Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

ОТЧЕТ

по дисциплине «Практикум по программированию»

**Лабораторная работа № 5**

**Выполнил:**

Cтудент гр. Group number В. О. Синцев

**Проверил**

Ст. преподаватель А.М. Журавская

Санкт-Петербург

2024 г.

## **Задание**

Вариант 7. На основе 2 или третьей лабораторных работ создать десктопное приложение. Разрешается использовать свою собственную идею для данной работы, т.е. не строго следовать наработкам из 2 и 3 лабораторных работ.

Предметная область: Фирма грузоперевозок. Фирма имеет список тарифов по перевозке грузов. Класс «тариф» хранит наименование тарифа и цену.

## **Код программы**

Class: LogisticsApp.java

import java.awt.\*;  
import java.util.List;  
import javax.swing.\*;  
  
  
public class LogisticsApp extends App{  
 public static void main(String[] args) {  
 LogisticsApp app = new LogisticsApp("Грузовичкофф");  
 app.showLoginScreen();  
 }  
  
 public LogisticsApp(String nameCompany){  
 this.nameCompany = nameCompany;  
 }  
  
 @Override  
 public void showLoginScreen() {  
 final JFrame loginFrame = new JFrame(nameCompany + " - Вход в систему");  
 loginFrame.setSize(400, 300);  
 loginFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 loginFrame.setLayout(new GridBagLayout());  
  
 // Настройка дизайнерского шрифта  
 Font customFont = new Font("Verdana", Font.*PLAIN*, 14);  
  
 JLabel userLabel = new JLabel("Имя пользователя:");  
 userLabel.setFont(customFont);  
 final JTextField usernameField = new JTextField(15);  
 usernameField.setFont(customFont);  
  
 JLabel passLabel = new JLabel("Пароль:");  
 passLabel.setFont(customFont);  
 final JPasswordField passwordField = new JPasswordField(15);  
 passwordField.setFont(customFont);  
  
 JButton loginButton = new JButton("Войти");  
 loginButton.setFont(customFont);  
 JButton registerButton = new JButton("Регистрация");  
 registerButton.setFont(customFont);  
  
 // Добавление анимации при наведении на кнопки  
 *addHoverEffect*(loginButton);  
 *addHoverEffect*(registerButton);  
  
 // логин  
 login(loginButton, usernameField, passwordField, loginFrame);  
  
 // регистрация  
 registration(registerButton, usernameField, passwordField, loginFrame);  
  
 // иконока  
 LogisticsApp.*setIconApp*("LogReg.png", loginFrame);  
  
 // мин размер  
 loginFrame.setMinimumSize(new Dimension(400, 300));  
  
 // Расположение компонентов с помощью GridBagLayout  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(10, 10, 5, 10);  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 loginFrame.add(userLabel, gbc);  
  
 gbc.gridx = 1;  
 loginFrame.add(usernameField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 1;  
 loginFrame.add(passLabel, gbc);  
  
 gbc.gridx = 1;  
 loginFrame.add(passwordField, gbc);  
  
 gbc.insets = new Insets(20, 10, 10, 10);  
 gbc.gridx = 0;  
 gbc.gridy = 2;  
 loginFrame.add(registerButton, gbc);  
  
 gbc.gridx = 1;  
 loginFrame.add(loginButton, gbc);  
  
 loginFrame.setLocationRelativeTo(null);  
 loginFrame.setVisible(true);  
 }  
  
  
 public void showMainApp() {  
 final JFrame frame = new JFrame("Грузоперевозки - Выбор тарифов");  
 frame.setSize(600, 400);  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 // Настройка дизайнерского шрифта  
 Font customFont = new Font("Verdana", Font.*PLAIN*, 14);  
  
 JPanel panel = new JPanel();  
 panel.setLayout(new GridBagLayout());  
  
 String[] tariffs = {"Эконом", "Стандарт", "Бизнес", "Премиум"};  
 String[] descriptions = {  
 "Базовый тариф для небольших грузов.",  
 "Оптимальный выбор для стандартных перевозок.",  
 "Повышенный комфорт и скорость доставки.",  
 "Максимальный сервис для особых клиентов."  
 };  
  
  
  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(10, 10, 10, 10);  
 gbc.fill = GridBagConstraints.*HORIZONTAL*;  
  
 for (int i = 0; i < tariffs.length; i++) {  
 final String tariff = tariffs[i];  
 final String description = descriptions[i];  
  
 JPanel tariffPanel = new JPanel(new BorderLayout());  
 tariffPanel.setBorder(BorderFactory.*createLineBorder*(Color.*GRAY*, 1));  
 tariffPanel.setBackground(Color.*WHITE*);  
  
 JLabel label = new JLabel("<html><b>" + tariff + "</b><br>" + description + "</html>");  
 label.setFont(customFont);  
  
 JButton addButton = new JButton("Добавить");  
 addButton.setFont(customFont);  
 *addHoverEffect*(addButton);  
  
 addButton.addActionListener(e -> {  
 cart.add(tariff);  
 JOptionPane.*showMessageDialog*(frame, tariff + " добавлен в корзину!");  
 });  
  
 tariffPanel.add(label, BorderLayout.*CENTER*);  
 tariffPanel.add(addButton, BorderLayout.*EAST*);  
  
 gbc.gridx = 0;  
 gbc.gridy = i;  
 panel.add(tariffPanel, gbc);  
 }  
  
  
  
 JButton showCartButton = new JButton("Показать корзину");  
 showCartButton.setFont(customFont);  
 *addHoverEffect*(showCartButton);  
  
 showCartButton.addActionListener(e -> showCart(frame));  
  
 JButton logoutButton = new JButton("Выйти");  
 logoutButton.setFont(customFont);  
 *addHoverEffect*(logoutButton);  
  
 logoutButton.addActionListener(e -> {  
 isLoggedIn = false;  
 currentUser = "";  
 frame.dispose();  
 showLoginScreen();  
 });  
  
 gbc.gridx = 0;  
 gbc.gridy = tariffs.length;  
 panel.add(showCartButton, gbc);  
  
 gbc.gridy = tariffs.length + 1;  
 panel.add(logoutButton, gbc);  
  
  
 JButton wish = new JButton("Добавить пожелания");  
 wish.setFont(customFont);  
 *addHoverEffect*(wish);  
 JPanel topPanel = new JPanel();  
 topPanel.add(wish);  
 topPanel.setBackground(Color.*PINK*);  
 frame.add(topPanel, BorderLayout.*NORTH*);  
  
  
 wish.addActionListener(e -> {  
 FileAppendLogger.*appendFromFile*(currentUser, frame);  
 });  
  
  
  
  
  
 panel.setBackground(Color.*LIGHT\_GRAY*);  
 frame.add(panel);  
 LogisticsApp.*setIconApp*("img.png", frame);  
 frame.setMinimumSize(new Dimension(600, 400));  
 frame.setLocationRelativeTo(null);  
 frame.setVisible(true);  
 }  
  
  
 private void showCart(JFrame parentFrame) {  
 // Создание модели для списка корзины  
 DefaultListModel<String> listModel = new DefaultListModel<>();  
 for (String item : cart) {  
 listModel.addElement(item);  
 }  
  
 // Создание диалогового окна для отображения корзины  
 JDialog cartDialog = new JDialog(parentFrame, "Корзина", true);  
 cartDialog.setSize(600, 400);  
 cartDialog.setLayout(new BorderLayout());  
  
 // Настройка дизайнерского шрифта  
 Font customFont = new Font("Verdana", Font.*PLAIN*, 14);  
  
 // Создание списка с возможностью выбора  
 JList<String> cartList = new JList<>(listModel);  
 cartList.setFont(customFont);  
  
 JScrollPane scrollPane = new JScrollPane(cartList);  
 JButton removeButton = new JButton("Удалить выбранное");  
 removeButton.setFont(customFont);  
 *addHoverEffect*(removeButton);  
  
 removeButton.addActionListener(e -> {  
 List<String> selectedItems = cartList.getSelectedValuesList();  
 if (!selectedItems.isEmpty()) {  
 for (String item : selectedItems) {  
 cart.remove(item);  
 listModel.removeElement(item);  
 }  
 JOptionPane.*showMessageDialog*(cartDialog, "Выбранные товары удалены из корзины!");  
 } else {  
 JOptionPane.*showMessageDialog*(cartDialog, "Пожалуйста, выберите товар для удаления.");  
 }  
 });  
  
 JButton removeAllButton = new JButton("Очистить корзину");  
 removeAllButton.setFont(customFont);  
 *addHoverEffect*(removeAllButton);  
  
 removeAllButton.addActionListener(e -> {  
 if (!cart.isEmpty()) {  
 cart.clear();  
 listModel.removeAllElements();  
 JOptionPane.*showMessageDialog*(cartDialog, "Корзина очищена");  
 }  
 else JOptionPane.*showMessageDialog*(cartDialog, "Корзина уже пуста!");  
 });  
  
  
 JButton checkoutButton = new JButton("Оформить заказ");  
 checkoutButton.setFont(customFont);  
 *addHoverEffect*(checkoutButton);  
  
 checkoutButton.addActionListener(e -> {  
 if (!cart.isEmpty()) {  
 cart.clear();  
 String filePath = "/Users/Vlas/IdeaProjects/laba4\_proganew/src/Order\_List.txt";  
 FileAppendLogger.*appendToFile*(filePath,"Пользователь: " + currentUser + "\n"+  
 "Товары: " + listModel.toString() + "\n"  
 );  
  
 listModel.clear();  
 JOptionPane.*showMessageDialog*(cartDialog, "Спасибо за ваш заказ!");  
 }  
 else {  
 JOptionPane.*showMessageDialog*(cartDialog, "Ваша корзина пуста.");  
 }  
 });  
  
  
  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.add(removeButton);  
 buttonPanel.add(checkoutButton);  
 buttonPanel.add(removeAllButton);  
  
 cartDialog.setMinimumSize(new Dimension(600, 400));  
  
 cartDialog.add(scrollPane, BorderLayout.*CENTER*);  
 cartDialog.add(buttonPanel, BorderLayout.*SOUTH*);  
 cartDialog.setLocationRelativeTo(parentFrame);  
 cartDialog.setVisible(true);  
  
 }  
  
  
}

Class: App.java

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public abstract class App implements ShowCart, ShowMainApp, ShowLoginScreen {  
 protected String nameCompany;  
 protected List<String> cart = new ArrayList<>();  
 protected Map<String, String> users = new HashMap<>();  
 protected boolean isLoggedIn = false;  
 protected String currentUser = "";  
  
 // меняем иконку  
 protected static void setIconApp(String path, JFrame Parentframe) {  
 ImageIcon icon = new ImageIcon(Thread.*currentThread*().getContextClassLoader().getResource(path));  
 Parentframe.setIconImage(icon.getImage());  
 }  
  
 // Метод для добавления анимации при наведении на кнопки  
 protected static void addHoverEffect(final JButton button) {  
 button.setFocusPainted(false);  
 button.setContentAreaFilled(false);  
  
 button.addMouseListener(new MouseAdapter() {  
 Color originalColor = button.getForeground();  
  
 @Override  
 public void mouseEntered(MouseEvent e) {  
 *animateButtonColor*(button, originalColor, Color.*BLUE*, 200);  
 }  
  
 @Override  
 public void mouseExited(MouseEvent e) {  
 *animateButtonColor*(button, Color.*BLUE*, originalColor, 200);  
 }  
 });  
 }  
  
 // Метод для анимации изменения цвета кнопки  
 protected static void animateButtonColor(final JButton button, Color startColor, Color endColor, int duration) {  
 Timer timer = new Timer(0, null);  
 int steps = 20;  
 int delay = duration / steps;  
 float[] startRGB = startColor.getRGBComponents(null);  
 float[] endRGB = endColor.getRGBComponents(null);  
 float[] delta = new float[3];  
 for (int i = 0; i < 3; i++) {  
 delta[i] = (endRGB[i] - startRGB[i]) / steps;  
 }  
 final int[] step = {0};  
 timer.addActionListener(e -> {  
 if (step[0] < steps) {  
 float[] newColor = new float[3];  
 for (int i = 0; i < 3; i++) {  
 newColor[i] = startRGB[i] + delta[i] \* step[0];  
 }  
 button.setForeground(new Color(newColor[0], newColor[1], newColor[2]));  
 step[0]++;  
 } else {  
 ((Timer) e.getSource()).stop();  
 }  
 });  
 timer.setDelay(delay);  
 timer.start();  
 }  
  
  
 // Слушатель на регистрацию  
 protected void registration(JButton registerButton, JTextField usernameField, JPasswordField passwordField, JFrame parentframe) {  
 registerButton.addActionListener(e -> {  
 String username = usernameField.getText();  
 String password = new String(passwordField.getPassword());  
 if (users.containsKey(username)) {  
 JOptionPane.*showMessageDialog*(parentframe, "Пользователь уже существует.");  
 } else if (!username.isEmpty() && !password.isEmpty()) {  
 users.put(username, password);  
 JOptionPane.*showMessageDialog*(parentframe, "Регистрация успешна!");  
 } else {  
 JOptionPane.*showMessageDialog*(parentframe, "Имя пользователя и пароль не могут быть пустыми.");  
 }  
 });  
 }  
  
  
 // слушатель на логин  
 protected void login(JButton loginButton, JTextField usernameField, JPasswordField passwordField, JFrame parentframe) {  
 loginButton.addActionListener(e -> {  
 String username = usernameField.getText();  
 String password = new String(passwordField.getPassword());  
 if (users.containsKey(username) && users.get(username).equals(password)) {  
 isLoggedIn = true;  
 currentUser = username;  
 JOptionPane.*showMessageDialog*(parentframe, "Вход выполнен!");  
 parentframe.dispose();  
 showMainApp();  
 } else {  
 JOptionPane.*showMessageDialog*(parentframe, "Неправильное имя пользователя или пароль.");  
 }  
 });  
 }  
  
  
}

Class: DatabaseManager.java

import javax.swing.\*;  
import java.awt.\*;  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.Scanner;  
  
public class FileAppendLogger {  
  
 public static void appendToFile(String filePath, String text) {  
 try (FileWriter writer = new FileWriter(filePath, true)) {  
 // Запись текущей даты и времени при запуске  
 writer.write("[" + *getCurrentTimestamp*() + "]\n");  
 writer.write(text + System.*lineSeparator*());  
  
 System.*out*.println("Данные успешно добавлены в файл.");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private static String getCurrentTimestamp() {  
 DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss");  
 return LocalDateTime.*now*().format(formatter);  
 }  
  
public static void appendFromFile(String currentUser, Frame parentFrame) {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setDialogTitle("Выберите файл для загрузки");  
  
 // Открытие диалогового окна  
 int result = fileChooser.showOpenDialog(null);  
  
 // Проверка, был ли файл выбран  
 if (result == JFileChooser.*APPROVE\_OPTION*) {  
 File selectedFile = fileChooser.getSelectedFile();  
 System.*out*.println("Выбран файл: " + selectedFile.getAbsolutePath());  
  
 // Чтение данных из выбранного файла  
 try (FileWriter writer = new FileWriter("/Users/Vlas/IdeaProjects/laba4\_proganew/src/WishList.txt", true);  
 Scanner scanner = new Scanner(selectedFile)) {  
 // Запись текущей даты и времени при запуске  
 writer.write("[" + *getCurrentTimestamp*() + "]\n");  
 writer.write("Пожелание от клиента. " + System.*lineSeparator*());  
 writer.write("Пользователь: "+currentUser+"\n" );  
 while (scanner.hasNextLine()) {  
 writer.write(scanner.nextLine() + System.*lineSeparator*());  
 }  
 writer.write(System.*lineSeparator*());  
 System.*out*.println("Данные успешно добавлены в файл.");  
 JOptionPane.*showMessageDialog*(parentFrame, "Данные успешно добавлены в файл.");  
  
 }  
 catch (FileNotFoundException e) {  
 JOptionPane.*showMessageDialog*(parentFrame, "Файл не найден: " + e.getMessage());  
 System.*err*.println("Файл не найден: " + e.getMessage());  
 }  
 catch (IOException e){  
  
 }  
 } else {  
 System.*out*.println("Файл не был выбран.");  
 JOptionPane.*showMessageDialog*(parentFrame, "Файл не был выбран.");  
 }  
  
  
  
}

}

Class: Converter.java

import javax.swing.\*;  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.Scanner;  
  
public class Converter {  
 private static String getCurrentTimestamp() {  
 DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss");  
 return LocalDateTime.*now*().format(formatter);  
 }  
 public static String getTextFromFile() {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setDialogTitle("Выберите файл для загрузки");  
  
 int result = fileChooser.showOpenDialog(null);  
  
 if (result == JFileChooser.*APPROVE\_OPTION*) {  
 File selectedFile = fileChooser.getSelectedFile();  
 System.*out*.println("Выбран файл: " + selectedFile.getAbsolutePath());  
  
 StringBuilder text = new StringBuilder();  
 try (Scanner scanner = new Scanner(selectedFile)) { // Закрывающая скобка после Scanner(selectedFile)  
 while (scanner.hasNextLine()) {  
 text.append(scanner.nextLine()).append(System.*lineSeparator*());  
 }  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 // Можно добавить сообщение пользователю  
 JOptionPane.*showMessageDialog*(null, "Файл не найден: " + e.getMessage(), "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 return text.toString();  
 } else {  
 // Пользователь отменил выбор файла  
 System.*out*.println("Файл не был выбран.");  
 return ""; // Или можно вернуть null, или выбросить исключение  
 }  
 }  
  
}

Class: PasswordUtil.java

import org.mindrot.jbcrypt.BCrypt;  
  
public class PasswordUtil {  
 // Метод для хеширования пароля  
 public static String hashPassword(String plainTextPassword) {  
 return BCrypt.*hashpw*(plainTextPassword, BCrypt.*gensalt*());  
 }  
  
 // Метод для проверки пароля  
 public static boolean checkPassword(String plainTextPassword, String hashedPassword) {  
 return BCrypt.*checkpw*(plainTextPassword, hashedPassword);  
 }  
}

Class: TariffAdvisor.java

import com.theokanning.openai.completion.chat.ChatCompletionRequest;  
import com.theokanning.openai.completion.chat.ChatCompletionResult;  
import com.theokanning.openai.completion.chat.ChatMessage;  
import com.theokanning.openai.completion.chat.ChatMessageRole;  
import com.theokanning.openai.service.OpenAiService;  
  
class TariffAdvisor {  
 private OpenAiService service;  
  
 public TariffAdvisor(OpenAiService service) {  
 this.service = service;  
 }  
  
 public String answerTariffQuestion(String userQuestion) {  
 String prompt = "Ты — эксперт по логистике. Вот доступные тарифы:\n" +  
 "Эконом: Базовый тариф для небольших грузов.\n" +  
 "Стандарт: Оптимальный выбор для стандартных перевозок.\n" +  
 "Бизнес: Повышенный комфорт и скорость доставки.\n" +  
 "Премиум: Максимальный сервис для особых клиентов.\n\n" +  
 "Ты также можешь рассчитать примерную стоимость до указанного адреса.\n" +  
 "Пользователь спрашивает: \"" + userQuestion + "\"\n\n" +  
 "Дай подробный и точный ответ, ссылаясь на тарифы и примерные стоимости, если это уместно."+  
 "Писать на одной строке не более 15 слов, постарайся делить текст на абзацы, так как текст не умещается в одну строку.";  
  
 ChatMessage systemMessage = new ChatMessage(ChatMessageRole.*SYSTEM*.value(), "Ты — помощник, хорошо разбирающийся в тарифах и логистике.");  
 ChatMessage userMessage = new ChatMessage(ChatMessageRole.*USER*.value(), prompt);  
  
 ChatCompletionRequest request = ChatCompletionRequest.*builder*()  
 .model("gpt-3.5-turbo")  
 .messages(java.util.Arrays.*asList*(systemMessage, userMessage))  
 .maxTokens(1000)  
 .temperature(0.7)  
 .build();  
  
 ChatCompletionResult result = service.createChatCompletion(request);  
 if (!result.getChoices().isEmpty()) {  
 return result.getChoices().get(0).getMessage().getContent();  
 } else {  
 return "Извините, я сейчас не могу ответить на ваш вопрос.";  
 }  
 }  
}

Class: ChatGPTDialog.java

import javax.swing.\*;  
import java.awt.\*;  
import java.util.Random;  
  
public class ChatGPTDialog extends JDialog {  
  
 private TariffAdvisor tariffAdvisor;  
 private JTextArea textArea;  
 private JTextField inputField;  
 private JButton sendButton;  
  
 // Настройки задержек  
 private static final Random *RANDOM* = new Random();  
 private static final int *BASE\_DELAY\_MS* = 15;  
 private static final int *RANDOM\_EXTRA\_DELAY\_MS* = 35;  
 private static final int *PUNCTUATION\_EXTRA\_DELAY\_MS* = 200;  
 private static final int *WORD\_SPACE\_EXTRA\_DELAY\_MS* = 50;  
  
 public ChatGPTDialog(Frame parent, TariffAdvisor tariffAdvisor) {  
 super(parent, "Чат с GPT", true);  
 this.tariffAdvisor = tariffAdvisor;  
  
 setLayout(new BorderLayout());  
  
 textArea = new JTextArea();  
 textArea.setEditable(false);  
 textArea.setFont(new Font("Verdana", Font.*PLAIN*, 14));  
 JScrollPane scrollPane = new JScrollPane(textArea);  
  
 JPanel inputPanel = new JPanel(new BorderLayout());  
 inputField = new JTextField();  
 sendButton = new JButton("Отправить");  
 inputPanel.add(inputField, BorderLayout.*CENTER*);  
 inputPanel.add(sendButton, BorderLayout.*EAST*);  
  
 add(scrollPane, BorderLayout.*CENTER*);  
 add(inputPanel, BorderLayout.*SOUTH*);  
  
 sendButton.addActionListener(e -> {  
 String question = inputField.getText().trim();  
 if (!question.isEmpty()) {  
 inputField.setText("");  
 // Получаем ответ от TariffAdvisor  
 String answer = tariffAdvisor.answerTariffQuestion(question);  
  
 // Очищаем текстовое поле перед выводом нового ответа  
 textArea.setText("");  
  
 // Выводим ответ "по-человечески"  
 printHumanLike(answer);  
 }  
 });  
  
 setSize(600, 400);  
 setLocationRelativeTo(parent);  
 }  
  
  
 private void printHumanLike(String message) {  
 char[] chars = message.toCharArray();  
  
 // Используем массив для хранения текущего индекса и Timer для анимации  
 final int[] currentIndex = {0};  
  
 // Создаём таймер, который срабатывает каждые X миллисекунд, добавляет по одному символу  
 Timer timer = new Timer(0, null);  
  
 timer.addActionListener(e -> {  
 if (currentIndex[0] < chars.length) {  
 char c = chars[currentIndex[0]];  
  
 // Добавляем символ в textArea  
 textArea.append(String.*valueOf*(c));  
  
 // Определяем задержку для следующего символа  
 int delay = *BASE\_DELAY\_MS* + *RANDOM*.nextInt(*RANDOM\_EXTRA\_DELAY\_MS* + 1);  
  
 if (isPunctuation(c)) {  
 delay += *PUNCTUATION\_EXTRA\_DELAY\_MS*;  
 }  
 if (c == ' ') {  
 delay += *WORD\_SPACE\_EXTRA\_DELAY\_MS* + *RANDOM*.nextInt(*RANDOM\_EXTRA\_DELAY\_MS* + 1);  
 }  
  
 // Обновляем задержку таймера для следующего цикла  
 timer.setDelay(delay);  
  
 // Скроллим вниз, чтобы видно было последний добавленный символ  
 textArea.setCaretPosition(textArea.getDocument().getLength());  
  
 currentIndex[0]++;  
 } else {  
 // Когда все символы добавлены, останавливаем таймер  
 timer.stop();  
 }  
 });  
  
 // Запускаем таймер с изначальной задержкой  
 timer.setDelay(10);  
 timer.start();  
 }  
  
 private boolean isPunctuation(char c) {  
 return c == '.' || c == ',' || c == '!' || c == '?' || c == ';' || c == ':';  
 }  
}

Interface: ShowCart.java

public interface ShowCart {  
 private void showCart(JFrame parentFrame){  
 }  
}

Interface: ShowMainApp.java

public interface ShowMainApp {  
 void showMainApp();  
}

Interface: ShowLoginScreen.java

public interface ShowLoginScreen {  
 void showLoginScreen();  
}

SQL-Запрос:

CREATE TABLE users (  
 id SERIAL PRIMARY KEY,  
 username VARCHAR(255) UNIQUE NOT NULL,  
 password VARCHAR(255) NOT NULL  
);  
  
CREATE TABLE carts (  
 id SERIAL PRIMARY KEY,  
 user\_id INT NOT NULL REFERENCES users(id) ON DELETE CASCADE,  
 product\_name VARCHAR(255) NOT NULL  
);  
  
CREATE TABLE orders (  
 id SERIAL PRIMARY KEY,  
 user\_id INT NOT NULL REFERENCES users(id) ON DELETE CASCADE,  
 order\_time TIMESTAMP NOT NULL DEFAULT *NOW*()  
);  
  
CREATE TABLE order\_items (  
 id SERIAL PRIMARY KEY,  
 order\_id INT NOT NULL REFERENCES orders(id) ON DELETE CASCADE,  
 product\_name VARCHAR(255) NOT NULL  
);  
  
CREATE TABLE wishes (  
 id SERIAL PRIMARY KEY,  
 user\_id INT NOT NULL REFERENCES users(id) ON DELETE CASCADE,  
 wish\_text TEXT NOT NULL,  
 created\_at TIMESTAMP NOT NULL DEFAULT *NOW*()  
);

**Пример работы программы**

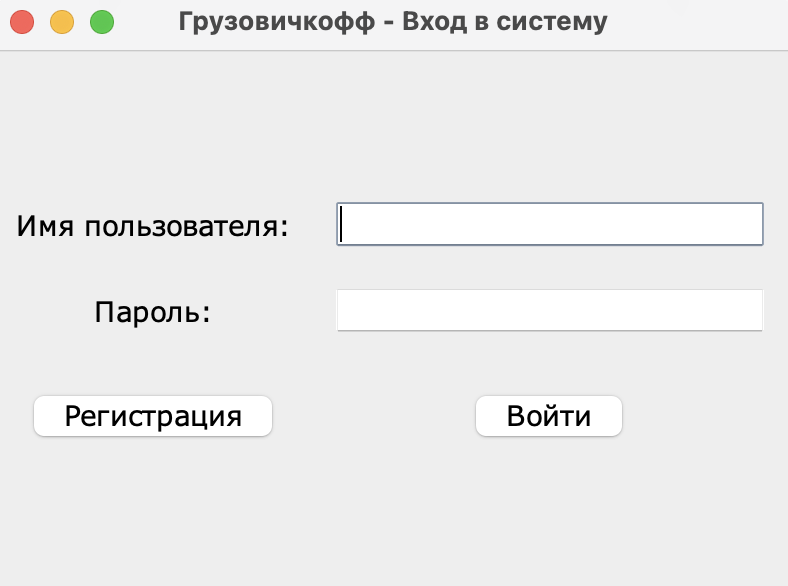


Рисунок 1 – Правильная работа программы(интерфейс регистрации и входа)

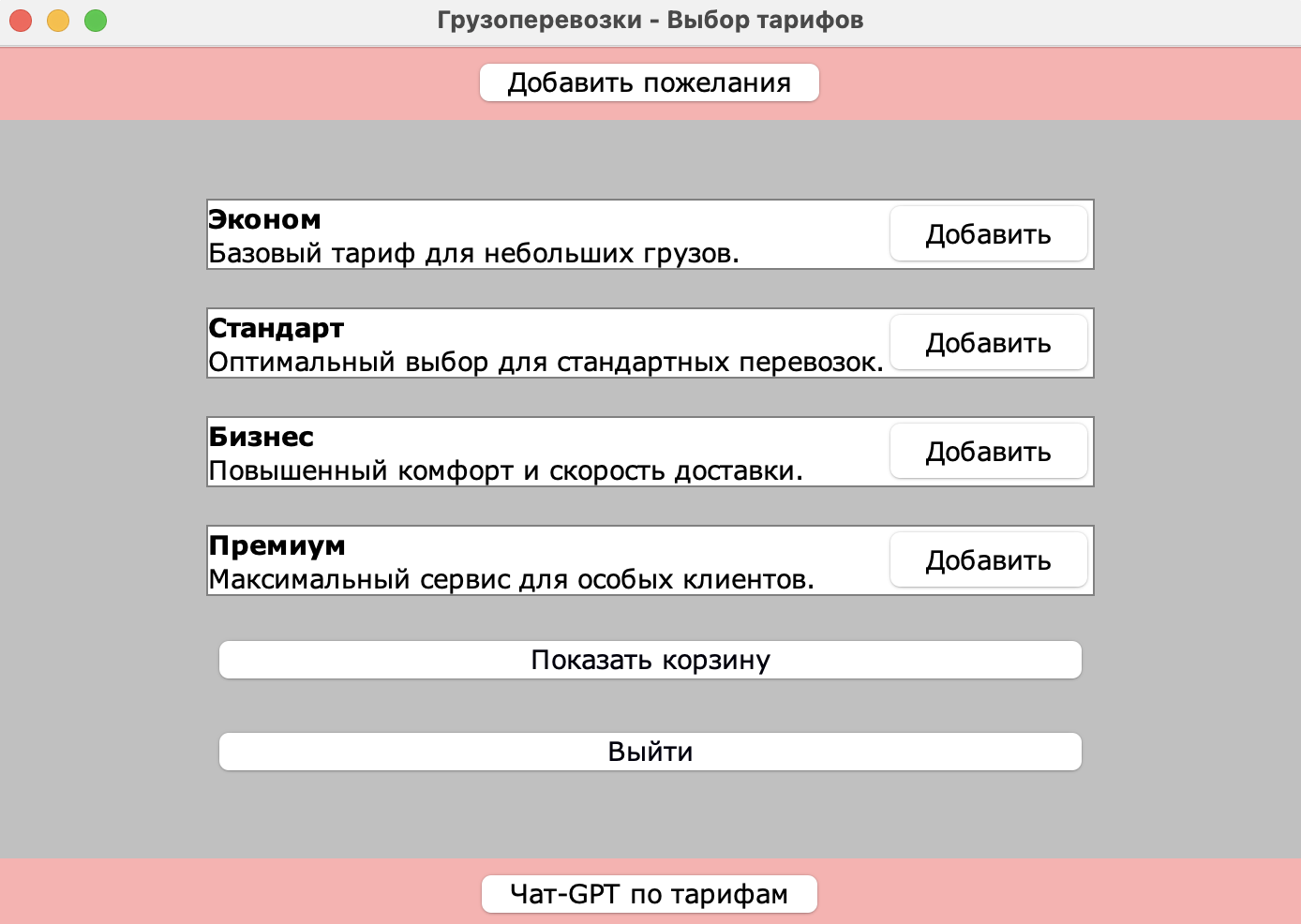


Рисунок 2 – Верная работа программы(главное меню программы)

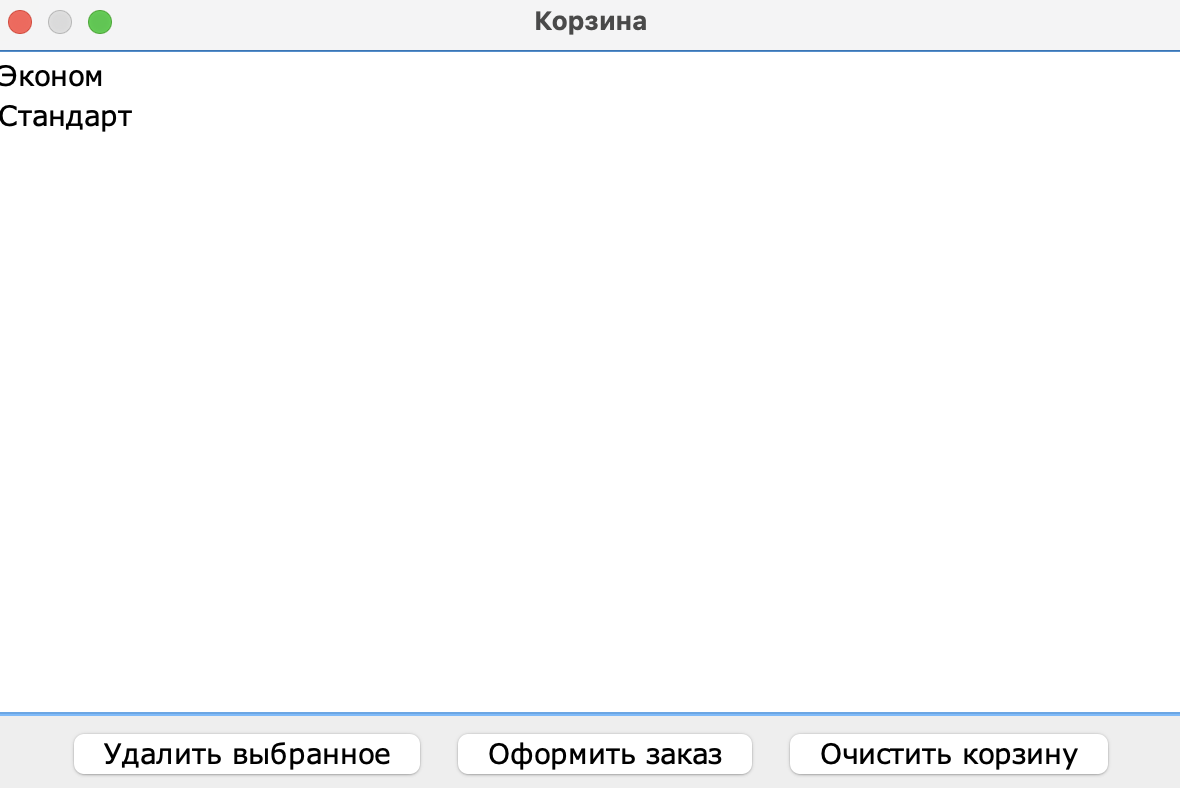
****

Рисунок 3 – Верная работа программы(корзина с заказами)

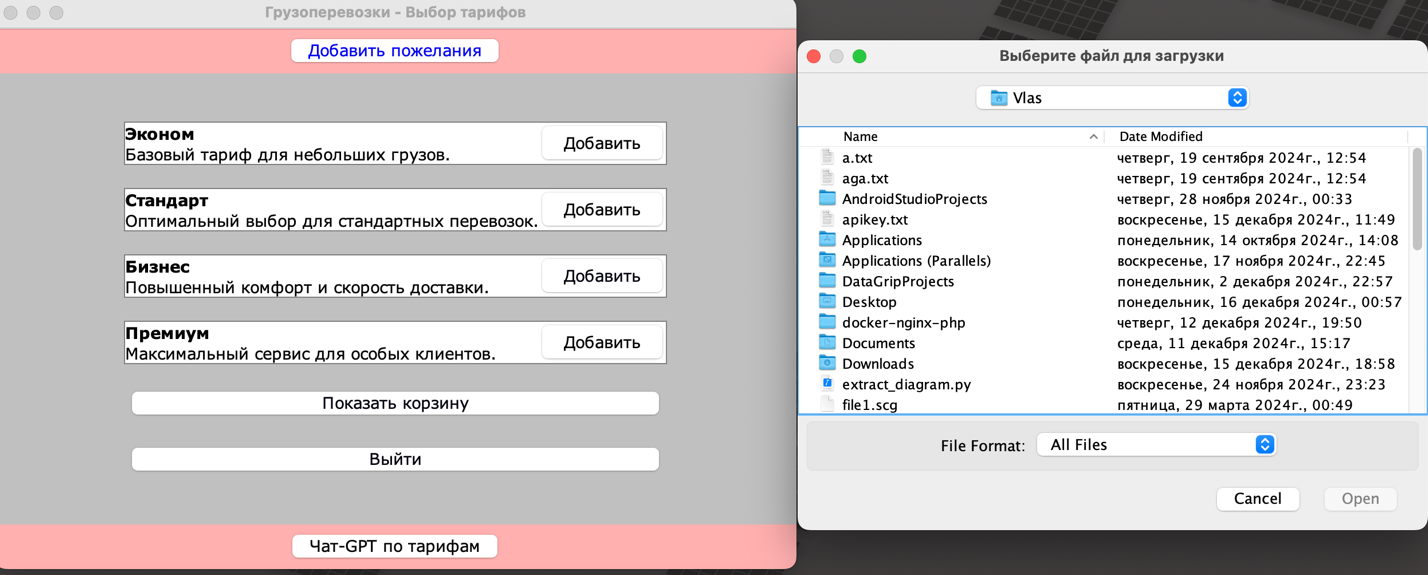


Рисунок 4 – Верная работа программы(импорт данных)

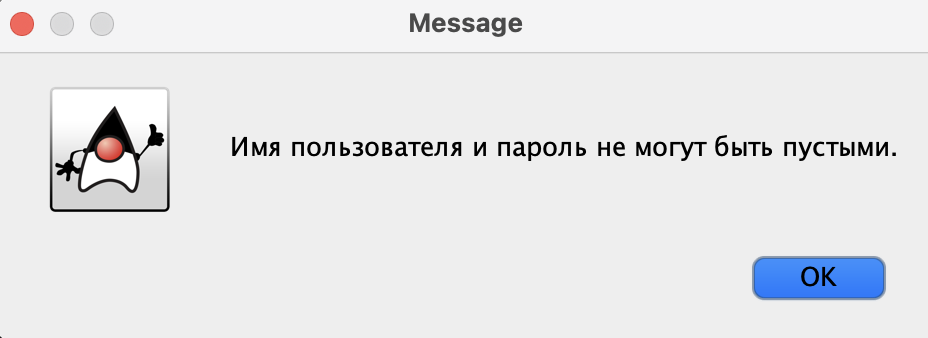


Рисунок 5 – Обработка попытки регистрации с пустыми полями

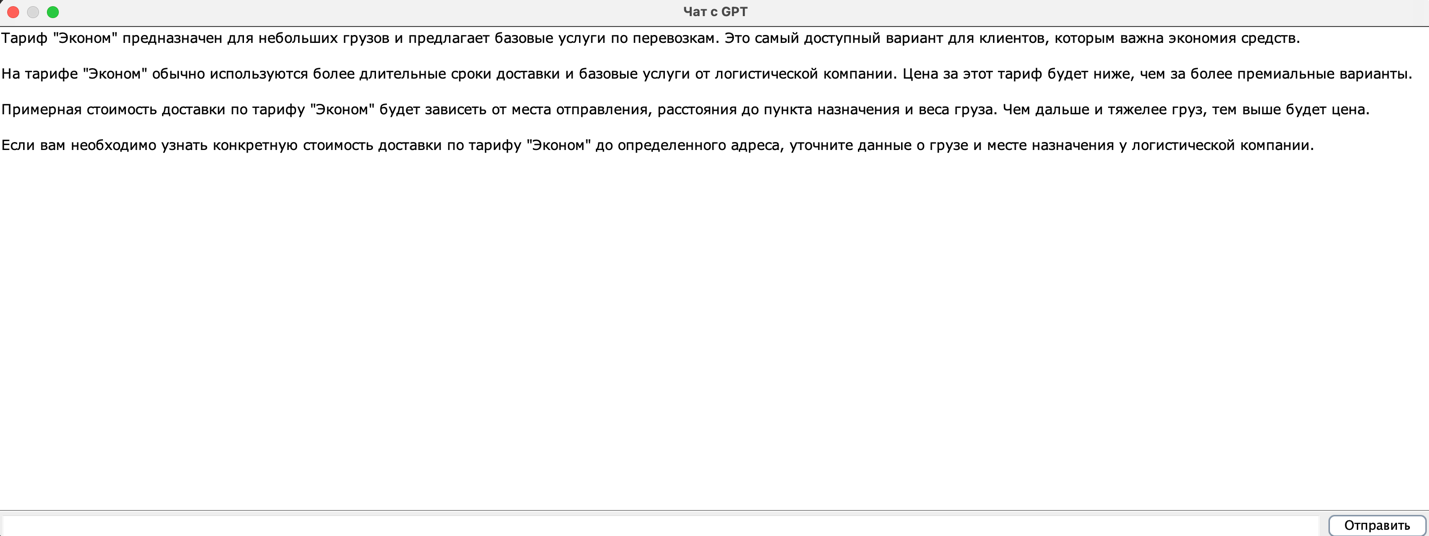


Рисунок 6 – Результат работы чата-GPT

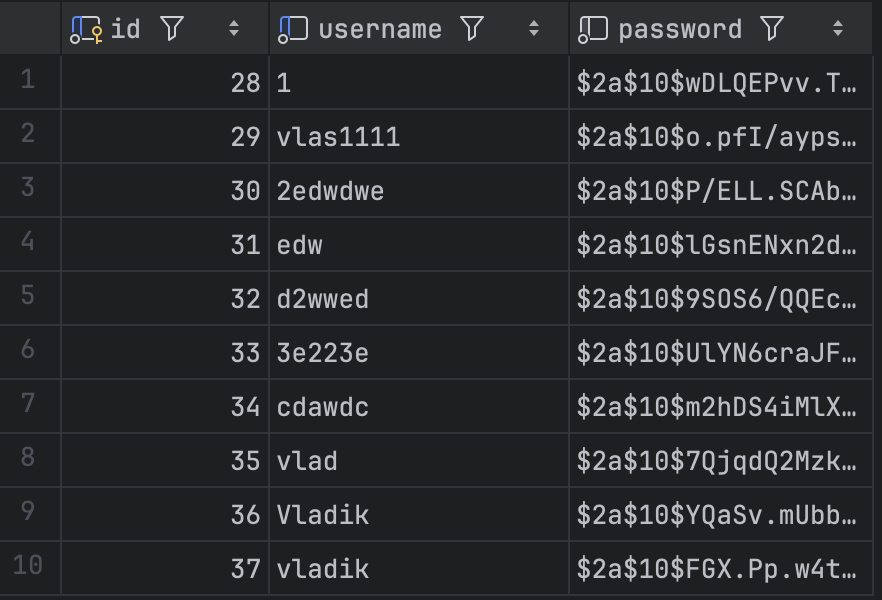


Рисунок 7– Верная работа программы(добавление информации о пользователях)

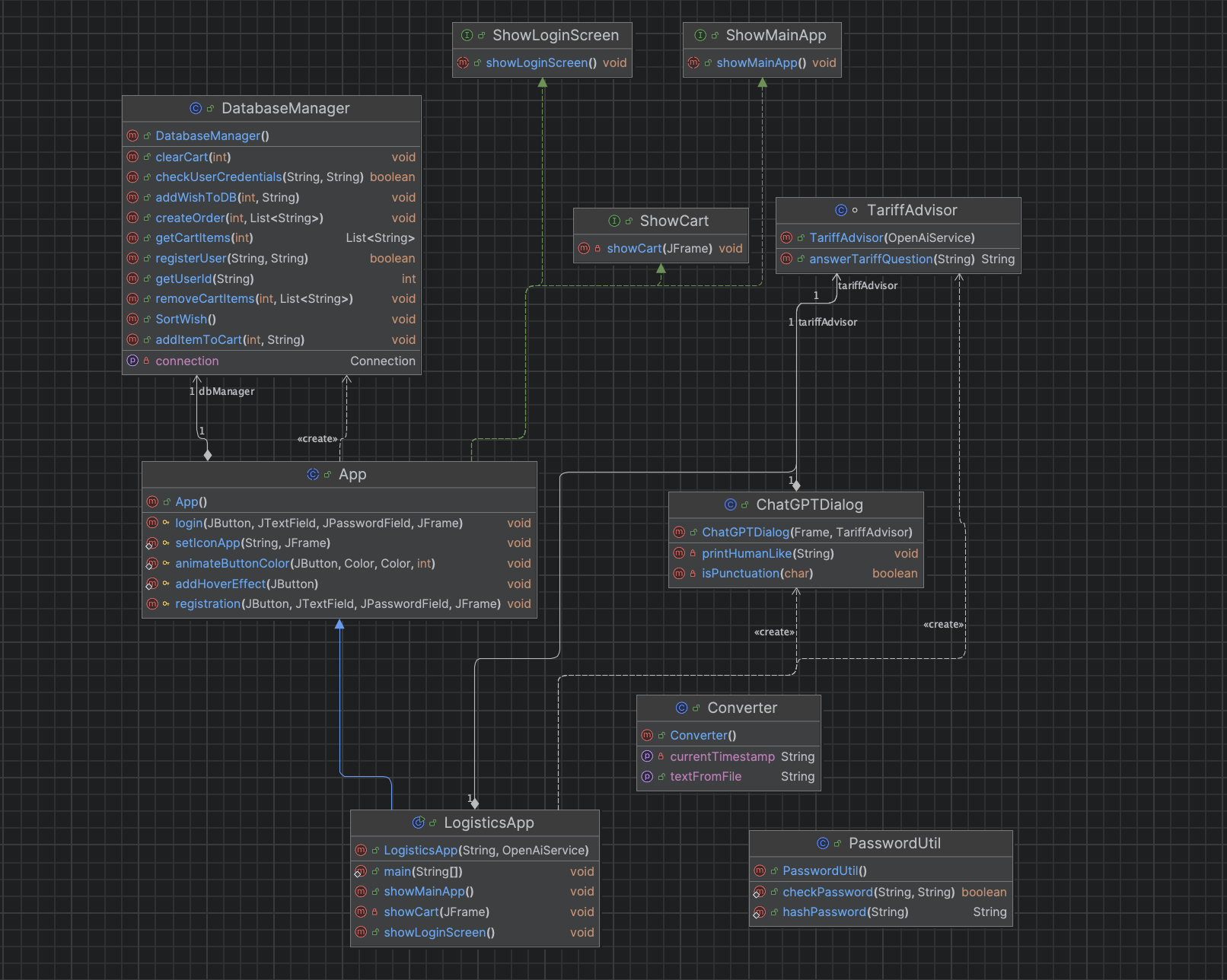


Рисунок 8 – Диаграмма классов

## **Вывод**

В результате выполнения данной лабораторной работы я ознакомился с устройством библиотеки Swing для создания GUI. Применил в своем приложении принципы инкапсуляции, наследования, абстракции и полиморфизма. Использовал в своей работе лямда-выражения(слушатели). Также подключился к OpenAI-API и успешно реализовал в виде чата-помощника в своем приложении. Подключился к базе данных Postgres и данные пользователя(пароль) хешировал с помощью безопасного алгоритма BCrypt. На выходе получилось приложение с гибким исходным кодом, с подключенной базой данных с данными, находящимися под защитой, которые не бояться “утечки” и полезной фичей в виде чата-ассистента, которая будет повышать прибыль грузоперевозочной компании, тем самым стимулируя бизнес на последующую расширяемость. Ссылка на Github: https://github.com/Vlasik2010/Vlasik.git