

# DOSSIER PROJET

DÉVELOPPEUR WEB ET WEB MOBILE

MONCHO Adrien



0659977054



admoncho@hotmail.com



Taglio Isolaccio

## Table des matières

INTRODUCTION.....	2
I- Liste des compétences couvertes par le projet.....	2
II- Résumé du projet.....	3
III- Cahier des charges .....	4
A – Spécificités techniques .....	4
IV- Spécifications fonctionnelles du projet.....	5
A – Description du fonctionnement de l'application .....	5
B – Architecture de l'application .....	6
C - La base de données.....	7
D - Les technologies .....	8
V – Réalisation du Candidat .....	9
A – Fonctionnalité de connexion.....	9
1- Style.....	10
2 - Javascript .....	10
3 - Requête Ajax .....	11
4 – PDO et Connexion à la base de données. ....	13
5 - Traitement du formulaire .....	14
B – Fonctionnalité Sidebar - tableau de bord .....	15
C – Fonctionnalité demande de congé .....	18
1 - Affichage des demandes en fonction du compte .....	21
VI – Présentation du jeu d'essai élaboré par le candidat .....	26
VII - Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité.....	27
VIII - Description d'une situation de travail ayant nécessité une recherche .....	28
IX - Extrait du site anglophone, utilisé dans le cadre de la recherche décrite précédemment .....	29



# INTRODUCTION

## **I- Liste des compétences couvertes par le projet.**

### **1- Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité :**

- a- Maquetter une application
- b- Réaliser une interface utilisateur web statique et adaptable
- c- Développer une interface utilisateur web dynamique

### **2- Développer la partie back-end d'une application web ou web mobile**

- a- Créer une base de données
- b- Développer les composants d'accès aux données
- c- Développer la partie back-end d'une application web ou web mobile

## II- Résumé du projet

Dans le cadre de ma certification de développeur d'application Web et Mobile et pour le besoin du centre de Formation Aflokkat, qui ne trouvait pas l'outil adéquat, il m'a été demandé de développer et de concevoir un ERP (**Enterprise Ressource Planning**) ou PGI (**Progiciel de Gestion Intégré**).

L'application a pour but d'assurer une gestion efficace de l'ensemble des collaborateurs tout en garantissant la confidentialité des informations et des échanges. Elle a été conçue pour simplifier, fluidifier et sécuriser la communication.

J'assure le développement du projet tant sur la partie Front End que Back End ainsi que la partie Base de données.

**AfloRH** centralise toutes les données de l'ensemble des salariés du centre et confère aux utilisateurs le droit de se connecter et se déconnecter du tableau de bord en fonction de leur rôle hiérarchique, avec une gestion des différents Rôles tels que Directeur, Stagiaire, Responsable RH, Formateur etc.

Les rôles du RH et du directeur posséderont tous les droits et privilèges nécessaires pour gérer efficacement l'application, comme la création de nouveau collaborateur ainsi que la création des rôles, ou traiter les différents types de congés.

Ces droits permettront l'utilisation de multiples fonctionnalités indispensables à la gestion optimisée des dossiers individuels de l'ensemble des collaborateurs tant dans l'approche règlementaire que managériale.

Ainsi, j'ai opté pour la mise en place d'un tableau dynamique intégrant l'état de présence de chaque collaborateur, qui se trouve en congé, en arrêt, ou en formation.

### III- Cahier des charges

Le centre de formation Aflokkat désire une application web de type gestion de ressources humaines :

Ne trouvant pas sur le marché le logiciel adéquat, il m'a été demandé de le concevoir, puis de le développer.

L'application sera découpée en plusieurs parties :

- Gestion des congés, suivi des collaborateurs (en congés, en arrêt maladie...).
- Dépôts des documents (arrêt maladie par exemple).
- Demande de rdv auprès des managers, validation des diverses demandes par les n+1, n+2.
- Création / Modification / Suppression / Validation de demandes : l'utilisateur crée une demande (congé, arrêt maladie ...) laquelle devra être validée par son supérieur direct.
- Mise en place de la plateforme de suivi des collaborateurs : l'utilisateur peut définir, suivre et vérifier la disponibilité des collaborateurs : il sait en temps réel qui est disponible, qui est en formation, qui est en arrêt, qui est en congé.

### A – Spécificités techniques

- Responsive Web Design : Oui (Mobile first)
- Moteur de recherche : (à venir)
- Gestions des utilisateurs : Oui
- Gestion des droits des utilisateurs : Oui
- Multilinguisme : (à définir)
- Live chat : (à venir)
- Progressive Web App : (à définir)
- Gestion de Token de sécurité : Oui

## IV- Spécifications fonctionnelles du projet

### A – Description du fonctionnement de l'application

Le projet se découpe en deux parties bien distinctes qui sont la page de connexion et de création de compte et la partie back office :

#### **La page connexion :**

L'utilisateur pourra tout simple créer un compte et se connecter, lorsqu'il créera son compte, il pourra choisir un rôle parmi une liste prédéfinie. Cette fonctionnalité permettra de définir les droits d'accès de l'utilisateur en fonction de son rôle dans l'entreprise.

Un contrôle des champs du formulaire a été mis en place afin d'apporter une sécurité supplémentaire au système.

Ces fonctionnalités seront amenées à évoluer dans une prochaine version du projet. Plus précisément, la partie création de compte sera modifiée pour devenir une demande d'adhésion permettant de restreindre l'accès à certaines parties du système aux seules personnes ayant été approuvées. Cette mesure vise à garantir la sécurité et la confidentialité des informations sensibles stockées dans le système de gestion des ressources humaines.

#### **Le Back office :**

Une fois connecté l'utilisateur aura accès à son tableau de bord et pourra choisir différentes options en fonction de son rôle dans l'entreprise. L'interface utilisateur sera adaptée en conséquence pour répondre aux besoins spécifiques de chaque utilisateur. Ci-après la liste des options :

- Voir l'état général de ses demandes
- Envoyer des documents
- Faire une demande de congé ou d'arrêt maladie
- Suivre les disponibilités de ses collaborateurs (suivi des con)
- Faire une demande rendez-vous
- Créer un compte Utilisateur
- Se déconnecter de son compte

Afin de répondre aux exigences du cahier des charges l'entièreté de l'application est responsive.

## B – Architecture de l'application

L'architecture 3 tiers ou multi-tier est une architecture de développement qui consiste à séparer les différentes couches de l'application en trois couches distinctes :



Le premier niveau (1) est l'interface utilisateur (UI) également appelée couche **Front-End**. Cette Couche est responsable de la gestion de l'interaction utilisateur, tel que la collecte et l'affichage de données, ainsi que la gestion des événements utilisateur. Elle communique (2) avec la couche applicative (3).

Le deuxième niveau (3) est la couche applicative, également appelé Back-End. Cette couche est responsable de la logique métier de l'application, du traitement des données et des règles métier. Elle communique(4) avec la couche de données (5).

Le troisième niveau (5) est la couche de données, qui est responsable du stockage et de la récupération des données. Cette couche peut être mise en œuvre à l'aide d'un système de gestion de base de données tel que SQL. Elle communique uniquement avec la couche applicative(7).

En utilisant une architecture 3 tiers, j'ai pu développer une application modulaire, maintenable et évolutif, car chaque couche peut être développée, testé et mise à jour indépendamment des autres couches. De plus elle facilite la maintenance et les mises à jour de l'application, car chaque couche peut être gérée séparément.

## C - La base de données

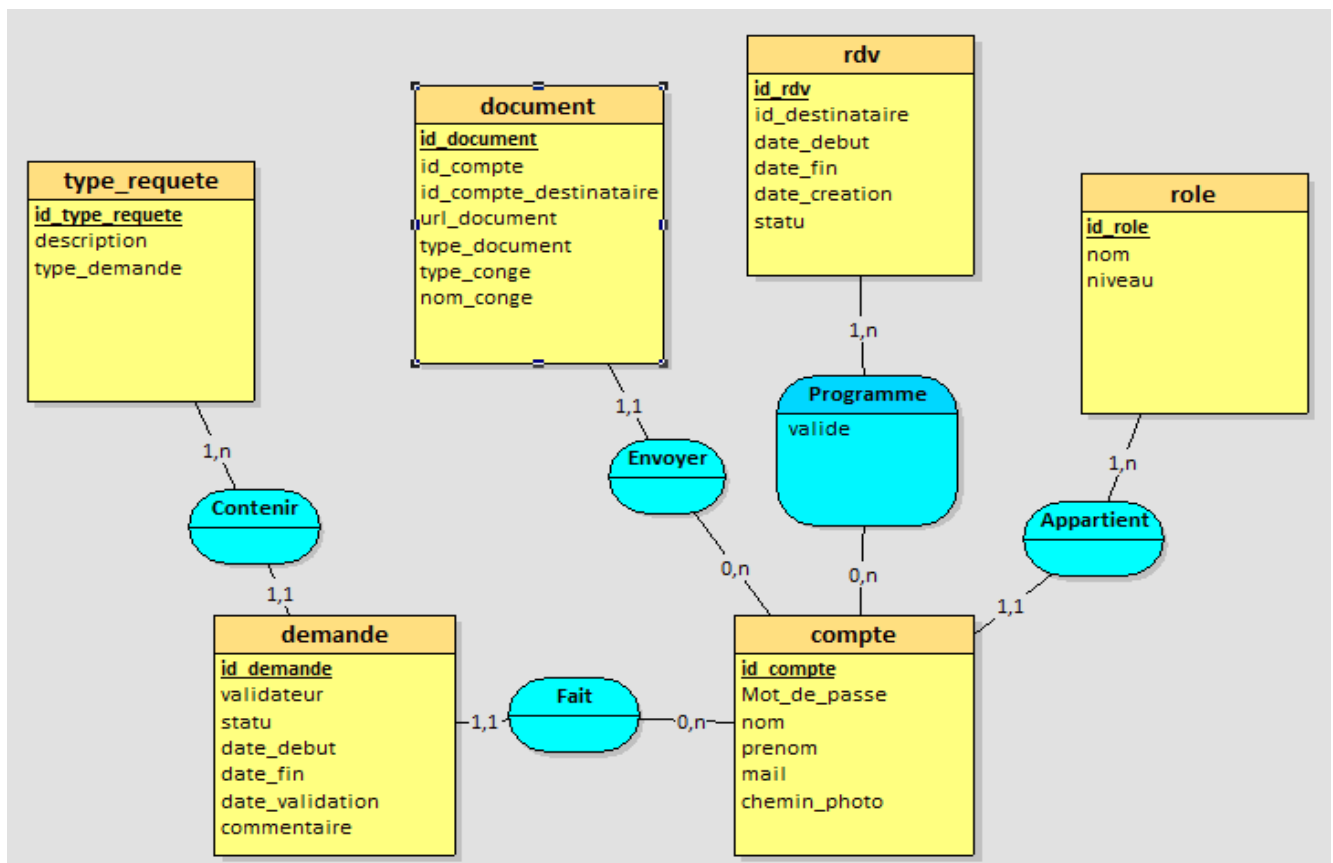


Figure 1: Schéma MCD de la base de données.

Afin de mieux comprendre le fonctionnement de l'application, je vais vous expliquer la structure de la base de données.

La table Compte est au centre de l'application, elle constitue la base et permet de visualiser les différentes relations entre les tables :

- Un Compte peut être associé à plusieurs rôles mais un rôle n'appartient qu'à un seul compte. Relation dite *one to many*.
- Un compte peut programmer plusieurs rendez-vous mais un rendez-vous ne peut être programmé que par un seul compte, pour gérer au mieux cette relation une table intermédiaire *rdv\_utilisateur* a été créée pour enregistrer les correspondances entre la table compte et rdv. Relation dite *many to many*.
- Un compte peut envoyer plusieurs documents mais un document ne peut être envoyé que par un seul compte. Relation dite *many to one*.
- Un compte peut faire une ou plusieurs demandes et une demande appartient à un seul compte.
- Une demande peut contenir plusieurs types de requête et un type de requête appartient à une seule demande.



## D - Les technologies

Liste des technologies Front End pour la réalisation du projet.

Liste des technologies Front End	
	Html5 et Css3 pour la base de structure de l'application et Style.
	Javascript pour la manipulation de DOM et écoute d'évènement.
	Librairie Javascript pour utilisation des requêtes Ajax.
	Bootstrap est une collection d'outils utiles à la création du design de sites et d'applications web. Il vient accompagner, l'ajout de composant complexe à réaliser à Html et CSS.
	Utilisation de la librairie SweetAlert.js, pour l'affichage de modal d'alerte.
Liste des technologies Back End	
	Pour pouvoir dynamiser l'application, traiter les formulaires, faire le lien avec ma base de données ...
	Utilisation MySQL le <b>système de gestion de données relationnelles</b> , pour créer ma base de données. Et utilisation du langage de programmation SQL pour l'écriture des requêtes.
Autres technologies :	
	Afin de m'aider dans le développement de l'application, j'ai utilisé l'outil de versionning Git.
	Utilisation Vs Code, Editeur de texte pour développeur.
	Utilisation de Whimsical, pour la création des wireframes de l'application.
	Utilisation d'un logiciel de modélisation conceptuelle de données.

## V – Réalisation du Candidat

Pour ce Chapitre, je vais vous exposer tout le processus de réalisation de fonctionnalité mentionné plus haut dans le *chapitre IV-A*, j'ai sélectionné les fonctionnalités les plus significative.

- Création et connexion à la Session
- Fonctionnement du tableau de Bord
- Processus de demande congé

## A – Fonctionnalité de connexion

Pour le formulaire de connexion et de création de compte, je suis parti sur un double formulaire animé en CSS et Javascript. J'ai commencé par établir la structure HTML.

```
<div class="container">
  <div class="bgGrey">
    <div class="box signin">
      <h2>Connectez-Vous</h2>
      <button class="signinBtn">S'identifier</button>
    </div>

    <div class="box signup">
      <h2>Demande d'adhésion?</h2>
      <button class="signupBtn">Adhérer</button>
    </div>
  </div>
  <div class="formBox">

    <div class="form signInForm">
      <form action="javascript:userConnect()" method="POST">
        <h3>Se connecter</h3>
        <input type="text" id="userMail" name="userMmail" placeholder="Votre Email" require>
        <input type="password" id="userPassword" name="userPassword" placeholder="Mot de passe" require>
        <input type="submit" name="connexion" value="Connexion" >
        <p>Vous avez oublié votre mot de passe ?
          <a href="#" class="forgot"> Cliquez ici</a>
        </p>
      </form>
    </div>

    <div class="form signUpForm">
      <form action="javascript:createCompte()" method="POST">
        <h3>Adhérer à AfloRH</h3>
        <input type="text" id="nameUser" name="NameUser" placeholder="Votre Nom" require>
        <input type="text" id="firstname" name="user" placeholder="Votre Prenom" require>
        <input type="email" id="mail" name="mail" placeholder="Adresse email" require>
        <input type="password" id="password" name="password" placeholder="Mot de passe" require>
        <label for="selectRole">Sélectionner un Rôle.</label>
        <select name="selectRole" id="selectRole"> </select>
        <input type="submit" name="user_register" value="Soumettre">
      </form>
    </div>
  </div>
</div>
```

## 1- Style

J'ai mis en place le style en initialisant des variables CSS, elles vont me permettre de :

```
:root {
  --text-white: #fff;
  --light-grey: #f2f2f2;
  --bg-grey: #d4d4d4;
  --blue-link: #03a9f4;
  --border-input: #333;
  /* Font-size */
  --text-size1: 3em;
  --text-size2: 2.2em;
  --text-size3: 1.5em;
  --text-size4: 1.2em;
  --text-size5: 1em;
  --text-size6: 0.7em;
}
```

1. Stocker des valeurs réutilisables.
2. Faciliter la maintenance et la modification des variables est réuni à un seul endroit.
3. Les noms des variables significatives peuvent aider à rendre le code plus lisible.

Figure 2 Initialisation des variables CSS

## 2 - Javascript

Afin de rendre dynamique et animer le formulaire, j'ai utilisé Javascript et la méthode de l'objet *document*.

*document*. *querySelector()* qui va me permettre de sélectionner un seul élément correspondant au sélecteur CSS spécifié, stocker dans une variable.

Je rappelle cette variable dans une fonction anonyme à un événement *onclick*. Dans la fonction, je rappelle mes autres variables et leur ajoute une classe *active*, ce qui aura pour effet de changer le comportement du formulaire. Dans le CSS je rappelle la classe *active* et l'assigne aux sélecteurs.

```
const signBtn = document.querySelector('.signinBtn');
const signupBtn = document.querySelector('.signupBtn');
const formBox = document.querySelector('.formBox');
const body = document.querySelector('body')

signupBtn.onclick = function(){
  formBox.classList.add('active');
  body.classList.add('active');
}
signinBtn.onclick = function(){
  formBox.classList.remove('active');
  body.classList.remove('active');
}
```

Figure 3: Code javascript

### 3 - Requête Ajax

Initialisation d'une fonction Ajax pour faire communiquer la parti front avec la partie Back end.

```
function userConnect() {  
  let userMail = $("#userMail").val();  
  let userPassword = $("#userPassword").val();
```

Dans cet extrait de requête Ajax, je commence par récupérer les valeurs de mes champs en utilisant la méthode **val()** de l'objet JQuery et en les stockant dans des variables.

```
$.ajax({  
  url: "php/formCtrl.php",  
  dataType: "JSON",  
  type: "POST",  
  data: {  
    request: "userConnect",  
    userMail: userMail,  
    userPassword: userPassword,  
  },
```

Je passe à la fonction **\$.ajax()** de JQuery qui va me permettre de faire un requête HTTP asynchrone. La requête pointe vers le fichier *formCtrl.PHP*, qui est le controller ou je traite mes formulaires.

```
success: function (response) {  
  if (response["status"] === 0) {  
    Swal.fire({  
      position: "top",  
      icon: "success",  
      title: response["msg"],  
      showConfirmButton: false,  
      timer: 3000,  
    });  
    setTimeout(() => {  
      document.location.href = "./php/admin.php";  
    }, "3000");
```

Ensuit on passe au *Error* ou *success*, Si le status défini coté PHP est strictement égale à 0, je rentre dans la parti Success, et la notification de la librairie javascript *Sweet Alert*, pour m'indiquer que je vais connecter, dans le cas contraire, faut réitérer la demande.

Coté PHP, J'utilise la structure de contrôle *switch case* pour gérer les différentes fonctionnalités de mes formulaires. Cette structure me permet de définir des cas correspondant aux différentes actions que l'utilisateur peut effectuer, et d'exécuter le bloc de code correspondant à chaque cas. De cette manière, je peux traiter les soumissions de formulaire de manière efficace et structurée, en fonction de l'action demandée par l'utilisateur.

```
switch ($_POST['request']) {
```

```
    case 'formConnect':  
        $nameUser = htmlspecialchars($_POST['nameUser']);  
        $firstname = htmlspecialchars($_POST['firstname']);  
        $mail = htmlspecialchars($_POST['mail']);
```

Dans l'exemple ci-dessus, pour que ma structure de contrôle fonctionne correctement, je récupère les valeurs qui ont été initialisées dans la requête en utilisant la variable superglobale `$_POST`. Ensuite, j'utilise le nom de la case correspondante en utilisant ces valeurs.

#### 4 – PDO et Connexion à la base de données.

Pour Réaliser et sécuriser l'application, j'ai utilisé la PDO (PHP Data Object) pour communiquer avec ma base de données. Elle offre plusieurs autres avantages.

```
$servName = 'localhost';
$username = 'dev';
$dbName = 'projet_rh';
$dbPassword = '!zP@vy@5P4Es].Ch';

// Connexion a la base de donnée
try{
    $db = new PDO("mysql:host=$servName; dbname=$dbName", $username, $dbPassword);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}catch(PDOException $e){
}
```

Figure 4: instantiation d'un nouvel objet PDO

#### Préparer les requêtes :

- La PDO me permet de préparer mes requêtes SQL.

#### Méthode BindParam() :

- Utilisation de la méthode *BindParam ()* dans la préparation des requêtes SQL afin d'être protégé contre les injections SQL.
- 

#### Méthode htmlspecialchars() :

- Utilisation de la méthode *htmlspecialchars ()* pour me prémunir des injections de script

#### Exécuter la requête

Après avoir préparé ma requête je peux l'exécuter.

```
$userMail = htmlspecialchars($_POST['userMail']);
$userPassword = $_POST['userPassword'];

$msg = "";
$status = 1;

$prepareSql = $db->prepare('SELECT `id_compte`, `mail`, `mot_de_passe`
                             FROM `compte`
                             WHERE `mail` = :mail ');
$prepareSql->bindParam('mail', $userMail);
$prepareSql->execute();
$recup = $prepareSql->fetch(PDO::FETCH_ASSOC);
```

## 5 - Traitement du formulaire

```
if ($recup) {  
    // error_log("1");  
    $userPasswordHash = $recup['mot_de_passe'];  
    if (password_verify($userPassword, $userPasswordHash)) {  
        $status = 0;  
        $msg = "Connexion Réussie !";  
        $userPasswordHash = password_hash($_POST['userPassword'], PASSWORD_BCRYPT);  
        $_SESSION['id_compte'] = $recup['id_compte'];  
    } else {  
        $status = 1;  
        $msg = "Votre mail ou votre mot de passe ne corresponde pas !";  
    }  
} else {  
    // error_log("4");  
    $status = 1;  
    $msg = "Vous devez entrer vos identifiants";  
}  
echo json_encode(array("msg" => $msg, "status" => $status));  
break;
```

Sur la parti création de compte je hashe le mot de passe, si toutes les conditions sont remplies, le mot passe est hashé et envoyé en base de données.

Dans l'exemple ci-dessus pour pouvoir me connecter, je vérifie si le mot de passe entré correspond, avec le mot de passe hashé en base de données avec la fonction **password\_verify()**, si tel est le cas la session se lance, et l'utilisateur est connecté.

Dans le cas contraire, Si les conditions requises ne sont pas remplies, le statut de la demande est modifié et un message d'alerte s'affiche à l'aide de la librairie Js *SweetAlert*. Cette fonctionnalité permet d'informer l'utilisateur que des actions supplémentaires sont nécessaires pour qu'il puisse se connecter.

## B – Fonctionnalité Sidebar - tableau de bord

Afin de faciliter la navigation, le tableau de bord comporte une sidebar pour rendre navigation intuitive. Ci – dessous la structure Html de la barre de navigation, le contenu du menu et générer dynamique avec du PHP.

```
<sidebar id="sidebar"></sidebar>
```

```
/* SIDEBAR */
#sidebar {
  position: fixed;
  max-width: 260px;
  width: 100%;
  background: var(--light);
  top: 0;
  left: 0;
  height: 100%;
  overflow-y: auto;
  scrollbar-width: none;
  transition: all 0.3s ease;
  z-index: 200;
}
```

Figure 5: extrait de code CSS de la sidebar

Pour ce faire une requêtes Ajax a été créer avec une instruction de code Javascript, qui va permettre de mettre à jours le contenu Html qui a l'id #sidebar.

La méthode **html()** de l'objet JQuery qui sélectionne l'id #sidebar pour remplacer le contenu html de cet élément par le contenu renvoyé par la requête AJAX. Le contenu à insérer est spécifié par **response['html']**.

```
///! B - 00 - Requete - Affichage des onglet en fonction de id_role
function loadMenu() {
  console.log(" B - 01 - Requete - Récupération des donnée.");
  $.ajax({
    url: "../php/displayData.php",
    dataType: "JSON",
    type: "POST",
    data: {
      request: "loadMenu",
    },
    success: function (response) {
      $("#sidebar").html(response["html"]);
      console.log("on est bon ");
    },
    error: function () {
      console.log("On est pas bon");
    },
  });
}
```



Le code suivant est le contenu de la sidebar, à la fin du code on peut voir la fonction PHP

**json\_encode()** pour renvoyer ces données à la requête Ajax.

```
$html .= '
<a href="#" class="brand"><i class="fa-brands fa-hive icon"></i> AfloRH</a>
<div class="user-sidebar">
|   <img src="" . $data['chemin_photo'] . "" alt="">
</div>
<ul id="sideBar" class="side-menu">
|   <li>
|   <a href="#" id="board" onclick="showComponent()" class="active">
|   <i class="fa-solid fa-house-user"></i> Dashboard</a>
|   </li>
|   <li class="divider" data-text="main">Main</li>

|   <li><a href="#" id="doc" onclick="displayDoc()">
|   <i class="fa-regular fa-folder-open"></i> Document</a>
|   </li>

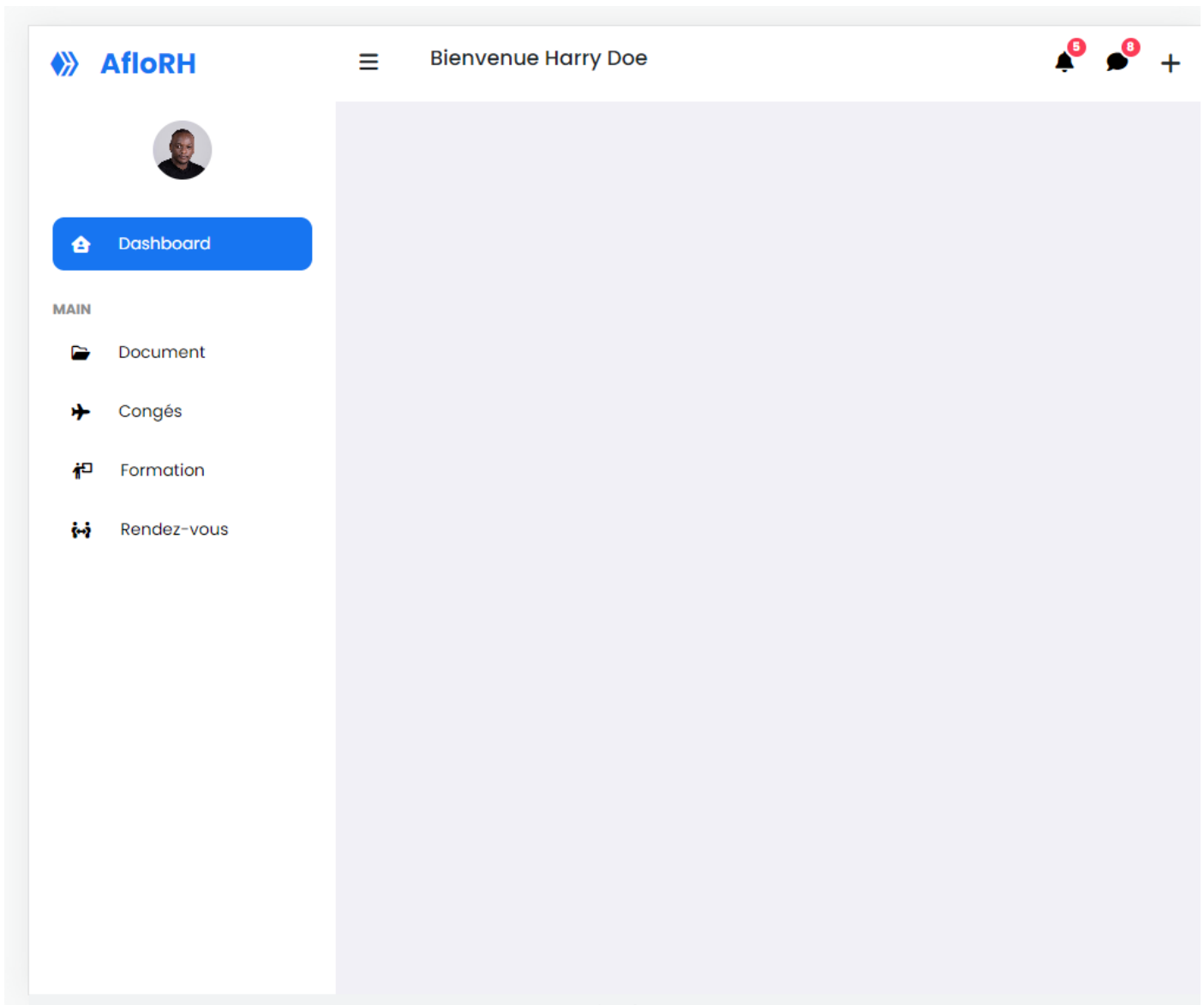
|   <li><a href="#" id="holiday" onclick="displayHoliday();">
|   <i class="fa-solid fa-plane"></i> Congés</a>
|   </li>

|   <!-- <li><a href="#"><i class="fa-solid fa-table-cells-large icon"></i> Ma
|   <li><a href="#" id="form" onclick="displayFormation()"><i class="fa-solid
|   <li><a href="#" id="rdv" onclick="displayRdv()"><i class="fa-solid fa-peop
|   </ul>';
// Stagiaire
}
```

```
echo json_encode(array("html" => $html, "status" => $status));
```

Ainsi le contenu de la sidebar s'affiche, Je réitère cette manipulation pour afficher le contenu des autres onglets en remplaçant l'id par une class *container-content* et en utilisant le même procédé, une requête AJAX pour chaque onglet. Cela permet de mettre à jour dynamiquement le contenu de la page.

## Maquette du tableau de bord



J'ai ajouté une fonction Javascript qui permet de réduire le menu lorsque l'on clique sur l'icône correspondante.

```
toggleSidebar.addEventListener("click", function () {  
  sidebar.classList.toggle("hide");  
});
```

Cet extrait de code Javascript ajout une écoute d'évènement au clique `addEventListener()`, Lorsque je vais cliquer sur l'icone de mon bouton la classe `hide` s'active et réduit ma sidebar.

## C – Fonctionnalité demande de congé

Maintenant que la navigation et le contenu se mettent à jour, j'ai accès au contenu de mes onglets. Je vais parler ici de la fonctionnalité pour faire une demande congé ou arrêt.

J'ai opté pour cette fonctionnalité, d'un double formulaire, un formulaire de demande d'arrêt maladie et un formulaire de demande congé, à cela j'ai ajouté deux interfaces, une pour voir les demandes en attentes, et une autre les demandes traitées. Les demandes traitées seront validées par le rôle supérieur.

*Exemple : une demande d'un formateur sera validée par le/la responsable en ressource humaine, et les demandes de le/ la responsable en ressource humaine sera validée par le directeur.*

Sur l'image ci-dessous, nous avons la navigation fonctionnant sur le même principe de la sidebar. Je fais appels à des fonctions javascript onclick () qui font appel à des requêtes Ajax.

```
$html .= '
<div onclick="displayHoliday()" id="arret" class="tabs__toggle is-active">Arrêt Maladie</div>
<div onclick="tabHoliday()" id="holidayForm" class="tabs__toggle">Congés</div>
<div onclick="tabDemandInProgress()" id="tabDemandInProgress" class="tabs__toggle">Demande en cours</div>
<div onclick="tabRequestValidated()" id="tabRequestValidated" class="tabs__toggle">Demande traitée</div>';
```

```
function tabHoliday() {
    $.ajax({
        url: "../php/tabFormCtrl.php",
        dataType: "JSON",
        type: "POST",
        data: {
            request: "tabHoliday",
        },
        success: function (response) {
            $(".tabs__toggle").removeClass("is-active");
            $(".tabs__toggle-responsive").removeClass("is-active");
            $("#holidayForm").addClass("is-active");
            $("#holiday_responsive").addClass("is-active");
            $(".tabs__content").html(response);
        },
        error: function () {
            console.log("On est pas bon");
        },
    });
}
```

Figure 6: Extrait d'une des requêtes Ajax qui gère l'affichage du deuxième formulaire.

Le Formulaire utilise une autre fonction Javascript, qui permet de gérer le traitement et l'envoi des données.

```
$html .= '<div class="tabs__title">
<h2>Demande de Congés</h2>
<div class="tabs__form">
    <div class="content-form">
        You, il y a 4 semaines • Projet_
        <form class="formHoliday" action="javascript:sendHoliday()">
            <div class="form-up">
                <div class="form-left">
```

Figure 7: Fonction Javascript qui renvoie vers une autre requête Ajax pour traiter le formulaire

## Select Dynamique

	id_type_requete	description	type_demande
<input type="checkbox"/> Éditer Copier Supprimer	1	Conges maladie Ordinaire(CMO)	1
<input type="checkbox"/> Éditer Copier Supprimer	9	Conges longue maladie (CLM)	1
<input type="checkbox"/> Éditer Copier Supprimer	12	Conges maladie longue duree(CLD)	1
<input type="checkbox"/> Éditer Copier Supprimer	2	Conges Paternite	2
<input type="checkbox"/> Éditer Copier Supprimer	3	Conges Maternite	2
<input type="checkbox"/> Éditer Copier Supprimer	4	Conges Parental	2

Figure 8: Extrait de base de données pour afficher les différents types de demande

Pour pouvoir afficher toutes les demandes de congé ou d'arrêt, j'ai dû créer un menu select dynamique. Je prépare une requête SQL qui me sélectionne tous les types de demande qui a pour valeur 2 afin de récupérer toutes les descriptions de type\_demande 2.

```
$execSql = $db->prepare('SELECT `id_type_requete`, `description`, `type_demande`
FROM `type_requete`
WHERE `type_demande` = 2');

$execSql->execute();
```

Ensuite j'exécute ma requête, et j'utilise la méthode `fetchAll` de l'objet PDO, pour qui me retourne un tableau associatif et me permet de récupérer toutes mes valeurs. Je stock cette méthode dans une variable.

```
$selectHolidays = $execSql->fetchAll(PDO::FETCH_ASSOC);
```

Ensuite je créer une boucle *foreach*, je lui passe la variable de l'objet à parcourir et la variable qui sera mise à jour. Je peux maintenant afficher mes données avec `fetchAll`.

```
<select name="" id="selecHoliday">;
foreach ($selectHolidays as $selectHoliday) {
    $html .= '<option value="' . $selectHoliday['type_demande'] . '>' . $selectHoliday['description'] . '</option>';
}
$html .= ' </select>
```

Pour finir, j'utilise la fonction `document.ready` de JQuery afin qu'au chargement de la page ma fonction se lance.

## 1 - Affichage des demandes en fonction du compte

Une fois qu'une demande de congé a été enregistrée, je voulais faire en sorte, que l'utilisateur connecté puisse voir uniquement sa demande.

```
$insertSql = $db->prepare('SELECT * FROM `compte`
INNER JOIN `role` ON `compte`.`id_role` = `role`.`id_role`
WHERE `compte`.`id_compte` = ' . $_SESSION['id_compte']);
$insertSql->execute();
$checkRequestCompte = $insertSql->fetch(PDO::FETCH_ASSOC);
```

La requête ci-dessus me permet de récupérer les informations du compte et celle du role associé pour l'utilisateur connecté. Maintenant je veux récupérer les demandes en attentes de ce compte.

Pour cela j'ai fait une deuxième requête :

```
$insertSql1 = $db->prepare('SELECT * FROM `compte`
INNER JOIN `demande` ON `compte`.`id_compte` = `demande`.`id_compte`
INNER JOIN `role` ON `compte`.`id_role` = `role`.`id_role`
WHERE `compte`.`id_compte` = ' . $_SESSION['id_compte'] . ' AND `demande`.`statu` = 1');
```

Cette requête me permet de sélectionner le compte avec une demande qui a un statut 1 associé à eux.

Pour finir je refais la même procédure pour les demandes qui ont un statut 2.

Pour ce qui est du role RH, même étape, elle va pouvoir consulter ses demandes mais aussi valider les demandes qui sont supérieur à son rang.

```
$insertSql1 = $db->prepare('SELECT * FROM `compte`
INNER JOIN `demande` ON `compte`.`id_compte` = `demande`.`id_compte`
INNER JOIN `role` ON `compte`.`id_role` = `role`.`id_role`
WHERE `role`.`niveau` > ' . $_SESSION['level'] . ' AND `demande`.`statu` = 1');
```

Cette requête va me permettre de récupérer toutes les demandes en attentes qui ont un niveau de supérieur a celui de l'utilisateur connecté.

Je peux faire la même manipulation, en changeant la requête pour que le role de directeur ne valide que les demandes de la responsable en ressources humaine. Pour clore cette fonctionnalité le directeur ne pourra pas faire de demande de congé.

Les différentes demandes pourront être supprimé ou validé par la Rh et/ou le Directeur.

## Capture d'écran des différents partis du formulaire

Rendu visuel du formulaire demande congé, j'ai rajouté un script javascript, afin d'afficher l'onglet active lorsque je change d'onglet.

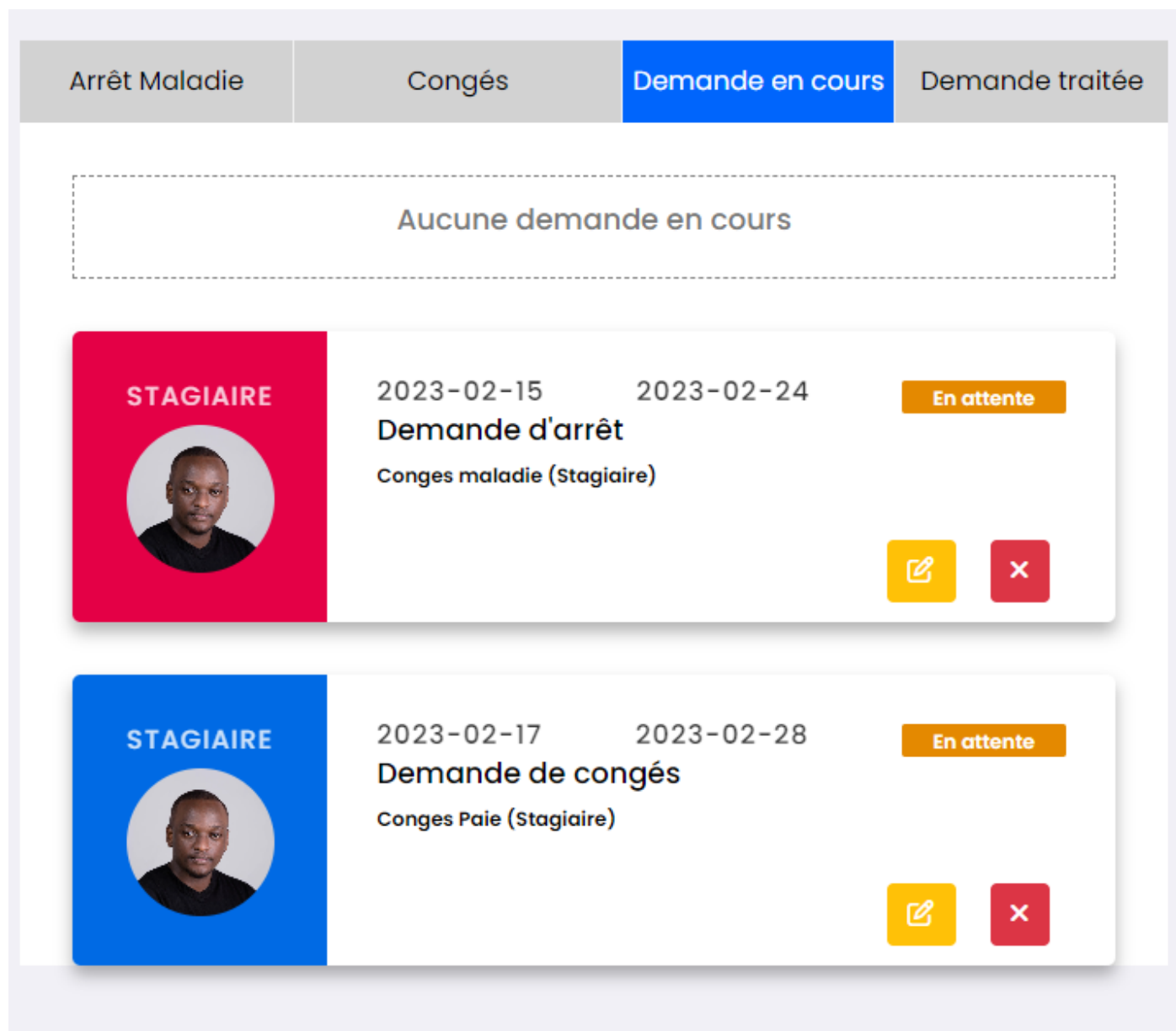
```
<div onclick="tabHoliday()" id="holidayForm" class="tabs__toggle">
<div onclick="tabHoliday()" id="holiday_responsive" class=" tabs__to
```

```
$(".tabs__toggle").removeClass("is-active");
$(".tabs__toggle-responsive").removeClass("is-active");
$("#holidayForm").addClass("is-active");
$("#holiday_responsive").addClass("is-active");
$(".tabs__content").html(response);
```

Ce code va supprimer la classe *is-active* de tous les éléments qui ont la classe *tabs\_\_toggle*, ensuite il ajoute la classe *is-active* aux éléments avec l'id *holidayForm*.

## L'interface d'un utilisateur

J'ai fait en sorte associé un code couleur à mes cartes afin de différencier les demandes d'arrêt, des demandes de congé.





**Interface du point de vue de la responsable en ressource humaine.**

Arrêt Maladie


Congés

Demande en cours

Demande traitée

Aucune demande en cours

RESSOURCE HUMAINE




2023-02-17


2023-02-24

En attente

Demande d'arrêt


Conges maladie Ordinaire (RH)





Demande d'arrêt à traiter

STAGIAIRE




2023-02-15


2023-02-24

En attente


Demande d'arrêt

Conges maladie (Stagiaire)





FORMATEUR




2023-02-13


2023-02-26

En attente

Demande d'arrêt

Conges maladie Ordinaire (Formateur)





**Interface du point de vue du directeur.**

Demande en cours	Demande traitée
<h2>Demander un arrêt maladie</h2> <p><b>Date de début</b></p> <input type="text" value="jj/mm/aaaa"/> <div></div> <p><b>Date de fin</b></p> <input type="text" value="jj/mm/aaaa"/> <div></div> <p><b>Sélectionner un type d'arrêt</b></p> <div>Conges maladie Ordinaire(CMO) <div></div></div> <div>test</div> <div>envoyer</div>	

## VI – Présentation du jeu d'essai élaboré par le candidat

Action	Résultat obtenu	Résultat attendu	Fonctionnel
L'utilisateur créer un compte	Le compte est bien créer		oui
L'utilisateur se connecte a son compte	L'utilisateur peut se connecter à sa Session		oui
Création d'une demande de congé ou d'arrêt maladie	L'utilisateur peut créer une demande		oui
L'utilisateur peut visualiser sa demande la modifier ou la supprimer	L'utilisateur peut visualiser sa demande		oui
Visualisation des demandes à traiter	(Pour Rh seulement), elle peut visualiser les demande de ses		oui
L'utilisateur peut se déconnecter	oui les l'utilisateur peut se déconnecter		oui

## VII - Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité.

### La Sécurité

Les mots de passe sont souvent la première ligne de défense contre les cyberattaques cherchant toujours à accéder à des informations sensibles stockées dans une base de données.

Afin de renforcer la sécurité des mots de passe stockés en base de données, les techniques de hachage sont largement utilisées pour chiffrer les mots de passe. Parmi les meilleures pratiques actuelles, la fonction PHP ***password\_hash()*** est souvent recommandée, car elle permet de hacher le mot de passe en utilisant l'algorithme ***PASSWORD\_BCRYPT*** considéré comme sûr pour le moment et non déchiffrable.

Chiffrer les mots de passe est une mesure de sécurité important, mais elle ne constitue pas une solution ultime en matière de sécurité. Il est également essentiel de mettre en place un processus de vérification des champs coté PHP :

**En PHP :** Il est important de s'assurer que les données saisies sont valides, et ne contiennent pas de codes malveillants, d'injection de script ou d'injection SQL.

Encore une fois PHP fait bien les choses et nous donne tous un panel d'outils pour éviter les injections :

- La fonction ***htmlspecialchars()***, va permettre d'encoder les caractères spéciaux et ainsi éviter les attaques d'injection de script.
- Utilisation des fonction ***bindValue()*** et /ou ***bindParam()*** peuvent être utilisées pour lier des valeurs au paramètre d'une requête SQL, ce qui empêche l'injection de code malveillant dans les requêtes SQL.

## VIII - Description d'une situation de travail ayant nécessité une recherche

Durant tout le long du projet, j'ai été confronté à des répétitions de code, notamment, lorsque je faisais deux requêtes SQL, et que je répéter deux boucle foreach. Je me suis demandé alors « n'y a-t-il pas un moyen d'optimiser ce code ? »

En cherchant un peu je suis tombé sur une fonction PHP *array\_merge()* qui me permet de fusionner plusieurs tableaux. Elle prend en argument un ou plusieurs tableaux et renvoie un tableau qui contient tous les éléments des tableaux d'entrée.

```
$selectRequests = array_merge($insertSql1->fetchAll(PDO::FETCH_ASSOC), $insertSql2->fetchAll(PDO::FETCH_ASSOC));
```

## IX - Extrait du site anglophone, utilisé dans le cadre de la recherche décrite précédemment

### array\_merge

(PHP 4, PHP 5, PHP 7, PHP 8)

array\_merge — Merge one or more arrays

#### Description

```
array_merge(array ...$arrays): array
```

Merges the elements of one or more arrays together so that the values of one are appended to the end of the previous one. It returns the resulting array.

If the input arrays have the same string keys, then the later value for that key will overwrite the previous one. If, however, the arrays contain numeric keys, the later value will *not* overwrite the original value, but will be appended.

Values in the input arrays with numeric keys will be renumbered with incrementing keys starting from zero in the result array.

*Rassemble les éléments d'un ou plusieurs tableaux en poussant les valeurs de l'un à la fin de l'autre. Cela nous retourne un tableau.*

*Si les tableaux en entrées ont les mêmes clés, alors la valeur finale pour cette clé remplacera la précédente. Cependant, si les tableaux contiennent des clés numériques, la valeur finale ne remplacera pas la valeur initiale mais l'ajoutera à l'existant.*

*Les clés numériques des tableaux d'entrées seront renumérotées avec une nouvelle incrémentation en partant de zéro dans le tableau final.*