Министерство цифрового развития, связи и массовых коммуникаций

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Информатика»

Лабораторная работа №5

по дисциплине

«Проектирование клиентских приложений»

Вариант №4

Выполнил:

студент гр.БЭИ2102

Борисов А. М.

Принял:

Доцент к.т.н

Халабия Р. Ф.

Москва 2024

**1. Постановка задачи**

Доработать программу из ЛР№4, добавив функционал экспорта в БД и импорта из БД, используя MySQL.

**2. Разработка схемы алгоритмов**

Разработанная схема программы представлена на рисунках 1-6.

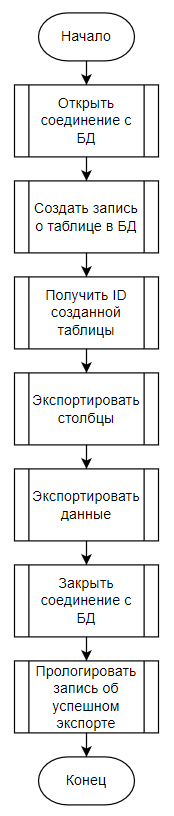


Рисунок 1 – Алгоритм экспорта таблицы в БД

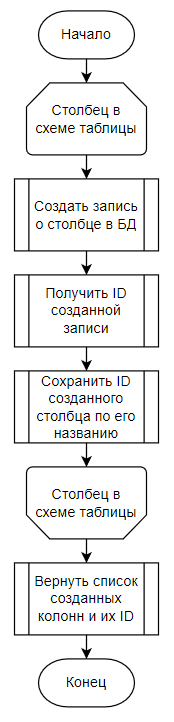


Рисунок 2 – Алгоритм экспорта столбцов в БД

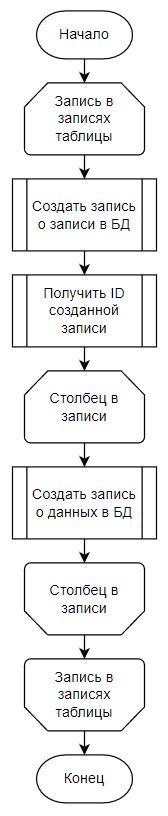


Рисунок 3 – Алгоритм экспорта данных в БД

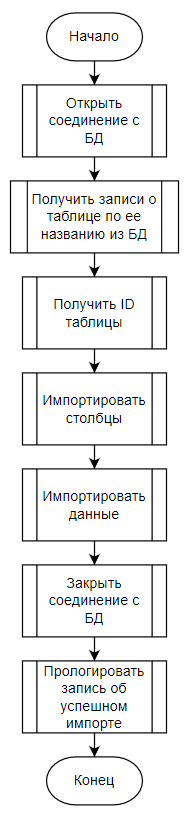


Рисунок 4 – Алгоритм импорта таблицы из БД

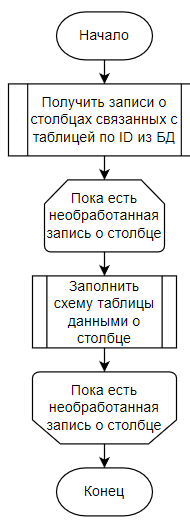


Рисунок 5 – Алгоритм импорта столбцов из БД

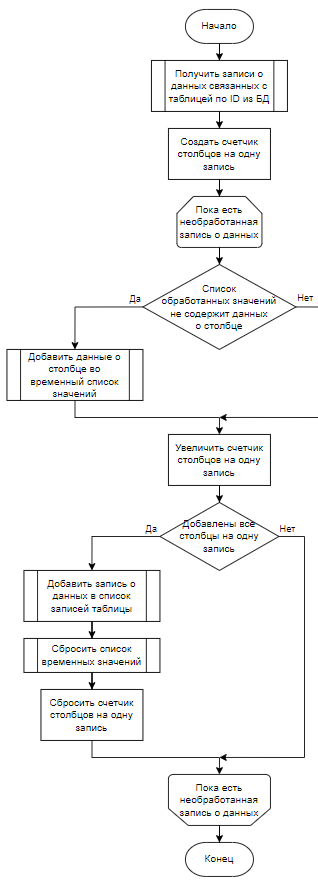


Рисунок 6 – Алгоритм импорта данных из БД

**3. Разработка программы**

**3.1 Разработка графического интерфейса пользователя**

Разработанный интерфейс программы, представлен на рисунке 7.

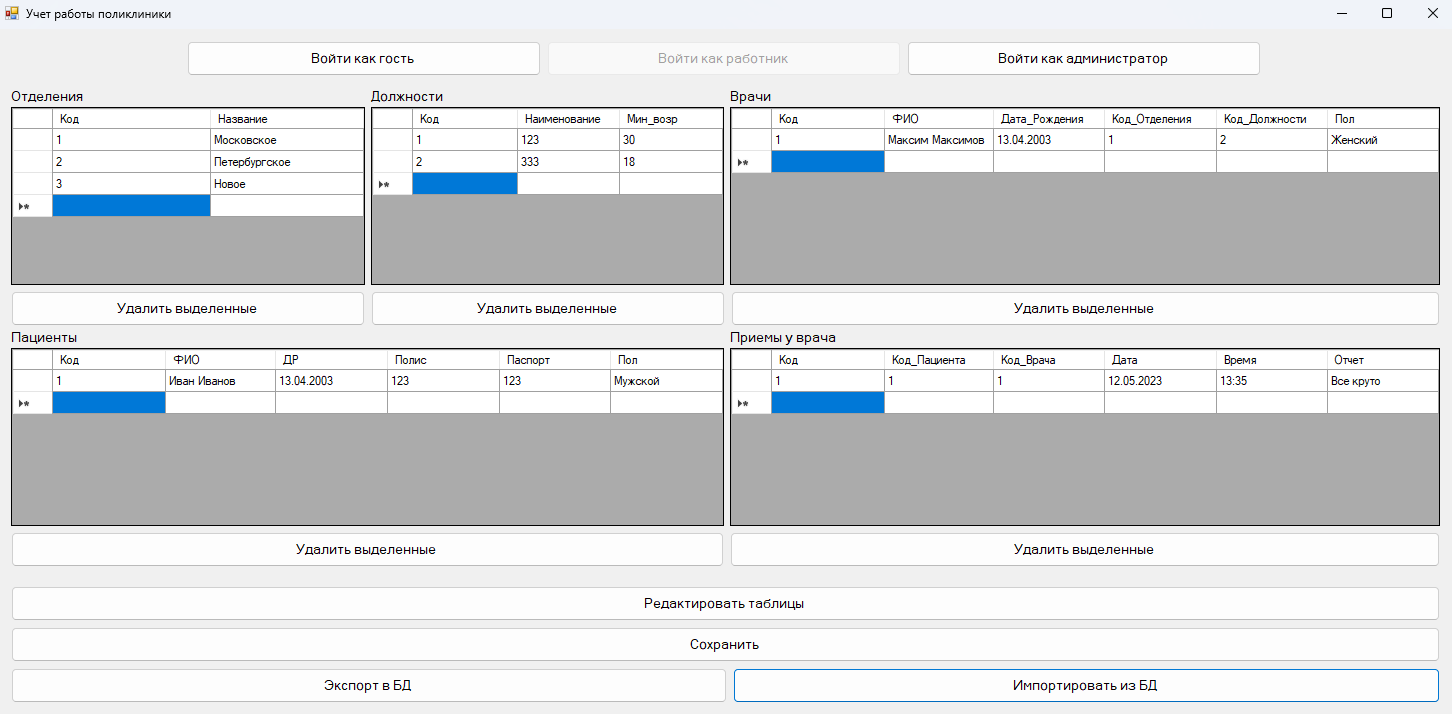


Рисунок 7 – Интерфейс программы

**3.2 Программный код**

Код класса работы с БД:

using HLP\_RKP\_LR3.Models;

using MySqlConnector;

using System;

using System.Collections.Generic;

namespace HLP\_RKP\_LR2.Models.Utils

{

internal class DBConnector

{

private const string connectionString = "Server=localhost;Database=lr5;Uid=root;Pwd=mysql;";

private const string TABLES\_TABLE = "client\_tables";

private const string COLUMNS\_TABLE = "columns";

private const string DATA\_TABLE = "data";

private const string ITEMS\_TABLE = "items";

private static void ExecuteCommandNonQuery(string query, MySqlConnection connection)

{

using (MySqlCommand command = new MySqlCommand(query, connection))

{

command.ExecuteNonQuery();

}

}

public static int GetLastId(MySqlConnection connection)

{

string query = "select last\_insert\_id()";

using (MySqlCommand command = new MySqlCommand(query, connection))

{

return Convert.ToInt32(command.ExecuteScalar());

}

}

public static MySqlDataReader ExecuteCommandReader(MySqlConnection connection, string query)

{

MySqlDataReader reader = null;

using (MySqlCommand command = new MySqlCommand(query, connection))

{

reader = command.ExecuteReader();

}

return reader;

}

public static void ClearOldData()

{

string query;

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

connection.Open();

query = $"SET FOREIGN\_KEY\_CHECKS = 0";

ExecuteCommandNonQuery(query, connection);

query = $"truncate {ITEMS\_TABLE}";

ExecuteCommandNonQuery(query, connection);

query = $"truncate {DATA\_TABLE}";

ExecuteCommandNonQuery(query, connection);

query = $"truncate {COLUMNS\_TABLE}";

ExecuteCommandNonQuery(query, connection);

query = $"truncate {TABLES\_TABLE}";

ExecuteCommandNonQuery(query, connection);

query = $"SET FOREIGN\_KEY\_CHECKS = 1";

ExecuteCommandNonQuery(query, connection);

connection.Close();

}

}

public static void ExportTable(string tableName, Dictionary<string, ItemTypes> schema, List<TableItem> items)

{

try

{

string query;

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

connection.Open();

query = $"insert {TABLES\_TABLE} (name) values (\"{tableName}\")";

ExecuteCommandNonQuery(query, connection);

int tableID = GetLastId(connection);

Dictionary<string, int> columnsWithIDs = ExportColumns(connection, tableID, schema);

ExportData(connection, tableID, columnsWithIDs, items);

connection.Close();

}

Logger.ExportToDB(tableName);

}

catch

{

Logger.Error($"Ошибка экспорта в БД таблицы \"{tableName}\"");

throw new Exception();

}

}

public static Dictionary<string, int> ExportColumns(MySqlConnection connection, int tableID, Dictionary<string, ItemTypes> schema)

{

string query;

Dictionary<string, int> columnsWithIDs = new Dictionary<string, int>();

foreach (KeyValuePair<string, ItemTypes> pair in schema)

{

query = $"insert {COLUMNS\_TABLE} (name, data\_type, table\_id) values (\"{pair.Key}\", \"{TypedItem.TypeToStringDict[pair.Value]}\", {tableID})";

ExecuteCommandNonQuery(query, connection);

int columnID = GetLastId(connection);

columnsWithIDs.Add(pair.Key, columnID);

}

return columnsWithIDs;

}

public static void ExportData(MySqlConnection connection, int tableID, Dictionary<string, int> columnsWithIDs, List<TableItem> items)

{

string query;

foreach (TableItem item in items)

{

query = $"insert {ITEMS\_TABLE}(table\_id) values ({tableID})";

ExecuteCommandNonQuery(query, connection);

int itemID = GetLastId(connection);

foreach (KeyValuePair<string, TypedItem> pair in item.Values)

{

query = $"insert {DATA\_TABLE} (value, column\_id, item\_id) values (\"{pair.Value.Value}\", \"{columnsWithIDs[pair.Key]}\", {itemID})";

ExecuteCommandNonQuery(query, connection);

}

}

}

public static void ImportTable(string tableName, Dictionary<string, ItemTypes> schema, List<TableItem> items)

{

try

{

string query;

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

connection.Open();

query = $"select \* from {TABLES\_TABLE} where name = \"{tableName}\"";

MySqlDataReader reader = ExecuteCommandReader(connection, query);

reader.Read();

int tableID = reader.GetInt32("id");

reader.Close();

ImportColumns(connection, tableID, schema);

ImportData(connection, tableID, items, schema);

connection.Close();

}

Logger.ImportFromDB(tableName);

}

catch

{

Logger.Error($"Ошибка импорта из БД таблицы \"{tableName}\"");

throw new Exception();

}

}

private static void ImportColumns(MySqlConnection connection, int tableID, Dictionary<string, ItemTypes> schema)

{

schema.Clear();

string query = $"select \* from {COLUMNS\_TABLE} where table\_id = {tableID}";

MySqlDataReader reader = ExecuteCommandReader(connection, query);

while (reader.Read())

{

string name = reader.GetString("name");

string data\_type = reader.GetString("data\_type");

ItemTypes type = TypedItem.StringToTypeDict[data\_type];

schema.Add(name, type);

}

reader.Close();

}

private static void ImportData(MySqlConnection connection, int tableID, List<TableItem> items, Dictionary<string, ItemTypes> schema)

{

items.Clear();

string query = $"select {COLUMNS\_TABLE}.name, {DATA\_TABLE}.value"

+ $" from {ITEMS\_TABLE}"

+ $" join {DATA\_TABLE} on {DATA\_TABLE}.item\_id = {ITEMS\_TABLE}.id"

+ $" join {COLUMNS\_TABLE} on {COLUMNS\_TABLE}.id = {DATA\_TABLE}.column\_id"

+ $" where {ITEMS\_TABLE}.table\_id = {tableID}";

MySqlDataReader reader = ExecuteCommandReader(connection, query);

int columnCount = 0;

Dictionary<string, string> itemValues = new Dictionary<string, string>();

while (reader.Read())

{

string name = reader.GetString("name");

string value = reader.GetString("value");

if (!itemValues.ContainsKey(name))

{

itemValues.Add(name, value);

}

columnCount++;

if (columnCount == schema.Count)

{

items.Add(new TableItem(schema, itemValues));

itemValues.Clear();

columnCount = 0;

}

};

reader.Close();

}

}

}

Код методов нажатия на кнопки экспорта в БД и импорта из БД:

private void exportDBBtn\_Click(object sender, EventArgs e)

{

TablesActions.ExportToDB();

}

private void importDBBtn\_Click(object sender, EventArgs e)

{

TablesActions.ImportFromDB(dgvPosition, dgvDepartment, dgvDoctor, dgvPatient, dgvAppointment);

}

**4. Результат работы программы**

Результаты работы программы представлен на рисунках 8-9.

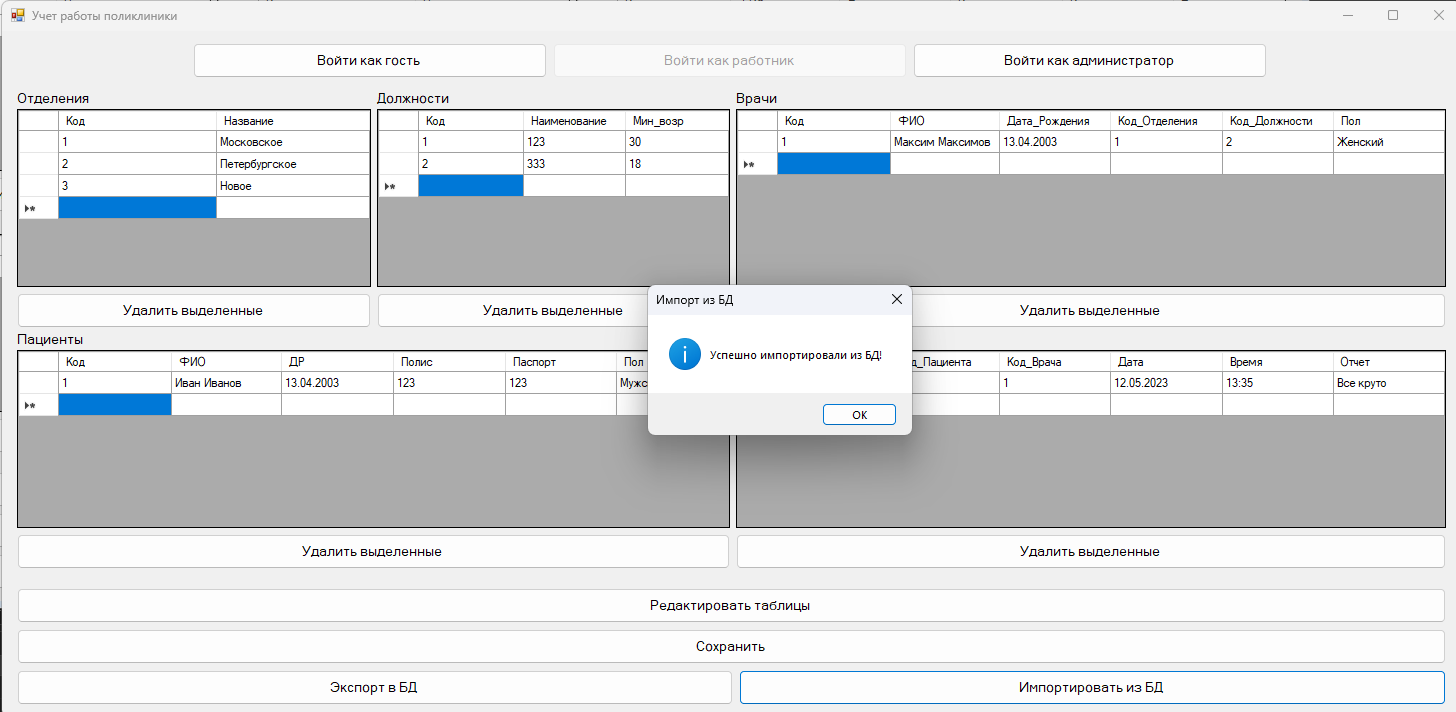


Рисунок 8 – Результат импорта из БД

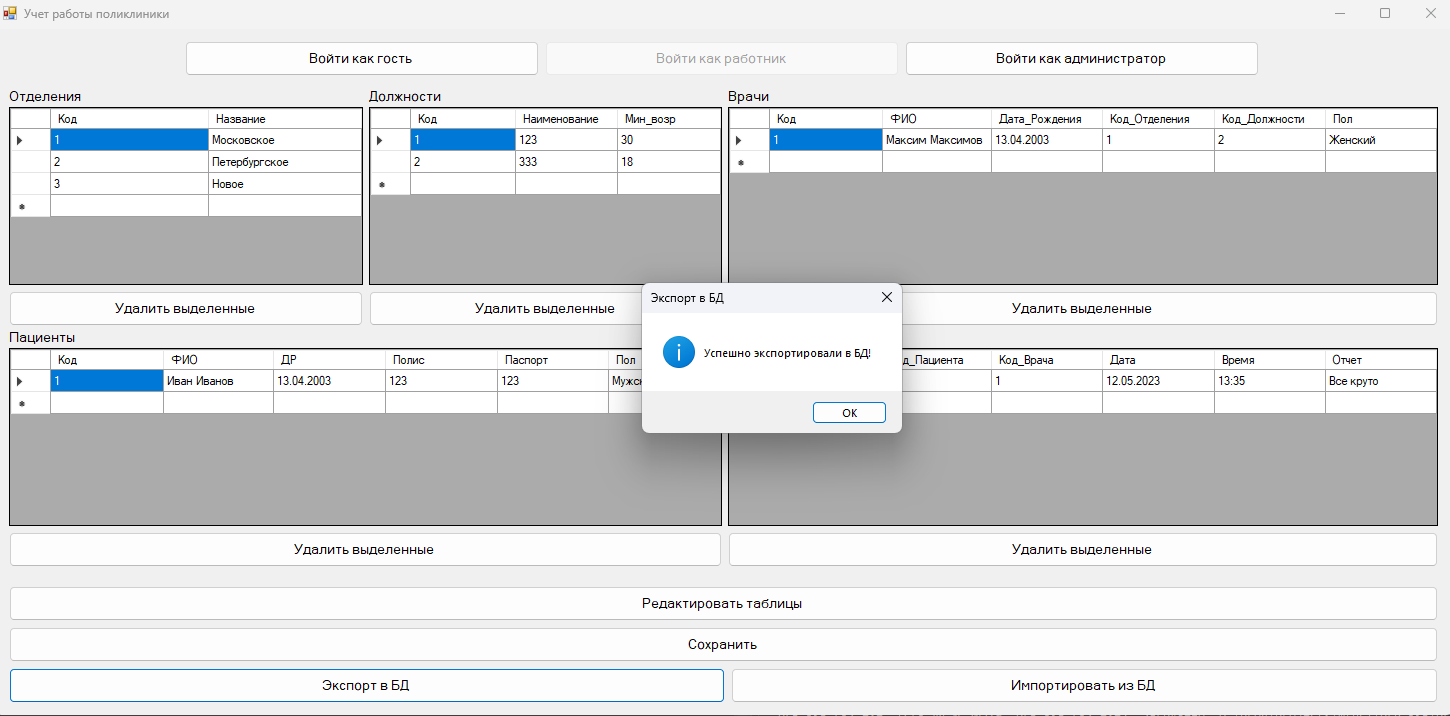


Рисунок 9 – Результат экспорта в БД