**Министерство образования и науки Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО**”

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИИ ВЕБ-СЕРВИСОВ»**

Студент: Су Минцзе

Группа: P41141

Преподаватель: Дергачев Андрей Михайлович

\

Санкт-Петербург

2021

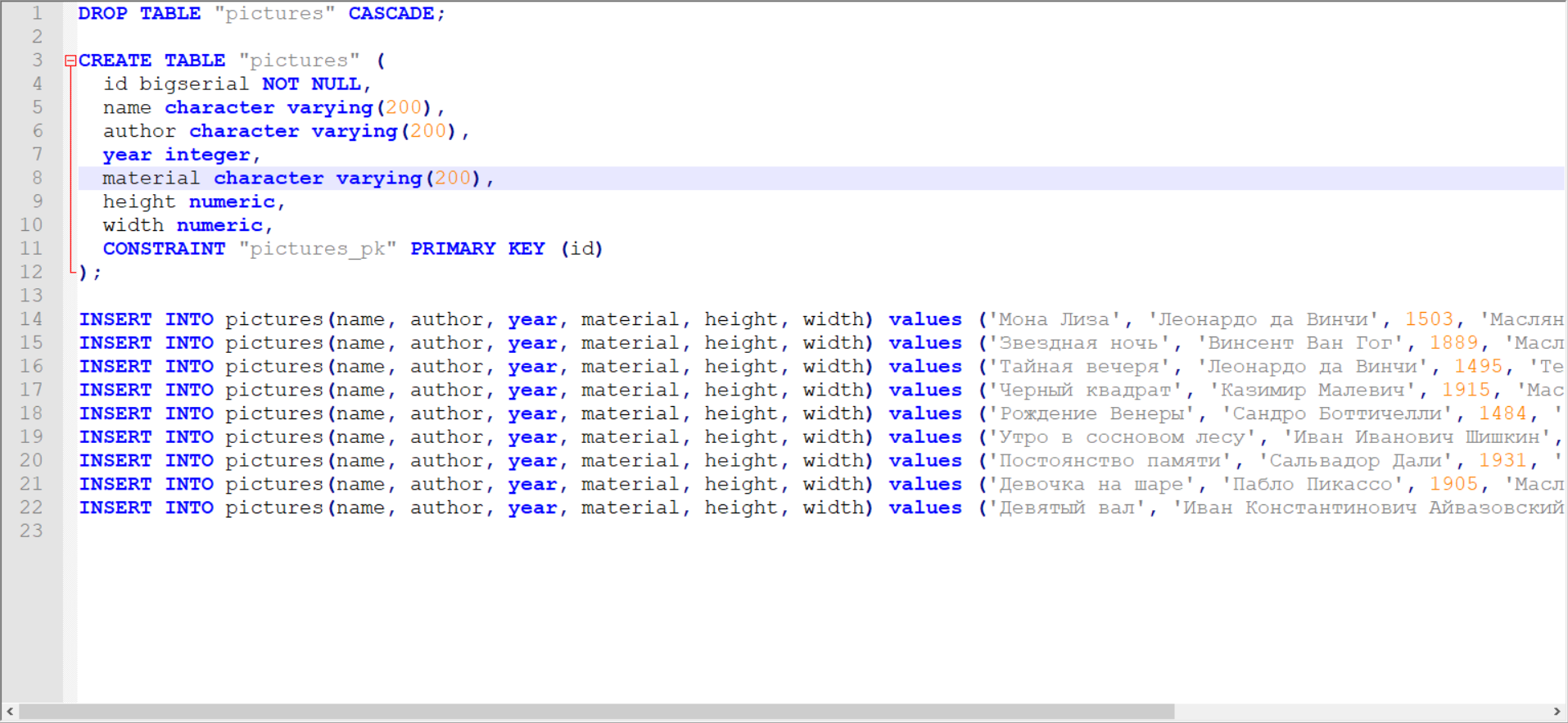
**Задание:**

В данной работе требуется создать таблицу в БД, содержащую не менее5 полей, а также реализовать возможность поиска по любым комбинациямполей с помощью SOAP-сервиса. Данные для поиска должны передаваться в

метод сервиса в качестве аргументов.Веб-сервис необходимо реализовать в виде standalone-приложения и J2EE-приложения. При реализации в виде J2EE-приложения следует на стороне сервера приложений настроить источник данных, и осуществлять его

инъекцию в коде сервиса.Для демонстрации работы разработанных сервисов следует также разработать и клиентское консольное приложение

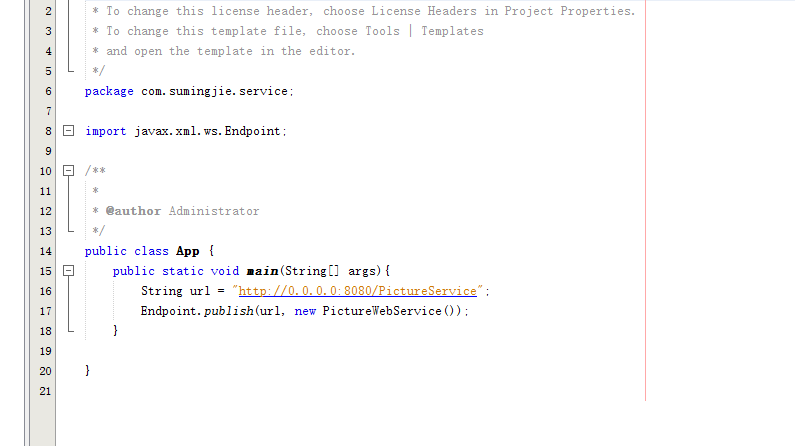
Листинг 1.1 – Файл Picture.sql



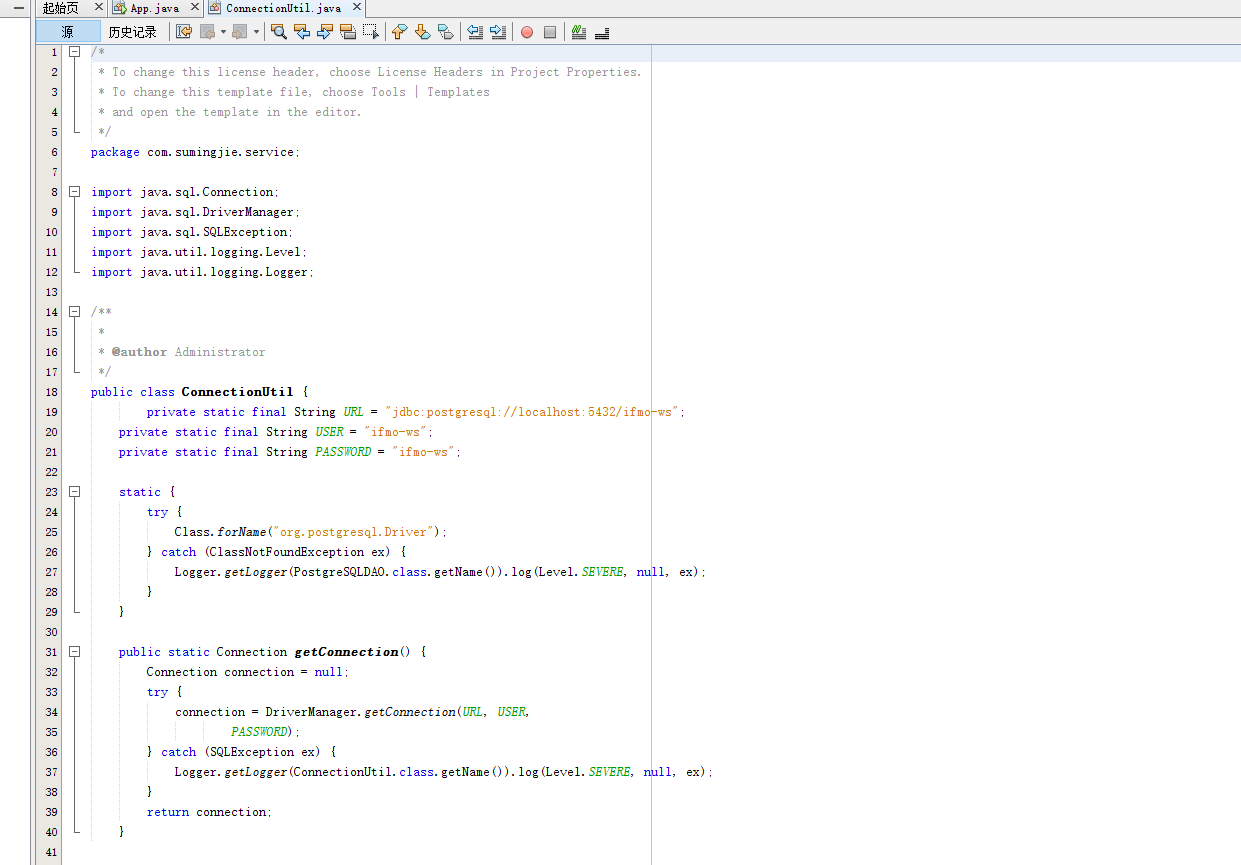
Код сервиса в виде standalone-приложения представлен в листингах 1.2-1.7. Класс App.java содержит main метод, и его основная цель – это запуститьвеб-сервис. ConnectionUtil.java используется для получения JDBC-соединений с базой данных. MyRequest.java представляет из себя структуру запроса, по которому будет выполняться поиск, который может включать поля id, name, author, year, material, height, width. Picture.java – POJO, который соответствует сущности, описанной в таблице picture базы данных. PictureWebServise.java содержит две операции: getAllPictures, который получает все картины, и findPictures, который ищет картины по параметру с типом MyRequest.

PostgreSQLDAO.java содержит методы для выборки данных из базы данных, а также установки этих данных в объекты класса Person.

Листинг 1.2 – Файл App.java



Листинг 1.3 – Файл ConnectionUtil.java



Листинг 1.4 – Файл MyRequest.java

@XmlAccessorType(XmlAccessType.FIELD)

@XmlRootElement(name = "MyRequest", namespace="http://service.sumingjie.com")

public class MyRequest implements Serializable{

@XmlElement(name = "id", required = false)

private int id;

@XmlElement(name = "name", required = false)

protected String name;

@XmlElement(name = "author", required = false)

private String author;

@XmlElement(name = "year", required = false)

private int year;

@XmlElement(name = "material", required = false)

private String material;

@XmlElement(name = "height", required = false)

private float height;

@XmlElement(name = "width", required = false)

private float width;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public int getYear() {

return year;

}

public void setYear(int year) {

this.year = year;

}

public String getMaterial() {

return material;

}

public void setMaterial(String material) {

this.material = material;

}

public float getHeight() {

return height;

}

public void setHeight(float height) {

this.height = height;

}

public float getWidth() {

return width;

}

public void setWidth(float width) {

this.width = width;

}

Листинг 1.5 – Файл Picture.java

public class Picture {

private int id;

private String name;

private String author;

private int year;

private String material;

private float height;

private float width;

public Picture() {

}

Picture(int id, String name, String author, int year, String material, float height, float width) {

this.id = id;

this.name = name;

this.author = author;

this.year = year;

this.material = material;

this.height = height;

this.width = width;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public int getYear() {

return year;

}

public void setYear(int year) {

this.year = year;

}

public String getMaterial() {

return material;

}

public void setMaterial(String material) {

this.material = material;

}

public float getHeight() {

return height;

}

public void setHeight(float height) {

this.height = height;

}

public float getWidth() {

return width;

}

public void setWidth(float width) {

this.width = width;

}

@Override

public String toString() {

return "Picture{" +

"name='" + name + '\'' +

", author='" + author + '\'' +

", year=" + year +

", material='" + material + '\'' +

", height=" + height +

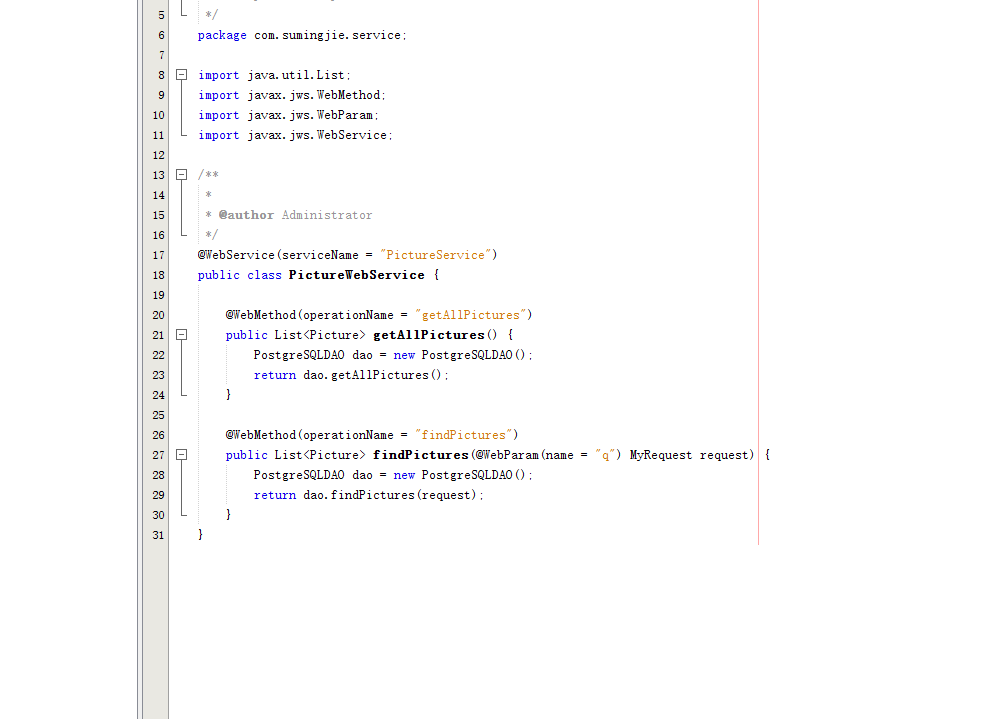
", width=" + width +

'}';

}

}

Листинг 1.6 – Файл PictureWebService.java



Листинг 1.7 – Файл PostgreSQLDAO.java

public class PostgreSQLDAO {

private Connection connection;

PostgreSQLDAO() {

this.connection = ConnectionUtil.getConnection();

}

public List<Picture> getAllPictures() {

return executeQuery("SELECT \* FROM pictures");

}

public List<Picture> findPictures(MyRequest request) {

List<Picture> pictures = new ArrayList<>();

String sql = "SELECT \* FROM pictures WHERE " +

"(id = " + request.getId() + " OR " + request.getId() + " = 0) AND " +

"(name = '" + request.getName() + "' OR '" + request.getName() + "' = '' OR '" + request.getName() + "' = '?') AND " +

"(author = '" + request.getAuthor() + "' OR '" + request.getAuthor() + "' = '' OR '" + request.getAuthor() + "' = '?') AND " +

"(year = " + request.getYear() + " OR " + request.getYear() + " = 0) AND " +

"(material = '" + request.getMaterial() + "' OR '" + request.getMaterial() + "' = '' OR '" + request.getMaterial() + "' = '?') AND " +

"(height = " + request.getHeight() + " OR " + request.getHeight() + " = 0) AND " +

"(width = " + request.getWidth() + " OR " + request.getWidth() + " = 0)";

return executeQuery(sql);

}

private List<Picture> executeQuery(String sql) {

List<Picture> pictures = new ArrayList<>();

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

String author = rs.getString("author");

int year = rs.getInt("year");

String material = rs.getString("material");

float height = rs.getFloat("height");

float width = rs.getFloat("width");

Picture picture = new Picture(id, name, author, year, material, height, width);

pictures.add(picture);

}

} catch (SQLException ex) {

Logger.getLogger(PostgreSQLDAO.class.getName()).log(Level.SEVERE, null, ex);

}

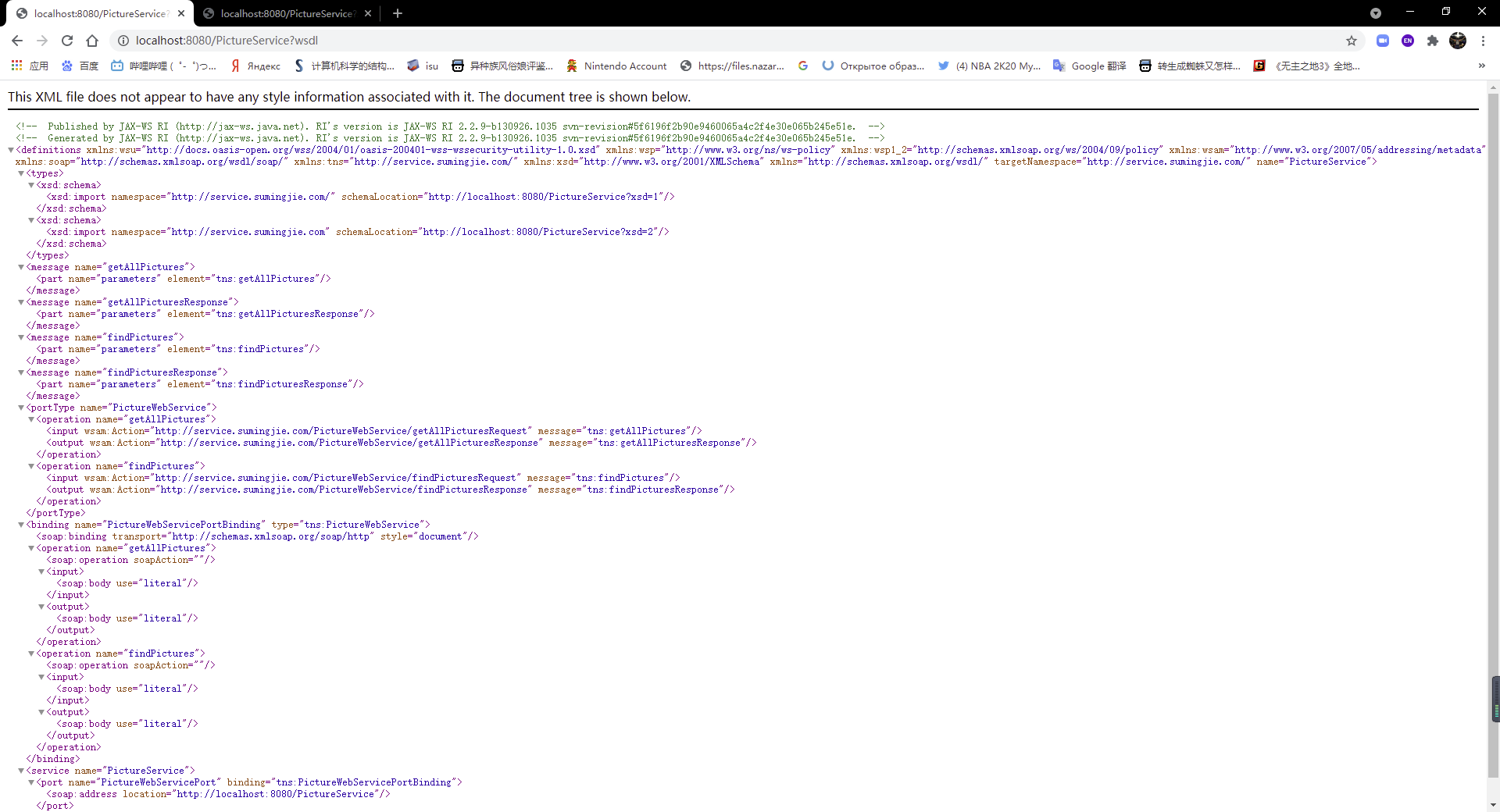
return pictures;

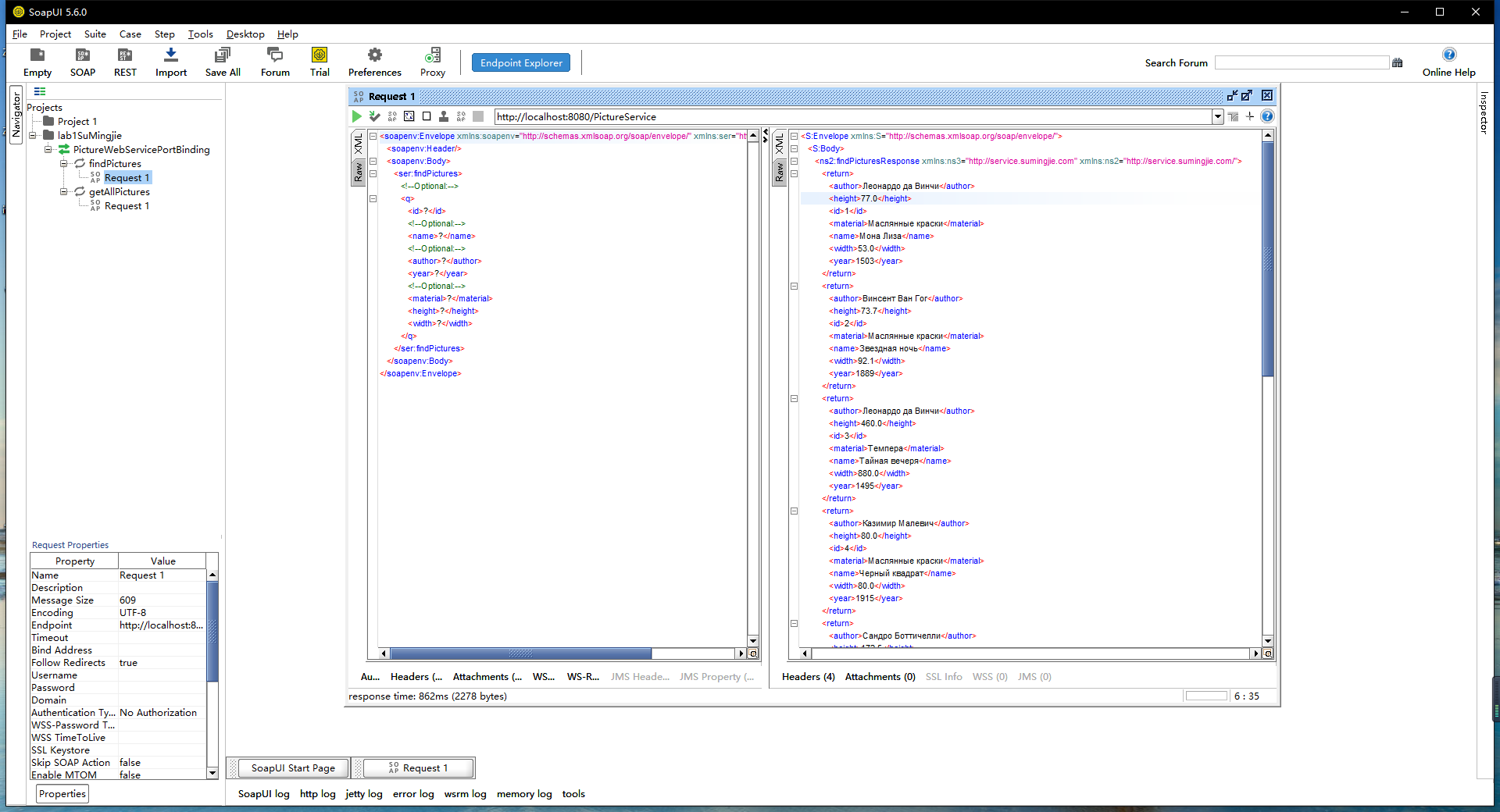
}

}

WSDL сервиса, представленный в листинге 1.8, доступен по адресу

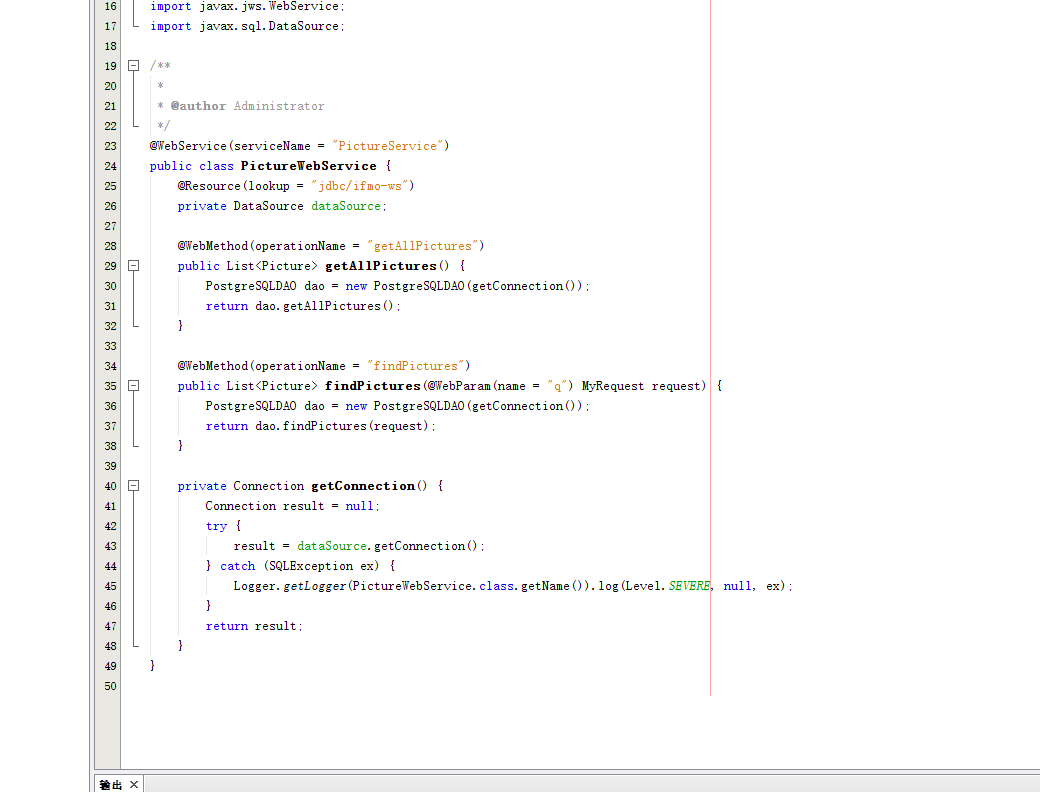
http://localhost:8080/PictureService?wsdl





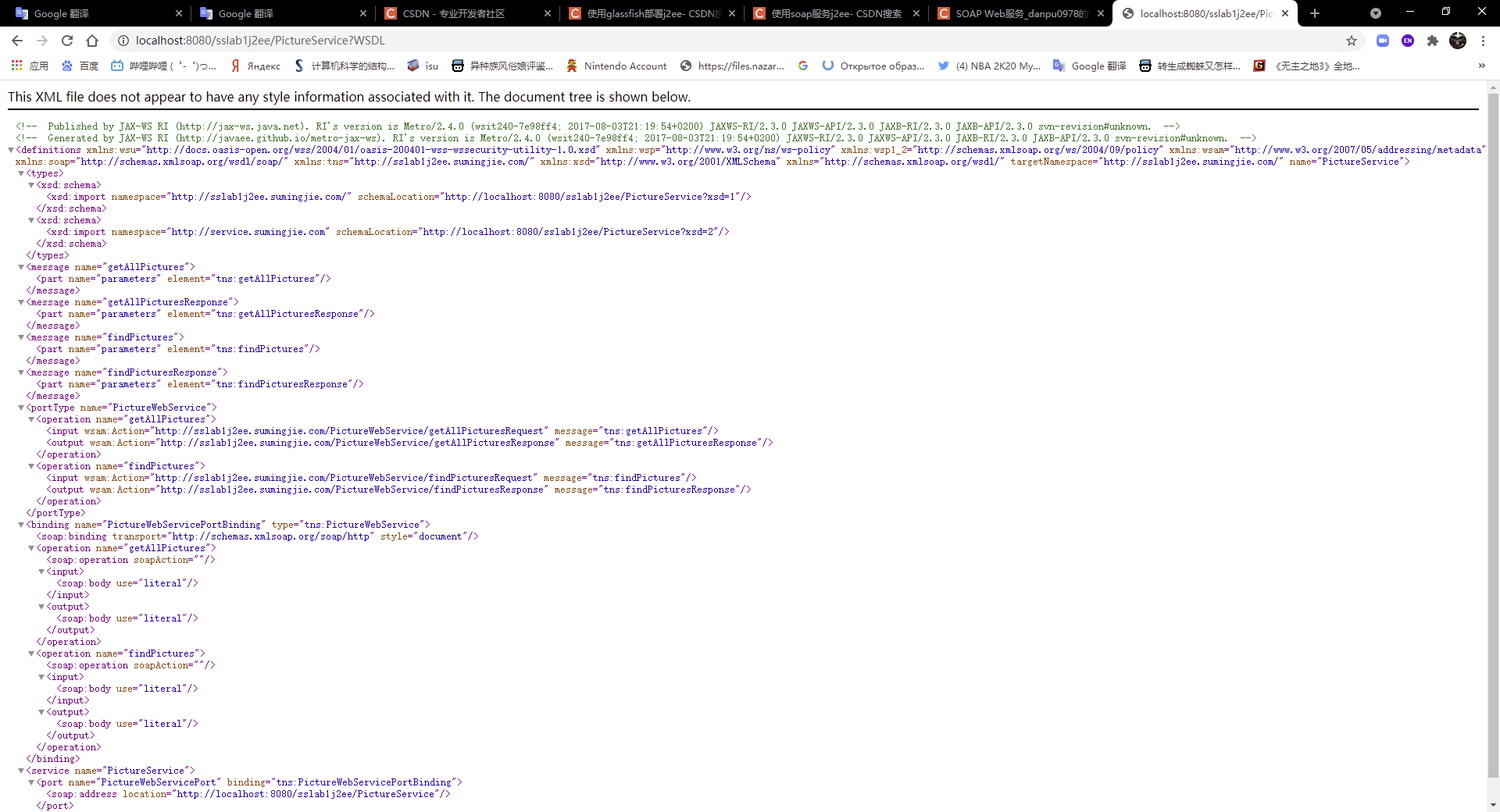
Код сервиса в виде J2EE-приложения представлен в листинге 1.9.

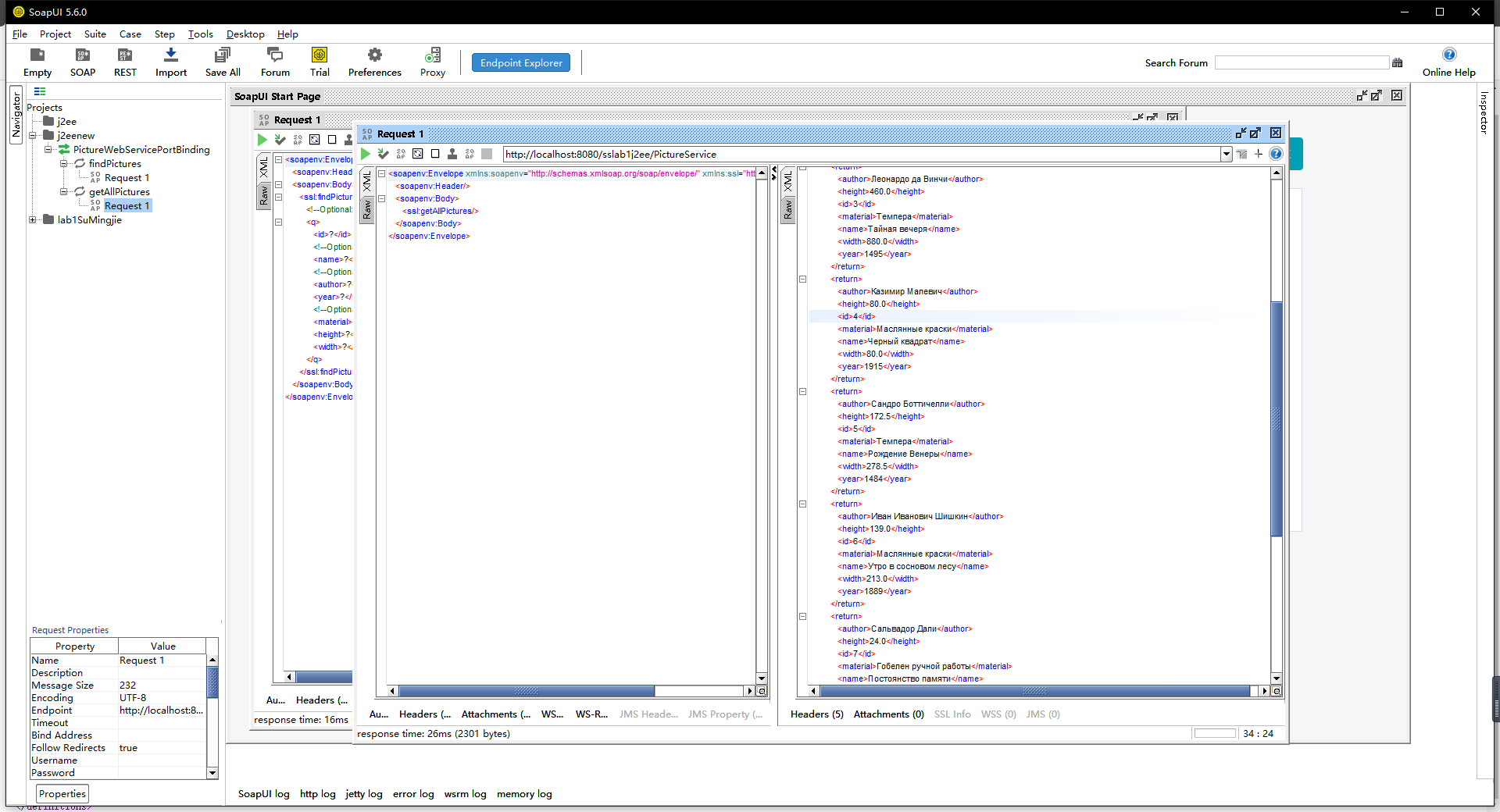
Классы MyRequest.java, Picture.java, PostgreSQLDAO.java аналогичны классам в standalone-приложении. PictureWebServise.java содержит две операции: getAllPictures, который получает все картины, и findPictures, который ищеткартины по параметру с типом MyRequest. Также содержит инъекцию источника данных, настроенного на стороне сервера приложений glassfish.



WSDL сервиса, представленный в листинге 1.10, доступен по адресуhttp://localhost:8080/sslab1j2ee/PictureService?wsdl

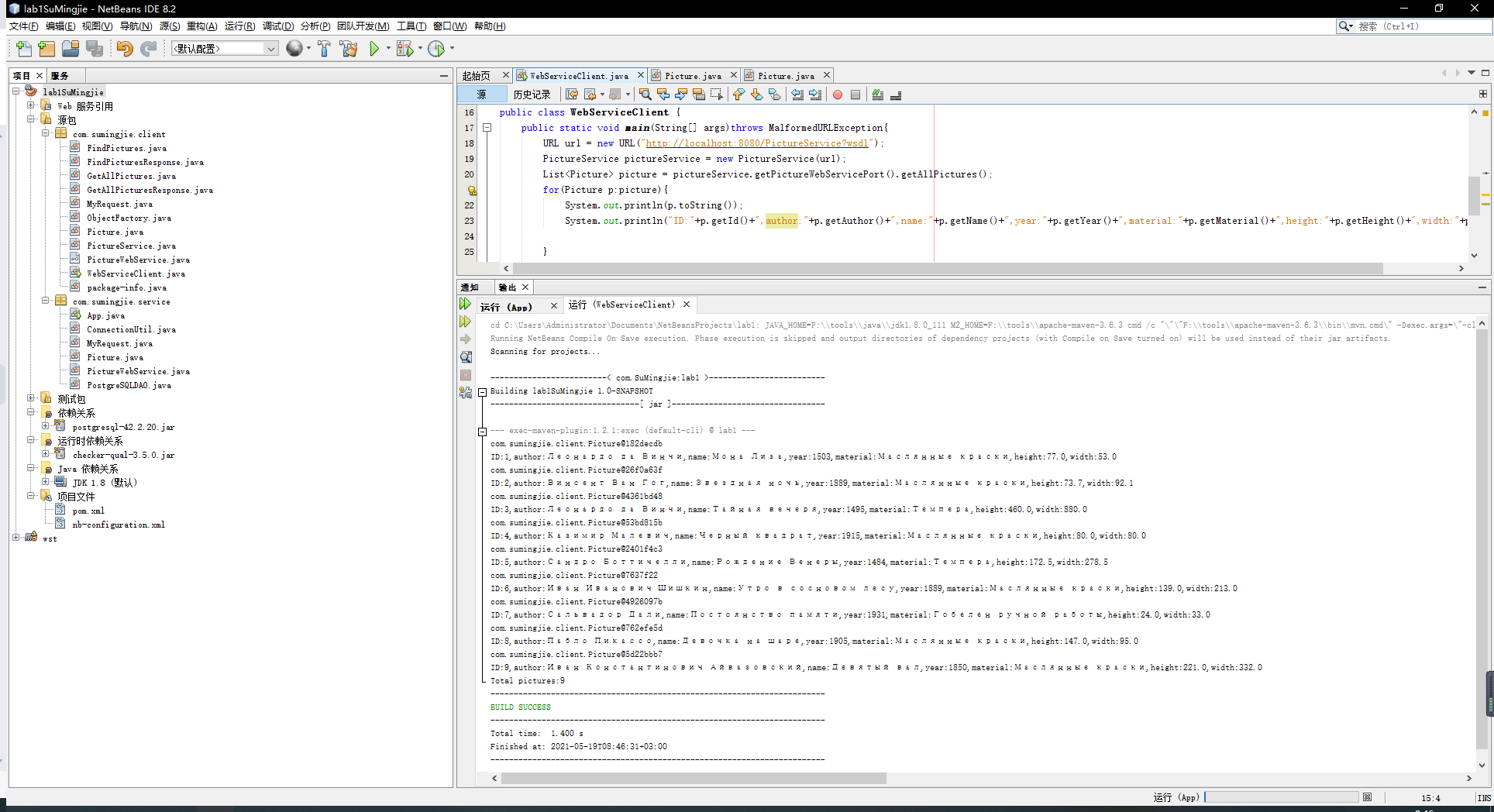
Листинг 1.10 – WSDL сервиса в виде J2EE-приложения





Листинг 1.11 – WebServiceClient.java





**Вывод:** в ходе выполнения работы была создана и заполнена таблица

pictures, а также реализована возможность поиска по любым комбинациям

полей с помощью SOAP-сервиса в виде standalone-приложения и

J2EE- приложения. Для демонстрации работы разработанных сервисов

было разработано клиентское консольное приложение