UFR sciences & techniques

UNIVERSITÉ DE BOURGOGNE

# PETAL

LACHAUD Samuel - PAZOLA Loïs - TISSERAND Orlane
BARSONI Dorian - AL-HALABI Zeina Helene - SENER Betul

# SOMMAIRE

# I) Introduction

## En quoi consiste PETAL

PETAL ou Plateforme d'Education et Travail Accessible en Ligne est le nom de notre projet de Développement d'application Web. Ce projet de fin de 6ème semestre de faculté d'informatique prend forme en groupe de 6 personne. Le projet est open-source et complètement libre de droit. Il peut être retrouvé à l'adresse suivante : https://github.com/samlach2222/PETAL

## Comment est-il programmé ?

PETAL est codé de manière conventionnelle pour une application web :

- Pour la partie Front-End (partie visible par l'utilisateur) nous utilisons le trio **HTML**, **CSS** et **JS**.
- Pour la partie Back-End (partie invisible pour l'utilisateur) nous utilisons **PHP** en version 7.03
- Pour la base de données nous utilisons **MySQL** en version 5.7.11

Nous utilisons en plus de cela plusieurs autres technologies :

- **JQuery**, qui permet une utilisation plus facile de JavaScript dans quelques cas et…
- **Ajax**, qui nous permet de gérer des parties de la page de manière asynchrone.
- **SweeAlert**, un plugin pour JavaScript permettant de rendre plus belles les alertes montrées à l'utilisateur

Nous avons choisi d'utiliser SweetAlert car il ne propose pas de rendre une tache de développement plus facile. Il est uniquement utilisé à des fins visuelles, cosmétiques.

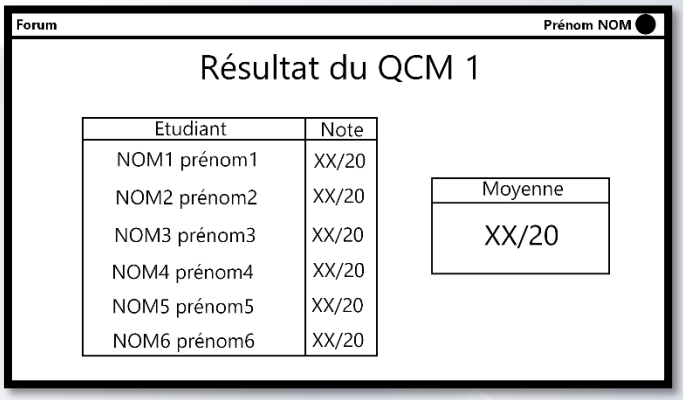
# II) Conception

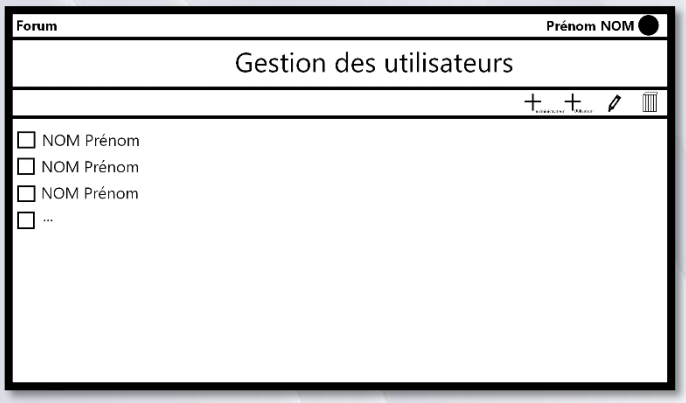
## Réalisation des maquettes

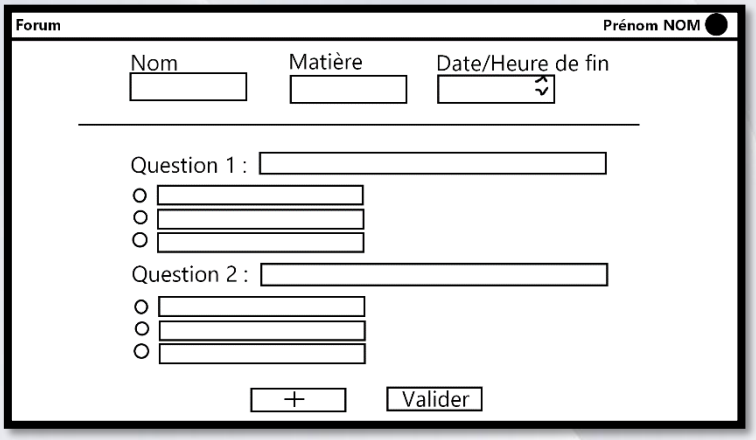
La première chose que nous avons réalisé dans la conception de notre projet a été de définir là où nous partons, et donc plus précisément, la maquette en fil de fer. Celle-ci permet d'avoir un aperçu des différentes sections de notre site web. Mais avant il a fallut découper le site en trois parties :

- La partie **Administrateur** : ensemble des pages permettant de manager les utilisateurs et les matières avec leurs cours et leurs QCM.
- La partie **Utilisateur** : ensemble des pages permettant à l'utilisateur d'utiliser le site et de naviguer parmi les ressources mises à disposition des administrateurs.
- La partie **All** : ensemble des pages ou éléments correspondant aux parties communes des deux parties précédentes

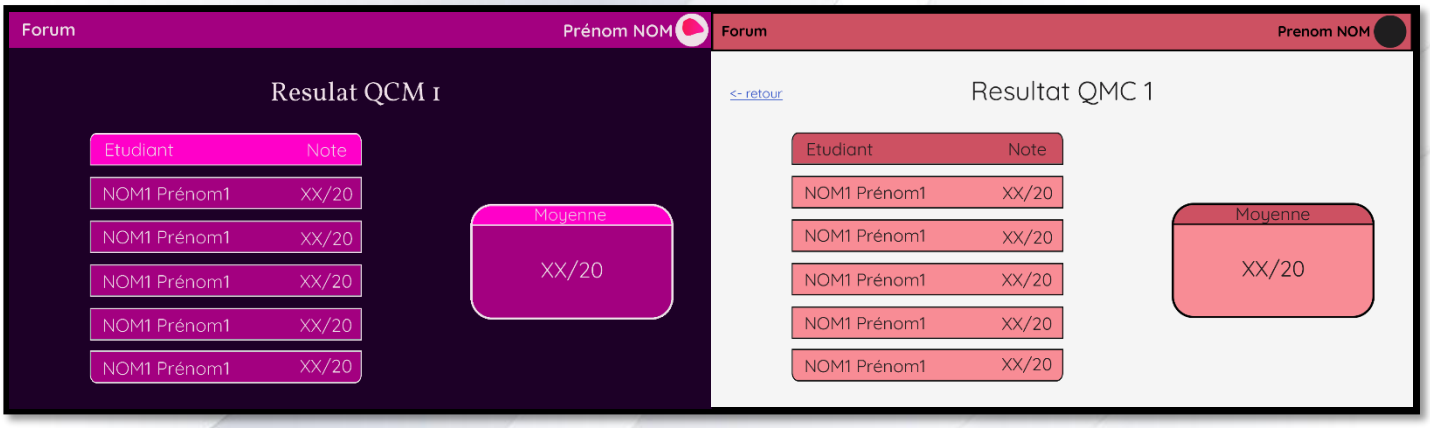
Voici donc quelques-unes de ces maquettes en fil de fer retrouvables à l'adresse suivante : https://github.com/samlach2222/PETAL/tree/main/Conception/Maquettes%20fil%20de%20fer

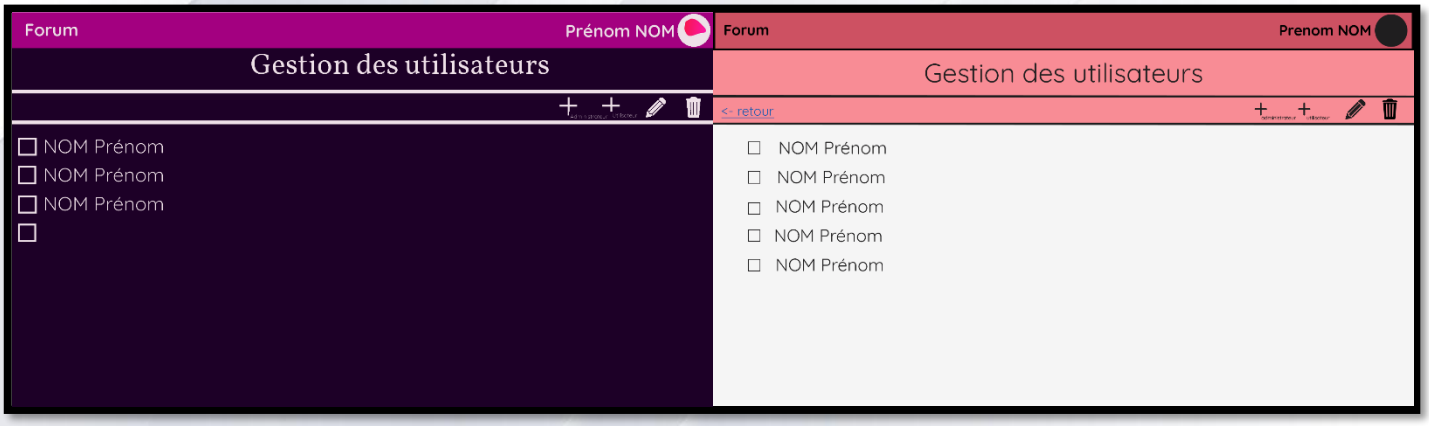


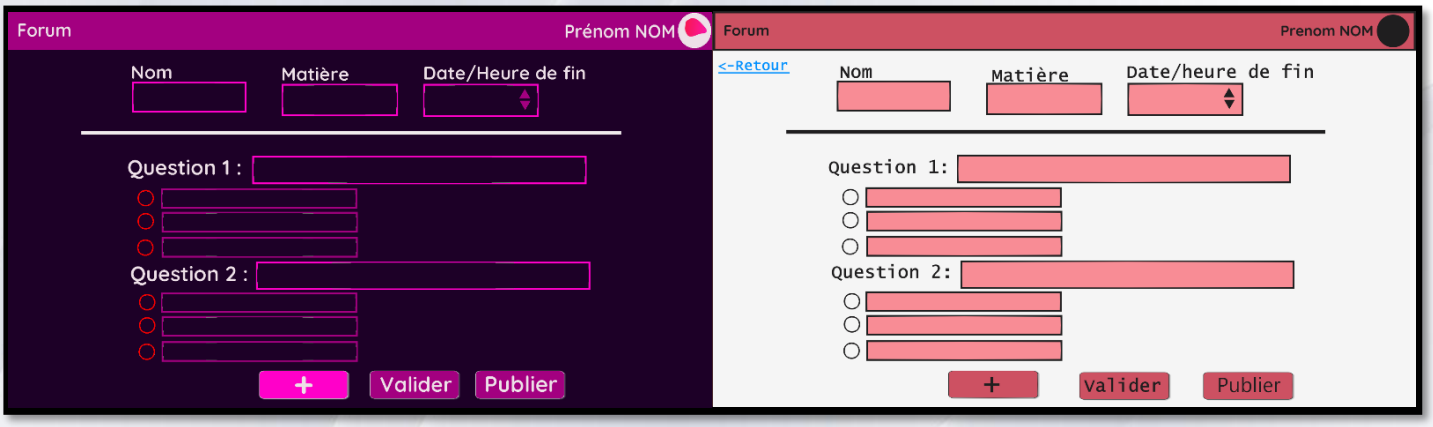



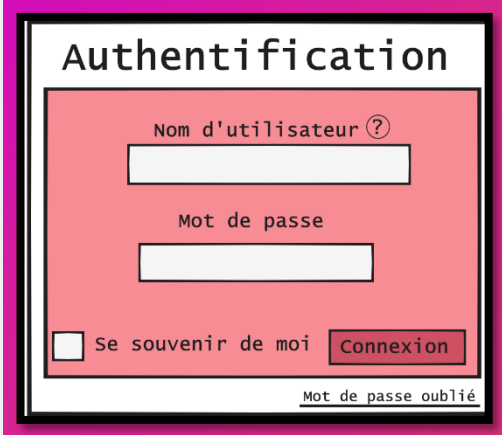
Pour donner suite à la création de la charte graphique, que nous verrons dans la partie suivante, nous avons réalisé les maquettes avec leurs couleurs en 2 versions pour les thèmes sombres et clairs. Voici donc quelques-unes retrouvables ici : https://github.com/samlach2222/PETAL/tree/main/Conception/Pages%20terminées




Enfin, la dernière partie concernant les maquettes est la description des balises pour partir sur une base commune pour les principaux contrôleurs. Voici un exemple sur la page de connexion retrouvable ici https://github.com/samlach2222/PETAL/tree/main/Conception/Maquettes%20avec%20balises :

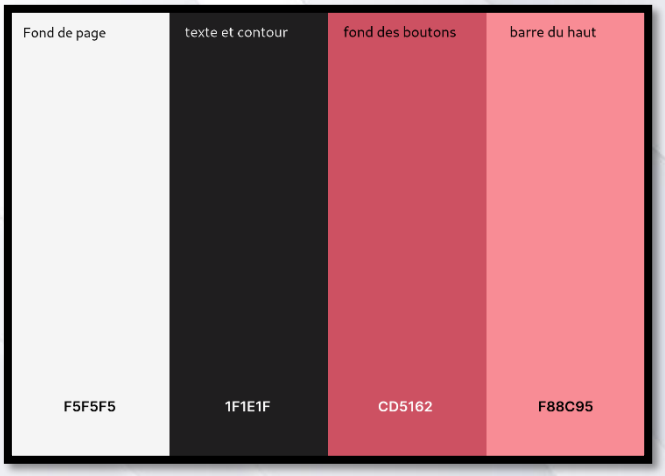



Div pour Authentification avec id=authentification, class=authentification, texte “Authentification” :

- Balise a avec id =mdp-oublie et comme texte “Mot de passe oublié”
- Div avec id=auth-center, class=authentication:
  - Label “Nom d’utilisateur”
  - Texte “?” avec un over indiquant les différents types d’identifiant
  - Input avec id=nom_user pour récupérer l’identifiant
  - Label “Mot de passe”
  - Input avec id=mdp pour récupérer et vérifier le mot de passe
  - Check box avec id=remember_me
  - Label “Se souvenir de moi”
  - Bouton avec id=connexion et comme texte “Connexion”

# Réalisation des chartes graphiques

Comme dit précédemment, avant de faire la maquette finale, il a été nécessaire de créer une charte graphique pour les deux thèmes. Ont alors été élaborés des panels de couleurs avec leurs utilités retrouvables ici : https://github.com/samlach2222/PETAL/tree/main/Conception/Charte%20Graphique




Nous avons donc premièrement le thème clair composé de 4 couleurs. Sur ces 4 couleurs, les deux premières forment les éléments les plus importants de la page. Le blanc est le fond de la page et le gris foncé le texte de la page. Nous avons ensuite deux couleurs de rose. Le rose le plus foncé correspond à la couleur attribuée au bandeau et aux boutons, et le plus clair correspond à la couleur d'accentuation des éléments en ayant besoin.


Puis nous avons le thème foncé, celui-ci est composé de 6 couleurs, nous retrouvons le fond de la page cette fois-ci dans un violet foncé et la couleur de texte en blanc cassé. La différence se fait sur les couleurs d'accentuation. En effet, nous avons 4 couleurs supplémentaires. La couleur « Logo Color 1 (Dark) » est la couleur du bandeau et des boutons. La version standard de cette couleur correspond aux accentuations comme les champs inputs ou des boutons supplémentaires. La couleur « Logo Color 2 (Dark) » est utilisée pour des touches d'accentuations pas très prononcées à l'œil, alors que sa version classique sert à attirer l'œil du visiteur rapidement.

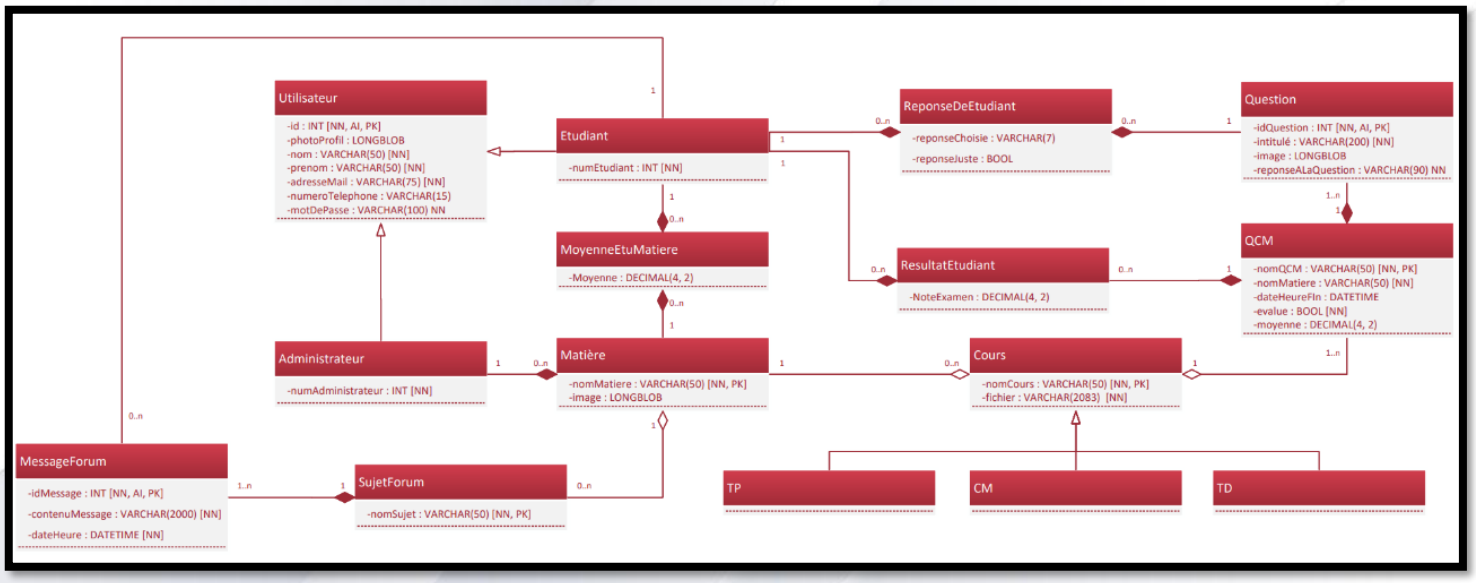
L'autre partie de cette charte graphique touche les polices. Nous avons choisi deux polices, je vais les laisser se présenter :

**Bonjour, je suis Volkorn et je suis la police de tous les titre du site PETAL. Vous pouvez me retrouver à l'adresse suivante : https://fonts.google.com/specimen/Vollkorn**

Bonjour, je suis QuickSand et je suis la police de tous les autres textes du site PETAL. Vous pouvez me retrouver à l'adresse suivante : https://fonts.google.com/specimen/Quicksand

# Conception BDD

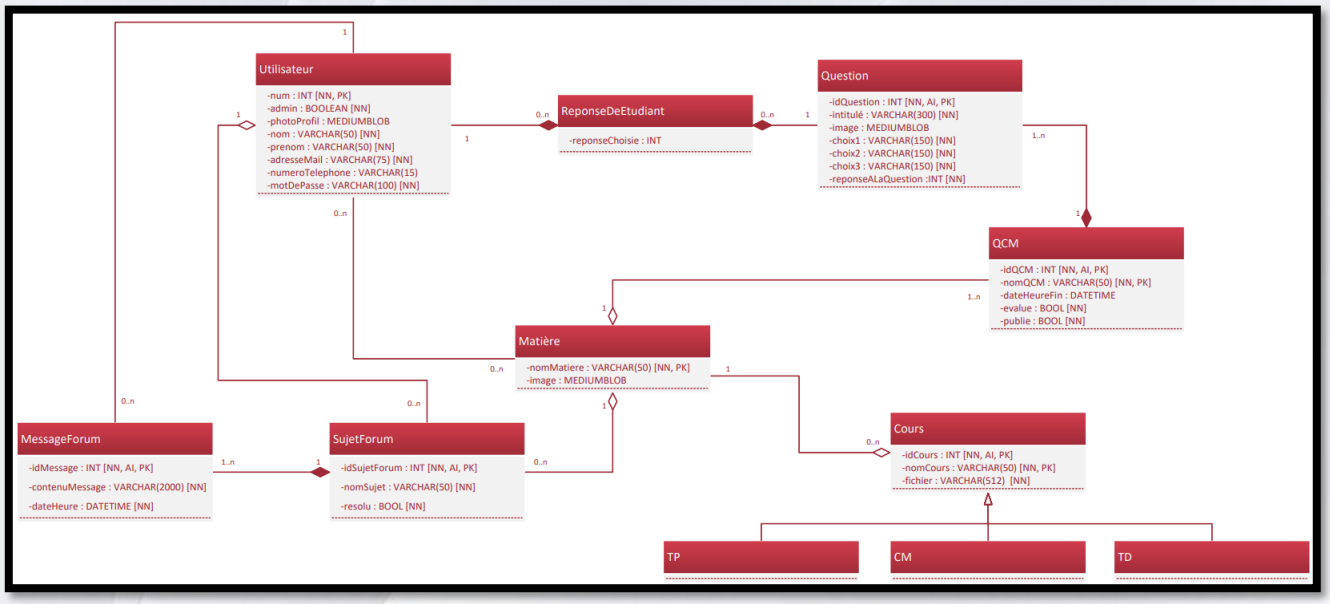
Après avoir prévu et conceptualisé l'intégralité de la partie visible de notre site, il est important de bien prévoir l'ensemble des données qui vont circuler et être stockée dans notre site web. Ainsi nous avons donc commencé par mettre en place un diagramme des différentes tables de la BDD. Voici la version originale :




Puis à partir de celui-ci, nous avons créé le modèle relationnel suivant et avons réalisé le fichier sql associé :

**Modèle relationnel :**
Utilisateur(id, photoProfil, nom, prenom, adresseMail, numeroTelephone, motDePasse, #nom)
Administrateur(#id, numAdministrateur)
Etudiant(#id, numEtudiant)
MoyenneEtuMatiere(#nomMatiere, #id, moyenne)
Matiere(nomMatiere, image, #numAdministrateur, #id)
SujetForum(nomSujet, #nomMatiere)
MessageForum(idMessage, contenuMessage, dateHeure, #nomSujet, #id)
Cours(nomCours, fichier, #nomMatiere, typeCours)
ResultatEtudiant(#id, #nomQCM, noteExamen)
QCM(nomQCM, #nomMatiere dateHeureFin, evalue, moyenne, #nomCours)
ReponseDeEtudiant(#id, #idQuestion, reponseChoisie, reponseJuste)
Question(idQuestion, intitulé, image, reponseALaQuestion, #nomQCM)

En fin de projet, nous nous sommes rendu compte régulièrement que notre conception de la BDD n'était pas bonne, nous l'avons donc régulièrement changé car nous avons manqué de rigueur (ou de recul) lors de la création initiale. Cette conception a été réalisée sur Microsoft Visio. Voici la version finale retrouvable ici : https://github.com/samlach2222/PETAL/tree/main/BDD



# III) Gestion de projet

## Base de notre projet

La gestion de projet est la base de celui-ci dans le cadre d'un projet en collaboration. Et la base de la gestion de projet c'est la collaboration entre les différents membres/ressources de ce projet. En effet PETAL c'est tout d'abord des chiffres :

- **6** étudiants en License d'informatique
- **14** semaines de projet
- Une moyenne de **8h** de travail par semaine
- Des pics de travail à plus de **40h** par semaine

Nous pouvons donc à partir de ces chiffres calculer les points du projet :

**1h par semaine = 0.25 points → 8h par semaine = 2 points**

**2 points * 6 étudiants * 14 semaines = 168 points**

**Notre projet comporte donc 168 points.**

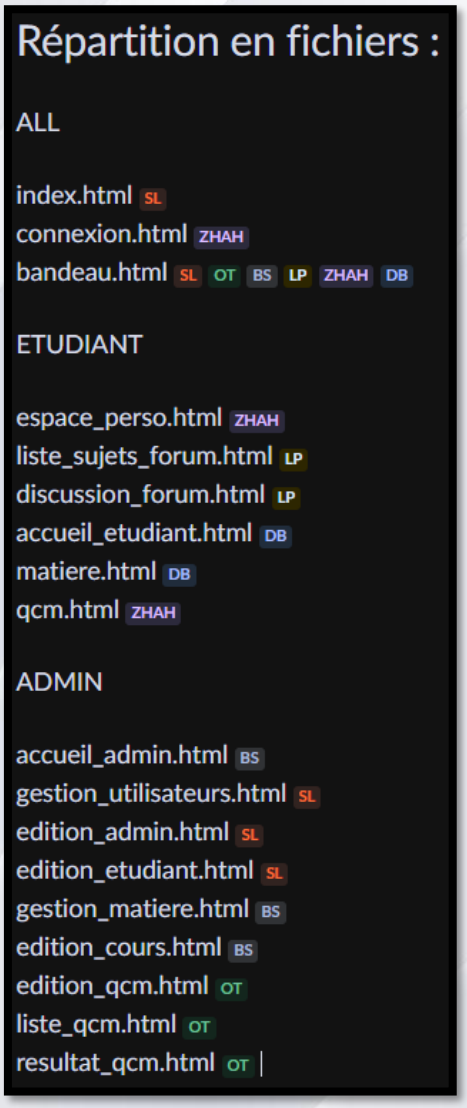
Nous avons fait alors un premier découpage des différentes tâches au départ du projet. Nous avons donc choisi un découpage en 4 niveaux de tâches.

| RANG 1 | RANG 2 | RANG 3 | RANG 4 |
|---|---|---|---|
| Conception | Maquette fil de fer | Réalisation des pages | • Partie administrateur<br>• Partie utilisateur<br>• Page Login |
| | Maquette Numérique | Réalisation des pages | • Partie administrateur<br>• Partie utilisateur<br>• Page Login |
| | | Charte graphique | • Charte graphique Dark Theme<br>• Charte graphique Light Theme |
| | | Contenu du site (textes dans les divers éléments exemple: div) | • Partie administrateur<br>• Partie utilisateur<br>• Page Login |
| BDD | • Lieu d'Upload (UWAMP)<br>• Diagramme des tables<br>• Structure des données (types de données)<br>• Langage (type de SQL) | | |
| Code | HTML | Découpage en pages | • Partie administrateur<br>• Partie utilisateur<br>• Page Login |
| | | Codage des pages | A REPARTIR |
| | CSS | Découpage en fichiers CSS | • Partie administrateur<br>• Partie utilisateur<br>• Page Login |
| | | Mise en place de thèmes | • CSS Dark Theme<br>• CSS Light Theme |
| | JS | Découper en fichier | • Partie administrateur<br>• Partie utilisateur<br>• Page Login |
| | | Conception des différentes méthodes et fonctions | A REPARTIR !! |
| | PHP | Découpage en fichier | • Partie administrateur<br>• Partie utilisateur<br>• Page Login |
| | | Conception des différentes méthodes et fonctions | A REPARTIR !! |

Nous avons alors déterminé 4 ensembles de ressources différentes :

- Les équipes des pages
  - Equipe Administrateurs SL OT BS
  - Equipe Utilisateur LP ZHAH DB
- Les équipes des thèmes
  - Equipe thème clair ZHAH DB OT
  - Equipe thème foncé SL LP BS

Puis quand il a fallu s'occuper des différentes pages du site, Ainsi la répartition suivante a pu être effectuée sur les pages HTML, les scripts PHP et JS ainsi que la création des thèmes CSS.




Parmi les 6 développeurs, nous pouvons trouver deux rôles : Le Scrum Master qui est en quelque sorte le chef de projet, et le Product Owner, qui s'occupe en grande partie de la communication avec le client (Profs) :

- **Scrum Master : Samuel LACHAUD**
- **Product Owner : Betul SENER**

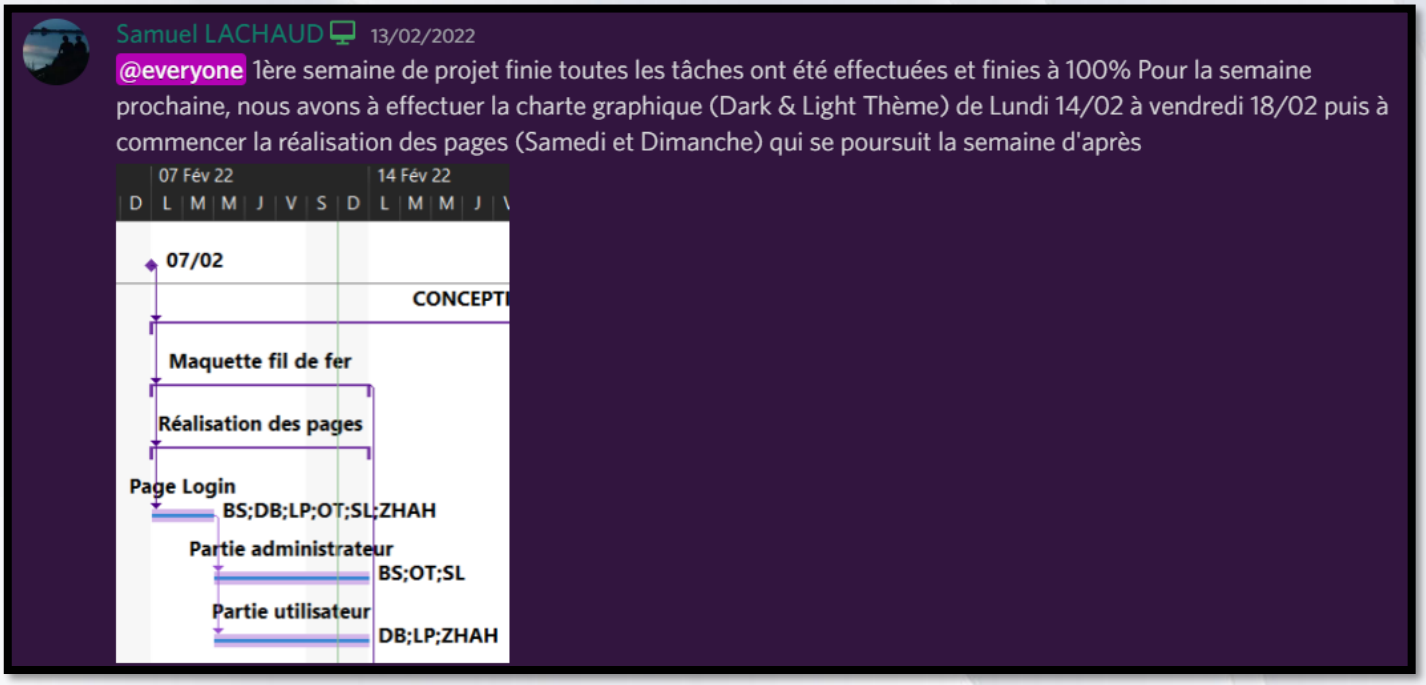
# Réunions et communications

La communication est la part la plus importante de la bien-réussite d'un projet collaboratif. Nous avons donc utilisé plusieurs outils facilitant le travail collectif.

Le premier outil et le plus connu est GitHub. En effet celui-ci nous a permis de gérer le code et d'organiser le travail par équipe sans se soucier de fusionner les modifications de chacun.

Le deuxième outil utilisé est Cozy. En effet il nous a permet, de la même manière que le ferais google docs, d'éditer nos documents à plusieurs, en ligne, mais cette fois hébergé en France et soucieux de la protection des données personnelles

La communication a également été assurée régulièrement via le serveur discord de notre projet. Chaque semaine une notification est envoyée à tout le monde pour signifier la fin d'une semaine de travail. Est alors décrit le travail à fournir pour la semaine suivante avec un aperçu du Gantt concernant le moment présent.



La communication à distance est donc bien assurée et en plus de demander régulièrement à chaque membre du projet des nouveautés dans des canaux dédiés sur discord, des réunions régulières ont été mises en place pour assurer une communication de groupe constante et pour assurer la direction dans laquelle va le projet PETAL.


Voici donc la liste des réunions qui ont été effectuées avec leurs dates :

**0ère semaine → 01/02**

**1ème semaine→ 08/02**

**2ème semaine →15/02**

**4ème semaine 28/02 et 02/03**

**5ème semaine 08/03**

**6ème semaine 15/03**

**7ème semaine 22/03**

**9ème semaine 30/03 et 31/03**

**SEMAINE CONFINEMENT**

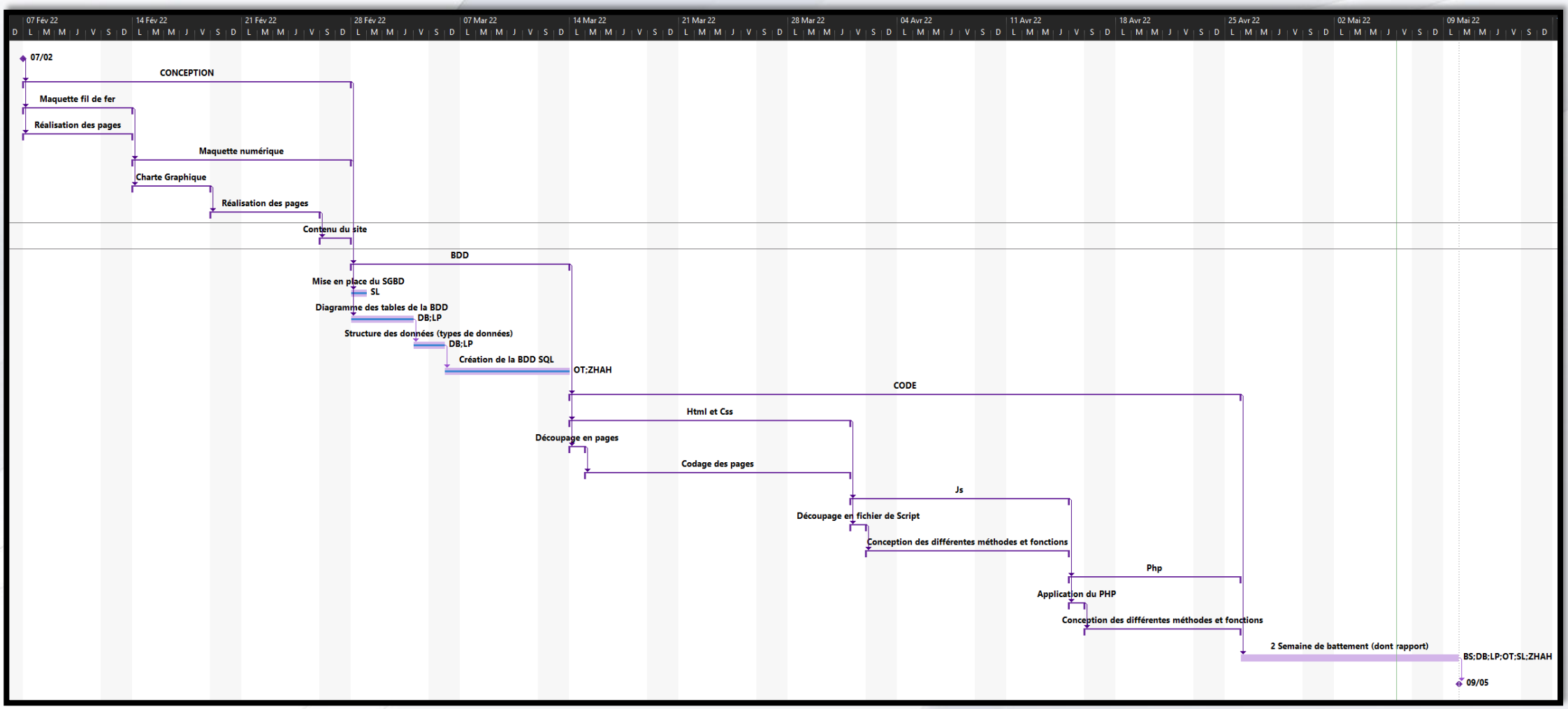
**11ème semaine 11/04**

**VACANCES**

**14ème semaine 05/05**

## Diagramme de Gantt

La partie la plus importante de notre gestion de projet, c'est sa partie prévisionnelle. Nous avons commencé le projet très tôt, à vrais dire, le jour même de la sortie du sujet. A l'aide de diverses réunions, nous avions découpé en plusieurs tâches, réparti celles-ci à travers les différentes ressources. Enfin, nous avons déterminé une temporalité pour chacune des tâches et sous-tâches et déterminé un diagramme de prévisionnel de Gantt réalisé avec Microsoft Project retrouvable ici : https://github.com/samlach2222/PETAL/tree/main/Gestion-Projet/Diagramme%20de%20Gantt. En voici donc un cours aperçu

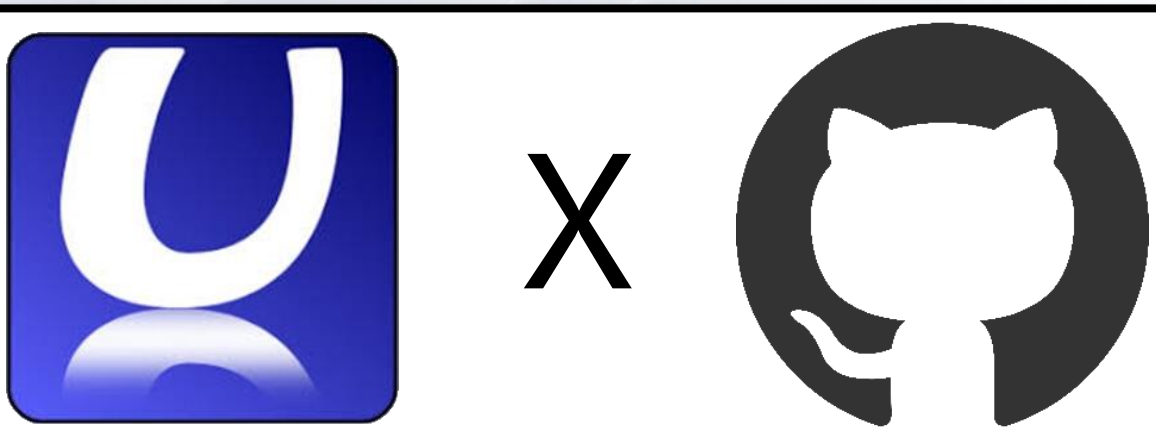

07 Fév 22
14 Fév 22
21 Fév 22
28 Fév 22
07 Mar 22
14 Mar 22
21 Mar 22
28 Mar 22
04 Avr 22
11 Avr 22
18 Avr 22
25 Avr 22
02 Mai 22
09 Mai 22
07/02
CONCEPTION
Maquette fil de fer
Réalisation des pages
Maquette numérique
Charte Graphique
Réalisation des pages
Contenu du site
BDD
Mise en place du SGBD
SL
Diagramme des tables de la BDD
DB;LP
Structure des données (types de données)
DB;LP
Création de la BDD SQL
OT;ZHAH
CODE
Html et Css
Découpage en pages
Codage des pages
Js
Découpage en fichier de Script
Conception des différentes méthodes et fonctions
Php
Application du PHP
Conception des différentes méthodes et fonctions
2 Semaine de battement (dont rapport)
BS;DB;LP;OT;SL;ZHAH
09/05

# IV) Développement du projet

## GitHub et uWamp

Avant même d'envisager le développement de PETAL, il a fallu imaginer la gestion de notre base de données et surtout le type d'hébergement de notre site. En effet il n'était pas envisageable pour nous de travailler depuis un serveur distant. Nous avons donc choisi une solution assez spéciale.

Nous utilisons GitHub comme gestionnaire de code, nous voulons utiliser uWamp qui est une version plus légère et plus pratique de WAMP. Or ce genre de serveur local n'est absolument pas prévu pour du travail en collaboration. Nous avons donc choisi de stocker uWamp directement sur notre GitHub.



UWAMP possède alors la même configuration pour tous les utilisateurs, les données de la base de données sont communes pour tout le monde et le code se trouvera donc pour tout le monde directement dans le dossier « WWW ». Cette manière de fonctionner est très pratique, l'installation est facile car il faut juste clone le repository GitHub et on possède le serveur et directement le site PETAL.

Cependant UWAMP possède un problème, il crée des fichiers de configuration pour chaque utilisateur, et crée des fichiers de base de données que l'on ne veut pas forcément commit. Nous utilisons donc pour cela la technologie **GITIGNORE** qui permet comme son nom l'indique d'ignorer les fichiers notés dans celui-ci, que ce soit précisément ou alors un ensemble à l'aide d'un regex. Cependant cette technique a des limites. C'est pour cela qu'on utilise un ou bien un **Pré-commit gitHook**.

En effet **GITIGNORE** sert uniquement à empêcher de commit des fichiers qu'on ne veut jamais qu'ils soient commis. Or nous avons besoin (pour une majorité des fichiers de bases de données que le fichier sois commit, mais qu'il ne puisse plus jamais l'être par les utilisateurs, que le fichier soit en lecture seule. Le **pré-commit** sert donc à ceci.

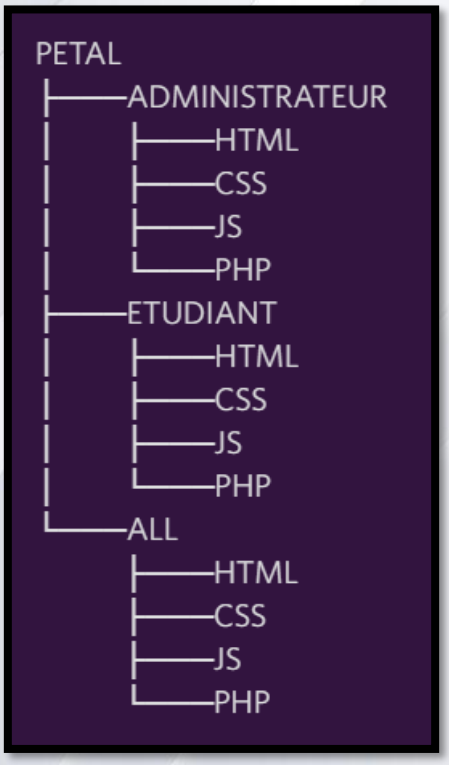
Nous utilisons donc le SGBD uWamp composé d'apache, de PHP 7.0.3 et de MySQL 5.7.11. Le tout hébergé sur le repository GitHub PETAL avec le blocage des fichiers non voulu (fichiers personnels) avec GITIGNORE et blocage des fichiers en lecture seule avec **PRE-COMMIT.** Voici donc le fichier **pre-commit.hook** retrouvable ici : https://github.com/samlach2222/PETAL/blob/main/pre-commit.hook

```
#!/bin/sh
version="Version 4.7"
echo "pre-commit: $version"
# Fonction end pour quitter le script
End(){
    echo "pre-commit: FIN, supprime le fichier temporaire stagedFiles.tmp"
    rm -f stagedFiles.tmp
    exit $1
}
# Interrompt dès qu'une erreur est rencontrée
set -e
# Arrête le script et demande d'exécuter le batch si les versions ne correspondent pas
echo "pre-commit: Check la version du hook pre-commit"
read -r firstLineBatch < "(Ré)initialiser pre-commit.bat"
# Enlève le caractère CR de firstLineBatch pour pouvoir comparer
if [[ "${firstLineBatch%$'\r'}" != "::$version" ]]; then
    cp -f "pre-commit.hook" ".git/hooks/pre-commit"
    echo -e "\npre-commit: Le hook pre-commit a été mis à jour"
    echo -e "pre-commit: VEUILLEZ COMMIT À NOUVEAU\n"
    End 1
fi
# Par défaut les commandes git n'output pas en UTF-8
# L'activer permet de supporter l'UTF-8 pour la suite de ce bash
echo "pre-commit: Active les outputs UTF-8 avec git pour ce repo"
git config core.quotepath off
echo "pre-commit: Crée et remplie le fichier stagedFiles.tmp avec les fichiers à commit"
git diff --name-only --no-renames --cached > stagedFiles.tmp
echo "pre-commit: Check s'il y a des fichiers dans PETAL parmi les fichiers à commit"
sortieRapide=$(awk 'BEGIN {awk_sortieRapide=1}
/PETAL\// {awk_sortieRapide=0; exit}
END {print awk_sortieRapide}' stagedFiles.tmp)
if [[ $sortieRapide == 1 ]]; then
    echo "pre-commit: Aucun fichier à commit n'est dans PETAL, sortie rapide"
    End 0
else
    echo "pre-commit: Il y a des fichiers à commit se trouvant dans PETAL"
fi
echo "pre-commit: Enlève les fichiers dans PETAL des fichiers à commit"
git reset -q HEAD -- PETAL
# Si au moins 1 fichier BDD est à commit, il faut
# commit tous les fichiers BDD
echo "pre-commit: Check s'il y a des fichiers BDD parmi les fichiers à commit"
FichiersBDDSelectionnes=$(awk 'BEGIN {awk_FichiersBDDSelectionnes=0}
/PETAL\/bin\/database\/mysql-5.7.11\/data\/petal_db\// {awk_FichiersBDDSelectionnes=1; exit}
/PETAL\/bin\/database\/mysql-5.7.11\/data\/mysql\/proc.MYD/ {awk_FichiersBDDSelectionnes=1; exit}
/PETAL\/bin\/database\/mysql-5.7.11\/data\/mysql\/proc.MYI/ {awk_FichiersBDDSelectionnes=1; exit}
END {print awk_FichiersBDDSelectionnes}' stagedFiles.tmp)
if [[ $FichiersBDDSelectionnes == 1 ]]; then
    echo "pre-commit: Au moins 1 fichier BDD à commit, ajoute tous les fichiers BDD"
    fichiersCaches=$(git ls-files -v "PETAL/bin/database/mysql-5.7.11/data/ibdata1" | { grep "^h" || true; })
    if [[ -z "$fichiersCaches" ]]; then
        git add "PETAL/bin/database/mysql-5.7.11/data/ibdata1"
    else
        echo "pre-commit: Réaffiche ibdata1 pour pouvoir le commit"
        git update-index --no-assume-unchanged "PETAL/bin/database/mysql-5.7.11/data/ibdata1"
        git add "PETAL/bin/database/mysql-5.7.11/data/ibdata1"
        echo "pre-commit: Recache ibdata1"
        git update-index --assume-unchanged "PETAL/bin/database/mysql-5.7.11/data/ibdata1"
    fi
    git add "PETAL/bin/database/mysql-5.7.11/data/petal_db/"
    git add "PETAL/bin/database/mysql-5.7.11/data/mysql/proc.MYD"
    git add "PETAL/bin/database/mysql-5.7.11/data/mysql/proc.MYI"
else
    echo "pre-commit: Pas de fichier BDD à commit"
fi
echo "pre-commit: Ajoute les fichiers Web et tous les autres fichiers utiles dans PETAL étant à la fois à commit et pouvant être
commit"
awk '
/PETAL\/www\// {system("git add " "\"" $0 "\"")}
/PETAL\/bin\/apache\/conf\/httpd_uwamp.conf/ {system("git add " "\"" $0 "\"")}
/PETAL\/bin\/php\/php-7.0.3\/php_uwamp.ini/ {system("git add " "\"" $0 "\"")}
' stagedFiles.tmp
if [[ `git diff --cached --numstat | wc -l` == 0 ]]; then
    echo -e "\npre-commit: ERREUR, IL NE RESTE AUCUN FICHIER À COMMIT\n"
    End 1
fi
End 0
```

Une fois installé dans le dossier git de l'utilisateur, ce fichier va donc bloquer le commit de fichiers en lecture seule. Il possède même un système de version afin de le mettre à jour et affiche à l'utilisateur des informations complémentaires comme la disponibilité d'une mise à jour du script ou un blocage.

# HTML / CSS

Comme on peut le voir sur le diagramme de Gantt, le développement HTML/CSS a duré 2 semaines et demie. Après avoir répartie la création des différentes pages, nous avons convenu d'une architecture de fichiers pour les stocker :




Avant de coder les différentes pages, toutes les ressources en commun ont développé le bandeau de chaque page. En effet celui-ci est un fichier HTML inclut à l'aide de PHP dans chacune des autres pages afin d'éviter la répétition inutile de code. Nous utilisons alors le système de calque suivant :

- La page est le calque de fond
- Le bandeau est un calque par-dessus

Les calques sont gérés grâce à l'indication CSS **Z-INDEX**

Une fois le bandeau codé, les différentes pages sont développés en suivant au plus possible les maquettes. Or, comme pour la base de données, nous nous sommes rendu compte de maquettes manquantes pour certaines fonctionnalités ou des parties à changer dans certaines maquettes, elles ont donc été refaites avant de proposer une version du code ajoutant ces fonctionnalités.

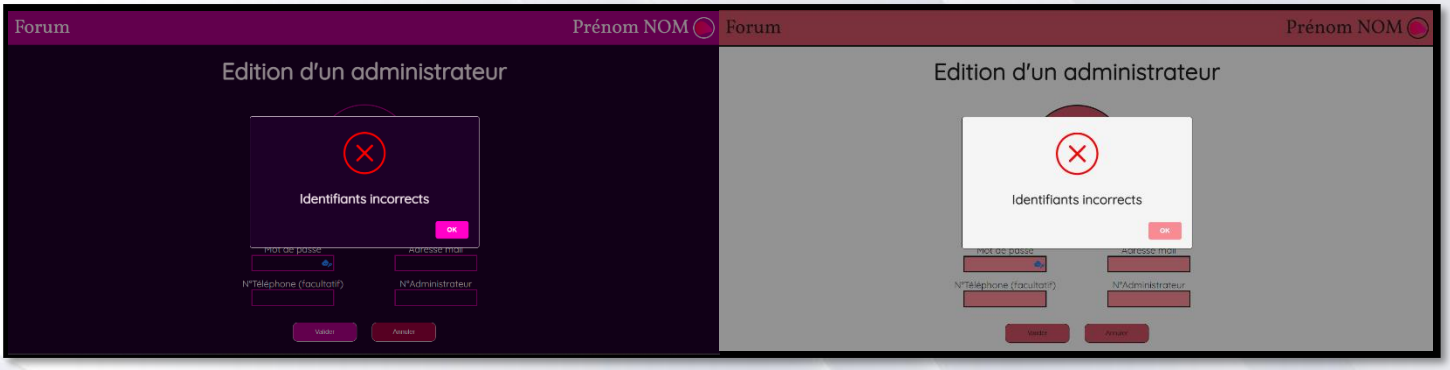
Pour la partie CSS, il existe 2 fichiers CSS pour chaque page (une pour le thème clair, et l'autre pour le foncé). Pour faciliter la tâche, le fichier CSS du bandeau clair est appelé dans chaque fichier CSS clair d'une page, et inversement.

# JS / PHP

Sur le diagramme de Gantt, on peut également apercevoir que les tâches suivantes sont le développement JavaScript (2,5 semaines) et le développement PHP (2.5 semaines).

Le développement JavaScript va concerner les différents scripts à exécuter localement sur le navigateur de l'utilisateur. Il nous sert donc principalement à des fins visuelles et esthétiques. Mais il sert surtout pour avertir efficacement sur les mauvaises actions de l'utilisateur. En effet, les cloisons des formulaires ou de toute autre entrée de l'utilisateur sont posées en PHP, mais le contact avec l'utilisateur est assuré en JavaScript.

Nous avons imposé une limite (depuis le SGBD) de 8Mo pour les envois de photos dans la base de données, cette limite est alors affichée à l'utilisateur par le biais de SweetAlert en JavaScript. En effet les alertes basiques de JavaScript sont peu ergonomiques, peu pratiques et vraiment laides. Nous utilisons donc SweetAlert qui nous permet de contrer ces points négatifs et surtout qui nous permet de les configurer en CSS. Voici donc les alertes dans les deux thèmes que propose PETAL :




Comme on peut le voir sur l'architecture des fichiers plus haut, les scripts PHP sont séparés des pages à afficher (répertoires PHP et HTML). Cela nous permet de pouvoir s'occuper de la page depuis un script à part et d'en faciliter ses mises à jour, d'en faciliter sa maintenance.

PHP gère également l'intégralité des communications avec la base de données. De plus, toutes les requêtes sont des requêtes préparées permettant de sécuriser celle-ci. Toutes les insertions sont vérifiées (taille de la valeur à entrer) avant qu'elles ne soient effectuées.

# V) Conclusion

Maintenant que nous avons expliqué les grandes lignes de PETAL. Nous pouvons donc lancer celui-ci. Je vais donc expliquer ici les différentes étapes permettant de mettre en place et d'utiliser

1) Téléchargez le site avec uwamp : https://github.com/samlach2222/PETAL/releases.
2) Récupérez l'un des deux scripts de remplissage de données (jeux de données) : https://github.com/samlach2222/PETAL/tree/main/BDD

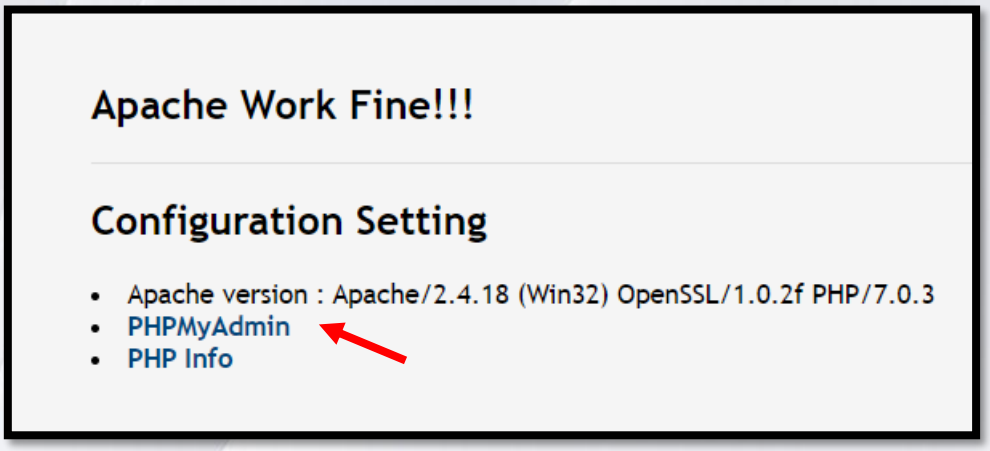
« PETAL données de test.sql » → Jeu de données complet possédant plusieurs utilisateurs et de nombreuse donnée

| Administrateur | Prénom | Nom | Mail | MotDePasse |
|---|---|---|---|---|
| **OUI** | Ada | Lovelace | Ada.Lovelace@mail.com | 1 |
| **NON** | Tim | Berners-Lee | Tim.Berners-Lee@mail.com | 2 |
| **NON** | George | Boole | George.Boole@mail.com | 3 |
| **OUI** | Grace | Hopper | Grace.Hopper@mail.com | 4 |
| **NON** | John | Mauchly | John.Mauchly@mail.com | 0 |

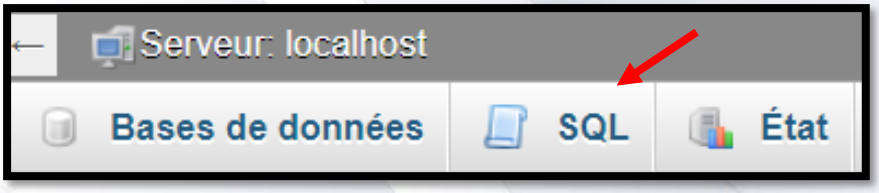
« PETAL base » → Jeu de données uniquement composé d'un administrateur possédant les identifiants suivants :

- Id : 1
- Nom : root
- Prenom : root
- Mail : root@mail.com
- MotDePasse : root

3) Lancez uwamp.exe et tapez « localhost » dans un navigateur web.
4) Cliquez sur phpMyAdmin :



5) Connectez-vous à PhpMyAdmin en utilisant les identifiants par défaut : root root.
6) Cliquez sur SQL dans la barre de navigation de la page :




7) Copiez-collez le contenu du fichier SQL chargé dans le champ SQL puis cliquez sur « Exécuter »
8) Une fois l'exécution terminée, rendez-vous sur la page de connexion de PETAL et connectez-vous :

# Contenu manquant

Nos pages sont fonctionnelles, permettent l'ensemble des fonctionnalités attendues dans le sujet. Cependant il y a un grand manquant : les QCM en XML. En effet par notre manière d'aborder la page des QCM ne nous permet pas, ou très difficilement d'aborder cette fonctionnalité. La page des QCM est dynamique à l'aide de JavaScript. On peut ajouter dynamiquement des nouvelles questions avec, si on le veut, des images. Ensuite lors de la publication ou de l'enregistrement du QCM, les données concernant le QCM (nom, date de fin, etc.) sont envoyées vers la table éponyme dans la BDD, puis chaque question est également envoyée dans la table question avec chacune des réponses. Ce mode de fonctionnement est facile à mettre en place, mais nécessite de nombreuses requêtes vers la BDD et est donc lourde pour la page et pour le trafic entre le front-end et le back-end. La solution de la mise en place de l'XML aurait pu rendre plus léger tout ce processus, mais par manque de temps, nous avons préféré choisir notre solution alternative, car bien que mauvaise sur un point de vue conception, elle nous permet tout de même d'implémenter la fonctionnalité des QCM.

Il nous manque également quelques petits points :

- Il n'y a pas de QCM initial pour déterminer les différentes matières d'un utilisateur
- Il n'y a pas de différence entre un professeur et un administrateur

# Améliorations possibles

J'ai, dans la partie précédente parlé de fonctionnalités absentes, bien que demandées dans le sujet, due à un manque de temps. Cependant il y a également des fonctionnalités supplémentaires, en partie développées ou images et conceptualisées qui n'ont pas pu être mises en place. Je vais donc les aborder ici. Commençons par les plus basiques d'entre elles :

- Pour le moment, seuls les administrateurs peuvent changer le mot de passe d'un utilisateur, ce qui rend impossible le changement par l'utilisateur lui-même.

- Un lien « mot de passe oublié » a été envisagé sur la conception et sur les maquettes. Or un problème de taille s'est posé à nous. Nous voulions envoyer un mail à l'utilisateur avec le mot de passe dedans (cela aurait également pu rejoindre le point précédent en mettant une page permettant de changer le mot de passe).

- Une case « se souvenir de moi » a été codée et implémentée dans ce commit : https://github.com/samlach2222/PETAL/commit/55850567b1ba73fa3b7a8a7c189c69b5e710c443. Or, à partir de ce commit, le hashage du mot de passe ne nous permet plus de garder cette option car elle va à l'encontre de la sécurité des données de la BDD, et surtout elle nous obligeait, en l'état à stocker en clair dans la base de données le mot de passes des utilisateurs : https://github.com/samlach2222/PETAL/commit/35670e6caefb06c272f71a6354199c8f88869ae8

- De la même manière qu'un changement de mot de passe du côté utilisateur aurait été bienvenue, le changement de photo de profil aussi.

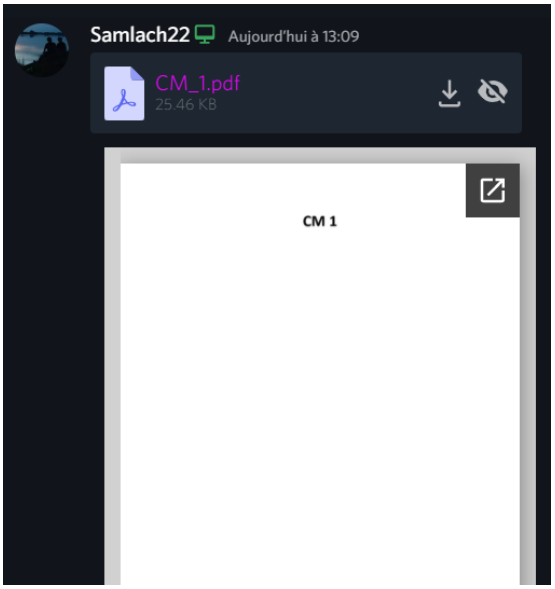
- Actuellement notre manière de gérer les images (photos de profil, ou tout autre fichier d'image faisant moins de 8Mo, la limite que l'on a imposée dans PETAL et dans la base de données) n'est pas la meilleure. En effet on les stocke directement dans la base de données dans des champs de type MEDIUMBLOB. Dans ces champs il est possible de mettre des fichiers qui seront alors stockés en hexadécimal. L'avantage de cette technique est que le fichier fait alors moins de taille que son format d'origine. Le désavantage de cette méthode est que dès que l'on veut l'afficher, nous l'encodons en base 64, et dès qu'on veut stocker une image dans la base de données, nous avons besoin de la décoder depuis de la base 64, processus couteux en temps et en ressource (à grande échelle bien sûr).

Le plus gros ajout qui a presque été mis au point est l'ouverture en affichage des différents fichiers directement depuis une page sur PETAL. En effet au lieu de proposer un téléchargement du fichier, nous aurions voulu proposer l'alternative de la visualisation directement sur le site. Pour des images, vidéos et sons, nous n'aurions aucun problème, cependant l'affichage de fichiers Microsoft Office, PDF ou bien même OBJ (objets 3D) sont complexes, très complexes.

Pour contrer cette complexité, nous nous sommes inspirés d'un Plugin que nous utilisons sur Discord qui s'appelle FileViewer : https://github.com/TheGreenPig/BetterDiscordPlugins/tree/main/FileViewer

Le plugin FileViewer permet d'ouvrir directement dans **Discord** à l'aide d'un **IFRAME** divers fichiers. Le plugin est développé en JavaScript et utilise divers services web : Google Drive pour les PDF, Office 365 Viewer pour les fichiers Office et viewstl pour les fichiers OBJ.

L'utilisation de ces services se fait de la manière suivante : Le plugin demande à l'un de ces 3 services une version embedded de ceux-ci avec en paramètre GET l'url du fichier Upload et l'affiche ensuite dans l'Iframe.




Notre idée à donc été de faire pareil, nous utilisons les mêmes services en ligne, et nous voulions les mettre dans un IFRAME sur notre page. Cependant, un gros problème empêche totalement la mise en place de ce processus. Nous exécutons notre site en local (localhost), le site est donc non-accessible depuis internet ainsi que toutes les données stockées. Nous ne pouvons donc pas envoyer nos fichiers par url GET car ils ne sont pas accessibles par les services. La fonctionnalité a donc été abandonnée.

Nous aurions pu mettre en place deux solution, cependant dans le cas d'un projet comme PETAL, elles ont toutes deux étés écartés :

- Héberger le site sur un véritable serveur web, accessible depuis internet (onéreux et non-utile dans le cadre d'un projet étudiant
- Envoyer une requête à une hébergeur de fichier en ligne qui possède des liens de téléchargement de fichiers directs, sans redirection, récupérer le lien et l'utiliser dans la requête au service (couteux en temps de développement, pour la planète, et peu éthique)