

Mathématiques et Informatique Appliquées aux Sciences Humaines et Sociales (MIASHS)
Technologie et handicap (HANDI)
UE Interface Homme - Machine / Programmation Web Accessible

>>>

Elaboration d'un site web accessible Opéra national de Paris

Bilo Boury, Charles Cascio, Nicolas Roelandt, Nassim Yousfi
27 juin 2017



Sommaire

Titre	1
Sommaire	3
1 Introduction	5
2 Conception	5
2.1 Répartition des tâches	5
3 Design du site	5
3.1 Design général	5
3.2 La charte graphique	6
3.2.1 Le choix des couleurs	6
3.2.2 Le choix de la police	6
3.3 Les composants	6
3.3.1 La barre de navigation comme repère	6
3.3.2 L'accueil	6
3.3.3 Une carte pour prévoir son déplacement	7
3.4 Les apports visuels	9
3.4.1 L'interraction au service de la lisibilité	9
3.4.2 Des éléments visuels pour apporter davantage de sens	9
3.4.3 Une mise en page réfléchie	9
3.5 Le responsive	9
3.6 Retour et éléments de réflexion	10
4 Réalisation de la base de données	12
4.1 Modèle Conceptuel de Données	12
5 PHP	12
5.1 Requêtes en base	13
5.1.1 Les listes déroulantes	13
5.1.2 Les saisies utilisateurs	14
5.2 Gestion des sessions	14
6 Gestion des réservations	15
6.1 Plan de salle	15
7 Versionnement	15
7.1 Logiciels de gestion du versionnement	15
7.1.1 GIT	15
7.1.2 GitKraken	15
7.2 Hébergement	16
7.3 Avantages	16
7.4 Inconvénients	17
7.5 Problèmes rencontrés	17
8 Evolutions possibles / points non traités	17
8.1 Fonctionnalités / Interface du site	17
8.2 Page gestion.php	17
8.3 Page spectacle.php	17
8.4 Page reservation.php	17

8.5	Page panier.php	18
9	Conclusion	18
1	Annexes	19
1.1	Usecases	19
1.1.1	Jean-Marc, prospect	19
1.1.2	Marcelle et Jacques, clients réguliers	19
1.1.3	Philippe, administrateur	20
1.2	Code python générant un jeu de test de réservations	21
1.3	Documentation de la base de données 1	24
1.3.1	proj_TypeUtilisateur	24
1.3.2	proj_Utilisateur	24
1.3.3	proj_Spectacle	25
1.3.4	proj_Salle	25
1.3.5	proj_Representation	25
1.3.6	proj_Categorie	26
1.3.7	proj_Place	26
1.3.8	proj_PrixPlace	27
1.3.9	proj_Reservation	27
1.3.10	messages_contacts	27
1.3.11	Disponibilité des places	27
1.3.12	Clés étrangères	28
Liste des figures		28
Liste des tables		30

1 Introduction

Ce projet nous a été confié dans le cadre du Master *Mathématiques et Informatique Appliquées aux Sciences Humaines et Sociales (MIASHS)* parcours *Technologie et handicap (HANDI)*. Il s'agit d'un projet associant les cours d'Interface Homme Machine (Dominique Archambault) et Programmation Web Accessible (Isis Truck) de la première année.

Le site présenté ici a été développé par une équipe de 4 étudiants :

- Bilo Boury
- Charles Cascio
- Nicolas Roelandt
- Nassim Yousfi

Cahier des charges Développer un site de réservation de places de spectacle le plus accessible possible.

L'objectif est de développer un site web permettant la réservation de place de spectacle. Le site doit être accessible et donc fonctionnel pour des personnes en situation de handicap (visuel ou moteur).

Nous avons voulu travailler sur un cas le plus concret possible, c'est pourquoi nous nous sommes inspirés du site de l'Opéra national de Paris.

Si les spectacles présentés dans ce projet ont réellement eu lieu, soit à l'Opéra Bastille ou au Palais Garnier, l'inspiration s'arrête là. Le plan de salle, les dates de représentation, les clients sont totalement fictifs.

2 Conception

Ce site a été développé majoritairement le soir et le week-end, sur une période d'un mois et demi environ, nous avons donc voulu le garder le plus simple et fonctionnel possible. D'une part car dans un soucis d'économie de temps et d'énergie, et d'autre part dans un souci d'accessibilité.

2.1 Répartition des tâches

1. maquette et design du site (HTML/CSS) : Charles Cascio
2. MCD et gestion des places et réservations : Bilo Boury
3. Implémentation de la base MySQL et PHP : Nicolas Roelandt
4. Implémentation PHP, gestion des sessions : Nassim Yousfi

Dans un premier temps, nous avons développé le design et le squelette du site et, parallèlement, le modèle de la base de données et son implémentation. Enfin, nous avons rendu le site dynamique en interfaçant le site avec la base à l'aide de PHP et Javascript.

3 Design du site

3.1 Design général

Si le site officiel¹ est superbe (voir fig. 1), nous ne nous en sommes pas servi pour développer le design de notre site.

Comme le montre les figures 2 et 5, le design a beaucoup évolué entre les différentes versions.

1. <https://www.operadeparis.fr>

Elaboration d'un site web accessible Opéra national de Paris

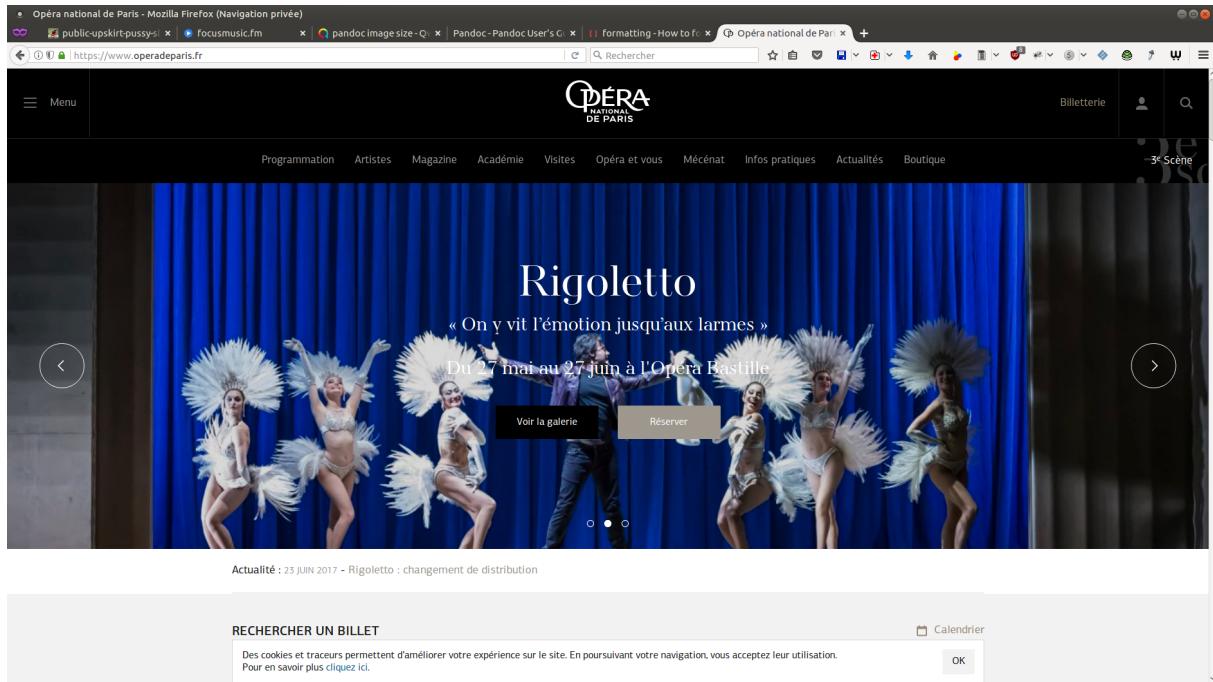


Figure 1. Site officiel de l'Opéra national de Paris

3.2 La charte graphique

3.2.1 Le choix des couleurs

Les couleurs ont d'abord été choisi pour correspondre aux goûts de la population ciblée CSP+. Le noir est la couleur du luxe, du haut de gamme. Le noir et le blanc sont des couleurs classiques, sobres. Ce choix de couleur permet de distinguer aisément le menu du reste de la page et donc se repérer facilement. Enfin ce thème blanc et noir a l'avantage de présenter de forts contrastes qui facilitent la lecture du texte.

En outre le pied de page en gris clair est discret. Il permet de faire le lien avec les autres présences sur le web de l'Opéra National de Paris sans effacer le contenu des pages. En effet, il donne accès aux différents réseaux sociaux de l'opéra de paris. Le vert a été choisi pour les boutons de validation et le bleu pour les boutons marquant le franchissement d'une étape ou l'envoi d'un formulaire.

3.2.2 Le choix de la police

Bootstrap choisit par défaut la meilleure police en fonction du navigateur. La taille de la police d'écriture a été choisi relativement grande pour un maximum de confort à la lecture.

3.3 Les composants

3.3.1 La barre de navigation comme repère

Le menu a été conçu de manière à être le plus simple et le plus lisible possible. Seuls les liens qui doivent être accessibles depuis n'importe quelle page ont été placés dans le menu. Ainsi l'ajout de sous menus n'a donc pas été nécessaire.

3.3.2 L'accueil

L'importance a été donné aux images sur la page d'accueil de façon à accueillir le visiteur de manière agréable et engageante. Cette première page propose directement des spectacles à découvrir et donne envie au visiteur de parcourir le reste du site.

3.3.3 Une carte pour prévoir son déplacement

Une carte a été ajoutée pour donner un point de repère géographique immédiat. Google maps a été préféré à leaflet pour l'accès à streetView et la visite à 360 degrés de l'intérieur de l'opéra.

Elaboration d'un site web accessible Opéra national de Paris

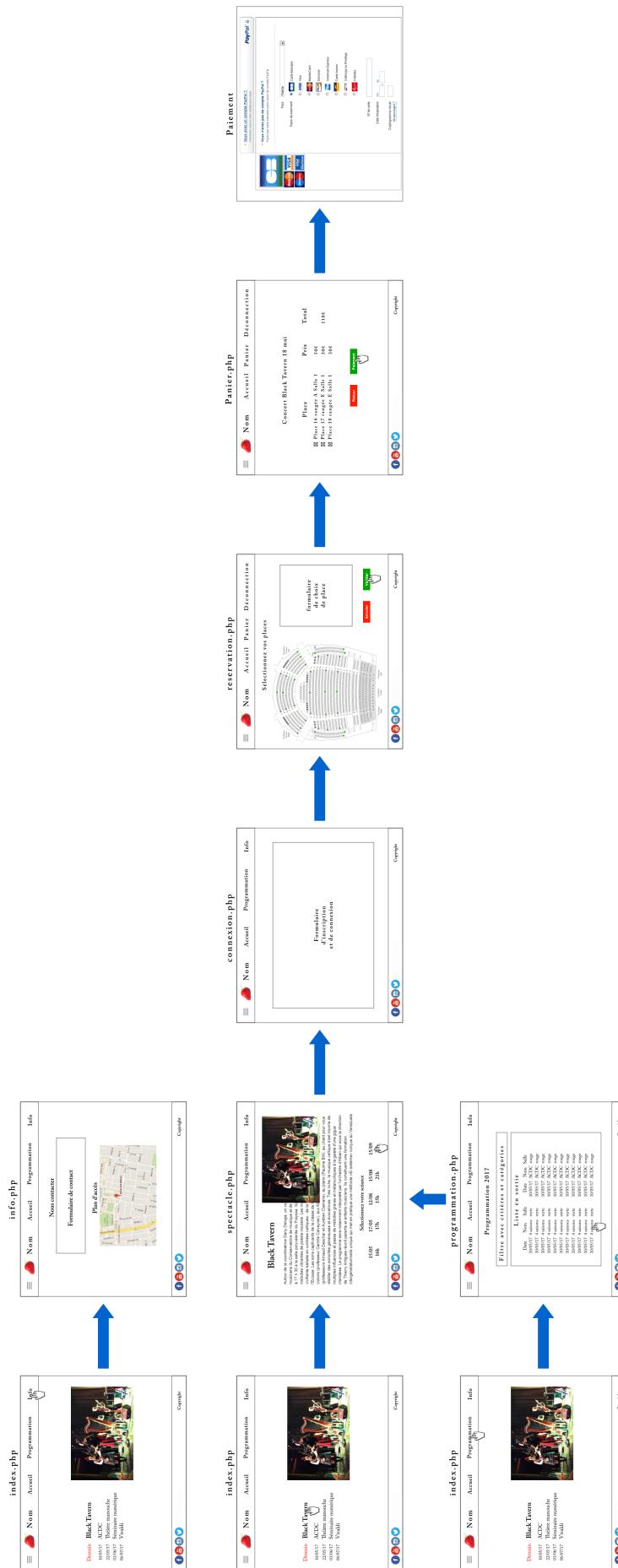


Figure 2. Première ébauche du design

3.4 Les apports visuels

3.4.1 L'interraction au service de la lisibilité

Les liens sont soulignés au passage de la souris et s'éclairent. En plus de donner des informations sur les éléments survolés par la souris, le visiteur a un retour direct sur son interaction avec les éléments du site.

3.4.2 Des éléments visuels pour apporter davantage de sens

Des icônes ont été ajoutés aux éléments importants comme les boutons d'action ou les liens principaux afin de se repérer aisément et de donner davantage de sens à ces éléments.

3.4.3 Une mise en page réfléchie

Une attention particulière a été portée sur le placement des éléments dans la page de façon à utiliser l'espace au maximum tout en aérant le plus possible le contenu. Les pages sont peu chargées, seules les informations nécessaires sont présentes. Ce choix de mise en page permet également d'éviter les ascenseurs horizontaux et donc de limiter les efforts des personnes à handicap moteur.

3.5 Le responsive

Un site pour tous les types d'écrans et tous les navigateurs. Les pages ont été pensées pour être totalement responsive, chose qui est facilitée par le modèle en grille de bootstrap et les flexboxs de HTML5. Ensuite le site a été conçu pour fonctionner sur tous les navigateurs (même IE8).



Figure 3. Sur smartphone menu fermé

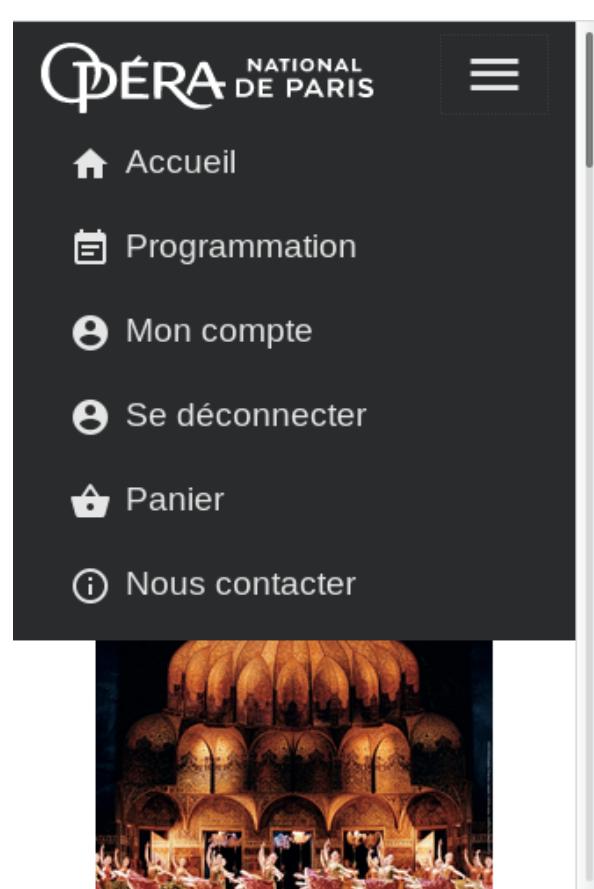


Figure 4. Sur smartphone menu ouvert

3.6 Retour et éléments de réflexion

Les premiers choix de design n'ont pas toujours été respectés pour des raisons de faisabilité (partie PHP). En outre il était prévu d'ajouter un lien d'évitement sur le menu et le pied de page. Toutefois, le menu et pied de page ont peu de liens. C'est pourquoi ils ne sont pas très gênants lors du parcours du site avec la touche de tabulation.

Elaboration d'un site web accessible Opéra national de Paris

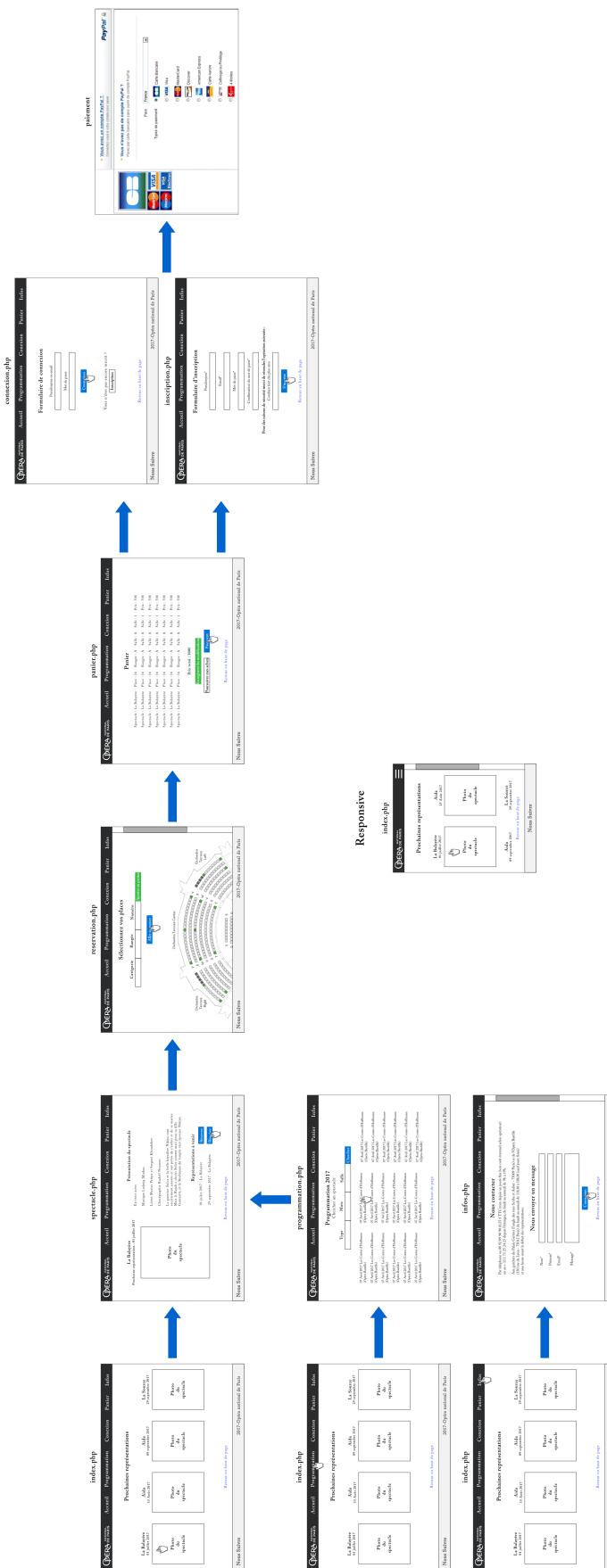


Figure 5. Design actuel

4 Réalisation de la base de données

La base de données a été implémentée avec MySQL, car cet implantation est courante pour des sites web. Il est donc aisément de trouver de la documentation à ce sujet ainsi que des packs logiciels fournissant la base de données et le serveur web pour un usage local.

Enfin, il s'agit de la base de donnée installée sur le serveur Handiman.

4.1 Modèle Conceptuel de Données

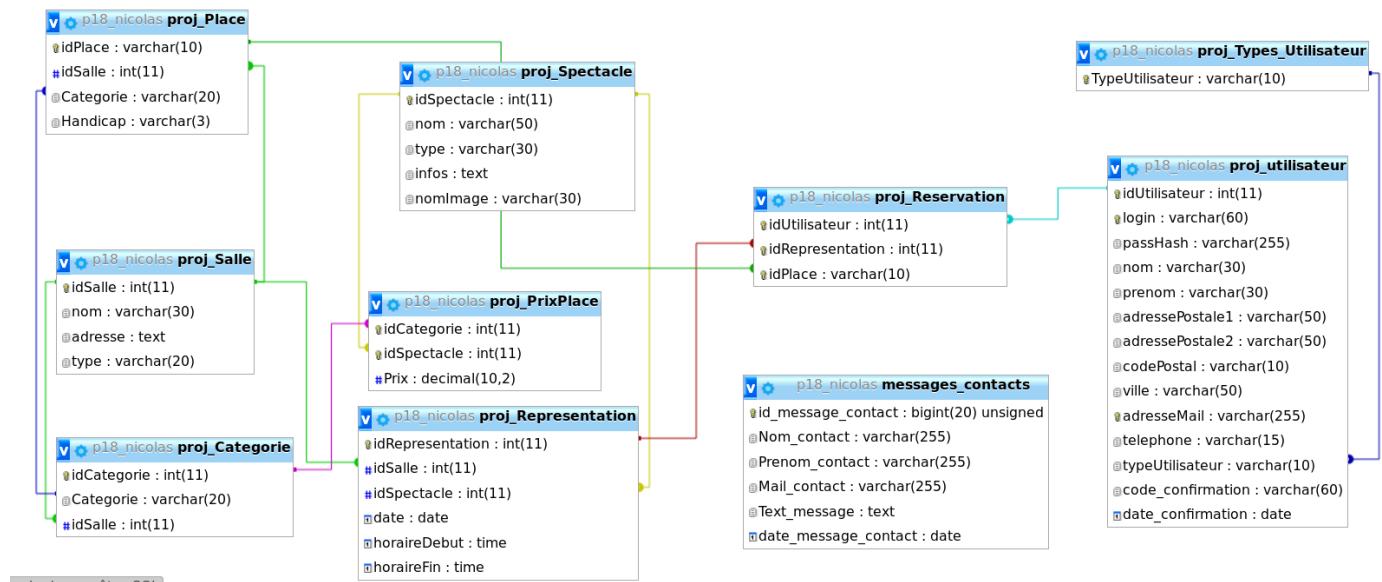


Figure 6. Modèle Conceptuel de Données

5 PHP

De façon général, nous avons essayé de conserver un maximum d'accessibilité pour notre site. Ainsi, nous avons volontairement favorisé l'usage de *PHP* par rapport à *Javascript*. Ceci afin de faciliter l'usage du site avec les lecteurs d'écrans.

De cette manière, la page de réservation de place comporte deux manières de choisir un siège :

- soit par un formulaire ;
- soit en cliquant sur la place voulue de la représentation de la salle.

Cette deuxième option a nécessité l'utilisation de *Javascript* mais n'est pas obligatoire.

Le block de code ci-dessous montre diverses techniques utilisées pour factoriser le code en insérant définissant des chemins relatifs vers les fichiers à inclure et en insérant le code nécessaire à la connexion à la base de données, à l'insertion du menu, du head, du footer et des fonctions *Javascript*.

```

<?php if ( session_status() == PHP_SESSION_NONE ) { session_start(); } ?>
<?php
    // Définition des chemins d'accès aux fichiers
    $path_root = "../";
    $path_structure = $path_root . "structure/";
    $path_pages = $path_root . "pages/";
    $path_images = $path_root . "images/";

    # inclure la connection à la base de données pour vérifier si les infos
    # existent ou pas
    require_once "$path_structure" . 'base.php';
  
```

```
# inclure la fonction debug
require_once "$path_structure).'fonctions.php';
?>

...
<!DOCTYPE html>
<html lang="fr">
<!-- Inclusion <head> -->
<?php include($path_structure."head.php"); ?>
<body>
<!-- Inclusion menu -->
<?php include($path_structure."menu.php"); ?>

...
<!-- Inclusion pied de page -->
<?php include($path_structure."footer.php"); ?>
<!-- Inclusion CDN et fichiers javascript -->
<?php include($path_structure."JS.php"); ?>

</body>
</html>
```

Le fait de procéder de cette manière favorise l'homogénéité du site et facilite la création et la maintenance des pages de notre site. Nous avons aussi utilisé PHP pour rendre le site dynamique à l'aide de requêtes au sein de la base de données et pour gérer les sessions utilisateurs.

5.1 Requêtes en base

5.1.1 Les listes déroulantes

Pour limiter les soucis de cohérences sur les requêtes, les listes déroulantes étaient liées avec les valeurs déjà présentes en base. Donc derrière chaque liste déroulante, il y a une requête de type *query* qui va chercher des informations précises dans la base et pré-rempli les balises html de type *SELECT*.

```
<!-- Type -->
<div class="form-group d-flex flex-column">
    <label for="typeSelect">Type</label>
    <select class="custom-select" id="typeSelect" name="typeSelect">
        <option selected>Choisir...</option>

        <?php
            // requête de sélection des options de
            // type de spectacle présents dans la base
            $sql = "SELECT DISTINCT (s.type) AS type
                    FROM proj_Spectacle AS s WHERE 1 ORDER BY type ASC";

            $req = $pdo->query($sql);

            while ($data=$req->fetch()) {
                echo "<option value=".$data->type.">".$data->type."</
                option>\n";
            }
            $req->closeCursor();
        ?>

    </select>
</div>
```

5.1.2 Les saisies utilisateurs

Pour toutes les requêtes nécessitant une saisie de l'utilisateur (champ texte par exemple), nous avons contrôlé la saisie à l'aide d'expressions régulières si besoin, toutes les saisies sont passées dans la fonction php *htmlspecialchars*.

Puis toutes les requêtes sont préparées avant d'être exécutées.

L'exemple de code ci-dessous montre le cas où un utilisateur souhaite changer son login. Tout d'abord, le remplissage de la variable `$_POST['inputpseudo']` est testé, la valeur fournie doit répondre à l'expression régulière définie. Sinon un message flash de type *danger* est affiché disant que le nouveau login n'est pas valide.

Puis la requête est préparée et exécutée avec les valeurs passées en paramètres. La saisie est contrôlée à ce moment avec *htmlspecialchars* pour éviter l'injection de code sql.

Un message flash est alors envoyé pour indiquer que l'opération s'est bien déroulée.

```
if (!empty($_POST['inputpseudo']) && !preg_match('#^[\w\-\_]+$#', $_POST['inputpseudo'])) {  
  
    $SESSION['flash']['danger'] = "Votre nouveau pseudo n'est pas valide !"  
}  
  
else if (!empty($_POST['inputpseudo']) && preg_match('#^[\w\-\_]+$#', $_POST['inputpseudo'])) {  
  
    // Je récupère l'id utilisateur qui vient de se connecter  
    // je l'affecte à la variable $user_id que je met en parametre dans ma  
    // requete  
    // sql pour changer les infos  
  
    $user_id = $SESSION['auth']->idUtilisateur;  
  
    $req = $pdo->prepare("UPDATE proj_utilisateur SET login = ? WHERE  
        idUtilisateur=?");  
  
    $req->execute([htmlspecialchars($_POST['inputpseudo']), $user_id]);  
}  
$SESSION['flash']['success'] = "Votre pseudo a été mis à jour !";
```

5.2 Gestion des sessions

La variable de session contient deux index :

- l'index `$_SESSION["auth"]` (authentification), qui contient tout les attributs liés à l'utilisateur inscrit dans la page d'inscription (*register.php*) ainsi que les informations nécessaires à la validation de son compte affectées dans le fichier *confirmation.php*.
- L'index `$_SESSION["flash"]`, qui contient tous les messages d'erreurs et de succès relatifs à la gestion des formulaires et des redirections.

Pour factoriser l'ouverture de la super variable dans toutes les pages, nous avons effectué cette ouverture dans le fichier *menu.php* qui est présent dans toutes les pages du site, néanmoins nous avons, à cause des nombreuses inclusions de fichiers PHP dont la variable de session est déjà déclarée, du prévenir l'éventualité d'une double ouverture de la session, ce qui engendrerait une erreur.

En utilisant cette instruction,

```
<?php if (session_status() == PHP_SESSION_NONE) session_start();?>
```

, nous vérifions d'abord si la variable de session existe déjà, dans le cas contraire et seulement dans ce cas là, la session est ouverte.

Un exemple de code de gestion des session vérifiant l'authentification de l'utilisateur est visible si dessous.

Il vérifie que la variable *auth* a bien été remplie à la connexion et sinon redirige, vers la page de connexion. Si l'utilisateur qui se connecte est de type admin, il est redirigé vers la page d'administration.

```
| if (!isset($_SESSION['auth'])) {  
| $_SESSION['flash']['danger'] = "Vous n'avez pas le droit d'accéder à cette  
|     page";  
| header('Location:connexion.php');  
| } if (isset($_SESSION['auth']) && ($_SESSION['auth']->typeUtilisateur == 'admin')) {  
|     header('Location:gestion.php');  
| } else {  
|     $_SESSION['flash']['success'] = "Bienvenue". $_SESSION['auth']->nom . " ". $_\br/>|         _SESSION['auth']->prenom . " !";  
|     //debug($_SESSION);  
| }  
?  
|>
```

6 Gestion des réservations

6.1 Plan de salle

7 Versionnement

7.1 Logiciels de gestion du versionnement

7.1.1 GIT

Dans le cadre du développement logiciel, la plupart des entreprises utilisent un logiciel de versionnement. Nous avons nous aussi souhaiter versionner notre travail. Nous avons choisi GIT car il est de plus en plus employé dans le monde professionnel et qu'il est open-source. Cela nous a permis d'acquérir des compétences de gestion de projet et de développement susceptibles d'intéresser un employeur.

De plus, en combinaison avec Github, cela nous permet d'avoir un version sauvegardée du code en ligne, palliant ainsi à d'éventuels soucis matériels.

Toutefois seul l'un d'entre nous avait déjà une expérience de ce logiciel de versionnement. L'apprentissage n'a pas était aisé pour tous mais nous avons tous progressé dans notre connaissance de ce logiciel. Afin de faciliter cet apprentissage, nous avons eu recours au logiciel **GitKraken** qui propose une interface graphique à git.

Au moment de la rédaction de ces lignes, 206 commits avaient été enregistrés, sur 12 branches différentes.

7.1.2 GitKraken

Pour nous aider à nous retrouver dans les branches et les dépôts, nous avons eu recours au logiciel **GitKraken** édité par la société Axosoft².

2. <https://www.gitkraken.com/>

Elaboration d'un site web accessible Opéra national de Paris

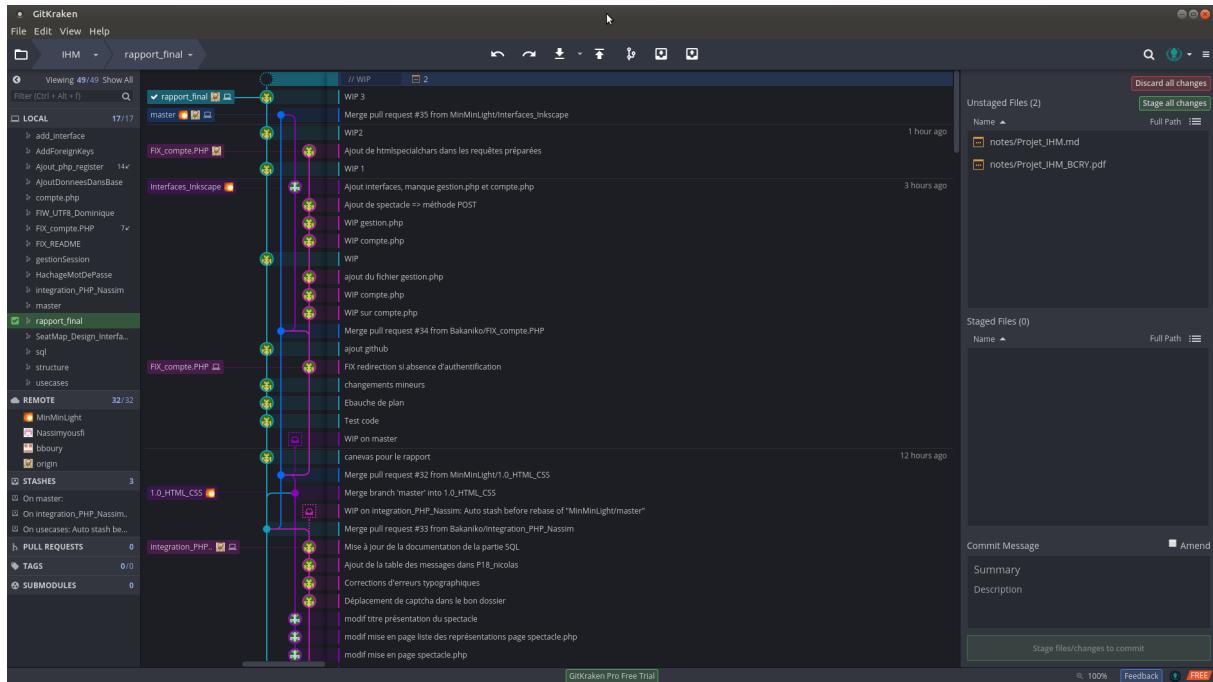


Figure 7. Interface de GitKraken

Outre une esthétique très travaillée, il permet de réaliser facilement plusieurs opérations courantes sans avoir à recourir à la ligne de commande (COMMIT, PULL, PUSH, ADD REMOTE). Il permet aussi de visualiser les branches des autres collaborateurs et leur avancement.

7.2 Hébergement

L'hébergement s'est fait principalement sur nos machines personnelles, puis le site a été déployé sur Handiman. Ainsi, chaque *Pull Request* était testée dans un serveur local avant d'être fusionnée à la branche *master*. Enfin, la branche *master* était téléchargée sur Handiman.

La version déployée est visible à cette adresse : <http://handiman.univ-paris8.fr/~nicolas/>

La base de code est hébergée et visible sur github : <https://github.com/MinMinLight/IHM>.

Nous aurions pu utiliser d'autres plate-formes tel que Bitbucket ou Gitlab / Framagit ; mais GitKraken s'intègre mieux avec Github et Bitbucket. De plus, beaucoup de projets libres ont recours à Github, et c'était l'occasion de se familiariser avec son interface.

La création et le lancement d'un projet nous ont semblé plus complexe sur des plate-formes telles que Gitlab / Framagit. Nous avons donc écarté ces solutions.

Avec du recul, il aurait probablement été plus intéressant d'héberger le code sur Bitbucket. En effet, c'est gratuit jusqu'à 5 collaborateurs (nous étions 4) et il est possible de rendre le projet privé. En effet, Github propose un hébergement gratuit pour les projets open-source, . Tout ce que nous publions dessus est donc librement accessible sur Internet. Ce qui fait que nous avons dû faire attention à ne pas publier nos codes d'accès ou nos mails personnels.

7.3 Avantages

L'avantage principal a été le fait d'avoir une distribution distribuée du code. Cela a aussi permis d'utiliser le code le plus à jour possible pour démarrer une nouvelle branche.

Github permet une communication entre les participants à travers les *Pull Request*, mais nous avons peu exploiter cette possibilité.

7.4 Inconvénients

Le fait du publier quelque chose sur github, de façon gratuite, fait qu'il a fallu faire attention à la sécurité. Afin de ne pas publier de mots de passe sur internet.

7.5 Problèmes rencontrés

- L'envoi de mail, fonctionne sur l'ordinateur de Nassim, ne fonctionne pas sur l'ordinateur de Nicolas ou sur Handiman. La *SendMail* n'est pas installé sur le serveur.
- un certain nombre de fichiers n'ont pas pu être chargés sur Handiman, des erreurs se produisaient. Par la suite, il n'était plus possible de supprimer ou remplacer les fichiers vides ou incomplets présents sur Handiman. Nous avons donc utiliser un serveur local pour terminer le projet.

8 Evolutions possibles / points non traités

8.1 Fonctionnalités / Interface du site

- gestion des achats (panier), paiement de la commande ;
- renvoi d'un mot de passe temporaire en cas d'oubli ;
- charger directement l'image depuis l'interface de gestion ;
- si un administrateur est connecté, remplacer panier par gestion qui renvoie vers la page *gestion.php* ;
- ajout d'un fil d'ariane en dessous du menu pour aider l'utilisateur à naviguer dans le site ;
-

8.2 Page gestion.php

- vérifier les saisies administrateur : format de date et horaires
- pouvoir dupliquer les données d'une représentation ou d'un spectacle pour gagner du temps de saisie
- insérer un message demandant la confirmation de l'action de suppression d'un spectacle ou d'une représentation
- dans l'onglet suppression, trouver un moyen de filtrer les représentations : par spectacle, par salle, par mois, par année. Éventuellement en reprenant le code de la page *programmation.php*. Il faudra peut-être avoir recours à Javascript et JQuery/Ajax.

8.3 Page spectacle.php

- gérer le cas où l'idSpectacle n'existe pas dans la base,
- si toutes les dates du spectacles sont passées, afficher quand même les informations, plus représentations passées (sans bouton réservé)

8.4 Page reservation.php

Cette page a nécessité l'utilisation d'un formulaire php et d'une image svg, ainsi que du code mêlant javascript, PHP et SQL.

Toutefois, bien que fonctionnelle sur l'ordinateur de la personne qui l'a développé, nous avons eu des soucis à l'intégration. En effet, les serveurs locaux présentaient des disparités de logiciels et de versions. Cette page ayant nécessité un temps de développement long, il ne nous restait plus de temps pour régler les problèmes découverts à l'intégration.

Après le choix de la place par le biais du formulaire ou de l'image, les informations sont insérées dans la variable `$_SESSION["panier"]`

8.5 Page panier.php

Bien qu'existe, la page panier.php n'est pas fonctionnelle car elle dépend directement des .

9 Conclusion

Bien que n'étant pas complètement terminé la réalisation de ce projet a été enrichissante à plusieurs niveaux.

Nous avons pu en effet apprendre à développer tous les composants d'un site web, en interfaçant son squelette à une base de données grâce à PHP, afin de le rendre dynamique. Nous avons rencontré un certain nombre de problèmes et essayé de trouver les meilleures solutions pour y remédier (fournir une alternative, avoir recours à un autre langage, un autre outil).

Nous avons aussi appris à collaborer pour réaliser ce site, notamment durant les phases d'intégration des différentes parties de chacun. En effet, nous n'avons pas toujours eu les mêmes pratiques de code et l'harmonisation n'a pas toujours été heureuse. Nous avons aussi progresser dans notre maîtrise de l'outil de versionnement Git.

1 Annexes

1.1 Usecases

1.1.1 Jean-Marc, prospect

Jean-Marc n'est pas encore client de l'Opéra.

Mais il souhaite assister à une représentation de Rigoletto dont il a vu les affiches dans la rue.

Le soir en rentrant du travail, il accède au site de l'opéra. La page d'accueil ne présente pas Rigoletto, il clique donc sur Programmation 2017 et est redirigé vers la page de programmation.

Il voit dans la liste des représentations en dessous du formulaire qu'une représentation de Rigoletto est prévue pour décembre. Il clique donc sur le lien qui le renvoie vers la description du spectacle.

Il peut admirer l'affiche, apprendre le nom du metteur en scène et des artistes. Un menu déroulant lui permet de choisir une représentation. Il s'aperçoit qu'il n'avait pas vu celle d'octobre auparavant. Il choisit celle-ci, plus proche dans le temps.

Puis il clique sur le bouton "Réserver", et est redirigé vers la page contenant le plan de salle. Il voit que les sièges les plus chers sont devant mais en dehors de ces moyens. Il sélectionne donc un siège libre dans le parterre.

Il constate que sa sélection apparaît dans le panier.

Ne souhaitant pas faire d'autres réservations, il se rend sur son panier. Sa place y est bien présente.

Il clique donc sur le bouton "Paiement". Le site lui demande alors de créer un compte.

Il saisie ses identifiants habituels ainsi que son adresse mail et valide. Il se rend sur sa boîte mail pour attendre le mail de confirmation. L'ayant reçu, il l'ouvre et clique sur le lien.

Il est alors redirigé vers son panier. Sa commande est toujours là ; il poursuit donc vers le paiement et cette fois, y accède directement.

Après avoir réglé son achat, il est redirigé vers son compte. Où il est invité à saisir le reste de ces coordonnées. Dans la partie intitulée "Mes réservations", il constate que sa réservation est présente.

Un lien lui permet de visualiser et d'imprimer son billet. Après l'avoir imprimé, il se déconnecte et quitte le site.

1.1.2 Marcelle et Jacques, clients réguliers

Marcelle et Jacques sont des clients réguliers de l'Opéra. Tous les deux pour Noël et leur anniversaire de mariage, ils réservent un billet pour un opéra.

En vue de leur anniversaire de mariage dans quelques mois, en mai, ils se rendent sur le site de l'Opéra national de Paris.

Ils cliquent sur le lien "Programmation 2017" qui les redirige vers la programmation de cette année. Habitues du site, ils utilisent le formulaire et choisissent un spectacle de type Opéra. Jacques n'aime pas les ballets, et le mois de mai à l'aide des listes déroulantes. Puis ils cliquent sur "Rechercher".

Le site leur propose alors, "Les contes d'Hoffmann" le 5 mai et "Aida" le 13 mai.

Ils cliquent sur les "Contes d'Hoffmann" et sont redirigés vers la page du spectacle. Après l'avoir consulté quelques instants, ils reviennent en arrière pour consulter la page d'"Aida". Préférant Verdi, ils choisissent cette représentation en cliquant sur le bouton "Réserver" placé à côté de la date qui les intéresse.

Habitues de l'Opéra national, ils choisissent deux places bien placées dans la fosse d'orchestre. Après avoir ajouté les deux places dans leur panier, ils cliquent sur le bouton "Étape suivante". Ils sont alors redirigés vers le panier qu'ils valident en cliquant sur "Paiement".

Il leur est alors demandé de se connecter. Après avoir saisi leurs identifiants, ils sont renvoyés vers la page de paiement. Après avoir réglé la transaction, ils sont redirigés vers leur compte, où ils peuvent imprimer leurs billets.

1.1.3 Philippe, administrateur

Philippe est un des administrateurs de l'Opéra National de Paris. C'est lui qui ajoute de nouveaux spectacles et planifie les représentations en accord avec le Directeur. Il les crée et les modifie si besoin. Il peut supprimer les objets inutiles ou erronés.

Ce matin, par exemple, il a remarqué que l'affiche des contes d'Hoffman n'est pas la bonne. Il se connecte donc sur le site. Il apprécie le fait que son interface soit la même que celle des clients, ainsi il peut voir régulièrement si il y a des soucis.

Il se rend donc sur le site et se rend sur la page de connexion. Le site reconnaît que Philippe est un administrateur et le redirige automatiquement vers la page de gestion.

Sur cette page, Philippe a la possibilité d'ajouter, modifier ou supprimer un spectacle. Il peut aussi administrer de la même façon les salles, réservations et utilisateurs.

En cliquant sur "Gestion des spectacles", il accède à la page permettant la gestion de ces derniers. Sur le côté, il peut choisir dans la liste tous les spectacles présents dans la base. Il sélectionne "Les contes d'Hoffmann" et valide.

La fiche du spectacle apparaît. En regardant le champ contenant le nom de l'image, il s'aperçoit qu'elle porte une extension peu courante : ".webp". Il vérifie le fichier sur le serveur et s'aperçoit que l'extension est en ".jpg". Il change donc l'extension dans le formulaire et valide. Puis il retourne sur l'accueil et contrôle le changement sur la fiche du spectacle. L'affiche officielle est maintenant affichée.

1.2 Code python générant un jeu de test de réservations

```
#!/usr/bin/env python
# -*- coding: <utf-8> -*-

#####
# This script generates random reservations #
# from a few parameters. It delivers it in #
# a csv file and a SQL file                 #
#                                             #
#                                             #
# project url:                            #
# https://github.com/MinMinLight/IHM      #
#                                             #
# website url:                           #
# http://handiman.univ-paris8.fr/~nicolas/ #
#                                             #
# inspiration:                           #
# http://sametmax.com/lencoding-en-python-une-bonne-fois-pour-toute/ #
#####

'''
Parameters:
- csv file with performance ids
- csv file with user ids
- csv file with place ids

Output format:
- 1 line = 1 row
- csv: idUtilisateur;idRepresentation;idPlace
- sql : INSERT INTO proj_Reservation (idUtilisateur, idRepresentation,
    idPlace) VALUES (idUtilisateur, idRepresentation, idPlace);

encodage de la sortie :
une_chaine = une_chaine.encode('utf8')

'''

import random
import csv

resaCSV = open('reservations.csv', 'wa')
resaSQL = open('reservations.sql', 'wa')
'''

User id loading
'''

# empty array
usersId = []
file = open("proj_utilisateur.csv", "r")
userCsv = csv.reader(file)
for row in userCsv:
    # print(row[1])
    usersId.append(row[0])

file.close()
# output check
print(usersId)

'''

Amount of place
'''
```

```
# empty array
places = []
file = open("proj_Place.csv", "r")
salleCsv = csv.reader(file)
for row in salleCsv:
    # print(row[1])
    places.append(row[0])

file.close()
# output check
print(places)

,
,
Number of performance
,

# empty array
performances = []
file = open("proj_Representation.csv", "r")
representationCsv = csv.reader(file)
for row in representationCsv:
    # print(row[1])
    performances.append(row[0])

file.close()
# output check
print(performances)

# Main algorithme
nbPlaces = len(places)

for performance in performances:
    i = 0
    pointer = 1

    while i < nbPlaces:
        # from place 1 to nbPlace + 1

        # if i is on the pointer
        if i == pointer:
            # then flip a coin
            isReserve = random.randint(0, 1)

            # if heads
            if isReserve == 1:

                # generates a number of reservations between 1 and 4
                nbReservation = random.randint(1, 4)
                # print u"reservation de " + str(nbReservation) + u" places"

                # pick a random user id
                userId = random.choice(usersId)

                # to avoid nbPlace overflow
                if (pointer + nbReservation > nbPlaces):
                    nbReservation = nbPlaces - pointer
                for j in range(0, nbReservation):
                    # print places[i + j] + u" isReserved by " + str(userId)
                    # + " for performance " + performance # output test
                    # print places[i + j] + u" isReserved by " + str(userId)
                    # + " for performance " + performance # output test
                    # print str(userId) + "," + performance + "," + places[i]
```

```
+ j] # output test
resaCSV.write((str(userId) + "," + performance + "," +
    places[i + j] + "\n").encode('utf8')) # 
        output in csv file
resaSQL.write("INSERT INTO `proj_Reservation` (
    idUtilisateur, `idRepresentation`, `idPlace`) VALUES
    (" + 
        (str(userId) + "\\", \" + performance + "
        \\", \" +
    places[i + j] + "\");\n").encode('utf8'))
        # output in sql file
pointer = i + nbReservation # pointer moves forward

# if tails
else:
    # print u"siege " + str(i) + " libre"

pointer += 1 # nothing happens, the pointer move forward

i += 1 # i moves forward

resaCSV.close() # fermeture du fichier reservations.csv
resaSQL.close() # fermeture du fichier reservations.sql
```

1.3 Documentation de la base de données 1

Notes table par table

Toutes les tables commencent par le préfix "proj_"

1.3.1 proj_TypeUtilisateur

TypeUtilisateur Défini les types d'utilisateurs possibles :

- admin : administre le site et la base de données
- manager : peut gérer les spectacles et les réservations mais pas le site et la base
- user : client du site, peut créer un compte et faire des réservations

Un utilisateur du site doit obligatoirement avoir un TypeUtilisateur, seul un admin peut changer le type d'un utilisateur.

varchar 10

1.3.2 proj_Utilisateur

IDU Identifiant Unique utilisateur, **clé primaire**

type numérique auto-incrémenté

login Identifiant de l'utilisateur (pseudo, adresse mail)

Varchar 30, **UNIQUE**

Il ne doit pas y avoir 2 login identiques dans la base

passHash Mot de passe haché par la fonction password_hash() de PHP ; doit être utilisé directement par password_verify().

varchar 255 NOT NULL

nom Nom de l'utilisateur

Varchar 30

prenom Prenom de l'utilisateur

Varchar 30

adressePostale1 Première ligne de l'adresse postale

Varchar 50

adressePostale2 Deuxième ligne de l'adresse postale

Varchar 50

codePostal Code postal : traité comme string pour les client étrangers

Varchar 10

ville Ville de résidence

Varchar 15

adresseMail email de contact, *devra correspondre à un certain format*

unique

varchar 50

telephone telephone de contact, traité comme string pour les numéros étrangers, *devra correspondre à un certain format*
varchar 15

typeUtilisateur référence le type d'utilisateur
Par défaut, réglé sur *user*, devra être changé ultérieurement par un admin
Clé étrangère sur **proj_TypeUtilisateur.TypeUtilisateur**
varchar 10

1.3.3 proj_Spectacle

Référence tous les spectacles proposés (faits, en cours, à venir)

idSpectacle Identifiant unique auto-incrémenté, **clé primaire**
Int

nom Nom du spectacle
varchar 50
(unique?)

type Référence le type de spectacle : concert, pièce de théâtre, danse . . .
Suggestion : Les valeurs devraient venir d'une table externe
varchar 30

infos contient les informations du spectacle :

- nom du metteur en scène / chorégraphe
- noms des acteurs/musiciens
- description courte

suggestion : peut être divisé en plusieurs champs dans un vrai site
TEXT

1.3.4 proj_Salle

Une salle est composée de places et appartient à un lieu

idSalle Identifiant unique
Numérique INT auto-incrémenté **clé primaire**

nom nom de la salle (salle n°3 ; Théâtre Odéon ; etc)
varchar 30

adresse adresse postale
TEXT

type type de salle : salle de concert, théâtre, piste de cirque...
varchar 20

1.3.5 proj_Representation

Une représentation est identifiée par un spectacle (auteur, acteurs, etc), une salle, un horaire et une date. Un spectacle peut avoir plusieurs représentations dans la même salle.

idRepresentation Identifiant unique
Numérique INT auto-incrémenté **clé primaire**

idSalle proj_Salle.idSalle, **clé étrangère**

idSpectacle proj_Spectacle.idSpectacle, **clé étrangère**

date date de la représentation, séparée du temps pour être plus simple à dupliquer
DATE

HoraireDebut Début de la représentation
TIME

HoraireFin Fin de la représentation
TIME

1.3.6 proj_Categorie

Catégories disponibles pour une salle

idCategorie Identifiant unique
Numérique INT auto-incrémenté **clé primaire**

Categorie Nom de la categorie
varchar 20

idSalle Les salles ont certaines catégories mais pas toutes. Il faut séparer les catégories de chaque salle, car la même catégorie n'aura pas le même prix dans un théâtre que dans un opéra.
proj_Salle.idSalle, **clé étrangère**

1.3.7 proj_Place

idPlace Identifiant unique
Numérique INT auto-incrémenté **clé primaire**

idSalle Une place appartient toujours à une salle
proj_Salle.idSalle, **clé étrangère**

idCategorie proj_Categorie.idCategorie, **clé étrangère**

Handicap Détermine si la place est accessible ou non.
varchar (3)

Rang Rangée à laquelle la place appartient
varchar (5)

Numero Numéro de la place : concaténation Rang + idPlace
varchar(10)

1.3.8 proj_PrixPlace

Le prix est fonction du spectacle, peut importe la représentation, et de la catégorie. La catégorie permet de remonter à la salle.

idCategorie proj_Categorie.idCategorie, **clé étrangère ; clé primaire**

idSpectacle proj_Spectacle.idSpectacle, **clé étrangère ; clé primaire**

prix Prix en euros
Décimal

1.3.9 proj_Reservation

Une réservation prend en compte 1 et 1 seule place, pour une unique représentation, par un unique utilisateur /client.

Avec le couple idPlace et idRepresentation, il est possible de retrouver le prix

IDU proj_Utilisateur.IDU, **clé étrangère ; clé primaire**

idRepresentation proj_Representation.idRepresentation, **clé étrangère ; clé primaire**

idPlace proj_Place.idPlace, **clé étrangère ; clé primaire**

1.3.10 messages_contacts

Cette table est destinée à contenir les messages envoyés à travers la page infos.phpMyAdmin

id_message_contact Id unique du message, généré à l'insertion
type *SERIAL*

Nom_contact Nom de l'expéditeur
varchar 30

Prenom_contact Prenom de l'expéditeur
varchar 255

Mail_contact email de l'expéditeur
varchar 255

Text_message Texte du message
type *TEXT*

date_message_contact Date d'enregistrement du message
type *date*

1.3.11 Disponibilité des places

Pas besoin de créer une liste spéciale :

Il suffit de trier les places pour le spectacle concerné où il y a (ou pas) de réservations de faite.

1.3.12 Clés étrangères

```
proj_Utilisateur(typeUtilisateur) -> proj_TypeUtilisateur(typeUtilisateur)
proj_Representation(idSalle) -> proj_Salle(idSalle)
proj_Representation(idSpectacle) -> proj_Spectacle(idSpectacle)
proj_Categorie(idSalle) -> proj_Salle(idSalle)
proj_Place(idSalle) -> proj_Salle(idSalle)
proj_Place(idCategorie) -> proj_Categorie(idCategorie)
proj_PrixPlace(idCategorie) -> proj_Categorie(idCategorie)
proj_PrixPlace(idSpectacle) -> proj_Spectacle(idSpectacle)
proj_Reservation(IDU) -> proj_Utilisateur(IDU)
proj_Reservation(idRepresentation) -> proj_Representation(idRepresentation)
proj_Reservation(idPlace) -> proj_Place(idPlace)
```

Liste des figures

1	Site officiel de l'Opéra national de Paris	6
2	Première ébauche du design	8
3	Sur smartphone menu fermé	9
4	Sur smartphone menu ouvert	9
5	Design actuel	11
6	Modèle Conceptuel de Données	12
7	Interface de GitKraken	16

Liste des tables