

Spis treści

1.	Wstęp	1
1.1.	Cel i koncepcja projektu.....	1
1.2.	Instalacja aplikacji.....	1
1.3.	Interfejs użytkownika i omówienie funkcjonalności	1
2.	Specyfikacja projektowa.....	9
2.1.	Opis i uzasadnienie wyboru technologii	9
2.2.	Przegląd innych rozwiązań.....	10
3.	Techniczna implementacja	10
3.1.	Pliki źródłowe .cpp.....	10
3.2.	Pliki nagłówkowe .h.....	24
3.3.	Baza danych	27
4.	Testowanie projektu	28
5.	Wymagania aplikacji	29
6.	Załączniki	29
6.1.	Kod źródłowy.....	29
6.2.	Oświadczenie o samodzielności wykonania projektu.....	30

1. Wstęp

1.1. Cel i koncepcja projektu

Celem projektu pod tytułem „Dziennik treningowy” jest stworzenie przystępnej, łatwej w obsłudze aplikacji do zapisywania postępów w treningach.

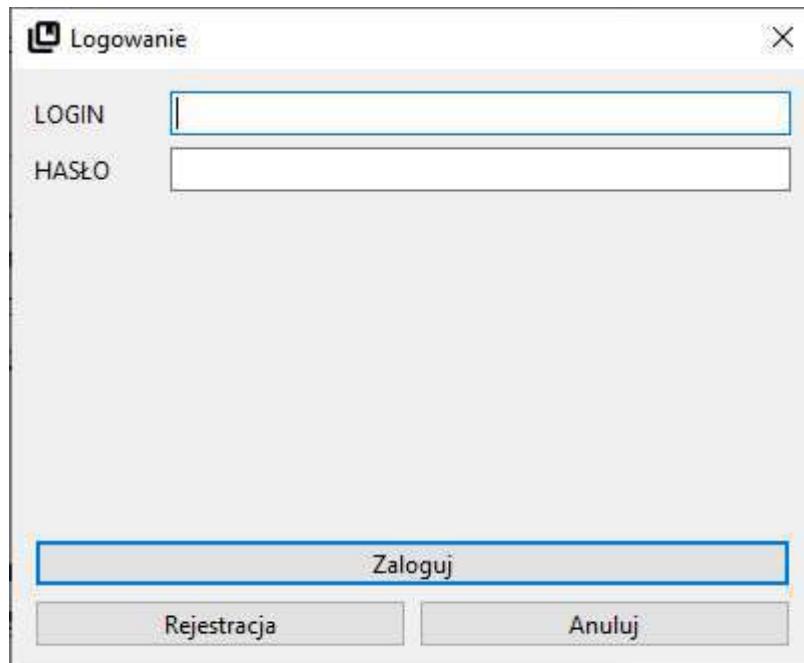
Program jest przeznaczony dla osób, które trenują oraz dla tych, którzy zamierzają trenować. Dlatego każdy ma możliwość rejestracji oraz logowania się do dziennika treningowego. Aplikacja posiada funkcję zapisywania, edytowania, usuwania i wyświetlania ćwiczeń. Podstawowe ćwiczenia są dostępne domyślnie w programie ale można też dodać własne ćwiczenia. Dziennik treningowy generuje postępy w ćwiczeniach oraz rankingi co do wagi obciążenia, z którym zostało wykonane ćwiczenie.

1.2. Instalacja aplikacji

Aby zainstalować aplikację należy rozpakować archiwum z plikami programu. Po rozpakowaniu otrzymamy plik Dziennik.exe, który jest gotowy do otwarcia i korzystania. Aby aplikacja działała poprawnie nie należy zmieniać położenia pliku wykonywalnego. Powiązany jest on z bazą danych i zmiana położenia może usunąć to połączenie.

1.3. Interfejs użytkownika i omówienie funkcjonalności

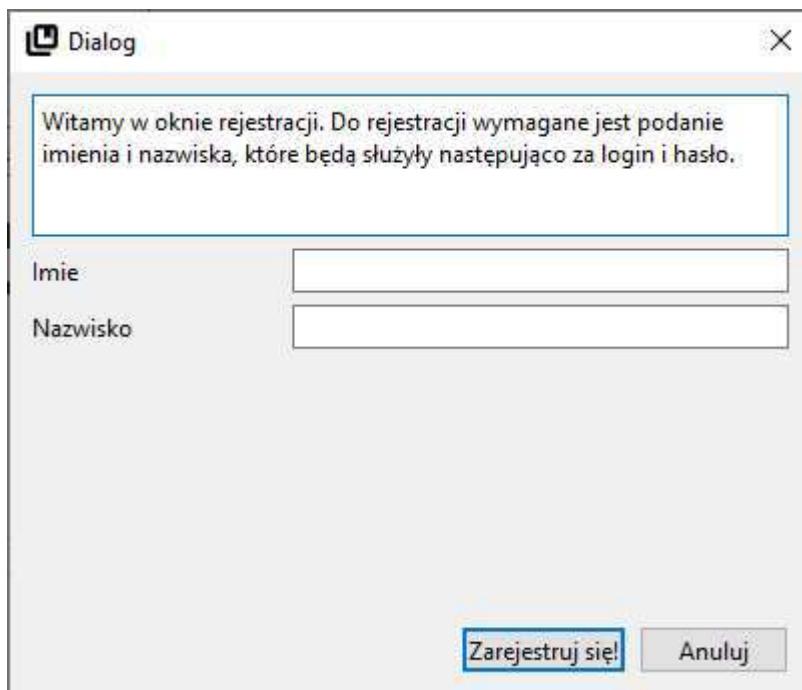
Po uruchomieniu aplikacji zostanie wyświetcone okno dialogowe do zalogowania się, bądź zarejestrowania się.



Zrzut ekranu 1 Logowanie

Login jest naszą nazwą użytkownika, a hasło kodem dostępu. Zaleca się stosować jako login – imię oraz jako hasło – nazwisko. Po wpisaniu danych należy nacisnąć zaloguj aby przejść do aplikacji.

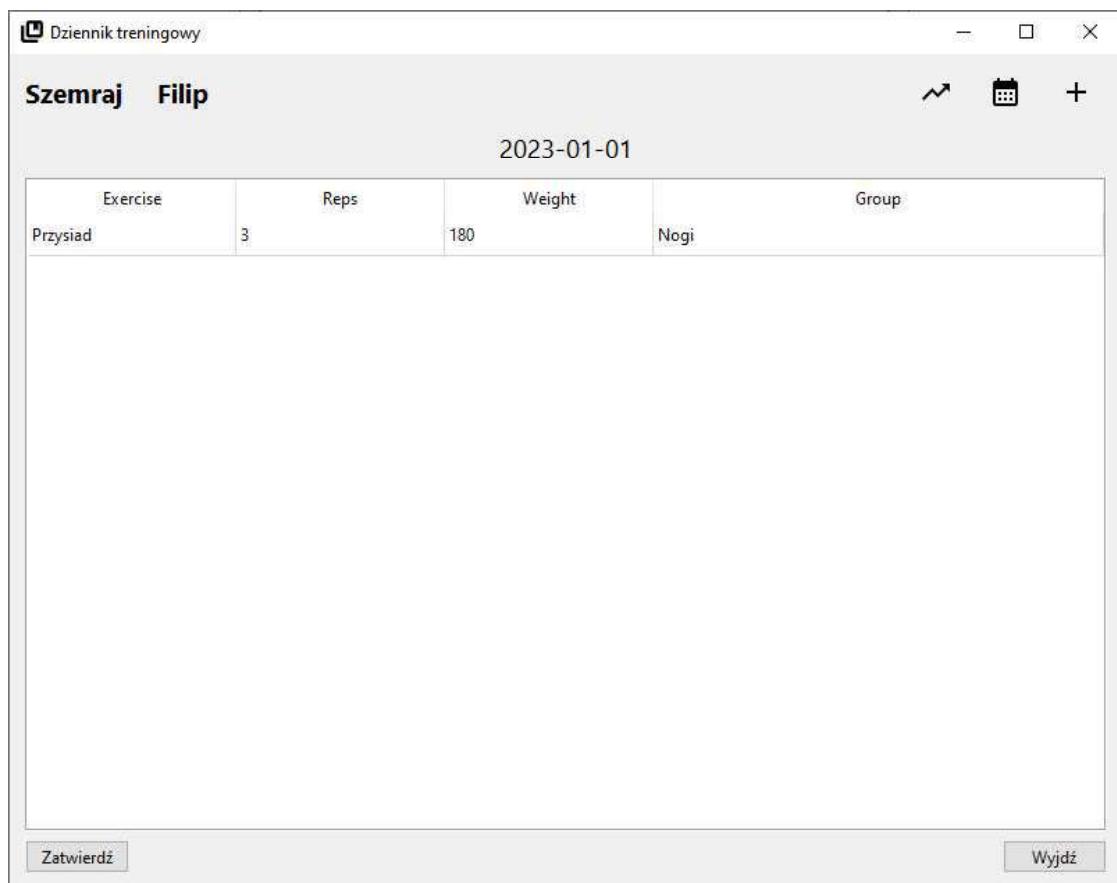
Jeżeli nie mamy własnego konta użytkownika możemy użyć opcji „Rejestracja”, aby utworzyć swój profil.



Zrzut ekranu 2 Rejestracja

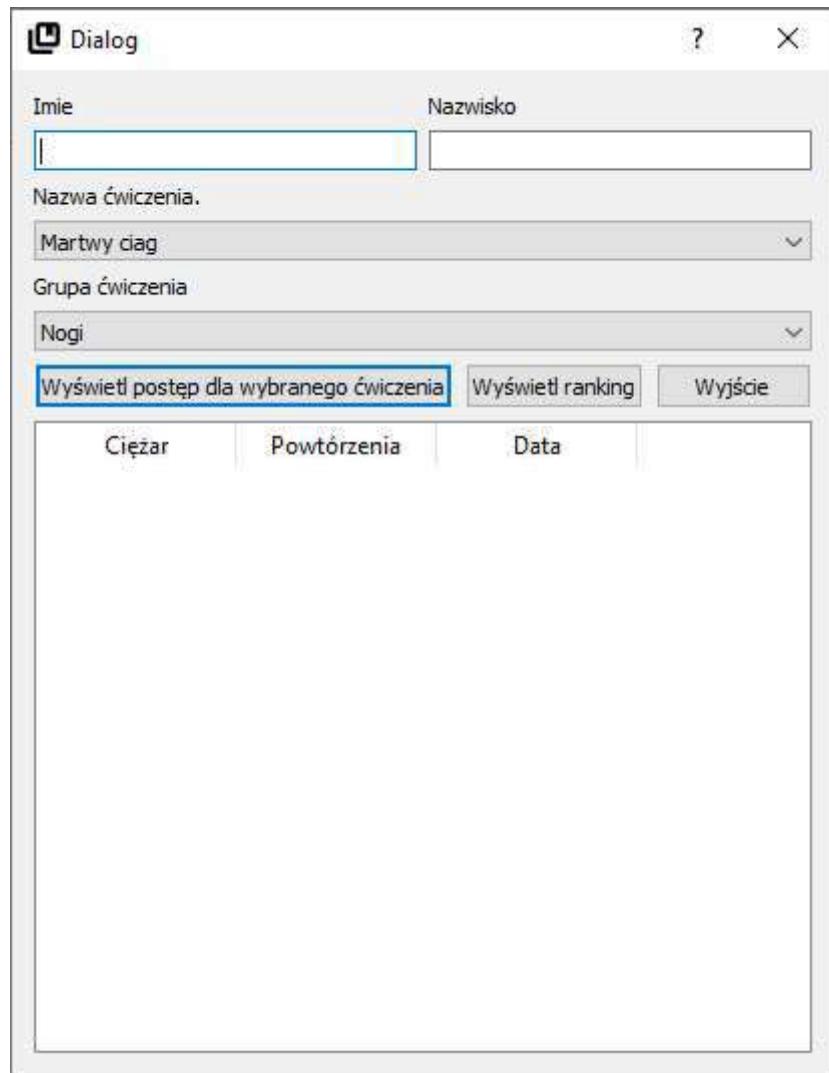
W oknie rejestracji wpisujemy swoje imię oraz nazwisko, które będą funkcjonować jako login i hasło, co jest opisane w polu tekstowym na górze okna dialogowego. Po wypełnieniu swoich danych należy nacisnąć przycisk „Zarejestruj się!”, a następnie zalogować się danymi podanymi przy rejestracji.

Po zalogowaniu się pokaże się główne okno aplikacji. Z tego poziomu możemy wykonać wszystkie operacje dostępne w programie.



Zrzut ekranu 3 Główne okno aplikacji

W lewym górnym rogu wyświetla się imię (login) i nazwisko (hasło) zalogowanego użytkownika. W prawym górnym rogu dostępne są trzy przyciski. Pierwszy od lewej otwiera okno dialogowe, w którym można wyświetlić rekordy oraz postępy dla danego ćwiczenia.



Zrzut ekranu 4 Postępy i rankingi

Drugim przyciskiem wybieramy datę, w której przeprowadzimy trening albo wybieramy datę, w której przeprowadzaliśmy trening.

styczeń 2023							
	pon.	wt.	śr.	czw.	pt.	sob.	niedz.
52	26	27	28	29	30	31	1
1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22
4	23	24	25	26	27	28	29
5	30	31	1	2	3	4	5

Zrzut ekranu 5 Kalendarz

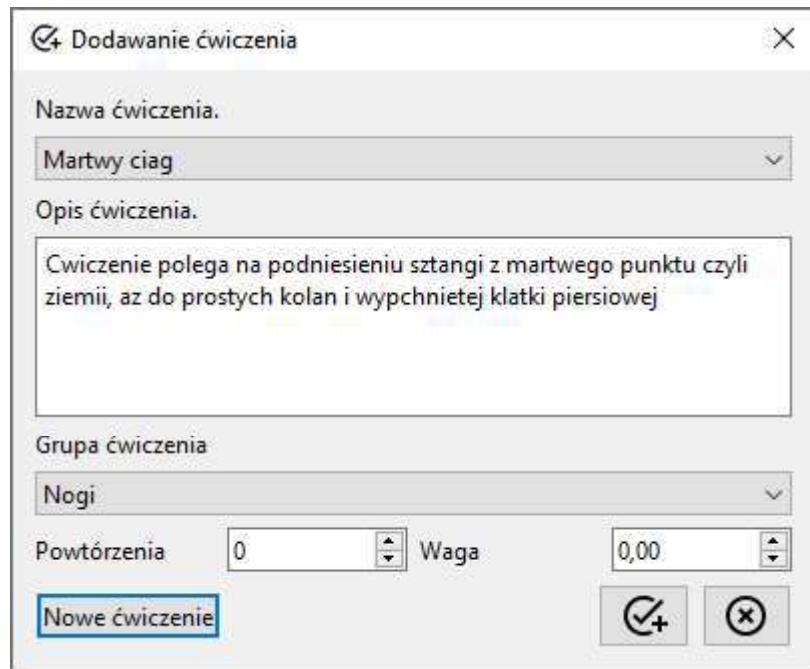
Po wybraniu daty, w której przeprowadziliśmy już trening, zostanie on automatycznie wyświetlony w centralnej części ekranu.

2023-01-01			
Exercise	Reps	Weight	Group
Przysiad	3	180	Nogi

Zrzut ekranu 6 Wyświetlanie ćwiczeń

Trzeci przycisk służy do dodania nowego ćwiczenia. Po wybraniu tej opcji zostanie wyświetlone nowe okno dialogowe, w których wybieramy następujące opcje:

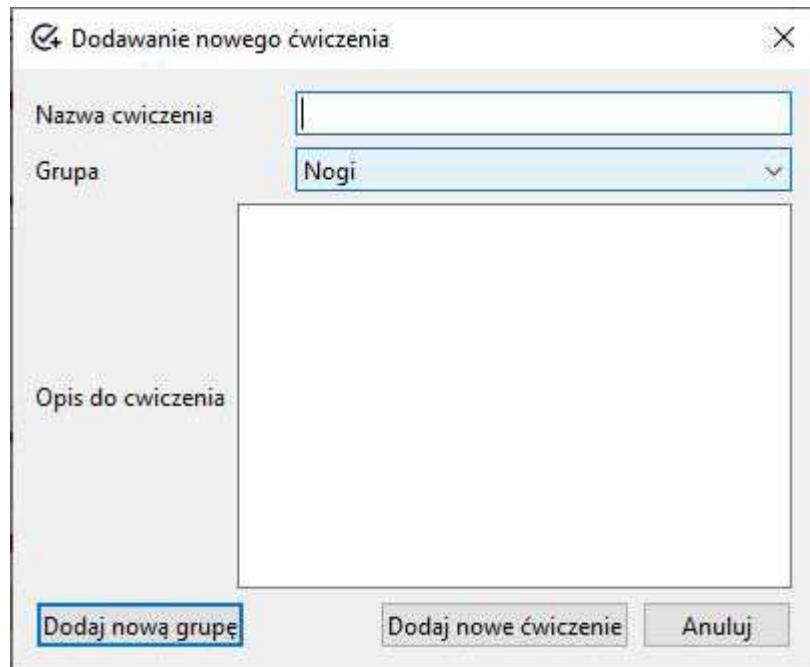
- nazwę ćwiczenia,
- grupę ćwiczenia,
- powtórzenia,
- wagę



Zrzut ekranu 7 Dodawanie ćwiczenia

Aby zatwierdzić wprowadzone dane naciskamy na przycisk znajdujący się w lewym dolnym rogu na lewo od przycisku zamknięcia okna oznaczonego krzyżkiem w kółku.

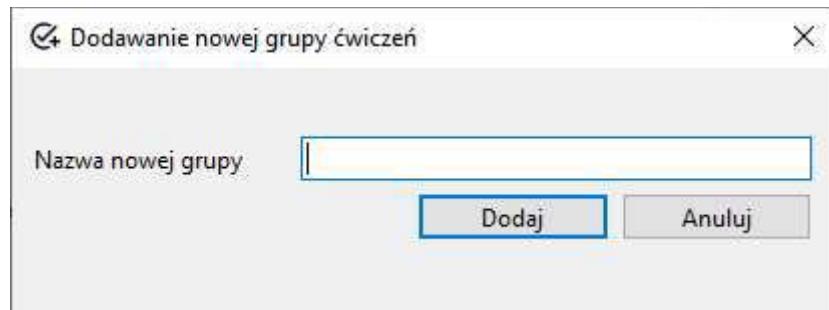
Jeżeli nie ma ćwiczenia, którego chcemy wprowadzić to możemy go dodać przyciskiem „Nowe ćwiczenie”, znajdującym się w lewym dolnym rogu.



Zrzut ekranu 8 Nowe ćwiczenie

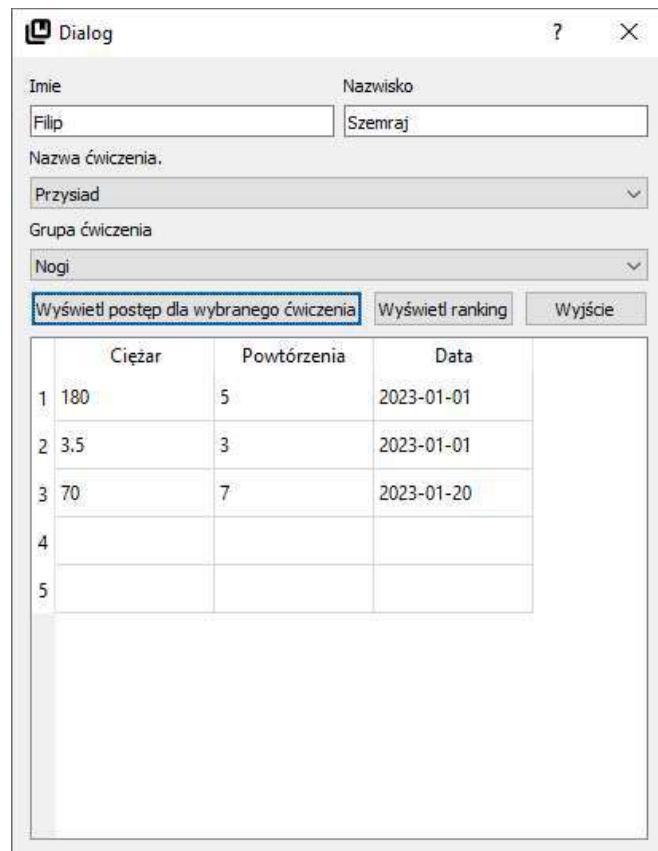
W oknie dialogowym dodania nowego ćwiczenia wprowadzamy nazwę tego ćwiczenia, jego grupę oraz opis. Dodawanie zatwierdzamy przyciskiem z dołu ekranu „Dodaj nowe ćwiczenie”.

Jeżeli w liście rozwijanej nie ma odpowiedniej grupy do wprowadzanego ćwiczenia, to możemy je dodać przyciskiem znajdującym się w lewym dolnym rogu „Dodaj nową grupę”.



Zrzut ekranu 9 Dodawanie nowej grupy ćwiczeń

W tym oknie dialogowym wprowadzamy nazwę grupy ćwiczeniowej i zatwierdzamy przyciskiem „Dodaj”.



Zrzut ekranu 10 Wyświetlanie postępów

W oknie dialogowym do wyświetlania postępów wyświetlane są postępy w danym ćwiczeniu, danego użytkownika według daty.

Dialog

Ciężar	Powtórzenia	Nazwa ćwiczenia	Dzień pójścia rekordu	Imię rekordzisty	Nazwisko rekordzisty
45	6	Martwy ciąg	2023-01-01	Weronika	Gil
180	5	Przysiad	2023-01-01	Filip	Szemraj
120	10	Wyciskanie leżąc	2023-01-02	Weronika	Gil

Wyjście

Zrzut ekranu 11 Rekordy

W oknie dialogowym rekordów wyświetlane są rekordy każdego ćwiczenia. Wyświetlane są dane o ciężarze, powtórzeniach, nazwie ćwiczenia, dacie i danych osoby posiadającej rekord.

Dziennik treningowy

Szemraj Filip

2023-01-01

Exercise	Reps	Weight	Group
Przysiad	3	3.5	Nogi
Martwy ciąg	3	50	Nogi
Martwy ciąg	3	60	Nogi

Zatwierdź Wyjdź

Zrzut ekranu 12 Usuwanie i edycja wpisów

Aby edytować wpis klikamy raz na wybraną komórkę i zaczynamy pisać. Edycje zatwierdzamy przyciskiem zatwierdź. Usuwanie rekordów przebiega przez wybranie wiersza i naciśnięcie klawisza backspace.

2. Specyfikacja projektowa

2.1. Opis i uzasadnienie wyboru technologii

Projekt został zrealizowany przy pomocy bibliotek i narzędzi programistycznych programu QT w wersji QT Creator 9.0.0. Projekt został napisany w języku C++ oraz SQL. Do zapisu danych została zaimplementowana baza danych SQLite 3. Biblioteki dołączone do programu to:

- QDialog – do wyświetlania okien dialogowych.
- QSqlDatabase – do połączenia z bazą danych SQLite
- QSql – do tworzenia zapytań w języku SQL, do sterowania bazą danych.
- QtDebug – do debugowania programu
- QApplication – do kontrolowania ustawień aplikacji
- QDate – do obsługi dat
- QObject

Program QT został wybrany z uwagi na przystępny, łatwy w obsłudze interfejs, mnogość darmowych, wbudowanych bibliotek, z których można wykorzystać wiele funkcji do budowy aplikacji okienkowej opartej na bazie danych. Program jest darmowy do użytku niekomercyjnego, co jest jedną z jego zalet. Kolejnym argumentem za wybrianiem tego programu było możliwość łatwego przesyłania postępów w projekcie między członkami zespołu poprzez wysłania zarchiwizowanego folderu z projektem.

Projekt wysłany w taki sposób może zostać uruchomiony do edycji przez każdego z zespołu bez problemów. Program QT posiada wiele elementów potrzebnych do budowy interfejsu oraz możliwość dodania własnych ikon lub grafik.

Projekt Dziennik treningowy został zrealizowany przy użyciu projektu Qt Widget Application. Głównym oknem jest plik mainwindow.cpp, reszta okien jest dialogowa. Dla każdego okna powstał plik nagłówkowy z rozszerzeniem .h, plik źródłowy z rozszerzeniem .cpp oraz plik formularza z rozszerzeniem .ui.

Do przechowywania danych została użyta baza danych napisana w języku SQL przy użyciu systemu zarządzania relacyjnych baz danych – SQLite 3. SQLite jest darmowym oprogramowaniem na licencji domeny publicznej. Został on wybrany głównie ze względu na kompatybilność z programem QT. Innymi czynnikami, które doprowadziły do wyboru tego systemu było to, że zajmuje mało pamięci, jest samowystarczalny, wysoce niezawodny oraz w pełni funkcjonalny.

2.2. Przegląd innych rozwiązań

Oprócz programu QT oraz systemu zarządzania relacyjnych baz danych – SQLite 3 rozważane były także inne rozwiązania.

Do utworzenia projektu mógł zostać wykorzystany program CodeBlocks z biblioteką WxWidgets. Nie został on jednak wybrany z uwagi na problemy z instalacją owego oprogramowania na komputerach członków zespołu. CodeBlocks z biblioteką WxWidgets posiadają także znacznie mniej przystępny interfejs oraz bardziej skomplikowany proces tworzenia aplikacji.

Aby przechowywać dane mogła zostać struktura danych. Po zgłębieniu zagadnienia uznaliśmy, że stworzenie aplikacji opartej na tej formie zapisywania danych byłaby zbyt skomplikowana i trudna w implementacji w programie QT.

3. Techniczna implementacja

3.1. Pliki źródłowe .cpp

Plik, który uruchamia cały projekt to main.cpp.

```
#include "login.h"
#include <QSqlDatabase>
#include <QtSql>
#include <QtDebug>
#include < QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    login loginWindow;
    loginWindow.show();

    return a.exec();
}
```

Listing 1 Plik main.cpp

Jest to standardowa konstrukcja pliku uruchamiającego projekt. Załączony do niego plik login.h pozwala utworzyć klasę login i obiekt loginWindow, który otwiera okno logowania (login.cpp) przez funkcje show.

```
login::login(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::login)
```

```

{
    ui->setupUi(this);

    mydb = QSqlDatabase::addDatabase("QSQLITE");
    mydb.setDatabaseName("baza_dziennik.db");
    if(mydb.open())
    {
        qDebug()<<"Database opened!";
    }
    else
    {
        qDebug() << "Error = "<< mydb.lastError();
    }
}

```

Listing 2 Plik login.cpp

W pliku login.cpp utworzone zostało ui przez funkcje ui(new Ui::login). Ui zostało zaimplementowane funkcją setupUi. Następnie zostało utworzone połączenie z bazą danych. Do pola mydb z klasy login.h zostało dołączone połączenie z bazą danych QSQLITE przez funkcję addDatabase. Następnie do pola mydb została dołączona ścieżka do bazy danych poprzez funkcje setDatabaseName. Kolejno w instrukcji warunkowej if otwierana jest baza danych. Jeżeli otwieranie przebiegnie pomyślnie to dla debugera zostanie przesłana informacja „Database opened!” przez funkcje qDebug. W przeciwnym razie zostanie przesłany błąd.

```

login::~login()
{
    delete ui;
}

void login::on_acceptB_clicked()
{

    QString username = ui->lineEdit_login->text();
    QString password = ui->lineEdit_password->text();
    QSqlQuery qry;
    if(qry.exec("SELECT * FROM user;"))
    {
        while(qry.next())
        {
            QString temp_username = qry.value(1).toString();
            QString temp_password = qry.value(2).toString();
            if(username == temp_username && password == temp_password)
            {
                qDebug() << "Zalogowano pomyślnie";
                dataWindow = new MainWindow(nullptr, &mydb, username, password);
                dataWindow->show();
            }
        }
    }
}

```

```

        login::~login();
        return;
    }
}
qDebug() << "Złe dane";
}
}

```

Listing 3 Plik login.cpp, destruktor, główna funkcja logująca

Destruktor zamyka okno i usuwa obiekt funkcją delete. Główna funkcja do logowania uruchamiana jest po naciśnięciu przycisku logowania „acceptB”. Tworzone są dwa stringi username oraz password. Przypisywane są do nich wartości wpisane przez użytkownika w polach tekstowychlineEdit_login i lineEdit_password. Następnie inicjowana jest zmienna qry do przechowania zapytania SQL. W instrukcji warunkowej if pobierana jest lista użytkowników z bazy danych. Kolejno w pętla while przehodzi przez wszystkie rekordy z poprzedniego zapytania. Utworzono są dwa stringi do przechowywania nazw i haseł z bazy danych. W instrukcji if sprawdzana jest zgodność wpisanych danych użytkownika z tymi będącymi w bazie danych. W przypadku gdy dane się zgadzają przesyłane są dla debugera informacje o pomyślnym zalogowaniu i tworzone jest nowe okno MainWindow przekazując połączenie z bazą danych oraz dane użytkownika. Obecnie okno jest zamykane. W wypadku gdy dane się nie zgadzają to wysyłane do debugera są informacje o wpisaniu niepoprawnych danych.

```

void login::on_DclineB_clicked()
{
    close();
}
void login::on_registerB_clicked()
{
    registerWindow = new registerUser(this, &mydb);
    registerWindow->setModal(true);
    registerWindow->show();
}

```

Listing 4 Plik login.cpp, zamykanie okna, tworzenie okna rejestracji

Funkcja uruchamiana po naciśnięciu przycisku DeclineB. W tej funkcji znajduję się funkcja close do zamykania okna. Funkcja uruchamiana po naciśnięciu przycisku registerB służy do utworzenia okna do rejestracji użytkownika. Tworzony jest obiekt registerUser przekazując do niego połączenie z bazą danych. Następnie uruchamiane jest nowe okno dialogowe.

```

registerUser::registerUser(QWidget *parent, QSqlDatabase *db) :
    QDialog(parent),
    mydb(db),
    ui(new Ui::registerUser)
{
    ui->setupUi(this);
}

```

```
mydb->open();
    filling=false;
}
```

Listing 5 Plik registeruser.cpp

W pliku registeruser.cpp tworzone jest ui podobnie jak w listingu 2. Przekazywane także jest połączenie z bazą danych.

```
void registerUser::on_registerB_clicked()
{
    QString name = ui->ImieLE->text();
    QString surname = ui->NazwiskoLE->text();

    if(qry.exec("SELECT * FROM user;"))
    {
        while(qry.next())
        {
            QString temp_username = qry.value(1).toString();
            QString temp_password = qry.value(2).toString();
            if(name == temp_username && surname == temp_password)
            {
                qDebug()<<"Uzytkownik z podanymi danymi juz istnieje";
            }
            else
            {
                qry.prepare("INSERT INTO user(name, surname) VALUES (:n, :s);");
                qry.bindValue(":n", name);
                qry.bindValue(":s", surname);
                if(qry.exec())
                {
                    qDebug()<<"Uzytkownik: "<<name<<surname<<" zostal dodany...";
                }
            }
        }
    }
}
```

Listing 6 Plik registeruser.cpp, dodawanie użytkownika

W funkcji uruchamianej po naciśnięciu przycisku registerB, utworzone są dwa stringi, do których zapisywane są dane wpisane przez użytkownika do pól tekstowych. Następnie w instrukcji if wykonywane jest zapytanie wybierające wszystkich użytkowników. Sprawdzane jest czy istnieje użytkownik z tymi samymi danymi, które podał użytkownik poprzez porównanie ich z rekordami z bazy danych. W przypadku gdy istnieje taki użytkownik to wysyłana jest informacje do debugera. W przeciwnym wypadku przygotowywane jest zapytanie SQL tworzące użytkownika. Do nazwy użytkownika i hasła przypisywane są dane wprowadzone wcześniej. Zapytanie jest wykonywane i gdy ta operacja się powiedzie wysyłane są dane użytkownika i komunikat o tym do debugera.

```

MainWindow::MainWindow(QWidget *parent, QSqlDatabase *db, QString n, QString s)
: QWidget(parent),
mydb(db),
name(n),
surname(s),
ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    mydb->open();
    ui->Imie_L->setText(name);
    ui->Nazwisko_L->setText(surname);

    qry.prepare("SELECT id_user FROM user WHERE name=:n AND surname=:s;");
    qry.bindValue(":n", name);
    qry.bindValue(":s", surname);
    qry.exec();
    qry.next();
    id = qry.value(0).toInt();

    filling=false;
    trainingDate.setDate(2023,01,01);
    ui->DateB->setText("2023-01-01");
    fillForm();
}

```

Listing 7 Plik mainwindow.cpp, tworzenie ui

W funkcji tworzącej okno MainWindow wraz z jego ui wykorzystywane jest istniejące już połaczenie z bazą danych. Pobierane są dane zalogowanej osoby, a następnie wyświetlane w lewym górnym rogu okna. Ustawiana jest także data na 1 stycznia 2023.

```

void MainWindow::fillForm()
{
    if(filling)
    {
        return;
    }
    filling = true;
    //Zamiana daty na taka z 0 na poczatku, jesli dzien lub miesiac jest 1 cyfrowy.
    QString temp = repairDate();
    qry.prepare("SELECT count(u.name) "
               "FROM training_day td "
               "JOIN exercise_training_day et ON et.id_training_day = td.id_training_day "
               "JOIN exercises_set e ON e.id_exercise = et.id_exercise "
               "JOIN user u ON u.id_user = td.id_user "
               "JOIN exercise_groups eg ON eg.id_exercise_group = e.id_exercise_group "
               "WHERE u.id_user = :i AND td.d = :temp "
               "ORDER BY e.name;");

}

```

```

qry.bindValue(":i", id);
qry.bindValue(":temp", QString(temp));
qry.exec();
qry.first();
ui->tableWidget->setRowCount(qry.value(0).toInt());
qry.prepare("SELECT e.name, et.reps, et.weight, eg.name "
    "FROM training_day td "
    "JOIN exercise_training_day et ON et.id_training_day = td.id_training_day "
    "JOIN exercises_set e ON e.id_exercise = et.id_exercise "
    "JOIN user u ON u.id_user = td.id_user "
    "JOIN exercise_groups eg ON eg.id_exercise_group = e.id_exercise_group "
    "WHERE u.id_user = :i AND td.d = :temp "
    "ORDER BY e.name;");
qry.bindValue(":i", id);
qry.bindValue(":temp", QString(temp));
if(qry.exec())
{
    int row=0;
    for(qry.first(); qry.isValid(); qry.next(), ++row)
    {
        ui->tableWidget->setItem(row, 0, new QTableWidgetItem(qry.value(0).toString()));
        ui->tableWidget->setItem(row, 1, new QTableWidgetItem(qry.value(1).toString()));
        ui->tableWidget->setItem(row, 2, new QTableWidgetItem(qry.value(2).toString()));
        ui->tableWidget->setItem(row, 3, new QTableWidgetItem(qry.value(3).toString()));
    }
}
filling = false;
}

```

Listing 8 Plik mainwindow.cpp, funkcja fillform

Funkcja fillform służy do wyświetlania treningów z danego dnia. Używana jest tu także funkcja repairdate do zmiany formatu daty. Przygotowywane jest złożone zapytanie SQL, które pokazuje wszystkie ćwiczenia wykonane w dany dzień. Data jest przechowywana w zmiennej temp. Pierwsza instrukcja SQL używana jest do ustalenia liczby wierszy w tabeli. Druga instrukcja SQL wyświetla nazwę ćwiczenia, ilość powtórzeń, wagę podnoszonego ciężaru, grupę mięśni na których przeprowadzane jest ćwiczenie.

```

QString MainWindow::repairDate()
{
    QString day;
    QString month;

    if(trainingDate.day()<10)
    {
        day = QString("0%1").arg(trainingDate.day());
    }
    else

```

```

{
    day = QString("%1").arg(trainingDate.day());
}

if(trainingDate.month()<10)
{
    month = QString("0%1").arg(trainingDate.month());
}
else
{
    month = QString("%1").arg(trainingDate.month());
}

QString year = QString("%1").arg(trainingDate.year());

QString temp = year+"-"+month+"-"+day;
return temp;
}

```

Listing 9 Plik mainwindow.cpp, funkcja repairdate

W pliku mainwindow.cpp występuje funkcja rapairdate. Służy ona do formatowania daty w ten sposób aby w datach, które mają jednocyfrowy dzień lub miesiąc zostało dodane 0 na początku. Tworzone są dwa stringi do zapisu dnia i miesiąca, a następnie w instrukcjach if sprawdzane jest czy są jedno czy dwu cyfrowe. Jeśli są dwu to data pozostaje taka sama, a jeżeli jedno cyfrowe to dodawane jest 0 do stringa. Następnie data jest łączona i zwracana w stringu temp.

```

void MainWindow::keyPressEvent(QKeyEvent *event)
{
    if(event->key() == Qt::Key_Backspace)
    {
        ui->tableView->removeRow(ui->tableView->currentRow());
    }
}

void MainWindow::on_Graphs_B_clicked()
{
    chartsWindow = new chartsOption(this, mydb, id);
    chartsWindow->setModal(false);
    chartsWindow->show();
}

void MainWindow::on_exitB_clicked()
{
    free(mydb);
    free(calendarWindow);
    free(windowExercise);
    free(chartsWindow);
}

```

```

    QApplication::quit();
    //close();
}

```

Listing 10 Plik mainwindow.cpp, usuwanie rekordów, nowe okno postępów, zamykanie aplikacji

Funkcja obsługująca usuwanie ćwiczeń sprawdza czy naciśnięty został przycisk backspace i gdy został usuwany jest zaznaczony wiersz. Kolejną funkcją jest funkcja do utworzenia nowego okna chartoption. Ostatnią funkcją jest zamknięcie programu i uwolnienie pamięci.

```

void MainWindow::receiveDate(QDate d)
{
    int day = d.day();
    int month = d.month();
    int year = d.year();
    trainingDate.setDate(year,month,day);

    //Zamiana daty na taka z 0 na poczatku, jesli dzien lub miesiac jest 1 cyfrowy.
    QString temp = repairDate();

    ui->DateB->setText(temp);
    fillForm();

}

void MainWindow::receiveDateForm(int reps, double weight, QString group, QString name)
{
    QString repsS = QString("%1").arg(reps);
    QString weightS = QString("%1").arg(weight);

    int maxRow = ui->tableWidget->rowCount();
    maxRow++;
    ui->tableWidget->setRowCount(maxRow);
    maxRow--;
    ui->tableWidget->setItem(maxRow, 0, new QTableWidgetItem(name));
    ui->tableWidget->setItem(maxRow, 1, new QTableWidgetItem(repsS));
    ui->tableWidget->setItem(maxRow, 2, new QTableWidgetItem(weightS));
    ui->tableWidget->setItem(maxRow, 3, new QTableWidgetItem(group));
}

```

Listing 11 Plik mainwindow.cpp, funkcja ustawiająca date i ustawiającą tabelę

Funkcja receivedate pobiera datę zaznaczoną w kalendarzu, łączy ją funkcją setdate oraz wyświetla na ekranie. Funkcja revicedateform pobiera w argumentach dane z ćwiczenia i następnie ustawia je w tabeli.

```

void MainWindow::on_Add_B_clicked()
{
    windowExercise = new adding_exercise(this, mydb);
}

```

```

windowExercise->setModal(true);
windowExercise->show();
connect(windowExercise, SIGNAL(sendDateForm(int, double ,QString, QString)), this,
SLOT(receiveDateForm(int, double ,QString, QString)));
}
void MainWindow::on_DateB_clicked()
{
    calendarWindow = new Calendar(this);
    calendarWindow->setModal(false);
    calendarWindow->show();
    connect(calendarWindow, SIGNAL(sendDate(QDate)), this, SLOT(receiveDate(QDate)));
}

void MainWindow::on_History_B_clicked()
{
    calendarWindow = new Calendar(this);
    calendarWindow->setModal(false);
    calendarWindow->show();
    connect(calendarWindow, SIGNAL(sendDate(QDate)), this, SLOT(receiveDate(QDate)));
}

void MainWindow::on_tableWidget_cellDoubleClicked()
{
    on_Add_B_clicked();
}

```

Listing 12 Plik mainwindow.cpp, otwieranie nowych okien

Funkcje zaprezentowane w listingu 12 odpowiadają za otwieranie okien takich jak: calendar oraz adding_exercise.

```

adding_exercise::adding_exercise(QWidget *parent, QSqlDatabase *db) :
    QDialog(parent),
    mydb(db),
    ui(new Ui::adding_exercise)
{
    ui->setupUi(this);
    mydb->open();
    filling=false;
    fillComboBox();
}
adding_exercise::~adding_exercise()
{
    delete ui;
}
void adding_exercise::fillComboBox()
{
    if(filling)

```

```

{
    return;
}
filling = true;
QString temp;
if(qry.exec("SELECT name FROM exercise_groups;"))
{
    qry.next();
    temp = qry.value(0).toString();
    do{
        ui->GroupEx_CB->addItem(qry.value(0).toString());
    }while(qry.next());
}
filling = false;
on_GroupEx_CB_textActivated(temp);
}

```

Listing 13 Plik adding_exercise.cpp, tworzenie okna, wypełnianie listy rozwijanej

W pliku adding_exercise.cpp zawarte są funkcje podobne do tych z poprzednich listingów oraz na funkcja wypełniania listy rozwijanej. Pobiera ona wszystkie rekordy z tabeli exercise_groups i przechodząc po każdym rekordzie w pętli while wpisywane są do listy rozwijanej.

```

void adding_exercise::on_GroupEx_CB_textActivated(const QString &arg1)
{
    if(filling)
    {
        return;
    }
    filling = true;
    QString temp;
    ui->Name_CB->clear();
    qry.prepare("SELECT es.name FROM exercise_groups eg "
               "JOIN exercises_set es ON es.id_exercise_group = eg.id_exercise_group "
               "WHERE eg.name = :group ;");
    qry.bindValue(":group", arg1);
    if(qry.exec())
    {
        qry.next();
        temp = qry.value(0).toString();
        do{
            ui->Name_CB->addItem(qry.value(0).toString());
        }while(qry.next());
    }
    filling = false;
    on_Name_CB_textActivated(temp);
}

```

```

void adding_exercise::on_Name_CB_textActivated(const QString &arg1)
{
    if(filling)
    {
        return;
    }
    filling = true;
    ui->Description_TB->clear();
    qry.prepare("SELECT es.description FROM exercise_groups eg "
               "JOIN exercises_set es ON es.id_exercise_group = eg.id_exercise_group "
               "WHERE es.name = :name ;");
    qry.bindValue(":name", arg1);
    qry.exec();
    qry.next();
    ui->Description_TB->setText(qry.value(0).toString());
    filling = false;
}

```

Listing 14 Plik adding_exercise.cpp, wyświetlanie wybranej opcji z list rozwijanych

Obie funkcje przedstawione w powyższym listingu dotyczą wyświetlania wybranej opcji w liście rozwijanej. Pobierane są wszystkie nazwy z danego pola i po wybraniu wyświetlane na liście rozwijanej.

```

void adding_exercise::on_DeclineB_clicked()
{
    close();
}
void adding_exercise::on_AcceptB_clicked()
{
    int reps = ui->RepsS->value();
    double weight = ui->WeightDS->value();
    QString group = ui->GroupEx_CB->currentText();
    QString name = ui->Name_CB->currentText();
    if(reps!=0 && !(group.isEmpty()) && !(name.isEmpty()))
    {
        emit sendDateForm(reps, weight ,group, name);
    }
}
void adding_exercise::on_addNewExB_clicked()
{
    newExerciseWindow = new adding_new_exercise(this, mydb);
    newExerciseWindow->setModal(true);
    newExerciseWindow->show();
}

```

Listing 15 Plik adding_exercise.cpp, dodawanie ćwiczenia, otwieranie nowego okna

W pierwszej funkcji pobierane są dane wpisane przez użytkownika w formularzu dodawania ćwiczenia, a następnie wywoływana funkcja senddateform, która zapisuje te dane. Kolejna funkcja tworzy okno adding_new_exercise.

```
void adding_new_exercise::on_Add_new_ex_B_clicked()
{
    QString name = ui->NameE->text();
    QString desc = ui->DescPT->toPlainText();
    if(name!=nullptr && desc!=nullptr)
    {
        QString group = ui->GroupExCB->currentText();
        int id_exercise_group;
        qry.prepare("SELECT id_exercise_group FROM exercise_groups "
                   "WHERE name=:g;");
        qry.bindValue(":g", group);
        qry.exec();
        qry.next();
        id_exercise_group=qry.value(0).toInt();
        qry.prepare("INSERT INTO exercises_set(name, description, id_exercise_group) "
                   "VALUES (:n, :d, :i);");
        qry.bindValue(":n", name);
        qry.bindValue(":d", desc);
        qry.bindValue(":i", id_exercise_group);
        if(qry.exec())
        {
            qDebug()<<"Pomyślnie dodano ćwiczenie: "<<name<<" do grupy: "<<group;
        }
    }
}
```

Listing 16 Plik adding_new_exercise.cpp, dodanie ćwiczeń do listy ćwiczeń

Funkcja przedstawiona powyżej przygotowuje zapytanie SQL, które dodaje ćwiczenia do listy ćwiczeń. Pobierane są wartości z pól tekstowych i zapisywane jako zmienne string. Zapytanie SQL wykonuje się i dodaje się nowe ćwiczenie.

```
void adding_new_group::on_AcceptB_clicked()
{
    QString name = ui->nameNewGroupL->text();
    if(name!=nullptr)
    {
        if(filling)
        {
            return;
        }
        filling = true;
        qry.prepare("INSERT INTO exercise_groups(name) "
                   "VALUES(:n);");
    }
}
```

```

qry.bindValue(":n", name);
if(qry.exec())
{
    qDebug() << "Dodano pomyślnie grupę o nazwie: " << name;
}
filling = false;
}
}

```

Listing 17 Plik adding_new_group.cpp, dodawanie nowej grupy ćwiczeń

Funkcja przedstawiona w listingu 17 odpowiada za dodanie nowej grupy ćwiczeń. Pobierana jest nazwa grupy, która ma zostać dodana do bazy i następnie przez instrukcje SQL jest zapisywana.

```

Calendar::Calendar(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Calendar)
{
    ui->setupUi(this);
}
Calendar::~Calendar()
{
    delete ui;
}
void Calendar::on_calendarWidget_selectionChanged()
{
    QDate d(ui->calendarWidget->selectedDate());
    emit sendDate(d);
}

```

Listing 18 Plik calendar.cpp

W pliku calendar.cpp został utworzony standardowy plik do obsługi kalendarza. Wykorzystywane do tego są funkcje wbudowane z biblioteki QCalendar.

```

void chartsOption::on_chartB_clicked()
{
    if(filling)
    {
        return;
    }
    filling = true;
    QString imie = ui->lineEdit_Imie->text();
    QString nazwisko = ui->lineEdit_Nazwisko->text();
    QString cwi = ui->Name_CB->currentText();
    qry.prepare("SELECT count(id_exercise) FROM exercise_training_day");
    qry.exec();
    qry.first();
    ui->tableWidget->setRowCount(qry.value(0).toInt());
}

```

```

qry.prepare("SELECT      exercise_training_day.weight,      exercise_training_day.reps,
training_day.d "
            "FROM exercise_training_day "
            "JOIN                      exercises_set          ON
exercises_set.id_exercise=exercise_training_day.id_exercise "
            "JOIN                      training_day           ON
training_day.id_training_day=exercise_training_day.id_training_day "
            "JOIN user  ON user.id_user=training_day.id_user "
            "WHERE  user.name='"+imie+"'  AND  user.surname='"+nazwisko+"'  AND
exercises_set.name='"+cwi+"'"
            "ORDER BY training_day.d ASC;");
if(qry.exec())
{
    int row=0;
    for(qry.first(); qry.isValid(); qry.next(), ++row)
    {
        ui->tableWidget->setItem(row, 0, new QTableWidgetItem(qry.value(0).toString()));
        ui->tableWidget->setItem(row, 1, new QTableWidgetItem(qry.value(1).toString()));
        ui->tableWidget->setItem(row, 2, new QTableWidgetItem(qry.value(2).toString()));
    }
}
filling = false;
}

```

Listing 20 Plik chartoption.cpp, wyświetlanie postępów

Funkcja wyświetlająca postępy pobiera dane użytkownika oraz rodzaj ćwiczenia dla którego mają zostać pokazane postępy. Przygotowywane są dwie instrukcje SQL. Pierwsza zlicza liczbę rekordów w tabeli ćwiczeń, aby ustawić wiersze w tabeli. Druga pobiera dane z bazy danych dla danych wpisanych przez użytkownika

```

void ranking::fillTable()
{
    if(filling)
    {
        return;
    }
    filling = true;
    qry.prepare("SELECT count(max) FROM (SELECT max(etd.weight) max "
                "FROM exercise_training_day etd "
                "JOIN exercises_set es ON es.id_exercise = etd.id_exercise "
                "JOIN training_day td ON td.id_training_day = etd.id_training_day "
                "JOIN user u ON u.id_user = td.id_user "
                "GROUP BY es.name);");
    qry.exec();
    qry.first();
    ui->tableWidget->setRowCount(qry.value(0).toInt());
}

```

```

qry.prepare("SELECT max(etd.weight), etd.reps, es.name, td.d, u.name, u.surname "
           "FROM exercise_training_day etd "
           "JOIN exercises_set es ON es.id_exercise = etd.id_exercise "
           "JOIN training_day td ON td.id_training_day = etd.id_training_day "
           "JOIN user u ON u.id_user = td.id_user "
           "GROUP BY es.name;");

if(qry.exec())
{
    int row=0;
    for(qry.first(); qry.isValid(); qry.next(), ++row)
    {
        ui->tableWidget->setItem(row, 0, new QTableWidgetItem(qry.value(0).toString()));
        ui->tableWidget->setItem(row, 1, new QTableWidgetItem(qry.value(1).toString()));
        ui->tableWidget->setItem(row, 2, new QTableWidgetItem(qry.value(2).toString()));
        ui->tableWidget->setItem(row, 3, new QTableWidgetItem(qry.value(3).toString()));
        ui->tableWidget->setItem(row, 4, new QTableWidgetItem(qry.value(4).toString()));
        ui->tableWidget->setItem(row, 5, new QTableWidgetItem(qry.value(5).toString()));
    }
}

filling = false;
}

```

Listing 21 Plik ranking.cpp

W pliku ranking.cpp znajduje się funkcja do wyświetlania rekordów co do danych ćwiczeń. Tworzone jest zapytanie SQL, które wybiera najlepsze wyniki i wyświetla je w tabeli.

Pozostałe niepokazane w listingach funkcje są odwzorowaniem ukazanych w poprzednich listingach funkcji. Np. funkcja do wypełniania listy rozwijanej wygląda tak samo w każdym pliku, tylko zmienione są dane do niej wprowadzane.

3.2. Pliki nagłówkowe .h

```

class adding_exercise : public QDialog
{
    Q_OBJECT

    QSqlDatabase *mydb;

public:
    explicit adding_exercise(QWidget *parent = nullptr, QSqlDatabase *db = nullptr);
    ~adding_exercise();

private slots:

    void on_GroupEx_CB_textActivated(const QString &arg1);
    void on_Name_CB_textActivated(const QString &arg1);

```

```

void on_DeclineB_clicked();
void on_AcceptB_clicked();
void on_addNewExB_clicked();
private:
    Ui::adding_exercise *ui;

    adding_new_exercise *newExerciseWindow;
    void fillComboBox();
    QSqlQuery qry;
    bool filling;
signals:
    void sendDateForm(int, double ,QString, QString);
};


```

Listing 22 Plik adding_exercise.h

W pliku `adding_exercise.h` zawarte są biblioteki dołączone do projektu oraz klasa o tej samej nazwie co plik. Klasa posiada metody do obsługi eventów, do wypełniania listy rozwijanej, pola do obsługi zapytań SQL, zmienną `bool` oraz sygnał przesyłania daty. Funkcje rozwijane są w plikach źródłowych.

Pliki `adding_new_exercise.h` oraz `adding_new_group.h` zawierają te same metody dla obsługi eventów, jak i inne metody do obsługi aplikacji.

```

class Calendar : public QDialog
{
    Q_OBJECT
public:
    explicit Calendar(QWidget *parent = nullptr);
    ~Calendar();
private slots:
    void on_calendarWidget_selectionChanged();
private:
    Ui::Calendar *ui;
signals:
    void sendDate(QDate);
};


```

Listing 23 Plik calendar.h

Plik `calendar.h` w swojej klasie zawiera metodę obsługującą kalendarz oraz funkcje `senddate`.

```

class chartsOption : public QDialog
{
    Q_OBJECT
    QSqlDatabase *mydb;
public:
    explicit chartsOption(QWidget *parent = nullptr, QSqlDatabase *db = nullptr);
    ~chartsOption();


```

```

private slots:
    void on_chartB_clicked();
    void on_ranking_clicked();
    void on_GroupEx_CB_textActivated(const QString &arg1);
    void on_pushButton_clicked();

private:
    Ui::chartsOption *ui;
    ranking *rankingWindow;
    void fillComboBox();
    QSqlQuery qry;
    QSqlQuery qry1;
    bool filling;
};


```

Listing 24 Plik chartoption.h

W pliku chartoption.h oprócz standardowych metod do obsługi eventów znajduje się wskaźnik na obiekty klas ranking.

Plik login.h ma podobną budowę. Zamiast wskaźnika na obiekty klas chart i ranking zawiera wskaźnik na obiekt klasy mainwindow i registeruser.

```

class MainWindow : public QWidget
{
    Q_OBJECT
public:
    MainWindow(QWidget *parent = nullptr, QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE"), QString name = "", QString surname = "");
    ~MainWindow();
private slots:
    void on_Add_B_clicked();
    void receiveDate(QDate);
    void receiveDateForm(int, double ,QString, QString);
    void on_DateB_clicked();
    void on_History_B_clicked();
    void on_tableWidget_cellDoubleClicked();
    void on_CommitB_clicked();
    void on_Graphs_B_clicked();
    void on_exitB_clicked();
private:
    //4 pierwsze były wyżej pod Q_OBJECT
    QSqlDatabase *mydb;
    QSqlDatabase mydbP;
    QString name;
    QString surname;
    //
    Ui::MainWindow *ui;
    void fillForm();
    QString repairDate();
};


```

```

bool filling;
int id;
QSqlQuery qry;
Calendar *calendarWindow;
adding_exercise *windowExercise;
chartsOption *chartsWindow;
QDate trainingDate;
protected:
    void keyPressEvent(QKeyEvent *event) override;
};

```

Listing 25 Plik mainwindow.h

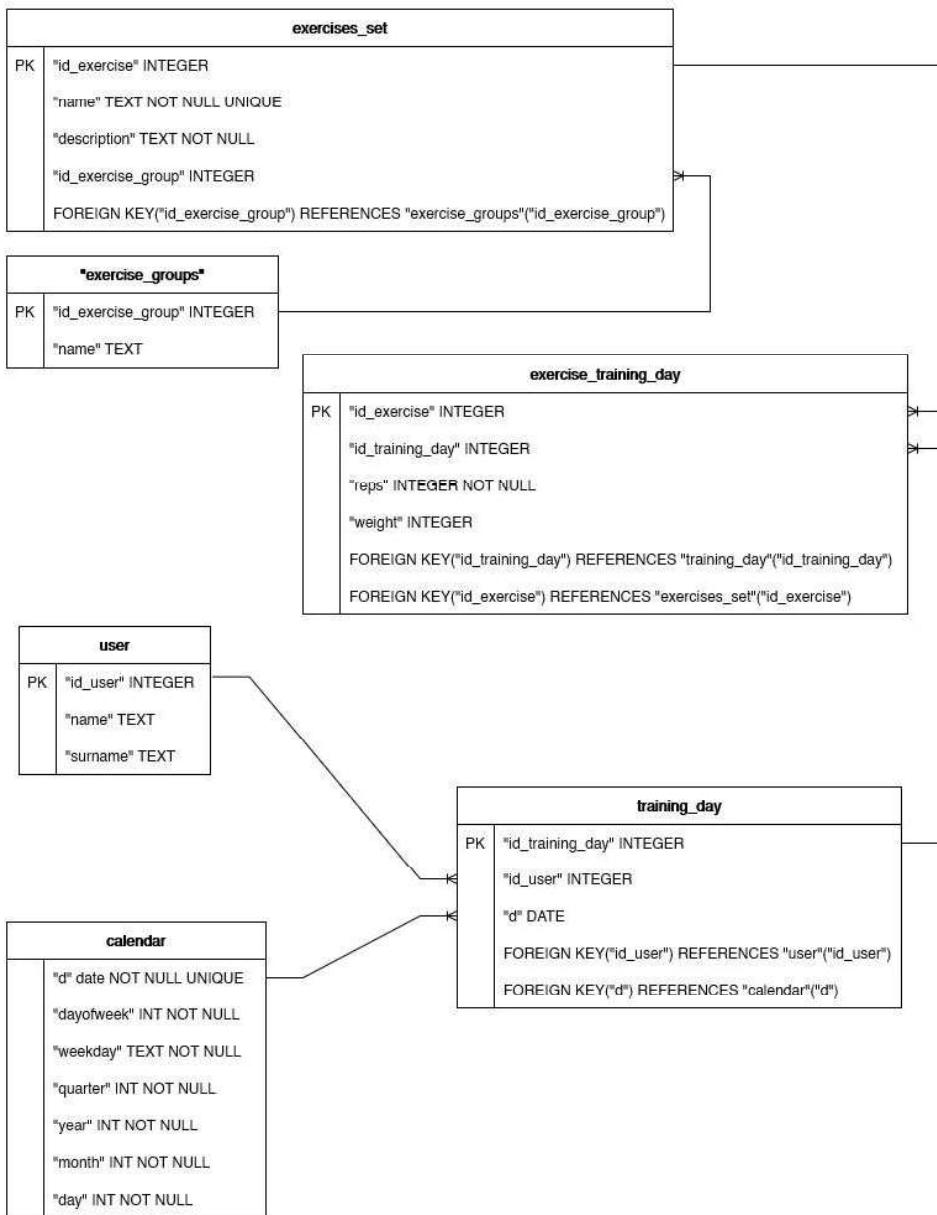
Plik mainwindow.h oprócz wcześniej opisanych metod obsługi eventów i innych metod wykorzystywanych w plikach źródłowych zawiera wskaźnik na obiekty klas calendar, adding_exercise, chartoption. Zapisane są tu także stringi przechowywujące imię i nazwisko zalogowanego użytkownika oraz obiekty klas qdate do obsługi dat oraz qsqldatabase do obsługi bazy danych.

Pliki ranking.h oraz registeruser.h posiadają omawiane wcześniej metody.

3.3. Baza danych

Baza danych napisana w języku SQL składa się z 6 tabel: user, calendar, training_day, exercise_training_day, exercise_groups oraz exercise_set. Pierwsza z nich przechowuje informacje o użytkowniku takie jak nazwa i hasło. Druga informacje o dacie takie jak dzień, tydzień, rok, miesiąc, kwartał. Trzecia przechowuje informacje o użytkowniku oraz o dacie. Jest ona powiązana z dwiema poprzednimi tabelami. Czwarta tabela zapisuje informacje co do treningu takie jak ilość powtórzeń, waga ciężaru oraz informacje o dniu treningowym z poprzedniej tabeli, z którą jest powiązana. Piąta tabela posiada jedynie informacje o nazwie grupy ćwiczeniowej. Ostatnia zawiera informacje co do poszczególnych ćwiczeń oraz grupę ćwiczeniową danego ćwiczenia poprzez powiązanie z poprzednią tabelą.

Poniżej znajduje się diagram bazy danych odzwierciedlający tabela w niej zawarte oraz połączenia między tabelami.



4. Testowanie projektu

W trakcie testowania aplikacji została przetestowana sprawność aplikacji. Sprawdzona została funkcja rejestracji, która przebiegła pomyślnie. Funkcja logowania również przebiegła pomyślnie. Możliwość edycji rekordów przez wpisywanie wartości w pola danego wiersza również działa. Opcja usuwania rekordów przez wybranie go z tabeli i naciśnięcie klawisza backspace przebiega pomyślnie. Przetestowana pomyślnie została też opcja wyboru daty. Dodawanie ćwiczeń, grup ćwiczeń oraz treningów działa bez zarzutu. W oknie dialogowym postępów i rekordów przetestowane zostały obie funkcje. Postępy po wprowadzeniu właściwych danych wyświetlane są poprawnie w tabeli. Najwyższe wyniki danych ćwiczeń są pomyślnie wyświetlane w tabeli w oknie dialogowym rekordów.

5. Wymagania aplikacji

Wymaganiem do uruchomienia aplikacji jest komputer z systemem Windows. Zalecaną wersją jest Windows 10 lub późniejsze.

Aplikacja była uruchamiana bez zarzutów na komputerach o podanych specyfikacjach:

- Procesor: AMD Ryzen 5 4600H, Intel Core i5-10300H
- RAM: 16,0 GB, 8GB
- System: Windows 10 Pro, Windows 10 Home

6. Załączniki

6.1. Kod źródłowy

Kod źródłowy znajduje się w archiwum instalacyjnym w folderze Dziennik_treningowy_source.