**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)
Санкт-Петербургский колледж телекоммуникаций им Э.Т. Кренкеля**

Отчет о выполнении итогового зачета

по

Прикладному программированию

Работа с данными

Принял:
Преподаватель Кривоносова Н.В.
Выполнил: студент группы КЗФ-051
Гаврилюк Вячеслав Викторович

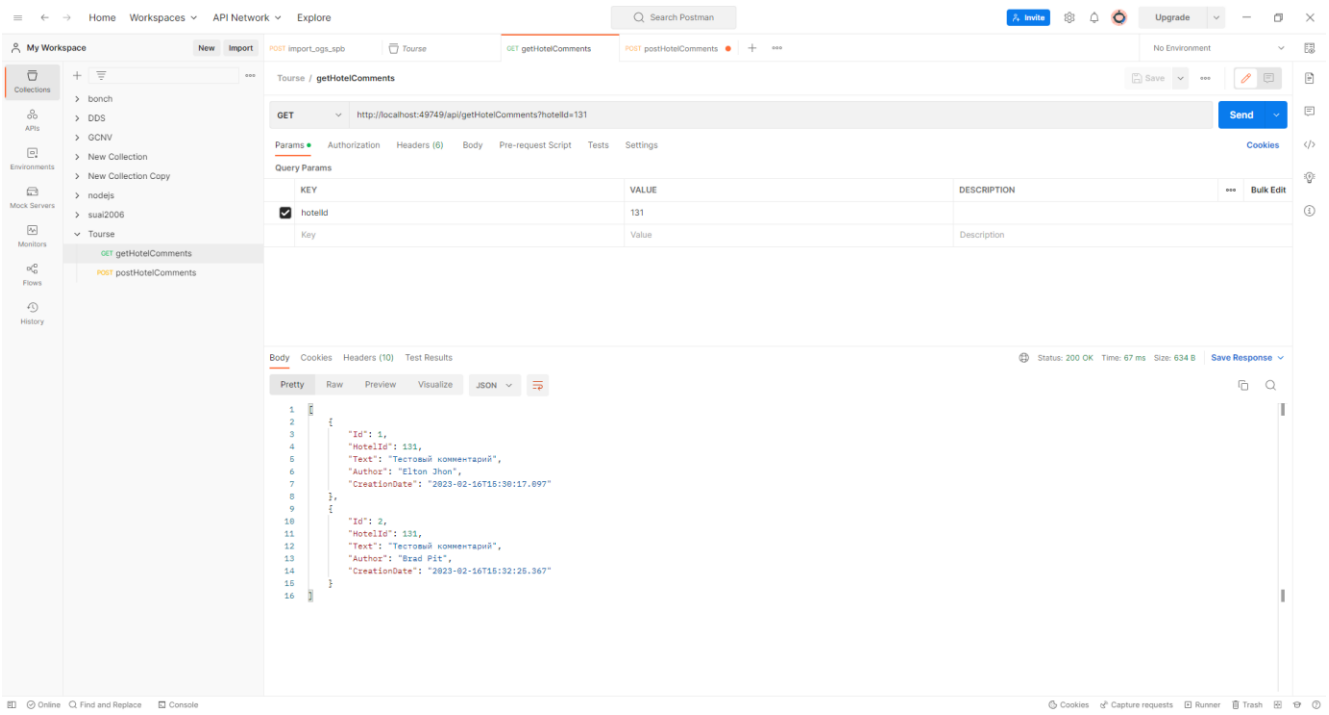
Санкт-Петербург
2023 год

Работа с данными.

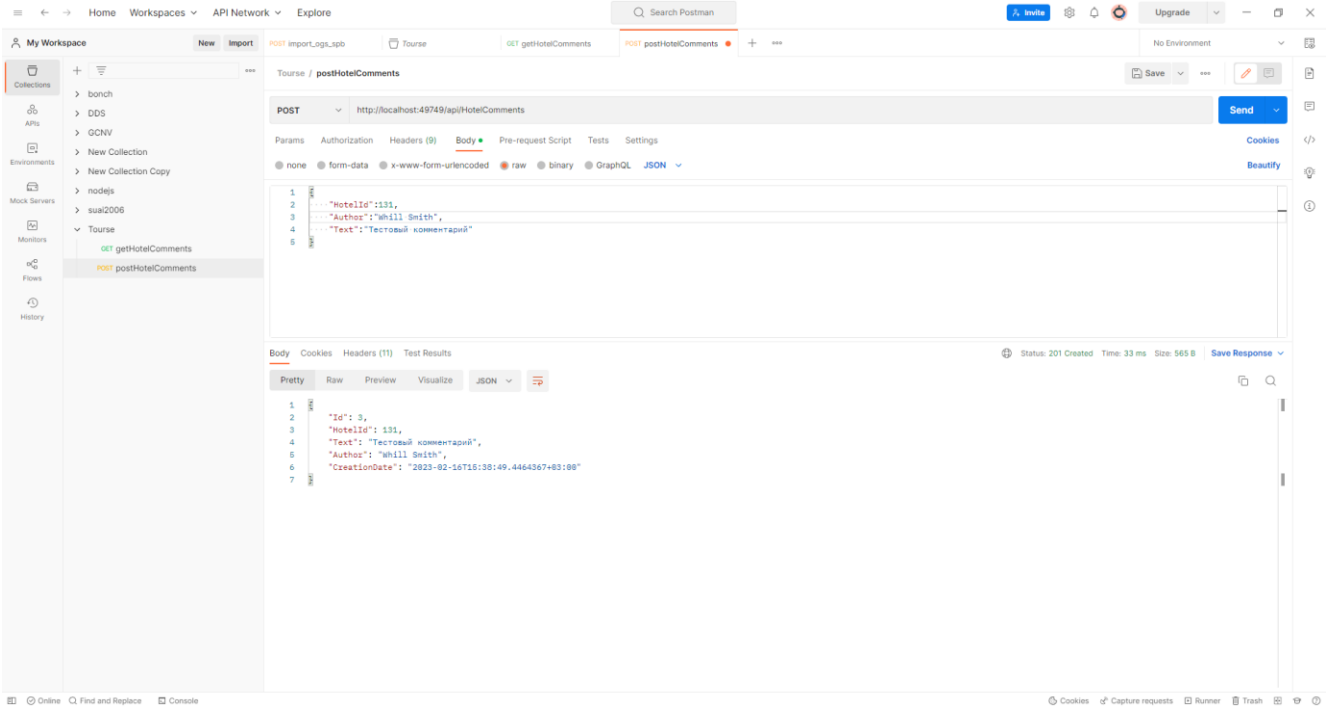
Задание.

Разработка API при взаимодействии с локальной БД

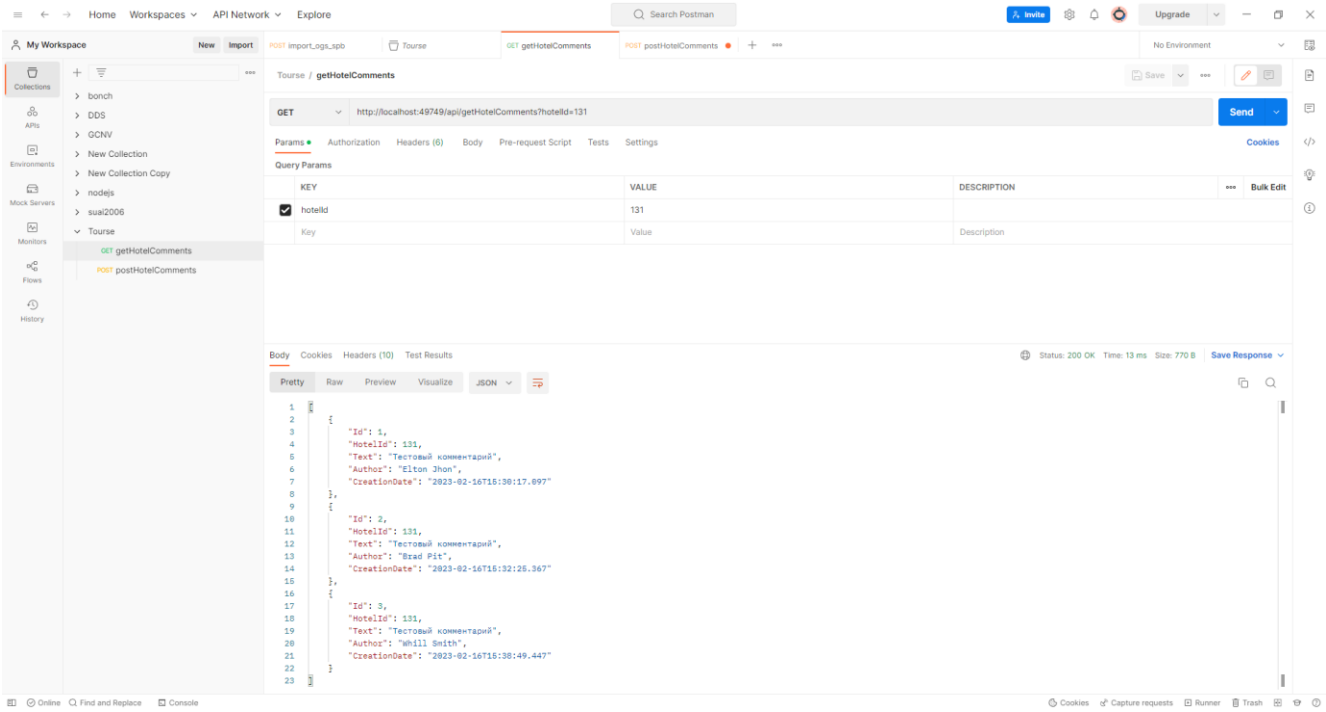
Пример GET запроса.



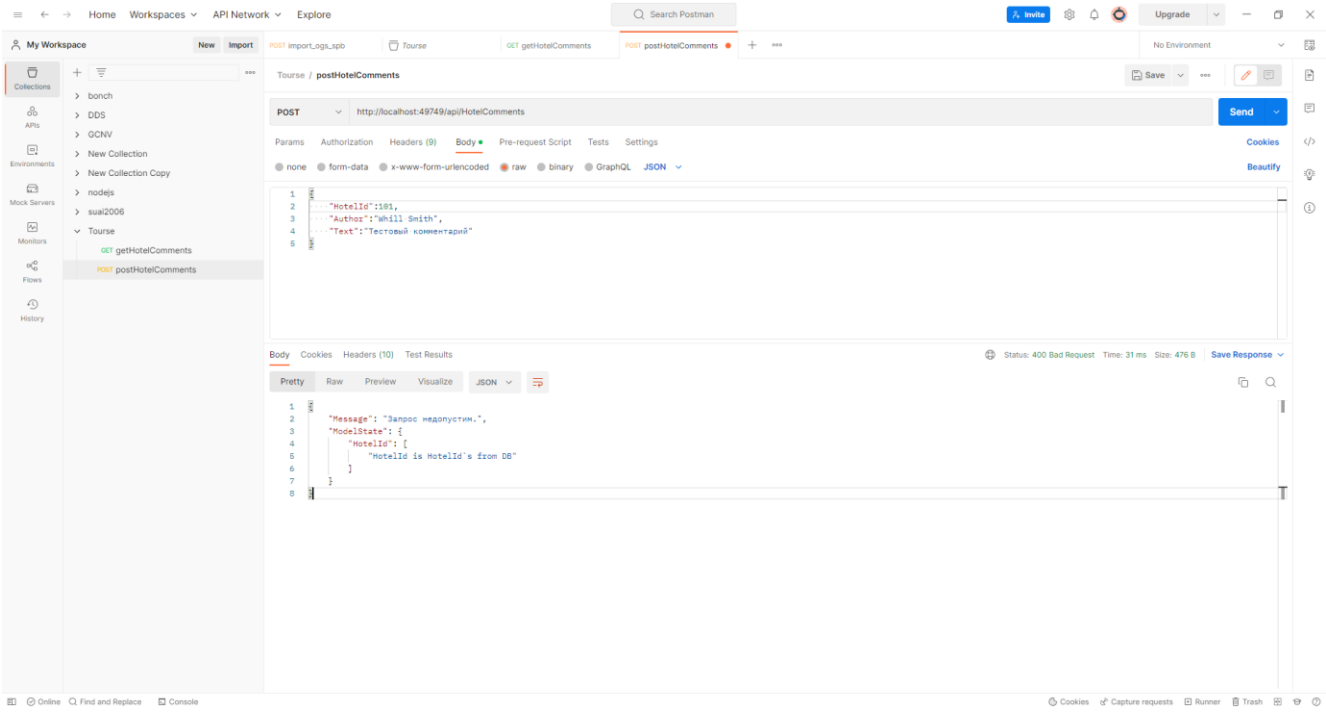
Пример POST запроса



Пример GET запроса.



Пример POST запроса с ошибкой



Листнинг кода

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Net;
using System.Net.Http;
```

```
using System.Web.Http;
using System.Web.Http.Description;
using WebAPI.Entities;
using WebAPI.Models;

namespace WebAPI.Controllers
{
    public class HotelsController : ApiController
    {
        private ToursBaseEntities db = new ToursBaseEntities();
        [ResponseType(typeof(List<ResponseHotel>))]
        // GET: api/Hotels
        public IHttpActionResult GetHotelComment()
        {
            return Ok(db.Hotel.ToList().ConvertAll(p => new ResponseHotel(p)));
        }

        // GET: api/Hotels/5
        [ResponseType(typeof(Hotel))]
        public IHttpActionResult GetHotel(int id)
        {
            Hotel hotel = db.Hotel.Find(id);
            if (hotel == null)
            {
                return NotFound();
            }

            return Ok(hotel);
        }

        // PUT: api/Hotels/5
        [ResponseType(typeof(void))]
        public IHttpActionResult PutHotel(int id, Hotel hotel)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            if (id != hotel.Id)
            {
                return BadRequest();
            }

            db.Entry(hotel).State = EntityState.Modified;

            try
            {
                db.SaveChanges();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!HotelExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return StatusCode(HttpStatusCode.NoContent);
        }
```

```
        // POST: api/Hotels
        [ResponseType(typeof(Hotel))]
        public IHttpActionResult PostHotel(Hotel hotel)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            db.Hotel.Add(hotel);
            db.SaveChanges();

            return CreatedAtRoute("DefaultApi", new { id = hotel.Id }, hotel);
        }

        // DELETE: api/Hotels/5
        [ResponseType(typeof(Hotel))]
        public IHttpActionResult DeleteHotel(int id)
        {
            Hotel hotel = db.Hotel.Find(id);
            if (hotel == null)
            {
                return NotFound();
            }

            db.Hotel.Remove(hotel);
            db.SaveChanges();

            return Ok(hotel);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }

        private bool HotelExists(int id)
        {
            return db.Hotel.Count(e => e.Id == id) > 0;
        }
    }
}


using Microsoft.Ajax.Utilities;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using System.Web.Http.Description;
using WebAPI.Entities;
using WebAPI.Models;

namespace WebAPI.Controllers
{
    public class HotelCommentsController : ApiController
```

```
    {
        private ToursBaseEntities db = new ToursBaseEntities();

        // GET: api/HotelComments
        public IQueryable<HotelComment> GetHotelComment()
        {
            return db.HotelComment;
        }
        [Route("api/getHotelComments")]
        public IHttpActionResult GetHotelComments(int hotelId)
        {
            var hotelComment = db.HotelComment.ToList().Where(p => p.HotelId ==
hotelId).ToList();
            return Ok(hotelComment);
        }

        // GET: api/HotelComments/5
        [ResponseType(typeof(HotelComment))]
        public IHttpActionResult GetHotelComment(int id)
        {
            HotelComment hotelComment = db.HotelComment.Find(id);
            if (hotelComment == null)
            {
                return NotFound();
            }

            return Ok(hotelComment);
        }

        // PUT: api/HotelComments/5
        [ResponseType(typeof(void))]
        public IHttpActionResult PutHotelComment(int id, HotelComment hotelComment)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            if (id != hotelComment.Id)
            {
                return BadRequest();
            }

            db.Entry(hotelComment).State = EntityState.Modified;

            try
            {
                db.SaveChanges();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!HotelCommentExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return StatusCode(HttpStatusCode.NoContent);
        }

        // POST: api/HotelComments
```

```
        [ResponseType(typeof(HotelComment))]
        public IHttpActionResult PostHotelComment(HotelComment hotelComment)
        {
            hotelComment.CreationDate = DateTime.Now;

            if (string.IsNullOrWhiteSpace(hotelComment.Author) ||
hotelComment.Author.Length > 100)
            {
                ModelState.AddModelError("Author", "Author is reqired string up to 100
symblos");
            }

            if (string.IsNullOrWhiteSpace(hotelComment.Text))
            {
                ModelState.AddModelError("Text", "Text is reqired string");
            }

            if (!(db.Hotel.ToList().FirstOrDefault(p => p.Id == hotelComment.HotelId) is
Hotel))
            {
                ModelState.AddModelError("HotelId", "HotelId is HotelId`s from DB");
            }

            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            db.HotelComment.Add(hotelComment);
            db.SaveChanges();

            return CreatedAtRoute("DefaultApi", new { id = hotelComment.Id },
hotelComment);
        }

        // DELETE: api/HotelComments/5
        [ResponseType(typeof(HotelComment))]
        public IHttpActionResult DeleteHotelComment(int id)
        {
            HotelComment hotelComment = db.HotelComment.Find(id);
            if (hotelComment == null)
            {
                return NotFound();
            }

            db.HotelComment.Remove(hotelComment);
            db.SaveChanges();

            return Ok(hotelComment);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
        private bool HotelCommentExists(int id)
        {
            return db.HotelComment.Count(e => e.Id == id) > 0;
        }
    }
}
```