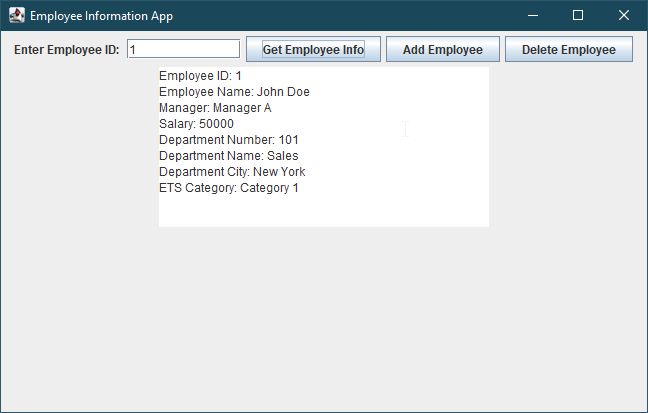
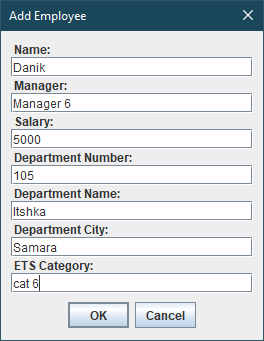
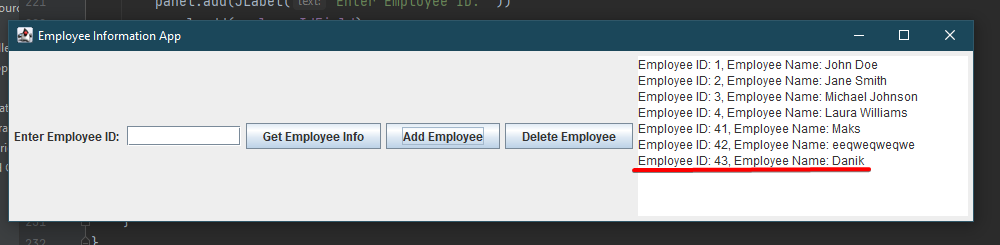
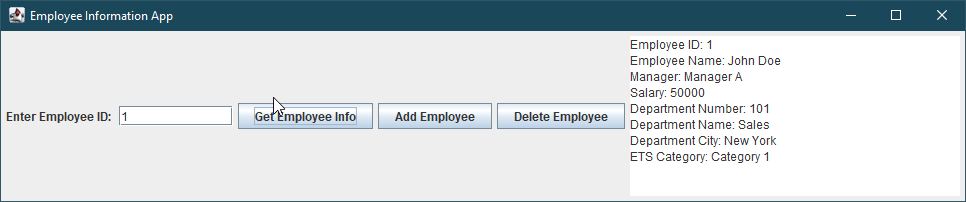


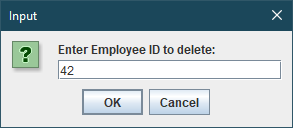
Пример работы программы:

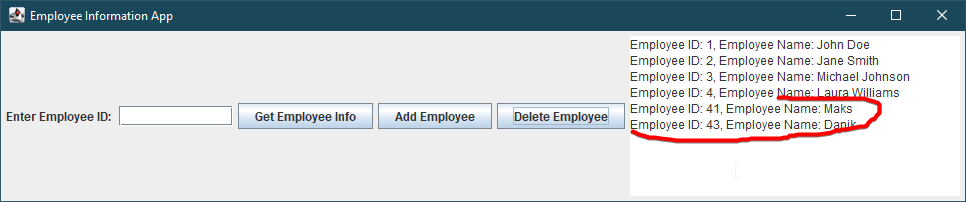












CREATE TABLE employees (

employee\_id SERIAL PRIMARY KEY,

name VARCHAR(100),

manager VARCHAR(100),

salary NUMERIC,

department\_number INT,

department\_name VARCHAR(100),

department\_city VARCHAR(100),

ets\_category VARCHAR(50)

);

INSERT INTO employees (name, manager, salary, department\_number, department\_name, department\_city, ets\_category)

VALUES

('John Doe', 'Manager A', 50000, 101, 'Sales', 'New York', 'Category 1'),

('Jane Smith', 'Manager B', 60000, 102, 'Marketing', 'Los Angeles', 'Category 2'),

('Michael Johnson', 'Manager C', 70000, 103, 'IT', 'San Francisco', 'Category 3'),

('Laura Williams', 'Manager A', 55000, 101, 'Sales', 'New York', 'Category 1');

Код программы:  
import javax.swing.\*  
import java.awt.\*  
import java.math.BigDecimal  
import java.sql.Connection  
import java.sql.DriverManager  
import java.sql.ResultSet  
  
class DatabaseManager {  
 private val url = "jdbc:postgresql://localhost:5432/test"  
 private val user = "postgres"  
 private val password = "password"  
  
 fun getConnection(): Connection {  
 return DriverManager.getConnection(url, user, password)  
 }  
  
 fun getEmployeeInfo(employeeId: Int): String {  
 val connection = getConnection()  
 val resultStringBuilder = StringBuilder()  
  
 try {  
 val statement = connection.createStatement()  
 val resultSet: ResultSet = statement.executeQuery("SELECT \* FROM employees WHERE employee\_id = $employeeId")  
  
 while (resultSet.next()) {  
 val employeeName = resultSet.getString("name")  
 val employeeNumber = resultSet.getInt("employee\_id")  
 val manager = resultSet.getString("manager")  
 val salary = resultSet.getBigDecimal("salary")  
 val departmentNumber = resultSet.getInt("department\_number")  
 val departmentName = resultSet.getString("department\_name")  
 val departmentCity = resultSet.getString("department\_city")  
 val etsCategory = resultSet.getString("ets\_category")  
  
 resultStringBuilder.append("Employee ID: $employeeNumber\n")  
 resultStringBuilder.append("Employee Name: $employeeName\n")  
 resultStringBuilder.append("Manager: $manager\n")  
 resultStringBuilder.append("Salary: $salary\n")  
 resultStringBuilder.append("Department Number: $departmentNumber\n")  
 resultStringBuilder.append("Department Name: $departmentName\n")  
 resultStringBuilder.append("Department City: $departmentCity\n")  
 resultStringBuilder.append("ETS Category: $etsCategory\n")  
 }  
  
 resultSet.close()  
 statement.close()  
 } catch (e: Exception) {  
 resultStringBuilder.append("Error: ${e.message}")  
 } finally {  
 connection.close()  
 }  
  
 return resultStringBuilder.toString()  
 }  
  
  
 fun addEmployee(name: String, manager: String, salary: Double, departmentNumber: Int, departmentName: String,  
 departmentCity: String, etsCategory: String): Boolean {  
 val connection = getConnection()  
  
 return try {  
 val statement = connection.createStatement()  
 statement.executeUpdate("INSERT INTO employees (name, manager, salary, department\_number, department\_name, " +  
 "department\_city, ets\_category) VALUES ('$name', '$manager', $salary, $departmentNumber, " +  
 "'$departmentName', '$departmentCity', '$etsCategory')")  
  
 statement.close()  
 true  
 } catch (e: Exception) {  
 e.printStackTrace()  
 false  
 } finally {  
 connection.close()  
 }  
 }  
  
 fun deleteEmployee(employeeId: Int): Boolean {  
 val connection = getConnection()  
  
 return try {  
 val statement = connection.createStatement()  
 statement.executeUpdate("DELETE FROM employees WHERE employee\_id = $employeeId")  
  
 statement.close()  
 true  
 } catch (e: Exception) {  
 e.printStackTrace()  
 false  
 } finally {  
 connection.close()  
 }  
 }  
}  
  
class EmployeeInformationApp : JFrame() {  
 private val dbManager = DatabaseManager()  
 private val employeeIdField = JTextField(10)  
 private val resultArea = JTextArea(10, 30)  
  
 private fun updateEmployeeInfo() {  
 // Очищаем текстовую область перед обновлением  
 resultArea.*text* = ""  
  
 // Выводим информацию о всех сотрудниках  
 val connection = dbManager.getConnection()  
 try {  
 val statement = connection.createStatement()  
 val resultSet: ResultSet = statement.executeQuery("SELECT \* FROM employees")  
  
 while (resultSet.next()) {  
 val employeeNumber = resultSet.getInt("employee\_id")  
 val employeeName = resultSet.getString("name")  
 resultArea.append("Employee ID: $employeeNumber, Employee Name: $employeeName\n")  
 }  
  
 resultSet.close()  
 statement.close()  
 } catch (e: Exception) {  
 resultArea.*text* = "Error: ${e.message}"  
 } finally {  
 connection.close()  
 }  
 }  
  
 init {  
 *title* = "Employee Information App"  
 *defaultCloseOperation* = *EXIT\_ON\_CLOSE* val panel = JPanel()  
 panel.*layout* = FlowLayout()  
  
 val getEmployeeInfoButton = JButton("Get Employee Info")  
 getEmployeeInfoButton.addActionListener **{** val employeeId = employeeIdField.*text*.*toIntOrNull*()  
 if (employeeId != null) {  
 val employeeInfo = dbManager.getEmployeeInfo(employeeId)  
 if (employeeInfo.*startsWith*("Error")) {  
 resultArea.*text* = employeeInfo  
 } else {  
 resultArea.*text* = employeeInfo  
 }  
 } else {  
 resultArea.*text* = "Invalid employee ID"  
 }  
 **}** val addEmployeeButton = JButton("Add Employee")  
 addEmployeeButton.addActionListener **{** val dialogPanel = JPanel()  
 dialogPanel.*layout* = BoxLayout(dialogPanel, BoxLayout.*Y\_AXIS*)  
  
 val nameField = JTextField(15)  
 val managerField = JTextField(15)  
 val salaryField = JTextField(15)  
 val departmentNumberField = JTextField(15)  
 val departmentNameField = JTextField(15)  
 val departmentCityField = JTextField(15)  
 val etsCategoryField = JTextField(15)  
  
 dialogPanel.add(JLabel("Name: "))  
 dialogPanel.add(nameField)  
 dialogPanel.add(JLabel("Manager: "))  
 dialogPanel.add(managerField)  
 dialogPanel.add(JLabel("Salary: "))  
 dialogPanel.add(salaryField)  
 dialogPanel.add(JLabel("Department Number: "))  
 dialogPanel.add(departmentNumberField)  
 dialogPanel.add(JLabel("Department Name: "))  
 dialogPanel.add(departmentNameField)  
 dialogPanel.add(JLabel("Department City: "))  
 dialogPanel.add(departmentCityField)  
 dialogPanel.add(JLabel("ETS Category: "))  
 dialogPanel.add(etsCategoryField)  
  
 val result = JOptionPane.showConfirmDialog(this, dialogPanel, "Add Employee",  
 JOptionPane.*OK\_CANCEL\_OPTION*, JOptionPane.*PLAIN\_MESSAGE*)  
  
 if (result == JOptionPane.*OK\_OPTION*) {  
 val name = nameField.*text* val manager = managerField.*text* val salary = salaryField.*text*.*toDoubleOrNull*() ?: 0.0  
 val departmentNumber = departmentNumberField.*text*.*toIntOrNull*() ?: 0  
 val departmentName = departmentNameField.*text* val departmentCity = departmentCityField.*text* val etsCategory = etsCategoryField.*text* if (dbManager.addEmployee(name, manager, salary, departmentNumber, departmentName,  
 departmentCity, etsCategory)) {  
 resultArea.*text* = "Employee added successfully."  
 // Обновляем информацию о сотрудниках после добавления  
 updateEmployeeInfo()  
 } else {  
 resultArea.*text* = "Error adding employee."  
 }  
 }  
 **}** val deleteEmployeeButton = JButton("Delete Employee")  
 deleteEmployeeButton.addActionListener **{** val employeeId = JOptionPane.showInputDialog(this, "Enter Employee ID to delete:")  
  
 if (employeeId != null) {  
 val id = employeeId.*toIntOrNull*()  
 if (id != null) {  
 if (dbManager.deleteEmployee(id)) {  
 resultArea.*text* = "Employee with ID $id deleted successfully."  
 // Обновляем информацию о сотрудниках после удаления  
 updateEmployeeInfo()  
 } else {  
 resultArea.*text* = "Error deleting employee."  
 }  
 } else {  
 resultArea.*text* = "Invalid Employee ID."  
 }  
 }  
 **}** // Первичное обновление информации о сотрудниках  
 updateEmployeeInfo()  
  
 panel.add(JLabel("Enter Employee ID: "))  
 panel.add(employeeIdField)  
 panel.add(getEmployeeInfoButton)  
 panel.add(addEmployeeButton)  
 panel.add(deleteEmployeeButton)  
 panel.add(resultArea)  
  
 *contentPane*.add(panel)  
 pack()  
 *isVisible* = true  
 }  
}  
  
fun main() {  
 EmployeeInformationApp()  
}