PROJET 13 PHP CRUD Backend + React Frontend

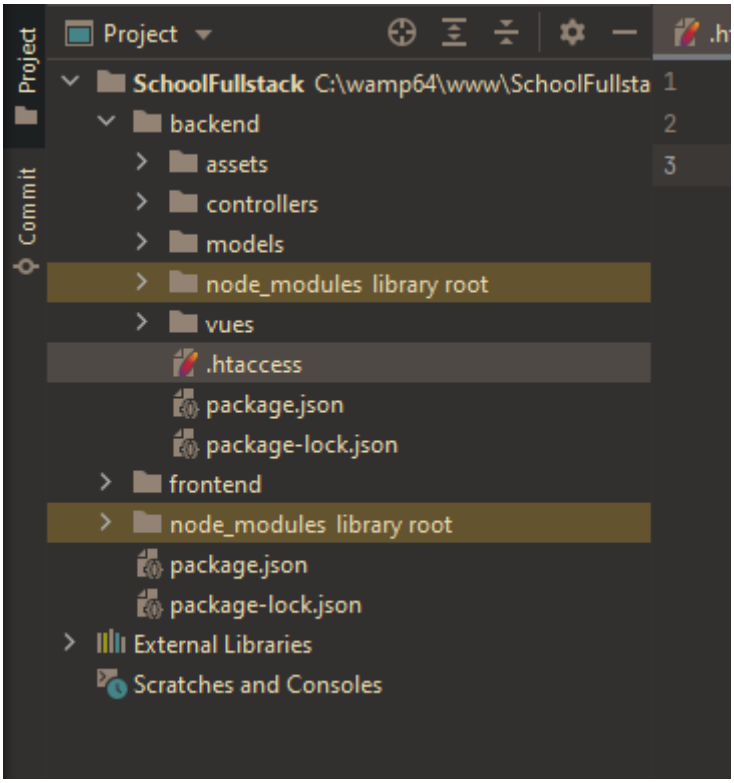
PARTIE A : LA CONNEXION

Ressources :

https://github.com/michelonlineformapro/Projet-12-Backend-PHP-MVC-Secure-React-Frontend

LES ETAPES :

1) Créer un projet PHP (démarrer wamp ou xamp (PHP + apache + MySQL)
2) Créer la structure de votre projet avec le patron de conception (Design Pattern) MVC (Model Vue Controller)



3) Créer un router et votre point entrée apache (.htaccess).
   a. Exemple : ici apache pointe vers le dossier vues + index.php ?=ma_route
   b.

4) Votre router doit pointer vers un fichier de connexion, à l'aide de session_start() et $_SESSION : interdire l'accès au tableau de bord si le visiteur n'est pas connecté.

```php
<?php
session_start();

//ob_start – Enclenche la temporisation de sortie (memoire tampon)
ob_start();

//Appel des controllers
require_once '../controllers/UtilisateurController.php';
require_once '../controllers/EtudiantController.php';

//Cle / Valeur URL
if(isset($_GET['url'])){
    $url = $_GET['url'];
}else{
    $url = 'connexion';
}

//Les routes

if($url === 'connexion'){
    $title = "Dev_School -CONNEXION-";
    require_once 'connexion.php';
}elseif ($url === 'dashboard' && $_SESSION['est_connecter'] === true){
    $title = "Dev_School -DASHBOARD-";
    require_once 'dashboard.php';
}elseif ($url === 'deconnexion'){
    require_once 'deconnexion.php';
```

5) Créer un fichier template.php qui contient gabarit html utilisable sut toutes les pages à l'aide de la mémoire tampon (ob_start() et ob_get_clean())
6) Créer une base de donnée et une table utilisateurs (insérer un ou plusieurs administrateur)
7) Créer une classe de connexion à MySQL avec l'objet PDO (privilégié le design pattern Singleton pour éviter les multiples instances de connexion)
8) Ajouter les options de vos futures requêtes http a votre de classe de connexion PDO:

```php
//Option json_encode http
header( header: "Access-Control-Allow-Origin: *");
header( header: "Access-Control-Allow-Methods: PUT, PATCH, GET, POST, DELETE");
header( header: "Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept");
```

9) Créer une classe (modèle) de connexion des utilisateurs qui hérite de la classe mère PDO de connexion à MySQL.

10) Créer une fonction publique de connexion à l'aide email et mot de passe.

a. Dans une condition lorsque la connexion matches, créer vos variables de session avec $_SESSION['estConnecter'] = true

b.Puis une redirection vers votre tableau de bord (sinon afficher les erreurs).

```php
<?php
//Appel de la classe MERE PDO
require_once "Database_modele.php";
class Utilisateur_modele extends Database_modele
{
    //Init des variables
    private $id_utilisateur;
    private $nom_utilisateur;
    private $email_utilisateur;
    private $password_utilisateur;

    //Connexion
    public function connecterUtilisateur(){
        //Connexion PDO
        $db = $this->getPDO();
        //Affectation des variables
        $this->email_utilisateur = $_POST['email'];
        $this->password_utilisateur = $_POST['password'];

        //Requête SQL
        $sql = "SELECT * FROM utilisateur WHERE email = ? AND password = ?";
        //Requête préparée
        $statement = $db->prepare($sql);
        //Liés les paramètre de la requête aux données de la table
        $statement->bindParam( param: 1,  &var: $this->email_utilisateur);
        $statement->bindParam( param: 2,  &var: $this->password_utilisateur);
        //Execution de la requête
        $statement->execute();
        //On compte les element de la table et on check les valeurs
        //Si il y a au main une valeur
        if($statement->rowCount() >= 1){
            $row = $statement->fetch( mode: PDO::FETCH_ASSOC);
            $this->id_utilisateur = $row['id'];
            $this->email_utilisateur = $row['email'];
            $this->password_utilisateur = $row['password'];
            //Check si ca matche + condition
            if($this->email_utilisateur === $row['email'] && $this->password_utilisateur === $row['password']){
                //On demarre une session
                session_start();
                //On stock les valeurs temoraire dans une session
                $_SESSION['est_connecter'] = true;
                $_SESSION['email'] = $this->email_utilisateur;
                $_SESSION['password'] = $this->password_utilisateur;
                //On fait une redirection vers le dashboard
                header( header: 'Location: dashboard');
            }else{
                //Sinon on declenche une erreur
                echo "<div class='mt-3 notification is-danger'>Erreur ! Merci de vérifié votre email et mot de passe</div>";
            }
        }else{
            if(!$_SESSION['est_connecter']){
                echo "<script>alert(\"Aucun utilisateur ne possèdent cet email et mot de passe\")</script>";
            }
            echo "<div class='mt-3 notification is-danger'>Aucun utilisateur ne possèdent cet email et mot de passe</div>";
        }
    }
}
```

11) Créer un fichier connexion_controller.php qui appelle son modèle et une fonction de connexion qui stock l'instance du modèle et appel la méthode de connexion, la fonction retourne une variable.

```php
<?php
require_once '../models/Utilisateur_modele.php';

//Connecter un utilisateur
function connexion(){
    //Instance du modele
    $utilisateur = new Utilisateur_modele();
    //Appel de la methode du modele
    $connecter_utilisateur = $utilisateur->connecterUtilisateur();
    return $connecter_utilisateur;
}
```

12) Votre router (index.php) appel la vue connexion.php, cette dernière affiche un formulaire de connexion avec la méthode POST.

a. Le bouton de soumission appel la fonction connexion de votre Controller.

b. Les erreurs sont donc gérées depuis le model et affichée dans votre vue en cas de mauvais email et mot de passe.

```php
///////////////CONNEXION////////////////////
if($url === 'connexion'){
    $title = "Dev_School -CONNEXION-";
    require_once 'connexion.php';

    ///////////////ACCES DASHBOARD////////////////////
}elseif ($url === 'dashboard' && $_SESSION['est_connecter'] === true){
    $title = "Dev_School -DASHBOARD-";
    require_once 'dashboard.php';
    ///////////////DECONNEXION////////////////////
}elseif ($url === 'deconnexion'){
    require_once 'deconnexion.php';
```

13) Le fichier dashboard.php (tableau bord) est inaccessible si le visiteur n'est pas connecté :

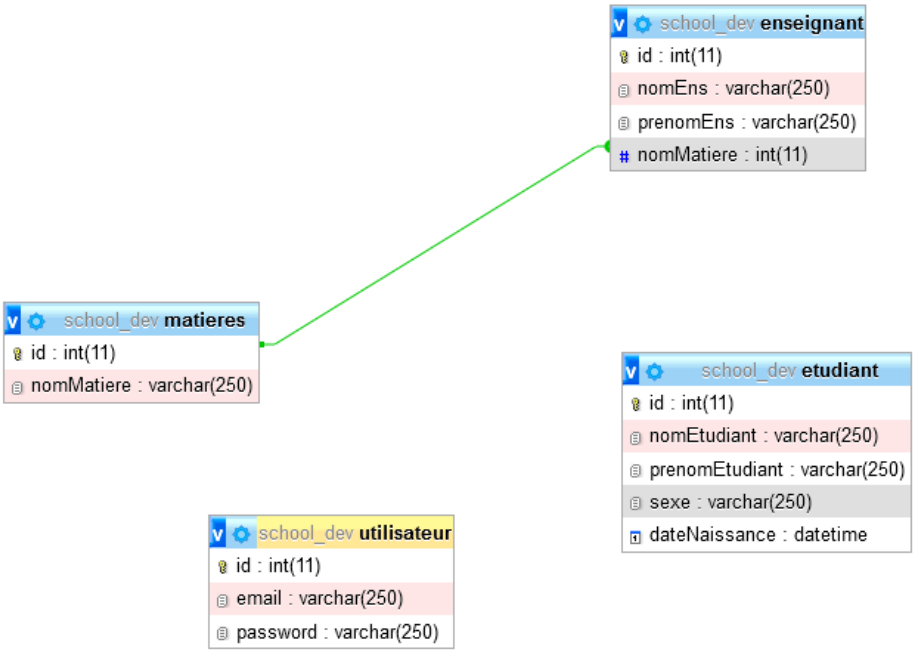
```php
<?php
require_once '../models/Etudiant_modele.php';
$etudiants = new Etudiant_modele();
$listeEtudiants = $etudiants->afficherEtudiants();


if($_SESSION['est_connecter']){

    ?>
    <div class="container is-fluid">
        <div class="notification is-success">
            <div class="title is-4">Bonjour :<b class="has-text-info"> <?= $_SESSION['email'] ?></b>
                <a href="deconnexion" class="is-pulled-right is-danger">DECONNEXION</a>
            </div>
        </div>
        <div class="box">
            <div class="title is-2 has-text-centered has-text-danger">TABLEAU DE BORD</div>
```

14) Dans votre router (index.php) : Si aucune route ne matche => exécuter une redirection :

```
    }
    //Si aucune route ne matches
}else{
    header( header: 'Location: http://localhost/SchoolFullstack/backend/connexion');
}

//ob_get_clean – Lit le contenu courant du tampon de sortie puis l'efface
$content = ob_get_clean();

require_once 'template.php';
```

PARTIE B : LE CRUD ECOLE

1) Exemple de base du MCD (Modèle Conceptuel de Données)
2) UTILISATEUR (id (int primary key, email (varchar 250), password(varcher 250))
3) ETUDIANT (id (int primary key), nomEtudiant (varchar 250), prenomEtudiant (varchar 250), sexe (varchar 250), dateNaissance (DATETIME))
4) ENSEIGNANT (id (int primaty key), nomEns (varchar 250) prenomEns(varchar 250), #nomMatiere (int foreign key INDEX))
5) MATIERE (id (int primary key), nomMatiere (varchar 250))



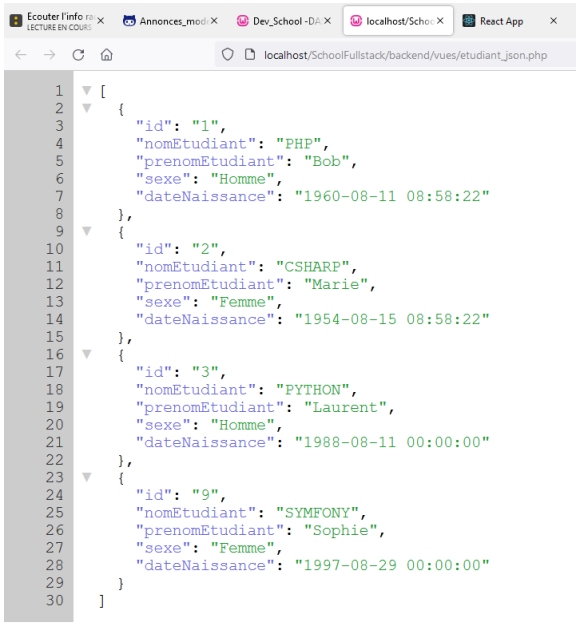
6) Dans un premier temps effectuer les opérations de CRUD dans votre etudiant_modele.php avec la table étudiant qui n'a aucune dépendance fonctionnelle (pas de clé étrangère)
    a. Les fonctions de votre Controller seront appelées dans votre router.

7) Transformer vos données SQL au format json à l'aide de la fonction PHP json_encode, les données seront affichées dans un fichier PHP(sans routing) => etudiant_json.php

EXEMPLE AVEC LA TABLE ENEIGNAT ET UNE JOINTURE /

```php
//Exporter les enseignat au format json
public function enseignantJson(){
    //init connexion
    $db = $this->getPDO();
    //Requête SQL

    $enseignantArray = [];
    $sql = "SELECT * FROM enseignant INNER JOIN  matieres ON enseignant.nomMatiere = matieres.id_matiere";
    //Recuperation des valeur avec query
    /*
     * PDO::query — Exécute une requête SQL, retourne un jeu de résultats en tant qu'objet PDOStatement
     */
    $enseignants = $db->query($sql);

    if($enseignants){
        //Init du compteur d'elements
        $i = 0;
        //Boucle de lecture
        while ($row = $enseignants->fetch( mode: PDO::FETCH_ASSOC)){
            $enseignantArray[$i]['idEns'] = $row['idEns'];
            $enseignantArray[$i]['nomEns'] = $row['nomEns'];
            $enseignantArray[$i]['prenomEns'] = $row['prenomEns'];
            $enseignantArray[$i]['nomMatiere'] = $row['nomMatiere'];
            //Incremente le compteur
            $i++;
        }

        //Si tous va bien on encode SQL
        echo json_encode($enseignantArray, flags: JSON_PRETTY_PRINT);
    }else{
        http_response_code( response_code: 404);
    }
}
```

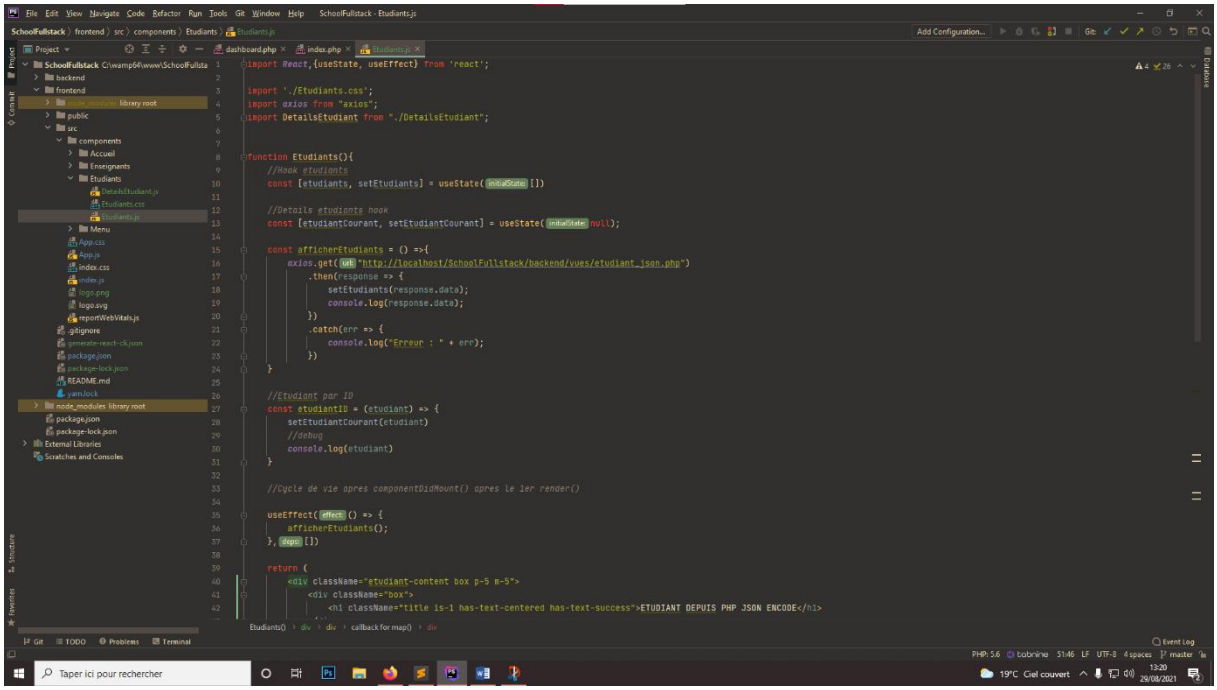
Un lien <a href= target_blank> appel votre fichier php et les données sérialisées au format json.



8) Créer un dossier Frontend et générer une CRA (create react app)

```
npx create-react-app mon-app
cd mon-app
npm start
```

9) Créer un composant étudiant et à l'aide des hook et axios (Requête http) => afficher tous les étudiants de votre backend :



10) Pour accéder a votre composant créer un système de routing à l'aide de :
    a. React Router : https://reactrouter.com/web/guides/quick-start

```jsx
import React from 'react';
import './Menu.css';
import {BrowserRouter as Router, Switch, Route, Link} from "react-router-dom";
import Etudiants from "../Etudiants/Etudiants";
import logo from '../../logo.png';
import Accueil from "../Accueil/Accueil";
import Enseignants from "../Enseignants/Enseignants";

export default function Menu(){
    return (
        <Router>
            <div>
                <nav className="navbar has-shadow" role={"navigation"}>
                    <div className={"navbar-start"}>
                        <span className={"navbar-item"}>
                            <Link className="title is-1 has-text-info" to="/">
                                <img src={logo}/>
                            </Link>
                        </span>
                        <span className={"navbar-item"}>
                            <Link to="/">ACCUEIL</Link>
                        </span>
                        <span className={"navbar-item"}>
                            <Link to="/etudiants">ETUDIANTS</Link>
                        </span>
                        <span className={"navbar-item"}>
                            <Link to="/enseignants">ENSEIGNANTS</Link>
                        </span>
                    </div>
                </nav>

                {/* La balise switch regarde a travers les enfants des Route et rend les elements qui matches */}
                <Switch>
                    {/* ici exact specifie le point d'entrée par defaut*/}
                    <Route exact path="/">
                        <Accueil/>
                    </Route>
                    <Route path="/etudiants">
                        <Etudiants/>
                    </Route>
                    <Route path="/enseignants">
                        <Enseignants/>
                    </Route>
                </Switch>
            </div>
        </Router>
    );
}
```

11) Afficher les détails de chaque étudiant à l'aide de la programmation modulaire et les passages des propriétés (props) React (composant réutilisable) dans un composant <DetailsEtudiant props={valeur ID}/>

HOOK :

```jsx
//Details etudiants hook
const [etudiantCourant, setEtudiantCourant] = useState(null);
```

FONCTION :

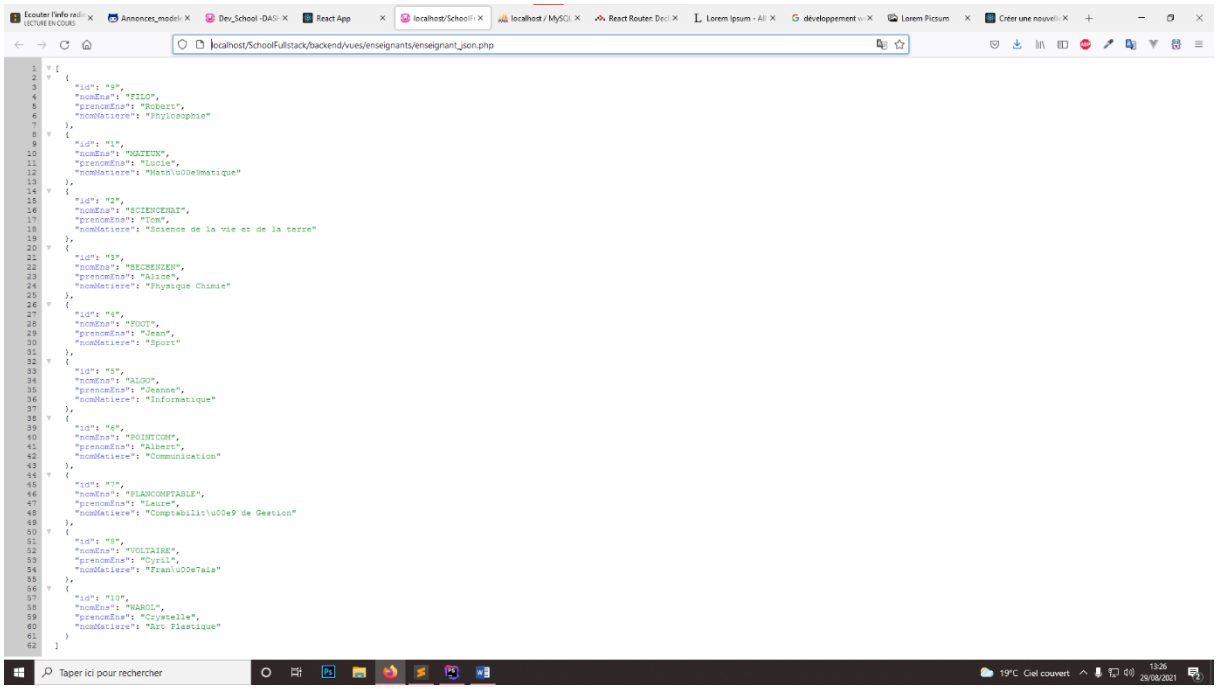
```jsx
//Etudiant par ID
const etudiantID = (etudiant) => {
    setEtudiantCourant(etudiant)
    //debug
    console.log(etudiant)
}
```

LA CONDITION TERNAIRE JSX :

```
return (
    <div className="etudiant-content box p-5 m-5">
        <div className="box">
            <h1 className="title is-1 has-text-centered has-text-success">ETUDIANT DEPUIS PHP JSON ENCODE</h1>
        </div>

        {etudiantCourant ? (
            <DetailsEtudiant etudiant={etudiantCourant}/>
        ) : (

            <div className="columns is-multiline">
                {etudiants.map(etudiant =>
                    <div key={etudiant.id} onClick={() => etudiantID(etudiant, etudiant.id)} className="mt-5 column is-2">
                        <div className="card">
                            <div className="card-image">
                                <figure className="image">
                                    <img src="https://picsum.photos/200" alt="" title=""/>
                                </figure>
                            </div>
                            <div className="card-content">
                                <div className="media">
                                    <div className="media-content">
                                        <p className="title is-4">PRENOM :<br /> {etudiant.prenomEtudiant}</p>
                                        <p className="title is-4">NOM :<br /> {etudiant.nomEtudiant}</p>
                                    </div>
                                </div>

                                <div className="content">
                                    <p>SEXE : {etudiant.sexe}</p>
                                    <p>DATE DE NAISSANCE : {etudiant.dateNaissance}</p>
                                </div>
                            </div>
                        </div>
                    </div>
                )}
            </div>
        )}
    </div>
);
```

PARTIE C : Les clés étrangères et json ?

1) Créer 2 tables (enseignant et matière) et une dépendance fonctionnelle entre les enseignants et le nom de la matière enseignée.
2) Effectuer les opérations de CRUD (Requête SQL avec jointure) pour ces 2 tables.
3) Pour les formulaires ajouter et editer : ajouter un boucle forEach qui liste toutes les matières
4) Sérialisé au format json les données SQL des enseignants et des matières vers une URL au format json à l'aide de PHP json_encode (sérialisation).



5) Répéter les opérations d'affichage dans la partie frontend avec React (deux nouveaux composant) avec l' url des enseignants et des matières.

LES + :

6) Ajouter des filtres de tri dans React (ex : barre de recherche, check box, liste de choix, etc…) à l'aide des fonctions filter() et includes() JavaScript et des conditions à l'aide des ternaires :
7) RAPPEL {condition1 ? ''OUI '' : ''NON ''}
8) OU {condition1 ? (Block JSX) : (Sinon autre block JSX)}

BRAVO VOUS ETES UN DEVELOPPEUR FULLSTACK PHP + JAVASCRIPT