**Министерство образования и науки Российской Федерации**

**федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**«Российский государственный университет**

**нефти и газа(НИУ) имени И. М. Губкина»**

Кафедра Автоматизированные системы управления

Дисциплина «Технология кроссплатформенной разработки»

**О Т Ч Е Т**

по лабораторной работе №2

«Разработка серверной части»

Студент: Олейникова Д.Д

Группа: АС-17-04

Преподаватель: Папилина Т.М.

Москва 2020 г.

1. ***Целью*** работы является разработка приложения Web API в ASP.Net Core.
   1. Описать модель (2-3 класса c бизнес-логикой, т.е. классы содержат не только данные, но и методы для работы с ними);
   2. Добавить в API контроллеры для работы с моделью: CRUD-операции и несколько (2-5) запросов на объединение данных из разных классов. Для работы с коллекциями пользоваться LINQ (программа минимум - Where и Select).
   3. +2 балла: добавить авторизацию: получение сводных отчетов только для авторизованных пользователей, добавление данных - только для администраторов.
   4. +2 балла: создать репозиторий, выложить код туда, прислать ссылку.
   5. +1 балл: собрать релиз-версию проект и скинуть архив с релизом и набором запросов для тестирования.
2. ***Задание***:
   1. **Создание веб-API с помощью ASP.NET CORE. Описать модель**

Создадим новый проект TodoApi на базе шаблона Веб-приложение AAP.NET Core.

По задумке проекта, пользователь введёт дневник задач. Модель задумки содержит два класса: Todoitem и Diary. Создадим их в проекте.

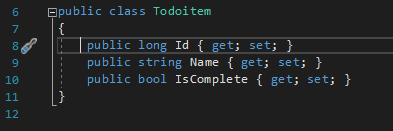


Рисунок 1

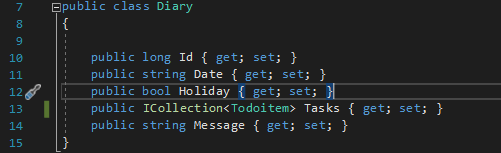


Рисунок 2

Класс Diary содержит массив Tasks типа Todoitem, чтобы пользователь мог вводить несколько задач в день. Свойство Id выступает в качестве уникального ключа реляционной базы данных.

Добавим контекст базы данных, который координирует функциональные возможности Entity Framework для модели данных. Создадим Todocontext и Diarycontext

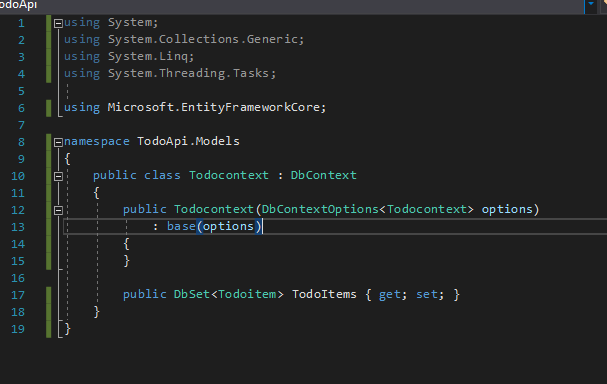


Рисунок 3

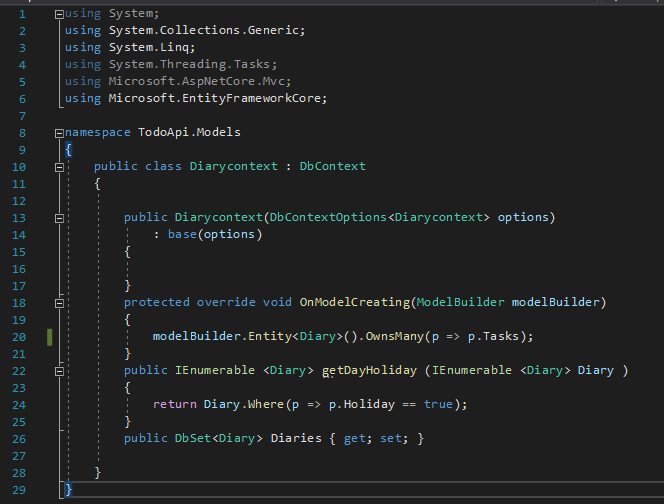


Рисунок 4

* 1. **Добавить в API контроллеры для работы с моделью**

Обновим файл Startup.cs для регистрации контекста базы данных.

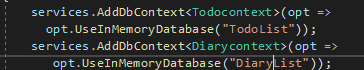


Рисунок 5

Сформируем контроллер с помощью встроенного шаблонного элемента «Контроллер API с действиями, использующий Entity Framework». Таким образом, мы получили CRUD операции (put, post, get, delete).

Добавим запрос «holidays», который будет возвращать записи дней, являющимися выходными. Для этого в Diarycontext напишем следующее:

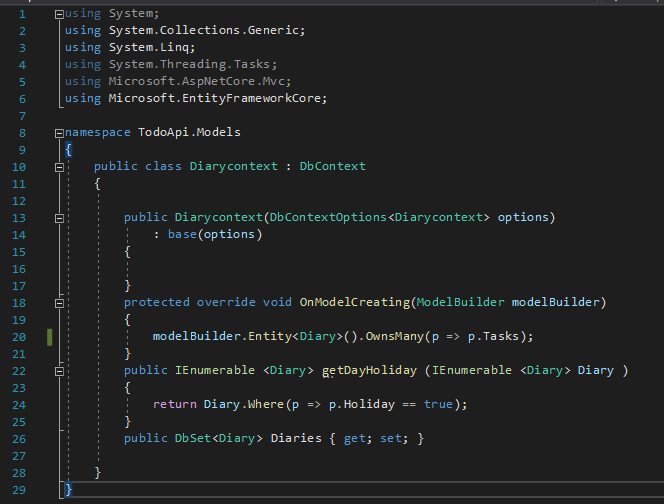


Рисунок 6

В контроллер добавим запрос «holidays»

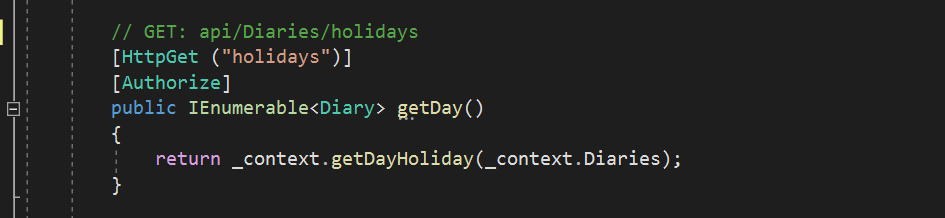


Рисунок 7

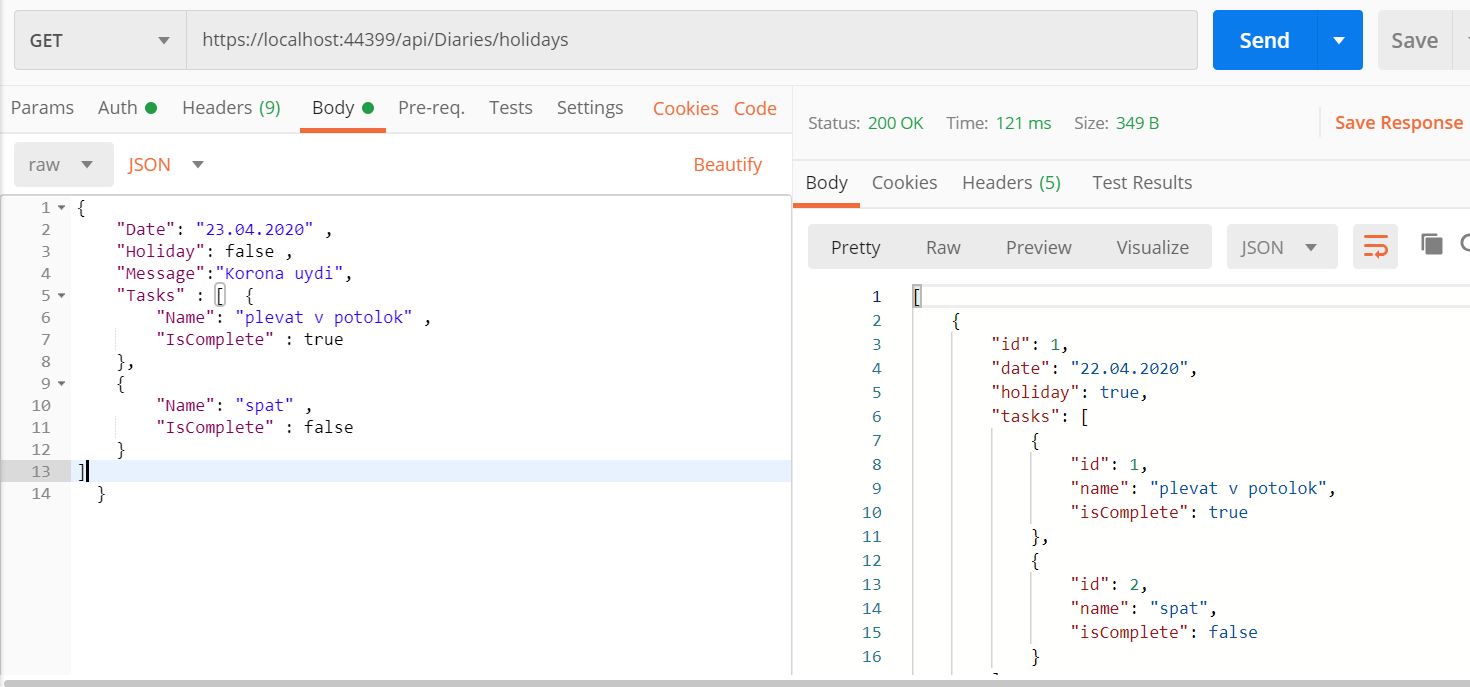
Убедимся, что все работает успешно, с помощью Postman.

Рисунок 8

Добавим запрос «firstday», который будет возвращать записи дней, являющимися первыми в месяце. Для этого в Diarycontext напишем следующее:

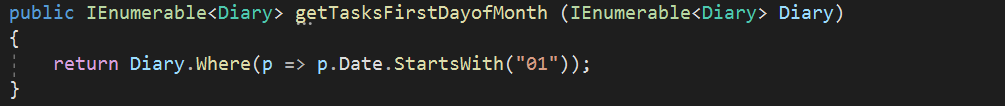


Рисунок 9

В контроллер добавим запрос «firstday»

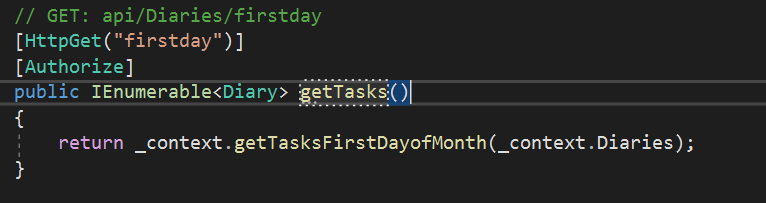


Рисунок 10

Убедимся, что все работает успешно, с помощью Postman.

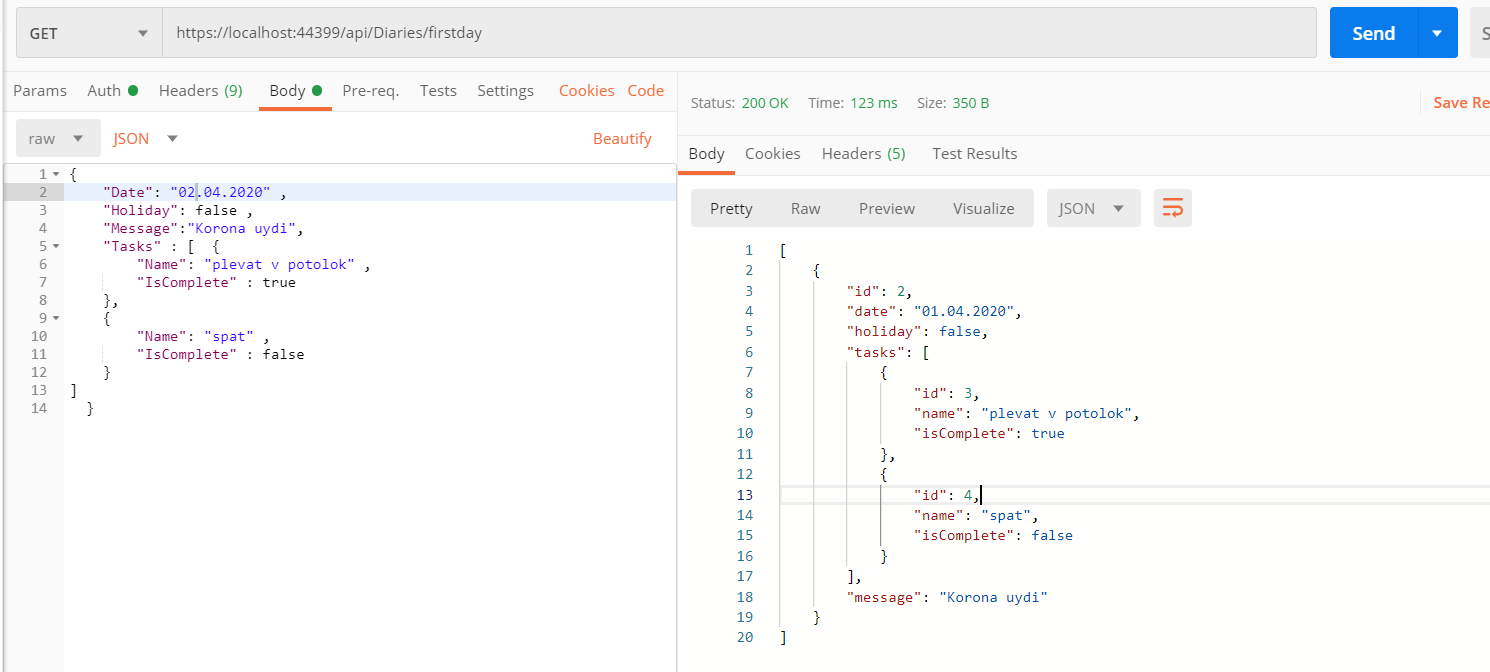


Рисунок 11

* 1. **Добавить авторизацию: получение сводных отчетов только для авторизованных пользователей, добавление данных - только для администраторов**

Установим пакет **Microsoft.AspNetCore.Authentication.JwtBearer**. для работы с JWT-токенами. Добавим класс AuthOptions. Пусть секретный ключ будет "superSecretKey\_CoVid19". Хранится токен будет год. Если нам передали значение «true», значит генерируем токен для администратора.

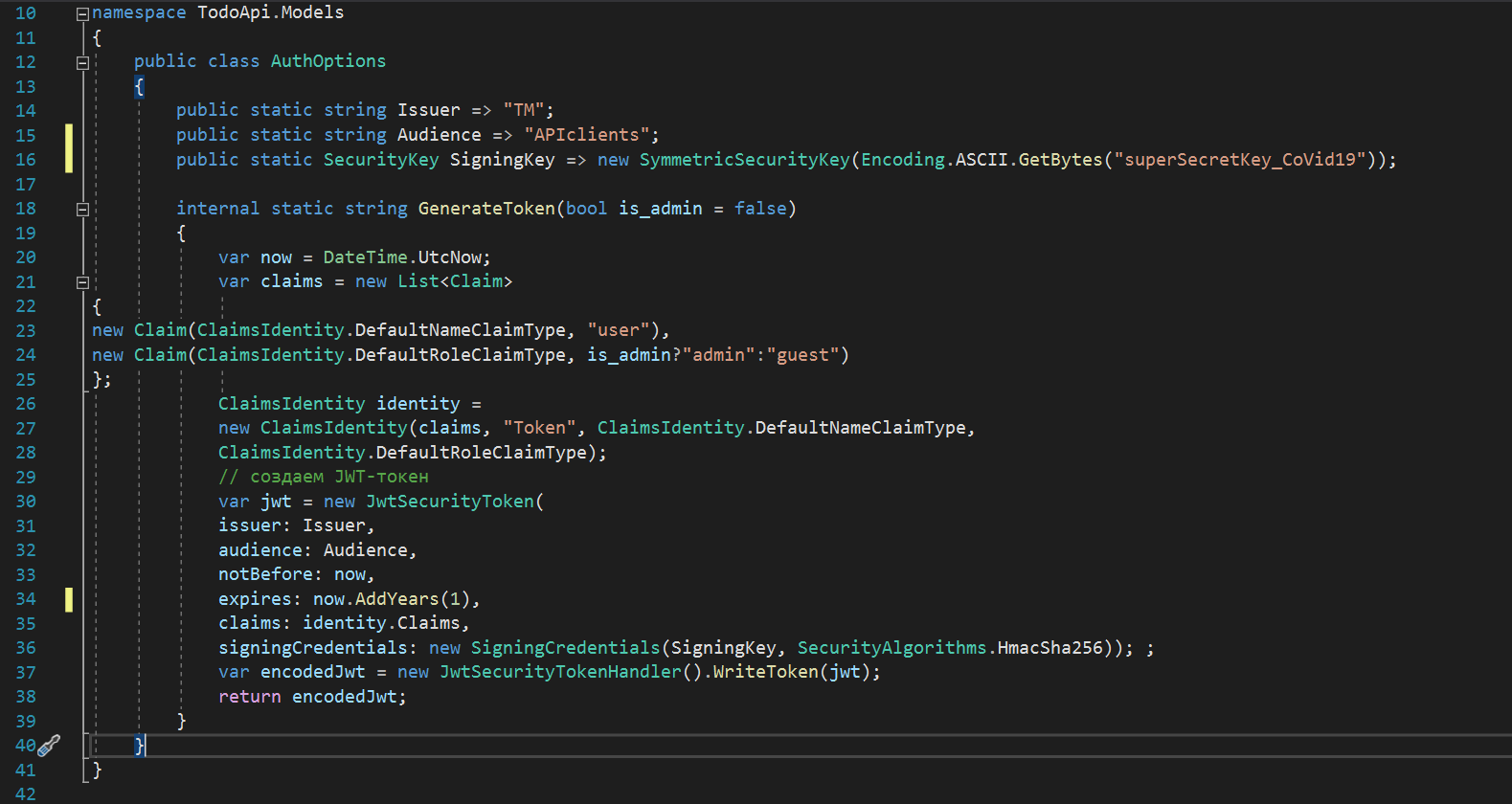


Рисунок 12

И для него контроллер AuthController

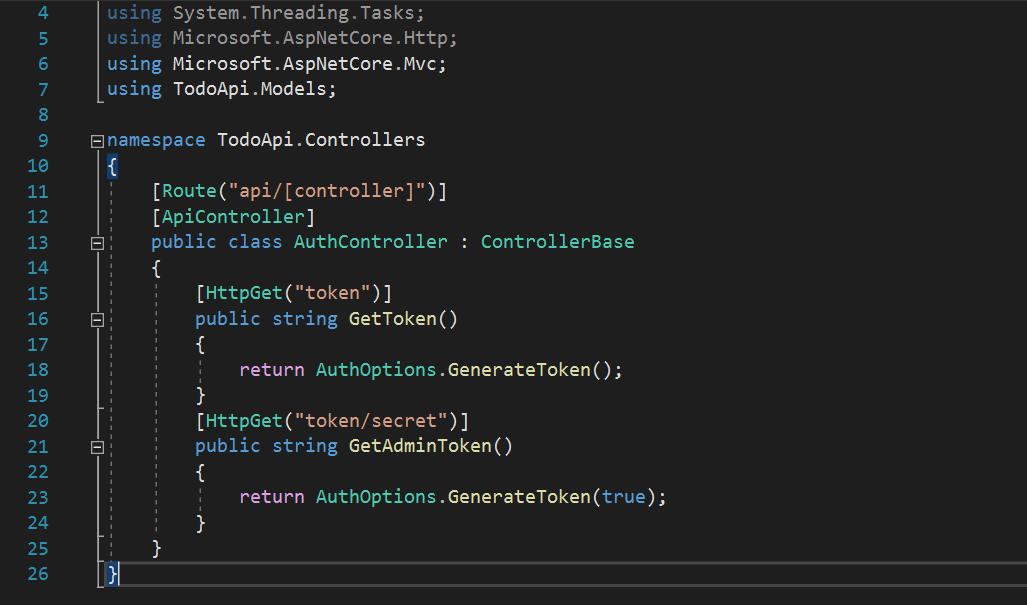


Рисунок 13

Для обработки запроса в контроллере создан метод Token, который сопоставлен с маршрутом «/token» и «/token/secret».

Теперь внесем изменения в DiariesController в связи с условием задачи: **получение сводных отчетов только для авторизованных пользователей, добавление данных - только для администраторов**

[Authorize(Roles = "admin")] для запросов put и post.

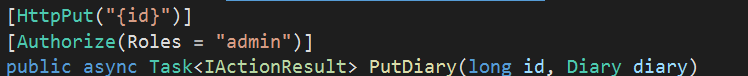


Рисунок 14

[Authorize] для запросов get

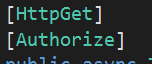


Рисунок 15

Проверим правильность работы с помощью Postman. Попробуем добавить и получить данные.

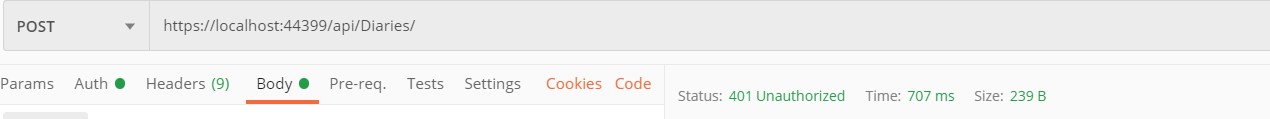


Рисунок 16

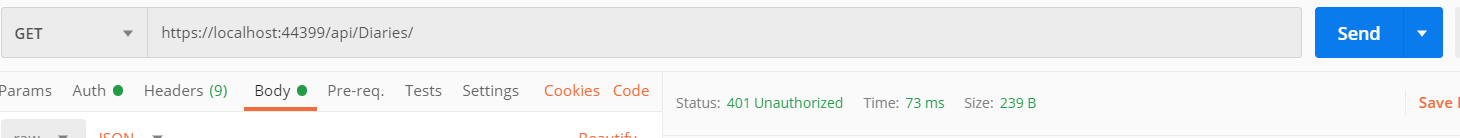


Рисунок 17

В обоих случаях ошибка: 401Unauthorized

Получим токен для авторизированного пользователя.

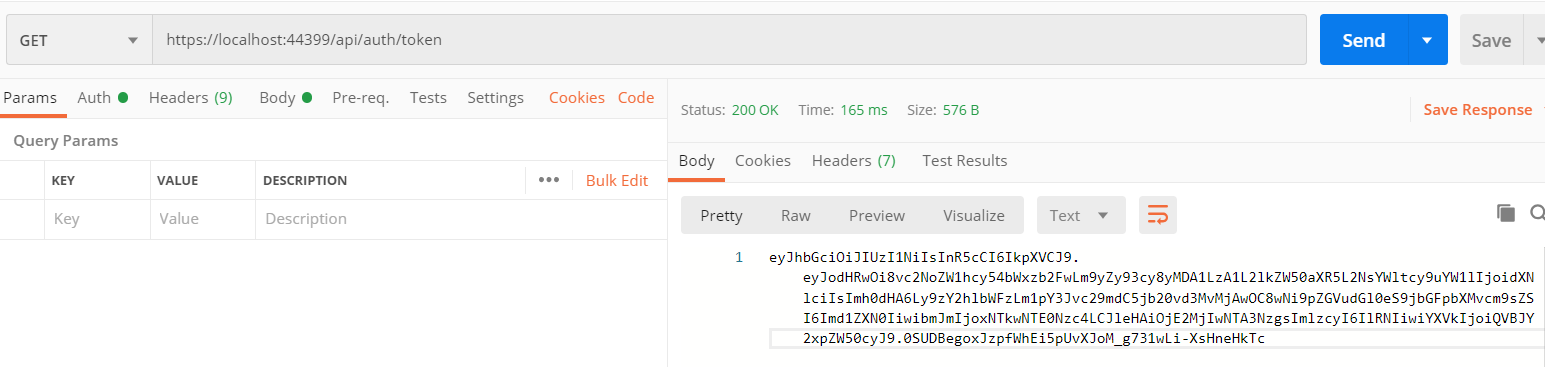


Рисунок 18

После чего скопируем сгенерированный токен и вставим его в поле auth.

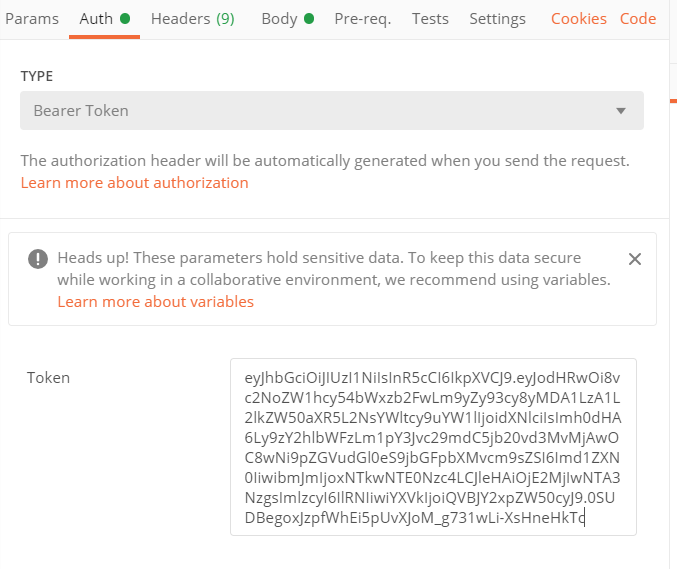


Рисунок 19

Теперь запрос выдает статус 200OK.

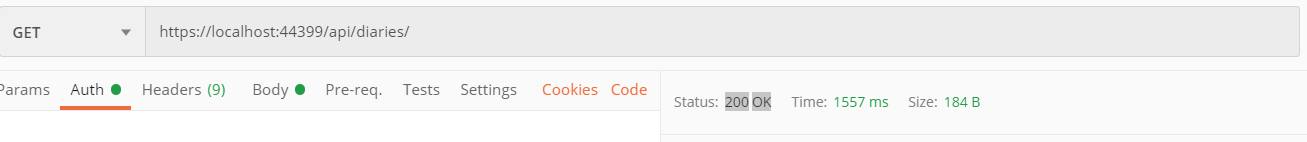


Рисунок 20

Получим токен для администратора и проделаем те же самые действия.

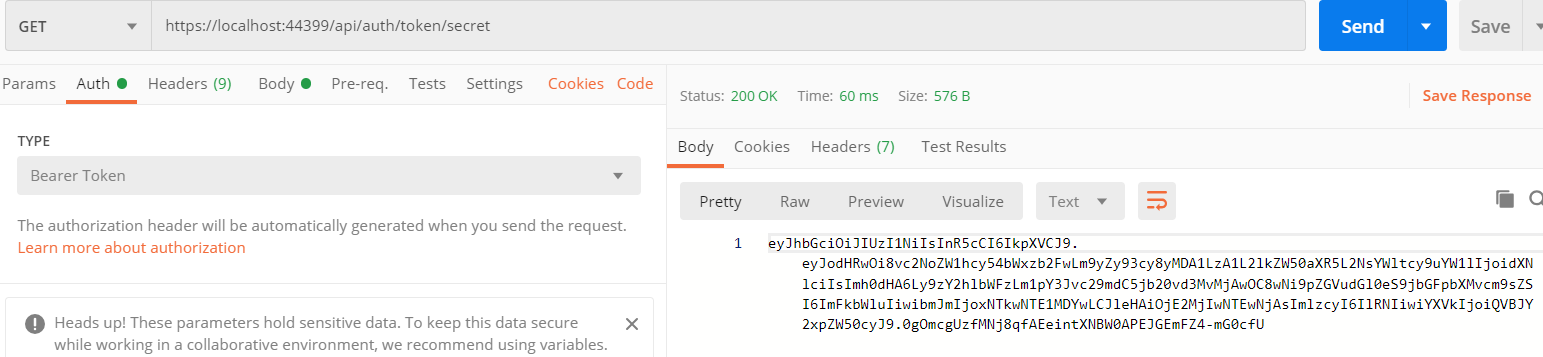


Рисунок 21

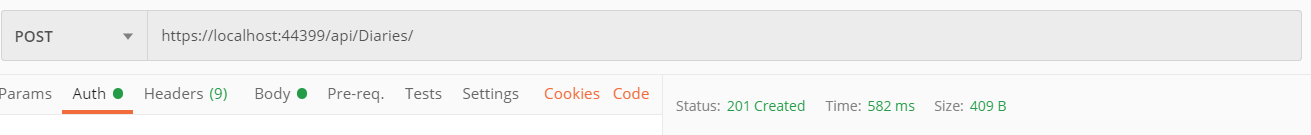


Рисунок 22

Теперь запрос выдает статус 201Created

Проверка пройдена успешно.

* 1. **Создать репозиторий, выложить код туда, прислать ссылку.**

<https://github.com/Dianaoleyn/ollab1>

* 1. **Собрать релиз-версию проекта и скинуть архив с релизом и набором запросов для тестирования.**

Набор запросов для тестирования:

{

"Date": "01.04.2020" ,

"Holiday": false ,

"Message":"Korona uydi",

"Tasks" : [ {

"Name": "plevat v potolok" ,

"IsComplete" : true

},

{

"Name": "spat" ,

"IsComplete" : false

}

]

}

api/Diaries/

api/Diaries/firstday

api/Diaries/holidays

api/auth/token

api/auth/token/secret