File Finder



Xavier Schwab et Sacha Grenier

SiT-1A – CPNV ES

Chefs de projet : Chevillat Jerome & Hurni Pascal

Table des matières

[1 Spécifications 4](#_Toc484607484)

[1.1 Titre 4](#_Toc484607485)

[1.2 Description 4](#_Toc484607486)

[1.3 Matériel et logiciels à disposition 4](#_Toc484607487)

[1.4 Prérequis 4](#_Toc484607488)

[1.5 Cahier des charges 4](#_Toc484607489)

[2 Planification Initiale 5](#_Toc484607490)

[3 Analyse 6](#_Toc484607491)

[3.1 Opportunités 6](#_Toc484607492)

[3.2 Document d’analyse et conception 6](#_Toc484607493)

[3.2.1 Page principale 7](#_Toc484607494)

[3.2.2 La page Planning des utilisateurs 9](#_Toc484607495)

[3.2.3 Page travailleurs 9](#_Toc484607496)

[3.2.4 Base de données 10](#_Toc484607497)

[3.3 Conception des tests 14](#_Toc484607498)

[3.4 Planification détaillée 14](#_Toc484607499)

[4 Dossier de Réalisation 16](#_Toc484607500)

[4.1.1 Mise en place de l’environnement 16](#_Toc484607501)

[4.1.2 Fichier de migration 18](#_Toc484607502)

[4.1.3 Travailleurs 19](#_Toc484607503)

[4.1.4 Menu de gestion de niveaux 21](#_Toc484607504)

[4.1.5 Planning général 24](#_Toc484607505)

[4.1.6 Planning des travailleurs 29](#_Toc484607506)

[4.2 Modifications 30](#_Toc484607507)

[4.2.1 Modifications à faire 30](#_Toc484607508)

[4.2.2 Modifications effectuées 31](#_Toc484607509)

[5 Tests 32](#_Toc484607510)

[5.1 Dossier des tests 32](#_Toc484607511)

[6 Conclusion 34](#_Toc484607512)

[6.1 Bilan des fonctionnalités demandées 34](#_Toc484607513)

[6.2 Bilan de la planification 34](#_Toc484607514)

[6.3 Bilan personnel 34](#_Toc484607515)

[7 Divers 35](#_Toc484607516)

[7.1 Journal de travail 35](#_Toc484607517)

[7.2 Webographie 35](#_Toc484607518)

[8 Annexes 36](#_Toc484607519)

[8.1 Cahier de charges 36](#_Toc484607520)

[8.2 Script SQL obtenu avec l’application jMerise 39](#_Toc484607521)

[8.3 Journal de travail 42](#_Toc484607522)

# Spécifications

## Titre

Plateforme de recherche avancée de documents sur un répertoire précis.

## Description

Ce projet nous a été demandé par un client. L’idée est de créer une application pour retrouver des fichiers qui avec le temps sont devenus trop nombreux.

Une réalisation générique de ce projet permettra d'utiliser la plateforme sur n’importe quel ordinateur équipé d’un système d’exploitation Windows.

## Matériel et logiciels à disposition

* Ordinateur standard CPNV
* Microsoft Project
* Logiciel de programmation (Visual studio)
* Logiciels de gestion de projet
* Logiciels de réalisation de documentation (Suite office)
* Un dépôt GIT

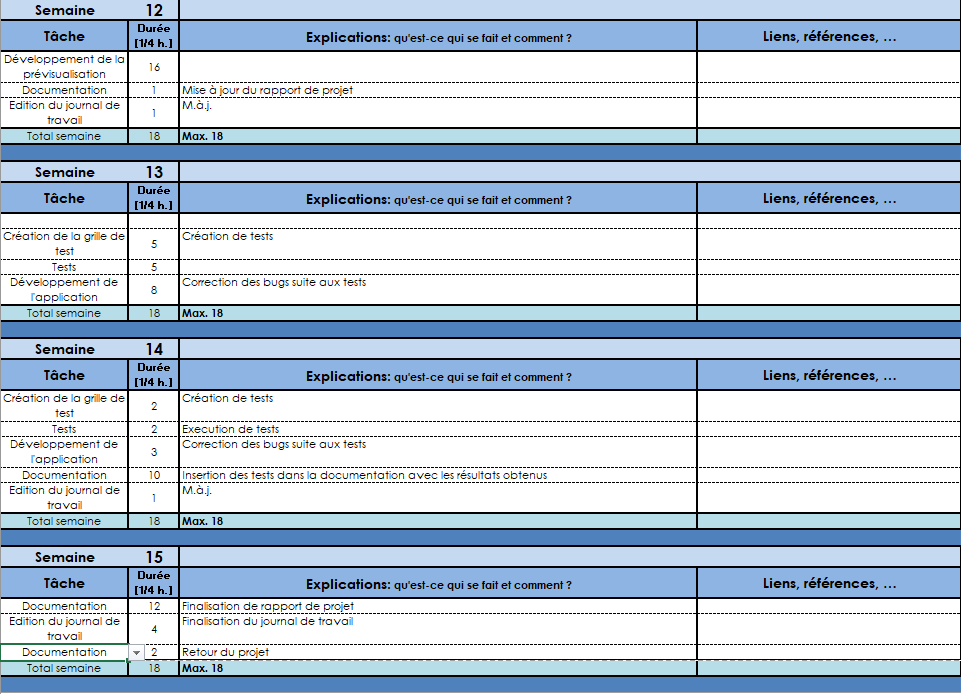
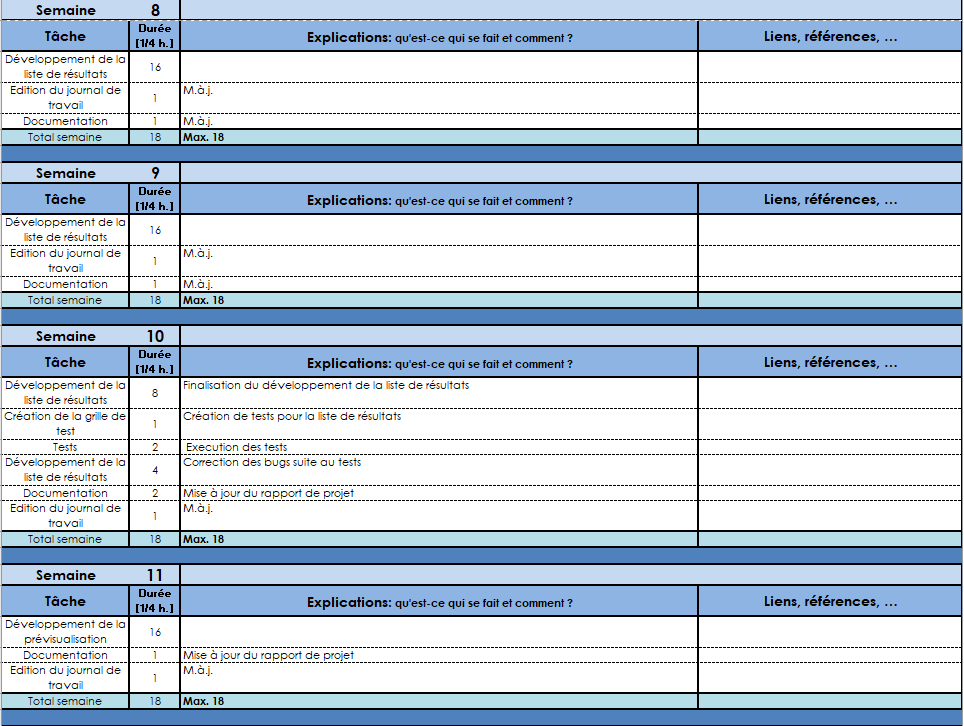
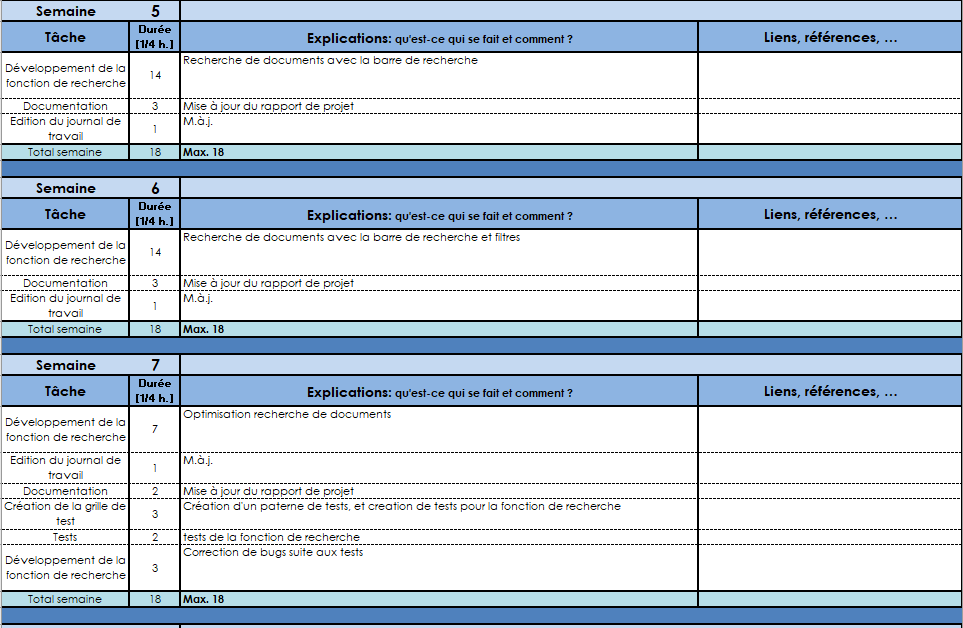
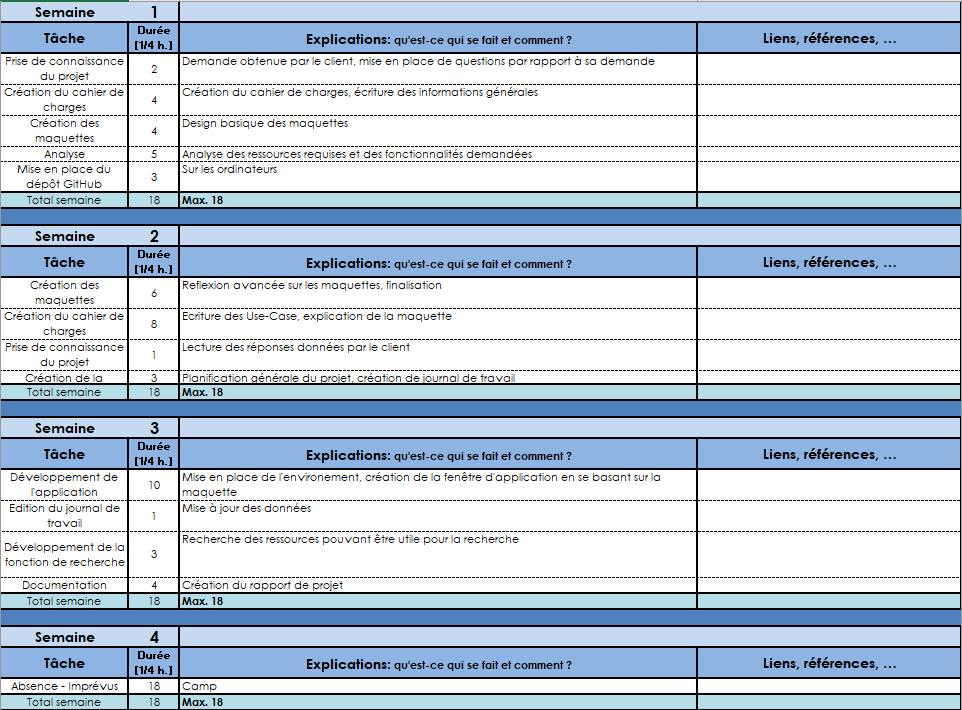
## Prérequis

Un niveau en C# CFC est nécessaire

## Cahier des charges

<https://github.com/SachaGrenier/MAW1.1/blob/master/CDCProjet/CDC/Cahier_des_charges.docx>

# Planification (Gantt)



# Analyse

## Opportunités

La création de cette application va nous permettre de :

* Nous améliorer en C#, découvrir de nouvelles fonctions, libraires, etc.
* Progresser en écriture de rapport de projet.
* Apprendre à utiliser le potentiel de chaque développeur en séparant le travail par tâches

Les difficultés seront :

* Le développement de la fonction de recherche à l’intérieur des documents
* Rendre la recherche la plus rapide possible
* Le travail en groupe peut apporter des conflits

## Document d’analyse et conception

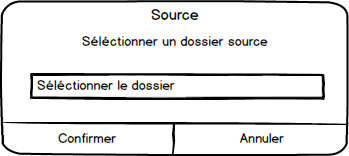
Le but est de créer une application qui permet d’effectuer des recherches avancées de documents sur un système Windows quelconque. Nous sommes mandatés par un client qui rencontre des difficultés à faire de recherches du a un grand nombre de documents.

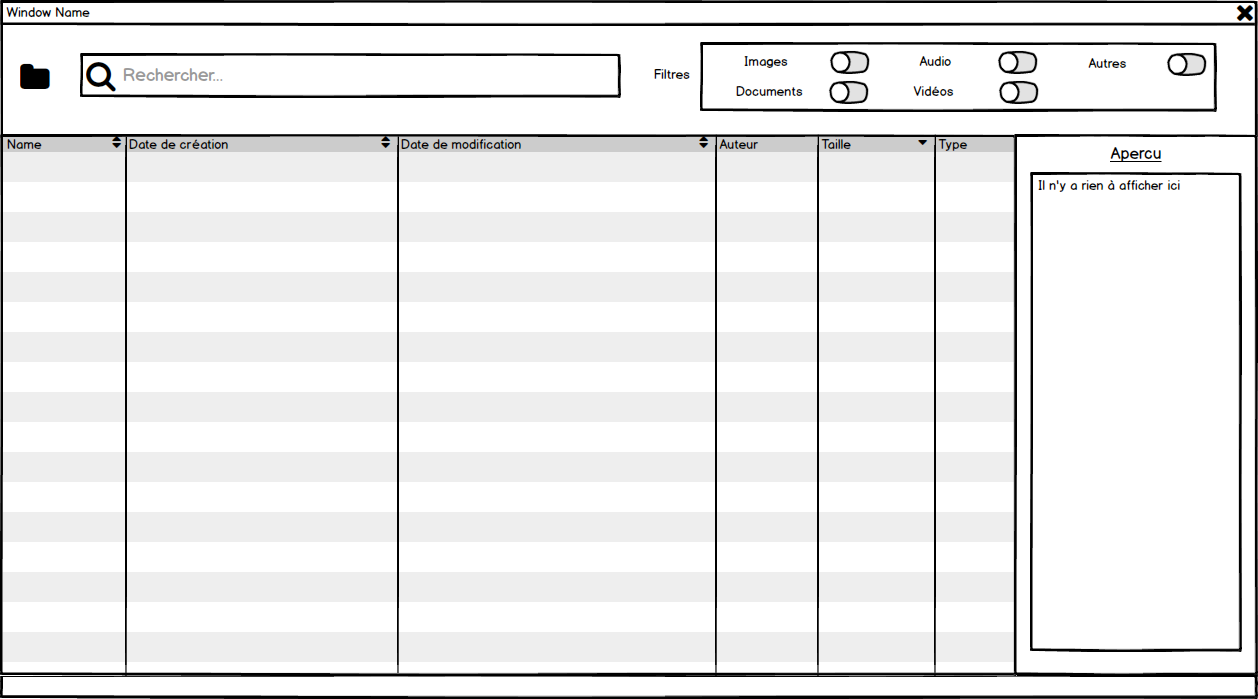
La fonctionnalité de recherche sera accompagnée de filtres qui permettront par exemple de visualiser uniquement les images, les vidéos ou les documents !

### Maquettes

En ce qui concerne le visuel de l’application, nous avons créé des maquettes correspondant à chaque étape de la recherche :

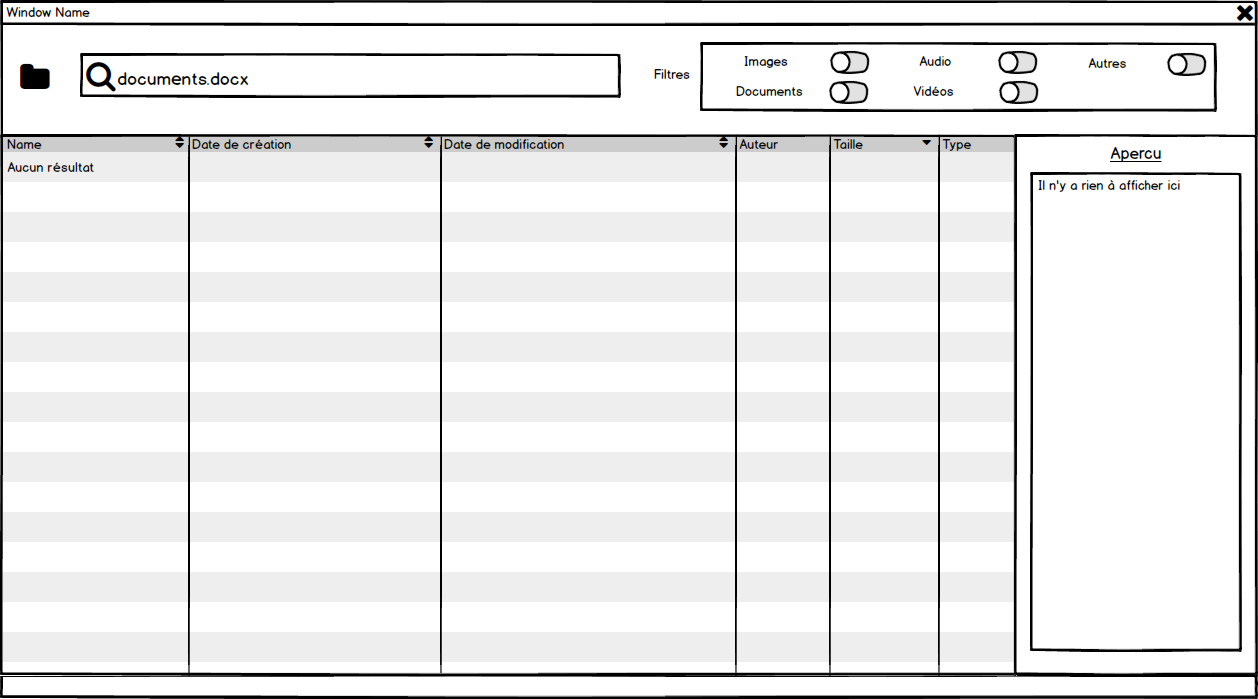
1. Démarrage de l’application.



1. Sélection du dossier d’où les recherches seront effectuées  
   Une fois le répertoire sélectionné, l’application se lance.

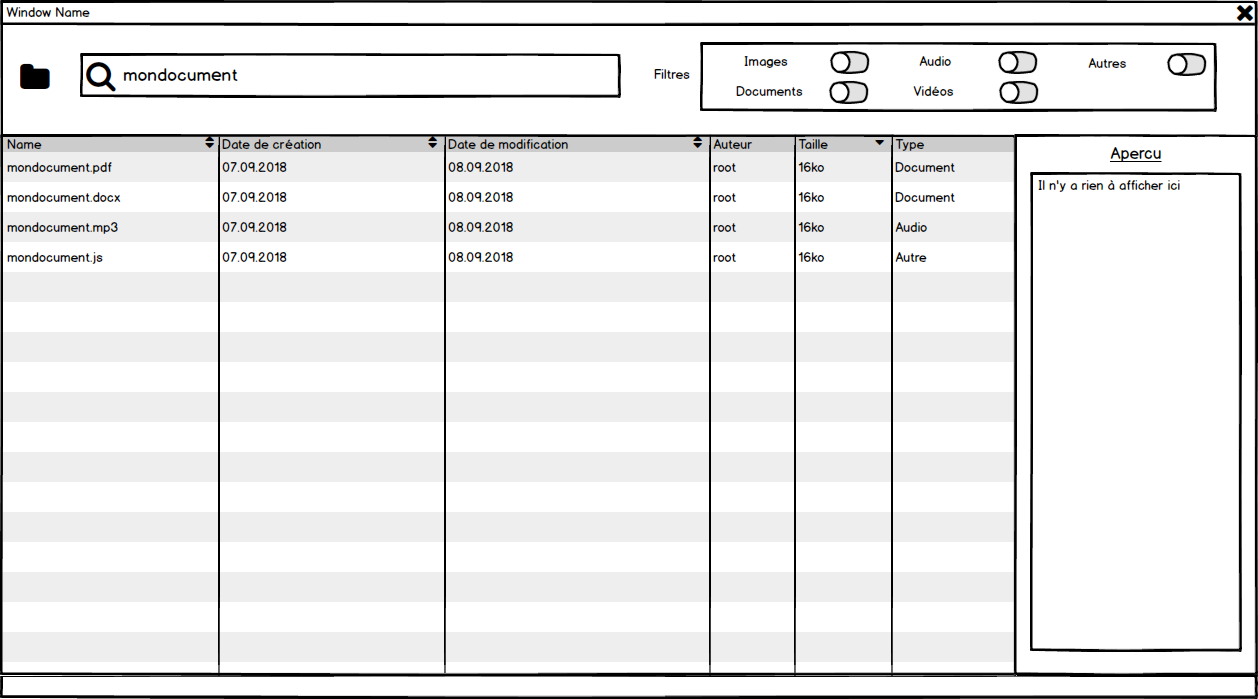
# Ce bouton permet de choisir un autre répertoire source

1. Ici, on recherche « documents.docx ». Etant donné qu’il n’existe pas, aucun résultat n’apparait.

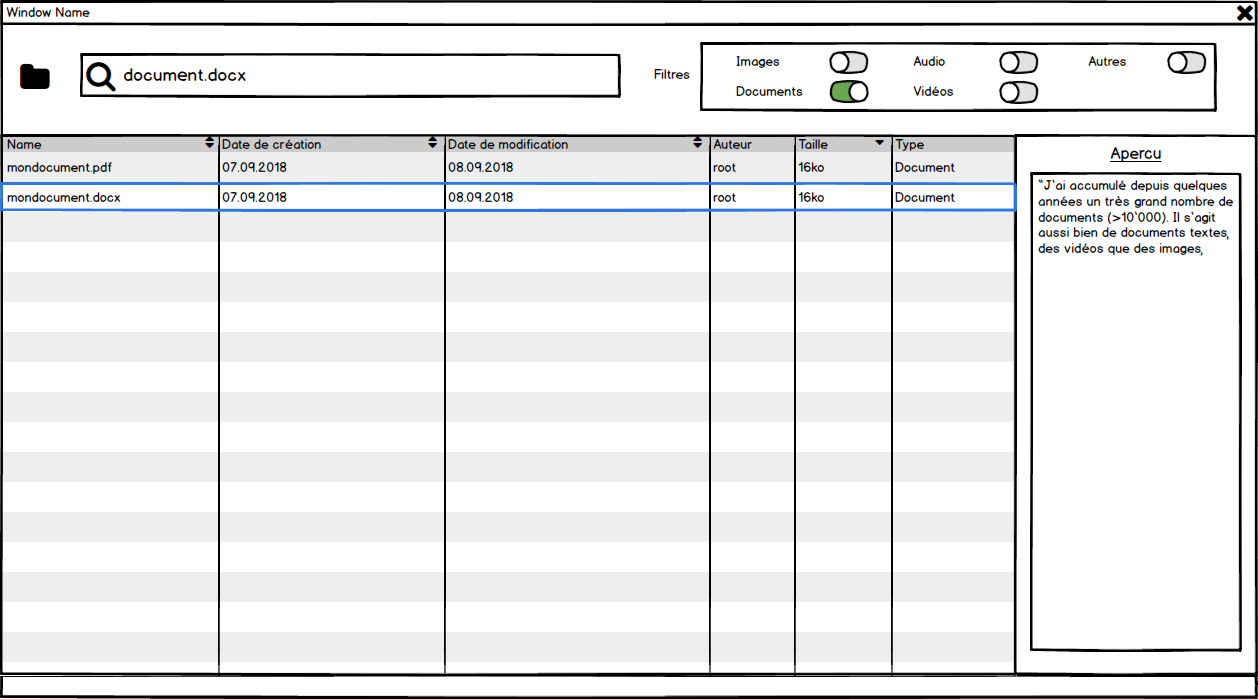


# La recherche est effectuée en temps réel

1. On recherche maintenant « mondocument » et plusieurs résultats apparaissent.

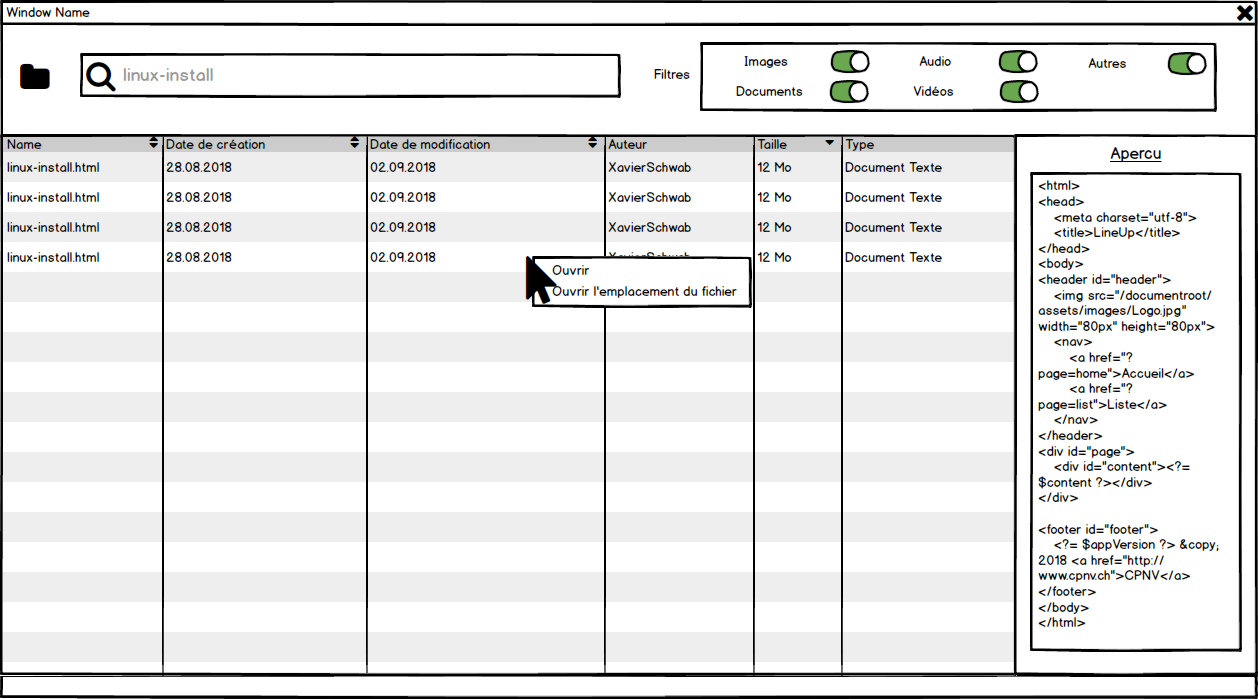


1. On applique le filtre documents et on clique sur le document qui nous intéresse. On peut voir à droite qu’une fenêtre avec un aperçu du contenu du document est apparue.



# L’aperçu du contenu du fichier fonctionnera uniquement avec des documents non formatés. Nous allons essayer de faire en sorte qu’il soit possible d’apercevoir le plus de documents possible. En ce qui concerne les images une miniature sera affichée, pour le reste rien n’est prévu à ce jour.

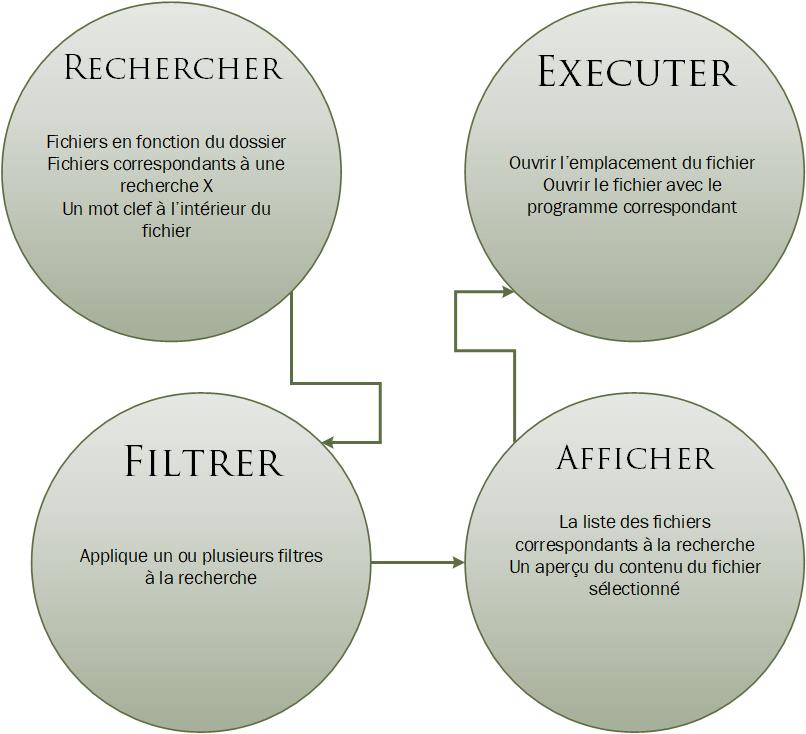
1. Dernière fonctionnalité, lorsque l’on fait un clic droit sur le document qui nous intéresse, un menu propose d’ouvrir avec soit l’application correspondant à l’extension soit l’explorateur windows avec l’emplacement du fichier.



### Architecture

En ce qui concerne l’architecture de nos méthodes, le schéma est plutôt simple :

.



## Conception des tests

Mes tests vont être effectués suivant une grille que je vais créer et qui aura les colonnes suivantes :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Fonctionnalité  à tester | Résultat attendu | Date | Résultat obtenu | Solutions | Etat du bug | Temps passé à résoudre |

Au fur et à mesure du projet, j’ajouterai des fonctionnalités à tester dans la grille.

## Répartition des tâches

Il est vital pour un projet de groupe de faire une répartition des tâches, notamment pour pouvoir utiliser correctement les capacités de chacun.

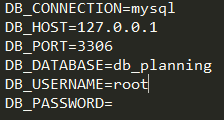
|  |  |
| --- | --- |
| **Tâches Xavier** | **Tâches Sacha** |
| Développement en général | Mise en place du git |
| Choix de technologies | Création des maquettes |
| Valide les documents | Création de la planification |
|  | Création du rapport de projet |
|  | Participation au développement |
|  |  |

# Dossier de Réalisation

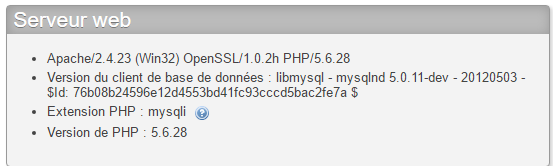
### Mise en place de l’environnement

Premièrement, j’ai installé une machine virtuelle avec VMware Workstation Player 12. J’y ai installé Windows 10, mais peu importe la version. Une fois que l’OS a été mis en place et que quelques outils y ont été installés (Notepad++, Sublime Text, Git Extensions), j’ai installé l’outil Composer qui permet notamment d’installer Laravel avec la commande suivante : « composer global require "laravel/installer" ».

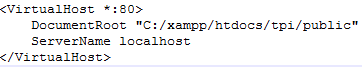
Laravel à présent installé, j’ai exécuté la commande « laravel new tpi ». Cette commande crée un répertoire dans le dossier courant de la console. Pour mon cas, c’était dans « C:\xampp\htdocs\ ». Ce répertoire fait office de structure de base pour mon projet. Il a installé tout ce dont j’avais besoin pour commencer mon projet. Il faut juste modifier le fichier «. env » pour lui permettre d’accueillir la base de données.



En ce qui concerne le serveur apache et le serveur MySQL, un serveur virtuel XAMPP a fait l’affaire. Voici les différentes versions de mes outils :



J’ai modifié le fichier vHosts pour qu’il aille directement pointer sur mon projet.



Lors de mon projet, j’ai remarqué que j’allais avoir besoin d’utiliser des formulaires. Laravel propose une libraire pour gérer les formulaires qui se trouve être très pratique. Pour l’installer, c’est très simple. Il faut se rendre à la racine du projet avec la console, puis exectuer la commande suivante :

composer require "laravelcollective/html":"^5.2.0"

Une fois le processus d’installation effectué par la commande, il faut modifier le fichier de configuration du projet pour y accueil la nouvelle librairie.

Le fichier est le suivant :

<https://github.com/SachaGrenier/TPI/blob/master/config/app.php>

Ce qu’il faut ajouter dans le tableau « providers » :

Collective\Html\HtmlServiceProvider::class,

Ce qu’il faut ajouter dans le tableau « aliases» :

'Form' => Collective\Html\FormFacade::class,

'Html' => Collective\Html\HtmlFacade::class,

J’ai aussi du ajouter la ibraire utilisée pour la génération des fichiers PDF. Au départ, j’ai installé wkhtmltopdf (<https://github.com/barryvdh/laravel-snappy>). Mais suite à des tests, je me suis rendu compte que ce n’était pas la libraire qui convenait aux besoins du projet. Je l’ai donc désinstallé, puis ai trouvé une librairie plus adéquate. Il s’agit de domPDF. La marche à suivre pour l’installer est la même que pour laravelcollective html : <https://github.com/barryvdh/laravel-dompdf>

Documentation de la libraire HTML : <https://laravelcollective.com/docs/5.2/html>

Documentation officielle pour Installer Laravel : <https://laravel.com/docs/5.4>

**Pour simplement importer mon projet :**

Premièrement, il faut avoir XAMPP et se rendre dans le dossier htdocs pour cloner mon projet avec la commande suivante :

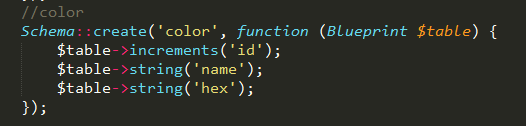
« git clone <https://github.com/SachaGrenier/TPI/> » . Ensuite, il faut modifier le fichier vHost (comme ci-dessus) pour pouvoir y accéder depuis le lien « <http://localhost> ». Enfin, il faut créer une base de données avec le nom : db\_planning puis exécuter le script suivant : <https://github.com/SachaGrenier/TPI/blob/master/TPI_files/db_planning.sql> (peut être fait directement dans PHPmyadmin depuis l’onglet « importer »).

### Fichier de migration

Ce fichier représente les fichiers MCD MLD et MPD que j’ai fournis précédemment. La base de données de Laravel fonctionne avec des migrations. C’est beaucoup plus pratique de modifier un élément de la base de données sans devoir tout supprimer. Pour créer un fichier de migration, il faut utiliser la console et inscire la commande suivante :

H:\TPI\screens\migration_screen.PNG

Dans ce cas-là, je donne déjà le nom de ma migration qui est « create \_all\_tables ». Une fois le fichier crée, il suffit de le remplir avec la tables et les champs souhaités. Ci-dessous un exemple de table dans une migration :

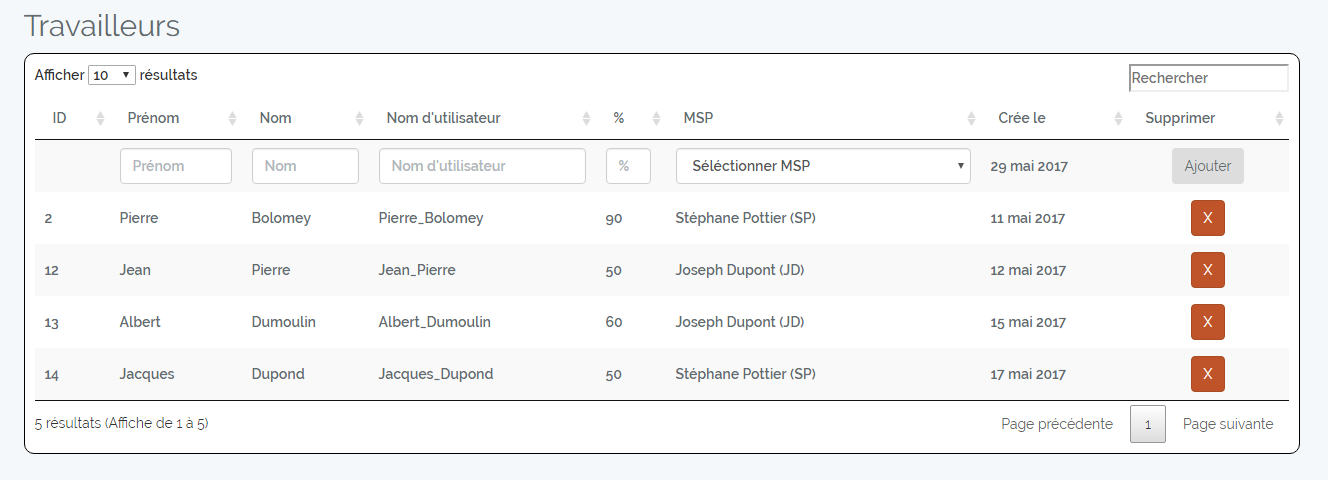


Lien vers le fichier de migration :

<https://github.com/SachaGrenier/TPI/blob/master/database/migrations/2017_05_11_062800_create_all_tables.php>

J’ai choisi de ne pas vous fournir le fichier de migration en annexe car le script SQL qui est déjà présent en soit fait l’affaire. Il est juste sous une autre forme, vous pouvez toujours y accéder avec le lien ci-dessus.

### Travailleurs



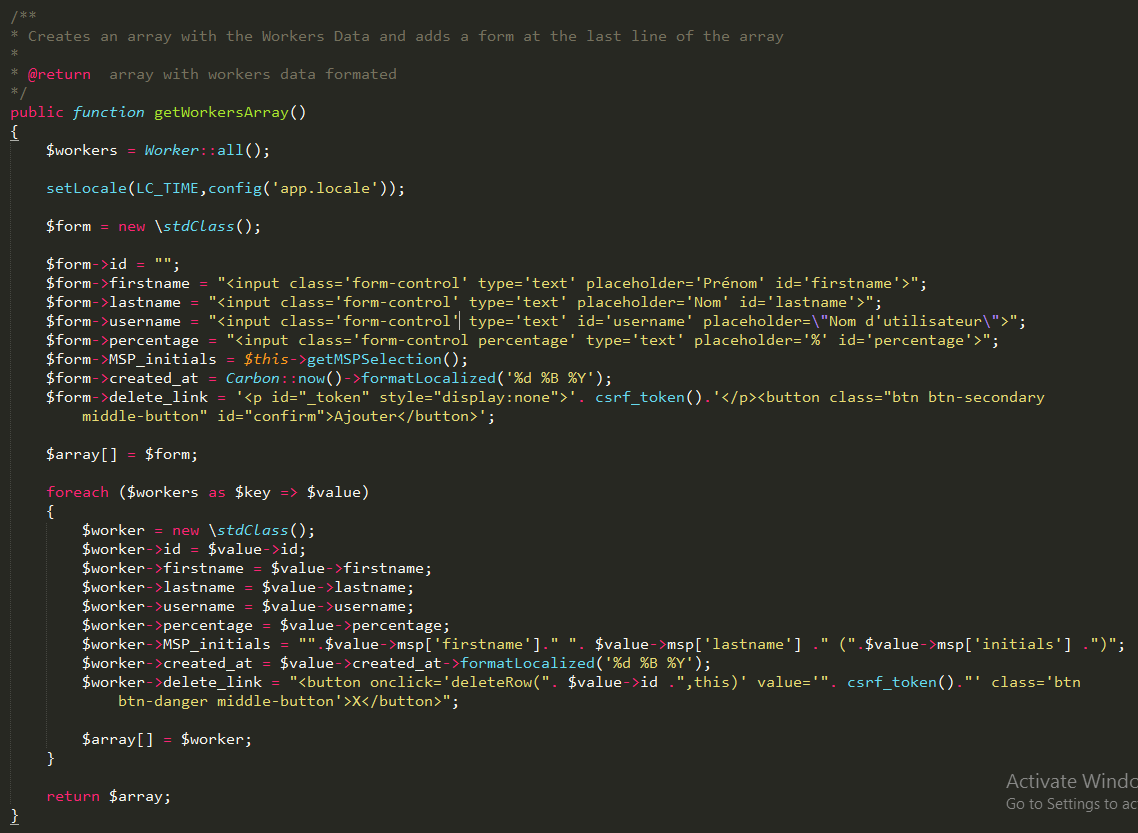
Cette page est séparée en deux tableaux : Le tableau des travailleurs et le tableau des Maîtres socioprofessionnels.

Pour ce qui est du tableau des travailleurs, j’ai utilisé la librairie JavaScript DataTables (<https://datatables.net/> ).

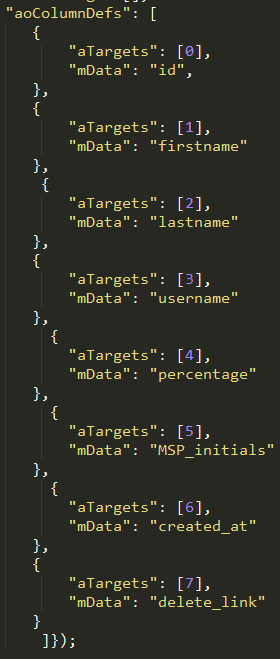
Les données des travailleurs sont récupérées en AJAX à l’aide de la fonction « getWorkersArray ». Cette fonction procède ainsi :

1. Récupération de tous les travailleurs inscrits dans la base de données.
2. Création d’un tableau vide.
3. Utilisation d’une boucle qui parcourt chaque travailleur et qui :
   1. Modifie le format de la date
   2. Modifie la manière dont est exprimé le MSP
   3. Propose un lien sous forme de bouton pour supprimer le travailleur

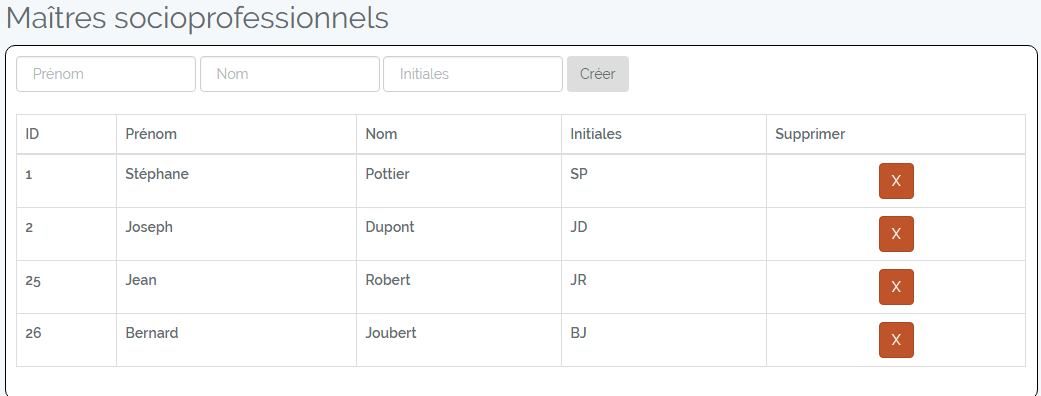
Une fois les objets modifiés, ils sont au fur et à mesure implémentés dans le tableau précédemment vide.



J’ai ajouté une ligne à la fin du tableau qui fait office de formulaire pour ajouter un travailleur.

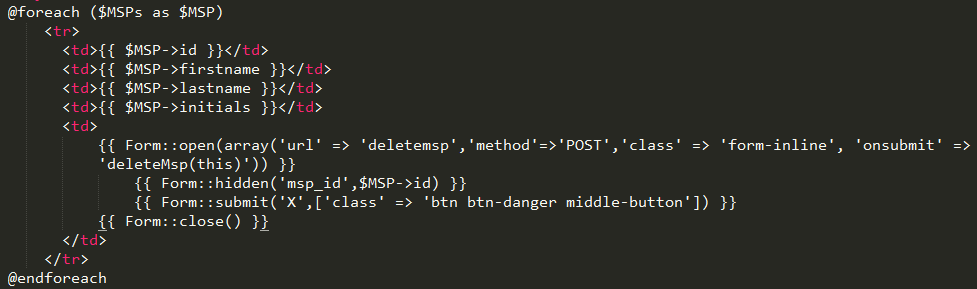
A la fin, le tableau est retourné et j’ai mappé DataTables pour qu’il puisse facilement mettre chaque case dans la bonne colonne. La valeur de « aTargets» définit à quelle colonne appartient la valeur de « mData».

La deuxième partie de la page est utilisée pour afficher le tableau des Maîtres socioprofessionnels.



Etant donné que ce tableau ne va pas beaucoup être utilisé, j’ai préféré utiliser un tableau simple, rempli en PHP et où Les formulaires ne sont pas dynamiques.

Voici comment le tableau des Maîtres socioprofessionnels est affiché :

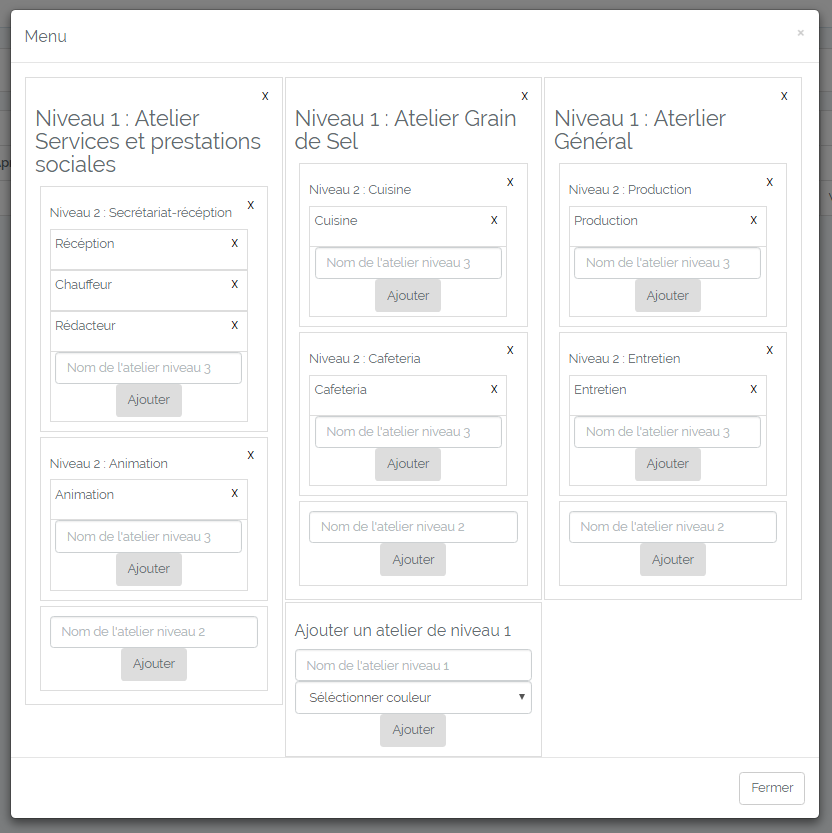


### Menu de gestion de niveaux

Ce menu est disponible sous forme de pop-up suite au clic sur le bouton « Gestion niveaux » présent sur la page de planning. Il représente tous les niveaux de la base de données, présentés sous forme hiérarchique.

Le tableau est généré en PHP et HTML mais est récupéré à l’aide d’AJAX. Etant donné que les formulaires sont aussi envoyés en AJAX, cela permet à l’utilisateur d’effectuer plusieurs actions sur son menu sans devoir rafraichir la page à chaque fois et relancer le menu en appuyant sur le bouton qui permet son affichage.

Chaque atelier, peu importe son niveau, peut être supprimé à l’aide du bouton sous forme de croix. Si l’utilisateur souhaite supprimer un atelier de niveau 2 alors que plusieurs ateliers de niveau 3 y sont attachés, une erreur intervient indiquant que l’opération est impossible. L’utilisateur doit alors d’abord supprimer les ateliers de niveau 3 pour réaliser son action.



Au premier abord ce menu semble un peu confus. Voici comment il faut le comprendre :

Niveau 1 : Ce sont les casiers principaux. Le titre est en grand et à l’intérieur se situent les ateliers concernant les niveaux plus élevés. Le dernier casier est constitué d’un formulaire pour ajouter un atelier de niveau 1. Dans le formulaire, on peut définir un nom d’atelier ainsi qu’une couleur.

Note : La couleur sélectionnée pour l’atelier de niveau 1 sera affichée par la suite

Niveau 2 : Ces casiers se situent directement dans un casier de niveau 1. Dans chaque casier de niveau 1, un formulaire simple avec juste un champ texte est à disposition pour ajouter un atelier de niveau 2.

Niveau 3 : Les casiers de niveau 3 représente directement les ateliers. Ils sont à l’intérieur des casiers de niveau 2. Comme pour le niveau 2, un formulaire simple est à disposition dans chaque casier de niveau 2 pour ajouter un atelier.

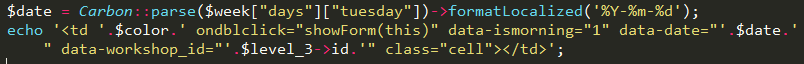
### Planning général

Voici le planning général en l’état actuel. Les logins des travailleurs sont inscrits dans les cellules pour représenter leur présence à un atelier.

Le tableau est généré en html et PHP, mais une grande partie des fonctions sont en JavaScript (jQuery) et Ajax. Il m’a paru important d’avoir un outil dynamique mais cela a rendu la tâche plus compliquée.

La base du tableau est générée en PHP et HTML. Pour ce faire, j’ai combiné 3 foreach (un par niveau). Dans chaque boucle, je récupère la liste des niveaux et avec ceci je crée ma ligne. Au final, le tableau apparait. La difficulté de cette partie du code est que j’ai dû prendre en compte les rowspan. Ceux-ci sont utilisés pour faire en sorte qu’une cellule prenne plus de place sur la hauteur, ce qui fait que la cellule du dessous ne doit pas exister. J’ai utilisé les rowspan pour le niveau 1 et 2.

Les cellules du tableau sont toutes générées de la même manière, c’est-à-dire avec des métadonnées mais toutes vides:



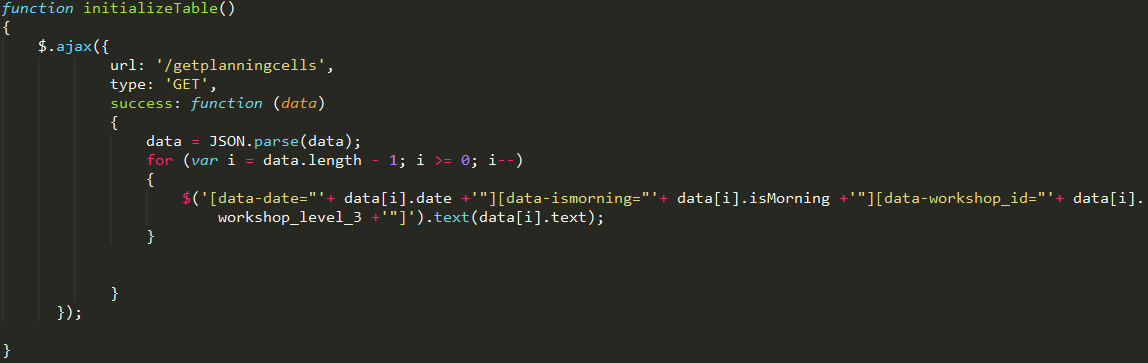
$date -> récupère la date du tableau $week, généré lors de la création de la vue et la formate.

$color -> couleur définie au début de la boucle foreach, utilisée dans toutes les cellules de la même ligne.

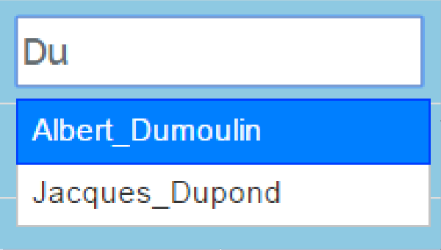
Ondblclick -> Evènement javascipt qui s’active lorsque la case est double cliquée par l’utilisateur. La fonction showForm() permet d’y afficher le champ pour ajouter un travailleur à la tâche.

Les métadonnées sont : ismorning,date et workshop\_id.

Pour les remplir avec les noms d’utilisateurs des travailleurs, j’ai créé une fonction qui récupère les travailleurs assignés à des tâches et qui les inscrit dans les cases, grâce aux métadonnées :

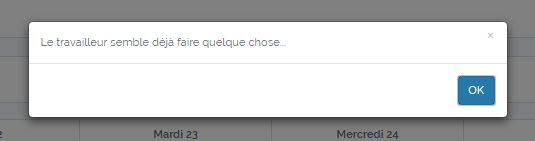


Pour assigner un travailleur à une tâche, un double clic sur une case fera apparaître un champ texte qui est en fait un « autocomplete ». Cela signifie qu’à partir du moment où certaines lettres que l’utilisateur à rentrées correspondent au prénom ou au nom d’un travailleur, il est proposé dans une liste :

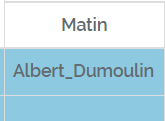


Pour valider la séléction, un simple clic sur le choix du travailleur fait office d’envoi de requête. A partir de ce moment, la fonction qui enregistre le travailleur dans l’atelier vérifie d’abord que le travailleur ne soit pas occupé à une autre tâche à ce moment la.

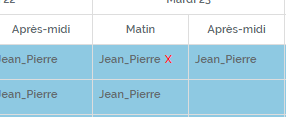
Dans ce cas la, un message d’erreur apparaît pour indiquer que l’utilisateur fait fausse route :



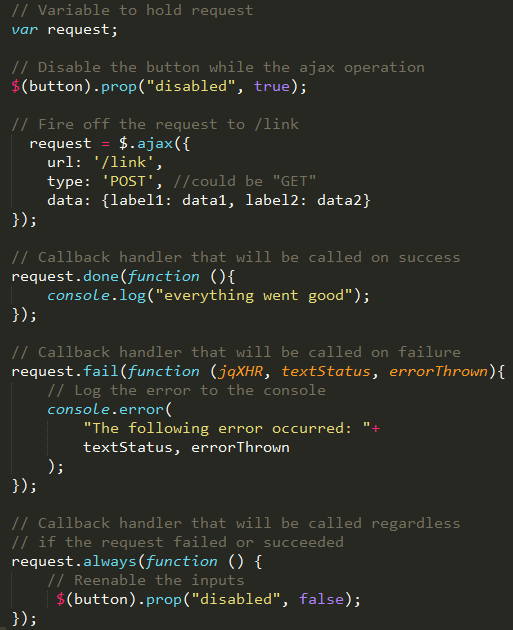
Si la requête est accéptée, le tableau se met à jour avec les nouvelles valeurs et le travailleur apparaît alors dans la table :



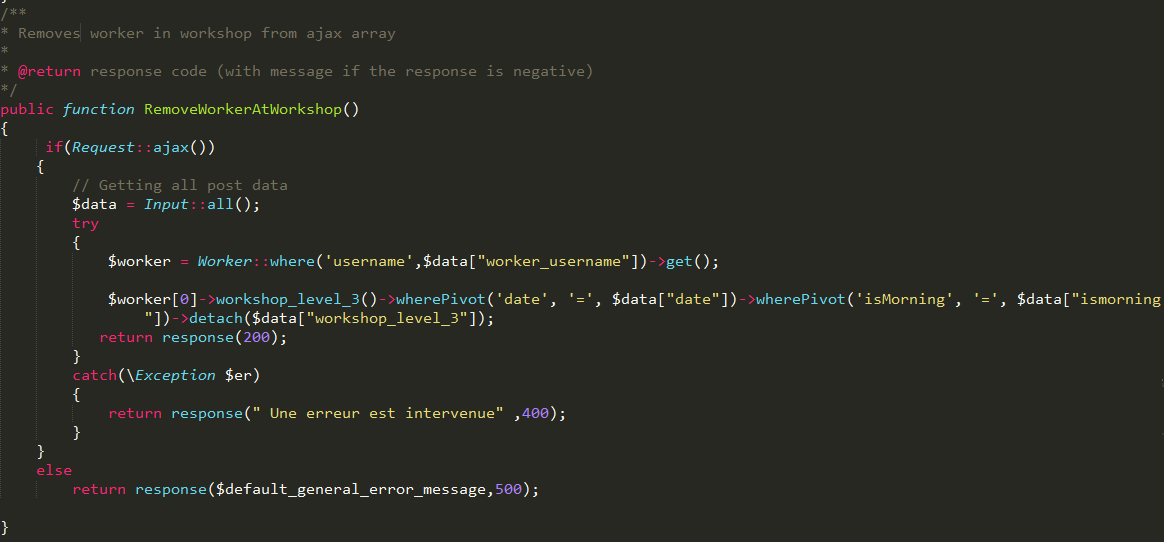
Il est aussi possible de supprimer un travailleur assigné à un atelier. Pour ce faire, lorsque l’utilisateur passe sa souris sur les cellules, une croix apparaît à coté des logins lors que ceux-ci sont inscrits :



Comme dit précédemment, les interactions avec la base de données sont effectuées à l’aide de AJAX. Voici le modèle de base que j’utilise pour effectuer mes requêtes :



Voici pour exemple de fonction qui traite une requête AJAX la fonction que j’ai utilisée pour supprimer une tâche de la base de données :



Ici, la fonction vérifie d’abord que des données envoyées ont bien étés récupérées.

Ensuite, les données sont enregistrées dans le tableau « $data », où les valeurs sont accessibles à l’aide de leurs clés. Toutes les interactions sont dans un « try catch », pour pouvoir fournir une erreur à l’utilisateur si les données qui ont été envoyés étaient incorrectes.

Dans tous les cas, une réponse http est envoyée à l’utilisateur suite à la requête. A savoir que si la réponse est positive (code 200), il n’aura pas de notification contrairement à une réponse négative (code 400 ou 500).

Le bouton tout en bas à gauche du planning permet de transformer le planning actuellement affiché sous forme de PDF. Lors de clic, le fichier PDF est généré et directement affiché dans le navigateur. Evidemment, si ce dernier ne peut pas lire les fichiers PDF, le fichier est téléchargé.

**Bugs connus :**

Lors du développement de mon projet j’ai remarqué quelques bugs. Voici une liste qui normalement est exhaustive :

* Responsivité ; il semble que le planning sous sa forme actuelle ne soit pas responsif. Il reste utilisable avec un petit format. Je n’ai pas pu tester s’il fonctionnait depuis un smartphone.

J’estime 3-4 jours supplémentaires nécessaires pour que le planning soit parfaitement responsif et utilisable sur smartphone (si ce n’est pas déjà le cas).

* Ajout-suppression de lignes ; En soit, l’ajout et la suppression des lignes ajoutées fonctionnent. Le problème est que à partir du moment où l’utilisateur souhaite utiliser la ligne ajoutée en y insérant un travailleur, la ligne du dessus se fait overwrite.

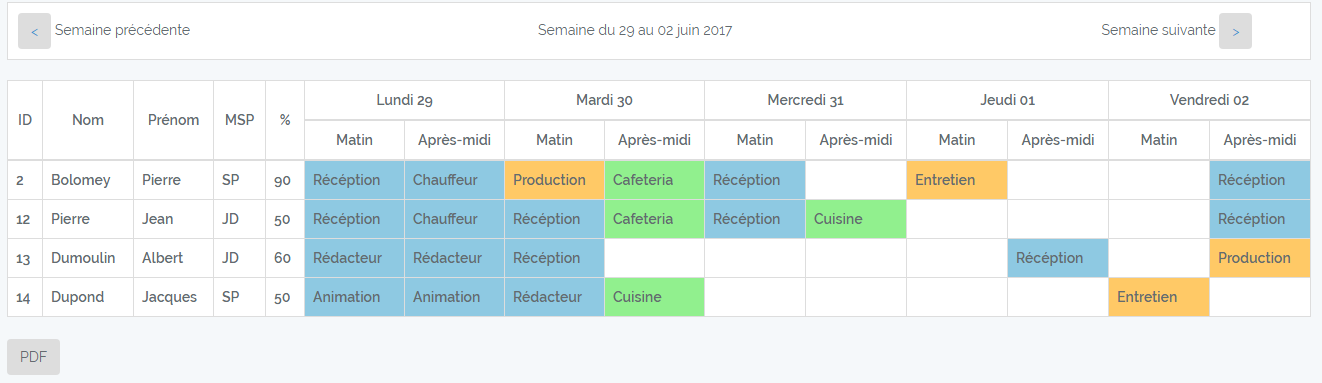
J’estime une semaine pour :

* + Faire en sorte que le travailleur inséré n’overwrite pas.
  + Le tableau puisse être généré avec les lignes qui ont été ajoutées.
  + Les lignes supprimables qui se font supprimées suppriment les tâches de la base de données.
* Historique modifiable ; C’était une fonction que je souhaitais implémenter. Les semaines précédentes de la semaine actuelles n’étaient pas sensées pouvoir être modifiées. Actuellement, le planning est modifiable peu importe la semaine.

J’estime 2 jours pour que le planning s’affiche en mode historique.

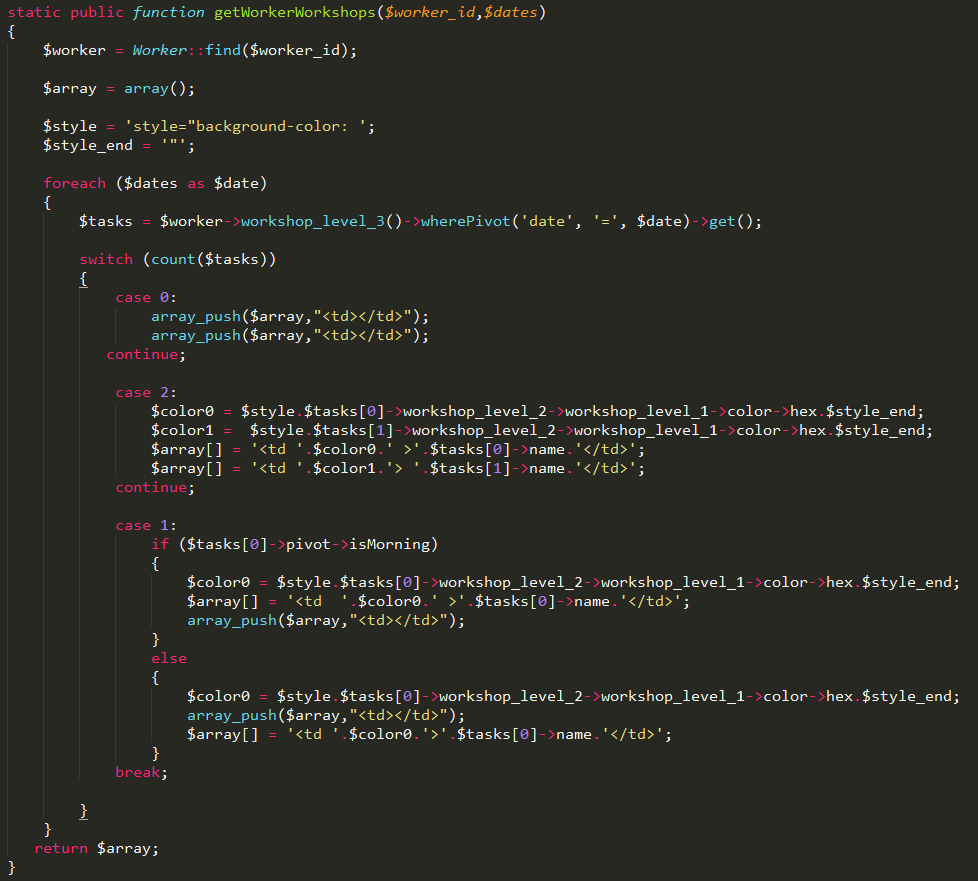
### Planning des travailleurs

Le planning des travailleurs se présente sous cette forme. Il est possible de parcourir les semaines pour y afficher le planning. Rien n’est modifiable de ce côté.



La fonction ci-dessous est utilisée pour récupérer le tableau de cellules des ateliers par travailleur, avec la couleur de l’atelier en fond.

Le paramètre $worker\_id sert à récupérer les ateliers effectués par le travailleur en utilisant son ID et $dates à fournir la plage de dates du planning. Le foreach parcourt les dates et insère à chaque fois deux cellules, pleines ou vides dans le tableau. La difficulté ici a été de gérer si la tâche était durant le matin ou l’après-midi et d’y insérer une case vide au bon endroit (switch).



## Modifications

### Modifications à faire

* Afficher la couleur de l’atelier de niveau 1 dans le menu de niveaux (15 min).
* Dans le planning, mettre en place un autofocus sur le champs texte qui apparait lors du double clic sur une cellule (15 min).
* Dans le planning, faire en sorte que la croix qui apparait quand l’utilisateur passe sa souris sur une cellule ne décale pas tout le planning (2h).
* Lors de la génération de planning sous format PDF, masquer la colonne action (15 min).
* Modifier le style général des fichier PDF générés (1h).
* Calculer le pourcentage effectif sur le planning des travailleurs (1h).

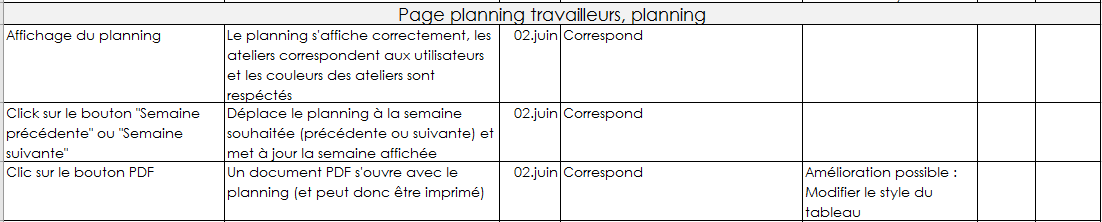
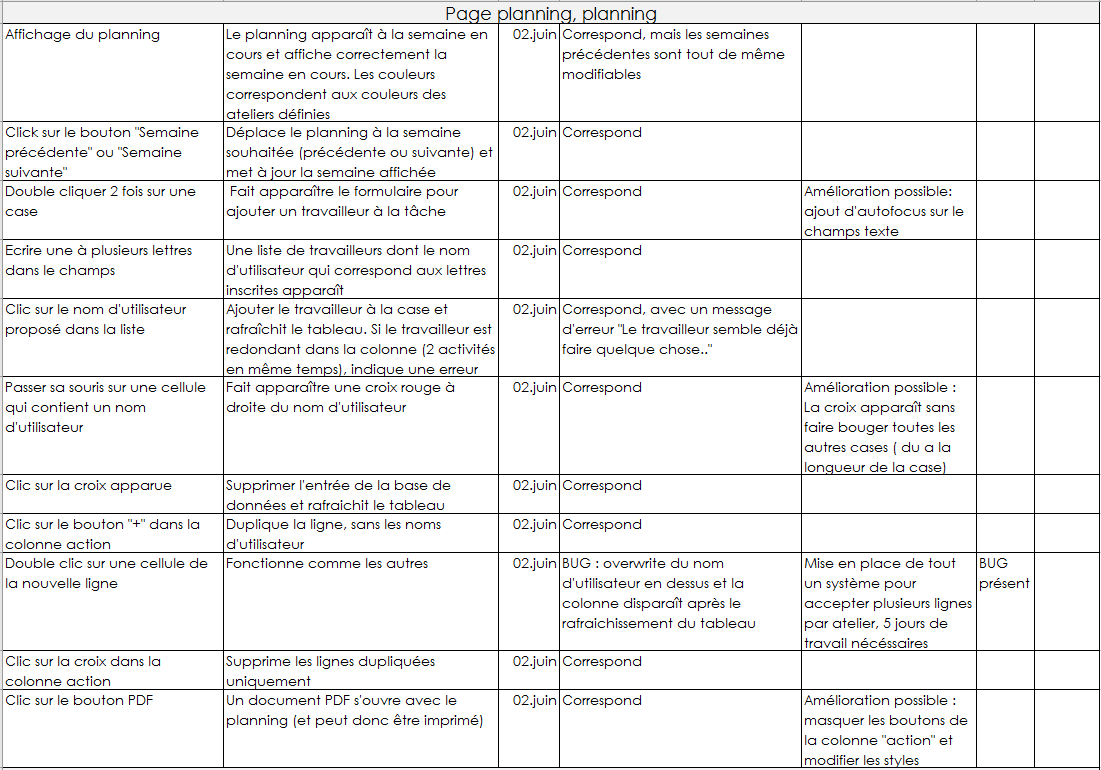
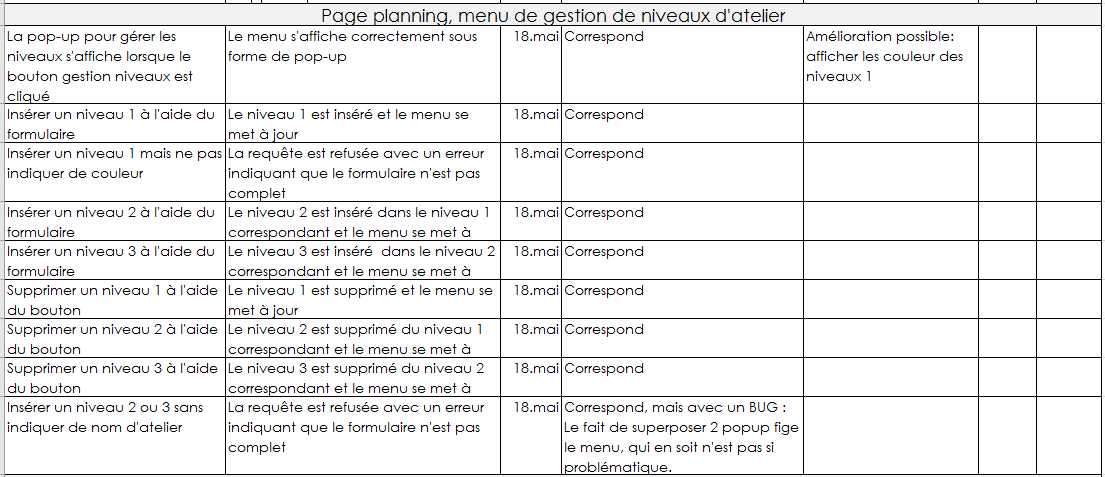
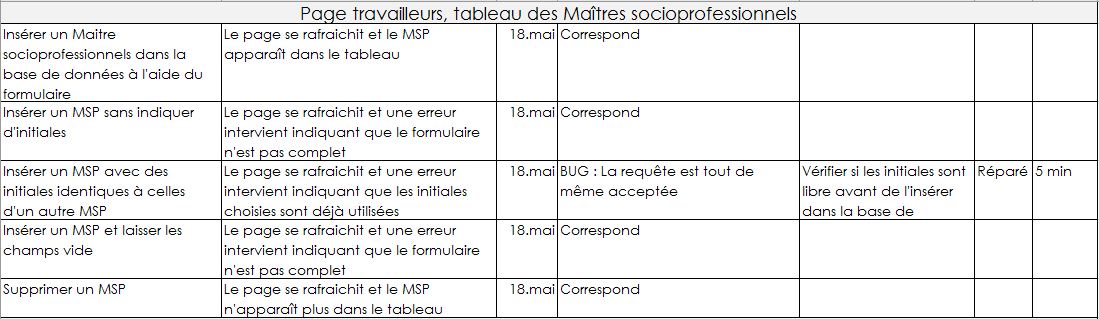
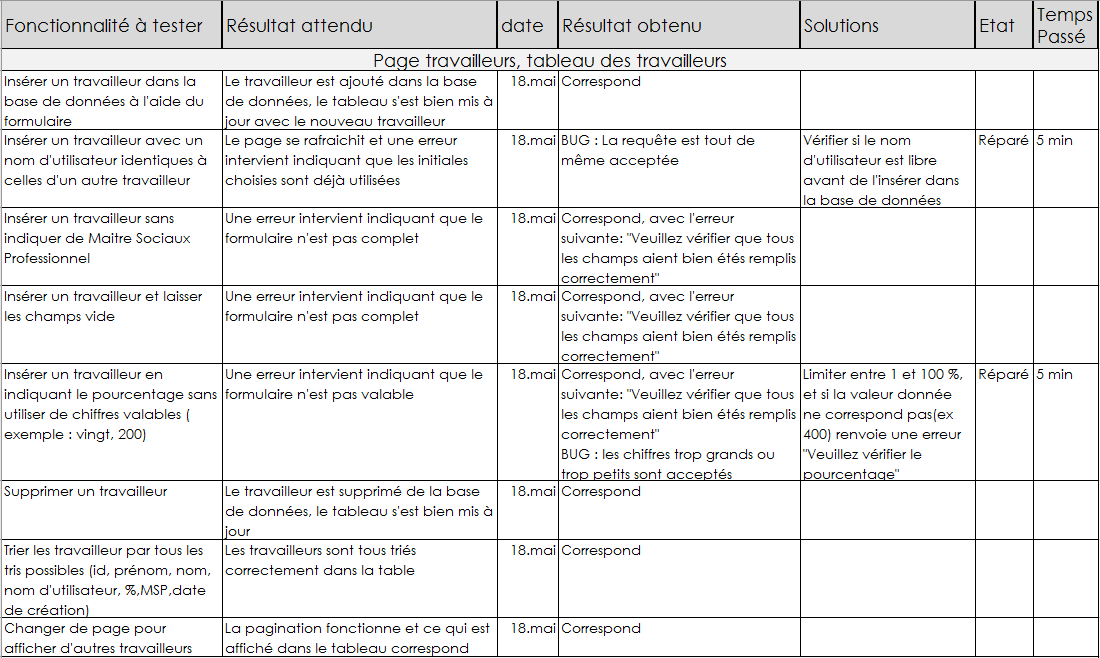
### Modifications effectuées

* Dans le tableau des travailleurs, j’ai créé une fonction qui vérifie si le nom d'utilisateur du travailleur est libre avant de l'insérer dans la base de données (5 min)
* Dans le formulaire pour ajouter un travailleur, j’ai limité entre 1 et 100 % le champ de pourcentage. Une fois la requête envoyée, si la valeur donnée ne correspond pas (ex 400) renvoie une erreur "Veuillez vérifier le pourcentage" (5 min).

# Tests

## Dossier des tests

Les tests ont été effectués en suivant la grille suivante :



# Conclusion

## Bilan des fonctionnalités demandées

Dans l’ensemble le projet est tout de même bien avancé. Il est vrai qu’une fonctionnalité majeure est manquante et que des améliorations sont possibles. Le problème principal pour moi a été le temps. J’ai perdu pas mal de temps avec les fonctions qui affichent le planning, car c’était d’une grande difficulté. J’ai aussi perdu un peu de temps avec la première librairie que je voulais utiliser pour générer les PDF, mais j’ai pu trouver une alternative et faire fonctionner le tout, avec difficulté.

J’ai réussi à rendre mon application générique, mais il est nécessaire d’y avoir 3 niveaux.

Rendre le site responsif était une partie facile grâce à Bootstrap. Mais pour le planning, c’était une autre histoire. Etant donné qu’il a été fait main, même avec les styles Bootstrap je n’ai pas réussi à le rendre responsif à cause du temps.

J’ai aussi trouvé que l’utilisation de GIT était très utile. J’ai pu constater mon avancement dans le projet grâce aux commits et pu vérifier mon code ajouté et supprimé avant les push.

Pour terminer mon projet j’estimerais une durée supplémentaire nécessaire de deux petites semaine.

## Bilan de la planification

Je pense avoir suivi correctement le planning. Je remarque que j’ai notamment perdu du temps sur l’analyse et sur le développement du planning.

## Bilan personnel

Si le projet était à refaire, j’aurais peut-être essayé d’utiliser une librairie pour le planning. Au moins une librairie qui m’aurais permis de gérer les rowspan plus facilement. Je dois avouer que les tableaux html sont très casse-cou.

Aussi, j’ai voulu rendre mon application le plus dynamique possible. Pour ce faire, j’ai utilisé pas mal de JavaScript. Ce projet m’a beaucoup apporté au niveau de jQuery & AJAX. Je connaissais un peu comment cela fonctionnait avant de comment mon projet mais j’ai pu en apprendre beaucoup. C’est en tous cas un bon point pour moi.

Pour moi, le projet est suffisamment bien commenté pour être repris et terminé par quelqu’un d’autre (ou même moi-même).

# Divers

## Journal de travail

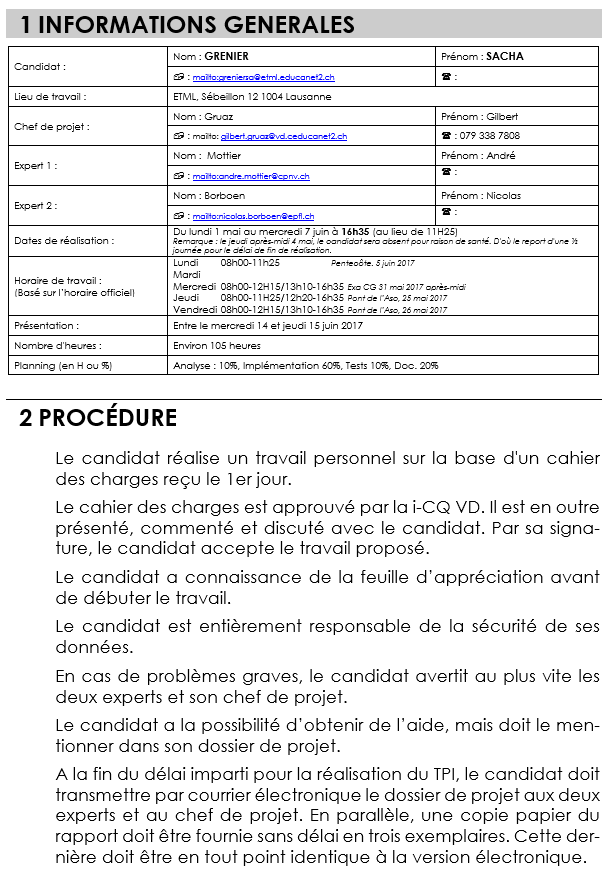
[Voir chapitre 8.3](#_Journal_de_travail)

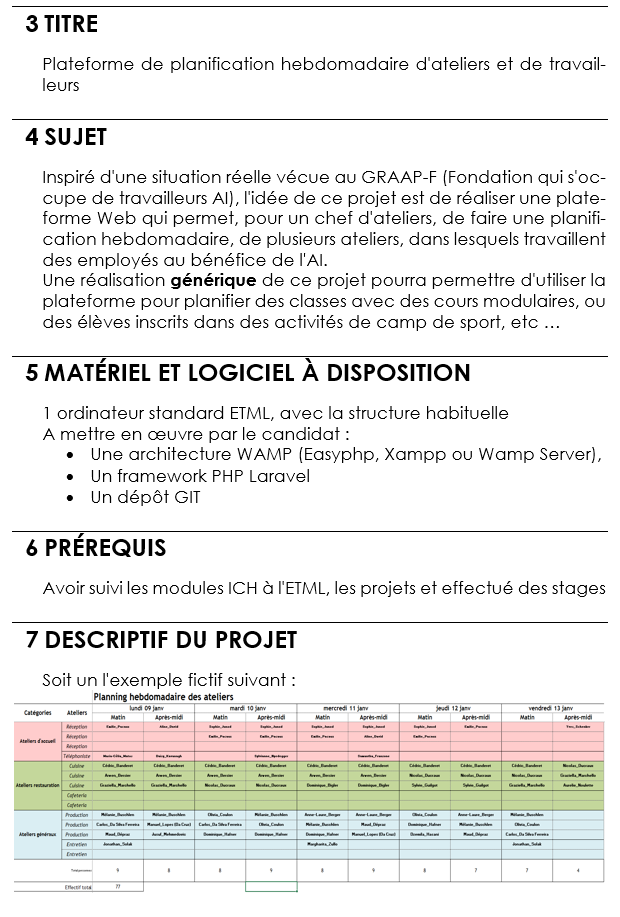
## Webographie

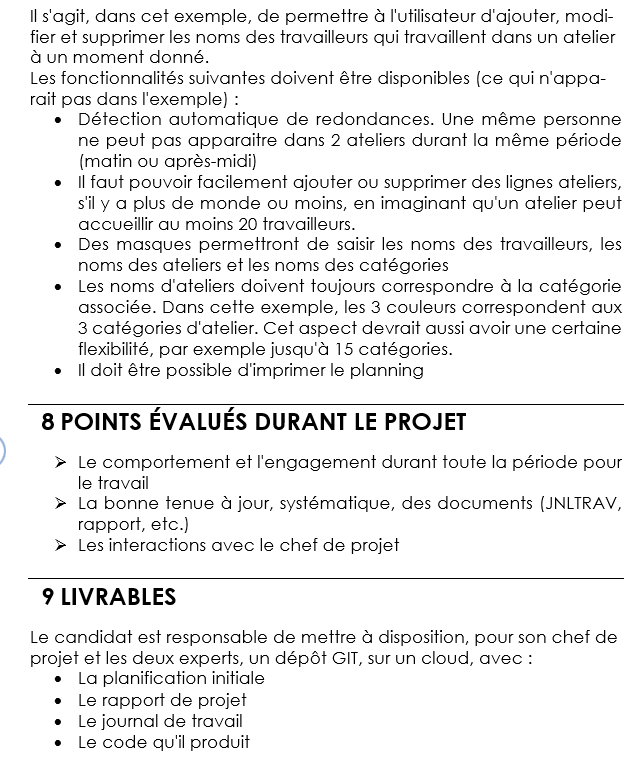
* Dépôt du projet  : <https://github.com/SachaGrenier/TPI>
* Laravel : <https://laravel.com/docs/5.4>
* Wkhtmltopdf : <https://github.com/barryvdh/laravel-snappy>
* Dompdf : <https://github.com/barryvdh/laravel-dompdf>
* Datatables : <https://datatables.net>
* jMerise : <http://www.jfreesoft.com/JMerise/>
* Librairie laravel HTML : <https://laravelcollective.com/docs/5.2/html>

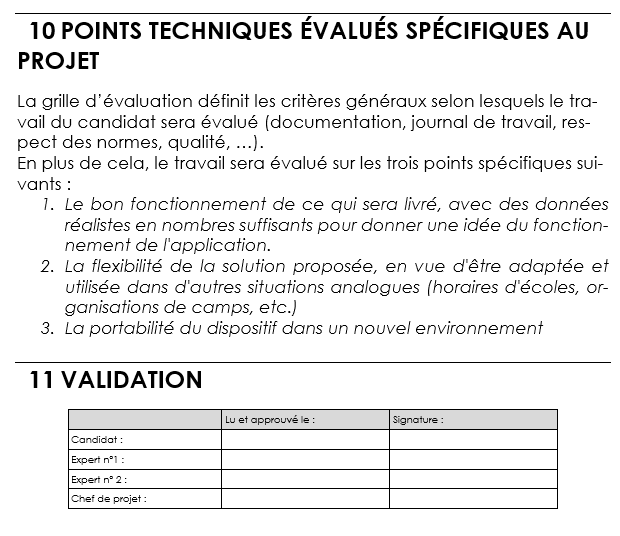
# Annexes

## Cahier de charges









## Script SQL obtenu avec l’application jMerise

#------------------------------------------------------------

# Script MySQL.

#------------------------------------------------------------

#------------------------------------------------------------

# Table: t\_atelier\_niv3

#------------------------------------------------------------

CREATE TABLE t\_atelier\_niv3(

idAtelierNiv3 int (11) Auto\_increment NOT NULL ,

ateNom Varchar (25) NOT NULL ,

idAtelierNiv2 Int NOT NULL ,

PRIMARY KEY (idAtelierNiv3 )

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: t\_atelier\_niv2

#------------------------------------------------------------

CREATE TABLE t\_atelier\_niv2(

idAtelierNiv2 int (11) Auto\_increment NOT NULL ,

ateNom Varchar (25) NOT NULL ,

idAtelierNiveau1 Int NOT NULL ,

PRIMARY KEY (idAtelierNiv2 )

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: t\_travailleur

#------------------------------------------------------------

CREATE TABLE t\_travailleur(

idTravailleur int (11) Auto\_increment NOT NULL ,

traNom Varchar (25) NOT NULL ,

traPrenom Varchar (25) NOT NULL ,

traPourcentage Int NOT NULL ,

idMSP Int NOT NULL ,

PRIMARY KEY (idTravailleur )

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: t\_atelier\_niv1

#------------------------------------------------------------

CREATE TABLE t\_atelier\_niv1(

idAtelierNiveau1 int (11) Auto\_increment NOT NULL ,

ateNom Varchar (25) NOT NULL ,

idCouleur Int NOT NULL ,

PRIMARY KEY (idAtelierNiveau1 )

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: t\_couleur

#------------------------------------------------------------

CREATE TABLE t\_couleur(

idCouleur int (11) Auto\_increment NOT NULL ,

couNomCouleur Varchar (25) NOT NULL ,

couCodeHexadecimal Varchar (40) NOT NULL ,

PRIMARY KEY (idCouleur )

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: t\_MSP

#------------------------------------------------------------

CREATE TABLE t\_MSP(

idMSP int (11) Auto\_increment NOT NULL ,

mspNom Varchar (25) NOT NULL ,

mspPrenom Varchar (25) NOT NULL ,

mspInitiales Varchar (4) NOT NULL ,

PRIMARY KEY (idMSP )

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: t\_tache

#------------------------------------------------------------

CREATE TABLE t\_tache(

taDate Date ,

taEstMatin Bool ,

idAtelierNiv3 Int NOT NULL ,

idTravailleur Int NOT NULL ,

PRIMARY KEY (idAtelierNiv3 ,idTravailleur )

)ENGINE=InnoDB;

ALTER TABLE t\_atelier\_niv3 ADD CONSTRAINT FK\_t\_atelier\_niv3\_idAtelierNiv2 FOREIGN KEY (idAtelierNiv2) REFERENCES t\_atelier\_niv2(idAtelierNiv2);

ALTER TABLE t\_atelier\_niv2 ADD CONSTRAINT FK\_t\_atelier\_niv2\_idAtelierNiveau1 FOREIGN KEY (idAtelierNiveau1) REFERENCES t\_atelier\_niv1(idAtelierNiveau1);

ALTER TABLE t\_travailleur ADD CONSTRAINT FK\_t\_travailleur\_idMSP FOREIGN KEY (idMSP) REFERENCES t\_MSP(idMSP);

ALTER TABLE t\_atelier\_niv1 ADD CONSTRAINT FK\_t\_atelier\_niv1\_idCouleur FOREIGN KEY (idCouleur) REFERENCES t\_couleur(idCouleur);

ALTER TABLE t\_tache ADD CONSTRAINT FK\_t\_tache\_idAtelierNiv3 FOREIGN KEY (idAtelierNiv3) REFERENCES t\_atelier\_niv3(idAtelierNiv3);

ALTER TABLE t\_tache ADD CONSTRAINT FK\_t\_tache\_idTravailleur FOREIGN KEY (idTravailleur) REFERENCES t\_travailleur(idTravailleur);

## Journal de travailH:\TPI\screens\JDT\w1.PNG

