2Министерство науки и высшего образования Российской Федерации

Муромский институт (филиал)

Федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

ЛАБОРАТОРНАЯ

РАБОТА №6

По Разработка корпоративных приложений

Тема Получение JSON в приложениях

Руководитель

Кульков Я.Ю.

(фамилия, инициалы)

(подпись) (дата)

Студент ПИН - 121

(группа)

Ермилов М.В.

(фамилия, инициалы)

(подпись) (дата)

Муром 2024

Лабораторная работа №6

Тема: получение JSON в приложениях.

Цели и задачи: изучить способы работы с API из сторонних приложений.

Ход работы: задание: подготовить JavaFX-приложение (в соответствии с темой курсовой работы), работающее с данными сервера по API.

Листинг кода 1 – EmployerController.java

package ru.kafpin.lr6\_bzz.controllers;

import javafx.beans.property.SimpleObjectProperty;

import javafx.beans.property.SimpleStringProperty;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.stage.Modality;

import javafx.stage.Stage;

import lombok.Setter;

import ru.kafpin.lr6\_bzz.MainApplication;

import ru.kafpin.lr6\_bzz.dao.\*;

import ru.kafpin.lr6\_bzz.domains.\*;

import javafx.scene.control.ComboBox;

import java.io.IOException;

import java.time.LocalDate;

import java.util.\*;

public class EmployerController {

@Setter

private Stage employerStage;

private final AutomobileDao automobileDao;

private final ClientDao clientDao;

private final EmployerDao employerDao;

private final ProvidedServiceDao providedServiceDao;

private final ServiceDao serviceDao;

private final ObservableList<ProvidedService> providedServices = FXCollections.observableArrayList();

private final ObservableList<Client> clients = FXCollections.observableArrayList();

private final ObservableList<Automobile> automobiles = FXCollections.observableArrayList();

private final ObservableList<Service> services = FXCollections.observableArrayList();

private final ObservableList<Employer> employers = FXCollections.observableArrayList();

private final ResourceBundle bundle = ResourceBundle.getBundle("employer", Locale.getDefault());

@FXML

private TableView<ProvidedService> tvProvidedServices;

@FXML

private TableColumn<ProvidedService, String> tcName;

@FXML

private TableColumn<ProvidedService, String> tcClientFIO;

@FXML

private TableColumn<ProvidedService, String> tcAutomobile;

@FXML

private TableColumn<ProvidedService, String> tcEmployerFIO;

@FXML

private TableColumn<ProvidedService, Integer> tcPrice;

@FXML

private TableColumn<ProvidedService, Date> tcDatetime;

@FXML

private TableView<Client> tvClients;

@FXML

private TableColumn<Client, String> tcFIO;

@FXML

private TableColumn<Client, String> tcPhone;

@FXML

private TableView<Automobile> tvAutomobiles;

@FXML

private TableColumn<Automobile, String> tcMark;

@FXML

private TableColumn<Automobile, String> tcModel;

@FXML

private TableColumn<Automobile, String> tcGosnumber;

@FXML

private DatePicker dpFrom;

@FXML

private DatePicker dpTo;

@FXML

private ComboBox<Client> cbClients;

public EmployerController(){

try {

automobileDao = new AutomobileDao();

clientDao = new ClientDao();

employerDao = new EmployerDao();

providedServiceDao = new ProvidedServiceDao();

serviceDao = new ServiceDao();

services.addAll(serviceDao.findALl());

employers.addAll(employerDao.findALl());

} catch (Exception e) {

throw new RuntimeException(e);

}

}

LocalDate localDatefrom = LocalDate.of(2020,1,1);

LocalDate localDateto = LocalDate.now();

@FXML

void initialize() {

dpFrom.setValue(localDatefrom);

dpTo.setValue(localDateto);

providedServices.addAll(providedServiceDao.findALlFromTo(localDatefrom,localDateto));

tcName.setCellValueFactory(s -> new SimpleStringProperty(s.getValue().getService().getName()));

tcClientFIO.setCellValueFactory(s -> new SimpleStringProperty(

automobileDao.findById(s.getValue().getAutomobile().getId()).getClient().toString()));

tcAutomobile.setCellValueFactory(s -> new SimpleStringProperty(s.getValue().getAutomobile().toString()));

tcEmployerFIO.setCellValueFactory(s -> new SimpleStringProperty(s.getValue().getEmployer().toString()));

tcPrice.setCellValueFactory(s -> new SimpleObjectProperty<Integer>(s.getValue().getService().getPrice()));

tcDatetime.setCellValueFactory(s -> new SimpleObjectProperty<Date>(s.getValue().getSqlDate()));

tvProvidedServices.setItems(providedServices);

tvProvidedServices.getSortOrder().add(tcDatetime);

clients.addAll(clientDao.findALl());

tcFIO.setCellValueFactory(s -> new SimpleStringProperty(

s.getValue().toString()));

tcPhone.setCellValueFactory(s -> new SimpleStringProperty(s.getValue().getPhone()));

tvClients.setItems(clients);

tvClients.getSortOrder().add(tcFIO);

tvClients.getSortOrder().add(tcPhone);

cbClients.setItems(clients);

}

@FXML

void onDataChanged(ActionEvent event) {

localDatefrom = dpFrom.getValue();

localDateto = dpTo.getValue();

providedServices.clear();

providedServices.addAll(providedServiceDao.findALlFromTo(localDatefrom,localDateto));

tvProvidedServices.setItems(providedServices);

tvProvidedServices.sort();

}

private void Error(String text){

Alert alert;

alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle(bundle.getString("error"));

alert.setContentText(null);

alert.setHeaderText(text);

alert.showAndWait();

}

@FXML

void onAddAutomobile(ActionEvent event) {

if(cbClients.getSelectionModel().getSelectedItem()!=null){

Automobile automobile = new Automobile();

automobile.setClient(cbClients.getSelectionModel().getSelectedItem());

if(showAutomobileDialog(automobile)){

Automobile existsAutomobile = automobileDao.findByGosnumber(automobile.getGosnumber());

if(existsAutomobile!=null){

Error("Невозможно создать автомобиль с указанным гос.номером, т.к. данный гос.номер присвоен другому автомобилю");

}

else{

automobileDao.save(automobile);

automobiles.clear();

automobiles.addAll(automobileDao.findALlCarsOfOwner(cbClients.getSelectionModel().getSelectedItem().getId()));

tvAutomobiles.sort();

}

}

}

else

Error(bundle.getString("clientnotchoicedforautoadd"));

}

@FXML

void onAddClient(ActionEvent event) {

Client client = new Client();

if(showClientDialog(client)){

Client existsClient = clientDao.findByPhone(client.getPhone());

if(existsClient!=null){

Error("Невозможно создать клиента с указанным номером телефона, т.к. номер занят другим клиентом");

}

else{

clientDao.save(client);

clients.clear();

clients.addAll(clientDao.findALl());

tvClients.sort();

}

}

}

@FXML

void onAddProvidedService(ActionEvent event) {

ProvidedService providedService = new ProvidedService();

if(showProvidedServiceDialog(providedService,false)){

providedServiceDao.save(providedService);

providedServices.clear();

providedServices.addAll(providedServiceDao.findALlFromTo(localDatefrom,localDateto));

tvProvidedServices.sort();

}

}

@FXML

void onEditAutomobile(ActionEvent event) {

if(cbClients.getSelectionModel().getSelectedItem()!=null){

Automobile automobile = tvAutomobiles.getSelectionModel().getSelectedItem();

if (automobile != null){

if(showAutomobileDialog(automobile)){

Automobile existsAutomobile = automobileDao.findByGosnumber(automobile.getGosnumber());

if(existsAutomobile!=null&& !Objects.equals(existsAutomobile.getId(), automobile.getId())){

Error("Невозможно изменить гос.номер выбранного автомобиля, т.к. введённый гос.номер присвоен другому автомобилю");

}

else{

automobileDao.update(automobile);

}

automobiles.clear();

automobiles.addAll(automobileDao.findALlCarsOfOwner(cbClients.getSelectionModel().getSelectedItem().getId()));

tvAutomobiles.sort();

}

}

else

Error(bundle.getString("autonotchoicedforautoupd"));

}

else

Error(bundle.getString("clientnotchoicedforautoupd"));

}

@FXML

void onEditClient(ActionEvent event) {

Client client = tvClients.getSelectionModel().getSelectedItem();

if (client != null){

if(showClientDialog(client)){

Client existsClient = clientDao.findByPhone(client.getPhone());

if(existsClient!=null&& !Objects.equals(existsClient.getId(), client.getId())){

Error("Невозможно изменить номер выбранного клиента, т.к. введённый номер занят другим клиентом");

}

else{

clientDao.update(client);

}

clients.clear();

clients.addAll(clientDao.findALl());

tvClients.sort();

}

}

else

Error(bundle.getString("clientnotchoicedup"));

}

@FXML

void onEditProvidedService(ActionEvent event) {

ProvidedService providedService = tvProvidedServices.getSelectionModel().getSelectedItem();

if (providedService != null){

if(showProvidedServiceDialog(providedService,true)){

providedServiceDao.update(providedService);

providedServices.clear();

providedServices.addAll(providedServiceDao.findALlFromTo(localDatefrom,localDateto));

tvProvidedServices.sort();

}

}

else

Error(bundle.getString("providedservicenotchoiceup"));

}

@FXML

void onRemoveAutomobile(ActionEvent event) {

if(cbClients.getSelectionModel().getSelectedItem()!=null){

Automobile automobile = tvAutomobiles.getSelectionModel().getSelectedItem();

if(automobile!=null){

if(showRemoveDialog("automobile")){

tvAutomobiles.getItems().remove(tvAutomobiles.getSelectionModel().getSelectedIndex());

automobileDao.deleteById(automobile.getId());

tvAutomobiles.sort();

providedServices.clear();

providedServices.addAll(providedServiceDao.findALlFromTo(localDatefrom,localDateto));

}

}

else

Error(bundle.getString("autonotchoicedfordel"));

}

else

Error(bundle.getString("clientnotchoicedforautodel"));

}

@FXML

void onRemoveClient(ActionEvent event) {

Client client = tvClients.getSelectionModel().getSelectedItem();

if(client!=null){

if(showRemoveDialog("client")){

tvClients.getItems().remove(tvClients.getSelectionModel().getSelectedIndex());

clientDao.deleteById(client.getId());

tvClients.sort();

providedServices.clear();

providedServices.addAll(providedServiceDao.findALlFromTo(localDatefrom,localDateto));

}

}

else

Error(bundle.getString("clientnotchoicedfordel"));

}

@FXML

void onRemoveProvidedService(ActionEvent event) {

ProvidedService providedService = tvProvidedServices.getSelectionModel().getSelectedItem();

if (providedService!=null){

if(showRemoveDialog("providedservice")){

tvProvidedServices.getItems().remove(tvProvidedServices.getSelectionModel().getSelectedIndex());

providedServiceDao.deleteById(providedService.getId());

tvProvidedServices.sort();

}

}

else

Error(bundle.getString("providedservicenotchoicedfordel"));

}

private boolean showProvidedServiceDialog(ProvidedService providedService, boolean update) {

FXMLLoader loader = new FXMLLoader(MainApplication.class.getResource("providedService-addEdit.fxml"),bundle);

try {

Parent root = loader.load();

Scene scene = new Scene(root);

Stage addStage = new Stage();

addStage.setTitle(bundle.getString("providedserviceinfo"));

addStage.setScene(scene);

addStage.initModality(Modality.APPLICATION\_MODAL);

addStage.initOwner(MainApplication.getMainStage());

EditProvidedServiceController editProvidedServiceController = loader.getController();

editProvidedServiceController.setProvidedService(providedService);

editProvidedServiceController.setEditStage(addStage);

editProvidedServiceController.setBundle(bundle);

editProvidedServiceController.setClients(clients);

editProvidedServiceController.setServices(services);

editProvidedServiceController.setEmployers(employers);

editProvidedServiceController.initialize();

if(update)

editProvidedServiceController.update();

addStage.showAndWait();

return editProvidedServiceController.isAction();

} catch (IOException e) {

System.out.println(e.getMessage());

return false;

}

}

private boolean showClientDialog(Client client) {

FXMLLoader loader = new FXMLLoader(MainApplication.class.getResource("client-addEdit.fxml"),bundle);

try {

Parent root = loader.load();

Scene scene = new Scene(root);

Stage addStage = new Stage();

addStage.setTitle(bundle.getString("clientinfo"));

addStage.setScene(scene);

addStage.initModality(Modality.APPLICATION\_MODAL);

addStage.initOwner(MainApplication.getMainStage());

EditClientController editClientController = loader.getController();

editClientController.setClient(client);

editClientController.setEditStage(addStage);

editClientController.setBundle(bundle);

addStage.showAndWait();

return editClientController.isAction();

} catch (IOException e) {

System.out.println(e.getMessage());

return false;

}

}

private boolean showAutomobileDialog(Automobile automobile) {

FXMLLoader loader = new FXMLLoader(MainApplication.class.getResource("automobile-addEdit.fxml"),bundle);

try {

Parent root = loader.load();

Scene scene = new Scene(root);

Stage addStage = new Stage();

addStage.setTitle(bundle.getString("automobileinfo"));

addStage.setScene(scene);

addStage.initModality(Modality.APPLICATION\_MODAL);

addStage.initOwner(MainApplication.getMainStage());

EditAutomobileController editAutomobileController = loader.getController();

editAutomobileController.setAutomobile(automobile);

editAutomobileController.setEditStage(addStage);

editAutomobileController.setBundle(bundle);

addStage.showAndWait();

return editAutomobileController.isAction();

} catch (IOException e) {

System.out.println(e.getMessage());

return false;

}

}

@FXML

void onClientSwitched(ActionEvent event) {

if(cbClients.getSelectionModel().getSelectedItem()!=null){

automobiles.clear();

automobiles.addAll(automobileDao.findALlCarsOfOwner(cbClients.getSelectionModel().getSelectedItem().getId()));

tcMark.setCellValueFactory(s -> new SimpleStringProperty(s.getValue().getMark()));

tcModel.setCellValueFactory(s -> new SimpleStringProperty(s.getValue().getModel()));

tcGosnumber.setCellValueFactory(s -> new SimpleStringProperty(s.getValue().getGosnumber()));

tvAutomobiles.setItems(automobiles);

tvAutomobiles.getSortOrder().add(tcMark);

tvAutomobiles.getSortOrder().add(tcModel);

tvAutomobiles.getSortOrder().add(tcGosnumber);

}

else {

automobiles.clear();

tvAutomobiles.setItems(automobiles);

}

}

private boolean showRemoveDialog(String removeditem) {

FXMLLoader loader = new FXMLLoader(MainApplication.class.getResource("removeform.fxml"),bundle);

try {

Parent root = loader.load();

Scene scene = new Scene(root);

Stage remove = new Stage();

remove.setResizable(false);

remove.setTitle(bundle.getString("remove"+removeditem));

remove.setScene(scene);

remove.initModality(Modality.APPLICATION\_MODAL);

remove.initOwner(MainApplication.getMainStage());

RemoveController removeController = loader.getController();

removeController.setRemoveStage(remove);

remove.showAndWait();

return removeController.isRemove();

} catch (IOException e) {

System.out.println(e.getMessage());

}

return false;

}

}

Листинг кода 2 – ClientDao.java

package ru.kafpin.lr6\_bzz.dao;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.core.type.TypeReference;

import com.fasterxml.jackson.databind.ObjectMapper;

import lombok.NoArgsConstructor;

import ru.kafpin.lr6\_bzz.domains.Client;

import java.io.\*;

import java.net.HttpURLConnection;

import java.net.URL;

import java.nio.charset.StandardCharsets;

import java.util.\*;

@NoArgsConstructor

public class ClientDao implements Dao<Client, Long> {

private URL url;

private final ObjectMapper mapper = new ObjectMapper();

private HttpURLConnection conn;

@Override

public Collection<Client> findALl() {

try {

url = new URL("http://127.0.0.1:8080/api/clients");

conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("GET");

conn.setRequestProperty("Accept", "application/json");

if(200 != conn.getResponseCode()){

System.out.printf("Response code = "+conn.getResponseCode());

return null;

}

}

catch (IOException e) {

System.out.println("URL/Connection error");

}

List<Client> list = null;

StringBuilder content = new StringBuilder();

try(BufferedReader bufferedReader =

new BufferedReader(

new InputStreamReader(conn.getInputStream(), StandardCharsets.UTF\_8))){

String line;

while((line = bufferedReader.readLine()) != null) {

content.append(line);

content.append("\n");

}

} catch (IOException e) {

System.out.println("bufferReader error");

}

try {

list = mapper.reader()

.forType(new TypeReference<List<Client>>() {})

.readValue(content.toString());

} catch (JsonProcessingException e) {

System.out.println(e);

System.out.println("Error of parsing");

}

return list;

}

@Override

public Client save(Client client) {

String json = parseSingleClientToJson(client);

try {

url = new URL("http://127.0.0.1:8080/api/clients");

conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("POST");

conn.setRequestProperty("Content-Type", "application/json");

conn.setRequestProperty("Accept", "application/json");

conn.setDoOutput(true);

}

catch (IOException e) {

System.out.println("URL/Connection error");

}

writeResponseToJson(json);

return parseJsonToSingleClient();

}

@Override

public Client update(Client client) {

String json = parseSingleClientToJson(client);

try {

url = new URL("http://127.0.0.1:8080/api/clients/"+client.getId());

conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("PUT");

conn.setRequestProperty("Content-Type", "application/json");

conn.setRequestProperty("Accept", "application/json");

conn.setDoOutput(true);

}

catch (IOException e) {

System.out.println("URL/Connection error");

}

writeResponseToJson(json);

return parseJsonToSingleClient();

}

@Override

public void deleteById(Long id) {

try {

url = new URL("http://127.0.0.1:8080/api/clients/"+id);

conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("DELETE");

conn.setRequestProperty("Accept", "application/json");

if(200 != conn.getResponseCode()){

System.out.printf("Response code = " + conn.getResponseCode());

}

}

catch (IOException e) {

System.out.println("URL/Connection error");

}

}

@Override

public Client findById(Long id) {

try {

url = new URL("http://127.0.0.1:8080/api/clients/"+id);

conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("GET");

conn.setRequestProperty("Accept", "application/json");

if(200 != conn.getResponseCode()){

System.out.printf("Response code = "+conn.getResponseCode());

return null;

}

}

catch (IOException e) {

System.out.println("URL/Connection error");

}

return parseJsonToSingleClient();

}

public Client findByPhone(String phone) {

try {

url = new URL("http://127.0.0.1:8080/api/clients/phone/"+phone);

conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("GET");

conn.setRequestProperty("Accept", "application/json");

if(200 != conn.getResponseCode()){

System.out.printf("Response code = "+conn.getResponseCode());

return null;

}

}

catch (IOException e) {

System.out.println("URL/Connection error");

}

return parseJsonToSingleClient();

}

private void writeResponseToJson(String json){

try(OutputStream os = conn.getOutputStream()) {

byte[] input = json.getBytes(StandardCharsets.UTF\_8);

os.write(input, 0, input.length);

} catch (IOException e) {

System.out.println("error of write outputstream");

}

}

private String parseSingleClientToJson(Client client){

String json = null;

try {

json = mapper.writeValueAsString(client);

} catch (JsonProcessingException e) {

System.out.println(e);

System.out.println("Error of write in json");

}

return json;

}

private Client parseJsonToSingleClient(){

Client client = null;

StringBuilder response = new StringBuilder();

try(BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream(), StandardCharsets.UTF\_8))){

String responseLine;

while ((responseLine = br.readLine()) != null) {

response.append(responseLine);

response.append("\n");

}

} catch (IOException e) {

System.out.println("bufferReader error");

}

System.out.println("response "+response);

try {

client = mapper.reader()

.forType(Client.class)

.readValue(response.toString());

} catch (JsonProcessingException e) {

System.out.println(e);

System.out.println("Error of parsing");

}

System.out.println("response "+client);

return client;

}

}

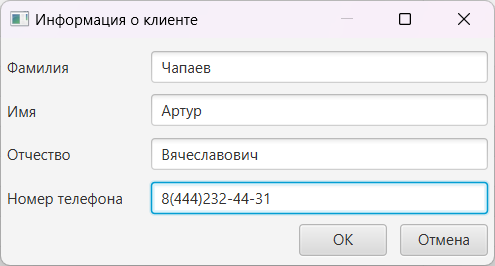


Рисунок 1 – Добавление клиента

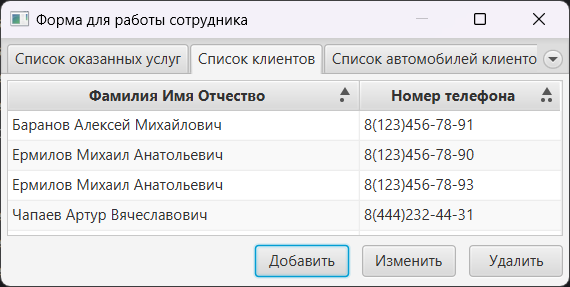


Рисунок 2 – Отображение клиента в списке JavaFX приложения

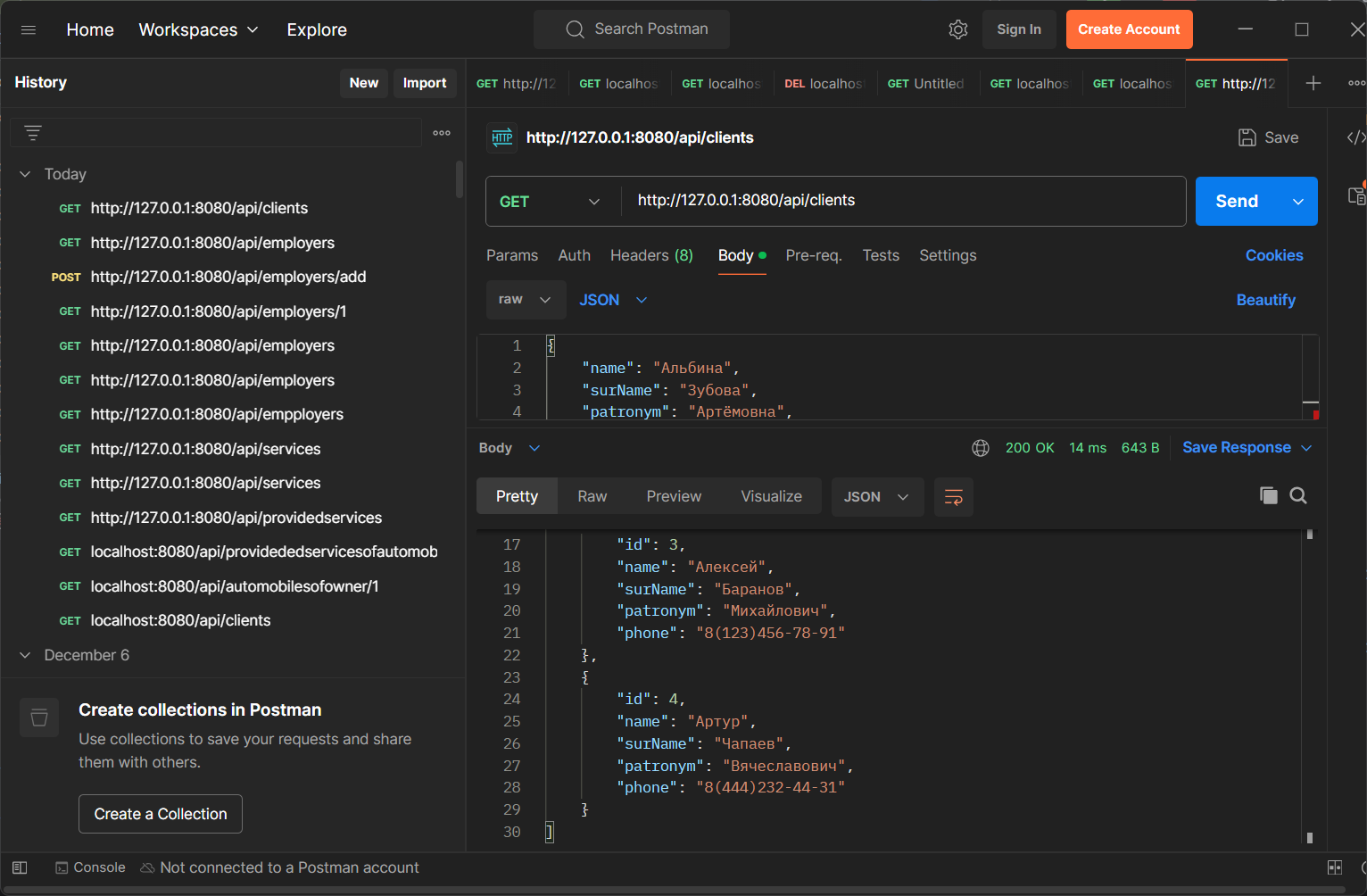


Рисунок 3 – Проверка добавления клиента в список на сервере

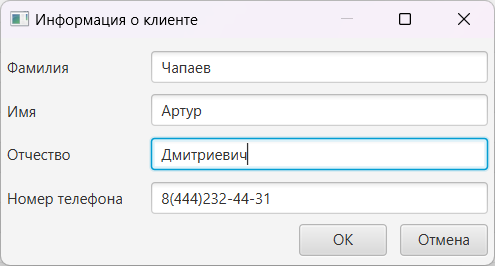


Рисунок 4 – Изменение информации о клиенте

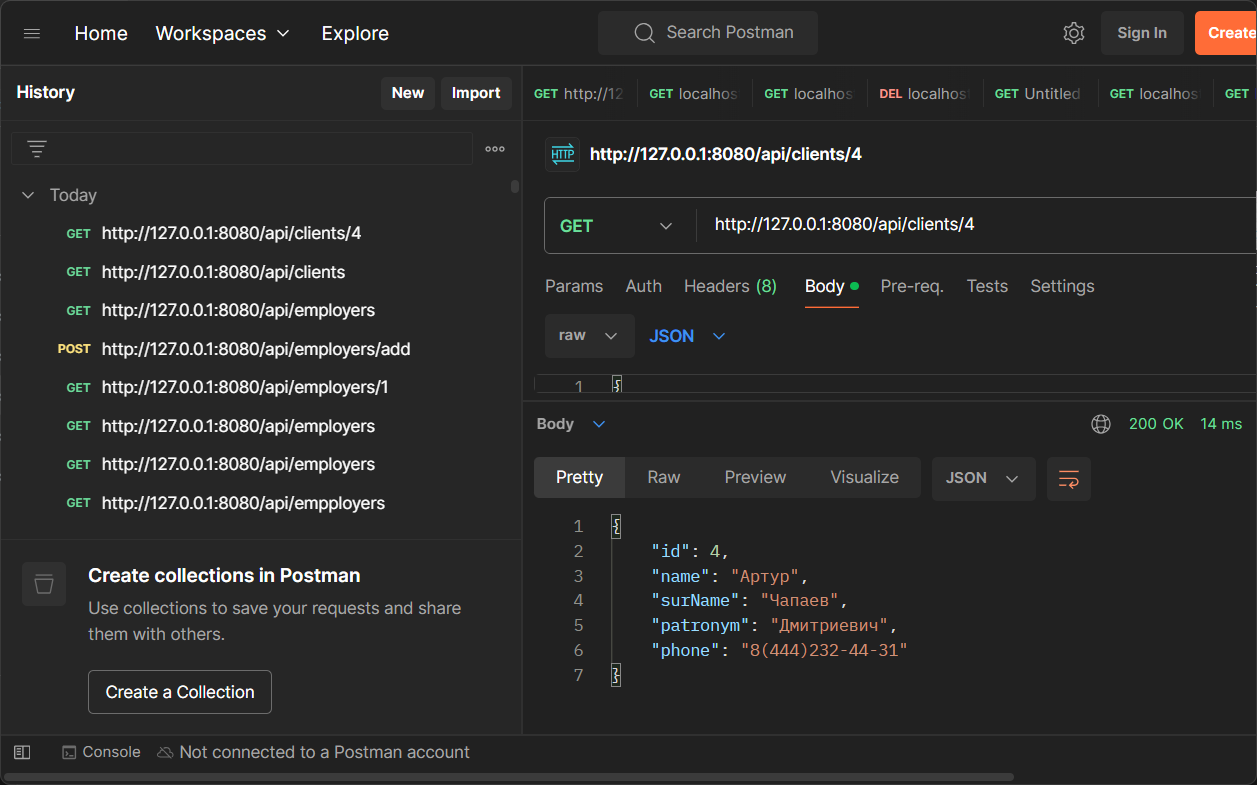


Рисунок 5 – Проверка изменения информации на сервере

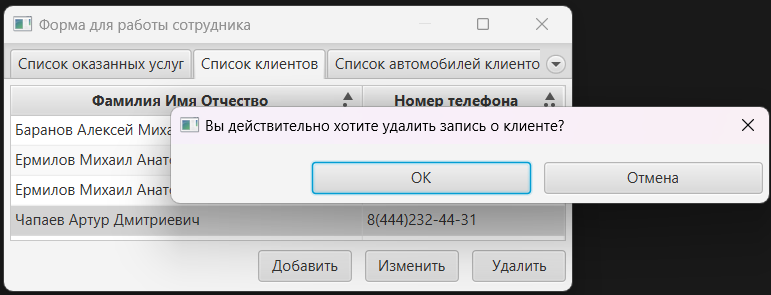


Рисунок 6 – Удаление выбранного клиента

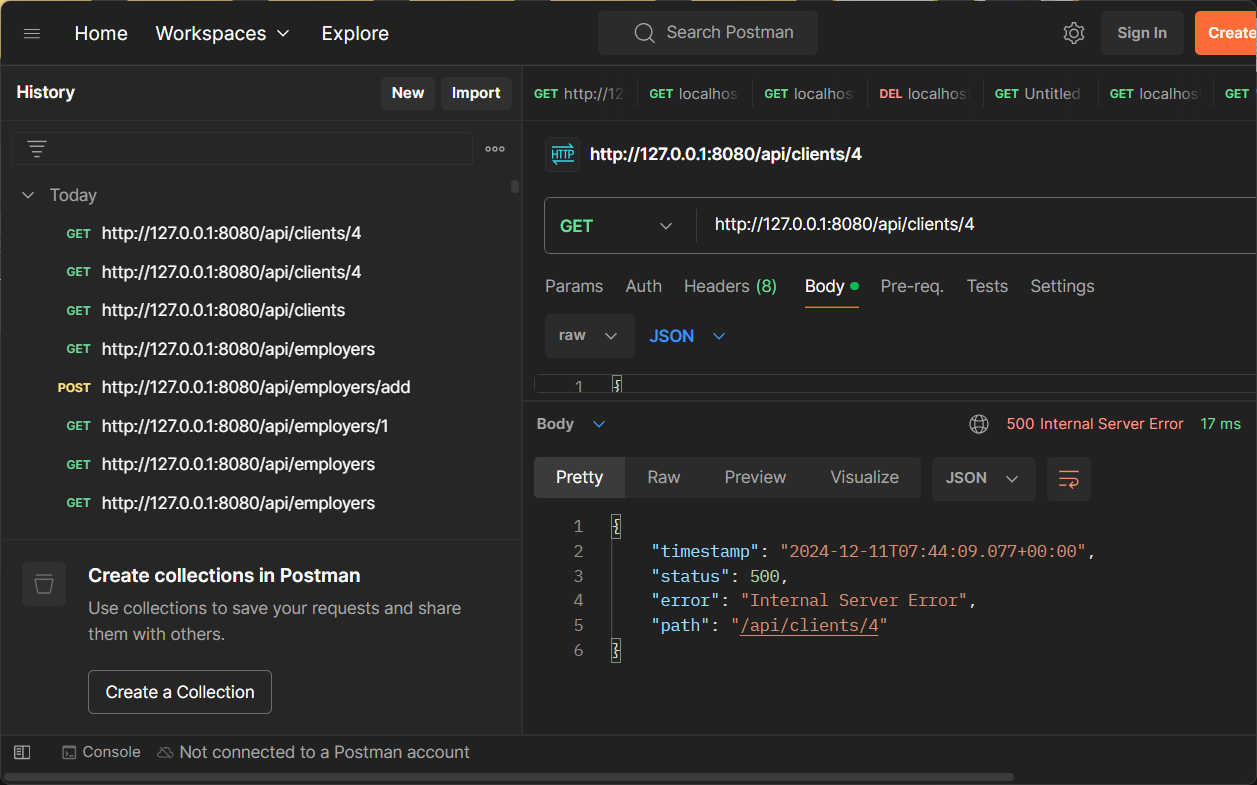


Рисунок 7 – Проверка удаления клиента на сервере

Вывод: в ходе работы подготовить JavaFX-приложение (в соответствии с темой курсовой работы), работающее с данными сервера по API.