**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Институт информационных технологий и управления**

**Кафедра компьютерных систем и программных технологий**

**ОТЧЕТ**

**по лабораторной работе №8**

**«Разработка простого клиентского приложения»**

**Информационное обеспечение систем управления**

Студент гр. 43501/1 Васильев А.М.

Преподаватель Мяснов А.В.

**Санкт-Петербург**

**2015**

# 1. Цель работы

Ознакомится с работой с БД с помощью API на примере СУБД Firebird 2.5 и языка Java.

# 2. Программа работы

Необходимо создать консольное приложение, выполняющее следующие функции:

* соединение с БД, выполнение фиксированного SQL-запроса и получение результатов запроса
* добавление данных в одну из таблиц БД
* выполнение хранимой процедуры

# 3. Разработка клиентского приложения

Используется среда NetBeans 8.0.2. Работа с СУБД осуществляется с помощью драйвера JDBC 2.6.6.

Листинг приложения:

package javaexample;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.io.UnsupportedEncodingException;

import java.util.Scanner;

import java.sql.\*;

import java.util.InputMismatchException;

import java.util.Vector;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author ЮЛМАРТ

\*/

public class JavaExample {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) throws UnsupportedEncodingException, ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

// TODO code application logic here

System.out.println("Hello world");

String strURL = "jdbc:firebirdsql://localhost/C:\\db\\nanofon.fdb?lc\_ctype=WIN1251&charSet=Cp1251";

// Инициализируемя Firebird JDBC driver.

// Эта строка действительна только для Firebird.

try {

// The newInstance() call is a work around for some

// broken Java implementations

Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();

} catch (Exception E) {

System.err.println("Unable to load driver.");

}

Connection conn = null;

//Создаём подключение к базе данных

conn = DriverManager.getConnection(strURL, "SYSDBA", "masterkey");

if (conn == null) {

System.err.println("Could not connect to database");

System.exit(-2);

}

Statement stmt = conn.createStatement();

Scanner scanner = new Scanner(System.in);

while (true) {

try {

System.out.println("Мобильный оператор, выберите действие");

System.out.println("1. Вывести стоимость звонков для всех тарифов");

System.out.println("2. Пополнить баланс");

System.out.println("3. Отправить SMS");

System.out.println("4. Выход");

int res = scanner.nextInt();

if (res < 1 || res > 4) {

System.out.println("Неверный вариант\n");

continue;

}

if (res == 1) {

ResultSet rs;

rs = stmt.executeQuery("select tariffs.tariff\_name,call\_prices.price as calls from tariffs,call\_prices,actions\_tariffs act where act.tariff=tariffs.id and act.action=call\_prices.id;");

while (rs.next()) {

System.out.println(rs.getString(1) + " " + rs.getString(2));

}

} else if (res == 2) {

Long number;

System.out.println("Please enter phone number");

number = scanner.nextLong();

if (number < 9210000000L || number > 9219999999L) {

System.out.println("Неверный номер");

System.exit(-1);

}

System.out.println("Введите сумму в рублях:");

Integer amount;

amount = scanner.nextInt();

if (amount < 10 || amount > 10000) {

System.out.println("Недопустимая сумма");

System.exit(-1);

}

String str = String.format("insert into payments values (gen\_id(gen\_payid,1),(select id from subs where number = (select id from numbers where number=%d)),%d,current\_timestamp)", number, amount);

System.out.println("Баланс пополнен");

stmt.executeUpdate(str);

str = String.format("update subs set balance = balance + %d where number = (select id from numbers where number = %d)", amount, number);

stmt.executeUpdate(str);

} else if (res == 3) {

CallableStatement call\_stmt = conn.prepareCall("{call sms(?,?)}");

Long number;

System.out.println("Введите ваш номер");

number = scanner.nextLong();

if (number < 9210000000L || number > 9219999999L) {

System.out.println("Неверный номер");

System.exit(-1);

}

ResultSet rs;

String str = String.format("select id from subs where number=(select id from numbers where number=%d)",number);

rs = stmt.executeQuery(str);

rs.next();

System.out.println(rs.getString(1));

Integer sub\_id = Integer.parseInt(rs.getString(1));

call\_stmt.setInt(1, sub\_id.intValue());

call\_stmt.setInt(2, 1);

call\_stmt.execute();

System.out.println("Сообщение отправлено");

} else if (res == 4) {

System.out.println("До свидания!");

System.exit(0);

}

} catch (InputMismatchException e) {

System.out.println("Wrong input");

System.exit(-1);

}

}

}

}

Порядок его работы:

1. Инициализация

2. Бесконечный цикл, в котором пользователь выбирает один из 4 вариантов действия, в том числе выход из приложения.

Тестирование приложения:

Мобильный оператор, выберите действие

1. Вывести стоимость звонков для всех тарифов

2. Пополнить баланс

3. Отправить SMS

4. Выход

1

Звонить маме 1.00

Простецкий 2.00

SMS-зависимый 3.00

Все по 3 3.00

Модем 3.00

Стабильный 3.00

Мобильный оператор, выберите действие

1. Вывести стоимость звонков для всех тарифов

2. Пополнить баланс

3. Отправить SMS

4. Выход

2

Введите номер телефона (10 цифр)

9218590115

Введите сумму в рублях:

100

Баланс пополнен

Мобильный оператор, выберите действие

1. Вывести стоимость звонков для всех тарифов

2. Пополнить баланс

3. Отправить SMS

4. Выход

3

Введите ваш номер (10 цифр)

9218590115

Сообщение отправлено

Мобильный оператор, выберите действие

1. Вывести стоимость звонков для всех тарифов

2. Пополнить баланс

3. Отправить SMS

4. Выход

4

До свидания!

# 4. Вывод

В ходе работы было создано простое клиентское приложение, выполняющее различные запросы к базе данных, добавляющее записи и вызывающее хранимые процедуры. API, предоставляемые драйвером СУБД Firebird для языка Java, позволяют использовать все возможности SQL-запросов в приложении. Поскольку большая часть логики была реализована на стороне БД, клиентское приложение получилось простым.

Трехзвенная архитектура— архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений). Клиент — это интерфейсный компонент комплекса, предоставляемый конечному пользователю. Этот уровень не должен иметь прямых связей с базой данных (по требованиям безопасности и масштабируемости), быть нагруженным основной бизнес-логикой (по требованиям масштабируемости) и хранить состояние приложения (по требованиям надёжности). На этот уровень обычно выносится только простейшая бизнес-логика. Сервер приложений (средний слой, связующий слой) располагается на втором уровне, на нём сосредоточена большая часть бизнес-логики. Вне его остаются только фрагменты, экспортируемые на клиента (терминалы), а также элементы логики, погруженные в базу данных (хранимые процедуры и триггеры). Сервер баз данных (слой данных) обеспечивает хранение данных и выносится на отдельный уровень, реализуется, как правило, средствами систем управления базами данных, подключение к этому компоненту обеспечивается только с уровня сервера приложений. К основным преимуществам такой архитектуры относится масштабируемость, надежность, высокая скорость работы и возможность балансировки нагрузки. Недостатки – сложность разработки приложений и дополнительные издержки в администрировании.