#### 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 №2

по дисциплине

"Интеграция кросс платформенных систем"

Выполнил:

ст. гр. ПРИм-124

Парахин К.В.

Приняла:

преп. Кафедры ИСПИ

Тимофеев А.А.

Владимир, 2025 г.

## Основы JAXP. Программный анализ XML-документов

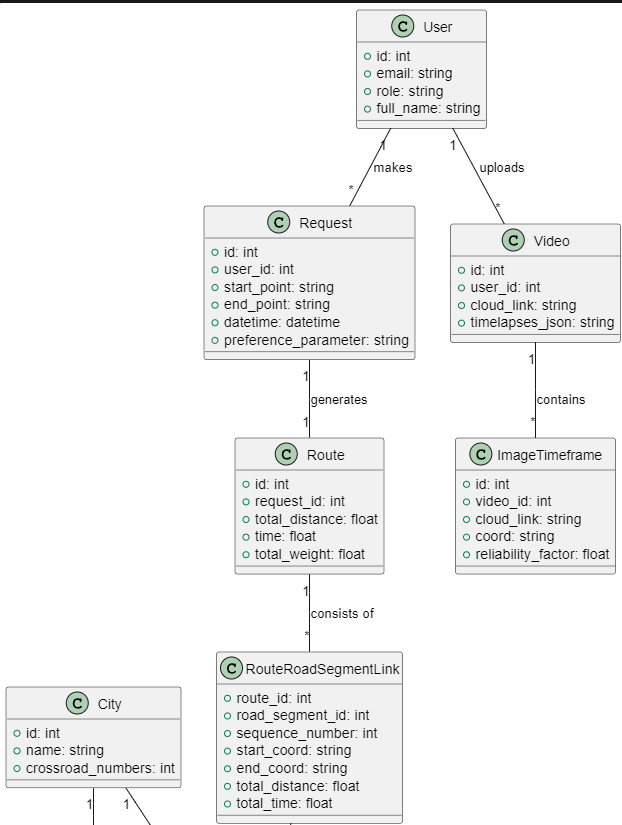
Данная работа предусматривает разработку программ(ы) анализа XML-документов, составленных в соответствии со схемой, разработанной в ЛР1. Суть анализа должна заключаться в формировании HTML-представления XML-документа, поданного программе на вход.

Возможные способы реализации:

1) две программы с идентичным поведением, использующие средства DOM API и SAX API, соответственно;

2) одна программа, использующая средства StAX API.

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



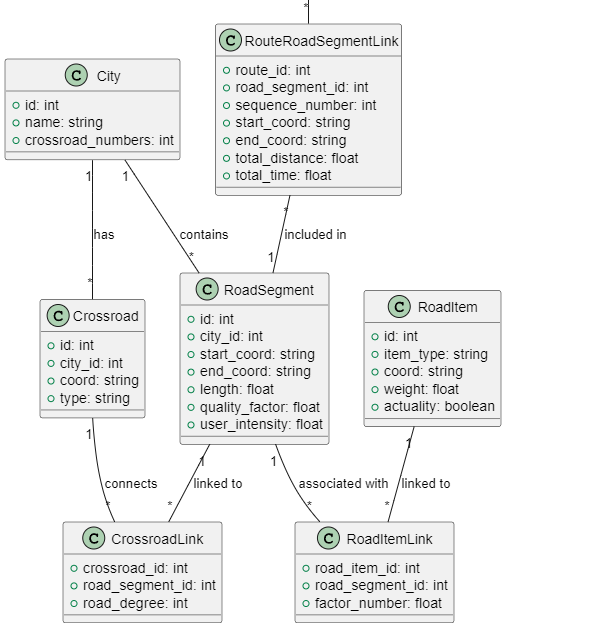


Рис. 1.1 – 1.2. Диаграмма классов UML

Далее нужно будет разработать программы на DOM API и SAX API – которые будут заниматься анализом XML-документов – выполненных на основе схем данных, соответствующим параметрам данной предметной области.   
  
За пример для схемы данных – для формирования XML-документа, будет взята данная XML-schema:

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

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

           targetNamespace="http://example.com/schema"

           elementFormDefault="qualified">

    <xs:simpleType name="emailType">

        <xs:restriction base="xs:string">

            <xs:minLength value="5"/>

            <xs:maxLength value="255"/>

            <xs:pattern value="^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"/>

        </xs:restriction>

    </xs:simpleType>

    <xs:simpleType name="coordType">

        <xs:restriction base="xs:string"/>

    </xs:simpleType>

    <xs:simpleType name="roleType">

        <xs:restriction base="xs:string">

            <xs:enumeration value="admin"/>

            <xs:enumeration value="user"/>

        </xs:restriction>

    </xs:simpleType>

    <xs:simpleType name="statusType">

        <xs:restriction base="xs:string">

            <xs:enumeration value="pending"/>

            <xs:enumeration value="completed"/>

            <xs:enumeration value="cancelled"/>

        </xs:restriction>

    </xs:simpleType>

    <xs:simpleType name="typeType">

        <xs:restriction base="xs:string">

            <xs:enumeration value="traffic\_light"/>

            <xs:enumeration value="roundabout"/>

        </xs:restriction>

    </xs:simpleType>

    <xs:complexType name="userType">

        <xs:sequence>

            <xs:element name="id" type="xs:int"/>

            <xs:element name="email" type="ns:emailType"/>

            <xs:element name="role" type="ns:roleType"/>

            <xs:element name="full\_name" type="xs:string"/>

        </xs:sequence>

    </xs:complexType>

    <xs:complexType name="requestType">

        <xs:sequence>

            <xs:element name="id" type="xs:int"/>

            <xs:element name="user\_id" type="xs:int"/>

            <xs:element name="start\_point" type="ns:coordType"/>

            <xs:element name="end\_point" type="ns:coordType"/>

            <xs:element name="datetime" type="xs:string"/>

            <xs:element name="preference\_parameter" type="xs:string"/>

            <xs:element name="status" type="ns:statusType"/>

        </xs:sequence>

    </xs:complexType>

    <xs:complexType name="crossroadType">

        <xs:sequence>

            <xs:element name="id" type="xs:int"/>

            <xs:element name="city\_id" type="xs:int"/>

            <xs:element name="coord" type="ns:coordType"/>

            <xs:element name="type" type="ns:typeType"/>

        </xs:sequence>

    </xs:complexType>

    <xs:complexType name="roadSegmentType">

        <xs:sequence>

            <xs:element name="id" type="xs:int"/>

            <xs:element name="city\_id" type="xs:int"/>

            <xs:element name="start\_coord" type="ns:coordType"/>

            <xs:element name="end\_coord" type="ns:coordType"/>

            <xs:element name="length" type="xs:float"/>

            <xs:element name="quality\_factor" type="xs:float"/>

            <xs:element name="user\_intensity" type="xs:float"/>

        </xs:sequence>

    </xs:complexType>

    <xs:element name="user" type="ns:userType"/>

    <xs:element name="request" type="ns:requestType"/>

    <xs:element name="crossroad" type="ns:crossroadType"/>

    <xs:element name="roadSegment" type="ns:roadSegmentType"/>

    <xs:complexType name="abstractRouteType" abstract="true">

        <xs:sequence>

            <xs:element name="route\_id" type="xs:int"/>

            <xs:element name="total\_distance" type="xs:float"/>

            <xs:element name="total\_time" type="xs:float"/>

        </xs:sequence>

    </xs:complexType>

    <xs:complexType name="routeType">

        <xs:complexContent>

            <xs:extension base="ns:abstractRouteType">

                <xs:sequence>

                    <xs:element name="route\_road\_segment" type="ns:roadSegmentType"/>

                </xs:sequence>

            </xs:extension>

        </xs:complexContent>

    </xs:complexType>

    <xs:key name="userKey">

        <xs:selector xpath="ns:user"/>

        <xs:field xpath="ns:user/id"/>

    </xs:key>

    <xs:keyref name="userKeyRef" refer="userKey">

        <xs:selector xpath="ns:request"/>

        <xs:field xpath="ns:request/user\_id"/>

    </xs:keyref>

</xs:schema>

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 анализа данного XML-документа – с помощью модели DOM API и модели SAX API

Начнем с модели DOM API. Для выполнения задачи будет использоваться фреймворк JAXP – предоставляющий API для работы с XML-документами

Программа выглядит таким образом – она формирует на каждом XML-объекте свой DOM-объект – который будет отображаться в разметке .html:

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import java.io.\*;

import org.xml.sax.SAXException;

public class DOMParser {

    public static void main(String[] args) {

        try {

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

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

            DocumentBuilder builder = factory.newDocumentBuilder();

            Document document = builder.parse(inputFile);

            document.getDocumentElement().normalize();

            Node userNode = document.getElementsByTagName("ns:user").item(0);

            String userEmail = ((Element) userNode).getElementsByTagName("email").item(0).getTextContent();

            String userName = ((Element) userNode).getElementsByTagName("full\_name").item(0).getTextContent();

            Node requestNode = document.getElementsByTagName("ns:request").item(0);

            String requestStartPoint = ((Element) requestNode).getElementsByTagName("start\_point").item(0).getTextContent();

            String requestEndPoint = ((Element) requestNode).getElementsByTagName("end\_point").item(0).getTextContent();

            String requestStatus = ((Element) requestNode).getElementsByTagName("status").item(0).getTextContent();

            NodeList roadSegmentList = document.getElementsByTagName("ns:roadSegment");

            StringBuilder htmlContent = new StringBuilder();

            htmlContent.append("<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n")

                       .append("<meta charset=\"UTF-8\">\n")

                       .append("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\n")

                       .append("<title>XML Data</title>\n</head>\n<body>\n<h1>Road Segments Data</h1>\n")

                       .append("<h2>User Info</h2>\n")

                       .append("<p><strong>Email:</strong> ").append(userEmail).append("</p>\n")

                       .append("<p><strong>Name:</strong> ").append(userName).append("</p>\n")

                       .append("<h2>Request Info</h2>\n")

                       .append("<p><strong>Start Point:</strong> ").append(requestStartPoint).append("</p>\n")

                       .append("<p><strong>End Point:</strong> ").append(requestEndPoint).append("</p>\n")

                       .append("<p><strong>Status:</strong> ").append(requestStatus).append("</p>\n")

                       .append("<h2>Road Segments</h2>\n")

                       .append("<table border=\"1\">\n<thead>\n<tr>\n<th>ID</th>\n<th>Start Coord</th>\n<th>End Coord</th>\n")

                       .append("<th>Length (m)</th>\n<th>Quality Factor</th>\n<th>User Intensity</th>\n<th>Efficiency</th>\n</tr>\n</thead>\n<tbody>");

            double totalLength = 0;

            for (int i = 0; i < roadSegmentList.getLength(); i++) {

                Node node = roadSegmentList.item(i);

                if (node.getNodeType() == Node.ELEMENT\_NODE) {

                    Element element = (Element) node;

                    String id = element.getElementsByTagName("id").item(0).getTextContent();

                    String startCoord = element.getElementsByTagName("start\_coord").item(0).getTextContent();

                    String endCoord = element.getElementsByTagName("end\_coord").item(0).getTextContent();

                    double length = Double.parseDouble(element.getElementsByTagName("length").item(0).getTextContent());

                    double qualityFactor = Double.parseDouble(element.getElementsByTagName("quality\_factor").item(0).getTextContent());

                    double userIntensity = Double.parseDouble(element.getElementsByTagName("user\_intensity").item(0).getTextContent());

                    double efficiency = (qualityFactor \* userIntensity) / length;

                    htmlContent.append("<tr>\n")

                               .append("<td>").append(id).append("</td>\n")

                               .append("<td>").append(startCoord).append("</td>\n")

                               .append("<td>").append(endCoord).append("</td>\n")

                               .append("<td>").append(length).append("</td>\n")

                               .append("<td>").append(qualityFactor).append("</td>\n")

                               .append("<td>").append(userIntensity).append("</td>\n")

                               .append("<td>").append(String.format("%.2f", efficiency)).append("</td>\n")

                               .append("</tr>");

                    totalLength += length;

                }

            }

            htmlContent.append("</tbody>\n<tfoot>\n<tr>\n<td colspan=\"3\">Total Length</td>\n")

                       .append("<td>").append(totalLength).append("</td>\n<td colspan=\"3\"></td>\n</tr>\n</tfoot>\n</table>\n</body>\n</html>");

            System.out.println(htmlContent.toString());

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

Далее будет использоваться SAX API – который уже обрабатывает XML-документ не на основе DOM-модели – а на простой модели событий, связанных с просмотром обработчиков исходного документа.

Для удобства работы и грамотного взаимодействия с объектами для основных сущностей: User, Request и RoadSegment – были созданы отдельные классы с методами обращения к их полям – благодаря чему получается более качественный ООП подход к формированию HTML-верстки

import org.xml.sax.\*;

import org.xml.sax.helpers.\*;

import javax.xml.parsers.\*;

import java.io.\*;

public class SAXParserExample {

    public static void main(String[] args) {

        try {

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

            SAXParserFactory factory = SAXParserFactory.newInstance();

            SAXParser saxParser = factory.newSAXParser();

            RoadSegmentHandler handler = new RoadSegmentHandler();

            saxParser.parse(inputFile, handler);

            System.out.println(handler.generateHTML());

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

class User {

    private String id;

    private String email;

    private String fullName;

    private String role;

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getFullName() { return fullName; }

    public void setFullName(String fullName) { this.fullName = fullName; }

    public String getRole() { return role; }

    public void setRole(String role) { this.role = role; }

}

class Request {

    private String id;

    private String userId;

    private String startPoint;

    private String endPoint;

    private String datetime;

    private String preferenceParameter;

    private String status;

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public String getUserId() { return userId; }

    public void setUserId(String userId) { this.userId = userId; }

    public String getStartPoint() { return startPoint; }

    public void setStartPoint(String startPoint) { this.startPoint = startPoint; }

    public String getEndPoint() { return endPoint; }

    public void setEndPoint(String endPoint) { this.endPoint = endPoint; }

    public String getDatetime() { return datetime; }

    public void setDatetime(String datetime) { this.datetime = datetime; }

    public String getPreferenceParameter() { return preferenceParameter; }

    public void setPreferenceParameter(String preferenceParameter) { this.preferenceParameter = preferenceParameter; }

    public String getStatus() { return status; }

    public void setStatus(String status) { this.status = status; }

}

class RoadSegment {

    private String id;

    private String cityId;

    private String startCoord;

    private String endCoord;

    private double length;

    private double qualityFactor;

    private double userIntensity;

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public String getCityId() { return cityId; }

    public void setCityId(String cityId) { this.cityId = cityId; }

    public String getStartCoord() { return startCoord; }

    public void setStartCoord(String startCoord) { this.startCoord = startCoord; }

    public String getEndCoord() { return endCoord; }

    public void setEndCoord(String endCoord) { this.endCoord = endCoord; }

    public double getLength() { return length; }

    public void setLength(double length) { this.length = length; }

    public double getQualityFactor() { return qualityFactor; }

    public void setQualityFactor(double qualityFactor) { this.qualityFactor = qualityFactor; }

    public double getUserIntensity() { return userIntensity; }

    public void setUserIntensity(double userIntensity) { this.userIntensity = userIntensity; }

}

class RoadSegmentHandler extends DefaultHandler {

    private StringBuilder htmlContent;

    private double totalLength;

    private String currentElement;

    private RoadSegment currentSegment;

    private User currentUser;

    private Request currentRequest;

    public RoadSegmentHandler() {

        htmlContent = new StringBuilder();

        totalLength = 0;

    }

    @Override

    public void startDocument() {

        htmlContent.append("<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n")

                   .append("<meta charset=\"UTF-8\">\n")

                   .append("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\n")

                   .append("<title>XML Data</title>\n</head>\n<body>\n<h1>Road Segments Data</h1>\n");

    }

    @Override

    public void endDocument() {

        htmlContent.append("</body>\n</html>");

    }

    @Override

    public void startElement(String uri, String localName, String qName, Attributes attributes) {

        currentElement = qName;

        if ("ns:roadSegment".equals(currentElement)) {

            currentSegment = new RoadSegment();

        } else if ("ns:user".equals(currentElement)) {

            currentUser = new User();

        } else if ("ns:request".equals(currentElement)) {

            currentRequest = new Request();

        }

    }

    @Override

    public void characters(char[] ch, int start, int length) {

        String content = new String(ch, start, length).trim();

        if ("email".equals(currentElement)) {

            currentUser.setEmail(content);

        } else if ("full\_name".equals(currentElement)) {

            currentUser.setFullName(content);

        } else if ("role".equals(currentElement)) {

            currentUser.setRole(content);

        } else if ("id".equals(currentElement)) {

            if (currentSegment != null) {

                currentSegment.setId(content);

            } else if (currentRequest != null) {

                currentRequest.setId(content);

            } else if (currentUser != null) {

                currentUser.setId(content);

            }

        } else if ("start\_coord".equals(currentElement)) {

            if (currentSegment != null) {

                currentSegment.setStartCoord(content);

            }

        } else if ("end\_coord".equals(currentElement)) {

            if (currentSegment != null) {

                currentSegment.setEndCoord(content);

            }

        } else if ("length".equals(currentElement)) {

            if (currentSegment != null) {

                currentSegment.setLength(Double.parseDouble(content));

            }

        } else if ("quality\_factor".equals(currentElement)) {

            if (currentSegment != null) {

                currentSegment.setQualityFactor(Double.parseDouble(content));

            }

        } else if ("user\_intensity".equals(currentElement)) {

            if (currentSegment != null) {

                currentSegment.setUserIntensity(Double.parseDouble(content));

            }

        } else if ("start\_point".equals(currentElement)) {

            if (currentRequest != null) {

                currentRequest.setStartPoint(content);

            }

        } else if ("end\_point".equals(currentElement)) {

            if (currentRequest != null) {

                currentRequest.setEndPoint(content);

            }

        } else if ("datetime".equals(currentElement)) {

            if (currentRequest != null) {

                currentRequest.setDatetime(content);

            }

        } else if ("preference\_parameter".equals(currentElement)) {

            if (currentRequest != null) {

                currentRequest.setPreferenceParameter(content);

            }

        } else if ("status".equals(currentElement)) {

            if (currentRequest != null) {

                currentRequest.setStatus(content);

            }

        }

    }

    @Override

    public void endElement(String uri, String localName, String qName) {

        if ("ns:roadSegment".equals(qName)) {

            double efficiency = (currentSegment.getQualityFactor() \* currentSegment.getUserIntensity()) / currentSegment.getLength();

            htmlContent.append("<tr>\n")

                       .append("<td>").append(currentSegment.getId()).append("</td>\n")

                       .append("<td>").append(currentSegment.getStartCoord()).append("</td>\n")

                       .append("<td>").append(currentSegment.getEndCoord()).append("</td>\n")

                       .append("<td>").append(currentSegment.getLength()).append("</td>\n")

                       .append("<td>").append(currentSegment.getQualityFactor()).append("</td>\n")

                       .append("<td>").append(currentSegment.getUserIntensity()).append("</td>\n")

                       .append("<td>").append(String.format("%.2f", efficiency)).append("</td>\n")

                       .append("</tr>");

            totalLength += currentSegment.getLength();

        }

    }

    public String generateHTML() {

        htmlContent.append("<h2>User Info</h2>\n")

                   .append("<p><strong>Email:</strong> ").append(currentUser.getEmail()).append("</p>\n")

                   .append("<p><strong>Name:</strong> ").append(currentUser.getFullName()).append("</p>\n")

                   .append("<h2>Request Info</h2>\n")

                   .append("<p><strong>Start Point:</strong> ").append(currentRequest.getStartPoint()).append("</p>\n")

                   .append("<p><strong>End Point:</strong> ").append(currentRequest.getEndPoint()).append("</p>\n")

                   .append("<p><strong>Status:</strong> ").append(currentRequest.getStatus()).append("</p>\n")

                   .append("<h2>Road Segments</h2>\n")

                   .append("<table border=\"1\">\n<thead>\n<tr>\n<th>ID</th>\n<th>Start Coord</th>\n<th>End Coord</th>\n")

                   .append("<th>Length (m)</th>\n<th>Quality Factor</th>\n<th>User Intensity</th>\n<th>Efficiency</th>\n</tr>\n</thead>\n<tbody>");

        htmlContent.append("</tbody>\n<tfoot>\n<tr>\n<td colspan=\"3\">Total Length</td>\n")

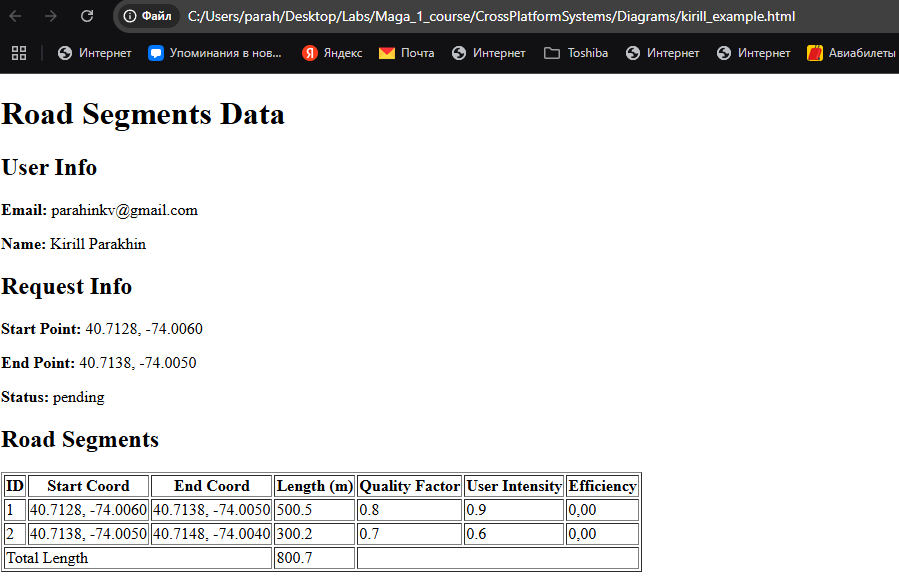
                   .append("<td>").append(totalLength).append("</td>\n<td colspan=\"3\"></td>\n</tr>\n</tfoot>\n</table>\n");

        return htmlContent.toString();

    }

}

По итогу, обе программы, используя чуть разные подходы – генерируют на основе представленного выше XML-документа – идентичный выходной HTML-документ



Вывод

В результате выполнения работы были изучены различные способы представления и анализа XML-документов, как с помощью программы на DOM API, так и SAX API