

Groupe 2

Sprint 2

Introduction

Dans ce deuxième audit, nous allons présenter les différentes tâches effectuées au cours du sprint 2.

Comme prévu, nous nous sommes concentrés sur la réalisation du scénario S3 correspondant toujours à l'incrément I1.

Nous avons aussi retravaillé le scénario S1 suite aux retours du premier audit.

Dans ce rapport nous allons présenter en détail :

- Le travail réalisé pour l'IHM
- Le déroulement du sprint2 et notre organisation
- Les choix techniques et les problèmes rencontrés lors du développement
- L'avancée du projet
- Nos objectifs pour le sprint3

Membres du groupe

- Celia Kezmane
- Nicolas Martinez
- Robin Miquel
- Matthieu Lehugeur

Prise en compte de l'audit 1

Remarques reçus lors de l'audit 1 :

- IHM :
 - Évaluation à faire
 - Prix en xx,xx (et pas xx.x)
 - Remplacer le camelCase par du langage naturel
- BD : Corriger une erreur de nomenclature
- Dev :
 - Faire des tests unitaires pour la BD
 - Pour le type de spectacle, remplacer les radios boutons par des checkboxes
- Git et gestion de projet : Ajouter une branche dev et des branches par feature
- Rapport : Numéro en bas de page, références des fichiers sur les diapos

Prise en compte de l'audit 1

Actions réalisées pour les prendre en compte :

- IHM :
 - Evaluation des heuristiques et évaluation automatique des sprint1 et sprint2
 - Correction des erreurs sur l'interface (prix, camelCase)
- BD : Erreur de nomenclature corrigée
- Dev :
 - Mise en place de tests unitaires pour la BD
 - Utilisation de checkboxes
- Git et gestion de projet : Branche master propre, ajout d'une branche dev et des branches pour les features
- Rapport : Numéro en bas de page, références sur les diapos

Prise en compte de l'audit 1

Ajout de
checkboxs



 Accueil Programmation Administration

Recherche

Dates

01 / 03 / 2020 au 31 / 03 / 2020

Catégorie de spectateurs

☒ Indifférent
☐ 1-5 ans
☐ Jeune Public
☐ Tout Public
☐ Adultes

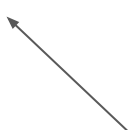
Type de spectacles

☒ Opéra
☒ Humoristique
☒ Drame
☒ Musical
☒ Cirque

Envoyer

Programmation du 01/03/2020 au 31/03/2020

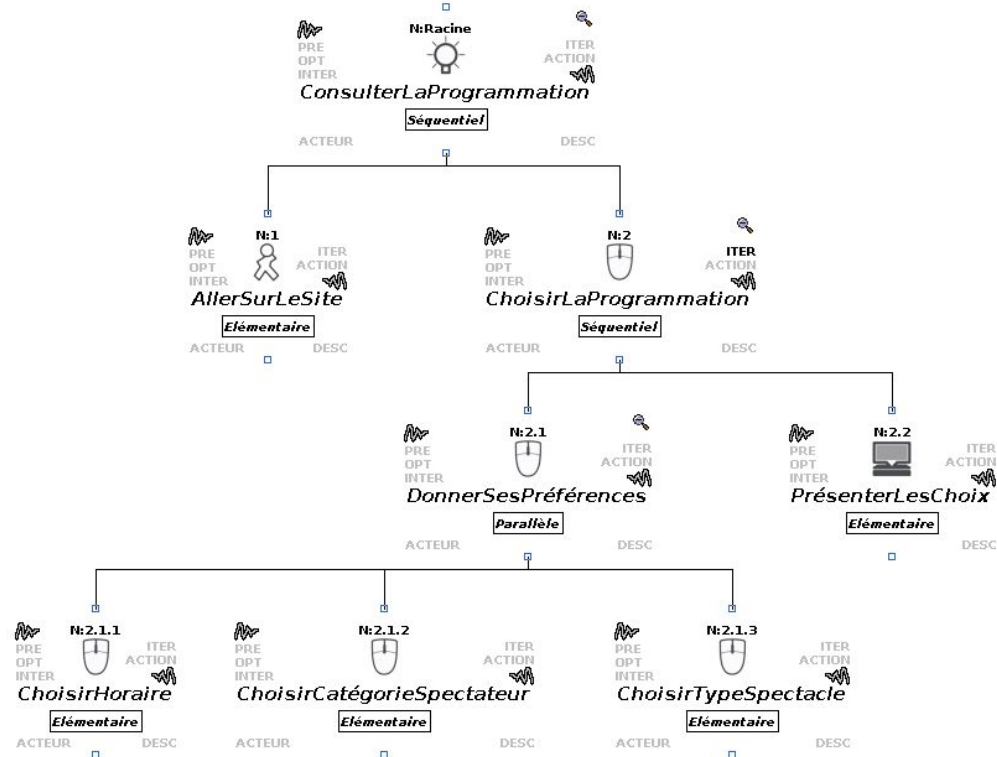
Horaire	Nom	Prix	Type de spectacle	Catégorie de spectateurs
01/03 à 18h00	Le Malade Imaginaire	12,00 €	Humoristique	Tout Public
09/03 à 20h00	Don Juan	10,00 €	Opéra	Adulte
10/03 à 18h30	Andromaque	15,00 €	Drame	Adulte
11/03 à 22h00	Romeo et Juliette	20,00 €	Drame	Tout Public
12/03 à 12h00	Les Perses	15,50 €	Drame	Adulte
13/03 à 18h00	Les Animaux	9,50 €	Cirque	1-5 Ans
13/03 à 20h00	L'avare	10,00 €	Humoristique	Tout Public
13/03 à 22h00	Andromaque	15,00 €	Drame	Adulte
14/03 à 15h00	Sonorites Etranges	10,00 €	Musical	Jeune Public



Respect du langage utilisateur

Retour sur l'IHM du sprint1

➤ Modèles de tâches



➤ Charte graphique

La charte graphique sert à "coder" la présentation de l'interface.

On donne des détails sur l'aspect général mais aussi sur l'aspect d'éléments précis (comme les formulaires ou la barre de navigation).

En plus de cela, pour chaque scénario, nous avons ajouté des détails les concernant.

➤ Maquette

Logo	Navigation dans le site																				
Choix des dates du spectacle	Affichage de la programmation dans un tableau bootstrap <table border="1"> <thead> <tr> <th>Horaire</th> <th>Nom</th> <th>Prix</th> <th>Type de spectacles</th> <th>Catégorie de spectateurs</th> </tr> </thead> <tbody> <tr> <td colspan="5">Résultat de la requête</td> </tr> <tr> <td colspan="5">Résultat de la requête</td> </tr> <tr> <td colspan="5">Résultat de la requête</td> </tr> </tbody> </table>	Horaire	Nom	Prix	Type de spectacles	Catégorie de spectateurs	Résultat de la requête					Résultat de la requête					Résultat de la requête				
Horaire		Nom	Prix	Type de spectacles	Catégorie de spectateurs																
Résultat de la requête																					
Résultat de la requête																					
Résultat de la requête																					
Choix de la catégorie de spectateurs																					
Choix du type de spectacle																					
Envoi des données au serveur																					

*** Site dans sa globalité ***

Utilisation de Bootstrap 4.3.1 sur l'ensemble du site : Quand on utilise une classe de Bootstrap, on l'indique avec cette police

➤ Parties générales

- Affichage des dates : Formats français JJ/MM/AAAA (quand ce n'est pas géré par le navigateur)
- Pour les tableaux, on affiche une ligne sur deux avec un fond gris : `table table-striped` et il n'y a pas de survol des lignes ou des cases

➤ Barre de navigation : `navbar navbar-expand-sm bg-dark navbar-dark`

- Fond gris foncé
- Icône du site en haut à gauche
- Onglets alignés à gauche
- Texte en gris clair puis en blanc une fois survolé
- Police et taille par défaut

➤ Reste de la page : `container-fluid`

- Fond blanc sur le reste de la page
- Titres des sections (Recherche, titre du résultat, ...) : balises HTML `<h2>`
- Sous titres des sections : balises HTML `<h4>`

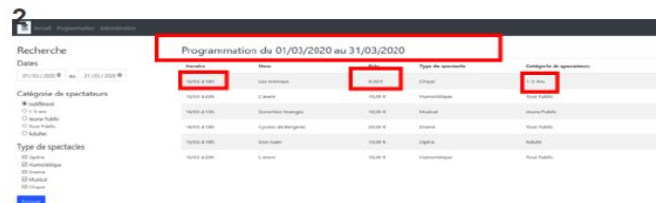
➤ Formulaires

- Situés à gauche de la page (sauf si ce n'est pas pertinent)
- Précédés par un titre
- Peuvent contenir des sous-titres si c'est pertinent
- On adapte la largeur en fonction de ce que le formulaire contient
- Pour les champs, on peut au choix choisir de les afficher :
 - Champ et de son label en ligne : `form-group row`
 - Label au dessus du champ : `form-group`
- Type de champs : On favorise des types limitants (number, time, ...) pour guider l'utilisateur
 - Pour les checkboxes et les radios boutons on met une marge de 0,9cm après le sous-titre
 - Pour les dates, pas de marges
- Bouton affichés en bleus et centré à gauche : `btn btn-primary`
- Survol par défaut de bootstrap : Checkboxes et radios boutons, s'affichant en bleu

➤ Messages d'erreur dans les formulaires : `invalid-feedback`

- En rouge et en petit juste en dessous du ou des inputs problématiques

➤ Evaluation : Heuristiques de Nielsen

Affichage du prix, type de spectacle, catégorie des spectateurs**Critères:**

Correspondance du système avec le monde réel

Description:

Le prix est affiché avec deux chiffres après la virgule et en €.

Le type de spectacle et la catégorie de spectateurs sont renseignés dans la langue de l'utilisateur (commence par une majuscule, présence d'espaces).

Les dates sont affichées au format français.

Si les deux dates renseignées dans les filtres sont identiques, on affiche "Programmation du #Date" au lieu de l'affichage par défaut "Programmation du #Date au #Date" qui ne semblerait pas naturel.

Les 9 heuristiques ont été évalués et les défauts constatés ont été pris en compte et corrigé

Filtres**Critères:**

Coherence et standards

Description:

Les filtres pour une recherche sont tous localisés à gauche et en dessous du mot "Recherche", chaque catégorie de filtre est repérée par un sous-titre mis en évidence.



➤ Corrections suite à l'évaluation des Heuristiques

1) Visibilité de l'état du système

- Ajout d'un message signalant à l'utilisateur qu'aucune `Representation` ne correspond à ses critères de recherche. Ainsi, il sait que si aucun résultat n'est affiché, c'est à cause de ses critères

2) Correspondance du système au monde réel

- Les prix de la forme 10.5€ ont été remplacé par 10,50€
- Les types de spectacle et les spectateurs cibles ne sont plus affiché en camelCase
- Quand la date de début et de fin sont identiques, on affiche uniquement "Programmation du xx/xx/xxxx et plus xx/xx/xxxx au xx/xx/xxxx"

5) Prévention des erreurs

- Nous avons ajouté aux inputs HTML le mot clé "required", l'utilisateur ne peut donc plus rentrer une date vide, ce qui causait une erreur au niveau de la BD

➤ Evaluation : Rapport automatique

L'étude de ce rapport est un peu compliquée. La plupart des informations nous ont semblées difficiles à prendre en comptes et à traiter.

Nous avons réalisé un rapport pour le site avant et après l'affichage des résultats afin d'avoir une étude complète.

Visual Guidance

Saliency

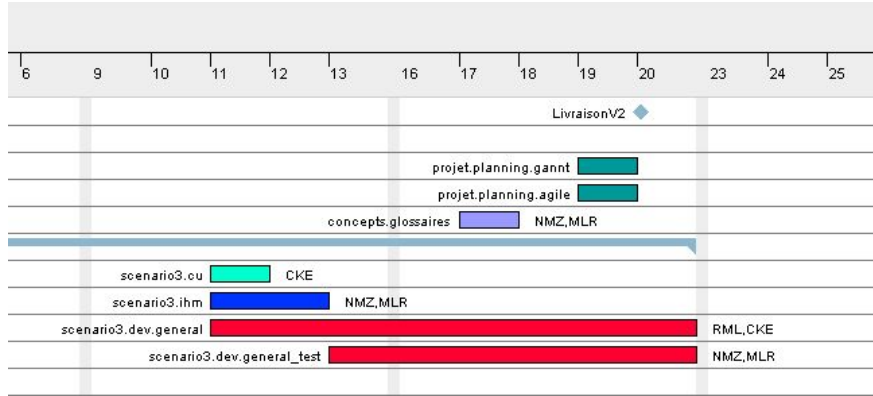
Saliency refers to visual conspicuity of regions on the interface. It predicts which areas attract attention when seeing the interface for the first time or when searching for information.

References: [1]

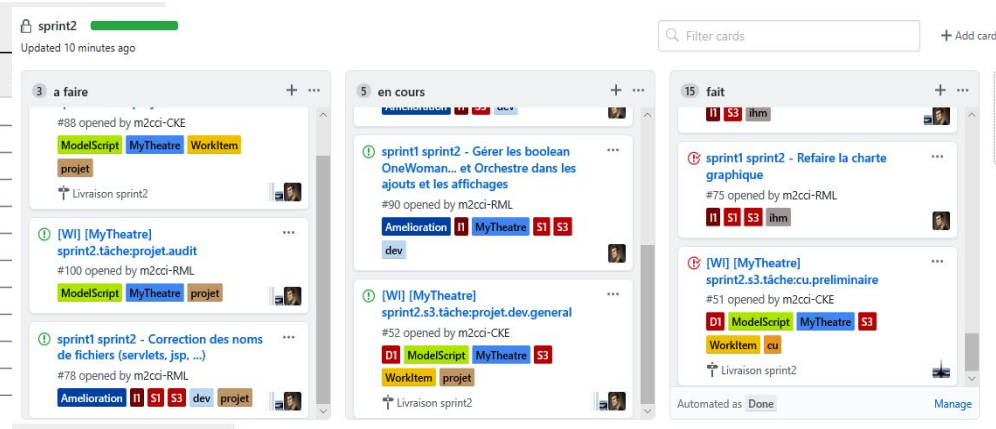
Evidence: ★★★★★

Relevance: ★★★★★☆

Planning effectif - Planning Github



Planning effectif



Planning Github

<https://github.com/m2cci/m2cci-1920-pi-GP02/blob/dev/projet/sprint2/plannings/effectif/diagrammes/plan.png>

<https://github.com/m2cci/m2cci-1920-pi-GP02/projects/3>

Organisation

Utilisation de Discord suite au confinement pour les meeting et les discussions

Pair programming via partage d'écran

Fonctionnement en GitFlow:

- Création d'un branche dev qui sert au développement, la branche master restant "propre"
- Création de branches par fonctionnalité (dev restant elle aussi propre)
- Pull request pour merge avec la branch dev

Ajout de tests unitaires sur la dao et la bd #64

 **Merged** m2cci-MLR merged 3 commits into `dev` from `sprint1.testdao` 6 days ago

 Conversation 0

 Commits 3

 Checks 0

 Files changed 21



m2cci-NMZ commented 6 days ago



Ajout de tests unitaires sur les methodes DAO et tests unitaires vérifiant le schema de la BD via une execution d'insertions sql negatives

Répartition du travail

RML: - IHM du scénario 3
 - Développement de la partie front end pour le scénario 3

CKE: - Tests unitaires sur la partie DAO
 - Cas utilisateurs
 - Développement de la partie front end

NMZ: - Tests unitaires sur la partie DAO
 - Amélioration du glossaire
 - Évaluation logicielle de l'IHM du scénario 3

MRL: - Tests unitaires sur la partie servlet controller
 - Amélioration du glossaire

Rétrospective du sprint2

➤ Ce qui n'a pas marché

- Rétrospective + audit 2 jours après = “redite” et perte de temps
- Stand-up meeting (avant confinement) : pas d'utilité car échange des informations en temps réel. Plus proche d'une concertation pour déterminer les objectifs du jour dans notre cas

➤ Ce qui a marché

- Branches git permet le développement indépendant d'une nouvelle option.
- Organisation sur Discord => Échange et programmation en groupe
- Bonne répartition des tâches / Meeting chaque matin pour savoir ce qui a été fait et ce qui reste à faire
- Création d'issue pour notifier un bug ou une amélioration à réaliser (+ planification agile)
- Pull-Request favorise la relecture du code

➤ Ce qu'il serait bien d'essayer

- Plus de revue de code, inclure des tâches de relecture
- Stand-up plus cohérents par rapport à la taille du groupe (passer plus rapidement sur le bilan du travail effectué et prendre du temps pour la répartition des tâches).

Zoom : Base de donnée

Tests automatiques pour la partie BD

```
@Test
public void testFKLesOperasNumSpe() throws Exception {
    System.out.println("Test FK LesOperas");

    try {
        this.readAndExecuteSqlNegative("bd/jddn7.sql");
        Assertions.fail("Insertion negative réussie");
    } catch (SQLException ex) {
        assertEquals(19, ex.getErrorCode());
    }
}
```

`PRAGMA foreign_keys = ON;`
`INSERT INTO LesOperas VALUES (3, 0);`

Le test essaie de faire une insertion négative qui viole une foreign key constraint

Le test échoue si l'insertion a lieu

Si une exception de de type SQL_CONSTRAINT (code d'erreur 19) est levée, le test réussit

Tests pour la partie servlet controler

Création d'une classe pour les méthodes utilisées par le servlet pour pouvoir les instancier et les tester

```
public class ControlerTools {  
  
    /**  
     * Calcule le nombre de semaines de la période délimitée par deux dates  
     * On compte chaque semaine dans lequel il y a un jour, si c'est du dimanche  
     * au lundi, c'est donc 2 semaines, celle du dimanche et celle du lundi  
     *  
     * @param d1 : Premier jour  
     * @param d2 : Dernier jour  
     * @return : int correspondant au nombre de semaine  
     */  
    public static int nbSemaineEntre(Date d1, Date d2) {  
        // Calcul du nombre de jour entre les deux dates  
        LocalDate date1 = d1.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();  
        LocalDate date2 = d2.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();  
        long nbJour = ChronoUnit.DAYS.between(date1, date2) + 1;    // On ajoute un pour inclure d1 et d2  
  
        // Calcul du nombre de semaines  
        int nbSem = 0;  
        int nbJourAEnlever = numeroJourSemaine(d2) + 1;    // On commence par enlever le nombre de jours dans la dernière semaine  
        while (nbJour > 0) {  
            nbSem++;  
            nbJour -= nbJourAEnlever;  
            nbJourAEnlever = 7;  
        }  
  
        return nbSem;  
    }  
}
```

```
@Test  
public void testNbSemaineEntreFinSemaine() throws Exception {  
    System.out.println("nbSemaineEntre");  
    Date d1 = new SimpleDateFormat("yyyy-MM-dd").parse("2020-03-01");  
    Date d2 = new SimpleDateFormat("yyyy-MM-dd").parse("2020-03-15");  
    int expectedResult = 3;  
    int result = ControlerTools.nbSemaineEntre(d1, d2);  
    assertEquals(expResult, result);  
}  
  
@Test  
public void testNbSemaineEntreDebutSemaine() throws Exception {  
    System.out.println("nbSemaineEntre");  
    Date d1 = new SimpleDateFormat("yyyy-MM-dd").parse("2020-03-02");  
    Date d2 = new SimpleDateFormat("yyyy-MM-dd").parse("2020-03-16");  
    int expectedResult = 3;  
    int result = ControlerTools.nbSemaineEntre(d1, d2);  
    assertEquals(expResult, result);  
}
```

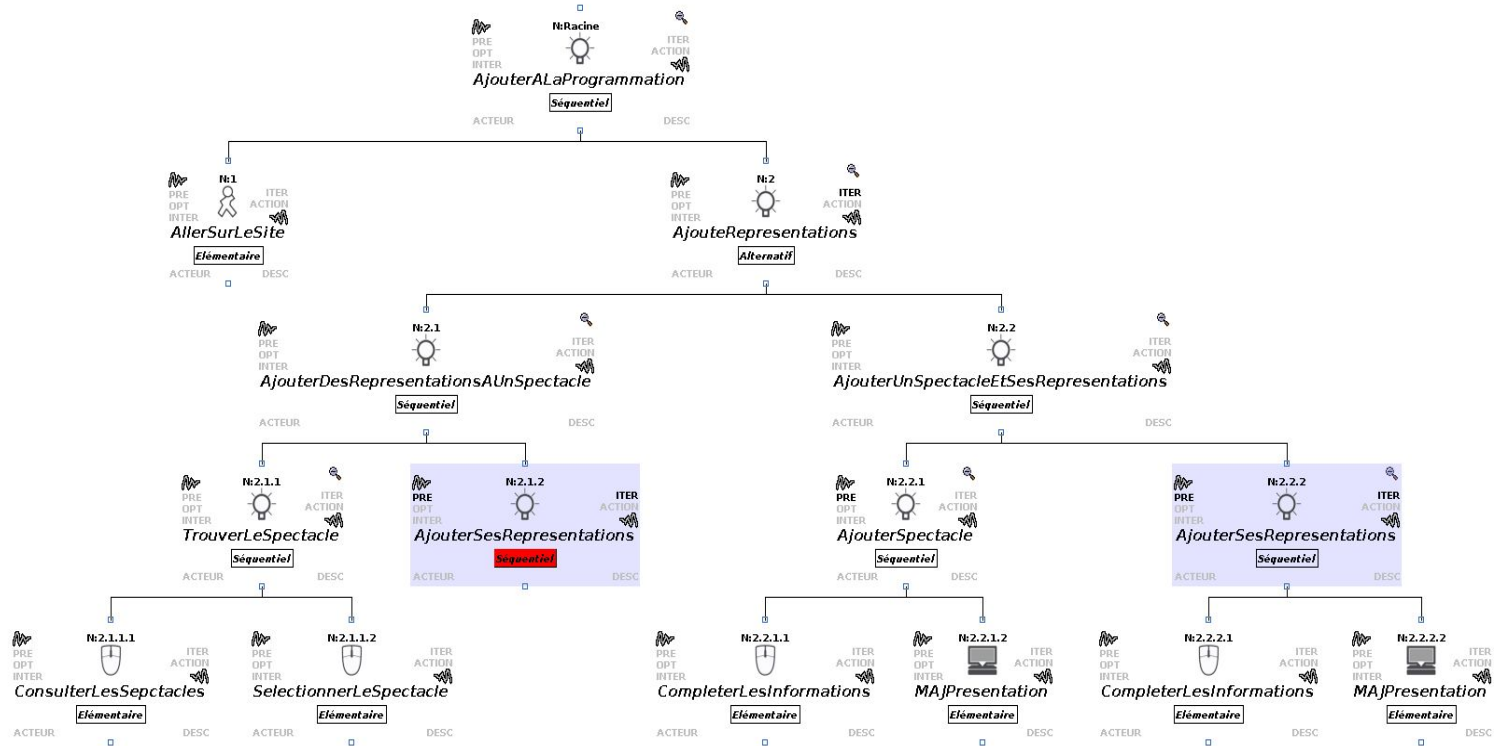

Réalisation du scénario 3

Description du problème

Suite à la lecture du scénario, nous avons retenu 3 cas d'utilisations :

- Ajout de `Spectacle` : Le `Gerant` doit pouvoir ajouter des `Spectacle`s au système
- Ajout de `Representation` : Le `Gerant` doit pouvoir ajouter des `Representation`s pour les `Spectacle`s qui sont dans le système
- Affichage de la `Programmation` : Le `Gerant` peut visualiser la `Programmation` de manière plus globale qu'un `Spectateur`. Les informations sont plus compactées et lui offrent une vision d'ensemble

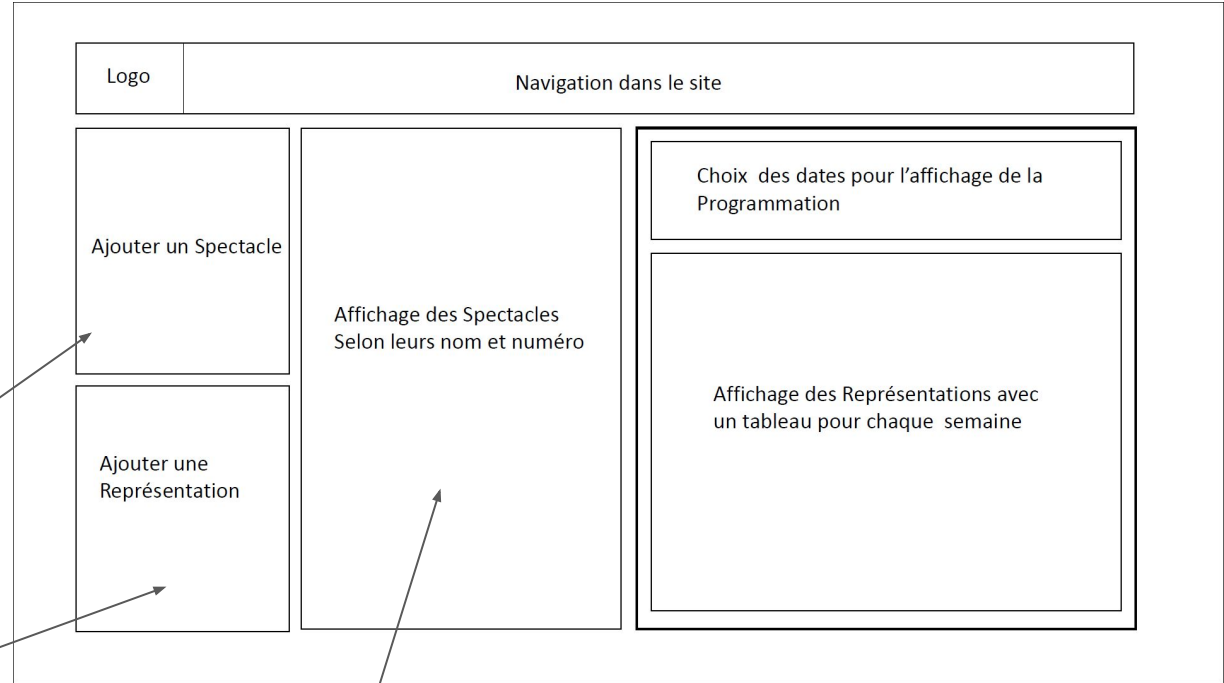
Modèle de tâches



Maquette

Nous avons fait le choix de mettre sur une même page l'ajout de `Spectacle`, de `Représentation`, ainsi que la consultation du planning des représentation.


Cela permet d'éviter à l'utilisateur de changer de page pour chacune de ces sous-tâches.



Formulaire pour ajouter les informations au système

Permet de connaître le numéro du spectacle pour l'ajout de `Représentation`s

Page Web

 Accueil Programmation Administration

Número

Nom

Prix

Cible

1-5 Ans

Type

Opéra

Ajouter

Représentation

Número du Spectacle

Horaire de la Représentation

jj / mm / aaaa

à

-- : --

Ajouter

N°

Nom du Spectacle

17

Andromaque

26

Bel Ami

45

Cyrano de Bergerac

25

Don Juan

20

L'avare

21

Le Cid

22

Le Malade Imaginaire

46

Les Animaux

23

Les Perses

24

Romeo et Juliette

47

Sonorites Etranges

27

Tous Petits

18

Une Piece

La Programmation

Du

01 / 03 / 2020

au

31 / 03 / 2020

Envoyer

Semaine du 24/02/2020 au 01/03/2020

Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
						18h00 Le Malade Imaginaire

Semaine du 02/03/2020 au 08/03/2020

Planning vide

Semaine du 09/03/2020 au 15/03/2020

Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
20h00 Don Juan	18h30 Andromaque	22h00 Romeo et Juliette	12h00 Les Perses	18h00 Les Animaux	15h00 Sonorites Etranges	18h00 Don Juan
	20h00 Une Piece			20h00 L'avare	18h00 Cyrano de Bergerac	20h00 L'avare
	22h00 Le Cid			22h00 Andromaque		22h00 Cyrano de Bergerac

Semaine du 16/03/2020 au 22/03/2020

Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
			15h00 Tous Petits		18h00 Romeo et Juliette	

20

Affichage des `Spectacle`s

- Le servlet principal fait une requête à la BD sur tous les spectacles
- Dans le jsp on les affiche dans un tableau avec uniquement le nom et le numéro du spectacle
- Si la liste est trop longue, il y a une barre de scrolling qui apparait, ainsi, les formulaires sont toujours visibles

Affichage des `Représentation`s

- Le servlet principal fait une requête sur toutes les représentations entre les dates du formulaire
- Il les trie en différentes listes pour chaque semaines et chaque jour de la semaine
- Le jsp affiche les représentations comme dans une sorte de calendrier, par `Horaire`s croissantes
- Si besoin, il y a aussi une barre de scrolling

Ajout des `Spectacle`s des des `Représentation`s

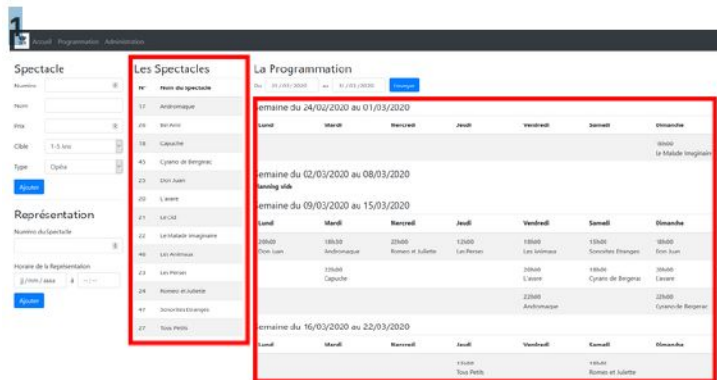
- Dans les formulaires, tous les champs sont required et les types des inputs sont adaptés aux données : il est normalement impossible de valider le formulaire si les données sont incorrectes
- Les deux formulaires sont indépendant, ils appellent chacun un servlet qui fait l'ajout à la BD
- Si l'ajout échoue à cause d'une exception SQL, on essaie de la traiter : On fait une requête pour vérifier si les clé primaires ou les clés étrangères sont violées, si c'est le cas, on affiche un message d'erreur
- Ces servlets appellent ensuite le servlet principal qui se sert d'afficher la page

Numéro du Spectacle

Ce numéro ne correspond à aucun spectacle, il faut d'abord ajouter le spectacle

Evaluation de l'IHM

Nous avons réalisé l'évaluation de l'IHM avec l'outil automatique et en étudiant les heuristiques.



État du système connu par l'utilisateur mais améliorable



Critères:

Visibilité de l'état du système

Description:

Comment est-elle bien mise en application ?

L'état du système est connu par l'utilisateur, notamment pour l'ajout des Spectacles et des Représentations. A chaque ajout, la liste des Spectacles et des Représentations sont mis à jour en direct pour que l'utilisateur puisse suivre l'évolution de sa Programmation.

Comment pourrait-elle être appliquée ?

Il pourrait être intéressant de signaler à l'utilisateur dans chacune des deux listes quel est le dernier ajout effectué afin qu'il le retrouve facilement.

Heuristiques

Suite à l'analyse des heuristiques, plusieurs problèmes nous sont apparus.

La liste complète est disponible dans le rapport heuristique mais nous retenons principalement les défauts suivants :

- L'utilisateur ne peut pas supprimer un ajout incorrecte avec notre interface
- L'utilisateur n'a pas accès au prix, type, cible des spectacles une fois ajoutés

Ces problèmes devront être corrigés dans le produit final. Ils vont demander de repenser certaines parties de l'interface et d'ajouter des éléments de JavaScript.

Cela va probablement retarder le projet.

Avancée du projet

Scénario 1: La consultation de la `Programmation` est disponible

Scénario 3 : Le `Gérant` peut ajouter des `Spectacle`s et des `Representation`s mais dans l'état les heuristiques de Nielsen ne sont pas assez respectées

Remarque : Dans les scénarios 1 et 3, la présence d'orchestre pour les spectacles musicaux et de OneWomanShow pour les Spectacle humoristiques n'est pas du tout pris en compte. Il faudra aussi corriger cette absence.

Objectifs pour le sprint3

Initialement, nous avons prévu de faire les scénarios suivantes pendant le dernier sprint. Cependant, il apparaît que dans l'état, l'IHM du scénario S3 est trop problématique et nous devons corriger ce point.

Nous allons aussi essayer de mettre en place une notion d'authentification pour que seul le gérant puisse accéder à l'administration.

Il faut aussi que l'on ajoute la gestion des orchestres et des OneWomanShow pour que les scénarios S1 et S3 soient considérés valides.

Une partie du groupe devrait néanmoins commencer la rédaction des différents fichiers ModelScript pour le scénario S1bis en parallèle de la correction des problèmes relevés.