Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

Кафедра информационных систем и программной инженерии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

«Технологии программирования»

на тему

Разработка регистрационной системы   
«Рекламное агентство»

Выполнил: студентка группы ИСТ-116 Кислова А.В.

Принял: доцент Вершинин В.В.

Владимир, 2018

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc534667309)

[1 ВВЕДЕНИЕ 2](#_Toc534667310)

[2 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc534667311)

[3 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc534667312)

[**3.1 Словарь предметной области** 4](#_Toc534667313)

[**3.2 Сценарий взаимодействия пользователей с системой** 4](#_Toc534667314)

[4 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 6](#_Toc534667315)

[**4.1 Создание диаграмм UML** 6](#_Toc534667316)

[**4.1.1 Диаграмма прецедентов** 6](#_Toc534667317)

[**4.1.2 Диаграмма последовательностей** 8](#_Toc534667318)

[**4.1.3 Диаграмма состояний** 9](#_Toc534667319)

[**4.1.4 Диаграмма классов** 10](#_Toc534667320)

[**4.1.5 Физическая модель базы данных** 10](#_Toc534667321)

[5 РЕАЛИЗАЦИЯ СИСТЕМЫ 12](#_Toc534667322)

[**5.1 Общие принципы организации системы** 12](#_Toc534667323)

[**5.2 Реализация модели** 12](#_Toc534667324)

[**5.3 Реализация механизма регистрации, аутентификации и авторизации** 12](#_Toc534667325)

[**5.4 Пользовательский интерфейс.** 12](#_Toc534667326)

[6 ЗАКЛЮЧЕНИЕ 17](#_Toc534667327)

[7 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc534667328)

[ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММНОЙ СИСТЕМЫ 19](#_Toc534667329)

[ПРИЛОЖЕНИЕ Б СТРУКТУРА БАЗЫ ДАННЫХ 28](#_Toc534667330)

# 1 ВВЕДЕНИЕ

В настоящее время практически полностью отсутствует монополизация.

Это помогает развиваться абсолютно любой сфере, т.к. любой человек может открыть компанию и заняться любимым делом. Это был плюс, очевидно, присутствует и минус: рынок насыщен организациями, предлагающими одинаковые услуги, как следствие, конкуренция высока. Сложно найти какой-либо товар, который производит лишь одна фирма.

Для того, чтобы клиент (покупатель) приобрёл именно твой товар, необходимо произвести качественную рекламу. Это берут на себя рекламные агентства.

Как уже было описано выше, монополизация отсутствует и среди рекламных агентств. Чтобы стать лидером в этой сфере, необходимо получать качественный результат в более сжатые сроки. Одна из возможностей повысить качество и понизить сроки выполнения – исключение ошибок, возникающих по вине человеческого фактора, то есть автоматизировать максимум процессов в компании.

# 2 ПОСТАНОВКА ЗАДАЧИ

В процессе разработки прототипа программной системы необходимо:

1. выполнить исследование и анализ предметной области;
2. разработать прототип ПС;
3. выполнить моделирование работы ПС;
4. разработать схему базы данных;
5. реализовать ПС с использованием выбранных средств и технологий.

Исходные данные:

1. язык моделирования UML;
2. платформа разработки ASP.NET;

Минимальный набор функций: поддержка различных типов пользователей; поддержка справочника менеджеров; поддержка справочника категорий рекламы; поддержка справочника клиентов; оформление заказа на размещение рекламы;

# 3 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Программная система «Рекламное агентство» поддерживает три типа пользователей: заказчик, менеджер, администратор. Заказчик делает заказ на рекламу. Менеджер просматривает заказы и переводит на различные статусы. Администратор удаляет пользователей и добавляет менеджеров.

## **3.1 Словарь предметной области**

* Заказ – набор информации, содержащий в себе информацию о рекламе.
* Заказчик – пользователь, авторизовавшийся под ролью «заказчик». Может создавать заказы на рекламу.
* Менеджер – пользователь, авторизовавшийся под ролью «менеджер». Может просматривать заказы и менять их статус.
* Администратор – пользователь, авторизовавшийся под ролью «администратор». Может просматривать список пользователей системы и удалять их, так же добавлять менеджера.

## **3.2 Сценарий взаимодействия пользователей с системой**

Взаимодействие пользователей с системой осуществляется посредствам интернет браузера.

Каждый пользователь имеет определенную роль, которая определяет его возможности. Можно выделить следующие роли и их возможности:

Заказчик

* Создать заказ
* Просмотреть типы рекламы

Менеджер

* Просмотреть заказ
* Присвоить статус заказу

Администратор

* Просмотреть список пользователей
* Удалить пользователя
* Добавить менеджера

Также, каждый зарегистрированный пользователь имеет возможность авторизоваться в системе и выйти из своего аккаунта.

# 4 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## **4.1 Создание диаграмм UML**

### **4.1.1 Диаграмма прецедентов**

**Наименование:** «Сделать заказ»

**Предусловие:** действующее лицо должно быть авторизированно в системе как заказчик.

**Действующее лицо:** заказчик

**Основной поток:** отправить заказ

Заказчик открывает шаблон заказа, заполняет данные об организации в соответствующие поля. После заполнения заказчик нажимает кнопку «Отправить».

**Альтернативный поток:** не все поля заполнены.

Заказчик заполнил не все необходимые поля. В этом случае он заполняет поля, пропущенные ранее.

**Альтернативный поток:** пользователь не авторизирован

Пользователь авторизуется.

**Постусловие:** если все необходимые поля заполнены, данные отправляются в базу данных. Заказу присваивается статус «заявка создана».

**Наименование:** «Удалить пользователя»

**Предусловие:** пользователь должен быть авторизован как администратор.

**Действующее лицо:** администратор

**Основной поток:** удаление пользователя

Администратор удаляет любого пользователя.

**Постусловие:** пользователь и все данные о нем, как и заказы у заказчиков, удаляются.

**Наименование:** «Изменить статус»

**Предусловие:** пользователь должен быть авторизован как менеджер

**Действующее лицо:** менеджер

**Основной поток:** изменение статуса

Менеджер меняет статус заказа на необходимый

**Постусловие:** статус заказа в бд меняется.

**Наименование:** «Зарегистрироваться»

**Предусловие:** номер телефона или электронная почта пользователя не должна быть зарегистрирована в системе.

**Действующее лицо:** Пользователь

**Основной поток:** Регистрация

Пользователю присваиваются логин и пароль

**Альтернативный поток:** номер телефона или эл.почта зарегистрированы в системе

Пользователь заново вводит данные для регистрации.

**Постусловие:** если введен уникальный номер телефона или эл.почта, пользователь получает соответствующие права.

**Наименование:** «Зарегистрировать менеджера»

**Предусловие:** данный менеджер не должен быть зарегистрирован в системе.

**Действующее лицо:** администратор

**Основной поток:** Регистрация

Пользователю присваиваются логин и пароль

**Альтернативный поток:** -

**Постусловие:** менеджер получает соответствующие права.

Диаграмма прецедентов представлена на рисунке 1.

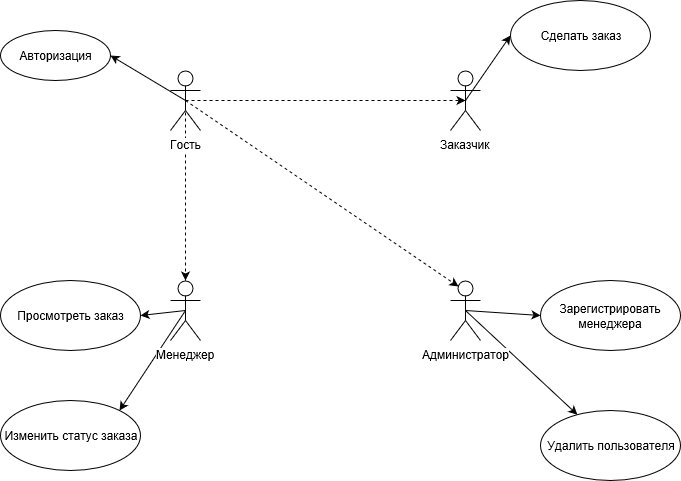


Рисунок 1. Диаграмма прецедентов

### **4.1.2 Диаграмма последовательностей**

Описание взаимодействий:

Заказчик открывает заявку и заполняет все данные. Заявка ждет сохранения, если все данные введены верно и заявка сохранена, то она переходит в статус «создана» иначе – остается в статусе «заполнена». Менеджер подтверждает актуальность заявки и либо удаляет её, либо отправляет в разработку. После этого заявка закрывается.

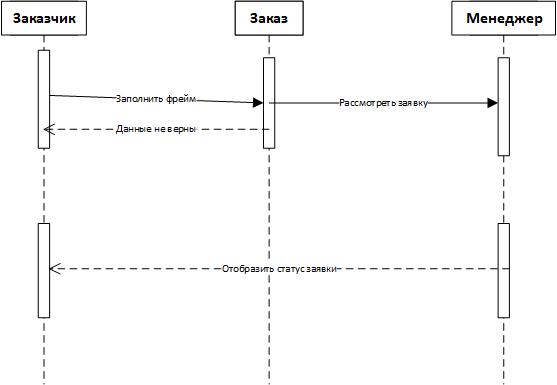


Рисунок 2. Диаграмма последовательностей для прецедента «Сделать заказ»

### **4.1.3 Диаграмма состояний**

Диаграмма состояний объекта «Заказ» представлена на рисунке 3.

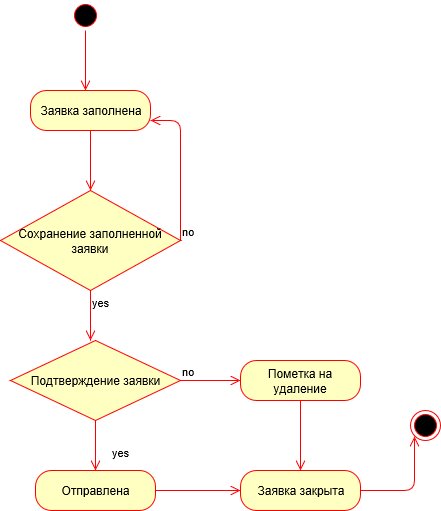


Рисунок 3. Диаграмма состояний заказ

### **4.1.4 Диаграмма классов**

Диаграмма классов демонстрирует классы системы, их атрибуты, методы и взаимосвязи между ними.

Диаграмма классов представлена на рисунке 4.

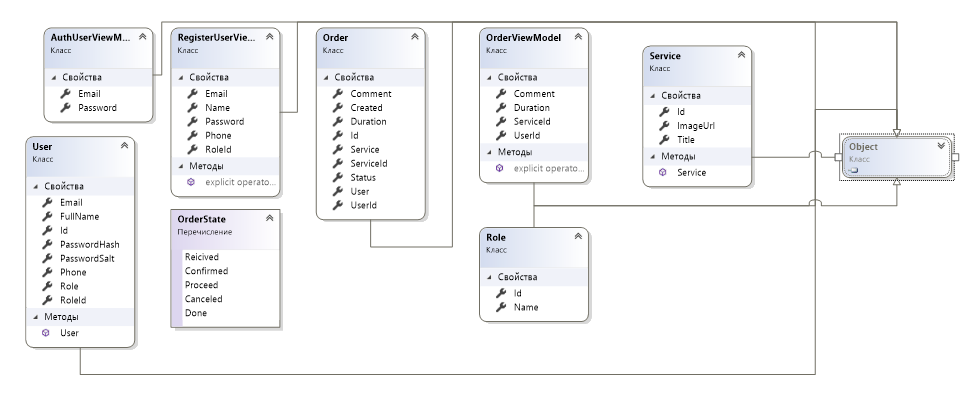


Рисунок 4. Диаграмма классов

### **4.1.5 Физическая модель базы данных**

Физическая модель базы данных представлена на рисунке 5. Дамп базы данных представлен в приложении Б.

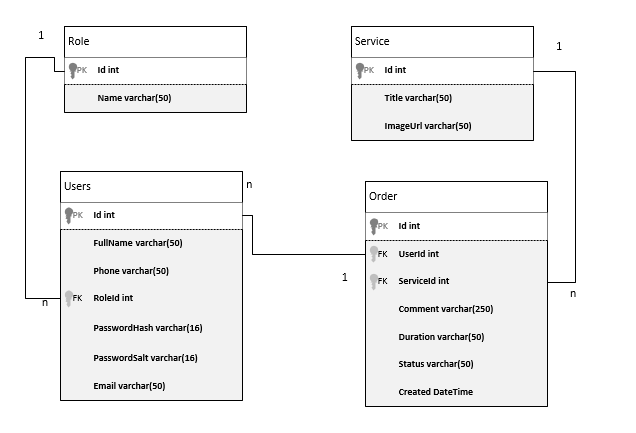


Рисунок 5. Физическая модель базы данных

# 5 РЕАЛИЗАЦИЯ СИСТЕМЫ

## **5.1 Общие принципы организации системы**

Разрабатываемая система реализована на платформе .NET Core, с использованием Entity Framework Core. Для представлений был использован ReactJS. В качестве хранилища выступает MS SQL. Вся разработка велась в Visual Studio Code.

## **5.2 Реализация модели**

Сначала были созданы все основные модели системы. Далее они были прописаны в контексте базы данных. С помощью инструмента миграций была сгенерирована база данных

## **5.3 Реализация механизма регистрации, аутентификации и авторизации**

Для реализации регистрации, аутентификации и авторизации пользователей были использованы стандартные методы шаблона Microsoft Identity.

Методы и представления имеют ограниченный доступ. Пользователям, в зависимости от роли, доступны различные функции. Это обеспечивается ограничениями, реализованными с помощью встроенных средств. Регистрацию менеджеров в системе производит администратор, заказчики же регистрируются сами.

## **5.4 Пользовательский интерфейс.**

Пользовательский интерфейс представлен на рисунках 6-13.

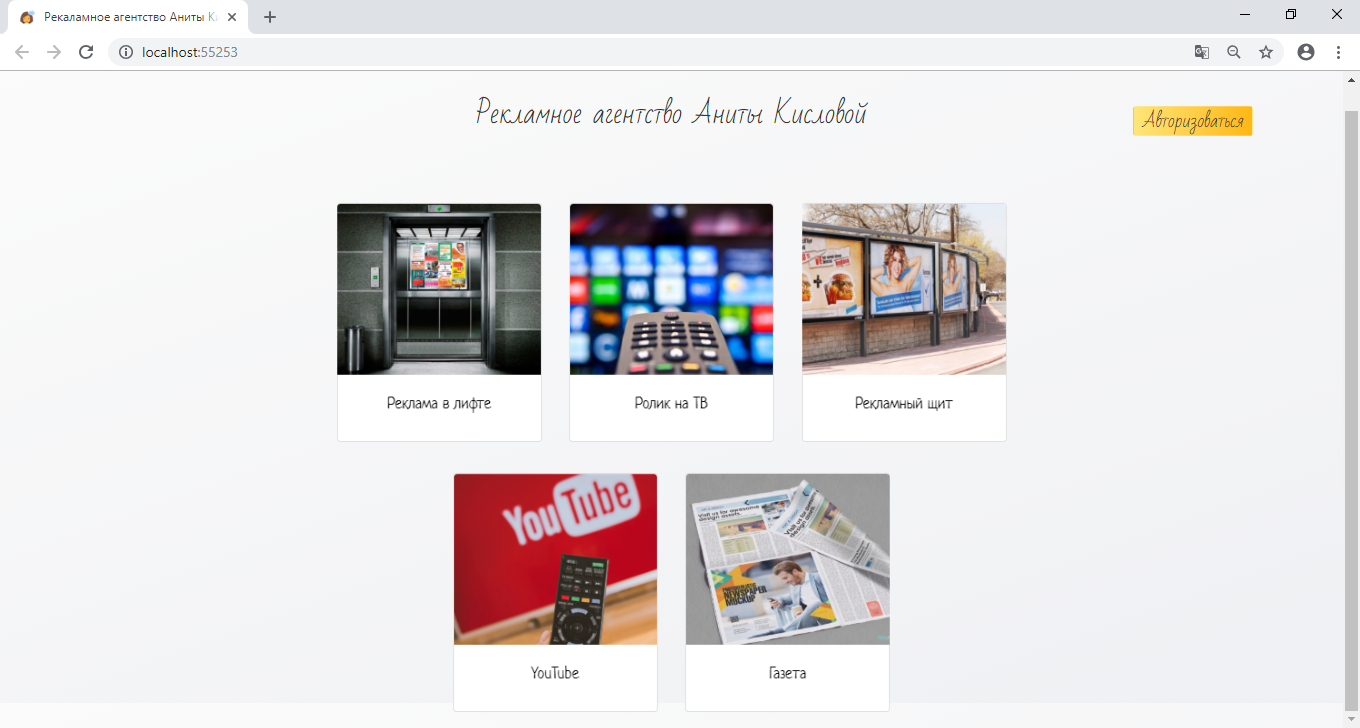


Рисунок 6. Главная страница для гостя

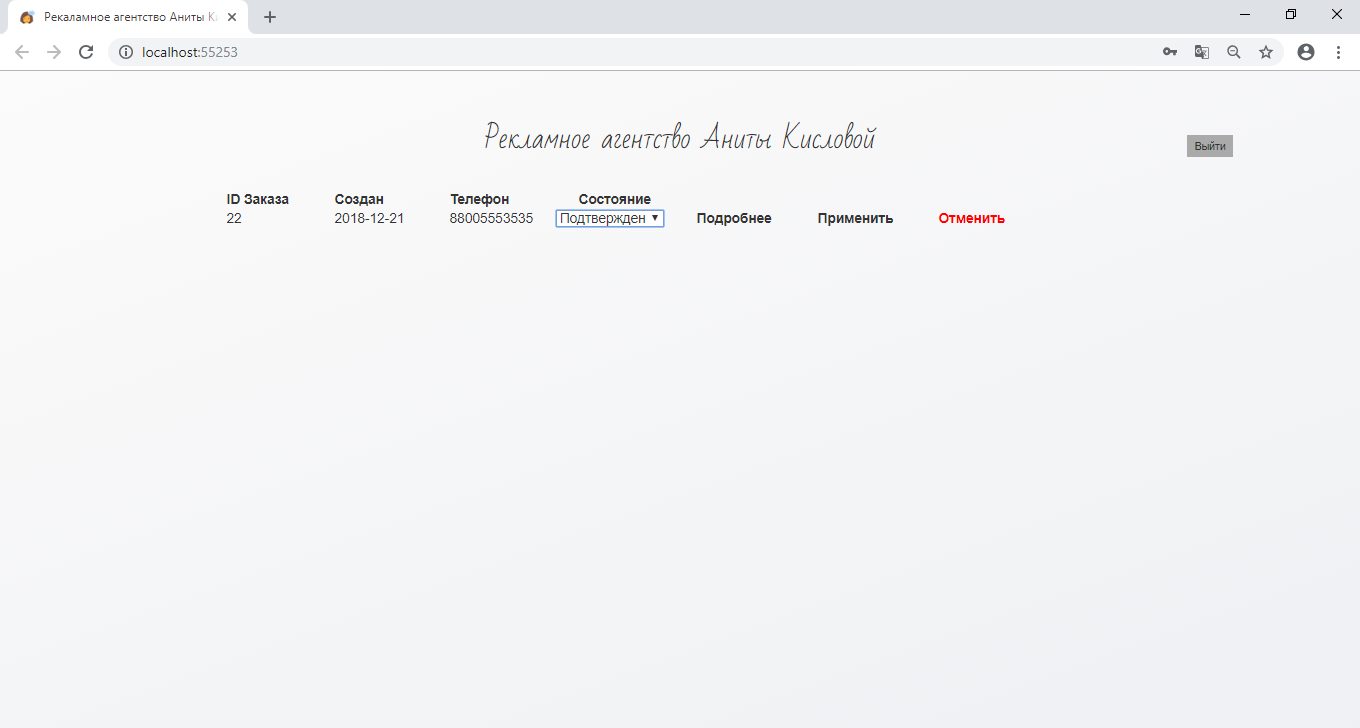


Рисунок 7. Главная страница для менеджера

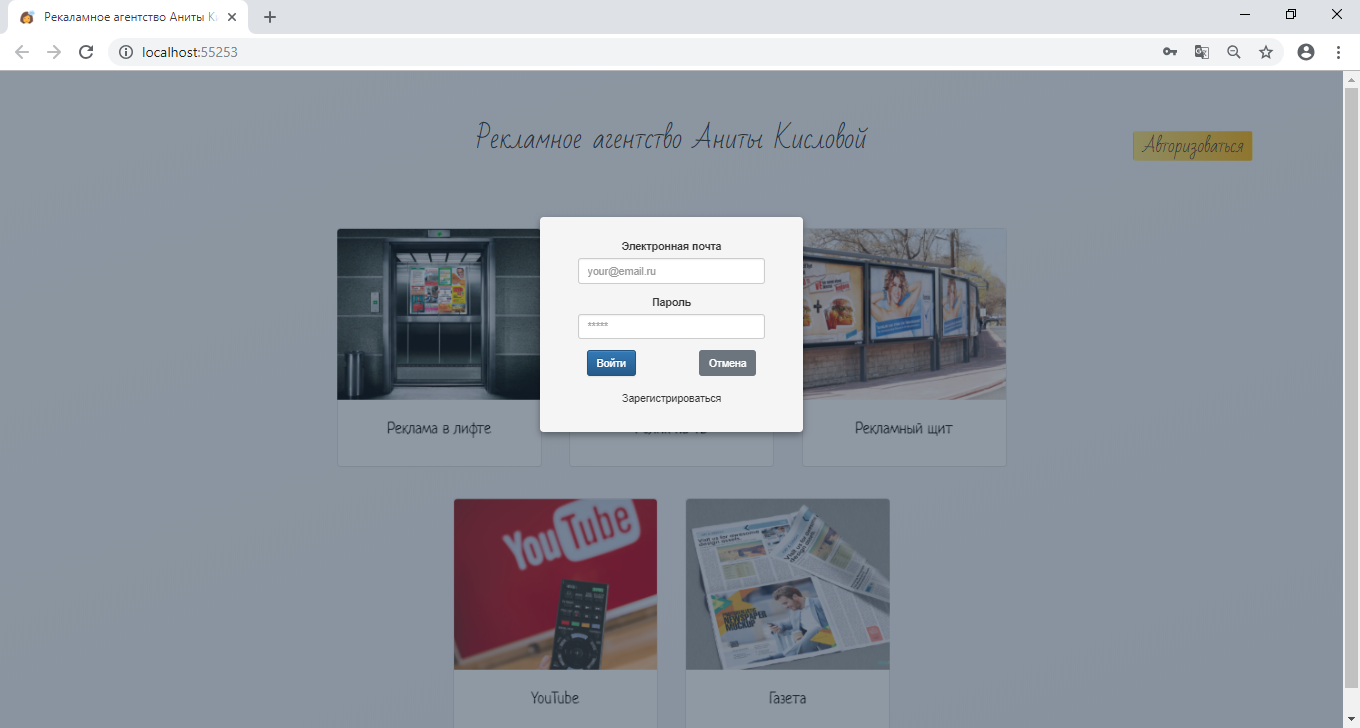


Рисунок 8. Окно авторизации

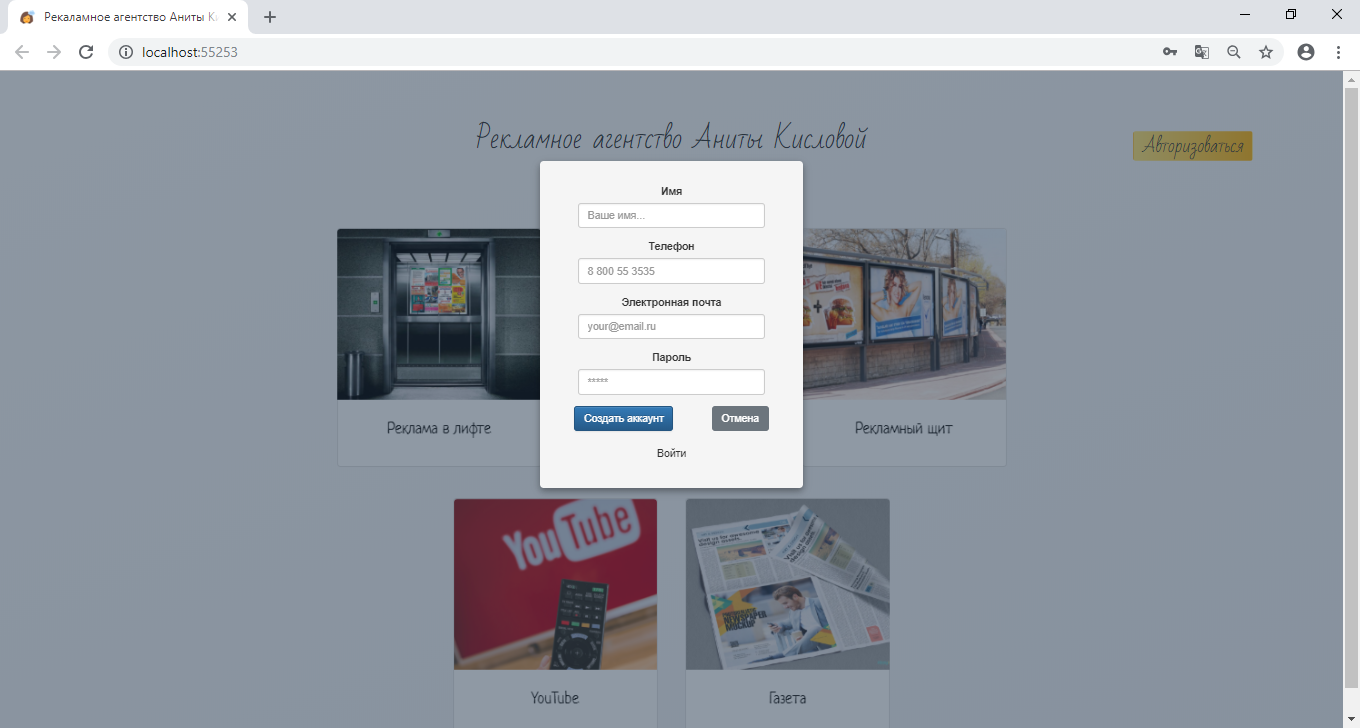


Рисунок 9. Окно регистрации.

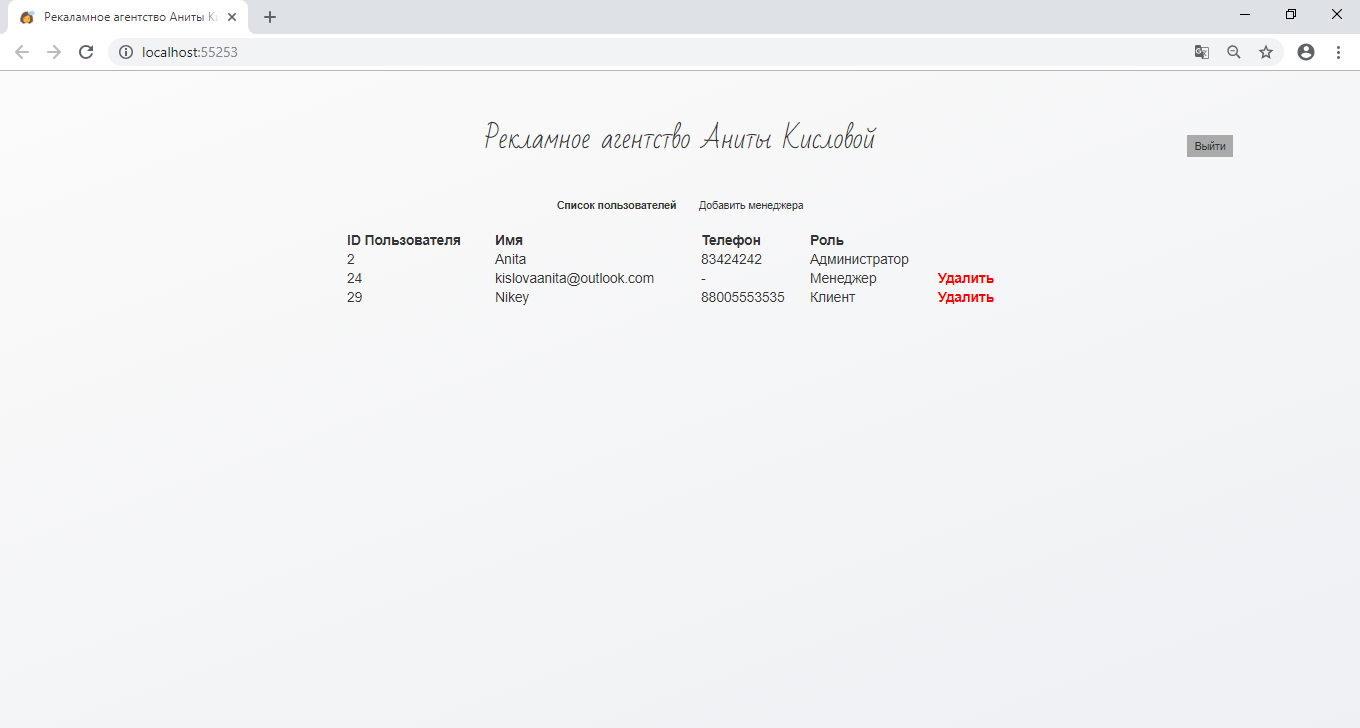


Рисунок 10. Главная страница администратора.

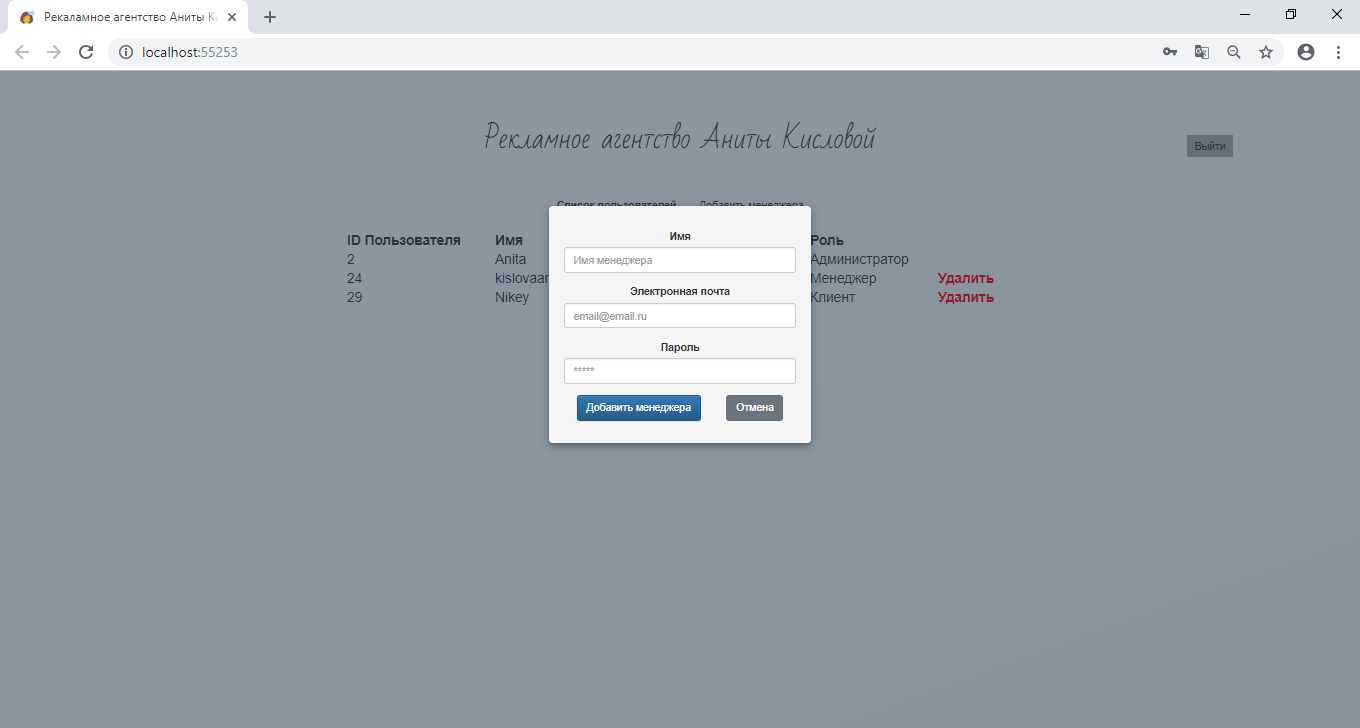


Рисунок 11. Добавление нового менеджера.

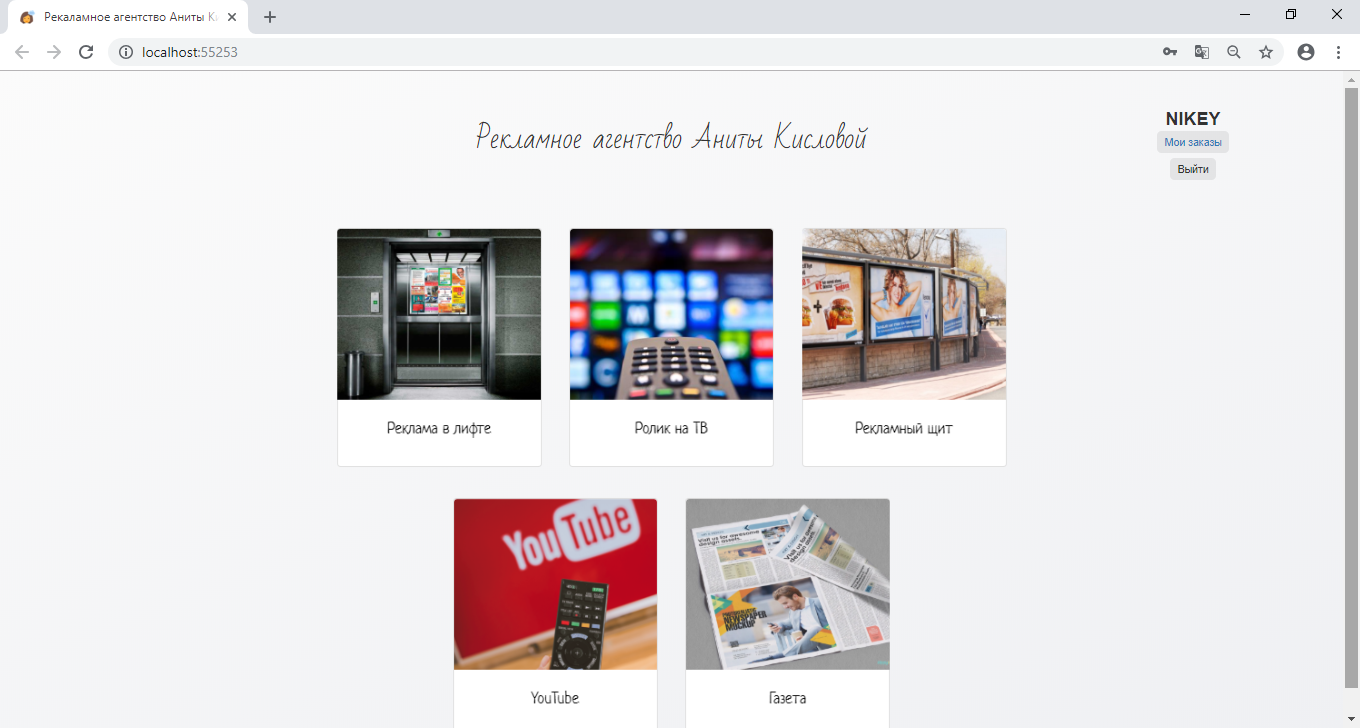


Рисунок 12. Главная страница заказчика.



Рисунок 13. Страница со списком заказов у заказчика.

# 6 ЗАКЛЮЧЕНИЕ

В ходе работы была разработана регистрационная система по информационной области «Рекламное агентство». При разработке приложения была использована система контроля версий.

Был реализован функционал для работы с системой разных типов пользователей. Приложение поддерживает ведение лога для отслеживания корректности работы системы.

# 7 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Буч Г. Объективно-ориентированное проектирование / Пер. с анг. Канкорд, 1996.
2. Шмуллер Дж. Освой самостоятельно UML. 2-е изд. М.: изд. дом «Вильямс», 2002
3. Фримен А., Сандерсон С. ASP.NET MVC 4 Framework с примерами на C# для профессионалов. 4-е издание. Вильямс, 2014
4. Гольцман В. MySQL 5.0 Библиотека программиста. СПБ.:Питер, 2010
5. Фримен А., Джозеф C. Раттц-мл - LINQ. Язык интегрированных запросов в C# 2010 для профессионалов Вильямс, 2011

# ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММНОЙ СИСТЕМЫ

Исходные коды программы:

Класс AuthUserViewModel

namespace AdvAgency.Models

{

public class AuthUserViewModel

{

public string Email { get; set; }

public string Password { get; set; }

}

}

Класс Order

namespace AdvAgency.Models

{

public class Order

{

public int Id { get; set; }

public int ServiceId { get; set; }

public Service Service { get; set; }

public int UserId { get; set; }

public User User { get; set; }

public string Comment { get; set; }

public string Duration { get; set; }

public OrderState Status { get; set; }

public DateTime Created { get; set; }

}

}

Класс OrderState

namespace AdvAgency.Models

{

public enum OrderState

{

Reicived,//принят

Confirmed,//подтвержден

Proceed,//выполняется

Canceled,//отменен

Done

}

}

Класс OrderViewModel

namespace AdvAgency.Models

{

public class OrderViewModel

{

public int UserId { get; set; }

public int ServiceId { get; set; }

public string Duration { get; set; }

public string Comment { get; set; }

public static explicit operator Order(OrderViewModel m)

{

return new Order()

{

UserId = m.UserId,

ServiceId = m.ServiceId,

Duration = m.Duration,

Comment = m.Comment

};

}

}

}

Класс RegisterUserViewModel

namespace AdvAgency.Models

{

public class RegisterUserViewModel

{

public string Name { get; set; }

public string Password { get; set; }

public string Phone { get; set; }

public string Email { get; set; }

public int RoleId { get; set; }

public static explicit operator User(RegisterUserViewModel m)

{

return new User()

{

FullName = m.Name,

Phone = m.Phone,

Email = m.Email,

RoleId = m.RoleId

};

}

}

}

Класс Role

namespace AdvAgency.Models

{

public class Role

{

public int Id { get; set; }

public String Name { get; set; }

}

}

Класс Service

namespace AdvAgency.Models

{

public class Service

{

public Service() { }

public int Id { get; set; }

public string Title { get; set; }

public string ImageUrl { get; set; }

}

}

Класс User

namespace AdvAgency.Models

{

public class User

{

public User() { }

public int Id { get; set; }

public String FullName { get; set; }

public String Email { get; set; }

public String Phone { get; set; }

public int RoleId { get; set; }

public Role Role { get; set; }

public byte[] PasswordHash { get; set; }

public byte[] PasswordSalt { get; set; }

}

}

Контроллеры

Класс AuthController

namespace AdvAgency.Controllers

{

[Authorize]

[Route("api/[controller]")]

public class AuthController : Controller

{

const string Admin = "1";

const string Manager = "2";

const string Client = "3";

private readonly IUserService \_userService;

private readonly IConfiguration \_config;

public AuthController(IUserService userService, IConfiguration config)

{

this.\_userService = userService;

this.\_config = config;

}

[AllowAnonymous]

[HttpPost("register")]

public IActionResult Create([FromBody] RegisterUserViewModel model)

{

model.RoleId = 3;

try

{

this.\_userService.Create((User)model, model.Password);

return Ok();

}

catch (AppException e)

{

return BadRequest(new

{

message = e.Message

});

}

catch

{

return BadRequest(new

{

message = "Произошла ошибка, попробуйте позже"

});

}

}

[AllowAnonymous]

[HttpPost("auth")]

public IActionResult Authenticate([FromBody] AuthUserViewModel model) {

if(model == null || model.Email == null || model.Password == null)

return BadRequest(new { message = "Электронная почта или пароль не валидны" });

var user = \_userService.Authenticate(model.Email,model.Password);

if (user == null)

return BadRequest(new { message = "Электронная почта или пароль не валидны" });

var tokenHandler = new JwtSecurityTokenHandler();

var key = Encoding.ASCII.GetBytes(\_config.GetSection("Config:SecretKey").Value);

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(new Claim[]

{

new Claim(ClaimTypes.Name, user.Id.ToString()),

new Claim(ClaimTypes.Role, user.RoleId.ToString()),

}),

Expires = DateTime.UtcNow.AddDays(7),

SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)

};

var token = tokenHandler.CreateToken(tokenDescriptor);

var tokenString = tokenHandler.WriteToken(token);

return Ok(new {

token = tokenString,

email = user.Email,

fullName = user.FullName,

phone = user.Phone

});

}

}

}

Класс ManagementController

namespace AdvAgency.Controllers

{

[Route("api/[controller]")]

public class ManagementController : Controller

{

const string Admin = "1";

const string Manager = "2";

const string Client = "3";

private readonly IUserService \_userService;

private readonly IUnitOfWork \_uof;

public ManagementController(IUserService userService, IUnitOfWork unitOfWork)

{

this.\_userService = userService;

this.\_uof = unitOfWork;

}

[Authorize(Roles = Admin)]

[HttpPost("createmanager")]

public IActionResult NewManager([FromBody] RegisterUserViewModel model)

{

try

{

model.RoleId = 2;

\_userService.Create((User)model, model.Password);

return Ok();

} catch (Exception ex)

{

return BadRequest();

}

}

[Authorize(Roles = Admin)]

[HttpGet("getusers")]

public IActionResult GetUsers()

{

try

{

var users = \_userService.Fetch(u => true);

users = users.Select(u =>

{

u.PasswordHash = null;

u.PasswordSalt = null;

return u;

});

return Ok( new

{

users

});

}

catch (Exception ex)

{

return BadRequest();

}

}

[Authorize(Roles = Admin)]

[HttpDelete("removeuser/{id}")]

public IActionResult RemoveUser([FromRoute]int id)

{

try

{

if (\_userService.Get(id).RoleId == 1) throw new Exception("Can't remove administrator");

\_userService.Delete(id);

\_uof.Complete();

return Ok();

}

catch (Exception ex)

{

return BadRequest( new {

message = ex.Message

});

}

}

}

}

Класс OrderController

namespace AdvAgency.Controllers

{

[Route("api/[controller]")]

public class OrderController : Controller

{

const string Admin = "1";

const string Manager = "2";

const string Client = "3";

private readonly IRepository<Order> \_orderRepo;

private readonly IUnitOfWork \_uof;

public OrderController(IRepository<Order> orderRepo, IUnitOfWork uof)

{

\_orderRepo = orderRepo;

\_uof = uof;

}

[Authorize(Roles = Client)]

[HttpPost("makeorder")]

public IActionResult MakeOrder([FromBody] OrderViewModel model)

{

try

{

var order = (Order)model;

order.UserId = int.Parse(User.Identity.Name);

order.Created = DateTime.Now;

\_orderRepo.Create(order);

\_uof.Complete();

return Ok();

}

catch (Exception e)

{

return BadRequest(new

{

message = "На сервере произошла ошибка, попробуйте позже"

});

}

}

[Authorize(Roles = (Manager + "," + Client))]

[HttpGet("getorders")]

public IActionResult GetOrders()

{

try

{

IEnumerable<Order> orders;

if (User.GetRoleId() == Client)

{

var userId = int.Parse(User.Identity.Name);

orders = \_orderRepo.Fetch(order => order.UserId == userId);

} else

{

orders = \_orderRepo.Fetch(o => true);

}

return Ok(new

{

orders

});

}

catch (Exception e)

{

return BadRequest(new

{

message = "На сервере произошла ошибка, попробуйте позже"

});

}

}

[Authorize(Roles = Manager)]

[HttpPut("setstatus/{id}")]

public IActionResult SetStatus([FromRoute] int id, [FromQuery] int status )

{

try

{

var order = \_orderRepo.Get(id);

if (order != null)

{

order.Status = (OrderState) status;

\_uof.Complete();

return Ok();

} else

{

return BadRequest("Заказ не найден");

}

} catch (Exception)

{

return BadRequest();

}

}

}

}

Класс ServiceController

namespace AdvAgency.Controllers

{

[Route("api/[controller]")]

public class ServiceController : Controller

{

private readonly IRepository<Service> \_servicesRepo;

public ServiceController(IRepository<Service> servRep)

{

\_servicesRepo = servRep;

}

[HttpGet("[action]")]

public IActionResult GetServices()

{

try

{

return Ok(new

{

services = \_servicesRepo.Fetch(s => true)

});

}

catch

{

return BadRequest(new

{

message = "Can't fetch data"

});

}

}

}

}

Класс Startup

namespace AdvAgency

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContextPool<AppContext>(

options => options.UseMySql(Configuration.GetSection("Config:ConnectionString").Value,

mySqlOptions =>

{

mySqlOptions.ServerVersion(new System.Version(8, 0, 13), ServerType.MySql);

}

));

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

services.AddSpaStaticFiles(configuration =>

{

configuration.RootPath = "ClientApp/build";

});

//конфигурирование JWT-аутендификации

var key = Encoding.ASCII.GetBytes(Configuration.GetSection("Config:SecretKey").Value);

services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(x =>

{

x.Events = new JwtBearerEvents

{

OnTokenValidated = context =>

{

var userService = context.HttpContext.RequestServices.GetRequiredService<IUserService>();

var userId = int.Parse(context.Principal.Identity.Name);

var user = userService.Get(userId);

if (user == null)

{

context.Fail("Unauthorized");

}

return Task.CompletedTask;

}

};

x.RequireHttpsMetadata = false;

x.SaveToken = true;

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(key),

ValidateIssuer = false,

ValidateAudience = false

};

});

//добавление зависимостей

services.AddScoped<IRepository<Service>, ServicesRepository>();

services.AddScoped<IRepository<User>, UserRepository>();

services.AddScoped<IRepository<Order>, OrderRepository>();

services.AddScoped<IUnitOfWork, UnitOfWork>();

services.AddScoped<IUserService, UserService>();

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseSpaStaticFiles();

app.UseAuthentication();

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

template: "{controller}/{action=Index}/{id?}");

});

app.UseSpa(spa =>

{

spa.Options.SourcePath = "ClientApp";

if (env.IsDevelopment())

{

spa.UseReactDevelopmentServer(npmScript: "start");

}

});

}

}

}

Класс IUserService

namespace AdvAgency.Services

{

public interface IUserService

{

User Authenticate(string email, string password);

IEnumerable<User> Fetch(Expression<Func<User,bool>> predicate);

User Get(int id);

User Get(Func<User,bool> predicate);

User Create(User user, string password);

void Update(User user, string password = null);

void Delete(int id);

}

}

Класс UserService

namespace AdvAgency.Services

{

public class UserService : IUserService

{

private readonly IRepository<User> \_userRepo;

private readonly IUnitOfWork \_unitOfWork;

public UserService(IRepository<User> userRep, IUnitOfWork uof){

this.\_userRepo = userRep;

this.\_unitOfWork = uof;

}

public User Authenticate(string email, string password)

{

if (string.IsNullOrEmpty(email) || string.IsNullOrEmpty(password))

return null;

var user = \_userRepo.Get(u => u.Email == email);

// check if username exists and password is correct

if (user == null || !VerifyPasswordHash(password, user.PasswordHash, user.PasswordSalt))

return null;

// authentication successful

return user;

}

public User Create(User user, string password)

{

if (string.IsNullOrWhiteSpace(password))

throw new AppException("Необходим пароль");

if (\_userRepo.Get(u => u.Email == user.Email) != null)

throw new AppException($"Пользователь с электронной почтой {user.Email} уже существует.");

byte[] passwordHash, passwordSalt;

CreatePasswordHash(password, out passwordHash, out passwordSalt);

user.PasswordHash = passwordHash;

user.PasswordSalt = passwordSalt;

\_userRepo.Create(user);

\_unitOfWork.Complete();

return user;

}

public void Delete(int id)

{

\_userRepo.Delete(new User() { Id = id });

}

public IEnumerable<User> Fetch(Expression<Func<User, bool>> predicate)

{

return \_userRepo.Fetch(predicate);

}

public User Get(int id)

{

return \_userRepo.Get(id);

}

public User Get(Func<User, bool> predicate)

{

throw new NotImplementedException();

}

public void Update(User user, string password = null)

{

throw new NotImplementedException();

}

private static void CreatePasswordHash(string password, out byte[] passwordHash, out byte[] passwordSalt){

if (password == null) throw new ArgumentNullException("password");

if (string.IsNullOrWhiteSpace(password)) throw new ArgumentException("Value cannot be empty or whitespace only string.", "password");

using (var hmac = new System.Security.Cryptography.HMACSHA512())

{

passwordSalt = hmac.Key;

passwordHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));

}

}

private static bool VerifyPasswordHash(string password, byte[] storedHash, byte[] storedSalt)

{

if (password == null) throw new ArgumentNullException("password");

if (string.IsNullOrWhiteSpace(password)) throw new ArgumentException("Value cannot be empty or whitespace only string.", "password");

if (storedHash.Length != 64) throw new ArgumentException("Invalid length of password hash (64 bytes expected).", "passwordHash");

if (storedSalt.Length != 128) throw new ArgumentException("Invalid length of password salt (128 bytes expected).", "passwordHash");

using (var hmac = new System.Security.Cryptography.HMACSHA512(storedSalt))

{

var computedHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));

for (int i = 0; i < computedHash.Length; i++)

{

if (computedHash[i] != storedHash[i]) return false;

}

}

return true;

}

}

}

# ПРИЛОЖЕНИЕ Б СТРУКТУРА БАЗЫ ДАННЫХ

Класс 20181125164828\_initial.cs

namespace AdvAgency.Migrations

{

public partial class initial : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Services",

columns: table => new

{

Id = table.Column<int>(nullable: false)

.Annotation("MySQL:AutoIncrement", true),

Title = table.Column<string>(nullable: true),

ImageUrl = table.Column<string>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Services", x => x.Id);

});

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Services");

}

}

}

Класс 20181130083213\_addUsers.cs

namespace AdvAgency.Migrations

{

public partial class addUsers : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Role",

columns: table => new

{

Id = table.Column<int>(nullable: false)

.Annotation("MySql:ValueGenerationStrategy", MySqlValueGenerationStrategy.IdentityColumn),

Name = table.Column<string>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Role", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Users",

columns: table => new

{

Id = table.Column<int>(nullable: false)

.Annotation("MySql:ValueGenerationStrategy", MySqlValueGenerationStrategy.IdentityColumn),

FullName = table.Column<string>(nullable: true),

Phone = table.Column<string>(nullable: true),

RoleId = table.Column<int>(nullable: false),

PasswordHash = table.Column<byte[]>(nullable: true),

PasswordSalt = table.Column<byte[]>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Users", x => x.Id);

table.ForeignKey(

name: "FK\_Users\_Role\_RoleId",

column: x => x.RoleId,

principalTable: "Role",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_Users\_RoleId",

table: "Users",

column: "RoleId");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Users");

migrationBuilder.DropTable(

name: "Role");

}

}

}

Класс 20181130225328\_addRoles.cs

namespace AdvAgency.Migrations

{

public partial class addRoles : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_Users\_Role\_RoleId",

table: "Users");

migrationBuilder.DropPrimaryKey(

name: "PK\_Role",

table: "Role");

migrationBuilder.RenameTable(

name: "Role",

newName: "Roles");

migrationBuilder.AddPrimaryKey(

name: "PK\_Roles",

table: "Roles",

column: "Id");

migrationBuilder.AddForeignKey(

name: "FK\_Users\_Roles\_RoleId",

table: "Users",

column: "RoleId",

principalTable: "Roles",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_Users\_Roles\_RoleId",

table: "Users");

migrationBuilder.DropPrimaryKey(

name: "PK\_Roles",

table: "Roles");

migrationBuilder.RenameTable(

name: "Roles",

newName: "Role");

migrationBuilder.AddPrimaryKey(

name: "PK\_Role",

table: "Role",

column: "Id");

migrationBuilder.AddForeignKey(

name: "FK\_Users\_Role\_RoleId",

table: "Users",

column: "RoleId",

principalTable: "Role",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

}

}

}

Класс 20181201093609\_userAddEmail.cs

namespace AdvAgency.Migrations

{

public partial class userAddEmail : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.AddColumn<string>(

name: "Email",

table: "Users",

nullable: true);

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropColumn(

name: "Email",

table: "Users");

}

}

}

Класс 20181217220204\_AddOrdersTable.cs

namespace AdvAgency.Migrations

{

public partial class AddOrdersTable : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Orders",

columns: table => new

{

Id = table.Column<int>(nullable: false)

.Annotation("MySql:ValueGenerationStrategy", MySqlValueGenerationStrategy.IdentityColumn),

ServiceId = table.Column<int>(nullable: false),

UserId = table.Column<int>(nullable: false),

Comment = table.Column<string>(nullable: true),

Duration = table.Column<string>(nullable: true),

Status = table.Column<int>(nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Orders", x => x.Id);

table.ForeignKey(

name: "FK\_Orders\_Services\_ServiceId",

column: x => x.ServiceId,

principalTable: "Services",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_Orders\_Users\_UserId",

column: x => x.UserId,

principalTable: "Users",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_Orders\_ServiceId",

table: "Orders",

column: "ServiceId");

migrationBuilder.CreateIndex(

name: "IX\_Orders\_UserId",

table: "Orders",

column: "UserId");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Orders");

}

}

}

Класс 20181218195419\_AddCreatedPropToOrder.cs

namespace AdvAgency.Migrations

{

public partial class AddCreatedPropToOrder : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.AddColumn<DateTime>(

name: "Created",

table: "Orders",

nullable: false,

defaultValue: new DateTime(1, 1, 1, 0, 0, 0, 0, DateTimeKind.Unspecified));

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropColumn(

name: "Created",

table: "Orders");

}

}

}