**Тестовое задание на вакансию «**[**Программист C#/ASP.NET**](https://hh.ru/vacancy/68655843?hhtmFrom=employer_vacancies)**»**

Disclaimer

Для сокращения времени, необходимого для реализации данной задачи, авторы постарались спроектировать его таким образом, чтобы Вы могли использовать максимум функционала "из коробки", которую предоставляет Microsoft Visual Studio.

Цель

В рамках данного тестового задания Вам необходимо создать простой веб-интерфейс и веб-сервис, который позволяет вести мониторинг пользователей в системе.

Для аутентификации пользователей достаточно использовать Basic Auth (username + password).

Для выполнения задания советуем воспользоваться одним из шаблонов:

* ASP.NET Core with React.js
* ASP.NET Core with Angular
* В крайнем случаи ASP.NET Core MVC

Инструментарий

1. Microsoft Visual Studio 2022

2. ASP.NET Core Identity (IdentityDbContext или его наследники) Для этого не забывайте при создании проекта указать правильные настройки аутентификации пользователей.

3. Для веб сайта рекомендуется использовать React.js или Angular (советуем воспользоваться шаблоном Visual Studio для экономии времени)

4. Для веб-сервиса ASP.NET Core Web API

5. Любая реляционная БД (SQLite и т.п.), которую нетрудно будет развернуть на локальной машине или докер контейнере.

Основные задачи (все обязательны к исполнению)

1. Использовать возможности ASP.NET Identity, чтобы создать пользователей и управления профилем. При запуске приложения сервис должен создать базу и добавить туда пользователя с правами Администратора.

2. Расширить UserIdentity, чтобы появились возможности

* хранить изображение добавленное пользователями.
* подсчёт входов в систему
* дата последнего входа в систему

3. Изменить страницу управления профилем пользователя таким образом, чтобы пользователь мог загружать аватарку. Предусмотреть поддержку следующих расширений для файлов с изображениями: svg, png, jpg.

4. Добавить отображение аватара пользователя на странице профиля.

5. На домашней странице отобразить полный список пользователей. Строка списка должна отображать следующий минимальный набор данных: имя пользователя, роль, аватар, количество входов в систему, дата последнего входа. Остальные данные по желанию.

6. Если пользователь не прошёл процесс авторизации в системе, то автоматически перенаправлять его на страницу Login.

7. Добавить возможность назначить пользователю роль администратора.

8. Добавить кнопку "Delete" в списке пользователей (напротив каждого пользователя), которую может видеть и использовать только **Администратор**.

9. Кнопка “Delete” не должна отображаться в строке текущего пользователя, если он имеет роль Администратора.

10. Предусмотреть следующие валидации:

* удаление несуществующего пользователя
* удаление пользователя без администраторских прав
* удаление текущего пользователя
* проверка на размер файла для аватара, размер не должен быть более 20КБ
* загрузки автара другому пользователю

Дополнительные задания\* (по желанию)

\*данные задания не обязательно выполнять, если чувствуете себя стесненным по времени.

1. а) Изменить метод контроллера WeatherForecast, чтобы он возвращал температурные данные по каждому из месяцев по выбранному году (минимум одно значение на 1 месяц)

б) Вывести график по обновлённым данным на странице FetchData со следующими полями: по горизонтали “месяц”, по вертикали “температура в градусах цельсия”. Предусмотреть вывод как положительных, так и отрицательных температур.

в) Если API возвращает более одного значения в месяц, предусмотреть расчёт среднего температурного значения в этот месяц

Для отображения графиков советуем использовать библиотеку plotly.js

2. Создать файл docker-compose, в котором будет развернуты Ваш UI, ASP.NET сервер и БД.

Успешного кодирования!