МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка базы данных фитнес приложения с реализацией технологии средства диагностики Oracle»

Выполнил студент Саков Евгений Владимирович

(Ф.И.О.)

Руководитель проекта Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: Блинова Е.А

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: Блинова Е.А

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

# Реферат

Курсовая работа изложена на 29 страницах, которые включают в себя 6 разделов, 3 подразделов, введение, архитектуру, реализацию, руководство пользователя, руководство тренера, заключение, используемую литературу и приложения. В работу добавлено 12 изображения.

В курсовой работе разработана модель базы данных, которая состоит из 10 основных таблиц. В базу данных включено 3 табличных пространства, 1 профиля безопасности, 2 роли, 44 процедуры, 1 пользователя.

Используемая СУБД – Oracle Database 12c.

Изучаемая технология – средства диагностики Oracle.

Основной целью курсовой работы является изучение и применение технологий средства диагностики Oracle.

Содержание

[Реферат 2](#_Toc59113133)

[Введение 4](#_Toc59113134)

[1. Постановка задачи 5](#_Toc59113135)

[1.1 Обзор и анализ прототипов 5](#_Toc59113136)

[2. Архитектура базы данных 8](#_Toc59113137)

[3. Описание процедур импорта и экспорта 10](#_Toc59113138)

[3.1. Процедура экспорта 10](#_Toc59113139)

[3.2. Процедура импорта 10](#_Toc59113140)

[5. Руководство пользователя 14](#_Toc59113141)

[6. Руководство тренера 17](#_Toc59113142)

[Заключение 19](#_Toc59113143)

[Список использованных источников 20](#_Toc59113144)

[ПРИЛОЖЕНИЕ А 21](#_Toc59113145)

[SQL-скрипты создания таблиц\ 21](#_Toc59113146)

[ПРИЛОЖЕНИЕ Б 24](#_Toc59113147)

[ПРИЛОЖЕНИЕ В. 27](#_Toc59113148)

# Введение

Современное общество невозможно представить без компьютера. Компьютеры настолько широко и глубоко внедрились в нашу жизнь, что очень трудно назвать какую-либо сферу деятельности человека, где бы они ни использовались. В связи с этим серьезные требования предъявляются как к аппаратной части современных компьютеров, так и к используемому программному обеспечению. В основном именно программное обеспечение, или, иными словами, программные продукты, обеспечивают возможность широкого использования компьютеров. Стоит лишь переустановить программное обеспечение компьютера или добавить какой-либо новый программный продукт, и можно будет решать на этом компьютере совершенно новые задачи.

Следовательно, используемые программные продукты должны соответствовать определенным критериям, обеспечивающим надежность работы компьютера и удобство работы пользователя.

В соответствии с заданием курсового проектирования следует создать базу данных (БД) «Фитнес-приложение» с использование программного проекта ORACLE DATABASE 12C.

Сегодня трудно себе представить сколько-нибудь значимую информационную систему, которая не имела бы в качестве основы или важной составляющей базу данных. Концепции и технологии БД складывались постепенно и всегда были тесно связаны с развитием систем автоматизированной обработки информации.

Проектирование базы данных «Фитнес-приложение» ведется для просмотра и учета различной информации.

# Постановка задачи

# Обзор и анализ прототипов

В данном обзоре я буду сравнивать наиболее популярные приложения с аналогичной тематикой, чтобы просмотреть достоинства и функционал всех приложений, а затем сделать свое, избегая минусов предыдущих.

Первый объект обзора - программа «SWORKIT» (рис.1). [1]

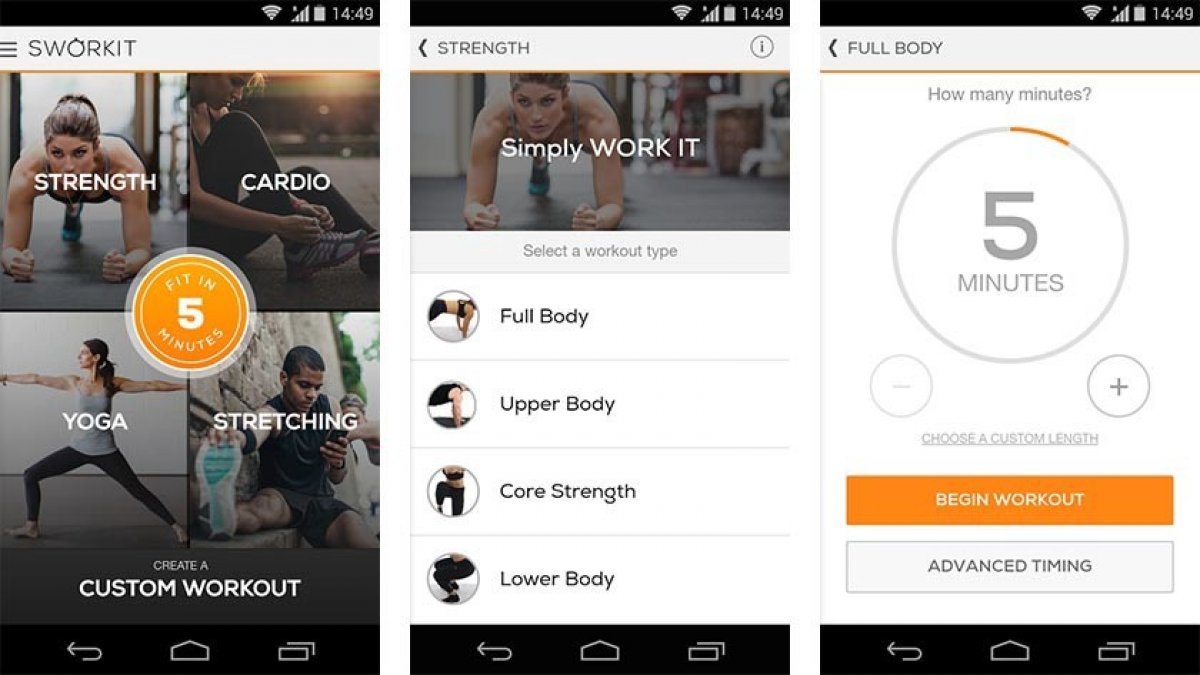


Рис. 1 – функционал приложения

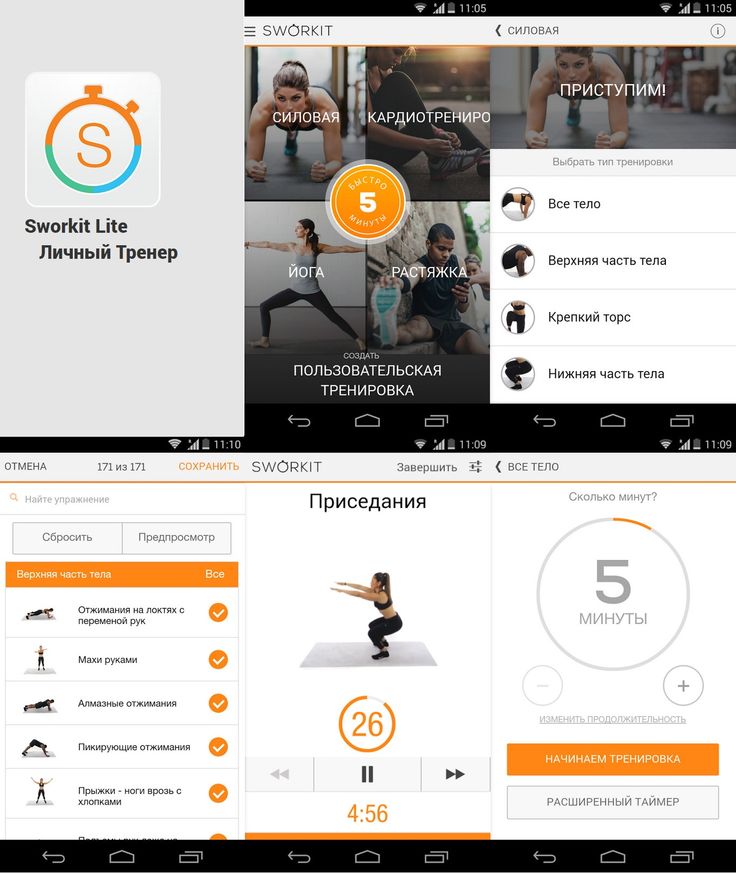


Рис. 2 – выполнение тренировок

Сразу можно выделить:

* Огромный функционал (рис.1);
* Присутствие секундомера поможет тренироваться правильно;
* Список упражнений, которые нужно сделать (рис. 2);
* Присутствует пример выполнения, что поможет новичкам в выполнении;

Подводя итоги, можно утверждать, что все сделано максимально удобно и приемлемо для пользователя в целом, является достойным примером.

Второй объект обзора - «Jefit» (рис. 3). [2]

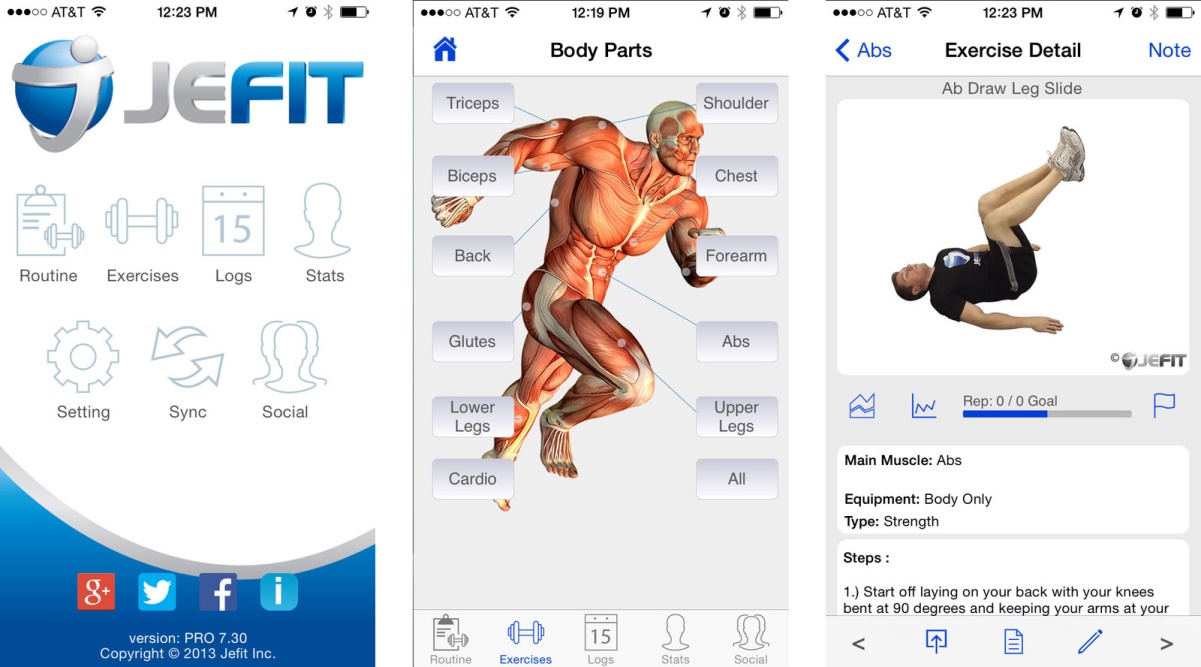


Рис.3 – функционал и главное меню

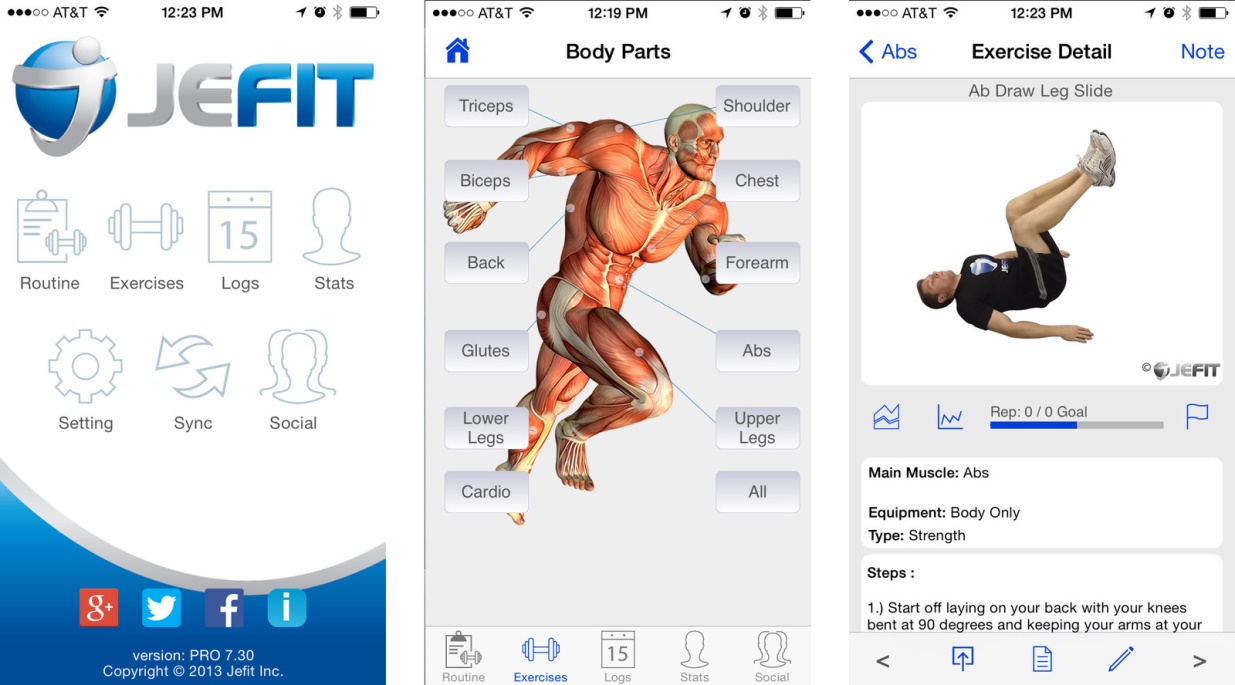


Рис.4 – выбор упражнения по виду мышц

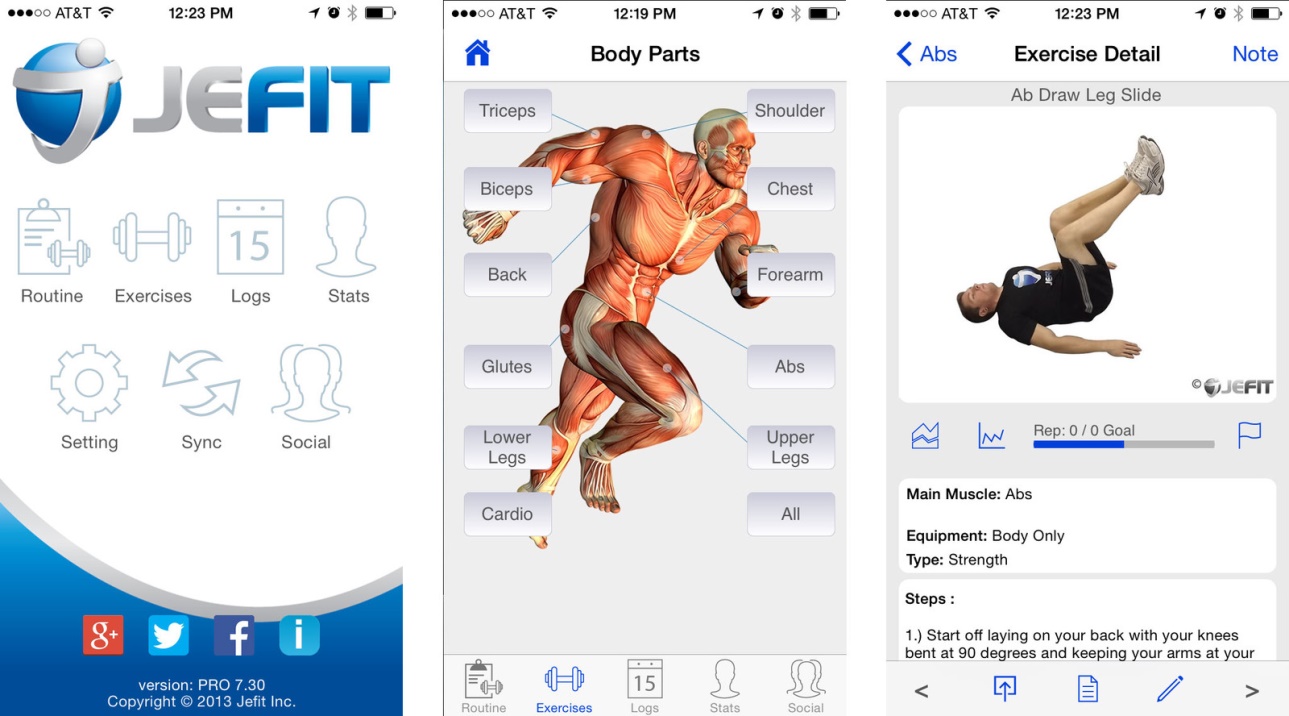


Рис.5 – пример выполнения

Сразу можно выделить:

-весь функционал перед пользователем (рис. 3);

-увеличивает познания в анатомии;

-удобно выбрать упражнение для группы мышц (рис. 4).

-пример выполнения и описание (рис. 5).

По сравнению с предыдущим аналогом, данный является более удобным и достаточно функциональным. Главная страница отображает весь функционал.

Можно отметить, что при работе остаются приятные ощущения, имеются практически все необходимые условия для выполнения задач пользователя.

Существует ряд аналогов, превосходящих мой программный продукт по объему функционала, однако моё приложение является десктопным и не требует доступа к Интернету для просмотра и администрирования различной информации.

# Архитектура базы данных

* 1. **Проектирование базы данных**

Для хранения и предоставления данных пользователю была разработана архитектура базы данных, которая содержит в себе девять таблиц. Диаграмма базы данных представлена на рисунке 2.

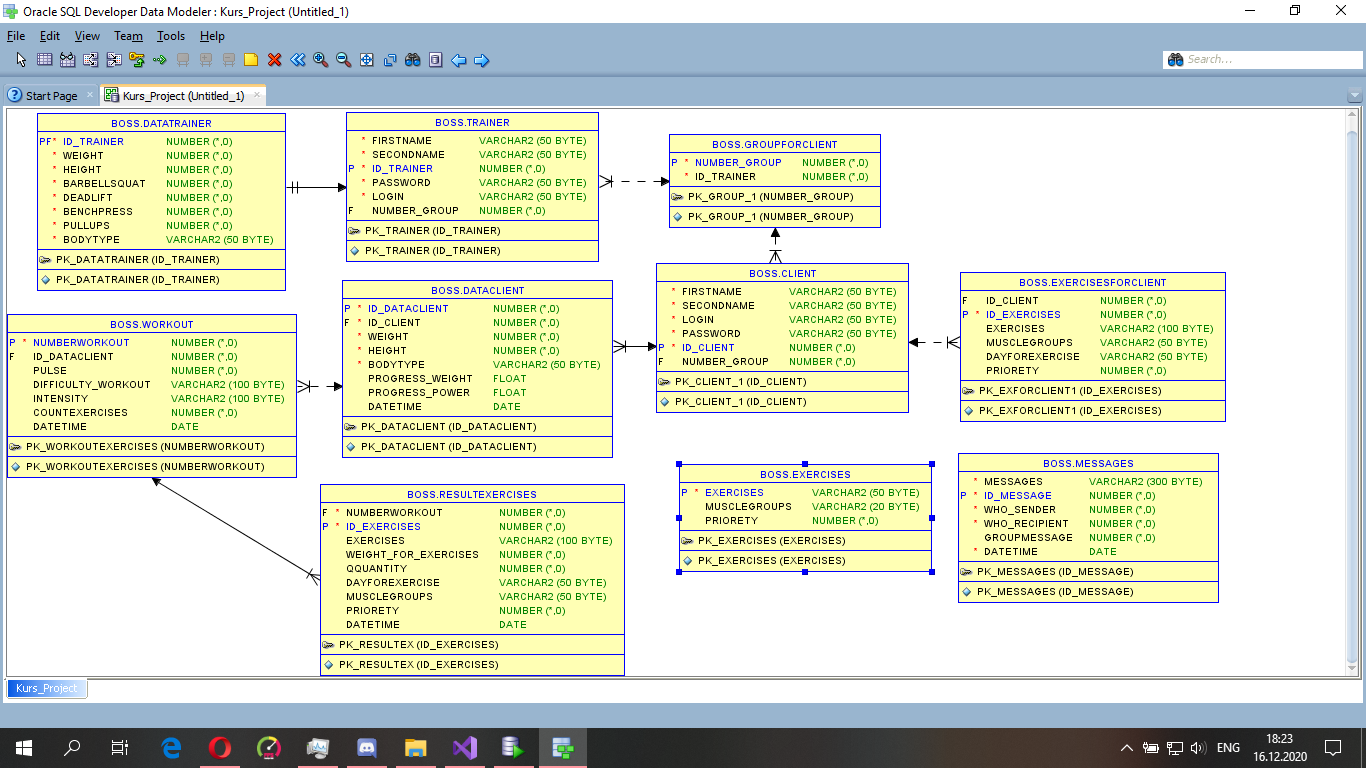


Рисунок 1. – Диаграмма базы данных

Таблица Client – содержит в себе личные данные о пользователе.

Таблица DataClient – содержит информацию о пользователе.

Таблица ExercisesForClient – содержит программу тренировок для пользователя.

Таблица ResultExercises – содержит результаты тренировок пользователей пользователя.

Таблица Group – содержит информацию о группах.

Таблица Trainer – содержит личную информацию о тренере.

Таблица DataTrainer – содержит информацию о тренере.

Таблица Messages – содержит информацию сообщениях пользователя и тренера.

Таблица Exercises – содержит информацию об упражениях.

Таблица Workout – содержит информацию о тренировках пользователя.

* 1. **Средства диагностики Oracle**

Средства диагностики Oracle реализованы при помощи трассировки.

--трассировка

alter session set sql\_trace=true;

exec DBMS\_TRACE.SET\_PLSQL\_TRACE(1);

--exec DBMS\_TRACE.SET\_PLSQL\_TRACE (32);

alter session set tracefile\_identifier='UnicalString4';

select tracefile from v$process where tracefile like '%UnicalString4%';

--в sql plus для преобразования трассировачного файла

TKPROF E:\app\ora\_install\_user\diag\rdbms\orcl\orcl\trace\orcl\_ora\_12056\_UnicalString2.trc text2.txt sys=no explain=y

select \* from BOSS.trainer;

select \* from BOSS.Datatrainer;

insert into trainer values('бот','бот',1,'123','user',null);

exec DBMS\_TRACE.CLEAR\_PLSQL\_TRACE;

alter session set sql\_trace=false;

Этот код работает по следующему принципу, вначале нужно запустить трассировку, чтобы можно было работать с трассировочными файлами, указали уровень трассировки, далее создаётся автоматически трассировочный файл мы меняем его название для уникальности и убеждаемся что он создался, после чего совершаем операции вывода для того чтобы они записались в трассировочный файл. Готовый трассировочный файл мы преобразуем в более читабельный вариант при помощи TKPROF. Когда трассировка не нужна выключаем ее.

# Описание процедур импорта и экспорта

Для экспорта и импорта данных были написаны скрипты которые используют пакеты DBMS\_XMLGEN и UTL\_FILE.

## 3.1. Процедура экспорта

Часть скрипта создания процедуры экспорта представлен на рисунке 3.1.

DECLARE

cSQLclient VARCHAR2(32000) := 'select \* from client';

resultclient clob;

qryCtx1 dbms\_xmlgen.ctxHandle;

f1 utl\_file.file\_type;

begin

qryCtx1 := DBMS\_XMLGEN.newContext(cSQLclient);

DBMS\_XMLGEN.setRowSetTag(qryCtx1, 'client');

DBMS\_XMLGEN.setRowTag(qryCtx1, 'columns');

resultclient := DBMS\_XMLGen.getXML(qryCtx1);

DBMS\_XMLGen.CloseContext(qryCtx1);

begin

l\_amount := 32767;

l\_pos :=1;

f1 := utl\_file.fopen('EXTRACT\_DIR', 'EXPORT\_CLIENT\_TO\_XML.xml', 'w');

LOOP

DBMS\_LOB.read (resultclient, l\_amount, l\_pos, l\_buffer);

utl\_file.put\_line(f1, l\_buffer);

l\_pos := l\_pos + l\_amount;

END LOOP;

EXCEPTION

WHEN OTHERS THEN

utl\_file.fclose(f1);

end;

END;

Для экспорта используется .xml – файл, откуда экспортируются данные.

## 3.2. Процедура импорта

Скрипт создания процедуры импорта представлен на рисунке 3.2.

DECLARE

cSQLclient CLOB;

line VARCHAR2(100);

f utl\_file.file\_type;

x pls\_integer := 0;

begin

begin

f := utl\_file.fopen('EXTRACT\_DIR', 'EXPORT\_CLIENT\_TO\_XML.xml', 'R');

loop

utl\_file.GET\_LINE(f,line);

cSQLclient:=CONCAT(cSQLclient,line);

x:=x+1;

exit when x=-5;

end loop;

exception

when others

then dbms\_output.put\_line(sqlerrm);

end;

utl\_file.fclose(f);

insert into xml\_tmp values (xmltype(cSQLclient));

insert into client

select FIRSTNAME,SECONDNAME,LOGIN,PASSWORD,ID,NUMBER\_GROUP

from xml\_tmp x, xmltable('/client/columns'

passing x.xml\_data columns FIRSTNAME VARCHAR2(50 BYTE), SECONDNAME VARCHAR2(50 BYTE),

LOGIN VARCHAR2(50 BYTE),PASSWORD VARCHAR2(50 BYTE),ID NUMBER,NUMBER\_GROUP NUMBER

) xt;

Полный листинг процедуры импорта и экспорта находится в Приложении Б. Импорт и экспорт построчно считывали данные так как данные были большие то использовался тип clob который может хранить в себе огромное количество данных.

1. **Описание процедур для работы с базой данных**

Весь доступ к данным осуществляется

Процедура добавления пользователя в базу данных

create or replace

PROCEDURE REG\_CLIENT(

client\_id number,

client\_firstname VARCHAR,

client\_secondname VARCHAR,

client\_login VARCHAR,

client\_password VARCHAR,

client\_group number)

AS

BEGIN

INSERT INTO client(id,firstname,secondname,login,password,number\_group) values(client\_id,client\_firstname,client\_secondname,client\_login,client\_password,client\_group);

END REG\_CLIENT;

Процедура принимает в качестве параметра идентификатор, имя ,фамилию, логин, пароль и номер группы, после добавляет пользователя в базу данных.

Процедура добавления результата

create or replace

PROCEDURE INSERT\_RESULT\_EXERCISES(

NUMBEREX number,

FIRST\_DAY VARCHAR,

FIRST\_DAY\_WEIGHT number,

FIRST\_DAY\_QUANTITY number,

SECOND\_DAY VARCHAR,

SECOND\_DAY\_WEIGHT number,

SECOND\_DAY\_QUANTITY number,

THIRD\_DAY VARCHAR,

THIRD\_DAY\_WEIGHT number,

THIRD\_DAY\_QUANTITY number,

ID\_CLIENT number,

DATETIME DATE)

AS

BEGIN

INSERT INTO RESULTEXERCISES values(NUMBEREX,FIRST\_DAY,FIRST\_DAY\_WEIGHT,FIRST\_DAY\_QUANTITY,

SECOND\_DAY,SECOND\_DAY\_WEIGHT,SECOND\_DAY\_QUANTITY,

THIRD\_DAY,THIRD\_DAY\_WEIGHT,THIRD\_DAY\_QUANTITY,

ID\_CLIENT,DATETIME);

END INSERT\_RESULT\_EXERCISES;

Процедура для получения данных об упражнениях пользователя участника.

create or replace

PROCEDURE GET\_INFO\_EXERCISESFORCLIENT

(exerforclient OUT SYS\_REFCURSOR )

AS

BEGIN

OPEN exerforclient FOR SELECT \* FROM EXERCISESFORCLIENT;

END;

Листинги других процедур находятся в приложения Б.

Практически все процедуры добавления действуют по одному принципу.

Получаем значения для добавление в таблицу, если это нужно, проверяем есть ли в таблице данные и только тогда добавляем в базу данных.

# Руководство пользователя

Поскольку в базе данных нет разделения на пользователей, то данное руководство является общим.

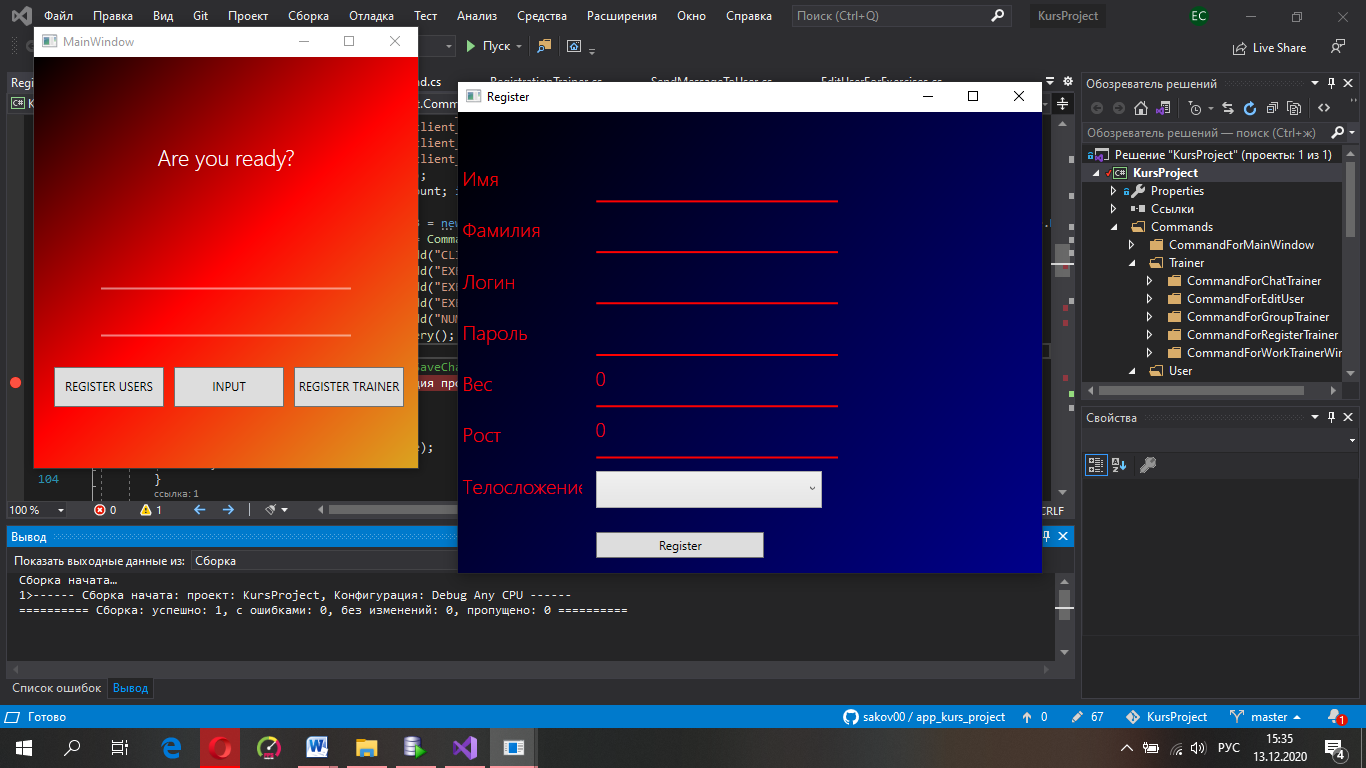


Рисунок 5.1.1 – Регистрация пользователя

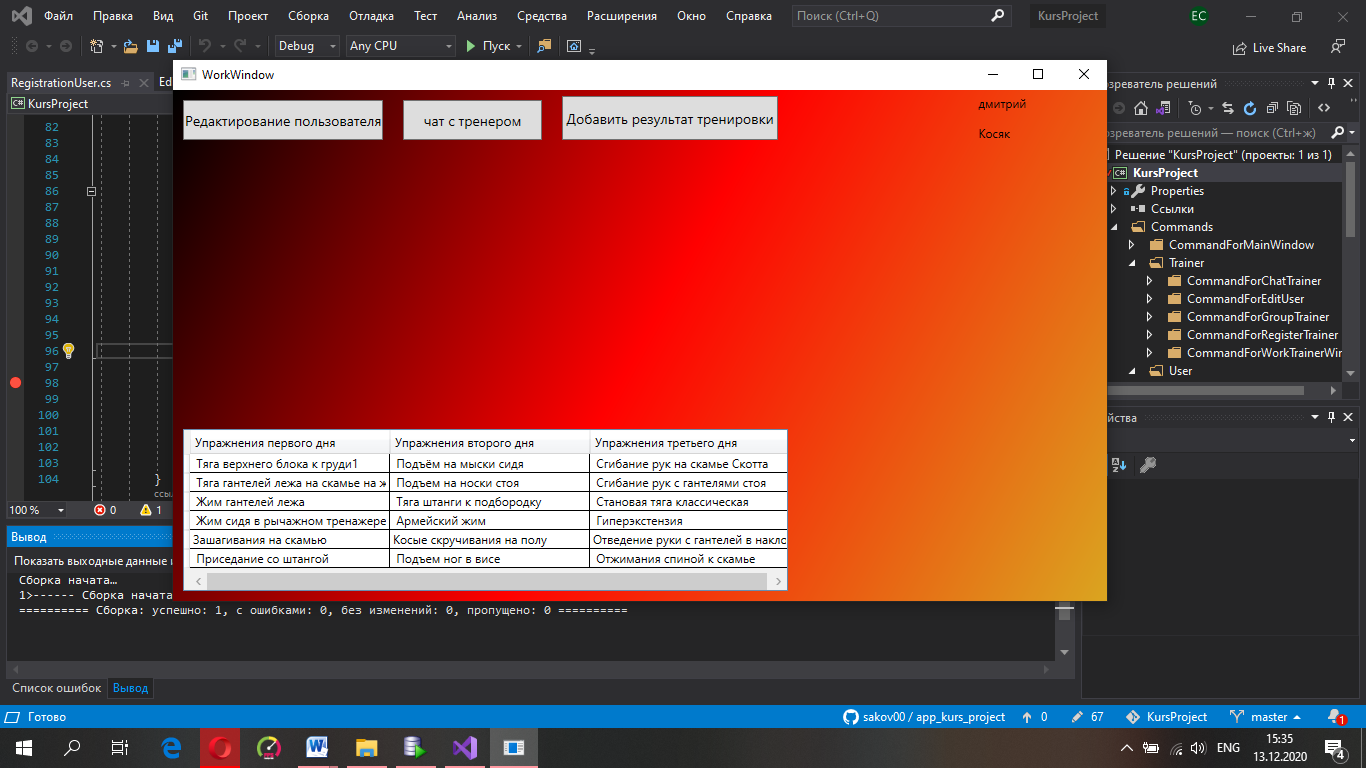


Рисунок 5.1.2 – Панель приложения пользователя

Пользователь программы может перейти по любой из вкладок и выбрать ему нужную информацию для изменения или просмотра. Снизу есть таблица в которой написана программа тренировок на день.

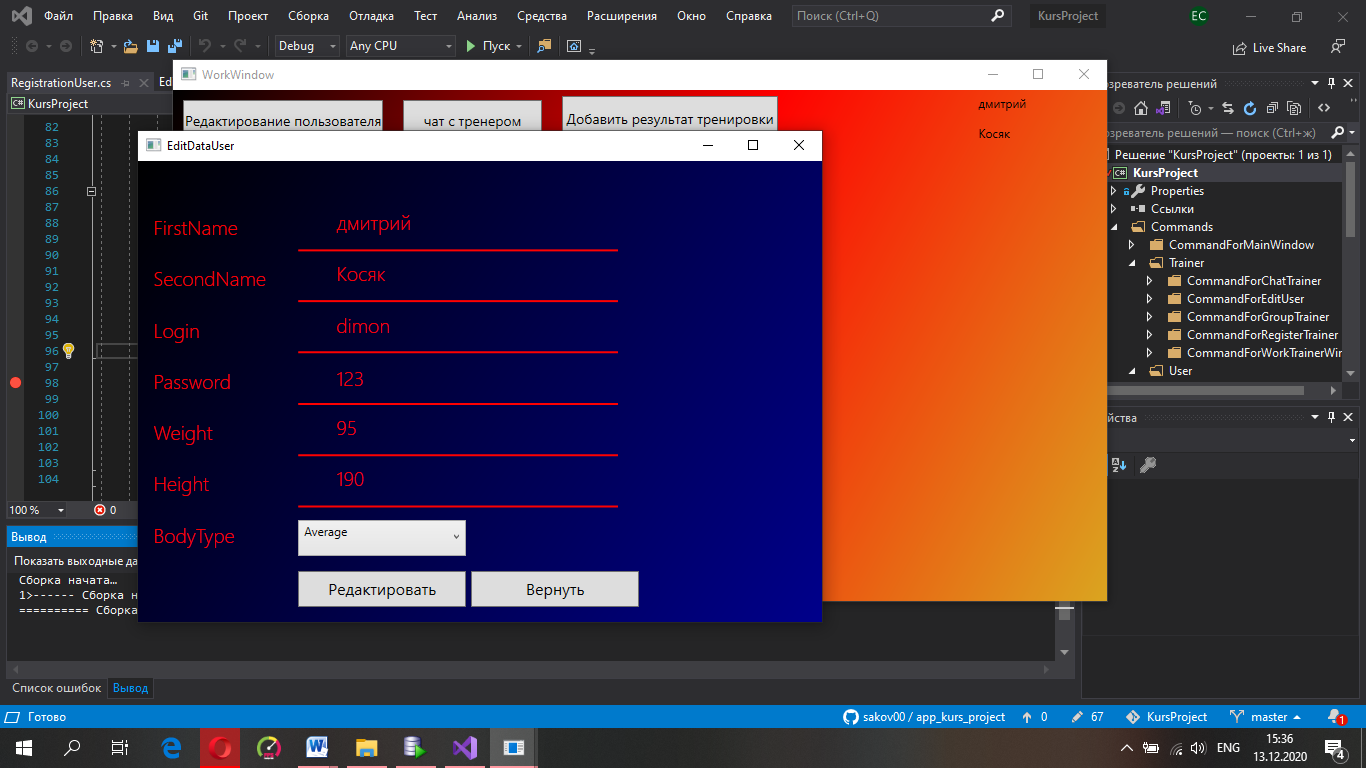


Рисунок 5.1.3 – редактирование данных пользователя

Данные которые будут изменены, вернуть их можно если только вы не нажали на редактировать так как если нажать то произойдут изменения в БД и нужно будет снова менять.

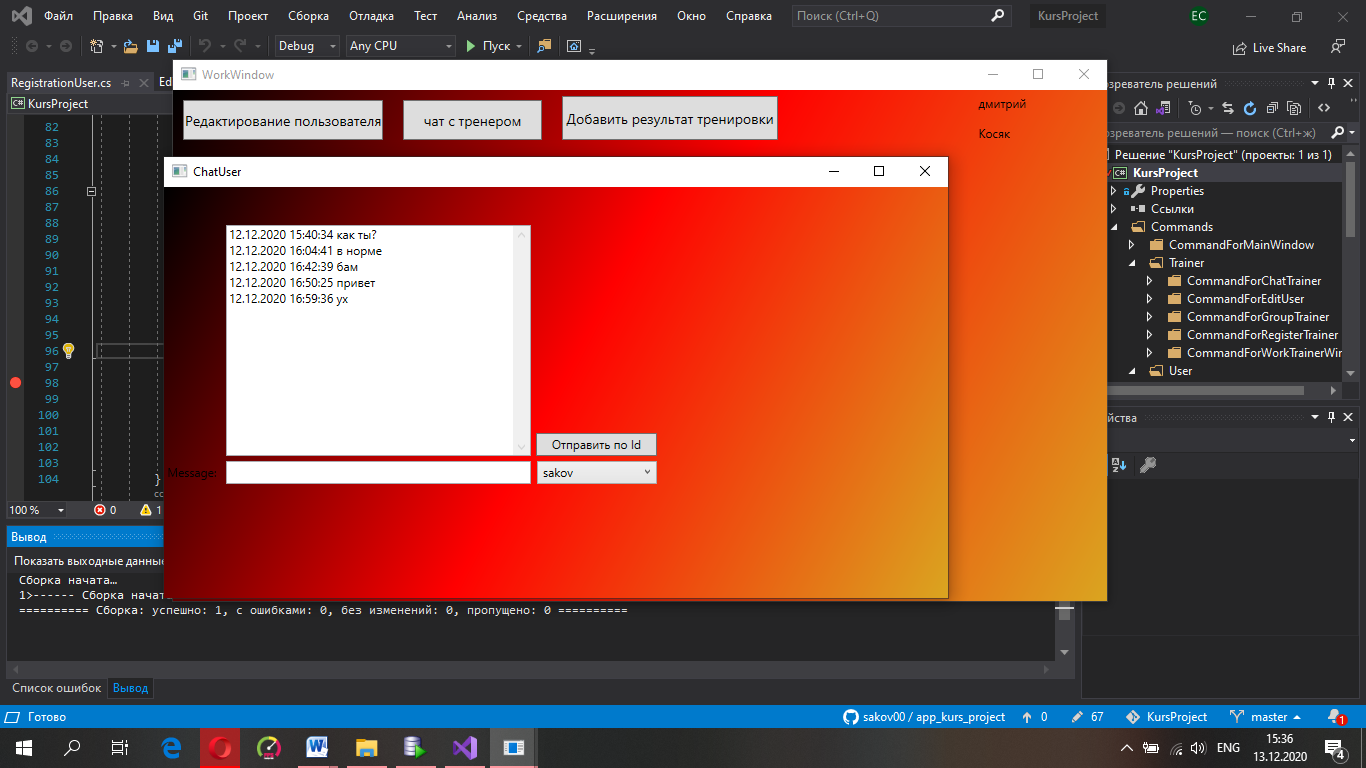


Рисунок 5.1.4 – чат пользователя с тренером

Пользователь может общаться только с теми кто состоит с ним в группе.

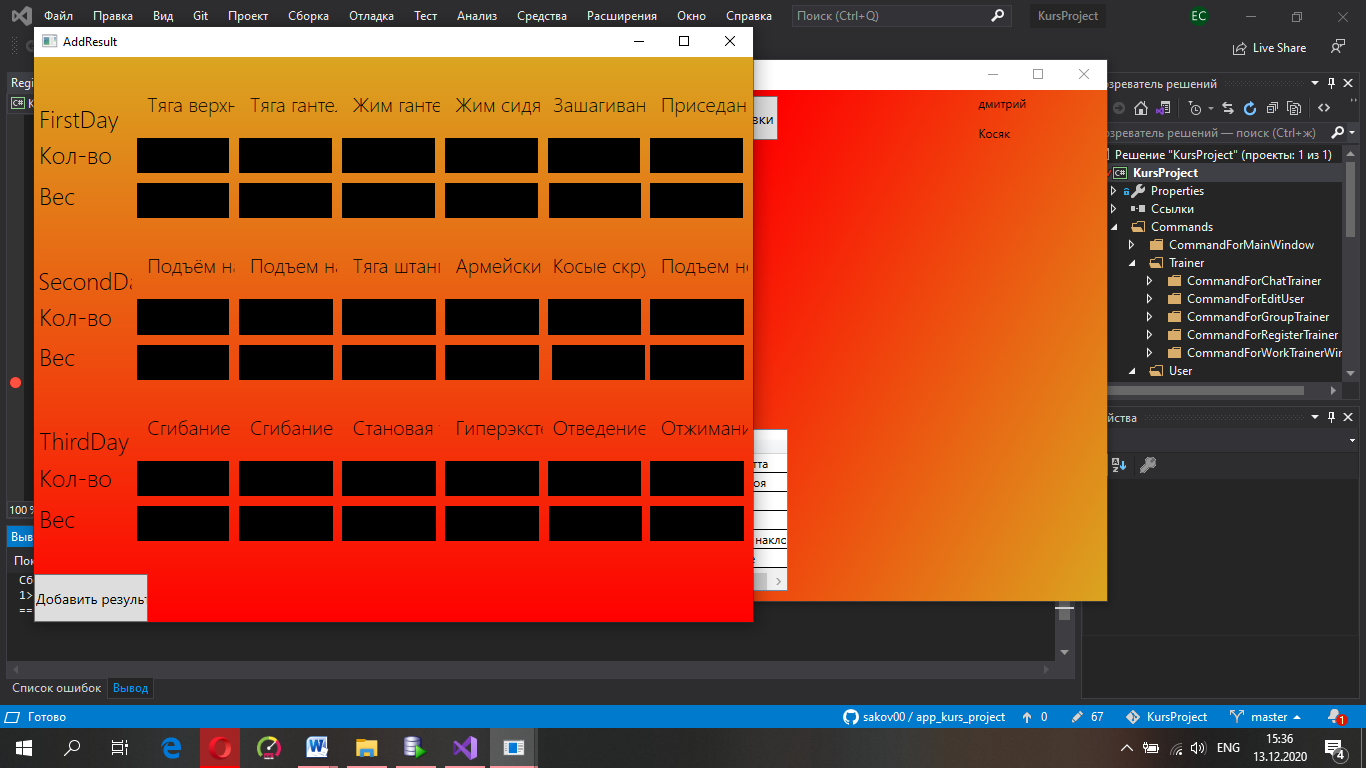


Рисунок 5.1.5 – добавить результат тренировок

Все операции и получение информации с БД выполняются через процедуры и никак иначе. Приложение написано с помощью WPF который использует C# на данном языке была написана приличная часть логики которая работает с данными.

# Руководство тренера

Поскольку в базе данных нет разделения на пользователей, то данное руководство является общим.

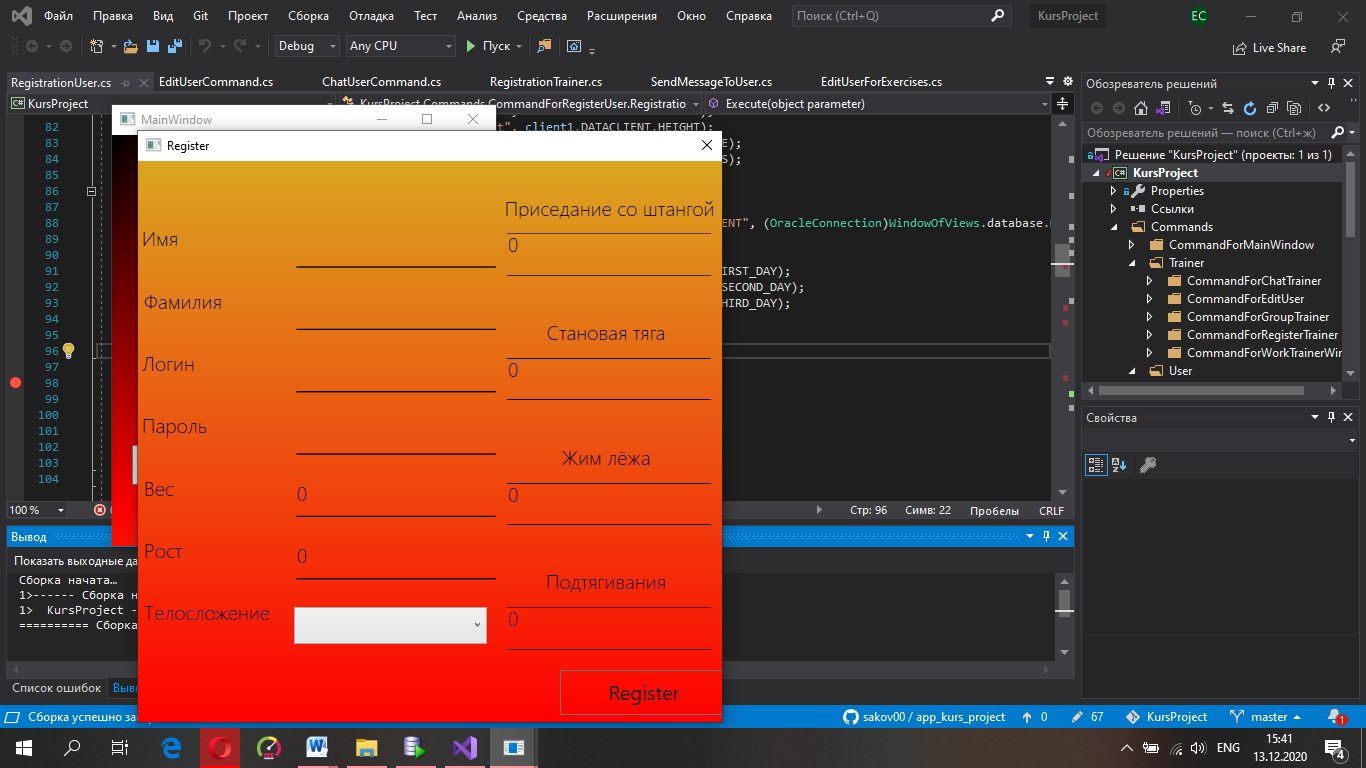


Рисунок 5.1.1 – Регистрация тренера

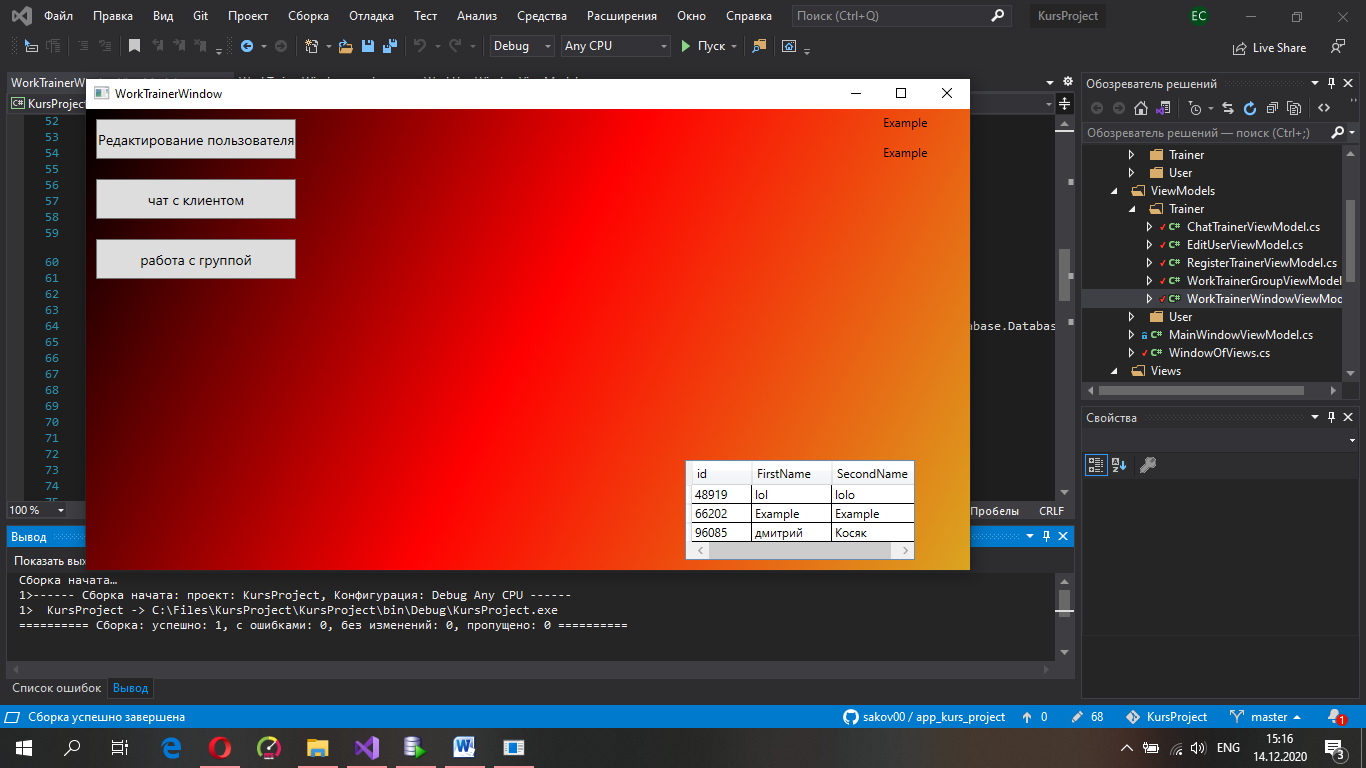


Рисунок 5.1.2 – Панель приложения тренера

Тренер программы может перейти по любой из вкладок и выбрать ему нужную информацию для изменения или просмотра. Снизу указаны все пользователи.

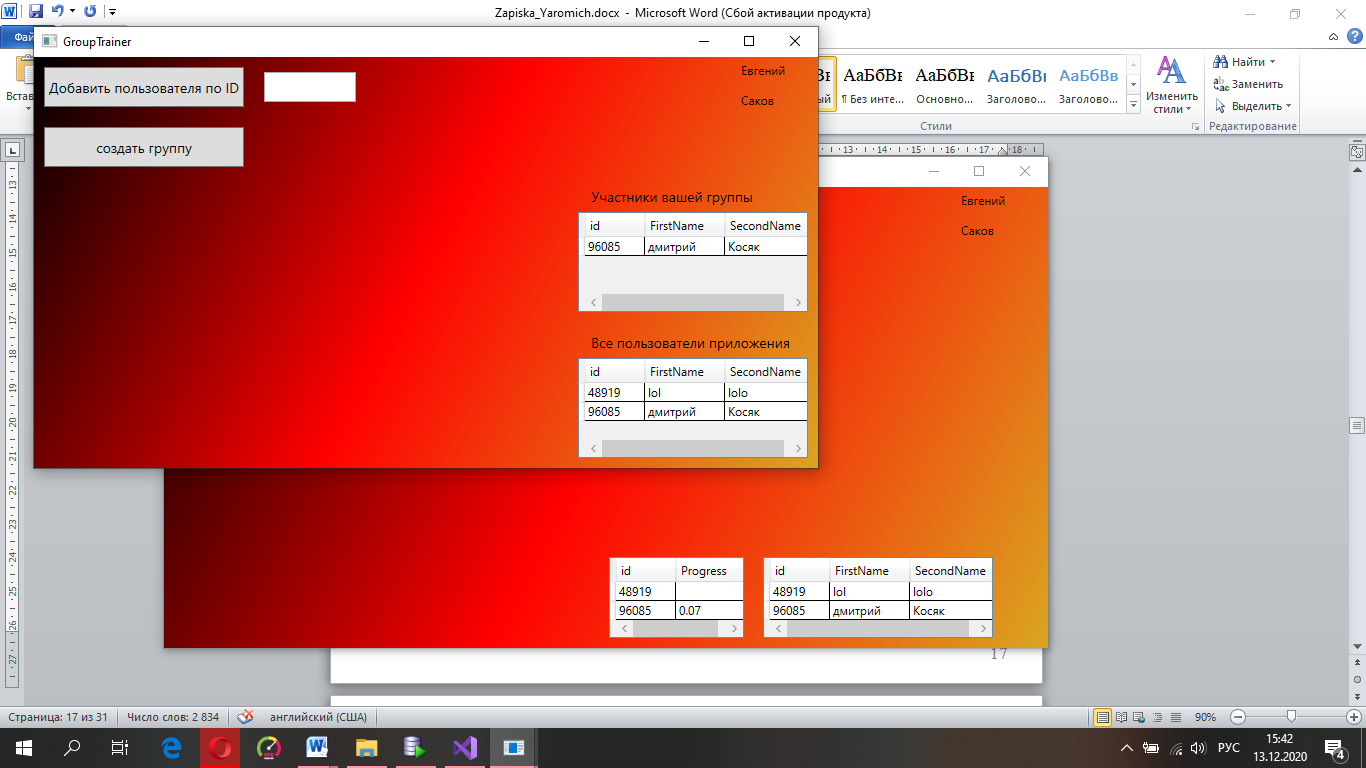


Рисунок 5.1.3 – создание группы тренером и добавление в нее пользователей

Группа нужна для того чтобы тренер мог взаимодействовать с пользователями в ней, то есть общаться в чате и изменять программу тренировок

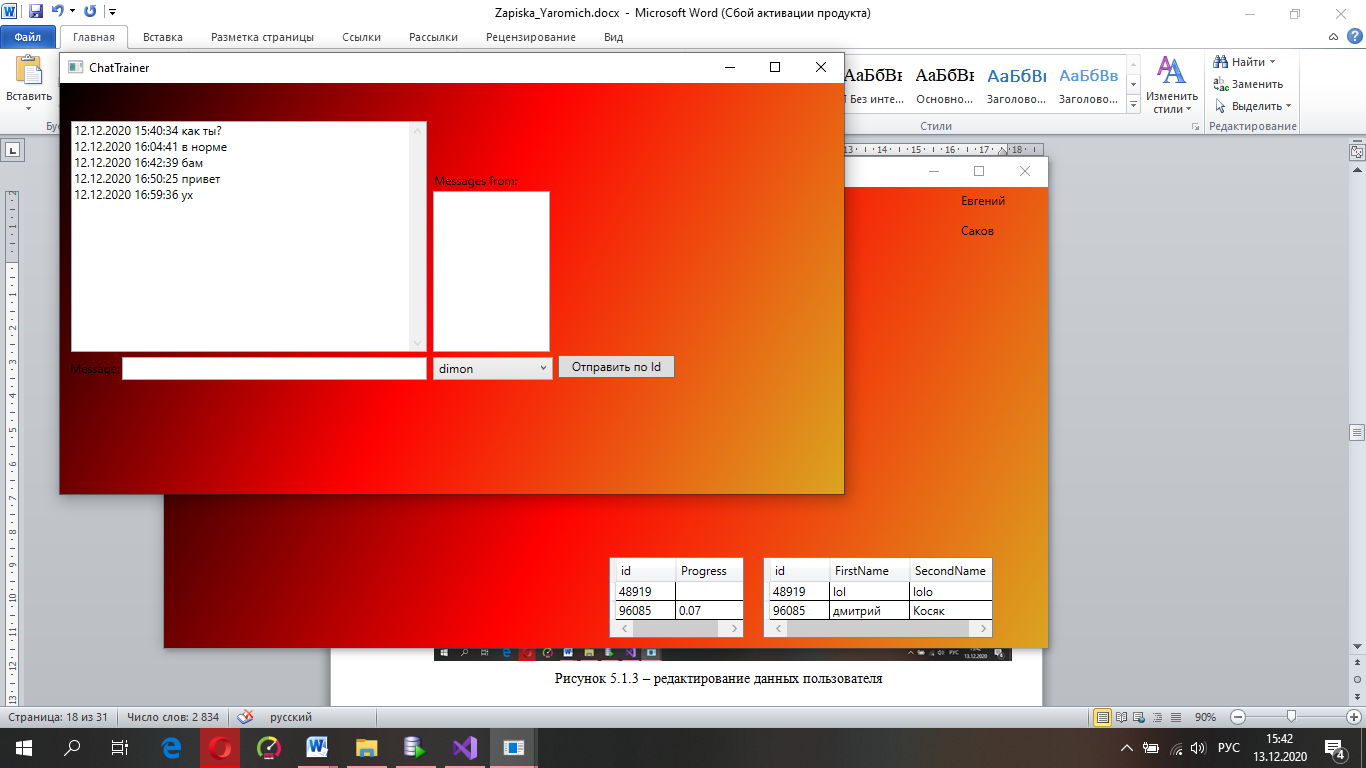


Рисунок 5.1.4 – чат тренера с пользователем

Тренер может общаться только с теми кто состоит в его группе

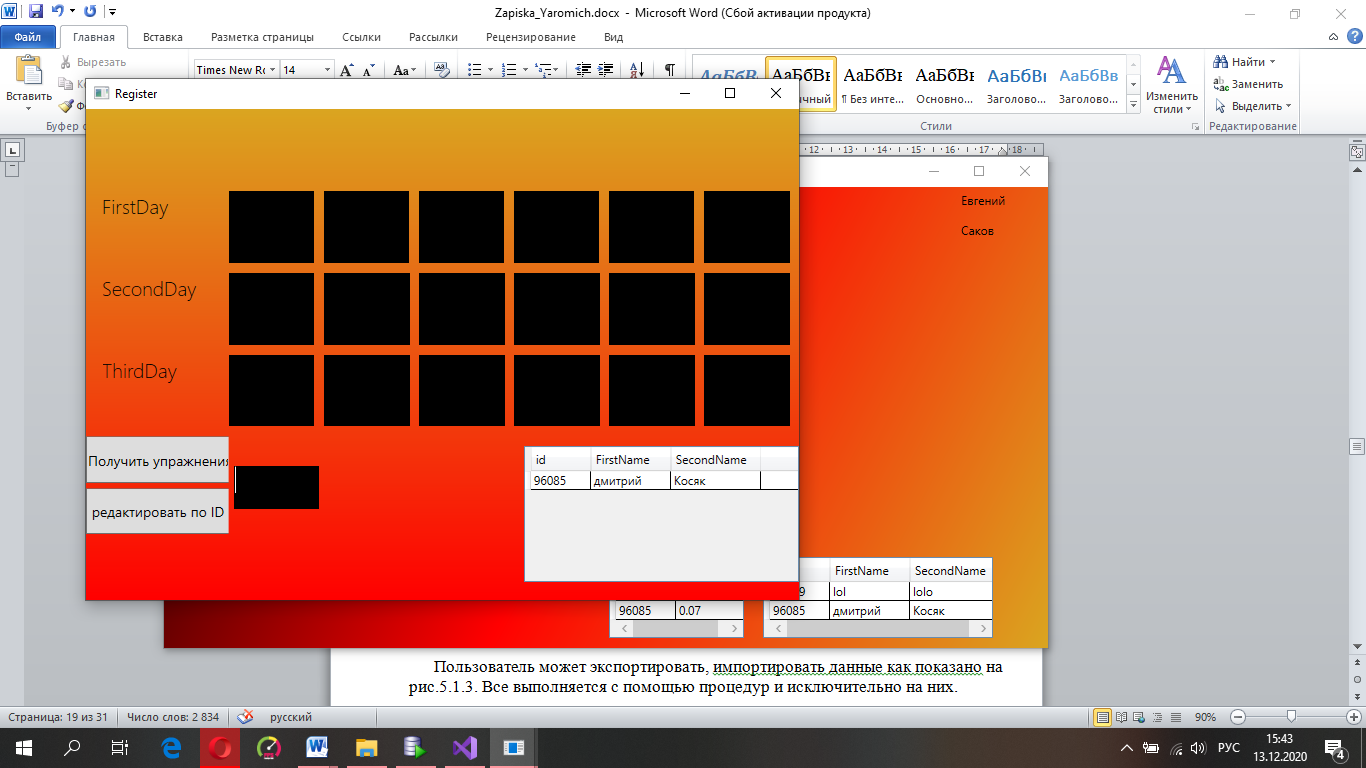


Рисунок 5.1.5 – изменить программу тренировок пользователя

Все операции и получение информации с БД выполняются через процедуры и никак иначе. Приложение написано с помощью WPF который использует C# на данном языке была написана приличная часть логики которая работает с данными.

# Заключение

В заключение курсовой работы необходимо сделать вывод о результатах проделанной работы. В данном курсовом проекте была разработана база данных «Фитнес приложение». Программа полностью автоматизирует труд организаторов ОИ и организует надежное хранение и обработку данных.

В соответствии с полученным результатом работы базы данных можно сделать вывод, что разработанная БД работает верно, а требования технического задания выполнены в полном объеме.

Выполненные цели:

* Регистрация и авторизация пользователя и тренера, с указанием всех данных
* Чат тренера и пользователя

Тренер:

* Редактирование программы тренировок пользователя
* Создание группы пользователей

Пользователь:

* вывод программы тренировок из базы данных, будет выводиться определённое количество упражнений на один день.
* Добавить результат тренировок, ввести количество повторений и если нужно вес.
* Редактирование пользователя, редактировать можно только персональные данные.

При написании курсовой работы был использован API-интерфейс WPF, что позволило сделать приложение более красивым и упростило разработку в разы.

Также стоит отметить то, что при написании курсового проекта для достижения желаемых результатов, было изучено множество возможностей технологии .NET.

# Список использованных источников

[1] METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com Дата доступа: 10.12.2019

[2] Oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru Дата доступа 10.12.2019

[3] stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com Дата доступа: 12.12.2019

[4] Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/tdddg_procedures.htm> Дата доступа: 10.11.2019

[5] oracle-patches.com Сайт об Oracle и всех его возможностях / Режим доступа: https://oracle-patches.com Дата доступа: 10.12.2019

# ПРИЛОЖЕНИЕ А

**SQL-скрипты создания таблиц**

CREATE TABLE Exercises(

Exercises varchar(50) NOT NULL,

MuscleGroups varchar(20) NULL,

Priorety int NULL,

CONSTRAINT PK\_Exercises PRIMARY KEY (Exercises));

CREATE TABLE Messages(

messages varchar(300) NOT NULL,

id\_Message int NOT NULL,

Who\_sender int NOT NULL,

Who\_recipient int NOT NULL,

GroupMessage int NULL,

DateTime date NOT NULL,

CONSTRAINT PK\_Messages PRIMARY KEY (id\_Message));-- Date=>DateTime

CREATE TABLE GroupForClient(-->>GroupForClient

Number\_Group int NOT NULL,

Id\_Trainer int NOT NULL,

CONSTRAINT PK\_Group\_1 PRIMARY KEY (Number\_Group));

CREATE TABLE Trainer(

FirstName varchar(50) NOT NULL,

SecondName varchar(50) NOT NULL,

Id\_Trainer int NOT NULL,

Password varchar(50) NOT NULL,

Login varchar(50) NOT NULL,

Number\_Group int NULL,

CONSTRAINT PK\_Trainer PRIMARY KEY (Id\_Trainer),

CONSTRAINT FK\_Trainer\_Group FOREIGN KEY(Number\_Group)

REFERENCES GroupForClient (Number\_Group));

CREATE TABLE DataTrainer(

Id\_Trainer int NOT NULL,

Weight int NOT NULL,

Height int NOT NULL,

BarbellSquat int NOT NULL,

Deadlift int NOT NULL,

BenchPress int NOT NULL,

Pullups int NOT NULL,

BodyType varchar(50) NOT NULL,

CONSTRAINT PK\_DataTrainer PRIMARY KEY (Id\_Trainer),

CONSTRAINT FK\_DataTrainer\_Trainer FOREIGN KEY(Id\_Trainer)

REFERENCES Trainer (Id\_Trainer));

/\*введение средаства диагностики схема бд

записка

проект\*/

CREATE TABLE Client(

FirstName varchar(50) NOT NULL,

SecondName varchar(50) NOT NULL,

Login varchar(50) NOT NULL,

Password varchar(50) NOT NULL,

Id\_Client int NOT NULL,

Number\_Group int NULL,

CONSTRAINT PK\_Client\_1 PRIMARY KEY (Id\_Client),

CONSTRAINT FK\_Client\_Group1 FOREIGN KEY(Number\_Group)

REFERENCES GroupForClient (Number\_Group));

CREATE TABLE ExercisesForClient(

Id\_Client int NULL,

Id\_Exercises int NULL,

Exercises varchar(100) NULL,

MuscleGroups varchar(50) NULL,

DayForExercise varchar(50) NULL,

Priorety int NULL,

CONSTRAINT PK\_ExForClient1 PRIMARY KEY (Id\_Exercises),

CONSTRAINT FK\_ExForClient\_Workout1 FOREIGN KEY(Id\_Client)

REFERENCES Client (Id\_Client));-- Number=>NumberEx

CREATE TABLE DataClient(

Id\_DataClient int NOT NULL,

Id\_Client int NOT NULL,

Weight int NOT NULL,

Height int NOT NULL,

Bodytype varchar(50) NOT NULL,

Progress\_Weight float NULL,

Progress\_Power float NULL,

DateTime date NULL,

CONSTRAINT PK\_DataClient PRIMARY KEY (Id\_DataClient),

CONSTRAINT FK\_DataClient\_Client FOREIGN KEY(Id\_Client)

REFERENCES Client (Id\_Client));

CREATE TABLE Workout(

NumberWorkout int NOT NULL,

Id\_DataClient int NULL,

Pulse int NULL,

difficulty\_Workout varchar(100) NULL,

intensity varchar(100) NULL,

CountExercises int NULL,

DateTime date NULL,

CONSTRAINT PK\_WorkoutExercises PRIMARY KEY (NumberWorkout),

CONSTRAINT FK\_WorkoutEx\_ExForClient FOREIGN KEY(Id\_DataClient)

REFERENCES DataClient (Id\_DataClient));

CREATE TABLE ResultExercises(

NumberWorkout int NOT NULL,

Id\_Exercises int NULL,

Exercises varchar(100) NULL,

Weight\_For\_Exercises int NULL,

Qquantity int NULL,

DayForExercise varchar(50) NULL,

MuscleGroups varchar(50) NULL,

Priorety int NULL,

DateTime date NULL,

CONSTRAINT PK\_ResultEx PRIMARY KEY (Id\_Exercises),

CONSTRAINT FK\_ResultEx\_ExForWorkout FOREIGN KEY(NumberWorkout)

REFERENCES Workout (NumberWorkout));-- Number=>NumberEx

ALTER TABLE ResultExercises ADD DateTime date NULL;

# ПРИЛОЖЕНИЕ Б

**Процедуры**

create or replace

PROCEDURE REG\_CLIENT(

client\_id number,

client\_firstname VARCHAR,

client\_secondname VARCHAR,

client\_login VARCHAR,

client\_password VARCHAR,

client\_group number)

AS

BEGIN

INSERT INTO client(id,firstname,secondname,login,password,number\_group) values(client\_id,client\_firstname,client\_secondname,client\_login,client\_password,client\_group);

END REG\_CLIENT;

create or replace

PROCEDURE INSERT\_RESULT\_EXERCISES(

NUMBEREX number,

FIRST\_DAY VARCHAR,

FIRST\_DAY\_WEIGHT number,

FIRST\_DAY\_QUANTITY number,

SECOND\_DAY VARCHAR,

SECOND\_DAY\_WEIGHT number,

SECOND\_DAY\_QUANTITY number,

THIRD\_DAY VARCHAR,

THIRD\_DAY\_WEIGHT number,

THIRD\_DAY\_QUANTITY number,

ID\_CLIENT number,

DATETIME DATE)

AS

BEGIN

INSERT INTO RESULTEXERCISES values(NUMBEREX,FIRST\_DAY,FIRST\_DAY\_WEIGHT,FIRST\_DAY\_QUANTITY,

SECOND\_DAY,SECOND\_DAY\_WEIGHT,SECOND\_DAY\_QUANTITY,

THIRD\_DAY,THIRD\_DAY\_WEIGHT,THIRD\_DAY\_QUANTITY,

ID\_CLIENT,DATETIME);

END INSERT\_RESULT\_EXERCISES;

create or replace

PROCEDURE GET\_INFO\_EXERCISESFORCLIENT

(exerforclient OUT SYS\_REFCURSOR )

AS

BEGIN

OPEN exerforclient FOR SELECT \* FROM EXERCISESFORCLIENT;

END;

create or replace

PROCEDURE EXPORT\_XML\_CLIENT AS

BEGIN

DECLARE

cSQLclient VARCHAR2(32000) := 'select \* from client';

resultclient clob;

qryCtx1 dbms\_xmlgen.ctxHandle;

f1 utl\_file.file\_type;

begin

qryCtx1 := DBMS\_XMLGEN.newContext(cSQLclient);

DBMS\_XMLGEN.setRowSetTag(qryCtx1, 'client');

DBMS\_XMLGEN.setRowTag(qryCtx1, 'columns');

resultclient := DBMS\_XMLGen.getXML(qryCtx1);

DBMS\_XMLGen.CloseContext(qryCtx1);

begin

l\_amount := 32767;

l\_pos :=1;

f1 := utl\_file.fopen('EXTRACT\_DIR', 'EXPORT\_CLIENT\_TO\_XML.xml', 'w');

LOOP

DBMS\_LOB.read (resultclient, l\_amount, l\_pos, l\_buffer);

utl\_file.put\_line(f1, l\_buffer);

l\_pos := l\_pos + l\_amount;

END LOOP;

EXCEPTION

WHEN OTHERS THEN

utl\_file.fclose(f1);

end;

END;

END EXPORT\_XML\_CLIENT;

create or replace

PROCEDURE IMPORT\_XML\_CLIENT AS

BEGIN

DECLARE

cSQLclient CLOB;

line VARCHAR2(100);

f utl\_file.file\_type;

x pls\_integer := 0;

begin

begin

f := utl\_file.fopen('EXTRACT\_DIR', 'EXPORT\_CLIENT\_TO\_XML.xml', 'R');

loop

utl\_file.GET\_LINE(f,line);

cSQLclient:=CONCAT(cSQLclient,line);

x:=x+1;

exit when x=-5;

end loop;

exception

when others

then dbms\_output.put\_line(sqlerrm);

end;

utl\_file.fclose(f);

insert into xml\_tmp values (xmltype(cSQLclient));

insert into client

select FIRSTNAME,SECONDNAME,LOGIN,PASSWORD,ID,NUMBER\_GROUP

from xml\_tmp x, xmltable('/client/columns'

passing x.xml\_data columns FIRSTNAME VARCHAR2(50 BYTE), SECONDNAME VARCHAR2(50 BYTE),

LOGIN VARCHAR2(50 BYTE),PASSWORD VARCHAR2(50 BYTE),ID NUMBER,NUMBER\_GROUP NUMBER

) xt;

delete from xml\_tmp;

END;

END IMPORT\_XML\_CLIENT;

# ПРИЛОЖЕНИЕ В.

Триггеры

Триггер для таблицы CLIENT

create or replace

trigger Trigger\_For\_CLIENT

after update or insert or delete on CLIENT

for each row

begin

dbms\_output.put\_line('Trigger\_For\_Table\_CLIENT');

if inserting then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'inserting','Trigger\_For\_Table\_CLIENT', :new.ID || ' ' || :new.LOGIN || ' ' || :new.PASSWORD|| ' ' || :new.FIRSTNAME|| ' ' || :new.SECONDNAME || ' ' || :new.NUMBER\_GROUP || ' ');

elsif updating then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'updating', 'Trigger\_For\_Table\_CLIENT', 'before: ' || :old.ID || ' ' || :old.LOGIN || ' ' || :old.PASSWORD|| ' ' || :old.FIRSTNAME|| ' ' || :old.SECONDNAME || ' ' || :old.NUMBER\_GROUP || 'after: ' || :new.ID || ' ' || :new.LOGIN || ' ' || :new.PASSWORD|| ' ' || :new.FIRSTNAME|| ' ' || :new.SECONDNAME || ' ' || :new.NUMBER\_GROUP || ' ');

elsif deleting then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'deleting', 'Trigger\_For\_Table\_CLIENT', :old.ID || ' ' || :old.LOGIN || ' ' || :old.PASSWORD|| ' ' || :old.FIRSTNAME|| ' ' || :old.SECONDNAME || ' ' || :old.NUMBER\_GROUP || ' ');

end if;

end;

Триггер для таблицы DATACLIENT

create or replace

trigger Trigger\_For\_DATACLIENT

after update or insert or delete on DATACLIENT

for each row

begin

dbms\_output.put\_line('Trigger\_For\_Table\_DATACLIENT');

if inserting then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'inserting','Trigger\_For\_Table\_DATACLIENT', :new.ID || ' ' || :new.WEIGHT || ' ' || :new.HEIGHT|| ' ' || :new.BODYTYPE|| ' ' || :new.PROGRESS);

elsif updating then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'updating', 'Trigger\_For\_Table\_DATACLIENT', 'before: ' || :old.ID || ' ' || :old.WEIGHT || ' ' || :old.HEIGHT|| ' ' || :old.BODYTYPE|| ' ' || :old.PROGRESS || 'after: ' || :new.ID || ' ' || :new.WEIGHT || ' ' || :new.HEIGHT|| ' ' || :new.BODYTYPE|| ' ' || :new.PROGRESS);

elsif deleting then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'deleting', 'Trigger\_For\_Table\_DATACLIENT', :old.ID || ' ' || :old.WEIGHT || ' ' || :old.HEIGHT|| ' ' || :old.BODYTYPE|| ' ' || :old.PROGRESS);

end if;

end;

Триггер для таблицы TRAINER

create or replace

trigger Trigger\_For\_TRAINER

after update or insert or delete on TRAINER

for each row

begin

dbms\_output.put\_line('Trigger\_For\_Table\_TRAINER');

if inserting then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'inserting','Trigger\_For\_Table\_TRAINER', :new.ID || ' ' || :new.LOGIN || ' ' || :new.PASSWORD|| ' ' || :new.FIRSTNAME|| ' ' || :new.SECONDNAME || ' ' || :new.NUMBER\_GROUP);

elsif updating then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'updating', 'Trigger\_For\_Table\_TRAINER', 'before: ' || :old.ID || ' ' || :old.LOGIN || ' ' || :old.PASSWORD|| ' ' || :old.FIRSTNAME|| ' ' || :old.SECONDNAME || ' ' || :old.NUMBER\_GROUP || 'after: ' || :new.ID || ' ' || :new.LOGIN || ' ' || :new.PASSWORD|| ' ' || :new.FIRSTNAME|| ' ' || :new.SECONDNAME || ' ' || :new.NUMBER\_GROUP);

elsif deleting then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'deleting', 'Trigger\_For\_Table\_TRAINER', :old.ID || ' ' || :old.LOGIN || ' ' || :old.PASSWORD|| ' ' || :old.FIRSTNAME|| ' ' || :old.SECONDNAME || ' ' || :old.NUMBER\_GROUP);

end if;

end;

Триггер для таблицы DATATRAINER

create or replace

trigger Trigger\_For\_DATATRAINER

after update or insert or delete on DATATRAINER

for each row

begin

dbms\_output.put\_line('Trigger\_For\_Table\_DATATRAINER');

if inserting then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'inserting','Trigger\_For\_Table\_DATATRAINER', :new.ID || ' ' || :new.WEIGHT || ' ' || :new.HEIGHT|| ' ' || :new.BARBELLSQUAT|| ' ' || :new.DEADLIFT || ' ' || :new.BENCHPRESS || ' ' || :new.PULLUPS || ' ' || :new.BODYTYPE);

elsif updating then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'updating', 'Trigger\_For\_Table\_DATATRAINER', 'before: ' || :old.ID || ' ' || :old.WEIGHT || ' ' || :old.HEIGHT|| ' ' || :old.BARBELLSQUAT || ' ' || :old.DEADLIFT || ' ' || :old.BENCHPRESS || ' ' || :old.PULLUPS || ' ' || :old.BODYTYPE || 'after: ' || :new.ID || ' ' || :new.WEIGHT || ' ' || :new.HEIGHT|| ' ' || :new.BARBELLSQUAT|| ' ' || :new.DEADLIFT || ' ' || :new.BENCHPRESS || ' ' || :new.PULLUPS || ' ' || :new.BODYTYPE);

elsif deleting then

insert into DIAGNOSTIC\_TABLE values (CURRENT\_DATE, 'deleting', 'Trigger\_For\_Table\_DATATRAINER', :old.ID || ' ' || :old.WEIGHT || ' ' || :old.HEIGHT|| ' ' || :old.BARBELLSQUAT || ' ' || :old.DEADLIFT || ' ' || :old.BENCHPRESS || ' ' || :old.PULLUPS || ' ' || :old.BODYTYPE);

end if;

end;