PHP Tech Interview Assignment

Introduction

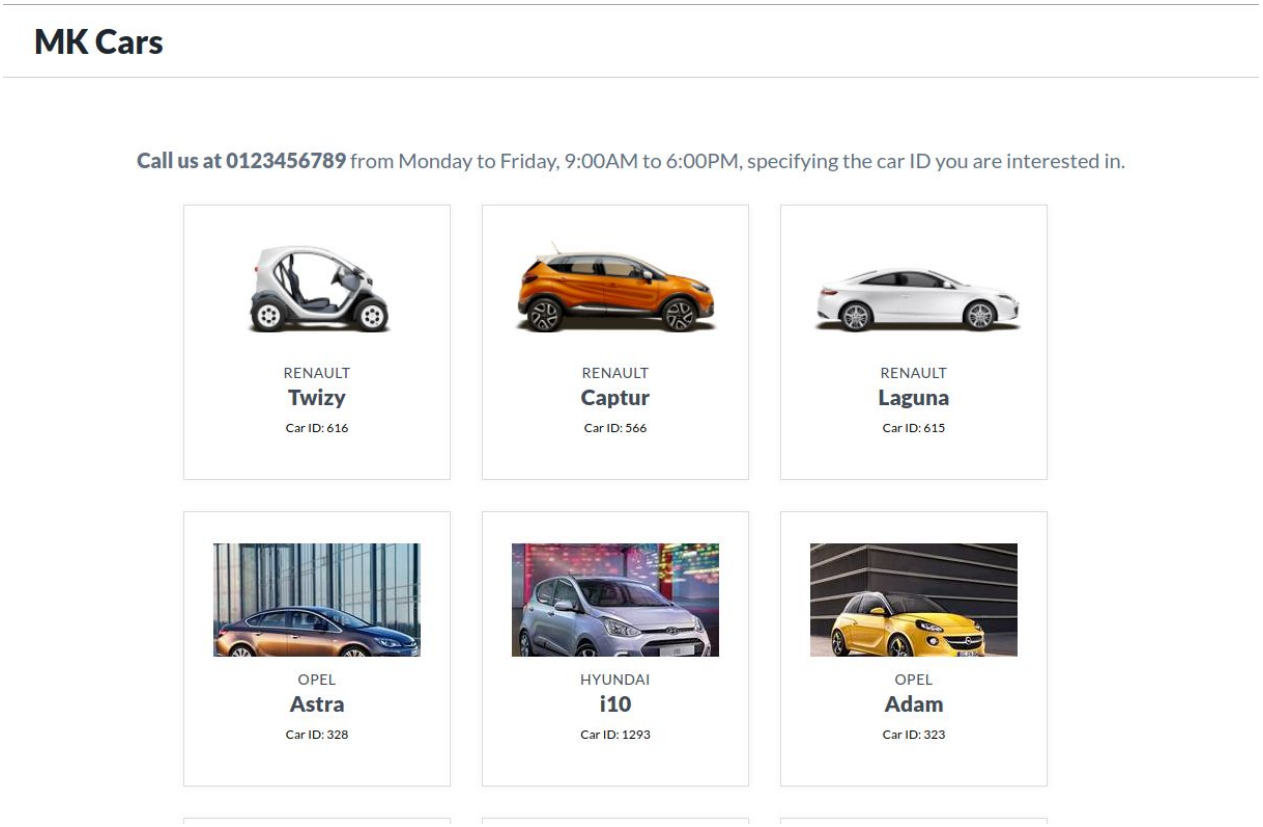
Welcome! At MotorK we believe that one of the best way to evaluate your capabilities and generate a discussion about software is to give you a small and self-contained assignment you can perform in a pressure free environment and with enough time on your hands. For this reason, we created this small *compito*.

Overview

- **Input**
    - A zip file containing a skeleton PHP [>7.0] application
- **Expected Output**
    - The link to a **git repository** with the above application extended with the requirements requested
- **Send back to**
    - Your contact at MotorK
- **Deadline**
    - Within 7 days, counting from the day the assignment was handed.

Description

MK Dealers, one of our clients, asked us to improve a web page made by a previous agency.



This page shows a list of cars: people who want to request a quote for a car have to phone the dealer and tell him the car ID they're interested in.

Mr. MK wants us to create a page with a form for each car, so that he can stop answering phone calls.

Part 1

- Refactor the current web page by making use of the MVC architecture pattern
- Create a detail page for each car
- Add the form to this detail page.
- Save the form submissions (also called *Leads*) in a SQLite database table containing these fields (Id, name, lastname, email, phone, cap, privacy, carId)

The implementation should be done in plain PHP, without using any framework.

Minimal UI and CSS is provided and it will not be part of the evaluation, so focus on the domain modeling and business logic. But if you have spare time, you could try to improve it.

Part 2

- Based on the car dimensions/tags (described below), devise a creative recommendation-engine that is able to return cars similar to the one requested. Print these suggestions in the cars' detail page.

Data source and description

Cars' data is provided through an API (not to be implemented) that has two endpoints:

- `/search`
  - This will return the list of all cars available.
- `/detail/{detailID}`
  - This will return data about a single car.

A car can be characterized by a set of categories, each corresponding to a given car property such as "**Look**", "**Fuel Type**" and "**Segment**". In all of these categories, there is an arbitrary number of tags that describe all the possibilities for that category, such as "*modern*", "*classic*" or "*original*" for the "**Look**" category.

Every car is represented by a set of these tags. From all the categories you will find in the input file, for this project, you should **only** consider the following:

- **Internal Space:** describes the available seating space in the car
  - Values: for single, 2-4 people, 4 people comfort, 5 people confort, 7 seats, > 7 seats
- **Segment:** the type of car
  - Values: citycar, utilitarian, compact, crossover, berlina-2v, berlina-3v, station, monovolume, suv, supercar
- **Fuel type:** the fuel used to power the car
  - Values: gasoline, gpl, gas, hybrid, electrical, diesel
- **Look:** subjective appreciation of its look
  - Values: classic, modern, sport, original

- **Price:** statistical percentile range of the car's price considering all models in a country
    - Values: 1 to 100 (1st percentile, 2nd percentile, … 100th percentile)

Evaluation Points

Your solution will be evaluated using the following criteria:

- Code Quality
    - Domain modelling, re-usability, performance and creativity;
- Unit Tests (covering all the required business logic);
- Degree to which solution fulfills all requirements requested;
- Documentation in English, providing information on the core aspects of the solution developed, and explaining the design choices taken.

Installation, usage and co.

All the technical details about the application can be found in the `README.md` included in the zip file.

Useful references

- PHPunit Manual
- Composer documentation
- Phinx documentation