Site de e-commerce basé sur l'architecture MVC.

**Explications globales**

Pour la base de données, il faut créer la base de données "db\_e\_shopping", puis lancer le fichier BDD/db\_e\_shopping.sql

Pour accéder à la bdd, il peut être nécessaire de modifier l'objet PDO (mot de passe, port) dans Modele/Modele.php et dans Vue/vueTunnel.php (La page Vue/vueTunnel.php n'a pas été intégrée à l'architectue MVC par Francis et Kevin).

Pour accéder aux pages admin il faut être logué en tant qu'utilisateur de niveau 1 (Vincent Reynaert ou Nicolas Sobczak). Tous les mots de passe sont "coucou".

Dans le fichier page\_listing.md, il y a un tableau détaillant les parties assignées à chacun. Chaque binôme a du faire une page, l'intégrer à l'architecture MVC, ajouter sa partie dans le css. Les noms des développeurs sont inscrits en haut de chaque page php mais pas dans le css. Ils ont développé la partie du css correspondant à leurs pages. Certains élèves ont voulu développer des pages en plus. Chacun a donc eu a faire du html, css, php, lecture de base de données et souvent de l'écriture aussi.

Une documentation a été créée dans le dossier "Documentation". Pour la visualiser, il faut lancer le fichier Documentation/html/index.html dans un navigateur.

**Explication de l'architecture MVC**

Dans l'architecture MVC, les rôles sont les suivants:

* modèle : données (accès et mise à jour)
* vue : interface utilisateur (entrées et sorties)
* contrôleur : gestion des événements et synchronisation



**Rôle du modèle**

Le modèle contient les données manipulées par le programme. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient.

Le modèle offre des méthodes pour mettre à jour ces données (insertion suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ses données. Dans le cas de données importantes, le modèle peut autoriser plusieurs vues partielles des données. Si par exemple le programme manipule une base de données pour les emplois du temps, le modèle peut avoir des méthodes pour avoir, tous les cours d'une salle, tous les cours d'une personnes ou tous les cours d'un groupe de Td.

**Rôle de la vue**

La vue fait l'interface avec l'utilisateur. Sa première tâche est d'afficher les données qu'elle a récupérées auprès du modèle. Sa seconde tâche est de recevoir tous les actions de l'utilisateur (clic de souris, sélection d'une entrées, boutons, …). Ses différents événements sont envoyés au contrôleur.

La vue peut aussi donner plusieurs vues, partielles ou non, des mêmes données. Par exemple, l'application de conversion de bases a un entier comme unique donnée. Ce même entier est affiché de multiples façons (en texte dans différentes bases, bit par bit avec des boutons à cocher, avec des curseurs). La vue peut aussi offrir la possibilité à l'utilisateur de changer de vue.

**Rôle du contrôleur**

Le contrôleur est chargé de la synchronisation du modèle et de la vue. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour que celle-ci se mette à jour. Certains événements de l'utilisateur ne concerne pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier.

Dans le cas d'une base de données des emplois du temps. Une action de l'utilisateur peut être l'entrée (saisie) d'un nouveau cours. Le contrôleur ajoute ce cours au modèle et demande sa prise en compte par la vue. Une action de l'utilisateur peut aussi être de sélectionner une nouvelle personne pour visualiser tous ses cours. Ceci me modifie pas la base des cours mais nécessite simplement que la vue s'adapte et offre à l'utilisateur une vision des cours de cette personne.

Le contrôleur est souvent scindé en plusieurs parties dont chacune reçoit les événements d'une partie des composants. En effet si un même objet reçoit les événements de tous les composants, il lui faut déterminer quelle est l'origine de chaque événement. Ce tri des événements peut s'avérer fastidieuse et peut conduire à un code pas très élégant (un énorme switch). C'est pour éviter ce problème que le contrôleur est réparti en plusieurs objets.