#### Министерство науки и высшего образования Российской Федерации

#### Федеральное государственное бюджетное образовательное учреждение

#### высшего образования

#### «Владимирский государственный университет

#### имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №3

по дисциплине

"Интеграция кросс платформенных систем"

Выполнил:

ст. гр. ПРИм-124

Парахин К.В.

Приняла:

преп. Кафедры ИСПИ

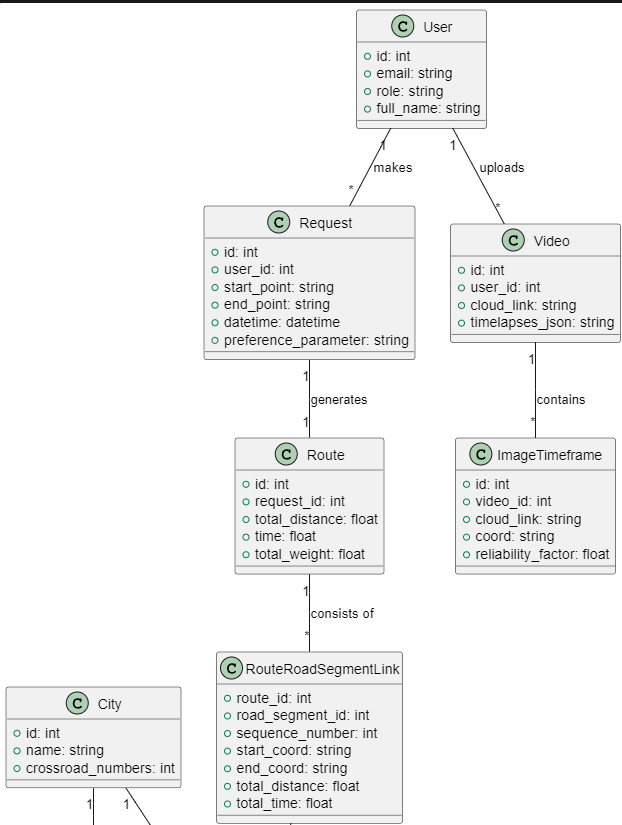
Тимофеев А.А.

Владимир, 2025 г.

## PОсновы XSLT. Преобразования средствами JAXP

Данная работа предусматривает разработку описания XSLT-преобразования, результат которого был бы эквивалентен результату, получаемому в ЛР2. При этом собственно XSLT-преобразование необходимо выполнить программно, используя средства JAXP, для программы сохраняются все условия, которые указаны в задании для предыдущей работы, но добавляются те, что указаны ниже.

Выполнение работы  
  
Предметная область связана с построением маршрутов по городу согласно качеству дорожного покрытия и дорожным условиям.  
Диаграмма классов UML данной модели выглядит так:



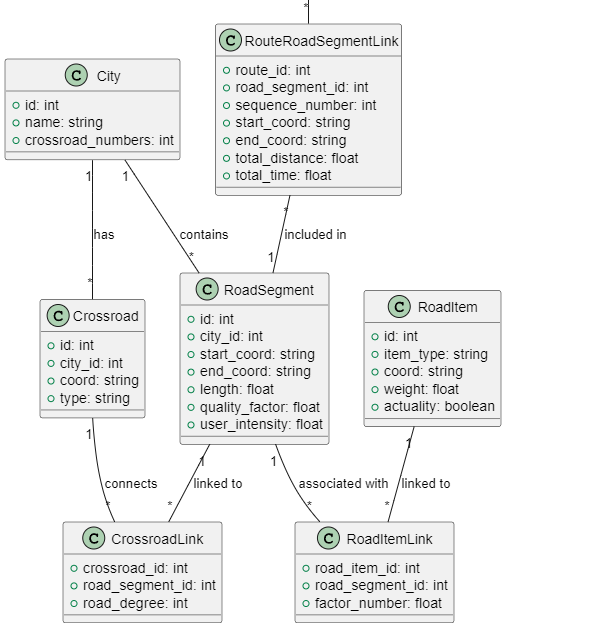


Рис. 1.1 – 1.2. Диаграмма классов UML

Далее нужно будет разработать программу на JAXP для выполнения XSLT-преобразования – выполненного на основе схем данных, соответствующим параметрам данной предметной области.

Сначала нужно написать пример XML-документа – который будет соответствовать требованиям (а именно наличие трех типов данных и данных, которые можно будет отображать в табличной форме)

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

<ns:document xmlns:ns="http://example.com/schema"

             xsi:schemaLocation="http://example.com/schema xml\_schema.xsd"

             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <ns:user>

        <id>1</id>

        <email>parahinkv@gmail.com</email>

        <role>admin</role>

        <full\_name>Kirill Parakhin</full\_name>

    </ns:user>

    <ns:request>

        <id>1</id>

        <user\_id>1</user\_id>

        <start\_point>40.7128, -74.0060</start\_point>

        <end\_point>40.7138, -74.0050</end\_point>

        <datetime>2025-03-09T12:00:00</datetime>

        <preference\_parameter>fastest</preference\_parameter>

        <status>pending</status>

    </ns:request>

    <ns:crossroad>

        <id>1</id>

        <city\_id>1</city\_id>

        <coord>40.7128, -74.0060</coord>

        <type>traffic\_light</type>

    </ns:crossroad>

    <ns:roadSegment>

        <id>1</id>

        <city\_id>1</city\_id>

        <start\_coord>40.7128, -74.0060</start\_coord>

        <end\_coord>40.7138, -74.0050</end\_coord>

        <length>500.5</length>

        <quality\_factor>0.8</quality\_factor>

        <user\_intensity>0.9</user\_intensity>

    </ns:roadSegment>

    <ns:roadSegment>

        <id>2</id>

        <city\_id>1</city\_id>

        <start\_coord>40.7138, -74.0050</start\_coord>

        <end\_coord>40.7148, -74.0040</end\_coord>

        <length>300.2</length>

        <quality\_factor>0.7</quality\_factor>

        <user\_intensity>0.6</user\_intensity>

    </ns:roadSegment>

</ns:document>

Далее необходимо реализовать программный код для описания XSLT-преобразования

XSLT-описание XML-документа должно содержать именованные параметризованные шаблоны, циклы, нумерацию/сортировку, ссылку на HTML v 4.01 спецификацию

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

<xsl:stylesheet version="1.0"

    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

    xmlns:ns="http://example.com/schema">

    <xsl:output method="html" version="4.01" encoding="UTF-8" doctype-system="http://www.w3.org/TR/html4/strict.dtd"/>

    <xsl:template match="/">

        <html>

        <head>

            <meta charset="UTF-8"/>

            <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

            <title>XML Data</title>

        </head>

        <body>

            <h1>Road Segments Data</h1>

            <h2>User Info</h2>

            <xsl:apply-templates select="ns:document/ns:user"/>

            <h2>Request Info</h2>

            <xsl:apply-templates select="ns:document/ns:request"/>

            <h2>Road Segments</h2>

            <table border="1">

                <thead>

                    <tr>

                        <th>ID</th>

                        <th>Start Coord</th>

                        <th>End Coord</th>

                        <th>Length (m)</th>

                        <th>Quality Factor</th>

                        <th>User Intensity</th>

                        <th>Efficiency</th>

                    </tr>

                </thead>

                <tbody>

                    <xsl:apply-templates select="ns:document/ns:roadSegment">

                        <xsl:sort select="length" data-type="number" order="descending"/>

                    </xsl:apply-templates>

                </tbody>

                <tfoot>

                    <tr>

                        <td colspan="3">Total Length</td>

                        <td>

                            <xsl:value-of select="sum(ns:document/ns:roadSegment/ns:length)"/>

                        </td>

                        <td colspan="3"></td>

                    </tr>

                </tfoot>

            </table>

        </body>

        </html>

    </xsl:template>

    <xsl:template match="ns:user">

        <p><strong>Email:</strong> <xsl:value-of select="ns:email"/></p>

        <p><strong>Name:</strong> <xsl:value-of select="ns:full\_name"/></p>

    </xsl:template>

    <xsl:template match="ns:request">

        <p><strong>Start Point:</strong> <xsl:value-of select="ns:start\_point"/></p>

        <p><strong>End Point:</strong> <xsl:value-of select="ns:end\_point"/></p>

        <p><strong>Status:</strong> <xsl:value-of select="ns:status"/></p>

    </xsl:template>

    <xsl:template match="ns:roadSegment">

        <tr>

            <td><xsl:value-of select="ns:id"/></td>

            <td><xsl:value-of select="ns:start\_coord"/></td>

            <td><xsl:value-of select="ns:end\_coord"/></td>

            <td><xsl:value-of select="ns:length"/></td>

            <td><xsl:value-of select="ns:quality\_factor"/></td>

            <td><xsl:value-of select="ns:user\_intensity"/></td>

            <td>0,00</td>

        </tr>

    </xsl:template>

</xsl:stylesheet>

Программа на JAXP должна использовать источник и результат преобразования различных типов (SAXSource/DOMResult), выполнять обработку ошибок.

Она загружает сам XML-документ и XSLT-описание для него, проводит создание источника (SAXSource) – c помощью фабрики и парсера проходится по XML-документу, с помощью трансформера изучает XSLT-описание – и по итогу отдает на выход через поток вывода HTML-разметку в отдельный файл

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.\*;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import javax.xml.transform.stream.StreamSource;

import org.w3c.dom.Document;

import java.io.File;

import java.io.FileOutputStream;

import java.io.OutputStream;

public class JAXPTransformer {

    public static void main(String[] args) {

        try {

            File xmlFile = new File("kirill\_example.xml");

            File xsltFile = new File("description.xslt");

            File outputFile = new File("kirill\_xslt\_example.html");

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

            factory.setNamespaceAware(true);

            DocumentBuilder builder = factory.newDocumentBuilder();

            Document document = builder.parse(xmlFile);

            DOMSource domSource = new DOMSource(document);

            TransformerFactory transformerFactory = TransformerFactory.newInstance();

            Transformer transformer = transformerFactory.newTransformer(new StreamSource(xsltFile));

            transformer.setOutputProperty(OutputKeys.INDENT, "yes");

            OutputStream outputStream = new FileOutputStream(outputFile);

            StreamResult result = new StreamResult(outputStream);

            transformer.transform(domSource, result);

            outputStream.close();

            System.out.println("Transformation completed successfully");

        } catch (Exception e) {

            System.err.println("Error during transformation: " + e.getMessage());

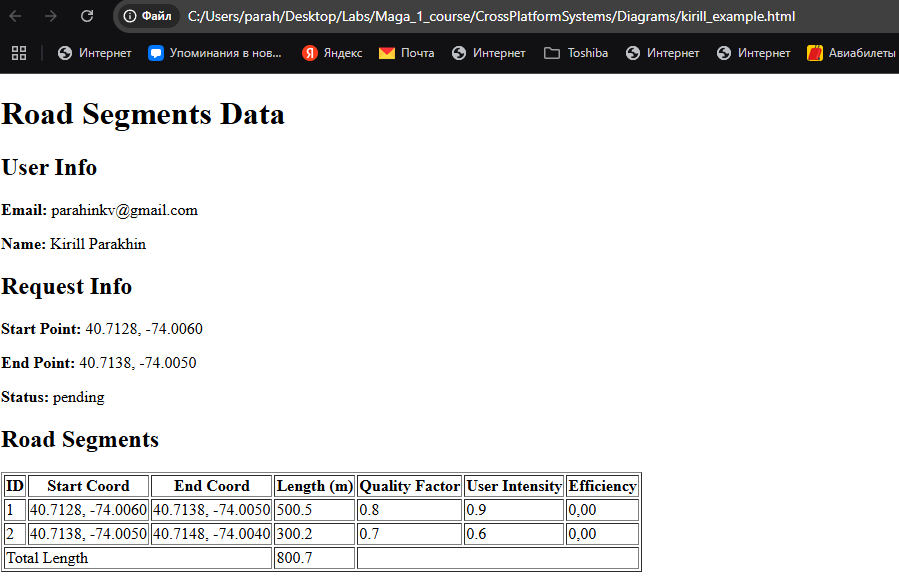
            e.printStackTrace();

        }

    }

}

По итогу, в результате XSLT-преобразования над XML-документом – получился HTML-документ, аналогичный полученному с помощью DOM API и SAX API в прошлой работе



Вывод

В результате выполнения работы было изучено XSLT-преобразование XML-документа, освоены различные способы анализа XML-документа с представлением его в виде HTML-документа.