Курсов проект

по

Проектиране и интегриране

на софтуерни системи

Хотел Борика

Уеб сайт и онлайн система за резервации

Изготвили:
Виктория Куцарова, 61457
Елена Орешарова, 61470
Валентин Змийчаров, 61481
Емил Гоцев, 61455

4-ти курс Софтуерно инженерство
зимен семестър 2014/2015

Фаза 2: Софтуерна имплементация на системата

1. Основна част на системата - ASP.NET MVC

Основният модул се състои от публична част и административен панел - Системата следва MVC (Model-View-Controller) архитектурата.

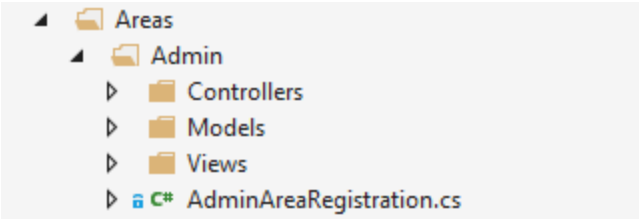
1.1 Публична част

Публичната част на системата съдържа 1 контролер (Home). За всеки от екраните (Стаи, Галерия, Контакти, Традиции, Празници) са дефинирани отделни Action методи с View-та към тях.

```
public partial class HomeController : BaseController
{
    public virtual ActionResult Index()
    {
        return View();
    }

    public virtual ActionResult Rooms()
    {
        var rooms = unitOfWork.RoomRepository.Get().ToList();

        for (int i = 0; i < rooms.Count; i++)
        {
            rooms[i].Pictures = Directory.GetFiles(Server.MapPath(Constants.RoomsImagesDir + rooms[i].RoomId +
        }

        return View(rooms);
    }

    public virtual ActionResult Contact()
    {
        return View();
    }

    public virtual ActionResult Gallery()
    {
        var albums = unitOfWork.AlbumRepository.Get().ToList();

        for (int i = 0; i < albums.Count; i++)
        {
```

1.2 Административен панел

Административната част на системата е отделена в “Area”:



За всеки един от компонентите (Галерия, стаи, резервация, търсене) са създадени отделни контролери с нужните Action методи. Те всички наследяват AdminBaseController, което предпазва от неоторизиран достъп чрез атрибута [Authorize].

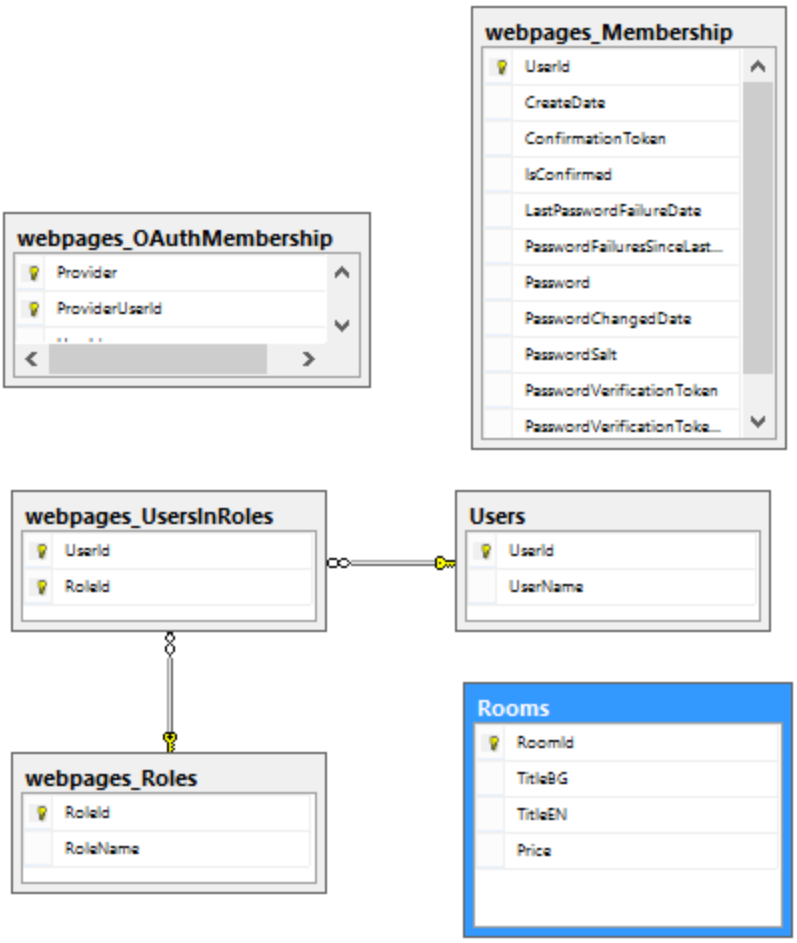
```
[Authorize]
public partial class AdminBaseController : Controller
{
    protected UnitOfWork unitOfWork;

    public AdminBaseController()
    {
        unitOfWork = new UnitOfWork();
    }

}
```

Дефиниран е и Account контролер, който се занимава с управлението на профилите: Вход/изход от системата, смяна на парола.

```
[InitializeSimpleMembership]
public partial class AccountController : Controller
{
    public virtual ActionResult Login()
    {
        if (Request.IsAuthenticated)
        {
            return RedirectToAction(MVC.Admin.Room.ActionNames.List, MVC.Admin.Room.Name, new { area
        }
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public virtual ActionResult Login(LoginVM model)
    {
        if (ModelState.IsValid && WebSecurity.Login(model.UserName, model.Password, persistCookie: fa
        {
            return RedirectToAction(MVC.Admin.Room.ActionNames.List, MVC.Admin.Room.Name, new { area
        }

        // If we got this far, something failed, redisplay form
        ModelState.AddModelError("", "Невалидно потребителско име или парола");
        return View(model);
    }
```

1.3 База данни

Системата използва база данни SQL Server 2012. Дефинирани са отделни таблици за стаи и за всички данни за потребителите. Картините към стаите и галерията не се държат в базата, а във файловото съдържание на системата. Комуникацията от сървъра към базата се осъществява чрез ADO.NET ORM - той създава edmx файл и отделни класове за всички таблици от базата.



1.4 Локализация

Публичната част на системата поддържа превод на английски. Home контролера наследява Base контролер, който при заявка към сървъра прочита подадените данни за език в url адреса и на база на това решава каква култура да зададе:

```
protected void SetLocalizationCulture()
{
    var language = LocalizationLanguage.bg.ToString();

    if (RouteData.Values["lang"] != null && !string.IsNullOrWhiteSpace(RouteData.Values["lang"].ToString()))
    {
        language = RouteData.Values["lang"].ToString();
    }

    var ci = new CultureInfo(language);
    Thread.CurrentThread.CurrentUICulture = ci;
    Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture(ci.Name);
}
```

Надписите се четат от ресурси Global.resx и Global.en.resx. Те представляват Dictionaries с еднакви ключове и различни стойности (съответно на ангийски и на български). В View-то се описва като: Global.Rooms и това се възприема като “Ако

текущата култура е BG, вземи стойнотта от Global.resx; Ако текущата култура е EN, вземи стойността от Global.en.resx”.
Global.resx:

ReservationsShort	Възползвайте се от нашите предложения и направете
Rooms	Стаи
RoomsShort	Предлагаме няколко вида стаи, сред които Двойна, Мансардна и
StElenaBody	на едноименния връх е изграден параклис, в който на празника
StElenaHeader	„Св. Елена и Константин"
StIlijaBody	отпразнува се на 20 юли или съботата преди него със събор
StIlijaHeader	Гайдарско надсвирване "Илинден - Равногор"
StPeterBody	се провежда всяка година в последната събота на месец юни.

Global.en.resx:

ReservationsShort	Take advantage of our offers and make your reservation now!
Rooms	Rooms
RoomsShort	We offer several types of rooms, including Double, Loft and
StElenaBody	at the peak of the same name is set up a chapel where a boiled
StElenaHeader	"St. Constantine and Elena"
StIlijaBody	people celebrate it on the 20th of July or the Saturday before it. The
StIlijaHeader	Bagpipe Contest "Ilinden - Ravnogor"
StPeterBody	takes place every year on the last Friday of June. The celebration

Употреба:

```
@{
    ViewBag.Title = Global.Rooms;
```

1.5 Комуникация

- Google Analytics: В началото на html съдържанието на всяка една от страниците се добавя кратък скрипт:

```
<script>
    (function (i, s, o, g, r, a, m) {
        i['GoogleAnalyticsObject'] = r; i[r] = i[r] || function () {
            (i[r].q = i[r].q || []).push(arguments)
        }, i[r].l = 1 * new Date(); a = s.createElement(o),
        m = s.getElementsByTagName(o)[0]; a.async = 1; a.src = g; m.parentNode.insertBefore(a, m)
    })(window, document, 'script', '//www.google-analytics.com/analytics.js', 'ga');

    ga('create', 'UA-44720939-1', 'auto');
    ga('send', 'pageview');
</script>
```

  След регистриране на сайта в Google Webmaster Tools, може да се гледа статистика за посещенията на сайта.
- Google Maps: Вграждат се директно чрез iframe. Може да се променят ширината и височината на картата.

- Модул за резервация: Клиентската и административната част се вграждат директно чрез iframe. След обновяване/добавяне/изтриване на стая се изпраща заявка към модула да обнови информацията:

```
string url = string.Format(
        @"http://127.0.0.1:3000/rooms/update
        ?external_id={1}
        &name_bg={2}
        &name_en={3}
        &price={3}",
        "", room.RoomId, room.TitleBG, room.TitleEN, room.Price)
        .Replace(Environment.NewLine, "").Replace(" ", "");
MakeRequest(url);

private void MakeRequest(string url)
{
    HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
    request.Method = "GET";
    var response = request.GetResponse();
}
```

- Модул за търсене: Изпраща параметри в query string-а на заявката. Тя от своя страна връща договорен JSON обект:

```
private JObject MakeRequest(string url)
{
    HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
    request.Method = "GET";
    request.AllowAutoRedirect = false;
    request.Accept = "application/json, text/plain, */*";
    request.ContentType = "application/application/json; charset=UTF-8";

    JObject jO;

    using (var response = request.GetResponse())
    using (var responseStream = response.GetResponseStream())
    using (var reader = new StreamReader(responseStream))
    {
        var responseFromServer = reader.ReadToEnd();
        jO = JsonConvert.DeserializeObject<JObject>(responseFromServer);
    }

    return jO;
}

public class SearchResult
{
    public string name { get; set; }
    public string location { get; set; }
    public string stars { get; set; }
    public string price { get; set; }
}
```

2. Услуга за справки - Java REST (Jersey)

2.1 Преглед на REST архитектурата

REST(Representational State Transfer) е архитектурен стил, който задава ограничения, като например единен интерфейс(използване на фиксиран брой операции за създаване, четене, промяна и триене на данни - `PUT`, `GET`, `POST` и `DELETE`), който предпоставя RESTful услугите да са подходящи за уеб услуги. Данните и функционалността при REST архитектурата се считат за ресурси и са достъпни чрез използването на(URI адреси), обикновено адреси в мрежата. Ресурсите са достъпни и могат да се обработват чрез няколко прости, добре дефинирани операции. REST стилът се използва заедно с клиент/сървър архитектура и най-често използва HTTP като протокол за комуникация. В REST архитектурния стил, клиентът и сървърът си обменят ресурси чрез използване на стандартизиран интерфейс и протокол.

2.2 Интерфейс на услугата за справки

Услугата за справки се достъпва чрез предварително дефиниран адрес и може да приема 3 query параметъра - брой звезди, местоположение и подредба. Всеки от параметрите е незадължителен и може да бъде изпуснат. Услугата връща списък с хотели в JSON формат, като всеки хотел има следния вид: {name: “hotel_name”, location: “hotel_location”, start: “number_of_stars”, price: “price_for_room”}.

2.3 Реализация на услугата

Услугата е реализирана с помощта на Java 7 и имплементацията на REST архитектурата за Java - Jersey. С помощта на maven се свалят всички нужни зависимости в проекта. След компилиране, кодът се пакетира до war файл, който може да бъде прочетен от web application server-и като tomcat, glassfish, jboss и др. За съхранение на данни се използва вградена Java SQL база - H2.

В web.xml се дефинира общият адрес на услугата.

```
<servlet-mapping>
    <servlet-name>hotel-search</servlet-name>
    <url-pattern>/api/*</url-pattern>
</servlet-mapping>
```

Чрез анотацията @Path(“hotels”) може да се достъпват методите в HotelController. Чрез анотацията @Path(“/search”) и анотацията @GET над метода findHotels, може да се търсят хотели по три параметър - брой звезди, местоположение и подредба.

```
@Path("hotels")
public class HotelController {

    private HotelService hotelService = new HotelServiceImpl();

    @GET
    @Path("/search")
    @Produces(MediaType.APPLICATION_JSON)
    public Response findHotels(@QueryParam("stars") Integer stars,
            @QueryParam("location") Integer location, @QueryParam("order") Integer order) {

        return BaseController.buildResponse(hotelService.getHotels(stars, location, order));
    }
}
```

Адресът, на който може да се достъпи метода findHotels, е: server_addres:server_port/hotel-search/api/hotels/search?star=num_of_stars&location=location_id&order=order_id. Трите параметъра са незадължителни. Създадени са номенклатури за градове и подредба, така че всеки град съответства на число, както и всяка подредба(по име, звезди или цена) също съответства на число.

Orders:	Location:
{Value = "1", Text = "Име"}, {Value = "2", Text = "Цена"}, {Value = "3", Text = "Звезди"	{Value = "1", Text = "Банско"}, {Value = "2", Text = "Белмекен"}, {Value = "3", Text = "Боровец"}, ...

HotelService представлява интерфейс, в който е дефиниран един метод - getHotels. Имплементацията на интерфейса се намира в класа HotelServiceImpl. В нея се създава sql заявка, която се изпраща до h2 базата и връща намерените резултати.

```
String query = "select h.name, l.name, stars, price "
        + "from hotels as h, locations as l "
        + " where h.location = l.id " + createWhereClause(stars, location)
        + (returnedOrder != null
        ? " order by " + returnedOrder + " asc"
        : "");
```

2.4 База данни

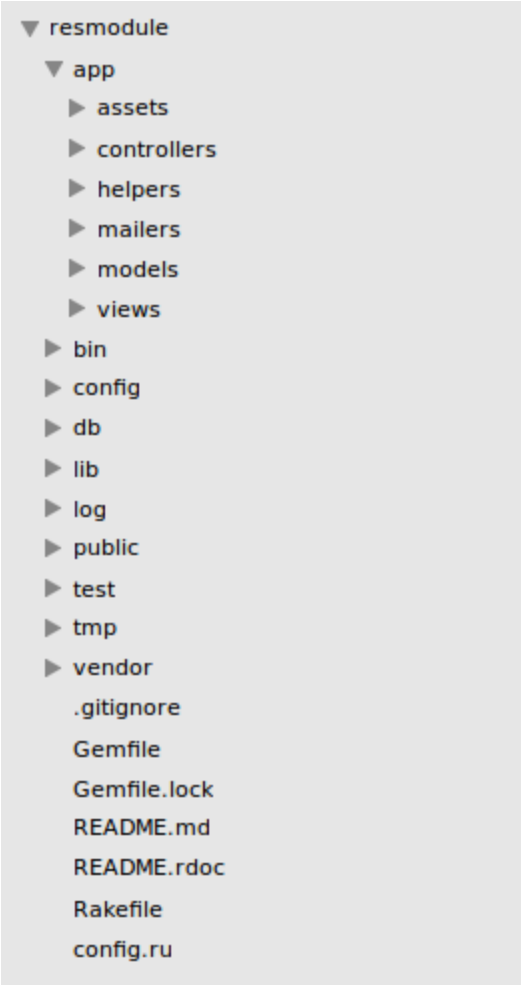
За съхранение на номенклатурите за градове, хотели и подредба е използвана вградената база h2(http://www.h2database.com/html/main.html). Тя е бърза и лека, поддържа JDBC API.

- Връзката към база се осъществява чрез класа DatabaseConnector.
- Създаването на схемата, както и добавянето на данни в таблиците се осъществява програмно - в класа DBUtil. Тя се създава еднократно при стартиране на сървъра, върху който услугата върви.

3. Модул за резервации - Ruby on Rails

3.1 Преглед на архитектурата

Rails като „framework“ налага използването на MVC като архитектурен стил на приложението. Друга характерна черта са множеството конфигурации по подразбиране, с които бързо може да се стартира даден проект (такива настройки има за базите данни, за изгледите и контролерите, за локализацията и др.). В модула за резервации има 2 модела, съответно за тях 2 контролера и няколко изгледа.
Директорийна структура на проекта:



3.2 Изгледи

Rails предоставя стандартни начини за представяне на изгледите, като двата по подразбиране са чрез ERB (генерира HTML) и JSON - те са и използваните в приложението.
В административната част изгледите са: списък с нови резервации, списък с обработени резервации, редактиране и преглеждане на детайлите на резервация.

В клиентската част има само изглед за създаване на нова резервация.

3.3 Синхронизация на наличните стаи с основния модул

Синхронизацията на наличните стаи се осъществява чрез URL, който се предоставя от модула за резервации и който се извиква от основния модул. Като параметри на URL-а се подават идентификационен номер на типа стая, имена на български и английски и цена. Например:
http://127.0.0.1:3000/rooms/update?external_id=29&name_bg=двойна&name_en=double&price=100
Записването или съответно обновяването на информацията за стаите се извършва от контролера RoomsController.

3.4 Бази данни и Active Record

За записване на информацията в база от данни се използва ActiveRecord - това е ORM „framework“, който Rails предоставя по подразбиране. ActiveRecord предоставя имплементации за всички популярни SQL бази данни (както комерсиални, така и такива с отворен код), но също така и за някои NoSQL бази данни.
За да се използва ActiveRecord е единствено нужно класовете на моделите да наследяват ActiveRecord::Base:

```
class Reservation < ActiveRecord::Base
  enum payment_method: [:cash, :bank]
  belongs_to :room
end
```

За модула е използвана SQLite, която се използва по подразбиране по време на разработка за Rails приложения.
При промени в схемата на базата от данни в Rails се използват т. нар. миграции. Те представляват Ruby код, който след това се превежда от ActiveRecord до SQL. Миграциите се намират обичайно в папката db и следващото е пример за такава:

```
class AddRoomIdToReservations < ActiveRecord::Migration
  def change
    add_column :reservations, :room_id, :integer
    add_index :reservations, :room_id
  end
end
```

3.5 Локализация

Езикът по подаразбиране е български, също както в основния модул. Локализацията става чрез различни .yml файлове в папката config/locales. В момента се поддържа освен български и английски. Следващото е част от файла bg.yml:

```yaml
bg:
  hello: "Здравей, свят!"
  helpers:
    label:
      reservation:
        create_date: "Дата на създаване"
        checkin: "Настаняване"
        checkout: "Освобождаване"
        client_names: "Имена на клиент"
        client_email: "Имейл на клиент"
        client_phone: "Телефон на клиент"
        details: "Допълнителни детайли"
        clients_count: "Брой клиенти"
        breakfast: "Закуска"
        payment_method: "Метод на плащане"
        status: "Статус"
        room_type: "Тип стая"
    submit:
      reservation:
        create: "Резервирай"
        update: "Запази"
  reservation:
    payment_method:
      bank: "Банков превод"
      cash: "В брой"
```