



ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ
ФИЛИАЛ ПЛОВДИВ

ФАКУЛТЕТ ПО ЕЛЕКТРОНИКА И АВТОМАТИКА
КАТЕДРА “КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ”

КУРСОВА РАБОТА

Дисциплина: “Програмиране за Интернет”

Специалност: “Компютърни системи и технологии”

Образователно-квалификационна

степен: Магистър

Изготвил: Мухарем Ярмаджъ

Фак.№: 615039

Проверил: гл. ас. д-р

Добринка Петрова

Курсовата работа представлява уеб приложение, в което са реализирани знанията, получени по предмета Програмиране за Интернет. Те обхващат:

- HTML;
- CSS;
- JavaScript;
- AJAX;
- Back-end server language;
- RegEx.

Темата е приложение за проверка на автомобили.

Използваните технологии са XAMPP, PHP, HTML, CSS, AJAX, JS и Symfony фреймворк.

XAMPP е безплатен и свободен софтуерен пакет, който съдържа Apache HTTP Server, MySQL база от данни и други необходими инструменти за използване на езиците за програмиране PHP (с PEAR) и Perl. Към него е включен допълнителния модул - phpMyAdmin. Той е уеб базиран инструмент за администрация на MySQL. Написан е на PHP. С негова помощ могат да се извършват следните действия с бази данни и техните полета.

За изготвянето на сървърната страна на уеб приложението е избран скриптовият език PHP. PHP базираните сайтове могат да работят под всякаква операционна система като UNIX, Windows или Linux. Използва се предимно в уеб програмирането и е един от най-популярните езици в сферата.

Страниците и техния облик са изградени чрез HTML и CSS.

HTML е основният маркиращ език за описание и дизайн на уеб страници. Чрез него се задава основната структура на дадена уеб страница. Когато някой кликне на линк в страница или въведе нов URL в адресното поле, браузърът изпраща заявка за документ от Web server-а, където е качена страницата, посредством Hypertext Transport Protocol (HTTP). Върнатата от сървъра страница е подредена посредством HTML структура.

CSS е език за описание на стилове. Обикновено се използва съвместно с HTML. Създаден е с цел да се раздели структурата и съдържанието на уеб страниците от тяхното оформление и визуално представяне.

JavaScript е интерпретируем език за програмиране, разпространяван с повечето уеб браузъри. Най-често се прилага към Интернет страница с цел добавяне на функционалност. JavaScript е програмен език, който позволява динамична промяна на поведението на браузъра в рамките на дадена HTML страницата. JavaScript функциите обикновено се свързват със събития на страницата - движение/натискане на мишката, клавиатурата или елемент от страницата, и други потребителски действия. Една от възможностите му е зареждане на данни чрез AJAX.

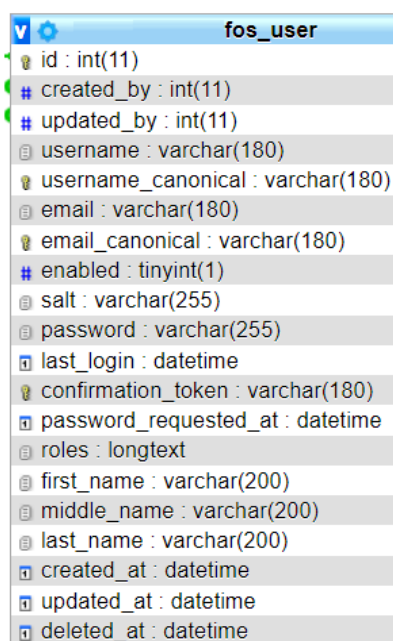
AJAX е похват в уеб разработките за създаване на интерактивни уеб приложения. Предимството на *Ajax* е, че посредством използването му уеб страниците се зареждат по-бързо. Посредством асинхронен обмен на малки порции данни „зад кадър“ могат да се променят само частично информации на уеб страницата. По този начин се намалява количеството информация, която се трансферира между сървъра и клиента.

Symfony framework е рамка за приложения с отворен код (open source framework), разработен от френската фирма Sensio Labs. Разработен е за да задоволи нуждите на фирмата, след което е пуснат под GPL лиценз за свободно използване. Symfony е web framework и е написан на езика PHP. За основа е използван Ruby on Rails и както него, следва принципите на MVC (model – view – controller). Symfony е съвкупност от PHP компоненти, framework, за изграждане на уеб приложения, философия и общност - всичко работещо в хармония. Symfony е един от водещите фреймуърци за изграждането на уеб сайтове от всякакво ниво на сложност. Той съдържа в себе си множество компоненти, които позволяват на разработчиците да изграждат своите проекти по-бързо и надеждно. Голямата общност от разработчици, които използват Symfony, дава

необходимата стабилна основа на framework-a, като осигуряват нужната подкрепа и голям брой готови решения.

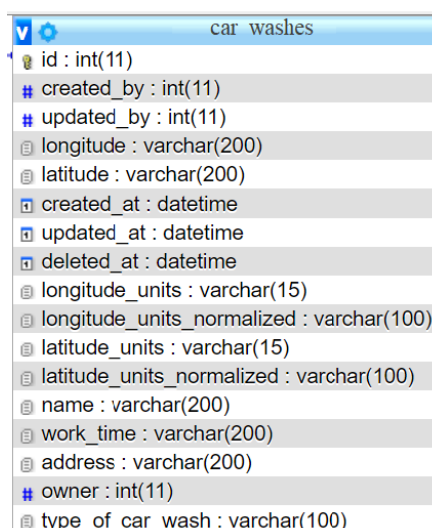
Базата данни, която използва приложението – Car Wash се състои от две таблици fos_user и car_washes, свързани с външен ключ.

Първата таблица fos_user – Фиг.1, съхранява потребителите на уеб приложението. Те се запазват в тази база данни след успешна регистрация, за която е добавена валидация, написана на JS, за проверка на празни полета и регулярни изрази за парола и валиден имейл адрес. А на втората таблица се записват данните за автомобилите – Фиг.2.



fos_user	
id	int(11)
created_by	int(11)
updated_by	int(11)
username	varchar(180)
username_canonical	varchar(180)
email	varchar(180)
email_canonical	varchar(180)
enabled	tinyint(1)
salt	varchar(255)
password	varchar(255)
last_login	datetime
confirmation_token	varchar(180)
password_requested_at	datetime
roles	longtext
first_name	varchar(200)
middle_name	varchar(200)
last_name	varchar(200)
created_at	datetime
updated_at	datetime
deleted_at	datetime

Фиг.1. Структура на таблица fos_user.

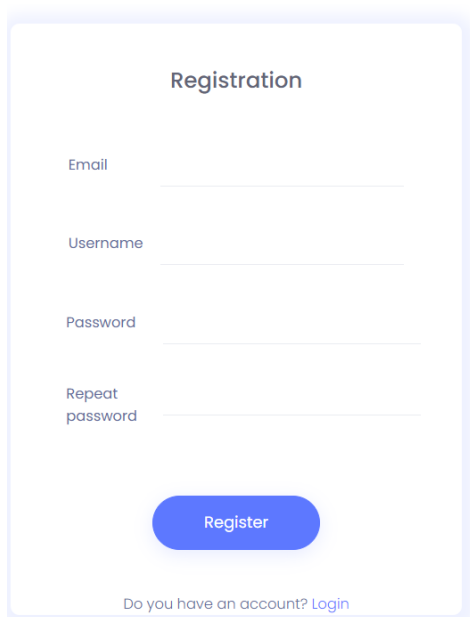


car_washes	
id	int(11)
created_by	int(11)
updated_by	int(11)
longitude	varchar(200)
latitude	varchar(200)
created_at	datetime
updated_at	datetime
deleted_at	datetime
longitude_units	varchar(15)
longitude_units_normalized	varchar(100)
latitude_units	varchar(15)
latitude_units_normalized	varchar(100)
name	varchar(200)
work_time	varchar(200)
address	varchar(200)
owner	int(11)
type_of_car_wash	varchar(100)

Фиг.2. Структура на таблица car washes

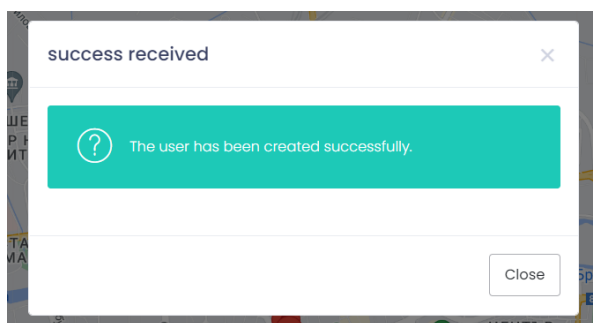
Типът на системата за управление на бази от данни е релационна система. Действията с базата данни, които се осъществяват по време на работата на приложението са – записване на данни, четене на данни, промяна на данни.

При регистрация в приложението се попълват всичките полета: имейл, потребителско име, парола, повторение на паролата. След попълване на всички полета, при което трябва да има уникалност на полетата „Username“ и „Email“ , а полетата „password“ и „repeat password“ са идентични. При натискане на бутона „Register“, потребителя успешно извършва регистрация и се добавя в базата от данни, при неправилно въвеждане на данни не се осъществява регистрация, дава се сигнал за невалидност или се нулират данните от някои полета, с което се показва, че трябва да се поправи допуснатата грешка.

A registration form titled "Registration" with a light blue border. It contains four input fields: "Email", "Username", "Password", and "Repeat password". Below the fields is a blue "Register" button. At the bottom, there is a link: "Do you have an account? [Login](#)".

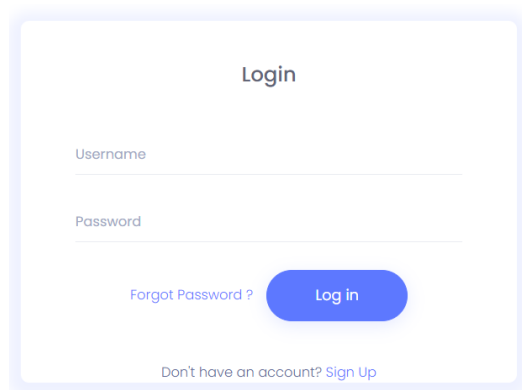
Фиг.3. Форма за регистрация.

При успешна регистрация в приложението се появява следното съобщение :



Фиг.4. Съобщение за успешна регистрация.

Когато се вписва в приложението информацията, която се попълва, трябва да се съдържа в базата от данни след успешна регистрация. Осъществява се потвърждение на данните от потребителя. Попълват се „username“ и „password“, които дават достъп до приложението. При неправилно въведени данни се дава сигнал. При забравена парола, потребителя избира „Forgot Password“ след което се отваря формата за обновяване на паролата. Потребителят трябва да попълни празното поле с неговото потребителско име или имейл запазени в приложението.



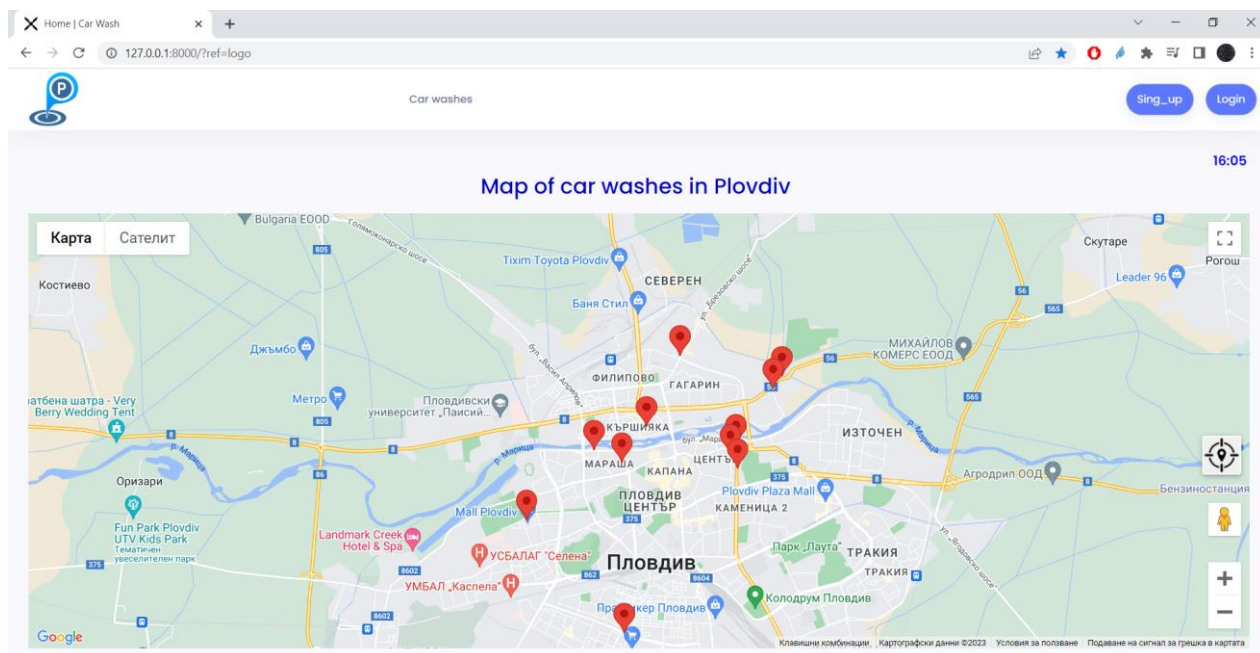
The login form is titled "Login" and is enclosed in a light blue border. It contains two input fields: "Username" and "Password". Below the "Password" field is a blue "Log in" button. To the left of the button is a link "Forgot Password?". At the bottom of the form, there is a link "Don't have an account? Sign Up".

Фиг.5. Форма за вписване в приложението.

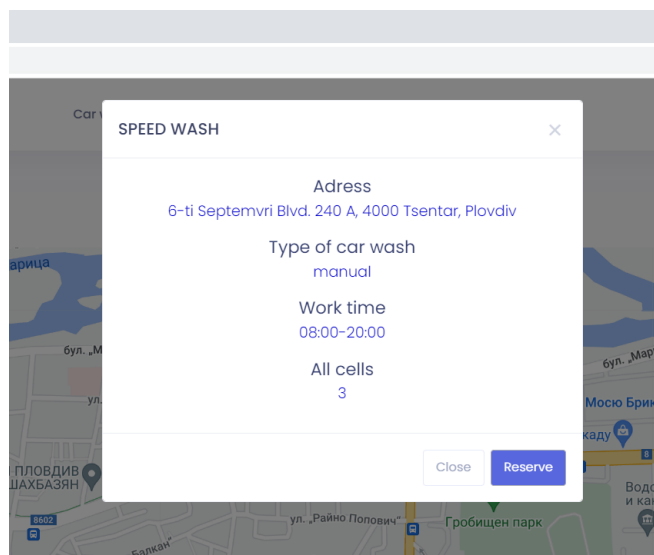
На началната страница на приложението е изобразена картата на град Пловдив.

На картата с червени пинчета са изобразени наличните автомивки за града.

Всяка от тях може да се кликне, а като се ховърне с мишката се появява името на посочения паркинг. При еднократно кликане върху пина, от изобразените на картата, се появява малък информационен прозорец за кликнатата автомивка със следната информация: името на автомивката, типа му, адреса, работното му време и общия брой клетки за ползване.

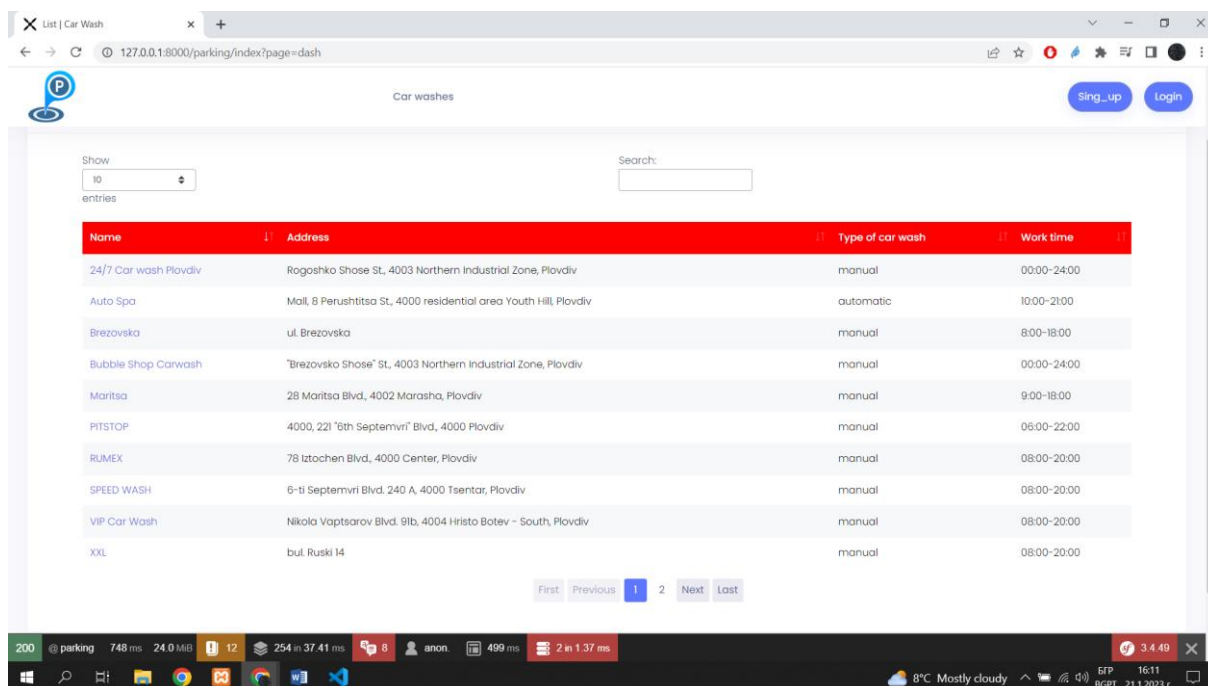


Фиг.6. Начална страница на приложението.



Фиг.7. Информационен прозорец за избраната автомивка.

В началната страница на приложението – Фигура 6, срещу логото се намира бутон с надпис „Car washes“. При кликане се отваря нов раздел със списъка от автомивки, кито са обозначени на картата. Списъка се формира от 3 колони: името на паркинга, адреса и работното му време. От „Search“ полето може да се търсят паркинги.



index.html.twig

```
{% extends 'base.html.twig' %}

{% block title %}{{ 'Home | Car Wash'|trans }}{% endblock %}

{% block content_breadcrumb %}
<a href="{{ path('homepage') }}" class="k-content__head-breadcrumb-link">{{ 'Home
page'|trans }}</a>
<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.css">
{% endblock content_breadcrumb %}

{% block content_body %}
<body>

    <script>
        var data = {{ info|json_encode|raw }};
    </script>
    <div>
        <p id='title-map'>Map of car washes in Plovdiv</p>
    </div>

    <div id="map"></div>

    <div id="floating-panel">
        <div id="map-canvas"></div>
        <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel"
            aria-hidden="true">
            <div class="modal-dialog">
```

```
<div class="modal-content">
  <div class="modal-header">
    <h4 class="modal-title" id="myModalLabel"></h4>
    <button type="button" class="close" data-dismiss="modal"><span aria-
hidden="true">×</span><span
      class="sr-only">Close</span></button>
  </div>
  <div class="modal-body">
    <div class="content"></div>
  </div>
  <div class="modal-footer">
    <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
    <a href="/reservation" class=" btn btn-primary " id="reserve-btn">
      Reserve
    </a>
  </div>
</div>
</div>
</div>
```

```
{% javascripts
  filter='uglifyjs2'
  '@AppBundle/Resources/public/js/map.js'

% }

<script type="text/javascript" src="{{ asset_url }}"?r={{ ver }}"></script>
{% endjavascripts %}
```

</body>

{% endblock %}

{% block stylesheets %}

<style>

```
#title-map {  
    margin-left: 550px;  
    font-size: 26px;  
    font-weight: 500;  
}
```

```
#map {  
    width: 100%;  
    height: 100%;  
}
```

```
.error {  
    border-block-color: red;  
}
```

```
.modal-h {  
    font-size: large;  
    font-weight: bolder;  
}
```

;

</style>

```
{% endblock % }
```

```
{% block javascripts % }
```

```
<script async defer
```

```
src="https://maps.googleapis.com/maps/api/js?key= "></script>
```

```
{% endblock % }
```

carwash.php

```
<?php
```

```
namespace carWashBundle\Entity;
```

```
use Doctrine\ORM\Mapping as ORM;
```

```
use Symfony\Component\Validator\Constraints as Assert;
```

```
use Doctrine\Common\Collections\ArrayCollection;
```

```
use Gedmo\Mapping\Annotation as Gedmo;
```

```
use FBaseBundle\Traits\BlameStampableEntity;
```

```
use FBaseBundle\Traits\SoftDeleteableEntity;
```

```
/**
```

```
 * carWash
```

```
 *
```

```
 * @ORM\Table(name="carWash")
```

```
 * @ORM\Entity(repositoryClass="carWashBundle\Repository\carWashRepository")
```

```
 * @Gedmo\SoftDeleteable(fieldName="deletedAt")
```

```
 */
```

```
class carWash
```

```
{
```

```
use BlameStampableEntity;
```

```
use SoftDeleteableEntity;
```

```
/**
```

```
 * @var int
```

```
 *
```

```
 * @ORM\Column(name="id", type="integer")
```

```
 * @ORM\Id
```

```
 * @ORM\GeneratedValue(strategy="AUTO")
```

```
 */
```

```
private $id;
```

```
/**
```

```
 * @ORM\ManyToOne(targetEntity="UsersBundle\Entity\User")
```

```
 * @ORM\JoinColumn(name="owner", referencedColumnName="id",  
nullable=false)
```

```
 * @Assert\NotBlank(message="carWash.form.error.owner_is_blank")
```

```
 */
```

```
private $owner;
```

```
/**
```

```
 * @ORM\Column(type="string", length=200, nullable=true)
```

```
 * @Assert\NotBlank
```

```
 */
```

```
private $name;
```

```
/**
```

```
 * @ORM\Column(type="string", length=200, nullable=true)
```

```
 * @Assert\Length(max = 70)
```

```
* @Assert\NotBlank
*/

private $address;

/**
 * @ORM\Column(type="string", length=200, nullable=true)
 * @Assert\Length(max = 70)
 */
private $longitude;

/**
 * @ORM\Column(type="string", length=15, nullable=true)
 */
private $longitude_units;

/**
 * @ORM\Column(type="string", length=100, nullable=true)
 */
private $longitude_units_normalized;

/**
 * @ORM\Column(type="string", length=200, nullable=true)
 * @Assert\Length(max = 70)
 */
private $latitude;

/**
 * @ORM\Column(type="string", length=15, nullable=true)
 */
```

```
private $latitude_units;
```

```
/**
```

```
 * @ORM\Column(type="string", length=100, nullable=true)
```

```
 */
```

```
private $latitude_units_normalized;
```

```
/**
```

```
 * @ORM\Column(type="string", length=200, nullable=true)
```

```
 * @Assert\Length(max = 70)
```

```
 */
```

```
private $workTime;
```

```
/**
```

```
 * @ORM\OneToMany(targetEntity="carWashBundle\Entity\carWashSpaces",  
mappedBy="carWash", orphanRemoval=true, cascade={"persist"})
```

```
 * @Assert\Valid()
```

```
 */
```

```
private $spaces;
```

```
/**
```

```
 * @ORM\Column(type="string", length=100, nullable=true)
```

```
 */
```

```
private $typeOfCarWash;
```

```
/**
```

```
 * Set typeOfCarWash
```

```
 *
```

```
 * @param string $typeOfCarWash
```

```
*  
  
* @return carWash  
  
*/  
  
public function setTypeOfCarWash($typeOfCarWash)  
{  
    $this->typeOfCarWash = $typeOfCarWash;  
  
    return $this;  
}  
  
/**  
 * Get typeOfCarWash  
 *  
 * @return string  
 */  
  
public function getTypeOfCarWash()  
{  
    return $this->typeOfCarWash;  
}  
  
  
public function __construct()  
{  
    $this->spaces = new ArrayCollection();  
}  
  
  
public function __toString()  
{  
    return $this->name;
```



```
}
```

```
/**
```

```
 * Get id.
```

```
 *
```

```
 * @return int
```

```
 */
```

```
public function getId()
```

```
{
```

```
    return $this->id;
```

```
}
```

```
public function getOwner()
```

```
{
```

```
    return $this->owner;
```

```
}
```

```
public function setOwner($owner)
```

```
{
```

```
    $this->owner = $owner;
```

```
    return $this;
```

```
}
```

```
/**
```

```
 * Set Name
```

```
 *
```

```
 * @param string $name
```

```
 *
```

```
* @return carWash
*/

public function setName($name)
{
    $this->name = $name;

    return $this;
}

/**
 * Get Name
 *
 * @return string
 */
public function getName()
{
    return $this->name;
}

/**
 * Set address
 *
 * @param string $address
 *
 * @return carWash
 */
public function setAddress($address)
{
    $this->address = $address;
```

```
        return $this;
    }

    /**
     * Get address
     *
     * @return string
     */
    public function getAddress()
    {
        return $this->address;
    }

    /**
     * Set longitude
     *
     * @param string $longitude
     *
     * @return carWash
     */
    public function setlongitude($longitude)
    {
        $this->longitude = $longitude;

        return $this;
    }

    /**
```

```
* Get longitude
*
* @return string
*/
public function getlongitude()
{
    return $this->longitude;
}

/**
 * Set latitude
 *
 * @param string $latitude
 *
 * @return carWash
 */
public function setlatitude($latitude)
{
    $this->latitude = $latitude;

    return $this;
}

/**
 * Get latitude
 *
 * @return string
 */
public function getlatitude()
```

```
{
    return $this->latitude;
}

/***** latitudeUnits
***** */

/**
 * Set latitudeUnits.
 *
 * @param string $latitudeUnits
 *
 * @return carWash
 */
public function setLatitudeUnits($latitudeUnits)
{
    $this->latitude_units = $latitudeUnits;

    return $this;
}

/**
 * Get latitudeUnits.
 *
 * @return string
 */
public function getLatitudeUnits()
{
    return $this->latitude_units;
}
```

```
}
```

```
/****** longitudeUnits
```

```
***** */
```

```
/**
```

```
* Set longitudeUnits.
```

```
*
```

```
* @param string $longitudeUnits
```

```
*
```

```
* @return carWash
```

```
*/
```

```
public function setLongitudeUnits($longitudeUnits)
```

```
{
```

```
    $this->longitude_units = $longitudeUnits;
```

```
    return $this;
```

```
}
```

```
/**
```

```
* Get latUnits.
```

```
*
```

```
* @return string
```

```
*/
```

```
public function getLongitudeUnits()
```

```
{
```

```
    return $this->longitude_units;
```

```
}
```

```
/******
```

```
***** */
```

```
/**
```

```
 * Set latUnitsNormalized.
```

```
 *
```

```
 * @param string|null $latUnitsNormalized
```

```
 *
```

```
 * @return carWash
```

```
 */
```

```
public function setLatUnitsNormalized($latUnitsNormalized = null)
```

```
{
```

```
    $this->lat_units_normalized = $latUnitsNormalized;
```

```
    return $this;
```

```
}
```

```
/**
```

```
 * Get latUnitsNormalized.
```

```
 *
```

```
 * @return string|null
```

```
 */
```

```
public function getLatUnitsNormalized()
```

```
{
```

```
    return $this->lat_units_normalized;
```

```
}
```

```
/** ***** Longitude
```

```
***** */
```

```
/**
```

```
 * Set longitudeUnitsNormalized.
```

```
 *
```

```
 * @param string|null $longitude_units_normalized
```

```
 *
```

```
 * @return carWash
```

```
 */
```

```
public function setLongitudeUnitsNormalized($longitudeUnitsNormalized = null)
```

```
{
```

```
    $this->longitude_units_normalized = $longitudeUnitsNormalized;
```

```
    return $this;
```

```
}
```

```
/**
```

```
 * Get longitudeUnitsNormalized.
```

```
 *
```

```
 * @return string|null
```

```
 */
```

```
public function getLongitudeUnitsNormalized()
```

```
{
```

```
    return $this->longitude_units_normalized;
```

```
}
```

```
/** ***** Latitude
```

```
***** */
```

```
/**
```



```
* Set latitudeUnitsNormalized.
```

```
*
```

```
* @param string|null $latitude_units_normalized
```

```
*
```

```
* @return carWash
```

```
*/
```

```
public function setLatitudeUnitsNormalized($latitudeUnitsNormalized = null)
```

```
{
```

```
    $this->latitude_units_normalized = $latitudeUnitsNormalized;
```

```
    return $this;
```

```
}
```

```
/**
```

```
* Get latitudeUnitsNormalized.
```

```
*
```

```
* @return string|null
```

```
*/
```

```
public function getLatitudeUnitsNormalized()
```

```
{
```

```
    return $this->latitude_units_normalized;
```

```
}
```

```
/**
```

```
*/
```

```
/**
```

```
* Set lngUnitsNormalized.
```

```
*
```

```
* @param string|null $lngUnitsNormalized
*
* @return carWash
*/
public function setLngUnitsNormalized($lngUnitsNormalized = null)
{
    $this->lng_units_normalized = $lngUnitsNormalized;

    return $this;
}

/**
 * Get lngUnitsNormalized.
 *
 * @return string|null
 */
public function getLngUnitsNormalized()
{
    return $this->lng_units_normalized;
}

/**
 * Set workTime
 *
 * @param string $workTime
 *
 * @return carWash
 */
public function setWorkTime($workTime)
```

```
{  
    $this->workTime = $workTime;  
  
    return $this;  
}  
  
/**  
 * Get workTime  
 *  
 * @return string  
 */  
public function getWorkTime()  
{  
    return $this->workTime;  
}  
  
/**  
 * Add spaces.  
 *  
 * @param \carWashBundle\Entity\carWashSpaces $spaces  
 *  
 * @return carWash  
 */  
public function addSpace(carWashSpaces $space)  
{  
    $space->setcarWash($this);  
    $this->spaces[] = $space;  
  
    return $this;  
}
```

```
}

/**
 * Remove spaces.
 *
 * @param \carWashBundle\Entity\carWashSpaces $spaces
 *
 * @return boolean TRUE if this collection contained the specified element, FALSE
otherwise.
 */
public function removeSpace(carWashSpaces $spaces)
{
    return $this->spaces->removeElement($spaces);
}

/**
 * Get spaces.
 *
 * @return \Doctrine\Common\Collections\Collection
 */
public function getSpaces()
{
    return $this->spaces;
    /*return $this->spaces->filter(function($contact) {
        return $contact->getContactType() && $contact->getContactType()-
>getType()->getNameKey()=='user.spaces';
    });*/
}
```

```
/**
 * Add rezervation.
 *
 * @param \carWashBundle\Entity\Reservation $rezervation
 *
 * @return carWash
 */
public function addReservation(Reservation $rezervation)
{
    $rezervation->setcarWash($this);
    $this->rezervation[] = $rezervation;

    return $this;
}

/**
 * Remove rezervation.
 *
 * @param \carWashBundle\Entity\Reservation $rezervation
 *
 * @return boolean TRUE if this collection contained the specified element, FALSE
otherwise.
 */
public function removeReservation(Reservation $rezervation)
{
    return $this->rezervation->removeElement($rezervation);
}

/**
```

```
* Get rezervation.
*
* @return \Doctrine\Common\Collections\Collection
*/
public function getReservation()
{
    return $this->rezervation;

    /*return $this->spaces->filter(function($contact) {
        return $contact->getContactType() && $contact->getContactType()-
>getType()->getNameKey()=='user.spaces';
    });*/
}

public function getFreeSpaces()
{
    return $this->spaces->filter(function($space) {
        return $space->getStatus()=='available';
    });
}

public function getBusySpaces()
{
    return $this->spaces->filter(function($space) {
        return $space->getStatus()=='busy';
    });
}

public function getSpaceNum()
{

```

```
return $this->spaces->filter(function($space) {  
    return $space->getPlaceNumber();  
});  
}  
}
```

map.js

```
let infoWindow1;
```

```
let pos;
```

```
function initMap() {  
    let map = new google.maps.Map(document.getElementById("map"), {  
        center: { lat: 42.15574631218075, lng: 24.742712542086 },  
        zoom: 13,  
        clickableIcons: true  
        //disableDefaultUI: !0  
    });
```

```
for (let k in data) {  
    let marker = new google.maps.Marker({  
        animation: google.maps.Animation.DROP,  
        position: { lat: data[k].lat, lng: data[k].long },  
        map: map,  
        title: data[k].name  
    });
```

```
const infowindow = new google.maps.InfoWindow({  
    content: ("<div id='content'> " + data[k].name + "</div>")
```

```
});
```

```
marker.addListener("mouseover", (e) => {
```

```
    infowindow.open({
```

```
        anchor: marker,
```

```
        map,
```

```
    })
```

```
});
```

```
marker.addListener("click", (e) => {
```

```
    let title = document.getElementById('myModalLabel').textContent =  
data[k].name;
```

```
    let templ = $('<div>');
```

```
    templ.html("");
```

```
    templ.append("<h>Address</h>");
```

```
    templ.append("<p> " + data[k].address + "</p>");
```

```
    templ.append("<h>Type of car wash</h>");
```

```
    templ.append("<p> " + data[k].typeOfCW + "</p>");
```

```
    templ.append("<h>Work time</h>");
```

```
    templ.append("<p>" + data[k].workTime + "</p>");
```

```
    templ.append("<h>All cells</h>");
```

```
    templ.append("<p>" + data[k].allSpaces + "</p>");
```

```
    $("h").addClass("modal-h");
```

```
    $("p").addClass("par");
```

```
    $('#reserve-btn').attr('href', "/reservation/" + data[k].parkingID);
```



```
$('#myModal').modal('show');

map.setZoom(16);
map.setCenter(marker.getPosition());

});
}

const centerControlDiv = document.createElement("div");
// Create the control.
const centerControl = createCenterControl(map);
// Append the control to the DIV.
centerControlDiv.appendChild(centerControl);
map.controls[google.maps.ControlPosition.RIGHT_CENTER].push(centerControl
Div);
infoWindow1 = new google.maps.InfoWindow();
}

window.initMap = initMap;

function createCenterControl(map) {
  const locationButton = document.createElement("button");

  // Set CSS for the control.
  locationButton.type = "button";
  locationButton.style.backgroundColor = "FFF";
  locationButton.style.border = "2px solid #fff";
  locationButton.style.borderRadius = "3px";
  locationButton.style.boxShadow = "0 2px 6px rgba(0,0,0,.3)";
```

```
locationButton.style.color = "rgb(25,25,25)";
locationButton.style.margin = "8px 0 22px";
locationButton.style.marginRight = "10px";
locationButton.style.marginTop = "80px";
locationButton.style.padding = '22px';
locationButton.title = "Click to find your location";
locationButton.style.backgroundImage = "url('https://img.icons8.com/external-smashingstocks-glyph-smashing-stocks/44/external-Current-Location-seo-and-marketing-smashingstocks-glyph-smashing-stocks.png')";

// Setup the click event listeners: simply set the map to Chicago.
locationButton.addEventListener("click", () => {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
      (position) => {
        pos = {
          lat: position.coords.latitude,
          lng: position.coords.longitude,
        };

        marker = new google.maps.Marker({
          animation: google.maps.Animation.DROP,
          position: pos,
          map: map,
          icon: 'https://img.icons8.com/external-smashingstocks-flat-smashing-stocks/41/external-car-transport-smashingstocks-flat-smashing-stocks-4.png',
          title: 'My location'
        });
```

```
        infoWindow1.setPosition(pos);
        map.setZoom(16);
        map.setCenter(pos);
    },
);

() => {
    handleLocationError(true, infoWindow1, map.getCenter());

}
} else {
    // Browser doesn't support Geolocation
    handleLocationError(false, infoWindow1, map.getCenter());
}
});

return locationButton;
}
```