

# Lumina Weather

*Product Requirements Document*

*version 1.0*

1. Technologies list.....	3
2. Development part.....	4
2.1 Appearance and design.....	4
2.1.1 Main page.....	4
2.1.2 Desktops, tablets and mobiles.....	6
2.2 Server side features.....	9
2.2.1 Main class LuminaWeather_API.....	9
2.2.2. Main controller Weather\IndexController..	11
2.2.3 Multilang possibilities .....	11
2.3 Client side features.....	12
2.3.1 Getting coordinates..	12
2.3.2 Getting weather by city name..	13
2.3.4 Getting all data..	14
2.3.5 Videos. ....	15
2.3.6 Latest news . ....	15
3. Annotation.....	16

# 1 Technologies list

For this exercise I'll use the next set of technologies

- Laravel 5.1 - base back-end framework
  - Angular JS - main javascript client-side framework
  - HTML5 – for project view skeleton
  - CSS3 – for style sheets
  - JQuery – javascript front-end library
  - Bootstrap3 – for responsive and attractive design
- APIs(<http://openweathermap.org/>, <http://maps.googleapis.com/maps/>) – for third-party functionalities.

PHP – is main programming language.

Let me explain, why I decided use this list of technologies

## Laravel

Laravel is modern and powerful PHP framework. It provides excellent architecture for project, comfortable services and best routing system. Besides, Laravel has a lot of ready functions and tools.

## Angular JS

This Javascript framework can work with other JS libraries and files. Besides, in the end of development you'll get clear and understandable code.

## Bootstrap

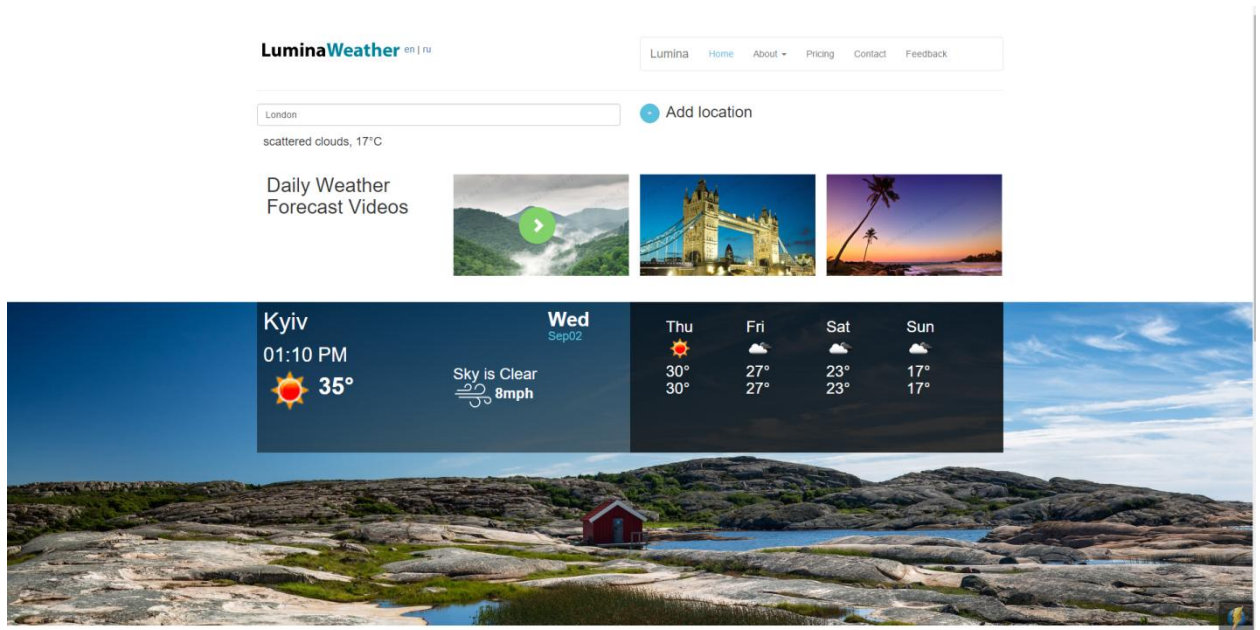
It's one of the best tools for responsive and smart design. A lot of things have been already written so I can use it with great pleasure.

## 2. Development part

### 2.1 Appearance and design

#### 2.1.1 Main page

First of all, I want to say, that WHOLE project's code, functional, solutions (HTML, CSS, Javascript, PHP etc) are fully mine. Whole my code is commented and clear.

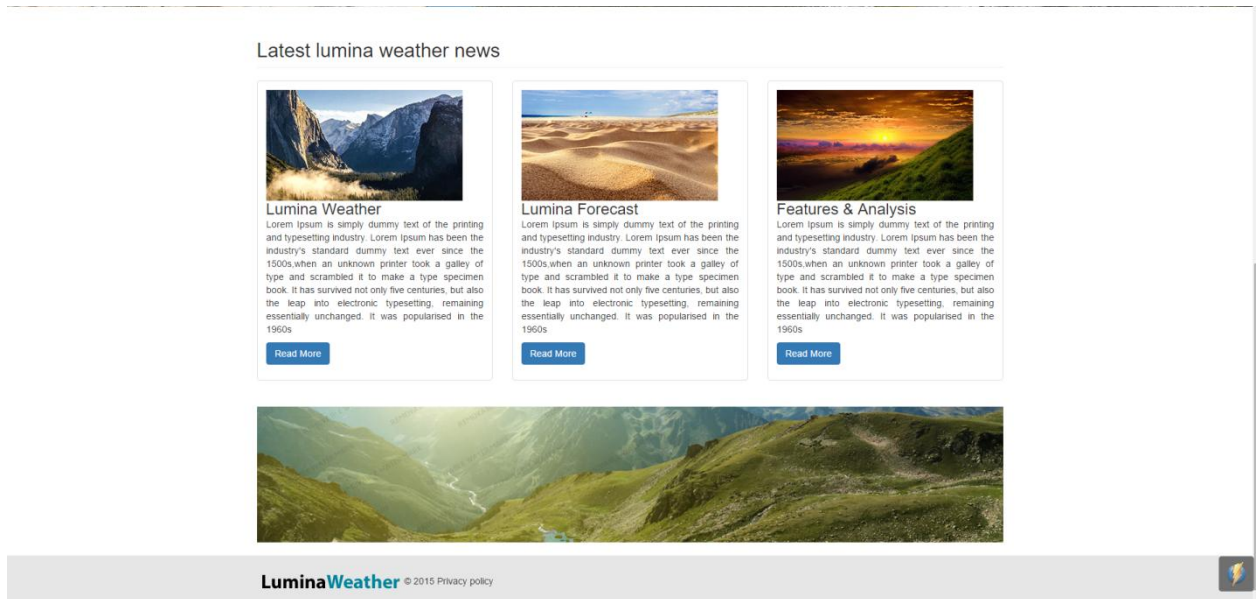


Pic.1 Main page (Page top)

Design for this exercise is simple, but rather attractive. We have

- Logo
- Main menu
- Input field for printing name of city
- Videos which can be used for showing weather features
- Two main blocks for displaying current and future weather
- Latest news about weather
- Logo/Banner/Advertisement
- Footer

All of this, you can see on Pic.1 and Pic.2



Pic.2 Main page (Page bottom)

About one specialty – background picture is automatically changed depending on times of day. You can see it on Pic.3. It's small thing, but it makes web-service more attractive.



Pic.3 Main Page (Evening time)

This features I did with JQuery (simple selector) and Javascript (new Date()).

### 2.1.2 Desktops, tablets and mobiles

As I said, I created responsive design for our service and we can tested it on different devices.

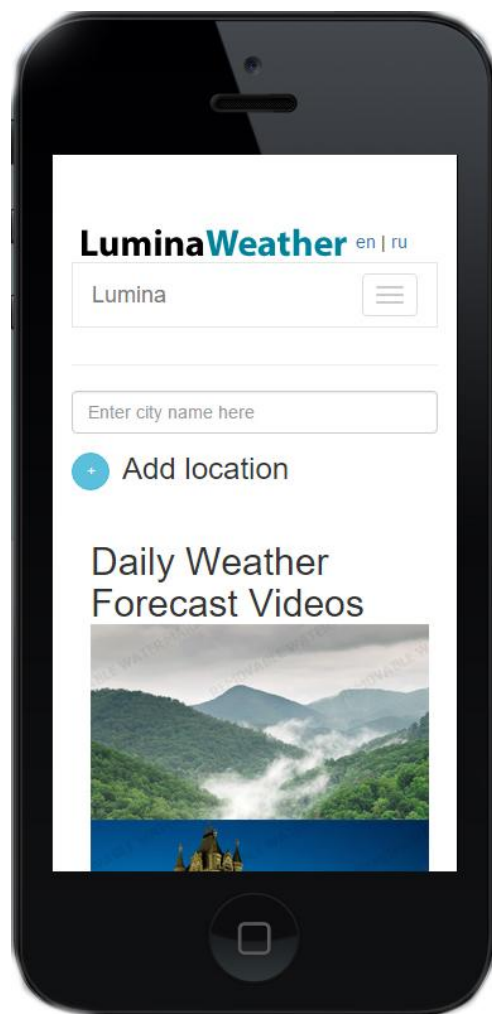
#### **Apple iPhone 5**

Css resolution 320x568

Pixel resolution 640x1136

Operating system iOS 6.0

Device type Phone



Pic 4 Main page via Apple iPhone

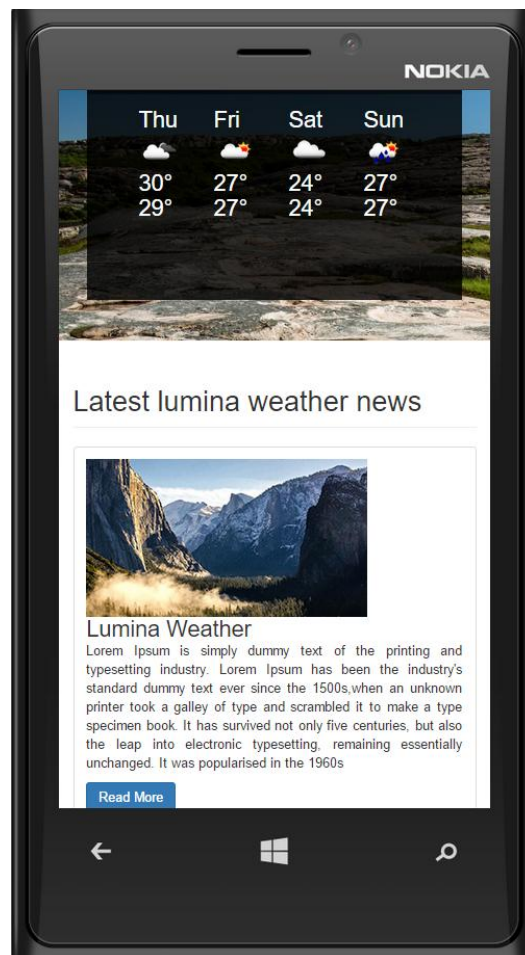
## Nokia Lumia 920

Css resolution 460x768

Pixel resolution 768x1280

Operating system Windows Phone 8

Device type Phone



Pic 5 Main page via Nokia Lumia

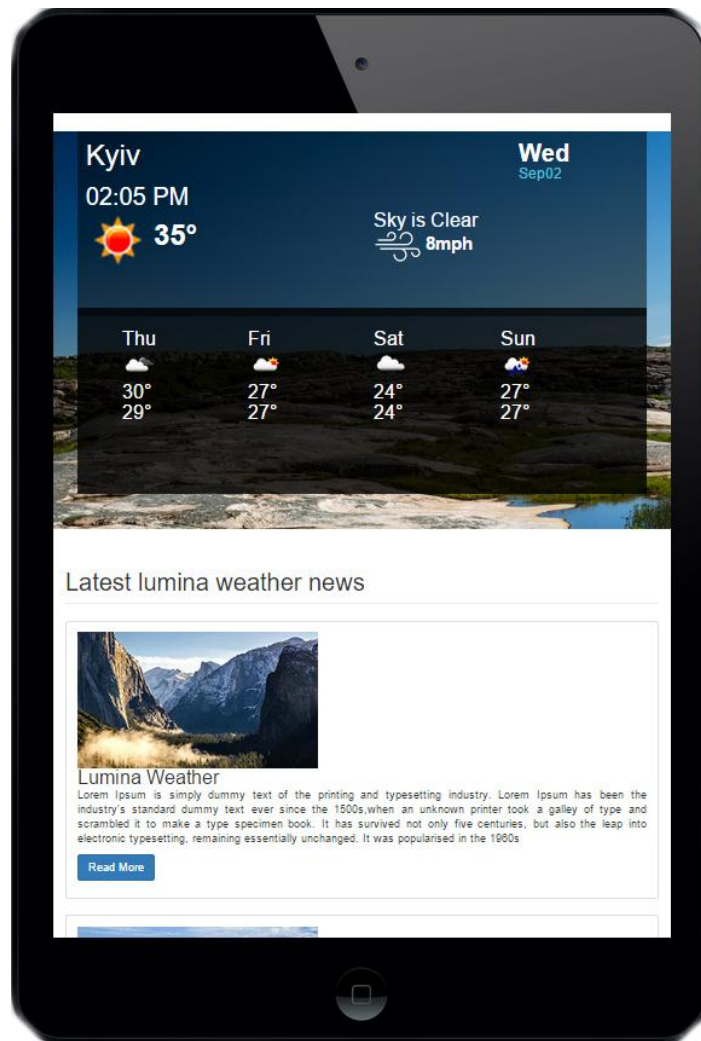
## Apple iPad mini

Css resolution 768x1024

Pixel resolution 768x1024

Operating system iOS 6.0

Device type Tablet



Pic 6 Main page via Apple iPad

## Realization

For getting these results, I used css framework Twitter Bootstrap 3. Bootstrap provides very simple grid system via dynamical columns class (.col-xs- .col-sm- .col-md- .col-lg-).

Whole page we divide on rows and use these columns in every row. And depending on screen – It becomes smaller/bigger.

### Example

```
<div class="row" >
  <div class="col-lg-6 col-md-9 col-sm-8">
    <div class="form-group">
```

```

        <input type="text" class="form-control"
id="weather"placeholder="{{ trans('messages.enter_city') }}" ng-model="jscity">
    </div>
</div>
.....
</div>

```

## 2.2 Server side features

### 2.2.1 Main class LuminaWeather\_API

Location: app/Models

This is major class of our functionality. In our situation – it's also Model part of MVC pattern. It has several methods for working and getting different results – current weather, 5 days forecast, current date, wind information etc.

One of the features – you can use it at any place of project. Besides – you can build your project as third-party service.

Let me explain – how it works

#### **public function \_\_init(\$longitude, \$latitude)**

Here we accept our longitude/latitude and get all need data.

```

//Current weather api: http://api.openweathermap.org/data/2.5/weather
$this->weatherToday =
json_decode(file_get_contents("http://api.openweathermap.org/data/2.5/weather?lat=$latitude&lon=$longitude"), true);

//Forecast api: http://api.openweathermap.org/data/2.5/forecast
$this->forecast =
json_decode(file_get_contents("http://api.openweathermap.org/data/2.5/forecast?lat=$latitude&lon=$longitude"), true);

```

```
//Current location api:http://maps.googleapis.com/maps/api/geocode/
$this->location = json_decode
(file_get_contents("http://maps.googleapis.com/maps/api/geocode/json?latlng=$latitude,$longitude&sensor=true"), true);
```

We get JSON results (file\_get\_contents()) from this APIs and make normal results with json\_decode ()

After that, we can get different parts from these results. Here is example function

```
/**
 * Get current temperature and icon
 *
 * @return array
 */
public function getIconTemperature() {
    if(empty($this->weatherToday)) {
        $this->weatherToday = Cache::get("weather");
    }
    $tempValue = round($this->weatherToday["main"]["temp"] - 273);
    $icon = $this->weatherToday["weather"][0]["icon"];
    $description = $this->weatherToday["weather"][0]["description"];
    return array("temp" => $tempValue, "icon" => $icon, "desc" =>
    $description);
}
```

### Cache possibilities

Laravel provides two basic tools for cache: Memcached and Redis. But for this simple exercise I used simple file.

```
'file' => [
    'driver' => 'file',
    'path' => storage_path('framework/cache'),
],
```

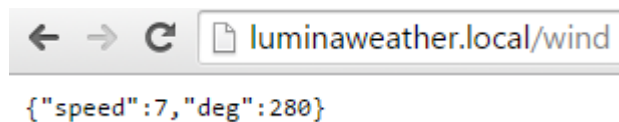
So after getting results – we save them for 90 minutes

```
$expiresAt = Carbon::now()->addMinutes(90);  
Cache::put("weather", $this->weatherToday, $expiresAt);  
Cache::put("forecast", $this->forecast, $expiresAt);  
Cache::put("location", $this->location, $expiresAt);
```

### 2.2.2. Main controller Weather\IndexController

Location: app\Http\Controllers\Weather

As usual controller of MVC – it has one main function – combine Models and Views. It has index method for init model and several methods for getting results (getCurrentWeatherAction(),getCurrentDateAction() etc.). I want to note, that I did such way because it gives one advantage for us – we don't must call this methods anymore. We can call – for example /wind and get JSON results and use it at anywhere.



Pic 7 JSON results

### 2.2.3 Multilang possibilities

Location: app\Http\Controllers\Weather\IndexController  
app\Http\Middleware\App

I provided two language supporting (ru, en).

**LuminaWeather** en | ru

So if you choose en(default) or ru – your chosen is saved in current session:  
\Session::set('locale', \$request->get("lang"));  
For this exercise, I translated only part of project. For future – you can add new translations at /resources/lang

+ Add location

Daily Weather  
Forecast Videos



Pic 8 Main page - en

+ Добавить местоположение

Видео о  
предстоящей  
погоде



Pic 9 Main page – ru

## 2.3 Client side features

### 2.3.1 Getting coordinates

Before getting current weather – we must get our location. I did it with basic HTML5 and Javascript.

```
if (navigator.geolocation) {
    $window.navigator.geolocation.getCurrentPosition(function (position) {
        $scope.$apply(function () {
            var latitude = position["coords"]["latitude"];
            var longitude = position["coords"]["longitude"];
            $http({
```

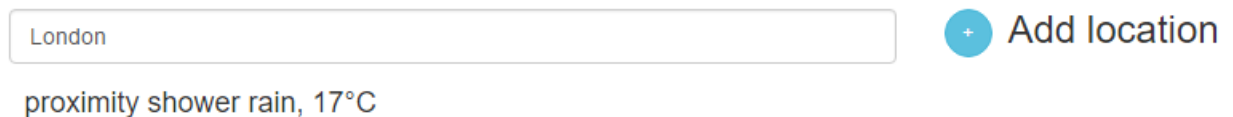
```

        url: "/save-location",
        method: "GET",
        params: {latitude: latitude, longitude: longitude},
        cache: true
    })
});
}, function (error) {
    console.log(error);
}, {timeout: 10000000, enableHighAccuracy: true, maximumAge: 0});
}

```

After getting results – I send asynchronous request to /save-location and there our location is saved.

### 2.3.2 Getting weather by city name



Pic 10 Getting weather by city name

This part was realized with AngularJS factory service (look: public/js/app.js).

```

//-----
angular.module("ImnApp", []).factory('weatherService', function ($http) {
    return {
        getWeather : function(city) {
            return $http.get('http://api.openweathermap.org/data/2.5/weather', {
                params: {
                    q: city
                }
            })
        }
    }
});

```

```

    }).then(function(response) {
        return response.data.weather[0].description + ", " +
        (response.data.main.temp - 273).toFixed() + "°C";
    })
}
}
});

```

We send one http request to [http://api.openweathermap.org /data/2.5/weather](http://api.openweathermap.org/data/2.5/weather) with one parameter – city which we getting from input filed.

#### 2.3.4 Getting all data

Here we also send request for getting results, but now, we do it with our routes.

```

//Here we will get all need information

//Today's weather

$http.get('/current-weather').success(function (response) {
    $scope.temperature = response["temp"];
    $scope.icon = response["icon"];
    $scope.description = response["desc"];
});

```

If we look route's file (routes.php) – we'll see such thing

```

Route::get('/current-weather',
'Weather\IndexController@getCurrentWeatherAction');

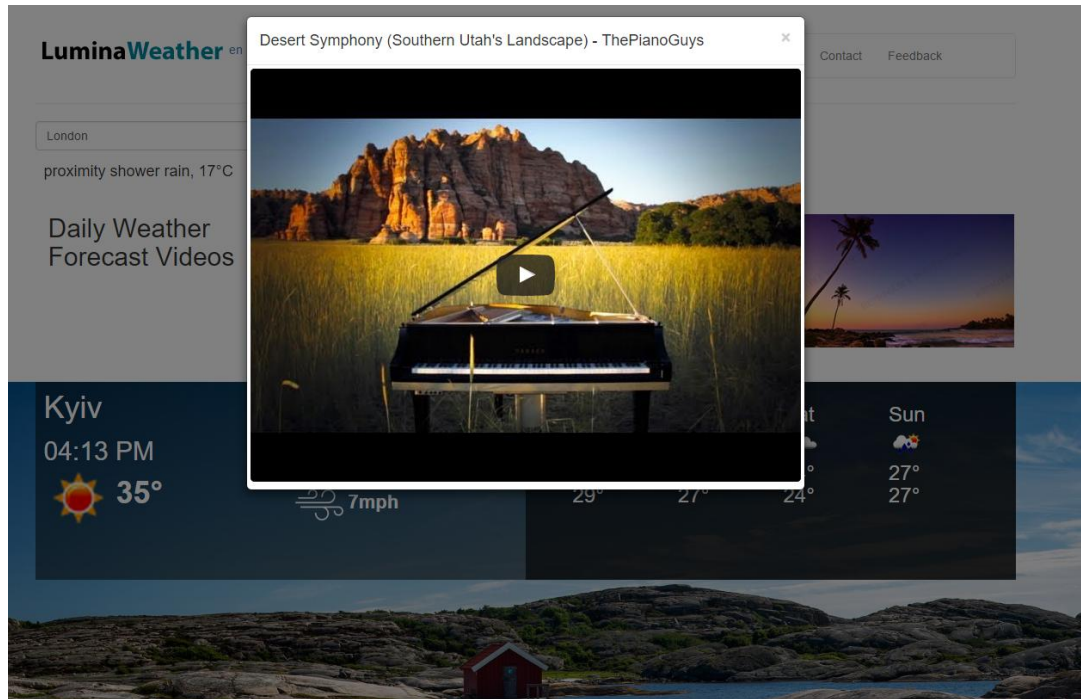
```

So if we make request to `/current-weather'`, `getCurrentWeatherAction` methods is executed

Simple notation: maybe, it'll be better to add all these parts to factory too, but I wanted to demonstrate different approaches for this task.

### 2.3.5 Videos

I provided small interactive thing as watching video. I used only one video, because I think, for simple exercise it'll be enough. In real project – we can save links for videos in database.

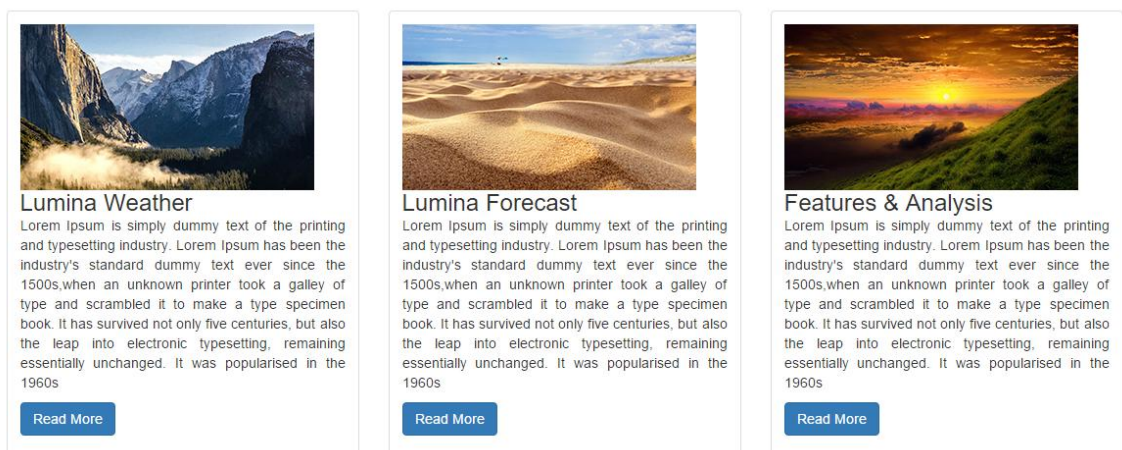


Pic 11 Watching videos

### 2.3.6 Latest news

Also small feature of this service. Being that we don't use database – I saved data in JSON string.

#### Latest lumina weather news



Pic 12 Latest news

### **3. Annotation**

Time for developing: 38-40 hours

Link for testing: <http://luminalearning.herokuapp.com/>

APIs that were used:

- <http://openweathermap.org>
- <http://maps.googleapis.com/maps/>
- <https://developers.google.com/youtube/>

Github: <https://github.com/johnyhunt/lumina>

Skype: xxleon\_huntxx