

Dossier de projet pour le titre professionnel de de développeur web & web mobile



Table des matières

Compétences du référentiel couvertes par le projet.....	3
Présentation personnelle.....	3
Résumé.....	4
Conceptualisation.....	5
Fonctionnalités.....	8
Architecture back-end.....	10
Outils utilisés.....	14
Faibles de sécurité.....	15
Recherche en anglais.....	16

Compétences du référentiel couvertes par le projet

Pour l'activité 1, "Développer la partie front-end d'une application web et web mobile en intégrant les recommandations de sécurité":

- Maquetter une application
- Réaliser une interface utilisateur web ou mobile statique et adaptable
- Développer une interface utilisateur web dynamique
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

Pour l'activité 2, "Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.":

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile
- Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.

Présentation personnelle

Je m'appelle Sataf Bassem j'ai 19 ans , actuellement au sein de la plateforme j'ai suivi le cursus de développeur web/ mobile et c'est un domaine dans lequel je m'épanoui pleinement.

Résumé

Dans le cadre de ma formation de développeur web & web mobile au sein de l'école de la Plateforme j'ai développé une boutique en ligne.

Passionné de musique depuis mon enfance j'ai décidé de porter ce projet sur la vente de vinyle.

Dans un monde où tout est de plus en plus digitalisé surtout dans un domaine comme la musique, le vinyle permet de retrouver des sensations perdu pour les plus nostalgiques ou d'en découvrir pour les nouveaux, On le distingue par ses rainures, la pochette relativement grande en papier et on l'apprécie une fois dans la platine avec son côté hypnotisant. D'autres personnes ne prennent même pas la peine de le déballer, ils préfèrent le garder le plus neuf possible pour enrichir leur collection, ou l'utilisent comme objet de décoration pour son côté "vintage".

Outre son côté visuel certaines personnes retrouvent dans le vinyle un grain de son sans pareil, si il est joué dans du matériel de qualité on obtient un son chaleureux caractérisé par son léger sifflement et a ses imperfections qui rendent si bien à l'oreille des passionnés.

Toutes ses raisons m'ont poussé vers le choix de faire une boutique en ligne de vente de vinyles et de platines.

Avec mes associés nous avons choisi le nom de vinyle Génération en tant que nom de domaine, Après avoir défini le thème de notre boutique nous sommes passé à la conceptualisation.

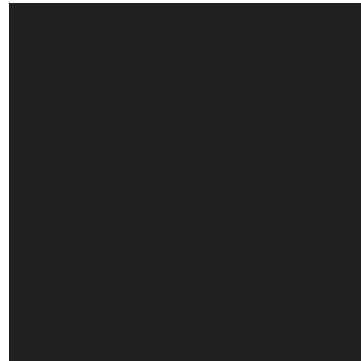
Conceptualisation

charte graphique

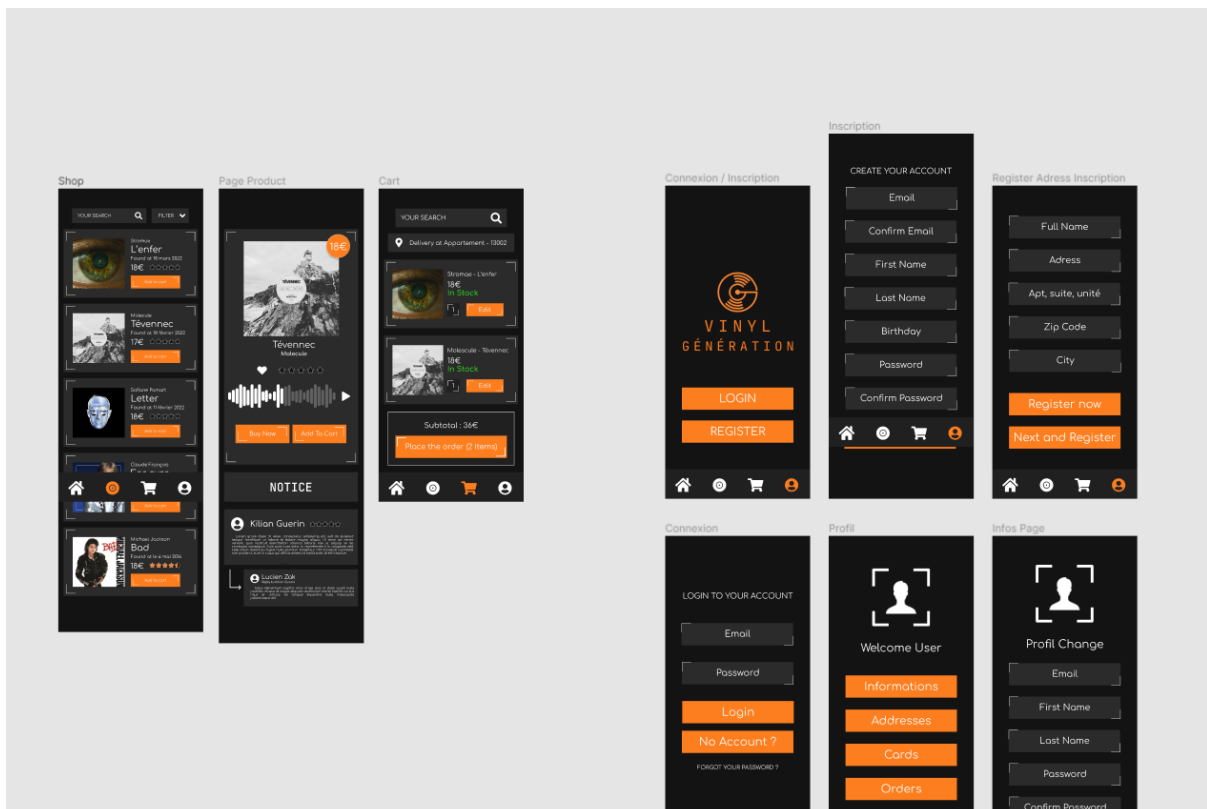
#FC7E1F



#202020



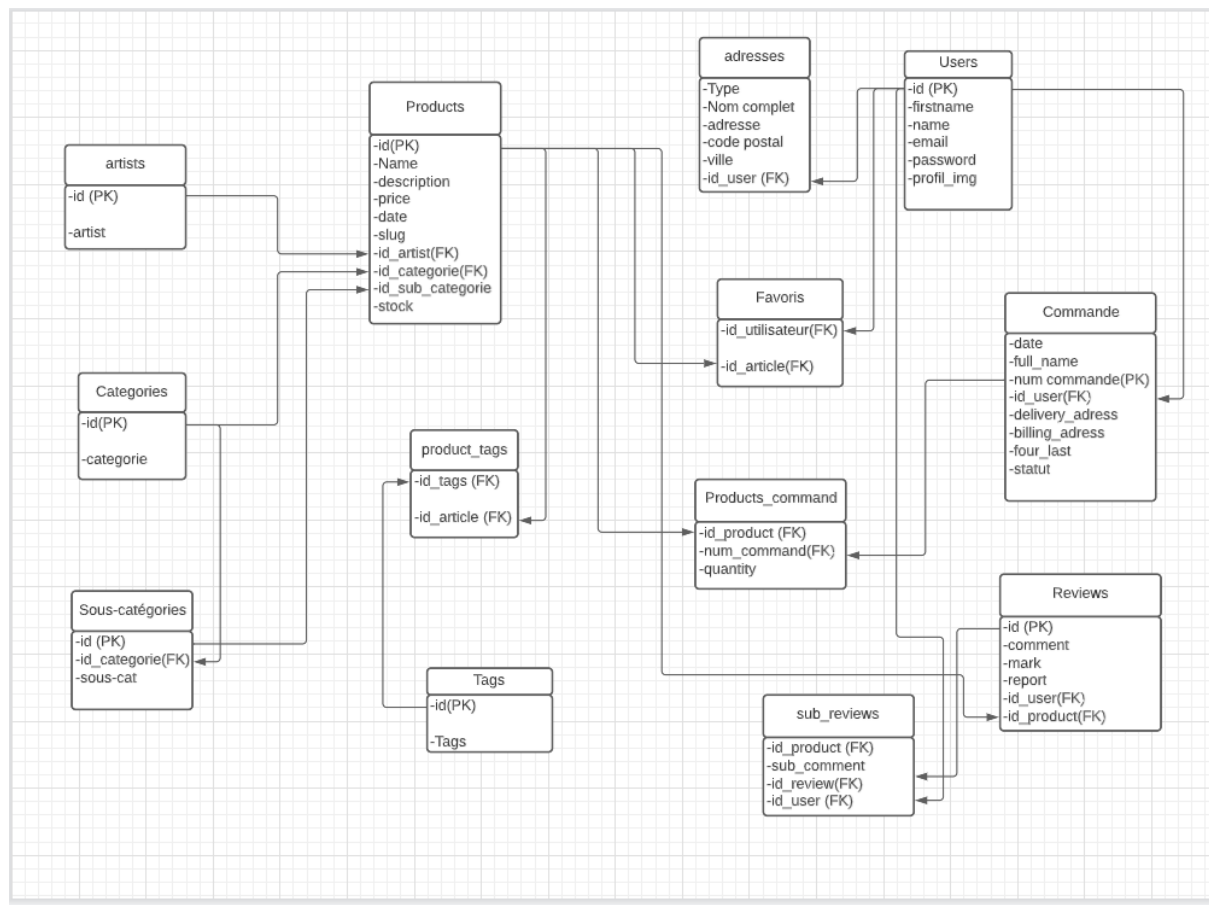
Maquettes



Conception de la base de données

Après avoir établi les besoins de mon site j'ai d'abord fait les conceptions selon la méthode "Merise" puis j'ai effectué la création de ma base de données selon le modèle ci-dessous :

(MLD)



Au total 13 tables, dont 4 tables principales autour desquelles s'articule la base de données.

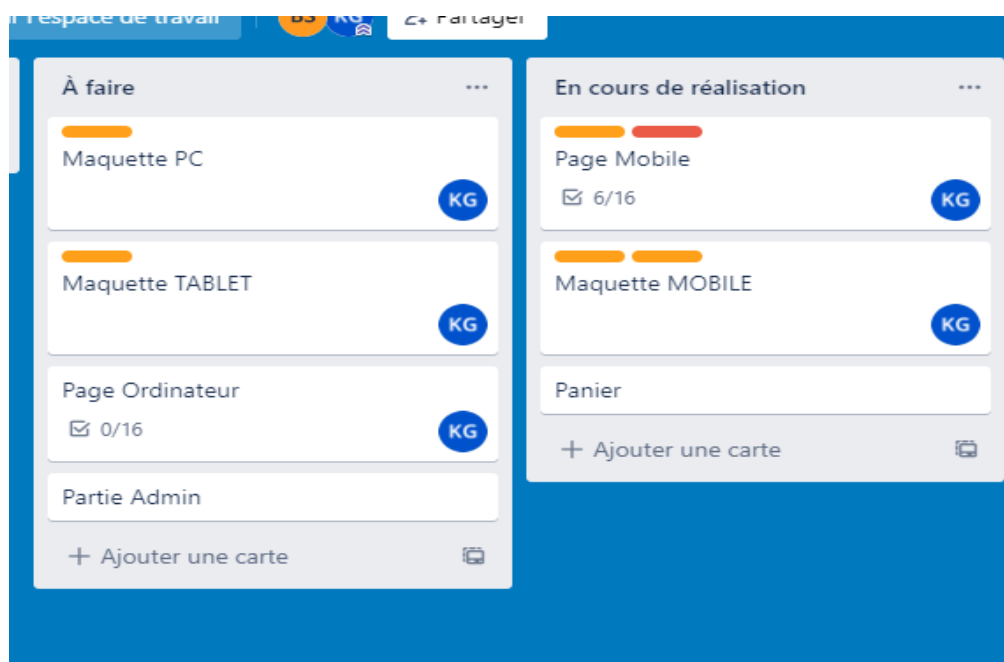
Products qui permet de stocker toutes les informations concernant les produits.

Users qui permet que chaque utilisateur puissent se connecter via email , password elle lié à la table commande qui elle permet garder en mémoire chaque commande passée de chaque utilisateur ainsi la table intermédiaire products_command permet de garder en mémoire tous les produits commandés dans chaque commande ainsi que la quantité demandée.

Certaines problématiques au moment de la création de la base de données ont pu être résolues par la création de tables intermédiaires telles que "Products_command" ou "Products_Tags".

Répartition des tâches

Pour se répartir les tâches équitablement avec les personnes présentes sur ce projet nous avons prit la décision d'utiliser l'outil trello



Fonctionnalités du site

Cibles :

Le site vise une clientèle utilisant plusieurs supports à savoir mobile , ordinateur et tablette c'est pour cette raison que le site sera entièrement responsive.

Authentification :

L'utilisateur aura la possibilité de s'inscrire et de se connecter à son espace personnel via un formulaire .

Page d'accueil dynamique :

La première page du site sera composée d'une présentation du projet , elle présentera les articles les plus récents ainsi que les articles les plus populaires.

Catalogue :

Un catalogue permettra à l'utilisateur de visualiser tous les articles proposés, il pourra les filtrer en fonction de ses envies.

L'utilisateur sera en mesure de voir le nom du produit, une photo de celui-ci, l'artiste (s' il s'agit d'un vinyle par exemple) et le prix.

Barre de recherche :

La barre de recherche aussi sera mise en place pour qu'un utilisateur qui recherche un article en particulier puisse trouver son compte.

Il sera en mesure de chercher un article via :

- ➡ Le nom de l'artiste
- ➡ Le nom de l'album
- ➡ les tags de l'article

Panier :

Le panier permettra à l'utilisateur de sauvegarder un ou plusieurs articles en vue de passer commande. La page affichera tous les articles sélectionnés par l'utilisateur ainsi que le total de ce panier qui correspond à la somme du prix des articles et donc à la somme à payer si la commande est passée. Il aura la possibilité de supprimer un article de celui-ci ou de changer sa quantité.

Panel administrateur :

L'administrateur qui détient les droits aura la possibilité de modifier le contenu du site via un panel d'où il pourra influencer sur :

- ➡ Gestion d'utilisateurs : Il aura la possibilité de supprimer un utilisateur
- ➡ Gestion des produits : Il sera en mesure d'ajouter un produit en lui définissant un nom, un prix, une description, une photo, une catégorie et une sous-catégorie, ou d'en modifier un produit déjà existant.
- ➡ Modération de l'espace commentaire et sous-commentaire : L'admin sera en capacité de supprimer un avis qu'il juge déplacé ou injustifié sous chaque produit
- ➡ Gestion des catégories et sous-catégories : l'admin aura la possibilité de créer une nouvelle catégorie ou d'en supprimer une déjà existante.

Système de paiement :

L'utilisateur sera en mesure de procéder à une commande en toute sécurité grâce à la mise en place d'un système de paiement sécurisé et pour ce faire nous utiliserons la solution proposée par la plateforme STRIPE .

Favoris :

L'utilisateur aura la possibilité de sauvegarder un article autrement qu'en l'ajoutant à son panier, en appuyant sur un cœur qui sera sur la page produit il pourra ainsi ajouter l'article en question à ses favoris et il aura la possibilité de retrouver tous ses favoris sur une page dédiée.

Livraison :

L'utilisateur pourra commander en point relais avec l'implantation de l'api de mondial relay.

Architecture back-end

➡ Architecture MVC : Model , View , Controller :

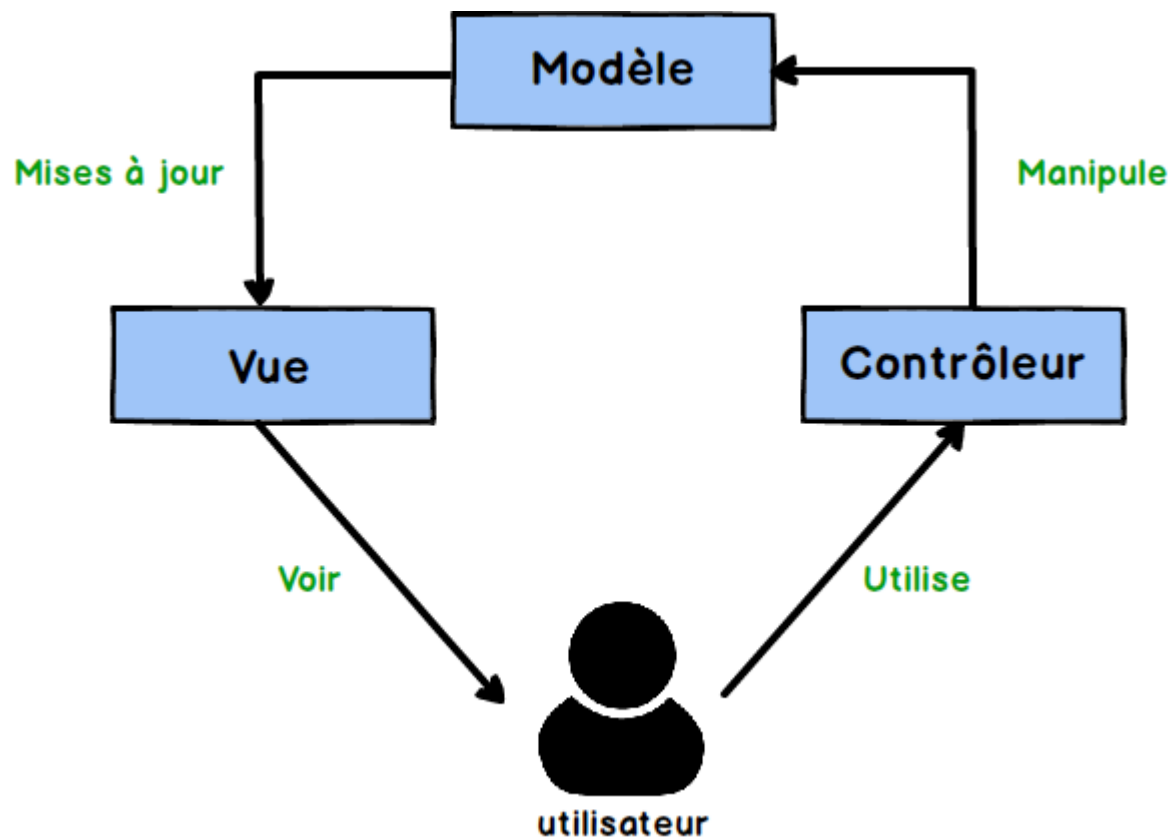
C'est une architecture qui sépare le code en différentes parties ce qui permet une meilleure répartition du travail au sein de l'équipe.

Cette architecture sépare le projet en 3 parties ayant des responsabilités différentes :

Le **Model** est la partie qui communique avec la base de données filtre les informations et les organise.

La **View** est la partie visible côté utilisateur composée principalement de code HTML elle est appelée par le controller.

Le **Controller** fait la liaison entre la view et le Model en créant une logique dans le code.



Routeur

Pour compléter l'architecture mvc j'ai intégré Alto Routeur via composer.

Alto Routeur est un petit routeur mais largement suffisant pour ce projet.

L'intégration d'un routeur permet une URL plus attractive et une sécurité en limitant strictement l'accès de l'utilisateur aux Views tout en restant sur la même page.

```
$router->map('GET', '/account/addresses', function () {  
    $newAddress = new AddressController;  
    $newAddress->AllUserAddresses();  
});
```


Chaque route instancie la méthode d'un Controller qui définit le rôle de cette page .
Dans ce cas, il s'agit d'une route qui permet à l'utilisateur de visualiser toutes ses adresses enregistrées.

```
public function AllUserAddresses()  
{  
    $titrepage = "Adresses";  
    $this->table = "addresses";  
  
    $id_user = $_SESSION['user']['id'];  
    $data = $this->setId_user($id_user)->getAllById_user();  
    $nb = $this->checkAddress($id_user);  
    $params = ['nb'=>$nb, 'data'=> $data, 'titre' => $titrepage, 'css' => 'account'];  
    return AbstractController::render('account.address', $params);  
}
```



```
public static function render($name, $params = [])
{
    require "../App/Views/header.view.php";
    require "../App/Views/$name.view.php";
    require "../App/Views/footer.view.php";
}
```

Chaque méthode retourne un renderer avec comme premier paramètre le nom de page sans le ".view" (exemple si je veux appeler la page "address.view" elle devient "address") et en deuxième paramètre les "params" qui est un tableau où l'on fait passer les informations nécessaires à la page, un titre de page, et le nom du fichier css utilisé.



```
<?php
foreach($params['data'] as $address)
{
```

Ainsi "params" est appelé de cette façon dans la vue.

Pour avoir un code et une architecture propre j'ai décidé de créer un model pour chaque table qui le nécessite, le projet s'articule sous le concept d'héritage ainsi tout les Models hérite d'une classe "Database" qui contient la méthode connect qui permet de se connecter à la base de données, et des fonctions génériques qui me permette d'avoir un code plus lisible et de ne pas me répéter. Par exemple, la méthode run qui permet de préparer une requête et de l'exécuter, ou encore la méthode getAll qui permet de récupérer toutes les données d'une table.

Outils utilisés

Durant ce projet j'ai parfois eu recours à l'utilisation de certains package ou librairies pour intégrer certaines fonctionnalités ou simplement pour un meilleur espace de travail :

- Composer : gestionnaire de paquets.
- Whoops : Gestion des messages d'erreur avec un affichage plus intuitif
- dumper : package permettant d'avoir l'équivalent d'un var_dump beaucoup plus visuel
- autoload : autoloader
- Alto Router : router
- RateYo : Librairie en JQuery permettant d'afficher le système d'étoile présent sur la page produit.
- Stripe : Api permettant un paiement sécurisé pour l'utilisateur.

Faibles de sécurité

Pour garantir la sécurité des utilisateurs, faire de la veille pour se tenir à jour est indispensable dans un domaine comme l'informatique.

Au cours de ce projet j'ai dû faire face à certaines failles que j'ai protégé au mieux que je pouvais avec mes connaissances.

Injection SQL :

Une Faille très connue mais qui n'en reste pas moins redoutable, en effet l'injection sql est une attaque qui les failles dans les sites qui communique directement avec leur bases de données ainsi une personne malveillante peut accéder à votre base de données ou en modifier le contenu en entrant un bout de requête dans le champs d'un formulaire par exemple.

Heureusement en 2022 il est possible d'éviter une attaque de ce genre en filtrant les caractères que l'utilisateur peut entrer dans le champ.

L'utilisation de pdo et la séparation de la requêtes et des paramètres permet aussi d'éviter ce genre d'attaque.

Pour ma part j'ai utilisé "htmlspecialchars" pour filtrer ce que l'utilisateur entre dans le champ et des requêtes à trous ce qui me permet d'être le moins vulnérable possible aux injections sql.

Faibles XSS :

Les failles xss sont très répandues sur internet, et utilisé dans de nombreuses attaques aujourd'hui. Malgré le fait qu'elles soient connus par la plupart des développeur elles sont toujours très présentes dans nos applications pour plusieurs raisons:

- Il est très facile de développer du code vulnérable à cette attaque
- Ces failles sont assez pénibles à corriger
- L'impact de ces failles est assez important lorsqu'elles sont exploitées. Elles peuvent donner par exemple lieu à du vol de données.

Ce type d'attaque intervient sur l'interface de l'utilisateur avec par exemple

l'affichage d'une pop up avec un message ou encore la redirection vers un site externe.

URL :

Grâce à l'utilisation d'un routeur je me prémunis face à l'extraction de données sensibles comme par exemple l'id d'un utilisateur en choisissant le contenu de mon URL .

Mots de passe :

J'ai décidé de protéger des données sensibles comme le mot de passe d'un utilisateur par le biais de hachage. Pour ce faire j'ai utilisé le hachage "Password Default" via php qui utilise l'algorithme bcrypt et qui évolue au cours du temps pour une meilleure scalabilité de mon projet.

Recherche en anglais

En choisissant de mettre Alto Routeur j'ai dû faire face à une problématique qui est le langage de la documentation officielle de Alto Routeur qui est l'anglais. Pour connaître le fonctionnement de ce routeur j'ai dû lire studieusement cette documentation officielle sans me laisser affecter par la barrière de la langue.

Altorouter

PHP5.3+ Routing Class. Supports REST, dynamic and reversed routing.

[View the Project on GitHub](#)
dannyyankooten/AltoRouter

[Download ZIP](#)

[View on GitHub](#)

[Star](#) 1,115

This project is maintained by [dannyyankooten](#)

Hosted on GitHub Pages
Theme by [orderedlist](#)

Example:
user_details

Example Mapping

```
// map homepage using callable
$route->map( 'GET', '/', function() {
    require __DIR__ . '/views/home.php';
});

// map users details page using controller#action string
$route->map( 'GET', '/users/[i:id]/', 'UserController#showDetails' );

// map contact form handler using function name string
$route->map( 'POST', '/contact/', 'handleContactForm' );
```

For quickly adding multiple routes, you can use the `addRoutes` method. This method accepts an array or any kind of traversable.

```
$route->addRoutes(array(
    array('PATCH', '/users/[i:id]', 'users#update', 'update_user'),
    array('DELETE', '/users/[i:id]', 'users#delete', 'delete_user')
));
```

Match Types

You can use the following limits on your named parameters. AltoRouter will create the correct regexes for you.

```
*                // Match all request URIs
[i]              // Match an integer
[i:id]           // Match an integer as 'id'
[a:action]       // Match alphanumeric characters as 'action'
[h:key]          // Match hexadecimal characters as 'key'
[:action]        // Match anything up to the next / or end of the string
[create|edit:action] // Match either 'create' or 'edit' as 'action'
[*]             // Catch all (Lazy, stops at the next trailing slash)
[*:trailing]     // Catch all as 'trailing' (Lazy)
[**:trailing]    // Catch all (possessive - will match the rest of the string)
[:format]?      // Match an optional parameter 'format' - a / or .
```

The character before the colon (the 'match type') is a shortcut for one of the following regular expressions

Dans cet exemple , la documentation nous explique comment créer nos routes avec un exemple assez précis , on peut aussi comprendre qu'il est possible de créer plusieurs routes plus rapidement en entrant un tableau de route.

Puis dans un second temps la documentation nous montre la syntaxe utilisée pour chaque cas de figure dans les cas où la route est dynamique et évolue en fonction d'où l'utilisateur se dirige.

Par exemple pour ne créer qu'une seule route pour tous les produits on créer une route du style "products/[i]" on définit "i" en tant que l'id du produits ainsi la page évoluera en fonction de chaque cas.