

**КОЛЛЕДЖ КОСМИЧЕСКОГО МАШИНОСТРОЕНИЯ И ТЕХНОЛОГИЙ**

**ОТЧЕТ**

по производственной практике ПП.01.01 по модулю ПМ.01  
«Разработка программных модулей программного обеспечения   
для компьютерных систем»

по специальности 09.02.03 «Программирование в компьютерных системах»

Выполнил студент гр. П1-18

Герасимов Д.А.

\_\_\_\_\_\_\_\_\_\_ (подпись)

Принял преподаватель

Гусятинер Л. Б.

\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2021

**Оглавление**

[Введение 3](#_Toc75755457)

[1. Общие сведения об организации 4](#_Toc75755458)

[1.1 Организационная структура и управление деятельностью ККМТ. 4](#_Toc75755459)

[1.2 Организационная структура и управление архивного отдела. 7](#_Toc75755460)

[1.3 Основные задачи архивного отдела. 8](#_Toc75755461)

[1.4 Функции архивного отдела. 8](#_Toc75755462)

[1.5 Нормативные документы, которые регламентируют профессиональную деятельность. 10](#_Toc75755463)

[1.6 Программное обеспечение предприятия. 10](#_Toc75755464)

[1.7 Задачи, подлежащие автоматизации. 10](#_Toc75755465)

[2. Содержание выполняемых видов работ 11](#_Toc75755466)

[2.1 Разработка спецификаций отдельных компонент 11](#_Toc75755467)

[2.2.1 Разработка кода программного продукта на основе готовых спецификаций на уровне модуля для программы Timer 11](#_Toc75755468)

[2.2.2 Разработка кода программного продукта на основе готовых спецификаций на уровне модуля для программы HelperDoc 13](#_Toc75755469)

[2.3.1 Отладка программного модуля с использованием специализированных программных средств по программе Timer 15](#_Toc75755470)

[2.3.2 Отладка программного модуля с использованием специализированных программных средств по программе HelperDoc 15](#_Toc75755471)

[2.4.1 Тестирование программных модулей в программе Timer 16](#_Toc75755472)

[2.4.2 Тестирование программных модулей в программе HelperDoc 17](#_Toc75755473)

[2.5.1 Оптимизация программного кода модуля по программе Timer 21](#_Toc75755474)

[2.5.2 Оптимизация программного кода модуля по программе HelperDoc 22](#_Toc75755475)

[2.6.1 Разработка компонентов проектной и технической документации с использованием графических языков спецификаций для программы Timer 23](#_Toc75755476)

[2.6.2 Разработка компонентов проектной и технической документации с использованием графических языков спецификаций для программы Timer 24](#_Toc75755477)

[Выводы 25](#_Toc75755478)

[Список литературы 26](#_Toc75755479)

[Дневник практики 27](#_Toc75755480)

[Приложения 29](#_Toc75755481)

[Приложение 1 29](#_Toc75755482)

[Приложение 2 38](#_Toc75755483)

[Приложение 3 43](#_Toc75755484)

[Приложение 4 44](#_Toc75755485)

[Приложение 5 54](#_Toc75755486)

[Приложение 6 66](#_Toc75755487)

Введение

На 3 курсе обучения в ККМТ, студентом группы П1-18 Герасимовым Дмитрием была пройдена производственная практика по модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем».

Студент получил и выполнил задание от руководителя, так же изучил организацию, для которой будет разработано данное приложение.

1. Общие сведения об организации

1.1 Организационная структура и управление деятельностью ККМТ.

* KKMT является составной частью организационной структуры Университета.
* Структура и штат ККМТ, a также изменения к ним утверждаются приказом ректора «Технологического университета» по представлению директора ККМТ, согласованному с проректором по учебно-методической работе в установленном порядке.
* KKMT имеет следующее структурное деление:

-отделения по специальностям;

-предметные и цикловые комиссии;

-другие учебные подразделения (лаборатории, мастерские, кабинеты), обеспечивающие образовательный процесс.

* Непосредственное управление деятельностью ККМТ осуществляет директор, назначаемый приказом ректора «Технологического университета», имеющий высшее образование и стаж работы не менее 5 лет на педагогических и руководящих должностях в системе профессиональном образования.
* Полномочия директора ККМТ определяются настоящим Положением, трудовым договором и должностной инструкцией, и другими локальными актами «Технологического университета».
* Заместители директора ККМТ;

-заместитель директора по учебной работе;

-заместитель директора по учебно-методической работе;

-заместитель директора по учебно-производственной работе;

-заместитель директора по учебно-воспитательной работе;

-заместитель директора по научно-методической работе.

* Координацию деятельности ККМТ осуществляет первый проректор и проректор по учебно-методической работе.
* C цепью объединения усилий руководства ККМТ, педагогических работников, учебно-вспомогательного персонала ККМТ для обеспечения качества подготовки специалистов создается Педагогический совет ККМТ.
* B состав Педагогического совета ККМТ входят по должности директор ККМТ, его заместители и руководители структурных подразделений ККМТ. В состав Педагогического совета ККМТ могут также входить сотрудники структурных подразделений «Технологического университета», организаций и учреждений, непосредственно связанных с деятельностью ККМТ по подготовке специалистов.
* Председателем Педагогического совета ККМТ является директор ККМТ.
* К компетенциям Педагогического совета KKMT относятся:

-разработка и предоставление Ученому совету «Технологического университета» на утверждение стратегии развития ККМТ;

-решение вопросов учебной, методической и воспитательной работы, вопросов взаимодействия с институтами «Технологического университета» по обеспечению функционирования системы непрерывного многоуровневого профессиональные образования;

-формирование предложений по контрольным цифрам приема студентов в ККМТ, рассмотрение планов работы приемной комиссии, отчетов по итогам приема и проведению профориентационной кампании;

-ежегодные заслушивание отчета директора ККМТ об основных направлениях деятельности;

-рассмотрение вопросов выдвижения кандидатур из состава работников ККМТ на соискание премий, присвоения почетных званий:

-рассмотрение других вопросов, связанных с деятельностью ККМТ, не относящихся к исключительной компетенции Ученого совета «Технологического университета».

Схема организационной структуры ККМТ на рисунке 1 и 2.

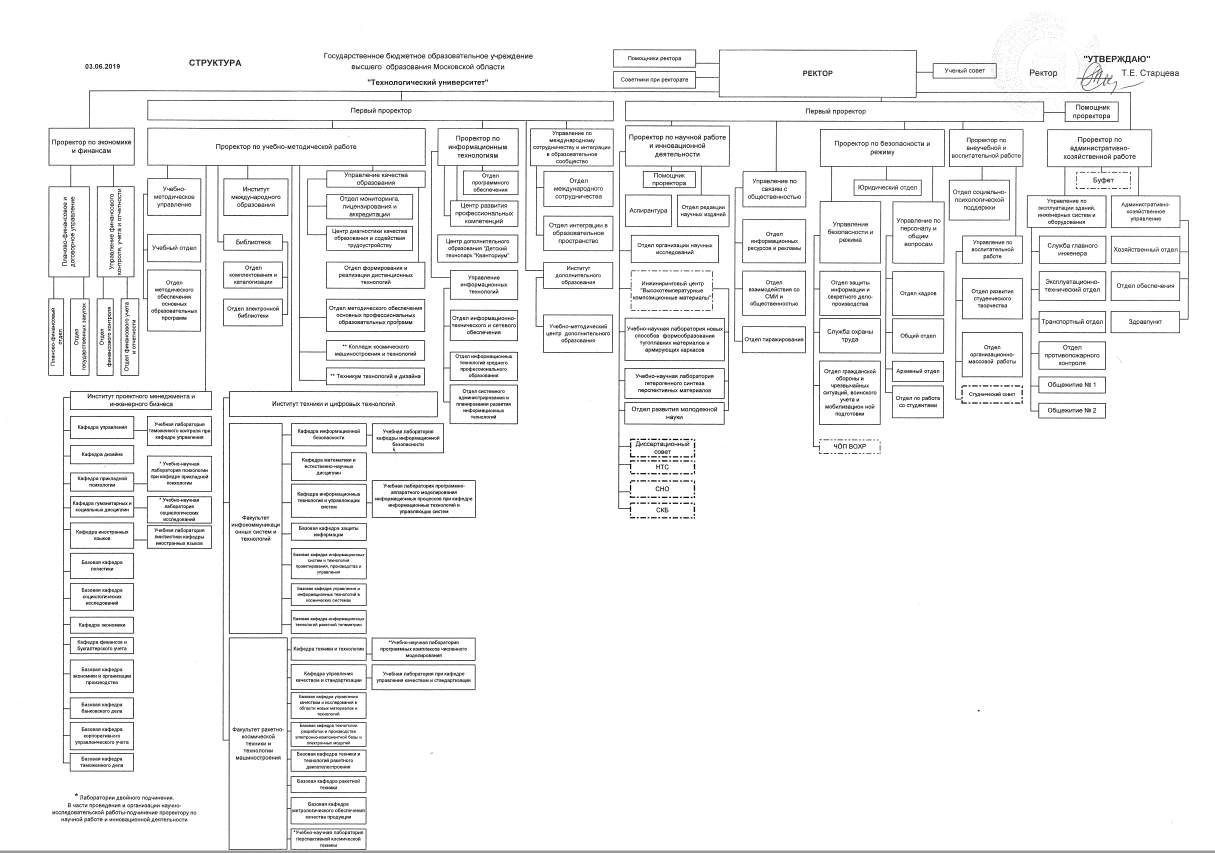


Рис.1 Организационная структура университета

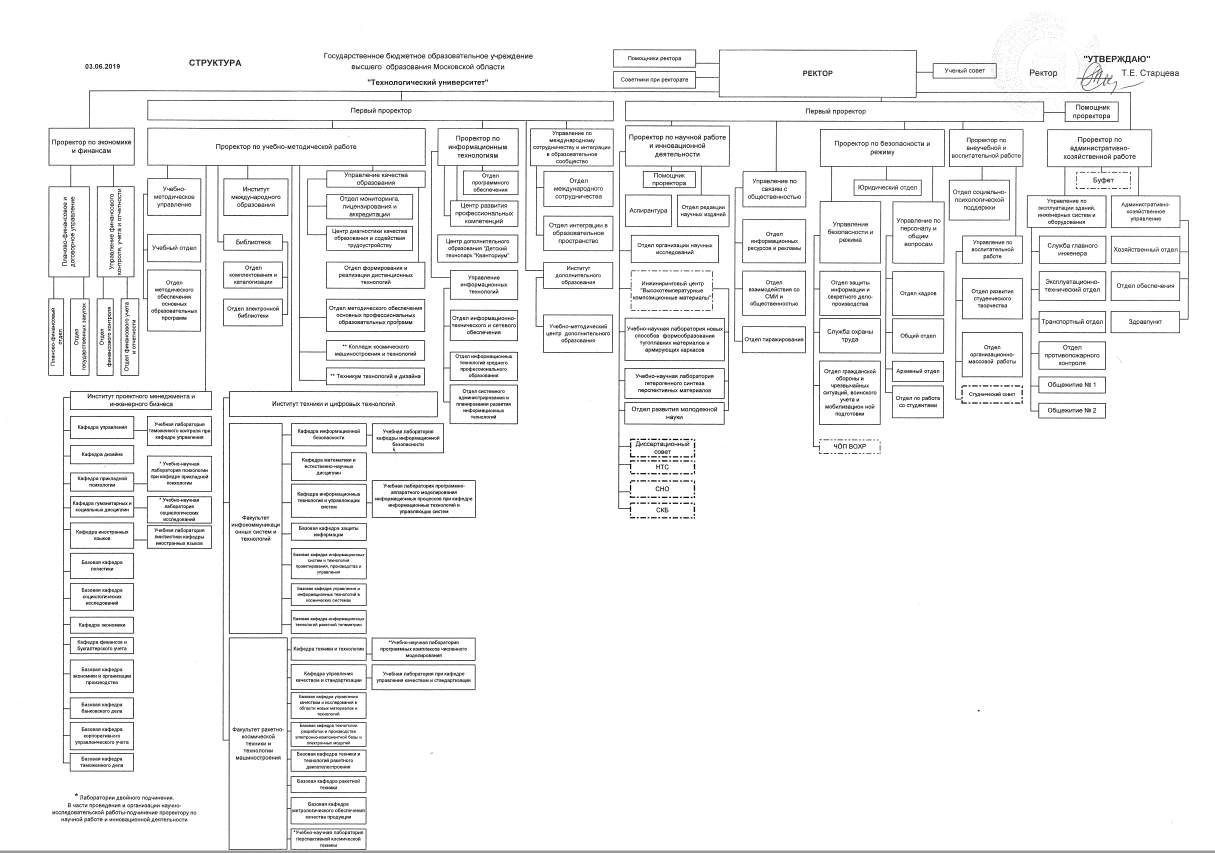


Рис. 2. Организационная структура университета

1.2 Организационная структура и управление архивного отдела.

1. Руководство архивным отделом осуществляет начальник отдела.
2. Структура, численность и штат работников архивного отдела, а также их изменения утверждаются приказом ректора Университета по представлению начальника центра кадровой политики и учета персонала по согласованию с курирующим проректором.
3. Должностные обязанности работников архивного отдела, включая начальника отдела, определяются должностными инструкциями, утверждаемыми ректором Университета.
4. В период временного отсутствия (болезнь, отпуск, командировка) начальника отдела его обязанности исполняет лицо, назначенное приказом ректора Университета, которое несет ответственность за надлежащее их исполнение.

1.3 Основные задачи архивного отдела.

Архивный отдел создан для организации, руководства, координации, контроля и совершенствования работы по обеспечению системы приема, учета и хранения документов в Университете. Исходя из поставленных целей, архивный отдел решает следующие задачи:

* Комплектование документами, состав которых предусмотрен разделом 3 настоящего Положения.
* Учет, обеспечение сохранности, создание научно-справочного аппарата, использование документов, хранящихся в архивном отделе.
* Организационно-методическое руководство, координация и контроль деятельности структурных подразделений по вопросам организации архивного дела.
* Подготовка и своевременная передача документов на государственное хранение с соблюдением требований, устанавливаемых государственной архивной службой.
* Внедрение новых методов организации работы, в том числе на основе использования современных информационных технологий.

1.4 Функции архивного отдела.

В соответствии с возложенными задачами архивный отдел осуществляет следующие функции:

* Планирует, принимает, учитывает и хранит документы структурных подразделений Университета, обработанные в соответствии с требованиями нормативных документов.
* Разрабатывает и внедряет современные унифицированные формы документов, утвержденные нормативными правовыми актами Университета.
* Осуществляет учет и обеспечивает полную сохранность принятых на хранение дел, соблюдение в помещениях архивного отдела условий, необходимых для обеспечения сохранности документов.
* Создает, пополняет и совершенствует научно-справочный аппарат к хранящимся в архивном отделе делам и документам.
* Обеспечивает использование документов:

-информирует курирующего проректора, начальника центра кадровой политики и учета персонала и работников Университета о составе и содержании документов архивного отдела; - выдает в установленном порядке дела, документы или копии документов в целях служебного и научного использования;

-исполняет запросы организаций и заявления граждан об установлении трудового стажа и другим вопросам социально-правового характера, в установленном порядке выдает копии документов и архивные справки;

-ведет учет использования документов, хранящихся в архивном отделе.

* Проводит экспертизу ценности документов, хранящихся в архивном отделе, организует и принимает участие в работе Экспертной комиссии Университета.
* Участвует в работе по составлению номенклатуры дел Университета, оказывает методическую и практическую помощь структурным подразделениям в ее составлении, совместно с общим отделом готовит проект сводной номенклатуры дел Университета.
* Контролирует правильность формирования и оформления дел в делопроизводстве, а также подготовку дел к передаче в архивный отдел.

1.5 Нормативные документы, которые регламентируют профессиональную деятельность.

* Номенклатура дел института.
* Положение об институте
* Должностные инструкции персонала института
* Документы по планированию деятельности института техника и цифровых технологий.
* Перечень нормативной и технической документации.
  1. Программное обеспечение предприятия.
* Windows 10
* Git version 2.24.1.2
* Goggle Chrome
* Android Studio
* IntelliJ IDEA Community Edition 2019.3
* Excel 2013
* DB Browser for SQLite Version 3.12.1
* JetBrains PyCharm Community Edition 2019.2.1

1.7 Задачи, подлежащие автоматизации.

* + - Внутренняя система анти-плагиата

1. Содержание выполняемых видов работ

2.1 Разработка спецификаций отдельных компонент

Общее задание было разделено на 3 этапа:

1. Изучение языка программирования Java. Работа с IntelliJ IDEA.
2. Изучение платформы Android. Работа с Android Studio.
3. Написание итогового проекта.
4. Написание помощника для работы в архиве

2.2.1 Разработка кода программного продукта на основе готовых спецификаций на уровне модуля для программы Timer

Модуль смены отображения текста кнопки приведён в Листинге 1. Данный модуль осуществляет смену отображения текста на кнопке в зависимости от её активности.

**Листинг 1. Модуль смены отображения текста кнопки.**

public void onClickMode(View view) {

if (!running) {

running = true;

mode.setText("Pause");

} else {

running = false;

mode.setText("Resume");

}

}

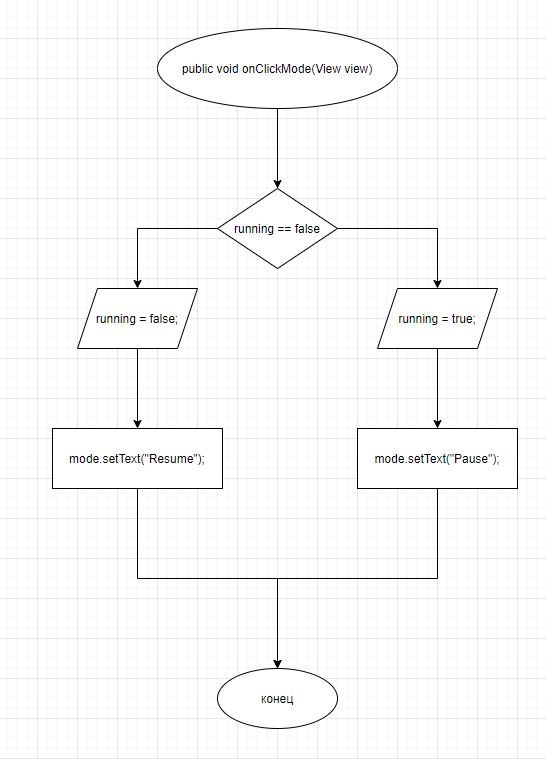


Рис. 3. Блок-схема листинга 1

2.2.2 Разработка кода программного продукта на основе готовых спецификаций на уровне модуля для программы HelperDoc

Модуль для загрузки данных в визуальную таблицу в Листинге 2. Данный модуль загружает данные из базы данных в визуальную Таблицу.

**Листинг 2. Модуль загрузки данных.**

def load\_data(self):

"""

Загружает все данные из бд в GUI

"""

all\_id = self.get\_all\_id()

table\_row = 0

self.ui.table.setRowCount(len(all\_id))

for pk in all\_id:

data = self.helper.create\_row(pk)

for i in range(self.ui.table.columnCount()):

self.ui.table.setItem(table\_row, i, QtWidgets.QTableWidgetItem(str(data[i])))

table\_row += 1

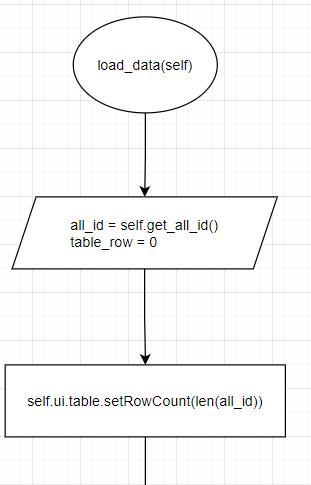


Рис. 4. Блок-схема листинга 2. Часть 1

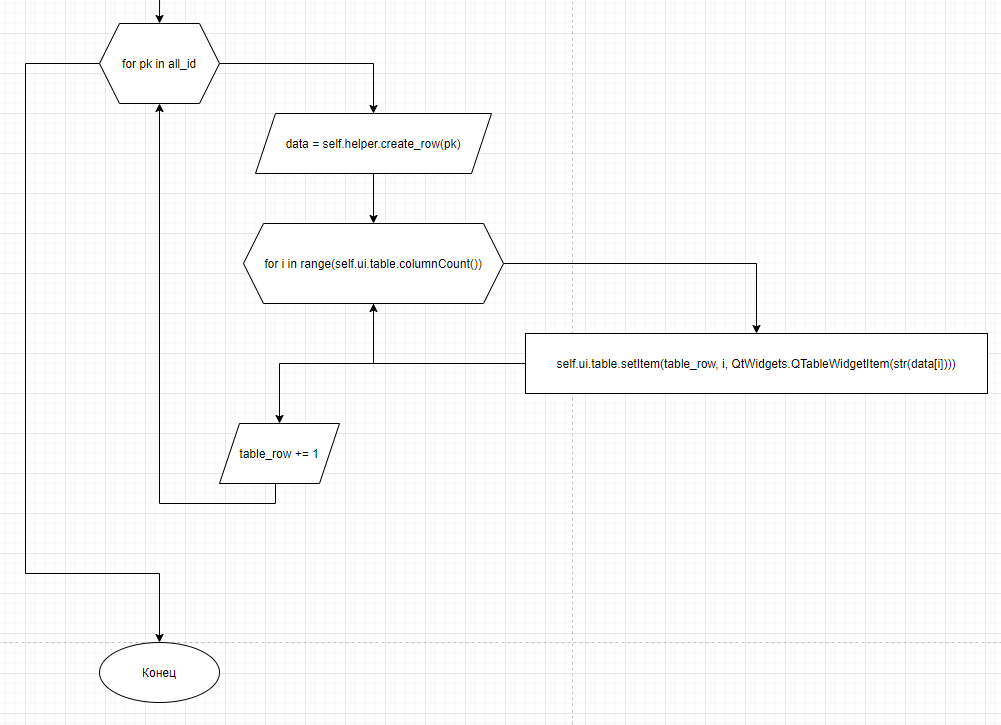


Рис. 5. Блок-схема листинга 2. Часть 2

2.3.1 Отладка программного модуля с использованием специализированных программных средств по программе Timer

Отладка проводилась с помощью функции “Make Project” в Android Studio.

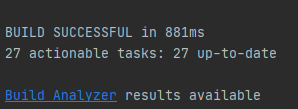


Рис. 6. Успешная сборка проекта

2.3.2 Отладка программного модуля с использованием специализированных программных средств по программе HelperDoc

Отладка проводилась с помощью Дебагера в Pycharm.

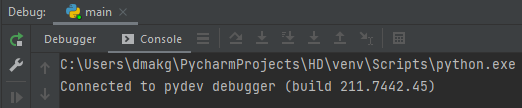


Рис. 7. Успешная сборка проекта

2.4.1 Тестирование программных модулей в программе Timer

Тестирование приложения с использованием входных данных введённых лично пользователем.

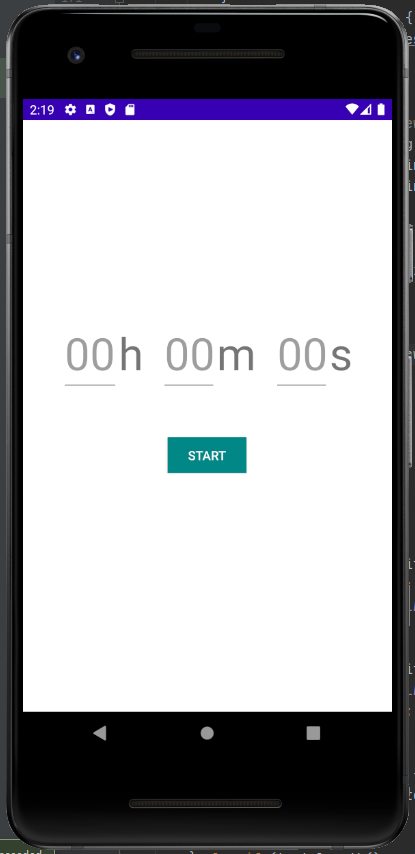


Рис. 8. Стартовый экран

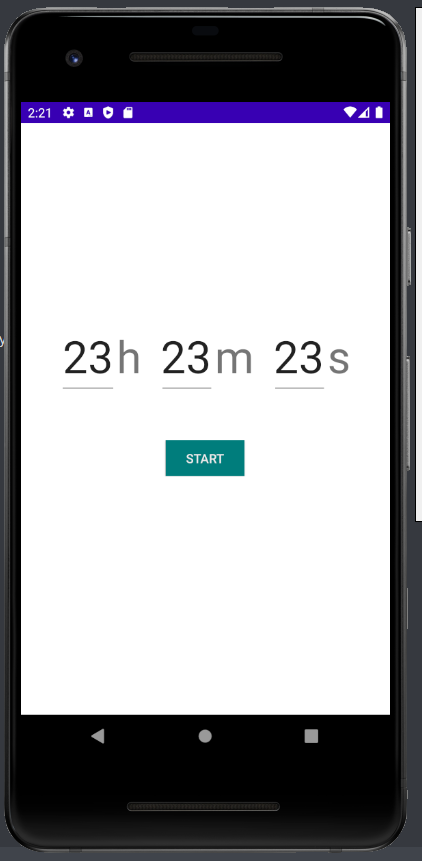


Рис. 9. Данные введённые пользователем

Выходных файлов у программы не имеется. Об окончании работы таймера пользователь узнает с помощью музыки, которая начнёт проигрываться в течение 10 секунд.

2.4.2 Тестирование программных модулей в программе HelperDoc

Тестирование приложения с использованием входных данных введённых из базы данных.

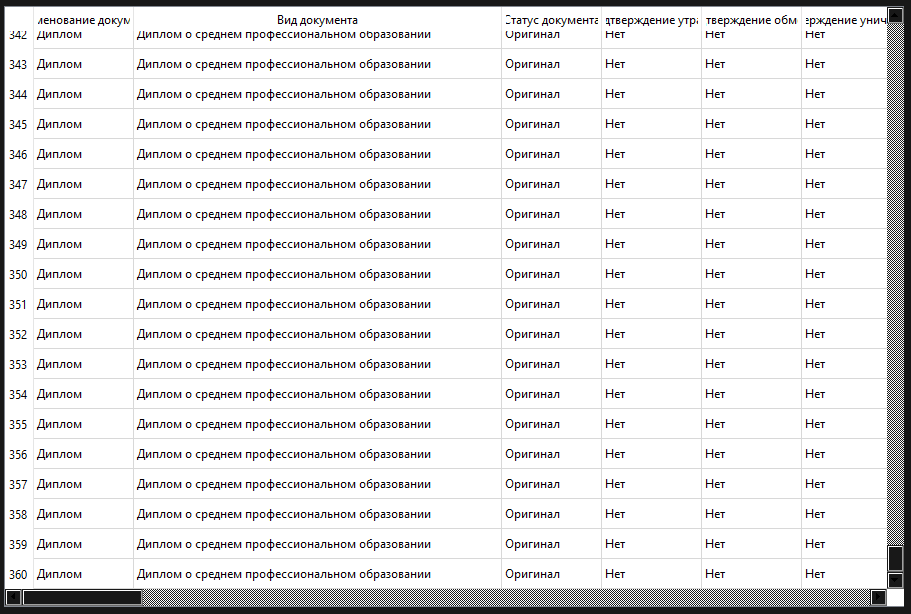


Рис 10. Входящие данные

Выходными данными является файлы Word-файлы. Они создаются при нажатии на кнопку “Создать документ”.

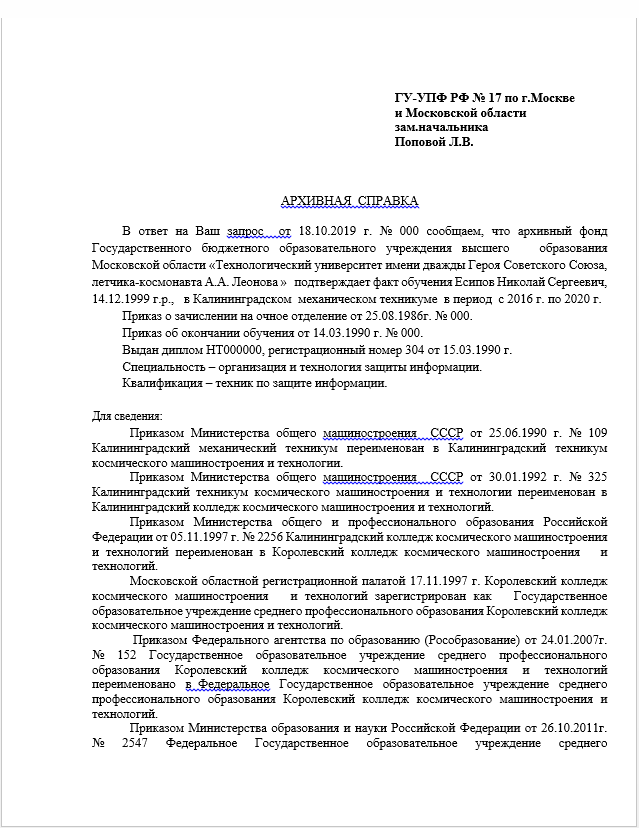


Рис 11. Выходной Word-файл. Часть 1.

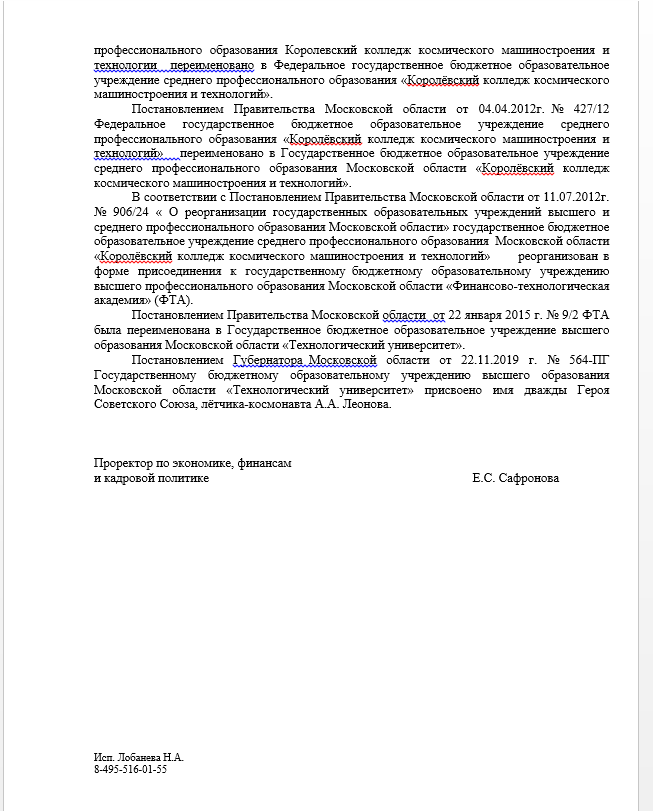


Рис 12. Выходной Word-файл. Часть 2.

2.5.1 Оптимизация программного кода модуля по программе Timer

Была оптимизирована функция определения текущего условия передачи в зависимости от времени в Листинге 4. Для оптимизации была удалена вся инициализация переменных из метода stopTimer() и вновь воссоздана но уже как свойство класса. Этим было ускорено быстродействие программы, а также уменьшено использование памяти.

**Листинг 3. Модуль остановки таймера до оптимизации.**

private void stopTimer() {

EditText hours = findViewById(R.id.hours);

EditText minutes = findViewById(R.id.minutes);

EditText seconds = findViewById(R.id.seconds);

TextView hoursText = findViewById(R.id.hours\_text);

TextView minutesText = findViewById(R.id.minutes\_text);

TextView secondsText = findViewById(R.id.seconds\_text);

Button start = findViewById(R.id.start);

Button cancel = findViewById(R.id.cancel);

Button mode = findViewById(R.id.mode);

running = false;

cancel.setVisibility(View.GONE);

mode.setVisibility(View.GONE);

start.setVisibility(View.VISIBLE);

// Даем возможность редактирования пользователю

MainActivityHelper.EditTextVisible(hours, hoursText);

MainActivityHelper.EditTextVisible(minutes, minutesText);

MainActivityHelper.EditTextVisible(seconds, secondsText);

hoursText.setText("");

minutesText.setText("");

secondsText.setText("");

}

**Листинг 4. Модуль остановки таймера после оптимизации.**

private void stopTimer() {

running = false;

cancel.setVisibility(View.GONE);

mode.setVisibility(View.GONE);

start.setVisibility(View.VISIBLE);

// Даем возможность редактирования пользователю

MainActivityHelper.EditTextVisible(hours, hoursText);

MainActivityHelper.EditTextVisible(minutes, minutesText);

MainActivityHelper.EditTextVisible(seconds, secondsText);

hoursText.setText("");

minutesText.setText("");

secondsText.setText("");

}

2.5.2 Оптимизация программного кода модуля по программе HelperDoc

Было оптимизированно добавление библиотек в программы.

**Листинг 5. Добавление библиотеки для работы с GUI до оптимизации.**

import PyQt6

**Листинг 6. Добавление библиотеки для работы с GUI после оптимизации.**

from PyQt6 import QtWidgets

from PyQt6.QtGui import QIcon

from PyQt6.QtWidgets import QMessageBox

from docxtpl import DocxTemplate

2.6.1 Разработка компонентов проектной и технической документации с использованием графических языков спецификаций для программы Timer

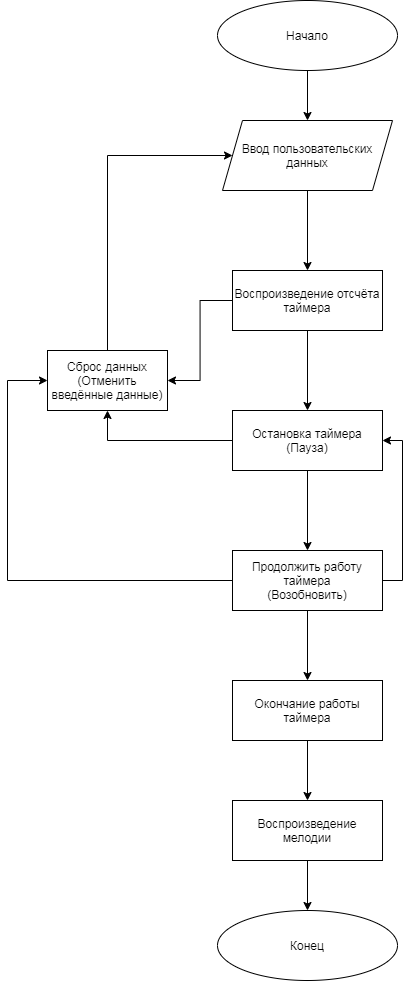


Рис 13. Общая блок-схема модуля

2.6.2 Разработка компонентов проектной и технической документации с использованием графических языков спецификаций для программы Timer

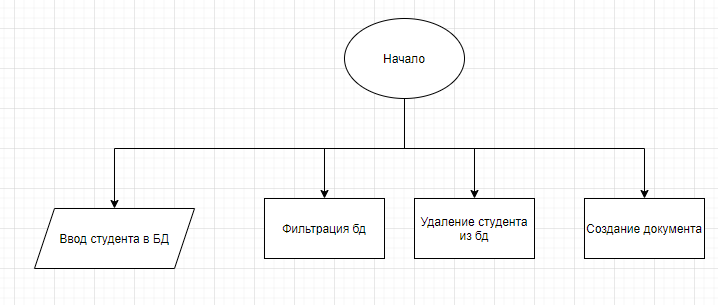


Рис 14. Общая блок-схема модуля

Выводы

Полученные навыки:

* + - Работа с IntelliJ IDEA
    - Работа с Android Studio
    - Сборка приложения в Android Studio
    - Работа с Pycharm

Полученные умения:

* + - Разработка программ на Java
    - Разработка Android-приложений
    - Работа с xml-файлами
    - Разработка GUI с помощью PyQt6

Список литературы

1. MS Windows 10 Pro Edu: <https://www.microsoft.com/>
2. MS Office 2013 Pro Plus : <https://www.microsoft.com/>
3. 7-Zip 19.00 (x64) : <https://www.7-zip.org/>
4. Android Studio : <https://developer.android.com/studio>
5. Git version 2.24.1.2 : <https://git-scm.com/>
6. Goggle Chrome : <https://www.google.com/intl/ru/chrome/>
7. IntelliJ IDEA Community Edition 2019.3 : <https://www.jetbrains.com/>
8. JetBrains PyCharm Community Edition 2019.2.1 : <https://www.jetbrains.com/>
9. Head First Android Development: A Brain-Friendly Guide. Гриффитс Дэвид, Гриффитс Дон.

Дневник практики

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работ | Отметка о выполнении |
| 27.04 | Установка и настройка среды JetBrains IntelliJ IDEA |  |
| 28.04 | Примитивные типы |  |
| 29.04 | Преобразование типов |  |
| 30.04 | Управляющие конструкции: условные операторы и циклы |  |
| 03.04 | Массивы и строки |  |
| 04.05 | Массивы и строки |  |
| 05.05-06.05 | Процедуры и функции |  |
| 07.05 | Пакеты и модификаторы доступа |  |
| 10.05 | Объявление класса |  |
| 11.05-12.05 | Наследование. Класс Object |  |
| 13.05-14.05 | Абстрактные классы и интерфейсы |  |
| 17.05 | Параметризованные классы (generic) |  |
| **18.05** | Параметризованные классы (generic) |  |
| 19.05-20.05 | Обработка исключений. Try-catch |  |
| 21.05 | Функциональные интерфейсы |  |
| 24.05-25.05 | Функциональные интерфейсы |  |
| 26.05 | Статические переменные и методы |  |
| 27.05-28.05 | Списки и динамические массивы |  |
| 31.05-02.06 | Множество и Словарь |  |
| 03.06-04.06 | Установка и настройка среды Android Studio |  |
| 07.06-09.06 | Создание приложений с активностью |  |
| 10.06 | Интент |  |
| 11.06 | Создание приложений с несколькими активностями |  |
| 14.06-15.06 | Создание приложений с несколькими активностями |  |
| 16.06-18.06 | Макеты и как с ними работать |  |
| 21.06-22.06 | Создание итоговых приложений |  |
| 23.06 | Тестирование и отладка приложения |  |
| 24.05-25.05 | Разработка документации |  |
| 28.05 | Сдача отчёта по практике |  |

Приложения

Приложение 1

Листинг 7. MainActivity.java

package com.example.timer;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.media.MediaPlayer;

import android.os.Bundle;

import android.os.Handler;

import android.text.Editable;

import android.text.InputType;

import android.view.View;

import android.view.WindowManager;

import android.widget.Button;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.TextView;

public class MainActivity extends Activity {

private EditText hours;

private EditText minutes;

private EditText seconds;

private TextView hoursText;

private TextView minutesText;

private TextView secondsText;

private Button start;

private Button cancel;

private Button mode;

private MediaPlayer sound;

private boolean running = false;

private TimeHelper timeHelper;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

hours = findViewById(R.id.hours);

minutes = findViewById(R.id.minutes);

seconds = findViewById(R.id.seconds);

hoursText = findViewById(R.id.hours\_text);

minutesText = findViewById(R.id.minutes\_text);

secondsText = findViewById(R.id.seconds\_text);

start = findViewById(R.id.start);

cancel = findViewById(R.id.cancel);

mode = findViewById(R.id.mode);

sound = MediaPlayer.create(this, R.raw.never\_gonna\_give\_you\_up);

timeHelper = new TimeHelper(hoursText, minutesText, secondsText);

if (savedInstanceState != null) {

running = savedInstanceState.getBoolean("running");

hours.setText(savedInstanceState.getString("hours"));

minutes.setText(savedInstanceState.getString("minutes"));

seconds.setText(savedInstanceState.getString("seconds"));

hoursText.setText(savedInstanceState.getString("hoursText"));

minutesText.setText(savedInstanceState.getString("minutesText"));

secondsText.setText(savedInstanceState.getString("secondsText"));

// Если был запущен код

if (running || secondsText.getText().length() != 0) {

start.setVisibility(View.GONE);

cancel.setVisibility(View.VISIBLE);

mode.setVisibility(View.VISIBLE);

mode.setText("Pause");

// Убираем возможность изменять

MainActivityHelper.TextViewVisible(hours, hoursText);

MainActivityHelper.TextViewVisible(minutes, minutesText);

MainActivityHelper.TextViewVisible(seconds, secondsText);

}

}

runTimer();

}

@Override

protected void onSaveInstanceState(@NonNull Bundle outState) {

super.onSaveInstanceState(outState);

outState.putBoolean("running", running);

outState.putString("hours", hours.getText().toString());

outState.putString("minutes", minutes.getText().toString());

outState.putString("seconds", seconds.getText().toString());

outState.putString("hoursText", hoursText.getText().toString());

outState.putString("minutesText", minutesText.getText().toString());

outState.putString("secondsText", secondsText.getText().toString());

}

private void runTimer() {

final Handler handler = new Handler();

handler.post(new Runnable() {

@SuppressLint("DefaultLocale")

@Override

public void run() {

handler.postDelayed(this, 1000);

boolean isEnd;

if (running) {

isEnd = timeHelper.minusSecond();

if (isEnd) {

stopTimer();

soundPlay();

}

}

}

});

}

private void stopTimer() {

running = false;

cancel.setVisibility(View.GONE);

mode.setVisibility(View.GONE);

start.setVisibility(View.VISIBLE);

// Даем возможность редактирования пользователю

MainActivityHelper.EditTextVisible(hours, hoursText);

MainActivityHelper.EditTextVisible(minutes, minutesText);

MainActivityHelper.EditTextVisible(seconds, secondsText);

hoursText.setText("");

minutesText.setText("");

secondsText.setText("");

}

private void soundPlay() {

sound.start();

}

public void onClickStart(View view) {

boolean isHoursZero = MainActivityHelper.isZero(hours);

boolean isMinutesZero = MainActivityHelper.isZero(minutes);

boolean isSecondsZero = MainActivityHelper.isZero(seconds);

System.out.println("[" + isHoursZero + " " + isMinutesZero + " " + isSecondsZero + "]");

if (isHoursZero || isMinutesZero || isSecondsZero) {

running = true;

start.setVisibility(View.GONE);

cancel.setVisibility(View.VISIBLE);

mode.setVisibility(View.VISIBLE);

mode.setText("Pause");

// Убираем возможность изменять

MainActivityHelper.TextViewVisible(hours, hoursText);

MainActivityHelper.TextViewVisible(minutes, minutesText);

MainActivityHelper.TextViewVisible(seconds, secondsText);

// Получаем введенные данные

int hoursValue = MainActivityHelper.getInt(hours);

int minutesValue = MainActivityHelper.getInt(minutes);

int secondsValue = MainActivityHelper.getInt(seconds);

// Приводим в правильный вид

if (secondsValue > 59) {

minutesValue += secondsValue / 60;

secondsValue %= 60;

}

if (minutesValue > 59) {

hoursValue += minutesValue / 60;

minutesValue %= 60;

}

// Устанавливаем TextView значения

hoursText.setText(String.valueOf(hoursValue));

minutesText.setText(String.valueOf(minutesValue));

secondsText.setText(String.valueOf(secondsValue));

}

}

public void onClickCancel(View view) {

stopTimer();

}

public void onClickMode(View view) {

if (!running) {

running = true;

mode.setText("Pause");

} else {

running = false;

mode.setText("Resume");

}

}

}

class MainActivityHelper {

public static void TextViewVisible(EditText et, TextView tv) {

et.setVisibility(View.GONE);

tv.setVisibility(View.VISIBLE);

}

public static void EditTextVisible(EditText et, TextView tv) {

et.setVisibility(View.VISIBLE);

tv.setVisibility(View.GONE);

}

public static int getInt(EditText et) {

String text = et.getText().toString();

if (text.equals("")) {

return 0;

} else if (text.length() == 2 && text.charAt(0) == '0') {

return (int)text.charAt(1);

}

return Integer.parseInt(text);

}

public static boolean isZero(EditText et) {

boolean state = true;

if (et.getText().length() == 0

|| et.getText().equals("00")

|| et.getText().equals("0")){

state = false;

}

return state;

}

}

class TimeHelper {

final private TextView hoursText;

final private TextView minutesText;

final private TextView secondsText;

public TimeHelper(TextView hoursText, TextView minutesText, TextView secondsText) {

this.hoursText = hoursText;

this.minutesText = minutesText;

this.secondsText = secondsText;

}

public boolean minusSecond() {

boolean isEnd = false;

int second = Integer.parseInt(secondsText.getText().toString());

if (second == 0) {

isEnd = minusMinute();

secondsText.setText("59");

} else {

secondsText.setText(String.valueOf(second - 1));

}

return isEnd;

}

public boolean minusMinute() {

boolean isEnd = false;

int minute = Integer.parseInt(minutesText.getText().toString());

if (minute == 0) {

isEnd = minusHour();

minutesText.setText("59");

} else {

minutesText.setText(String.valueOf(minute - 1));

}

return isEnd;

}

public boolean minusHour() {

boolean isEnd = false;

int hour = Integer.parseInt(hoursText.getText().toString());

if (hour == 0) {

isEnd = true;

} else {

hoursText.setText(String.valueOf(hour - 1));

}

return isEnd;

}

}

Приложение 2

Листинг 8. main\_activity.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="16dp"

android:gravity="center"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginBottom="50dp"

android:orientation="vertical"

android:gravity="center"

>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:gravity="center\_horizontal"

>

<EditText

android:id="@+id/hours"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="10dp"

android:textSize="50sp"

android:hint="@string/hours"

android:inputType="number"

android:maxLength="2"

/>

<TextView

android:id="@+id/hours\_text"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="12dp"

android:layout\_marginRight="4dp"

android:textSize="50sp"

android:visibility="gone"

android:text=""

android:textColor="@color/black"

/>

<TextView

android:id="@+id/hours\_sign"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginRight="10dp"

android:textSize="50sp"

android:text="@string/hours\_sign"

/>

<EditText

android:id="@+id/minutes"

android:layout\_width="62dp"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="10dp"

android:textSize="50sp"

android:hint="@string/minutes"

android:inputType="number"

android:maxLength="2"

/>

<TextView

android:id="@+id/minutes\_text"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="12dp"

android:layout\_marginRight="4dp"

android:textSize="50sp"

android:visibility="gone"

android:text=""

android:textColor="@color/black"

/>

<TextView

android:id="@+id/minutes\_sign"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginRight="10dp"

android:textSize="50sp"

android:text="@string/minutes\_sign"

/>

<EditText

android:id="@+id/seconds"

android:layout\_width="62dp"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="10dp"

android:textSize="50sp"

android:hint="@string/seconds"

android:inputType="number"

android:maxLength="2"

/>

<TextView

android:id="@+id/seconds\_text"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="12dp"

android:layout\_marginRight="4dp"

android:textSize="50sp"

android:visibility="gone"

android:text=""

android:textColor="@color/black"

/>

<TextView

android:id="@+id/seconds\_sign"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginRight="10dp"

android:textSize="50sp"

android:text="@string/seconds\_sign"

/>

</LinearLayout>

<LinearLayout

android:id="@+id/buttons"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:gravity="center\_horizontal"

android:layout\_marginTop="50dp"

>

<Button

android:id="@+id/start"

android:layout\_width="wrap\_content"

android:layout\_height="40dp"

android:layout\_marginHorizontal="50dp"

android:onClick="onClickStart"

android:text="@string/start"

android:background="@color/teal\_700"

android:textColor="@color/white"

tools:ignore="UsingOnClickInXml" />

<Button

android:id="@+id/cancel"

android:layout\_width="wrap\_content"

android:layout\_height="40dp"

android:layout\_marginHorizontal="50dp"

android:onClick="onClickCancel"

android:text="@string/cancel"

android:visibility="gone"

android:background="@color/teal\_700"

android:textColor="@color/white"

tools:ignore="UsingOnClickInXml" />

<Button

android:id="@+id/mode"

android:layout\_width="wrap\_content"

android:layout\_height="40dp"

android:layout\_marginHorizontal="50dp"

android:onClick="onClickMode"

android:text="@string/mode"

android:visibility="gone"

android:background="@color/teal\_700"

android:textColor="@color/white"

tools:ignore="UsingOnClickInXml" />

</LinearLayout>

</LinearLayout>

</RelativeLayout>

Приложение 3

Листинг 9. strings.xml

<resources>

<string name="app\_name">Timer</string>

<string name="hours">00</string>

<string name="hours\_sign">h</string>

<string name="minutes">00</string>

<string name="minutes\_sign">m</string>

<string name="seconds">00</string>

<string name="seconds\_sign">s</string>

<string name="start">Start</string>

<string name="cancel">Cancel</string>

<string name="mode">Pause</string>

</resources>

Приложение 4

Листинг 10. main.py

import sys

from PyQt6 import QtWidgets

from PyQt6.QtGui import QIcon

from PyQt6.QtWidgets import QMessageBox

from docxtpl import DocxTemplate

from design import Ui\_HelperDoc

from helper import Helper

class HelperDoc(QtWidgets.QMainWindow):

def \_\_init\_\_(self):

"""

Главный класс. Объединяет GUI и функциональную часть

"""

super().\_\_init\_\_()

self.helper = Helper('db.db')

self.word = DocxTemplate("template.docx")

self.ui = Ui\_HelperDoc()

self.ui.setupUi(self)

self.init\_UI()

def init\_UI(self):

"""

Добавляет функционал всем кнопкам

:return:

"""

self.setWindowTitle('HelperDoc')

self.setWindowIcon(QIcon('favicon.ico'))

self.ui.suggest\_input.setPlaceholderText('Поиск...')

self.load\_data()

self.ui.clear.clicked.connect(lambda: self.clear\_data())

self.ui.add\_row.clicked.connect(lambda: self.add\_row())

self.ui.suggest\_input\_button.clicked.connect(lambda: self.find\_rows())

self.ui.table.setEditTriggers(QtWidgets.QTableWidget.EditTrigger.NoEditTriggers)

self.ui.remove\_row.clicked.connect(lambda: self.remove\_row())

self.ui.create\_doc.clicked.connect(lambda: self.create\_doc())

def get\_all\_id(self):

"""

:return: список всех id из таблицы 'table\_document'

"""

# [(1,), (2,), (3,)]

all\_id = self.helper.bd.select(self.helper.table\_document, 'id', {})

# [1, 2, 3]

return [pk[0] for pk in all\_id]

def load\_data(self):

"""

Загружает все данные из бд в GUI

"""

all\_id = self.get\_all\_id()

table\_row = 0

self.ui.table.setRowCount(len(all\_id))

for pk in all\_id:

data = self.helper.create\_row(pk)

for i in range(self.ui.table.columnCount()):

self.ui.table.setItem(table\_row, i, QtWidgets.QTableWidgetItem(str(data[i])))

table\_row += 1

def get\_row\_by\_position(self, position, type\_dict=False):

"""

:param position: позиция нужной строки в бд

:param type\_dict: тип возвращаемых данных (true - dict; false - list)

:return: Строка типа dict или list

"""

if not type\_dict:

row = list()

for col in range(self.ui.table.columnCount()):

row.append(self.ui.table.item(position, col).text())

else:

row = {

'title\_doc': self.ui.table.item(position, 0).text(),

'type\_doc': self.ui.table.item(position, 1).text(),

'status\_doc': self.ui.table.item(position, 2).text(),

'loss\_confirmation': self.ui.table.item(position, 3).text(),

'confirmation\_exchange': self.ui.table.item(position, 4).text(),

'destruction\_confirmation': self.ui.table.item(position, 5).text(),

'education\_level': self.ui.table.item(position, 6).text(),

'series\_doc': self.ui.table.item(position, 7).text(),

'number\_doc': self.ui.table.item(position, 8).text(),

'issue\_date': self.ui.table.item(position, 9).text(),

'registration\_number': self.ui.table.item(position, 10).text(),

'code\_profession': self.ui.table.item(position, 11).text(),

'title\_profession': self.ui.table.item(position, 12).text(),

'title\_qualification': self.ui.table.item(position, 13).text(),

'education\_program': self.ui.table.item(position, 14).text(),

'admission\_year': self.ui.table.item(position, 15).text(),

'graduation\_year': self.ui.table.item(position, 16).text(),

'study\_period': self.ui.table.item(position, 17).text(),

'last\_name': self.ui.table.item(position, 18).text(),

'name': self.ui.table.item(position, 19).text(),

'middle\_name': self.ui.table.item(position, 20).text(),

'birth\_date': self.ui.table.item(position, 21).text(),

'gender': self.ui.table.item(position, 22).text(),

'snills': self.ui.table.item(position, 23).text(),

'country\_code': self.ui.table.item(position, 24).text(),

'education\_form': self.ui.table.item(position, 25).text(),

'education\_receipt\_form': self.ui.table.item(position, 26).text(),

'financing\_source': self.ui.table.item(position, 27).text(),

}

return row

def clear\_data(self):

"""

Очистка боковой панели от введенных данных

"""

self.ui.title\_doc\_line.clear()

self.ui.type\_doc\_line.clear()

self.ui.status\_doc\_line.clear()

self.ui.loss\_confirmation\_line.clear()

self.ui.confirmation\_exchange\_line.clear()

self.ui.destruction\_confirmation\_line.clear()

self.ui.education\_level\_line.clear()

self.ui.series\_doc\_line.clear()

self.ui.number\_doc\_line.clear()

self.ui.issue\_date\_line.clear()

self.ui.registration\_number\_line.clear()

self.ui.code\_profession\_line.clear()

self.ui.title\_profession\_line.clear()

self.ui.title\_qualification\_line.clear()

self.ui.education\_program\_line.clear()

self.ui.admission\_year\_line.clear()

self.ui.graduation\_year\_line.clear()

self.ui.study\_period\_line.clear()

self.ui.last\_name\_line.clear()

self.ui.name\_line.clear()

self.ui.middle\_name\_line.clear()

self.ui.birth\_date\_line.clear()

self.ui.gender\_line.clear()

self.ui.snills\_line.clear()

self.ui.country\_code\_line.clear()

self.ui.education\_form\_line.clear()

self.ui.education\_receipt\_form\_line.clear()

self.ui.financing\_source\_line.clear()

def get\_inputted\_data(self):

"""

Вернет полученные данные из боковой панели

"""

data = list()

data.append(self.ui.title\_doc\_line.text())

data.append(self.ui.type\_doc\_line.text())

data.append(self.ui.status\_doc\_line.text())

data.append(self.ui.loss\_confirmation\_line.text())

data.append(self.ui.confirmation\_exchange\_line.text())

data.append(self.ui.destruction\_confirmation\_line.text())

data.append(self.ui.education\_level\_line.text())

data.append(self.ui.series\_doc\_line.text())

data.append(self.ui.number\_doc\_line.text())

data.append(self.ui.issue\_date\_line.text())

data.append(self.ui.registration\_number\_line.text())

data.append(self.ui.code\_profession\_line.text())

data.append(self.ui.title\_profession\_line.text())

data.append(self.ui.title\_qualification\_line.text())

data.append(self.ui.education\_program\_line.text())

data.append(self.ui.admission\_year\_line.text())

data.append(self.ui.graduation\_year\_line.text())

data.append(self.ui.study\_period\_line.text())

data.append(self.ui.last\_name\_line.text())

data.append(self.ui.name\_line.text())

data.append(self.ui.middle\_name\_line.text())

data.append(self.ui.birth\_date\_line.text())

data.append(self.ui.gender\_line.text())

data.append(self.ui.snills\_line.text())

data.append(self.ui.country\_code\_line.text())

data.append(self.ui.education\_form\_line.text())

data.append(self.ui.education\_receipt\_form\_line.text())

data.append(self.ui.financing\_source\_line.text())

return data

def add\_row(self):

"""

Добавление строки в бд. Ввод организуется из боковой панели

"""

if self.mbox\_execute("Вы уверены, что хотите добавить данные?"):

# new\_row\_position = self.ui.table.currentRow() + 1

new\_row\_position = self.ui.table.rowCount()

inputted\_data = self.get\_inputted\_data()

for i in range(len(inputted\_data)):

self.ui.table.setItem(new\_row\_position, i, QtWidgets.QTableWidgetItem(inputted\_data[i].strip()))

try:

result = self.helper.add\_all\_bd(inputted\_data)

if result:

self.ui.table.insertRow(new\_row\_position)

for i in range(len(inputted\_data)):

self.ui.table.setItem(new\_row\_position, i, QtWidgets.QTableWidgetItem(inputted\_data[i].strip()))

except Exception:

self.mbox("Некоректный ввод данных")

def includes\_row(self, row, wanted):

"""

Есть ли в списке row строка wanted.

param row: список

param wanted: строка которая ищется в списке row

return: True, если wanted есть в row. False, если нет

"""

for col in row:

if wanted in str(col):

return True

return False

def find\_rows(self):

"""

Загружает все подходящие строки

"""

text = self.ui.suggest\_input.text().strip()

if text == '':

self.load\_data()

else:

need\_rows = list()

for pk in self.get\_all\_id():

row = self.helper.create\_row(pk)

if self.includes\_row(row, text):

need\_rows.append(row)

len\_need\_rows = len(need\_rows)

self.ui.table.setRowCount(len\_need\_rows)

for i in range(len\_need\_rows):

for j in range(self.ui.table.columnCount()):

self.ui.table.setItem(i, j, QtWidgets.QTableWidgetItem(need\_rows[i][j]))

def remove\_row(self):

"""

Удаление строки из бд и из GUI

"""

if self.mbox\_execute("Вы уверены, что хотите удалить данные?"):

if self.ui.table.rowCount() > 0:

current\_row\_position = self.ui.table.currentRow()

print(current\_row\_position)

row = self.get\_row\_by\_position(current\_row\_position)

result = list()

result.append(self.helper.del\_all\_bd(row))

print(result)

self.ui.table.removeRow(current\_row\_position)

else:

self.mbox("Невозможно удалить, так как таблица пуста")

def create\_doc(self):

"""

Создание документа по шаблону

"""

current\_row\_position = self.ui.table.currentRow()

if current\_row\_position != -1:

row = self.get\_row\_by\_position(current\_row\_position, True)

context = {

'last\_name': row['last\_name'],

'name': row['name'],

'middle\_name': row['middle\_name'],

'birth\_date': row['birth\_date'],

'admission\_year': row['admission\_year'],

'graduation\_year': row['graduation\_year'],

'education\_form': row['education\_form'].lower()[:-2] + "ое",

'title\_doc': row['title\_doc'].lower(),

'registration\_number': row['registration\_number'],

'title\_profession': row['title\_profession'].lower(),

'title\_qualification': row['title\_qualification'].lower(),

}

print(context)

self.word.render(context)

name\_f = context['last\_name'] + '\_' + context['name'] + '\_' + context['middle\_name'] + ".docx"

try:

self.word.save(f"./Документы/{name\_f}")

# print(subprocess.Popen(r'explorer /select,"./Документы"'))

# self.mbox(f"Файл \"{name\_f}\" успешно создан!", "Успех")

except Exception:

self.mbox(f"Файл с именем \"{name\_f}\" открыт. Закройте его и перевыполните действие")

def mbox\_execute(self, body, title='Предупреждение'):

"""

Диалоговое окно с выбором действия [Yes / No]

param body: Главный текст для сообщения

param title: Оглавление диалогового окна

return: True, если выбран 'Yes'. False, если 'No'

"""

dialog = QMessageBox()

dialog.setText(body)

dialog.setWindowTitle(title)

dialog.setStandardButtons(QMessageBox.StandardButton.Yes | QMessageBox.StandardButton.No)

result = dialog.exec()

if result == QMessageBox.StandardButton.Yes:

return True

elif result == QMessageBox.StandardButton.No:

return False

def mbox(self, body, title='Ошибка'):

"""

Диалоговое окно

param body: Главный текст для сообщения

param title: Оглавление диалогового окна

"""

dialog = QMessageBox()

dialog.setText(body)

dialog.setWindowTitle(title)

dialog.exec()

app = QtWidgets.QApplication([])

application = HelperDoc()

application.show()

sys.exit(app.exec())

Приложение 5

Листинг 11. helper.py

from sqlite import Sqliter

class Helper:

def \_\_init\_\_(self, bd):

self.bd = Sqliter(bd)

self.table\_profession = "profession"

self.table\_qualification = "qualification"

self.table\_education\_program = "education\_program"

self.table\_student = "student"

self.table\_document = "document"

@staticmethod

def get\_profession(row, is\_form\_list=False):

"""

return: Вернет список или словарь с необходимыми данными

"""

if is\_form\_list:

return [row[11], row[12]]

return {

'code': row[11],

'title': row[12]

}

def add\_profession\_bd(self, row):

"""

Добавление данных в бд

"""

if row == -1:

return False

data = dict()

if (type(row) is list) or (type(row) is tuple):

data = self.get\_profession(row)

if type(row) is dict:

data = row

is\_added = False

# pk = self.bd.get\_profession\_id(data['title'])

pk = self.bd.get\_id(self.table\_profession, 'code', {'title': data['title']})

if pk == -1:

is\_added = True

self.bd.add\_profession(data['code'], data['title'])

return is\_added

def del\_profession\_bd(self, row):

"""

Удаление данных из бд

"""

is\_del\_success = False

data = self.get\_profession(row)

if data == -1:

return is\_del\_success

# pk = self.bd.get\_profession\_id(data['title'])

pk = self.bd.get\_id(self.table\_profession, 'code', {'title': data['title']})

num\_id = len(self.bd.select(self.table\_student, '`id`', {'profession\_code': pk}))

if num\_id == 0:

self.bd.del\_profession(pk)

is\_del\_success = True

return is\_del\_success

@staticmethod

def get\_qualification(row, is\_form\_list=False):

"""

return: Вернет список или словарь с необходимыми данными

"""

if is\_form\_list:

return [row[13]]

return {

'title': row[13]

}

def add\_qualification\_bd(self, row):

"""

Добавление данных в бд

"""

if row == -1:

return False

data = dict()

if (type(row) is list) or (type(row) is tuple):

data = self.get\_qualification(row)

if type(row) is dict:

data = row

is\_added = False

# data\_id = self.bd.get\_qualification\_id(data['title'])

pk = self.bd.get\_id(self.table\_qualification, 'id', data)

if pk == -1:

is\_added = True

self.bd.add\_qualification(data['title'])

return is\_added

def del\_qualification\_bd(self, row):

"""

Удаление данных из бд

"""

is\_del\_success = False

data = self.get\_qualification(row)

if data == -1:

return is\_del\_success

# pk = self.bd.get\_qualification\_id(\*data)

pk = self.bd.get\_id(self.table\_qualification, 'id', data)

# кол-во используемых id

num\_id = len(self.bd.select(self.table\_student, '`id`', {'qualification\_id': pk}))

if num\_id == 0:

self.bd.del\_qualification(pk)

is\_del\_success = True

return is\_del\_success

@staticmethod

def get\_education\_program(row, is\_form\_list=False):

"""

return: Вернет список или словарь с необходимыми данными

"""

if is\_form\_list:

return [row[14]]

return {

'title': row[14]

}

def add\_education\_program\_bd(self, row):

"""

Добавление данных в бд

"""

if row == -1:

return False

data = dict()

if (type(row) is list) or (type(row) is tuple):

data = self.get\_education\_program(row)

if type(row) is dict:

data = row

is\_added = False

# data\_id = self.bd.get\_education\_program\_id(data['title'])

pk = self.bd.get\_id(self.table\_education\_program, 'id', data)

if pk == -1:

is\_added = True

self.bd.add\_education\_program(data['title'])

return is\_added

def del\_education\_program\_bd(self, row):

"""

Удаление данных из бд

"""

is\_del\_success = False

data = self.get\_education\_program(row)

if data == -1:

return is\_del\_success

# pk = self.bd.get\_education\_program\_id(\*data)

pk = self.bd.get\_id(self.table\_education\_program, 'id', data)

# кол-во используемых id

num\_id = len(self.bd.select(self.table\_student, '`id`', {'education\_program\_id': pk}))

if num\_id == 0:

self.bd.del\_education\_program(pk)

is\_del\_success = True

return is\_del\_success

def get\_student(self, row, is\_form\_list=False):

"""

return: Вернет список или словарь с необходимыми данными

"""

# qualification\_id = self.bd.get\_qualification\_id(row[13])

qualification\_id = self.bd.get\_id(self.table\_qualification, 'id', {'title': row[13]})

# education\_program\_id = self.bd.get\_education\_program\_id(row[14])

education\_program\_id = self.bd.get\_id(self.table\_education\_program, 'id', {'title': row[14]})

if (qualification\_id == -1) or (education\_program\_id == -1):

return -1

try:

birth\_date = row[21].strftime("%d.%m.%Y")

except Exception:

birth\_date = row[21]

if is\_form\_list:

return [

row[19], row[18], row[20], birth\_date,

row[22], row[23], row[24], row[25], row[26], row[27], row[15],

row[16], row[17], row[11], qualification\_id, education\_program\_id,

]

return {

'name': row[19],

'last\_name': row[18],

'middle\_name': row[20],

'birth\_date': birth\_date,

'gender': row[22],

'snills': row[23],

'country\_code': row[24],

'education\_form': row[25],

'education\_receipt\_form': row[26],

'financing\_source': row[27],

'admission\_year': row[15],

'graduation\_year': row[16],

'study\_period': row[17],

'profession\_code': row[11],

'qualification\_id': qualification\_id,

'education\_program\_id': education\_program\_id,

}

def add\_student\_bd(self, row):

"""

Добавление данных в бд

"""

if row == -1:

return False

data = dict()

if (type(row) is list) or (type(row) is tuple):

data = self.get\_student(row)

if type(row) is dict:

data = row

is\_added = False

data\_of\_list = [

data['name'],

data['last\_name'],

data['middle\_name'],

data['birth\_date'],

data['gender'],

data['snills'],

data['country\_code'],

data['education\_form'],

data['education\_receipt\_form'],

data['financing\_source'],

data['admission\_year'],

data['graduation\_year'],

data['study\_period'],

data['profession\_code'],

data['qualification\_id'],

data['education\_program\_id'],

]

# data\_id = self.bd.get\_student\_id(\*data\_of\_list)

pk = self.bd.get\_id(self.table\_student, 'id', data)

if pk == -1:

is\_added = True

self.bd.add\_student(\*data\_of\_list)

return is\_added

def del\_student\_bd(self, row):

"""

Удаление данных из бд

"""

data = self.get\_student(row)

if data == -1:

return False

# pk = self.bd.get\_student\_id(\*data)

pk = self.bd.get\_id(self.table\_student, 'id', data)

self.bd.del\_student(pk)

return True

def get\_document(self, row, is\_form\_list=False):

"""

return: Вернет список или словарь с необходимыми данными

"""

student = self.get\_student(row)

if student == -1:

return -1

# student\_id = self.bd.get\_student\_id(\*student)

student\_id = self.bd.get\_id(self.table\_student, 'id', student)

if student\_id == -1:

return -1

try:

issue\_date = row[9].strftime("%d.%m.%Y")

except Exception:

issue\_date = row[9]

if is\_form\_list:

return [

row[0], row[1], row[7], row[8], row[3],

row[4], row[5], row[6], issue\_date, row[10],

row[2], student\_id,

]

return {

'title': row[0],

'type': row[1],

'series': row[7],

'number': row[8],

'loss\_confirmation': row[3],

'exchange\_confirmation': row[4],

'destruction\_confirmation': row[5],

'education\_level': row[6],

'issue\_date': issue\_date,

'registration\_number': row[10],

'status': row[2],

'student\_id': student\_id,

}

def add\_document\_bd(self, row):

"""

Добавление данных в бд

"""

if row == -1:

return False

data = dict()

if (type(row) is list) or (type(row) is tuple):

data = self.get\_document(row)

if type(row) is dict:

data = row

is\_added = False

data\_of\_list = [

data['title'],

data['type'],

data['series'],

data['number'],

data['loss\_confirmation'],

data['exchange\_confirmation'],

data['destruction\_confirmation'],

data['education\_level'],

data['issue\_date'],

data['registration\_number'],

data['status'],

data['student\_id']

]

# data\_id = self.bd.get\_document\_id(\*data\_of\_list)

pk = self.bd.get\_id(self.table\_document, 'id', data)

if pk == -1:

is\_added = True

self.bd.add\_document(\*data\_of\_list)

return is\_added

def del\_document\_bd(self, row):

"""

Удаление данных из бд

"""

data = self.get\_document(row)

if data == -1:

return False

# pk = self.bd.get\_document\_id(\*data)

pk = self.bd.get\_id(self.table\_document, 'id', data)

self.bd.del\_document(pk)

return True

def add\_all\_bd(self, row):

"""

Добавление всех данных в бд

"""

result\_add = {

'profession': self.add\_profession\_bd(self.get\_profession(row)),

'qualification': self.add\_qualification\_bd(self.get\_qualification(row)),

'education\_program': self.add\_education\_program\_bd(self.get\_education\_program(row)),

'student': self.add\_student\_bd(self.get\_student(row)),

'document': self.add\_document\_bd(self.get\_document(row))

}

# print(result\_add)

return bool(sum(result\_add.values()))

def del\_all\_bd(self, row):

"""

Удаление полной строки из бд

"""

result\_del = {

'document': self.del\_document\_bd(row),

'student': self.del\_student\_bd(row),

'education\_program': self.del\_education\_program\_bd(row),

'qualification': self.del\_qualification\_bd(row),

'profession': self.del\_profession\_bd(row),

}

return result\_del['document']

def create\_row(self, pk\_document):

"""

Создание строки для удобного форматирования

"""

try:

document = self.bd.select(self.table\_document, '\*', {'id': pk\_document})[0]

student = self.bd.select(self.table\_student, '\*', {'id': document[12]})[0]

profession = self.bd.select(self.table\_profession, '\*', {'code': student[14]})[0]

education\_program = self.bd.select(self.table\_education\_program, '\*', {'id': student[15]})[0]

qualification = self.bd.select(self.table\_qualification, '\*', {'id': student[16]})[0]

data = (

document[1], document[2], document[11],

document[5], document[6], document[7],

document[8], document[3], document[4],

document[9], document[10], profession[0],

profession[1], qualification[1],

education\_program[1], student[11], student[12],

student[13], student[2], student[1],

student[3], student[4], student[5],

student[6], student[7], student[8],

student[9], student[10]

)

# print(data)

return data

except Exception:

return -1

def close(self):

"""

Закрытие соединения с бд

"""

self.bd.close()

Приложение 6

Листинг 12. sqlite.py

import sqlite3

class Sqliter:

def \_\_init\_\_(self, name\_db):

"""

Подключение к бд

:param name\_db: имя базы данных

"""

self.connection = sqlite3.connect(name\_db)

self.cursor = self.connection.cursor()

self.table = str()

self.table\_profession = "profession"

self.table\_qualification = "qualification"

self.table\_education\_program = "education\_program"

self.table\_student = "student"

self.table\_document = "document"

@staticmethod

def is\_valid(table):

"""

Валидна ли таблица

:param table: название таблицы

:return: True - если название таблицы подходит; False - если нет

"""

is\_valid = False

if len(table) != 0 or type(table) == str:

new\_table = "".join(table.split('\_'))

is\_valid = new\_table.isalnum()

return is\_valid

def connect\_table(self, table):

"""

Функция create\_table создаёт таблицу в БД

:param table: имя таблицы

:return: True - если подключились к таблице, False - если нет

"""

if table is None:

table = self.table

with self.connection:

is\_valid = self.is\_valid(table)

if is\_valid:

self.table = table

return is\_valid

def get\_qualification\_id(self, title):

"""

Вернет id записи от таблицы qualification.

:return:

"""

try:

with self.connection:

data = self.cursor.execute(

f'SELECT `id` FROM {self.table\_qualification} WHERE `title` = \'{title}\'').fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_qualification(self, title: str):

"""

Функция add добавляет данные в таблицу qualification

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_qualification}` (title) VALUES (?)", (title,))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_qualification: Такой пользователь уже существует")

return is\_valid

def del\_qualification(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_qualification} WHERE `id` = {pk}')

self.save()

def get\_education\_program\_id(self, title):

"""

Вернет id записи от таблицы student.

:return:

"""

try:

with self.connection:

data = self.cursor.execute(

f'SELECT `id` FROM {self.table\_education\_program} WHERE `title` = \'{title}\'').fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_education\_program(self, title: str):

"""

Функция add добавляет данные в таблицу education\_program

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_education\_program}` (title) VALUES (?)", (title,))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_education\_program: Такой пользователь уже существует")

return is\_valid

def del\_education\_program(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_education\_program} WHERE `id` = {pk}')

self.save()

def get\_profession\_id(self, title):

"""

Вернет id записи из таблицы profession

:return:

"""

try:

with self.connection:

data = self.cursor.execute(

f'SELECT `code` FROM {self.table\_profession} WHERE `title` = \'{title}\'').fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_profession(self, code: str, title: str):

"""

Функция add добавляет данные в таблицу profession

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_profession}` (code, title) VALUES (?,?)", (code, title))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_profession: Такой пользователь уже существует")

return is\_valid

def del\_profession(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_profession} WHERE `code` = \'{pk}\'')

self.save()

def get\_student\_id(self, name, last\_name, middle\_name, birth\_date, gender,

snills, country\_code, education\_form, education\_receipt\_form,

financing\_source, admission\_year, graduation\_year, study\_period,

profession\_code, qualification\_id, education\_program\_id

):

"""

Вернет id студента от таблицы student

:return:

"""

try:

with self.connection:

command = f'SELECT `id` FROM {self.table\_student} WHERE `name` = ? AND \

`last\_name` = ? AND `middle\_name` = ? AND \

`birth\_date` = ? AND `gender` = ? AND \

`snills` = ? AND `country\_code` = ? AND \

`education\_form` = ? AND \

`education\_receipt\_form` = ? AND \

`financing\_source` = ? AND \

`admission\_year` = ? AND \

`graduation\_year` = ? AND \

`study\_period` = ? AND \

`profession\_code` = ? AND \

`qualification\_id` = ? AND \

`education\_program\_id` = ?'

args = (

name, last\_name, middle\_name, birth\_date, gender,

snills, country\_code, education\_form, education\_receipt\_form,

financing\_source, admission\_year, graduation\_year, study\_period,

profession\_code, qualification\_id, education\_program\_id

)

data = self.cursor.execute(command, args).fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_student(self, name, last\_name, middle\_name, birth\_date, gender,

snills, country\_code, education\_form, education\_receipt\_form,

financing\_source, admission\_year, graduation\_year, study\_period,

profession\_code, qualification\_id, education\_program\_id

):

"""

Функция add добавляет данные в таблицу student

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_student}` (name, last\_name, middle\_name, birth\_date, \

gender, snills, country\_code, education\_form, education\_receipt\_form, financing\_source, \

admission\_year, graduation\_year, study\_period, profession\_code, qualification\_id, \

education\_program\_id) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)",

(name, last\_name, middle\_name, birth\_date, gender,

snills, country\_code, education\_form, education\_receipt\_form,

financing\_source, admission\_year, graduation\_year, study\_period,

profession\_code, qualification\_id, education\_program\_id))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_student: Такой пользователь уже существует")

return is\_valid

def del\_student(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_student} WHERE `id` = {pk}')

self.save()

def get\_document\_id(self, title, type, series, number, loss\_confirmation, exchange\_confirmation,

destruction\_confirmation, education\_level, issue\_date, registration\_number,

status, student\_id):

"""

Вернет id студента от таблицы student

:return:

"""

try:

with self.connection:

data = self.cursor.execute(

f'SELECT `id` FROM {self.table\_document} WHERE `title` = \'{title}\' AND \

`type` = \'{type}\' AND `series` = \'{series}\' AND \

`number` = \'{number}\' AND `loss\_confirmation` = \'{loss\_confirmation}\' AND \

`exchange\_confirmation` = \'{exchange\_confirmation}\' AND \

`destruction\_confirmation` = \'{destruction\_confirmation}\' AND \

`education\_level` = \'{education\_level}\' AND `issue\_date` = \'{issue\_date}\' AND \

`registration\_number` = \'{registration\_number}\' AND `status` = \'{status}\' AND \

`student\_id` = \'{student\_id}\'').fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_document(self, title, type, series, number, loss\_confirmation, exchange\_confirmation,

destruction\_confirmation, education\_level, issue\_date, registration\_number,

status, student\_id):

"""

Функция add добавляет данные в таблицу profession

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_document}` (title, type, series, number, \

loss\_confirmation, exchange\_confirmation, destruction\_confirmation, education\_level, issue\_date, \

registration\_number,status, student\_id) VALUES (?,?,?,?,?,?,?,?,?,?,?,?)", (title, type, series, number,

loss\_confirmation,

exchange\_confirmation,

destruction\_confirmation,

education\_level,

issue\_date,

registration\_number,

status, student\_id))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_document: Такой пользователь уже существует")

return is\_valid

def del\_document(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_document} WHERE `id` = {pk}')

self.save()

def get\_id(self, table: str, name\_pk: str, data: dict):

"""

Вернет id студента от таблицы student

:return:

"""

try:

with self.connection:

command = f"SELECT `{name\_pk}` FROM `{table}`"

count = len(data)

if count == 0:

return -1

command += " WHERE "

column\_values = list()

for key, value in data.items():

count -= 1

if value is not None:

command += f"`{key}` = ?"

column\_values.append(value)

if count:

command += " AND "

need\_id = self.cursor.execute(command, tuple(column\_values)).fetchall()

if len(need\_id) != 0:

return need\_id[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def get\_last\_id(self, table, name\_pk):

"""

Вернет последнее id записи из таблицы которую вы передадите

:return:

"""

try:

with self.connection:

return self.cursor.execute(f"SELECT MAX({name\_pk}) FROM {table}").fetchall()[0][0]

except sqlite3.IntegrityError:

return -1

def select(self, table: str, need: str, data: dict):

"""

Вернет выбранные данные (need) из выбранной таблицы (table).

Выборка происходит благодаря переданным данным (data)

:return:

"""

try:

with self.connection:

command = f"SELECT {need} FROM `{table}`"

if len(data) == 0:

return self.cursor.execute(command).fetchall()

command += " WHERE "

column\_values = list()

count = len(data)

for key, value in data.items():

command += f"`{key}` = ?"

column\_values.append(value)

count -= 1

if count:

command += " AND "

return self.cursor.execute(command, tuple(column\_values)).fetchall()

except sqlite3.IntegrityError:

return -1

# def get

def update(self, table: str, pk: dict, data: dict):

"""

Обновит выбранные данные (data) из выбранной таблицы (table).

Выборка происходит благодаря id таблицы (pk)

:param: все словари имеют следующий вид => {'имя\_колоны': значение\_колоны}

:return:

"""

try:

with self.connection:

column\_values = list()

command = f"UPDATE `{table}`"

key\_pk = list(pk.keys())[0]

command += f" SET `{key\_pk}` WHERE "

column\_values.append(pk[key\_pk])

count = len(data)

for key, value in data.items():

column\_values.append(value)

command += f"`{key}` = ?"

count -= 1

if count:

command += ", "

return self.cursor.execute(command, tuple(column\_values)).fetchall()

except sqlite3.IntegrityError:

return -1

# Функция save сохраняет изменения в БД

def save(self):

self.connection.commit()

# print(f"{self.cursor.rowcount} отредактированно строк")

# Функция close закрывает БД

def close(self):

self.connection.close()