Source Control System

Overall Task Description

Source control system is an application that allows people to work in teams and collaborate in software programming. You know what source control system is and actually you know half of the server code already!
The system provides a way to create and see software projects and contribute source code in it. Only part of the information is public. Registered users can create projects; make sorting, filtering and paging on the results; add collaborators and commit source code.
Your task is to implement a SPA application using AngularJS and consuming a provided ASP.NET Web API 2 server.

REST API Server

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://spa.bgcoder.com/

AngularJS Client

Implement routes for the client application.
The routes can be either public or private (accessible only by registered users).
Invalid routes should redirect to the home page.

- **`#/`**
  - Shows the home page
  - Shows latest 10 public software projects
  - Shows latest 10 public source code commits
  - Shows statistics for the projects, commits and users (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 projects and commits pages (links can be in any manner)
  - 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
  - Validate the fields as much as possible on the form
  - Should redirect to the home page after successful registration
  - Public
- **`#/projects` (public part)**
  - Should show last 10 added projects
  - Public
- **`#/projects` (private part)**
  - Should show pages of projects (10 per page) and all filters the API exposes
  - Should give link to the create project route
  - Visualize pictures for the License field

  - Use dropdown lists where possible
  - Visualize dates in human-readable format
  - Private
- **`#/projects/:id` (private part)**
  - Should show information about the project with the provided id
  - Should show project commits (with server side paging and username filter)
  - Should show link to adding new commit
  - The commits should have client side filtering by only current users commit on the currently shown page
  - The commits should have client side sorting options by from and to creation time on the currently shown page
  - Should give link to the add commit route
  - If the current user is collaborator to the project, he/she can add other users by e-mail to be collaborators to the same project
  - Should show all collaborators to the project
  - Private
- **`#/commits/:id` (private part)**
  - Should show information about the commit with the provided id
  - Private
- **`#/projects/add` (private part)**
  - Available only for registered users
  - Should have form with validation for adding new project
  - Should redirect to the projects page after successful creation
  - Use dropdown lists where possible
  - Use appropriate UI controls
  - Private
- **`#/projects/:id/addcommits` (private part)**
  - Available only for registered users
  - Should have form with validation for adding new commit to the project with the id from the route
  - Should redirect to the project page after successful creation
  - Use appropriate UI controls
  - Private

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 services (excluding the identity services)
- At least **2** Kendo UI widgets (use Kendo for Angular)
- 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.

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/account/register**	Registers a new user in the system.
POST (public)	**token**	Logs in a user in the system.
POST	**api/account/logout**	Logs out a user from the system.
GET (public)	**api/statistics**	Gets statistics for the system.
GET	**api/licenses**	Gets all available licenses.
GET (public)	**api/projects**	Gets latest 10 added projects sorted by their time of creation.
GET	**api/projects/{projectId}**	Gets project details by its id.
POST	**api/projects**	Creates a new project with current user as collaborator.
PUT	**api/projects/{projectId}**	Adds collaborator to a project.
GET	**api/projects/all**	Gets filtered projects. Available query string filters are: - "**Page**" (starts from 1), - "**PageSize**" (default is 10), - "**Filter**" (searches in name and description), - "**OrderBy**" ("date", "collaborators" or "name"), - "**OrderType**" ("asc" or "desc"), - "**ByUser**" (user e-mail), - "**OnlyPublic**" (true or false) For example: api/projects/all?filter=test&page=2&orderby=name
GET	**api/projects/collaborators/{projectId}**	Gets all collaborators in a project with the corresponding id.
GET (public)	**api/commits**	Gets latest 10 commits sorted by their time of creation.
GET	**api/commits/{commitId}**	Gets commits details by its id.
POST	**api/commits**	Creates new commit from the current user.
GET	**api/commits/byproject/{projectId}**	Gets latest commits by project id. Available query string filters are: - "**Page**" (starts from 1), - "**PageSize**" (default is 10), - "**ByUser**" (user e-mail),

RESTful API Details

You are given only the POST methods' details. For all other API methods, you will need to explore the responses by yourself.

Users

Register

api/users/register		**POST**
Request	`{ "email": "test@test.com", "password": "12345678", "confirmPassword": "12345678" }`	
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" }

Logout

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

Projects

Create project

api/projects	POST
Headers	"Authorization" = "Bearer deYtpuVHa4Ba-1t-..._hdTI-1Pa-bNiASNQz"

Request	{
	"name": "Test",
	"description": "Test Description",
	"licenseId": 1,
	"private": true
	}

Add collaborator

api/projects/{id}	PUT
Headers	"Authorization" = "Bearer deYtpuVHa4Ba-1t-..._hdTI-1Pa-bNiASNQz"
Request	"test@test.com"

Commits

Add commits to project

api/commits	PUT
Headers	"Authorization" = "Bearer deYtpuVHa4Ba-1t-..._hdTI-1Pa-bNiASNQz"

Request	{
	"projectId": 1,
	"sourceCode": "using MyTested.WebApi; // :D"
	}

You have 8 hours to work. Good luck!