|  |
| --- |
|  |
| RAPPORT DU PROJET |
| **Gestion des Étudiants** |

|  |
| --- |
| **Encadré par :**   * **M.SAADI MOSTAFA**     **Réalisé par :**   * **DARHOUANE ASSIA** * **ROMANE HIBA** |
|  |

***Juin 2022***

**Sommaire**

*Remerciement ………………………………………………………..3*

*1. Introduction ………………………………….…………………….4*

*2. Objectifs du projet :..................................................................4*

*3. Pages HTML……….……………………………………………...5*

*3-1 Page “ comptes.php ’’*

*3-2 Page “ ajouter\_compte.php ’’*

*3-3 Page “ ClassEtudiant.php ’’*

*3-4 Page “ enregistrer\_compte.php ’’*

*3-5 Page “ supprimer\_compte.php ’’*

*3-6 Page “ modifier\_compte.php ’’*

*3-7 Page “ enregistrer\_modifier\_compte.php ’’*

*4- Affichage des exécutions……….………………………………...28*

*5-Conclusion…………………………………………………………..31*

**Remerciement**

Nous voulons exprimer par ces quelques lignes de remerciements notre gratitude envers vous Mr. SAADI MOSTAFA , de nous avoir donné la chance de tester nos connaissances en développement web et de réaliser ce projet.

Nous vous remercions aussi pour vos efforts, vos conseils et votre disponibilité durant ce semestre en espérant d’être à la hauteur de vos attentes .

1. *Introduction* :

Ce Mini projet est la meilleure occasion pour mettre en pratique nos connaissances et savoir-faire acquis pendant la formation.

Dans ce cadre est situé le thème de ce projet qui s’intéresse à la réalisation d’un site web dédié pour la Gestion des étudiants.

2. Objectifs du projet :

Développement d’une application de gestion des étudiants basée sur les technologies WEB :

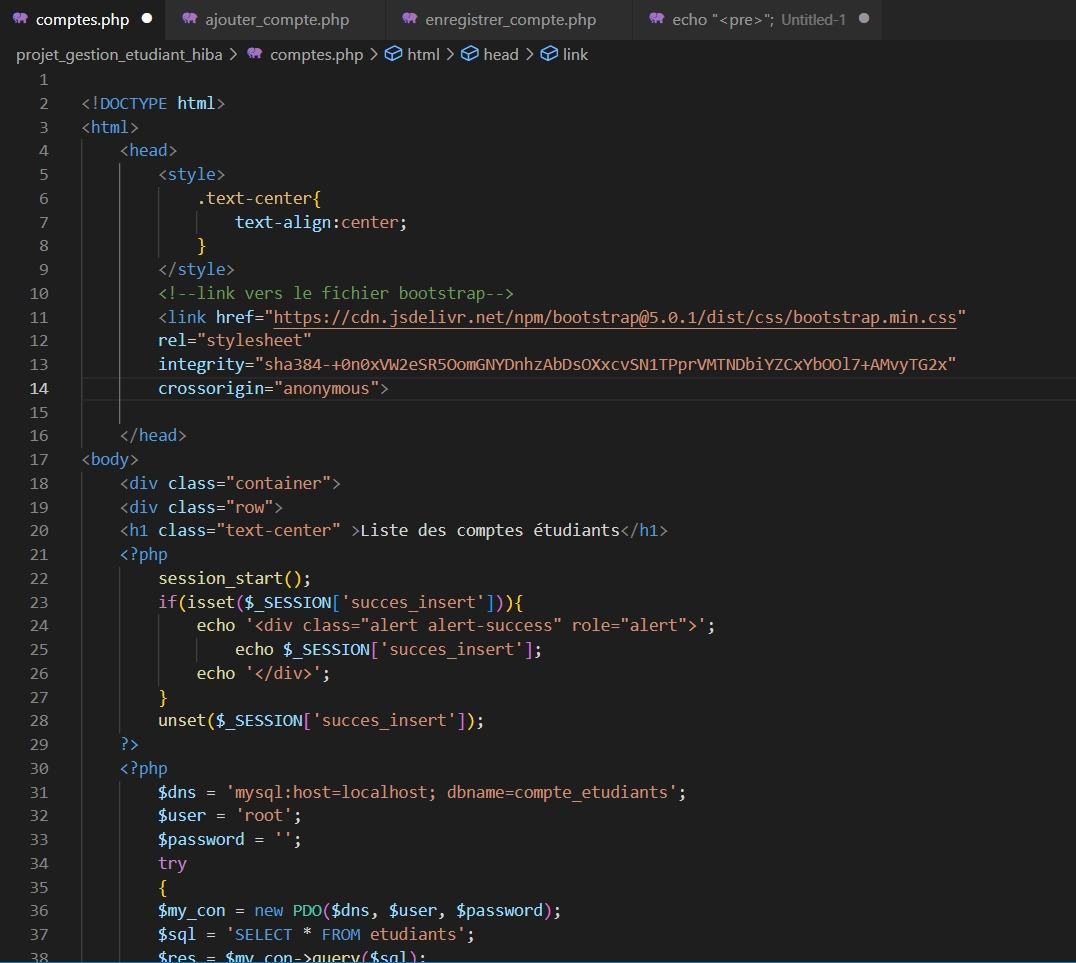
* Fournir une liste des étudiants à partir de la DataBase.
* Ajouter un nouvel étudiant.
* Assurer la modification et la suppression des étudiants déjà existants.

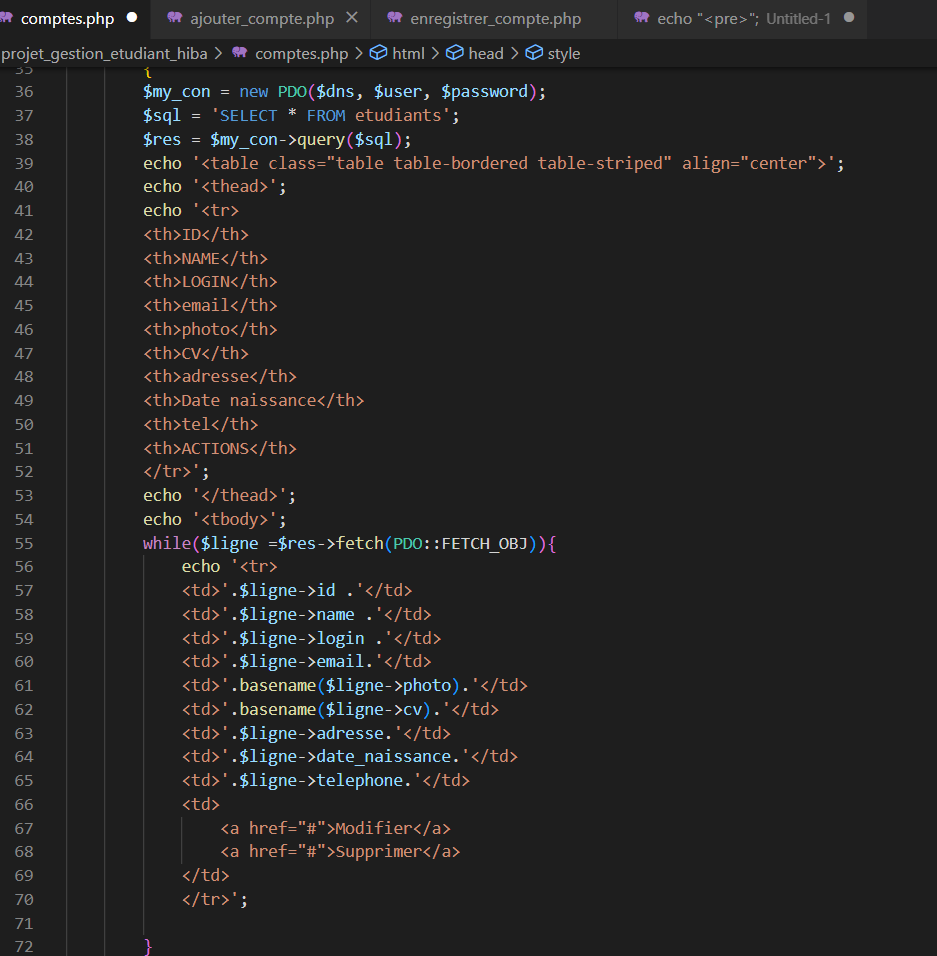
3. Pages HTML :

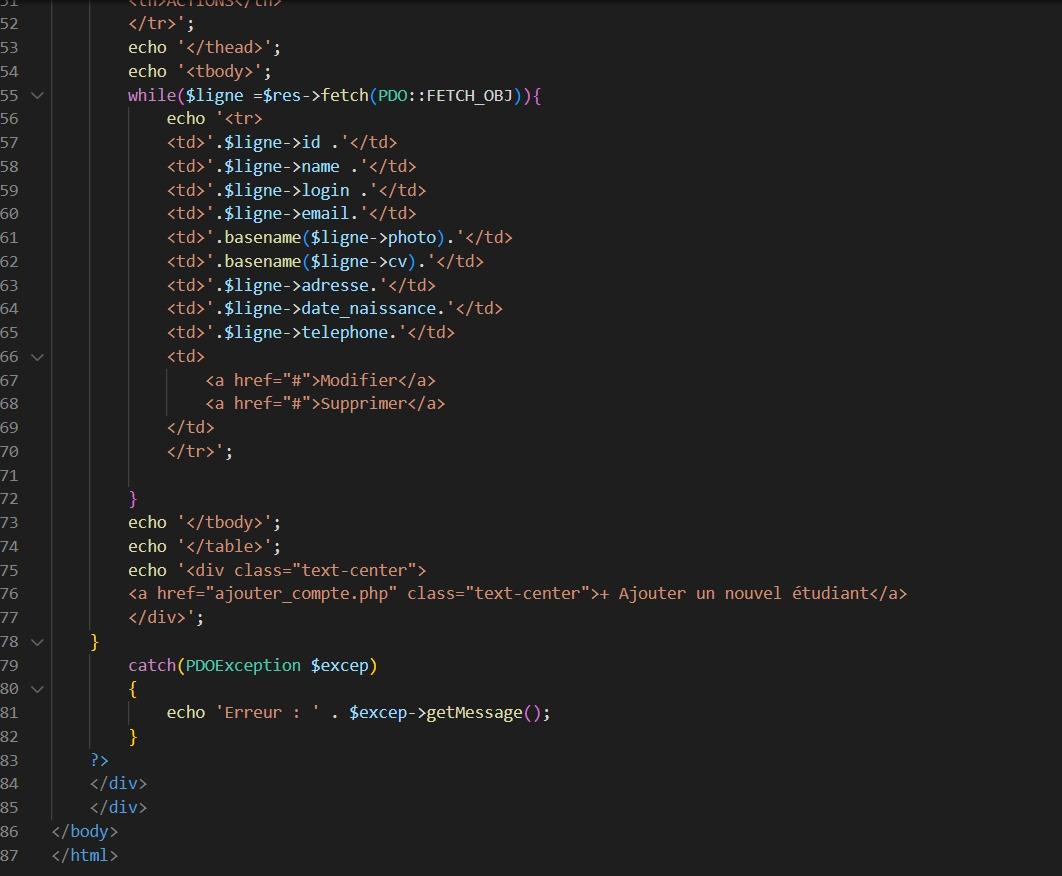
3-1 Page “ comptes.php ’’:

Cette page permet de lister les enregistrements des comptes étudiants de la base de données “***compte\_etudiants***”, comme illustré dans l’écran ci-dessous :

* On s'intéresse d’abord à l’appel du fichier Bootstrap qui organise et gère la mise en page de notre site web (adapté à toutes les plateformes : PC Ordinateur, Mobile ou Tablettes) sans avoir besoin de coder toute la partie [CSS](https://fr.tuto.com/blog/2020/10/css.htm).
* *Que se passera-t-il après l’enregistrement d’un étudiant avec succès ?*
* Le code PHP ci-joint contient une session qui constitue un moyen de stocker les données propres à chaque utilisateur sur le serveur. Grâce à cette fonctionnalité, on aura une redirection vers la même page tout en affichant une alerte “L’étudiant est enregistré avec succès”.
* On utilise -*PDO: PHP Data Objects-* afin d’interroger la base de données via des requêtes [SQL](https://www.chiny.me/glossaire.php#sql), et on passe à l’affichage en utilisant *PDO::FETCH\_OBJ* qui retourne les résultats sous forme d'un objet dont les noms de propriétés correspondent aux noms des colonnes.



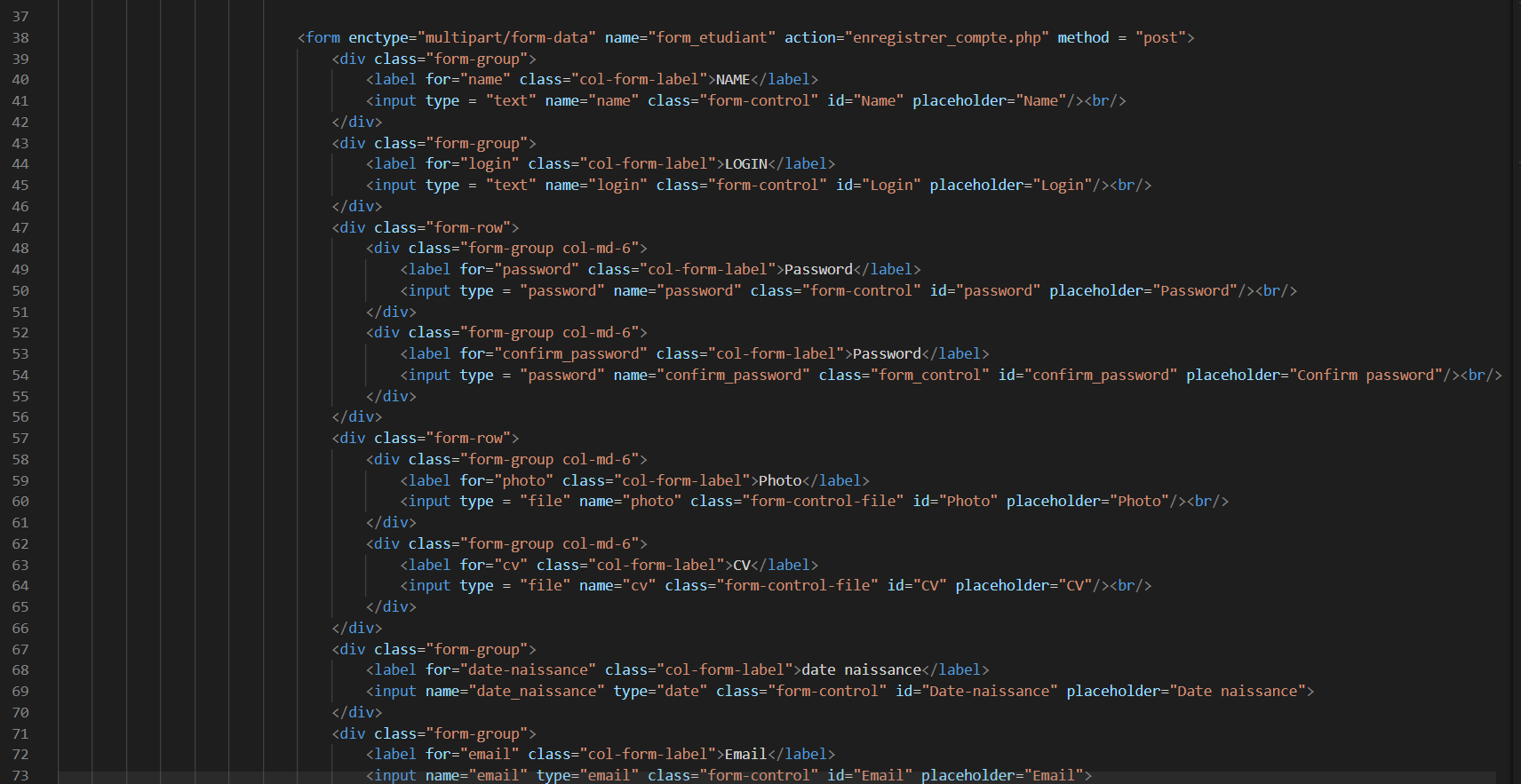
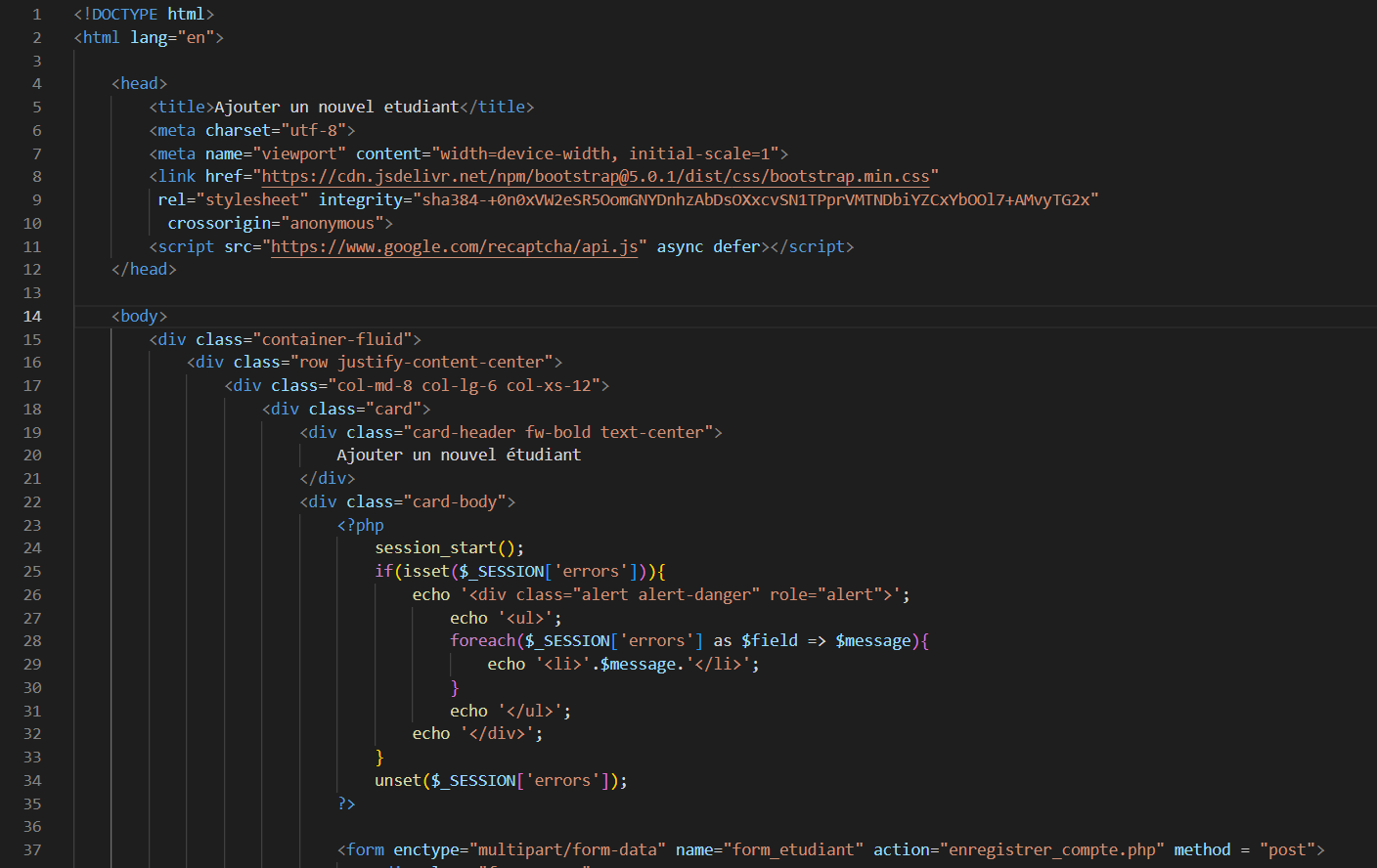


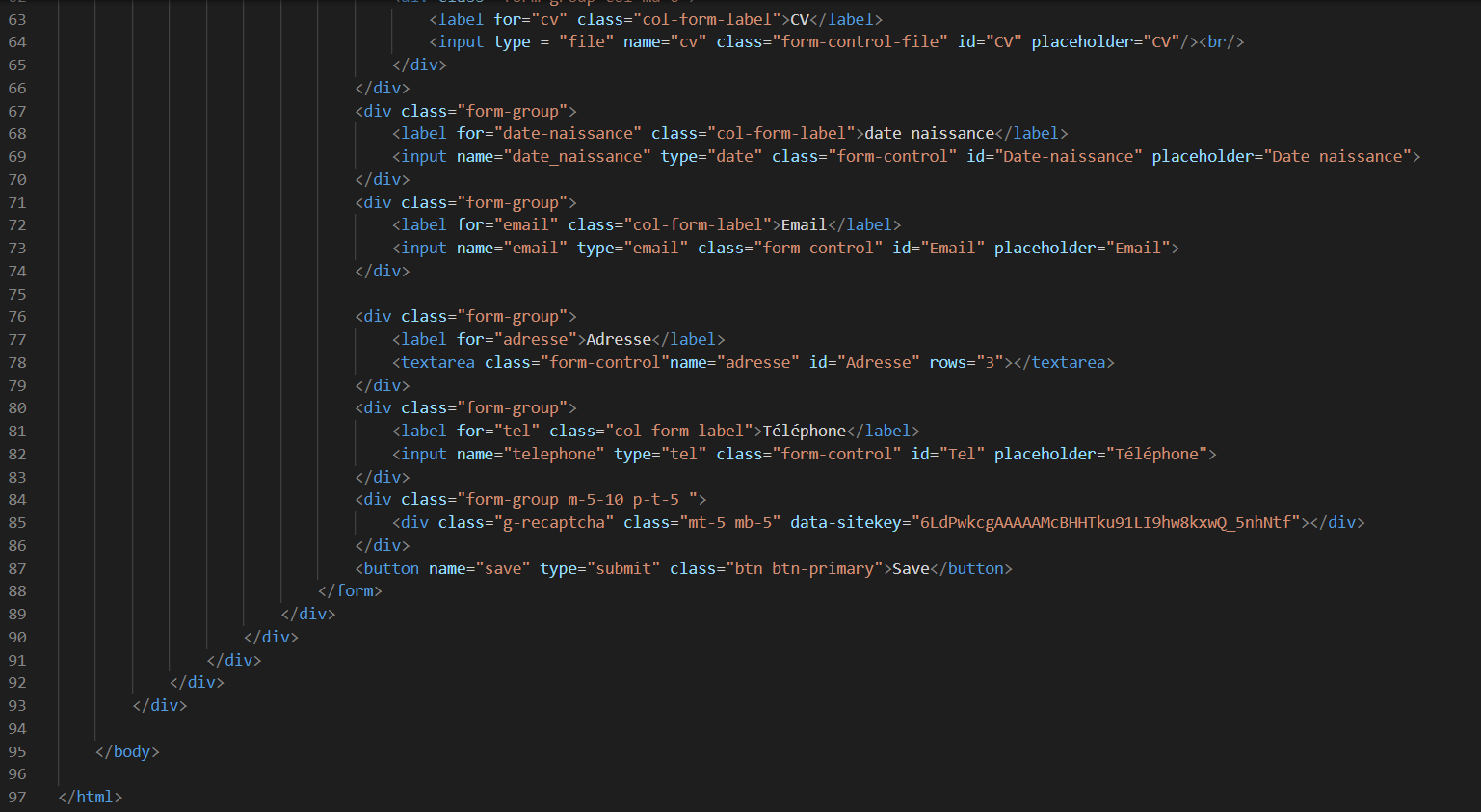


3-2 Page “ ajouter\_compte.php ’’:

Cette page contient principalement un formulaire qui permet à l’utilisateur de s’enregistrer:

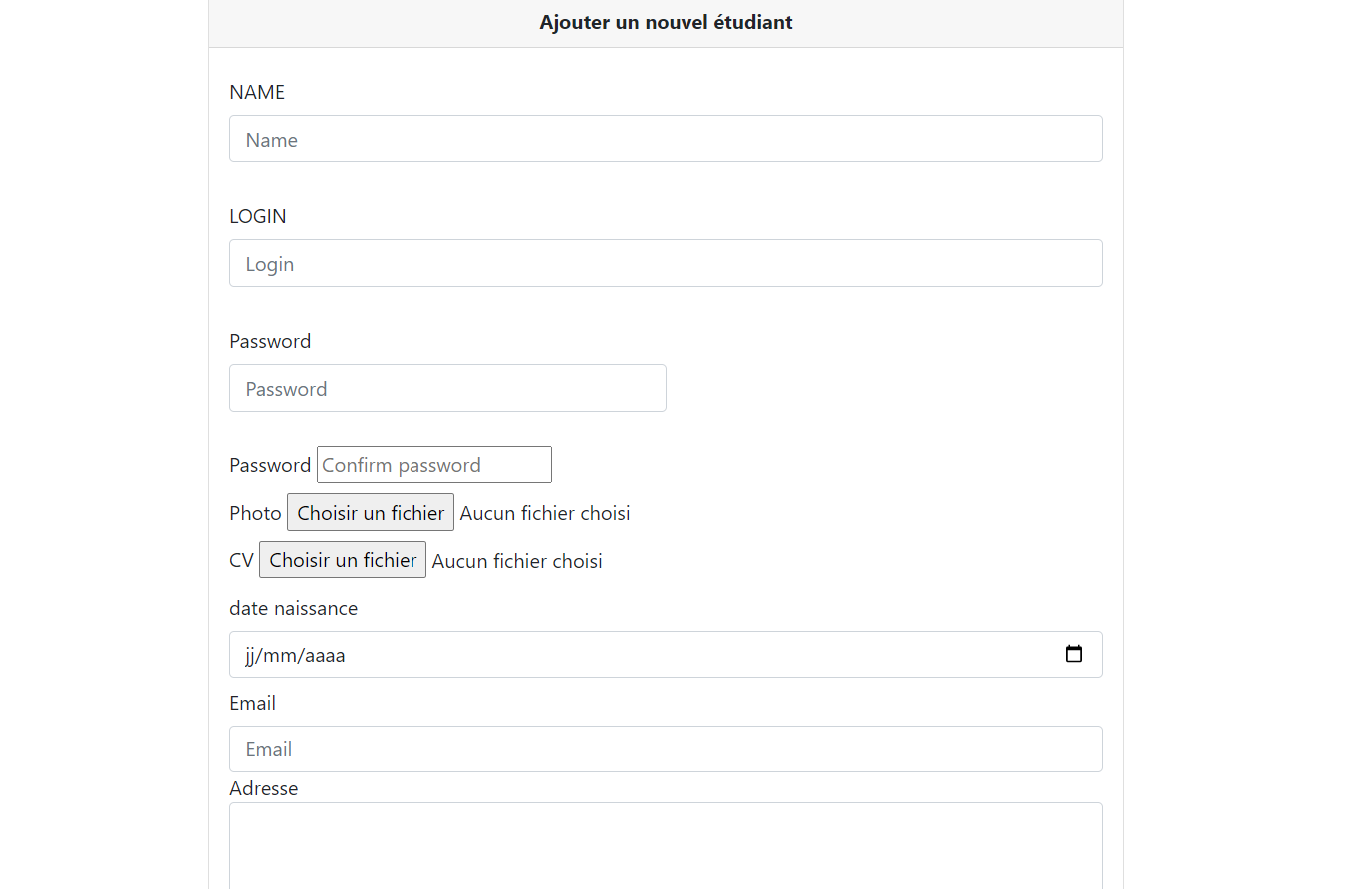
* Comme c’est expliqué ci-dessus on appelle le fichier Bootstrap.
* On définit une session afin de capter les erreurs dans le cas des champs vides, on notifie l’utilisateur à travers des alertes de message en rouge (voir résultats d’affichage).

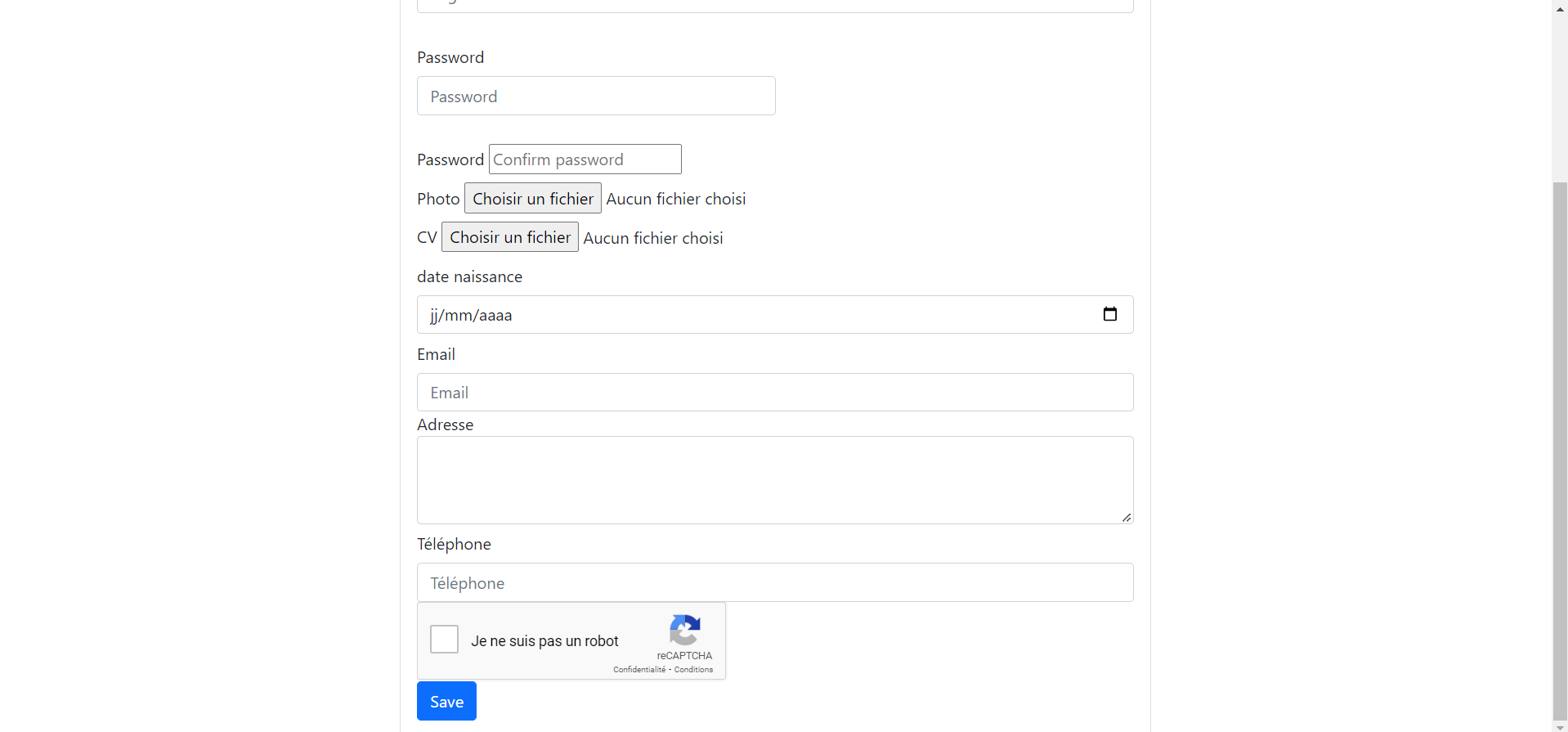




Affichage D'exécution:

* Formulaire d’enregistrement :





3-3 Page “ ClassEtudiant.php ’’:

Cette page contient la partie principale du traitement du formulaire , par exemple la détection des champs vides , tout ce passe au niveau de cette partie, et grâce à la programmation orientée objet POO ,on aura une structure générale du code plus claire, plus modulable et plus facile à maintenir et à déboguer.

* La classe - Etudiant - dont on déclare:
* Un tableau -fields- contenant les champs du formulaire (name,login,password…)
* Sa fonction principale est -ajouter- elle prend comme argument le tableau des données POST , tout en effectuant un test de la méthode d’envoi POST .

→ On intercepte les erreurs au niveau de chaque champ à l’aide de la méthode privée validateFormulaire.

→ Concernant le test !empty($errorValidation), sert à détecter que l’erreur se trouve dans un champ du formulaire puisque le tableau n’est pas vide, on collecte toutes les erreurs captées dans la variable global Session .

→Au cas où l'étudiant s’est enregistré avec succès , on stocke le message de succès dans la variable globale session suivi d’une redirection vers la page des comptes.

* La privateFunction -saveEtudiant - , où se passe l’enregistrement dans la base de données

→ appel de la méthode storeFile pour le CV ainsi que la photo.

→Connexion avec la base de données.

→L’utilisation de la requête SQL insert into qui insère les données dans la table etudiants.

* Private function -storeFile- ,on l’avait déclarée comme private car elle sera accessible à l'intérieur de la classe seulement, elle prend les fichiers déposés dans le formulaire et les stocke dans un dossier nommé -pathToStoreFiles- déclarée.
* La privateFunction -validateFormulaire - ,mentionnée en haut , nous permet la validation des champs et prend en argument le $\_POST du formulaire :

→ le premier test nous retourne false s’il n'y a pas de données

→ la boucle foreach itère sur chaque élément du tableau associatif de cette manière champs=>valeur.

→ le $responseValidation contient 2 valeurs : **false** dans le cas du champ vide ou inexistant , et le message d’erreur et cela se fait à l'aide de la méthode -validateField- .

→ Puisque le formulaire contient les deux champs “Password “ et “confirm Password” , pour la vérification on aboutit au test if ,sinon on affiche l’erreur “le mot de passe et la confirmation ne sont pas identiques”.

→ Concernant la méthode -validateField- , on en aura besoin d’un tableau nommé response où on stocke le statut et le message , cela se fait selon un test de l'existence d’un nom du champ .

* La public Function -supprimer- , nous permet la suppression d’un compte :

→ on se connecte, mais en utilisant cette fois-ci la requête de suppression DELETE FROM .

→ Si la suppression a été effectuée avec succès , on aura le message “ l’étudiant a été supprimé avec succès” en utilisant la variable globale session.

* La public Function -editer- , nous permet de retourner les données qu’on désire modifier d’un compte .

→ en utilisant la requête SELECT \* FROM.

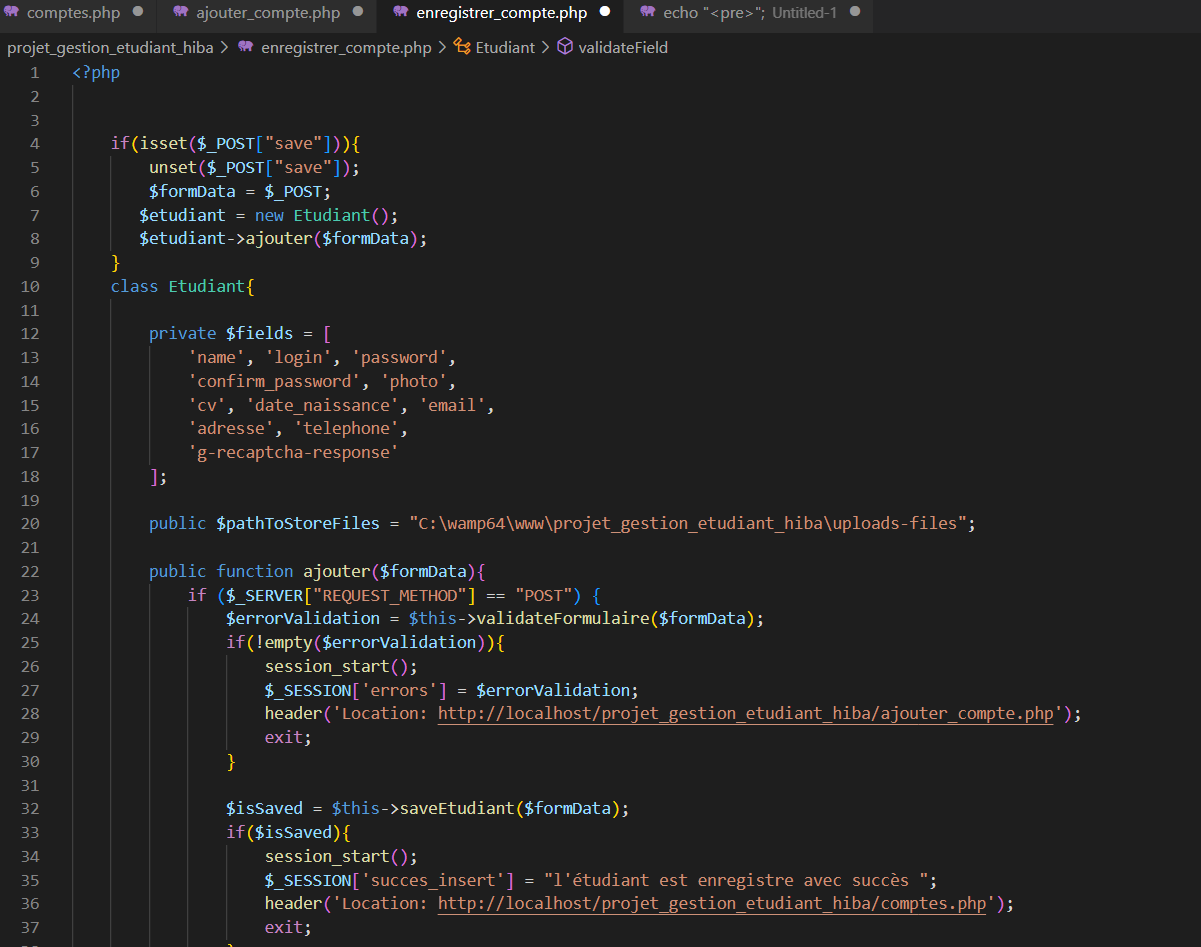
* La public Function -modifier- , sert à modifier les données dans la base de données

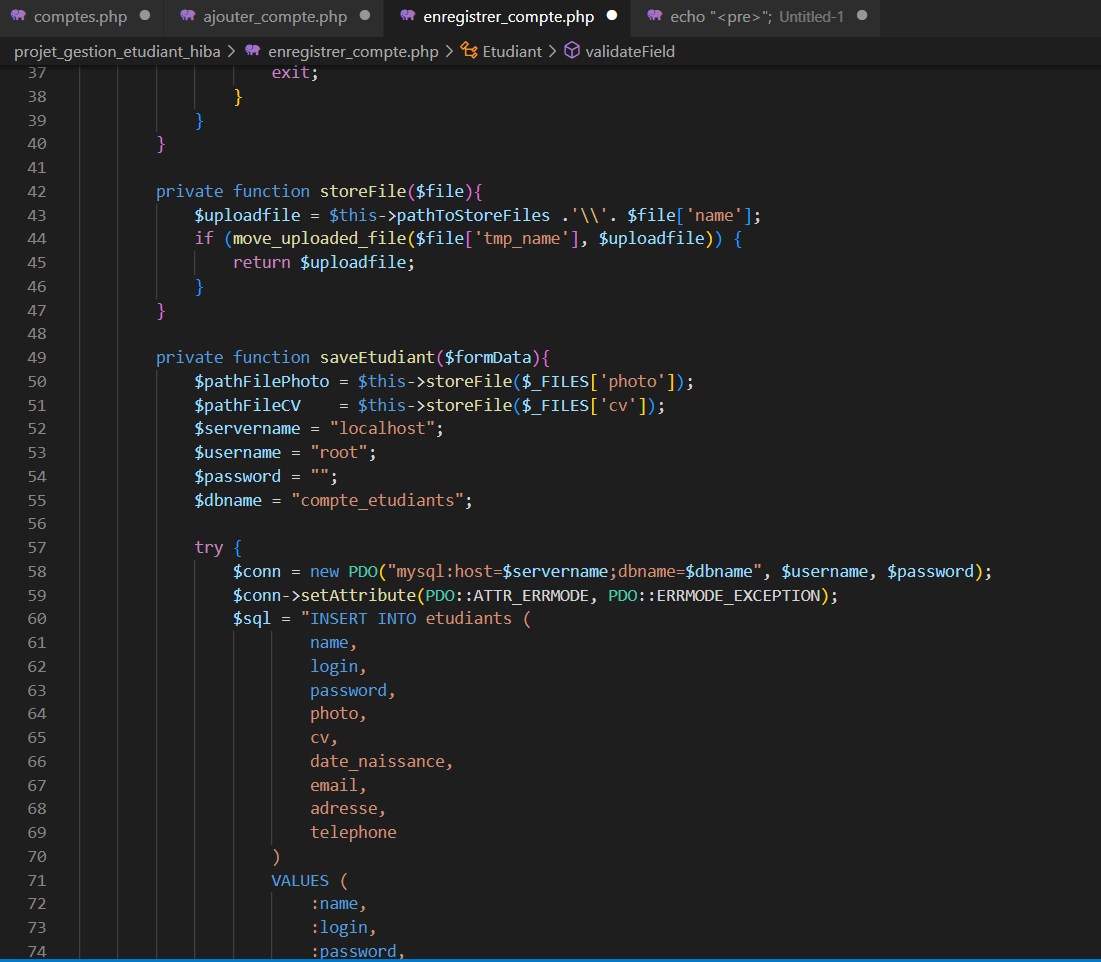
→ on teste tout d’abord les erreurs des champs, et on les envoie dans Session.

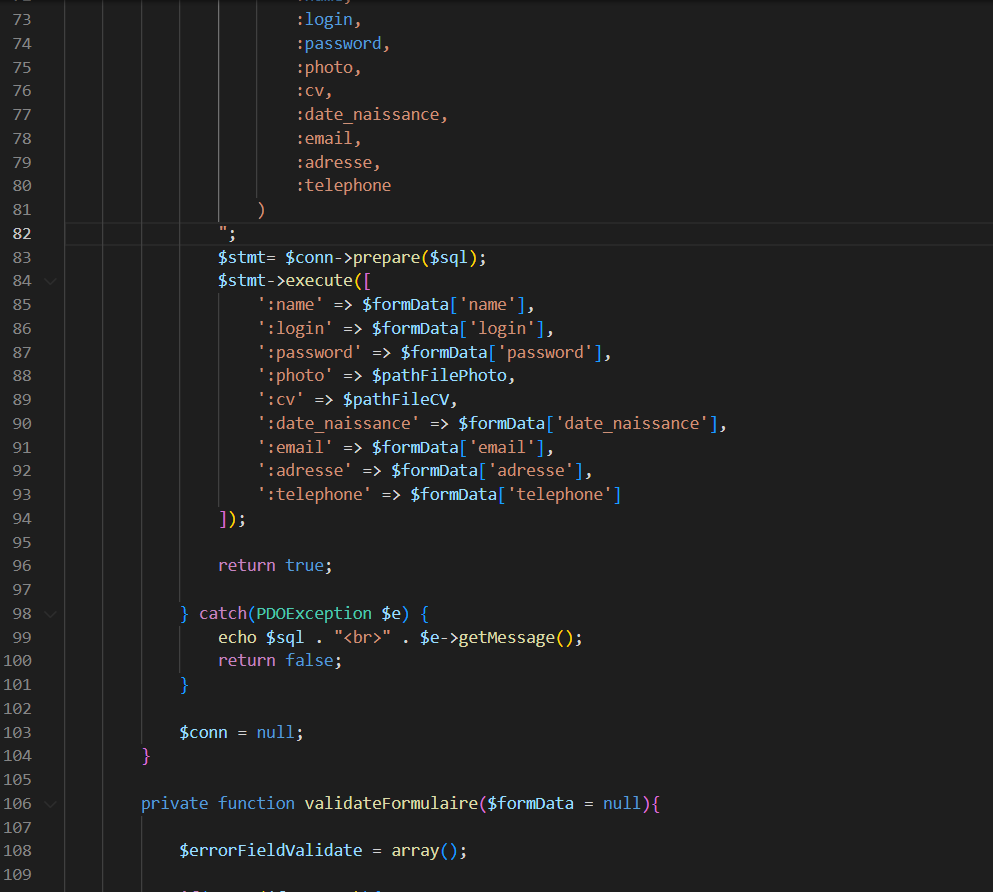
→ au cas où , il n’ y a pas d’erreurs, on modifie les données par la méthode modifyEtudiant.

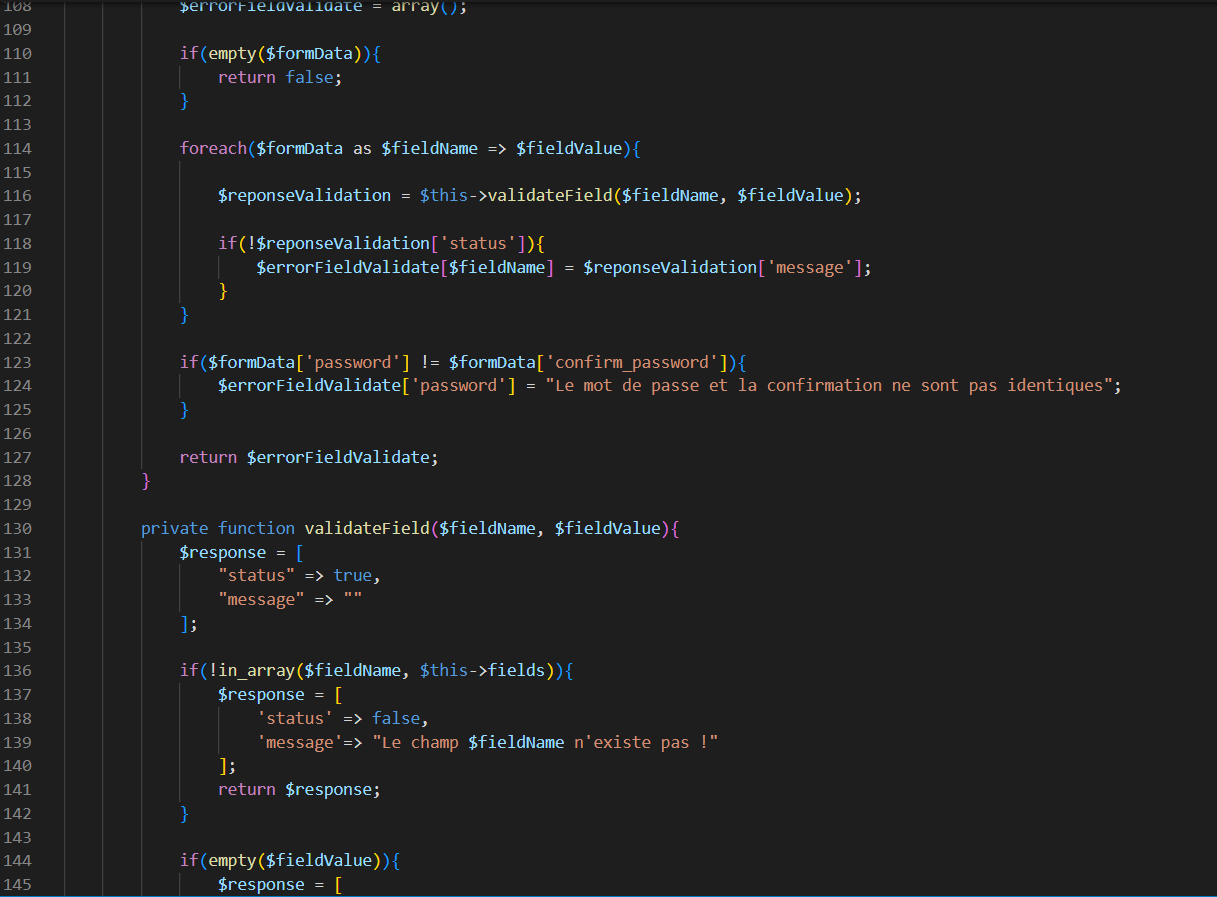
* La public Function -modifyEtudiant- , sert à modifier les données dans la base de données

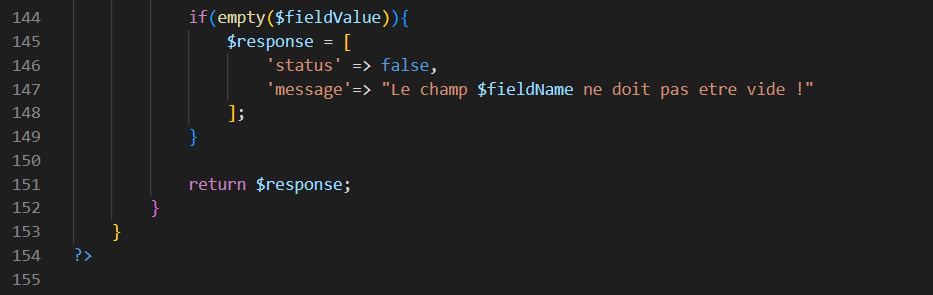
→ en utilisant la requête UPDATE.

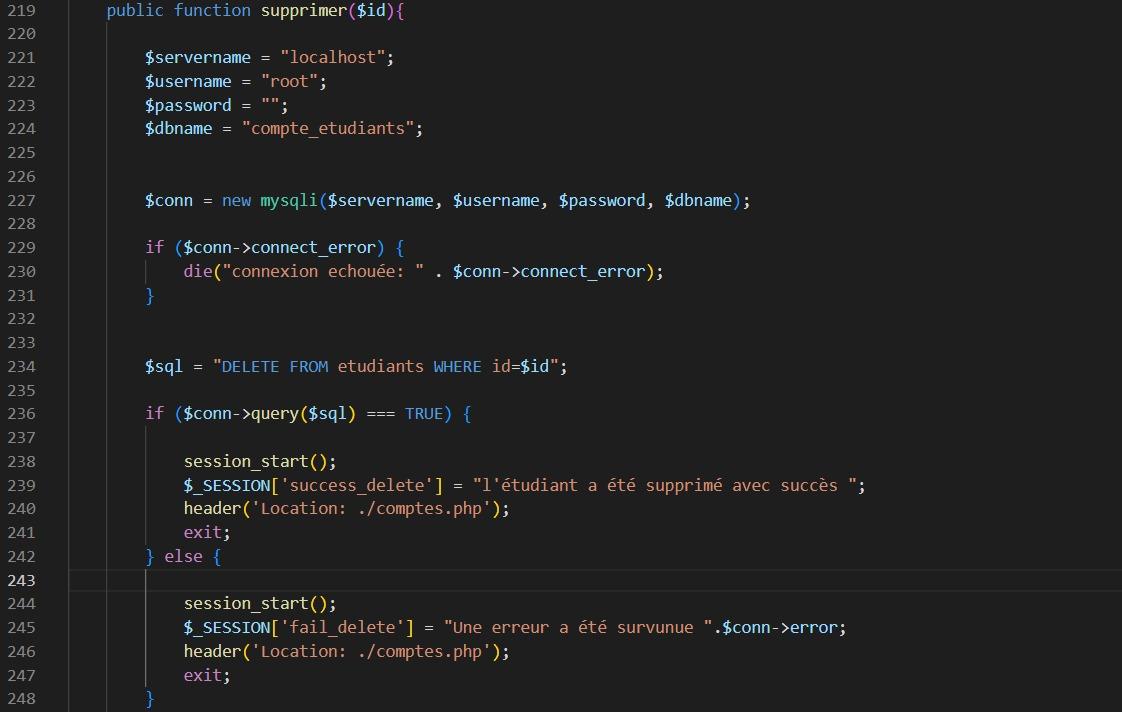


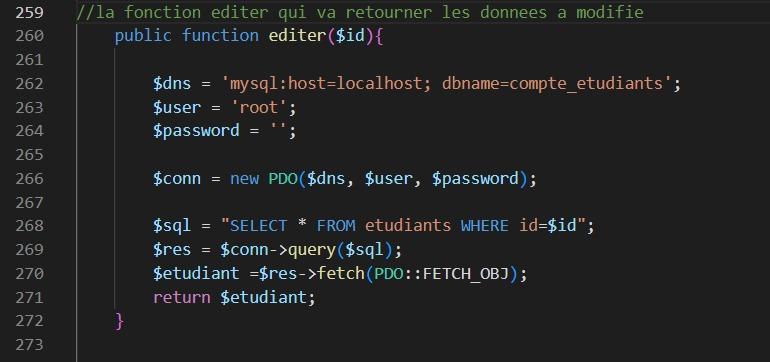
















3-4 Page “ enregistrer\_compte.php ’’:

* Cette page est conçue pour l’instanciation et l’appel des méthodes de la classe Etudiant , elle sert aussi à l’envoi des données du formulaire.

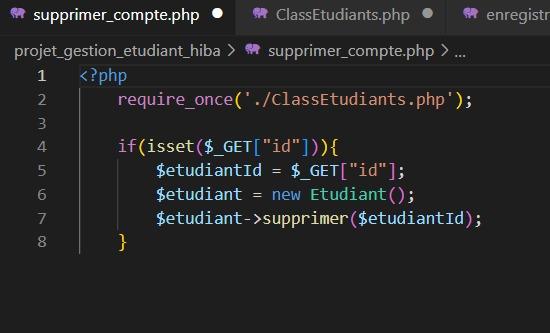
→ require\_once: sera utilisée pour ajouter le fichier contenant des fonctions ainsi qu’elle vérifie que si le fichier a déjà été ajouté une seule fois.

→ Grâce au test , on peut stocker les données de l’étudiant dans la base de données.



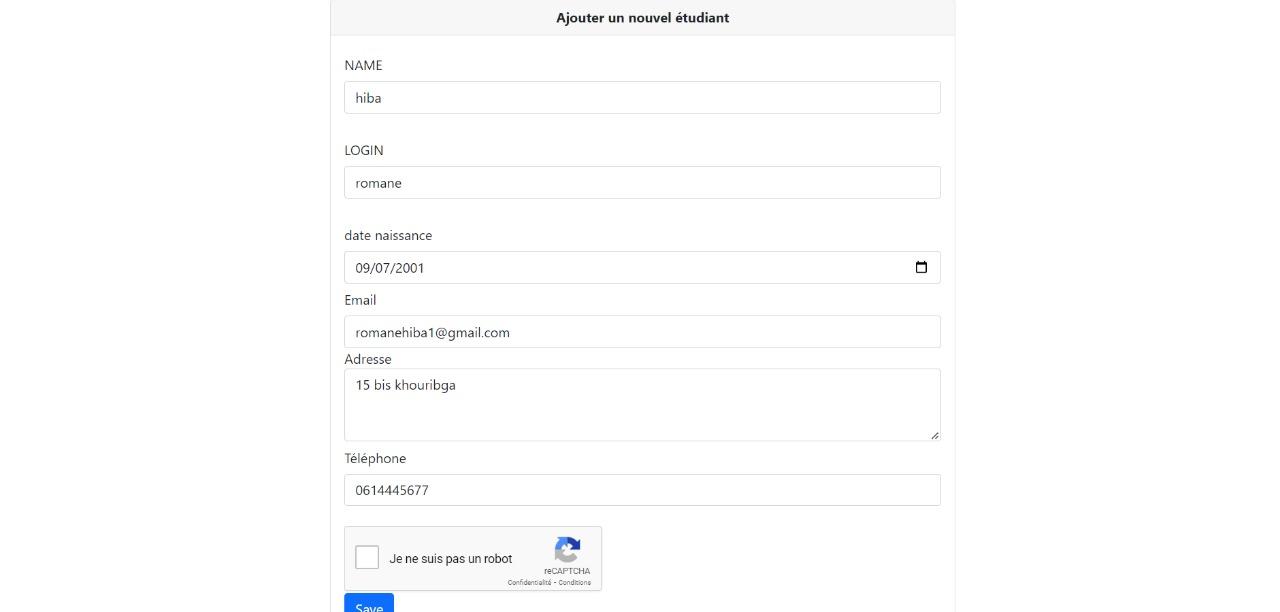
3-5 Page “ supprimer\_compte.php ’’:

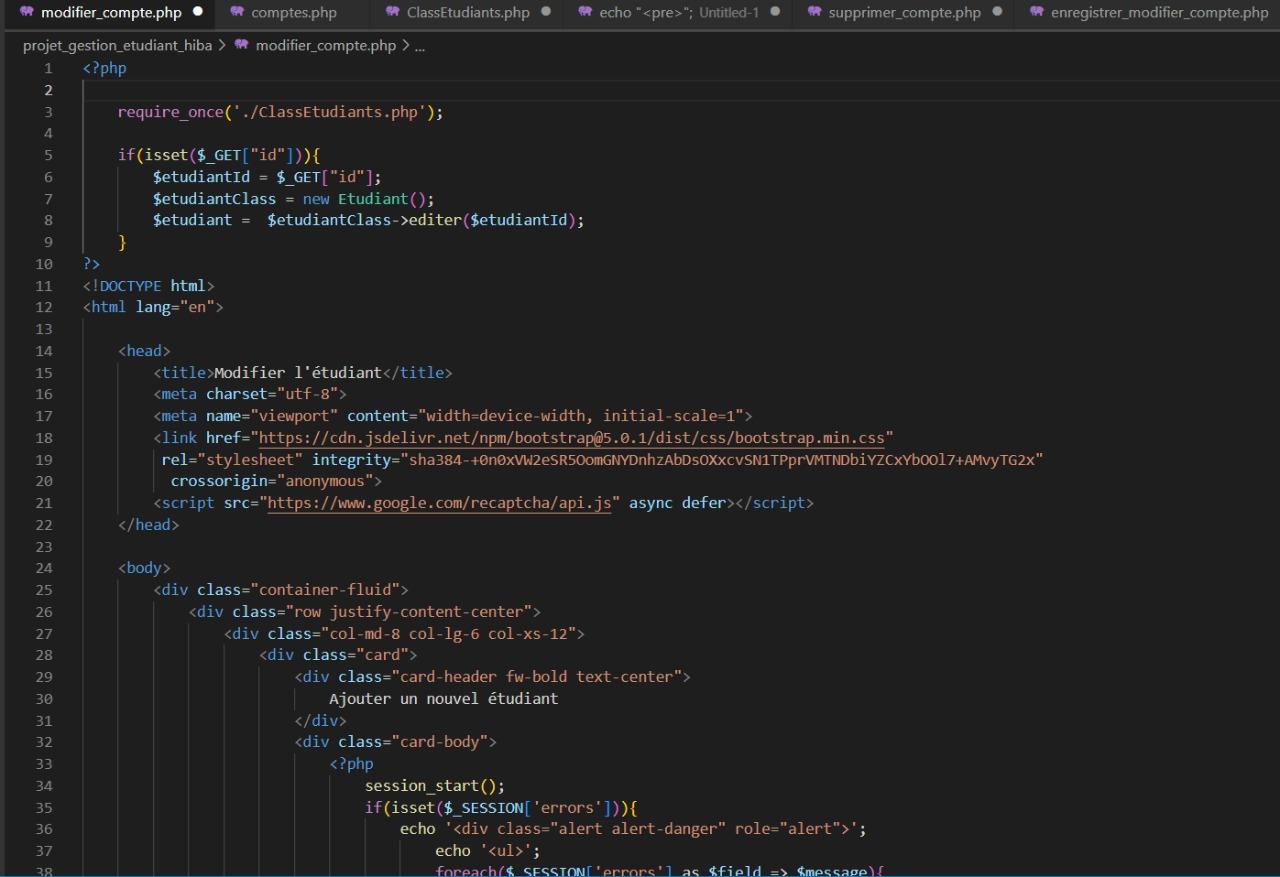
* Cette page est dédiée à la suppression des données envoyées par le formulaire à l'aide de l’id envoyé comme paramètre dans l’url en utilisant la méthode GET.

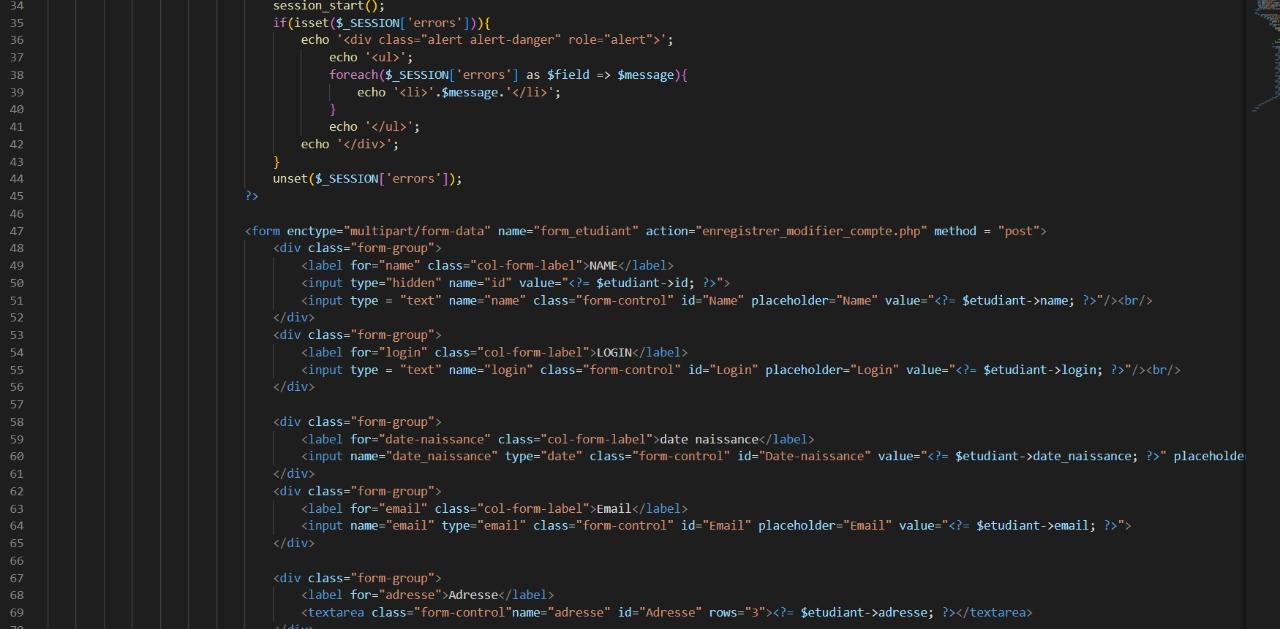


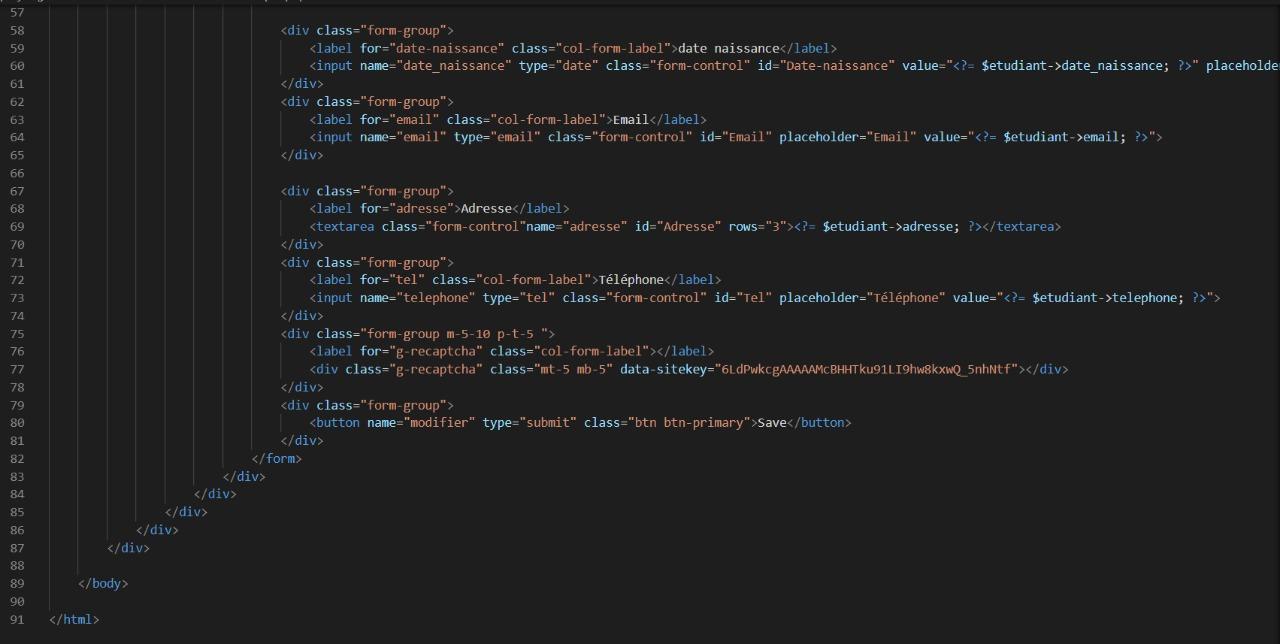
3-6 Page “ modifier\_compte.php ’’:

* Ce fichier permet de récupérer les données à modifier dans une variable étudiant , il prend chaque attribut de la variable (objet) il les dépose dans la valeur de chaque champ, tout en affichant les données initiales de la manière suivante :



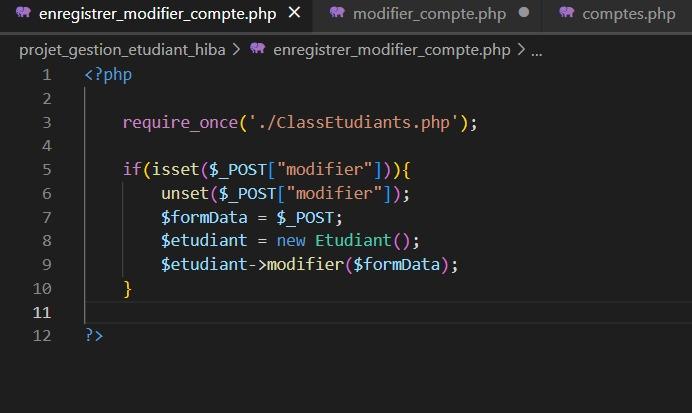




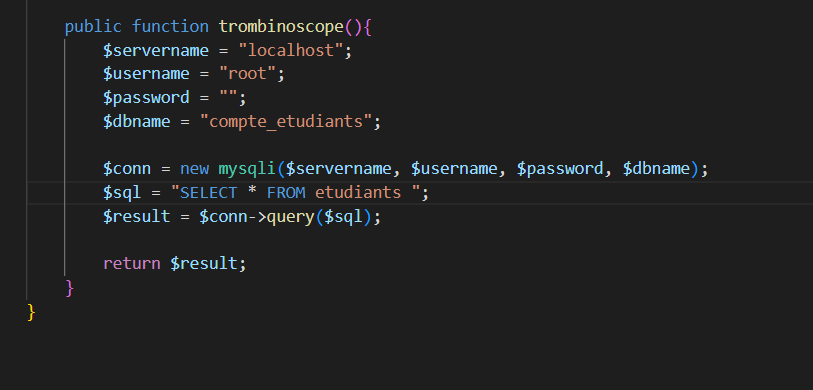


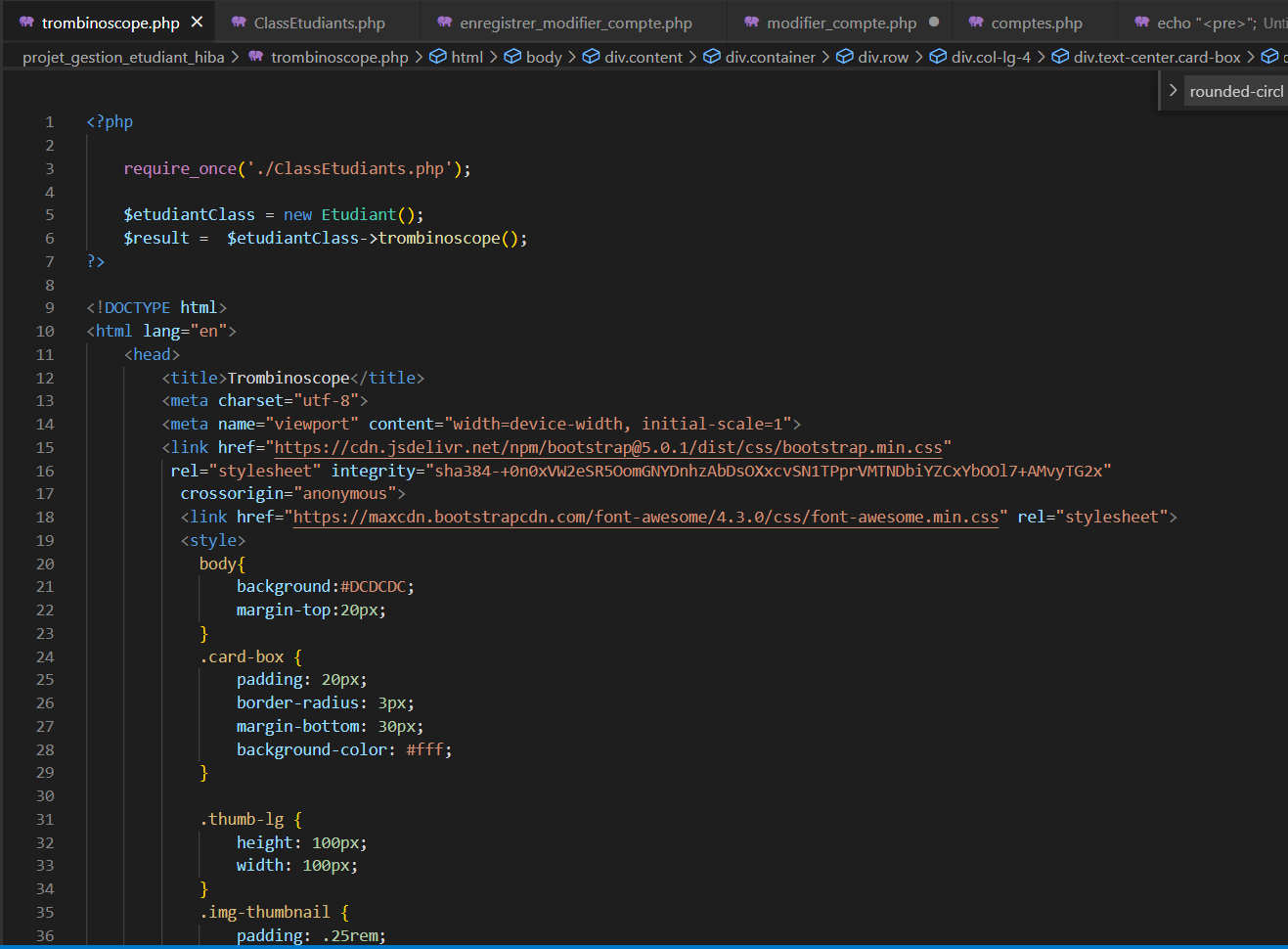
3-7 Page “ enregistrer\_modifier\_compte.php ’’:

* A l’aide la méthode modifier, on enregistre les données du compte modifié:



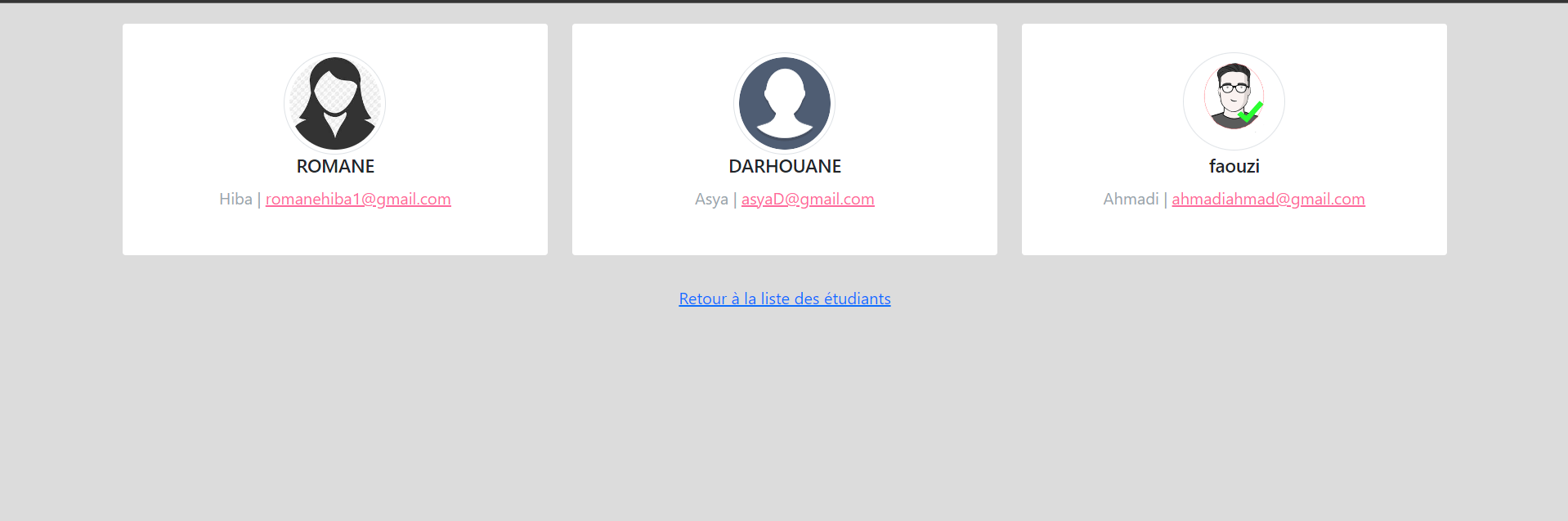
3-8 Page “trombinoscope.php ’’:





Une image contenant texte

Description générée automatiquement

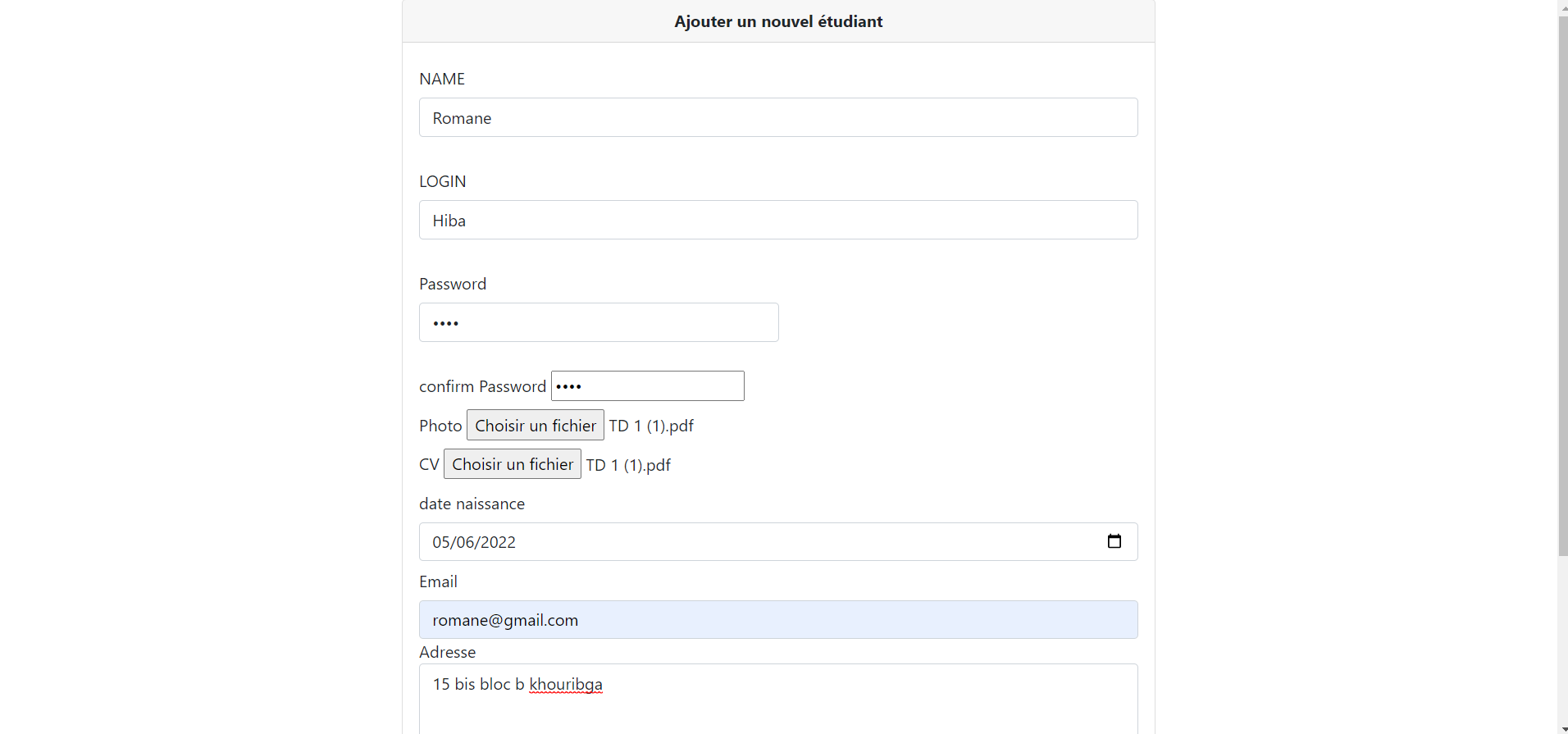
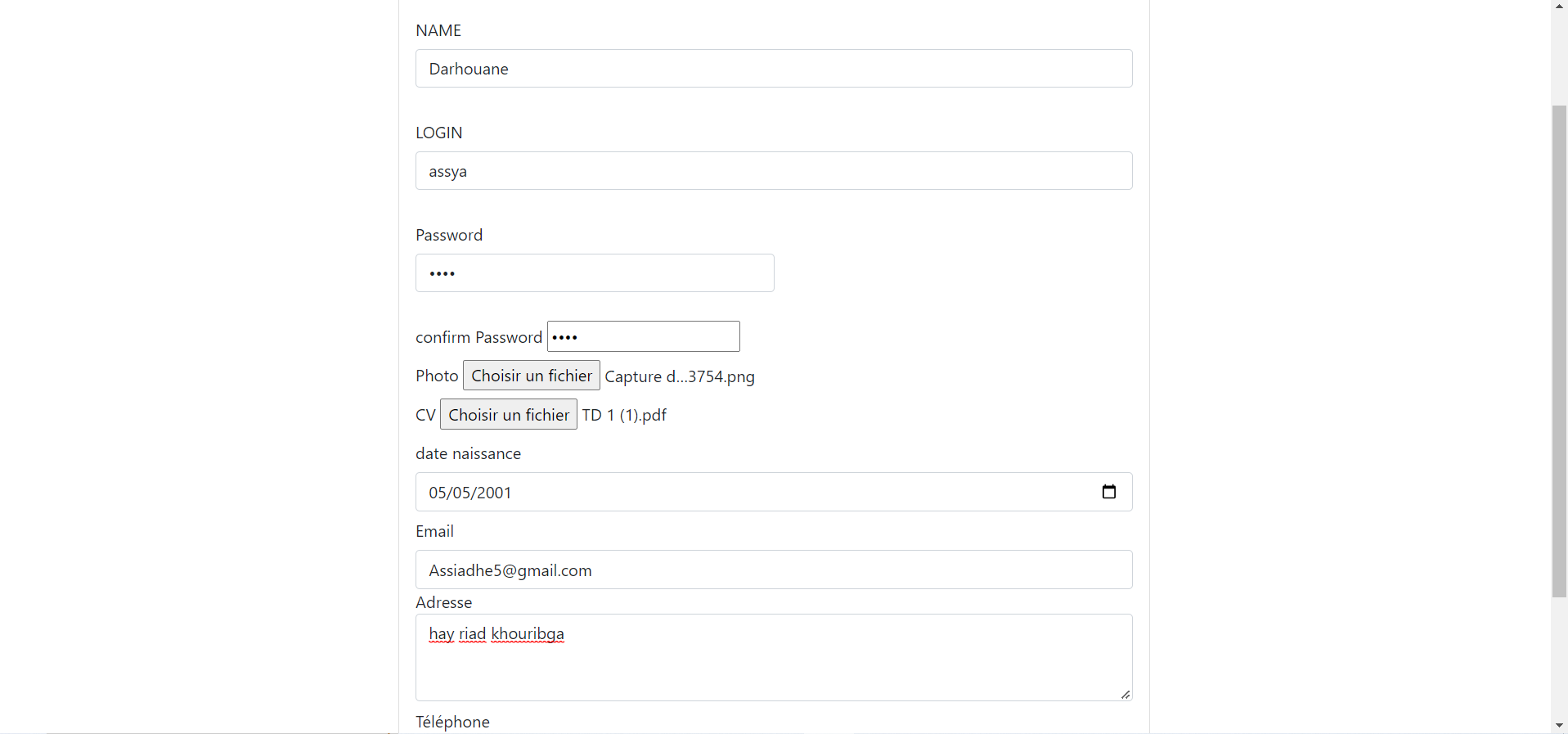


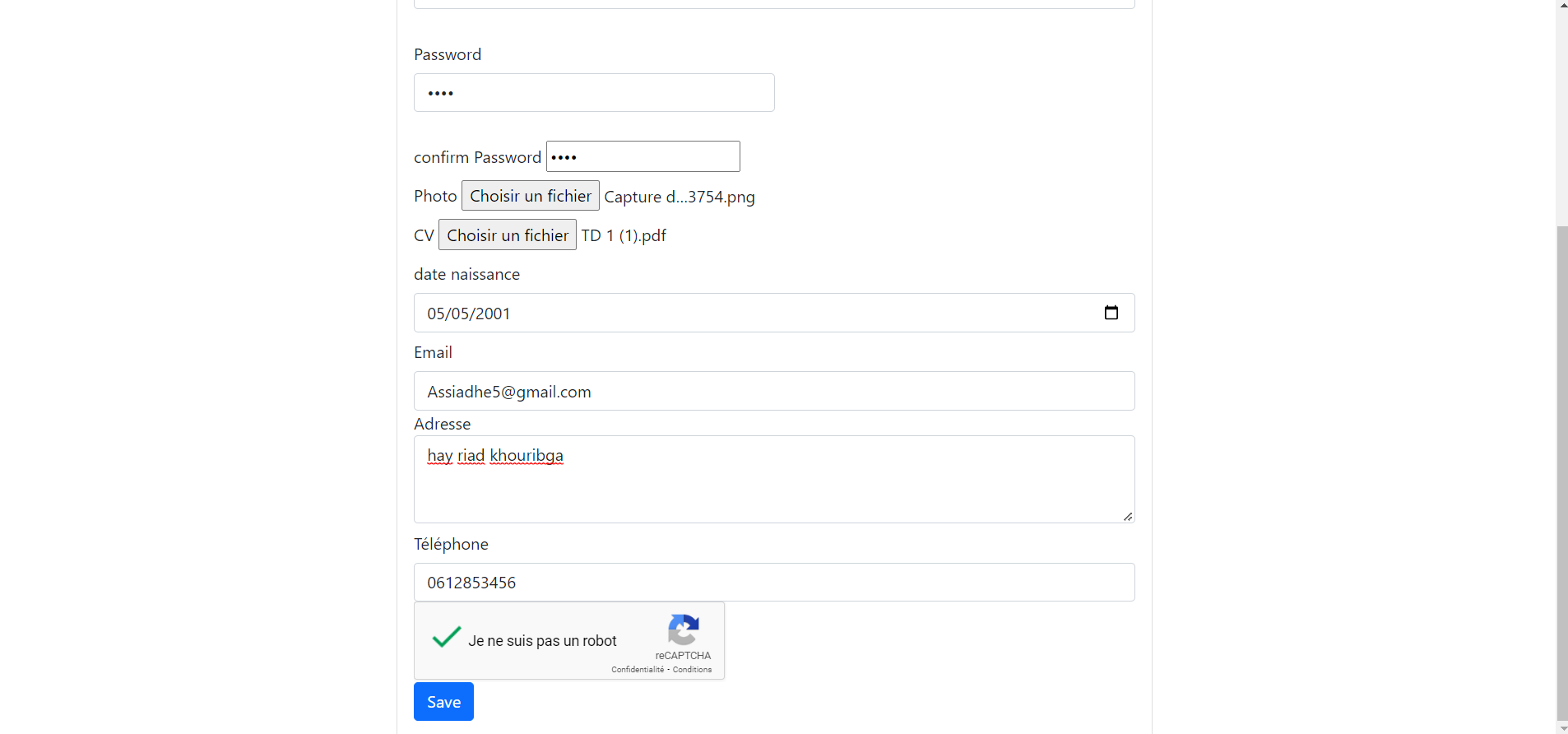
Une image contenant texte

Description générée automatiquement

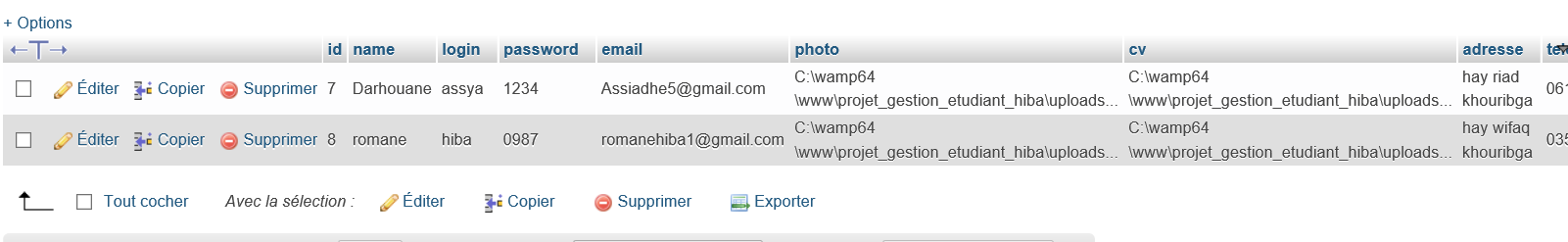
4-Affichage d'exécution :

→ remplissage du formulaire:

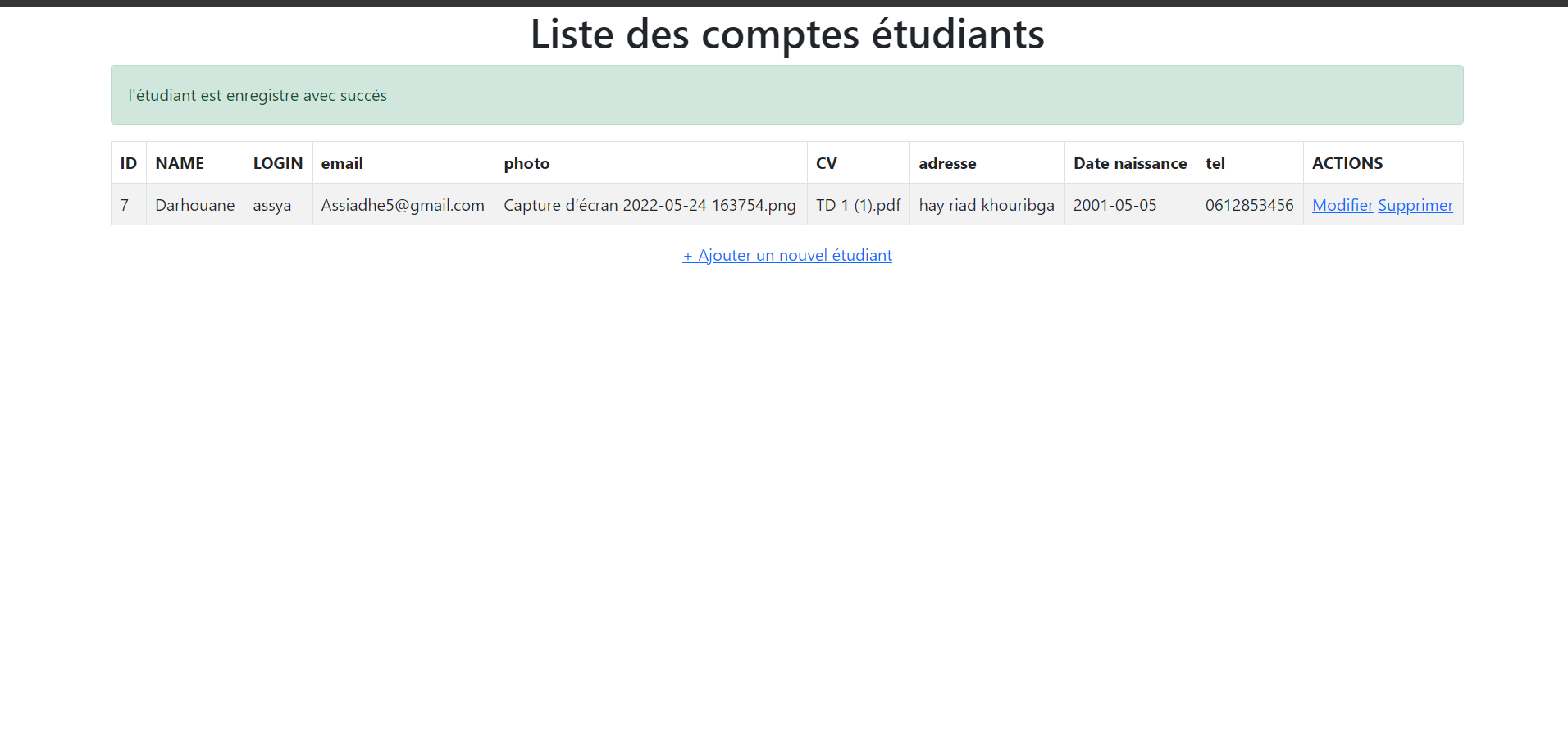




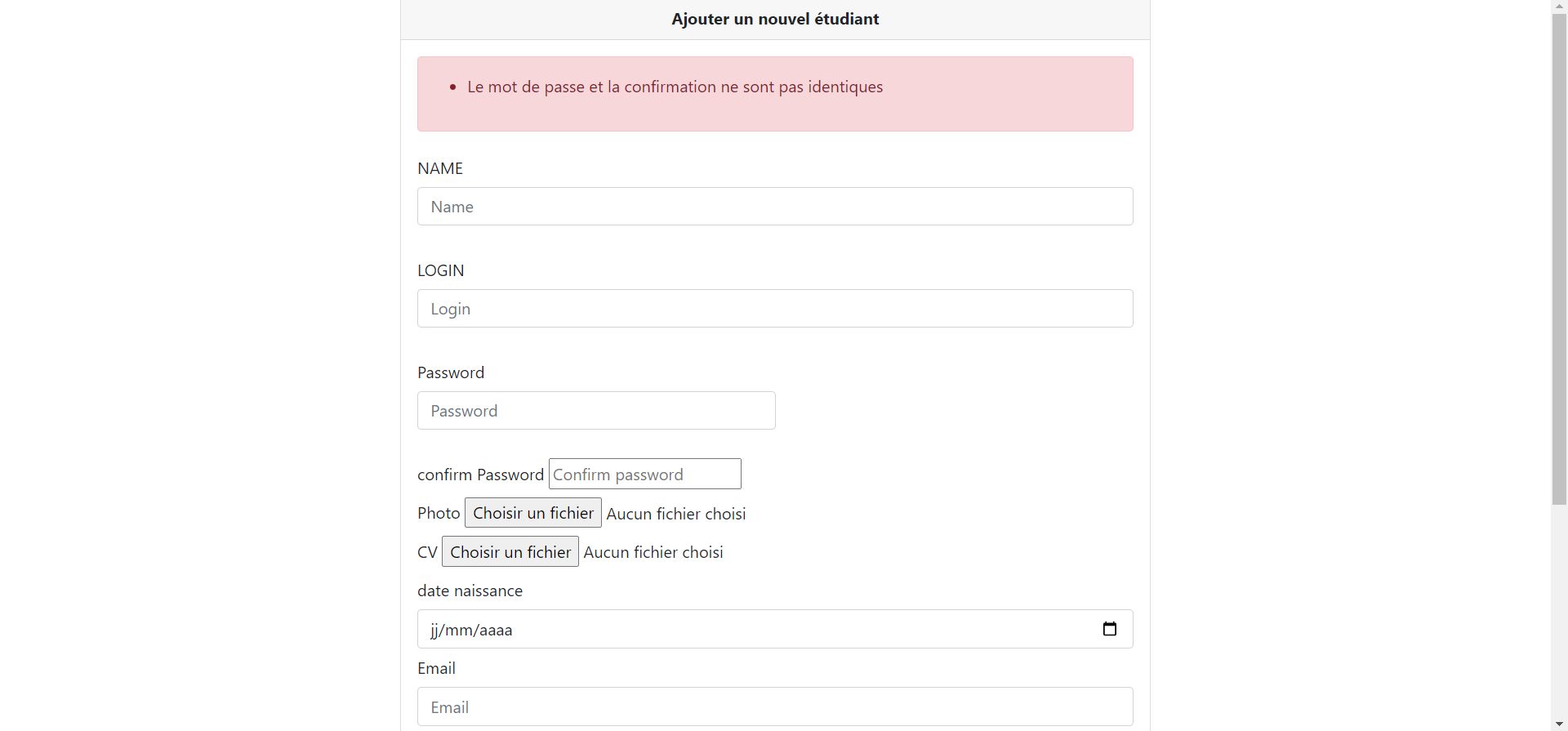
→ Vérification sur la base données:



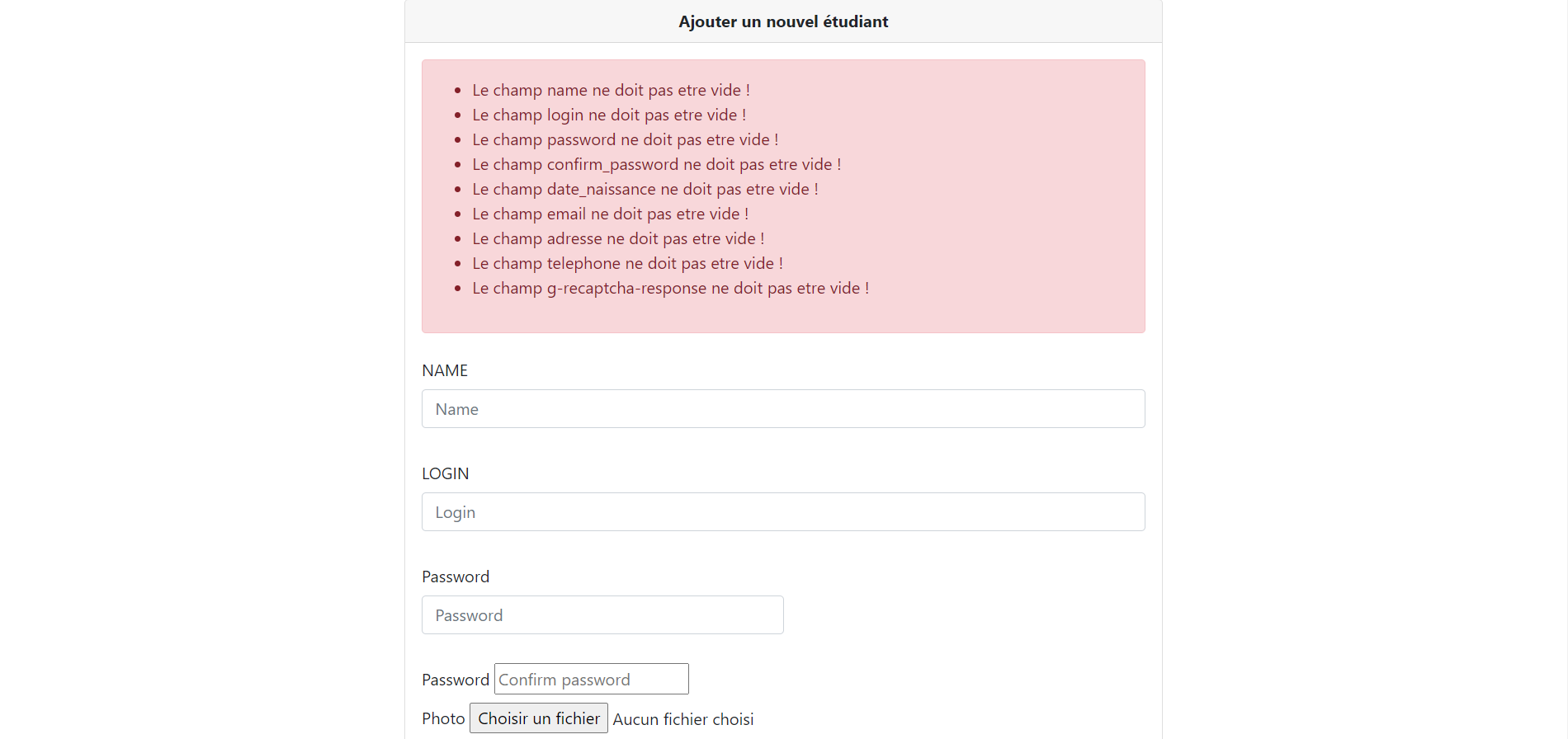
→ Alerte de message:



→ Mot de passe différents:



→ Avertissement du champ vide:



→ Après modification d’un compte :



Conclusion

**Bien que l’on ait rencontré des difficultés. Notre avis général sur ce projet est très positif car il nous a permis de réaliser de gros progrès en développement Web. Il nous a permis d’exploiter pleinement les capacités PHP.**