



RAPPORT FINAL DE SAÉ



Trello - Trollé

Réalisé par
Eliott BASSIER
Zakaria HASSAIRY
Ilan VELTER
Yann THOMAS

Projet encadré par
Cyril NADAL

Pour l'obtention du BUT 2

ANNÉE UNIVERSITAIRE 2023-2024

Remerciements

Avant de détailler notre projet de fin d'études, il nous tient à cœur de remercier toutes les personnes ayant contribué de près ou de loin à la réalisation de notre projet, sans qui le travail aurait été bien plus difficile.

Nous exprimons nos vifs remerciements à notre encadrant pédagogique M. Nadal pour son soutien, sa disponibilité et son aide tout au long de nos travaux.

Résumé

Le projet Trello-Trollé a été réalisé dans le cadre de la SAE* 4.A1 de la formation de BUT Informatique de l'IUT de Montpellier-Sète.

Ce projet s'est déroulé en deux grandes parties: une première consistait à analyser l'application web Trello-Trollé qui nous était donnée. Cette application comportait de nombreux problèmes, que ce soit des failles de sécurité, des problèmes d'utilisation, une architecture de code mal conçue...

Ensuite, dans la 2e partie du projet, soit la partie concernée par ce rapport final, nous avons corrigé lesdits problèmes et changé le comportement de certaines fonctionnalités afin d'améliorer l'expérience utilisateur.

Sommaire

1. Table des figures.....	5
1.1. Base de données :.....	5
1.1.1. Ancienne Base de Données.....	5
1.1.2. Nouvelle Base de Données.....	5
1.2. Stockage des Données Sensibles.....	6
1.2.1. Ancien stockage des données de connexion à la BDD.....	6
1.2.2. Nouveau stockage des données de connexion à la BDD.....	7
1.2.3. Ancien Stockage des mots de passe.....	7
1.2.4. Nouveau Stockage des mots de passe.....	8
1.3. Gestion de Projet.....	8
1.3.1. Tableau de gestion des tâches.....	8
1.3.2. Tableau de gestion de tâches (fin de projet).....	9
2. Glossaire.....	10
3. Introduction.....	12
4. Analyse.....	13
4.1. Problèmes liés à la base de données.....	13
4.1.1. Normalisation.....	13
4.2. Failles de sécurité.....	13
4.2.1. Mot de passe des utilisateurs.....	13
4.2.2. Stockage des informations de connexion.....	14
4.3. Fonctions incompréhensibles et/ou non commentés.....	14
4.4. Problèmes du site.....	15
5. Rapport technique.....	17
5.1. Normalisation.....	17
5.2. Faille de sécurité.....	17
5.2.1. Stockage des informations de connexion.....	17
5.2.2. Fonctions incompréhensibles et/ou non commentés.....	18
5.2.3. Problèmes du site.....	18
6. Méthodologie et Organisation du Projet.....	20
6.1. Gestion d'équipe.....	20
6.2. Méthode de développement et outils.....	20
6.3. Bilan critique.....	21
7. Conclusion.....	22
8. Bibliographie.....	23
8.1. Bibliothèques de code utilisées.....	23

1. Table des figures

1.1. Base de données :

1.1.1. Ancienne Base de Données

```
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));
```

Ancienne base de données fournie pour la phase d'audit du projet.

1.1.2. Nouvelle Base de Données

```
CREATE TABLE Trello_Utilisateur(  
  login VARCHAR(30),  
  nom VARCHAR(30),  
  prenom VARCHAR(30),  
  email VARCHAR(255),  
  mdphache VARCHAR(255),  
  PRIMARY KEY(login),  
  UNIQUE(email)  
);  
  
CREATE TABLE Trello_Tableau(  
  idTableau SERIAL,  
  codeTableau VARCHAR(255),  
  titreTableau VARCHAR(50),  
  loginProprietaire VARCHAR(30) NOT NULL,  
  PRIMARY KEY(idTableau),  
  UNIQUE(codeTableau),  
  FOREIGN KEY(loginProprietaire) REFERENCES Trello_Utilisateur(login) ON DELETE CASCADE ON UPDATE CASCADE  
);  
  
CREATE TABLE Trello_Colonne(  
  idcolonne SERIAL,  
  titrecolonne VARCHAR(50),  
  idTableau INT NOT NULL,  
  ordreColonne INT NOT NULL DEFAULT 1,  
  PRIMARY KEY(idcolonne),  
  FOREIGN KEY(idTableau) REFERENCES Trello_Tableau(idTableau) ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```

CREATE TABLE Trello_Colonne(
    idcolonne SERIAL,
    titrecolonne VARCHAR(50),
    idTableau INT NOT NULL,
    ordreColonne INT NOT NULL DEFAULT 1,
    PRIMARY KEY(idcolonne),
    FOREIGN KEY(idTableau) REFERENCES Trello_Tableau(idTableau) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Trello_Carte(
    idCarte SERIAL,
    titreCarte VARCHAR(50),
    descriptifCarte TEXT,
    couleurCarte VARCHAR(7),
    idcolonne INT NOT NULL,
    ordreCarte INT NOT NULL DEFAULT 1,
    PRIMARY KEY(idCarte),
    FOREIGN KEY(idcolonne) REFERENCES Trello_Colonne(idcolonne) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Trello_EtreMembre(
    login VARCHAR(30),
    idTableau INT,
    PRIMARY KEY(login, idTableau),
    FOREIGN KEY(login) REFERENCES Trello_Utilisateur(login) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(idTableau) REFERENCES Trello_Tableau(idTableau) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Trello_EtreAffecte(
    login VARCHAR(30),
    idCarte INT,
    PRIMARY KEY(login, idCarte),
    FOREIGN KEY(login) REFERENCES Trello_Utilisateur(login) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(idCarte) REFERENCES Trello_Carte(idCarte) ON DELETE CASCADE ON UPDATE CASCADE
);

```

Résultat de la décomposition de l'ancienne base de données grâce au théorème de Casey-Delobel.

1.2. Stockage des Données Sensibles

1.2.1. Ancien stockage des données de connexion à la BDD

```

static private array $configurationBaseDeDonnees = array(
    'nomHote' => '162.38.222.142',
    'nomBaseDeDonnees' => 'iut',
    'port' => '5673',
    'login' => 'a_completer',
    'motDePasse' => 'a_completer'
);

```

Stockage des données de connexion à la base de données PostgreSQL non sécurisé

1.2.2. Nouveau stockage des données de connexion à la BDD

```
/**
 * Tente de récupérer et de charger les variables d'environnement contenues dans le .env à la racine du projet, s'il existe.
 * Si le chargement échoue, les valeurs par défaut sont conservées (logins serveur IUT)
 * @return bool vrai si le chargement s'est fait avec succès, faux sinon.
 */
!usage  🧑  Ilan Velter
static public function loadEnv(): bool
{
    if (!file_exists( filename: __DIR__ . '/.././.env')) {
        return false;
    }
    $env = parse_ini_file( filename: __DIR__ . '/.././.env');
    if (!isset($env["nomHote"]) || !isset($env["nomBaseDeDonnees"]) || !isset($env["port"]) || !isset($env["login"]) || !isset($env["motDePasse"])) {
        return false;
    }

    self::$configurationBaseDeDonnees = array(
        'nomHote' => $env["nomHote"],
        'nomBaseDeDonnees' => $env["nomBaseDeDonnees"],
        'port' => $env["port"],
        'login' => $env["login"],
        'motDePasse' => $env["motDePasse"]
    );
    ConfigurationSite::setSource( source: $env["source"] ?? "docker");
    return true;
}
```

Stockage des données de connexion à la base de données PostgreSQL sécurisé et renseigné dans le .gitignore

1.2.3. Ancien Stockage des mots de passe.

```
public function setMdpHache(string $mdpHache): void
{
    $this->mdpHache = $mdpHache;
}
!usage  🧑  Malo Gasquet
public function setMdp(string $mdp): void
{
    $this->mdp = $mdp;
}
```

Ancien Stockage des mots de passes en clair dans la Base de données permettant leur récupération facile

1.2.4. Nouveau Stockage des mots de passe

```
5 usages  👤 Malo Gasquet
public function getMdpHache(): string
{
    return $this->mdpHache;
}

1 usage  👤 Malo Gasquet
public function setMdpHache(string $mdpHache): void
{
    $this->mdpHache = $mdpHache;
}
```

Nouveau Stockage des mots de passes hachés uniquement dans la Base de données permettant leur récupération sécurisée

1.3. Gestion de Projet

1.3.1. Tableau de gestion des tâches

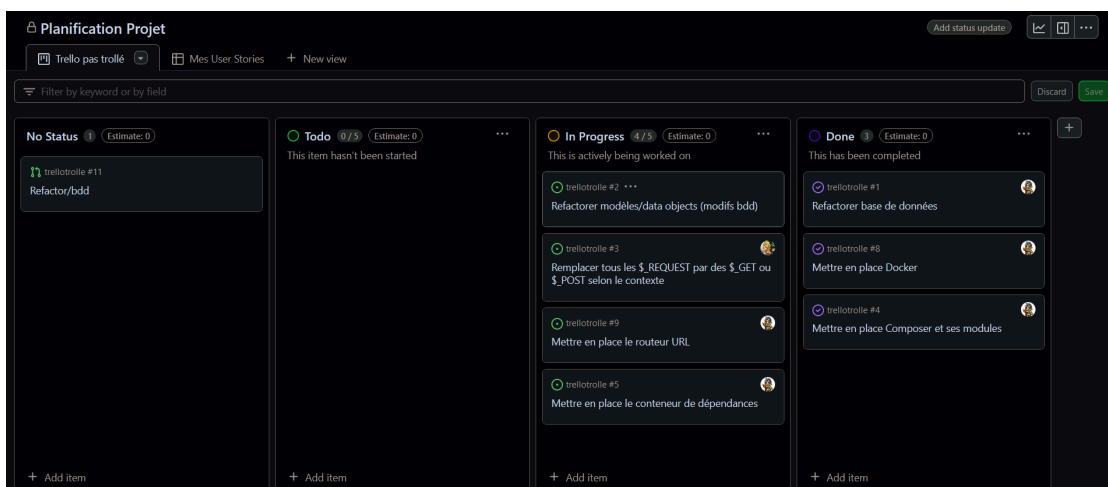


Tableau des différentes User Stories hébergé sur GitHub Projects en cours de projet

1.3.2. Tableau de gestion de tâches (fin de projet)

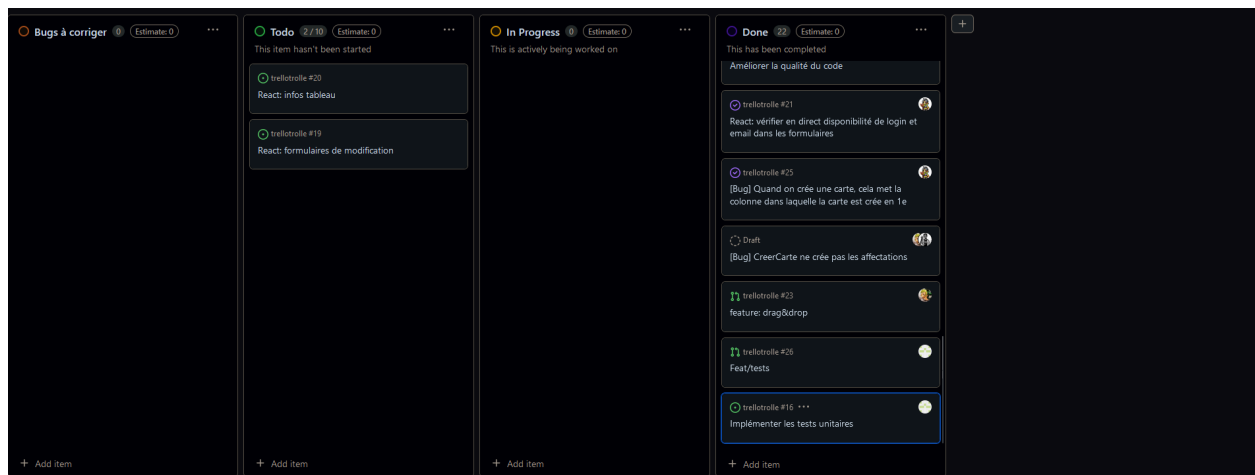


Tableau des différentes User Stories hébergé sur GitHub Projects au moment du rendu du projet

2. Glossaire

Trello : application web qui aspire à aider à la gestion de projet, en planches listant des cartes, chacune représentant des tâches.

Javascript : Langage de programmation utilisé principalement pour créer des fonctionnalités interactives sur les sites web.

3ème forme normale : Un principe de conception de bases de données relationnelles visant à réduire la redondance des données et à améliorer l'intégrité en exigeant que chaque attribut non-clé soit fonctionnellement dépendant de la clé primaire et de rien d'autre.

Casey-Delobel : Théorème stipulant qu'une base de données relationnelle peut être réorganisée en une forme normale supérieure sans perdre d'informations.

GitLab/GitHub : Des plateformes de gestion de code source basées sur Git, un système de contrôle de version distribué.

PHPDoc : Un outil de documentation automatique pour le langage de programmation PHP.

TODO : "À faire" en français, notation utilisée dans le code source pour marquer des parties qui nécessitent une modification future ou une implémentation ultérieure.

GitHub Projects : Application similaire à Trello, intégré directement à GitHub

DataObjects : Représente une classe ou une structure de données, utilisée pour interagir avec la base de données. Représente notamment la partie Modèle dans le modèle MVC.

Mot de passe haché : Une version cryptographiquement transformée du mot de passe de l'utilisateur, stockée dans une base de données pour en préserver la sécurité en cas de violation, rendant la récupération du mot de passe d'origine difficile.

.env : Fichier de configuration utilisé dans le développement de logiciels pour stocker des variables d'environnement sensibles, telles que les clés d'API ou les informations de connexion à la base de données.

.gitignore : Fichier utilisé pour spécifier intentionnellement les fichiers et répertoires qui ne doivent pas être suivis ou versionnés par Git.

pattern : Une expression régulière spécifiée dans la balise d'entrée HTML (<input>) pour définir un format spécifique attendu pour les données saisies par l'utilisateur.

HTML : Acronyme de *HyperText Markup Language*, est le langage de balisage standard utilisé pour créer et structurer le contenu des pages Web.

SAE : Acronyme de “*Situation d'Apprentissage et d'évaluation*”, est un projet effectué dans le cadre du diplôme du Bachelor universitaire de technologie, regroupant différentes ressources effectuées au cours de l'année.

reactive : librairie Javascript fournie par l'IUT permettant la création d'objets dynamiques dans une interface web réagissant aux différentes actions de l'utilisateur sur la page

3. Introduction

Trello-Trollé est un site clone du célèbre site web Trello*. Sur ce clone, nous avons donc la possibilité de s'inscrire et de se connecter, de gérer son compte, de gérer les tableaux et ses éléments : les colonnes, les cartes et les utilisateurs membres d'un tableau. Ces membres possèdent presque tous les droits dans la gestion d'un tableau (ajout, suppression, modification des colonnes et des cartes) sauf la gestion des autres membres du tableau et la suppression définitive du tableau, puisque seul le propriétaire du tableau possède ces droits.

Cependant, le site Trello-Trollé qui nous a été fourni est uniquement codé en PHP et ne comporte aucun code dynamique en JavaScript*, donc à chaque action, le site demande un nouveau chargement de page via une requête au serveur. De plus, le code du site comporte de nombreux problèmes, comme des problèmes de sécurité. Enfin, la base de données n'est pas organisée correctement et comporte ainsi de nombreuses redondances, ce qui peut causer de nombreuses erreurs de gestion des données.

L'objectif principal de ce projet était de, tout d'abord, réaliser un rapport (*audit*) des implémentations de fonctionnalités à revoir ou à refaire mais également les modifications à apporter à la base de données. Ces modifications visent à améliorer l'application, que ce soit au niveau de l'expérience utilisateur, de la qualité du code, mais également pour régler les éventuels problèmes de sécurité. Lors de la 2e partie du déroulement du projet, nous avons ensuite modifié le code, la base de données et éventuellement d'autres éléments comme l'apparence (*le design*) du site afin de remédier à tous les éléments que nous avons listé dans la première partie.

Dans la section d'analyse qui suit cette introduction, nous allons identifier les lacunes majeures du site Trello-Trollé, mettant en évidence les besoins d'améliorations et de sécurité. Par la suite, dans le rapport technique, nous allons effectuer un audit détaillé des fonctionnalités existantes, recensant celles à revoir ou à refaire, et proposer des solutions techniques pour y remédier. Enfin, la partie sur la méthodologie et l'organisation du projet décrit la gestion de l'équipe et les outils utilisés, tandis que le bilan critique, évaluant les étapes clés du projet, tire des leçons pour l'avenir. Nous pouvons également retrouver les annexes du rapport en fin de document, comportant entre autres un glossaire.

4. Analyse

Dans cette partie du rapport, nous allons tout d'abord énoncer les principaux problèmes qui ont été relevés lors de la phase d'analyse, en précisant ce qui ne va pas et la façon dont il est possible de les corriger.

4.1. Problèmes liés à la base de données

4.1.1. Normalisation

Dans l'état actuel de l'application, il y a un souci au niveau de la base de données. Il existe seulement une seule table `app_db` qui contient toutes les colonnes de l'application.

(cf. [Ancienne Base de Données](#))

Pour une application fonctionnelle et mieux structurée, il faudrait normaliser la base de données jusqu'à la 3ème forme normale* en séparant les données dans différentes relations. Pour cela, il est nécessaire d'utiliser le théorème de décomposition de Casey-Delobel*, qui va nous permettre de décomposer, sans perdre les dépendances fonctionnelles.

4.2. Failles de sécurité

4.2.1. Mot de passe des utilisateurs

Ensuite, ce code comporte plusieurs failles de sécurité. Par exemple, la présence de deux colonnes "mdp" et "mdpHaché" dans la base de données et dans le code est problématique. Cela signifie que les mots de passe sont stockés en clair, ce qui facilite la fuite des mots de passe des utilisateurs en cas d'attaque informatique.

(cf. [Ancien Stockage des mots de passe.](#))

Pour résoudre ce problème, il serait nécessaire de modifier la base de données pour supprimer la colonne "mdp" et conserver uniquement le mot de passe haché. De plus, il faudrait adapter le code pour que lors de l'inscription de l'utilisateur, son mot de passe soit directement haché sans être stocké en clair.

4.2.2. Stockage des informations de connexion

Et de même pour le fichier ConfigurationBaseDeDonnees.php qui contient les informations de connexion à la base de données PostgreSQL. Si quelqu'un a accès au code source du site d'une manière quelconque, il aura ainsi un accès total à la base de données de l'application.

(cf. [Ancien stockage des informations de connexion](#))

Il est préférable d'opter pour l'utilisation des variables d'environnement dans un fichier .env externe qui ne sera pas poussé sur le dépôt git, et/ou utiliser la fonctionnalité de "secrets" d'un dépôt sur GitHub/GitLab*. PHP possède les fonctions getenv() et parse_ini_file(\$nomFichier) pour extraire les informations d'un fichier de configuration.

4.3. Fonctions incompréhensibles et/ou non commentés

L'application possède également des soucis au niveau du code, et plus précisément du code incompréhensible dû à l'absence de commentaire. Il existe également des fonctions qui ont l'air d'avoir aucune fonctionnalité dans l'application.

```
public static function fun(string $n)
{
    if($n == 0) {
        Session::getInstance()->telemetry($n, b: $n-1, self::lire(cle: 'telemetry'));
    }
    for($i=0;$i<intval($n);$i++) {
        self::fun(n: Cookie::contient(cle: 'telem') ? $i-1 : (intval($n)+$i));
    }
}

public function telemetry($a, $b, $c)
{
    ConnexionUtilisateur::important($a, y: $b ? null : (($c+$a) > $a*$a ? $b : 24));
}
```

Dans les deux précédentes fonctions, il est possible de renommer d'abord les méthodes et même les paramètres, car ils n'ont pas de noms explicites. On ignore également l'utilité de ces méthodes, car il manque de commentaires explicatifs. Ces commentaires peuvent être ajoutés soit en utilisant PHPDoc* pour documenter la méthode, en

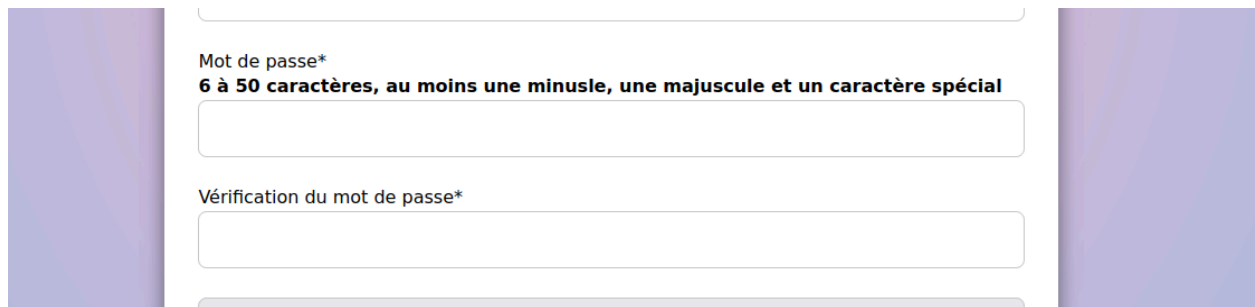
précisant les paramètres, l'objectif de la fonction et la valeur de retour, soit en commentant chaque ligne de code.

```
public static function important($x, $y)
{
    //Je crois que ça ne marche pas hahahaha
    //Je vais simplement retirer le code pour le moment
}
```

Ici, on peut voir la présence d'une méthode non implémentée pour l'instant, on peut préciser avec PHPDoc*, et même ajouter la mention TODO:* dans le commentaire: la plupart des IDE tiennent compte de cette mention et il est ainsi facile de rechercher les méthodes qui n'ont pas encore été implémentées.

4.4. Problèmes du site

Le site a des problèmes d'implémentation, notamment lors de la création d'un compte. L'utilisateur est invité à choisir un mot de passe comportant entre 6 et 50 caractères, avec au moins une minuscule, une majuscule et un caractère spécial.



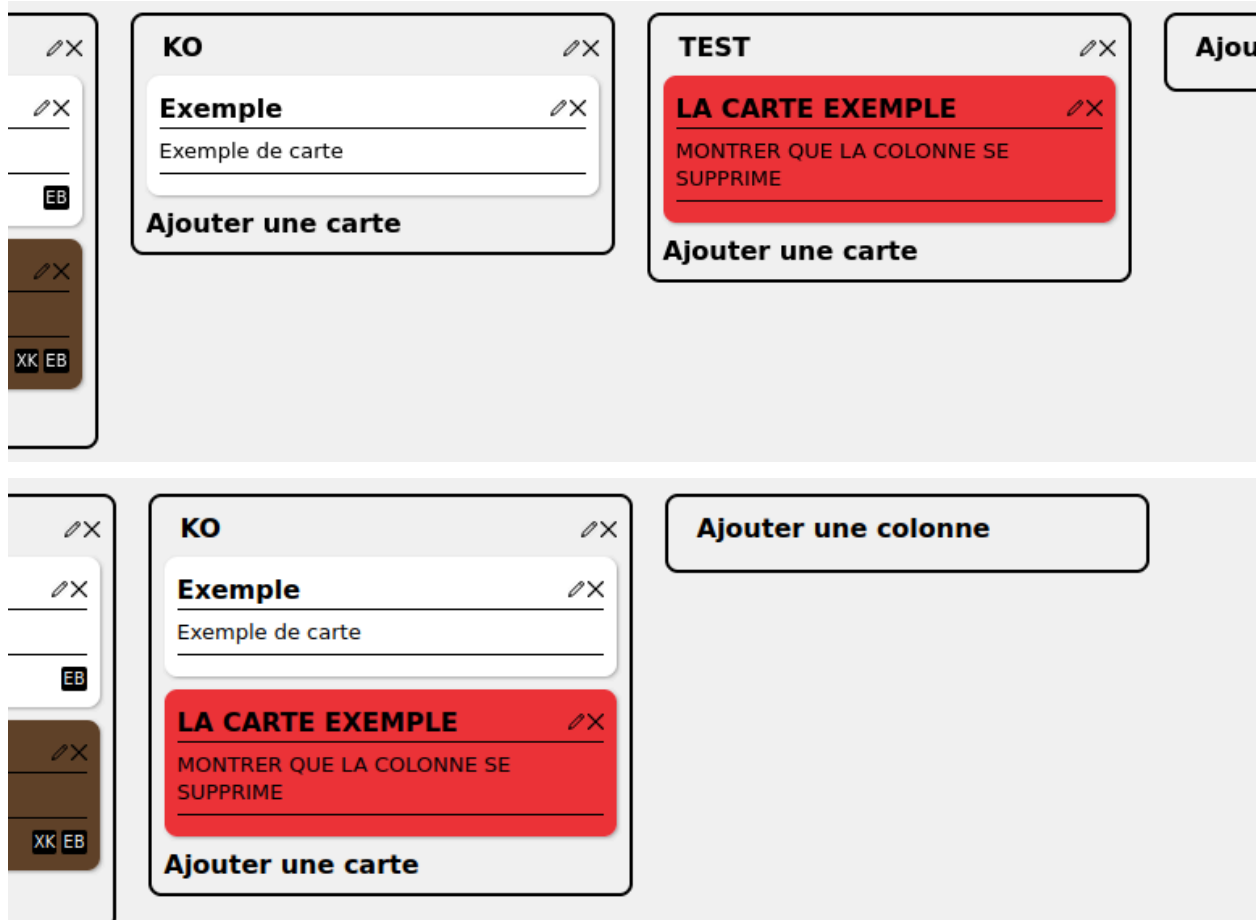
The screenshot shows a web form for creating a password. It consists of two input fields. The first field is labeled 'Mot de passe*' and has a validation message below it: '6 à 50 caractères, au moins une minuscule, une majuscule et un caractère spécial'. The second field is labeled 'Vérification du mot de passe*'. The form is flanked by two vertical purple bars.

Outre le fait que le mot "minuscule" est ici mal orthographié, la phrase de spécification ne prend pas en compte une caractéristique importante pour la validation du mot de passe.

```
pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&* _+=\ -]).{6,50}"
```

En fait, le modèle de champ de mot de passe spécifie la nécessité d'avoir au moins un chiffre dans le mot de passe. Cela n'est pas clairement expliqué, ce qui peut empêcher l'utilisateur de s'inscrire sur le site web Trello-Trollé.

Enfin, le site possède des fonctionnalités ne fonctionnant pas complètement, comme ici où si l'on supprime la dernière carte d'une colonne, alors cette colonne est supprimée.



Il serait nécessaire de modifier la fonction qui gère la fonctionnalité pour assurer son fonctionnement complet.

5. Rapport technique

5.1. Normalisation

Nous avons tout d'abord changé la modélisation de la base de données, en effectuant une décomposition avec le théorème de Casey-Delobel*, qui nous a permis d'obtenir une base de données 3ème Forme Normale*.

(cf. [Nouvelle Base de Données](#))

Cette décomposition nous a permis dans un premier temps, d'enlever les redondances et donc les incohérences que pouvait avoir l'ancienne décomposition. Elle nous permet également d'avoir une base de données qui est plus facilement gérable et compréhensible.

5.2. Faille de sécurité

Nous avons donc pu supprimer la colonne affichant le mot de passe en clair des utilisateurs, et on a pu refactorer le code, plus principalement les DataObjects* pour les modifier et s'adapter à la nouvelle modélisation.

(cf. [Nouveau Stockage des mots de passe](#))

On a conservé ici que les mots de passes hachés* afin d'éviter une fuite de mots de passes en cas d'attaque informatique sur la base de données.

5.2.1. Stockage des informations de connexion

On peut désormais se connecter à la base de données à partir d'un fichier .env* contenant toutes les informations de connexions sous variable d'environnement

(cf. [Nouveau stockage des données de connexion à la BDD](#))

Ceci est beaucoup plus sécurisé car vu que le fichier `.env*` est présent sur le `.gitignore*`, cela signifie qu'il ne sera pas envoyé sur GitHub* et donc les informations de connexions ne seront pas affichées publiquement.

5.2.2. Fonctions incompréhensibles et/ou non commentés

Les fonctions citées précédemment n'ont pas été conservées ou utilisées dans le code source de l'application. Nous avons également commenté des fonctions avec PHPDoc* pour les rendre plus compréhensible pour les développeurs :

```
/**
 * Permet de savoir si un utilisateur a des droits d'accès sur un tableau
 * @param int $idTableau
 * @param string $login
 * @return bool
 * @throws ServiceException
 */
16 usages  ⓘ Ilan Velter
public function aDroitsDaccès(int $idTableau, string $login): bool
{
    return $this->estProprietaire($idTableau, $login) || $this->estMembre($idTableau, $login);
}
```

Les fonctions du projet possèdent également des noms explicites qui permettent de savoir à peu près leurs utilités à la première lecture.

5.2.3. Problèmes du site

Pour ce qui est du mot de passe, on a changé les indications concernant les caractères minimums, et a ajouté "et un chiffre" pour suivre le `pattern*` du code HTML* (Nous avons également corrigé la petite erreur sur "minuscule" qui était écrite : "minusle")

Mot de passe*

6 à 50 caractères, au moins une minuscule, une majuscule, un caractère spécial et un chiffre

Vérification du mot de passe*

```
pattern="(?!.*\d)(?!.*[a-z])(?!.*[A-Z])(?!.*[!@#$%^&*_=+\- ]).{6,50}"
```

On a corrigé le bogue concernant les colonnes qui s'effaçaient automatiquement si on supprimait la dernière carte

<div>En cours</div> <div><div>Exemple</div><div>Exemple de carte</div></div> <div>Ajouter une carte</div>	<div>Fini</div> <div><div>Exemple</div><div>MONTRER QUE LA COLONNE NE SE SUPPRIME PLUS</div></div> <div>Ajouter une carte</div>
<div>En cours</div> <div><div>Exemple</div><div>Exemple de carte</div></div> <div>Ajouter une carte</div>	<div>Fini</div> <div>Ajouter une carte</div>

6. Méthodologie et Organisation du Projet

6.1. Gestion d'équipe

Afin de nous organiser sur cette SAE, nous avons décidé d'utiliser principalement GitHub*, d'abord pour gérer le versionning de code, mais également l'outil de gestion de projet GitHub Projects* similaire à Trello. Son avantage principal est qu'il est intégré à GitHub*, il est donc facile de lier les issues avec les différentes cartes du tableau.

(cf. [Tableau de gestion des tâches](#))

6.2. Méthode de développement et outils

Quelle méthode de gestion de projet avez-vous choisie et pourquoi ? À quels ajustements avez-vous procédé ?

Montrez la planification réalisée au début du projet et le déroulement réel du projet pour faire une analyse comparative. Temps perdu ? Différence par rapport à ce qui était planifié au début ?

(cf. [Tableau de gestion de tâches \(fin de projet\)](#))

Tout d'abord, on a décidé d'effectuer cette SAE* en utilisant les méthodes agiles que nous avons pu apprendre le semestre précédent, car on a pu s'apercevoir que c'était beaucoup plus pratique que travailler en cascade.

Cependant, comme expliqué précédemment, nous avons utilisé GitHub Projects* au lieu de Trello* que nous avons pu utiliser lors de la dernière SAE*. Nous n'avons également pas conservé les points de complexité attribués aux User Stories. On a préféré discuter entre nous pour définir les User Stories prioritaires par rapport aux autres. Enfin, au vu du temps limité pour compléter la SAE*, nous avons décidé de ne pas diviser cette période en différents sprints. Finalement, il nous a manqué de temps pour le reactive*.

6.3. Bilan critique

Nous sommes satisfaits de la façon dont nous avons géré le projet ce semestre, même si nous avons dû composer avec des délais assez courts. En ce qui concerne les points faibles, il pourrait être utile de réintroduire les points de complexité dans les User Stories, afin d'avoir toujours une vision claire des priorités du projet.

7. Conclusion

Dans le cadre de ce projet, nous avons travaillé sur un faux Trello* en utilisant principalement du PHP et JavaScript. Les objectifs principaux étaient de refactoriser le code existant et d'ajouter des fonctionnalités JavaScript*.

Nous avons réussi à atteindre ces objectifs en améliorant la structure du code PHP* et en implémentant avec succès les fonctionnalités JavaScript* demandées.

Il pourrait être intéressant dans le futur de changer la charte graphique du site, ou même de pouvoir avoir une synchronisation en temps réel du site, tel que le site Trello. On pourrait également rendre le projet responsive afin qu'elle soit disponible pour les appareils mobiles.

Les problèmes techniques principaux que nous avons rencontrés étaient liés à la modification du code déjà existant. Nous devons maintenir les anciennes fonctionnalités tout en ajoutant de nouvelles.

Grâce à GitHub, nous avons pu nous organiser et communiquer plus efficacement sur les tâches à accomplir.

Enfin, sur le plan de la méthode de travail, nous avons adopté une approche agile, en travaillant étape par étape pour assurer une progression régulière du projet. Nous avons également accordé une attention particulière à la mise en place de bonnes pratiques de développement pour garantir la maintenabilité du projet à long terme.

8. Bibliographie

8.1. Bibliothèques de code utilisées

[1]

« phpmailer/phpmailer - Packagist ». [En ligne]. Disponible sur:
<https://packagist.org/packages/phpmailer/phpmailer>

- Documentation en ligne du module PHPMailer disponible sur le site Packagist répertoriant un grand nombre de modules de Composer.

« Symfony Documentation ». [En ligne]. Disponible sur:
<https://symfony.com/doc/current/index.html>.

- Documentation en ligne de Symfony disponible directement sur leur site

« phpunit/phpunit - Packagist ». [En ligne]. Disponible sur:
<https://packagist.org/packages/phpunit/phpunit>.

- Documentation en ligne du module PHPUnit disponible sur le site Packagist.

« twig/twig - Packagist ». [En ligne]. Disponible sur:
<https://packagist.org/packages/twig/twig>.

- Documentation en ligne du module twig disponible sur le site Packagist.

« PSR-4: Autoloader - PHP-FIG ». [En ligne]. Disponible sur:
<https://www.php-fig.org/psr/psr-4/>.

- Module PSR4 Autoloader disponible sur leur site.