Trip Exchange System

# Overall Task Description

Trip Exchange system is an application that makes possible shared car trips for people who do not know each other.

The system provides a way to see a range of drivers and their trips. Only part of the information is public. Registered users can create trips; make sorting, filtering and paging on the results; join trips.

Your task is to implement a SPA application using AngularJS and consuming a provided ASP.NET Web API 2 server.

# REST API Server

## Description

You are given a REST API server (as well as documentation for it) locally to use from Visual Studio. The REST API server uses the standard Microsoft Identity system with different routes. Additionally, you may consume the same services at the following link: <http://spa2014.bgcoder.com/>

More details on the services are also available here: <http://spa2014.bgcoder.com/Help>

The REST API server is also open sourced at GitHub: <https://github.com/NikolayKostov-IvayloKenov/TripExchangeSystem>

You can also find “*Postman.requests.json*” file that contains most of the requests exported from Postman. You can use it for debugging.

# AngularJS Client

## Description

Implement routes for the client application.

The routes can be either public or private (accessible only by registered users).

You are provided with screenshots of a sample application for better understanding.

Invalid routes should redirect to the home page.

### The routes

* #/
  + Shows the home page
  + Shows latest 10 available trips
  + Shows last 10 registered drivers
  + Shows statistics for the trips and drivers (you should cache these statistics on the client until next refresh of the application)
  + Shows register and login (or logged user, if any)
  + Shows links to trips and drivers pages
  + Public
* #/unauthorized
  + This page should be used in case of unauthorized access to a part of the application
  + Public
* #/register
  + Should provide form for registering new user
  + The user should be able to choose whether he is driver or not. If the user chooses he/she is a driver, he/she should provide a car name
  + Validate the fields as much as possible on the form
  + Should redirect to the home page after successful registration
  + Public
* #/drivers (public part)
  + Should show last 10 registered drivers with their upcoming trips and total trips in the system
  + Public
* #/trips (public part)
  + Should show latest 10 upcoming trips with their driver name, from and to cities and departure time.
  + Public
* #/drivers (private part)
  + Should show pages of drivers (10 per page) and username filter
  + Private
* #/drivers/:id (private part)
  + Should show information about the driver with the provided id
  + Should show the driver’s trips
  + The driver’s trips should have client side filtering by only logged user trips and only selected driver trips
  + The driver’s trips should have client side sorting options by from, to and descending departure time
  + Private
* #/trips (private part)
  + Should show pages of trips (10 per page)
  + Should give link to the create trip route
  + Trips should have server-side filters for including already finished trips and only current logged user’s trips
  + Trips should have server-side filters for from and to cities
  + Trips should have server-side ordering by driver, date, number of seats , from and to cities in both ascending and descending order
  + Use dropdown lists where possible
  + Visualize dates in human-readable format
  + Visualize pictures for the Is Mine field
  + Private
* #/trips/create (private)
  + Available only for registered drivers
  + Should have form with from and to cities, available seats and departure time
  + Should redirect to the trips page after successful creation
  + Use dropdown lists where possible
  + Use numeric textbox and date time picker
  + Validate the fields as much as possible on the form
  + Private
* #/trips/:id (private)
  + Should show information about the trip with the provided id
  + Should give possibility for joining the trip
  + Should show all the passengers’ names

# General requirements

The application’s core logic must be written with AngularJS. You are free to use any other libraries like Twitter Bootstrap, jQuery, UnderscoreJS, etc.

Your application must have:

* At least 3 custom controllers (excluding the identity controllers)
* At least 3 custom directives
* At least 1 custom filter
* At least 4 custom (excluding the identity) services
* Visualization for all success or error messages coming from the server

Write high quality code and follow the best practices.

Beautiful UI is not necessary, but should be usable enough. You may use Twitter Bootstrap, if you are familiar with it.

You are provided with a skeleton application with implemented register, login, logout and session. You may use it, if you want. ☺

# RESTful API Overview

The RESTful API is exposing web services to work with the server. The API has the following web service endpoints:

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Web service endpoint** | **Description** |
| POST (public) | **api/users/register** | Registers a **new user** in the system. |
| POST (public) | **api/users/login** | **Logs in** a user in the system. |
| GET | **api/users/userInfo** | Gets the information about the **current logged user** |
| POST | **api/users/logout** | **Logs out** a user from the system. |
| GET (public) | **api/stats** | Gets **statistics** for the system – total trips, total users, total drivers and total finished trips. |
| GET (public) | **api/cities** | Gets list of strings with **all cities** sorted by name |
| POST | **api/trips** | **Creates a new trip** with current user as driver. |
| GET | **api/trips/{tripId}** | Gets **trip details** by its id. |
| PUT | **api/trips/{tripId}** | **Joins a trip** (with available seats) by given id – assigns the current user as a passenger. |
| GET (public) | **api/trips** | Gets **top 10 upcoming trips** sorted by their time of departure in ascending order. |
| GET | **api/trips?page={P}** | Gets the trips on **page {P}** where their departure time is in the future (upcoming only). All trips are sorted by their time of departure in ascending order and are at most 10. If page parameter is missing its default value is 1. |
| GET | **api/trips?page={P}&orderBy ={S}&orderType={O}** | Gets the trips on page {P}.  All trips are ordered by “**driver**”, “**date**”, “**seats**” (free seats), “**from**”, “**to**” and are at most 10. OrderType is “**asc**” or “**desc**”. If order is missing, it is assumed “date” as default in “asc” order. |
| GET | **api/trips?page={P}&from={F}&to={T}** | Gets the trips on page {P}.  Trips are filtered by **from town** {F} and/or **to town** {T} (one of which may be missing). If from or to is missing, the server returns trips from/to every town. |
| GET | **api/trips?page={P}&finished=true** | Gets the trips on page {P}, **including all the already finished trips** (with departure time earlier than the current time). By default, if finished is missing, the server returns only the upcoming trips. |
| GET | **api/trips?page={P}&onlyMine =true** | Gets the trips on page {P}. Returns **only the current user’s created or joined trips**. If onlyMine is missing, the server returns everyone’s trips. |
| GET (public) | **api/drivers** | Gets **top 10 drivers**, sorted by number of total trips (desc) and then by username (asc). |
| GET | **api/drivers?page={P}** | Gets the **drivers from page {P}**, sorted by number of total trips (desc) and then by username (asc). |
| GET | **api/drivers?page={P}&username={U}** | Gets the **drivers from page {P}**, sorted by number of total trips (desc) and then by username (asc) and **filtered by username** (filter is “*contains*” and case-insensitive). |
| GET | **api/drivers/{driverId}** | Gets **driver by its id**, along with his/her 50 most recent upcoming trips (in which this user is driver or passengers). |

**\*NOTE: Parameters *orderBy*, *orderType*, *from*, *to*, *finished* and *onlyMine* on the *trips* API point are optional and can be combined with the *page* parameter. For example *api/trips?page=5&sort=seats&to=Sofia* is valid request.**

**\*NOTE: Paging numeration always starts from 1 (there is no page 0).**

# RESTful API Details

## Users

### **Register**

|  |  |  |  |
| --- | --- | --- | --- |
| api/users/register | | | POST |
| **Request** | { "email": "test@test.com",  "password": "12345678",  "confirmPassword": "12345678",  "isDriver": "true",  "car": "Mazda 6 2014" } | | |
| **Response** |  | 200 OK | |

### **Login**

|  |  |  |
| --- | --- | --- |
| api/users/login | | POST |
| **Request** | { "grant\_type": "password",  "username": "test@test.com",  "password": "12345678" } | |
| **Response** | { "access\_token": "deYtpuVHa4Ba-1t-...\_hdTI-1Pa-bNiASNQz",  "token\_type": "bearer",  "expires\_in": 1209599,  "userName": "test@test.com",  ".issued": "Fri, 26 Sep 2014 10:39:16 GMT",  ".expires": "Fri, 10 Oct 2014 10:39:16 GMT" } | |

### **User info**

|  |  |  |
| --- | --- | --- |
| api/users/userInfo | | GET |
| **Headers** | "Authorization" = "Bearer deYtpuVHa4Ba-1t-...\_hdTI-1Pa-bNiASNQz" | |
| **Response** | { "email": "test@test.com",  "isDriver": true,  "car": "Mazda 6 2014" } | |

### **Logout**

|  |  |  |  |
| --- | --- | --- | --- |
| ****api/users/logout**** | | | POST |
| **Headers** | "Authorization" = "Bearer deYtpuVHa4Ba-1t-...\_hdTI-1Pa-bNiASNQz" | | |
| **Response** |  | 200 OK | |

## Statistics

|  |  |  |
| --- | --- | --- |
| api/stats | | GET |
| **Response** | { "trips": 1337,  "finishedTrips": 919,  "users": 654,  "drivers": 321 } | |

## Cities

|  |  |  |
| --- | --- | --- |
| api/cities | | GET |
| **Response** | ["Aheloy","Ahtopol","Aksakovo",...,"Zlataritsa","Zlatitsa","Zlatograd"] | |

## Trips

### **Create trip**

|  |  |  |
| --- | --- | --- |
| api/trips | | POST |
| **Headers** | "Authorization" = "Bearer deYtpuVHa4Ba-1t-...\_hdTI-1Pa-bNiASNQz" | |
| **Request** | { "from": "Sofia",  "to": "Kyustendil",  "availableSeats": "4",  "departureTime": "2014-09-29 13:37:00" } | |
| **Response** | { "id": "bcf79ac5-0e77-4a21-aa21-279c93375231",  "driverId": "fe411b06-cef6-4c95-b95a-13a3310a1f05",  "driverName": "test@test.com",  "from": "Sofia",  "to": "Kyustendil",  "departureDate": "2014-09-29T13:37:00",  "numberOfFreeSeats": 4,  "isMine": true,  "passangers": ["test@test.com"] } | |

**\* REQUIREMENTS: The logged user should be driver in order to be able to create trips. availableSeats should be at least 1. departureTime should be in the future. from and to should be from the list of available cities (/api/cities)**

**\* NOTE: isMine value means that the currently logged user is part of the trip (either driver or passenger)**

**\* NOTE: All drivers are passengers too.**

### **Get trip detail**

|  |  |  |
| --- | --- | --- |
| api/trips/7362e399-d051-42bd-95a0-109a2673dfd5 | | GET |
| **Headers** | "Authorization" = "Bearer deYtpuVHa4Ba-1t-...\_hdTI-1Pa-bNiASNQz" | |
| **Response** | { "id": "7362e399-d051-42bd-95a0-109a2673dfd5",  "driverId": "c73d0bc9-47b7-4cc4-abb2-cd5516b94596",  "driverName": "test02@test.com",  "from": "Ahtopol",  "to": "Etropole",  "departureDate": "2014-10-11T17:09:19.203",  "numberOfFreeSeats": 3,  "isMine": false,  "passangers": [  "test02@test.com",  "test06@test.com"  ]  } | |

### **Join trip**

|  |  |  |
| --- | --- | --- |
| api/trips/7362e399-d051-42bd-95a0-109a2673dfd5 | | PUT |
| **Headers** | "Authorization" = "Bearer deYtpuVHa4Ba-1t-...\_hdTI-1Pa-bNiASNQz" | |
| **Response** | { "id": "7362e399-d051-42bd-95a0-109a2673dfd5",  "driverId": "c73d0bc9-47b7-4cc4-abb2-cd5516b94596",  "driverName": "test02@test.com",  "from": "Ahtopol",  "to": "Etropole",  "departureDate": "2014-10-11T17:09:19.203",  "numberOfFreeSeats": 2,  "isMine": true,  "passangers": [  "test02@test.com",  "test06@test.com",  "test@test.com"  ]  } | |

**\*NOTE: In case of any error (invalid trip id, past trip, user already in the trip, no seats available, etc.) you will receive status code 400 (with JSON object with property message, containing the text of the error). This applies for all other queries.**

### **Get trips**

|  |  |  |
| --- | --- | --- |
| api/trips?page=1&orderBy=seats&orderType=asc&from=Sofia&finished=true&onlyMine=false | | GET |
| **Headers** | "Authorization" = "Bearer deYtpuVHa4Ba-1t-...\_hdTI-1Pa-bNiASNQz" | |
| **Response** | [{  "id": "0ea1b7d8-b639-4fb1-a79d-d97fab0c79e4",  "driverId": "84630007-63ad-4ccc-8264-1797e6a43279",  "driverName": "test34@test.com",  "from": "Sofia",  "to": "Boychinovtsi",  "departureDate": "2014-11-09T17:09:19.3",  "numberOfFreeSeats": 3,  "isMine": false,  "passengers": ["test13@test.com", "test34@test.com"]  }, {  "id": "bcf79ac5-0e77-4a21-aa21-279c93375231",  "driverId": "fe411b06-cef6-4c95-b95a-13a3310a1f05",  "driverName": "test@test.com",  "from": "Sofia",  "to": "Kyustendil",  "departureDate": "2014-09-29T13:37:00",  "numberOfFreeSeats": 4,  "isMine": true,  "passengers": ["test@test.com"]  }, {  "id": "c58dec3b-ec72-4e85-8658-401065abaedc",  "driverId": "9297512a-6817-435a-b0b7-df11277f400d",  "driverName": "test10@test.com",  "from": "Sofia",  "to": "Dolna Mitropoliya",  "departureDate": "2014-07-11T17:09:18.957",  "numberOfFreeSeats": 4,  "isMine": false,  "passengers": ["test10@test.com"]  }, {  "id": "8cf1097e-8710-46ef-a1d7-5893ca5cbef3",  "driverId": "fe411b06-cef6-4c95-b95a-13a3310a1f05",  "driverName": "test@test.com",  "from": "Sofia",  "to": "Kyustendil",  "departureDate": "2014-09-29T13:37:00",  "numberOfFreeSeats": 4,  "isMine": true,  "passengers": ["test@test.com"]  }] | |

## Drivers

### **Get drivers**

|  |  |  |
| --- | --- | --- |
| api/drivers?page=1&username=TEST.com | | GET |
| **Headers** | "Authorization" = "Bearer deYtpuVHa4Ba-1t-...\_hdTI-1Pa-bNiASNQz" | |
| **Response** | [{  "id": "1c25d2fc-f2f5-4e63-bef1-01a2c4374e26",  "name": "test44@test.com",  "numberOfUpcomingTrips": 3,  "numberOfTotalTrips": 10  }, {  "id": "36f11831-545c-4228-b0b6-be54832884e6",  "name": "test01@test.com",  "numberOfUpcomingTrips": 1,  "numberOfTotalTrips": 8  }, {  "id": "52f9631d-0af7-49b0-81b1-5b0476131022",  "name": "test34@test.com",  "numberOfUpcomingTrips": 3,  "numberOfTotalTrips": 8  }] | |

**\*NOTE: If you send the request anonymously you will receive** **the top 10 drivers, sorted by number of trips (in descending order) and then by username (in ascending order) with no paging and filtering.**

### **Get driver detail**

|  |  |  |
| --- | --- | --- |
| api/drivers/52f9631d-0af7-49b0-81b1-5b0476131022 | | GET |
| **Headers** | "Authorization" = "Bearer deYtpuVHa4Ba-1t-...\_hdTI-1Pa-bNiASNQz" | |
| **Response** | {  "id": "e11ba9c5-d0cc-465a-aa55-354340e42baf",  "name": "test36@test.com",  "numberOfUpcomingTrips": 3,  "numberOfTotalTrips": 9,  "trips": [{  "id": "f466253d-a1ef-41c6-9f84-5c2722a01d73",  "driverId": "e11ba9c5-d0cc-465a-aa55-354340e42baf",  "driverName": "test36@test.com",  "from": "Botevgrad",  "to": "Simitli",  "departureDate": "2014-10-05T17:09:19.16",  "numberOfFreeSeats": 1,  "isMine": false  }, {  "id": "c639f81c-1f3e-4b00-9533-0c88e96a520e",  "driverId": "c73d0bc9-47b7-4cc4-abb2-cd5516b94596",  "driverName": "test02@test.com",  "from": "Ardino",  "to": "Belitsa",  "departureDate": "2014-10-31T17:09:19.27",  "numberOfFreeSeats": 1,  "isMine": false  }, {  "id": "080dc395-09ac-4f76-8e17-a188a9de0fae",  "driverId": "70bd20b5-2d78-427b-b359-3eb1b891d0d7",  "driverName": "test26@test.com",  "from": "Saedinenie",  "to": "Maglizh",  "departureDate": "2014-11-15T17:09:19.323",  "numberOfFreeSeats": 3,  "isMine": false  }]  } | |