Planning web



Sacha Grenier – Cin4b

ETML - Lausanne

Durée : 105 heures

Chef de projet : Gruaz Gilbert

Experts : Mottier André, Borboen Nicolas

Table des matières

[1 Spécifications 4](#_Toc484093842)

[1.1 Titre 4](#_Toc484093843)

[1.2 Description 4](#_Toc484093844)

[1.3 Matériel et logiciels à disposition 4](#_Toc484093845)

[1.4 Prérequis 4](#_Toc484093846)

[1.5 Cahier des charges 4](#_Toc484093847)

[2 Planification Initiale 5](#_Toc484093848)

[3 Analyse 6](#_Toc484093849)

[3.1 Opportunités 6](#_Toc484093850)

[3.2 Document d’analyse et conception 6](#_Toc484093851)

[3.2.1 Page principale 7](#_Toc484093852)

[3.2.2 La page Planning des utilisateurs 9](#_Toc484093853)

[3.2.3 Page travailleurs 9](#_Toc484093854)

[3.2.4 Base de données 10](#_Toc484093855)

[3.3 Conception des tests 14](#_Toc484093856)

[3.4 Planification détaillée 14](#_Toc484093857)

[4 Dossier de Réalisation 16](#_Toc484093858)

[4.1.1 Mise en place de l’environnement 16](#_Toc484093859)

[4.1.2 Fichier de migration 17](#_Toc484093860)

[4.1.3 Page « Travailleurs » 18](#_Toc484093861)

[4.1.4 Menu de gestion de niveaux 20](#_Toc484093862)

[4.1.5 Planning général 22](#_Toc484093863)

[4.2 Modifications 24](#_Toc484093864)

[5 Tests 25](#_Toc484093865)

[5.1 Dossier des tests 25](#_Toc484093866)

[6 Conclusion 27](#_Toc484093867)

[6.1 Bilan des fonctionnalités demandées 27](#_Toc484093868)

[6.2 Bilan de la planification 27](#_Toc484093869)

[6.3 Bilan personnel 27](#_Toc484093870)

[7 Divers 27](#_Toc484093871)

[7.1 Journal de travail 27](#_Toc484093872)

[7.2 Bibliographie 27](#_Toc484093873)

[7.3 Webographie 27](#_Toc484093874)

[8 Annexes 27](#_Toc484093875)

[8.1 Script SQL obtenu avec l’application jMerise 28](#_Toc484093876)

# Spécifications

## Titre

Plateforme de planification hebdomadaire d’ateliers et de travailleurs

## Description

Inspiré d’une situation réelle vécue au GRAAP-F (fondation qui s’occupe de travailleurs AI), l’idée de ce projet est de réaliser une plateforme Web qui permet, pour un chef d’ateliers, de faire une planification hebdomadaire de plusieurs ateliers dans lesquels travaillent des employés au bénéfice de l’AI.

Une réalisation générique de ce projet permettra d'utiliser la plateforme pour planifier des classes avec des cours modulaires, ou des élèves inscrits dans des activités de camp de sport, etc.

## Matériel et logiciels à disposition

1 ordinateur standard ETML, avec la structure habituelle

A mettre en œuvre par le candidat :

* Une architecture WAMP (Xampp),
* Un framework PHP Laravel
* Un dépôt GIT

## Prérequis

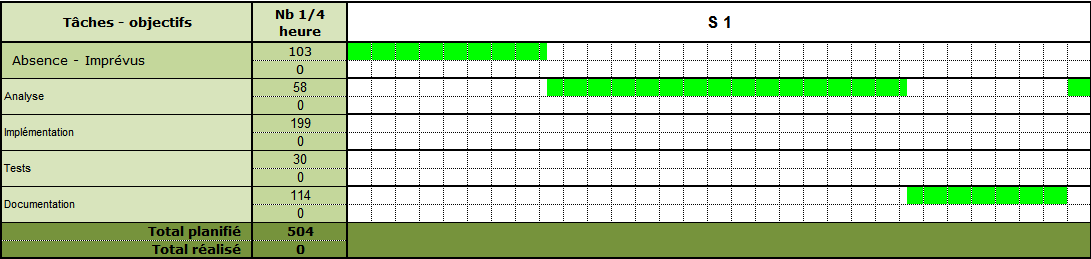
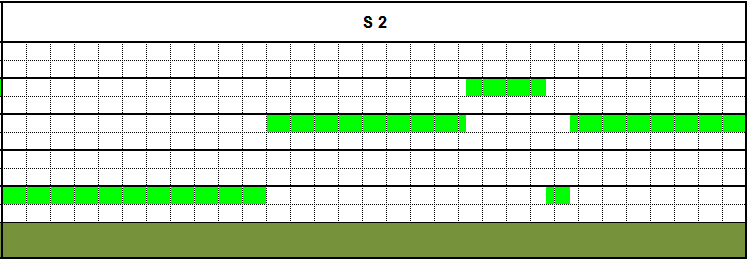
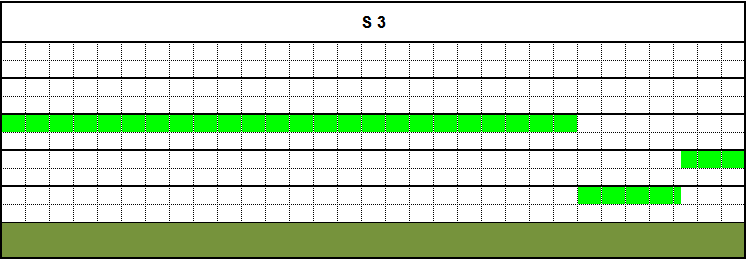
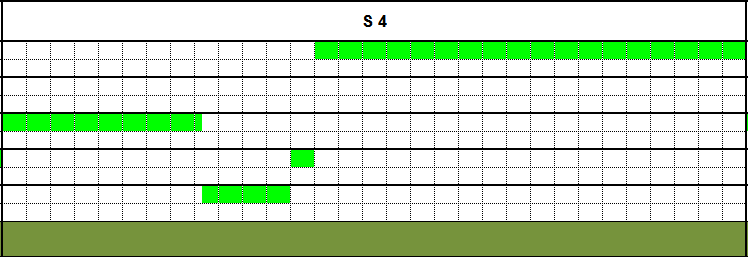
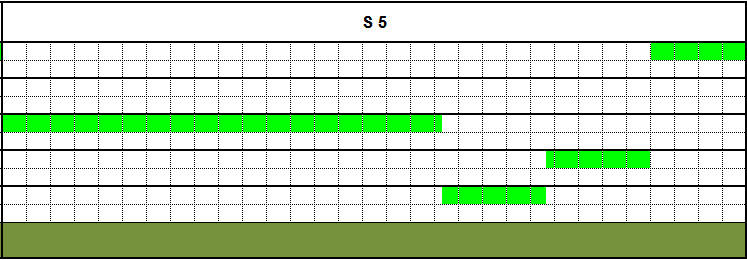
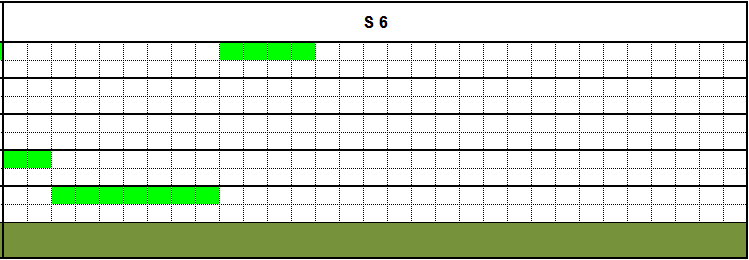
Avoir suivi les modules ICH à l'ETML, les projets et effectué des stages

## Cahier des charges

<https://github.com/SachaGrenier/TPI/blob/master/TPI_files/H-TPI-greniersa-web-Cahier_de_charges.docx>

# Planification Initiale

Voici le Gantt que j’ai obtenu en faisant ma planification initiale :



# Analyse

## Opportunités

La création de ce site web va me permettre de :

* M’améliorer avec le framework Laravel
* M’améliorer en Javascript, jQuery et AJAX
* Créer une grille de planning et la rendre dynamique

Les difficultés seront :

* Le temps à disposition
* Créer la grille sans utiliser de librairie

Ainsi que les solutions possibles

Si les spécifications de départ ne laissent pas de doutes sur la manière de réaliser un projet, ce chapitre ne fera que renvoyer le lecteur aux spécifications.

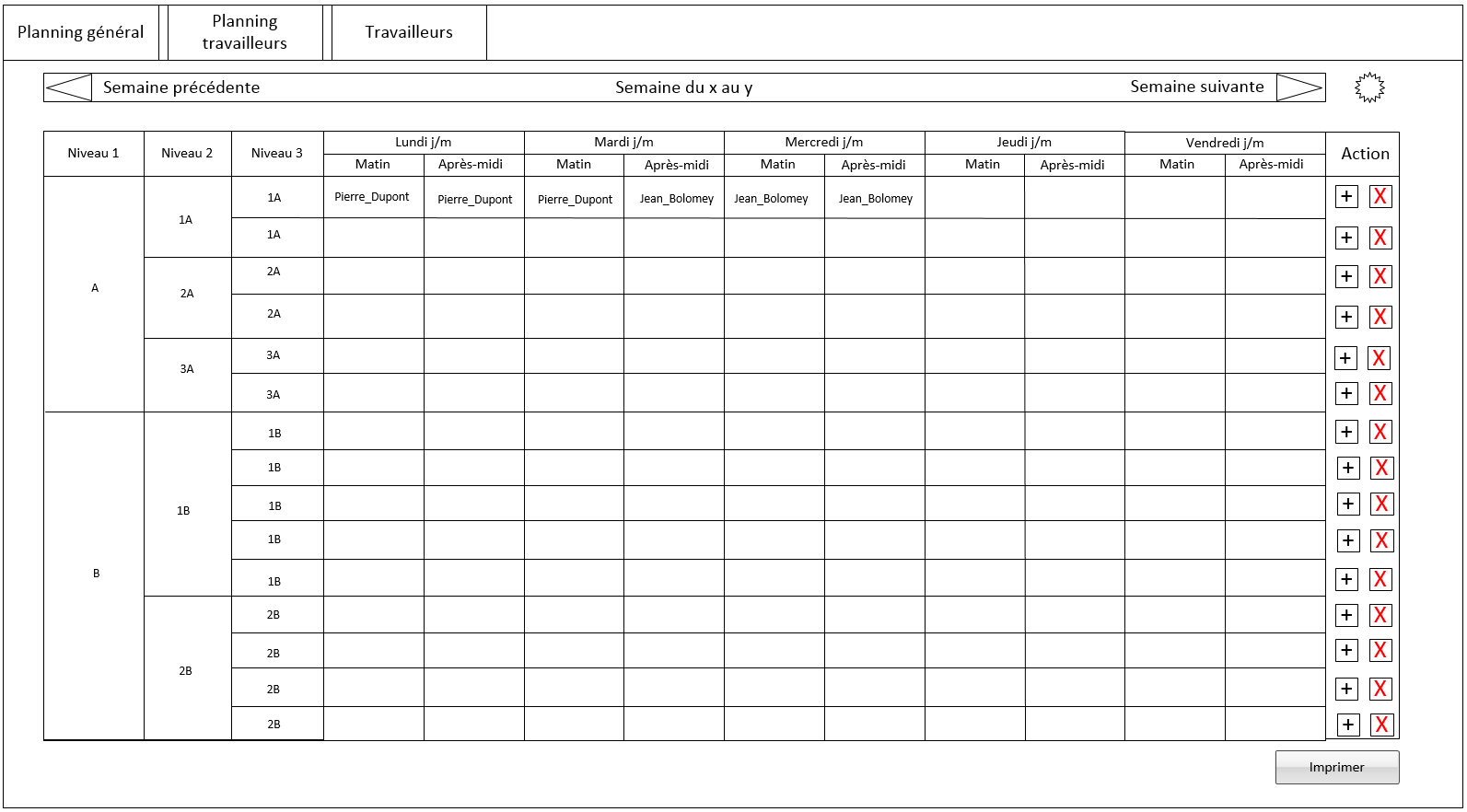
## Document d’analyse et conception

Le but du site est de créer un planning générique qui pourra être utilisé dans plusieurs domaines. Pour me mettre en situation, j’ai choisi le domaine donné dans le cahier de charges qui est le GRAAP-F. L’objectif est donc d’attribuer des tâches à des utilisateurs et de les implémenter dans un planning sous forme de grille hebdomadaire. Il doit être possible d’attribuer des tâches à des utilisateurs lors de la semaine courante, mais aussi lors des semaines suivantes. Les semaines précédentes seront affichées mais ne pourront pas être modifiées, elles feront office « d’historique ».

En ce qui concerne les pages du site, j’ai créé des maquettes concernant à chaque page du site.

### Page principale

Cette page affichera le planning général, avec diverses possibilités d’ajout, suppression et modification de données.



1

2

3

4

5

6

7

8

Description :

1) Menu d’entête, comportant les liens vers les 3 pages du site.

2)Barre de navigation indiquant la semaine actuellement affichée dans la grille, avec deux boutons en forme de triangle pour se déplacer entre les semaines.

3)Boutons + (ajouter) et x (supprimer) qui affecteront les lignes. Normalement ces boutons seront cachés et visibles uniquement lorsque l’utilisateur passe sa souris sur la ligne souhaitée. Le bouton + ajoutera une ligne en dessous et prendra exactement la même catégorie et la même tâche. Le bouton X supprimera la ligne du tableau, avec confirmation de l’utilisateur.

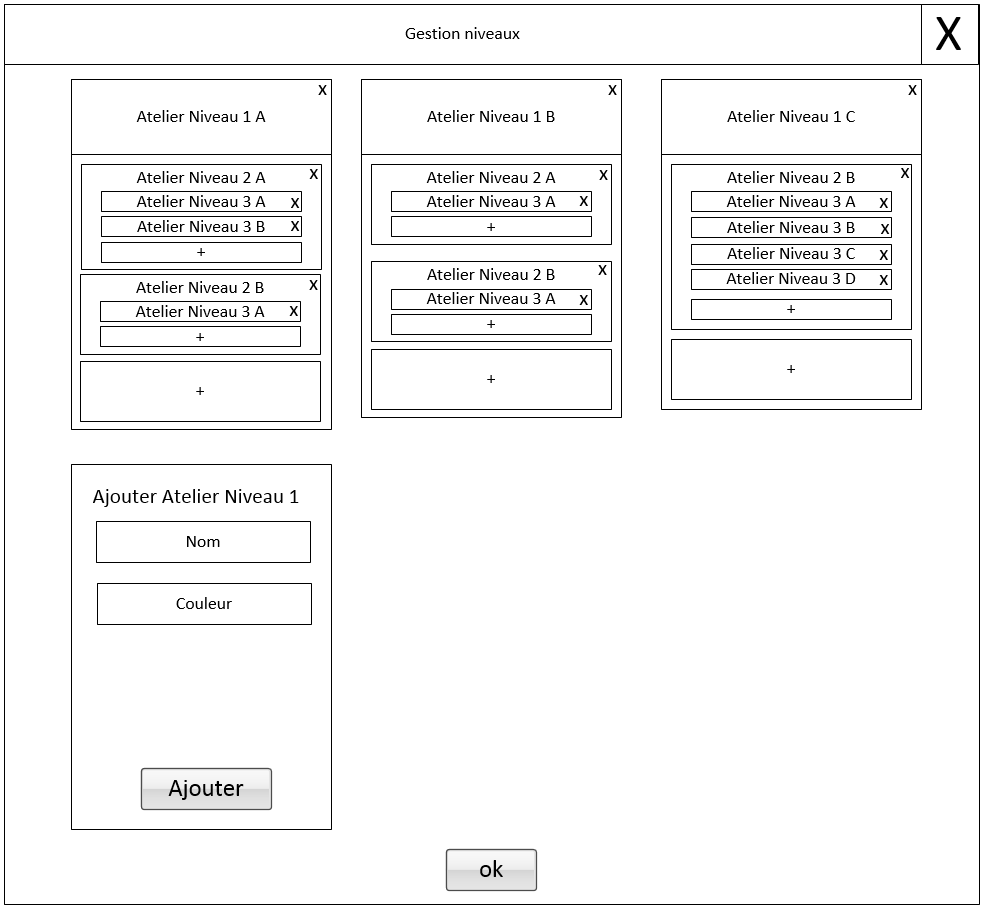
4)Colonne correspondant au Mardi, qui se sépare en Matin et Après-midi.

5)Les catégories et tâches qui en héritent seront affichées ici. Au moins une ligne par tâche sera toujours affichée.

6) Exemple de cellule, correspondant au mardi après-midi. Ici, l’utilisateur pourra cliquer pour afficher champ de recherche, qui une fois complété de quelques lettres proposera les différents utilisateurs correspondants (Autocomplete).

7) Bouton d’impression pour imprimer (imprimer ou enregistrer sous format PDF) la semaine affichée.

8)Bouton qui ouvrira un menu sous forme de pop-up, et qui servira à modifier les catégories et les tâches.

Menu : 

1

4

3

2

1

Description :

1)Boutons pour fermer la pop-up

2)Formulaire pour ajouter un atelier de niveau 1

3)Bouton + pour ajouter un atelier de niveau 2. Une fois le bouton cliqué, un champ texte apparaitra et appuyer sur la touche entrée ajoutera l’atelier.

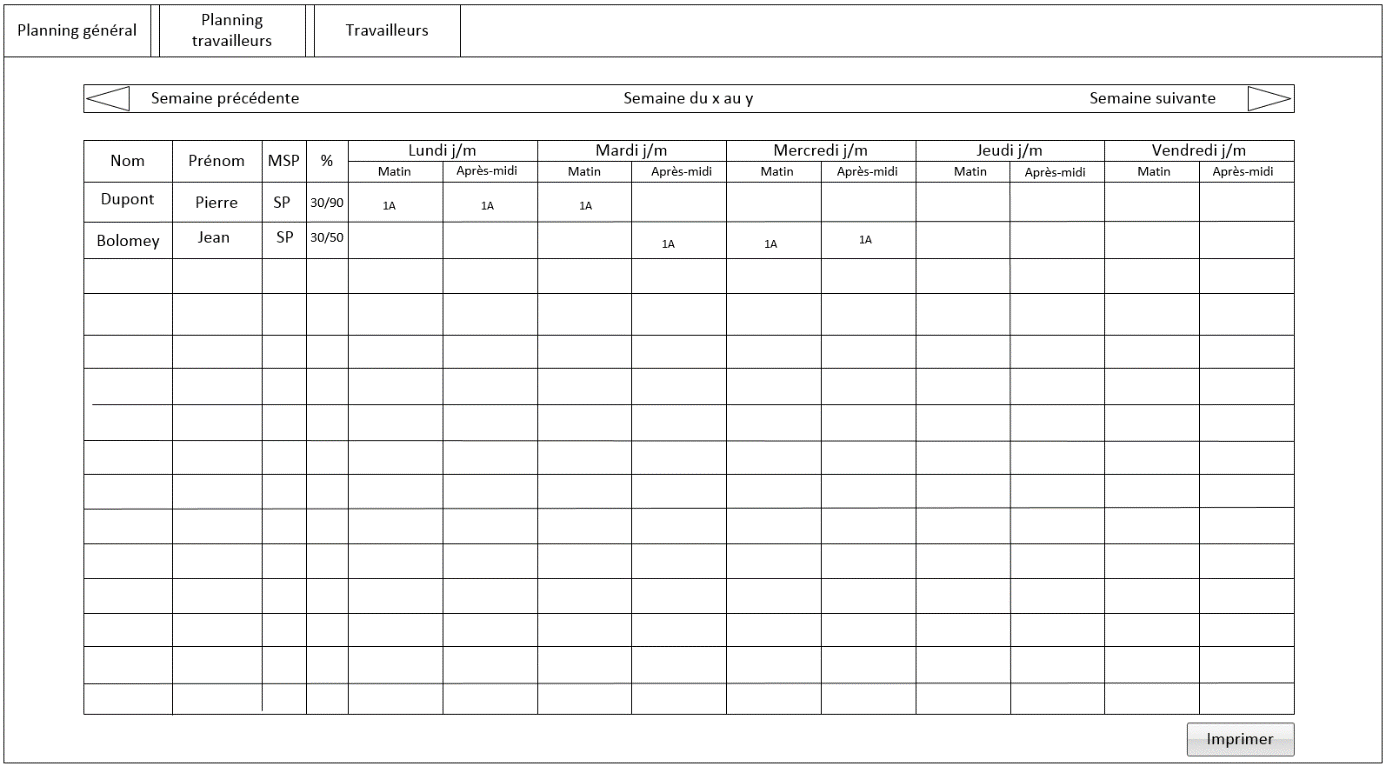
4)Idem que pour le point 3) mais pour ajouter un atelier de niveau 3

Peu importe les niveaux, tous les ateliers seront supprimables à l’aide de la croix qui se situe à droite de chaque atelier. Bien évidemment, si un atelier de niveau 2 est supprimé, tous les ateliers de niveau 3 associés le seront aussi.

### La page Planning des utilisateurs

Cette page sera uniquement en lecture et rien de sera modifiable. Avec les boutons en forme de triangle, il sera possible de se déplacer à travers les semaines et d’y afficher le planning. Elle représente la même grille que pour la page principale mais sous une autre forme. Voici le modèle :

2



4

3

5

1

Description :

1) Barre de navigation indiquant la semaine actuellement affichée dans la grille, avec deux boutons en forme de triangle pour se déplacer entre les semaines.

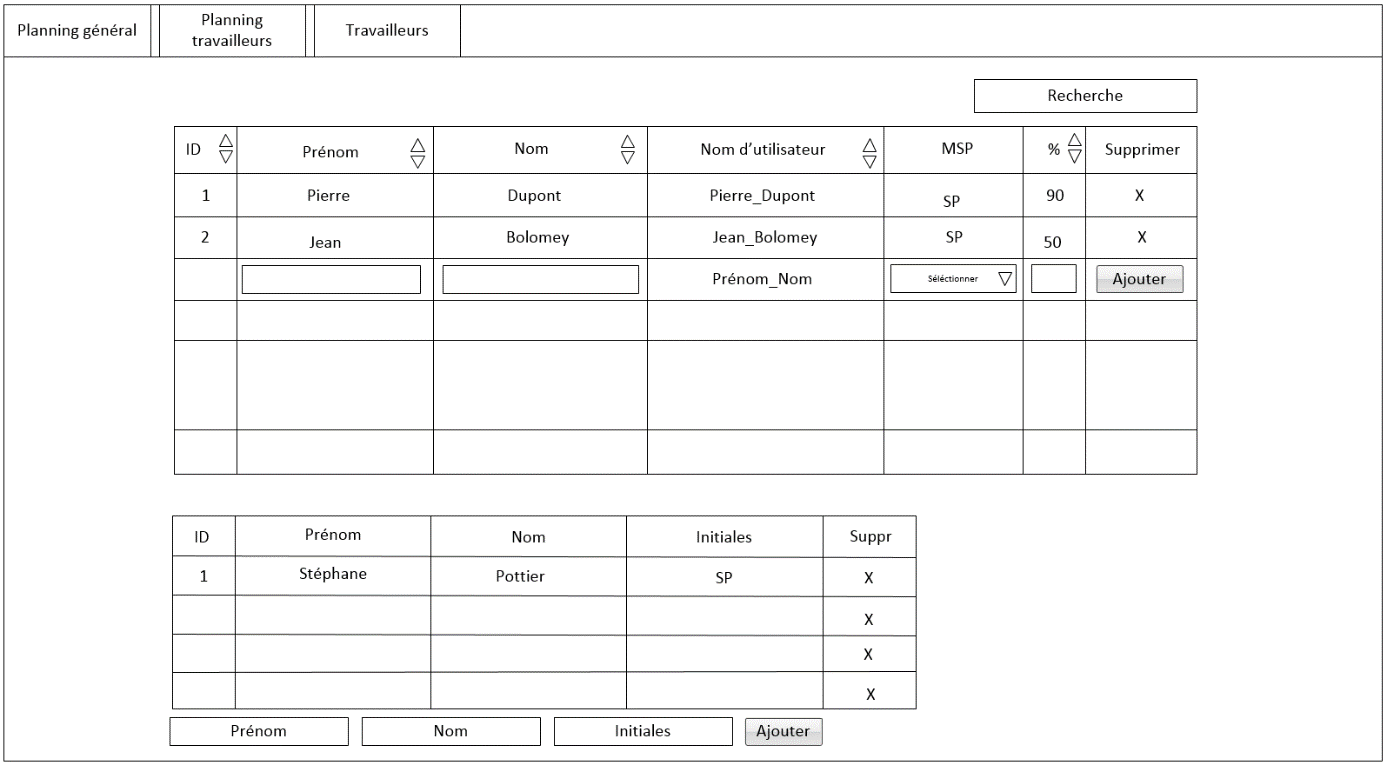
2)Colonnes indiquant le nom et le prénom des utilisateurs

3)Colonne MSP, indiquant le Maitre socioprofessionnel

4)Collonne %, indiquant le pourcentage effectif/prévu   
5) Bouton d’impression pour imprimer (imprimer ou enregistrer sous format PDF) la semaine affichée.

### Page travailleurs

Cette page affichera la liste des travailleurs sous forme de tableau. Pour ce faire, je vais utiliser la libraire DataTables (<https://datatables.net/>) qui propose un tableau dynamique où il est possible de trier les données et qui utilise une pagination.



3

2

1

Description :

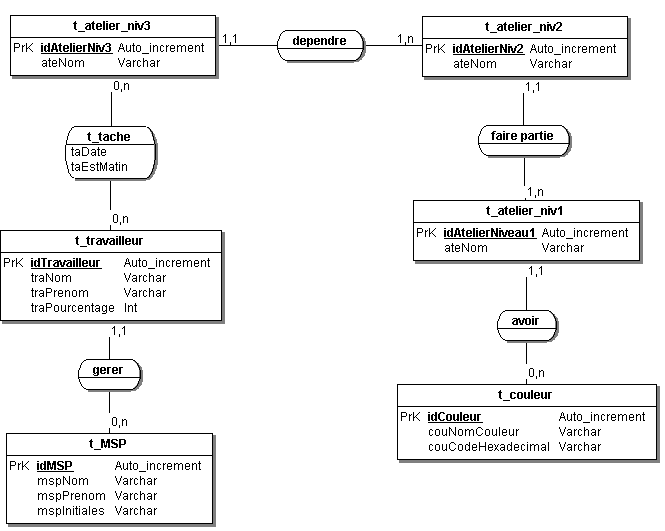
1)Tableau dynamique comportant la liste des travailleurs généré par DataTables. La dernière ligne est sous forme de formulaire pour ajouter un travailleur dans la liste.

2)Liste des Maitres socioprofessionnel, sous forme de tableau simple, sans utiliser DataTables.

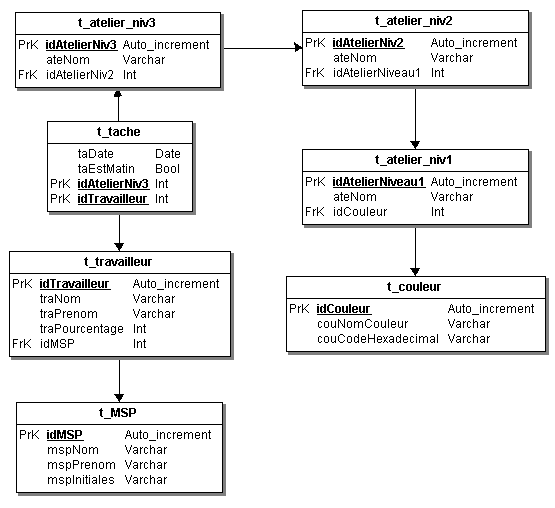
3)Formulaire pour ajouter un MSP à la liste

### Base de données

Pour la base de données, je vais utiliser MySQL. Laravel fourni quelques outils pour aider à la construire, notamment un système de migrations que je vais utiliser pour créer mes tables. Le MCD, MLD et le script SQL suivants ont été réalisés à l’aide de l’application jMerise, que vous pouvez consulter ici : <http://www.jfreesoft.com/JMerise/>. Cet outil permet de créer un MCD qui se transforme ensuite en MLD et en un MPD, sous forme de script SQL. J’ai donc dû nommer mes tables spécifiquement pour que le MLD construit par jMerise corresponde aux normes ETML.

On y retrouve donc 5 tables basiques et une table intermédiaire.   
Voici le MCD : 

Etant donné que les cardinalités entre la table « t\_atelier\_niv3 » et « t\_travailleur » sont de 0,N, j’ai donc prévu la table intermédiaire « t\_tache ». Elle correspond à un atelier X assigné à un travailleur Y.

S’en suit le MLD : 

Suite à ceci l’application jMerise m’a proposé un script SQL pour créer la base de données. Vous le trouverez dans les annexes, ou en cliquant ici : [Script SQL](#_Script_SQL_obtenu)

Et enfin le MPD, dans lequel j’ai modifié le nom des champs pour respecter les conventions du framework Laravel (snake case) et que vous pouvez retrouver ici : <https://laravel.com/docs/5.4/eloquent#eloquent-model-conventions>



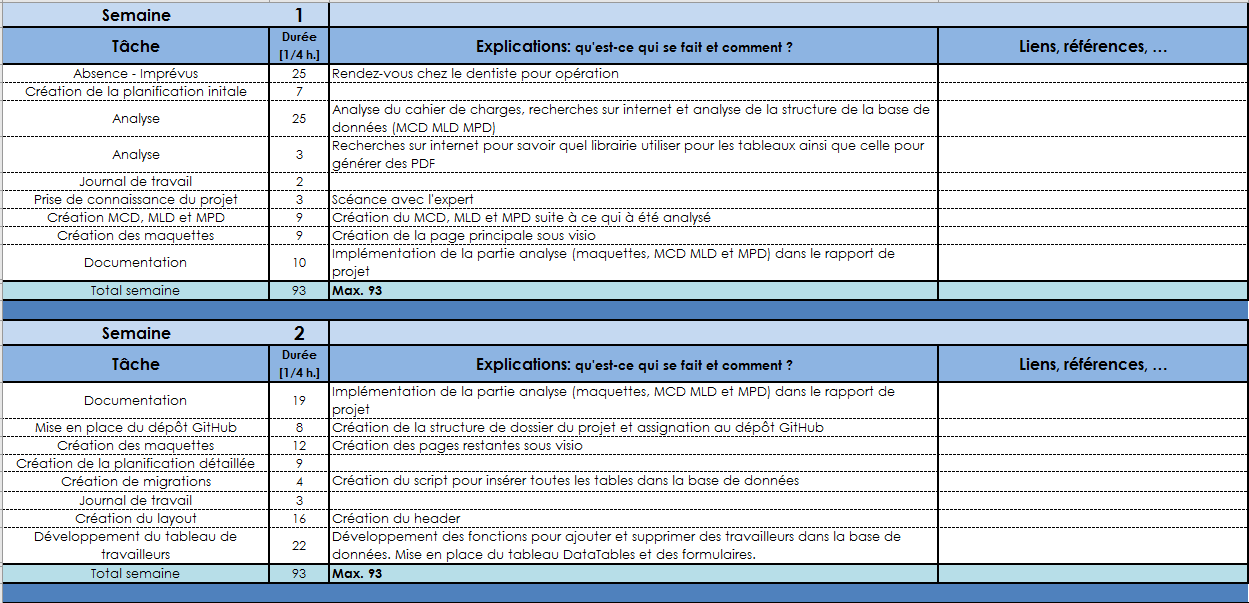
## Conception des tests

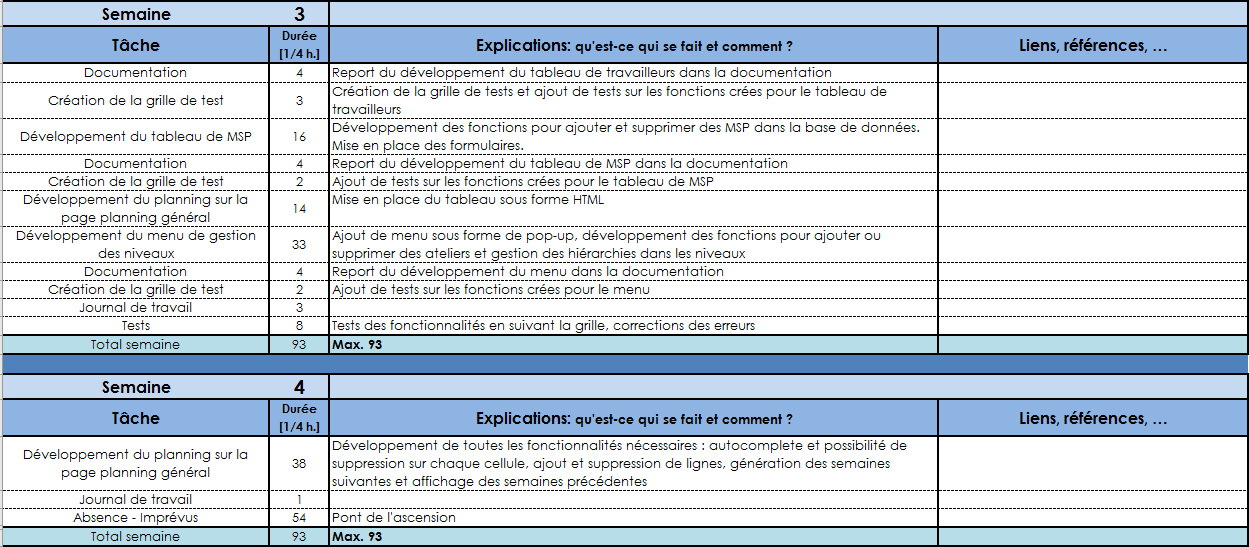
Mes tests vont être effectués suivant une grille que je vais créer et qui aura les colonnes suivantes :

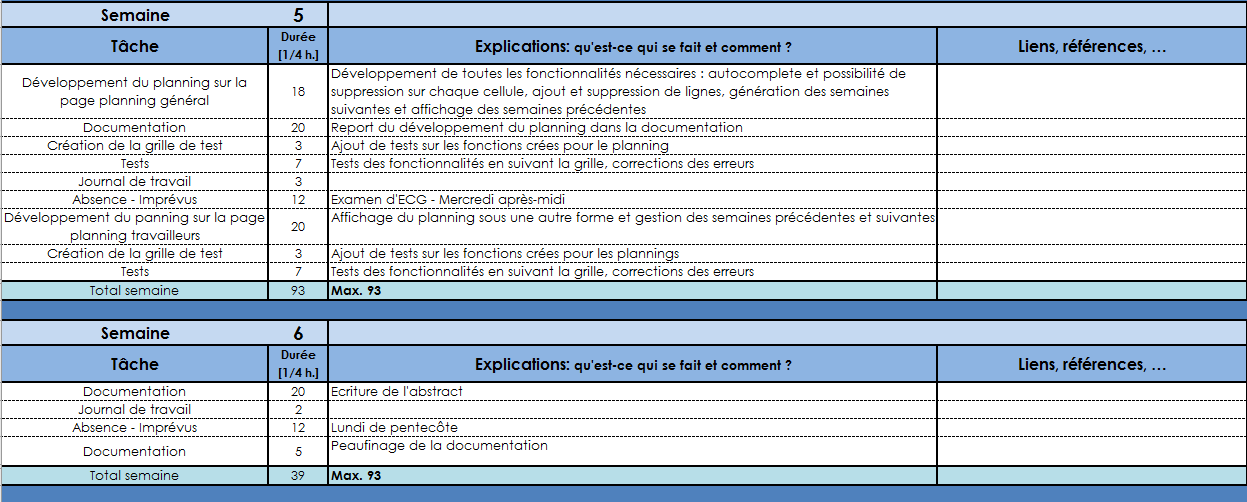
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Fonctionnalité  à tester | Résultat attendu | Date | Résultat obtenu | Solutions | Etat | Temps passé |

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

## Planification détaillée







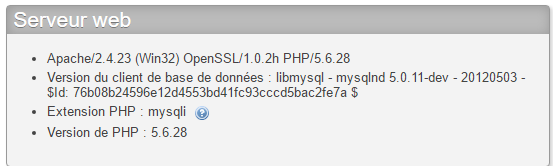
# Dossier de Réalisation

### Mise en place de l’environnement

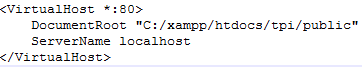
Premièrement, j’ai installé une machine virtuelle avec VMare Workstation Player 12. J’y ai installé Windows 10, mais peu importe la version. Une fois que l’OS a été mis en place et que quelques outils y ont été installés (Notepad++, Sublime Text, Git Extensions), j’ai installé l’outil Composer qui permet 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.

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>

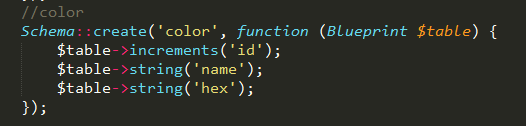
Documentation officielle pour Installer Laravel : <https://laravel.com/docs/5.4>

### 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

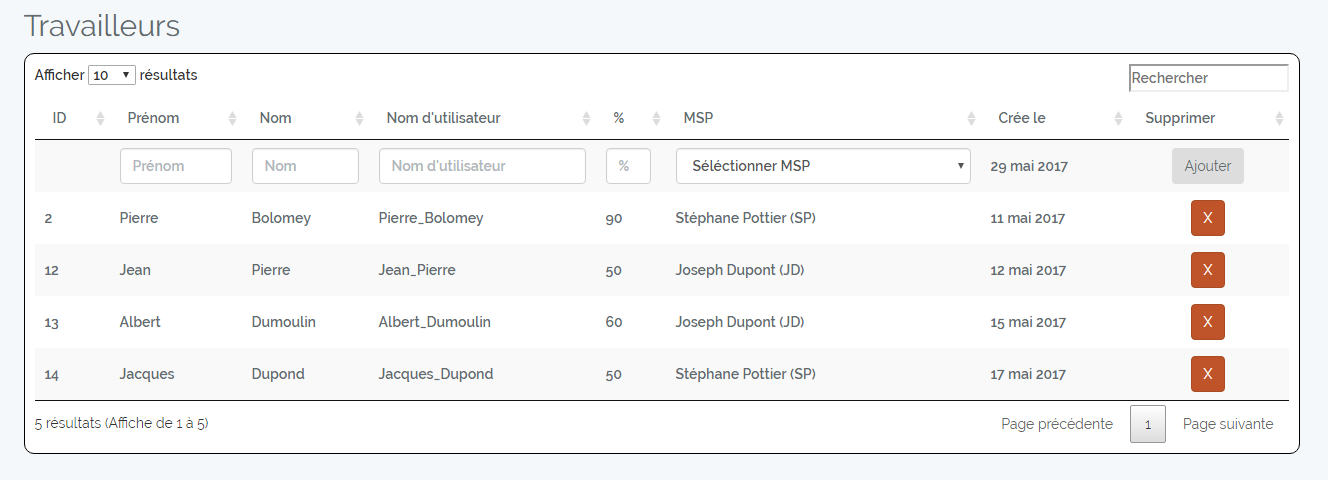
Dans ce cas-là, je donne déjà le nom de ma migration qui est « create \_all\_tables ». Une fois le fichier crée, il suffit de le remplir avec la tables et les champs souhaités. Ci-dessous un exemple de table dans une migration :



Lien vers le fichier de migration :

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

### Travailleurs



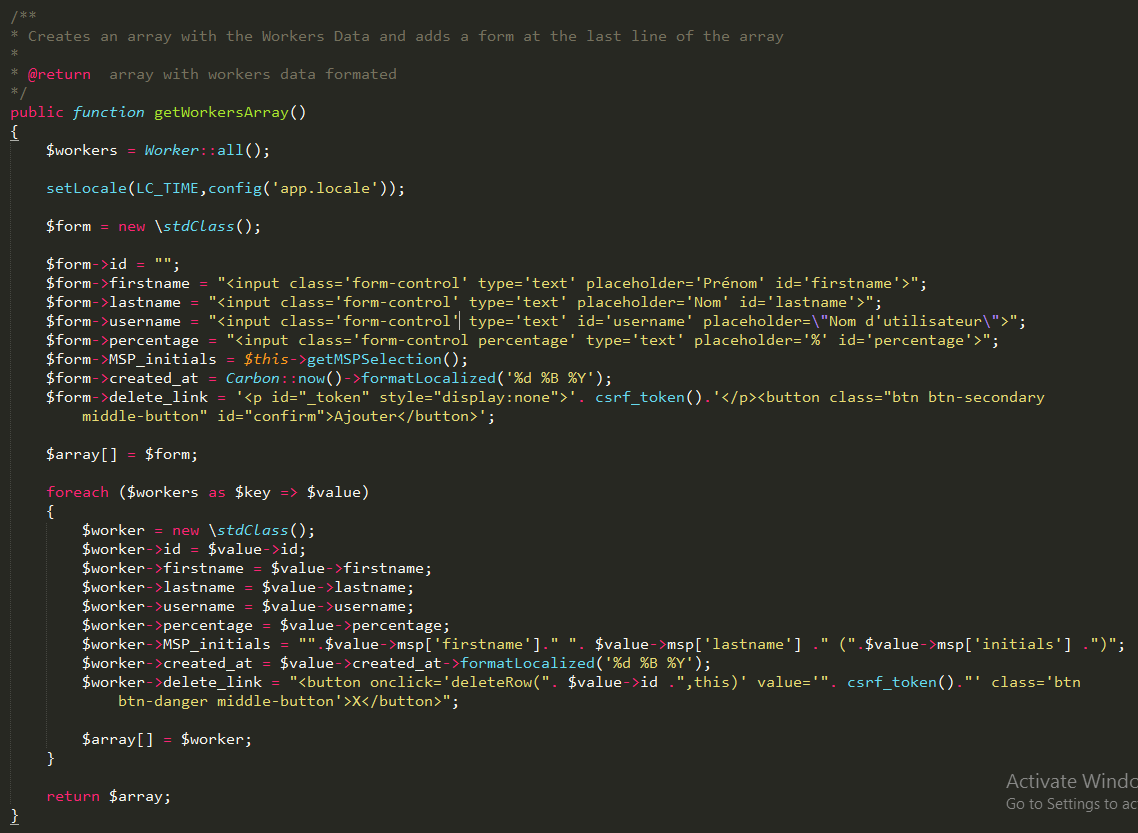
Cette page est séparée en deux tableaux : Le tableau des travailleurs et le tableau des Maîtres socioprofessionnel.

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

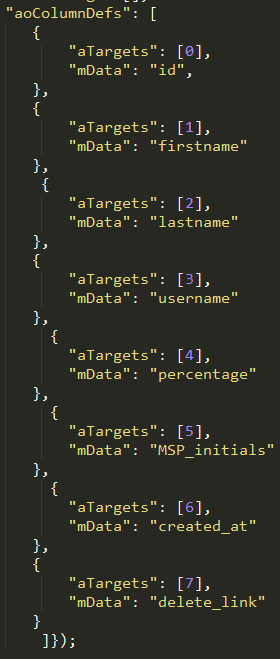
Les données des travailleurs sont récupérées en AJAX à l’aide de la fonction « getWorkersArray ». Cette fonction procède ainsi :

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

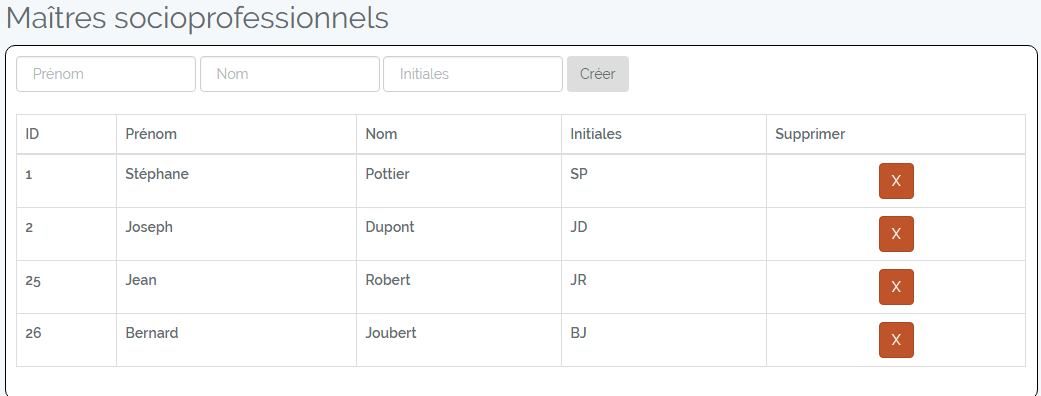
Une fois les objets modifiés, ils sont au fur et à mesure implémentés dans le tableau précédemment vide.



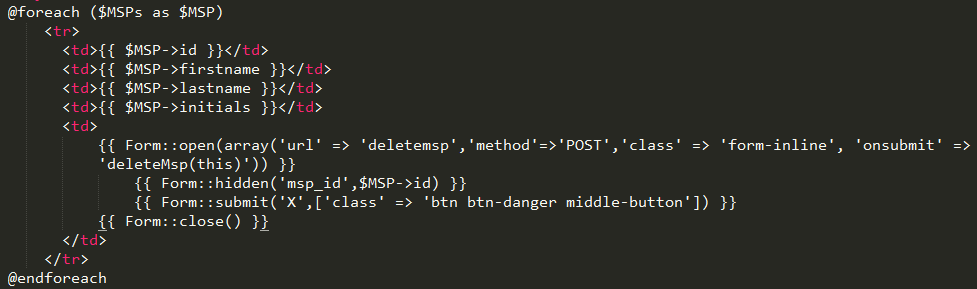
J’ai ajouté une ligne à la fin du tableau qui fait office de formulaire pour ajouter un travailleur.

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.

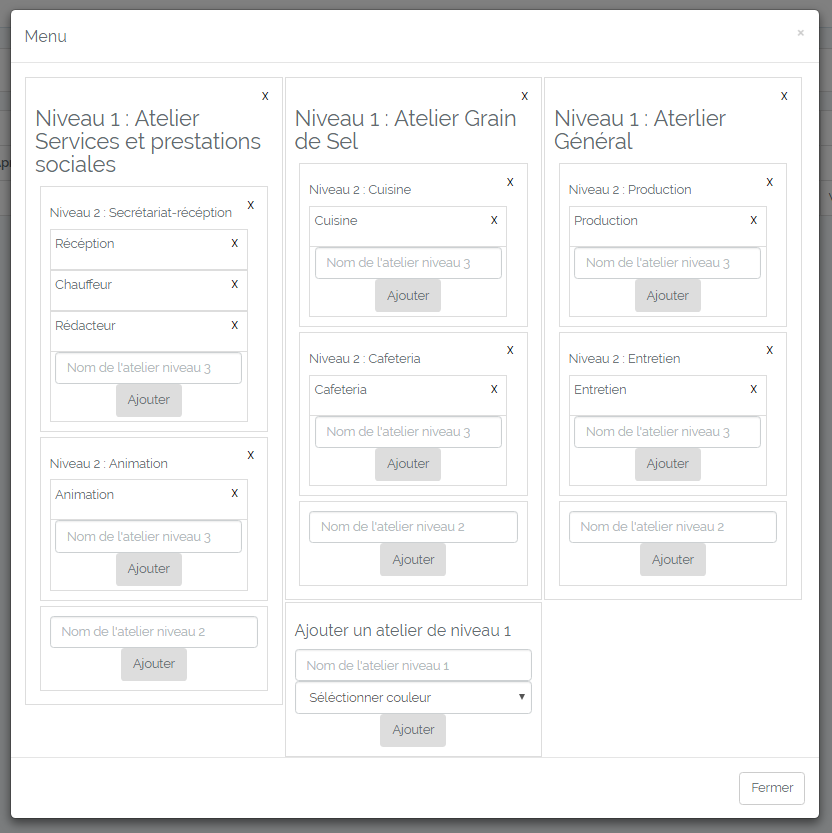


### Menu de gestion de niveaux

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 :

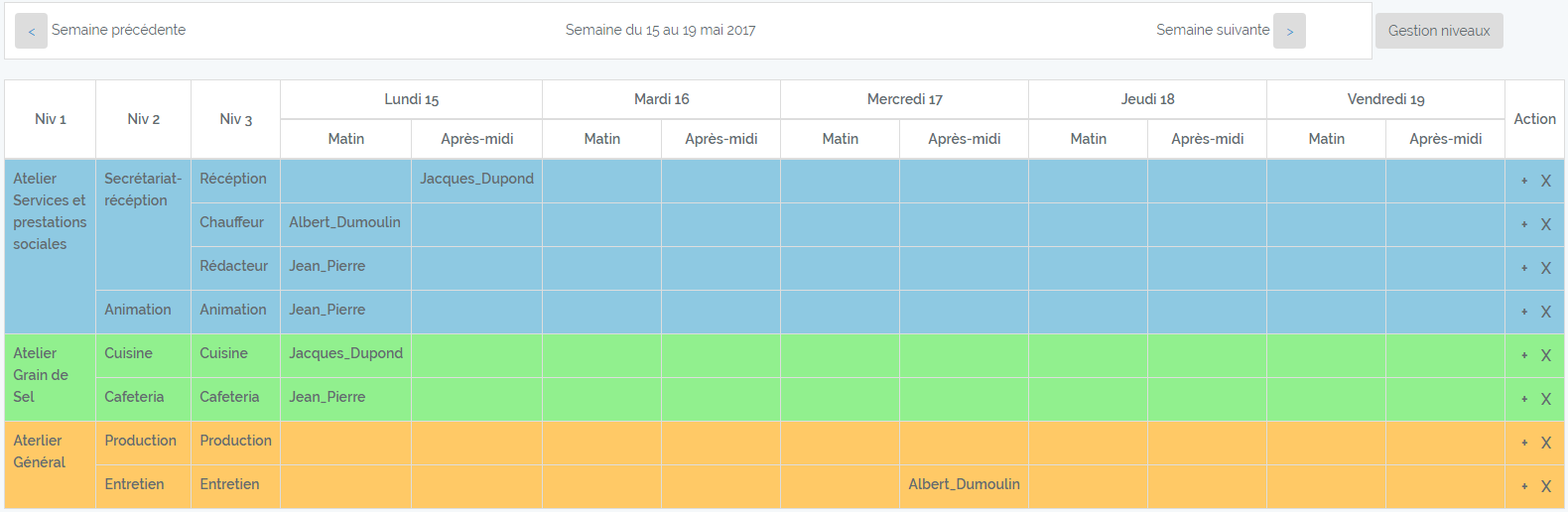
Niveau 1 : Ce sont les casiers principaux. Le titre est en grand et à l’intérieur se situent les ateliers concernant les niveaux plus élevés. Le dernier casier est constitué d’un formulaire pour ajouter un atelier de niveau 1. Dans le formulaire, on peut définir un nom d’atelier ainsi qu’une couleur.

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.

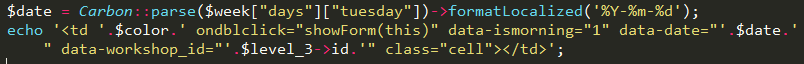
### Planning général

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:



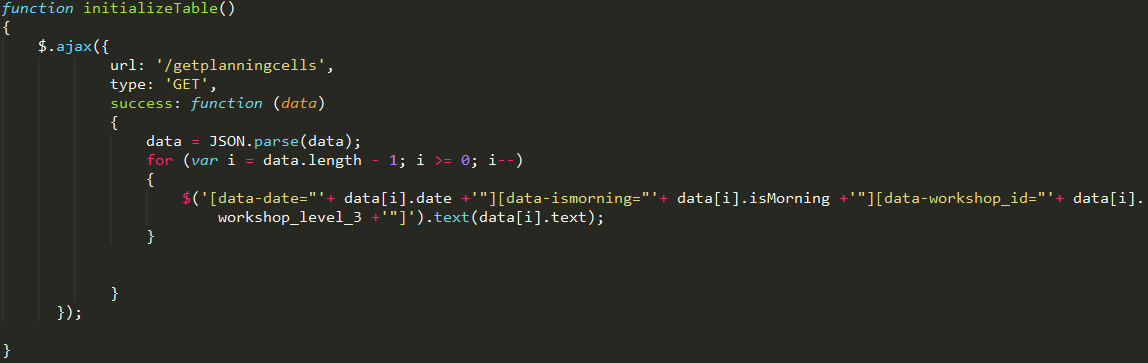
$date -> récupère la date du tableau $week, généré lors de la création de la vue et la formate.

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

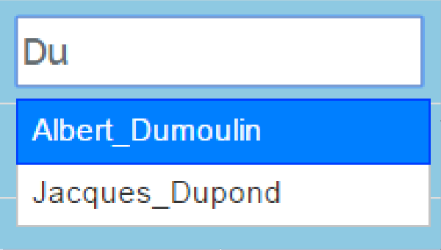
Ondblclick -> Evènement javascipt qui s’active lorsque la case est double cliquée par l’utilisateur. La fonction showForm() permet d’y afficher le champ pour ajouter un travailleur à la tâche.

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 :

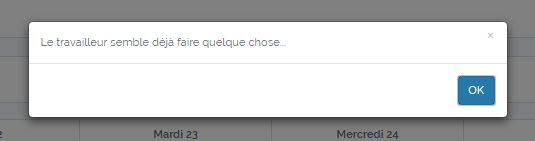


Pour assigner un utilisateur à une tâche, un double clic sur une case fera apparaître un champ texte qui est en fait un « autocomplete ». Cela signifie qu’à partir du moment où certaines lettres que l’utilisateur à rentrées correspondent au prénom ou au nom d’un travailleur, il est proposé dans une liste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

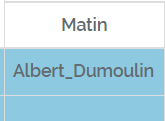


Pour valider la séléction, un simple clic sur le choix de l’utilisateur fait office d’envoi de requête. A partir de ce moment, la fonction qui enregistre le travailleur dans l’atelier vérifie d’abord que le travailleur ne soit pas occupé à une autre tâche à ce moment la.

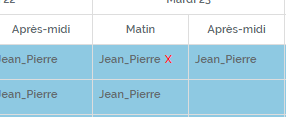
Dans ce cas la, un message d’erreur apparaît pour indiquer que l’utilisateur fait fausse route :



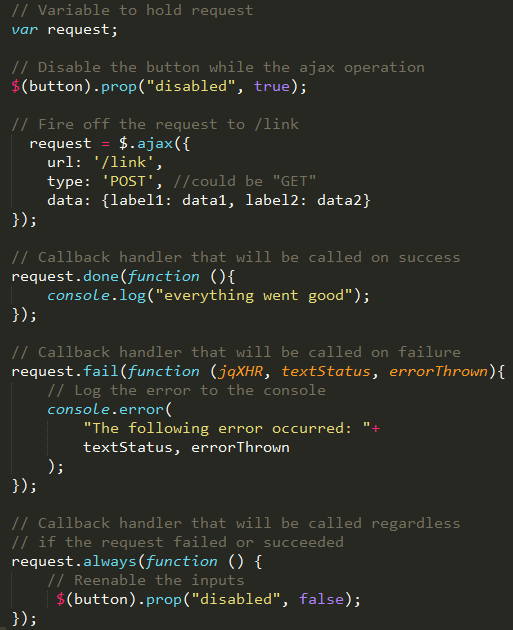
Si la requête est accéptée, le tableau se met à jour avec les nouvelles valeurs et le travailleur apparaît alors dans la table :



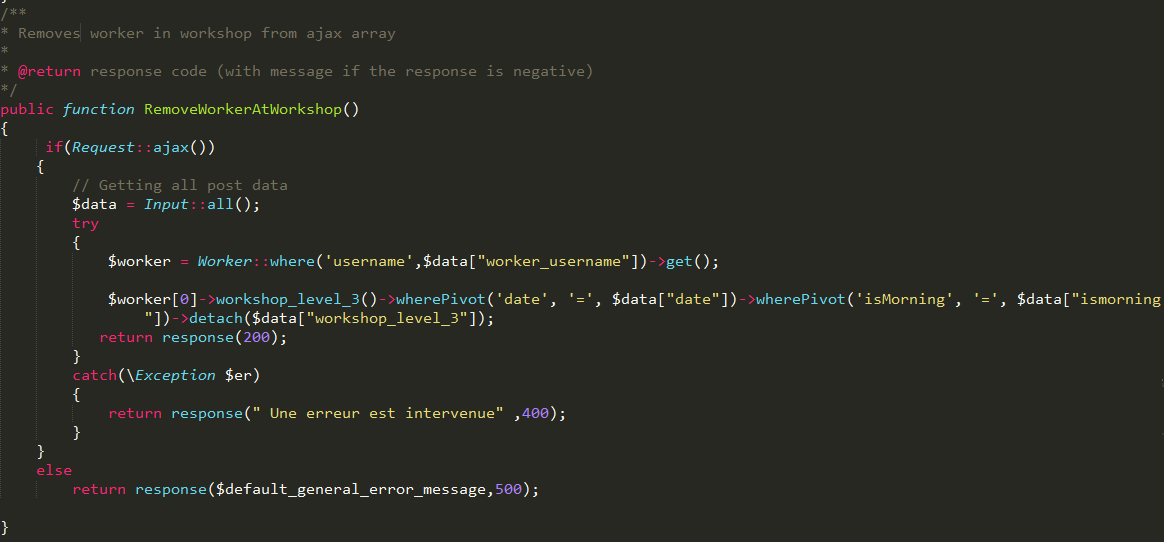
Il est aussi possible de supprimer un utilisateur à une tâche. Pour ce faire, lorsque l’utilisateur passe sa souris sur les cellules, une croix apparaît à coté des logins lors que ceux-ci sont inscrits :



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 :



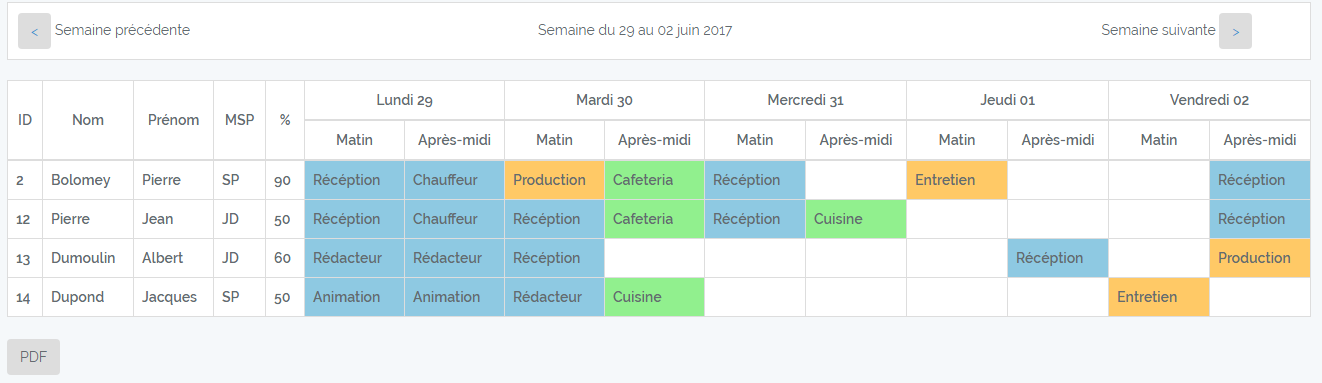
Ici, la fonction vérifie d’abord que des données envoyées ont bien étés récupérées.

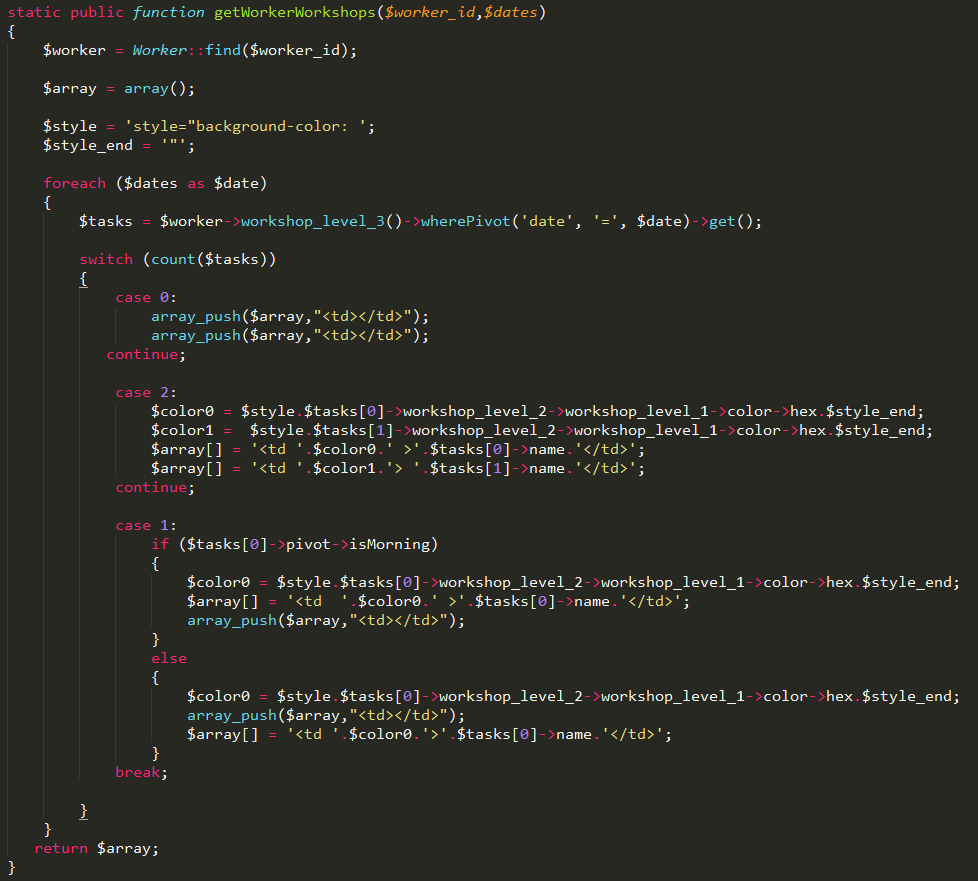
Ensuite, les données sont enregistrées dans le tableau « $data », où les valeurs sont accessibles à l’aide de leurs clés. Toutes les interactions sont dans un « try catch », pour pouvoir fournir une erreur à l’utilisateur si les données qui ont été envoyés étaient incorrectes.

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).

Problèmes rencontrés :

### Planning des travailleurs





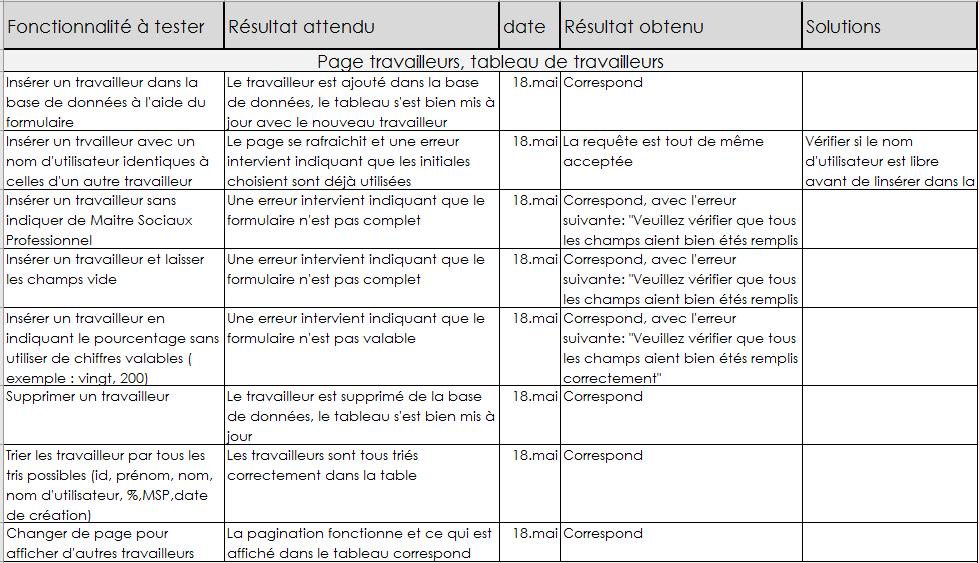
## Modifications

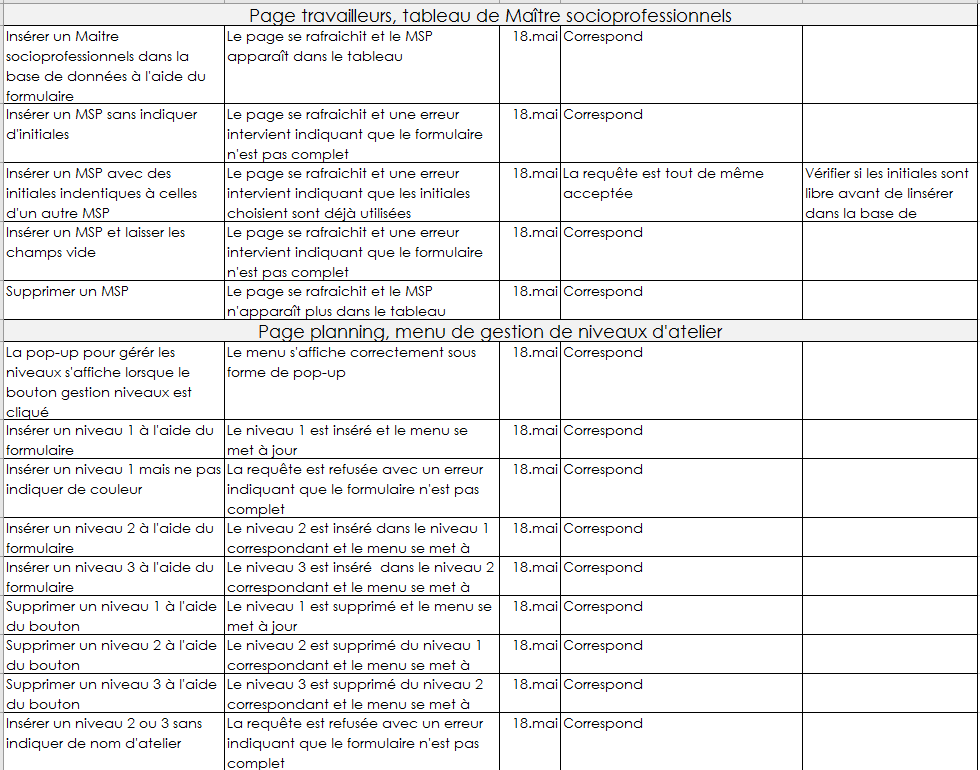
Historique des modifications demandées (ou nécessaires) aux spécifications détaillées.

Date, raison, description, etc.

# Tests

## Dossier des tests

Les tests ont été effectués en suivant la grille suivante : 



# Conclusion

## Bilan des fonctionnalités demandées

Il s’agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.

Si ce n’est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu’il reste à accomplir pour terminer le tout.

## Bilan de la planification

Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différences entre les planifications initiales et détaillées avec le journal de travail.

## Bilan personnel

Si c’était à refaire:

Qu’est-ce qu’il faudrait garder ? Les plus et les moins ?

Qu’est-ce qu’il faudrait gérer, réaliser ou traiter différemment ?

Qu’est que ce projet m’a appris ?

Suite à donner, améliorations souhaitables, …

Remerciements, signature, etc.

# Divers

## Journal de travail

Date, activité (description qui permet de reproduire le cheminement du projet), durée, liens et références sur des documents externes. Lorsqu’une activité de recherches a été entreprise, il convient d’énumérer ce qui a été trouvé, avec les références.

## Bibliographie

Références des livres, revues et publications utilisés durant le projet.

## Webographie

Références des sites Internet consultés durant le projet.

# Annexes

Cahier des charges

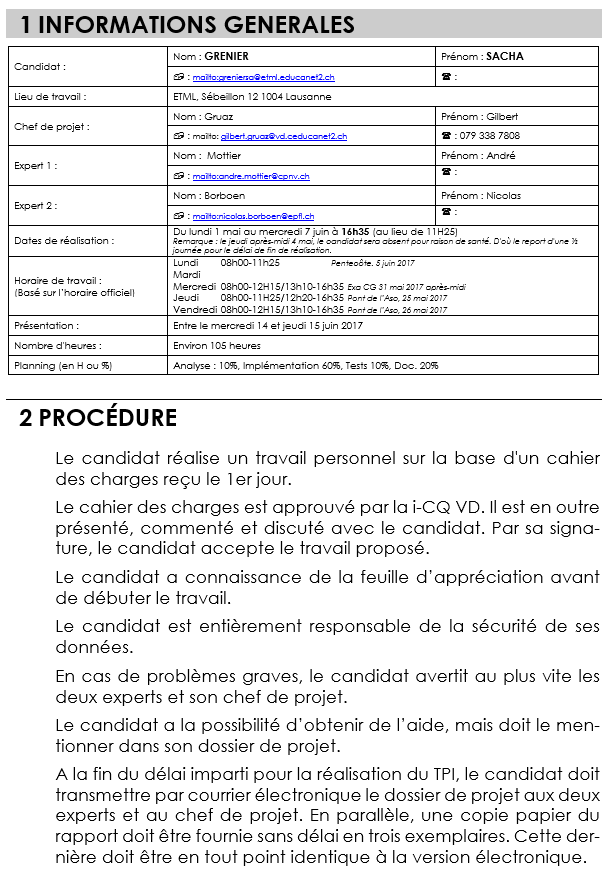
Listing du code source (partiel ou, plus rarement complet)

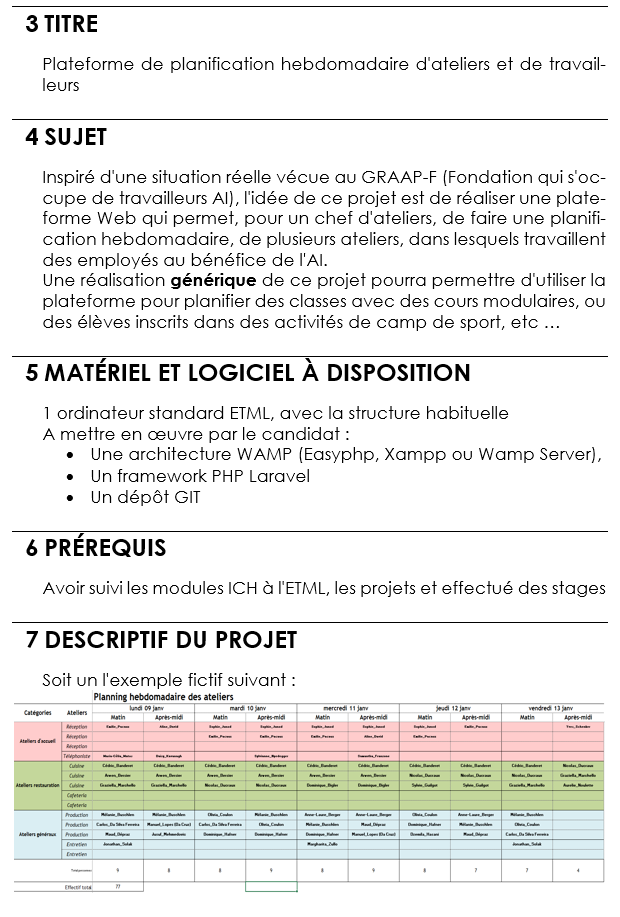
Guide(s) d’utilisation et/ou guide de l’administrateur

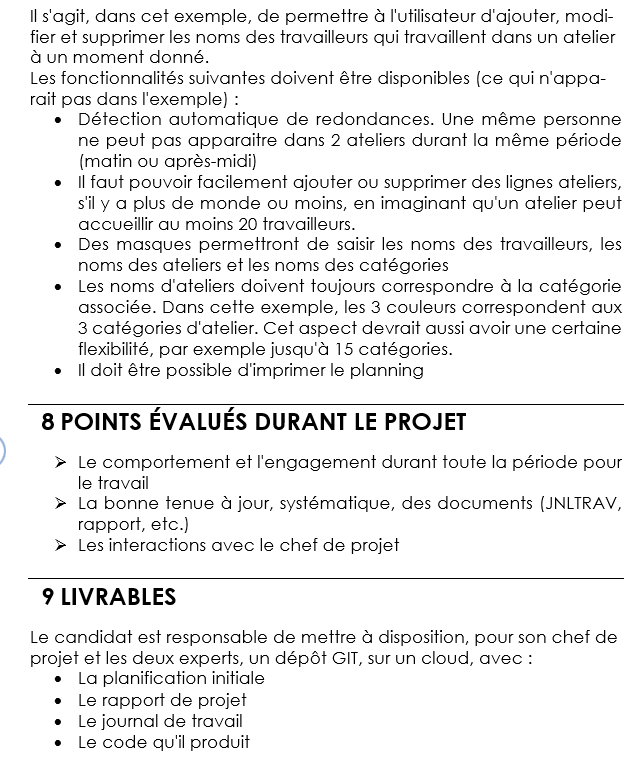
Etat ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).

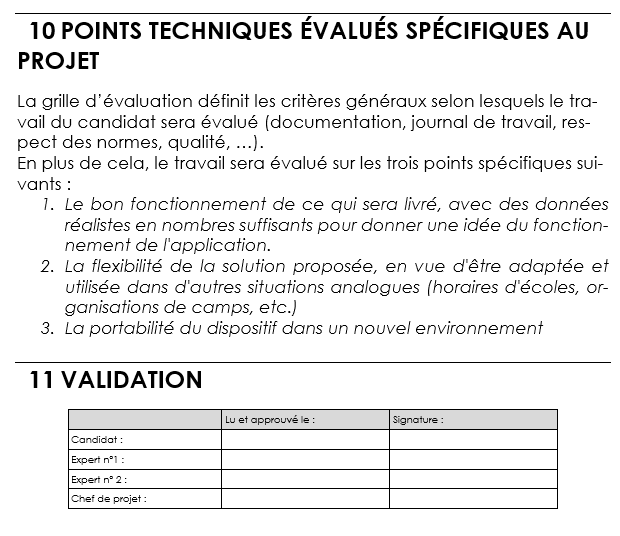
Extraits de catalogue, documentation de fabricant, etc.

## 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);