ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И
ОПТИКИ»

Факультет программной инженерии и компьютерной техники
Кафедра вычислительной техники

Отчёт по лабораторной работе №5

ПО КУРСУ

«Технологии веб-сервисов»

Реализация CRUD с помощью REST-сервиса

Выполнил: студент группы P4110
Маркитантов М. В.

Проверил: канд. техн. наук,
доцент Дергачев А. М.

Санкт-Петербург
2018

Задание:

В данной работе в веб-сервис, разработанный в четвертой работе, необходимо добавить методы для создания, изменения и удаления записей из таблицы.

Метод создания должен принимать значения полей новой записи, метод изменения – идентификатор изменяемой записи, а также новые значения полей, а метод удаления – только идентификатор удаляемой записи.

Метод создания должен возвращать идентификатор новой записи, а методы обновления или удаления – статус операции. В данной работе следует вносить изменения только в standalone-реализацию сервиса.

В соответствии с изменениями сервиса необходимо обновить и клиентское приложение.

Выполнение работы:

В результате выполнения четвертой лабораторной работы был написан REST-сервис в виде standalone-приложения. Дополним классы *PictureResource.java* и *PostgreSQLDAO.java* методами *create*, *update* и *delete*, которые позволяют создавать, обновлять и удалять записи из таблицы pictures соответственно.

Метод *create* получает на вход объект *picture*, представленный в JSON-формате. В случае успешного добавления возвращает id записи, иначе 0.

Метод *update* получает на вход id, по которому будет обновлена запись, и объект *picture*, представленный в JSON-формате, который содержит данные для обновления указанной записи. По этим данным метод *createUpdateQuery* создает часть строки sql запроса, который содержит данные из частично заполненного объекта *picture*. В случае успешного обновления записи возвращает 1, иначе 0.

Метод *delete* получает на вход id, по которому будет удалена запись. В случае успешного удаления записи возвращает 1, иначе 0.

Исходный код классов *PictureResource.java* и *PostgreSQLDAO.java* представлен в листингах 5.1-5.2.

Листинг 5.1 – Файл PictureResource.java

```
package com.maxart.service;

import java.sql.SQLException;
import java.util.List;
import javax.ws.rs.*;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.UriInfo;

@Path("/pictures")
@Produces({MediaType.APPLICATION_JSON})
public class PictureResource {

    @GET
    public List<Picture> find(@Context UriInfo info) {
        String id = info.getQueryParameters().getFirst("id");
        String name = info.getQueryParameters().getFirst("name");
        String author = info.getQueryParameters().getFirst("author");
        String year = info.getQueryParameters().getFirst("year");
        String material = info.getQueryParameters().getFirst("material");
        String height = info.getQueryParameters().getFirst("height");
        String width = info.getQueryParameters().getFirst("width");
        return new PostgreSQLDAO().findPictures(id, name, author, year,
material, height, width);
    }

    @GET
    @Path("/{id}")
    public List<Picture> getOne(@PathParam("id") int id) {
        return new PostgreSQLDAO().findOne(id);
    }

    @POST @Consumes("application/json")
    public String create(Picture picture) {
        PostgreSQLDAO dao = new PostgreSQLDAO();
        return "{\"result\":" + dao.createPicture(picture) + "}";
    }

    @PUT @Consumes("application/json")
    @Path("/{id}")
    public String update(@PathParam("id") int id, Picture picture) {
        PostgreSQLDAO dao = new PostgreSQLDAO();
        return "{\"result\":" + dao.updatePicture(id, picture) + "}";
    }

    @DELETE
    @Path("/{id}")
    public String delete(@PathParam("id") int id) {
        PostgreSQLDAO dao = new PostgreSQLDAO();
        return "{\"result\":" + dao.deletePicture(id) + "}";
    }
}
```

Листинг 5.2 – Файл PostgreSQLDAO.java

```
package com.maxart.service;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class PostgreSQLDAO {

    private Connection connection;

    PostgreSQLDAO() {
        this.connection = ConnectionUtil.getConnection();
    }

    public List<Picture> findPictures(String id, String name, String author,
String year, String material, String height, String width) {
        StringBuilder sb = new StringBuilder("");
        StringBuilder query = new StringBuilder("");
        boolean where = false;
        if (id != null) {
            sb.append("id = ").append(Integer.parseInt(id)).append(" AND ");
            where = true;
        }

        if (name != null) {
            sb.append("name = '").append(name).append("' AND ");
            where = true;
        }

        if (author != null) {
            sb.append("author = '").append(author).append("' AND ");
            where = true;
        }

        if (year != null) {
            sb.append("year = ").append(Integer.parseInt(year)).append(" AND
");
            where = true;
        }

        if (material != null) {
            sb.append("material = '").append(material).append("' AND ");
            where = true;
        }

        if (height != null) {
            sb.append("height = ").append(Float.parseFloat(height)).append("
AND ");
            where = true;
        }

        if (width != null) {
            sb.append("width = ").append(Float.parseFloat(width)).append("
AND ");
            where = true;
        }

        if (where) {
            if (sb.toString().endsWith(" AND ")) {
                sb.setLength(sb.length() - 5);
```

```
            }

            query.append("SELECT * FROM pictures WHERE
").append(sb.toString());
        } else {
            query.append("SELECT * FROM pictures");
        }

        return executeQuery(query.toString());
    }

    public List<Picture> findOne(int id) {
        String query = "SELECT * FROM pictures WHERE id = " + id;
        List<Picture> pictures = executeQuery(query);
        return pictures;
    }

    public int createPicture(Picture picture) {
        String sql = "INSERT INTO pictures (name, author, year, material,
height, width) VALUES(?, ?, ?, ?, ?, ?)";
        PreparedStatement preparedStatement = null;
        int id = 0;
        try {
            preparedStatement = this.connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
            preparedStatement.setString(1, picture.getName());
            preparedStatement.setString(2, picture.getAuthor());
            preparedStatement.setInt(3, picture.getYear());
            preparedStatement.setString(4, picture.getMaterial());
            preparedStatement.setFloat(5, picture.getHeight());
            preparedStatement.setFloat(6, picture.getWidth());

            int affectedRows = preparedStatement.executeUpdate();
            ResultSet generatedKeys = preparedStatement.getGeneratedKeys();
            if (generatedKeys.next()) {
                id = (int) generatedKeys.getLong(1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return id;
    }

    public int updatePicture(int id, Picture picture) {
        String sql = "UPDATE pictures SET" + createUpdateQuery(picture) + "
WHERE id=?";
        PreparedStatement preparedStatement = null;
        int affectedRows = 0;
        try {
            preparedStatement = this.connection.prepareStatement(sql);
            preparedStatement.setInt(1, id);
            affectedRows = preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return affectedRows;
    }

    public int deletePicture(int id) {
        String sql = "DELETE FROM pictures WHERE id = ?";

        PreparedStatement preparedStatement = null;
```

```
        int affectedRows = 0;
        try {
            preparedStatement = this.connection.prepareStatement(sql);
            preparedStatement.setInt(1, id);
            affectedRows = preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return affectedRows;
    }

    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;
    }

    private String createUpdateQuery(Picture picture) {
        StringBuilder stringBuilderField = new StringBuilder("(");
        StringBuilder stringBuilderValues = new StringBuilder("(");
        if (picture.getId() > 0) {
            stringBuilderField.append("id,");
            stringBuilderValues.append(picture.getId()).append(",");
        }

        if (picture.getName() != null) {
            stringBuilderField.append("name,");

stringBuilderValues.append("'").append(picture.getName()).append("',");
        }

        if (picture.getAuthor() != null) {
            stringBuilderField.append("author,");

stringBuilderValues.append("'").append(picture.getAuthor()).append("',");
        }

        if (picture.getYear() > 0) {
            stringBuilderField.append("year,");
            stringBuilderValues.append(picture.getYear()).append(",");
        }

        if (picture.getMaterial() != null) {
            stringBuilderField.append("material,");
```

```
stringBuilderValues.append("'").append(picture.getMaterial()).append("',");
        }

        if (picture.getHeight() > 0) {
            stringBuilderField.append("height,");
            stringBuilderValues.append(picture.getHeight()).append(",");
        }

        if (picture.getWidth() > 0) {
            stringBuilderField.append("width,");
            stringBuilderValues.append(picture.getWidth()).append(",");
        }

        if (stringBuilderField.toString().endsWith(",")) {
            stringBuilderField.setLength(stringBuilderField.length() - 1);
            stringBuilderValues.setLength(stringBuilderValues.length() - 1);
        }

        stringBuilderField.append(")");
        stringBuilderValues.append(")");

        return stringBuilderField.toString() + " = " +
stringBuilderValues.toString();
    }
}
```

Код клиента содержит файлы *Picture.java*, полученный в результате выполнения предыдущей лабораторной работы. В *App.java*, исходный код которого представлен в листинге 5.3, были добавлены новые методы:

- метод *sendRequest* выполняет POST, PUT или DELETE запрос, принимает на вход *client*, *url*, *method* и *json* – данные тела запроса, представленные в JSON-формате;
- метод *status* выполняет GET запрос */pictures*, который печатает список всех картин.

В классе *App.java* последовательно выполняются запросы:

- метод GET */pictures?author=Леонардо да Винчи*;
- метод GET */pictures?author=Леонардо да Винчи&year=1495*;
- метод GET */pictures?id=7*;
- метод GET */pictures/7*;
- метод POST */pictures*, данные в JSON-формате: name=Богатыри, author=Виктор Михайлович Васнецов, year=1881, material=Маслянные краски, height=295.3, width=446;

- метод POST */pictures*, данные в JSON-формате: name=picture, author=author, year=2018, material=Акварель, height=30, width=40;
- метод PUT */pictures/11*, данные в JSON-формате: name=My own picture, author=ITMO, year=2018;
- метод PUT */pictures/22*, данные в JSON-формате: name=My own picture, author=ITMO, year=2018;
- метод DELETE */pictures/11*.

Результат выполнения приведен на рисунке 5.1.

Листинг 5.3 – App.java

```
package com.maxart.client;

import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.GenericType;
import com.sun.jersey.api.client.WebResource;

import javax.ws.rs.core.MediaType;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

public class App {
    private static final String URL = "http://localhost:8080/pictures";

    public static void main(String[] args) {
        Client client = Client.create();
        System.out.println("Simple hard code client for service");
        status(client);

        System.out.println("Query: /pictures?author=Леонардо да Винчи,
\nMethod: GET");
        display(findPictures(client, URL, "author=Леонардо да Винчи"));
        System.out.println();

        System.out.println("Query: /pictures?author=Леонардо да
Винчи&year=1495, \nMethod: GET");
        display(findPictures(client, URL, "author=Леонардо да
Винчи&year=1495"));
        System.out.println();

        System.out.println("Query: /pictures?id=7, \nMethod: GET");
        display(findPictures(client, URL, "id=7"));
        System.out.println();

        System.out.println("Query: /pictures/7, \nMethod: GET");
        display(findPictures(client, URL + "/7", ""));
        System.out.println();

        System.out.println("Query: /pictures, \nMethod: POST, \nData:
name=Богатыри, author=Виктор Михайлович Васнецов, " +
                "year=1881, material=Маслянные краски, height=295.3,
width=446");
```

```
        String json = "{\"name\":\"Богатыри\"," +
                "\"author\":\"Виктор Михайлович Васнецов\"," +
                "\"year\":1881, " +
                "\"material\":\"Маслянные краски\"," +
                "\"height\":295.3, " +
                "\"width\":446}";

        System.out.println("Result: " + sendRequest(client, URL, "POST",
json));
        System.out.println();
        status(client);

        System.out.println("Query: /pictures, \nMethod: POST, \nData:
name=picture, author=author, " +
                "year=2018, material=Акварель, height=30, width=40");
        json = "{\"name\":\"picture\"," +
                "\"author\":\"author\"," +
                "\"year\":2018, " +
                "\"material\":\"Акварель\"," +
                "\"height\":30, " +
                "\"width\":40}";
        System.out.println("Result: " + sendRequest(client, URL, "POST",
json));
        System.out.println();
        status(client);

        System.out.println("Query: /pictures/11, \nMethod: PUT, \nData:
name=My own picture, author=ITMO, year=2018");
        json = "{\"name\":\"My own picture\"," +
                "\"author\":\"ITMO\"," +
                "\"year\":2018}";
        System.out.println("Result: " + sendRequest(client, URL + "/11",
"PUT", json));
        System.out.println();
        status(client);

        System.out.println("Query: /pictures/22, \nMethod: PUT, \nData:
name=My own picture, author=ITMO, year=2018");
        System.out.println("Result: " + sendRequest(client, URL + "/22",
"PUT", json));
        System.out.println();
        status(client);

        System.out.println("Query: /pictures/11, \nMethod: DELETE");
        System.out.println("Result: " + sendRequest(client, URL + "/11",
"DELETE", ""));
        System.out.println();
        status(client);
    }

    private static void status(Client client) {
        System.out.println("Query: /pictures, \nMethod: GET");
        display(findPictures(client, URL, ""));
        System.out.println();
    }

    private static String sendRequest(Client client, String url, String
method, String json) {
        WebResource webResource = client.resource(url);

        ClientResponse response = null;
        if (method.equals("POST"))
            response =
webResource.type(MediaType.APPLICATION_JSON).post(ClientResponse.class,
```

```
json);
        if (method.equals("PUT"))
            response =
webResource.type(MediaType.APPLICATION_JSON).put(ClientResponse.class, json);
        if (method.equals("DELETE"))
            response =
webResource.type(MediaType.APPLICATION_JSON).delete(ClientResponse.class);

        if (response != null) {
            if (response.getStatus() !=
ClientResponse.Status.OK.getStatusCode()) {
                throw new IllegalStateException("Request failed");
            }

            return response.getEntity(String.class);
        }

        return "Please specify method type (POST, PUT, DELETE)";
    }

    private static List<Picture> findPictures(Client client, String url,
String query) {
        WebResource webResource = client.resource(url);
        if (!query.isEmpty()) {
            Map<String, String> map = getQueryMap(query);

            Set<String> keys = map.keySet();
            for (String key : keys) {
                webResource = webResource.queryParam(key, map.get(key));
            }
        }

        ClientResponse response =
webResource.accept(MediaType.APPLICATION_JSON).get(ClientResponse.class);
        if (response.getStatus() != ClientResponse.Status.OK.getStatusCode())
{
            throw new IllegalStateException("Request failed");
        }

        GenericType<List<Picture>> type = new GenericType<List<Picture>>() {
        };

        return response.getEntity(type);
    }

    private static Map<String, String> getQueryMap(String query) {
        String[] params = query.split("&");
        Map<String, String> map = new HashMap<String, String>();
        for (String param : params) {
            String name = param.split("=")[0];
            String value = param.split("=")[1];
            map.put(name, value);
        }
        return map;
    }

    private static void display(List<Picture> pictures) {
        for (Picture picture : pictures) {
            System.out.println(picture);
        }
    }
}
```

```
Simple hard code client for service
Query: /pictures,
Method: GET
Picture{id=1, name='Мона Лиза', author='Леонардо да Винчи', year=1503, material='Маслянные краски', height=77.0, width=53.0}
Picture{id=2, name='Звездная ночь', author='Винсент Ван Гог', year=1889, material='Маслянные краски', height=73.7, width=92.1}
Picture{id=3, name='Тайная вечеря', author='Леонардо да Винчи', year=1495, material='Темпера', height=460.0, width=880.0}
Picture{id=4, name='Черный квадрат', author='Казимир Малевич', year=1915, material='Маслянные краски', height=80.0, width=80.0}
Picture{id=5, name='Рождение Венеры', author='Сандро Боттичелли', year=1484, material='Темпера', height=172.5, width=278.5}
Picture{id=6, name='Утро в сосновом лесу', author='Иван Иванович Шишкин', year=1889, material='Маслянные краски', height=139.0, width=213.0}
Picture{id=7, name='Постоянство памяти', author='Сальвадор Дали', year=1931, material='Гобелен ручной работы', height=24.0, width=33.0}
Picture{id=8, name='Девочка на шаре', author='Пабло Пикассо', year=1905, material='Маслянные краски', height=147.0, width=95.0}
Picture{id=9, name='Девятый вал', author='Иван Константинович Айвазовский', year=1850, material='Маслянные краски', height=221.0, width=332.0}

Query: /pictures?author=Леонардо да Винчи,
Method: GET
Picture{id=1, name='Мона Лиза', author='Леонардо да Винчи', year=1503, material='Маслянные краски', height=77.0, width=53.0}
Picture{id=3, name='Тайная вечеря', author='Леонардо да Винчи', year=1495, material='Темпера', height=460.0, width=880.0}

Query: /pictures?author=Леонардо да Винчи&year=1495,
Method: GET
Picture{id=3, name='Тайная вечеря', author='Леонардо да Винчи', year=1495, material='Темпера', height=460.0, width=880.0}

Query: /pictures?id=7,
Method: GET
Picture{id=7, name='Постоянство памяти', author='Сальвадор Дали', year=1931, material='Гобелен ручной работы', height=24.0, width=33.0}

Query: /pictures/7,
Method: GET
Picture{id=7, name='Постоянство памяти', author='Сальвадор Дали', year=1931, material='Гобелен ручной работы', height=24.0, width=33.0}
```

Рисунок 5.1, лист 1 – Результат выполнения клиентского консольного приложения

```
Query: /pictures,
Method: POST,
Data: name=Богатыри, author=Виктор Михайлович Васнецов, year=1881, material=Маслянные краски, height=295.3, width=446
Result: {"result":10}

Query: /pictures,
Method: GET
Picture{id=1, name='Мона Лиза', author='Леонардо да Винчи', year=1503, material='Маслянные краски', height=77.0, width=53.0}
Picture{id=2, name='Звездная ночь', author='Винсент Ван Гог', year=1889, material='Маслянные краски', height=73.7, width=92.1}
Picture{id=3, name='Тайная вечеря', author='Леонардо да Винчи', year=1495, material='Темпера', height=460.0, width=880.0}
Picture{id=4, name='Черный квадрат', author='Казимир Малевич', year=1915, material='Маслянные краски', height=80.0, width=80.0}
Picture{id=5, name='Рождение Венеры', author='Сандро Боттичелли', year=1484, material='Темпера', height=172.5, width=278.5}
Picture{id=6, name='Утро в сосновом лесу', author='Иван Иванович Шишкин', year=1889, material='Маслянные краски', height=139.0, width=213.0}
Picture{id=7, name='Постоянство памяти', author='Сальвадор Дали', year=1931, material='Гобелен ручной работы', height=24.0, width=33.0}
Picture{id=8, name='Девочка на шаре', author='Пабло Пикассо', year=1905, material='Маслянные краски', height=147.0, width=95.0}
Picture{id=9, name='Девятый вал', author='Иван Константинович Айвазовский', year=1850, material='Маслянные краски', height=221.0, width=332.0}
Picture{id=10, name='Богатыри', author='Виктор Михайлович Васнецов', year=1881, material='Маслянные краски', height=295.3, width=446.0}

Query: /pictures,
Method: POST,
Data: name=picture, author=author, year=2018, material=Акварель, height=30, width=40
Result: {"result":11}

Query: /pictures,
Method: GET
Picture{id=1, name='Мона Лиза', author='Леонардо да Винчи', year=1503, material='Маслянные краски', height=77.0, width=53.0}
Picture{id=2, name='Звездная ночь', author='Винсент Ван Гог', year=1889, material='Маслянные краски', height=73.7, width=92.1}
Picture{id=3, name='Тайная вечеря', author='Леонардо да Винчи', year=1495, material='Темпера', height=460.0, width=880.0}
Picture{id=4, name='Черный квадрат', author='Казимир Малевич', year=1915, material='Маслянные краски', height=80.0, width=80.0}
Picture{id=5, name='Рождение Венеры', author='Сандро Боттичелли', year=1484, material='Темпера', height=172.5, width=278.5}
Picture{id=6, name='Утро в сосновом лесу', author='Иван Иванович Шишкин', year=1889, material='Маслянные краски', height=139.0, width=213.0}
Picture{id=7, name='Постоянство памяти', author='Сальвадор Дали', year=1931, material='Гобелен ручной работы', height=24.0, width=33.0}
Picture{id=8, name='Девочка на шаре', author='Пабло Пикассо', year=1905, material='Маслянные краски', height=147.0, width=95.0}
Picture{id=9, name='Девятый вал', author='Иван Константинович Айвазовский', year=1850, material='Маслянные краски', height=221.0, width=332.0}
Picture{id=10, name='Богатыри', author='Виктор Михайлович Васнецов', year=1881, material='Маслянные краски', height=295.3, width=446.0}
Picture{id=11, name='picture', author='author', year=2018, material='Акварель', height=30.0, width=40.0}
```

Рисунок 5.1, лист 2 – Результат выполнения клиентского консольного приложения

```
Query: /pictures/11,
Method: PUT,
Data: name=My own picture, author=ITMO, year=2018
Result: {"result":1}

Query: /pictures,
Method: GET
Picture{id=1, name='Мона Лиза', author='Леонардо да Винчи', year=1503, material='Маслянные краски', height=77.0, width=53.0}
Picture{id=2, name='Звездная ночь', author='Винсент Ван Гог', year=1889, material='Маслянные краски', height=73.7, width=92.1}
Picture{id=3, name='Тайная вечеря', author='Леонардо да Винчи', year=1495, material='Темпера', height=460.0, width=880.0}
Picture{id=4, name='Черный квадрат', author='Казимир Малевич', year=1915, material='Маслянные краски', height=80.0, width=80.0}
Picture{id=5, name='Рождение Венеры', author='Сандро Боттичелли', year=1484, material='Темпера', height=172.5, width=278.5}
Picture{id=6, name='Утро в сосновом лесу', author='Иван Иванович Шишкин', year=1889, material='Маслянные краски', height=139.0, width=213.0}
Picture{id=7, name='Постоянство памяти', author='Сальвадор Дали', year=1931, material='Гобелен ручной работы', height=24.0, width=33.0}
Picture{id=8, name='Девочка на шаре', author='Пабло Пикассо', year=1905, material='Маслянные краски', height=147.0, width=95.0}
Picture{id=9, name='Девятый вал', author='Иван Константинович Айвазовский', year=1850, material='Маслянные краски', height=221.0, width=332.0}
Picture{id=10, name='Богатыри', author='Виктор Михайлович Васнецов', year=1881, material='Маслянные краски', height=295.3, width=446.0}
Picture{id=11, name='My own picture', author='ITMO', year=2018, material='Акварель', height=30.0, width=40.0}

Query: /pictures/22,
Method: PUT,
Data: name=My own picture, author=ITMO, year=2018
Result: {"result":0}

Query: /pictures,
Method: GET
Picture{id=1, name='Мона Лиза', author='Леонардо да Винчи', year=1503, material='Маслянные краски', height=77.0, width=53.0}
Picture{id=2, name='Звездная ночь', author='Винсент Ван Гог', year=1889, material='Маслянные краски', height=73.7, width=92.1}
Picture{id=3, name='Тайная вечеря', author='Леонардо да Винчи', year=1495, material='Темпера', height=460.0, width=880.0}
Picture{id=4, name='Черный квадрат', author='Казимир Малевич', year=1915, material='Маслянные краски', height=80.0, width=80.0}
Picture{id=5, name='Рождение Венеры', author='Сандро Боттичелли', year=1484, material='Темпера', height=172.5, width=278.5}
Picture{id=6, name='Утро в сосновом лесу', author='Иван Иванович Шишкин', year=1889, material='Маслянные краски', height=139.0, width=213.0}
Picture{id=7, name='Постоянство памяти', author='Сальвадор Дали', year=1931, material='Гобелен ручной работы', height=24.0, width=33.0}
Picture{id=8, name='Девочка на шаре', author='Пабло Пикассо', year=1905, material='Маслянные краски', height=147.0, width=95.0}
Picture{id=9, name='Девятый вал', author='Иван Константинович Айвазовский', year=1850, material='Маслянные краски', height=221.0, width=332.0}
Picture{id=10, name='Богатыри', author='Виктор Михайлович Васнецов', year=1881, material='Маслянные краски', height=295.3, width=446.0}
Picture{id=11, name='My own picture', author='ITMO', year=2018, material='Акварель', height=30.0, width=40.0}
```

Рисунок 5.1, лист 3 – Результат выполнения клиентского консольного приложения

```
Query: /pictures/11,
Method: DELETE
Result: {"result":1}

Query: /pictures,
Method: GET
Picture{id=1, name='Мона Лиза', author='Леонардо да Винчи', year=1503, material='Маслянные краски', height=77.0, width=53.0}
Picture{id=2, name='Звездная ночь', author='Винсент Ван Гог', year=1889, material='Маслянные краски', height=73.7, width=92.1}
Picture{id=3, name='Тайная вечеря', author='Леонардо да Винчи', year=1495, material='Темпера', height=460.0, width=880.0}
Picture{id=4, name='Черный квадрат', author='Казимир Малевич', year=1915, material='Маслянные краски', height=80.0, width=80.0}
Picture{id=5, name='Рождение Венеры', author='Сандро Боттичелли', year=1484, material='Темпера', height=172.5, width=278.5}
Picture{id=6, name='Утро в сосновом лесу', author='Иван Иванович Шишкин', year=1889, material='Маслянные краски', height=139.0, width=213.0}
Picture{id=7, name='Постоянство памяти', author='Сальвадор Дали', year=1931, material='Гобелен ручной работы', height=24.0, width=33.0}
Picture{id=8, name='Девочка на шаре', author='Пабло Пикассо', year=1905, material='Маслянные краски', height=147.0, width=95.0}
Picture{id=9, name='Девятый вал', author='Иван Константинович Айвазовский', year=1850, material='Маслянные краски', height=221.0, width=332.0}
Picture{id=10, name='Богатыри', author='Виктор Михайлович Васнецов', year=1881, material='Маслянные краски', height=295.3, width=446.0}
```

Рисунок 5.1, лист 4 – Результат выполнения клиентского консольного приложения

Вывод: в ходе выполнения работы был реализован CRUD с помощью REST-сервиса в виде standalone-приложения. Для демонстрации работы разработанного сервиса было разработано клиентское консольное приложение.