École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise

ENSIIE - Évry

Projet Web 2019-2020

Groupe n°30 : VodkalIIsE

Site Web VocasIItE

Rapport de Projet Web : VocasIItE

LEFOULON François LEFRANÇOIS Pierre-Marie
DUBROMER Valentin UTH Rathea

Mai 2020

Table des matières

1 Introduction

L'objectif du projet est la création d'un site pour l'association VocalIIsE, qui réalise des concerts chantés. En effet, il n'y a pour l'instant pas de manière standard de regrouper les chansons des différents concerts (soirées), de savoir qui chantera quelle chanson, de retrouver le programme d'un concert, ou encore d'avoir un affichage des paroles lisible et pratique lors des répétitions.

Les problématiques sont les suivantes :

1. Permettre aux utilisateurs d'ajouter des chansons avec leurs paroles et un ou plusieurs lien(s) d'écoute pour les répétitions, de manière ergonomique
2. Permettre d'afficher proprement les soirées prévues et les chansons associées
3. Permettre de retrouver les chansons et chanteurs d'une soirée passée ou à venir ; s'inscrire pour chanter
4. Limiter l'accès à certaines données à certains utilisateurs en fonction de leur rôle dans l'association
5. Mettre en place une authentification à plusieurs niveaux avec, de préférence, une compatibilité AriseID.

2 Répartition des tâches

1. Pierre-Marie : création de la BDD, Makefile, aspect général et design, création et modification de chansons et de soirées, recherche de chansons, visualisation des chansons et soirées, authentification, autorisations
2. Valentin : authentification, tests, aspect des pages publiques
3. Rathea : tests, aspect des pages publiques
4. François : création de la BDD, barre de navigation, recherche de soirées, autorisations

3 Design du site

Nous utilisons le framework CSS *Bulma* (bulma.io), qui propose de nombreux outils à l'aspect sobre et moderne, qui s'adaptent autant à une utilisation sur ordinateur que sur site mobile, permettant à VocasIItE d'être responsive.
Bulma contient notamment :

1. Des cadres pour les différentes parties de la page qui aident à l'agencement des éléments et dont la taille et la disposition s'ajustent automatiquement
2. Une barre de navigation
3. De jolis formulaires
4. Des éléments individuels stylisés comme des boutons et des champs input

Cependant, cela ne nous a pas empêché d'écrire de bonnes pages de CSS, notamment pour le formulaire de la page editSong.php

Un script du site fontawesome nous donne accès à sa bibliothèque d'icônes vectorielles, utilisées un peu partout.

Ces outils ont largement simplifié le travail. Une partie plutôt difficile était la gestion de la barre de navigation, car il a fallu chercher longuement dans la documentation pour savoir exactement quelles classes ajouter dans chaque balise, et comment les agencer.

4 Base de Données

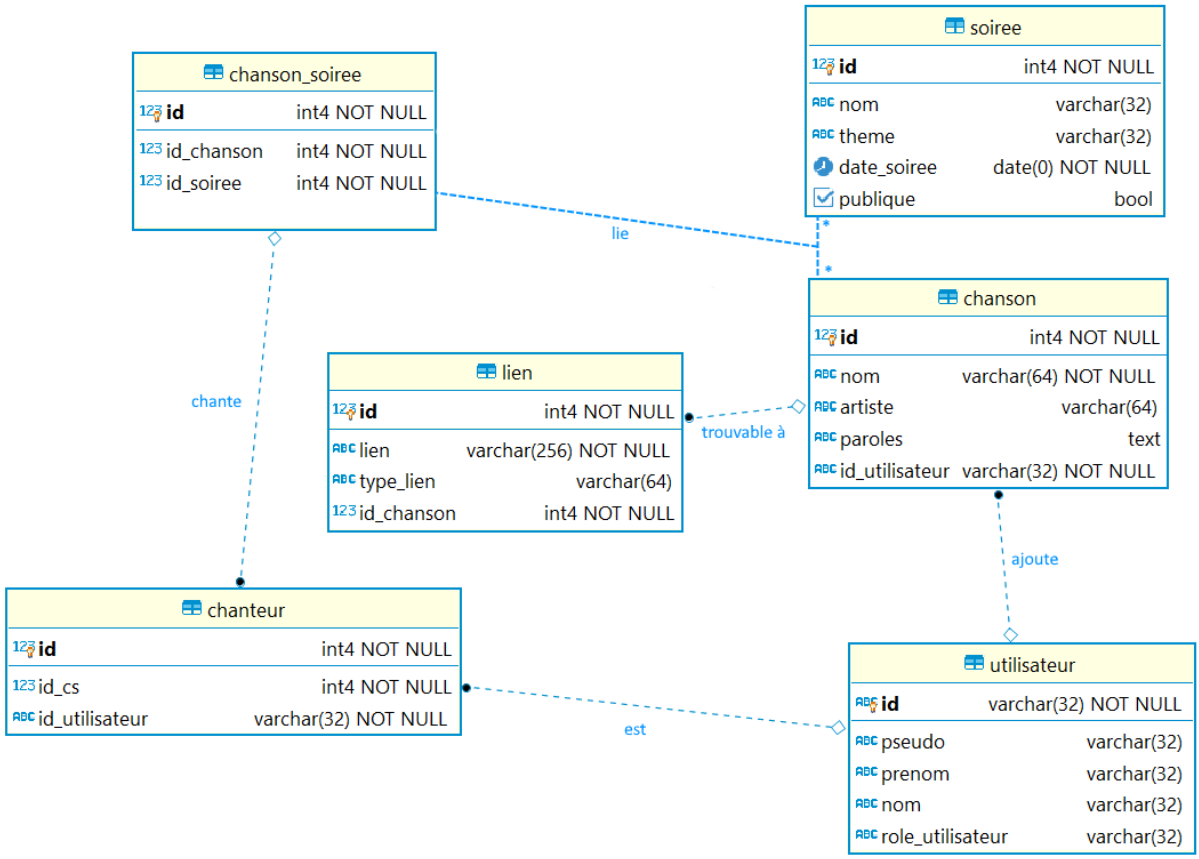

Figure 1: Diagramme UML commenté de la base de donnée

5 CRUD

Nous utilisons une base de données PostgreSQL, pour laquelle nous avons pris soin de créer un utilisateur *ensiie* et un mot de passe *ensiie* à qui on donne tous les droits sur cette dernière. Ceci est évidemment limité au cadre du projet et sera modifié lors d'une implémentation définitive. Les requêtes sont préparées dans le PHP grâce à PDO, afin d'éviter les injections.

Il était prévu de stocker les données des utilisateurs dans des objets PHP "Entity", puis d'utiliser des méthodes Hydrator pour mettre à jour la BDD en fonction des objets. En fin de compte, ça ne s'est pas avéré utile car cela semblait superflu et plus délicat à manipuler, même si nous

aurions pu, de cette manière, faire de l'objet en PHP.
A la place, tout est géré dans les fichiers PHP concernés, notamment les requêtes SQL qui sont faites sur le moment, quand il y en a besoin.

Nous avons cependant écrit un script *autoloader.php* qui charge tout ce dont nous avons besoin pour chaque page. Par exemple, ce fichier contient un appel à *session_start* et inclut des fichiers de *src/Model*, comme la classe qui permet de se connecter à la base de données.

6 Recherche (de chansons, de soirées)

L'objectif était de faire des pages de recherche agréables et élégantes.
Impensable, donc, de faire un formulaire qui irait poster dans une autre page en redirigeant l'utilisateur. Nous souhaitions pouvoir rester dans la même page, c'est pourquoi nous sommes passés par la technologie *AJAX*.

En effet, l'affichage des résultats est géré côté client, en JavaScript. C'est aussi d'ici que nous effectuons des requêtes *XMLHttpRequest*.
Un eventListener écoute la moindre modification du formulaire (entrée d'un caractère par exemple), puis on essaye de faire une requête, celles-ci étant limitées à une par seconde pour éviter une avalanche de requêtes si quelqu'un écrit très rapidement.

Le fichier recevant les requêtes s'apelle *searchSongs.php* ou *searchSoirees.php*. Ce fichier fait une requêtes SQL et utilise plusieurs *echo* sur le résultat pour formater le tout en JSON.
Le code JS récupère le JSON, le parse et s'occupe de l'affichage dans une liste.
Un résultat JSON contient aussi les permissions qu'a l'utilisateur du site sur chaque résultat, pour afficher les bons boutons pour chaque chanson ou soirée.

La recherche de chanson passe par deux champs (titre et artiste) et celle de soirée est plus simple : l'utilisateur sélectionne simplement une année scolaire pour en afficher les soirées.Ces années scolaires sont générées au début dans le formulaire par du code JS : on commence par l'année 2018-2019 et on créé autant de choix qu'il y a d'années scolaires jusqu'à celle en cours incluse.

Vous trouverez un schéma explicatif au début de la page suivante.

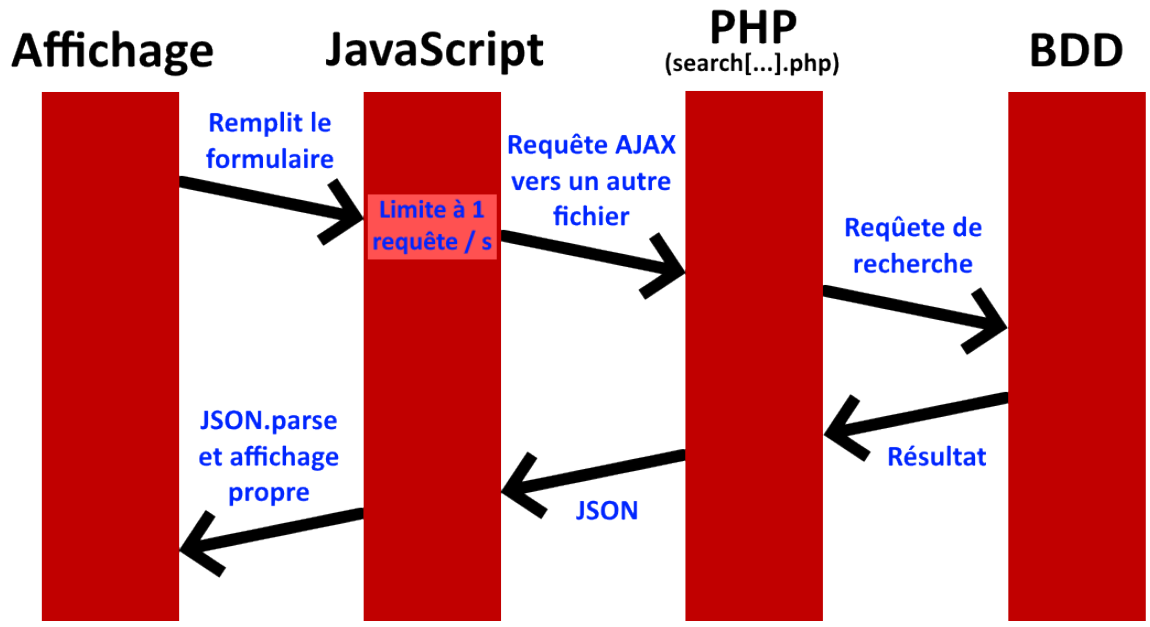

Figure 2: Explication de la recherche

7 Création, édition, affichage de chansons et soirées

L'affichage des chansons est optimisé pour les répétitions, puisqu'on a facilement accès à un ou plusieurs liens pour écouter la chanson, mis en valeur par des icônes, et la possibilité de zoomer sur le texte.

Lorsqu'un utilisateur crée ou modifie une chanson, il peut la taper à l'aide d'un langage style Markdown, qui est parsé en HTML à l'aide de la librairie ShowdownJS incluse dans le projet.

Sur la page de recherche, les icônes pour voir, éditer et supprimer une chanson/soirée sont affichées, ou non, selon les autorisations de l'utilisateur, sous la forme d'icônes sur les résultats de la recherche. Cela a représenté un travail important pour conditionner l'affichage des différentes icônes et leurs actions en fonction des droits de l'utilisateur.
En effet, il n'a pas fallu se limiter à l'affichage, mais aux réels autorisations côté *backend*, et comme cela a été fait à la fin, il a fallu revenir sur toutes les pages et les modifier en fonction de chaque droit pour chaque rôle.

Pour savoir quelles personnes chantent à une soirée, il suffit de survoler l'icône "liste" de chaque chanson de la soirée. Le nombre de personnes chantant à chaque soirée étant relativement réduit dans l'association, cela fonctionne bien, il n'y aura jamais de débordement du tag "title" et il n'y a pas besoin d'une page à part entière.

8 Authentification

Nous utilisons l'API OAuth2 avec des scripts fournis par l'association ARISE, dans le but d'épargner aux utilisateurs la création d'un nouveau compte et assurer la meilleure UX aux iiens.

En particulier, pas besoin de créer un système de modification des profils utilisateur car ARISE permet déjà cela sur le site *iiens.net*. Cette intégration nous permet aussi de savoir si un mem-

bre fait partie de l'association VocalIIsE et quel rang il a, et l'on peut directement attribuer le rôle de membre aux utilisateurs qui se connectent, ainsi qu'un identifiant, un prénom, pseudo et nom.

Il faut cependant stocker des informations sur les utilisateurs sur la base de données (pour le fonctionnement général du site), et donc les traiter différemment selon s'ils se connectent pour la première fois au site - on doit soit créer une instance d'utilisateur soit éventuellement la mettre à jour si elle existe. Pour cela, on utilise un particularité de PostgreSQL : un requête *INSERT ... ON CONFLICT DO ....*

Pour utiliser OAuth, on déclare une instance d'OAuthAriseClient, initialisée à l'aide de clés d'application fournies par ARISE, dont les méthodes permettent de récupérer les informations de l'utilisateur.
Nous avons rajouté deux fonctions très utiles pour la gestion des droits dans *autoloader.php* : isAuthenticated(), qui renvoie un booléen, et getDroits(), qui renvoie une chaîne de caractères décrivant le rôle de l'utilisateur actuel du site (récupère des informations dans *$_SESSION*).

La navbar contient les boutons de connexion/déconnexion, implémentés comme

1. Si l'on est déconnecté : un formulaire qui envoie une requête POST à "OAuth.php" qui se charge de la connexion AriseID et un bouton qui redirige vers "loginAdmin.php" qui s'occupe de la connexion administrateur.

2. Si l'on est connecté : soit un formulaire qui envoie une requête POST à "OAuth.php" pour la déconnexion, page qui va ensuite nous rediriger vers "logout.php", soit un bouton qui est un lien direct vers cette page en cas de connexion administrateur.

Un script "login.php" est appelé après chaque connexion, admin ou AriseID, et stocke toutes les bonnes informations dans $_SESSION à l'aide d'une requête vers la base de donnée.
Pour cela, le champ $_SESSION["Id"] doit contenir l'identifiant de l'utilisateur, et ce dernier doit exister dans la base de données.

9 Autres remarques

Il était prévu de faire une page affichant les différents membres de l'association avec leurs rôles et informations respectifs, ainsi que la possibilité pour les administrateurs de modifier ces dernières ; cela n'a pas été fait par manque de temps, et parce que le site contenait déjà assez de fonctionnalités.

10 Conclusion

Le site fonctionne bien, l'UX semble agréable. Les difficultés liées à l'appropriation des outils autres que HTML, CSS, JS ou PHP (par exemple *OAuth* ou *Bulma*) ont demandé du temps mais rien n'a chamboulé le développement. Le site répond à toutes les attentes de la problématique, et continuera d'être mis à jour pendant la vie de l'association, une fois hébergé.

Il était parfois délicat de se coordiner sur les tâches à accomplir, il est arrivé que deux personnes travaillent sur la même page en même temps.
Nous avons quand même pu donner des tâches à tout le monde, même s'il faut reconnaître que la charge de travail n'était pas équitablement répartie.

Dans l'ensemble, nous sommes contents et fiers de ce projet, qui en plus d'être utile à long terme, nous a permis à tous d'en apprendre beaucoup plus sur les technologies web et comment s'en servir.