BTS SIO - SLAM

PROJET EN COURS DE FORMATION

CONCEPTION D'UNE APPLICATION WEB D'AIDE A LA GESTION DE PROJET POUR UN CABINET DE CONSEIL

Participants:

- Alexandre Boyer

SESSION 2022-2023

Table des matières

1. Cahier des charges

Contexte

OASYS Consulting est une société de services de la région Toulousaine intervenant dans les domaines de l'infogérance informatique et de la formation. Elle compte 3 salariés mais, en fonction des projets, elle fait intervenir d'autres sociétés partenaires ou travailleurs indépendants.

Ces intervenants extérieurs refacturent leurs prestations à OASYS Consulting.
Aujourd'hui, OASYS utilise encore une application de gestion et de facturation basée sur Access. Cette application est vieillissante, non maintenue et inaccessible depuis internet.

Il faut donc développer une nouvelle application web pour aider Oasys a la gestion de leur projet.

Objectif

Le but de cette application est de stocker et gérer les données de l'activité de OASYS, y compris l'enregistrement des clients, des projets, des interventions des salariés et/ou intervenants externes sur chaque phase.

L'application doit également permettre aux chefs de projets de suivre l'avancement des projets qui leur sont attribué.

Fonctionnalités requises

1. Enregistrement des clients de OASYS : l'application doit permettre d'enregistrer les informations relatives aux clients de OASYS, y compris leur nom, leur adresse, leur adresse e-mail, leur numéro de téléphone, etc.

2. Enregistrement des projets et de leurs étapes : l'application doit permettre d'enregistrer les projets de OASYS, y compris leur code, leur libellé, leur date de début et de fin, le client associé, le domaine, ainsi que leurs différentes étapes.

3. Enregistrement des interventions des salariés et/ou intervenants externes : l'application doit permettre d'enregistrer les interventions effectuées par les techniciens sur chaque phase du projet, y compris la durée de l'intervention en jours et en heures, les dates d'intervention, l'état d'avancement du projet, etc.

4. Enregistrement des éléments de facturation au client final : l'application doit permettre d'enregistrer les éléments de facturation liés aux prestations de service effectuées, y compris la durée de la prestation, le tarif journalier, le coût total de la prestation, etc.

5. Suivi de l'activité : l'application doit permettre aux chefs de projets de se connecter à l'application par Internet et d'éditer des fiches de suivi de projets, permettant de connaître qui est intervenu, l'état d'avancement, les commentaires des intervenants, etc.

6. Suivi de la part des projets réalisés en interne et en externe : l'application doit permettre aux chefs de projets et aux dirigeants de suivre la part des projets réalisés en interne et la part réalisée par les intervenants externes, ainsi que le coût de la sous-traitance.

2. Etudes des outils possibles

IDE

- Visual Studio Code
- PHP Storm
- Sublime text

Pour notre cas, l'IDE (integrated development environment) choisi sera PHP Storm qui est un environnement de développement spécialisé pour le développement web et qui est distribué par JetBrains.

Il a été sélectionné pour intégration Git avec Github qui est très bien faites et très facile à prendre en main et utiliser.

Système de gestion des versions

Le système de gestion des versions choisi est Git, hébergé sur GitHub

Langages utilisés

Pour notre cas d'application web il a été utilisé du PHP pour le backend.
Et pour le frontend du HTML/CSS/JavaScript.

Il aurait pû également être utilisé Java ou Python pour la backend. Mais PHP étant le langage serveur le plus utilisé et le plus documenté pour du web, celui est fortement recommandé.

Système de gestion de base de données

Le SGBD utilisé a été MySQL avec PHPMyAdmin pour l'administrer.
MySQL étant le SGBD le plus utilisé et le plus documenté encore une fois, le choix s'est porté sur lui.
De plus le projet dans le volume de données qu'il s'apprêtait a stocker ne nous demandais pas de choisir des technologie comme du NoSQL et dans ce cas précis, MySQL était tout indiqué.

Framework

Deux framework majeur ont été utilisés.
Tout d'abord Bootstrap pour réaliser le front end. Bootstrap permet de réaliser des designs a la fois simpliste mais avec un aspect travaillé et moderne. Etant très familier avec celui-ci, je l'ai choisi pour mon front.

Pour le Dashboard, C'est Chart.js qui a permis de réaliser le graphique. Celui-ci est facile à mettre en

place et a utilisé pour injecter des données issues de la BDD et rendre des graphiques, ce qui a une vraie valeur ajoutée.

3. Mise en œuvre

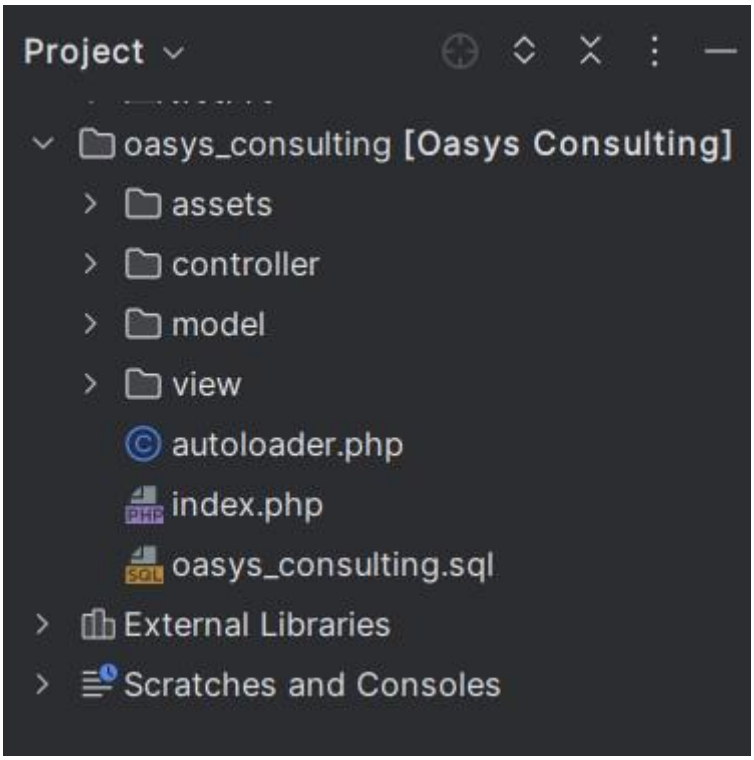
3.1. Architecture

Présentation de l'architecture :

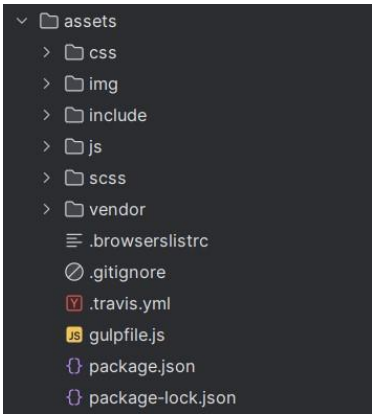
L'architecture utilisé est l'architecture MVC (Modèle – Vue – Controlleur). L'architecture MVC sépare une application en trois composants : le modèle (gestion des données), la vue (affichage) et le contrôleur (traitement des interactions). Cela facilite la maintenance, la réutilisation du code et la flexibilité de l'application.

Mise en place sur le projet :

L'architecture du projet Oasys consulting basé sur le MVC est montré ci-dessous :



Le dossier Assets détenant out les fichiers Images, CSS, JS et tout ce qui est chargé par le projet en général.



3.2. Présentation du fonctionnement d'une fonctionnalité

La fonctionnalité de création d'un projet étant très complète, celle-ci va être présenté pour illustrer le fonctionnement de base du projet, car en effet les fonctionnalités sont basées sur le même paterne de fonctionnement.

Le controller :

Le Controller contient tous les appels au méthode utilisé par les fonctionnalités présente sur la page. Il récupère toutes les valeurs données par l'user et les transmet dans les appels aux méthodes présente dans les modèles.

Par exemple pour la page de création de projet le controller fait appel a des fonctions qui vont récupérer la liste des clients et des User
pour les injecter dans la vue dans le formulaire de création de projet

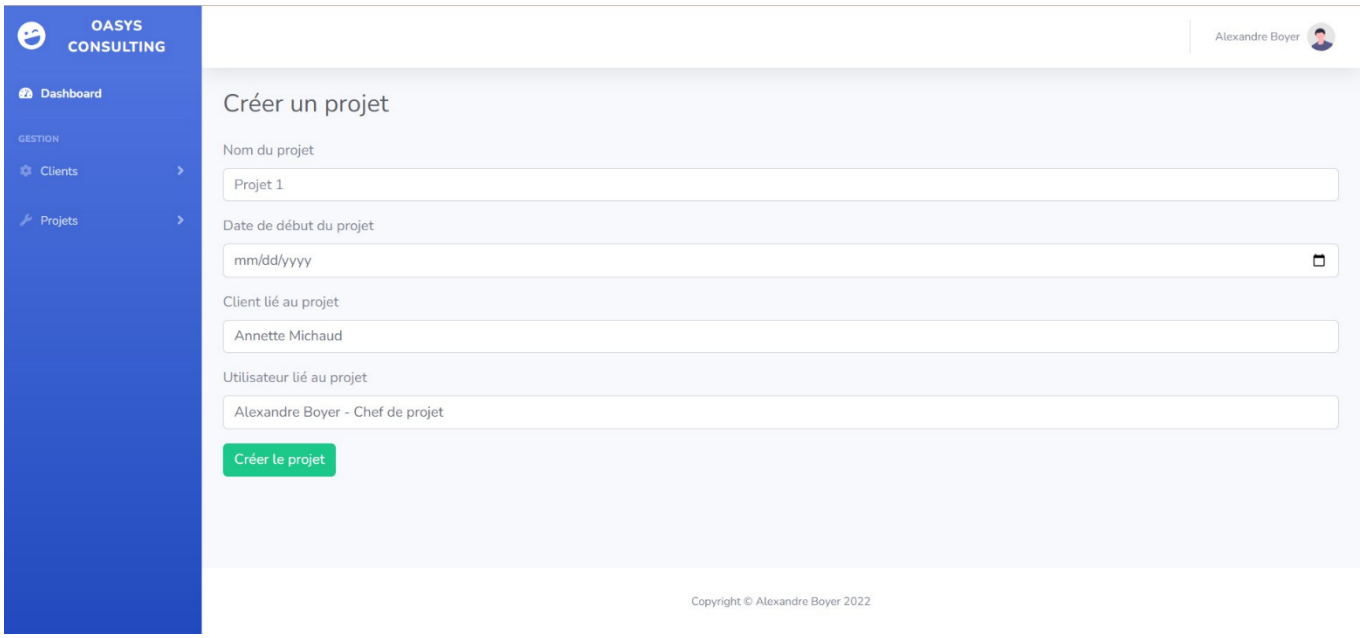
```php
<?php

namespace controller;

2 usages  Alexandre Boyer
class Create_projectController extends Controller
{
    Alexandre Boyer
    public function resolve(): void
    {
        $clientSERVICE = new \model\service\clientSERVICE();
        $userSERVICE = new \model\service\userSERVICE();

        $vars = [
            "listClients" => $clientSERVICE->getClient(),
            "listChefDeProjet" => $userSERVICE->getChefDeProjet(),
            "resultNotification" => false
        ];
```

La Vue :



Dès que l'user a cliqué sur « Créer le projet » dans la vue, cela va déclencher cela dans le controller :

```php
if(isset($_POST['nom_projet']) && isset($_POST['date_debut_projet']) && isset($_POST['id_client']) && isset($_POST['id_user']))  {
    try {
        $projetSERVICE = new \model\service\projetSERVICE();
    }
    catch(\Exception $e) {
        echo "impossible de créer le projet";
        die();
    }

    $vars["resultNotification"]  = $projetSERVICE->createProject($_POST['nom_projet'],$_POST['date_debut_projet'],$_POST['id_client']

    $this->render( view: "create_project", $vars);
    $this->redirect( action: "create_project");
}else{
    $this->render( view: "create_project", $vars);
}
```

Les modèles :

Il existe trois types de modèle :

- Les modèles DAO lié à l'action sur la BDD
- Les méthodes Métier qui servent à créer les objets métier comme les projets par exemple grâce aux données récupérer par le DAO.
- Puis il y a les modèles service qui sont les méthodes finales qui peuvent être appelé par les Controller

Dans le controller est instancié La classe projetService qui contient toutes les méthodes liées aux actions sur les objets projets

Puis est appelé (si tous les champs du formulaire HTML de la vue sont « SET ») la méthode la

classe : createProject et on lui passe en paramètre les infos remplies par l'user dans le formulaire.

Voici donc la méthode createProject appelé :

```
1 usage  ± Alexandre Boyer
public function createProject($libelle_projet,$date_de_debut_projet,$id_client,$id_user) {
    return $this->projetDAO->createProject($libelle_projet,$date_de_debut_projet,$id_client,$id_user);
}
```

Celle-ci fait elle-même appel a une fonction du DAO qui va créer le projet en base de donnée :

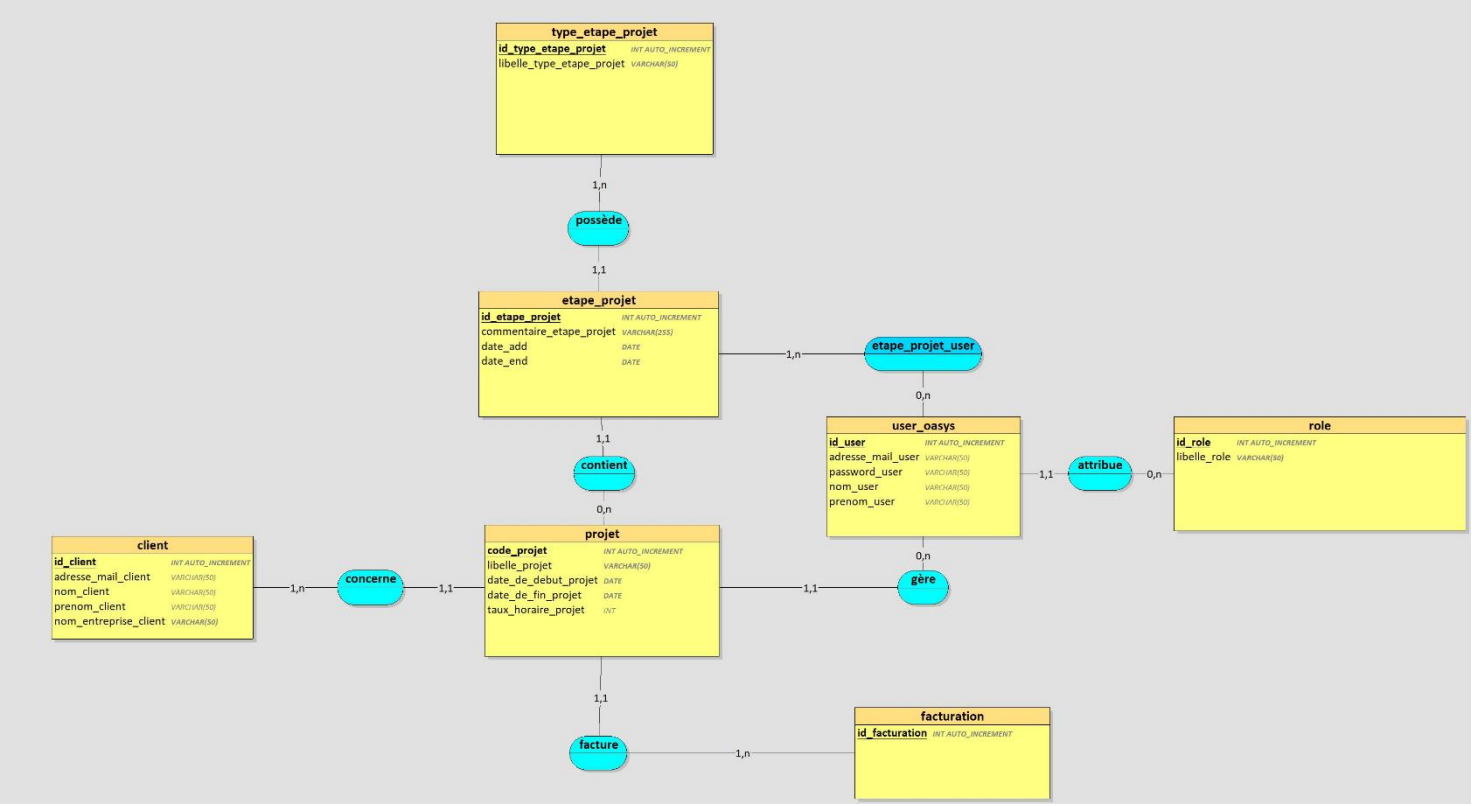
```
public function createProject($libelle_projet,$date_de_debut_projet,$id_client,$id_user)
{ // Créer un projet
    $createproject = $this->Connection->prepare( query: "INSERT INTO " . self::TABLE . " (libelle_projet,date_de_debut_projet,
    id_client,id_user)
        VALUES (:libelle_projet,:date_de_debut_projet,:id_client,:id_user)");
    $res = $createproject->execute(array(
        "libelle_projet" => $libelle_projet,
        "date_de_debut_projet" => $date_de_debut_projet,
        "id_client" => $id_client,
        "id_user" => $id_user
    ));

    return $res;
}
```

Voila comment marche une fonctionnalité de base grâce a l'architecture MVC et comment cela a été refléchi dans ce projet.

3.3. Analyse de la base de données

3.3.1. Modèle conceptuelle de données (MCD – Méthode Merise)



3.3.2. Modèle Logique de données (MLD – Méthode Merise)

facturation = (id_facturation *INT AUTO_INCREMENT***);**

client = (id_client *INT AUTO_INCREMENT*, adresse_mail_client *VARCHAR(50)*, nom_client *VARCHAR(50)*, prenom_client *VARCHAR(50)*, nom_entreprise_client *VARCHAR(50)***);**

type_etape_projet = (id_type_etape_projet *INT AUTO_INCREMENT*, libelle_type_etape_projet *VARCHAR(50)***);**

role = (id_role *INT AUTO_INCREMENT*, libelle_role *VARCHAR(50)***);**

user_oasys = (id_user *INT AUTO_INCREMENT*, adresse_mail_user *VARCHAR(50)*, password_user *VARCHAR(50)*, nom_user *VARCHAR(50)*, prenom_user *VARCHAR(50)*, *#id_role***);**

projet = (code_projet *INT AUTO_INCREMENT*, libelle_projet *VARCHAR(50)*, date_de_debut_projet *DATE*, date_de_fin_projet *DATE*, taux_horaire_projet *INT*, *#id_user, #id_facturation, #id_client***);**

etape_projet = (id_etape_projet *INT AUTO_INCREMENT*, commentaire_etape_projet *VARCHAR(255)*, date_add *DATE*, date_end *DATE*, *#code_projet, #id_type_etape_projet***);**

etape_projet_user = (*#id_user, #id_etape_projet*);