# Тестовое задание для разработчика .NET

## Задача

Поиск проектов на Github, сохранение в локальную базу данных, вывод результатов на экран.

## Технологии

.NET Core, Razor Pages, bootstrap

## Описание

Для хранения локальных данных можно использовать MySql, MariaDB, PostgreSql, SqLite.

### Веб страница

Разработать приложение, используя bootstrap4 либо bootstrap5. На первой странице расположены: поле ввода текста, кнопка «Поиск».

Приложение должно реализовывать следующий механизм:

1. Пользователь вводит в поле поиска текст, нажимает кнопку «Поиск»
2. Приложение производит поиск введенных данных по локальной бд
3. Если данные не найдены, то производится отправка запроса на API Github *https://api.github.com/search/repositories?q=subject*, где **subject** – введенный текст
4. Полученный ответ сохраняется в локальную бд данных в формате: строка поиска, результат (json)
5. Производится отдача страницы с результатами

При показе страницы с результатами необходимо сохранять в поле поиска введенный текст.

Результаты выводить в виде карточек (bootstrap cards). В карточке отображать:

* Имя проекта
* Автор
* Количество звезд (Stargazers)
* Количество просмотров (Watchers)

При нажатии на карточку осуществлять переход на репозиторий Github

### REST API

Реализовать REST API для:

1. Осуществления поиска POST /api/find , строку поиска передавать в теле (body) запроса. Механизм поиска такой же, как и при поиске со страницы в браузере.
2. Просмотра списка поисков GET /api/find – выдавать результаты в виде json массива с объектами, в составе которых:

* Имя проекта
* Автор
* Количество звезд (Stargazers)
* Количество просмотров (Watchers)
* Ссылка на репозиторий

1. Реализовать функцию сброса результатов поиска по поисковой строке DELETE /api/find/{id}, где {id} – идентификатор поиска

## Дополнительные задачи

Дополнительные задачи не являются обязательной частью. Тем не менее, выполнение этих задач будет плюсом.

1. В REST API при просмотре списка поисков добавить параметр для вывода результатов постранично. Учитывать номер страницы при отдаче списка, а так же константу «количество на страницу» при построении запросов.
2. Авторизация в REST API c использованием IdentityModel. Результат авторизации — JWT токен. Запросы к REST API доступны только при наличии в заголовках (headers) параметра Authorization: Bearer полученный\_токен\_jwt

## Оценка результата

При оценке результата будут приняты во внимание следующие критерии:

1. Логичная структура приложения
2. Оформление кода
3. Использование DI
4. Наличие миграций для базы данных