HTECGROUP-FE PROJECT DOCUMENTATION

Stefan Milovanović

Used technologies

For the purpose of development of the project, there were used two popular frontend technologies:

1. Angular 4
2. Bootstrap (grid system only)

Styleguides

Javascript code is written by Robert C. Martin’s book Clean Code: A Handbook of Agile Software Craftsmanship. All class names start with capital letter and every separated word in the name starts with capital letter. All the variables are written in the camel case notation. All component names are written with small letters where each word in the name is divided by “-“. Component names contain .component and end with .ts extension.

BEM (Block Element Modifier) methodology is used for writing css classes.

Project Organization

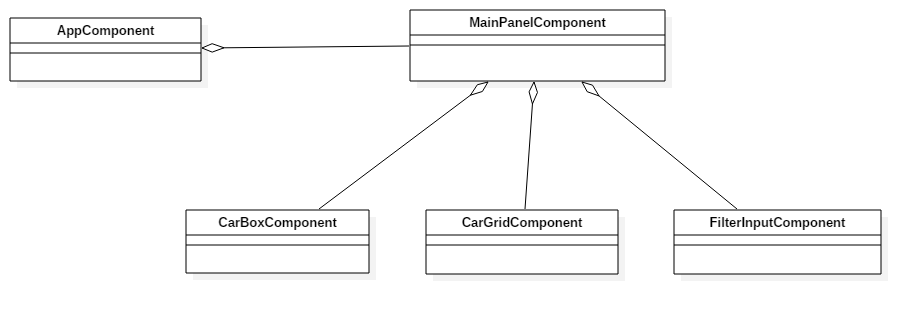
Project is divided in three main parts:

1. Components
2. Models
3. Services

Component folder contains all components that are used in the realization of this project:

1. Car-box component
2. Car-grid component
3. Filter-input component
4. Main-panel component

UML Class Diagram below shows how the components are connected.



Main Panel component

Main panel component contains all informations that are rendered. It contains logic for rendering other components, when they should be rendered, or when animation should be started. Basicly it is root component for all other components except AppComponent that is root component for the project, and is also main module.

Public functions within the component are:

* updateDisplay(searchResult:Array<Car>) – method that updates search result, when another search is completed.
* addCarForTracking(carGrid:CarGridComponent, car:Car) – method that class the same method from the car grid component and adds car in the tracking list.
* startAnimation(animationSpeed:number, carGrid:CarGridComponent) – this method starts an animation when start button is hit.

This component is very simple both with functionality and view logic.

Filter Input component

This component is used for the search. Basicly it send http request to data.json and receives data from the request. When data is received it emits event that search is finished and provides result to the main panel.

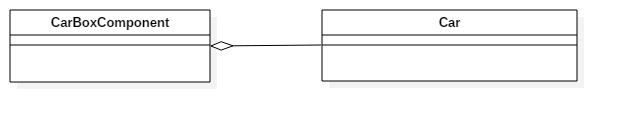
Functions within this component are:

* search() – method that is called when search button or enter is hit, it searches for the data, and emits event searchFinished for the MainComponent.
* \_filterCars(cars:Array<Car>) – method that filters cars by the search (query) string, and is used within search function.

Car Box component

Car Box component represents one box that contains information about one car. It contains logic for applying particular border based on whether a box is selected or not. Also it has function that is used for selecting car, function emits event when car is selected.

Function within this component are:

* selectCar() – method that applies border whether car is selected, and emits onCarSelected event.
* \_applyBorderStyle() – method that applies border style whether car is selected or not, it is used within selectCar method.

Car Grid component

Car Grid component is the most complex component in this project. It has a lot of private/public methods that are used for the animation. Basicly it renders grid based on how much cars are added for tracking. It renders semaphores and speed limit signs, and creates a row for each car in the tracking list. It contains logic for animation, animation changes (semaphore wait, speed limits etc.). This component has information about whole traffic, about tracking cars, and it has few dictionaries.

Here will be explained main elements of the component.

1. trackDictionary - private attribute, that represents associative array where key is id of a car, and the value is position of the car on the grid. Basicly it represents in which column is car contained in the grid.
2. semaphoreDictionary – private attribute, that represents associative array where key is JSON string of specific traffic light(semaphore) , and the value is DOM id of semaphore HTMLElement.
3. speedLimitsDictionary – private attribute, same as semaphoreDictionary just for speed limits.
4. standing- is array that contains cars , in the order of how the reached the end. So it contains information about who is winner, second place,third place etc.
5. speedLimitReachedDictionary – associative array where key is id of a car and a value is bool that shows if car has reached a speed limit sign.
6. semaphoreReachedDictionary- same as speedLImitReachedDictionary , just for semaphores
7. semaphoreIntervalIds- associative array that contains information for each semaphore where semaphore as JSON string is key and intervalID is valu. IntervalId represents id returned from SetInterval function. This is used when semaphore need to be restarted.

Main function for this component that starts animation is animate().

It looks like this:

animate() {

this.\_resetSemaphores();

this.\_activateSemaphores();

this.trackingCars.forEach((car: Car) => {

let duration = this.\_calculateTimerDelayBySpeed(car.speed);

this.\_animateCar(car, duration / this.\_animationSpeed);

});

}

Function \_animateCar is used to animate specific car with given speed(duration of the SetTimeout function).

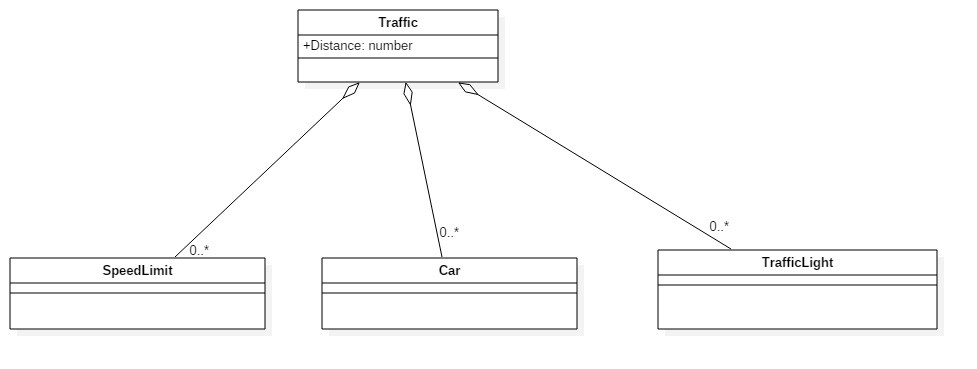
All functions are written using Clean Code instruction, so they are readable and understandable.

Models

There are registered four model classes:

* Car
* SpeedLimit
* TrafficLight
* Traffic

First three are contained in the Traffic class. Diagram is shown below.



Services

Services are classes that provide data or functionality. The are created as Injectable classes, so whenever their use is required the can be injected in any class. There are three services used for this project:

1. RequestService
2. TimeService
3. TrafficService

Request service

Request service is modeled as façade pattern. It encapsulates http requests on the lower level. It has private attribute url, that contains address of the targeting server.

From the public api there are :

* createGetRequest(data:any, action:string) – creates get request based on the given data, and is targeting given action. It returns observable with the result.
* createPostRequest(data:any, action:string) – creates post request based on the given data, and is targeting given action. It returns observable with the result.

Time service

Time service is class that contains one static method, calculateTimespanInSeconds. It is used for calculation how much seconds has passed between two dates. The function looks like this:

public static calculateTimespanInSeconds(start:Date, end:Date){

let diff = end.getTime() - start.getTime();

let seconds = Math.floor(diff / (1000));

return seconds;

}

Traffic service

Traffic service is service that injects RequestService, and uses it to send http request to data.json. There is one public method, getTraffic(), that returns traffic data, converts it to Traffic model, and return it as Observable.

Summary

Here is mentioned basic implementation od the project. This document contains basic information about methods, classes, components etc.

CarGridComponent has a lot of functions, so I mentioned just important things so it would be easier to understand it’s logic. Every functionality has own function so that is the reason for the larger number of functions. Each function is properly named by convection, and it will not be difficult to understand it, and recognized it’s functionality.