



Fiche de production

Situation E4

Mise en place d'un système d'inscription et de connexion  
d'un utilisateur

Dans le cadre de la réalisation d'un site de vente en ligne, dans notre cas, le projet Dendō jitensha, il est indispensable de mettre en place un système permettant l'inscription d'un utilisateur et sa connexion après inscription (voir fiche de description de la situation).

Cette fiche de production a pour objectif de présenter le procédé permettant la mise en place de cette fonctionnalité au sein de l'application Web développée.

Pour permettre une présentation complète de la situation, plusieurs points seront abordés :

- présentation de manière globale de la situation de projet ;
- présentation fonctionnelle de la situation ;
- mise en place de tests permettant de démontrer le bon fonctionnement des modules.

Les différents sujets abordés seront revus de manière plus approfondie dans la table des matières présente sur la page suivante.

## Table des matières

<b>I. CONTEXTE .....</b>	<b>P.1-3</b>
1a. Description de la situation.....	p.1
1b. Explication des attentes du projet.....	p.1-2
1c. Définitions des besoins associés à la réalisation du projet .....	p.2-3
<b>II. PRESENTATION DE LA PARTIE FRONT END .....</b>	<b>P.3-7</b>
2a. Définition des éléments de la partie visuel du projet .....	p.3
2b. Espace d'inscription.....	p.4-5
2c. Espace de connexion .....	p.5-6
2d. Aperçu de l'espace personnel d'un utilisateur .....	p.6
2e. Espace de déconnexion.....	p.7
<b>III. PRESENTATION DE LA PARTIE BACK END .....</b>	<b>P.7-13</b>
3a. Présentation de l'architecture de la base de données.....	p.7-8
3b. Présentation du script de création de la base de données .....	p.8
3c. Description du système d'inscription d'un utilisateur .....	p.9-10
3d. Description du système de connexion d'un utilisateur .....	p.10-11
3e. Comment afficher les informations d'un utilisateur donné ? .....	p.11-12
3f. Système de déconnexion .....	p.13
<b>IV. TEST DES SYSTEMES MIS EN PLACE .....</b>	<b>P.14</b>
4a. Test du système d'inscription .....	p.14
4b. Test du système de connexion.....	p.15
4d. Test de la déconnexion .....	p.15
<b>CONCLUSION .....</b>	<b>P.16</b>
<b>PARTIE ANNEXE (RECAPTCHA) .....</b>	<b>P.17-1</b>

## **I- Contexte**

*Cette première partie a pour objectif de contextualiser la situation rencontrée dans le cadre la réalisation de ce projet E4.*

### **1a. Description de la situation**

Le société Dendō jitensha, acteur majeur sur le marché chinois de la vente de vélo électrique souhaite développer son activité, en intégrant le marché français de ce même secteur d'activité d'une part et en mettant en place des solutions numériques qui permettront d'accroître la visibilité de la société en attirant une potentielle nouvelle clientèle, en partie grâce au développement d'une application Web. Il s'agira plus précisément d'un site de vente en ligne (e-commerce), qui permettra la présentation des produits vendus par Dendō jitensha et bien sûr la possibilité d'achat par les utilisateurs. On retrouvera bien sûr d'autres fonctionnalités présentes dans la majeure partie des sites de e-commerce présent sur le Web.

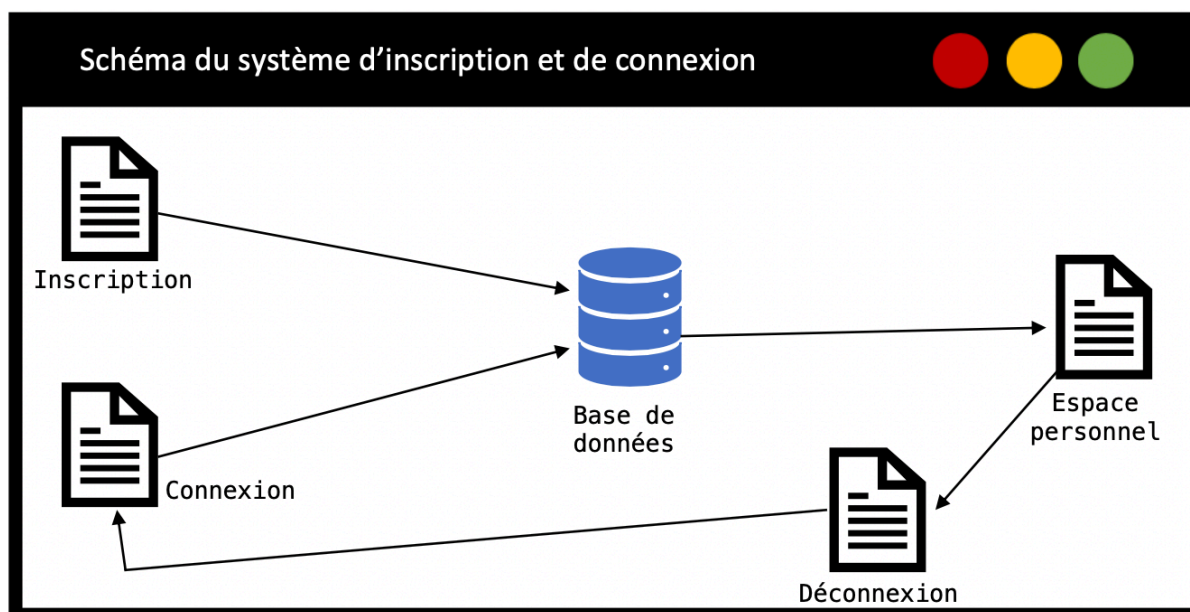
Dans un tel projet, il est indispensable de mettre en place un système permettant de « cloisonner » les espaces des utilisateurs les uns des autres. D'un point de vue sécuritaire, ce point paraît indispensable, car cela permettra de personnaliser les informations affichées en fonction de l'utilisateur qui utilise la solution, d'éviter qu'un utilisateur accède aux informations personnelles d'un autre, etc...

### **1b. Explication des attentes du projet**

Le système mis en place devra permettre :

- l'inscription d'un utilisateur, d'un point de vue fonctionnel, il s'agira de mettre en place un formulaire permettant à un utilisateur de créer un espace personnel en renseignant ses informations (nom, prénom, adresse email, mot de passe, etc...) et de stocker ces dernières de manière sécurisée au sein de la base de données ;
- à la suite de cette étape, l'utilisateur aura confirmation visuelle du bon déroulé de son inscription et donc de son ajout à la base de données ;
- dans le cas où un utilisateur est déjà inscrit, il devra avoir la possibilité de se connecter à son espace personnel par l'intermédiaire d'un formulaire de connexion. Les informations entrées permettront de vérifier la présence de l'utilisateur dans la base de données, de vérifier ses informations et de les afficher ;
- à la suite de sa connexion, un utilisateur sera en capacité de se déconnecter de son espace personnel, s'il le souhaite. Une page de confirmation de déconnexion sera déconnexion sera également mise en place.

Il est possible de visualiser ces différents points à l'aide de la figure ci-dessous :



*Il sera également possible d'implémenter une fonctionnalité permettant la suppression d'un compte utilisateur, mais elle ne sera pas présentée dans ce dossier.*

### 1c. Définitions des besoins associés à la réalisation du projet

De manière à permettre la mise en place d'un tel système, il sera nécessaire de mettre en place un environnement de développement sur lequel sera installés un langage serveur permettant la gestion de la partie back end (dans notre cas, le langage choisi et le langage PHP + utilisation du gestionnaire de librairie nommé Composer de manière à mettre en place un système d'autoloading performant). Devra également être présent un Système de Gestion de Base de Données, autrement appelé SGBD, de manière à mettre en place la base de données qui permettra le stockage des informations des utilisateurs et qui permettra également de manipuler les données stockées par le biais de requêtes SQL (qui devra donc également être présent sur l'environnement). Enfin, la présence d'un serveur Web (autrement appelé serveur http) sera indispensable dans le cadre de la réalisation des tests de la solution, dans notre cas, il s'agira d'un serveur Apache.

D'autres langages seront également utilisés dans la partie front end cette fois tel que le HTML5, le CSS3 et du JavaScript. Il s'agit ici de langages interprétés par les navigateurs Web (Mozilla Firefox, Google Chrome, Safari, etc...), il faudra donc avoir accès à au moins l'un d'entre eux de manière à pouvoir tester la solution dans son ensemble.

Il sera possible en fonction du système d'exploitation utilisé, d'installer un serveur LAMP, MAMP ou encore WAMP de manière à avoir accès à ces différents langages et outils.

## II- Présentation de la partie front end


Cette partie permettra de réaliser une présentation de la partie visuelle du projet par le biais de maquettes et ainsi présenter une interface graphique qui se rapprochera du résultat final.

## 2a. Définition des éléments de la partie visuelle du projet

Pour permettre la mise en place d'un tel projet, il faudra développer les pages suivantes :

- page d'inscription + validation d'inscription ;
- page de connexion, la validation de connexion se fera par l'arrivée d'un utilisateur sur son espace personnel ;
- page présentant l'espace personnel d'un utilisateur ;
- présence visuelle de la fonctionnalité permettant la déconnexion d'un utilisateur, la validation de déconnexion se fera par l'arrivée de l'utilisateur sur la page de connexion ou d'accueil.

## 2b. Espace d'inscription



**Création de compte :**

First name

Last name

Email

Password

Confirm your password

[Déjà inscrit ? Connecte toi !](#)

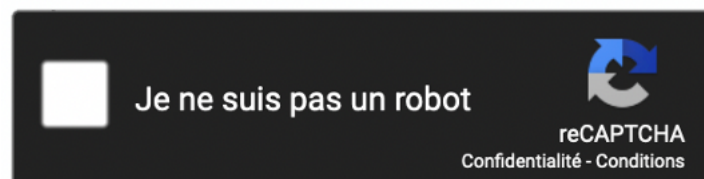
La figure ci-dessus présente la page d'inscription au site Dendō jiten sha.

Ce formulaire contient les éléments suivants :

- un champ de type texte pour récupérer le prénom de l'utilisateur (nommé « First name » sur la figure) ;
- un champ de type texte pour récupérer le nom de famille de l'utilisateur (nommé « Last name » sur la figure) ;
- un champ de type email de manière à pouvoir obtenir l'adresse email d'un utilisateur dans le bon format (nommé « Email » sur la figure) ;
- un champ de type password pour que l'utilisateur puisse définir son mot de passe (nommé « Password » sur la figure) ;
- un second champ de type password pour que l'utilisateur confirme son mot de passe et ainsi éviter les possibles fautes d'orthographe dans le mot de passe choisi (nommé « Confirm your password » sur la figure) ;
- une fois le formulaire d'inscription rempli, l'utilisateur pourra confirmer sa demande d'inscription en cliquant sur le bouton « Inscription » ;
- *si un utilisateur déjà inscrit se retrouve sur cette page d'inscription, il aura la possibilité de se diriger vers l'espace de connexion en cliquant sur le lien « Déjà inscrit ? Connecte-toi ! »*

Une fois l'étape d'inscription finalisée, l'utilisateur sera redirigé vers une page de confirmation d'inscription qui lui indiquera qu'il a bien été inscrit ou une fenêtre lui indiquant que la procédure d'inscription a échoué, le cas échéant.

En addition, il sera possible d'intégrer un espace supplémentaire à ce formulaire d'inscription permettant de vérifier si un utilisateur est bien un être humain et non un bot de manière à limiter les chances de création de faux profils sur le site. Une solution de type reCAPTCHA pourrait être intéressante :



Une section annexe sera disponible en fin de dossier pour vous présenter l'ajout de cette fonctionnalité, car elle n'est pas obligatoirement à implémenter dans le cadre de la mise en place d'espaces personnels.

## 2c. Espace de connexion



L'espace de connexion sera accessible depuis la page d'accueil du site, ou depuis le formulaire d'inscription. Cette espace ne sera pas réellement une page à part entière mais sera en réalité une fenêtre modale sous forme de pop-up. D'après le site de Wikipédia, une fenêtre modale est une fenêtre, dans une interface graphique, qui prend le contrôle total du clavier et de l'écran. Elle est en général associée à une question à laquelle il est impératif que l'utilisateur réponde avant de poursuivre, ou de modifier quoi que ce soit.

Donc, dès que l'utilisateur cliquera sur le bouton « Connexion », la fenêtre modale suivante s'affichera :

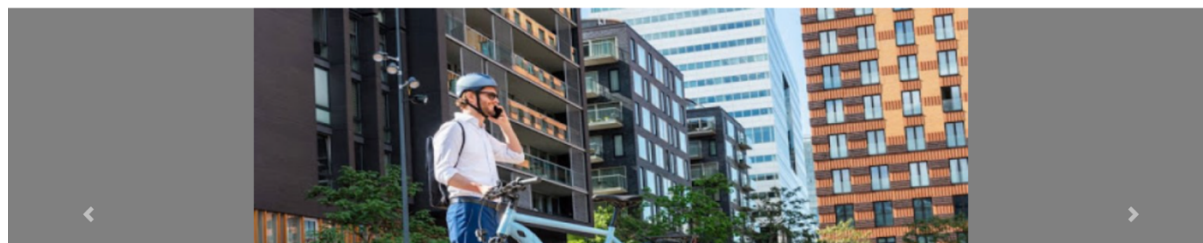
A white modal form with a subtle drop shadow. It contains two input fields: the first is labeled 'Mail:' and has the placeholder text 'Entrez votre email'; the second is labeled 'Mot de passe:' and has the placeholder text 'Entrez votre mot de passe'. Below these fields is a blue button with the text 'Connexion' in white.

Lorsque les champs auront été remplis par l'utilisateur et que ce dernier appuiera sur le bouton « Connexion », l'utilisateur sera redirigé sur la page d'accueil du site. Après connexion la page d'accueil se verra quelque peu modifiée. En effet, l'utilisateur aura donc accès à son espace personnel et le bouton de connexion disparaîtra pour être remplacé par le bouton de déconnexion.



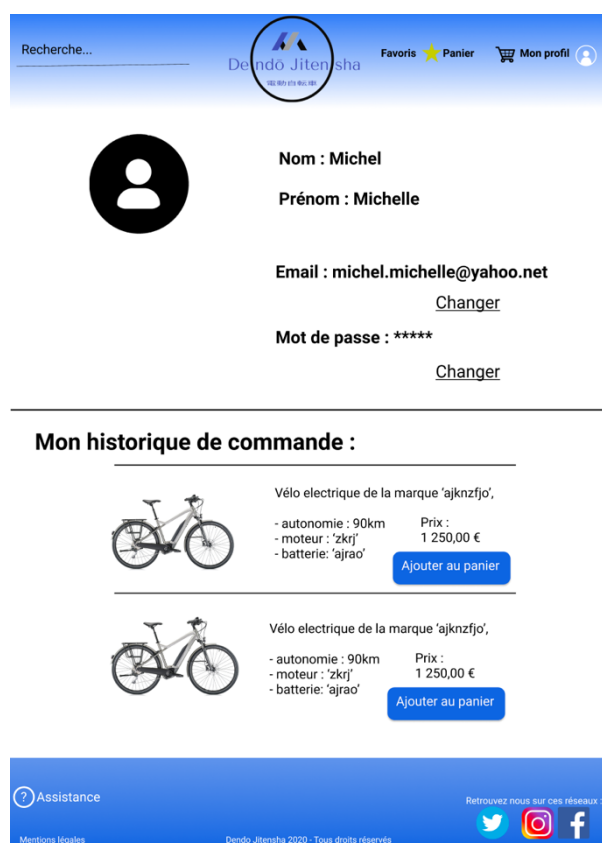


## Des vélos tous beau pour pas cher



En-tête de la page d'accueil du site

### 2d. Aperçu de l'espace personnel d'un utilisateur



L'espace utilisateur paraît être le meilleur moyen de tester le bon fonctionnement du système d'espaces personnels. En effet, c'est majoritairement sur cette page que les informations vont changer en fonction de l'utilisateur connecté. Sur la figure à gauche, on peut constater que le profil présente les informations suivantes :

- le nom ;
- le prénom ;
- l'adresse email ;
- le mot de passe ;
- l'historique de commande.

## 2e. Espace de déconnexion

La déconnexion d'un utilisateur se fera par l'intermédiaire d'un bouton de déconnexion, ce qui permettra la destruction des sessions et/ou cookies qui ont pu être enregistré(es) ainsi la redirection de l'utilisateur vers la page de connexion d'accueil du site (en mode non-connecté).

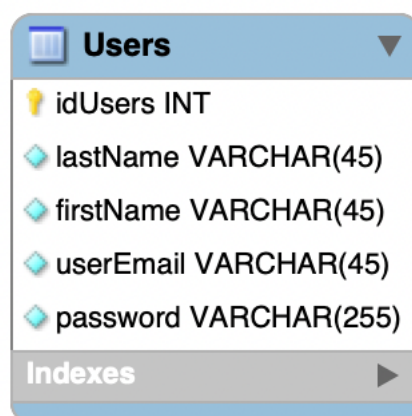
## III- Présentation de la partie back end

Dans cette partie du dossier, je vous propose de vous présenter la base de données qui sera mise en place pour permettre le bon fonctionnement des espaces utilisateurs interdépendants. Cette section permettra donc de présenter la partie base de données qui comprend donc l'architecture de cette dernière et le script de création de cette même base de données.

Ensuite, il sera important de lier le jeu de données créé avec l'interface utilisateur, par le biais de la mise en place des scripts PHP, qui permettront l'ajout de données pour les sauvegarder de manière persistante. Il sera également possible de manipuler les données stockées et ainsi permettre d'afficher des informations en fonction de l'utilisateur, d'effectuer des vérifications (connexion et inscription), ...

### 3a. Présentation de l'architecture de la base de données

Dans le cadre de cette situation, la mise en place de la base de données nommée « E4\_ESPACES\_PERSONNELS » sera nécessaire. Elle contiendra les informations suivantes :



La création d'une seule table permettra de mettre en place un système permettant la création et la gestion d'espaces personnels. Cette dernière contiendra les champs suivants :

- idUsers, qui permettra d'identifier un utilisateur en fonction d'un identifiant donné ;
- lastName et firstName, qui permettront de stocker le nom et le prénom des utilisateurs. Ces informations seront utilisées dans de nombreux modules du site tel que l'affichage des informations personnelles d'un utilisateur sur son espace, la mise en place des informations de paiement, de facturation, de livraison, ... ;

- userEmail, qui permettra le stockage de l'adresse email d'un utilisateur. Ce champ sera utilisé pour permettre l'identification d'un utilisateur lors de sa connexion. Il s'agira également d'une information qui permettra de savoir si un utilisateur est déjà inscrit sur le site ou non (au moment de son inscription) pour éviter la duplication de comptes ;
- password, qui stockera de manière sécurisée (chiffrement) le mot de passe que l'utilisateur aura choisi. Il s'agira de la seconde information (avec l'adresse email) qui sera demandée au moment de la connexion de l'utilisateur de manière à vérifier les informations de ce dernier.

La figure présentant la table Users sera bien entendu amenée à évoluer avec l'apparition d'autres tables au sein de la base de données, ce qui permettra par exemple de lier un compte utilisateur avec son panier électronique, son historique de commandes, etc...

### 3b. Présentation du script de création de la base de données

Pour permettre la mise en place de la base de données qui sera utilisée par le module consacré à la création et la gestion d'espaces utilisateurs, le script SQL suivant sera utilisé :

```
DROP DATABASE E4_ESPACES_PERSONNELS;  
CREATE DATABASE E4_ESPACES_PERSONNELS;  
USE E4_ESPACES_PERSONNELS;  
  
CREATE TABLE IF NOT EXISTS Users(  
    idUsers INT NOT NULL AUTO_INCREMENT,  
    lastName VARCHAR(45) NOT NULL,  
    firstName VARCHAR(45) NOT NULL,  
    userEmail VARCHAR(45) NOT NULL,  
    password VARCHAR(255) NOT NULL,  
    PRIMARY KEY (idUsers)  
);
```

### 3c. Description du système d'inscription d'un utilisateur

Pour permettre l'inscription d'un utilisateur, comme indiqué dans la partie précédente, ce dernier devra passer par un formulaire d'inscription. Il lui sera donc nécessaire de renseigner un certain nombre d'informations, à savoir :

- un prénom ;
- un nom de famille ;
- une adresse email ;

- un mot de passe (un champ mot de passe de vérification doit être rempli pour éviter qu'un utilisateur fasse une faute de frappe et ne s'en rende pas compte).

Les champs constitutifs du formulaire d'inscription possèdent l'attribut `required` au sein du code HTML, ce qui signifie que les champs doivent obligatoirement être renseignés pour que le formulaire d'inscription puisse être envoyé puis traité.

Une fois le formulaire envoyé, le fichier **`registrationValidation.php`** se verra « remettre » l'ensemble des informations renseignées par l'utilisateur via des variables super-globales `$_POST` (voir fichier : [https://github.com/Dams-debug/E4\\_espaces\\_personnels/blob/main/Inscription/php/registrationValidation.php](https://github.com/Dams-debug/E4_espaces_personnels/blob/main/Inscription/php/registrationValidation.php)).

Ces données vont subir un certain nombre de traitement. Dans un premier, les informations seront passées en arguments à la fonction `htmlspecialchars()` qui permettra d'empêcher les potentiels scripts envoyés par les utilisateurs par le biais des champs du formulaire de s'exécuter.

Ensuite les 2 mots de passes transmis sont comparés de manière à réduire la possibilité que l'utilisateur se soit trompé au moment de la saisie de son mot de passe (fautes de frappe). Si les deux mots de passe sont identiques, le script se poursuit, sinon un message indiquant à l'utilisateur que les 2 mots de passes sont différents apparaît. Dans le cas où les deux mots de passes sont bien identiques, la connexion vers la base de données est réalisée, en créant une instance de la classe PDO, à laquelle sont passées les différentes informations de connexion. Juste après une instance de la classe `DatabaseManager` (voir fichier : [https://github.com/Dams-debug/E4\\_espaces\\_personnels/blob/main/Inscription/php/class/DatabaseHandling/DatabaseManager.php](https://github.com/Dams-debug/E4_espaces_personnels/blob/main/Inscription/php/class/DatabaseHandling/DatabaseManager.php)) est créée. Cette dernière permettra de réaliser les manipulations auprès de la base de données.

Ensuite, une nouvelle instance est créée, il s'agit ici d'une instance de la classe `RegistrationInformations` (voir fichier : [https://github.com/Dams-debug/E4\\_espaces\\_personnels/blob/main/Inscription/php/class/Informations/RegistrationInformations.php](https://github.com/Dams-debug/E4_espaces_personnels/blob/main/Inscription/php/class/Informations/RegistrationInformations.php)), qui prendra en arguments une liste de données (nom, prénom, email et mot de passe). Cette classe permettra le stockage des informations de l'utilisateur durant le reste du processus.

À ce stade, l'ensemble des informations transmises par l'utilisateur sont valides. La prochaine étape consiste à vérifier qu'un utilisateur ne possède pas déjà un compte, utilisant l'adresse email renseignée, pour éviter les duplications de comptes et de ce fait, éviter que plusieurs puissent avoir la même adresse email (sachant qu'une adresse mail est unique). C'est à ce moment-là que la méthode `checksBeforeRegistration()` de l'instance de la classe `DatabaseManager` est utilisée. Cette méthode prend un paramètre, l'adresse email du potentiel nouvel utilisateur. Cette méthode effectuera une requête sur la base de données en utilisant l'adresse email passée en argument, au moment de l'appel de la méthode, de manière à récupérer les informations d'une entrée, sur la table `Users`, uniquement si son champ `userEmail` contient l'adresse email fournie. Si le résultat de la requête est null, alors l'utilisateur ne possède pas de compte avec cette adresse email. Au contraire, si le résultat de la requête SQL contient des informations, c'est qu'un compte utilisateur, avec cette adresse email, est déjà présent. Dans ce cas, un message indiquant à l'utilisateur qu'il possède déjà un compte, apparaît à l'écran. S'il ne possède pas encore de compte, la méthode `addUser` de

l'instance de la classe DatabaseManager est appelée et les informations de l'utilisateur sont passées en arguments de cette méthode. Le code présent dans cette dernière méthode permettra l'ajout de l'utilisateur et de ses informations, à la base de données avec chiffrement du mot de passe.

### **L'espace utilisateur est créé.**

#### **3d. Description du système de connexion d'un utilisateur**

Une fois l'espace utilisateur créé, il est nécessaire de mettre en place un système permettant au nouvel utilisateur de pouvoir se connecter à son espace. Pour se faire, la mise en place d'un formulaire de connexion est indispensable. Pour pouvoir être exploité, le formulaire de connexion doit contenir deux informations :

- l'adresse email que l'utilisateur a utilisée au moment de la création de son espace personnel ;
- le mot de passe qu'il a choisi.

Les deux champs sont à remplir obligatoirement, tout comme les champs du formulaire d'inscription.

Une fois le formulaire de connexion rempli et envoyé, le fichier connectionValidation.php (voir fichier : [https://github.com/Dams-debug/E4\\_espaces\\_personnels/blob/main/Connexion/php/connectionValidation.php](https://github.com/Dams-debug/E4_espaces_personnels/blob/main/Connexion/php/connectionValidation.php)) sera chargé de réaliser les différents traitements avant de permettre la connexion d'un utilisateur. Tout comme dans la précédente partie, les informations renseignées par l'utilisateur sont rendues « inoffensives » (never trust user input).

Ensuite, la connexion vers la base de données est réalisée, en créant une instance de la classe PDO, à laquelle sont passées les différentes informations de connexion. Juste après une instance de la classe DatabaseManager (voir fichier : [https://github.com/Dams-debug/E4\\_espaces\\_personnels/tree/main/Connexion/php/class/DatabaseHandling](https://github.com/Dams-debug/E4_espaces_personnels/tree/main/Connexion/php/class/DatabaseHandling)) est créée. Cette dernière permettra de réaliser les manipulations auprès de la base de données. Ensuite, une nouvelle instance est créée, il s'agit ici d'une instance de la classe LoginInformations (voir fichier : [https://github.com/Dams-debug/E4\\_espaces\\_personnels/blob/main/Connexion/php/class/Informations/LoginInformations.php](https://github.com/Dams-debug/E4_espaces_personnels/blob/main/Connexion/php/class/Informations/LoginInformations.php)), qui prendra en arguments une liste de données (email et mot de passe). Cette classe permettra le stockage des informations de l'utilisateur durant le reste du processus.

La première étape consiste à vérifier si l'utilisateur qui souhaite se connecter, possède bien un compte. Pour se faire la méthode checksBeforeConnection de la classe DatabaseManager est appelée et l'adresse mail de l'utilisateur qui souhaite se connecter, est passée en argument. Cette méthode va simplement vérifier si une entrée de la tables Users possède bien un champ userEmail avec l'adresse passée. Si une entrée est trouvée, l'utilisateur possède bien un compte. Sinon, ce dernier verra apparaître à l'écran, un message indiquant qu'il ne possède pas d'espace personnel qui utilise cette adresse email. Dans le cas où le compte existe bien, les informations, du compte existant, sont récupérées et le mot de passe

renseigné est comparé avec le mot de passe stocké en base de données. S'ils sont identiques l'utilisateur se verra connecté à son espace privé, il verra apparaître le message « Le mot de passe est incorrect ! », le cas échéant.

### **L'utilisateur est connecté !**

#### 3e. Comment afficher les informations d'un utilisateur donné ?

L'une des raisons principales de la mise en place d'espaces personnels, est de pouvoir afficher des informations personnalisées en fonction de l'utilisateur qui possède le compte.

C'est pourquoi, une fois connecté, les informations affichées à l'écran seront adaptées à l'utilisateur, grâce aux informations stockées en session. Le choix de l'utilisation des sessions plutôt que des cookies par exemple se justifie par le fait que les sessions sont supprimées à chaque fois que le navigateur Web utilisé est fermé par l'utilisateur, ce qui, en cas d'oubli de déconnexion de la part de ce dernier, permettra une fermeture de la session en cours et limitera le risque qu'une personne non autorisée puisse accéder aux informations personnelles de l'utilisateur.

#### **La mise en œuvre du système de session se fait de la manière suivante :**

Création d'une session :

```
session_start();
```

Cette ligne doit impérativement être la première ligne du fichier (directement après l'ouverture de la balise <?php. Elle devra apparaître non seulement au moment de la création de la session mais également sur l'ensemble des fichiers qui auront la nécessité d'accéder aux informations stockées en session.

Ensuite, les informations souhaitées sont stockées en session :

```
$_SESSION['lastName'] = $res['lastName'];  
$_SESSION['firstName'] = $res['firstName'];  
$_SESSION['userEmail'] = $res['userEmail'];  
  
header("location:../../Profil/profil.php");
```

La dernière permet de rediriger automatiquement l'utilisateur vers sa page de profil.

Sur le fichier qui compose le profil de l'utilisateur, apparaît en première, cette ligne :

```
session_start();
```

Elle donne ainsi accès aux informations stockées précédemment.

Enfin, l'affichage des informations peut se faire de la manière suivante :

```
<p>Nom : <?php echo $_SESSION['lastName'] ?> </p>  
<p>Prénom : <?php echo $_SESSION['firstName'] ?> </p>
```

```
<p>Email : <?php echo $_SESSION['userEmail'] ?> </p>
```

### 3f. Système de déconnexion

Si un utilisateur souhaite se déconnecter manuellement ou que le système d'utilisation de cookies est préféré, la mise en place d'un système de déconnexion est indispensable.

Pour mettre en place un tel système, un lien de déconnexion sera présent sur le profil de l'utilisateur. Ce lien entraînera l'exécution du code suivant :

```
<?php

/* Fichier permettant de supprimer la session en cours */

session_start();

session_destroy();
unset($_SESSION['lastName']);
unset($_SESSION['firstName']);
unset($_SESSION['userEmail']);

/* Redirection vers la page de connexion */

header("location:./connectionValidation.php");
```

Ce script entraînera la destruction de la session en cours et le « dé paramétrage » des paramètres de cette dernière. L'utilisateur sera ensuite redirigé vers la page de validation de connexion. Sur ce fichier, la condition en début de fichier renverra false :

```
if (isset($_POST['email']) && isset($_POST['password']))
```

En effet, les informations entrées par l'utilisateur au moment de sa connexion n'existent plus, le script considère donc qu'il vient de se déconnecter. L'utilisateur sera enfin redirigé vers la page de connexion.

Pour empêcher un utilisateur déconnecté de retourner sur sa page privé sans repasser par le formulaire de connexion, le code suivant a été mis en place au début du fichier représentant le profil de l'utilisateur :

```
if (!isset($_SESSION['userEmail'])) {  
    header("location:../Connexion/connection.html");  
}
```


Ce code indique que s'il n'existe pas d'informations possédant l'identifiant « userEmail » dans la session courante, alors l'utilisateur est redirigé vers la page de connexion.

#### **IV- Test des systèmes mis en place**

Dans cette partie il sera question de vous présenter le résultat de la mise en place des différents modules de manière à vous en donner un aperçu.

##### **4a. Test du système d'inscription**






**Création de compte :**

First name  Last name

Email

Password

Confirm your password

☒ Je ne suis pas un robot  reCAPTCHA  
Confidentialité - Conditions

Déjà inscrit ? Connecte toi !

Après avoir rempli le formulaire d'inscription (voir figure ci-dessus), l'utilisateur John Doe verra apparaître un message du type :


**Vous avez bien été ajouté à notre base de données !**

[> Retourner au formulaire d'inscription](#)

Ce message confirme son ajout à la base de données.

#### 4b. Test du système de connexion

Une fois inscrit le formulaire suivant permettra à l'utilisateur John Doe d'accéder à son espace personnel :



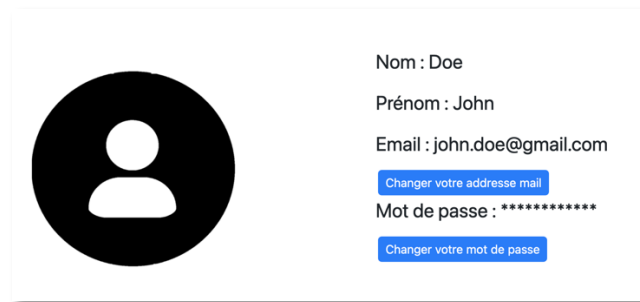
**Connexion**

Mail:

Mot de passe



Après envoi des informations du formulaire, l'utilisateur est bien redirigé vers son espace privé qui contiendra en partie, les informations présentées sur la figure suivante :



A user profile card with a white background and a thin grey border. On the left is a large black circle containing a white silhouette of a person. To the right of the circle, the following text is displayed: 'Nom : Doe', 'Prénom : John', 'Email : john.doe@gmail.com'. Below the email is a blue button with white text that says 'Changer votre adresse mail'. Below the button is the text 'Mot de passe : \*\*\*\*\*'. At the bottom is another blue button with white text that says 'Changer votre mot de passe'.

Nom : Doe
Prénom : John
Email : john.doe@gmail.com
<a href="#">Changer votre adresse mail</a>
Mot de passe : *****
<a href="#">Changer votre mot de passe</a>

#### 4d. Test de la déconnexion

Enfin, une fois que l'utilisateur souhaite se déconnecter manuellement, il lui suffit de cliquer sur le lien de déconnexion pour qu'il soit redirigé vers l'espace de connexion du site.

## **Conclusion**

En conclusion, on peut dire que la mise en place d'espaces personnels est impérative dans le cadre du développement d'un site de e-commerce. En effet, la segmentation des espaces utilisateurs permettra de sécuriser les informations personnelles de tous les utilisateurs du site de manière à permettre la visualisation des produits du site, leur ajout à un panier électronique, la configuration d'informations de facturation, ...

Pour permettre le déploiement d'un système de compte utilisateur dans ce type de projet, il est nécessaire de mettre en place un certain nombre de modules, tel que :

- un système d'inscription ;
- un système de connexion ;
- un système de déconnexion.

Le tout fonctionnant de manière homogène et permettant d'améliorer la satisfaction des utilisateurs.

## reCAPTCHA

Plus tôt dans ce dossier, nous avons abordé la possibilité de mettre en place un reCAPTCHA. Il s'agit d'un système qui permet de détecter, au moment de l'inscription, si un utilisateur est humain ou non (bot). Cette API mise en place par Google permettra donc de réduire le risque de voir apparaître de faux comptes sur le site.

Pour mettre en place ce type de dispositif, il est possible de suivre la documentation officielle (<https://developers.google.com/recaptcha/intro>).

La mise en place de ce système nécessitera l'ajout des balises suivantes, au fichier contenant le formulaire de connexion :

- dans le header :

```
<script src="https://www.google.com/recaptcha/api.js" async  
defer></script>
```

Cette ligne permet d'intégrer l'api de Google.

- dans le formulaire :

```
<div class="g-recaptcha" data-  
sitekey="6LdZCvUZAAAAAJke49Yn9N02MI0FXAgFqKQUzpN0"></div>
```

Cette ligne permet d'ajouter l'élément reCAPTCHA au formulaire. Noté la présence de la clé publique dans cette balise HTML, elle sera importante dans la suite de cette partie.

Lors de l'inscription d'un utilisateur, un test de ce type pourra être proposé (il n'est pas facultatif et doit être réussi) :



C'est par le biais des informations présentes dans la variable `$_POST["g-recaptcha-response"]` que le test de validation sera effectué (donc après envoi du formulaire).

Les informations contenues dans cette variable seront transmises à une instance de la classe ReCaptcha (voir figure suivante) :

```
<?php

namespace Registration;

class ReCaptcha {

    private $_secret;

    function __construct($secret) {

        $this->secret = $secret;

    }

    public function checkCode($code) {

        if(empty($code))
        {
            return false;
        }

        $url =
"https://www.google.com/recaptcha/api/siteverify?secret={$this
->secret}&response={$code}";

        if(function_exists('curl_version'))
        {
            $curl = curl_init($url);

            curl_setopt($curl, CURLOPT_HEADER, false);

            curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

            // Si le support Google n'est pas accessible, on
            abandonne la requête au bout d'1 seconde :
            curl_setopt($curl, CURLOPT_TIMEOUT, 1);
```

```
        $response = curl_exec($curl);
    }
    else
    {
        $response = file_get_contents($url);
    }

    if(empty($response) || is_null($response))
    {
        return false;
    }

    $json = json_decode($response);
    return $json->success;
}
}
```

C'est la méthode `checkCode` de cette instance qui permettra de réaliser le test.

Dans un premier temps, la correspondance des clés privée et publique doit être relevé. En effet, au début de la mise en place d'un reCAPTCHA, depuis la documentation officielle, un couple clé publique/clé privée sera généré et leur association est donc indispensable.

L'attribut privé `_secret` prend la valeur de la clé privée qui sera passée en argument au moment de l'instanciation de la classe. La méthode `checkCode` prendra en paramètre la valeur de `$_POST["g-recaptcha-response"]` qui correspond à un code généré après l'envoi du formulaire et donc réussite du test du reCAPTCHA.

Le tout est envoyé à une URL qui aura été créé et qui contiendra la clé privée et le code généré présent dans `$_POST["g-recaptcha-response"]`. L'URL mènera à un fichier JSON qui contiendra un certain nombre d'éléments dont un champ ayant pour identifiant « `success` » qui indique si le test est validé par l'API ou non. Il suffira de récupérer ces informations à l'aide de la fonction `curl` ou `file_get_content` pour pouvoir traiter le retour. L'élément « `success` » nous intéresse particulièrement, car il pourra être `true` ou `false` et définira donc si le test est valide ou non. Ce résultat sera utilisé dans la mise en place d'une condition supplémentaire dans le fichier de validation d'inscription (si le résultat est vrai on poursuit l'exécution du script, sinon on l'arrête).