Compte Rendu Projet Post It

ghouibi ghassen

January 7, 2019

Contents

1 Introduction

Le but de projet est de disséminer des post-it virtuels sur un repère infinie,Ces derniers vont suivre les mouvements du curseur à l'aide de la souris pour être chargé par le navigateur.
Pour eviter les conflits on introduit une base de données, d'où la nécessité d'avoir un compte pour chaque utilisateur.

1.1 Architecture du code

Le projet s'adapte à l'architecture MVC (Model,View,Controller).
Model: comporte l'access à la base de données donc contient les données à afficher pour l'utilisateur.
View : comporte les différentes vu on trouve l'interface pour se connecter ou s'inscrire églament on trouve une page d'acceuil et une page pour la modification de mot de passe et un lien de mot de passe oublié .
Controller : comporte les fichiers view controller et post-it controller qui va suivre les actions de l'utilisateur autrement dit qui vont fourni un résultat en fonction de l'action effectuées par l'utilisateur.

2 Model

2.1 utilisateur

L'utilisation des modules mysql et bcrypt-nodejs,Pour cette partie j'ai utilisé Xampp, tout simplement pour la facilité d'utilisation et surtout la syntax de MariaDB trés similaire à SQL.
L'utilisation bcrypt-nodejs pour crypter le mot de passe dans la base données grâce à ce module si jamais la base de données est perdu c'est difficile de déchiffrer les mots de passes donc les comptes utilisateurs sont protégé.

ajouterUnUtilisateurDansLaBaseDeDonnes

Cette fonction va nous permettre d'ajouter un utilisateur dans la base données donc chaque utilisateur aura un nom et un prénom un email et un mot de passe un idenfiant qui servira a identifier l'utilisateur dans les prochaines rêquêtes et accès à la base de données et aussi qui sera utile dans l'utilisation des token vu qu'on stockera l'adresse email dans le token.

mettreAjourLecompteUtilisateur

Cette fonction permet de mettre à jour le mot de passe de l'utilisateur quand une demande de réinitialisation sera demandé par l'utilisateur à traves un lien qui va être envoyer par email.

ajouterAlaBaseDeDonnees

Cette fonction permettra d'ajouter un post-it à la base de données donc le post-it sera présent dans la base de données sous la forme suivant (idUtilisateur,idPostit,coordonnéesX,coordonnéesY,distance,angle,text,couleur).

L'id de l'utilisateur comme détaillé auparavant pour éviter les conflits entre les utilisateurs alors que idPostit évitera aussi les mêmes erreurs lors de la suppression par exemple .Au début avant d'utiliser les tokens et les routes sécurisés je me suis retrouvé à effacer des post-it d'autres utilisateurs donc les id sont trés important.

chercherLutilisateurDansLabaseDeDonnees

Cette fonction permettra de voir si l'utilisateur existe ou pas dans la base de données et comparer les mots de passes cette fonction et surtout utiliser dans le cas de connection ,et qui va renvoyer un résultat soit 0 soit 1.

chercherLidDeLutilisateur

Cette fonction permettra de renvoyer l'id de l'utilisateur et elle sera fortement solliciter à chaque identification pour l'accès à la base de données.

SupprimerUnPostIt

Cette fonction permet de supprimer un post-it à l'aide de l'identifiant de l'utilisateur et du post-it,Et comme expliquer auparavant c'est pour ça l'ajout de identifiant de post-it est important.Imaginons si un utilisateur bouge son post-it et veut bien le supprimer ? dans ce cas la l'utilisateur va supprimer un post-it qui existe pas déjà vu que avant je supprimer avec les paramètres de post-it mais dans ce cas la on peut supprimer le post-it même si ce dernier change de position vu que son identifiant est unique.

chargerLesPostitDeLaBaseDeDonnees

Cette fonction permet de charger les Post-it à partir de la base de données donc logiquement dans la plupart de cas on aura plus qu'un post-it donc on fera un tableau d'objet qui sera envoyer au serveur à la fin de chargement de tous les post-its dans le tableau.

mettreAjourText

Cette fonction permettra de faire la mise à jour du texte présent dans le post-it ,comme prévu on utilise l'id de l'utilisateur et de post-it (pareil de la suppression cela permet de modifier le post-it quand il sera dans une position différente à ce qu'il lui atribuée de début).

nombredePostit

Cette fonction va extraire le nombre maximum de post-it par l'utilisateur,cette fonction est importante pour agrandir le repère infinie de l'utilisateur.

3 View

L'utilisation du format ejs ,Bootstrap et recaptcha .
Le format Embedded JavaScript était d'une grande utilité pour l'envoi de message rapide et facilement dans les fichiers password et inscription, par contre je voulais profiter un peu plus de cet outil au début avant de faire les rêquetes client avec xhr je fesais comme ceci

```
res.render('index',{token:myToken}) <%= token %>
```

```
var test=JSON.parse( '{ "token" : "<%- token %>"}');
```

Par contre cette méthode n'est fortement déconseiller vu qu'on pourra accéder au token vu qu'il sera dans une balise,Mais au début ça me permet d'avancer.
L'utilisation de bootstrap c'était dans le but de simplifier le travail en css et aussi pour la raison que niveau design et couleur je m'y connais pas trop.
recaptcha n'était pas d'une grande utilité dans ce projet je voulais juste voir comment ca se présente et essaye une fonction basique dessus.
Au début la décomposition était views/(les vues) ,partials/(footer.ejs,header.ejs).
Aprés j'ai abonnder cette décomposition vu que j'ai pas plusieurs vues.

3.1 Connexion

Le fichier Connexion.ejs comporte deux champs un pour la connexion qui va être transmis via une rêquete dans le fichier 'screen-controller' et l'autre champ sera pour les tentatives successive d'un utilisateur à se connecter avec un mot de passe qui n'est pas apporiré par exemple .
Donc on va jouer sur l'apparition d'une partie à la fois , ce choix parceque oublié le mot de passe et se connecter à un compte ce sont deux actions liées autrement dit dépend l'une de l'autre.
Dans le cas ou l'utilisateur à bien saisie les informations l'utilisateur recoit un token qui sera stocker dans le LocalStorage signé par le serveur et aussi un element qui s'appelle clicked dans le SessionStorage on verra à quoi il servira aprés.

3.2 Home

Ce fichier comporte rien de spécial à part l'inclusion des fichiers javascript screencontroller,et post-it controller et tools.

3.3 Index

Ce fichier comporte une photo d'acceuil et la barre de navigation. Une petite photo pour faire jolie c'est tout !,Ce fichier servira justement comme redirection des rêquetes du genre mot de passe oublié ou déconnexion.

3.4 Inscription

Ce fichier comporte un formulaire d'inscription et la rêquete d'inscription se fera dans la balise form .
Ce formulaire répond à l'utilisateur en fonction des données saisie dans le cas ou l'utilisateur à un compte un message d'erreur va apparaître le compte existe déjà,Sinon l'utilisateur reçoit un email de confirmation.

3.5 Password

Ce fichier comporte la vue de modification de mot de passe il y a un formulaire qui demande l'email et le nouveau mot de passe deux fois si jamais les mots de passe ne sont pas identiques l'utilisateur reçoit un message que les mots de passes ne sont pas identiques et si l'utilisateur à bien saisie les informations et n'a pas cocher le recaptcha il reçoit un message qu'il faut cocher cette case. Dans ces deux cas le mot de passe ne sera pas changer et dans le cas ou tout se passe comme prévu le changement de mot de mot sera effectué et la modification dans la base de données sera faite et une redirection vers la page de connexion sera effectué.

4 Controller

4.1 screencontroller

Ce fichier comporte une classe Vue qui permet de manipuler les vues dans le cas ou l'utilisateur a un token donc l'utilisateur n'a plus besoin de se connecter ou d'accèder au page d'index ou autre donc il aura une redirection vers la page home peut importe la rêquete effectué de la part de l'utilisateur. Aussi dans fichier on a la rêquete de connexion qui permettra de controller la vue connexion si jamais l'utilisateur remplit le formulaire 3 fois de suite avec des erreurs soit dans le mot de passe ou dans l'email on aura un changement de vue entre le formulaire de connexion et un autre formulaire permet d'envoyer un email à l'utilisateur à traves la balise form.
Dans le cas ou l'utilisateur clique sur mot de passe oublié et renseigne son email il recoit un email avec un lien pour modifier son mot de passe qui sera la page password.

4.2 post-itcontroller

La grande partie du projet se situe ici,l'utilisateur se connecte et trouvera une interface comme-ceci





Tout d'abord la recuperation de l'hitorique de post-it,cela s'effectue à traves une rêquete donc grâce à la fonction recupererPostItDuServeur,on effectue cette rêquete en ajoutant au header le token pour permettre au serveur d'identifier la personne ,la réponse sera un objet JSON donc on parse cette objet et on boucle aprés on créer le post-it grâce à elementFactory et à la fin on ajoute le post-it au tableau de post it.
Aussi La fonction boucleDeselectionDePostit qui se présente pour détecter des evenements fréquents et surtout le dragstart, dragend et click pourquoi tout simplement parceque on connait pas le nombre de post-it qui sont présent et à n'importe qu'elle moment on peut avoir un nouveau element donc le choix est orienté vers querySelectorAll.
Le button jaune permet à l'utilisateur de se déconnecter donc le token sera supprimer du localStorage.
Le button rouge c'est une sorte de corbeille qu'on peut supprimer des éléments en faisant un drag un drop vers la corbeille.
Le button vert permet d'ajouter un nouveau post-it.
Le button bleupermet de mettre les post-it en question et quand on clique sur se button le point noir au milieu servira d'une sorte d'indicateur de souris qui suivra le mouvement de la souris.

4.2.1 Ajouter

La partie qui permettera l'ajout de post-it à besoin d'un contenu de post-it donc un text saisie de la part de l'utilisateur aprés on construit l'objet le choix était sur des objets tout simplement pour faciliter la manipulation des données logiquement on aura plus qu'un post-it donc le tout est stocker dans un tableau les coordonnées de post-it sont choisie au hasard entre 0 et la taille de l'écran.S'il y a pas de collision le post-it est fixer à c'est coordonnées la avec une couleur random par contre il manque des paramètres c'est l'angle

et la distance ,dans cette partie la j'ai essayé au début des calcules a partir des coordonnées absolue de post-it ou des coordonnées par rapport à la page mais sans qu'ils aboutissent vraiment donc j'ai opté pour le produit scalaire mais c'était un peu dificille et il avait beaucoup de conflits au final j'ai opté pour la méthode de transformer l'ecran en repère orthonomée c'est à dire si la souris est dans la position (0,0) elle sera donc dans la position (-innerWith,innerHeight) et pareil pour toute autre coordonnées donc l'origine sera (innerWith/2,innerHeight/2). comme ça on peut calculer l'angle entre ces deux segments l'aide de arctg.aprés la création et le calcul de la distance et de l'angle on peut maintenant créer l'element grace a elementFactory,on fera un push dans le tableau de post-it et on envoie ces données au serveur,Au Début c'était au moment de déconnexion ou on fait la rêquete POST vers le serveur aprés j'ai changer d'avis . Au final Le button ajouter se construit grace à la classe ButtonAjouter,la séparation des classes était volontaires pour mieux se repèrer au cas de problème.

4.2.2 Supprimer

Le deuxième point sera le button de supprimer ,ce dernier doit ecouter les evenements suivant click,dragover,dragenter,dragleave et drop.J'ai choisi de faire un drag and drop pour facilité la modification du texte en click donc pour modifier le text d'un post-it il suffit de cliquer sur le post-it concerner et une boîte de dialog apparaît pour écrire le nouveau text et on envoie ces modifications au serveur bien sur avec le token en header pour être bien identifier par le serveur et le serveur se chargera de faire la rêquete SQL de modification du text. Pour la suppression de post-it lors de création de post-it on utilise l'element draggable a true qui va nous permettre de bouger notre post-it je me suis basé sur une video sur youtube qui explique comment faire un drag and drop et je me suis inpisré donc la suppression fait appel à la class DragAndDrop, qui aura dragStart,dragEnd,dragOver,dragEnter,dragLeave,dragDrop,rappel et clicked qui permmettra la modification du text expliquer juste avant, donc ces fonctions s'appuye beaucoup sur le css le dragStart il faut cliquer et rester appuye sur le button de la souris dans ce cas on récupère l'id de l'element séléctionner , on l'attribut à la class hold , et on le rend invisible gràace à la classe invisible, aprés on se dirige vers la corbeille pour la suppression dans se cas on attribut l'element à la classe hovered aprés on aura le dragover c'est à dire on est au bon endroit pour faire un drop dans ce cas on va se servir encore de sessionStorage pour faire une alert à l'utilisateur pour afficher un message que le post-it sera supprime s'il fait un drop dans le cas l'utilisateur peut faire un drop ailleur et donc le post-it ne sera pas supprimé et reprend sa position initial, et il reprend sa class habituelle fill,Dans le cas de drop on va récuperer tous d'abord l'id de l'element et on l'envoie au serveur qui est de son tour fait la rêquete SQL pour la supression donc ce cas comme expliquer

auparavant dans le fichier utilisateur on aura besoin de l'id de l'utilisateur et celui de post-it comme ça même si le post-it change de position on peut le supprimer grace à ça,l'id de utilisateur sera accessible à partir du token qui sera envoyer dans le header c'est si le token est modifier par exemple la suppression ne sera pas effectuer . on enchaîne avec la suppression du body et du tableau de post-it et aussi la suppression de alerted de sessionStorage.

4.2.3 Play

Le button play il va écouter que les clicks il sera créer grace à la class ButtonPlay en s'appuyant sur la fonction ButtonFactory pareil que les autres buttons donc l'evenement déclenchera la fonction play qui sera composé sur 2 états oui ou non une autre fois on utilise le sessionStorage et était initialiser avec oui au début de connexion dans le screen controller dans notre cas clicked sera initialiser avec oui donc on va écouter l'evenement mousemove c'est à dire tant que la souris en mouvement et on fera un alert de nombre de post-it présent vu que certain post-it ne sont pas dans le repère de l'utilisateur . Dans le deuxième cas on remet la petite bulle qui suit la souris à l'origine autrement dit au mileu de l'ecran. le premier cas fera appel à la fonction target cette fonction récupère les coordonnées de l'element qui suit la souris et lui attribut les coordonnées de la souris alors on va tracer les coordonnées de la souris de la même manière pour les post-its conversion des coordonnées et calculer l'angle,vu qu'on a l'ecran partagé sur 4 partie donc l'angle de la souris évolue de 0 jusqu'a 90 degrée et si l'angle de la souris et l'angle de post-it sont égaux et les positions sont aussi égaux tout simplement parceque on peut avoir le même angle mais pas dans la même partie du repère orthonommé (imaginaire) dans ce cas on fera un deplacement donc la fonction deplacement interviens ça dépend des cas si l'element est à gauche on ferra un + pour la position s'il est à droite on diminue comme expliquer auparavant . e

4.2.4 Deconnexion

Dans le cas ou l'utilisateur se deconnecte tous simplement on fera une redirection vers la page d'index et on supprime toutes les informations qui lui concerne comme le token et l'element clicked dans le sessionStorage

5 Serveur

6 Conclusion