

**Софийски университет „Св. Климент Охридски“**

**Факултет по математика и информатика**

**ДОКУМЕНТАЦИЯ**

на тема:

„Коли БГ“

курсов проект по

**"Мобилно интернет съдържание"**

Изготвил:

**Иван Вичев, ф.н.80926,**

**Специалност: Компютърни науки, 4.курс**

**Дата: Ръководител:**

**05.06.2017. проф. д-р Васил Цунижев**

**Съдържание**:

**1.Функционален анализ.   
2.Технологичен анализ.**

**3. Функционално проектиране User Guide**

**4. Технологично програмиране “Technical Reference”  
5.Тестване . Преносимост, екран от различни типове устройства.**

**1.Функционален анализ. Алгоритъм,функционалност,интерфейси**

**За база на проекта е използван:**

<https://github.com/Omega-time/time-tracker-frontend>

**От даденият сайт е взета структурата на съдържанието на Web като в последствие бива разширена за mobile. Функционалността е конструирана благодарение на, HTML5, CSS3 и JS, както и библиотеката JQuery, използвана за улеснение при писането на скриптовете, както и list.js която спомага манипулацията на списъците. DOM (Document Object Model) са елементи от JQuery които улесняват създаването на анимации и AJAX приложения. От примерния проект са използвани следните функционалности:**

* **CSS3 стилизация;**
* **Ефекти;**
* **Събития;**
* **DOM;**
* **AJAX;**

**Проектът има адаптивен дизайн, или иначе казано приспособява се към резолюцията на устройството на което бива достъпен. Адаптивният дава най-доброто преживяване за потребителя, без значение от какво устройство има достъп до него- таблет, мобилен телефон или компютър.**

**Използвана още е и библиотеката bootstrap и по точно нейната разновидност bootswatch която позволява използването на всички функционалности на bootstrap плюс различна цветова палитра.**

**Също така е използвана и библиотеката list.js която позволява лесната манипулация на списъци.**

**2.Технологичен анализ.**

**Езици, протоколи-анализ на производителността, преносимостта**

**В проекта са използвани следните библиотеки и технологии:**

* **HTML5 – стандартен език за маркиране, който се използва за създаването на уеб страници**
* **CSS3 - език описващ форматирането и визуализацията на съдържанието**
* **Javascript - език за създаване на динамични уеб страници**
* **jQuery -** **jQuery е JavaScript библиотека, създадена, за да улесни използването на клиентски скриптове в HTML**
* **list.js - позволява лесната манипулация на списъци.**
* **Twitter Bootstrap -** **Библиотеката за Bootstrap представя универсален набор от стилове, които създадени за решаването на множество проблеми, свързани с web дизайна.**

**3. Функционално проектиране User Guide**

**Сайта позволява като основна функционалност добавянето промяната и триенето на информация за автомобили давайки пълна свобода на потребителите да манипулират информацията с цел постоянното и обновяване.**

**Също така позволява търсене по година на производство или модел като по този начин улеснява достъпа на потребителите до желаната информация.**

**4. Технологично програмиране “Technical Reference”**

**Този сайт е създаден с цел да направи достъпа до базова информация за различни видове модели автомобили лесен и приятен позволявайки на потребителите да променят и допълват информацията за автомобилите като по този начин помагат на други потребители да направят по лесен избор при закупуване на автомобил. Съдържанието на сайта е адаптирано към различни видове устройства като основните функционалности свързани с адаптацията са**

**менюто: преминава в dropdown меню при промяна на резолюцията под тази за таблет като в допълнение се залепя за върха на страницата за улеснен достъп на всякъде по страницата;**

**както и промяната на input полетата и таблицата с информацията при промяна на резолюцията също както и полетата за търсене които подпомагат лесното търсене в страницата.**

**Също така таблицата се конструира динамично използвайки JS.**

**Изображението на началната страница също така е адаптивно и променя размера си спрямо размера на устройството :**

**img{**

**display: block;**

**margin: 0 auto;**

**width: 80%;**

**}**

**Също така допълнителен CSS бива използван за други елементи като центриране на текст в таблица или оразмеряване на input полетата**

**КОД:**

    td,

    th {

        text-align: center;

    }

    tr {

        height: 50px;

    }

    form {

        text-align: center;

    }

    input {

        width: 30%;

    }

    .data-input{

    width: 100%;

   float: left;

   display: block;

    }

    span{

        display: block;

    }

    .contact-info {

        width: 60%;

        margin: 0px auto;

    }

    #home-info{

        text-align: center;

    }

    address{

        text-align: justify;

    }

    .navbar{

        position: fixed;

        width: 100%;

    }

    .container{

        padding-top:40px;

    }

**КОД:(Меню)**

    <nav class="navbar navbar-default main-menu">

        <div class="container-fluid">

            <div class="navbar-header">

                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#mynav">

                                <span class="icon-bar"></span>

                                <span class="icon-bar"></span>

                                <span class="icon-bar"></span>

                </button>

            </div>

            <div class="collapse navbar-collapse" id="mynav">

                <ul id="menu" class="nav navbar-nav">

                    <li><a href="home.html" class="navbar-brand">Начало</a></li>

                    <li><a href="cars.html" class="navbar-brand">Коли</a></li>

                    <li><a href="contacts.html" class="navbar-brand">За контакти</a></li>

                </ul>

            </div>

        </div>

    </nav>

**КОД:(Полета)**

<div class="container">

        <h3>Добавяне на кола: </h3>

        <div class="form-group">

            <form id="add\_form" class="well">

                <label>

                        <span>Модел:</span>

                        <input id="model" type="text" name="model" class="data-input">

                    </label>

                <label>

                        <span>Година:</span>

                        <input id="year" type="text" name="year" class="data-input">

                    </label>

                <label>

                        <span>Описание:</span>

                        <input id="description" type="text" name="description" class="data-input">

                    </label>

                <br><br><br>

                <input id='add-car-btn' type="submit" value="Добави" class="btn btn-primary">

            </form>

        </div>

        <h3>Промяна на кола:</h3>

        <div class="form-group">

            <form id="edit\_form" class="well">

                <label>

                    <span>Модел:</span>

                    <input id="edit\_model" type="text" name="model" class="data-input">

                </label>

                <label>

                    <span>Година:</span>

                    <input id="edit\_year" type="text" name="year" class="data-input">

                </label>

                <label>

                    <span>Описание:</span>

                    <input id="edit\_description" type="text" name="description" class="data-input">

                </label>

                <br><br><br>

                <input id="update\_button" type="submit" value="Промени" class="btn btn-primary">

            </form>

            <br>

            <h3>Търсене:</h3>

            <div class="well">

                <form>

                    <input list="models" name="models">

                    <datalist id="models"></datalist>

                    <input id='filter-by-model-btn' class="btn-success" type="submit" value="По модел">

                </form>

                <form>

                    <input list="years" name="years">

                    <datalist id="years"></datalist>

                    <input id="filter-by-year-btn" class="btn-success" type="submit" value="По година">

                </form>

                <br>

                <form>

                    <input id="restore-table-btn" type="submit" value="Изчисти" class="btn btn-warning">

                </form>

            </div>

        </div>

        <div id="table"></div>

    </div>

**Код:(Таблица)**

function buildPage() {

var option;

var tbody = document.createElement('tbody');

var thead = document.createElement('thead');

tbody.setAttribute('id', 'tbody');

tbody.setAttribute('class', 'list');

var existingCars = CarRepository.getAllCars();

var currentTable = document.createElement('table');

currentTable.setAttribute('id', 'myTable');

var table = document.getElementById('table');

currentTable.setAttribute('class', 'table');

table.innerHTML = '';

var button;

var tr = document.createElement('tr');

var th = document.createElement('th');

th.appendChild(document.createTextNode('Модел'));

tr.appendChild(th);

th = document.createElement('th');

th.appendChild(document.createTextNode("Година"));

tr.appendChild(th);

th = document.createElement('th');

th.appendChild(document.createTextNode('Описание'));

tr.appendChild(th);

thead.appendChild(tr);

currentTable.appendChild(thead);

var existingCarsLength = existingCars.length;

for (var i = 0, tr, td; i < existingCarsLength; i++) {

tr = document.createElement('tr');

tr.setAttribute('id', i);

td = document.createElement('td');

td.setAttribute('class', 'model');

td.appendChild(document.createTextNode(existingCars[i].model));

tr.appendChild(td);

td = document.createElement('td');

td.setAttribute('class', 'year');

td.appendChild(document.createTextNode(existingCars[i].year));

tr.appendChild(td);

td = document.createElement('td');

td.setAttribute('class', 'description');

td.appendChild(document.createTextNode(existingCars[i].description));

tr.appendChild(td);

button = document.createElement('button');

button.appendChild(document.createTextNode('Edit'));

button.setAttribute('value', existingCars[i].id);

button.setAttribute('onclick', 'AppModule.editCar(value)');

td = document.createElement('td');

td.appendChild(button);

tr.appendChild(td);

button = document.createElement('button');

button.appendChild(document.createTextNode('X'));

button.setAttribute('value', existingCars[i].id);

button.setAttribute('onclick', 'AppModule.removeCar(value)');

button.setAttribute("class", "btn-danger");

td = document.createElement('td');

td.appendChild(button);

tr.appendChild(td);

tbody.appendChild(tr);

}

currentTable.appendChild(tbody);

table.appendChild(currentTable);

var descriptions = [];

var models = [];

var years = [];

for (var i = 0; i < existingCarsLength; ++i) {

if (!(contains(models, existingCars[i].model))) {

models.push(existingCars[i].model);

var modelsClass = document.getElementById('models');

option = document.createElement('option');

option.setAttribute('value', existingCars[i].model);

modelsClass.appendChild(option);

}

if (!(contains(years, existingCars[i].year))) {

years.push(existingCars[i].year);

var yearsClass = document.getElementById('years');

option = document.createElement('option');

option.setAttribute('value', existingCars[i].year);

yearsClass.appendChild(option);

}

}

var options = {

valueNames: ['model', 'year', 'description']

};

var userList = new List('myTable', options);

}

buildPage();

**За търсенето, добавянето, триенето и променянето на коли също е използван JS като може да бъде намерен в :**

1. **Car.js – модела който използваме за да създадем кола**

**Код:**

/\*\*

\* A module simulating the properties of a real life car with its model,

\* year of construction and description.

\* \*/

var Car = (function () {

    /\*\*

     \* Function simulating an auto incrementing static variable.

     \*/

    var incr = (function () {

    var i = 0;

    return function () {

    return i++;

    }

   })();

    function Car(model, year, description) {

        this.id = incr();

        this.model = model;

        this.year = year;

        this.description = description;

}

return Car;

} ());

1. **CarRepository.js - Реализацията на CRUD както и зареждането на началните стойности на коли в таблицата**

**Код:**

var CarRepository = (function () {

    this.carArray = [];

    /\*\*

     \* Constructor with four initial cars to display

     \* \*/

    function CarRepository() {

    }

    /\*\*

     \* Method used to add initial cars to the car repository.

     \* It first initialises four different cars and then adds

     \* them to the carArray that simulates the repository itself.

     \* \*/

    CarRepository.prototype.setupInitialCars = function () {

        var carOne = new Car("Honda", 2012, "yellow");

        var carTwo = new Car("VW", 2012, "red");

        var carThree = new Car("Mercedes", 2016, "black");

        var carFour = new Car("BMW", 2014, "yellow");

        this.carArray = [carOne, carTwo, carThree, carFour];

    }

    /\*\*

     \* Next four methods implement the CRUD functionality for the module.

     \* C = addCar

     \* R = getCar

     \* U = updateCar

     \* D = deleteCar

     \* \*/

    /\*\*

     \* Method used to add new car to the repository by its model, year and description.

     \* \*/

    CarRepository.prototype.addCar = function (model, year, description) {

        var carToAdd = new Car(model, +year, description);

        this.carArray.push(carToAdd);

    }

    /\*\*

     \* Method used to get a single car from the repository by its index.

     \* \*/

    CarRepository.prototype.getCar = function (id) {

        for (var i = 0; i < this.carArray.length; i++) {

            if (this.carArray[i].id == id) {

                return this.carArray[i];

            }

        }

    }

    /\*\*

     \* Method used to update a car by finding it by its index and then replacing its

     \* model,year and description with the new values.

     \* \*/

    CarRepository.prototype.updateCar = function (id, model, year, description) {

        for (var i = 0; i < this.carArray.length; i++) {

            if (this.carArray[i].id == id) {

                this.carArray[i].model = model;

                this.carArray[i].year = +year;

                this.carArray[i].description = description;

            }

        }

    }

    /\*\*

     \* Method used to delete a car by its index.

     \* \*/

    CarRepository.prototype.deleteCar = function (id) {

        for (var i = 0; i < this.carArray.length; i++) {

            if (this.carArray[i].id == id) {

                this.carArray.splice(i, 1);

            }

        }

    }

    /\*\*

     \*  Method that returns an array of all cars in the repository.

     \* \*/

    CarRepository.prototype.getAllCars = function () {

        return this.carArray;

    }

    return CarRepository;

} ());

1. **И последно AppModule.js – функциите за търсене и манипулация на данните**

**Код:**

/\*\*

\* Method that clears all the datalists and destroyes the table and then

\* rebuilds the page by calling the buildPage method with no filters.

\*/

function resetAllContent() {

$('#models').text('');

$('#years').text('');

$('#descriptions').text('');

$('#table').text('');

buildPage();

}

/\*\*

\* Function called onclick of the restore-table-btn that calls the resetAllContent method.

\*/

var clearChanges = document.getElementById('restore-table-btn')

clearChanges.onclick = function () {

resetAllContent();

}

/\*\*

\* Method called after the filter-by-model-btn is clicked.

\* It resets the content and rebuilds with the filter for models.

\* \*/

var filterByModelButton = document.getElementById('filter-by-model-btn');

filterByModelButton.onclick = function () {

resetAllContent();

var options = {

valueNames: ['model', 'year', 'description']

};

var userList = new List('myTable', options);

userList.search($('input[name=models]').val(), ['model']);

}

/\*\*

\* Method called after the filter-by-year-btn is clicked.

\* It resets the content and rebuilds with the filter for years.

\* \*/

var filterByYearButton = document.getElementById('filter-by-year-btn');

filterByYearButton.onclick = function () {

resetAllContent();

var options = {

valueNames: ['model', 'year', 'description']

};

var userList = new List('myTable', options);

userList.search($('input[name=years]').val(), ['year']);

}

/\*\*

\* Method called after the add-car-btn is clicked.

\* It gets the data from the add\_form and validates it.

\* if the information is valid it calls addCar with the values,

\* resets the form and resetsAllContent.

\* It resets the content and rebuilds with the filter for models.

\* \*/

var addCarButton = document.getElementById('add-car-btn');

addCarButton.onclick = function () {

var model = document.getElementById("add\_form").elements[0].value;

var year = document.getElementById("add\_form").elements[1].value;

var description = document.getElementById("add\_form").elements[2].value;

if (model == '' || year == '' || description == '' || year < 1800 || year > 2020 || isNaN(year)) {

alert("Can't have null values, or the year is invalid");

return;

}

CarRepository.addCar(model, year, description);

document.getElementById("add\_form").reset();

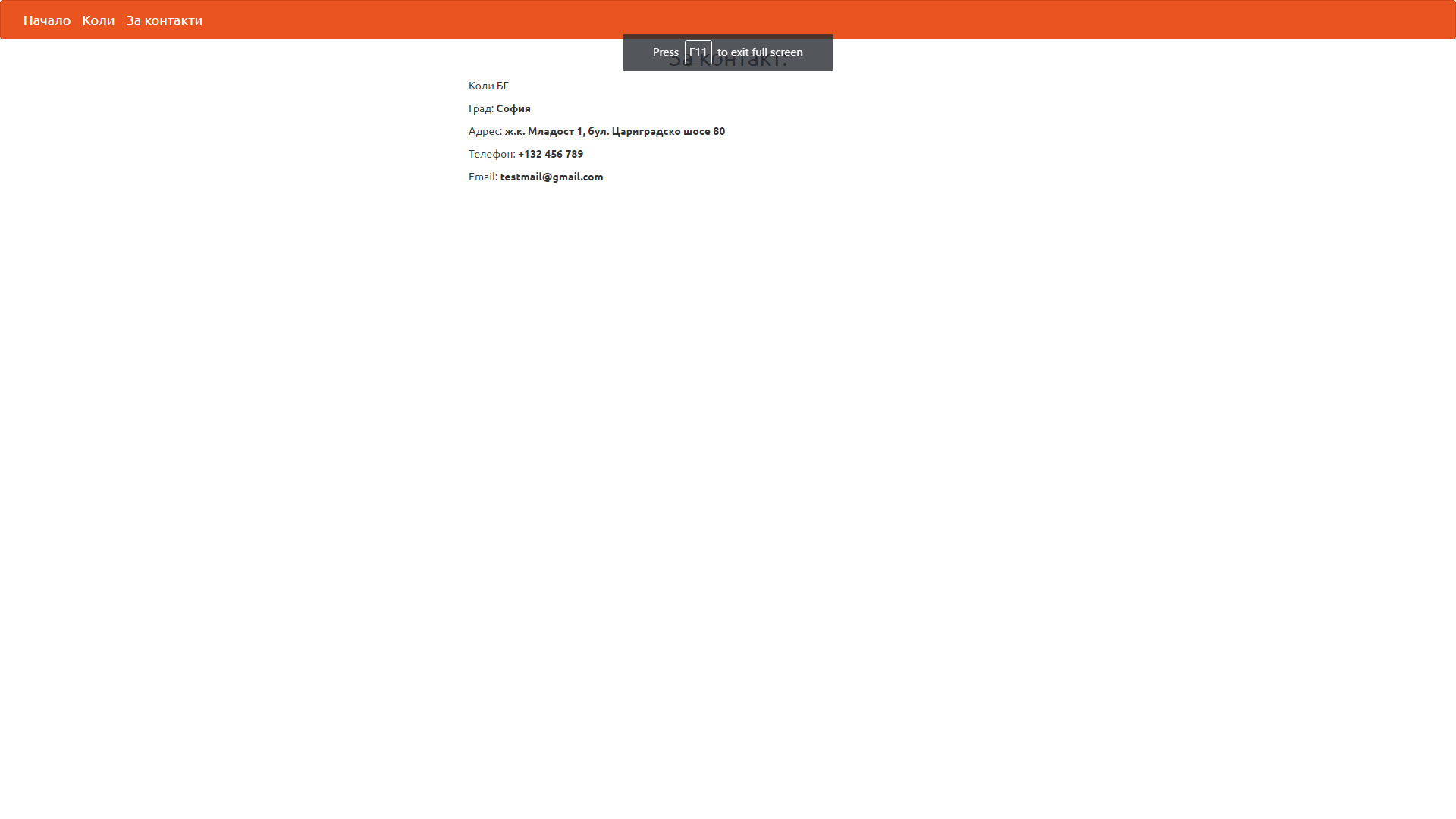
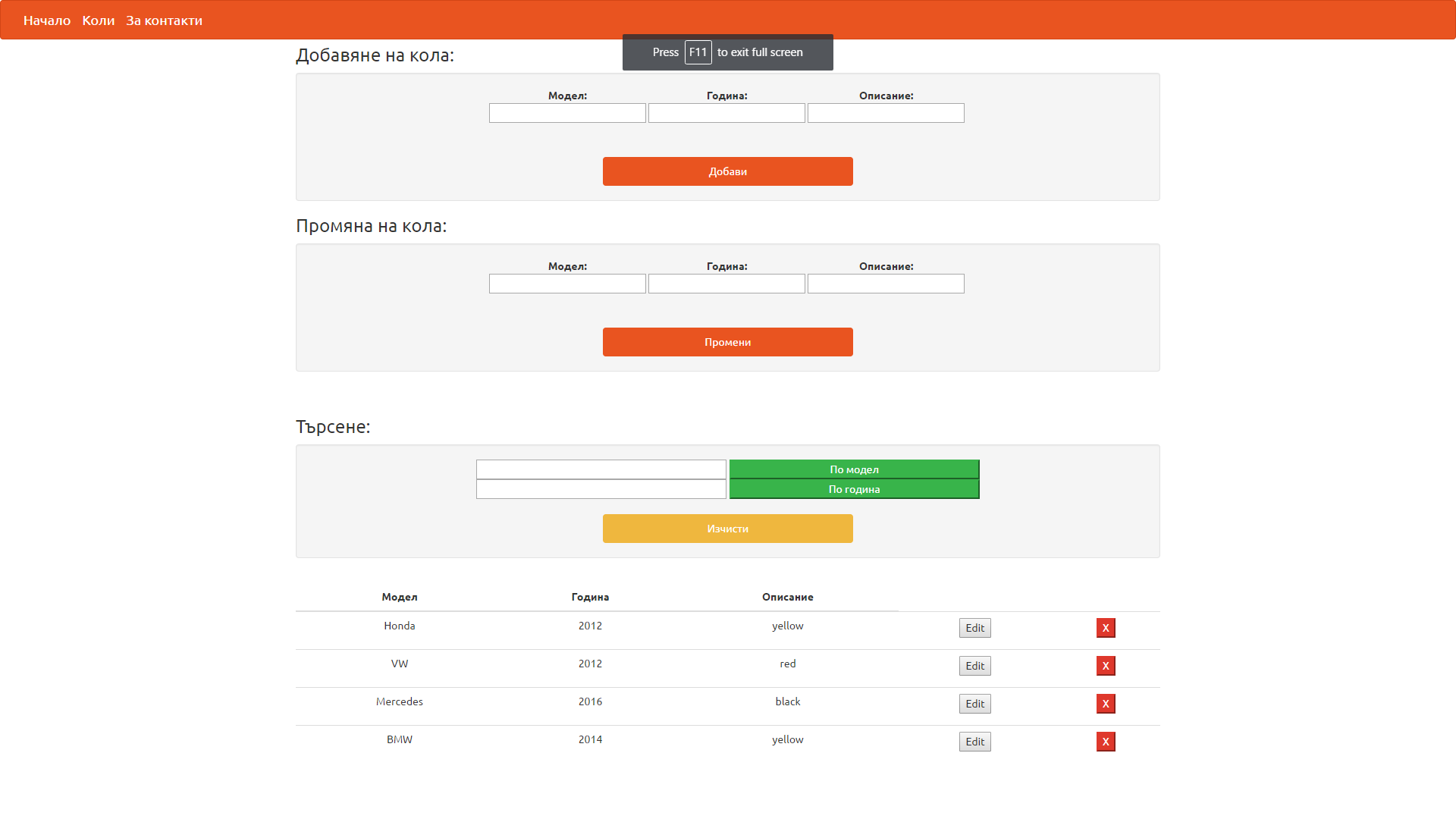
resetAllContent();

}

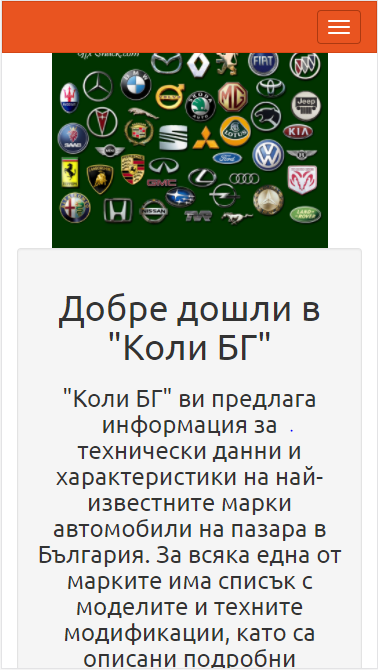
**5.Тестване . Преносимост, екран от различни типове устройства.**

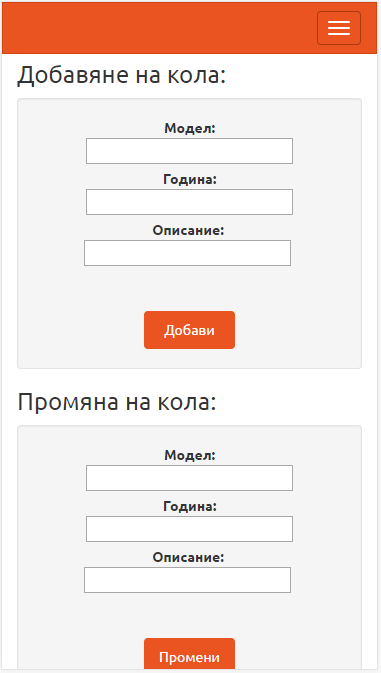
**Desktop:**

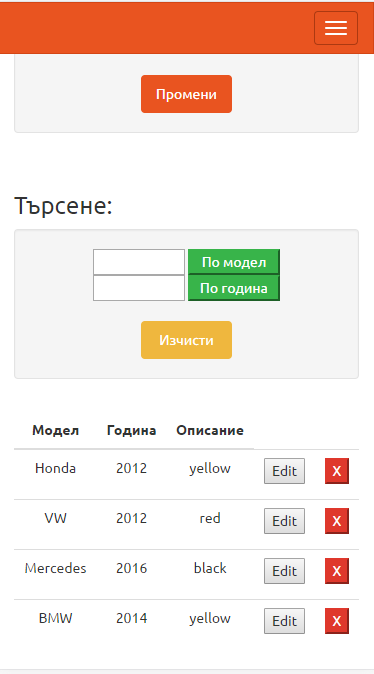




**iPhone 6:**







**iPad:**