SAÉ 203

Introduction

L'objectif de cette SAÉ est d'apporter un certain nombre de corrections, d'améliorations et de nouvelles fonctionnalités à un site de création de formulaires en ligne

Le site original fourni permet de créer un formulaire composé d'une seule question, avec un lien pour inviter des contacts à répondre. Le créateur du formulaire se charge lui-même de la transmission du lien. Les personnes recevant le lien peuvent ensuite choisir une réponse à la question parmi celles proposées

Les principaux ajouts demandés sont :
- Adaptation du site pour mobiles (responsive)
- Affichage des résultats
- Création de compte

Principaux jalons :
- Mardi : Mise en ligne du site, en local et sur le serveur
- Mercredi : Affichage des résultats, ajout de questions supplémentaires
- Jeudi : Création et gestion des comptes
- Vendredi : Gestion des options / rendu final

La gestion du responsive et le design des pages se fait en parallèle.

Mission 0 : prise en main

L'archive fournie (en .zip ou .tar.gz, au choix) contient les fichiers du projet, organisés comme suit :

- application
  - config : les fichiers de configuration
    - config.php.example à renommer en config.php et adapter pour l'installation
    - routes.php à modifier pour ajouter des pages au site
  - core : les fichiers du framework, à ne pas modifier
  - pages : chaque fichier correspond à une page du site ou une action
  - views : contient les vues blades
  - models : contient les fonctions permettant de récupérer ou modifier des données dans la base
  - helpers : pour mettre les autres fonctions utilisées à plusieurs endroits
  - cache : contient les fichiers générés par blade. Le répertoire doit être modifiable par le serveur (pensez à modifier les droits !)
- public : les fichiers css, images et autres fichiers référencés par les pages du site
- vendor : généré automatiquement par composer, ne pas modifier
- readme.txt : instructions pour l'installation
- qel.sql : fichier contenant le code sql pour créer la base de données du site

Installation

Le site est à installer en local sur votre machine (ou une machine des salles tp), ainsi que sur le serveur mis en place pour la SAÉ. Il est recommandé d'éviter d'utiliser la dernière version de php (php 8.1) qui pose des problèmes avec les bibliothèques utilisées.

- Serveur SAÉ : 172.31.144.142
- Comptes : groupe_xx
- Accès : ssh, scp, ftp
- PhpMyAdmin (même login/mot de passe, http://172.31.144.142/phpmyadmin
- Le répertoire public_html est visible sur http://172.31.144.142/~groupe_xx/
- Le projet sera placé directement dans public_html/qel/ et donc visible sur http://172.31.144.142/~groupe_xx/qel/

La procédure complète d'installation est détaillée dans le readme.txt du projet

Le framework

Un framework est un ensemble de fichiers et une façon d'organiser le code permettant de faciliter le développement d'un site web. Il permet d'être plus efficace, en évitant d'avoir à tout réinventer pour chaque site.

Le framework utilisé ici est volontairement minimaliste, et similaire à ce qui a été fait en TP php pour faciliter la prise en main.

Voici ce qui se passe quand on demande la page
http://172.31.144.142/~groupe_xx/qel/action=index

Le fichier index.php analyse de la requête pour trouver la page à afficher. C'est le fichier application/config/route.php qui indique comment traiter l'action index , avec la ligne suivante :

add_route('index');

Cette ligne indique que l'action 'index' est valide. Le fichier correspondant se trouve dans application/pages, avec le même nom et l'extension '.php'
L'action index est donc associée au fichier application/pages/index.php qui est donc exécuté

On voit que ce nouveau fichier php est très court, et contient juste la commande

echo $blade->run('index');

Le fichier blade correspondant est affiché. Son nom complet est applications/views/index.blade.php

Le fichier blade commence par

@extends('templates.main')

Cela signifie que la structure de la page est définie dans le fichier blade applications/views/templates/main.blade.php

Dans les cas plus complexes, la page appelle des fonctions qui interrogent la base de données, et qui sont alors définies dans des fichiers du répertoire applications/models/. Ces fichiers sont inclus dans le php de la page. Certaines fonctions sont déjà écrites, les autres sont créées mais vides (à vous de les compléter en fonction des indications en commentaire et de vos besoins au cours du projet). Il est recommandé de ne pas modifier les signatures des fonctions proposées (c'est à dire la première ligne)

Principales conventions

Le site existant respecte un certain nombre de conventions. Il vous est demandé de respecter vous aussi ces conventions dans votre code.

- Les mêmes noms sont utilisés partout où c'est possibles : pour les paramètres ($_GET comme $_POST), les variables en php, les champs dans la base de données, et les variables blade. Pour les paramètres, leur présence est vérifiée par le framework conformément à ce qui est indiqué dans les routes, et les paramètres ainsi déclarés sont transformés en variables (par exemple, $_GET['action'] est copié dans $action)

- Généralement, chaque action correspond soit à un traitement, soit à une page. Les traitements se terminent par une redirection. Les pages se terminent par l'affichage d'une vue blade. Les fichiers php correspondant sont tous dans le répertoire pages

- Des constantes sont définies en fonction de la configuration pour faciliter l'écriture des liens et la migration (installation sur un serveur différent). Il est important d'utiliser ces constantes dans l'écritures de tous les liens, afin d'éviter tout problème de lien lors du déploiement. Avant d'écrire un chemin, pensez à regarder les constantes disponible dans config.php, ainsi que les fonctions prévues à cet effet (dans les modèles), et souvent déjà écrites.

- En plus des conventions du framework, ce site utilise des icônes colorées avec ou sans texte à coté. En blade, le code pour les utiliser est le suivant :

  {!!make_icon('calendar-plus',' Créez votre questionnaire maintenant !','icon-neutral')!!}

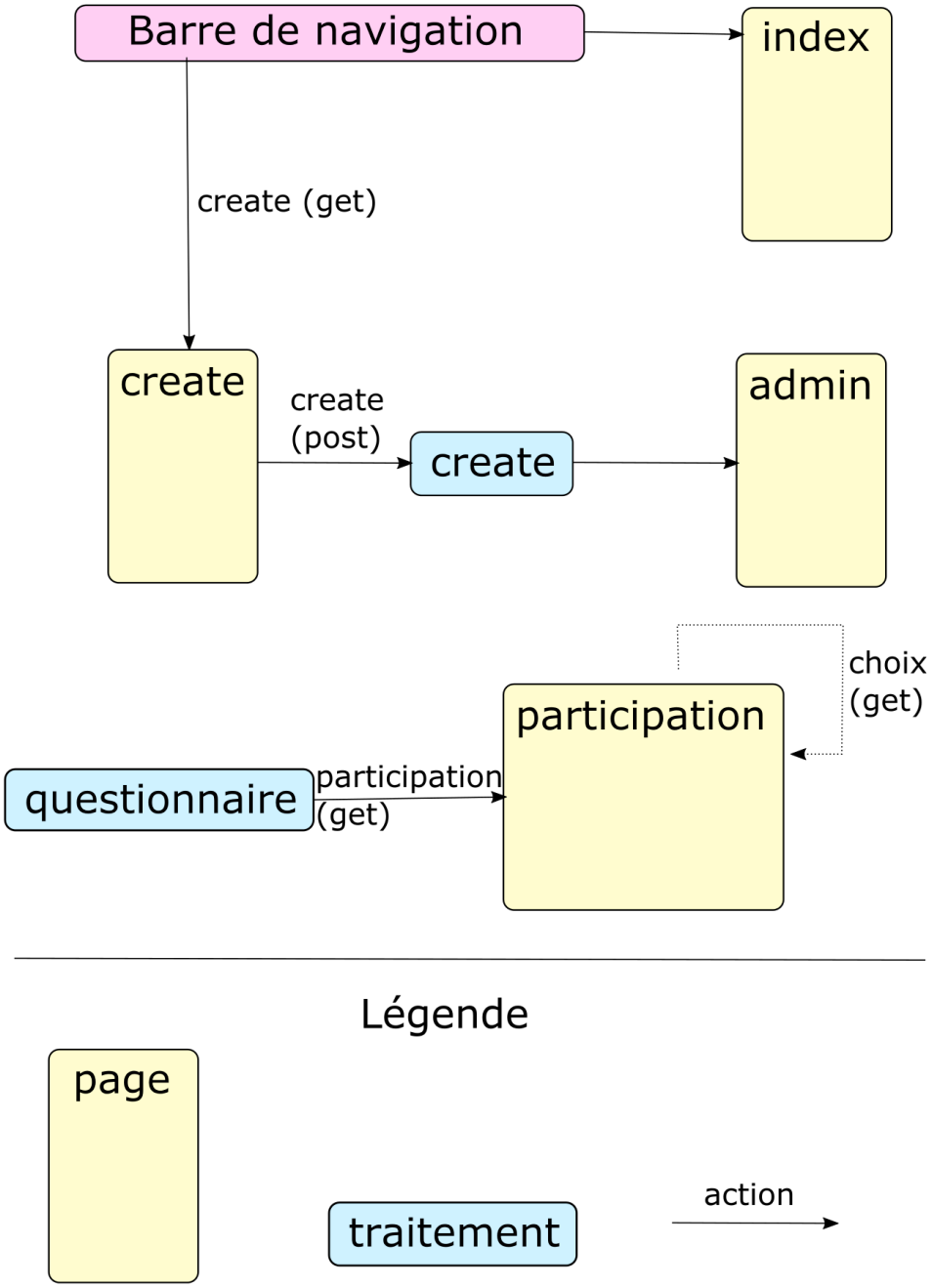
  Le premier paramètre fait référence à un fichier du répertoire public/icon. Par exemple, calendar-plus correspond à bx-calendar-plus.png (le préfixe et l'extension sont ajoutés automatiquement). Les deux autres paramètres sont optionnels, et correspondent au texte et à une classe pour le style.

Les icônes déjà présentes ont été récupérées en png via https://boxicons.com , en blanc (ffffffff) et 144x144. Vous pouvez facilement ajouter les vôtres de la même manière.

Plan du site

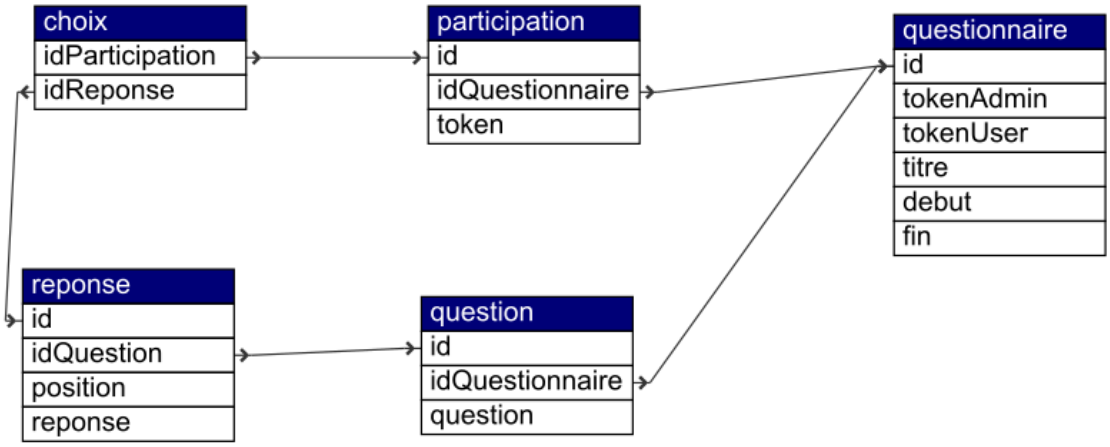
Le site original est organisé comme indiqué ci-dessous :

Pour simplifier le schéma, la barre de navigation est présentée à part. On évite ainsi de répéter les liens vers index et create. Les liens qui partent des traitements sont réalisés via une redirection (commande header('Location: …'))



Base de données

Le fichier qel.sql contient la base de données du site. Voici son schéma :



Un questionnaire comporte une ou plusieurs questions. Chaque question est reliée à des réponse. Les différents choix de réponse d'une même personne sont regroupés dans une participation à un questionnaire.

Les tokens sont des suites aléatoires de caractères qui permettent d'autoriser certaines pages ou actions. On aurait pu utiliser les id pour cela, mais un id est trop facile à deviner (ils commence par 1 et augmentent généralement de 1 à chaque fois…) pour servir d'autorisation.

La base de données est à installer via phpmyadmin. Il sera nécessaire de la compléter pour réaliser certaines missions, et la version complétée fait partie des livrables finaux (n'oublier de l'exporter et de la mettre dans vos fichiers).

Mission 1 : ajout des mentions légales

Présentation

Tout site internet doit présenter un certain nombre d'informations. Il manque sur le site actuel une page présentant ces informations légales, il faut donc l'ajouter. La page sera accessible avec l'action 'legal'

Le design de cette page est libre, mais doit rester cohérent avec le reste du site

Marche à suivre

Il est recommandé de procéder étape par étape, en testant à chaque fois.

1. Ajout d'un lien dans le footer (fichier main.blade.php), vers l'action 'legal'. Inspirez-vous des liens de navigation pour utiliser la bonne syntaxe .
2. Ajout d'une ligne dans routes.php pour faire le lien entre l'action 'legal' et la page 'pages/legal'
3. Création du fichier pages/legal.php, avec un contenu simple pour le test.
4. Création du fichier legal.blade.php, et appel de celui-ci à partir de pages/legal.php
5. Rédaction du contenu dans le fichier blade.

Contenu

Le contenu et le design sont libres, dans le respect de l'objectif de page. Les informations minimales à fournir sont les suivantes :

- Editeur (qel)
- Hébergeur (l'iut)
- Développeur (mmi création — vous)
- Conditions d'utilisation
- Cookies (il n'y a que le cookie de session, pas de problème rgpd)
- Données personnelles (il n'y en pas pour l'instant)

Mission 2 : affichage des résultats

Présentation

Actuellement, il est impossible de trouver sur le site combien de fois chaque réponse a été donnée. S'agissant de l'objectif du site, il faut y remédier rapidement

Marche à suivre

Il est recommandé de procéder étape par étape, en testant à chaque fois

1. Dans pages/participation.php, on va appeler get_responses_with_freq() (définie dans models/participation.php) avec les bons paramètres, au lieu de get_reponses_by_token() : l'idée est de récupérer les mêmes informations qu'avant, avec les fréquences en plus
2. Dans elements/question.blade.php, on va changer l'affichage d'une réponse :
   @php($pourcent = $question['total']==0 ? 0 : ((int)((100*$reponse['count'])/ $question['total'])))

   {{$reponse['reponse']}}
   {{$pourcent}}%
   Au lieu de juste:
   {{$reponse['reponse']}}
3. Il reste maintenant à compléter le CSS pour obtenir une barre de longueur adaptée

Mission 3 : ajout de questions

Présentation et analyse

On souhaite maintenant avoir plusieurs questions dans un même formulaire.

A partir de la page d'administration, il faudra donc mettre à nouveau la partie du formulaire initial ayant permis de créer la première question, pour pouvoir en ajouter de nouvelles. Néanmoins, un nouveau problème se pose alors : comment garantir que tous ceux qui remplissent le formulaire ont bien eu les mêmes questions ?

La réponse passe par l'ajout d'un statut au questionnaire, qui permet d'indiquer s'il est encore possible de modifier le formulaire, ou si on peut y répondre (les deux étant mutuellement exclusifs). Cela permettra aussi d'empêcher les modifications après la date de fin.

Il faudra donc ajouter dans la base un statut à chaque questionnaire, avec les valeurs suivantes possibles :

- creation (le questionnaire est modifiable, mais on ne peut pas y répondre)
- publie (le questionnaire est définitif, on peut y répondre)
- termine (on ne peut plus rien modifier)

Le passage vers 'publie' se fera par un bouton, le passage vers termine en testant la date.

Pour faciliter les tests, il y a une fonction has_passed() dans core/date.php qui permet de tester si une date est passée, et une constante peut-être définie dans config.php pour que le résultat de cette fonction soit automatiquement mis à vrai ou faux. Il n'est ainsi pas nécessaire d'attendre la date de fin pour tester son effet.

Marche à suivre

Il est recommandé de procéder étape par étape, en testant à chaque fois

1. Ajout du statut dans la base de données, avec une valeur par défaut (creation)
2. Ajout d'un bouton (dans la partie cta) pour publier un questionnaire (changer le statut vers 'publie'). Le bouton renverra vers la page d'administration, avec le même token, et avec un paramètre GET en plus (par exemple 'publie')
3. Ajout de la route avec 'publie' en deuxième paramètre pour la page d'administration. Attention à mettre cette règle avant celle existante : les règles sont testées dans l'ordre et la première applicable est utilisée, on met donc les règles avec plus de paramètres en premier.
4. Dans la page d'admin, si 'publie' est défini on appelle set_statut() pour publier
5. Prise en compte du statut dans l'affichage : lorsque le statut le permet, ajout d'un formulaire pour ajouter une question. Le formulaire sera forcément en POST, avec comme action la page d'administration. Comme on est en POST, il est nécessaire

pour récupérer le token de l'envoyer à partir du formulaire, on l'ajoute donc en paramètre caché (input type="hidden" value=…). N'oublier pas d'ajouter la route correspondante

6. Gestion de l'ajout d'une question dans la page d'administration (s'inspirer de ce qui a été fait à la création du formulaire, toutes les fonctions utiles pour modifier la base de données existent déjà dans les modèles)
7. Mise à jour du statut en fonction de la date : compléter et utiliser la fonction update_status().

Mission 4 : ajout des comptes utilisateurs

Présentation

On souhaite ajouter la possibilité de créer des comptes, pour gérer plus facilement les différents questionnaires créés ou auxquels on a participé

Le cahier des charges est le suivant :

- On utilise un email comme login, le mot de passe ne sera pas stocké en clair (utilisation de sha1).
- La création de compte doit rester optionnelle, pour la création comme pour la consultation des questionnaires
- Si on est connecté, la consultation d'un questionnaire doit relier le compte et le questionnaire. On distingue les questionnaires créés et ceux auxquels on a juste répondu
- La page d'un utilisateur doit afficher les questionnaires reliés, par type (créé ou répondu), par ordre chronologique inverse
- Le comportement de la page de profil dépend du fait d'être loggé ou pas. Un utilisateur loggé voit ses formulaires, un non loggé un formulaire d'inscription/connexion

Le design des pages de connexion et/ou inscription ainsi du profil (liste des évènements) est libre, mais doit rester cohérent avec le reste du site

Tâches à réaliser

1. Ajout de la table utilisateur dans la base de données, ainsi que le nécessaire pour la relier aux deux types de questionnaires (créés/remplis). Il peut être utile de raisonner en termes de cardinalice (cf cours de BDD, feuille sur les modifications)
2. Ajout du lien vers la page de profil ou le formulaire de connexion dans la barre de navigation
3. Création de la page de profil, avec lancement du blade correspondant à la situation (profil ou login, à créer tous les deux). Les formulaires à afficher sur le profil sont récupérés en utilisant les fonctions dans models/user.php
4. Ajout du formulaire de login avec une action (sur la même page) qui connecte l'utilisateur et affiche le profil. La validation du login se fait avec la fonction log_in, toujours dans user.php
5. Affichage de la liste des questionnaires. On réutilise pour ça elements/questionnaire.blade.php
6. Création des liens entre utilisateur et questionnaire quand on est connecté à son compte et que l'on consulte ou créé un formulaire en utilisant un token.

Mission 5 : options (conditions d'affichage)

Présentation

On souhaite ajouter la possibilité de définir quand les résultats seront visibles : toujours, après avoir répondu, ou une fois la date de fin du questionnaire atteinte

Tâches à réaliser

1. Modification de la base de données pour mémoriser pour chaque questionnaire les conditions d'affichage du résultat
2. Ajout de l'option dans le formulaire de création, et dans les fonctions correspondantes
3. Affichage des résultats prenant en compte cette option

Mission 6 : option (ajout de réponses)

Présentation

En plus des réponses proposées, on souhaite dans certains cas autoriser des réponses libres à certaines questions : il sera alors possible pour celui qui répond d'ajouter sa propre réponse

Tâches à réaliser

1. Ajout de l'information (question ouverte ou non) aux questions dans la base de données. Il faut faut aussi mémoriser pour chaque réponse si elle est dans les propositions originales ou pas (on pourra effacer une réponse devenue inutile, mais seulement si elle a été ajoutée pas si elle fait partie des questions d'origines)
2. Affichage dans les formulaires de l'option et d'un nouveau champ de saisie quand cela est nécessaire
3. Traitement des nouvelles réponses (ajout proprement dit)
4. Attention, dans le cas où un utilisateur change d'avis, il est possible qu'une réponse qu'il a créé ne soit plus du tout utilisée. On doit alors la supprimer

Mission 7 : design et responsive

Présentation

Il est demandé de rendre le site actuel responsive. Cela nécessite d'imaginer une mise en page alternative, adaptée aux terminaux mobiles.

Il est parfaitement acceptable de modifier la structure du template utilisé, à condition que l'affichage sur les écrans larges ne soit pas altéré.

L'utilisation d'un menu hamburger peut aider à adapter la barre de navigation

Livrables

Le travail demandé dans chaque mission sera mis en ligne sur la machine SAE au fur et à mesure (chaque soir au minimum)

Le rendu est constitué des éléments suivants :

- Tous les fichiers du répertoire public_htlm sur votre compte sur la machine SAE
- La base de donnée actuelle correspondant à ce compte (même login), accessible via phpmyadmin
- Le fichier qel.sql, qui doit permettre de recréer votre base (sans les données des utilisateurs, juste ce qui est nécessaire au fonctionnement du site).
- Un fichier synthèse.txt, à la racine du compte sur votre machine SAE, avec pour chaque mission :
  - Le nom et le numéro de la mission
  - Les étapes complétées et testées, celles commencées et les autres
  - Une courte description (2/3 lignes) de ce qui fonctionne, reste à faire (le plus court des deux) ou pose problème (ex : quel bug)