# Exercises: Application Flow & Middleware

Problems for exercises and homework for the [“C# MVC Frameworks - ASP.NET Core” course @ SoftUni](https://softuni.bg/trainings/2197/csharp-mvc-frameworks-asp-net-core-november-2018).

# Eventures

**Eventures** **Inc**. is a fast-rising newly made Start-Up Company, which specializes in **Event Tickets Sales**. It is said to be the killer of systems like Eventim, Eventbride, etc.

You have been appointed as the developer of the **main web application**. This is a great responsibility, so do your best and do not dissapoint your employers. The application functionality is not that complex, and it will be **separated** into **several parts**, each part consisting of **several tasks**.

Your current task is to create the **architecture** and **core logic** of the **application**, so get started.

## Data Storage

The core application logic requires **2 data models** to be implemented:

### User

Has the following properties:

* Username – a string (from IdentityUser)**.**
* Password – a string (from IdentityUser)**.**
* Email – a string (from IdentityUser)**.**
* First Name – a string.
* Last Name – a string.
* Unique Citizen Number (UCN) – a string.
* Role – can be User / Admin

### Event

Has the following properties:

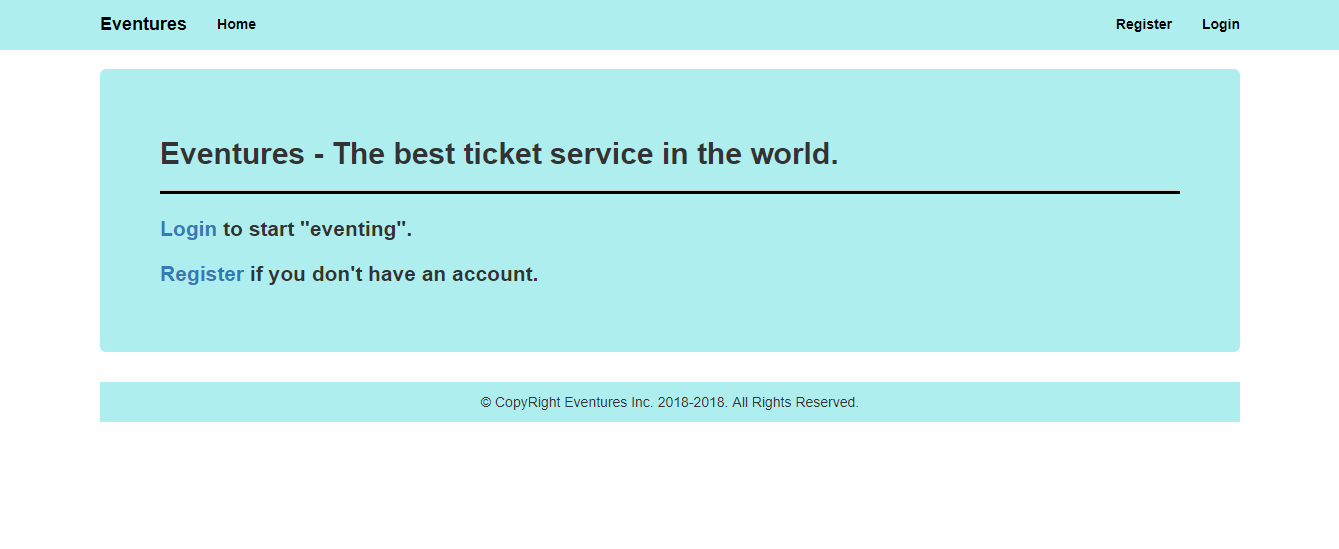
* Id – a **UUID**.
* Name – a string.
* Place – a string.
* Start – a DateTime object.
* End – a DateTime object.
* Total Tickets – an integer.
* Price Per Ticket – a **decimal** value.

## Front-End

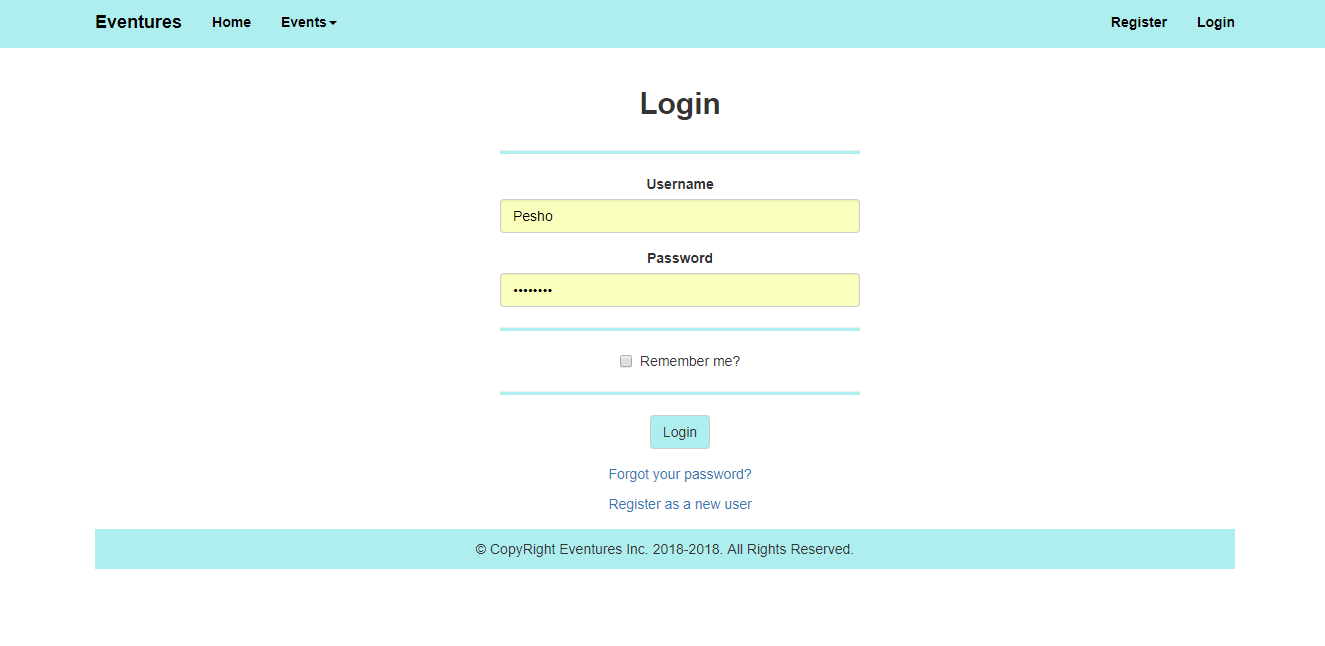
The Front-End might be a little different in some small components than the things you’ve done so far. However, do not panic, it is not that hard to do it. You can do it! 😉

There are a few templates you must implement, they are not that much, but they are tricky.

### Index Template

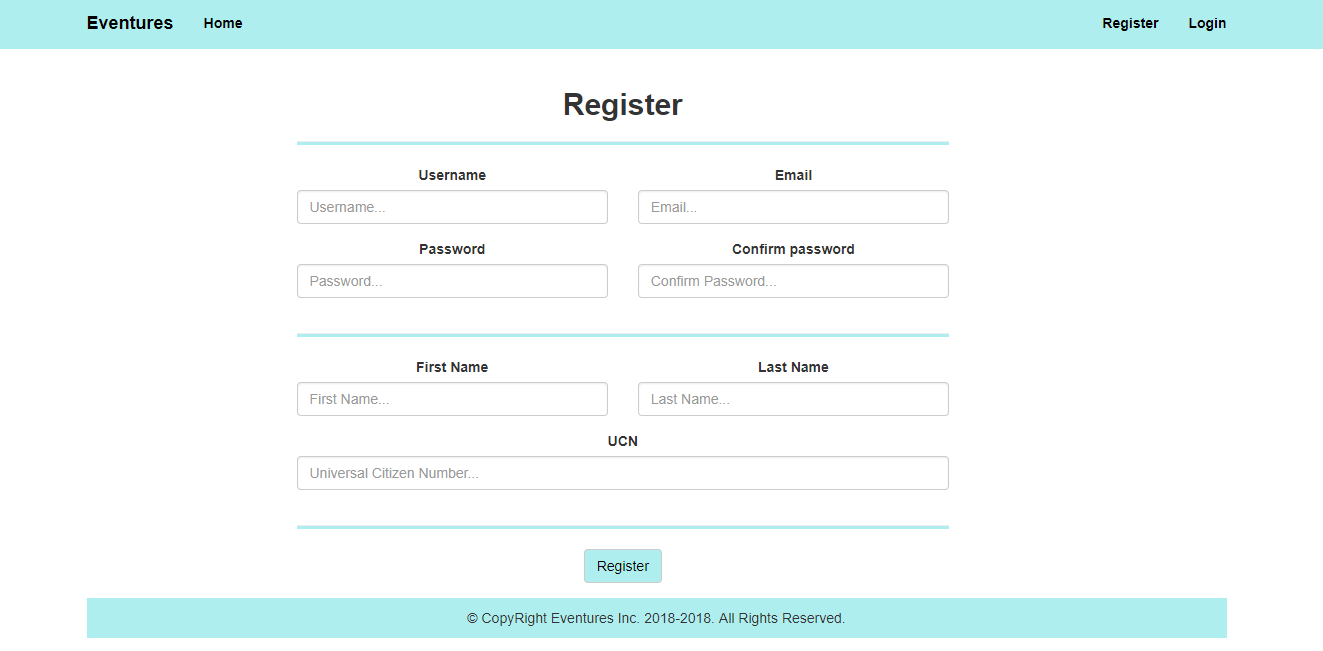


### Login Template

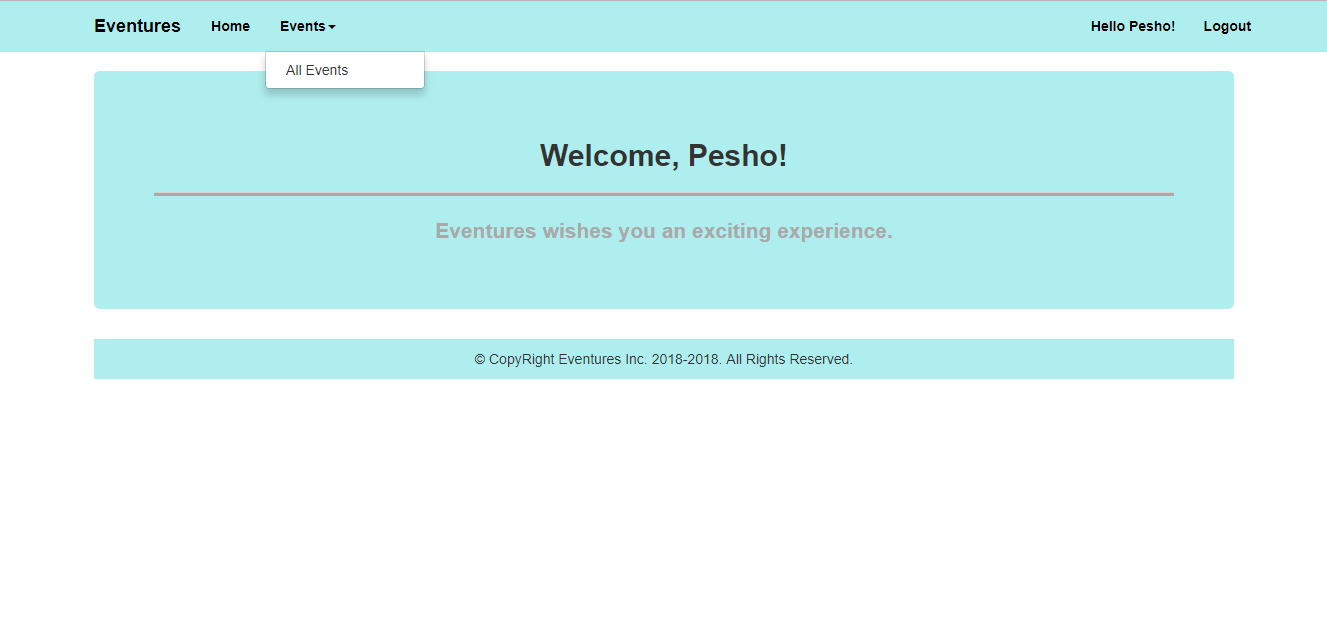


The Forgot your password? Is a just a link with no functionality, for now.

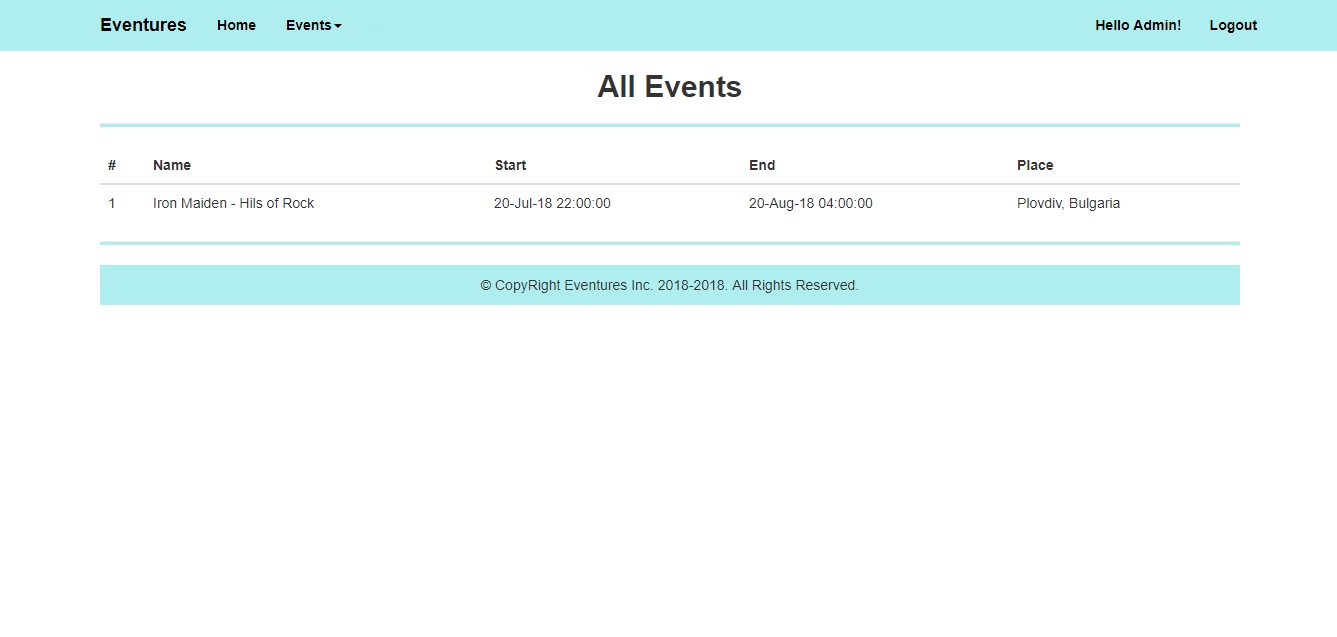
### Register Template



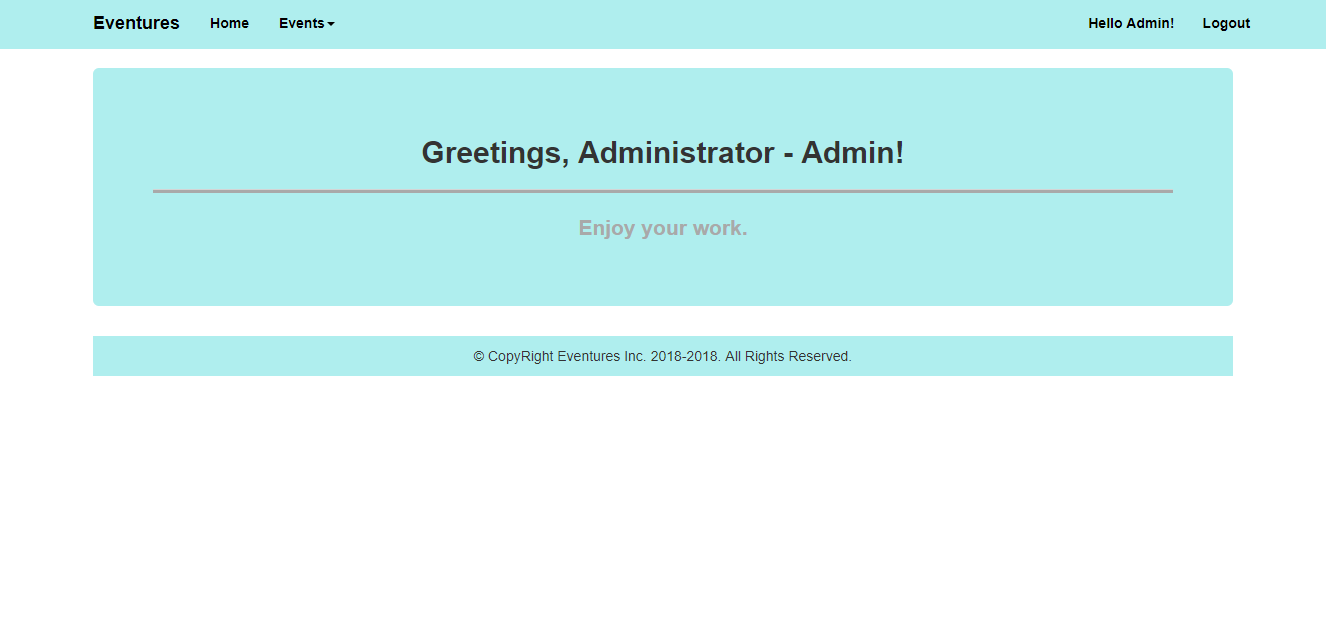
### Home (logged-in users)



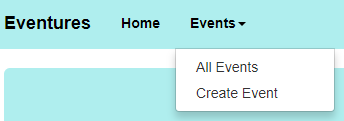
### All Events (logged-in)



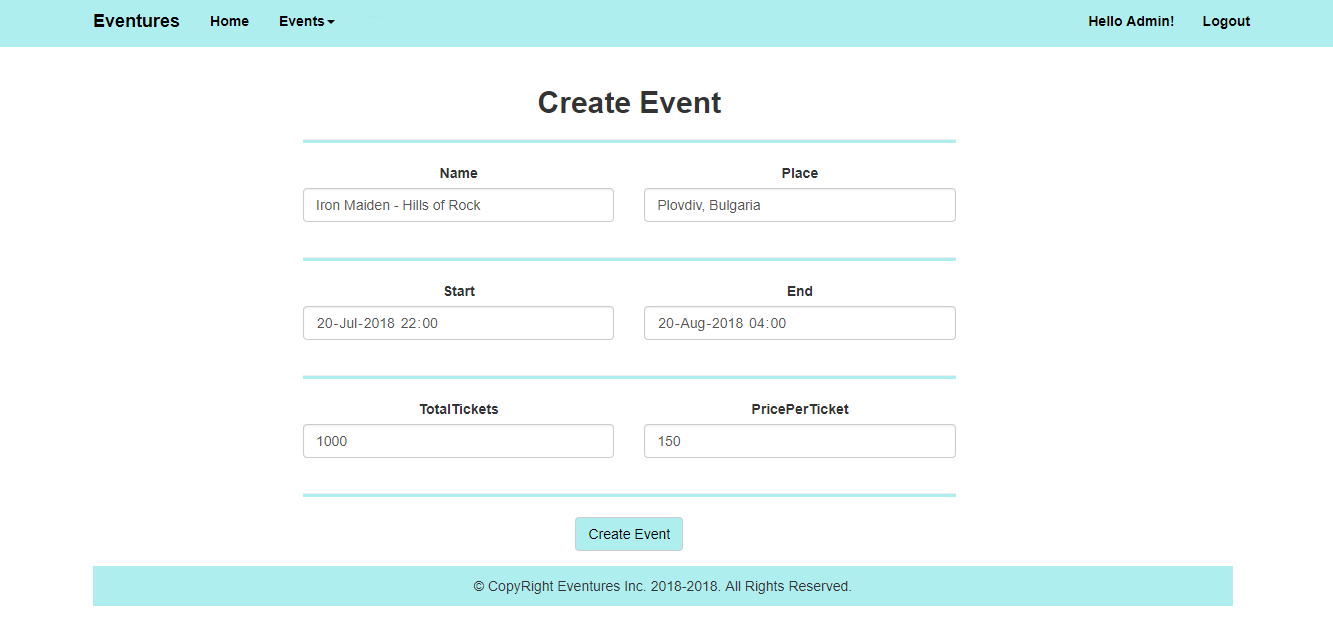
### Admin Home (logged-in Admin)



#### Admin Navigation



### Create Events



**Note**: The color for the application is #AFEEEE.

## Business Logic

### Technical Requirements

The application should be an **ASP.NET Core Web** app. As such it should use **the most** of the **ASP.NET Core MVC Framework**.

Use **ASP.NET Core Identity** for authentication. Add the following **roles** to your User functionality – (‘USER’, ‘ADMIN’).

### Functionality

The application should provide its Guest users (**not logged-in**) the functionality to register and login.

The application should provide its Regular users (**logged-in** Users with Role – User) the functionality to **view all** Events.

The application should provide its Admin users (**logged-in** Users with Role – Admin) the functionality to **create** **new** Events, **view all** Events.

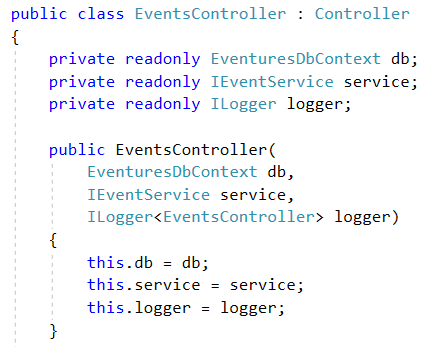
## Logging Provider

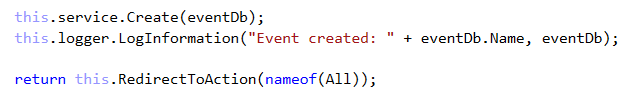
A logging provider **displays or stores logs**. For example, the Console provider displays logs on the console. The default project template calls the **CreateDefaultBuilder** extension method, which adds the logging providers Consolе, Debug, EventSource (starting in ASP.NET Core 2.2).

If you use **CreateDefaultBuilder**, you can replace the default providers with your own choices. Call **ClearProviders**, and add the providers you want.

### Create Logs

Get an **ILogger<TCategoryName>** object from DI and **create information log for created Event**.





## Middleware

Create a Middleware, which **seeds** the **database** upon **Application startup**. It should **seed** the database with the Roles and it should seed a single Admin User.

## Filters

Create an ActionFilter which **Logs** each Admin Create Event activity, after the action itself. The log should output a single message:

[{currentDateAndTime}] Administrator {Username} create event {EventName} ({EventStart} / {EventEnd}).