Intégration des données dynamiques dans le Framework LARAVEL

A cette étape du projet vous disposez d’un framework fonctionnel permettant de naviguer entre les pages / fonctionnalités de votre boutique

**Nous allons maintenant connecter votre boutique à la base de données afin de faire persister les données. 🎉🎉**

Cette intégration des données dynamiques se fera en **3 étapes**:

* Dans un premier temps il va falloir connecter votre base de données à votre projet LARAVEL
* Dans un second temps en ré-utilisant les **requêtes SQL** que vous avez déjà écrites
* Dans un troisième temps en mettant en oeuvre l’**ORM Eloquant** intégré à LARAVEL

Les étapes 5 à 8 sont à réaliser individuellement.

Les étapes 9 et 10 sont à réaliser sur le projet commun de l’îlot

# 

# 

# Projet étape 5: Lier une base de données à Laravel

### Modalités

* Travail en îlot, en autonomie
* Production individuelle

### Objectif (compétences)

* Accéder à votre base de données depuis LARAVEL

### Résultat attendu

* Avoir une connexion entre votre base de données et LARAVEL

### Consignes

* Etape 1 : Modifier votre fichier ***.env*** avec les identifiants de connexion à votre base de données
* Etape 2 : Modifier votre fichier ***/config/database.php*** en ajoutant les variables d’environnement définies dans le fichier ***.env*** auparavant.
* Etape 3 : Effectuer un ***php artisan migrate:refresh*** pour vérifier que la connexion à la base de données est fonctionnelle

### Ressources

* <https://laravel.com/docs/5.7/database#running-queries>

# Projet étape 6: Accès aux données & URL Dynamique

### Modalités

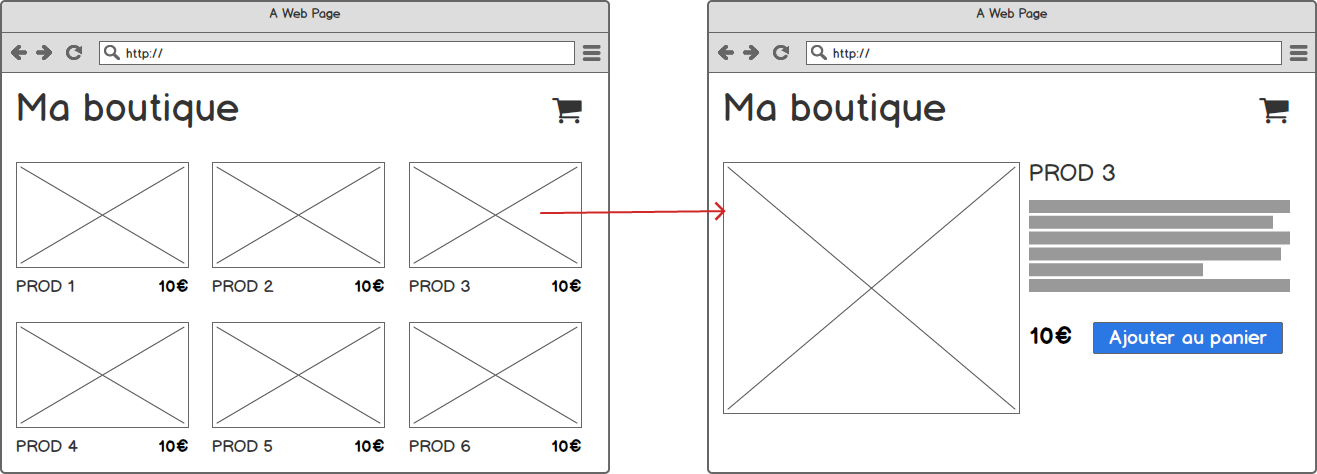
* Travail en îlot, en autonomie
* Production individuelle

### Objectif (compétences)

* Mettre en oeuvre des données dynamiques de votre boutique.

### Résultat attendu

* Lister l’ensemble des produits présents dans la base de données.
* Cliquer sur un produit de la liste pour afficher la fiche du produit



### Consignes

* Etape 1 : Créer un controller (si cela n’est pas déjà fait)
* Etape 2 : “connecter” vos Routes au Controller
* Etape 3 : “connecter” vos Views au Controller
* Etape 4 : Utiliser une **“Raw SQL Queries”** pour récupérer les données dans BDD puis les afficher dans vos Views

### Ressources

* <https://laravel.com/docs/5.7/database#running-queries>
* <https://laravel.com/docs/5.7/controllers>
* <https://laravel.com/docs/master/helpers#urls>
* BDD créée en module BDD

### Livrable

* Une page contenant la listes des produits de la BDD /!\ Commit GIT attendu
* Une page contenant le nom et prix d’un produit /!\ Commit GIT attendu

# Projet étape 7 : Accès aux données via “Model”

### Modalités

* Travail en îlot, en autonomie
* Production individuelle

### Objectif (compétences)

* Mettre en oeuvre un Model avec un **ORM** (**O**bject **R**elationnal **M**apping).

### Résultat attendu.

Nous souhaitons reprendre le travail précédent en utilisant cette fois un ORM :

* Lister l’ensemble des produits présents dans la base de données.
* Cliquer sur un produit de la liste pour afficher la fiche du produit

### Consignes

* Pour votre table MySQL contenant les produits, créez le Model correspondant.

**/!\ Uniquement la table des Produits pour le moment.**

### Ressources

* <https://www.grafikart.fr/formations/laravel/eloquent>
* <https://laracasts.com/series/laravel-from-scratch-2018/episodes/8>
* Documentation Laravel <https://laravel.com/docs/5.7/eloquent>
* BDD créée lors du module BDD
* 😉En cas de problème à la première migration : <https://laravel-news.com/laravel-5-4-key-too-long-error>

### Livrable

* Une page contenant la liste des produits triées par nom en utilisant l‘ORM
* Une page contenant la liste des produits triées par prix croissant en utilisant l‘ORM
* Une page contenant un produits en utilisant l‘ORM

/!\ Commit GIT attendu

|  |
| --- |
| Une fois la page d’accueil du réseau social intégré dans LARAVEL, un îlot réalisera un code-review Groupe.1476296081_file_search.png |

# Projet étape 8 : Mise à jour des données via Model

### Modalités

* Travail en îlot, en autonomie
* Production individuelle

### Objectif (compétences)

* Mettre à jour la base de données à travers la saisie dans un formulaire via un ORM.

### Specification client.

* Le client veut pouvoir ajouter un nouveau produit par la saisie dans un formulaire.
* Le client veut pouvoir mettre à jour un produit existant via un formulaire.
* Le client veut pouvoir supprimer un produit existant.

### Ressources

* Le schéma de transmission des requêtes du **Mini Projet “Formulaire Laravel”**
* Documentation Laravel :
  + <https://laravel.com/docs/5.7/requests>
  + <https://laravel.com/docs/5.7/controllers#resource-controllers>

### Livrable

* Description fonctionnelle **de la ressource “produit”** mis à jour (avec les méthodes HTTP GET, POST, PUT, DELETE)
* Pages d’ajout et de modification d’un produit (**nom et prix uniquement**)
* Données mise à jours dans la BDD

## 1476296081_file_search.png

|  |
| --- |
| L'îlot réalisera un code-review et mergera dans la branche principale du projet le code le plus adapté |

# Projet étape 9 : Mise en place des relations entre Models et intégration des données dynamiques à notre boutique

### Modalités

* Travail en îlot en autonomie.
* Répartition du travail

### Objectifs

* Mettre en place des relations entre les Models.
* Utiliser les Models dans le code de votre Boutique.

### Résultat attendu

* Nous souhaitons mettre en place des Models pour l’ensemble des données **de manière progressive**
* Nous souhaitons définir les relations entre les Models.
* Nous souhaitons avoir des données dynamiques sur l’ensemble des pages conformément au cahier des charges.

### Consignes

**Vous procéderez par étape** pour créer des Models pour chaque table MySQL.

Pour chaque nouveau Model, **vous testerez le bon fonctionnement** dans votre navigateur.

Pour chaque nouveau Model, **vous définirez dans celui-ci les relations** avec les autres modèles ( et vice et versa ) .

Voici une proposition de déroulement (à adapter à votre modèle de base de données)

* Panier
* Client
* Adresses
* ….

**TESTER A CHAQUE ETAPE !**

### Ressources

* <https://www.grafikart.fr/formations/laravel/eloquent-relations>
* 😍 <https://laracasts.com/series/laravel-from-scratch-2018/episodes/16>
* <https://laravel.com/docs/5.7/eloquent-relationships>
* <https://laravel.com/docs/5.7/controllers#resource-controllers>
* <https://laravel.com/docs/5.7/helpers#urls>
* <https://laravel.sillo.org/les-relations-avec-eloquent-12>
* <http://laraveldaily.com/pivot-tables-and-many-to-many-relationships/>

### Livrable

* Models créés pour chaque table MySQL.
* Relations entre Models
* Boutique fonctionnelle et accessible depuis votre serveur.

### Modalités

* Travail en îlot en autonomie.
* Répartition du travail

### Production

* La description fonctionnelle de votre application complétée (Nom de l’intégrateur, Nom de la table SQL, Nom du Modèle, Nom du Contrôleur, Méthode du contrôleur, Méthode HTTP)

# 

# 

# Projet étape 10 : Validation de formulaire (pour les plus rapides)

### Modalités

* Travail en îlot, en autonomie
* Production de groupe.

### Spécification client

* Nous souhaitons que les données saisies par un utilisateur soient contrôlées avant l’enregistrement dans la base de données.

### Objectif

* Découvrir les mécanismes de validation de données côté serveur

### Consignes

* Vous devez mettre à jour votre application pour intégrer de nouveaux contrôles sur les données saisies dans les formulaires.

### Ressources

* <https://laravel.com/docs/5.7/validation>
* 😍 <https://laravel.com/docs/5.7/validation#form-request-validation>

### Livrable

* Données soumises contrôlées