Система за управление на резервации в хотели

Софтуерни системи, базирани на услуги 2016

Изготвили:

Валентин Змийчаров Ф.Н. 24952

Владислав Беличев Ф.Н. 24976

Съдържание

Система за управление на резервации в хотели 1
1. Цел на разработваното приложение 3
2. Анализ на изискванията към приложението 3
   2.1 Функционални изисквания 3
   2.2 Нефункционални изисквания 4
3. Обосновка на използваните програмни технологии 4
   3.1 .NET Framework 4
   3.2 Java 4
4. Проектиране на услугите и приложението, което ги използва 5
5. Описание на програмната реализация 5
   5.1 Уеб приложение 5
   5.2 Java SOAP услуги 7
   5.3 Комуникация с външна система - Yandex Translate API 10
   5.4 .NET REST услуги 10

1. Цел на разработваното приложение

Системата е уеб приложение, което предоставя възможност за търсене на хотел в България по град и име на английски език. След избор тя предоставя подробна информация за хотела. Информацията може да бъде преведена на български и руски. За всеки хотел се предоставя възможност за резервация (Брой възрастни / деца + датите на престой). Всеки потребител може да наблюдава извършените през системата резервации.

2. Анализ на изискванията към приложението

2.1 Функционални изисквания

- **Търсене на хотели** - избор измежду наличните в системата хотели с търсене по име на град и име на хотел
- **Преглед на детайли за хотел** - системата предоставя подробна информация за избрания хотел:
  - Наименование
  - Град
  - Адрес
  - Номер за връзка
  - Описание
  - Брой стаи
  - Цена за единична стая
  - Цена за двойна стая
  - Цена за апартамент
  - Брой звезди
  - Удобства
- **Превод на информацията за хотел (Yandex translate API)** - информацията за всеки хотел може да бъде преведена на български и руски чрез предоставеното от Yandex API за машинен превод
- **Извършване на резервация** - след избор на хотел, потребителят може да извърши резервация за него като избере дати на престой и брой възрастни / деца
- **Преглед на извършени резервации** - всеки потребител може да преглежда извършените резервации през системата

2.2 Нефункционални изисквания

- Уеб-базиран интерфейс, който поддържа минимум последните версии на браузърите Internet Explorer, Mozilla Firefox, Safari и Chrome.
- Защитена връзка към външните източници.
- Системата връща отговор на заявка в рамките на 2 секунди + времето за отговор на външните системи.
- Системата е налична 24/7 освен при нужда от обновяване.

3. Обосновка на използваните програмни технологии

3.1 .NET Framework

Уеб приложението, което обединява имплементираните услуги е реализирано чрез ASP.NET5 MVC6. Това е най-новата технология на Microsoft за разработка на уеб приложения и дори е в **RC (Release candidate)** версия. За целта се използва Visual Studio 2015 под Windows 10.

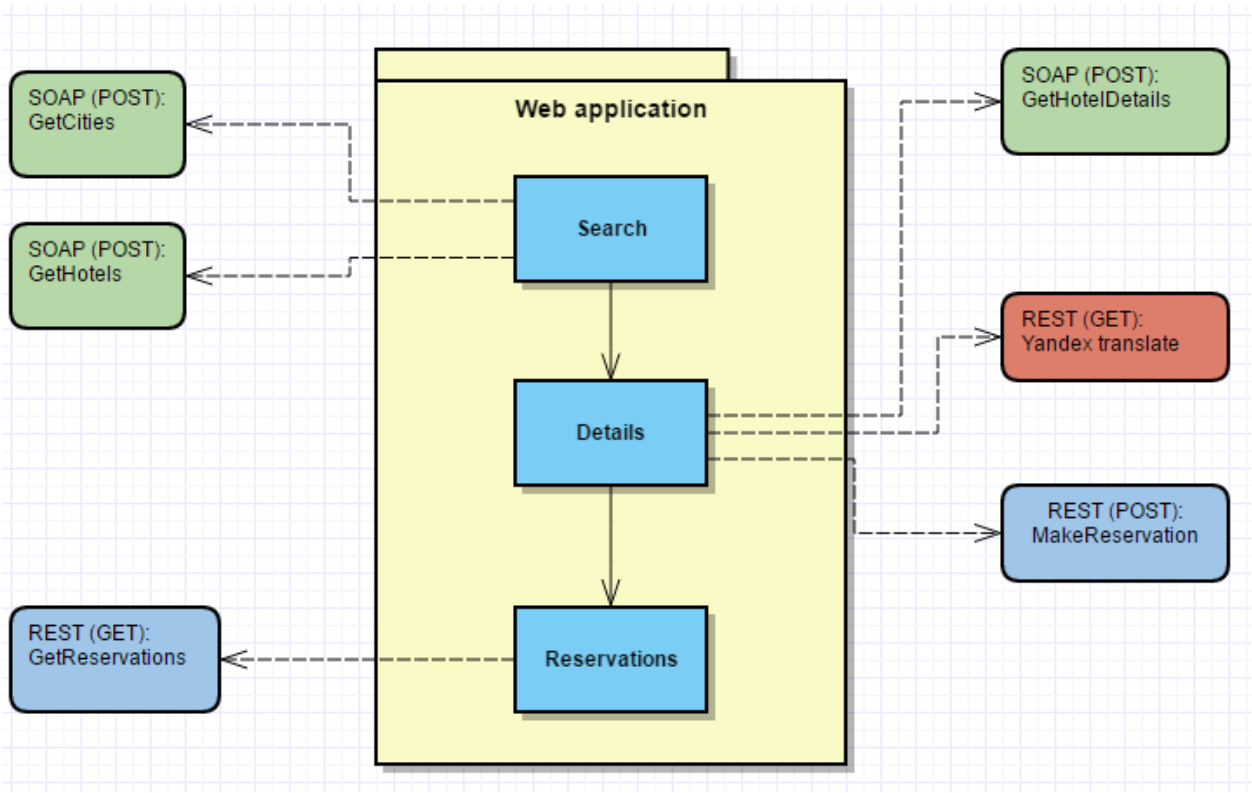
Услугите свързани с резервацията също са имплементирани на ASP.NET5 и използват **REST** протокол. Реализирани са чрез новия за .NET вид контролер, при който след добавяне на съответните атрибути се пояснява, видът на методите му:

```
[Produces("application/json")]
[Route("api/HotelReservations")]
public class HotelReservationsController : Controller
```

3.2 Java

SOAP-базираната уеб услуга, предоставяща информация за градове и хотели е реализирана на **Java 8** с помощта на фреймуърка **Spring MVC**. Това е актуална технология с много възможности. Средата за разработка е **Spring Tools Suite** (customization of Eclipse). Освен това е използвана и **build системата Gradle** за успешното конструиране на Java проекта - компилация, набавяне на необходимите библиотеки, генериране на source файлове (Java класове) на база .xsd дефиниции на обектите (хотели и градове) и операциите върху тях, генериране на изпълним .war файл. Благодарение на тези технологии и билиотеки програмиста се грижи само за формата на данните и тяхното въвеждане, конфигурирането на портовете за уеб услугата и прихващането на заявките към сървъра. Маршализацията и демаршализацията на съобщенията се извършва автоматично с помощта на библиотеката **JAXB**.

4. Проектиране на услугите и приложението, което ги използва



Диаграмата показва нагледно връзката между приложението и имплементираните услуги. Екранът Search достъпва SOAP услугите за градове и хотели. След избор на хотел, потребителят е пренасочен към екрана Details. Той извиква SOAP услуга за извличане на детайна информация за избрания хотел. Външната услуга за превод на текст от Yandex се извиква, за да преведе данните за хотела на искания от потребителя език. Ако потребителят желае може да направи резервация за хотела, при което системата се обръща към REST API-то с POST заявка: MakeReservation и потребителят е пренасочен към списъка с резервации. Там системата извиква REST API GET метода GetReservations, за да визуализира списъка с направени резервации.

5. Описание на програмната реализация

5.1 Уеб приложение

Уеб приложението е реализирано чрез .NET и използва MVC (Model-View-Controller) архитектура. Има един основен (Home) контролер, който чрез различни Action method-и извиква различни страници от системата.

- **Index action** - Търсене на хотел по град и име. Обръща се към Java SOAP услугите за взимане на списък с градове по зададен термин за търсене, хотели по даден град и термин за търсене. След избор на хотел се извършва POST заявка към /Home/Search. След проверка, че входът е валиден, потребителят се пренасочва към страница с детайли за избрания хотел
- **Details action** - По зададено id на хотел и език (незадължително) визуализира подробна информация за хотела. Този метод извиква Java SOAP услугата за детайли за хотел и визуализира получената информация. Ако параметърът за език е подаден, извиква Yandex Translate API, за да преведе информацията за хотела. В този екран е предоставена форма за резервация със следните полета за попълване: Дата от/до; Брой възрастни; Брой деца. При коректно въведени данни се извиква .NET POST API-то за извършване на резервация за конкретен хотел. При успешна резервация, потребителят е пренасочен към страница със списък с извършените през системата резервации.
- **HotelReservations action** - Списък с извършените през системата резервации. Представени са под формата на таблица и включват името на избрания хотел, дата на извършване на резервацията, дата от/до, брой деца и брой възрастни.

5.2 Java SOAP услуги

SOAP услугите за градове и хотели са дефинирани в .xsd файлове по следния начин:

```
<xs:element name="getCitiesRequest">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="term" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="getCitiesResponse">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="cities" type="tns:cities"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:complexType name="cities">
    <xs:sequence>
        <xs:element name="city" type="tns:city" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="city">
    <xs:sequence>
        <xs:element name="name" type="xs:string"/>
    </xs:sequence>

    <xs:attribute name="id" type="xs:unsignedInt" use="required"/>
</xs:complexType>

</xs:schema>
```

Фиг. 1. XSD дефиниция на уеб услуга за градове

Услугата за градове предоставя следните възможности:

- Търсене по id на град
- Търсене на градове по зададена част от тяхното име

Услугата за хотели предоставя следните възможности:

- Търсене по id на хотел
- Търсене на хотели по зададена част от тяхното име и/или id на града, в който се намират
- Извличане на детайлна информация за хотел по негово id

На база .xsd дефинициите на обектите в SOAP услугите и операциите върху тях чрез Gradle build script и библиотеката JAXB се генерират .java файлове, репрезентиращи тези обекти и операции. След това Gradle build script компилира приложението, извличайки необходимите библиотеки, от които то зависи и го пакетира като Spring boot application.

От създадените JAVA класове се създават инстанции, представящи обекти в SOAP услугите (градове и хотели) и по този начин се създава статична база данни за хотели и градове. JAVA класовете, представящи операциите върху обектите се използват за автоматична маршализация и демаршализация на клиентските заявки и отговори, прихващани от сървърното приложение.

Автоматичо се създава и WSDL дефиниция за всяка една услуга на база .xsd представянето ѝ (достъпна на http://localhost:8080/ws/hotels.wsdl). Това е необходимо за клиентското приложение, което консумира услугата:

```
@Bean(name = "hotels")
public DefaultWsdl11Definition hotelsWsdl1Definition(XsdSchema hotelsSchema) {
    DefaultWsdl11Definition wsdl11Definition = new DefaultWsdl11Definition();
    wsdl11Definition.setPortTypeName("HotelsPort");
    wsdl11Definition.setLocationUri("/ws/hotels");
    wsdl11Definition.setTargetNamespace("http://hotels.sources.com");
    wsdl11Definition.setSchema(hotelsSchema);
    return wsdl11Definition;
}
```

Фиг. 2. Автоматично генериране на WSDL за хотели

На последно място се създават методите за прихващане на клиентските заявки. Те изглеждат по следния начин:

```
@PayloadRoot(namespace = NAMESPACE_URI, localPart = "getCitiesRequest")
@ResponsePayload
public GetCitiesResponse getCities(@RequestPayload GetCitiesRequest request) {
    GetCitiesResponse response = new GetCitiesResponse();
    response.setCities(cityRepository.findCities(request.getTerm()));

    return response;
}
```

Фиг. 3. Прихващане на клиентска заявка за градове

Примерна клиентска заявка за градове е следната (извършва се чрез метод POST на XML документ на адреса, на който е налична услугата за градове - http://localhost:8080/ws/cities):

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:gs="http://cities.sources.com">
   <soapenv:Header/>
   <soapenv:Body>
      <gs:getCitiesRequest>
         <gs:term>v</gs:term>
      </gs:getCitiesRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Фиг. 4. Примерна заявка за търсене на градове по част от тяхното име

Отговорът на тази заявка изглежда по следния начин:

```
vbelichev@ubuntuvm:~/Projects/services-su-hotel/soa-java-soap/src/main/resources/samples$
curl --header "content-type: text/xml" -d @cities-request.xml http://localhost:8080/ws/cit
ies -s | xmllint --format -
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:getCitiesResponse xmlns:ns2="http://cities.sources.com">
      <ns2:cities>
        <ns2:city id="1">
          <ns2:name>Varna</ns2:name>
        </ns2:city>
        <ns2:city id="3">
          <ns2:name>Plovdiv</ns2:name>
        </ns2:city>
        <ns2:city id="5">
          <ns2:name>Veliko Tarnovo</ns2:name>
        </ns2:city>
      </ns2:cities>
    </ns2:getCitiesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Фиг. 5. Примерен отговор на клиентска заявка за градове

5.3 Комуникация с външна система - Yandex Translate API

Yandex предоставя безплатна услуга за превод на текст. Уеб приложението извиква услугата, когато потребителят иска да преведе информацията за даден хотел на друг език. Всеки един от атрибутите на хотела се подава по отделно на услугата и върнатият резултат се визуализира. На услугата се подават секретен ключ (получава се след регистрация в Yandex), текстът, който трябва да се преведе и двата езика (от език1 към език2). Извикването на услугата е реализирано в TranslateRepository:

```
public class YandexTranslateRepository : ITranslateRepository
{
    public string Translate(string input, string toLang, string fromLang = "en")
    {
        YandexTranslateResult yandexResult = null;

        using (var client = new HttpClient())
        {
            client.BaseAddress = new Uri("https://translate.yandex.net/");
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

            var response = client.GetAsync(string.Format("api/v1.5/tr.json/translate?key={0}&text={1}&lang={2}-{3}"
                , "trnsl.1.1.20160329T125742Z.7cf7029a7299121f.e2035c38942a41a74d94459866c16182668762f8"
                , input
                , fromLang
                , toLang))
            .GetAwaiter().GetResult();
            if (response.IsSuccessStatusCode)
            {
                var jsonContent = response.Content.ReadAsStringAsync().Result;
                yandexResult = JsonConvert.DeserializeObject<YandexTranslateResult>(jsonContent);
            }
        }

        if (yandexResult != null && yandexResult.Text != null && yandexResult.Text.Count > 0)
            return yandexResult.Text[0];
        else
            return input;
    }

}
```

Фиг. 6. Комуникация с Yandex

5.4 .NET REST услуги

.NET услугите се извикват чрез REST протокол. Те пишат и извличат информация от SQL Server база от данни, съдържаща резервациите, направени през системата. Поддържат се 2 услуги:

- GET (“api/HotelReservations”) - Връща списък с направените резервации, подредени по дата на извършване на резервацията.
- POST (“api/HotelReservations”: hotelReservation entity) - Ако входът е валиден, записва нова резервация с подадените параметри. Ако не, връща грешка.