Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance - EWEMBE
Nom d'usage - Entrez votre nom d'usage ici.
Prénom - Dhellye Shade
Adresse - 926 Rue Font-couverte, 34070 Montpellier

Titre professionnel visé

TP5 – Développeur Web et Mobile

MODALITÉ D'ACCÈS :

☒ Parcours de formation
☐ Validation des Acquis de l'Expérience (VAE)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.
Ce titre est délivré par le Ministère chargé de l'emploi.

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen**.

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel** (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- une déclaration sur l'honneur à compléter et à signer ;
- des documents illustrant la pratique professionnelle du candidat (facultatif)
- des annexes, si nécessaire.

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

http://travail-emploi.gouv.fr/titres-professionnels

Sommaire

Exemples de pratique professionnelle

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Compétence n°1 *Installer et configurer son environnement de travail en fonction du projet web ou web mobile*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Le processus d'installation et de configuration de l'environnement de travail est une étape importante dans notre projet de développement. Nous avons commencé par l'installation de Visual Studio Code comme éditeur de code principal, choisi pour sa polyvalence et ses nombreuses fonctionnalités adaptées au développement web.

En parallèle, nous avons installé MAMP pour disposer d'un environnement local complet intégrant

- Apache : C'est un serveur web très populaire. Il gère les requêtes des utilisateurs et envoie les pages web en réponse.
- MySQL : C'est un système de gestion de base de données relationnelle. Il stocke et gère les données du site web de manière efficace.
- PHP : C'est un langage de programmation côté serveur. Il permet de créer du contenu dynamique, d'interagir avec la base de données et de traiter les formulaires.

MAMP nous a permis de gérer efficacement notre base de données locale à travers phpMyAdmin, facilitant ainsi les tests et le développement.

Pour notre micro-framework PHP Modèle-Vue-Contrôleur, nous avons procédé au téléchargement du fichier ZIP contenant le framework. Après avoir décompressé l'archive dans le répertoire de notre choix, nous avons installé les dépendances nécessaires avec la commande "composer install".

Pour la gestion de la base de données, nous avons utilisé phpMyAdmin via MAMP pour créer et configurer notre base de données. Les paramètres de connexion à la base de données ont été implémentés dans le fichier .env du framework, garantissant une connexion sécurisée et efficace entre notre application et les données.

L'ensemble de ces installations et configurations nous a permis de disposer d'un environnement de développement complet, stable et sécurisé, parfaitement adapté aux besoins de notre projet et facilitant le processus de développement web

2. Précisez les moyens utilisés :

Pour mener à bien mon projet, j'ai utilisé plusieurs outils essentiels :

- **Éditeur de code** : Visual Studio Code pour écrire et organiser mon code efficacement.
- **Serveur de développement local** : MAMP pour tester mon application en local avant le déploiement.
- **Système de contrôle de version** : Git pour gérer les différentes versions de mon projet et assurer un suivi rigoureux des modifications.

3. Avec qui avez-vous travaillé ?

J'ai installée et configuré mon environnement de travail seul, en m'appuyant sur des ressources en ligne et la documentation des outils utilisés.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *Diginamic*

Chantier, atelier, service ▸ Projet en formation

Période d'exercice ▸ Du : 16*/09/2024* au : *20/09/2024*

5. Informations complémentaires *(facultatif)*

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Compétence n°2 ▸ *Maquette des interfaces utilisateur web ou web mobile*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour répondre à la demande de l'entreprise KENTECK, qui souhaitait un logiciel de gestion interne pour leurs projets, nous avons débuté par l'analyse des besoins des différents profils d'utilisateurs (PDG, chef de projet, développeurs). Parmi ces trois profils, nous avons identifié le chef de projet comme utilisateur principal, il est au cœur de la gestion des tâches et du suivi des projets. Cette décision stratégique nous a permis de structurer notre approche de maquettage en nous concentrant d'abord sur ses besoins spécifiques.

Le processus de maquettage des interfaces utilisateur a été réalisé sur Figma, avec une approche centrée sur la simplicité et l'efficacité, spécifiquement orientée vers les besoins du chef de projet. Notre conception s'est focalisée principalement sur le format bureau, en accord avec l'utilisation prévue de l'application dans un contexte professionnel.

Suite à une série d'échanges avec le **chef de projet** de KENTECK, nous avons pu établir une **user story** détaillée reflétant ses besoins:

En tant que	Je souhaite	Afin
Chef de projet	Créer (CRUD) des utilisateurs avec le rôle développeur	De gérer efficacement les accès et les permissions des membres de l'équipe de développement
Chef de projet	Créer(CRUD) des projets	D'organiser et suivre les différents projets de l'entreprise de manière structurée
Chef de projet	Créer des tâches prévisionnelles	De planifier et répartir efficacement la charge de travail entre les développeurs
Chef de projet	Génère des rapports personnalisés sur les projets réalisés qui me permettent de visualiser les données sous forme de tableaux ou de graphiques.	D'analyser la performance des projets et prendre des décisions éclairées basées sur des données concrètes
Chef de projet	Voir le planning semaine par semaine de tous les développeurs	D'avoir une vision claire de la disponibilité et de la charge de travail de chaque développeur
Chef de projet	Voir l'agenda de tous les développeurs (tâches prévisionnelles + tâches réelles)	De suivre l'avancement réel des tâches et d'identifier les écarts potentiels avec le planning prévisionnel

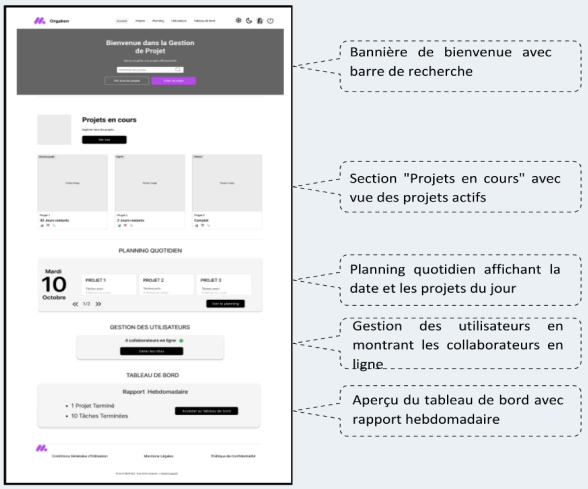
En nous basant sur les user stories détaillées du chef de projet, nous avons créé notre maquette sur Figma. Notre priorité était d'avoir une interface intuitive répondant aux six besoins principaux identifiés : la gestion des utilisateurs développeurs, la création de projets, la gestion des tâches prévisionnelles, la génération de rapports, la visualisation du planning hebdomadaire et le suivi de l'agenda des développeurs.

La structure de notre maquette s'est articulée autour d'un menu de navigation principal donnant accès aux cinq pages essentielles :

- Accueil : Page d'entrée de l'application
- Projets : Gestion et suivi des projets
- Planning : Vue hebdomadaire des tâches
- Utilisateurs : Gestion CRUD des développeurs
- Tableau de bord : Vue d'ensemble avec métriques

Nous aurons aussi une page de connexion et de support, ce sont des pages secondaires. Elles font partie de l'interface mais ne sont pas dans le menu principal car elles ont des fonctions spécifiques : la connexion est préalable à l'utilisation, et le support est accessible via un lien en pied de page.

ACCEUIL



Bannière de bienvenue avec barre de recherche

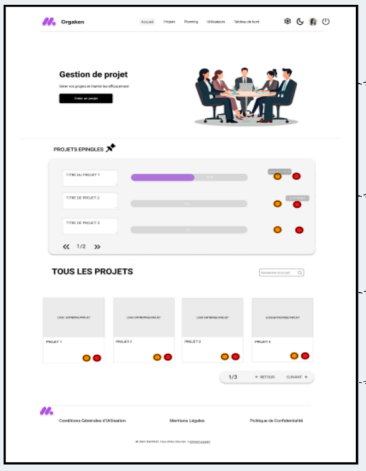
Section "Projets en cours" avec vue des projets actifs

Planning quotidien affichant la date et les projets du jour

Gestion des utilisateurs en montrant les collaborateurs en ligne

Aperçu du tableau de bord avec rapport hebdomadaire

PROJETS



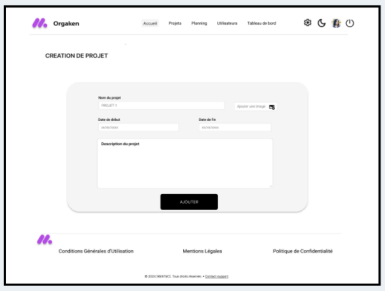
Entête avec titre "Gestion de projet" et Bouton pour la creation de project

Section "Projets épinglés" avec barre de progression et contrôles et bouton supprimer & modifier

Section "Tous les projets" avec vue en grille des projets & barre de recherche

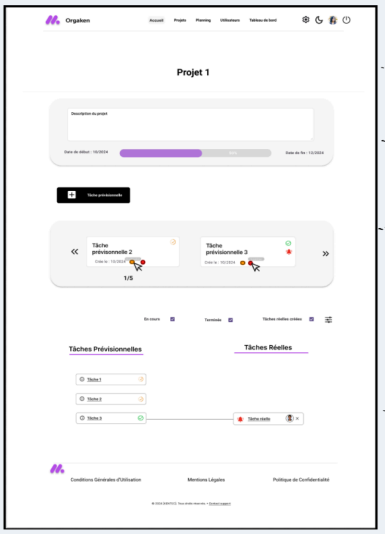
Système de pagination

FORMULAIRE CREATION DE PROJET



Dans la page gestion de projet, une fois un projet créé via le formulaire (nom, dates, description), nous accédons à sa page détaillée

INFORMATION SUR LE PROJET



Vue détaillée du projet : Titre du projet

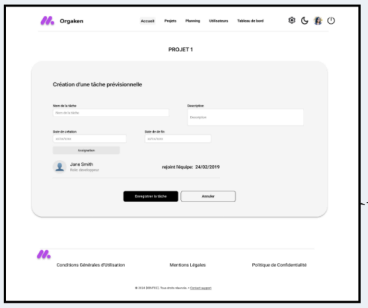
Description du projet, Barre de progression avec date de début et de fin

Créer des tâches prévisionnelles via le bouton dédié dès que c'est fait nous avons la possibilité de le supprimer ou modifier

Visualiser les tâches existantes dans deux sections

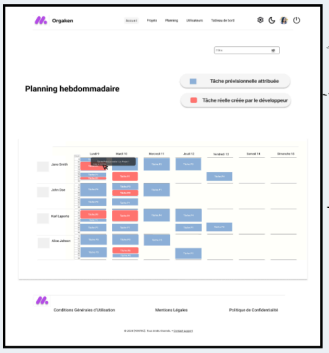
- Tâches Prévisionnelles : liste des tâches planifiées
- Tâches Réelles : tâches effectivement créées par les développeurs

CREATION TACHES PREVISIONNELLES



Dans la page détaillée du projet, le bouton "Tâche prévisionnelle" ouvre ce formulaire de création.

ACCEUIL

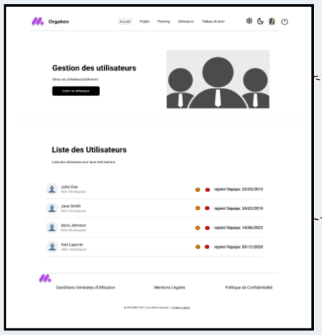


Filtre en haut à droite

Légende des types de tâches :bleu pour tâches prévisionnelles, rouge pour tâches réelles

Vue hebdomadaire du lundi au dimanche, liste des développeurs sur la gauche

UTILISATEUR



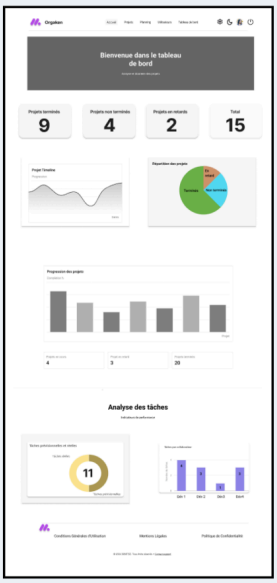
Entête avec titre "Gestion des utilisateurs" et Bouton pour la création d'un utilisateur

Section "Liste des utilisateurs" avec son non/prénom suivi du bouton supprimer & modifier

CREATION D'UTILISATEUR

Formulaire de creation d utilisateur avec nom/prénom/role & bouton enregistrer ou annule

TABLEAU DE BORD



Cards des métriques (projets terminés, non terminés, en retard)

Section "Projets en cours" avec vue des projets actifs

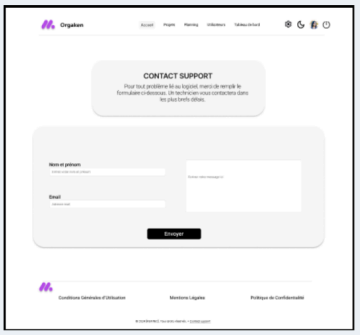
Graphique à barres de progression des projets

Section analyse des tâches avec graphiques de performance des devellopeurs

CONNEXION

Formulaire avec champs identifiant/mot de passe & Option "mot de passe oublié"

CONTACT



Formulaire épuré avec champs nom/prénom/email & bouton envoye

2. Précisez les moyens utilisés :

Pour concevoir mon projet, j'ai utilisé plusieurs outils :

- **Outil de prototypage** : Figma, pour créer des maquettes interactives et définir l'interface utilisateur.
- **Tableau numérique HUAWAI IdeaHub S 2 65 Pouces présent à Diginamic** : Pour réaliser les premières esquisses et structurer visuellement les idées avant la conception numérique.

3. Avec qui avez-vous travaillé ?

Pour la réalisation de la maquette, j'ai travaillé en groupe avec trois autres personnes. Nous étions quatre au total, chargés de concevoir et structurer l'interface du projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *Diginamic*

Chantier, atelier, service ▸ Projet en formation

Période d'exercice ▸ Du : *16/10/2024* au : *20/10/2024*

5. Informations complémentaires *(facultatif)*

La maquette complète du projet est disponible sur Figma à l'adresse suivante :
https://www.figma.com/design/QyPVPh3b6UTYSs0YSKijFw/ORGA-KEN?node-id=0-1&p=f&t=brpWvGUSG4HolEKn-0

Cette maquette présente l'ensemble des interfaces utilisateur, du design système et des composants.

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Compétence n°3 *Réaliser des interfaces utilisateur statiques web ou web mobile*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Suite à la validation de la maquette, j'ai été chargée de développer la page de gestion des utilisateurs en commençant par mettre en place une architecture modulaire. Le framework utilisé base.php comme template principal, où nous avons décidé de séparer le header et le footer dans des fichiers distincts pour optimiser la maintenance du code :

```
<body>
    <?php include('header.php'); ?>

    <main>
        <div class="container">
        <?php include_once $view; ?>
        </div>
    </main>

    <?php include('footer.php'); ?>

    <script src="/build/app.js"></script>
</body>
```

Cette organisation modulaire facilite la maintenance et évite la duplication de code, avec le JavaScript chargé en fin de page pour optimiser les performances.

La page de gestion des utilisateurs s'appuie sur une structure HTML5 sémantique organisée en deux sections principales.
La section (user-section) présente une mise en page en deux colonnes : une colonne avec le contenu textuel incluant un titre h1 "Gestion des utilisateurs", un paragraphe explicatif et un bouton pour créer un utilisateur et une colonne avec une image d'illustration.

```
<section class="user-section">
    <div class="user-content">
        <h1>Gestion des utilisateurs</h1>
        <p>Gérez vos utilisateurs facilement</p>
        <button id="addUserBtn" class="btn btn-primary">Créer un utilisateur</button>
    </div>
    <div class="primaryimg">
        <img src="build/images/user.png" alt="Utilisateurs" width="500" height="400">
    </div>
```

La seconde section (user-content) affiche la liste des utilisateurs avec un titre h2 et un descriptif.

```
<section class="user-content">
    <h2>Liste des Utilisateurs</h2>
    <p>Liste des utilisateurs avec leurs informations</p>
```

En effet, j'ai utilisé des éléments HTML sémantiques (titres hiérarchisés, paragraphes, boutons, image avec attribut alt) et des conteneurs div avec des classes explicites, créant ainsi une base robuste pour l'application des styles CSS.
Pour le CSS, j'ai appliqué un style en utilisant des techniques modernes :

```
.user-content h2 {
    font-size: 1.5rem;
    margin-bottom: 0.5rem;
    font-weight: bold;
    color: $text-color;
}

.primaryimg {
    flex: 1;
    display: flex;
    justify-content: flex-end;
}

#addUserBtn {
    background-color: $primary;
    border: none;
    padding: 0.75rem 1.5rem;
    border-radius: 0.25rem;
    font-weight: 500;
    color: $white;
}
```

```
.btn-edit, .btn-delete {
    background: none;
    border: none;
    padding: 0.5rem;
    cursor: pointer;
    transition: transform 0.2s ease, color 0.2s ease;

    i {
        color: inherit;
    }

    &:hover {
        transform: scale(1.1);
    }
}

.btn-edit .fas  {
    color: $orange;
    &:hover {
        color: darken($orange, 10%);
    }
}
```

Les titres et textes sont stylisés pour une hiérarchie visuelle claire, avec des tailles (1,5rem pour h2) et des marges appropriées.

Pour l'agencement, j'utilise Flexbox sur la classe Primaryimg afin d'aligner l'image correctement. Le bouton d'ajout présente un design distinctif avec une couleur primaire ($primary: #6637EE;)

et des bords arrondis. Les boutons d'action ont un style minimaliste avec des transitions sur le survol, incluant un effet d'échelle et des changements de couleur progressifs (facilité de 0,2 s). L'utilisation de variables ($primary, $orange) assure la cohérence des couleurs à travers l'interface.

Cependant, notre page n'est pas une interface complètement statique car elle intègre des éléments dynamiques comme:
- l'affichage dynamique des utilisateurs avec PHP
- Formulaire modal interactif pour ajouter/modifier les utilisateurs
- Boutons d'action dynamiques (édition, suppression).

Ces aspects dynamiques seront détaillés dans la compétence 4 : Développer la partie dynamique des interfaces utilisateur web ou web mobile.

2. Précisez les moyens utilisés :

J'ai utilisé **HTML5** pour structurer le contenu, **CSS** pour la mise en forme et le design, et **PHP** pour gérer la logique côté serveur et l'interactivité dynamique.

3. Avec qui avez-vous travaillé ?

Pour réaliser l'interface statique de la page utilisateur, j'ai travaillé seul. Cependant, je me suis inspiré de la maquette conçue au préalable par le groupe, qui a guidé la structure et le design de l'interface.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *Diginamic*

Chantier, atelier, service ▸ Projet en formation

Période d'exercice ▸ Du : *21/10/2024* au : *25/11/2024*

5. Informations complémentaires *(facultatif)*

Après avoir développé l'interface utilisateur, j'ai décidé d'ajouter des éléments d'accessibilité. J'ai donc réalisé un test d'accessibilité avec l'onglet Accessibility dans les DevTools intégrés à Google Chrome (le 15 mars 2024) pour m'assurer que mon interface soit utilisable par tous les utilisateurs, y compris ceux ayant des besoins spécifiques.

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

Pour ce faire, comme vous pouvez le voir dans les captures d'écran, j'ai ajouté l'attribut role="main" .

```
<section class="user-section">
    <div class="user-content">
        <h1>Gestion des utilisateurs</h1>
        <p>Gérez vos utilisateurs facilement</p>
        <button id="addUserBtn" class="btn btn-primary"  aria-haspopup="dialog">Créer un utilisateur </button>
    </div>
    <div class="primaryimg">
        <img src="build/images/user.png" alt="Interface de gestion des utilisateurs" width="500" height="400">
    </div>
</section>
```

role="main"permet aux utilisateurs de sauter directement au contenu principal, évitant de naviguer à travers les menus et en-têtes. J'ai ajouté l' attribut aria-haspopup="dialog" sur le bouton pour créer un utilisateur ce qui permet a le lecteur d'écran d'annoncer à l'utilisateur quelque chose comme : "Créer un utilisateur, bouton, ouvre une boîte de dialogue".Remplacer le texte alternatif "Utilisateurs" par "Interface de gestion des utilisateurs".

J'ai aussi personnalisé le texte alternatif pour chaque avatar en incluant le prénom de l'utilisateur spécifique.

Par exemple, au lieu d'annoncer simplement "Avatar" pour tous les avatars (ce qui ne permet pas de les distinguer), un lecteur d'écran annoncera "Avatar de Jean" ou "Avatar de Marie" selon l'utilisateur affiché

```
<div class="user-avatar">
    <img src="/placeholder.svg?height=60&width=60"
    alt="Avatar de <?php echo htmlspecialchars($user->getFirstname()); ?>">
</div>
```

Et pour finir j'ai rajouté aria-label sur les boutons d'action , fournit un texte descriptif pour les boutons qui n'ont que des icônes. Au lieu d'entendre simplement "bouton", l'utilisateur entend "Modifier Jean Dupont" ou "Supprimer Marie Martin"

```
<div class="actions">
    <button class="btn-edit" data-id="<?php echo $user->getId(); ?>"
     aria-label="Modifier <?php echo htmlspecialchars($user->getFirstname() . ' ' . $user->getSurname()); ?>">
        <i class="fas fa-edit" aria-hidden="true"></i>
    </button>
    <button class="btn-delete" data-id="<?php echo $user->getId(); ?>"
    aria-label="Supprimer <?php echo htmlspecialchars($user->getFirstname() . ' ' . $user->getSurname()); ?>">
        <i class="fas fa-trash" aria-hidden="true"></i>
    </button>
</div>
```

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Compétence n°4 *Développer la partie dynamique des interfaces utilisateur web ou web mobile.*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Il était nécessaire d'implémenter des éléments dynamiques dans la structure statique afin d'assurer une meilleure interaction et d'adapter l'affichage en fonction des besoins des utilisateurs.

J'ai bien conscience que le PHP fonctionne côté serveur (back-end), mais il génère du code HTML qui sera ensuite interprété par le navigateur côté client (front-end), créant ainsi une interaction complète entre les deux environnements.
C'est dans cette optique que j'ai développé les parties dynamiques de l'interface utilisateur, comme décrit en détail ci-dessous.

J'ai d'abord travaillé sur l'affichage initial avec PHP, en utilisant une boucle foreach pour générer la liste des utilisateurs. Le code suivant crée une carte pour chaque utilisateur avec un identifiant unique (data-id) essentiel pour les interactions futures :

```
<?php if (!empty($users)): ?>
    <?php foreach ($users as $user): ?>
        <div class="user-card" data-id="<?php echo $user->getId(); ?>">
```

Sur chaque utilisateur il y aura son nom, son rôle et son email.

```
<div class="user-details">
    <h3><?php echo htmlspecialchars($user->getSurname() . ' ' . $user->getFirstname()); ?></h3>
    <p class="role">Rôle: <?php echo htmlspecialchars($user->getRole()); ?></p>
    <p class="email">Email: <?php echo htmlspecialchars($user->getEmail()); ?></p>
</div>
```

J'ai utilisé `htmlspecialchars()` pour afficher le rôle et l'email de l'utilisateur car cette fonction permet d'encoder les caractères HTML spéciaux comme les chevrons (`<` et `>`). Cela empêche l'exécution de code malveillant injecté, protégeant ainsi l'application contre les failles de type Cross-Site Scripting (XSS). Cette mesure de sécurité est essentielle pour la protection des données des utilisateurs.

Pour les btn edit et delete et j'ai utilisé JavaScript pour implémenter les fonctionnalités dynamiques

```
            <div class="actions">
                <button class="btn-edit" data-id="<?php echo $user->getId(); ?>">
                    <i class="fas fa-edit"></i>
                </button>
                <button class="btn-delete" data-id="<?php echo $user->getId(); ?>">
                    <i class="fas fa-trash"></i>
                </button>
            </div>
```

Lorsque l'utilisateur clique sur ces boutons, le code JavaScript récupère l'identifiant unique de l'utilisateur stocké dans l'attribut data-id.
Cela me permet d'identifier le bon utilisateur pour effectuer les opérations d'édition ou de suppression, en communiquant de manière sécurisée avec le serveur.

```
    if (editBtn) {
        const userId = editBtn.dataset.id;
        if (userId) openModal('update', userId);
    } else if (deleteBtn) {
        const userId = deleteBtn.dataset.id;
        if (userId && confirm('Êtes-vous sûr de vouloir supprimer cet utilisateur ?')) {
            deleteUser(userId);
        }
    }
```

- ➔ Si l'utilisateur a cliqué sur un bouton **modifier** (.btn-edit), on récupère l'id de l'utilisateur (editBtn.dataset.id).
- ➔ Si l'id existe, **j'ouvre la modale** (openModal('update', userId)) pour modifier l'utilisateur.

Dans le cas contraire:

- ➔ Si l'utilisateur a cliqué sur un bouton **supprimer** (.btn-delete), on récupère l'id de l'utilisateur (deleteBtn.dataset.id).
- ➔ Une **confirmation** est demandée avec (confirm('Êtes-vous sûr de vouloir supprimer cet utilisateur ?')).
- ➔ Si l'utilisateur **confirme**, on appelle deleteUser(userId) pour supprimer l'utilisateur.

Avant de pouvoir effectuer l'édition ou la suppression d'un utilisateur, il est nécessaire de pouvoir le créer. Pour cela, j'ai développé un formulaire dynamique sous forme de modale. Cette approche permet à l'utilisateur de saisir les informations nécessaires à la création d'un nouvel utilisateur, tout en évitant le rechargement de la page. Cela offre une expérience fluide et réactive.

En utilisant une modale pour la saisie des informations, j'ai évité le rechargement complet de la page, ce qui s'inscrit dans une démarche d'**éco-conception**.

En effet, l'utilisation de fenêtres modales avec des requêtes asynchrones (Fetch) permet d'optimiser considérablement les performances de l'application. Cette approche réduit le volume de données transférées, car seul le contenu nécessaire est chargé au lieu de recharger la page entière.

Concernant l' ajout d' un utilisateur lorsque le chef de projet soumet le formulaire en cliquant sur le bouton "Ajouter", un événement de type submit est déclenché. J'ai ajouté un écouteur d'événements sur ce formulaire pour capturer cet événement et empêcher le comportement par défaut qui entraînerait un rechargement de la page :

```
userForm.addEventListener('submit', async function(e: SubmitEvent): Promise<void> {
    e.preventDefault();
```

Ensuite, j'utilise l'objet FormData pour récupérer toutes les informations saisies par l'utilisateur dans le formulaire. Ces données sont ensuite envoyées à l'API via la méthode createUser du service userService, en utilisant l'instruction await pour attendre la réponse du serveur.

Afin de gérer à la fois la création et la modification des utilisateurs, j'ai mis en place une logique qui différencie ces deux actions. Si le formulaire est en mode "create", j'envoie les données à userService.createUser(formData) pour les enregistrer dans la base de données. En revanche, si le formulaire est en mode "update", j'appelle userService.updateUser(userId, formData) afin de modifier les informations d'un utilisateur existant.

```
    const formData = new FormData(userForm);
    const mode = userForm.dataset.mode;
    const userId = (document.getElementById('userId') as HTMLInputElement).value;

    try {
        const data = mode === 'create'
            ? await userService.createUser(formData)
            : await userService.updateUser(userId, formData);
```

Si l'opération réussit, plusieurs actions sont effectuées :

- **En cas de création**, j'ajoute le nouvel utilisateur à la liste visible grâce à la fonction addUserToList(data.user)
- **En cas de mise à jour**, j'actualise la carte utilisateur via la fonction updateUserCard(data.user)

Dans les deux cas, la modale est fermée avec la fonction closeModal(), et un message de confirmation est affiché pour informer l'utilisateur que l'opération a été réalisée avec succès.

```
        if (data.success) {
            if (mode === "create") {
                addUserToList(data.user);
                closeModal();
                alert("Votre utilisateur a bien été créé");
            } else {
                updateUserCard(data.user);
                closeModal();
                alert("Utilisateur mis à jour avec succès");
            }
        } else {
            throw new Error(data.message || 'Une erreur est survenue');
        }
```

Cette étape permet alors, d'ajouter de nouveaux utilisateurs à notre système et ouvre la possibilité de gérer ces utilisateurs par la suite, que ce soit pour les modifier ou les supprimer.

2. Précisez les moyens utilisés :

PHP , Javascript

3. Avec qui avez-vous travaillé ?

J'ai relevé ce défi seule, en m'aidant de tutoriels trouvés sur Internet. Bien sûr, j'ai aussi utiliser les connaissances acquises durant cette formation pour avancer et finaliser ce projet

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *Diginamic*

Chantier, atelier, service ▸ Projet en formation

Période d'exercice ▸ Du : 20/12/2024 au : *15/01/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Compétence n°5 *Mettre en place une base de données relationnelle*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Une base de données permet de stocker, organiser et gérer des données de manière structurée. Pour notre application de gestion de projets, on a utilisé phpMyAdmin, l'interface de gestion de base de données fournie avec MAMP.
Cet outil favorise la manipulation des données à travers des requêtes SQL (par **exemple** pour la création ou la consultation de l'entité User):

Pour créer la table User :

```sql
CREATE TABLE User (
    id INT PRIMARY KEY AUTO_INCREMENT,
    role VARCHAR(255),
    surname VARCHAR(255),
    firstname VARCHAR(255),
    email VARCHAR(255),
    password VARCHAR(255)
);
```

Ou encore pour consulter les utilisateurs :

```sql
-- Récupérer tous les utilisateurs
SELECT * FROM User;

-- Rechercher un utilisateur spécifique
SELECT * FROM User
WHERE email = 'jean.dupont@email.com';

-- Récupérer tous les administrateurs
SELECT * FROM User
WHERE role = 'admin';
```

Pour concevoir notre base de données, on a commencé par établir le **Modèle Conceptuel de Données** (MCD) en utilisant l'application Looping. C'est important de le faire parce qu'elle nous montre la structure de notre base de données avant même de commencer son implémentation.

Dans notre MCD (annexe 1), on retrouve les entités suivantes avec leurs propriétés :

USER : Id (INT), role (VARCHAR(255)), surname (VARCHAR(255)), firstname (VARCHAR(255)), email (VARCHAR(255)), password (VARCHAR(255))

PROJECT : Id (INT), creation_date (DATE), starting_date (DATE), ending_date (DATE), project_name (VARCHAR(255)), description (TEXT)

TASK : Id (INT), description (VARCHAR(255)), starting_date (DATE), ending_date (DATE)

CLIENT : Id (INT), siret (INT), name (VARCHAR(255))

PRIORITY : Id (INT), label (VARCHAR(255))

TYPE OF TASK : Id (INT), label (VARCHAR(255))

STATUT : Id (INT), label (VARCHAR(255))

Nous avons ensuite défini les relations entre ces entités en précisant leurs cardinalités (annexe 1). Dans notre projet, nous avons identifié les trois types de cardinalités suivantes :

One-to-One (1,1) : Par exemple, dans notre application, une tâche ne peut avoir qu'une seule priorité à la fois, et cette priorité est spécifique à cette tâche.

One-to-Many (1,N) : Dans notre cas, un utilisateur peut concevoir plusieurs projets, mais chaque projet ne peut être conçu que par un seul utilisateur. C'est la relation qu'on retrouve entre l'entité USER et PROJECT via l'association "conceived".

Many-to-Many (N,N) : Par exemple, dans notre application, un utilisateur peut participer à plusieurs projets, et un projet peut avoir plusieurs utilisateurs. Cette relation est gérée par l'association "is part of" entre USER et PROJECT, qui nécessitera une table intermédiaire lors de l'implémentation.

Après avoir défini notre Modèle Conceptuel de Données (MCD), nous avons procédé à la création du Modèle Logique de Données (MLD) (annexe 2). Elle consiste à convertir le MCD en un modèle plus détaillé, en tenant compte des contraintes techniques du système de gestion de base de données (SGBD) que nous utilisons.

L'application Looping a une fonction automatique pour transformer un MCD en MLD, ce qui nous a facilité le processus. Pour se faire,le logiciel change les associations en tables relationnelles. Par exemple, la relation "is part of" entre les entités USER et PROJECT a donné lieu à une table intermédiaire afin de gérer la participation des utilisateurs à plusieurs projets.

Elle a également défini automatiquement les clés primaires et les clés étrangères pour assurer l'intégrité référentielle des données. Chaque table dispose d'une clé primaire unique (Id) et, lorsqu'une relation existe entre deux entités, le logiciel a ajouté les clés étrangères nécessaires pour établir les liaisons entre les tables.

2. Précisez les moyens utilisés :

On a utilisé l'application Looping pour réaliser le MCD et le MLD.
Ce logiciel nous a permis de concevoir visuellement les entités et leurs relations, puis de transformer automatiquement le MCD en MLD.

3. Avec qui avez-vous travaillé ?

Pour le MCD et MLD, j'ai travaillé en collaboration avec mon groupe

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *Diginamic*

Chantier, atelier, service ▸ Projet en formation

Période d'exercice ▸ Du : *15/01/2025* au : *1/01/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Compétence n°6 - *Développer des composants d'accès aux données SQL*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour envoyer et récupérer des informations depuis une base de données nous avons besoin des **composants d'accès aux données.** Ils servent d'intermédiaires entre l'application et la base de données

Pour mettre en place ces composants dans notre projet, j'ai étudié le framework. Téléchargé depuis GitHub, j'ai analysé la hiérarchie des fichiers (Annexe 3)pour comprendre son architecture et identifier les points d'intervention.

J'ai remarqué qu'à la racine du framework se trouve un fichier .env.exemple qui me donne une structure de base avec des données contenant les informations de configuration. J'ai copié ce fichier dans le fichier .env qui n' est pas versionné. Il était initialement configuré comme suit :

.env

```
DSN=mysql:host=localhost;dbname=filrouge
USERNAME=admin
PASSWORD=admin

CONTROLLER_NAMESPACE=Sthom\App\Controller\
MODEL_NAMESPACE=Sthom\App\Model\
DEBUG=true
```

Comme vous pouvez le remarquer nous avons une chaîne de connexion à la base de données, suivi des identifiants pour se connecter et les espaces de noms (namespaces) pour les contrôleurs et les modèles. Le mode débogage de l'application est activé. Il semblerait que le framework suis une architecture MVC (Modèle-Vue-Contrôleur)

J'ai par la suite modifier ce fichier pour l'adapter à mon environnement de développement en remplaçant les identifiants de connexion par les miens.
J'ai changé le nom de la base de données de "filrouge" à "orgaken", ajusté le port à 8889, spécifié l'encodage UTF8mb4, et mis à jour les identifiants d'accès avec mes propres informations de connexion.

```
DSN="mysql:host=localhost;port=8889;dbname=orgaken;charset=utf8mb4"
USERNAME=root
PASSWORD=Abc123!Def45


CONTROLLER_NAMESPACE=Sthom\App\Controller\
MODEL_NAMESPACE=Sthom\App\Model\
DEBUG=true
```

Pendant l' exploration, j'ai analysé la classe Database {} située dans le dossier database du framework, elle gère la connexion en utilisant les paramètres de configuration du fichier .env.
De plus, j 'ai vu qu'il avait les trois éléments du pattern Singleton dans la classe Database:

La variable statique pour stocker l'instance unique est ici:

```php
class Database {

    /**
     * @var PDO|null
     * Cette variable statique contient l'instance de la connexion à la base de données
     */
    private static ?PDO $connexion = null;
```

La variable $connexion n'est accessible que depuis l'intérieur de la classe (car elle private), elle peut être de type soit un objet PDO (PHP Data Objects), soit null (?)

Le constructeur privé :

Cela signifie que le constructeur ne peut être appelé que depuis l'intérieur de la classe Database.
À l'intérieur de ce constructeur, la classe utilise PDO (PHP Data Objects), qui agit comme une interface sécurisée entre l'application PHP et ma base de données.
PDO prévient les injections SQL grâce à son système de requêtes préparées.

```php
    /**
     * @method __construct
     * Le constructeur est privé pour empêcher l'instanciation de la classe
     * Il initialise la connexion à la base de données en utilisant les informations de connexion définies dans les configurations
     * de notre framework
     * @throws \Exception
     */
    private function __construct()
    {
        try {
            self::$connexion = new PDO($_ENV['DSN'], $_ENV['USERNAME'], $_ENV['PASSWORD']);
        } catch(\PDOException $e) {
            throw new \Exception("Erreur de connexion à la base de données, veuillez vérifier vos paramètres de connexion");
        }
    }
```

Pour établir la connexion, le constructeur récupère les paramètres nécessaires depuis les variables d'environnement ($_ENV['DSN'], $_ENV['USERNAME'],$_ENV['PASSWORD']) que j'ai configuré dans le fichier .env. Cette approche sépare le code des informations sensibles de connexion, ce qui est une bonne pratique en termes de sécurité et de maintenance.

La méthode publique statique pour accéder à cette instance unique:

Cette méthode getConnexion() est le point central du pattern Singleton. Elle vérifie d'abord si une connexion à la base de données existe déjà.

Si la connexion n'existe pas, elle en crée une nouvelle en appelant le constructeur privé. Dans tous les cas, elle renvoie une instance de PDO ,soit fraîchement créée soit celle qui préexistait , ce qui garantit que toute l'application utilise la même connexion à la base de données.

```
    /**
     * @method getConnexion
     * Cette méthode permet de récupérer l'instance de la connexion à la base de données
     * Si la connexion n'existe pas, elle est créée.
     * Son instance est ensuite utilisée pour exécuter des requêtes SQL
     * @return PDO
     */
    public final static function getConnexion(): PDO {
        if (self::$connexion === null) {
            new static();
        }
        return self::$connexion;
    }
}
```

Après avoir analysé la connexion à la base de données avec Database.php, j'ai exploré la classe AbstractRepository.php qui fournit toutes les méthodes CRUD nécessaires pour manipuler les données.

Voici deux exemples de fonctions clés que j'ai étudiées:
findAll() - Cette fonction permet de récupérer toutes les entrées d'une table:

```
public final function findAll(): array // retrouver les en
{
    $query = $this->queryBuilder
        ->select()
        ->buildSelect();

    return $this->executeMultiple($query);
}
```

delete() permet simplement de supprimer une entrée en utilisant son ID:

```
public final function delete(int $id): void
{
    $query = $this->queryBuilder
        ->where('id', '=', $id)
        ->delete();

    $this->executeStatement($query);
}
```

Ces classes abstraites servent de base pour tous les repositories de l'application (Chacun de ces repositories héritera de l'AbstractRepository et bénéficie de toutes les méthodes CRUD (findAll, save, delete, etc.) sans avoir à les réécrire.).

Voici un exemple de comment je l'ai utilisé :

```php
class UserRepository extends AbstractRepository
{


    public function __construct()
    {
        parent::__construct(User::class);
    }
```

Ce UserRepository peut directement utiliser toutes les méthodes comme findAll(), find(id), save(user), delete(id), etc., sans avoir à les réécrire.

2. Précisez les moyens utilisés :

Le framework PHP téléchargé depuis GitHub
Une base de données MySQL (accessible via le port 8889)
MAMP

3. Avec qui avez-vous travaillé ?

J'ai travaillé avec le framework qui était déjà développé par Sébastien Thomas.
Les fichiers Database.php et AbstractRepository.php étaient déjà présents quand j'ai téléchargé le framework depuis GitHub.

Mon rôle a été de configurer le fichier .env avec mes propres informations de connexion et d'utiliser ces composants déjà existants pour mon projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *Diginamic.*

Chantier, atelier, service ▸ Projet en formation

Période d'exercice ▸ Du : 12/12/2025 au : 20/01/2025

5. Informations complémentaires *(facultatif)*

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Compétence n°7 - *Développer des composants métier côté serveur*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour la partie développer des composants métier côté serveur, j'ai travaillé sur les éléments qui gèrent les opérations liées aux utilisateurs (le controller (UserController.php) et le modèle (User.php)).

le modèle User.php qui constitue l'élément central de la gestion des utilisateurs. Cette classe respecte les principes de l'encapsulation en programmation orientée objet.

```php
class User
{
    // Définit le nom de la table en base de données
    public const TABLE = 'user';

    // Propriétés privées avec valeurs par défaut
    private ?int $id;
    private ?string $surname = '';
    private ?string $firstname = '';
    private ?string $email = '';
    private ?string $password = '';
    private ?string $role = '';
```

Pour respecter le principe d'encapsulation, toutes les propriétés sont privées et accessibles uniquement via des getters et setters

```php
    // Getter pour l'ID
    public function getId(): int
    {
        return $this->id;
    }

    // Setter pour l'ID
    public function setId(int $id): void
    {
        $this->id = $id;
    }
```

Dans le UserController, j'ai développé trois méthodes principales:

- **Méthode index()**

La méthode index() permet d'afficher la liste de tous les utilisateurs enregistrés dans le système :

```php
public final function index() {
    $users = $this->repository->findAll();
    $this->render('user/index', ['users' => $users]);
}
```

Cette méthode récupère tous les utilisateurs en utilisant la méthode findAll() de l'AbstractRepository que nous avons vue précédemment, puis les transmet à la vue 'user/index' pour affichage.

- **Méthode create()**

```php
public function create()
{
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        try {

            $requiredFields = ['surname', 'firstname', 'email', 'role', 'password'];
            foreach ($requiredFields as $field) {
                if (empty($_POST[$field])) {
                    throw new \Exception("Le champ $field est requis.");
                }
            }
```

La méthode create() permet d'ajouter un nouvel utilisateur.Dans cette méthode, j'ai d'abord vérifié que la méthode HTTP est bien de type POST. C'est essentiel pour la sécurité puisque la création d'utilisateurs ne devrait pas être accessible via d'autres méthodes HTTP comme GET.

Ensuite, je définis un tableau $requiredFields qui contient tous les champs obligatoires pour créer un utilisateur : nom de famille, prénom, email, rôle et mot de passe.

La boucle foreach parcourt chaque champ requis et vérifie s'il est vide dans le tableau $_POST (qui contient les données envoyées par le formulaire). Si un champ est manquant, je lance une exception avec un message d'erreur précisant quel champ est manquant. Après cette validation, j'ai créé l'objet utilisateur et initialisé ses propriétés avec les valeurs reçues :

```php
            $user = new User();

            $user->setSurname($_POST['surname']);
            $user->setFirstname($_POST['firstname']);
            $user->setEmail($_POST['email']);
            $user->setPassword(password_hash($_POST['password'], PASSWORD_DEFAULT));
            $user->setRole($_POST['role'] ?? 'développeur');
```

J'utilise les méthodes setter pour définir chaque propriété de l'utilisateur avec les données du formulaire. Pour le mot de passe, j'applique la fonction password_hash() qui génère un hachage sécurisé, empêchant ainsi le stockage en clair des mots de passe en base de données.

L'opérateur de fusion null (??) permet d'attribuer un rôle par défaut 'développeur' si aucun rôle n'est spécifié.

Après avoir initialisé toutes les propriétés de l'objet User, j'ai implémenté un mécanisme de gestion des erreurs pour l'insertion en base de données:

```php
try {
    $this->repository->insert($user);
} catch (\Exception $e) {
    throw new \Exception("Erreur lors de l'insertion de l'utilisateur dans la base de données.");
}
```

- **Méthode update()**

```php
public function update($id)
{
    if ($_SERVER['REQUEST_METHOD'] === 'POST' || $_SERVER['REQUEST_METHOD'] === 'GET') {
        try {
            $user = $this->repository->find($id);

            if (!$user) {
                $this->json([
                    'success' => false,
                    'message' => 'Utilisateur non trouvé'
                ], 404);
                return;
            }
```

Pour modifier un utilisateur, la méthode vérifie que la requête est bien de type POST ou GET.J'ai choisi d'accepter ces deux méthodes pour plus de flexibilité, permettant ainsi la mise à jour des utilisateurs via ces deux requêtes. Puis je récupère l'utilisateur par son ID.

Si l'utilisateur n'existe pas, je renvoie immédiatement une réponse JSON avec un statut 404 (Not Found) et j'interromps l'exécution de la méthode. J'ai ensuite mis à jour uniquement les champs qui sont présents dans la requête .

```
    $data = $_POST;

    if (isset($data['surname'])) {
        $user->setSurname($data['surname']);
    }
    if (isset($data['firstname'])) {
        $user->setFirstname($data['firstname']);
    }
    if (isset($data['email'])) {
        $user->setEmail($data['email']);
    }
    if (isset($data['role'])) {
        $user->setRole($data['role']);
    }
```

Pour ce faire, je récupère d'abord toutes les données envoyées dans la variable $data . Pour chaque propriété possible de l'utilisateur, j'utilise la fonction isset() qui vérifie si le champ existe dans les données reçues

Si le champ existe, j'utilise le setter approprié pour mettre à jour cette propriété spécifique Et si le champ n'existe pas, je ne fais rien pour cette propriété, conservant ainsi sa valeur actuelle.

Après cela , J'utilise la méthode save() du repository qui détecte automatiquement qu'il s'agit d'une mise à jour (et non d'une création) car l'objet utilisateur possède déjà un ID.

```
$this->repository->save($user);

$this->json([
    'success' => true,
    'message' => 'Utilisateur mis à jour avec succès',
    'user' => $this->formatUserData($user)
]);
```

Après la sauvegarde, je renvoie une réponse JSON avec un statut de succès, un message explicatif et les données mises à jour de l'utilisateur(formatées via la méthode formatUserData).

Pour finir j'ai rajouté catch pour gérer les erreurs qui pourraient survenir pendant la mise à jour (problèmes de base de données, contraintes non respectées, etc.). La réponse inclut le message d'erreur pour faciliter le débogage.

```
} catch (\Exception $e) {
    $this->json([
        'success' => false,
        'message' => 'Erreur lors de la mise à jour de l\'utilisateur : ' . $e->getMessage()
    ], 500);
}
```

Aussi, si la requête n'est ni POST ni GET, je renvoie une erreur 405 (Method Not Allowed).

```
    } else {
        $this->json([
            'success' => false,
            'message' => 'Méthode non autorisée'
        ], 405);
    }
}
```

2. Précisez les moyens utilisés :

Le framework PHP et la base de donnée

3. Avec qui avez-vous travaillé ?

J'ai travaillé avec Sébastien Thomas.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *Diginamic*

Chantier, atelier, service ▸ Projet en formation

Période d'exercice ▸ Du : *25/01/2025* au : *20/02/2025*

5. Informations complémentaires *(facultatif)*

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Master II en Management de la stratégie digitale marketing et commerciale	**IDECA Business School, Foundation 1999**	28 novembre 2022
Certificat de formation en anglais général - Niveau B1 (Pré-Intermédiaire)	**English Path Language School, Malta**	11 août 2023
Certificat d'achèvement - Programme intensif de conversation anglaise - Niveau B1/B2 (CECR)	**Access English, Toronto, Canada**	12 juillet 2024

Déclaration sur l'honneur

Je soussignée Dhellye Shade EWEMBE ,

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l'auteure des réalisations jointes.

Fait à Montpellier le 17/03/2025

pour faire valoir ce que de droit.

Signature :

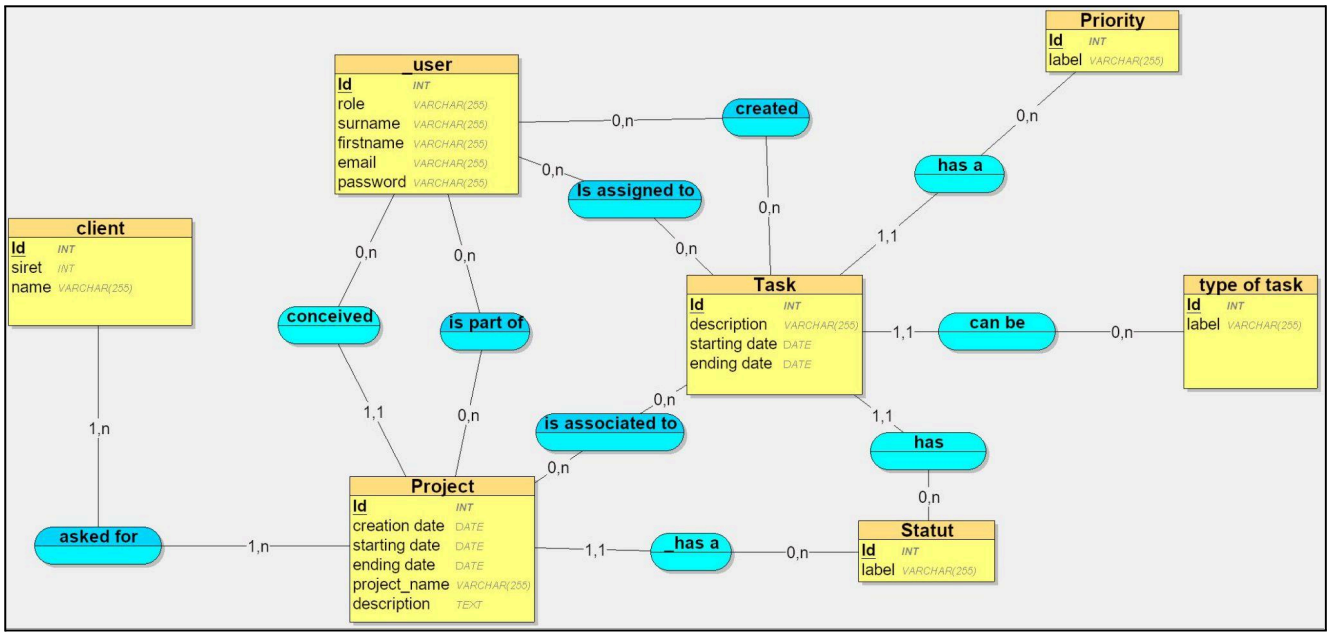
Documents illustrant la pratique professionnelle

(facultatif)

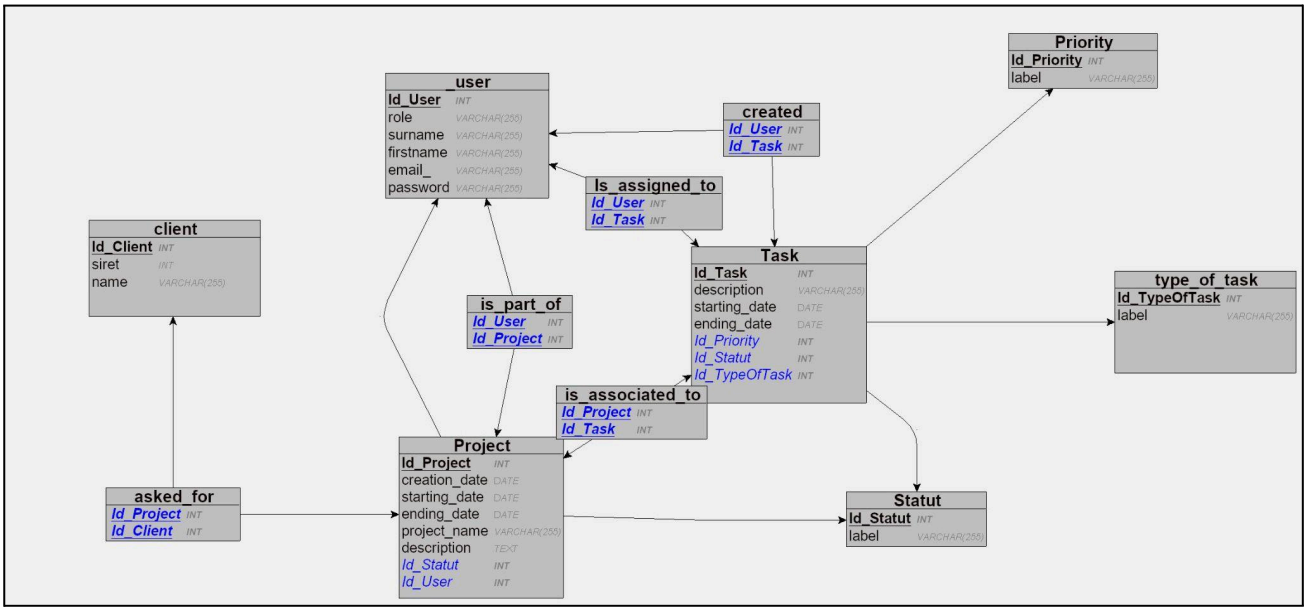
Intitulé
Annexe 1: Modèle Conceptuel de donnée (MCD)
Annexe 2: Modèle Logique de donnée (MLD)
Annexe 3:Structure de la base de données 'Orgaken' - Tables utilisées pour le projet
Annexe 4 : Résultat d'une requête SQL - Affichage des utilisateurs dans phpMyAdmin"
Annexe 5: Structure des fichiers du framework PHP
Annexe 6 : Logs de requêtes API pour l'utilisateur 131 incluant des opérations de: consultation, mise à jour et suppression.
Annexe 7: Interface de gestion d'utilisateurs d'une application Orgaken"

ANNEXES

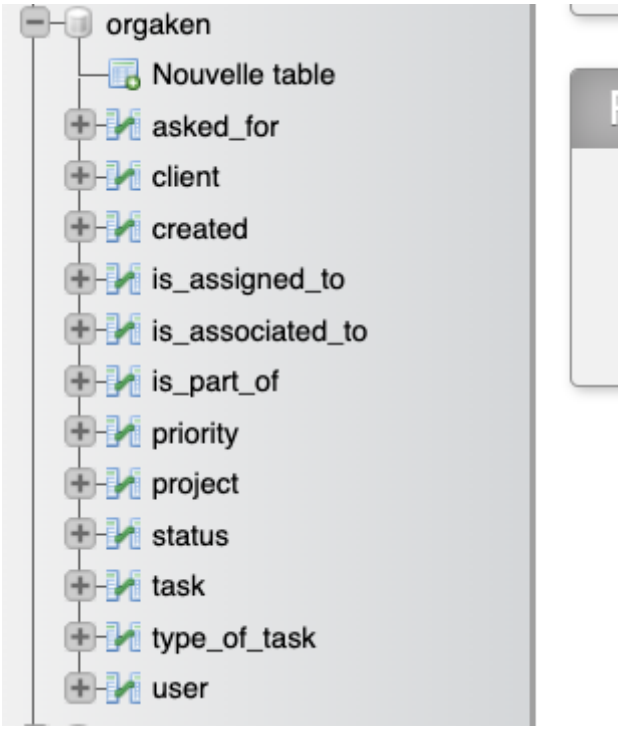
Annexe 1: Modèle Conceptuel de donnée (MCD)



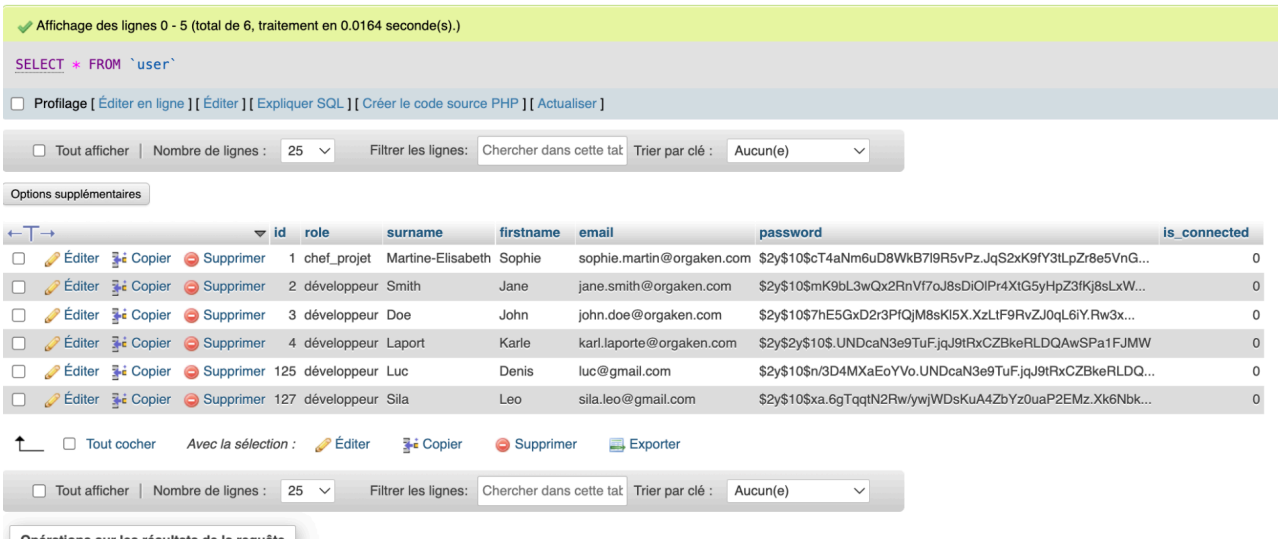
Annexe 2: Modèle Logique de donnée (MLD)



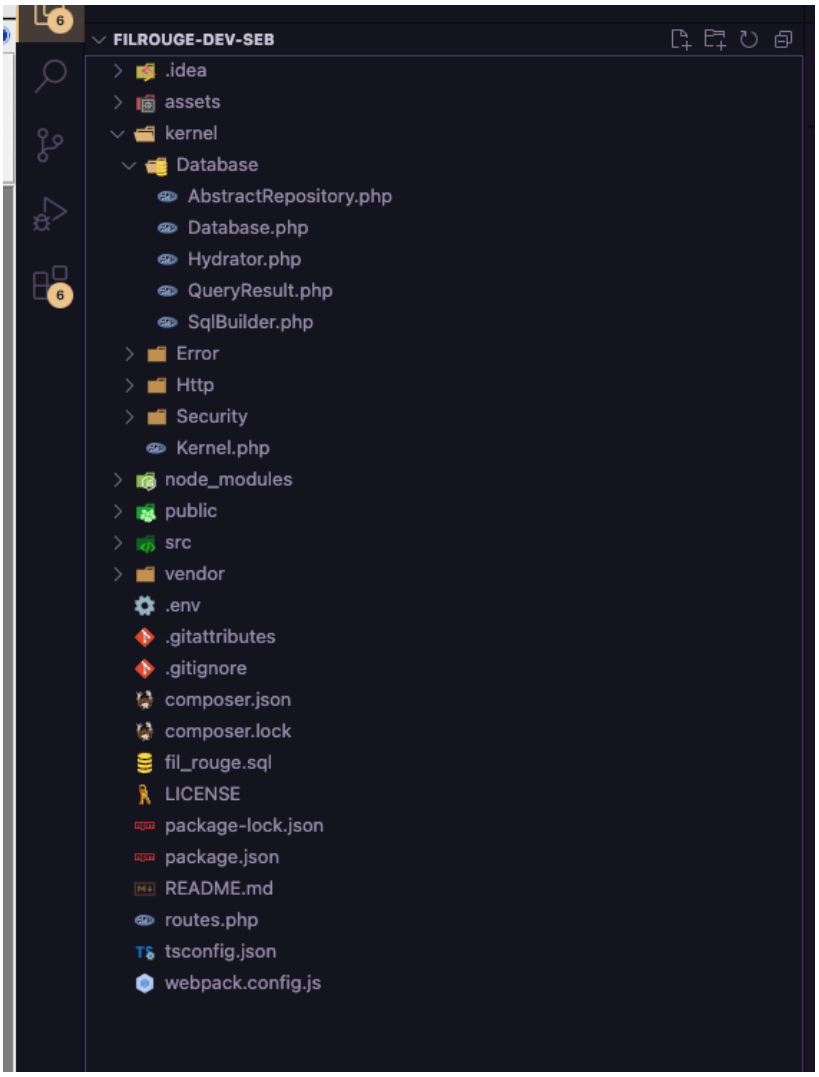
Annexe 3:Structure de la base de données 'orgaken' - Tables utilisées pour le projet



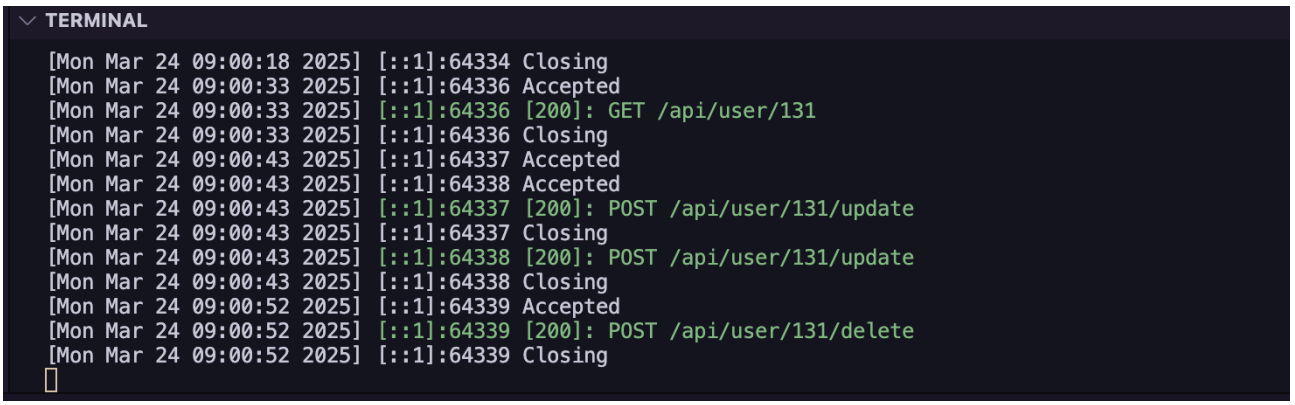
Annexe 4 : Résultat d'une requête SQL - Affichage des utilisateurs dans phpMyAdmin"



Annexe 5: Structure des fichiers du framework PHP



Annexe 6 : Logs de requêtes API pour l'utilisateur 131 incluant des opérations de consultation, mise à jour et suppression.



Annexe 7: Interface de gestion d'utilisateurs de l’application Orgaken"