Documentation

The core aspects of the solution developed

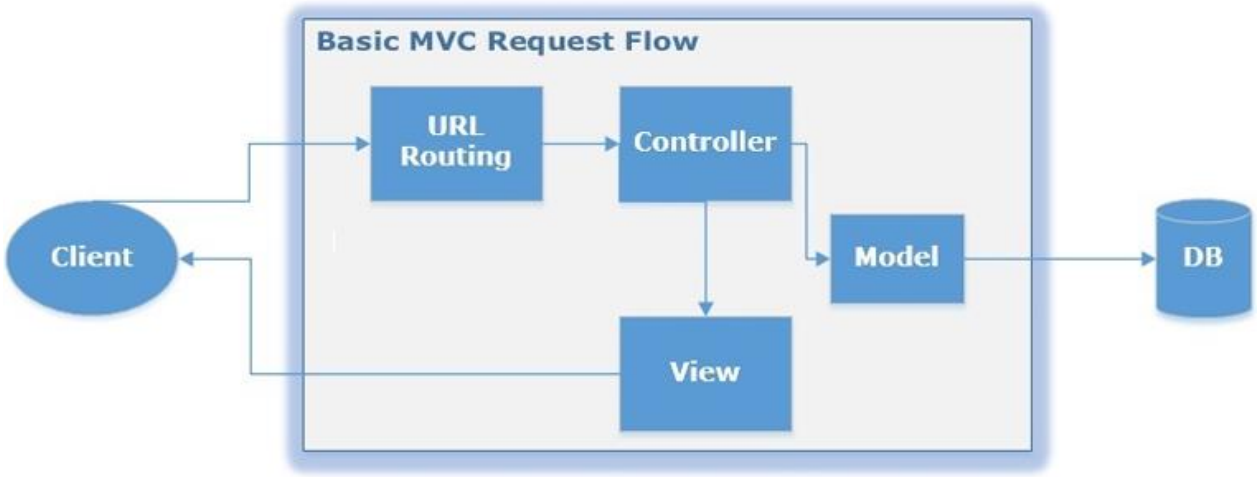
In this assignement we used multi aspects for to design our website :

1. Api that contains the code to power the simple API for cars data
2. Composer to declare and install the libraries that the main project needs.
3. MVC pattern for makes our code easy to organize.
4. A database for save all form entries.
5. PhpUnit allows the implementation of regression tests by verifying that the executions correspond to the predefined assertions.

The design pattern : MVC Architecture

As your website grows, you'll have trouble organizing your code. This assignement are meant to show you a good way to design your website. A *design pattern is* a series of best practices for organizing your site.

One of the most famous *design patterns is* called **MVC** (Model - View - Controller): it is the one we will discover in this assignement.



1. Presentation of the MVC design pattern

The **MVC** pattern makes it easy to organize your source code. So far, we have programmed monolithically: our web pages mix processing (PHP), data access (SQL) and presentation (HTML tags). We will now separate all these parts for clarity.

1.1 M: The model

The model is responsible for data management, including interactions with the database. This is, for example, the class **CarModel** contains four functions :

- getCars : to fetch all list cars
- getDetailCar : to fetch a detail for one car
- getSimilarCars : to fetch similar cars for one car requested
- submitQuote : to store form data into database.

1.2 V : The view

In the view are grouped all the lines of code that generate the HTML page that will be sent to the user. Views are files that contain almost exclusively HTML code, with the exception of a few that echodisplay variables pre-filled by the controller. However, a loop foreach is allowed for views that display a list of items. **The view does not perform any processing, calculations** .

In our assignement, the view would be the view/index.php and view/detail.php . The code in these files are used to display a web page containing all the cars contained in the variable cars and the detail for each car in the variable carDetail.

1.3 C: the controller

The controller manages the logic of the code that makes decisions. It is sort of the intermediary between the model and the view: the controller will ask the model the data, analyze it, make decisions and call the appropriate view by giving it the text to display in the view. The controller contains only PHP.

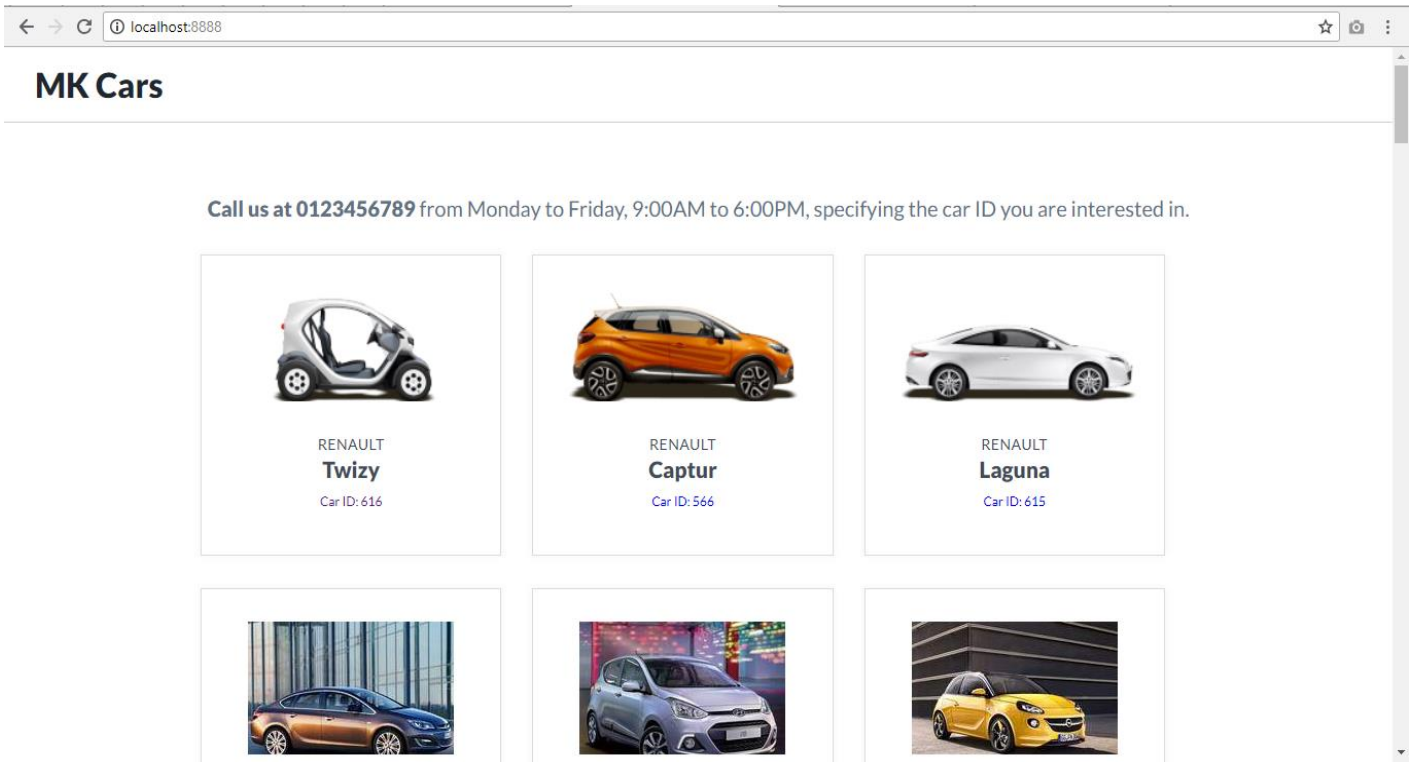
There are a multitude of MVC implementations :

1. A big single controller
2. One controller per model
3. A controller for each action of each model

We choose here the intermediate version and start to create a controller for **CarModel**. Here is the controller src/CarController.php on our example:

Our controller is broken down into several parts:

1. The declaration of the class is charged CarModel;
2. we use the model to retrieve the table of all cars with
   $cars = CarModel::getCars();
3. we then call the view that will generate us the web page with
   include CONFIG_VIEWS_DIR . '/index.php';



localhost:8888/detail/1410

MK Cars

KIA
Rio

SUBMODEL: BERLINA 2 VOL. 5 PORTE
YEAR: 2015
INTERNAL SPACE: 2-4 PEOPLE
ROOF: FIXED
FUEL TYPE: GASOLINE
LOOK: MODERN

Nome

Cognome

Email

Telefono

CAP

Telefono

Ho letto e accetto la privacy policy

REQUEST QUOTE

Similar Cars: