ПРИЛОЖЕНИЕ Д. ТЕКСТ ПРОГРАММЫ

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Российский экономический университет имени Г.В. Плеханова»
Московский приборостроительный техникум

УТВЕРЖДЕН

РАЗРАБОТКА БАЗЫ ДАННЫХ И ВЕБ-API ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ПО ДОСТАВКЕ ГАЗА

Текст программы

2022

АННОТАЦИЯ

В данном программном документе приведен код программы веб-api для мобильного приложения по доставке газа.

В разделе «Текст программы» указаны следующие пункты:

- Наименование программы;
- Область применения программы;
- Модули;
- Код программы.

СОДЕРЖАНИЕ

1. ТЕКСТ ПРОГРАММЫ

1.1. Наименование программы

Наименование – «Разработка базы данных и веб-api для мобильного приложения по доставке газа»

1.2. Область применения программы

Программа предназначена для клиентов, желающих заказать баллоны газа через мобильное приложение и водителей, принимающих и выполняющих заказы клиентов.

1.3. Модули

Таблица 1- Модули

Модуль	Описание	Количество строк кода	Размер (в Кбайтах)
1	2	3	4
ClientController.cs	Контроллер клиента	144	5.24
DriverController.cs	Контроллер водителя	144	5.24
OrderController.cs	Контроллер заказов	148	6.26
ProductController.cs	Контроллер товаров	52	1.4

1.4. Код программы

ClientController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using GasDeliveryApi.Models;
using GasDeliveryApi.Models.Views;

namespace GasDeliveryApi.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
```

```
public class ClientController : ControllerBase
{
    /// <summary>
    /// Список пар: номер телефона + код авторизации, для верефикации телефона
    /// </summary>
    private static List<PhoneValid> phones = new List<PhoneValid>();

    /// <summary>
    /// Контекст базы данных
    /// </summary>
    private GasDeliveryDBContext _context;

    /// <summary>
    /// Инициализация контекста данных
    /// </summary>
    /// <param name="context">контекст данных</param>
    public ClientController(GasDeliveryDBContext context)
    {
        _context = context;
    }

    /// <summary>
    /// Метод, генерирующий код авторизации (в дальнейшем нужно код отправлять на номер телефона)
    /// </summary>
    /// <param name="phoneValid">Объект с номером телефона</param>
    /// <returns>Возвращает сгенерированный код, или http-код 404, если аккаунт с таким номером уже
есть</returns>
    [HttpPost]
    [Route("[action]")]
    public ActionResult<int> GetCode(PhoneValid phoneValid)
    {
        if (_context.Users.Where(c => c.Phone == phoneValid.Phone).FirstOrDefault() != null)
        {
            return NotFound();
        }

        var result = phones.Where(p => p.Phone == phoneValid.Phone).FirstOrDefault();
        if (result != null)
        {
            phones.Remove(result);
```

```
            }

            int code = new Random().Next(100000, 999999);

            phones.Add(new PhoneValid()
            {
                Phone = phoneValid.Phone,
                Code = code
            });

            return new ObjectResult(code);
        }

        /// <summary>
        /// Метод, проверяющий корректность ранее отправленного кода
        /// </summary>
        /// <param name="phoneValid">Объект с номером телефона и кодом авторизации</param>
        /// <returns>Возвращает http-код 200 или 404, в зависимости от успешности проверки кода</returns>
        [HttpPost]
        [Route("[action]")]
        public ActionResult CheckCode(PhoneValid phoneValid)
        {
            var result = phones.Where(p => p.Phone == phoneValid.Phone && p.Code == phoneValid.Code).FirstOrDefault();
            if (result == null)
            {
                return NotFound();
            }

            phones.Remove(result);
            return Ok();
        }

        /// <summary>
        /// Метод регистрации нового клиента
        /// </summary>
        /// <param name="clientView">Объект, содержащий все данные, необходимые для регистрации</param>
        /// <returns>Возвращает http-код 200, 404 или 500, в зависимости от выполнения регистрации</returns>
```

```
[HttpPost]
[Route("[action]")]
public ActionResult RegClient(ClientView clientView)
{
    if (_context.Users.Where(c => c.Phone == clientView.Phone).FirstOrDefault() != null)
    {
        return NotFound();
    }

    try
    {
        PersonalInfo personalInfo = new PersonalInfo()
        {
            LastName = clientView.LastName,
            FirstName = clientView.FirstName,
            Patronymic = clientView.Patronymic
        };
        _context.PersonalInfos.Add(personalInfo);

        User user = new User()
        {
            Phone = clientView.Phone,
            RoleId = _context.UserRoles.Where(u => u.Name == "Клиент").Select(u => u.Id).FirstOrDefault()
        };
        _context.Users.Add(user);

        Client client = new Client()
        {
            PersonalInfo = personalInfo,
            User = user
        };
        _context.Clients.Add(client);

        _context.SaveChanges();

        return Ok();
    }
    catch
    {
        return StatusCode(500);
```

```
            }
        }

        /// <summary>
        /// Тестовый метод, выводящий список пар: номер телефона + код авторизации
        /// </summary>
        /// <returns>Список пар: номер телефона + код авторизации</returns>
        [HttpGet]
        [Route("[action]")]
        public IEnumerable<PhoneValid> GetPhones()
        {
            return phones;
        }
    }
}
```

DriverController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using GasDeliveryApi.Models;
using GasDeliveryApi.Models.Views;

namespace GasDeliveryApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class DriverController : ControllerBase
    {
        /// <summary>
        /// Список пар: номер телефона + код авторизации, для верефикации телефона
        /// </summary>
        private static List<PhoneValid> phones = new List<PhoneValid>();

        /// <summary>
        /// Контекст базы данных
        /// </summary>
```

```
        private GasDeliveryDBContext _context;

        /// <summary>
        /// Инициализация контекста данных
        /// </summary>
        /// <param name="context">контекст данных</param>
        public DriverController(GasDeliveryDBContext context)
        {
            _context = context;
        }

        /// <summary>
        /// Метод, генерирующий код авторизации (в дальнейшем нужно код отправлять на номер телефона)
        /// </summary>
        /// <param name="phoneValid">Объект с номером телефона</param>
        /// <returns>Возвращает сгенерированный код, или http-код 404, если аккаунт с таким номером уже
есть</returns>
        [HttpPost]
        [Route("[action]")]
        public ActionResult<int> GetCode(PhoneValid phoneValid)
        {
            if (_context.Users.Where(u => u.Phone == phoneValid.Phone).FirstOrDefault() != null)
            {
                return NotFound();
            }

            var result = phones.Where(p => p.Phone == phoneValid.Phone).FirstOrDefault();
            if (result != null)
            {
                phones.Remove(result);
            }

            int code = new Random().Next(100000, 999999);

            phones.Add(new PhoneValid()
            {
                Phone = phoneValid.Phone,
                Code = code
            });
```

```
            return new ObjectResult(code);
        }

        /// <summary>
        /// Метод, проверяющий корректность ранее отправленного кода
        /// </summary>
        /// <param name="phoneValid">Объект с номером телефона и кодом авторизации</param>
        /// <returns>Возвращает http-код 200 или 404, в зависимости от успешности проверки кода</returns>
        [HttpPost]
        [Route("[action]")]
        public ActionResult CheckCode(PhoneValid phoneValid)
        {
            var result = phones.Where(p => p.Phone == phoneValid.Phone && p.Code == phoneValid.Code).FirstOrDefault();
            if (result == null)
            {
                return NotFound();
            }

            phones.Remove(result);
            return Ok();
        }

        /// <summary>
        /// Метод регистрации нового водителя
        /// </summary>
        /// <param name="clientView">Объект, содержащий все данные, необходимые для регистрации</param>
        /// <returns>Возвращает http-код 200, 404 или 500, в зависимости от выполнения регистрации</returns>
        [HttpPost]
        [Route("[action]")]
        public ActionResult RegDriver(DriverView driverView)
        {
            if (_context.Users.Where(c => c.Phone == driverView.Phone).FirstOrDefault() != null)
            {
                return NotFound();
            }

            try
```

```
            {
                PersonalInfo personalInfo = new PersonalInfo()
                {
                    LastName = driverView.LastName,
                    FirstName = driverView.FirstName,
                    Patronymic = driverView.Patronymic
                };
                _context.PersonalInfos.Add(personalInfo);

                User user = new User()
                {
                    Phone = driverView.Phone,
                    RoleId = _context.UserRoles.Where(u => u.Name == "Водитель").Select(u => u.Id).FirstOrDefault()
                };
                _context.Users.Add(user);

                Driver driver = new Driver()
                {
                    PersonalInfo = personalInfo,
                    User = user
                };
                _context.Drivers.Add(driver);

                _context.SaveChanges();

                return Ok();
            }
            catch
            {
                return StatusCode(500);
            }
        }

        /// <summary>
        /// Тестовый метод, выводящий список пар: номер телефона + код авторизации
        /// </summary>
        /// <returns>Список пар: номер телефона + код авторизации</returns>
        [HttpGet]
        [Route("[action]")]
```

```
        public IEnumerable<PhoneValid> GetPhones()
        {
            return phones;
        }
    }
}
```

OrderController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using GasDeliveryApi.Models;
using GasDeliveryApi.Models.Views;

namespace GasDeliveryApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrderController : ControllerBase
    {
        /// <summary>
        /// Контекст базы данных
        /// </summary>
        private GasDeliveryDBContext _context;

        /// <summary>
        /// Инициализация контекста данных
        /// </summary>
        /// <param name="context">контекст данных</param>
        public OrderController(GasDeliveryDBContext context)
        {
            _context = context;
        }

        /// <summary>
        /// Регистрация нового заказа
        /// </summary>
```

```
        /// <param name="orderView">Объект, содержащий все данные, необходимые для регистрации</param>
        /// <returns></returns>
        [HttpPost]
        [Route("[action]")]
        public ActionResult RegOrder(OrderView orderView)
        {
            try
            {
                var address = _context.AddressDeliveries.Where(a => a.Address == orderView.Address.Address &&
                                        a.ApartmentNum == orderView.Address.ApartmentNum &&
                                        a.FloorNum == orderView.Address.FloorNum &&
                                        a.FrontDoorNum == orderView.Address.FrontDoorNum).FirstOrDefault();
                if (address == null)
                {
                    _context.AddressDeliveries.Add(orderView.Address);
                }

                var client = _context.Clients.Where(c => c.User.Phone == orderView.ClientPhone).FirstOrDefault();
                if (client == null)
                {
                    return StatusCode(500);
                }

                Ordered order = new Ordered()
                {
                    Address = orderView.Address,
                    DateDelivery = DateTime.Parse(orderView.DateDelivery),
                    DesiredTimeFrom = TimeSpan.Parse(orderView.DesiredTimeFrom),
                    DesiredTimeTo = TimeSpan.Parse(orderView.DesiredTimeTo),
                    Sum = 0.0,
                    Client = client,
                    Status = _context.OrderStatuses.Where(s => s.Name == "Обрабатывается").FirstOrDefault()
                };
                _context.Ordereds.Add(order);

                double allSumm = 0.0;
                foreach (var i in orderView.Products)
                {
                    OrderCompo orderCompos = new OrderCompo()
```

```
                    {
                        Order = order,
                        ProductId = i.Id,
                        Quantity = i.Quantity,
                        Sum = _context.Products.Where(p => p.Id == i.Id).Select(p => p.Price).FirstOrDefault() *
i.Quantity
                    };
                    _context.OrderCompos.Add(orderCompos);
                    allSumm += orderCompos.Sum;
                }
                order.Sum = allSumm;

                _context.SaveChanges();
                return Ok();
            }
            catch
            {
                return StatusCode(500);
            }
        }

        [HttpGet]
        [Route("[action]")]
        public ActionResult<OrderView> GetFreeOrders()
        {
            var result = (from order in _context.Ordereds
                          where order.Status == _context.OrderStatuses.Where(s => s.Name ==
"Обрабатывается").FirstOrDefault()
                          select new OrderView()
                          {
                              Id = order.Id,
                              Address = order.Address,
                              ClientLastName = order.Client.PersonalInfo.LastName,
                              ClientFirstName = order.Client.PersonalInfo.FirstName,
                              ClientPhone = order.Client.User.Phone,
                              DateDelivery = order.DateDelivery.ToString("dd.MM.yyyy"),
                              DesiredTimeFrom = order.DesiredTimeFrom.ToString(),
                              DesiredTimeTo = order.DesiredTimeTo.ToString(),
                              Summ = order.Sum,
                              OrderStatus = order.Status.Name,
```

```
                    Products = (from orderCompos in _context.OrderCompos
                                where orderCompos.OrderId == order.Id
                                select new OrderProductView()
                                {
                                    Id = orderCompos.Product.Id,
                                    Name = orderCompos.Product.Name,
                                    Quantity = orderCompos.Quantity,
                                    Summ = orderCompos.Sum
                                }).ToList()
                    }).ToList();
            return new ObjectResult(result);
        }

        [HttpPost]
        [Route("[action]")]
        public ActionResult<Ordered> SetDriver(DriverOrderView driverOrder)
        {
            var orderDb = (from order in _context.Ordereds
                           where order.Id == driverOrder.OrderId
                           select order).FirstOrDefault();

            var driverDb = (from driver in _context.Drivers
                            where driver.User.Phone == driverOrder.DriverPhone
                            select driver).FirstOrDefault();

            orderDb.DriverId = driverDb.Id;
            orderDb.ExactTime = TimeSpan.Parse(driverOrder.ExactTime);
            orderDb.StatusId = _context.OrderStatuses.Where(s => s.Name == "Принят в работу").Select(s =>
s.Id).FirstOrDefault();
            _context.Ordereds.Update(orderDb);
            _context.SaveChanges();

            var result = (from order in _context.Ordereds
                          where order.Id == driverOrder.OrderId
                          select order).FirstOrDefault();

            return new ObjectResult(result);
        }
    }
}
```

ProductController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using GasDeliveryApi.Models;
using GasDeliveryApi.Models.Views;

namespace GasDeliveryApi.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class ProductController : ControllerBase
   {
      /// <summary>
      /// Контекст базы данных
      /// </summary>
      private GasDeliveryDBContext _context;

      /// <summary>
      /// Инициализация контекста данных
      /// </summary>
      /// <param name="context">контекст данных</param>
      public ProductController(GasDeliveryDBContext context)
      {
         _context = context;
      }

      [HttpPost]
      [Route("[action]")]
      public ActionResult AddProduct(Product product)
      {
         try
         {
            Product newProduct = new Product()
            {
               Name = product.Name,
               Price = product.Price,
```

```
                Description = product.Description
            };
            _context.Products.Add(newProduct);
            _context.SaveChanges();
            return Ok();
        }
        catch
        {
            return StatusCode(500);
        }
    }
  }
}
```