Хотелски резервации
XML проект - **Документация**

Дата: 21 December 2024

Автор: Петя Личева

Факултетен номер: 3MI0700022

Резюме

Задачата, която ще изпълнява моята XML система е резервация на хотелски стаи и разплащането при резервация. Моята система предлага достъп до различни хотели, които от своя страна имат различни видове стаи. Стаите поддържат различни удобства, максимално допустим брой гости, вид на стаята и други. Тези стаи могат да се наемата за определени периоди от време (, които се определят от началната дата на престой до крайната дата на престой) . Разплащането се осъществява по различни методи - в кеш или с кредитна / дебитна карта като се указва и личните данни на госта, който ще извършва разплащането - имена, имейл, телефонен номер и прочие.

Структура на проекта

- css - папка със стиловете на проекта;
- js - папка с javascript файловете, които генерират случайни данни, екземплярите с въпросните случайни данни и прилага различните трансформации;
- resources - папка с index.html (началната страница на проекта) и необходимите XSLT трансформации;
- Hotel_Reservations_XML_Project_Documentation.pdf - документацията на проекта в pdf формат;
- Hotel_Reservations_XML_Project_Presentation.pptx - презентацията на проекта;
- reservations.xsd - xsd файла, който задава формата на XML схемата спрямо изискванията за проекта;
- test.xml - тестово съдържание, което показва как работи схемата;

Версия “XML схема”

Описание на предметната област и на заданието

Предметната област на реализация на проекта е системата по резервации на хотелски стаи в различни хотели. Приложението е разработено на база проучвания на подобни приложения, например - https://www.booking.com и https://www.airbnb.com . Приложението таргетира фирми и услуги в туристическия сектор от нивото на споменатите вече сайтове като цели предоставяне на нов формат, в който да се представят резервациите на хотелски стаи, стаи или цели апартманети от къщи за гости, като се стреми да покрие и двата сектора - хотелиерство и малки, частни, семейни бизнеси, които се занимават с подобна дейност.

Дефиниране на схемата

Кодът на с хемата

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="transactions">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="transaction">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="reservation">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="room" type="Room"/>
                                        <xsd:element name="guests">
                                            <xsd:complexType>
                                                <xsd:sequence>
                                                    <xsd:element name="guest"
type="Guest"
                                                                 maxOccurs="unbounded"/>
                                                </xsd:sequence>
                                            </xsd:complexType>
                                        </xsd:element>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                            <xsd:element name="card" type="DebitCard"/>
                        </xsd:sequence>
                        <xsd:attribute name="id" type="xsd:integer" use="required"/>
                        <xsd:attribute name="payingTool" use="required"
type="PayingTool"/>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <!-- Helper complex types -->
    <xsd:complexType name="Hotel">
        <xsd:sequence>
            <xsd:element name="name" type="Name"/>
            <xsd:element name="address" type="Address"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Price">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="value" type="xsd:float"/>
```

```
                <xsd:attribute name="currency" type="CurrencyName"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:simpleType name="RoomType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="apartment"/>
            <xsd:enumeration value="single"/>
            <xsd:enumeration value="double"/>
            <xsd:enumeration value="deluxe"/>
            <xsd:enumeration value="president"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="DebitCard">
        <xsd:sequence>
            <xsd:element name="iban" type="IBAN"/>
            <xsd:element name="balance" type="xsd:float"/>
        </xsd:sequence>
        <xsd:attribute name="currency" type="xsd:string"/>
    </xsd:complexType>
    <xsd:simpleType name="PayingTool">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="by_card"/>
            <xsd:enumeration value="in_cash"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="Room">
        <xsd:sequence>
            <xsd:element name="roomNo" type="xsd:integer"/>
            <xsd:element name="floorNo" type="xsd:integer"/>
            <xsd:element name="maxCapacity" type="xsd:string"/>
            <xsd:element name="hasPersonalBath" type="xsd:boolean"/>
            <xsd:element name="isAvailable" type="xsd:boolean"/>
            <xsd:element name="hotel" type="Hotel"/>
            <xsd:element name="amenities">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="amenity" type="Amenity"
maxOccurs="unbounded"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="price" type="Price"/>
        </xsd:sequence>
        <xsd:attribute name="type" type="RoomType"/>
    </xsd:complexType>

    <!-- Helper complex types with validations -->
    <xsd:simpleType name="Name">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="([A-Z][a-z]+)|([А-Я][а-я]+)"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="Email">
        <xsd:restriction base="xsd:string">
```

```
            <xsd:pattern value="[A-Za-z0-9\.\-
_+%]+@((gmail\.com)|(yahoo\.com)|(hotmail\.com)|(aol\.com)|(outlook\.com)|(comcast\.net)|
(icloud\.com)|(msn\.com)|(hotmail\.co\.uk)|(sbcglobal\.net)|(live\.com)|(yahoo\.co\.in)|(
me\.com)|(att\.net)|(mail\.ru)|(bellsouth\.net)|(rediffmail\.com)|(cox\.net)|(yahoo\.co\.
uk)|(verizon\.net)|(ymail\.com)|(hotmail\.it)|(kw\.com)|(yahoo\.com\.tw)|(mac\.com)|(live
\.se)|(live\.nl)|(yahoo\.com\.br)|(googlemail\.com)|(libero\.it)|(web\.de)|(allstate\.com
)|(btinternet\.com)|(online\.no)|(yahoo\.com\.au)|(live\.dk)|(earthlink\.net)|(yahoo\.fr)
|(yahoo\.it)|(gmx\.de)|(hotmail\.fr)|(shawinc\.com)|(yahoo\.de)|(moe\.edu\.sg)|(163\.com)
|(naver\.com)|(bigpond\.com)|(statefarm\.com)|(remax\.net)|(rocketmail\.com)|(live\.no)|(
yahoo\.ca)|(bigpond\.net\.au)|(hotmail\.se)|(gmx\.at)|(live\.co\.uk)|(mail\.com)|(yahoo\.
in)|(yandex\.ru)|(qq\.com)|(charter\.net)|(indeedemail\.com)|(alice\.it)|(hotmail\.de)|(b
luewin\.ch)|(optonline\.net)|(wp\.pl)|(yahoo\.es)|(hotmail\.no)|(pindotmedia\.com)|(orang
e\.fr)|(live\.it)|(yahoo\.co\.id)|(yahoo\.no)|(hotmail\.es)|(morganstanley\.com)|(wellsfa
rgo\.com)|(juno\.com)|(wanadoo\.fr)|(facebook\.com)|(edwardjones\.com)|(yahoo\.se)|(fema\
.dhs\.gov)|(rogers\.com)|(yahoo\.com\.hk)|(live\.com\.au)|(nic\.in)|(nab\.com\.au)|(ubs\.
com)|(uol\.com\.br)|(shaw\.ca)|(t-
online\.de)|(umich\.edu)|(westpac\.com\.au)|(yahoo\.com\.mx)|(yahoo\.com\.sg)|(farmersage
nt\.com)|(anz\.com)|(yahoo\.dk)|(dhs\.gov))"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="Phone">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="(\+(359)|0)[0-9]{9}"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="Guest">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="name" type="Name" use="required"/>
                <xsd:attribute name="family" type="Name" use="required"/>
                <xsd:attribute name="email" type="Email"/>
                <xsd:attribute name="phone" type="Phone"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:simpleType name="CurrencyName">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\w{3,5}"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="Currency">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="name" type="CurrencyName"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:simpleType name="IBAN">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Z]{2}\d{2}[A-Z0-9]{1,30}"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="Amenity">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
```

```
                <xsd:attribute name="name" type="Name"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="Address">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="country" type="Name" use="required"/>
                <xsd:attribute name="town" type="Name" use="required"/>
                <xsd:attribute name="location" type="xsd:string" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:schema>
```

Елементи

- Transactions - описва множество от поне 1 транзакция; при точно 1 транзакция може да се изпусне този елемент с цел опростяване на записа;
- Transaction - описва конкретна транзакция, която в себе си съдържа id на транзакцията, начин на плащане, който може да е (“с карта” или “в кеш”), резервацията, за която е била отворена транзакцията и картата, с която е извършено плащането (в случай, че не плащаме в кеш, а с карта);
- Reservation - описва конкретна резервация, която в себе си съдържа стаята, която е била наета и списък на гостите на въпросната стая;
- Room - описва конкретна стая, която включва номер на стаята, ноер на етажа, максимален капацитет на стаята, дали има лична баня, дали стаята е налична при резервиране и след това, хотела, в който се намира стаята, списък с удобствата на стаята, цена и тип стая.
- Guests - описва множество от поне 1 гост на дадена стая; при точно 1 гост може да се изпусне този елемент с цел опростяване на записа;
- Guest - описва конкретен гост на дадена стая, който включва име, фамилия, имейл и телефон като името и фамилията са задължителни компоненти за разлика от имейла и телефона;
- Card - описва конкретна дебитна карта, която включва валута на картата, айбан и баланс;
- IBAN - описва айбана на дадена дебитна карта и го валидира с регулярен израз;
- Balance - описва баланса на дебитна карта като число с плаваща запетая и го валидира;
- RoomNo - описва номера на стая в хотел като цяло число и го валидира;
- FloorNo - описва номера на етажа на дадена хотелска стая като цяло число и го валидира;
- MaxCapacity - описва максималния капацитет на стаята като цяло число и го валидира;
- HasPersonalBath - описва наличието / отсъствието на лична баня в дадена хотелска стая като булева стойност и го валидира;
- IsAvailable - описва наличността на дадена стая (нейния статус) като булева стойност true при свободна стая и false - при заета стая и го валидира;
- Hotel - описва конкретен хотел чрез името и адреса му;
- Name - описва полетата име на клиент, фамилия на клиент, име на хотел, име на удобство в стая, име на държава, име на град и т.н. като низ от символи, започващ с

главна буква (латинска или българска) и продължаващ с малки букви (латински или български);
- Address - описва конкретен адрес на конкретен хотел, който има държава, град и локация;
- Amenities - описва множество от поне 1 удобство в конкретна хотелска стая; при точно 1 удобство може да се изпусне този елемент с цел опростяване на записа;
- Amenity - описва конкретно удобство в дадена хотелска стая, което има име на удобството;
- Price - описва конкретна цена на дадена резервация, която има стойност и валута;

Атрибути

- Id - пореден номер на дадена трансакция (уникално цяло число);
- PayingTool - вид на плащане на дадена трансакция - в кеш или с дебитна карта;
- Value (price) - стойност на цената на дадена резервация;
- Currency (price) - валута на цената на дадена резервация;
- Currency (debit card) - валута на дебитната карта на даден клиент;
- Type (room) - тип стая - апартаманет, единична, двойна, делукс или президентски апартманет;
- Name (client) - име на клиент (започва с главна латинска или българска буква и продължава с малки букви - латински или български);
- Family - фамилия на клиент (започва с главна латинска или българска буква и продължава с малки букви - латински или български);
- Email - имейл на даден клиент (<латински букви>.<латински букви>@<валиден домейн>);
- Phone - телефонен номер на даден клиент (валидни български номера, започващи с +359 или 0 и продължаващи с още 9 цифри);
- Name (currency) - име на дадена валута (от 3 до 5 главни латнски букви, използвани за съкращение името на дадена валута, например BGN, USD и т.н.);
- Name (amenity) - име на удобство в дадена хотелска стая (започва с главна латинска или българска буква и продължава с малки букви - латински или български);
- Country - име на дадена държава (започва с главна латинска или българска буква и продължава с малки букви - латински или български);
- Town - име на даден град (започва с главна латинска или българска буква и продължава с малки букви - латински или български);
- Location - локация (низ от символи - без ограничение на символите, които се използват);

Добавяне на примерно съдържание, за доказване чрез примери за работоспособността на схемата

Тестово съдържание, което да покаже, че схемата работи:

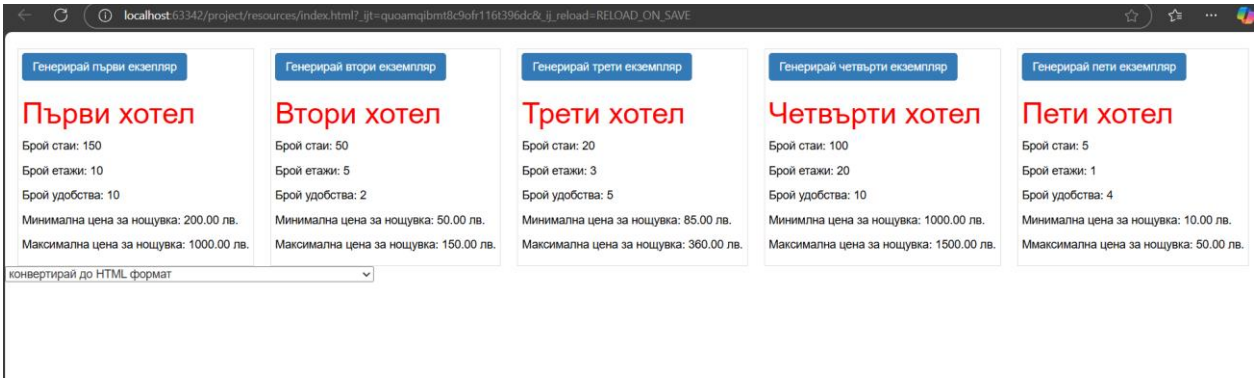
```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="xml-to-plain-text.xsl"?>

<transactions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:noNamespaceSchemaLocation="reservations.xsd">
```

```
    <transaction id="1" payingTool="by_card">
        <reservation>
            <room>
                <roomNo>101</roomNo>
                <floorNo>1</floorNo>
                <maxCapacity>2</maxCapacity>
                <hasPersonalBath>true</hasPersonalBath>
                <isAvailable>false</isAvailable>
                <hotel>
                    <name>Paradise</name>
                    <address country="Bulgaria" town="Plovdiv" location="ul. Malina
123"/>
                </hotel>
                <amenities>
                    <amenity>TV</amenity>
                    <amenity>Air-Conditioner</amenity>
                </amenities>
                <price currency="BGN">400.00</price>
            </room>
            <guests>
                <guest name="Georgi" family="Ivanov" email="givanov@gmail.com"
phone="0843589999"/>
                <guest name="Mihaela1" family="Georgieva"/>
            </guests>
        </reservation>
        <card currency="BGN">
            <iban>BG12FNL89655</iban>
            <balance>1500.00</balance>
        </card>
    </transaction>
</transactions>
```

Горният xml пример не е валиден, тъй като както казах и по-рано имената и фамилиите на клиентите трябва да започват с главна буква и да продължават с малки букви (независимо дали на латиница или на кирилица), а във въпросния пример името на втория гост - Михаела Георгиева, има грешка, тъй като в името освен допустими символи има и цифри (Mihaela1). Това е и нещото, което ни казва грешката при опит за валидация на екземпляра. Ако обаче вместо Mihael1, напишем Mihaela и пуснем наново валидатора, откриваме, че екземпляра става валиден и съответно вече не би следвало да ни се показват такива съобщения като тези по-долу.

Версия “Описание на DOM, съответстващ на схемата”



Идеята на тази форма е, че всеки от хотелите, описани на нея представят различни екземпляри, които имат конкретни характеристики и с всеки от бутоните се генерира (чрез DOM) по случаен начин екзепляр, отговарящ на характеристиките на някой от описаните хотели. Даденият екземпляр има случйно избрани данни от предварително заредени големи масиви със случайни стойности с цел случайно генериране на валидни XML екземпляри спрямо създадената от мен XML схема. Съответно преди да се натисне някой бутон, който да генерира случаен екземпляр от различен тип хотел, можем да изберем в какъв формат бихме желали приожението да ни покаже данните след генерирането, например - HTML, друг XML формат или в чист текст.

Валидни екземпляри на документа

- Първи екземпляр:

```
<transactions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="reservations.xsd">
   <transaction id="14" payingTool="in_cash">
   <reservation>
   <room type="double">
   <roomNo>43</roomNo>
   <floorNo>0</floorNo>
   <maxCapacity>2</maxCapacity>
   <hasPersonalBath>false</hasPersonalBath>
   <isAvailable>true</isAvailable>
   <hotel>
   <name>Рай</name>
   <address country="Италия" town="Рим" location="Улица: Кленова улица, Район: Южна част"/>
   </hotel>
   <amenities>
   <amenity>Ресторант</amenity>
   <amenity>Бар</amenity>
   <amenity>Закуска</amenity>
   <amenity>Фитнес</amenity>
   <amenity>Бар</amenity>
   <amenity>Спа</amenity>
   <amenity>Чадъри</amenity>
```

<amenity>Шезлонги</amenity>
<amenity>Портиер</amenity>
<amenity>Тераса</amenity>
</amenities>
<price currency="SEK">549</price>
</room>
<guests>
<guest name="Лазар" family="Славчев" email="Lazar.Slavchev@rediffmail.com" phone="0887654321"/>
<guest name="Петър" family="Ганев" email="Petar.Ganev@rediffmail.com" phone="+359878987654"/>
</guests>
</reservation>
<card currency="USD">
<iban>GQ7050002001010843110111</iban>
<balance>323.35227634175055</balance>
</card>
</transaction>
</transactions>

- Втори екземпляр:

```
<transactions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="reservations.xsd">
   <transaction id="77" payingTool="in_cash">
   <reservation>
   <room type="double">
   <roomNo>28</roomNo>
   <floorNo>3</floorNo>
   <maxCapacity>2</maxCapacity>
   <hasPersonalBath>true</hasPersonalBath>
   <isAvailable>true</isAvailable>
   <hotel>
   <name>Рай</name>
   <address country="Египет" town="Гиза" location="Улица: Кинг стрийт, Район: Площад"/>
   </hotel>
   <amenities>
   <amenity>Бар</amenity>
   <amenity>Интернет</amenity>
   </amenities>
   <price currency="CNY">75</price>
   </room>
   <guests>
   <guest name="Яна" family="Богданов" email="Yana.Bogdanov@hotmail.com" phone="+359876543210"/>
   <guest name="Людмила" family="Златев" email="Lyudmila.Zlatev@outlook.com" phone="0876123456"/>
   </guests>
   </reservation>
   <card currency="EUR">
   <iban>LV97HABA0012345678910</iban>
   <balance>69.02438219313728</balance>
   </card>
   </transaction>
</transactions>
```

- Трети екземпляр:

<transactions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="reservations.xsd">
<transaction id="635" payingTool="by_card">
<reservation>
<room type="president">
<roomNo>11</roomNo>
<floorNo>2</floorNo>
<maxCapacity>2</maxCapacity>
<hasPersonalBath>true</hasPersonalBath>
<isAvailable>false</isAvailable>
<hotel>
<name>Модерн</name>
<address country="Египет" town="Кайро" location="Улица: Парк авеню, Район: Център"/>
</hotel>
<amenities>
<amenity>Транспорт</amenity>
<amenity>Минибар</amenity>
<amenity>Ресторант</amenity>
<amenity>Фитнес</amenity>
<amenity>Бар</amenity>
</amenities>
<price currency="SEK">164</price>
</room>
<guests>
<guest name="Катя" family="Недев" email="Katya.Nedev@cox.net" phone="0888654321"/>
<guest name="Йоан" family="Русев" email="Yoan.Rusev@me.com" phone="0877654321"/>
</guests>
</reservation>
<card currency="CHF">
<iban>BF10301340200154009450000643</iban>
<balance>227.79389511740607</balance>
</card>
</transaction>
</transactions>

- Четвърти екземпляр:

<transactions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="reservations.xsd">
<transaction id="804" payingTool="in_cash">
<reservation>
<room type="president">
<roomNo>8</roomNo>
<floorNo>4</floorNo>
<maxCapacity>2</maxCapacity>
<hasPersonalBath>true</hasPersonalBath>
<isAvailable>false</isAvailable>
<hotel>
<name>Сияние</name>
<address country="Индия" town="Мумбай" location="Улица: Бейкър стрийт, Район: Западна част"/>
</hotel>
<amenities>
<amenity>Пералня</amenity>
<amenity>Чадъри</amenity>

```
   <amenity>Закуска</amenity>
   <amenity>Климатик</amenity>
   <amenity>Шезлонги</amenity>
   <amenity>Телевизор</amenity>
   <amenity>Транспорт</amenity>
   <amenity>Румсервиз</amenity>
   <amenity>Сейф</amenity>
   <amenity>Фитнес</amenity>
   </amenities>
   <price currency="JPY">1455</price>
   </room>
   <guests>
   <guest name="Тодор" family="Захариев" email="Todor.Zahariev@att.net" phone="+359889876543"/>
   <guest name="Мартин" family="Димитров" email="Martin.Dimitrov@bellsouth.net" phone="0878123456"/>
   </guests>
   </reservation>
   <card currency="SEK">
   <iban>TG53TG0090604310346500400070</iban>
   <balance>1092.1232541751012</balance>
   </card>
   </transaction>
</transactions>
```


- Пети екземпляр:


```
<transactions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="reservations.xsd">
   <transaction id="932" payingTool="by_card">
   <reservation>
   <room type="deluxe">
   <roomNo>1</roomNo>
   <floorNo>0</floorNo>
   <maxCapacity>4</maxCapacity>
   <hasPersonalBath>true</hasPersonalBath>
   <isAvailable>false</isAvailable>
   <hotel>
   <name>Мираж</name>
   <address country="Мексико" town="Монтерей" location="Улица: Бейкър стрийт, Район: Младост"/>
   </hotel>
   <amenities>
   <amenity>Спа</amenity>
   <amenity>Шезлонги</amenity>
   <amenity>Пералня</amenity>
   <amenity>Чадъри</amenity>
   </amenities>
   <price currency="CHF">11</price>
   </room>
   <guests>
   <guest name="Габриела" family="Славчев" email="Gabriela.Slavchev@bellsouth.net"
   phone="+359889876543"/>
   <guest name="Стефания" family="Янакиев" email="Stefaniya.Yanakiev@icloud.com" phone="+359878123456"/>
   <guest name="Кристиян" family="Димитров" email="Kristiyan.Dimitrov@hotmail.co.uk"
   phone="+359878123456"/>
   <guest name="Людмила" family="Русев" email="Lyudmila.Rusev@bellsouth.net" phone="+359877876543"/>
```

```
    </guests>
    </reservation>
    <card currency="GBP">
    <iban>CY17002001280000001200527600</iban>
    <balance>32.51763396062301</balance>
    </card>
    </transaction>
</transactions>
```

Конструиране и валидиране на всеки екземпляр чрез DOM

На следните снимки ще видите javascript кода, който стои зад генерирането на валидни XML екземпляри спрямо XSD схемата на проекта посредством DOM.

```
// Helper functions - generates a small part of the whole xml file
function generateRoomNo(roomsCount) {
    return Math.floor(Math.random() * roomsCount);
}

function generateFloorNo(floorsCount) {
    return Math.floor(Math.random() * floorsCount);
}

function generateRoomMaxCapacity(roomType) {
    switch (roomType) {
        case "apartment":
            return 8;
        case "double":
            return 2;
        case "deluxe":
            return 4;
        case "president":
            return 2;
        case "single":
        default:
            return 1;
    }
}

function generateHotelName(lang) {
    let randomIndex;
    switch (lang) {
        case "BG": {
            randomIndex = Math.floor(Math.random() * hotelNamesBG.length);
            return hotelNamesBG[randomIndex];
        }
        case "EN":
        default: {
            randomIndex = Math.floor(Math.random() * hotelNamesEN.length);
            return hotelNamesEN[randomIndex];
        }
    }
```

```
}

function generateCountry(lang) {
    let randomIndex;
    switch (lang) {
        case "EN": {
            randomIndex = Math.floor(Math.random() * countriesEN.length);
            return countriesEN[randomIndex];
        }
        case "BG":
        default: {
            randomIndex = Math.floor(Math.random() * countriesBG.length);
            return countriesBG[randomIndex];
        }
    }
}

function generateTown(lang, country) {
    switch (lang) {
        case "EN": {
            const countryTowns = townsEN[country];
            return countryTowns[Math.floor(Math.random() * countryTowns.length)];
        }
        case "BG":
        default: {
            const countryTowns = townsBG[country];
            return countryTowns[Math.floor(Math.random() * countryTowns.length)];
        }
    }
}

function getRandomLocationEN() {
    const randomStreet = streetsEN[Math.floor(Math.random() * streetsEN.length)];
    const randomDistrict = districtsEN[Math.floor(Math.random() * districtsEN.length)];

    return `Street: ${randomStreet}, District: ${randomDistrict}`;
}

function getRandomLocationBG() {
    const randomStreet = streetsBG[Math.floor(Math.random() * streetsBG.length)];
    const randomDistrict = districtsBG[Math.floor(Math.random() * districtsBG.length)];

    return `Улица: ${randomStreet}, Район: ${randomDistrict}`;
}

function generateLocation(lang) {
    if (lang === "BG") {
        return getRandomLocationBG();
    } else {
        return getRandomLocationEN();
    }
}
```

```
function generateAddress(htmlDoc, lang) {
    const addressElement = htmlDoc.createElement('address');
    const country = generateCountry(lang);
    addressElement.setAttribute('country', country);
    addressElement.setAttribute('town', generateTown(lang, country));
    addressElement.setAttribute('location', generateLocation(lang));

    return addressElement;
}

function generateHotel(htmlDoc, lang) {
    const hotelElement = htmlDoc.createElement('hotel');
    const hotelNameElement = htmlDoc.createElement('name');
    const hotelNameTextNode = htmlDoc.createTextNode(generateHotelName(lang));
    hotelNameElement.appendChild(hotelNameTextNode);
    hotelElement.appendChild(hotelNameElement);
    const hotelAddressElement = generateAddress(htmlDoc, lang);
    hotelElement.appendChild(hotelAddressElement);

    return hotelElement;
}

function generateAmenity(lang) {
    let randomIndex;
    switch (lang) {
        case "BG": {
            randomIndex = Math.floor(Math.random() * amenitiesInBG.length);
            return amenitiesInBG[randomIndex];
        }
        case "EN":
        default: {
            randomIndex = Math.floor(Math.random() * amenitiesInEN.length);
            return amenitiesInEN[randomIndex];
        }
    }
}

function generateCurrencyName() {
    const randomIndex = Math.floor(Math.random() * currencies.length);

    return currencies[randomIndex];
}

function generatePriceValue(htmlDoc, min, max) {
    return Math.floor(Math.random() * (max - min) + min);
}

function generatePrice(htmlDoc, lang, min, max) {
    const priceElement = htmlDoc.createElement('price');
    priceElement.setAttribute('currency', generateCurrencyName());

    const value = generatePriceValue(htmlDoc, min, max);
    priceElement.appendChild(htmlDoc.createTextNode(value));
```

```
    return priceElement;
}

function generateRoom(htmlDoc, lang, min, max, roomMaxCapacity, type, roomsCount,
floorsCount, amenitiesCount) {
    const roomElement = htmlDoc.createElement('room');

    const roomNoElement = htmlDoc.createElement('roomNo');
    roomNoElement.appendChild(htmlDoc.createTextNode(generateRoomNo(roomsCount)));
    roomElement.appendChild(roomNoElement);

    const floorNoElement = htmlDoc.createElement('floorNo');
    floorNoElement.appendChild(htmlDoc.createTextNode(generateFloorNo(floorsCount)));
    roomElement.appendChild(floorNoElement);

    const maxCapacityElement = htmlDoc.createElement('maxCapacity');
    maxCapacityElement.appendChild(htmlDoc.createTextNode(roomMaxCapacity));
    roomElement.appendChild(maxCapacityElement);

    const hasPersonalBathElement = htmlDoc.createElement('hasPersonalBath');
    hasPersonalBathElement.appendChild(htmlDoc.createTextNode(Math.random() <= 0.5 ?
"false" : "true"));
    roomElement.appendChild(hasPersonalBathElement);

    const isAvailableElement = htmlDoc.createElement('isAvailable');
    isAvailableElement.appendChild(htmlDoc.createTextNode(Math.random() <= 0.5 ? "false"
: "true"));
    roomElement.appendChild(isAvailableElement);

    const hotelElement = generateHotel(htmlDoc, lang);
    roomElement.appendChild(hotelElement);

    const amenitiesElement = htmlDoc.createElement('amenities');
    for (let i = 0; i < amenitiesCount; i++) {
        const amenityElement = htmlDoc.createElement('amenity');
        amenityElement.appendChild(htmlDoc.createTextNode(generateAmenity(lang)));
        amenitiesElement.appendChild(amenityElement);
    }
    roomElement.appendChild(amenitiesElement);

    const priceElement = generatePrice(htmlDoc, lang, min, max);
    roomElement.appendChild(priceElement);

    roomElement.setAttribute('type', type);
    return roomElement;
}

function generateName(lang) {
    let randomIndex;
    if (lang === "BG") {
        randomIndex = Math.floor(Math.random() * bulgarianNames.length);
```

```
        return bulgarianNames[randomIndex];
    } else {
        randomIndex = Math.floor(Math.random() * englishNames.length);
        return englishNames[randomIndex];
    }
}

function generateFamily(lang) {
    let randomIndex;
    if (lang === "BG") {
        randomIndex = Math.floor(Math.random() * bulgarianFamilies.length);
        return bulgarianFamilies[randomIndex];
    } else {
        randomIndex = Math.floor(Math.random() * englishFamilies.length);
        return englishFamilies[randomIndex];
    }
}

function translateBulgarianToEnglish(firstName, secondName) {
    function translateName(name) {
        return name.split('').map(char => bulgarianToEnglishMap[char] || char).join('');
    }

    return {
        translatedFirstName: translateName(firstName),
        translatedSecondName: translateName(secondName)
    };
}

function generateEmail(name, family) {
    const translatedNames = translateBulgarianToEnglish(name, family);

    const randomIndex = Math.floor(Math.random() * emailDomains.length);
    return translatedNames.translatedFirstName + "." +
translatedNames.translatedSecondName + emailDomains[randomIndex];
}

function generatePhone() {
    const randomIndex = Math.floor(Math.random() * phones.length);
    return phones[randomIndex];
}

function generateGuest(htmlDoc, lang) {
    const guestElement = htmlDoc.createElement('guest');

    const name = generateName(lang);
    guestElement.setAttribute('name', name);

    const family = generateFamily(lang);
    guestElement.setAttribute('family', family);

    const email = generateEmail(name, family);
    guestElement.setAttribute('email', email);
```

```
    const phone = generatePhone();
    guestElement.setAttribute('phone', phone);

    return guestElement;
}

function generateReservation(htmlDoc, lang, min, max, type, roomMaxCapacity, roomsCount,
floorsCount, amenitiesCount) {
    const reservationElement = htmlDoc.createElement('reservation');

    const roomElement = generateRoom(htmlDoc, lang, min, max, roomMaxCapacity, type,
roomsCount, floorsCount, amenitiesCount);
    reservationElement.appendChild(roomElement);
    const guestsElement = htmlDoc.createElement('guests');
    reservationElement.appendChild(guestsElement);

    for (let i = 0; i < roomMaxCapacity; i++) {
        const guestElement = generateGuest(htmlDoc, lang);
        guestsElement.appendChild(guestElement);
    }

    return reservationElement;
}

function generatePayingTool() {
    const randomIndex = Math.floor(Math.random() * payingTools.length);
    return payingTools[randomIndex];
}

function generateRoomType() {
    const roomTypes = ["apartment", "single", "double", "deluxe", "president"];
    const randomIndex = Math.floor(Math.random() * roomTypes.length);
    return roomTypes[randomIndex];
}

function generateIBAN(htmlDoc) {
    const IBANElement = htmlDoc.createElement('iban');
    const randomIndex = Math.floor(Math.random() * IBANs.length);
    IBANElement.appendChild(htmlDoc.createTextNode(IBANs[randomIndex]));

    return IBANElement;
}

function generateBalance(htmlDoc, min, max) {
    const balanceElement = htmlDoc.createElement('balance');
    balanceElement.appendChild(htmlDoc.createTextNode(Math.random() * (max - min) +
min));

    return balanceElement;
}
```

```
function generateDebitCard(htmlDoc, min, max) {
    const debitCardElement = htmlDoc.createElement('card');

    const ibanElement = generateIBAN(htmlDoc);
    debitCardElement.appendChild(ibanElement);

    const balanceElement = generateBalance(htmlDoc, min, max);
    debitCardElement.appendChild(balanceElement);

    const currency = generateCurrencyName();
    debitCardElement.setAttribute('currency', currency)

    return debitCardElement;
}

function generateTransaction(htmlDoc, lang, minPricePerNight, maxPricePerNight,
roomsCount, floorsCount,
                             amenitiesCount, minBalance, maxBalance) {
    const transactionIDStep = 1000;
    const roomType = generateRoomType();
    const roomMaxCapacity = generateRoomMaxCapacity(roomType);

    const transactionElement = htmlDoc.createElement('transaction');
    transactionElement.setAttribute('id', Math.floor(Math.random() * transactionIDStep));
    transactionElement.setAttribute('payingTool', generatePayingTool());

    const reservationElement = generateReservation(htmlDoc, lang, minPricePerNight,
maxPricePerNight, roomType,
        roomMaxCapacity, roomsCount, floorsCount, amenitiesCount);
    transactionElement.appendChild(reservationElement);

    const cardElement = generateDebitCard(htmlDoc, minBalance, maxBalance);
    transactionElement.appendChild(cardElement);

    return transactionElement;
}

const lang = "BG";

// formats the XML content
function formatXML(xmlString) {
    const PADDING = "  ";
    let formatted = "";
    const regex = /(>)(<)(\/*)/g;
    xmlString = xmlString.replace(regex, "$1\n$2$3");

    let pad = 0;
    xmlString.split("\n").forEach((line) => {
        if (line.match(/<\/\w/)) {
            pad -= 1;
        }
```

```
        formatted += PADDING.repeat(pad) + line.trim() + "\n";

        if (line.match(/<\w[^>]*[^/]>.*$/)) {
            pad += 1;
        }
    });

    return formatted.trim();
}

// downloads a specific XML file
function downloadXMLFile(xmlContent, filename) {
    const blob = new Blob([xmlContent], {type: 'application/xml'});

    const link = document.createElement('a');
    link.href = URL.createObjectURL(blob);

    link.download = filename;
    document.body.appendChild(link);
    link.click();

    document.body.removeChild(link);
    URL.revokeObjectURL(link.href);
}

function transformXMLString(xmlString, xsl) {
    const parser = new DOMParser();
    const xml = parser.parseFromString(xmlString, "application/xml");

    if (window.ActiveXObject || "ActiveXObject" in window) { // For IE
        const xsltProcessor = new ActiveXObject("Msxml2.XSLTemplate.6.0");
        const xslDoc = new ActiveXObject("Msxml2.FreeThreadedDOMDocument.6.0");
        const xmlDoc = new ActiveXObject("Msxml2.DOMDocument.6.0");

        xslDoc.loadXML(xsl);
        xmlDoc.loadXML(xmlString);

        xsltProcessor.stylesheet = xslDoc;
        const output = xsltProcessor.createProcessor();
        output.input = xmlDoc;
        output.transform();

        document.getElementById("specimenParagraph").textContent = output.output;
    } else if (document.implementation && document.implementation.createDocument) {
        const xsltProcessor = new XSLTProcessor();
        xsltProcessor.importStylesheet(xsl);
        const resultDocument = xsltProcessor.transformToFragment(xml, document);
        document.getElementById("specimenParagraph").innerHTML = "";
        document.getElementById("specimenParagraph").appendChild(resultDocument);
    } else {
        alert("XSLT transformations are not supported in your browser.");
    }
```

```
}

// generates a hotel specimen by definite parameters - rooms count in a hotel, floors
count in a hotel, amenities count
// in a regular hotel room, min price per a night in the hotel, max price per night in
the hotel
function generateSpecimen(roomsCount, floorsCount, amenitiesCount, minPrice, maxPrice,
filename) {
    const documentImplementation = document.implementation;
    const htmlDoc = documentImplementation.createDocument(null, null);

    const PItext = "version=\"1.0\" encoding=\"UTF-8\"";
    const processingInstructions = htmlDoc.createProcessingInstruction('xml', PItext);
    htmlDoc.insertBefore(processingInstructions, htmlDoc.documentElement);

    const rootElement = htmlDoc.createElement('transactions');
    rootElement.setAttribute('xmlns:xsi', 'http://www.w3.org/2001/XMLSchema-instance');
    rootElement.setAttribute('xsi:noNamespaceSchemaLocation', 'reservations.xsd');
    const transactionElement = generateTransaction(htmlDoc, lang, minPrice, maxPrice,
roomsCount, floorsCount,
        amenitiesCount, minPrice, maxPrice);
    rootElement.appendChild(transactionElement);

    const xmlSerializer = new XMLSerializer();

    const header = document.getElementById("specimenHeader");
    switch (filename) {
        case "firstSpecimen":
            header.innerText = "Екземпляр на първия хотел:\n";
            break;
        case "secondSpecimen":
            header.innerText = "Екземпляр на втория хотел:\n";
            break;
        case "thirdSpecimen":
            header.innerText = "Екземпляр на третия хотел:\n";
            break;
        case "fourthSpecimen":
            header.innerText = "Екземпляр на четвъртия хотел:\n";
            break;
        case "fifthSpecimen":
            header.innerText = "Екземпляр на петия хотел:\n";
            break;
        default:
            break;
    }

    header.style.color = "green";

    const contentParagraph = document.getElementById("specimenParagraph");
    contentParagraph.style.display = "block";
    htmlDoc.appendChild(rootElement);
    const xmlContent = formatXML(xmlSerializer.serializeToString(htmlDoc));
    downloadXMLFile(xmlContent, filename);
```

```
    const transformationSelect =
document.getElementById('transformation_format_select').value;
    if(transformationSelect === 'конвертирай до HTML формат') {
        loadFile("xml-to-html.xsl", function (xsl) {
            transformXMLString(xmlContent, xsl);
        });
    } else if (transformationSelect === 'конвертирай до XML формат (информация за
транзакцията)') {
        loadFile("xml-to-transaction-xml.xsl", function (xsl) {
            transformXMLString(xmlContent, xsl);
        });
    } else if (transformationSelect === 'конвертирай до XML формат (информация за наетата
стая)') {
        loadFile("xml-to-room-xml.xsl", function (xsl) {
            transformXMLString(xmlContent, xsl);
        });
    } else if (transformationSelect === 'конвертирай до Plain Text формат (информация за
транзакцията)') {
        loadFile("xml-to-transaction-plain-text.xsl", function (xsl) {
            transformXMLString(xmlContent, xsl);
        });
    } else if (transformationSelect === 'конвертирай до Plain Text формат (информация за
наетата стая)') {
        loadFile("xml-to-room-plain-text.xsl", function (xsl) {
            transformXMLString(xmlContent, xsl);
        });
    } else {
        alert('Моля изберете в какъв формат предпочитате да виждате данните за
генерираните резервации!');
    }
}

// specific functions that generates each of the specimens
function generateFirstSpecimen() {
    const roomsCount = 150;
    const floorsCount = 10;
    const amenitiesCount = 10;
    const minPrice = 200.00;
    const maxPrice = 1000.00;
    const filename = "firstSpecimen";

    generateSpecimen(roomsCount, floorsCount, amenitiesCount, minPrice, maxPrice,
filename);
}

function generateSecondSpecimen() {
    const roomsCount = 50;
    const floorsCount = 5;
    const amenitiesCount = 2;
    const minPrice = 50.00;
    const maxPrice = 150.00;
    const filename = "secondSpecimen";
```

```
    generateSpecimen(roomsCount, floorsCount, amenitiesCount, minPrice, maxPrice,
filename);
}

function generateThirdSpecimen() {
    const roomsCount = 20;
    const floorsCount = 3;
    const amenitiesCount = 5;
    const minPrice = 85.00;
    const maxPrice = 360.00;
    const filename = "thirdSpecimen";

    generateSpecimen(roomsCount, floorsCount, amenitiesCount, minPrice, maxPrice,
filename);
}

function generateFourthSpecimen() {
    const roomsCount = 100;
    const floorsCount = 20;
    const amenitiesCount = 10;
    const minPrice = 1000.00;
    const maxPrice = 1500.00;
    const filename = "fourthSpecimen";

    generateSpecimen(roomsCount, floorsCount, amenitiesCount, minPrice, maxPrice,
filename);
}

function generateFifthSpecimen() {
    const roomsCount = 5;
    const floorsCount = 1;
    const amenitiesCount = 4;
    const minPrice = 10.00;
    const maxPrice = 50.00;
    const filename = "fifthSpecimen";

    generateSpecimen(roomsCount, floorsCount, amenitiesCount, minPrice, maxPrice,
filename);
}
```

Версия “Набор от XSLT трансформации”

4 различни стилови таблици (XSLT) за трансформиране на XML документите в HTML код

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="html" encoding="UTF-8" indent="yes"/>

    <!-- First XSLT Transformation Template: from XML to HTML -->
```

```
    <xsl:template match="/transactions/transaction">
        <html>
            <head>
                <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"/>
                <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
                <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
            </head>
            <body>
                <h1>Reservation Details:</h1>
                <span><strong>Transaction ID:</strong> <xsl:value-of
select="/transactions/transaction/@id"/></span><br/>
                <span><strong>Paying Tool:</strong> <xsl:value-of
select="/transactions/transaction/@payingTool"/></span>
                <xsl:apply-templates select="reservation/room"/>
                <xsl:apply-templates select="reservation/guests"/>
                <xsl:apply-templates select="card"/>
            </body>
        </html>
    </xsl:template>

    <!-- Second XSLT Transformation Template: from XML to HTML -->
    <xsl:template match="reservation/room">
        <h3><strong>Room Details:</strong></h3>
        <table class="table">
            <tr>
                <th>Floor No:</th>
                <td><xsl:value-of select="floorNo"/></td>
            </tr>
            <tr>
                <th>Room No:</th>
                <td><xsl:value-of select="roomNo"/></td>
            </tr>
            <tr>
                <th>Max Capacity:</th>
                <td><xsl:value-of select="maxCapacity"/></td>
            </tr>
            <tr>
                <th>Has Personal Bath:</th>
                <td><xsl:value-of select="hasPersonalBath"/></td>
            </tr>
            <tr>
                <th>Is Available:</th>
                <td><xsl:value-of select="isAvailable"/></td>
            </tr>
            <tr>
                <th>Price:</th>
                <td>
                    <xsl:value-of select="price"/>
                    (<xsl:value-of select="price/@currency"/>)
                </td>
            </tr>
        </table>
```

```
        <hr style="border: 1px solid gray;"/>

        <!-- Hotel Details -->
        <div style="display: inline-block;">
            <h3><strong>Hotel:</strong></h3>
            <p>
                <strong>Name:</strong> <xsl:value-of select="hotel/name"/><br/>
                <strong>Address:</strong> <xsl:value-of
select="hotel/address/@country"/>,
                <xsl:value-of select="hotel/address/@town"/>,
                <xsl:value-of select="hotel/address/@location"/>
            </p>
        </div>

        <!-- Amenities Details -->
        <div style="display: inline-block; margin-left: 5rem;">
            <h3><strong>Amenities:</strong></h3>
            <xsl:for-each select="amenities/amenity">
                <span><xsl:value-of select="."/></span><br/>
            </xsl:for-each>
        </div>
    </xsl:template>

    <!-- Third XSLT Transformation Template: from XML to HTML -->
    <xsl:template match="reservation/guests">
        <div style="display: inline-block; margin-left: 5rem;">
            <h3><strong>Guests:</strong></h3>
            <xsl:for-each select="guest">
                <span><xsl:value-of select="@name"/> <xsl:value-of
select="@family"/></span><br/>
            </xsl:for-each>
        </div>
    </xsl:template>

    <!-- Fourth XSLT Transformation Template: from XML to HTML -->
    <xsl:template match="card">
        <div style="display: inline-block; margin-left: 5rem;">
            <h3><strong>Paying Details:</strong></h3>
            <p>
                <strong>IBAN:</strong> <xsl:value-of select="iban"/><br/>
                <strong>Balance:</strong> <xsl:value-of select="balance"/>
                (<xsl:value-of select="@currency"/>)
            </p>
        </div>
    </xsl:template>
</xsl:transform>
```

В горният код виждате 4 XSLT трансформации от XML до HTML код, които представят по-различна визуализация на XML екземплярите като данните за хотелската стая са представени в табличен вид, а допълнителни данни за трансакцията по дадена резервация, данни за хотела, в който се намира дадена стая, удобствата на стаята и допълнителни данни относно начина на

плащане са изнрсени в отделни контейнери (div-ове), така че потребителите да виждат данните за своите резервации по по-красив и интуитивен за тях наин.

2 различни стилови таблици (XSLT) за трансформиране на XML документите в друг XML код

```
<?xml version="1.0" encoding="UTF-8" ?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
    <!-- Second XSLT Transformation Template: from XML to other XML format -->
    <xsl:template match="/">
        <rooms>
            <xsl:for-each select="transactions/transaction/reservation/room">
                <room>
                    <roomNumber>
                        <xsl:value-of select="roomNo"/>
                    </roomNumber>
                    <floorNumber>
                        <xsl:value-of select="floorNo"/>
                    </floorNumber>
                    <maxCapacity>
                        <xsl:value-of select="maxCapacity"/>
                    </maxCapacity>
                    <hasPersonalBath>
                        <xsl:value-of select="hasPersonalBath"/>
                    </hasPersonalBath>
                    <amenities>
                        <xsl:for-each select="../amenities/amenity">
                            <amenity>
                                <xsl:value-of select="."/>
                            </amenity>
                        </xsl:for-each>
                    </amenities>
                </room>
            </xsl:for-each>
        </rooms>
    </xsl:template>
</xsl:transform>
<?xml version="1.0" encoding="UTF-8" ?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="xml" encoding="UTF-8" indent="yes"/>

    <!-- First XSLT Transformation Template: from XML to other XML format -->
    <xsl:template match="/">
        <transactions>
            <xsl:for-each select="transactions/transaction">
                <transaction>
                    <transactionId><xsl:value-of select="@id" /></transactionId>
                    <roomNumber><xsl:value-of select="reservation/room/roomNo"
/></roomNumber>
```

```
                    <guests>
                        <xsl:for-each select="reservation/guests/guest">
                            <guest>
                                <name><xsl:value-of select="@name" /></name>
                                <family><xsl:value-of select="@family" /></family>
                            </guest>
                        </xsl:for-each>
                    </guests>
                    <cardBalance><xsl:value-of select="card/balance" /></cardBalance>
                </transaction>
            </xsl:for-each>
        </transactions>
    </xsl:template>
</xsl:transform>
```

Горните две представяния на данни са от XML към друг XML формат и са свързани с по-компактен пренос на данни за трансакции и данни за наета хотелска стая от бизнес към бизнес. Едно такова представяне е изключително от полза, когато трябва да уеднаквяваме начините за комуникация от една система към друга, например, ако имаме за задача да направим софтуер за резервации, който да може да събира данни за хотели, апартманети и резервации от 2 различни системи каквито са - https://www.booking.com и https://www.airbnb.com например.

2 различни стилови таблици (XSLT) за трансформиране на XML документите в чист текст

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="text" encoding="UTF-8"/>

    <!-- Second XSLT Transformation Template: from XML to Plain Text -->
    <xsl:template match="/">
        Room No:
        <xsl:value-of select="/transactions/transaction/reservation/room/roomNo"/>
        Floor No:
        <xsl:value-of select="/transactions/transaction/reservation/room/floorNo"/>
        Max Capacity:
        <xsl:value-of select="/transactions/transaction/reservation/room/maxCapacity"/>
        Has Personal Bath:
        <xsl:value-of
select="/transactions/transaction/reservation/room/hasPersonalBath"/>
        Amenities:
        <xsl:for-each
select="/transactions/transaction/reservation/room/amenities/amenity">
            <xsl:value-of select="."/>
            <xsl:text>; </xsl:text>
        </xsl:for-each>
        <xsl:text>&#10;</xsl:text>
    </xsl:template>
</xsl:stylesheet>
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
    <xsl:output method="text" encoding="UTF-8" />

    <!-- First XSLT Transformation Template: from XML to Plain Text -->
    <xsl:template match="/">
        <xsl:for-each select="transactions/transaction">
            Transaction ID: <xsl:value-of select="@id" />
            Paying Tool: <xsl:value-of select="@payingTool" />
            Room No: <xsl:value-of select="reservation/room/roomNo" />
            Floor No: <xsl:value-of select="reservation/room/floorNo" />
            Price: <xsl:value-of select="reservation/room/price" /> <xsl:text>
(</xsl:text> <xsl:value-of select="reservation/room/price/@currency"/>
<xsl:text>)</xsl:text>
            Guests:
            <xsl:for-each select="reservation/guests/guest">
                <xsl:value-of select="@name" />
                <xsl:text> </xsl:text>
                <xsl:value-of select="@family" />
                <xsl:text>; </xsl:text>
            </xsl:for-each>
            <xsl:text>&#10;</xsl:text>
            Payed By: <xsl:value-of select="reservation/guests/guest[@email or
@phone]/@name"/>
            <xsl:text> </xsl:text><xsl:value-of select="reservation/guests/guest[@email
or @phone]/@family"/>
            Card Balance: <xsl:value-of select="card/balance" /> <xsl:text> (</xsl:text>
<xsl:value-of select="card/@currency"/> <xsl:text>)</xsl:text>
            <xsl:text>&#10;</xsl:text>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

Горните 2 представяния на информацията относно трансакциите и стаите са в чист текст, което е удобен начин за представяне на информация при системи, в които данните имат плоска структура, тоест няма твърде много нива на наследяване или твърде голяма абстракция. Случаят с хотелските резервации и по-точно формата, който създавам в този проект не е твърде абстрактен, което предполага лесно съхранение на данни и тяханата лесна манипулируемост дори в чист текст. Освен това предимството на един такъв формат е че той е лесно четим дори от човек, а при достатъчно добра конструкция има претенциите за лесна работа и с машината.

Заключения, които се описват с особен фокус върху:

Подобрения с добавяне на допълнителни атрибути, логика и т.н.

Доразвиване на схемата с добавяне на повече трансформации