PID – Projet Réservations

# Contexte

Dans le cadre de la formation en *Bachelier en Informatique de gestion*, d’une part, et en *BES de Web Developer*, d’autre part, le présent projet s’inscrit dans une double finalité, à savoir :

* se former au développement d’une application Web dynamique au moyen d’outils de développement RAD et en développant une méthodologie professionnelle dans un contexte de travail collaboratif (travail de groupe et support en classe) ;
* se préparer au Travail de Fin d’Études en explorant les besoins fonctionnels du TFE dans un contexte moins contraignant, et en développant les compétences adéquates et son autonomie.

# Gestion du projet

* <https://openclassrooms.com/courses/4192086-gerez-votre-projet-informatique-facilement>
* <https://openclassrooms.com/courses/4296701-gerez-un-projet-digital-avec-une-methodologie-en-cascade>
* <https://openclassrooms.com/courses/4507926-initiez-vous-a-la-gestion-de-projet-agile>

Itération 0 – Découverte du projet

# Introduction

## Présentation du projet

Le projet consiste à informatiser la gestion des réservations de spectacles d’une société de production. Celle-ci gère un catalogue reprenant des spectacles, leurs auteurs et leurs metteurs en scènes, les comédiens, ainsi que les lieux et les dates de représentations.

L’**internaute** pourra consulter librement le catalogue des spectacles affichant le lieu et les prochaines dates de représentation. Il pourra effectuer des recherches, des tris et des filtres à travers les pages du catalogue.

Le **membre** pourra réserver des places pour une représentation d’un spectacle, consulter la liste de ses réservations et modifier ses données de profil.

L’**administrateur** pourra gérer son catalogue à travers un back-office sécurisé. Par exemple, il pourra ajouter, modifier et supprimer un spectacle manuellement, importer/exporter des données au format CSV, mais aussi mettre à jour la liste des spectacles grâce aux nouveautés publiées par un **Web service tiers**.

À son tour, l’**application** devra produire d’une part son propre Web service (une API authentifiée avec système d’affiliation), d’autre part un flux RSS (par exemple, la liste des prochaines représentations).

## Cahier de charges techniques

* Développer une application Web en respectant les contraintes suivantes
  + utiliser la **programmation orientée objet** (langages *Java*, *PHP* ou *Python*),
  + exploiter un **framework** (*Spring Boot*, *Symfony*, *Laravel*, *CakePHP*, *Django*),
  + gérer les erreurs de programmation au moyen d’outils ou de **techniques de débogage** et y apporter une solution pertinente,
  + utiliser à bon escient la **documentation** disponible,
  + **optimisation** du code, du cache et des échanges avec la base de données,
  + gérer un **catalogue d’éléments** (produits, membres ou services) :
    - effectuer des recherches dans le catalogue,
    - paginer, trier et filtrer les résultats de la requête,
    - sélectionner des éléments du catalogue et effectuer des actions groupées,
    - sélectionner un élément du catalogue et en afficher le détail,
    - gérer la sécurisation et les droits d’accès aux contenus (droits d’accès par rôle administrateur, utilisateur public, utilisateur enregistré, gestionnaire, etc.),
    - afficher des contenus de manière différenciée (accessibilité, langue, sécurité, fonctionnalités, disponibilité de l’information, etc.), en fonction des profils utilisateurs,
    - optimisé par **programmation asynchrone** (AJAX, JSON, XML,…),
    - exploiter des fichiers de données structurées (XML, CSV, texte,..) en lecture/écriture (importer/exporter le catalogue),
    - intégrer des **services internes et tiers** (RESTful Web services),
  + déployer un **back-office d’administration** (interface administrateur sécurisé) :
    - gérer les contenus dynamiques,
    - modifier ses données de profil,
    - configurer une interface utilisateur (couleur, affichage,…),
    - interagir avec un système de gestion de bases de données (récupérer, ajouter, modifier, supprimer des enregistrements, etc.) ;
* Veiller à la sécurité
  + utiliser des outils spécifiques de protection et d’identification,
  + protéger l’application contre les injections SQL, attaques XSS, vols de session, détournement de cookies, etc.,
  + définir la réécriture d’url,
  + configurer des paramétrages et restrictions d’accès au serveur.

## Données initiales

Suite à une précédente tentative d’informatisation inaboutie, la société de production dispose déjà de l’export des données de l’ancienne application sous forme de fichier au format XML. Partant de ce fichier, elle souhaite que vous conceviez une base de données relationnelle.

### Extrait du fichier XML

<?xml version="1.0" encoding="utf-8"?>

<reservations>

<lieu id="delvaux">

<designation>Espace Delvaux / La Vénerie</designation>

<adresse>

<rue>3 rue Gratès</rue>

<localite>1170 Watermael-Boitsfort</localite>

</adresse>

<website></website>

<tel></tel>

</lieu>

…

<spectacle id="ayiti" reservable="1">

<title>Ayiti</title>

<img src="/wrapped/imgs/ayiti.jpg" />

<auteur>Daniel Marcelin et Philippe Laurent</auteur>

<scenographie>Daniel Marcelin et Philippe Laurent</scenographie>

<distribution>Daniel Marcelin</distribution>

<lieu ref="delvaux" />

<representations>

<representation complet="1">

<date>12/10/2012</date>

<heure>13:30</heure>

<lieu ref="delvaux" />

</representation>

<representation>

<date>12/10/2012</date>

<heure>20:30</heure>

<lieu ref="dexia" />

</representation>

</representations>

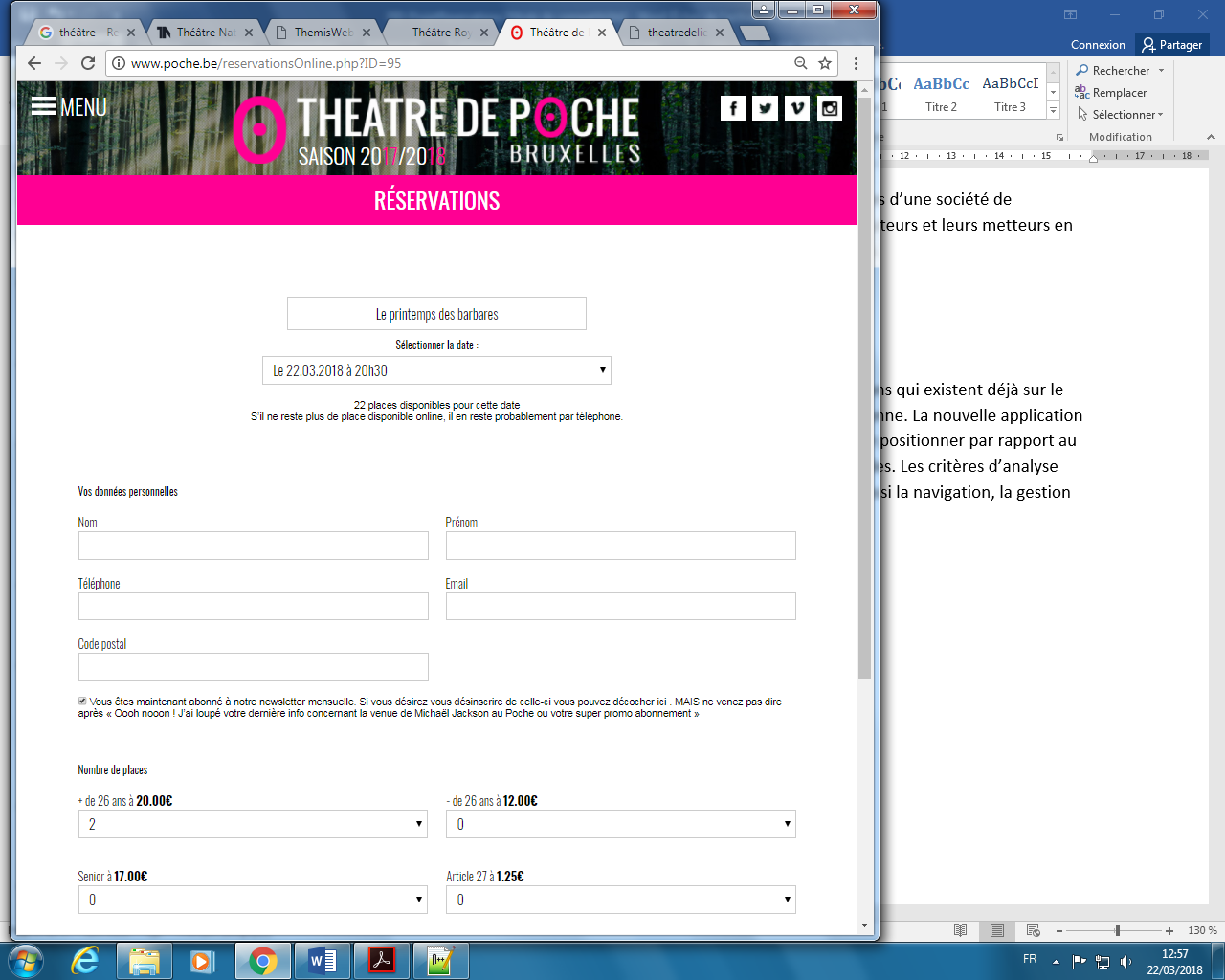
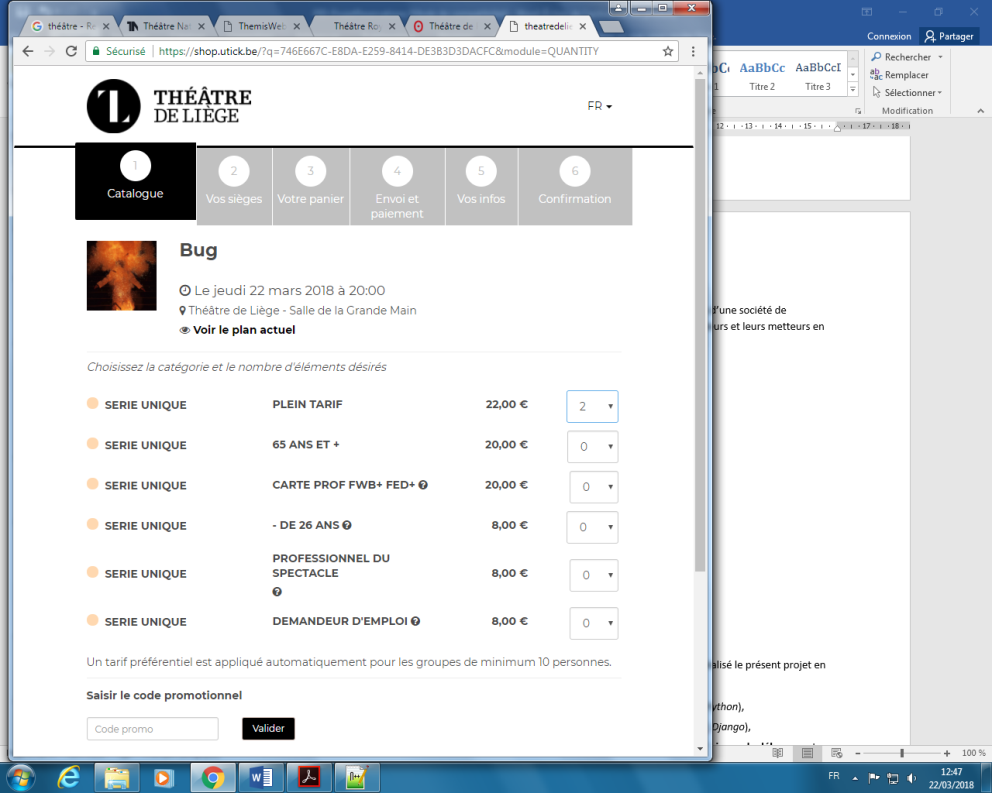
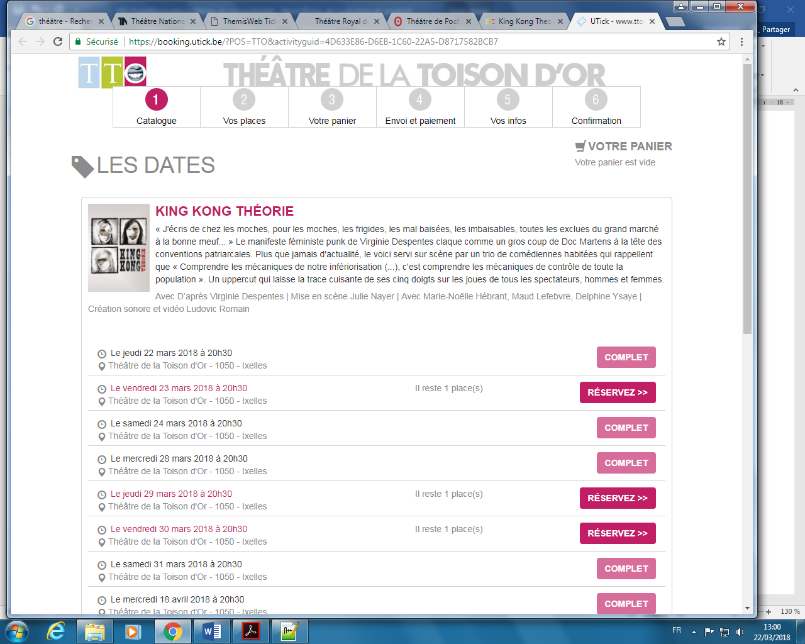
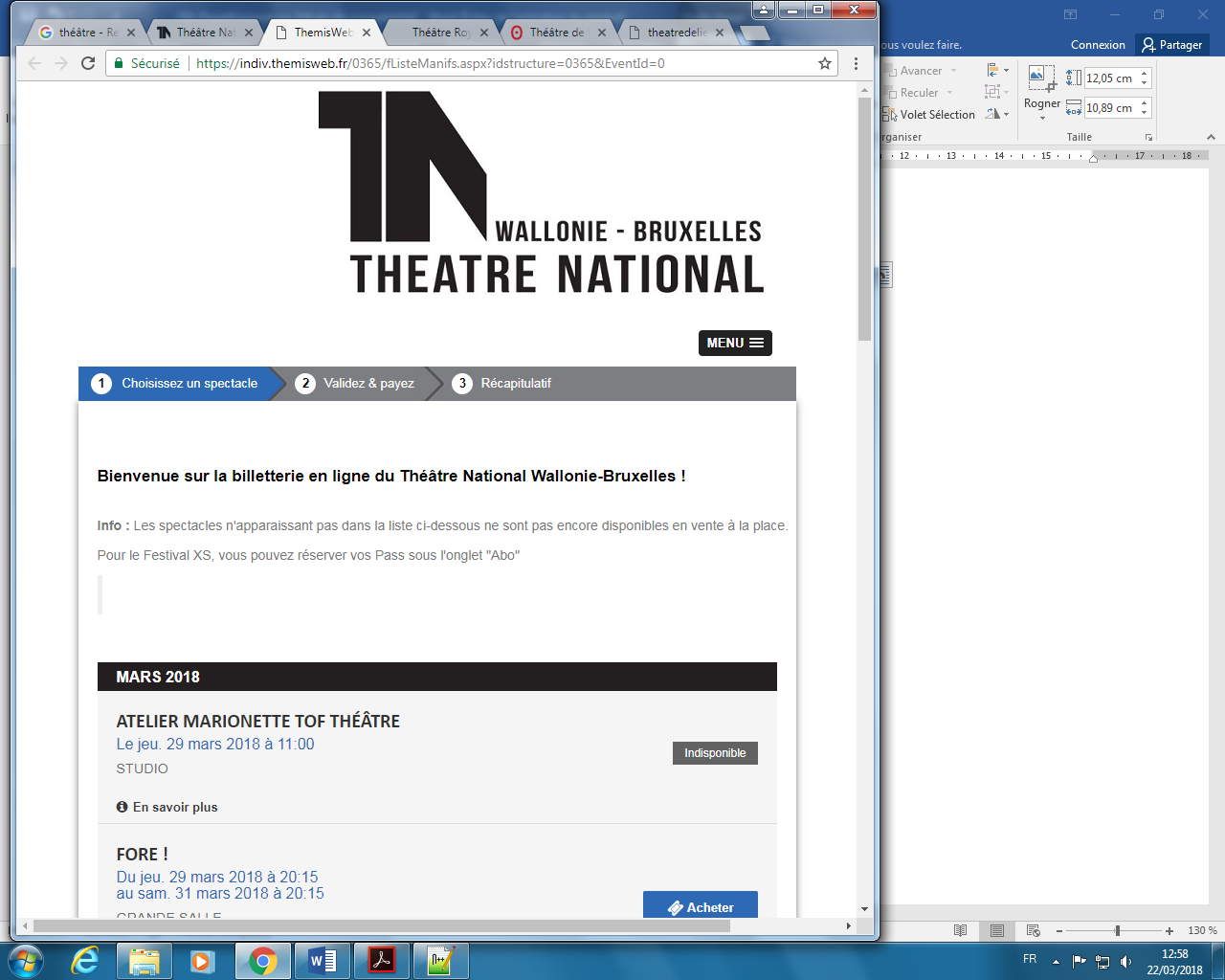
</spectacle>

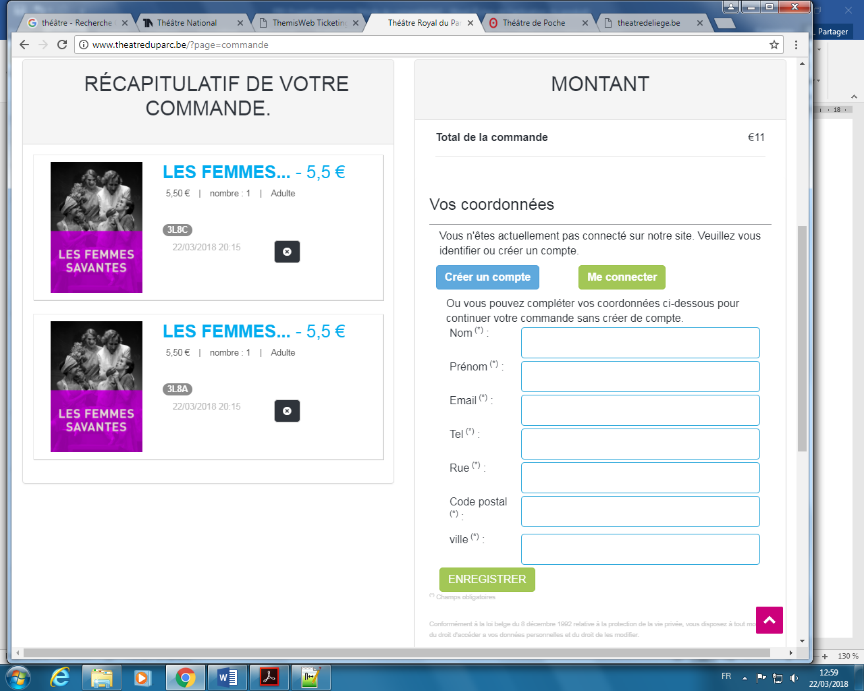
…

</reservations>

# Étude comparative

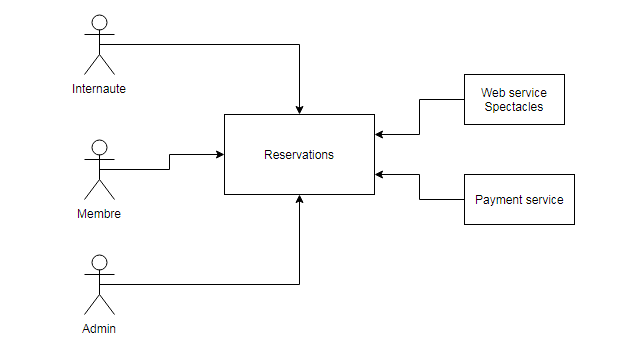
Une analyse de la concurrence ou une étude comparative des applications qui existent déjà sur le marché permet de comprendre comment ce genre d’application fonctionne. La nouvelle application pourra bénéficier de l’analyse de leurs avantages et inconvénients, et se positionner par rapport au marché, quitte à proposer des améliorations ou des innovations originales. Les critères d’analyse peuvent porter sur le processus de la fonctionnalité elle-même, mais aussi la navigation, la gestion des erreurs, l’accessibilité, l’ergonomie, les technologies utilisées, etc.



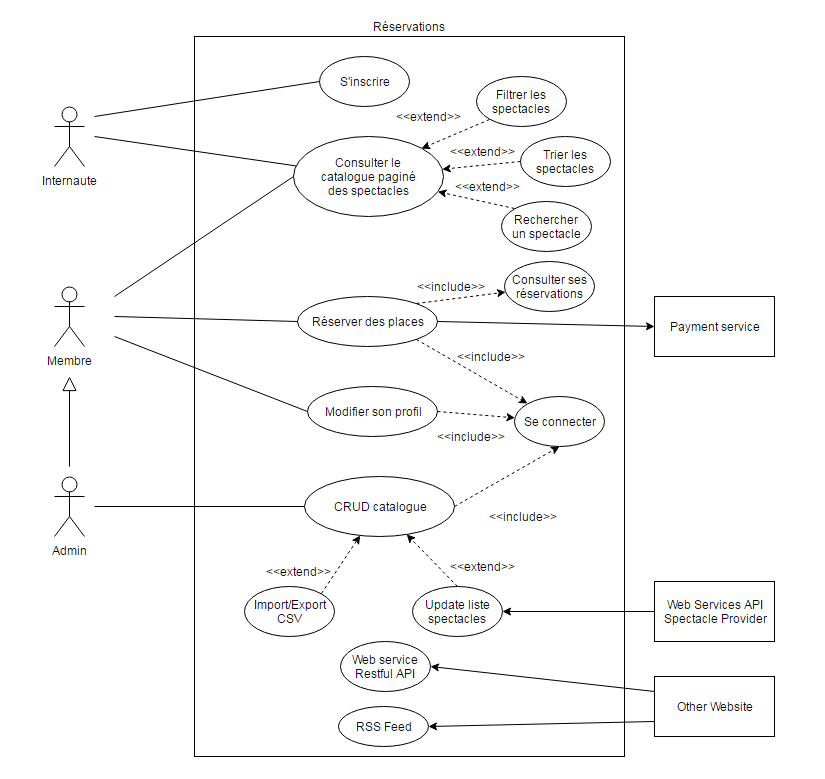


# Analyse[[1]](#footnote-1)

## Diagramme de contexte



## Cas d’utilisation



# Base de données



# Contraintes et règle métier

L’étudiant devra justifier à travers la rédaction d’un **rapport écrit** qu’il a réalisé le présent projet en veillant à respecter les contraintes suivantes :

* utiliser la **programmation orientée objet** (langages *Java*, *PHP* ou *Python*),
* exploiter un **framework backend** (*Spring Boot*, *Symfony*, *Laravel*, *CakePHP*, *Django*),
* exploiter un **framework** ou une **librairie frontend** (*Angular*, *VueJS* ou *jQuery*),
* intégrer des **services internes et tiers** (RESTful Web services),
* gérer les erreurs de programmation au moyen d’**outils ou de** **techniques de débogage** et y apporter une solution pertinente,
* utiliser à bon escient la **documentation** disponible,
* **optimiser** le code, le cache et les échanges avec la base de données,
* Veiller à la **sécurité** des données et du système en général.

# Modalités

# Planning

L’itération 0 doit être effectuée en classe. Elle ne fait pas l’objet d’une remise.

PID – Projet Réservations

Itération 1 – Installation du framework

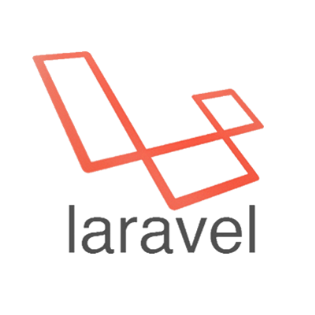
# Introduction

L’objectif de cette première itération consiste à effectuer les tâches suivantes :

1. Installer et configurer le *framework* ;
2. Utiliser un gestionnaire de dépendances pour organiser les librairies et composantes ;
3. Utiliser et configurer un environnement de développement intégré ;
4. Utiliser un logiciel de gestion des versions et configurer une solution d’hébergement ;
5. Utiliser un logiciel de gestion de projet et définir l’organigramme des tâches.

# Modalités

Vous devez implémenter l’application en utilisant au choix un des frameworks côté serveur basé sur le langage PHP, Java ou Python : *Symfony*, *Laravel*, *CakePHP*, *Spring Boot* ou *Django*. Côté client, nous nous limiterons à JavaScript et la librairie *jQuery*. La partie statique HTML/CSS sera facilitée par le framework *Twitter Bootstrap*.



# Planning

L’itération 1 doit être effectuée en classe. Elle doit faire l’objet d’une remise sur le dépôt de versioning en ligne.

PID – Projet Réservations

Itération 2 – Prise en main du framework

# Introduction

L’objectif de cette deuxième itération consiste à suivre un tutoriel simple qui permet de découvrir le workflow d’un développement avec framework. Il s’agit d’une mini-phase de formation.

# Modalités

Vous devez suivre un tutoriel mettant en œuvre les activités suivantes :

1. Installation et configuration du framework,
2. Mapping relationnel simple d’une entité,
3. Implémentation d’un template simple de type liste des données,
4. Implémentation du contrôleur correspond et des actions CRUD de base (index, show, save, delete, update, create).

# Références

Voici une liste de tutoriels classés par langage :

* PHP
  + Symfony
    - <https://symfony.com/blog/introducing-the-symfony-demo-application> (Demo Application)
  + Laravel
    - <https://laracasts.com/series/laravel-from-scratch-2017>   
      (Laracasts « Laravel from scratch »)
    - Spark
      * <https://spark.laravel.com/docs/5.0/quickstart>
  + CakePhp
    - <https://book.cakephp.org/3.0/fr/tutorials-and-examples.html> (CMS)
* Java
  + SpringBoot
    - <https://spring.io/guides/gs/spring-boot/>
* Python
  + Django
    - <https://docs.djangoproject.com/fr/2.0/#index-first-steps> (prise en main)
    - <https://docs.djangoproject.com/fr/2.0/intro/> (tutoriel complet)

# Planning

L’itération 2 doit être effectuée en classe. Elle ne doit pas faire l’objet d’une remise sur le dépôt de versioning en ligne.

PID – Projet Réservations

Itération 3 – Mapping relationnel

# Introduction

L’objectif de cette troisième itération consiste à ajouter les fonctionnalités suivantes :

1. Réaliser le mapping Objet-Relationnel d’une table sans clé étrangère de la base de données conformément au schéma de base de données et à la syntaxe de l’ORM proposé par le framework ;
   1. Générer et personnaliser l’entité ;
   2. Générer et personnaliser les jeux de données (fixtures/seeder).
2. Implémenter au moins un template de type liste de données en utilisant la syntaxe du moteur de template proposé par le framework ;
3. Implémenter un contrôleur chargé d’extraire et d’afficher les données d’une table dans le template de type liste ;
4. Définir une route permettant l’affichage la page Web qui affiche la liste ;
5. Terminer le mapping Objet-Relationnel de la base de données en terminant par les tables possédant des clés étrangères.

# Analyse

## Diagramme de classes participantes

## Règles métier

* Interopérabilité
* Risques
* Sécurité

## Dictionnaire de données

Artists

|  |  |  |  |
| --- | --- | --- | --- |
| **NOM** | **TYPE (long)** | **DESCRIPTION** | **CONTRAINTES /RÈGLE DE CALCUL** |
| id | Entier long | Id de l’artiste | Clé primaire |
| firstname | Texte (60) | Prénom de l’artiste |  |
| lastname | Texte (60) | Nom de l’artiste |  |

Types

|  |  |  |  |
| --- | --- | --- | --- |
| **NOM** | **TYPE (long)** | **DESCRIPTION** | **CONTRAINTES /RÈGLE DE CALCUL** |
| id | Entier long | Id du type | Clé primaire |
| type | Texte (60) | Fonction de l’artiste (métier) |  |

Artist\_type

|  |  |  |  |
| --- | --- | --- | --- |
| **NOM** | **TYPE (long)** | **DESCRIPTION** | **CONTRAINTES /RÈGLE DE CALCUL** |
| id | Entier long | Id de l’artiste | Clé primaire |
| artist\_id | Entier long | Id de l’artiste | Clé étrangère |
| type\_id | Entier long | Id du type d’artiste | Clé étrangère |

Show

|  |  |  |  |
| --- | --- | --- | --- |
| **NOM** | **TYPE (long)** | **DESCRIPTION** | **CONTRAINTES /RÈGLE DE CALCUL** |
| id | Entier long | Id de l’artiste | Clé primaire |
| location\_id | Entier long | Id du lieu (locations) | Clé étrangère, Null autorisé |
| slug | Texte (60) | Label court d’identification | Titre slugifié automatiquement, Unique |
| title | Texte (255) | Titre du spectacle |  |
| poster\_url | Texte (255) | URL de l’affiche | Null autorisé |
| bookable | Boolean | Réservable ou non |  |
| price | Double | Prix du spectacle |  |
| description | Texte long | Description du spectacle |  |
| created\_at | DATE | Date de création | Null autorisé |

Localities

|  |  |  |  |
| --- | --- | --- | --- |
| **NOM** | **TYPE (long)** | **DESCRIPTION** | **CONTRAINTES /RÈGLE DE CALCUL** |
| id | Entier long | Id de l’artiste | Clé primaire |
| postal\_code | Texte (6) | Code postal | Unique |
| locality | Texte (60) | Désignation officielle de la localité | Unique |

Locations

|  |  |  |  |
| --- | --- | --- | --- |
| **NOM** | **TYPE (long)** | **DESCRIPTION** | **CONTRAINTES /RÈGLE DE CALCUL** |
| id | Entier long | Id de l’artiste | Clé primaire |
| locality\_id | Entier long | Id de la localité (localities) | Clé étrangère, Null autorisé |
| slug | Texte (60) | Label court d’identification | Titre slugifié automatiquement, Unique |
| designation | Texte (60) | Désignation officielle du lieu |  |
| address | Texte (255) | Addresse du lieu | Null autorisé |
| website | Texte (255) | Site Web du lieu | Null autorisé |
| phone | Texte (30) | Numéro de téléphone du lieu | Null autorisé |

Collaborations (artist\_type\_show)

|  |  |  |  |
| --- | --- | --- | --- |
| **NOM** | **TYPE (long)** | **DESCRIPTION** | **CONTRAINTES /RÈGLE DE CALCUL** |
| id | Entier long | Id de l’artiste | Clé primaire |
| artist\_type\_id | Entier long | Id de l’artiste associé à un type (fonction) | Clé étrangère |
| show\_id | Entier long | Id du spectacle | Clé étrangère |

TODO : Ajouter les autres dictionnaires de données

# Contraintes et règle métier

# Modalités

L’itération 3 doit faire l’objet d’une remise sur le dépôt de versioning en ligne.

PID – Projet Réservations

Itération 4 – Prototypage

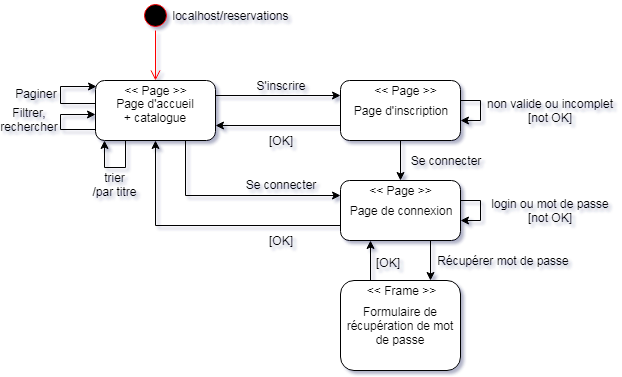
Implémenter les maquettes HTML au moyen de templates et d’un framework HTML/CSS (Bootstrap, Foundation,…).

# Analyse

## Diagramme de navigation

Les diagrammes de navigation, issus des use cases, nous permettent d’identifier les écrans (interfaces graphiques, pages web) dont l’application a besoin.

### Navigation *S’inscrire* + *Se connecter*



Mais il faut également analyser les use cases en identifiant dans les scénarios les composants visuels (titres, texte et images dynamiques, formulaires) et interactifs (boutons, hyperliens) nécessaires.

## Diagrammes de cas d’utilisation

### Use case *S’inscrire*

#### Scénario nominal

|  |  |
| --- | --- |
| 1. Un internaute clique sur le lien/bouton S’inscrire présent dans l’en-tête de toutes les pages. | 1. Le système affiche le formulaire d’inscription. |
| 1. L’internaute encode :    1. un login,    2. un mot de passe (min. 6 caractères, au moins une majuscule et un caractère spécial),    3. la confirmation du mot de passe,    4. son prénom,    5. son nom,    6. son email,    7. sa langue (choix dans une liste). | 1. De façon asynchrone, le système vérifie :    1. La disponibilité du login,    2. La validité du mot de passe,    3. La correspondance des mots de passe,    4. La disponibilité de l’email.   Un message de notification s’affiche pour chaque cas d’erreur et le bouton pour s’inscrire reste désactivé.  Lorsque tous les champs sont validés, le bouton *S’inscrire* est activé. |
| 1. L’internaute clique sur le bouton *S’inscrire* après avoir rempli tous les champs de façon adéquate. | 1. Le système valide les données reçues côté serveur. 2. Le système enregistre les données dans la base de données (table *users*) :    1. Le mot de passe est hashé,    2. Le rôle *member* est assigné par défaut. 3. Le système envoie un email de confirmation à l’internaute. 4. Le système redirige l’internaute vers la page de connexion (login/mot de passe) avec un message de confirmation l’invitant à se connecter. |

#### Scénario alternatif

|  |  |
| --- | --- |
| 1. L’internaute clique sur le bouton *S’inscrire* après avoir rempli tous les champs mais les données n’ont pas pu être validées côté client ou ont été altérées entretemps (piratage). | 1. Le système invalide les données reçues côté serveur. 2. Le système redirige l’internaute vers le formulaire d’inscription avec un message d’erreur reprenant les validations qui ont échouées. |

Il ressort du diagramme de navigation et de ce scénario le besoin de créer les deux templates et le fragment de page suivants : un formulaire d’inscription, un formulaire de connexion et un menu d’authentification.

Vous pouvez utiliser des outils de développement rapide d’application (RAD) tels que les nombreux framework HTML/CSS/JS (*Bootstrap* ou *Foundation*) disponibles sur le Web pour personnaliser des templates et y intégrer les composants graphiques nécessaires, ainsi que des librairies afin de faciliter la programmation front-end (*jQuery*).

* <https://jquery.com/download/>
* <https://getbootstrap.com/docs/4.1/examples/>
* <https://bootsnipp.com> (page *snippets*, *forms* ou *builder*)

### Use case *Consulter le catalogue paginé des spectacles*

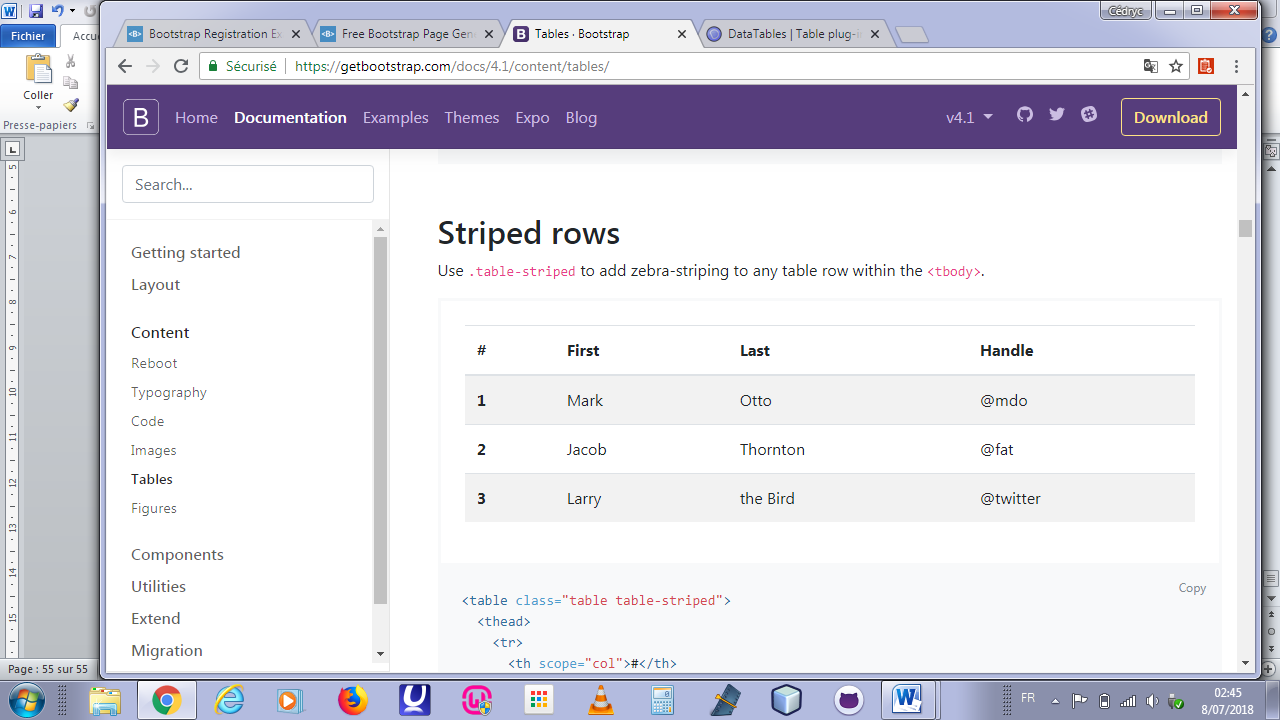
#### Scénario nominal

|  |  |
| --- | --- |
| 1. Un internaute ou un membre clique sur le lien *Catalogue* présent dans le menu dans l’en-tête de toutes les pages. | 1. Le système affiche le catalogue présentant la liste des spectacles triés par titre et paginés par 10. |
| 1. L’internaute clique sur un des liens du menu de pagination pour changer de page. | 1. Le système affiche les spectacles suivants correspondants à la page demandée. |
| 1. L’internaute clique sur un des liens de tri (titre, lieu, réservable, prix). | 1. Le système affiche la première page des spectacles triés selon le critère demandé. |
| 1. L’internaute sélectionne un critère parmi les listes déroulantes des formulaires de filtre (lieu, réservable). | 1. Le système affiche la première page des spectacles filtrés selon le critère demandé. |
| 1. L’internaute entre un mot-clé dans le formulaire de recherche. | 1. Le système affiche les spectacles correspondants au critère entré. |
| 1. L’internaute clique sur un des liens du menu de pagination pour changer de page. | 1. Le système affiche les spectacles suivants ou précédents correspondants à la page demandée en conservant les critères de tri, de filtre ou de recherche. |

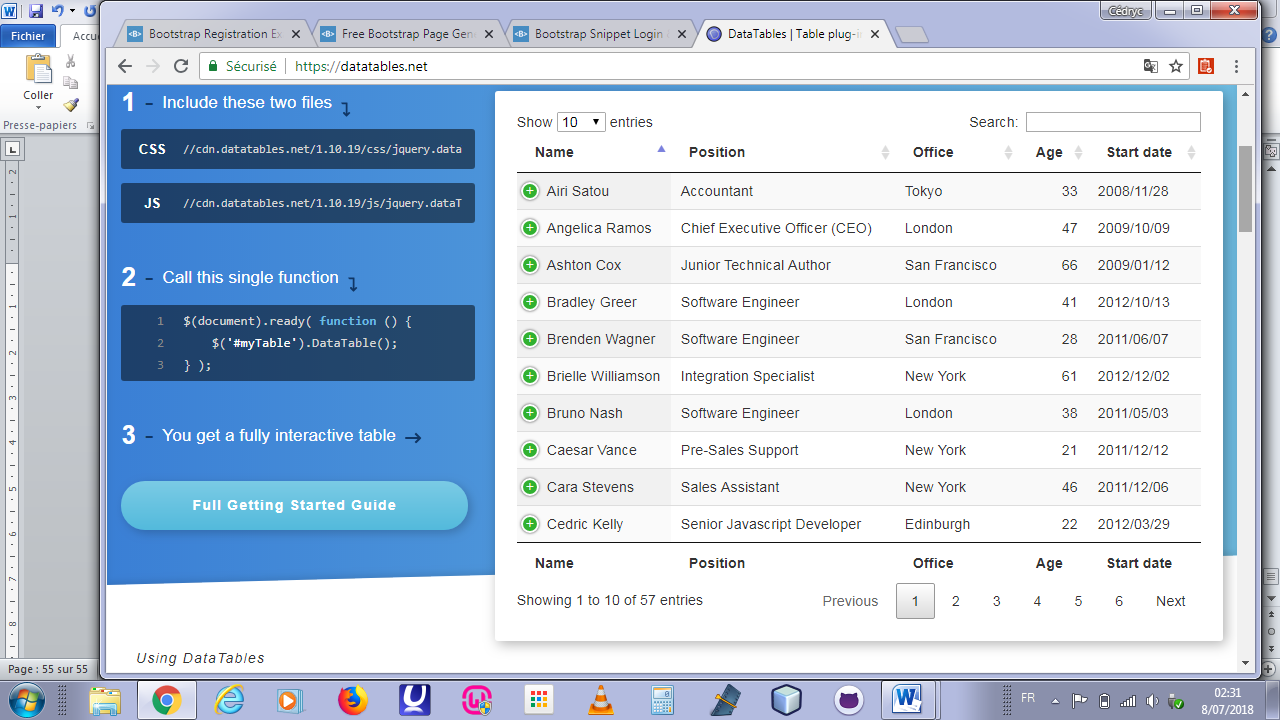
Il ressort de ce scénario le besoin de créer un template avec un tableau présentant la liste des spectacles.

Nous pouvons chercher ce composant graphique parmi les ceux de *Bootstrap* ou utiliser la librairie *Datatables* qui offre des fonctionnalités supplémentaires (tri, filtre, recherche et pagination) :

* <https://getbootstrap.com/docs/4.1/content/tables/>



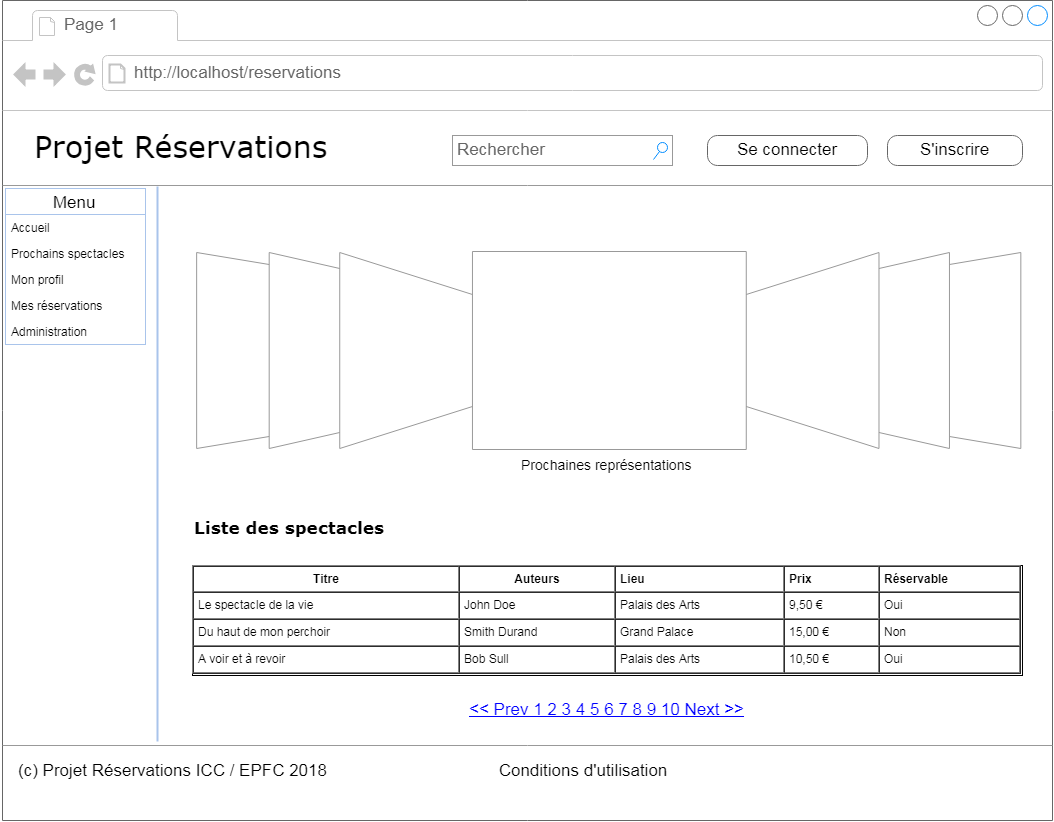
* <https://datatables.net/>



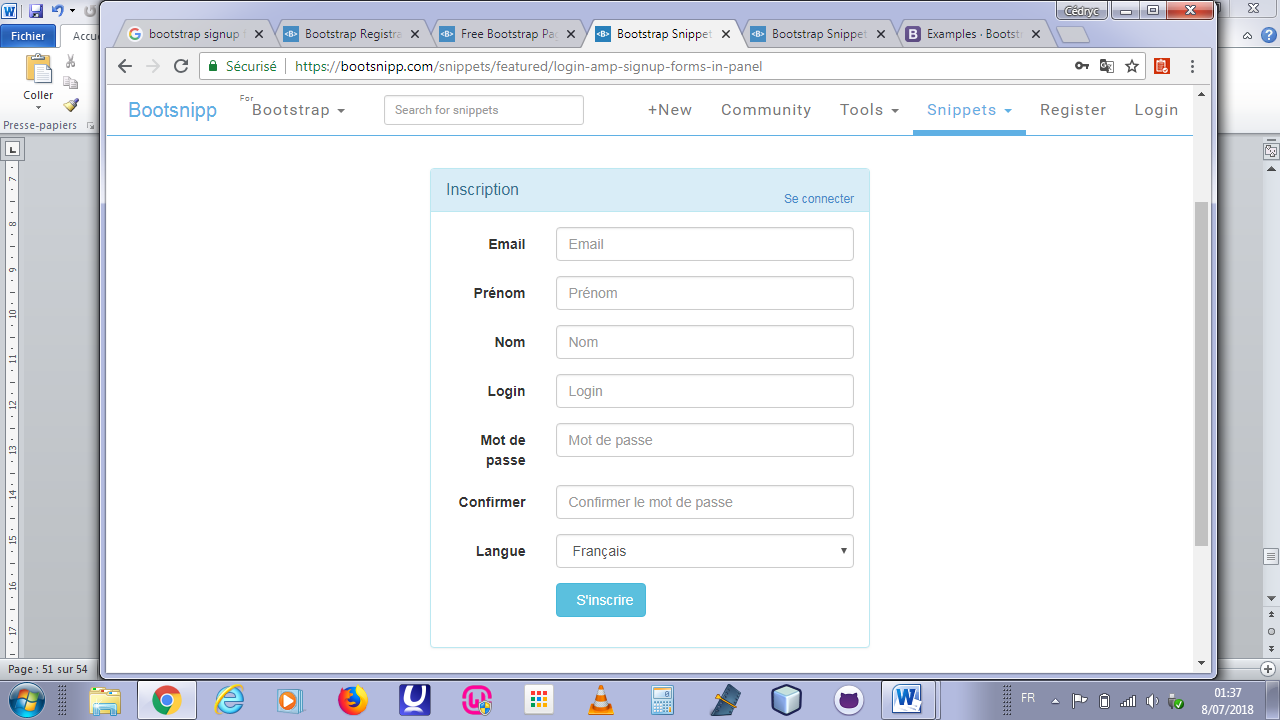
# Écrans et navigation

## Mockups

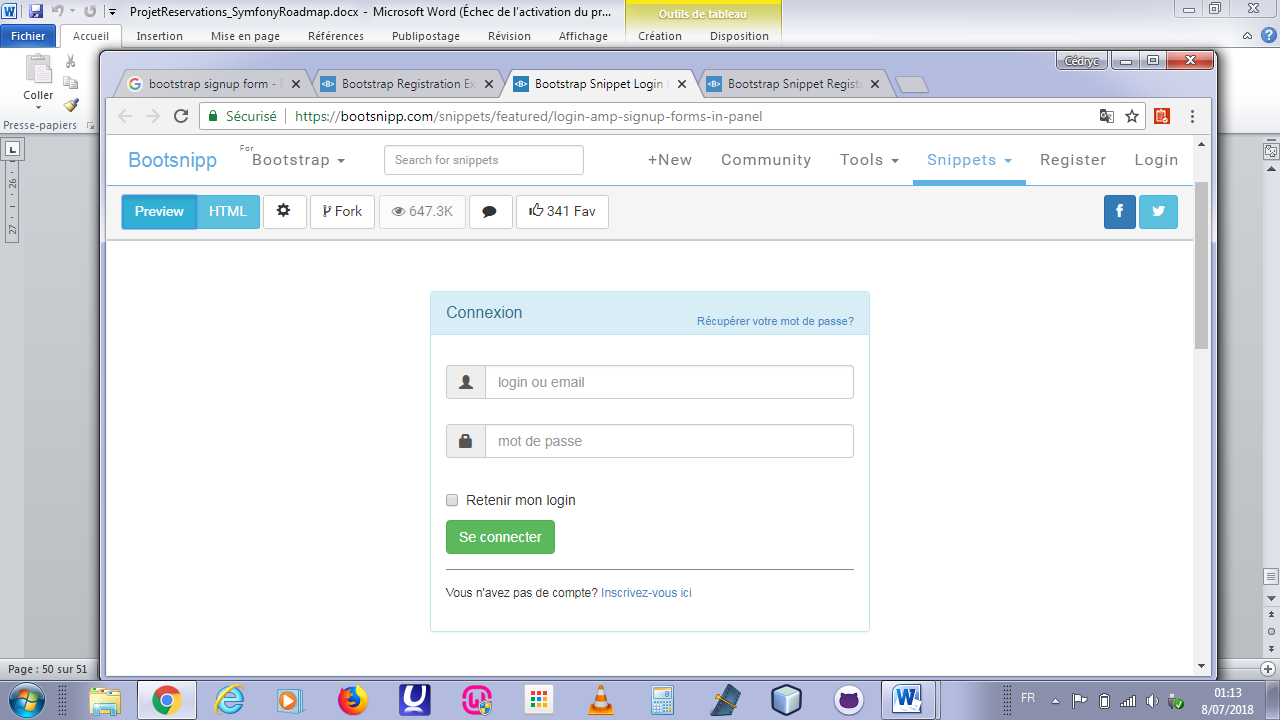
### Trame de page principale



### Formulaire d’inscription



### Formulaire de connexion



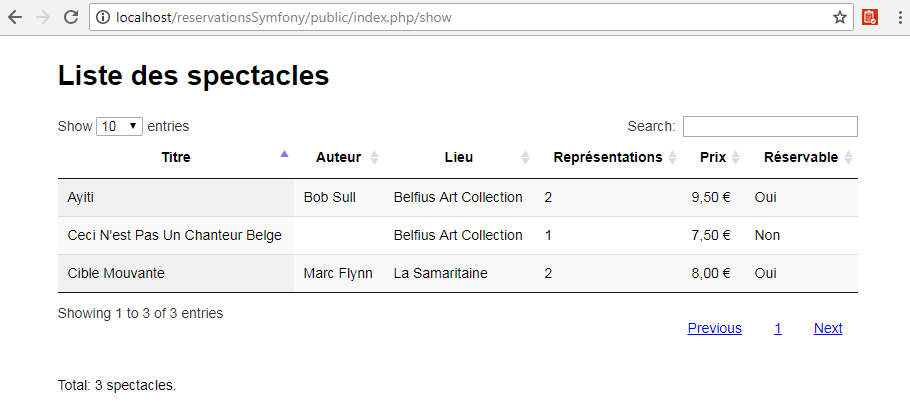
### Menu d’authentification

Ce menu sera intégré dans l’en-tête de la page d’accueil.



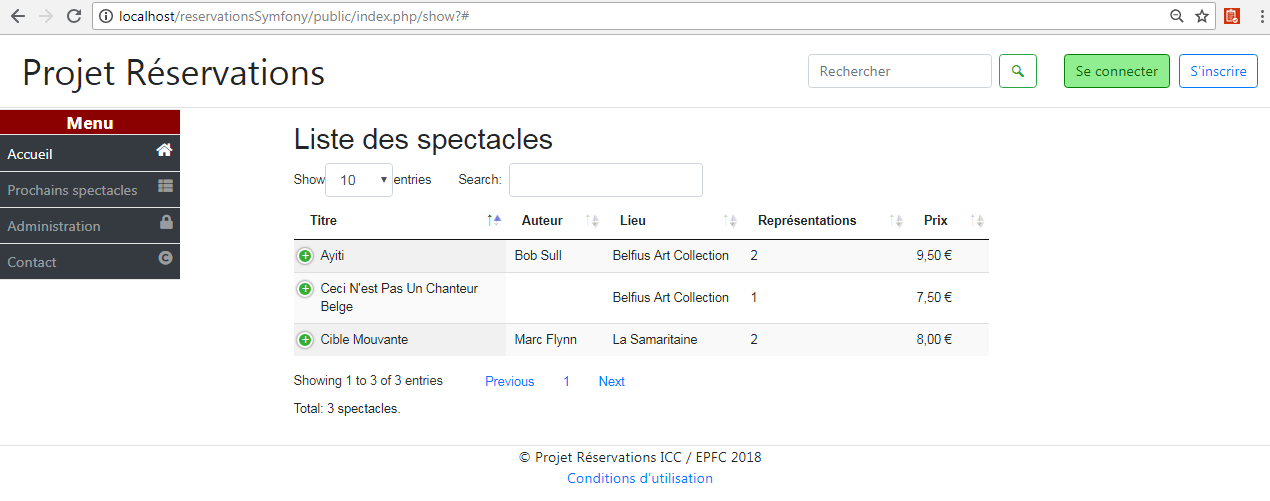
### Catalogue des spectacles

Ce catalogue sera intégré dans la page d’accueil.



### Menu de navigation

Ce menu sera intégré dans l’en-tête ou dans la sidebar de toutes les pages (sauf administration).



PID – Projet Réservations

Itération 5 –Formulaires et validation

Dans une application Web les formulaires permettent d’ajouter et de modifier des données. Cependant, il faut veiller à la validité de nouvelles données fournies par l’utilisateur pour des raisons à la fois de sécurité mais aussi d’intégrité et de cohérence des données stockées.

PID – Projet Réservations

Itération 6 –Authentification

Cette itération regroupe plusieurs problématiques liées au principe d’authentification :

* L’inscription des membres,
* La connexion et la déconnexion,
* La récupération du mot de passe,
* L’autorisation d’accès à une ressource ou une fonctionnalité en fonction du rôle d’un membre.

PID – Projet Réservations

Itération 7 – API RESTful

Un exemple d’application double (client + serveur) implémentant les principes d’une API Restful est disponible sur GitHub : <https://github.com/cedryc-ruth/caviste2018.git>

Veuillez utiliser une API publique du type OpenData. Généralement, il suffit de créer un compte pour avoir accès à l’API.

Exemples d’API :

* London Theater : <https://developer.londontheatredirect.com/Documentation>
* Théâtre de la ville de Paris: <https://api.theatredelaville-paris.com/>
* Théâtre contemporain : <https://www.theatre-contemporain.net/images/partenaires/descriptif_opendata_api_cris-v4.pdf>
* Agenda.Brussels : <https://api.brussels/store/apis/info?name=agenda.brussels&version=0.0.1&provider=admin>

1. Diagrammes réalisés sur <https://www.draw.io/> [↑](#footnote-ref-1)