excellente compatibilité avec

nos besoins en matière de stockage et de gestion des informations liées aux utilisateurs, aux tableaux, aux colonnes et aux cartes.

Pour le développement et la gestion du code, notre équipe utilise PHPStorm comme environnement de développement intégré (IDE) pour le développement en PHP. Ce choix s'est imposé naturellement en raison de sa praticité, de ses fonctionnalités avancées et de la familiarité de l'équipe avec cet outil.

Enfin, pour garantir une gestion optimisée des environnements de développement, de test et de production, nous avons adopté Docker pour la conteneurisation de nos applications. Cette technologie nous permet de créer, de déployer et de gérer des conteneurs de manière efficace, assurant ainsi une cohérence et une portabilité optimales de nos applications tout au long du cycle de vie du projet.

## 6.4. Suppression des failles de sécurité

**1.** Afin d'améliorer l'application, il est essentiel d'avoir un site sécurisé, c'est pourquoi nous avons pris des mesures rigoureuses pour éliminer toutes les vulnérabilités de sécurité que nous avons identifiées. En mettant en œuvre une série de mesures préventives, nous nous sommes assurés que notre site offre un environnement en ligne robuste et protégé.

Nous avons dans un premier temps entrepris une démarche proactive en éliminant les failles d'échappement de caractères de notre site. Ces vulnérabilités potentielles, qui permettraient à des tiers malveillants d'insérer du code indésirable ou nuisible dans nos systèmes, ont été identifiées et corrigées de manière exhaustive. En prenant cette mesure préventive, nous renforçons la sécurité de notre plateforme en garantissant que les données et les interactions des utilisateurs sont protégées contre toute exploitation indésirable. Notre engagement envers la sécurité de nos utilisateurs est fondamental, et nous continuerons à surveiller et à améliorer nos pratiques de sécurité pour maintenir un environnement en ligne sûr et fiable.

Pour cela, nous avons décidé stratégiquement de migrer nos anciennes vues vers des gabarits Twig. Par conséquent, le fichier précédemment nommé "vueGenerale.php" a été transformé en "base.html.twig", les autres fichiers Twig hériteront tous de celui-ci.

Cette transition vers l'utilisation de Twig offre plusieurs avantages, notamment une meilleure gestion des échappements de caractères, ce qui réduit considérablement le risque de failles de sécurité.

Le code de la figure 26 assure maintenant l'échappement de caractères en HTML.

```
<div class="colonne">
  <div class="titre icons_menu">
    <span>{{ colonne.getTitreColonne() }}</span>
```

*Figure 26 :Correction échappement de caractères*

Nous pouvons tester en mettant une balise script pour le nom d'une colonne, les caractères HTML seront donc bien échappés.

**2.** Face à la problématique liée à l'affichage direct des mots de passe lors de l'utilisation de la fonction "Mot de passe oublié", et confrontés à des contraintes de temps, nous avons été dans l'impossibilité d'établir un service de réinitialisation sécurisé par email. Dans l'optique de prioriser la sécurité des comptes utilisateurs, cette fonctionnalité a été intégralement désactivée. Cette action représente une solution provisoire, anticipant la future mise en place d'un mécanisme de réinitialisation conforme aux standards de sécurité, garantissant ainsi une gestion optimale de la confidentialité des utilisateurs.

# 7. Réalisation

---

## 7.1. Architecture

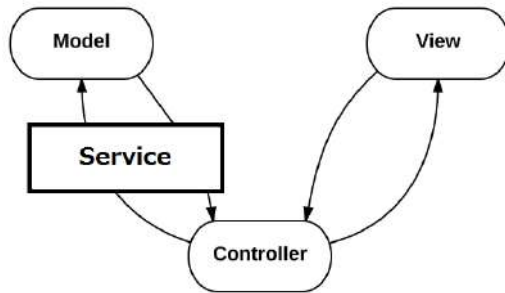


Figure 27 : Schéma MVCS<sup>2</sup>

L'adoption antérieure du modèle MCV entravait la conformité aux principes SOLID, ce qui a nécessité l'intégration d'une couche de services supplémentaires pour passer à une architecture MCVS<sup>2</sup>. L'adoption de l'architecture MVCS<sup>2</sup> pour le développement d'un site web procure de multiples bénéfices essentiels à la réalisation d'applications solides, modulaires et facilement maintenables.

Premièrement, comme démontré par la figure 27, le modèle MVCS<sup>2</sup> présente une démarcation distincte des responsabilités, permettant ainsi à chaque composant du système de se focaliser sur une fonction spécifique. Le modèle s'occupe de la gestion des données ainsi que de la logique métier, pendant que la Vue se consacre à l'affichage des données à l'utilisateur. Le Contrôleur assure la navigation et module les interactions entre le modèle et la vue, tandis que la couche Service gère les opérations métier complexes et facilite les échanges avec des ressources externes.

Cette structuration en couches distinctes améliore grandement la collaboration au sein des équipes de développement, en permettant à chaque développeur de se concentrer sur des sections précises de l'application sans interférer avec les autres. Elle favorise également la réutilisation du code, renforce la modularité et simplifie le processus de maintenance, en facilitant notamment les mises à jour ou le remplacement sélectif des composants du système sans affecter le reste de l'application.

## 7.2. Routing

Fonctionnellement, le site n'est pas encore tout à fait stable. Il est composé de deux types de connexions (Sessions et JWT), deux types de contrôleurs (Interface et API), et cela n'est pas tout à fait optimisé. La connexion sessions est la connexion qui se fait pour l'utilisateur du site et la JWT est celle qui est nécessaire pour l'utilisateur de l'API.

L'application qui existait avant le réusinage n'utilisait que des *query strings* en php ainsi que du HTML et CSS. Nous avons apporté au projet un réusinage au niveau des routes, en transformant les query strings en RouteurURL. Pour ce faire, nous avons utilisé les composants de symfony composer : symfony/http-foundation et symfony/routing. En les utilisant, nous mettons en place un système de routage qui nous permet de définir des URL claires pour chaque page de notre site, simplifiant la navigation pour les utilisateurs et améliorant l'expérience utilisateur globale. Nous avons fait ceci pour les différents avantages que cela offre, et nous avons également mis en place un système d'API afin de minimiser la lourdeur des requêtes, et faire fonctionner le site avec une partie cliente mieux développée<sup>[3][4]</sup>. Il se trouve cependant que le temps a cruellement manqué, et que le site n'exploite pas pleinement les avantages de l'API. En effet, seules les fonctions de suppression de tableau, carte et colonne, ainsi que la fonction d'ajout de colonne ont pu être faites. Mais cela est largement améliorable, par exemple avec la fonctionnalité de modification de ces attributs (en faisant des popups JavaScript), aussi au niveau de l'ajout des membres, modification des titres des attributs et j'en passe, que nous aurions beaucoup aimé faire.

# 8. Validation, résultats et perspectives

## 8.1. Validation

La validation de notre système a été effectuée selon un protocole rigoureux visant à garantir l'atteinte des objectifs de qualité, de performance et de sécurité. Nous avons adopté une approche holistique en effectuant des tests fonctionnels, des évaluations de performance et des analyses de sécurité pour s'assurer une conformité aux exigences. Grâce à l'implémentation d'une suite exhaustive de tests automatisés, nous avons pu atteindre une grande couverture de test, encapsulant l'ensemble des composants du système tels que les repositories, les entités et les services, voir figure 28. Cette stratégie méthodique a permis d'assurer une fiabilité et l'efficacité de l'application dans divers scénarios d'utilisation.

▼	Folder	DataObject	100% files, 98% lines
	File	Carte.php	100% lines
	File	Colonne.php	100% lines
	File	InterfaceDataObject.php	
	File	Tableau.php	96% lines
	File	Utilisateur.php	100% lines
>	Folder	HTTP	50% files, 33% lines
▼	Folder	Repository	100% files, 52% lines
	File	AbstractRepository.php	32% lines
	File	CarteRepository.php	46% lines
	File	CarteRepositoryInterface.php	
	File	ColonneRepository.php	50% lines
	File	ColonneRepositoryInterface.php	
	File	ConnexionBaseDeDonnees.php	20% lines
	File	ConnexionBaseDeDonneesInterface.php	
	File	RepositoryGeneriqueInterface.php	
	File	TableauRepository.php	80% lines
	File	TableauRepositoryInterface.php	
	File	UtilisateurRepository.php	84% lines
	File	UtilisateurRepositoryInterface.php	
▼	Folder	Service	100% files, 88% lines
>	Folder	Exception	
	File	CarteService.php	92% lines
	File	CarteServiceInterface.php	
	File	ColonneService.php	90% lines
	File	ColonneServiceInterface.php	
	File	TableauService.php	87% lines
	File	TableauServiceInterface.php	
	File	UtilisateurService.php	85% lines
	File	UtilisateurServiceInterface.php	

Figure 28 La couverture des tests

La figure 28 représente la couverture que les tests ont sur notre projet.

Il est à préciser que certains pourcentages sont faibles car certaines classes ne sont pas entièrement vérifiées et c'est en partie dû à l'IDE. Parfois celui-ci va compter les sauts de ligne ou ne pas compter les lignes avec des accolades ce qui fait baisser le pourcentage de couverture des tests.

L'environnement Docker a joué un rôle essentiel dans l'optimisation de la phase de validation en fournissant un cadre homogène. Cette standardisation a permis de minimiser les disparités entre les environnements de développement, de test et de production.

La méthode de validation adoptée, combinée à la puissance et à la flexibilité offertes par Docker, a été déterminante dans la création d'un système robuste, performant et sécurisé. Cette approche garantit non seulement la satisfaction des critères de qualité initiaux, mais ouvre également la voie à des améliorations continues et à l'évolution future de l'application.

## 8.2. Résultats

La validation du projet a suivi une approche méthodique, en se basant sur l'analyse de l'existant, la normalisation de la base de données, l'optimisation de l'architecture de l'application, l'amélioration de la sécurité et l'analyse ergonomique. Cette démarche a permis de mettre en évidence la conformité de l'application aux objectifs fixés, tout en identifiant les axes d'amélioration.

**Base de données :** Le processus de normalisation a permis de résoudre les problèmes de redondance et d'incohérence, menant à une structure de données optimisée. La mise en œuvre d'une conception de base de données normalisée a contribué à l'efficacité des opérations CRUD, tout en simplifiant la maintenance du système Figure 23.

**Architecture de l'application :** L'adoption des principes DRY<sup>1</sup> et SOLID a amélioré la qualité du code et facilité sa maintenance. L'intégration d'API REST a permis de structurer les interactions entre le frontend et le backend de manière efficace, tout en offrant une expérience utilisateur fluide.

**Sécurité :** Les tests de sécurité ont permis d'identifier et de corriger plusieurs vulnérabilités, notamment celles liées à l'injection XSS et à la gestion des authentifications. L'accent mis sur la sécurité a renforcé la confiance dans l'utilisation de l'application.



En conclusion, le résultat de la validation du projet montre une amélioration significative de l'application tant sur le plan fonctionnel que technique. Les efforts déployés pour optimiser la base de données, l'architecture de l'application, la sécurité et l'ergonomie ont porté leurs fruits, ouvrant la voie à de futures évolutions. La mise en place d'un cadre de test rigoureux et l'utilisation de technologies modernes telles que Docker ont été des facteurs clés dans le succès de cette phase de validation.

## 8.3. Perspective

### 1. Extension des Fonctionnalités

Tel que le site est fait lors de ce rendu, il est tout à fait possible de poursuivre son amélioration. On peut déjà remarquer quelques défauts qui sont améliorables, tel que le temps de chargement de la page qui est très long, réglable par une amélioration de l'API, et des extensions au niveau de l'expérience utilisateur, comme avec le déplacement drag-and-drop des cartes au sein même du tableau, qui pourrait se faire avec davantage de JavaScript en communication avec l'API qui est déjà prête à accueillir la fonctionnalité. Voyons donc ici les perspectives d'avenir de l'application *Trello-Trollé*.

Le site comporte également beaucoup de tests, permettant de couvrir totalement (ou presque) le code, mais davantage pourraient être fait. En effet, au niveau de la sécurité du site, de l'API, ou bien simplement des services par exemple, d'autres tests pourraient s'ajouter. Il est également important de noter que certains tests ne passent pas, que nous avons volontairement choisi de ne pas régler faute de temps. Une perspective serait donc un réusinage du code pour en améliorer sa qualité, après avoir terminé l'API. Vous trouverez par ailleurs une branche "generalisation\_service" dans laquelle un prototype de généralisation des services a été faits, ce qui pourrait être intéressant d'implémenter. Pour continuer dans cette idée, il y a également des fonctionnalités qui n'ont pas pu être réusinées, telle que la modification de mot de passe par exemple, qui avant n'était pas du tout sécurisée. Afin d'améliorer encore le site, il faudrait pour cette fonctionnalité ajouter un token dans la Base de Données pour ainsi créer un lien de réinitialisation temporaire de mot passe (à envoyer par mail) pour l'utilisateur qui le demande.

En termes visuels et d'interactions, le site a aussi largement de quoi être amélioré. Nous aurions adoré ajouter par exemple un drag-and-drop dans les tableaux pour améliorer l'expérience utilisateur, ou une fonctionnalité de modification de fond d'écran pour les tableaux, des styles d'écritures plus adaptées, des formes plus enfantines... en fait, tout ce qui a été noté dans l'analyse ergonomique.

En somme, beaucoup de perspectives d'amélioration sont possibles. Le projet a besoin de temps pour davantage de qualité, et peut-être à l'avenir, gagner en popularité.

## 9. Organisation du projet

---

### 9.1. Outils de communication

Les outils de communication que nous avons choisis pour ce projet ont été sélectionnés en fonction de leur adaptabilité, leur accessibilité et leur efficacité dans différents contextes de travail.

Trello a été privilégié pour la répartition des tâches et le suivi de l'avancement en raison de sa simplicité d'utilisation et de sa fonctionnalité collaborative. Il permet une visualisation claire des responsabilités de chacun et facilite la coordination des efforts.

Instagram a été utilisé pour la communication interne en raison de sa popularité parmi les membres de l'équipe, ce qui favorise une interaction rapide et informelle. C'est un outil intuitif qui permet de partager des informations de manière visuelle et interactive, tout en renforçant les liens amicaux au sein de l'équipe.

Discord s'est imposé comme un choix naturel pour les appels, le partage d'écran et de ressources diverses (liens, bouts de code, images, etc.) en raison de sa polyvalence et de ses fonctionnalités avancées. Il offre un environnement convivial pour la collaboration en temps réel.

En ce qui concerne les rencontres en présentiel, nous avons opté pour des lieux flexibles tels que des salles libres ou chez un membre du groupe, car ils offrent un cadre confortable et décontracté pour des discussions productives entre membres de l'équipe.

Enfin, Google Docs a été retenu pour les rendus tels que le rapport d'analyse et technique en raison de sa facilité d'accès, de sa compatibilité avec différents types de fichiers et de sa capacité à permettre une édition collaborative en temps réel.

En somme, ces outils ont été choisis pour leur gratuité, leur simplicité d'utilisation, leur adaptabilité à différents besoins de communication et leur capacité à favoriser une collaboration efficace au sein de l'équipe.

## 9.2. Méthode de développement

Tout au long du projet, nous avons adopté une approche de développement incrémental. Cette méthode se caractérise par la réalisation successive de petites étapes ou "incréments" pour atteindre un objectif global.

Aucune tâche spécifique n'a été assignée pour ce projet ; chaque membre de l'équipe était libre de contribuer selon ses préférences et de se concentrer sur la partie qu'il souhaitait réaliser. Malgré cette liberté, une communication régulière a été maintenue entre les membres de l'équipe afin de se tenir mutuellement informés de l'avancement et de coordonner efficacement nos efforts.

## 9.3. Retour critique

### **BAVOILLOT Léanne :**

Mon avis sur la SAE est mitigée, le temps réduit nous a obligé à directement régler les erreurs qu'on pouvait trouver dans le code, et donc pas privilégier l'organisation et la division des tâches, certains s'occupaient d'une partie du code bien précis (Test, API) et d'autres que sur le refactor... on ne pouvait donc pas toucher à tout. Malheureusement, on n'a pas pu faire tout ce qu'on voulait et gérer l'organisation comme il le fallait par cause de manque de temps... Malgré ça, il y avait beaucoup de différentes tâches à faire, on a ainsi pu améliorer plusieurs compétences dans ce projet.

### **CAPO Mathys :**

Personnellement, je ne pense pas avoir beaucoup appris à cause du temps imparti, mais j'ai pu au contraire fortifier et valider mes compétences.

Je trouve qu'au sein de l'équipe, il y a eu une super cohésion et communication.

Les aspects négatifs sont pour moi le fait que l'on ait codé sur peu de branche et qu'au final de nombreux tests ne passent plus.

### **DE MURCIA Enzo :**

La contrainte majeure que j'ai rencontrée au cours de ce projet a été le manque de temps, ce qui a limité mon engagement à la hauteur de mes aspirations. La simultanéité de multiples engagements scolaires a représenté un obstacle significatif à mon investissement dans le projet.

**JOST Victor :**

J'ai bien aimé le principe du projet, le fait qu'il y ait déjà une base à réusiner, des fonctions sur lesquelles s'appuyer qui sont déjà faites... et une mise en pratique simple des différents TD. J'aurais vraiment aimé avoir davantage de temps pour faire une meilleure API, et une meilleure interactivité et un meilleur visuel. J'en garde tout de même une bonne expérience.

**MAYLIN Enzo :**

Toutes les SAE ont enrichi mes compétences. Les cours et travaux dirigés m'ont fourni une fondation solide basée sur des principes à suivre. Et les SAE me donnaient une certaine liberté essentielle pour que j'apprenne en m'amusant. Cependant, le défi de cette SAE n'était pas centré sur l'apprentissage ou la création, mais plutôt sur la capacité à recracher en un temps court ce qui a été abordé lors des TD.

## 9.4. Bilan critique

Concernant les points à améliorer, plusieurs membres de l'équipe ont mentionné des contraintes liées au temps et à l'organisation du projet. Le manque de temps imparti a été un frein significatif pour plusieurs d'entre nous, limitant ainsi notre capacité à approfondir certains aspects techniques et à gérer efficacement la répartition des tâches. Cette contrainte a également eu un impact sur la qualité des tests et du code.

Par ailleurs, bien que la communication et la cohésion au sein de l'équipe aient été globalement positives, il serait bénéfique d'améliorer la coordination et la collaboration pour une meilleure répartition des responsabilités et une optimisation du workflow. Enfin, certains membres auraient souhaité avoir davantage de liberté et de flexibilité dans l'organisation du projet, notamment en ce qui concerne les objectifs à atteindre.

En conclusion, malgré les défis rencontrés tels que les contraintes de temps et les défis organisationnels, le projet a été une expérience enrichissante pour tous les participants. Chacun a pu renforcer ses compétences existantes et en acquérir de nouvelles grâce à la diversité des tâches et des responsabilités. La communication au sein de l'équipe a été essentielle, bien que certains regrettent le manque de temps pour approfondir certains aspects du projet. Globalement, le projet a été perçu comme une expérience positive, combinant apprentissage autonome et collaboration.

# 10. Conclusion

---

Le projet avait pour but un apprentissage approfondi des compétences précédemment acquises dans les travaux dirigés, soient l'usage de Twig, du routage par chemins, API REST et l'usage de javascript pour la dynamisation du site. D'autres objectifs tels que l'acquisition d'expérience dans la gestion de projet, de communication, gestion de conflit et management sont aussi notables. Et il y a également un objectif d'acquisition du BUT qui entre en jeu, bien que cela soit subjectif comparé au reste.

Durant cette courte période durant laquelle nous avons travaillé sur le projet, la plupart des objectifs ont été atteints pour les différents membres du groupe. Mais comme toute chose, certaines choses n'ont pas marché aussi bien que ce que l'on aurait souhaité. Le temps a manqué, et nous avons dû rendre un travail que nous pensions insuffisant. Cependant, c'est aussi cela qui nous permet d'avancer, le fait que nous avons gagné de l'expérience.

En termes de perspectives, que nous avons détaillé plus tôt, le projet est ouvert à de grandes améliorations. Cela est surtout dû au fait que nous n'avons pas pu en faire tout ce que nous aurions voulu. Pour plus détails, voir *Perspectives*.

En somme, le projet nous a, à tous, fait comprendre d'une façon beaucoup plus précise les technologies précédemment citées. Aujourd'hui, nous saisissons un peu mieux l'importance d'un code de qualité, qui facilite grandement la résolution de problèmes internes au code. Au-delà de l'aspect technique, l'expérience en équipe fût très enrichissante. Nous avons tous des personnalités différentes, qui n'ont pas toujours l'habitude de se mélanger pour rendre un travail qui nous convient à tous. C'est cela même que nous avons travaillé lors de toute la durée du projet.

En ce qui concerne le dernier objectif, nous espérons que notre travail saura éveiller en vous la satisfaction suffisante à l'accomplissement de ce dernier.

# 11. Les sources

---

[1] PostgreSQL, "CREATE TRIGGER." [En ligne]. Disponible :  
<https://www.postgresql.org/docs/current/sql-createtrigger.html>

[2] PostgreSQL, "CREATE PROCEDURE." [En ligne]. Disponible :  
<https://www.postgresql.org/docs/current/sql-createprocedure.html>

[3] Romain Lebreton, "R.4.01 - Développement Web - JavaScript." [En ligne]. Disponible :  
<https://romainlebreton.github.io/R.4.01-DeveloppementWeb-JavaScript/>

[4] Romain Lebreton, "R4.A.10 - Complément Web." [En ligne]. Disponible :  
<https://romainlebreton.github.io/R4.A.10-ComplementWeb/>