Cahier des charges - Place

1 But de la prestation

Vous devez réaliser un site Web dynamique mimiquant le fonctionnement de l'application *Place* du site Reddit lorsqu'elle était en fonctionnement.
Voir : `https://www.reddit.com/r/place`

L'utilisateur pourra s'identifier sur la page d'accueil (ou créer un compte) et voir la grille de dessin (canevas) s'afficher sur une page. Ensuite, l'utilisateur pourra placer un pixel dans la zone de son choix avec une couleur choisie parmi une liste prédéfinie.

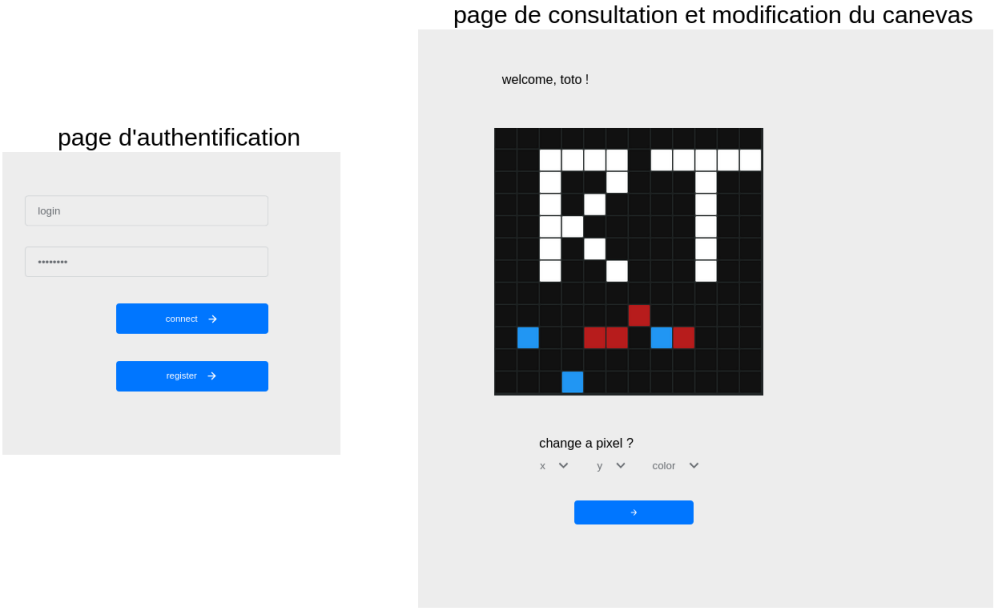


FIGURE 1 – prototype de l'application

Les technologies à utiliser sont celles vues en cours lors des différentes Ressources de l'année : HTML, CSS, SQL, PHP et Javascript. L'accent sera mis sur les fonctionnalités possibles sur le site plutôt que sur l'aspect graphique, il s'agit ici de mettre en place une maquette pour démontrer que l'application est faisable (on parle de *Proof of Concept*)

Vous devrez rendre sur Ecampus pour le **Vendredi 10/06 à 17h** par binôme :
— l'intégralité de votre code source et des fichiers nécessaires au fonctionnement de l'application (scripts, fichiers sqlite, images, CSS, etc)
— un rapport faisant office de documentation et d'explication du code. Il doit présenter comment initialiser le projet, comment lancer le serveur et interagir en tant qu'utilisateur. Il présente aussi les différentes méthodes et comment le projet fonctionne dans son ensemble.

2 Environnement de travail

Votre projet devra être développé dans les mêmes conditions de travail que celles disponibles dans les salles de TP informatique du rez-de-chaussée en RT. Ce sont sous ces conditions que l'enseignant évaluera votre travail :

- Ubuntu 18.04 LTS (mais une autre distribution GNU/Linux peut faire l'affaire si les bons paquets sont installés)
- PHP version 7.2.24 (une version plus récente peut faire l'affaire)
- paquets `sqlite3` et `sqlitebrowser`
- navigateur Firefox pour test des fonctionnalités (version 100.0.2)

L'application sera composée de plusieurs pages en HTML/CSS et PHP/Javascript, et les données à enregistrer seront stockées dans une base *sqlite*. L'application pourra être lancée avec un serveur PHP à l'écoute sur le port 50000.

Il peut être intéressant pour votre binôme de mettre votre code sur un logiciel de synchronisation des données style GitHub ou un répertoire partagé sur Unicloud. Pensez à faire des sauvegardes régulières !

3 Fonctionnalités

3.1 Éléments de base, triés par ordre de priorité

- Une fonction en PHP afin d'initialiser le projet : elle crée à minima la base de données avec deux tables *Utilisateur* et *Pixel*, crée un utilisateur "admin" et remplit la table Pixel avec une grille par défaut blanche de 16*16 pixels.

- Page d'authentification avec identifiant et mot de passe : l'utilisateur pourra se connecter ou créer un compte basique s'il n'en a pas. Cette page vérifie les informations dans la table *Utilisateur* et agit en conséquence. Si tout est ok, on redirige vers la page du canevas.

- Page d'affichage du canevas : va chercher la liste des pixels définis dans la table *Pixel* et les affiche sous forme de grille avec une balise `svg`. Un bouton de déconnexion sera disponible.

- Sur la même page, possibilité de modifier un pixel en tant qu'utilisateur en donnant dans un formulaire des coordonnées (x,y) et une couleur souhaitée (parmi une liste prédéfinie). On va ensuite modifier dans la table *Pixel* l'élément concerné et recharger le canevas.

- Stocker la date de la dernière modification afin d'empêcher une modification si la dernière modification date de moins de 1 minute.

3.2 Volet créativité

Une fois toutes les fonctionnalités de base implémentées, libre à vous de réaliser des éléments supplémentaires selon vos envies. L'idée ici est de faire preuve de créativité et de se faire plaisir en ajoutant une plus-value personnalisée à votre projet ! Je vous donne quelques idées possibles à la volée pour vous inspirer, mais libre à vous de faire autre chose...

- ajouter une page d'administration qui permet de "tricher" en modifiant n'importe quel pixel sans limite de temps
- modifier le nombre de couleurs disponibles, par exemple avec un *widget* permettant de choisir une couleur en hexadécimal
- ajouter un compteur décroissant indiquant le nombre de secondes restantes avant de pouvoir modifier le canevas, et griser le bouton "envoyer" tant que la condition temporelle n'est pas respectée
- proposer la sélection d'un pixel en cliquant dessus plutôt qu'en rentrant ses coordonnées dans le formulaire
- ajouter une page d'administration qui permet de modifier la taille du canevas (sans casser la disposition précédente)
- stocker les mots de passe des utilisateurs de manière un peu plus sécurisée, par exemple avec un hachage `MD5`
- vérifier qu'on ne puisse pas créer de nom d'utilisateur avec espace ou un nom déjà existant dans la base de données
- afficher au survol d'un pixel le nom de l'utilisateur qui a fait la dernière modification
- faire une page de statistiques sur les utilisateurs les plus actifs

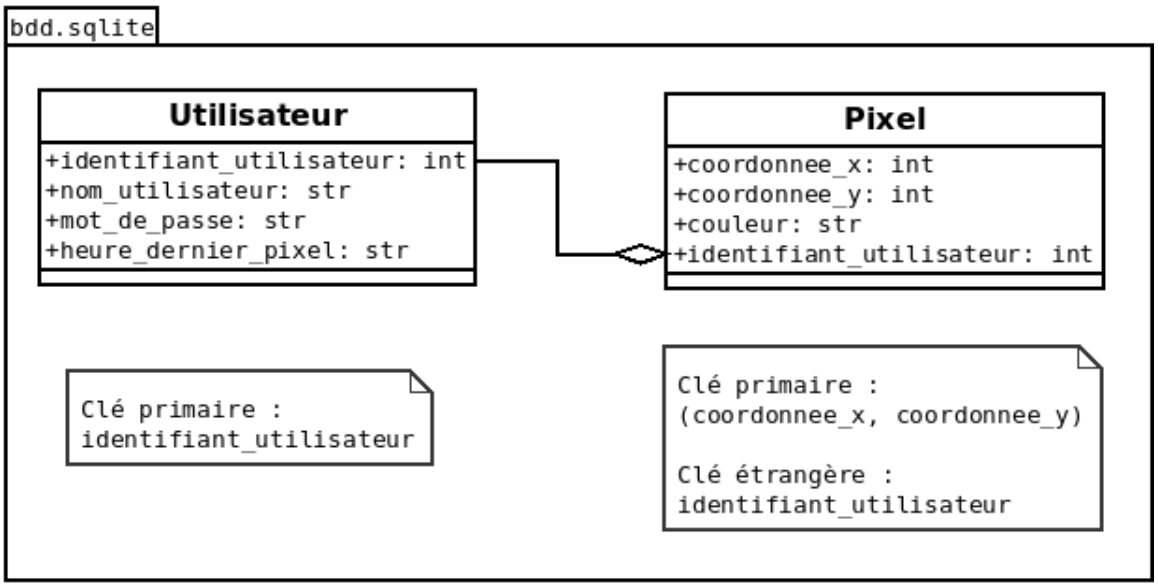
4 Conseils techniques

Vous êtes livrés à vous-même pour la réalisation du projet, mais je vous donne dans cette section quelques préconisations et vous indique comment procéder pour ne pas partir dans tous les sens.

De manière générale, avant de vous lancer dans le développement il peut être intéressant de mettre sur papier le fonctionnement générale de votre application : pages requises, méthodes requises, interactions et dépendances, ordre de priorité des éléments, etc.

4.1 Stockage des données

Il y a deux éléments à stocker, les utilisateurs et les pixels du canevas. On se propose donc une simple base de données créée avec *sqlite* avec le schéma relationnel suivant :



Pour la table *Utilisateur* rien de bien complexe : l'attribut "heure_dernier_pixel" contiendra la date (sous forme de chaîne de caractères) à laquelle l'utilisateur a placé un pixel pour la dernière fois afin d'implémenter si besoin le contrôle temporel.

La table *Pixel* contient chaque pixel individuel avec ses coordonnées en abscisse et ordonnée, la couleur qu'il contient et l'identifiant de l'utilisateur qui a fait la dernière modification (par défaut ce sera le 1 pour l'administrateur)

Pour créer la base, soit vous pouvez travailler en faisant des requêtes, soit vous pouvez utiliser l'outil graphique "DB Browser for SQLite" qui est très intuitif et permet de faire des requêtes et vérifier l'état de la base à n'importe quel moment.
NB : attention si vous ouvrez votre fichier *sqlite* avec l'outil graphique il est bloqué en lecture/écriture, à ne pas faire en même temps que les tests de l'application !

4.2 Affichage du canevas

Vu qu'on dispose de coordonnées pour chacun des pixels, il existe plusieurs manières de faire l'affichage d'une grille de pixels en HTML. Je vous préconise d'utiliser l'élément `svg` qui est tout désigné pour faire de l'affichage de dessins vectoriels.
`https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial`

Il existe de nombreuses documentations en ligne à condition de chercher les bons mots-clés, mais je ne vais pas vous donner la solution directe. L'idée de base est d'avoir un élément parent `svg` avec la propriété `viewBox` de la taille voulue. Cet élément contiendra autant de balises enfant `rect` qu'il y a de pixels sur le canevas. Chacun des rectangles contiendra ses coordonnées et fera une taille de 1*1. Attention à la manière dont les axes sont affichés, c'est inversé par rapport aux normes de géométrie : `https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial/Positions`

Je vous conseille de faire dans un premier temps une page de test en HTML pur (ou une petite boucle PHP) pour réaliser l'affichage d'un canevas statique. Puis une fois que

vous avez compris comment faire l'affichage, vous pourrez passer à l'affichage dynamique en fonction de ce qui est contenu dans la base de données.

4.3 Interaction côté serveur

Les interactions côté serveur correspondent à tout ce qui fait appel à la base de données pour vérifier des informations de connexion, pour afficher les différents pixels ou pour modifier un pixel. Pour cela, le langage *PHP* est adapté et ce sont des éléments sur lesquels vous avez déjà travaillé.

Pour interagir avec la base de données, on utilisera les objets `PDO`. Il faut récupérer les informations utilisateur dans les formulaires avec le dictionnaire `$_POST` (ou `$_GET`) et les passer en paramètre avec un `bindValue`. L'état d'authentification de l'utilisateur sera gérée avec la notion de session. Tous ces éléments ont été travaillés lors des TP3 et 4 de la ressource 209, n'hésitez pas à reprendre vos travaux.

4.4 Interaction côté client

Si vous souhaitez implémenter des fonctionnalités supplémentaires (volet créativité) vous pourrez utiliser le langage *Javascript* afin de proposer un niveau d'interactivité plus confortable pour l'utilisateur. On pensera à utiliser des `EventListener` afin de capter les mouvements et clics de l'utilisateur.

5 Notation

Le barème n'est pas fixé car votre enseignant a une (légère) tendance à sous-estimer le temps de travail à investir pour un projet de développement. La notation sera donc faite de manière comparative par rapport à la production de l'ensemble de la promotion initiale selon les éléments réalisés. A titre informatif, il est prévu de noter :

— les fonctionnalités de base sur 12
— la documentation technique sur 3
— le volet créativité sur 5

Liste non exhaustive des items de base qui seront comparés d'un binôme à l'autre :

— application sans erreur, se lance bien en écoute sur le port 50000
— base sqlite fonctionnelle
— fonction d'initialisation de la base et du projet
— page de login fonctionnelle
— possibilité de créer un compte
— démarrage d'une session à la connexion
— possibilité de se déconnecter
— affichage d'un canevas "statique"
— affichage du canevas selon les données de la base
— interface de modification du canevas