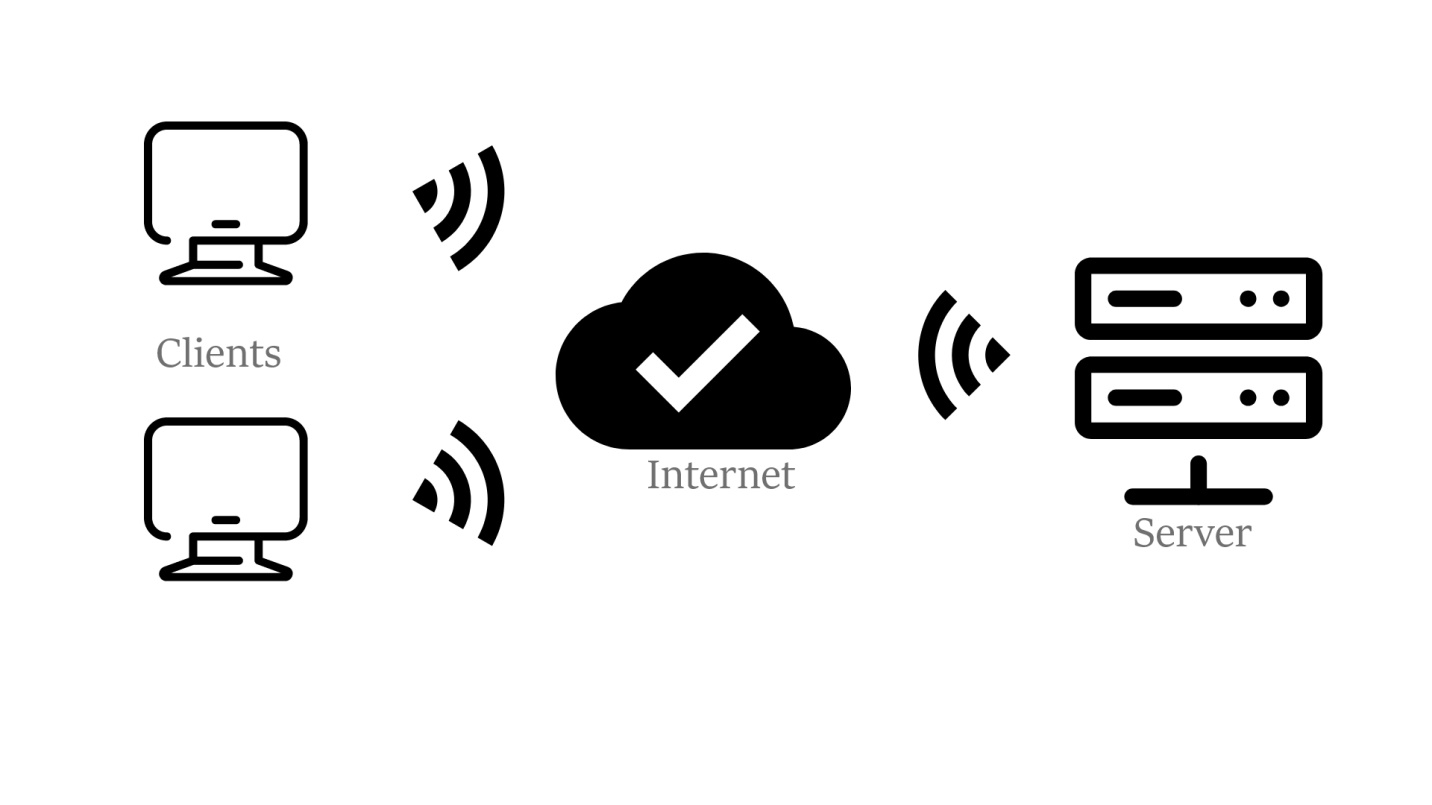
Application Architecture

Conceptual Architecture

On the conceptual layer our application will use the Client-Server Architecture.

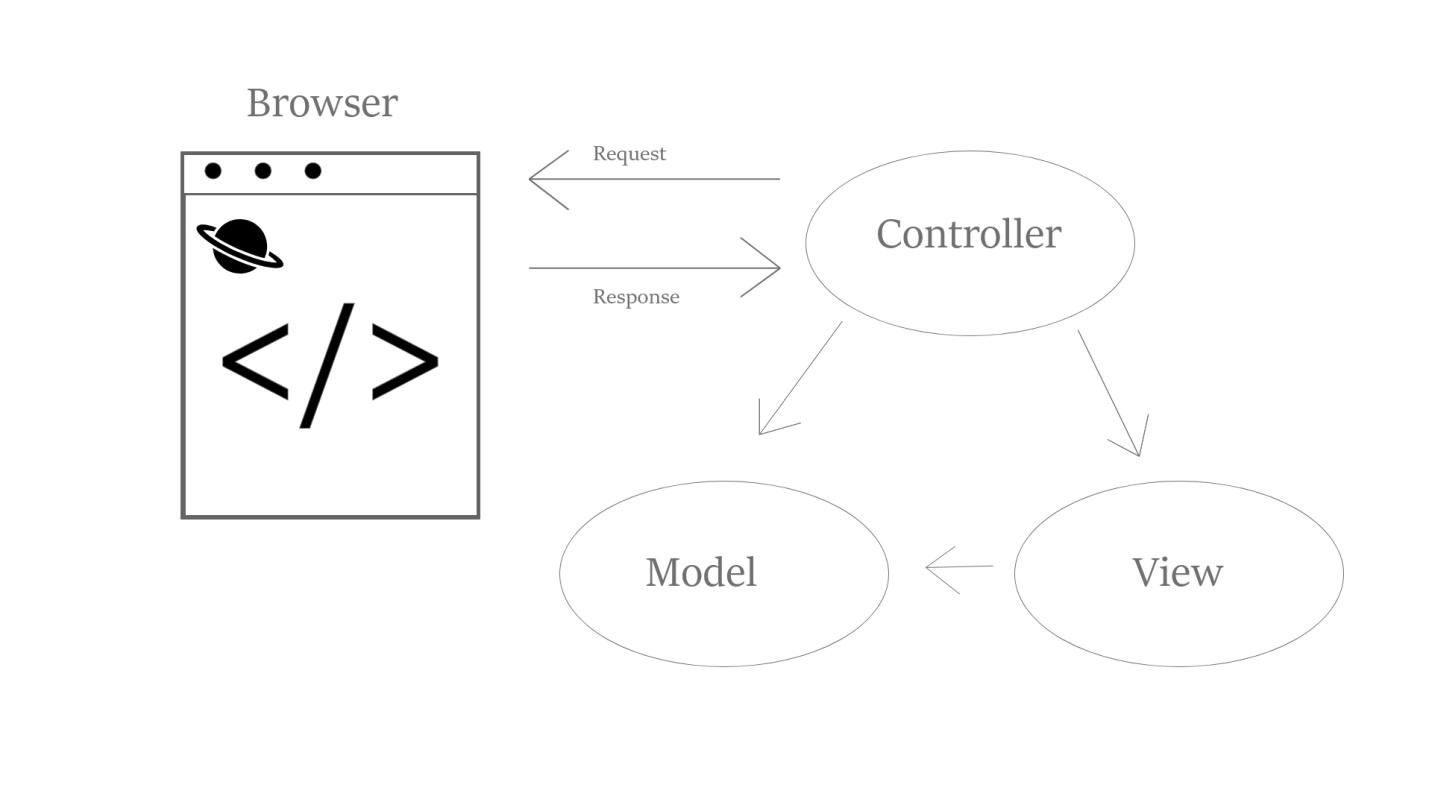


Client server architecture is a computing model in which the server hosts, distributes and controls part of the resources and services to be used by the client. The design is made up of one or more client systems connected to the main server through a network so called internet.

This architecture is suitable for our idea because it is the standard for most web-applications.

Execution Architecture

Our web application will use the MVC architectural style, which separates an application into three components - Model, View, and Controller



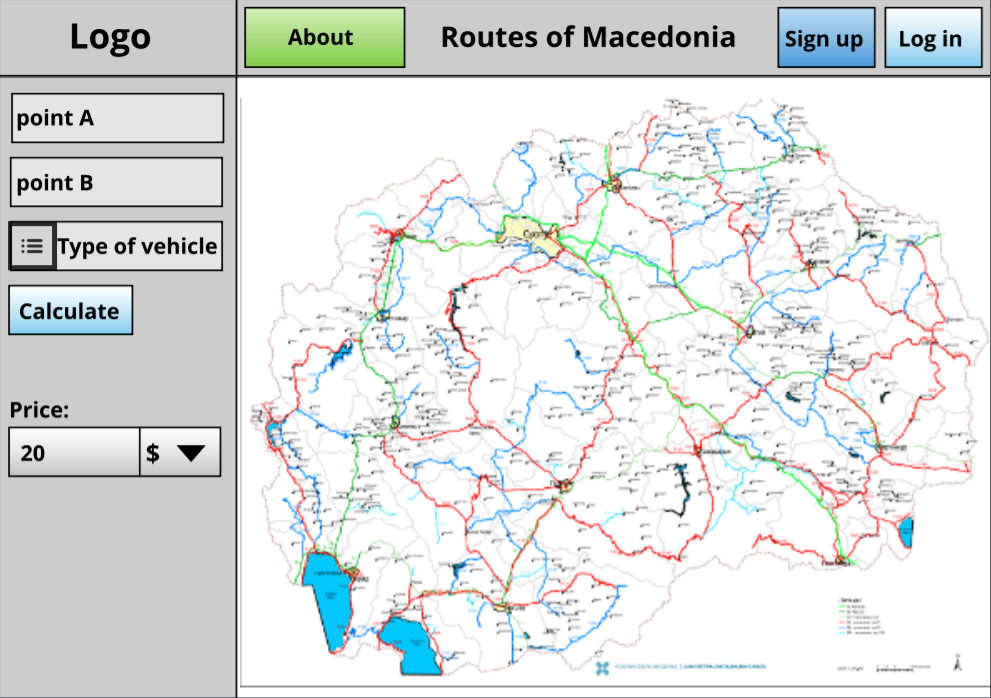
* **Model**: represents the shape of the data. A class in C# is used to describe a model. Model objects store data retrieved from the database.
* **View**: View in MVC is a user interface. View display model data to the user and also enables them to modify them.
* **Controller**: The controller handles the user request. Typically, the user uses the view and raises an HTTP request, which will be handled by the controller. The controller processes the request and returns the appropriate view as a response.

As per the above figure, when a user enters a URL in the browse, it goes to the server and routed to a controller. A controller then executes the related view and models for that request, and creates the response which it sends it back to the browser.

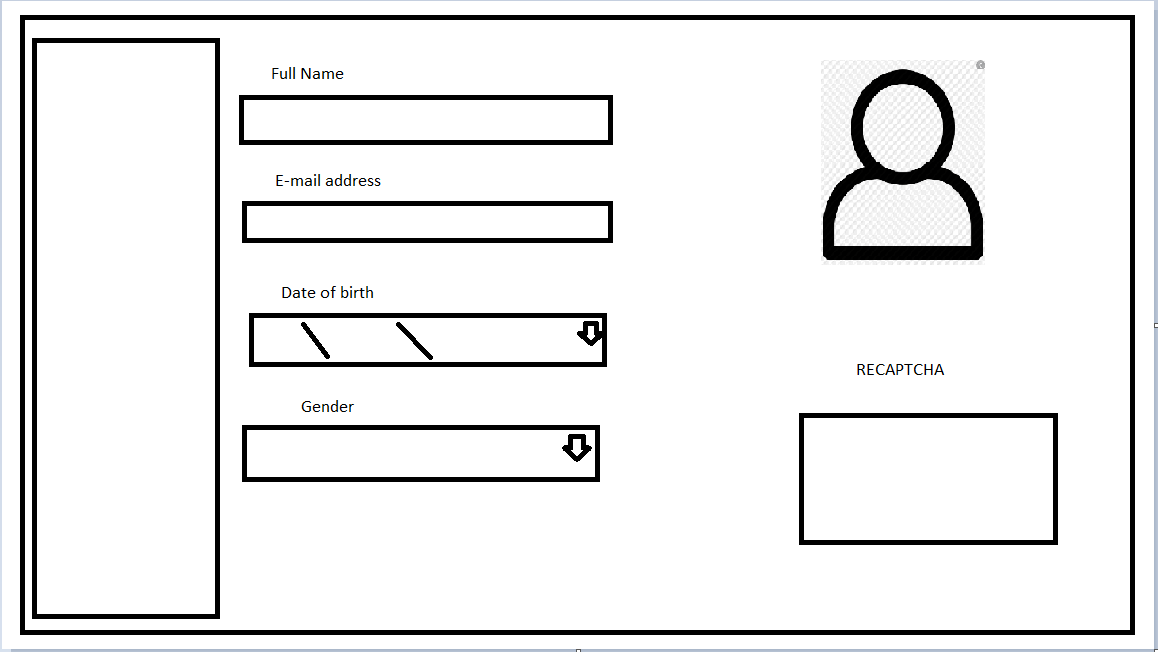
Implementation Architecture

MVC was the ideal option regarding that it is widely used for web applications, as well as the fact that there are many existing frameworks like AngularJS, Struts, Ruby on Rails , Django, Spring and similar.

The Model encapsulates the application state and is represented on the following picture. As it can be seen, the functionality is similar to that of a basic Google Maps section, with a couple of additional functions meant for our users’ interest.

The design of our model is a “friend” of simplicity as it calculates optimal routes between point A and point B. The calculation can be done using different parameters such as cheapest or shortest. Before the calculation, there is one in-between step – the selection of vehicle type, which is crucial to calculate the total price for tolls. 

The View part is the user interface and it renders the model, displays the control buttons, requests updates from the model and most of the time and also what happens to be in our case – there are multiple views. The picture represented above is just one of the views. Another view is the Sign Up/Log In view where Users enter their personal information and register in the database of the server.

And lastly there are the Controllers – the functions or the tools of manipulation with the app.

The Controller and the View together make the UI. The characteristics of the controllers are that they react on user input, modify the model, and dispatch the view.

Our primary and prototypical controllers are the:

1. The Sign up and Log in buttons which link the user to another page.



1. Calculate button which sums up the price of the selected route.



1. The About button which links the user to a page with all the information connected to the application as well as contact data.



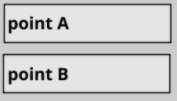
1. Select Vehicle dropdown chooser – enables the user to choose from several types of vehicles and by that the back-end algorithm knows what formula for calculation should be used



1. CTRL+Scroll wheel – enables the user to zoom in our out while hovering over the map, changing the view to the customer’s choice.



1. Point A and B – can be selected by left-clicking on a location of the map, or entering coordinates manually from the keyboard



1. Price field with dropdown chooser- The software informs the user of the cost of a certain route and the user can choose the desired currency he/she wants the funds to be displayed in.

