

Министерство науки и высшего образования  
Российской Федерации

Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Кафедра прикладной математики

Master Thesis  
**Developing informational system for collecting  
and analyzing student's digital footprint**



Факультет: ПМИ  
Группа: ПММ-03  
Студент: Moskalev D.I.

Новосибирск

2022

## **АННОТАЦИЯ**

Отчёт 125 с., 5 ч., 84 рис., 11 табл., 21 источник, 5 прил.

DATA ANALYSIS, DATA MINING, BIG DATA, MACHINE LEARNING,  
DEEP LEARNING, ETL, ЦИФРОВОЙ СЛЕД СТУДЕНТА.

Объектом исследования является студенческий журнал, состоящий из таблиц и рукописного текста, с данными о посещаемости и успеваемости студентов по предметам из расписания университета.

Цель работы – разработка информационной системы для своевременного определения факторов влияющих на посещения предметов студентами, используя ETL процессы в рамках мобильного приложения и визуализацией данных в режиме реального времени.

В процессе работы были изучены эффективные методы анализа данных, выполнена обработка рукописного текста, сгенерирована собственная OCR модель для распознавания рукописных символов, сделаны два мобильных приложения, веб-сервис и визуальная аналитическая панель (дашборд) для визуализации данных.

Результатом работы является реализация мобильных приложений и визуального представления выходных данных используя веб-сервис и дашборд с возможностью фильтрации результатов.

# **СОДЕРЖАНИЕ**

ВВЕДЕНИЕ .....	5
1. ЦИФРОВОЙ СЛЕД СТУДЕНТА .....	7
1.1. ОСНОВНЫЕ ПРОБЛЕМЫ УЧЕТА ПОСЕЩАЕМОСТИ СТУДЕНТОВ ...	7
1.2. ИСПОЛЬЗУЕМЫЕ ТЕРМИНЫ И ОБОЗНАЧЕНИЯ .....	8
2. ЗАДАЧА КЛАССИФИКАЦИИ СИМВОЛОВ .....	10
2.1. ПОСТАНОВКА ЗАДАЧИ.....	10
2.2. ОБЗОР ИСТОЧНИКОВ ИНФОРМАЦИИ .....	13
2.3. ВЫБРАННЫЕ МЕТОДЫ РЕШЕНИЯ .....	13
3. ОПИСАНИЕ РАЗРАБОТАННЫХ ПРОГРАММ .....	18
3.1. МОБИЛЬНОЕ ПРИЛОЖЕНИЕ SYMBOL CLASSIFIER .....	18
3.2. МОБИЛЬНОЕ ПРИЛОЖЕНИЕ STUDENT DIGITIZER .....	21
3.3. МОДУЛЬ БАЗЫ ДАННЫХ .....	25
3.4. МОДУЛЬ ВИЗУАЛИЗАЦИИ ДАННЫХ.....	28
4. ЭТАПЫ РЕШЕНИЯ ЗАДАЧИ .....	30
4.1. АНАЛИЗ ВХОДНЫХ ДАННЫХ .....	30
4.2. РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ.....	40
4.3. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ .....	43
4.4. СОЗДАНИЕ ДАШБОРДА .....	45
4.5. СОЗДАНИЕ ВЕБ-СЕРВИСА .....	45
5. ИССЛЕДОВАНИЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ .....	46
5.1. МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ .....	46
5.2. ВИЗУАЛИЗАЦИЯ ПОЛУЧЕННЫХ ДАННЫХ .....	75
ЗАКЛЮЧЕНИЕ .....	85
СПИСОК ЛИТЕРАТУРЫ .....	86
ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ ПРЕДОБРАБОТКИ ДАННЫХ.....	88
ПРИЛОЖЕНИЕ Б. ТЕКСТ ПРОГРАММЫ КЛАССИФИКАЦИИ ДАННЫХ ..	94
ПРИЛОЖЕНИЕ В. ТЕКСТ ПРИЛОЖЕНИЯ SYMBOL CLASSIFIER .....	95
ПРИЛОЖЕНИЕ Г. ТЕКСТ ПРИЛОЖЕНИЯ STUDENT DIGITIZER .....	99
ПРИЛОЖЕНИЕ Д. ТЕКСТ ВЕБ-СЕРВИСА ВИЗУАЛИЗАЦИИ ДАННЫХ....	124

## **ВВЕДЕНИЕ**

В настоящее время актуальность анализа данных успеваемости студентов с целью возможности редактирования учебного процесса обуславливается тем, что большинство учебных организаций получают данные только к концу семестра, когда они уже не всегда актуальны. В данном проекте реализован способ получения данных в течение недели после прошедшего занятия.

Актуальность данной тематики обуславливается отсутствием современных бесплатных систем и программ, которые считывают в форматированном виде рукописный или печатный текст с помощью фотокамеры мобильного телефона в виде электронных таблиц.

Постепенно, объем данных для анализа данных стал настолько большим, что его невозможно было обработать вручную и появилась необходимость в автоматизации процесса мониторинга успеваемости обучающихся, поэтому сейчас активно развиваются сферы по анализу данных (Data Analysis) и машинного обучения (Machine Learning), позволяющие обрабатывать большие данные (Big Data) эффективно и точно. Актуальность и новизну теме добавляет и тот факт, что после решения задачи не требуется хранить первоначальные файлы или какие-либо промежуточные данные как в бумажном, так и в цифровом виде.

Используемые в работе средства анализа данных и машинного обучения позволяют преобразовать и визуализировать исходные данные, при этом существенно сэкономив ресурсы, а сделанные мобильные приложения готовы для дальнейшего использования в учебном процессе. Каждый год появляются новые статьи и методы для улучшения точности в разных отраслях анализа данных и оптического распознавания текста (OCR), что показывает интерес к задаче.

В качестве сферы анализа данных выбрана учебная отрасль, так как она является одной из самых важных и крупных, а полученное решение позволяет использовать приложение для любого письменного языка лишь с незначительными изменениями, которые зависят от формата входных данных. Поддерживаются как

печатный, так и отсканированный вариант студенческого журнала. Данные заполнения студенческого журнала вымышленные и любое совпадение случайно.

Своевременное определение возможного снижения числа студентов позволяют скорректировать учебный план или провести собрание, чтобы определить причины непосещения конкретных предметов студентами, принять меры для сохранения количества обучающихся в текущем семестре и персонализировать расписание или преподавательский состав для данной студенческой группы различными доступными способами в дальнейшем, чтобы улучшить заинтересованность и показатели вовлеченности в учебный процесс.

В рамках выпускной квалификационной работы было необходимо разработать информационную систему сбора и анализа цифрового следа студентов. Для этого требовалось:

1. Определить основные технологии для решения задачи.
2. Найти общедоступные наборы данных рукописных символов.
3. Произвести подготовку данных для дальнейшего обучения модели.
4. Создать программную реализацию классификации символов.
5. Построить ETL систему извлечения, перемещения и загрузки данных.
6. Создать мобильные приложения с графическим интерфейсом.
7. Сделать визуальные представления полученных результатов.

В первом разделе магистерской диссертации описывается понятие цифрового следа студента и термины, используемые в процессе написания работы.

Во втором разделе магистерской диссертации рассмотрена задача классификации, методов решения и источников информации.

В третьем разделе магистерской диссертации рассмотрены выбранные языки программирования, технологии и подробное описание модулей программ.

В четвертом разделе магистерской диссертации проведен анализ данных и представлен поэтапный процесс проектирования и решения задачи.

В пятом разделе магистерской диссертации отображены полученные результаты мобильных приложений и визуализации данных.

# **1. ЦИФРОВОЙ СЛЕД СТУДЕНТА**

Цифровой след студента – полученные обезличенные цифровые данные студента из студенческого журнала, состоящие из номера и количества пропусков определенных учебных занятий. Главным достоинством сбора и анализа цифрового следа студентов является гибкость и универсальность. Любая учебная организация на основании данных за предыдущий отчетный период времени может получить достаточно точное представление о текущем состоянии учебного процесса и принять меры исходя из анализа определенного сегмента учащихся.

## **1.1. ОСНОВНЫЕ ПРОБЛЕМЫ УЧЕТА ПОСЕЩАЕМОСТИ СТУДЕНТОВ**

Практическое применение сбора и анализа цифрового следа студентов позволяет ответить на множество интересующих вопросов.

### **1.1.1. Причины пропусков и отчислений студентов**

Среди важных причин, по которым студенты университетов пропускают занятия, отчисляются либо переходят в другие университеты (факультеты), наиболее распространены:

- Сложность учебной программы или отдельных предметов;
- Организация лекций или практик со стороны преподавателей;
- Более привлекательные предложения университетов-конкурентов;
- Недовольство предметом, факультетом или университетом;
- Аналогичная специальность с более привлекательными условиями;
- Персональные факторы студентов.

Если университет (факультет) не готовы предоставить условия лучше, чем в другом университете (факультете), то студент начинает поиск и с некоторой вероятностью меняет университет или факультет.

### **1.1.2. Эффективность мер по поддержанию количества студентов**

Для понимания эффективности принятых мер необходимо взять показатели за новый период, совпадающий по длительности, сезонности, признакам и сегменту студентов.

Поскольку анализ данных можно сделать для любой отрасли, в которой ведется база данных, то чтобы оценить эффективность мер по поддержанию количества студентов, можно построить визуальное представление полученных данных для соответствующей учебной организации.

## **1.2. ИСПОЛЬЗУЕМЫЕ ТЕРМИНЫ И ОБОЗНАЧЕНИЯ**

**Анализ данных (Data Analysis)** – инструмент обработки данных на подготовительном и завершающем этапах решения задачи.

**БД (Database)** – база данных.

**Большие данные (Big Data)** – серия подходов, инструментов и методов для работы с данными больших объемов.

**Глубокое обучение (Deep Learning)** – метод машинного обучения, который предполагает самостоятельное выстраивание общих правил в виде искусственной нейронной сети на примере данных во время процесса обучения.

**Датасет (Dataset)** – набор данных.

**Дашборд (Dashboard) [1]** – информационная панель или визуальная модель сгруппированных данных, которая на одном экране представляет все данные таких образом, что необходимые аналитические процедуры и выводы легко вытекают из этой модели.

**Классификация (Classification)** – совокупность методов машинного обучения, в которых предсказывается принадлежность целевого признака к определенному классу.

**Машинное обучение (Machine Learning)** – подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных учиться.

**Метрика (Metric)** – числовое или визуальное представление результатов для оценки качества модели и сравнения алгоритмов.

**Признак (Feature)** – входной параметр, который может иметь числовой или категориальный тип.

**Свёртка (Convolution)** – процесс применения фильтра к изображению.

**Свёрточные нейронные сети (Convolutional Neural Network)** – алгоритм, который может принимать входное изображение, присваивать важность (веса и смещения) различным объектам в изображении.

**СУБД (Database Management System)** – комплекс программно-языковых средств, позволяющих создавать базы данных и управлять данными.

**Тензор (Tensor)** – обобщение векторов и матриц.

**Точность (Accuracy)** – метрика точности решения модели классификатора на тестовой выборке.

**Функция активации (Activation function)** – функция, которая определяет выходное значение нейрона в зависимости от результата взвешенной суммы входов и порогового значения.

**Целевой признак (Target feature)** – признак, который определяется в процессе решения задачи.

**Шум (Noise)** – отсутствие, неверный формат или не уникальные данные.

**ETL (Extract, Transform, Load)** – конвейер процессов по сбору, перемещению и загрузке данных.

**OCR (Optical character recognition)** – перевод рукописного или печатного текста в текстовые данные, использующиеся для представления символов в компьютере.

## 2. ЗАДАЧА КЛАССИФИКАЦИИ СИМВОЛОВ

В этой главе рассмотрено понятие задачи многоклассовой классификации, выполнен обзор источников информации и методов решения задачи.

### 2.1. ПОСТАНОВКА ЗАДАЧИ

Обобщенная постановка задачи мульти-классификации представляет собой неизвестную целевую зависимость:

$$y^* : X \rightarrow Y, \quad (2.1)$$

где  $X$  – множество описаний входных объектов,  $Y$  – конечное известное множество классов. Значения отображения (2.1) известны на всех объектах конечной обучающей выборки набора данных вида:

$$X_n = [(x_1, y_1), \dots, (x_n, y_n)], \quad (2.2)$$

где  $n$  – количество объектов. Вероятностная постановка задачи предполагает пространство, состоящее из множества пар с неизвестной вероятностной мерой  $P: X \times Y$ , где  $X$  – конечное известное множество описаний объектов,  $Y$  – конечное известное множество классов.

При этом имеется обучающая выборка зависимостей (2.2), сгенерированная согласно вероятностной мере  $P$ .

В обеих постановках задачи необходимо построить алгоритм  $a: X \rightarrow Y$ , который может однозначно определить класс произвольного объекта  $x \in X$  из набора данных. Таким образом, задача решается методами классификации, которые по признакам и опираясь на обучающую выборку, с определенной вероятностью определяют принадлежность к одному из классов.

Признаком называется отображение  $f: X \rightarrow D_f$ , где  $D_f$  – множество допустимых значений признака. Если заданы признаки  $features = \{f_1, \dots, f_n\}$ , то вектор  $x = (f_1(x), \dots, f_n(x))$  называется признаковым описанием объекта  $x \in X$ , а множество  $X = D_{f_1} \times \dots \times D_{f_n}$  называют признаковым пространством [2].

В задаче мульти-классификации допустимое множество значений признаков  $D_f$  имеет многоклассовый признак:  $D_f = \{f_1, f_2, \dots, f_n\}$ . Нумерация классов:  $f_1 = 0, f_2 = 1, f_n = n - 1$ .

Основные этапы решения задачи для мобильного приложения “*Symbol Classifier*” можно разделить на обучение модели и обработку результатов (рисунок 1).

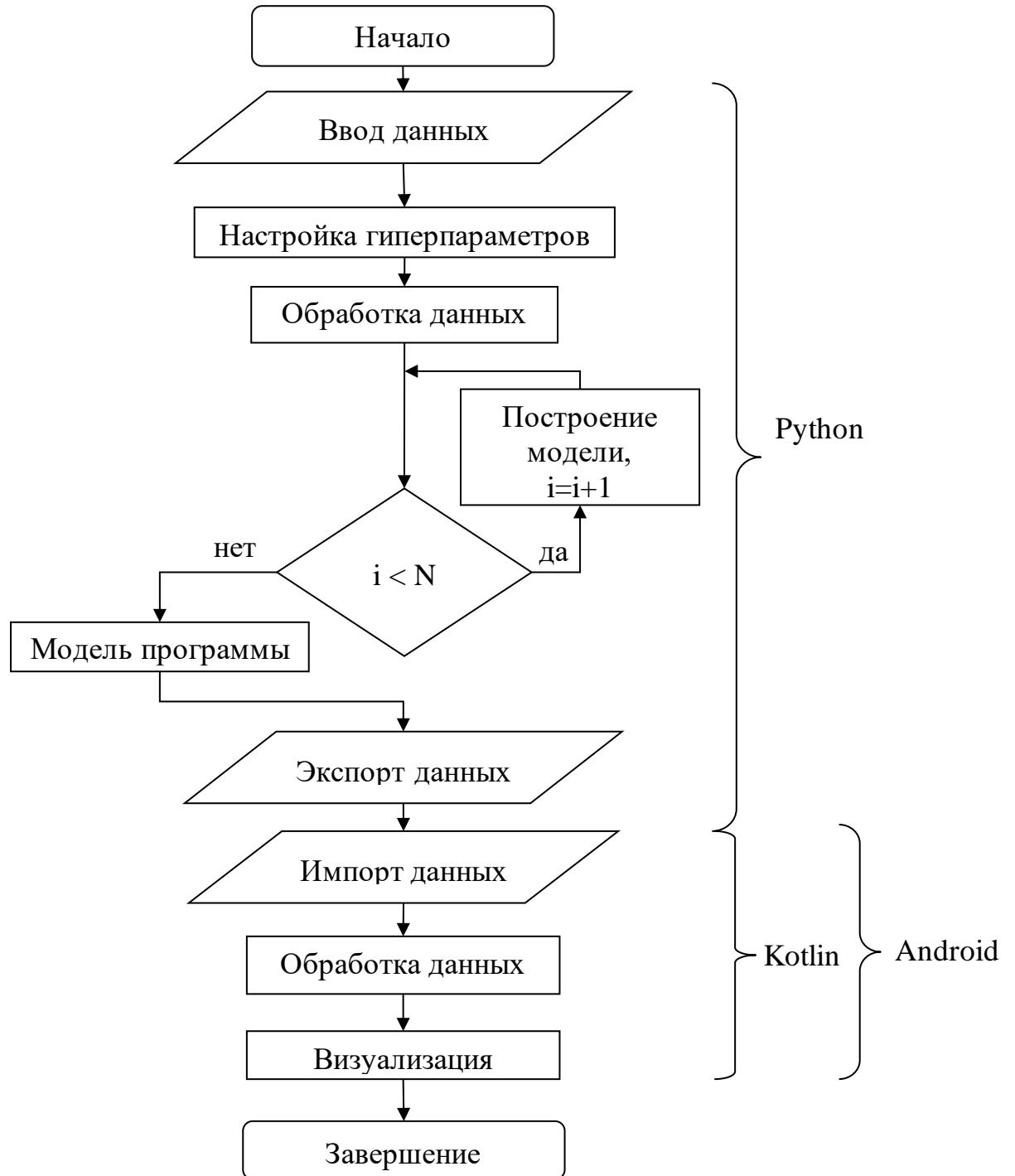


Рисунок 1 – Алгоритм решения задачи для приложения “*Symbol Classifier*”

Таким образом полученная модель на языке программирования Python [3] является входными данными для обработки символов на языке Kotlin [4].

Основные этапы решения задачи для мобильного приложения “*Student Digitizer*” можно разделить на извлечение, обработку и визуализацию данных (рисунок 2).

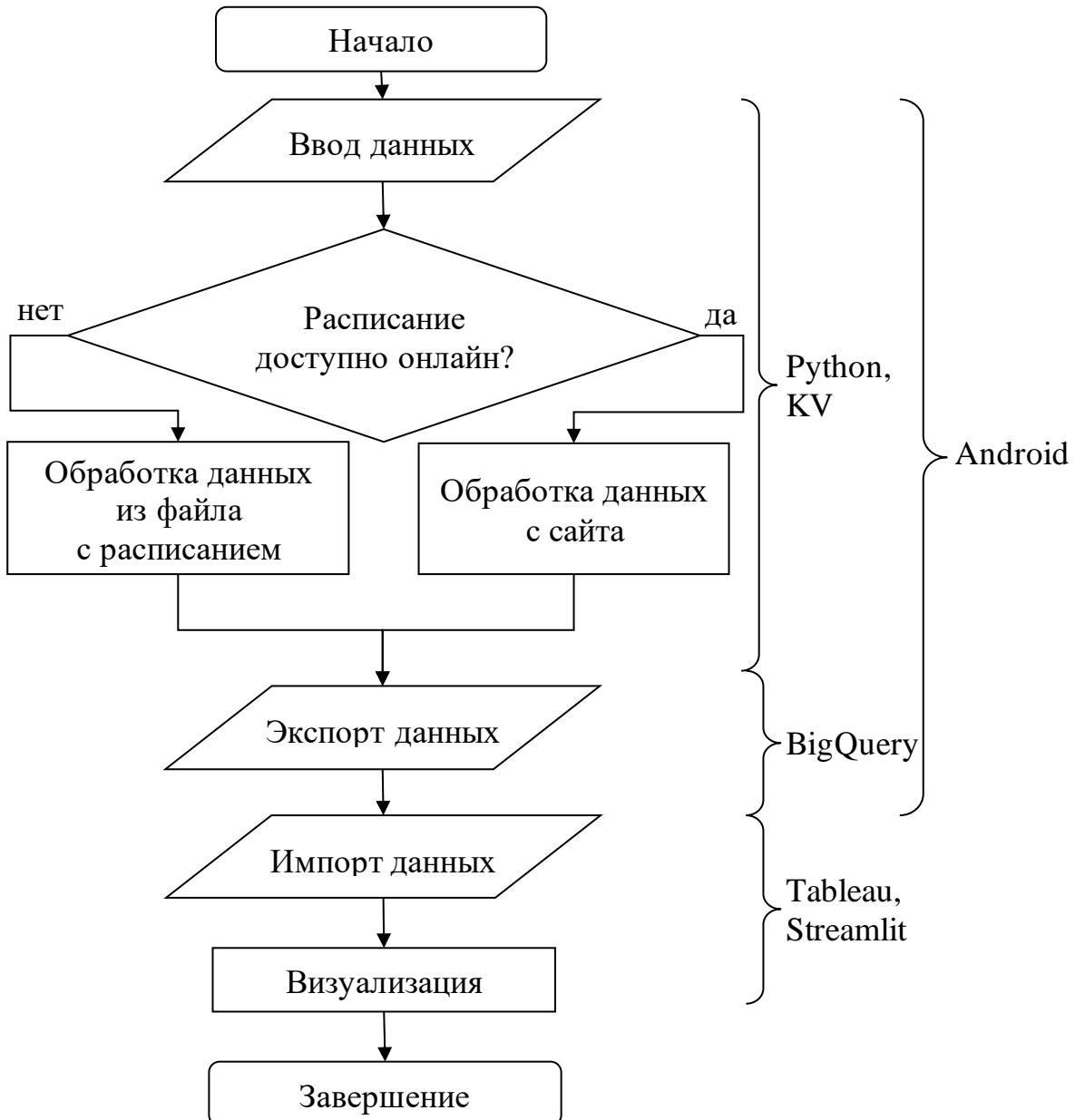


Рисунок 2 – Алгоритм решения задачи для приложения “*Student Digitizer*”

Таким образом обработанные данные на языке программирования Python записываются базу данных BigQuery [5] с помощью SQL запросов и визуализируется в Tableau [6] в виде дашборда и на веб-сервисе с помощью фреймворка Streamlit.

## **2.2. ОБЗОР ИСТОЧНИКОВ ИНФОРМАЦИИ**

Разработка информационной системы для анализа цифрового следа студентов является одной из инновационных и актуальных задач, где применяются техники анализа данных и машинного обучения, поэтому для ее решения был взят набор данных (датасет) букв алфавита в общедоступном открытом ресурсе для создания собственной модели решения и дальнейшего применения на основе обученных данных. Классифицированные символы важно учитывать в студенческом журнале для правильной обработки данных.

В качестве набора символьных данных был выбран общедоступный датасет “*База рукописных символов русского алфавита*” [7]. Основным преимуществом выбранного датасета можно назвать большой объем информации и разделение регистра символов.

## **2.3. ВЫБРАННЫЕ МЕТОДЫ РЕШЕНИЯ**

Для решения задачи многоклассовой классификации были выбраны свёрточные нейронные сети, позволяющие классифицировать размеченные графические данные с высокой точностью за счет большого числа слоев и параметров архитектуры модели. Рассмотрим основные параметры каждого слоя нейронной сети (таблица 1). На рисунке 3 изображена архитектура свёрточной многослойной нейронной сети.

Таблица 1 – Основные параметры слоев свёрточной нейронной сети

Слой №	Название	Параметры	Функция активации
1	Conv2D	160	relu
2	Conv2D	16512	relu
3	MaxPooling 2D	0	
4	Conv2D	590336	relu
5	Flatten	0	
6	Dense	393219	softmax

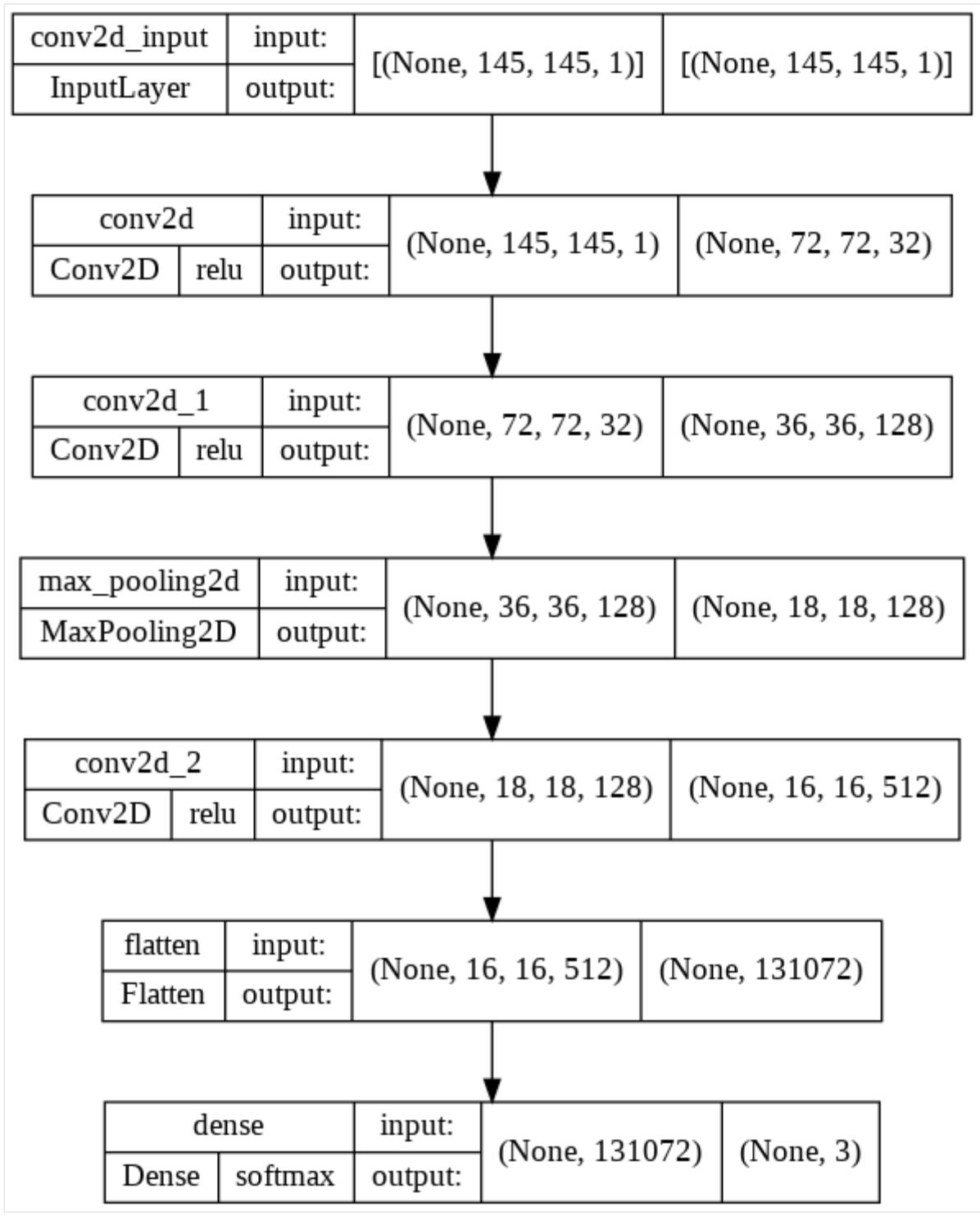


Рисунок 3 – Архитектура свёрточной нейронной сети для решения задачи

### 2.3.1. Описание слоев

Слой “Conv2D” – определяет взвешенное ядро. Производится свёртка, результатом которой становятся тензоры.

Слой “MaxPooling 2D” – используется для уменьшения размера ввода и извлечения максимума из подвыборки.

Слой “Flatten” – используется для конвертации данных в меньшую размерность.

Слой “Dense” – вычисляет  $output = activation(dot(input, kernel) + bias)$ , где  $activation$  – это активатор,  $kernel$  – взвешенная матрица, применяемая к входящим тензорам,  $bias$  – константа, для настройки модели.

Слои “MaxPooling 2D” и “Flatten” преобразуют данные без дополнительных вычислений, поэтому не требуют функций активации.

### 2.3.2. Описание функции активации Relu

Relu (Rectified Linear Unit) [8] – скалярная нелинейная функция активации (2.3), которая возвращает либо само положительное число, либо 0 (рисунок 4).

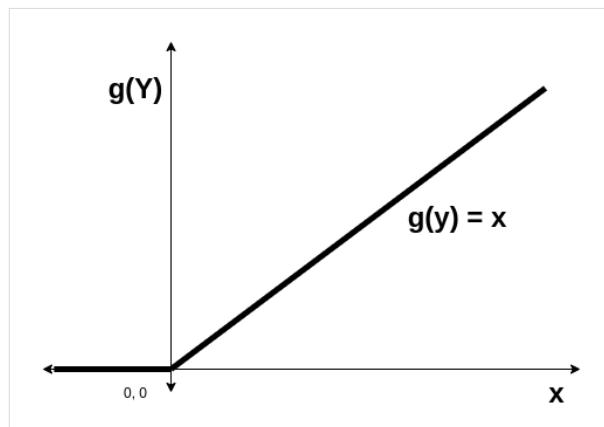


Рисунок 4 – Функция активации Relu

$$f(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} \quad (2.3)$$

Рассмотрим основные преимущества и недостатки активационной функции Relu (таблица 2).

Таблица 2 – Особенности активационной функции Relu

Преимущества	Недостатки
Нелинейная функция	Для чисел меньше 0 нельзя выполнять обратное распространение ошибки,
Вычислительно-эффективная функция	так как функция возвращает всегда 0

### 2.3.3. Описание функции активации Softmax

Softmax – векторная функция активации, которая масштабирует числа в вероятности и возвращает вектор с вероятностями каждого возможного результата [9]. Для нейронных сетей, которым необходимо классифицировать входные данные по многочисленным категориям, Softmax часто используется исключительно для выходного слоя и рассчитывается по формуле 2.4.

$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^n \exp(y_j)}, \quad (2.4)$$

где  $y$  – входной вектор Softmax функции  $S$  размерностью  $n \times n$  элементов,  $y_i$  – элемент входного вектора любого значения,  $\exp(y_i)$  – стандартная функция экспоненты для  $y_i$  элемента,  $\sum_{j=1}^n \exp(y_j)$  – нормализация значений от 0 до 1,  $n$  – количество классов.

Рассмотрим основные преимущества и недостатки активационной функции Softmax (таблица 3).

Таблица 3 – Особенности активационной функции Softmax

Преимущества	Недостатки
Может использоваться для многоклассовой классификации	
Функция нормализует выходные данные для каждого класса от 0 до 1 и делит их на сумму значений, показывая вероятность того, что входное значение находится в определенном классе.	

Рассмотрим результаты обучения модели данных (таблица 4).

Таблица 4 – Результирующие метрики

Номер итерации	Точность (валидационная выборка), %
1	0,9870
2	0,9852
...	
19	0,9981
20	<b>0,9981</b>

В таблице 5 приведены результаты сравнения классификации символов с OCR сервисом Nanonets [10].

Таблица 5 – Сравнение созданной модели данных с OCR сервисом

Класс	Модель, %	OCR сервис, %	Δ, %
0	0,99996	0,86571	0,13425
1	0,999995	0,714521	0,285474
2	0,998339	0,877604	0,120735
<b>Итого</b>	<b>2,998294</b>	<b>2,457835</b>	<b>0,540459</b>

Видно, что для нескольких классов можно использовать собственную модель данных, однако в общем случае и для большего количества классов лучше использовать OCR сервис, так как сервис:

- Учитывает регистр букв;
- Умеет делать конкатенацию букв;
- Делает быструю цифровизацию текста;
- Автоматически приводит текст к нужному языку исходя из контекста;
- Поддерживает разные графические форматы изображений;
- Лучше классифицирует мульти-классовые изображения.

В дальнейшем используем OCR сервис для обработки данных, исходя из его преимуществ.

### **3. ОПИСАНИЕ РАЗРАБОТАННЫХ ПРОГРАММ**

В этой главе рассмотрены параметры разработанных программ, базы данных и веб-сервиса для решения задачи.

#### **3.1. МОБИЛЬНОЕ ПРИЛОЖЕНИЕ SYMBOL CLASSIFIER**

*Название приложения:* SymbolClassifier.apk

*Программное обеспечение и технологии использующиеся в процессе написания приложений:* для разработки мобильного приложения использовалась интегрированная среда разработки (IDE) Android Studio версии 2021.1.1 Patch 2.

*Язык программирования:* для написания приложения был выбран фреймворк Keras языка программирования Python версии 3.8.9 и Kotlin с версией плагина 1.6.20. Что касается языка программирования Python, то он получил большую известность в сфере задач анализа данных и машинного обучения за счет универсальности и возможности гибкой установки модулей. Дополнительно, Python может быть использован для написания мобильных приложений. В настоящее время Python находится на лидирующем месте среди всех языков программирования, благодаря своей универсальности и удобству. Язык Kotlin является рекомендуемым языком программирования [11] для создания мобильных приложений на Android и поддерживает стиль “Material Design”.

*Назначение приложений:* классификация символов.

*Сведения о функциональных ограничениях на применение:* наличие устройства на базе Android.

*Способ вызова программы:* запуск программы осуществляется нажатием ярлыка приложения.

Алгоритм программы представлен на рисунке 5.

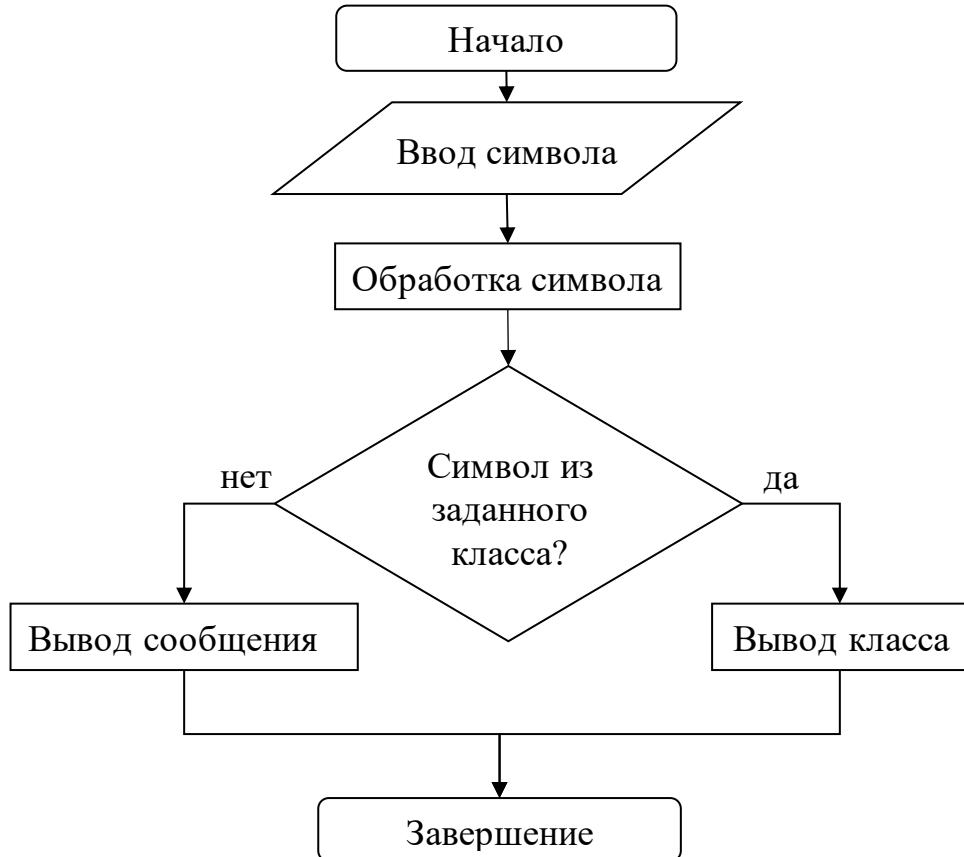


Рисунок 5 – Алгоритм программы для классификации заданного символа

*Используемые методы:* в качестве методов решения были выбраны многослойные свёрточные нейронные сети и методы компьютерного зрения (OpenCV).

*Связи программы с другими программами:* данная программа является частью варианты входных данных приложения “*Student Digitizer*”.

### 3.1.1. Входные и выходные данные приложения

*Описание входных данных:* исходные данные содержатся в файле модели “*converted\_model.tflite*” и представляют собой взаимосвязи между параметрами элементов датасета и их классом.

*Формат:* матрица входных данных имеет размер  $4320 \times 3$ , где 4320 – количество элементов для обучения модели и прогнозирования, 3 – количество классов, которые предлагается использовать для классификации символов. Для каждого символа было выбрано по 1440 элементов.

На рисунке 6 приведены данные для обучения модели “converted\_model.tflite”, которая используется в приложении.

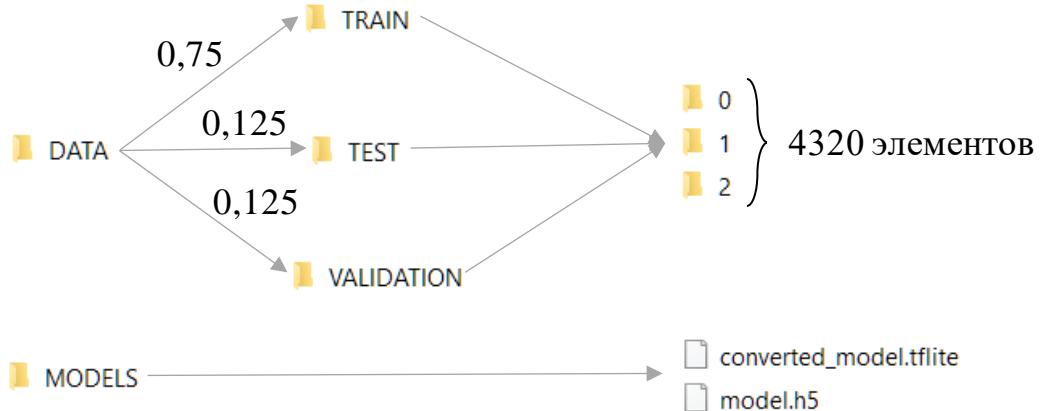


Рисунок 6 – Входные и выходные данные для мобильного приложения

Для корректной работы методов потребуется преобразовать названия классов из категориальных в числовые. Прежде чем приступить к анализу данных и обучению модели, необходимо проанализировать исходные наборы данных для определения качества выборки и сбалансированности подвыборок.

*Организация и предварительная подготовка входных данных:* На этапе предобработки данных происходит распределение данных (DATA) для тренировки (TRAIN), тестирования (TEST) и валидации (VALIDATION).

Предобработка данных:

- Последовательная нумерация файлов;
- Нормализация диапазона уровней изображений ( $0\text{--}255 \rightarrow 0\text{--}1$ );
- Приведение к единому размеру изображений;
- Перевод категориальных классов в числовые.

После проверки корректности данных разделяем выборку на подвыборки. Соотношение обучающей, тестовой и валидационной выборок выбрано 75%, 12.5% и 12.5% соответственно.

*Обучающая выборка* – это набор данных, который используется для разработки модели машинного обучения. *Валидационная выборка* – выборка, которая используется в процессе разработки модели машинного обучения для подбора оптимального набора гиперпараметров. *Тестовая выборка* – набор данных,

который не используется непосредственно в процессе обучения модели, однако позволяет протестировать модель.

*Описание и организация выходных данных:* результатом работы приложения является вывод на экран класса символа и точности классификации.

## **3.2. МОБИЛЬНОЕ ПРИЛОЖЕНИЕ STUDENT DIGITIZER**

*Название приложения:* StudentDigitizer.apk

*Программное обеспечение и технологии использующиеся в процессе написания приложений:* для разработки мобильного приложения использовалась интегрированная среда разработки (IDE) Pycharm версии 2021.3 (Professional Edition), Android Studio и Google Colab для сборки приложения.

*Языки программирования:* для написания приложения были выбраны языки программирования Python, SQL и Kivy (KV) [12]. Язык SQL используется для записи в базу данных (БД) BigQuery. Для графического интерфейса пользователя (GUI) были выбраны фреймворки Kivy версии 2.1.0rc1 и KivyMD [13] версии 1.0.0.dev0, так как KivyMD дополняет стандартный фреймворк и поддерживает стиль “Material Design” и использует собственный язык программирования.

*Назначение приложений:* извлечение и обработка данных для записи в БД.

*Сведения о функциональных ограничениях на применение:* наличие фотокамеры в Android устройстве и доступа в сеть Интернет для записи в базу данных и/или извлечения расписания в режиме онлайн.

*Способ вызова программы:* запуск программы осуществляется нажатием ярлыка приложения.

Алгоритм программы представлен на рисунке 7.

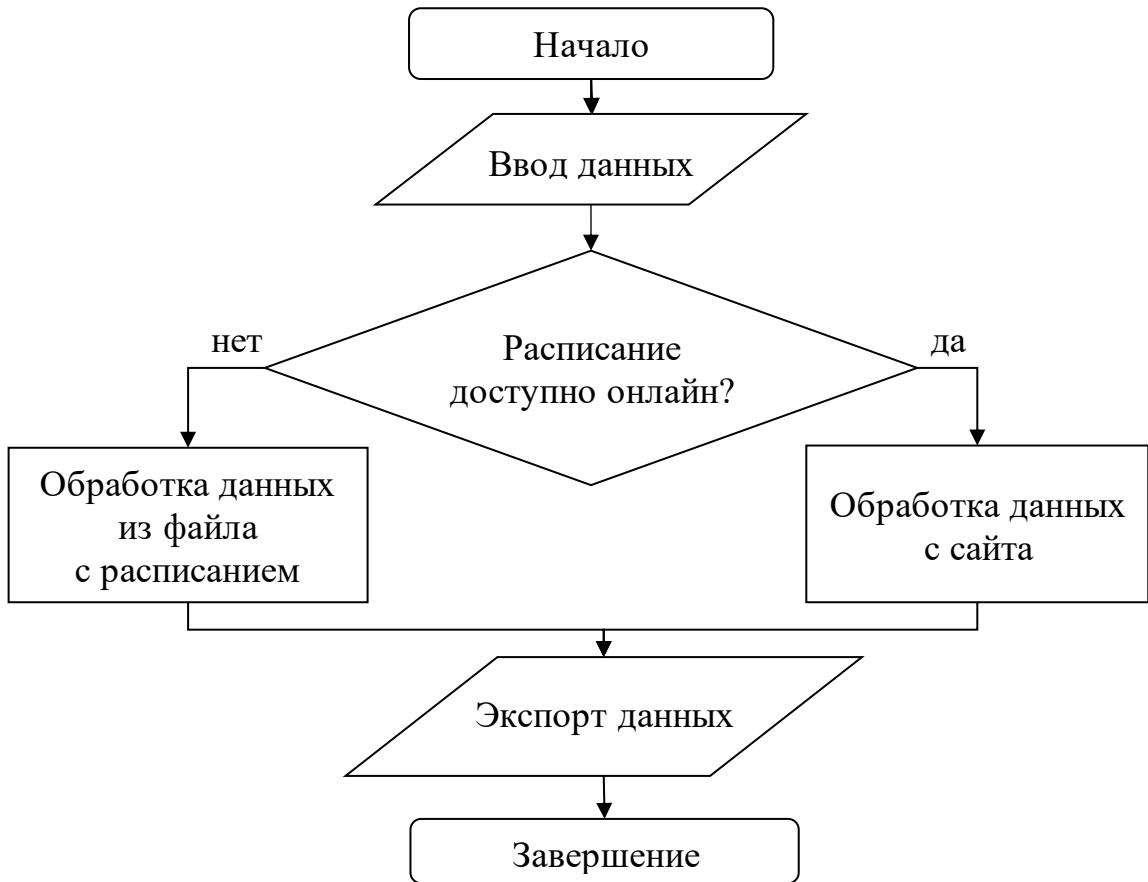


Рисунок 7 – Алгоритм приложения “*Student Digitizer*”

*Используемые методы:* для разработки приложения использовались ETL процессы сбора, перемещения и загрузки данных. Для обмена данными использовалась технология REST API для отправки и загрузки данных с сервера.

*Связи программы с другими программами:* данные программа связана с базой данных и визуализацией результатов работы приложения.

### 3.2.1. Входные и выходные данные приложения

*Описание входных данных:* исходные данные содержатся на сайте университета или в HTML файле расписания занятий студентов.

*Формат:* матрица входных данных после предобработки имеет размер  $40 \times 29$ , где 40 – количество строк, 29 – количество столбцов, которые предлагаются использовать для анализа и обработки данных.

Пример вымышленных входных данных в виде разворота студенческого журнала приведены на рисунках 8–9.

Номер	Фамилия, имя, отчество студента	Учёный							Последовательность							Протокол учета изъятого предмета
		20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет	Предмет
Иванов А.	Иванов А.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Б.	Иванов Б.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Г.	Иванов Г.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Д.	Иванов Д.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Е.	Иванов Е.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Ж.	Иванов Ж.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов З.	Иванов З.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов И.	Иванов И.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов К.	Иванов К.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Л.	Иванов Л.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов М.	Иванов М.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Н.	Иванов Н.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов О.	Иванов О.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов П.	Иванов П.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Р.	Иванов Р.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов С.	Иванов С.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Т.	Иванов Т.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Ф.	Иванов Ф.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Иванов Э.	Иванов Э.	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н	Н
Всего отсутствовало	1	2	11	0	1	1	1	1	1	1	1	1	1	1	1	10
Подпись преподавателя	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Подпись старосты	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Рисунок 8 – Сфотографированные входные данные для мобильного приложения

Месяц	Фамилия, инициалы студента	Числитель							Поминаемость							Пропущено часов занятий всего ученик по указанию преподавателей	Замечания деканата и преподавателей
		25	26	27	28	29	30	*	*	*	*	*	*	*	*		
Иванов А.		H	H			H		H	H	H	H	H	H	H		14	
Иванов Б.		H	H	H	H	H		H	H	H	H	H	H	H		6	
Иванов В.		H	H	H	H	H										1	
Иванов Г.		H	H	H	H	H										1	
Иванов Д.		H	H	H	H	H										1	
Иванов Е.		H	H	H	H	H										1	
Иванов Ж.		H		H	H	H										1	
Иванов З.			H		H	H										1	
Иванов Й.		H		H		H										1	
Иванов К.		H		H		H										10	
Иванов Л.		H		H		H										1	
Иванов М.		H		H		H										1	
Иванов Н.		H		H		H										1	
Иванов О.		H		H		H										1	
Иванов П.			H													H	
Иванов Р.			H													H	
Иванов С.			H			H										H	
Иванов Т.			H			H										H	
Иванов Ф.			H			H										H	
Иванов Э.			H			H										H	
Всего отсутствовало		1	2	11	0	1	1	10	1	1	1	1	1	1	1	1	10
Полиник преподавателя		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	
Подпись старосты		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	

Рисунок 9 – Отсканированные входные данные для мобильного приложения

Для полной обработки данных потребуется соединить две страницы в один разворот студенческого журнала.

*Организация и предварительная подготовка входных данных:* страницы журнала для верного считывания через OCR сервис должны быть четкими и читабельными. Предобработка данных нужна для структурированного сбора данных с сайта или файла и записи табличных данных в БД.

*Описание и организация выходных данных:* выходные данные представляют собой 2 таблицы, которые содержат обезличенную статистику активности студентов в группе и по учебным предметам университета.

### 3.3. МОДУЛЬ БАЗЫ ДАННЫХ

*Название схемы:* Table1. На рисунке 10 изображена схема и базы данных.

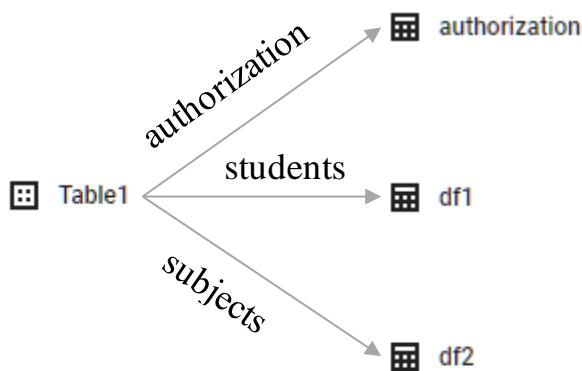


Рисунок 10 – Схема и базы данных

*Программное обеспечение и технологии использующиеся в процессе написания приложений:* бессерверная СУБД Google BigQuery.

*Язык программирования:* для разработки БД и записи/чтения данных был выбран язык SQL.

*Назначение базы данных:* чтение данных для визуализации данных.

*Сведения о функциональных ограничениях на применение:* наличие доступа к сети.

*Способ вызова базы данных:* стандартные запросы языка SQL.

Алгоритм программы представлен на рисунке 11.

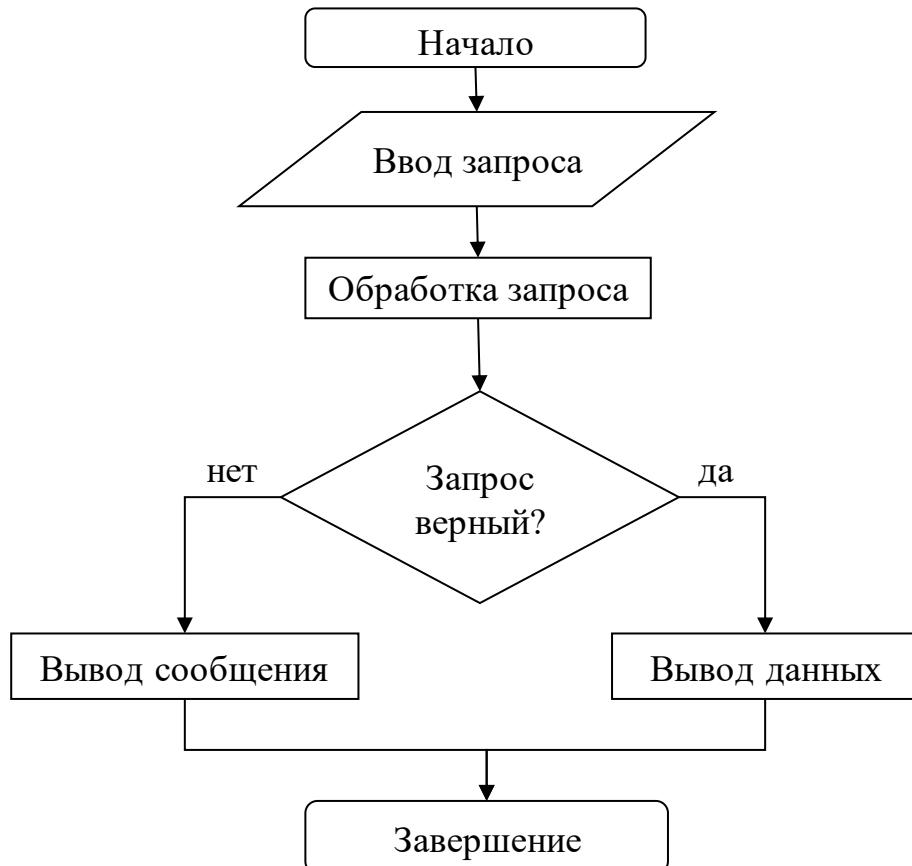


Рисунок 11 – Алгоритм запроса к базе данных

*Используемые методы:* Standard SQL.

*Связи базы данных с другими программами:* текущая БД позволяет визуализировать полученные результаты мобильных приложений.

### 3.3.1. Входные и выходные данные приложения

*Описание входных данных:* исходные данные для авторизации в приложении содержатся таблице “authorization”, данные студентов содержатся в приложении “df1”, данные учебных предметов в таблице “df2”.

*Формат:* рассмотрим таблицы 6–8.

Таблица 6 – Параметры таблицы авторизации “authorization”

Название	Описание	Тип	Свойства
login	Логин	STRING	NULLABLE
password	Пароль		
id_model	ID модели		
key_model	Ключ модели		
access_type	Тип доступа		

Таблица 7 – Параметры таблицы студентов “df1”

Название	Описание	Тип	Свойства
number	Номер	INTEGER	NULLABLE
student	Студент	STRING	
lectures_all	Пропуски всех занятий студентом	INTEGER	
group	Группа	STRING	
week_n	Номер недели	INTEGER	
date	Дата	TIMESTAMP	

Таблица 8 – Параметры таблицы предметов “df2”

Название	Описание	Тип	Свойства
subject	Предмет	STRING	NULLABLE
group	Группа	STRING	
week_n	Номер недели	INTEGER	
date	Дата	TIMESTAMP	
total	Пропуски предмета всеми студентами группы	INTEGER	

Таблицы студентов и предметов связаны по условию “group” = “group” соответственно и типу связи “многие ко многим” (many-to-many).

*Организация и предварительная подготовка входных данных:* предобработка данных выполняется в мобильном приложении до записи в БД.

*Описание и организация выходных данных:* выходные данные структурированы в таблицах.

### 3.4. МОДУЛЬ ВИЗУАЛИЗАЦИИ ДАННЫХ

*Названия приложений:* Dashboard (Tableau), Web Service (Streamlit).

*Программное обеспечение и технологии использующиеся в процессе написания приложений:* Tableau версии 2022.1.0, Streamlit версии 1.7.0.

*Язык программирования:* для написания веб-сервиса был выбран язык программирования Python.

*Назначение приложений:* визуализация полученных данных в виде дашборда и графиков.

*Сведения о функциональных ограничениях на применение:* наличие доступа к сети.

*Способ вызова программ:* просмотр дашборда доступен по ссылке [14], просмотр веб-сервиса доступен по адресу: <http://localhost:8501/> нажатием кнопки запуска.

Алгоритм программы визуализации результатов решения задачи представлен на рисунке 12.

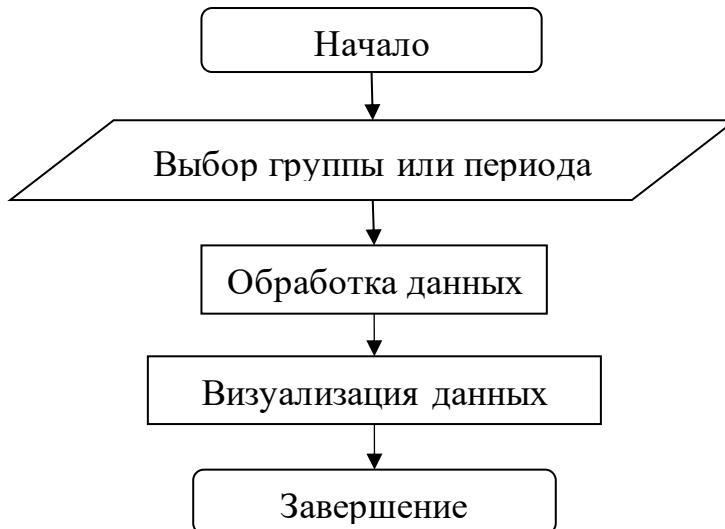


Рисунок 12 – Алгоритм программы визуализации результатов решения задачи

*Используемые методы:* программное обеспечение Tableau и фреймворк Streamlit.

*Связи программы с другими программами:* текущие визуализации данных представляют полученные результаты решения задачи.

### **3.4.1. Входные и выходные данные приложения**

*Описание входных данных:* дашборд представляет собой отдельные объекты на разных листах (таблица 9).

Таблица 9 – Описание листов в дашборде

График	Описание
Groups	Фильтр по группе или периоду
Subjects	Предметы и количество пропусков в каждой группе
Students	Пропуски каждого студента из группы
Dynamics	Динамика пропусков предметов группой
Dynamics_detail	Подробная динамика пропусков предметов группой
Percentage	Процентная диаграмма доли пропусков группы
Total	Общее количество пропусков
DASHBOARD	Дашборд

*Формат:* 8 листов, из которых 7 листов отдельные объекты дашборда.

*Организация и предварительная подготовка входных данных:* подбор оптимальных параметров для каждого объекта и группировка входных данных.

*Описание и организация выходных данных:* выходные данные представляют собой функциональный дашборд и веб-сервис с графиками полученных данных решения задачи.

## 4. ЭТАПЫ РЕШЕНИЯ ЗАДАЧИ

В этой главе рассмотрены основные этапы сбора и анализа цифрового следа студентов (рисунок 13). Сервисы использовались в рамках бесплатного уровня использования (free tier) [15–17].

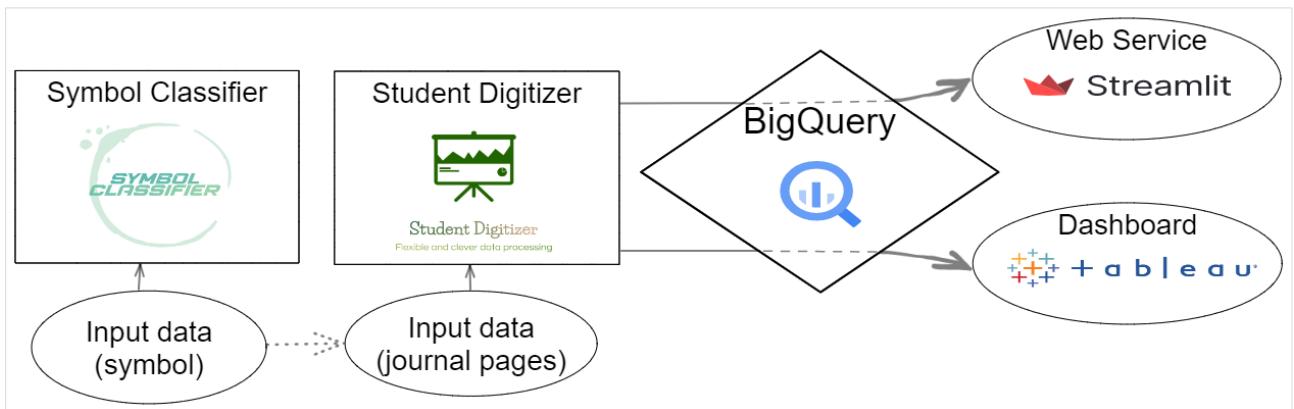


Рисунок 13 – Основные этапы сбора и анализа цифрового следа студентов

### 4.1. АНАЛИЗ ВХОДНЫХ ДАННЫХ

Рассмотрим входные онлайн данные (с подключением к сети) и оффлайн (без подключения к сети) расписания занятий [18].

Онлайн вариант представляет собой сбор данных с сайта университета. На рисунке 14 представлено расписание для различных групп студентов.

# Расписание занятий

Чёрным шрифтом отмечены группы, расписание для которых подготавливается.

Факультет автоматики и вычислительной техники

Факультет летательных аппаратов

Механико-технологический факультет

Факультет мехатроники и автоматизации

Факультет прикладной математики и информатики

Факультет радиотехники и электроники

Физико-технический факультет

Факультет энергетики

Факультет бизнеса

Факультет гуманитарного образования

Заочное отделение

Институт социальных технологий

## Очное отделение

Бакалавриат, специалитет

### 1 курс

ПМИ-11

ПМИ-12

ПМИ-13

ПМ-11

ПМ-12

ПМ-13

ПМ-14

ПМ-15

### 2 курс

ПМИ-01

ПМИ-02

ПМ-01

ПМ-02

ПМ-04

ПМ-05

### 3 курс

ПМИ-91

ПМИ-92

ПМ-91

ПМ-92

ПМ-93

ПМ-95

### 4 курс

ПМИ-81

ПМИ-82

ПМ-81

ПМ-83

ПМ-84

Магистратура

### 1 курс

ПМИМ-11

ПММ-11

ПММ-12

### 2 курс

ПМИМ-01

ПММ-01

ПММ-03

Рисунок 14 – Группы студентов

### 4.1.1. Расписания занятий на сайте университета

Рассмотрим расписание произвольной группы (рисунки 15–20).

## ФПМИ · ПММ-11

Сегодня 12 июня 2022, воскресенье

18 учебная неделя

	Предмет	Аудитория
ПН	08:30-10:00	
	10:15-11:45	
	12:00-13:30	
	14:00-15:30	НЕДЕЛИ 10 12 Межкультурное взаимодействие и коммуникация в профессиональной сфере · Домников П. А. · Практика
		НЕДЕЛИ 2 4 6 8 Межкультурное взаимодействие и коммуникация в профессиональной сфере · Домников П. А. · Лекция
	15:45-17:15	НЕДЕЛИ 2 4 6 8 10 12 Межкультурное взаимодействие и коммуникация в профессиональной сфере · Домников П. А. · Практика
	17:30-19:00	НЕДЕЛИ 2 4 6 8 Современные компьютерные технологии · Персова М. Г., Вагин Д. В. · Лекция
		НЕДЕЛИ 10 12 Современные компьютерные технологии · Персова М. Г., Вагин Д. В. · Лабораторная
	19:15-20:45	НЕДЕЛИ 2 4 6 8 10 12 Современные компьютерные технологии · Персова М. Г., Вагин Д. В. · Лабораторная
	21:00-22:30	

Рисунок 15 – Расписание занятий на понедельник

<b>ВТ</b>	08:30-10:00		
	10:15-11:45		
	12:00-13:30		
	14:00-15:30	НЕДЕЛИ 1 3 5 7 9    Объектно-ориентированный подход при разработке наукоемкого программного обеспечения · Рояк М. Э. · Лекция	1-307
	15:45-17:15	НЕДЕЛИ 1 3 5 7 9    Объектно-ориентированный подход при разработке наукоемкого программного обеспечения · Рояк М. Э. · Лекция	1-307
	17:30-19:00		
	19:15-20:45		
	21:00-22:30		

Рисунок 16 – Расписание занятий на вторник

<b>СР</b>	08:30-10:00		
	10:15-11:45		
	12:00-13:30		
	14:00-15:30	Иностранный язык · Практика	5-287
	15:45-17:15	НЕДЕЛИ 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18    Управление инновациями · Горевая Е. С. · Лекция	5-6
	17:30-19:00	НЕДЕЛИ 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18    Управление инновациями · Горевая Е. С. · Практика	5-6
	19:15-20:45		
	21:00-22:30		

Рисунок 17 – Расписание занятий на среду

<b>ЧТ</b>	08:30-10:00	НЕДЕЛИ 2 4 6 8 10 12 14    Моделирование стохастических динамических систем · Чубич В. М. · Практика	1-2086
	10:15-11:45	НЕДЕЛИ 2 4 6 8 10 12 14    Моделирование стохастических динамических систем · Чубич В. М. · Лекция	1-307
	12:00-13:30	НЕДЕЛИ 2 4 6 8 10 12 14    Компьютерные технологии моделирования и анализа данных · Лемешко Б. Ю. · Лекция	1-2036
	14:00-15:30	НЕДЕЛИ 2 4 6 8 10 12 14    Компьютерные технологии моделирования и анализа данных · Лемешко Б. Ю. · Практика	1-2036
	15:45-17:15		
	17:30-19:00		
	19:15-20:45		
	21:00-22:30		

Рисунок 18 – Расписание занятий на четверг

<b>ПТ</b>	08:30-10:00		
	10:15-11:45		
	12:00-13:30		
	14:00-15:30		
	15:45-17:15		
	17:30-19:00		
	19:15-20:45		
	21:00-22:30		

Рисунок 19 – Расписание занятий на пятницу

<b>сб</b>	08:30-10:00	
	10:15-11:45	
	12:00-13:30	
	14:00-15:30	
	15:45-17:15	
	17:30-19:00	
	19:15-20:45	
	21:00-22:30	

Рисунок 20 – Расписание занятий на субботу

#### 4.1.2. Расписание из HTML файла

Рассмотрим онлайн вариант расписания другой группы (HTML файл без CSS стилей) на рисунках 21–26. В данном случае применяются аналогичные способы обработки данных, за исключением этапа подключения к сайту с расписанием занятий.

## ФПМИ · ПММ-03

Сегодня 23 марта 2021, вторник 7 учебная неделя

Предмет

Аудитория

ПН

08:30-10:00

10:15-11:45

12:00-13:30

14:00-15:30

Иностранный язык 2-408

2-408

15:45-17:15

Прямые и итерационные методы решения больших разреженных систем уравнений · [Домников П. А.](#), 1-2086  
1-2086

17:30-19:00

по чётным Прямые и итерационные методы решения больших разреженных систем уравнений · [Домников П. А.](#), 1-2086  
1-2086

19:15-20:45

Рисунок 21 – Расписание занятий на понедельник

вт  
08:30-10:00

10:15-11:45

12:00-13:30

14:00-15:30

недели 1 3 5 7 9 11 13 Объектно-ориентированный подход при разработке научного программного обеспечения · [Рояк М. Э.](#), 1-307  
1-307

недели 2 4 6 8 10 12 Компьютерные технологии моделирования и анализа данных · [Персова М. Г.](#), 1-307  
1-307

15:45-17:15

недели 1 3 5 7 9 11 13 Объектно-ориентированный подход при разработке научного программного обеспечения · [Рояк М. Э.](#), 1-307  
1-307

недели 2 4 6 8 10 12 14 Компьютерные технологии моделирования и анализа данных · [Персова М. Г.](#), 1-307  
1-307

17:30-19:00

недели 1 3 5 7 9 11 13 Компьютерные технологии моделирования и анализа данных · [Персова М. Г.](#), 1-307  
1-307

19:15-20:45

Рисунок 22 – Расписание занятий на вторник

ср  
08:30-10:00

по нечётным Методология представления научно-технических результатов · [Домников П. А.](#), 1-307  
1-307

10:15-11:45

по нечётным Методология представления научно-технических результатов · [Домников П. А.](#), 1-307  
1-307

12:00-13:30

14:00-15:30

15:45-17:15

17:30-19:00

19:15-20:45

Рисунок 23 – Расписание занятий на среду

ЧТ  
08:30-10:00

10:15-11:45

12:00-13:30

14:00-15:30

15:45-17:15

17:30-19:00

19:15-20:45

Рисунок 24 – Расписание занятий на четверг

ПТ  
08:30-10:00

10:15-11:45

12:00-13:30

14:00-15:30

недели 2 4 6 8 10 14 Современные компьютерные технологии · [Персова М. Г.](#), [Вагин Д. В.](#), [Киселев Д. С.](#) 1-203а, 1-203б  
1-203а, 1-203б

15:45-17:15

недели 2 4 6 8 Современные компьютерные технологии · [Персова М. Г.](#), [Вагин Д. В.](#) 1-426  
1-426

недели 10 Современные компьютерные технологии · [Персова М. Г.](#), [Вагин Д. В.](#), [Киселев Д. С.](#) 1-203а, 1-203б  
1-203а, 1-203б

17:30-19:00

19:15-20:45

Рисунок 25 – Расписание занятий на пятницу

c6  
08:30-10:00

10:15-11:45

12:00-13:30

14:00-15:30

15:45-17:15

17:30-19:00

19:15-20:45

Рисунок 26 – Расписание занятий на субботу

Важно понять взаимосвязь между данными из расписания и структурой студенческого журнала для правильной записи учебных предметов. Для 1 учебного дня в журнале предусмотрено 4 столбца (рисунок 27).

Фамилия, инициалы студента	1	2	...					
	1	2	3	4	1	2	3	4

## Рисунок 27 – Запись учебных предметов

Далее, вместо номера пары вставляются в порядке очередности аббревиатуры учебных предметов из онлайн или офлайн расписания (рисунок 28), при этом рукописные названия предметов автоматически заменяются.

### Рисунок 28 – Аббревиатуры пар из расписания

Количество пропусков пересчитывается в процессе обработки информации исходя из данных журнала.

Для анализа пропусков студентов в студенческом журнале достаточно классифицировать следующие символьные данные:

- Б – пропуск по уважительной причине;
- Н – неявка на занятие;
- О – опоздание на занятие.

В дальнейшем нумерация классов Б – 0, Н – 1, О – 2 для обработки данных (таблица 10). Эти классы наиболее значимы, так как правильность их классификации позволяет использовать агрегатные функции и получать правильные результаты. Любые другие классы данных либо считаются автоматически, либо только проверяется их заполненность произвольными данными. Если символ не относится к какому-либо из классов с точностью не менее 75%, то считается, что символ принадлежит классу “*Nothing*”, который не указан в модели. Все классы содержат только заглавные буквы, чтобы улучшить метрику точности классификации.

Таблица 10 – Параметры модели для приложения “*Symbol Classifier*”

Класс	Количество изображений		
	Тренировка	Тестирование	Валидация
0			
1	1080	180	180
2			

Итоговая модель конвертируется в портативный формат TensorFlow Lite (tflite). Рассмотрим результаты обработки вымышленных входных данных с помощью Nanonets OCR (рисунки 29–30).

		Письменность										Чтение											
		Месяц					Год					Месяц					Год						
		Фамилия, инициалы студента	20				21				22				23				24				
		Иванов А.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Б.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Г.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Д.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Е.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Ж.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов З.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов И.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов К.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Л.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов М.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов О.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов П.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Р.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов С.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Т.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Ф.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
Предмет		Иванов Э.	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	И	С	Е	М	
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
			Было отсутствовало	Было в превосходительной степени	Было в ступени																		

Рисунок 29 – Результаты обработки с фотографированных входных данных

Месяц	Год	Фамилия, инициалы студента	Носимаемости												Всего ученых преподавателей
			1	2	3	4	5	6	7	8	9	10	11	12	
Иванов А.	1990	Иванов А.	*	*	*	*	*	*	*	*	*	*	*	*	14
Иванов Б.	1990	Иванов Б.	H	H	H	H	H	H	H	H	H	H	H	H	6
Иванов Г.	1990	Иванов Г.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов Д.	1990	Иванов Д.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов Е.	1990	Иванов Е.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов Ж.	1990	Иванов Ж.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов З.	1990	Иванов З.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов И.	1990	Иванов И.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов К.	1990	Иванов К.	H	H	H	H	H	H	H	H	H	H	H	H	10
Иванов Л.	1990	Иванов Л.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов М.	1990	Иванов М.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов Н.	1990	Иванов Н.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов О.	1990	Иванов О.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов П.	1990	Иванов П.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов Р.	1990	Иванов Р.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов С.	1990	Иванов С.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов Т.	1990	Иванов Т.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов Ф.	1990	Иванов Ф.	H	H	H	H	H	H	H	H	H	H	H	H	1
Иванов Э.	1990	Иванов Э.	H	H	H	H	H	H	H	H	H	H	H	H	1
Всего отсутствовало	1990		1	2	11	0	1	1	10	1	1	1	1	1	10
Поплыть преподавателя	1990		*	*	*	*	*	*	*	*	*	*	*	*	*
Помыть старосты	1990		*	*	*	*	*	*	*	*	*	*	*	*	*

Рисунок 30 – Результаты обработки отсканированных входных данных

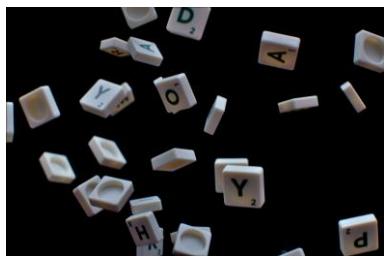
## 4.2. РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Основным условием задачи было создание мобильного приложения, которое по входным данным студенческого журнала может получить данные по студентам и предметам для визуализации результатов.

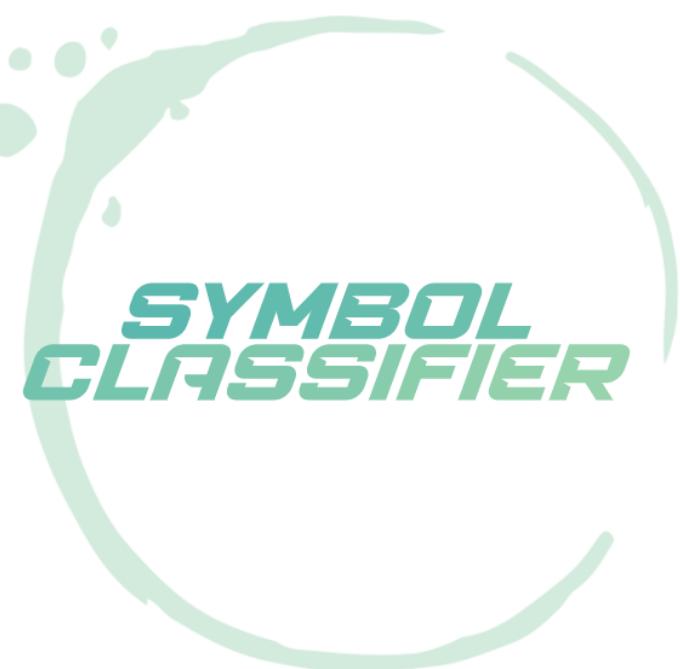
Для решения задачи были разработаны 2 приложения “*Symbol Classifier*” и “*Student Digitizer*”, при этом за один и тот же период для одной и той же группы студентов в процессе написания программы данные могли загружаться несколько раз.

Рассмотрим ярлык и логотип приложения “*Symbol Classifier*” (рисунок 31).

Ярлык



Логотип



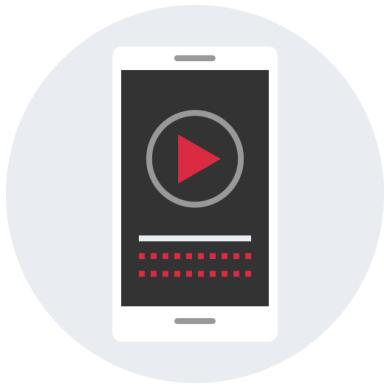
a)

б)

Рисунок 31 – Ярлык (а) и логотип (б) приложения “*Symbol Classifier*”

Рассмотрим ярлык и логотип приложения “*Student Digitizer*” (рисунок 32).

Ярлык



Логотип



## Student Digitizer

Flexible and clever data processing

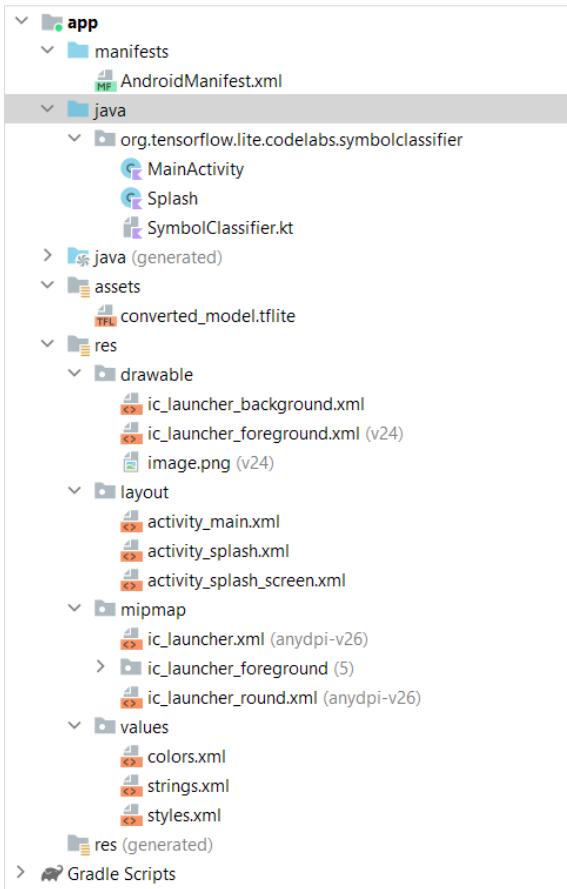
а)

б)

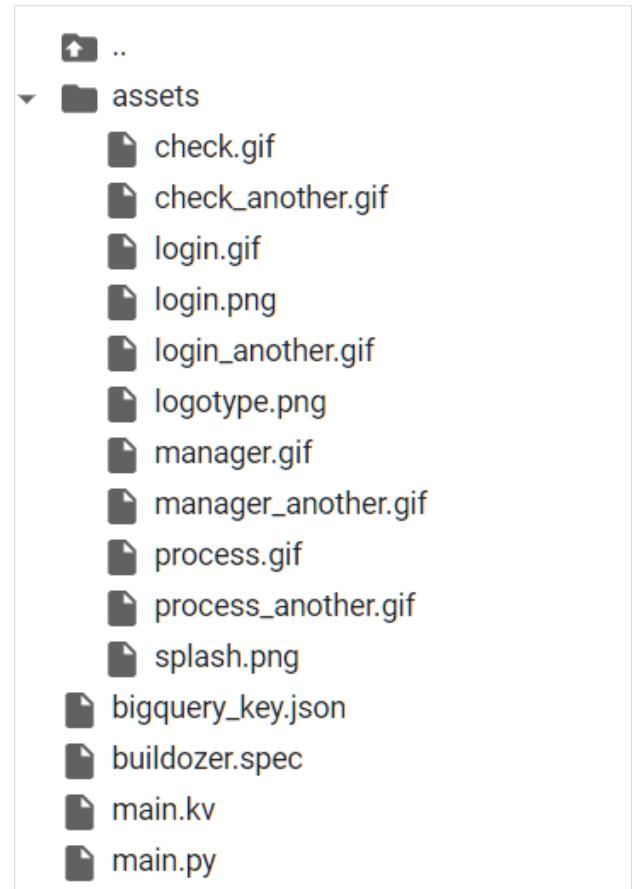
Рисунок 32 – Ярлык (а) и логотип (б) приложения “*Student Digitizer*”

Для более подробного сравнения разработанных мобильных приложений рассмотрим структуру приложений “*Symbol Classifier*” и “*Student Digitizer*” (рисунок 33). Для сборки приложения по классификации символов использовалась среда разработки Android Studio, основное приложение по сбору и анализу цифрового следа студентов собиралось в сервисе Google Colab [19].

“Symbol Classifier”



“Student Digitizer”



a)

б)

Рисунок 33 – Структуры мобильных приложений

“Symbol Classifier” (a) и “Student Digitizer” (б)

Тестирование приложений производилось на Android 11 версии, с 1.5 Гигабайт оперативной памяти и процессоре с 4 ядрами. Для стабилизации результатов в приложении “Symbol Digitizer” для классификации символов использовался порог отсечения 0.75.

Тестирование мобильного приложения “Student Digitizer” проводилось по фотографиям студенческого журнала. Для стабилизации результатов производилась проверка правильности входных данных.

Оба Android приложения могут быть опубликованы в магазине приложений Google Play. Используемые логотипы Android указаны в таблице 11.

Таблица 11 – Логотипы из мобильных приложений [20]

Логотип	Обозначение
	Нажатие на кнопку в мобильном приложении
	Автоматическая загрузка или обработка данных

### 4.3. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Для сбора сведений и системного анализа предметной области использовался предметный подход, так в ходе решения задачи база данных (БД) не была фиксированной. В этом случае БД может использоваться для решения задач и в других учебных сферах, что делает решение более универсальным.

Для проектирования БД были выбраны основные атрибуты, доступные как входные данные для анализа данных.

Рассмотрим таблицу “authorization” для авторизации (рисунок 34).

Row	login	password	id_model	key_model	access_type
1	user		***	***	user

Рисунок 34 – Таблица “authorization”

Рассмотрим пример заполнения таблиц “df1” для статистики по студентам, “df2” – по предметам (рисунок 35).

Row	number	student	lectures_all	group	week_n	date
1	1	Student1	0	ПММ-03	7	2022-01-31 00:00:00 UTC
2	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
3	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
4	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
5	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
6	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
7	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
8	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
9	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
10	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
11	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
12	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
13	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
14	1	Student1	0	ПММ-03	10	2022-01-08 00:00:00 UTC
15	1	Student1	0	ПММ-03	10	2022-01-11 00:00:00 UTC
16	1	Student1	0	ПММ-03	10	2022-01-11 00:00:00 UTC
17	1	Student1	0	ПММ-03	10	2022-01-11 00:00:00 UTC
18	1	Student1	0	ПММ-03	10	2022-01-11 00:00:00 UTC
19	1	Student1	0	ПММ-03	10	2022-01-11 00:00:00 UTC

Row	subject	group	week_n	date	total
1	ИЯП	ПММ-12	1	2022-02-07 00:00:00 UTC	0
2	КТМИАД	ПММ-12	1	2022-02-07 00:00:00 UTC	1
3	ПИИМРБРСУ	ПММ-12	1	2022-02-07 00:00:00 UTC	1
4	ОППРНПО	ПММ-12	1	2022-02-07 00:00:00 UTC	2
5	ИЯП	ПММ-11	1	2022-02-07 00:00:00 UTC	0
6	ОППРНПО	ПММ-11	1	2022-02-07 00:00:00 UTC	2
7	ИЯП	ПММ-12	1	2022-02-09 00:00:00 UTC	4
8	КТМИАД	ПММ-12	1	2022-02-09 00:00:00 UTC	5
9	ОППРНПО	ПММ-12	1	2022-02-09 00:00:00 UTC	10
10	ПИИМРБРСУ	ПММ-12	1	2022-02-09 00:00:00 UTC	14
11	ИЯП	ПММ-12	2	2022-02-09 00:00:00 UTC	0
12	СКТ	ПММ-12	2	2022-02-09 00:00:00 UTC	1
13	УИ	ПММ-12	2	2022-02-09 00:00:00 UTC	14
14	МВИКВПС	ПММ-12	2	2022-02-09 00:00:00 UTC	3
15	РУРИС	ПМ-92	2	2022-02-15 00:00:00 UTC	0
16	УРВС	ПМ-92	2	2022-02-15 00:00:00 UTC	0
17	ЯПИМТ	ПМ-92	2	2022-02-15 00:00:00 UTC	0
18	МО	ПМ-92	2	2022-02-15 00:00:00 UTC	1
19	ОЭЗ	ПМ-92	2	2022-02-15 00:00:00 UTC	1

а)

Рисунок 35 – Пример заполнения выходных таблиц  
студентов (а) и предметов (б)

б)

## **4.4. СОЗДАНИЕ ДАШБОРДА**

Основными данными для визуализации в виде дашборда является информация о посещаемости учебных занятий студентом группы.

Полный список источников данных представляет собой разворот студенческого журнала (2 страницы). В результате получается детальная статистика с фильтрами, которые можно менять в режиме реального времени.

К основным визуализируемым показателям можно отнести:

- Предметы и их посещаемость студентами;
- Динамику пропусков занятий;
- Процентное соотношение пропусков среди студенческих групп.

## **4.5. СОЗДАНИЕ ВЕБ-СЕРВИСА**

В качестве MVP веб-сервиса был сделан функциональный дизайн страницы веб-сервиса с подключенным взаимодействием с Google BigQuery. В качестве frontend и backend частей был выбран фреймворк Streamlit.

Тестирование проводилось используя сравнения визуализации данных в дашборде и БД. Веб-сервис реализован по адресу: <http://localhost:8501/>.

## 5. ИССЛЕДОВАНИЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

В этой главе показаны результаты решения задачи в виде мобильных приложений и визуализация данных через дашборд и веб-сервис.

### 5.1. МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ

Основные скриншоты мобильных приложений (рисунки 36–64).

#### 5.1.1. Приложение Symbol Classifier

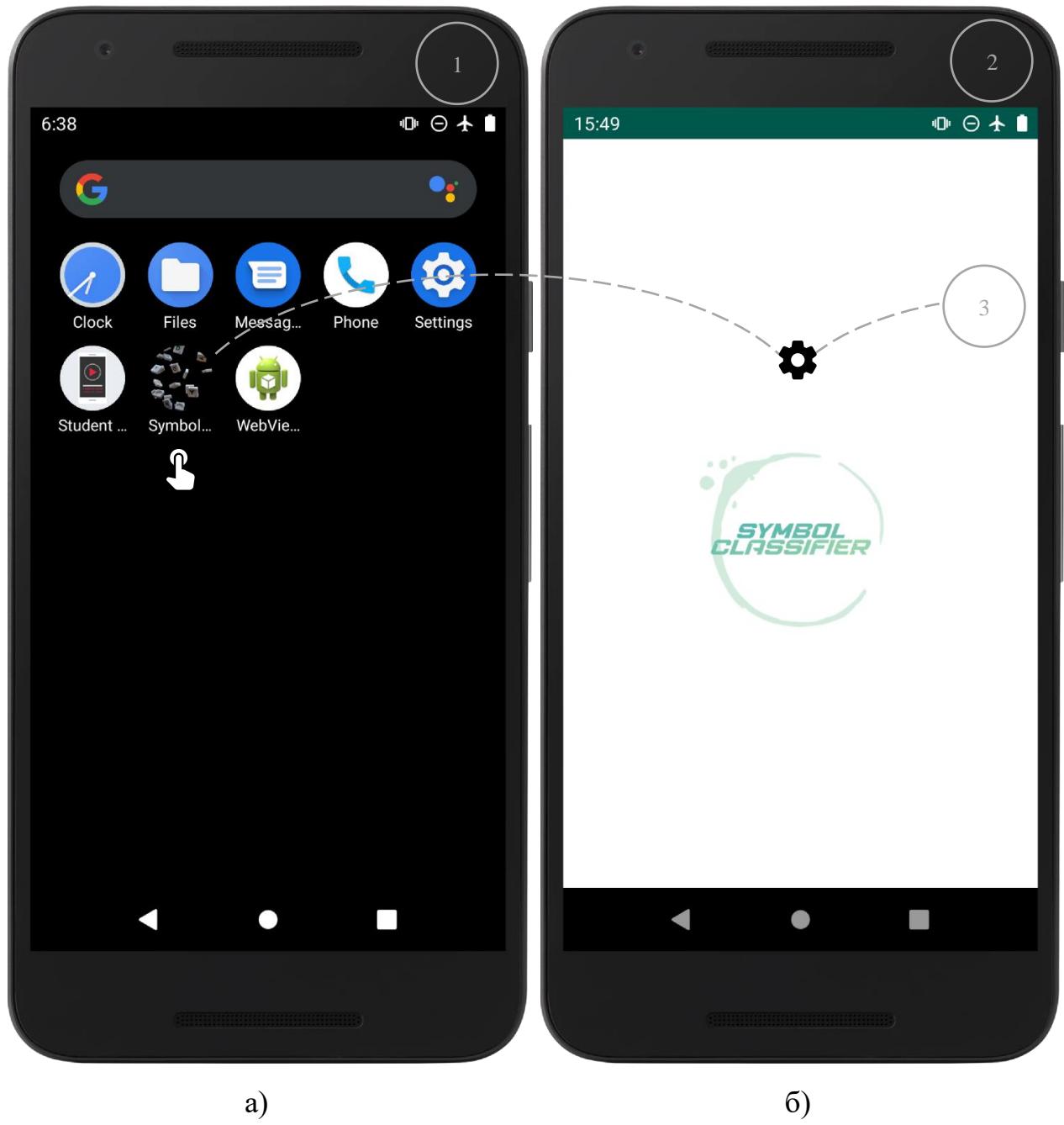


Рисунок 36 – Приложение на рабочем экране (а) и заставка (б)

Мобильное приложение “*Symbol Classifier*” динамически считает метрику точности классификации во время выполнения действий на экране (рисунок 37).



Рисунок 37 – Классификация символов “Б” (а) и “Н” (б)

Класс “*Nothing*” означает, что нарисованный символ не относится к какому-либо из классов 0, 1, 2 (рисунок 38).

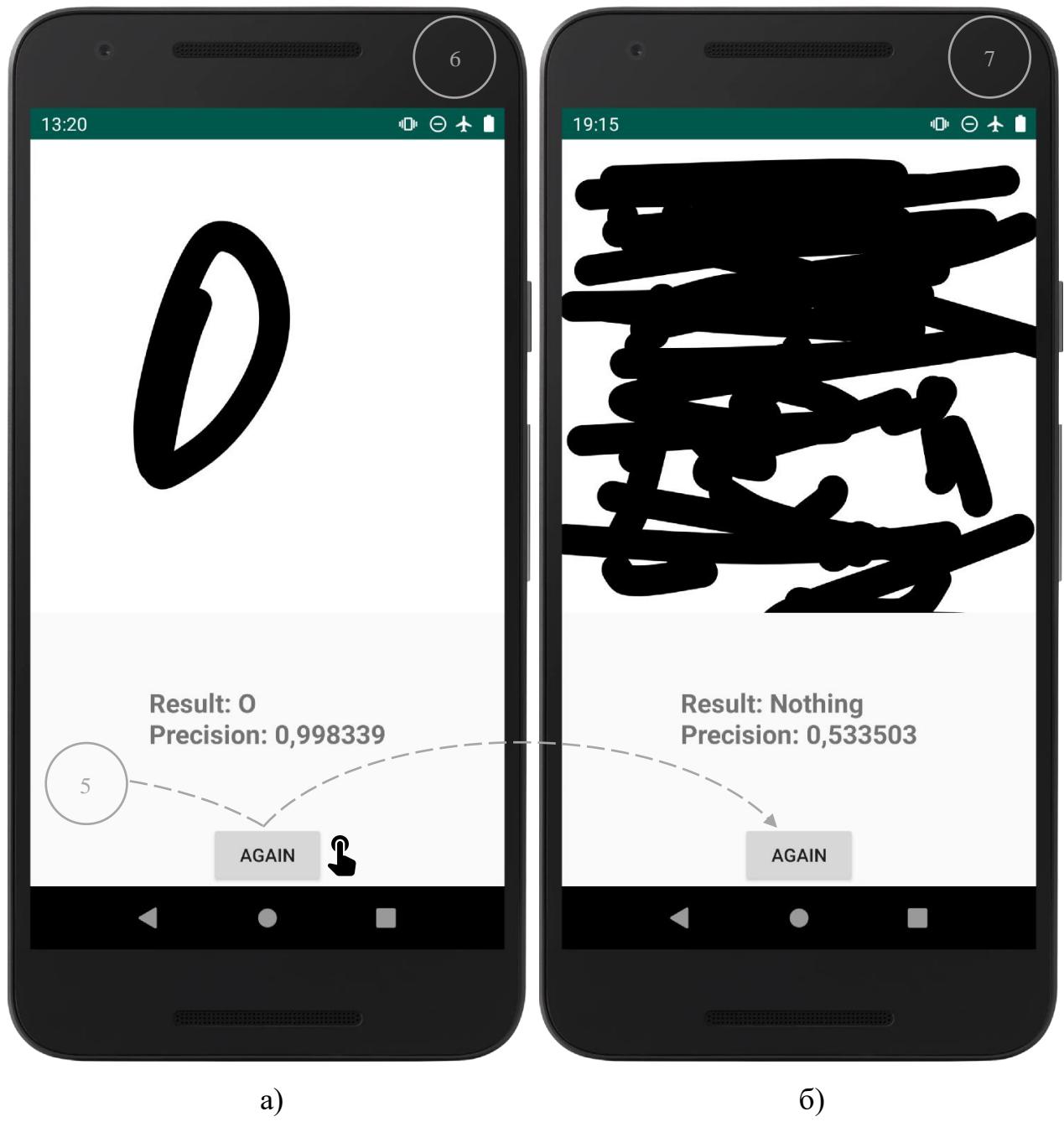


Рисунок 38 – Классификация символов “O” (а) и “Nothing” (б)

### 5.1.2. Приложение Student Digitizer

Основные скриншоты мультиязычного мобильного приложения “*Student Digitizer*” с различными темами для решения задачи сбора и анализа цифрового следа студентов. Приложение включает анимированные GIF файлы [21].

Английская версия приложения, энергосберегающая тема (рисунок 39).

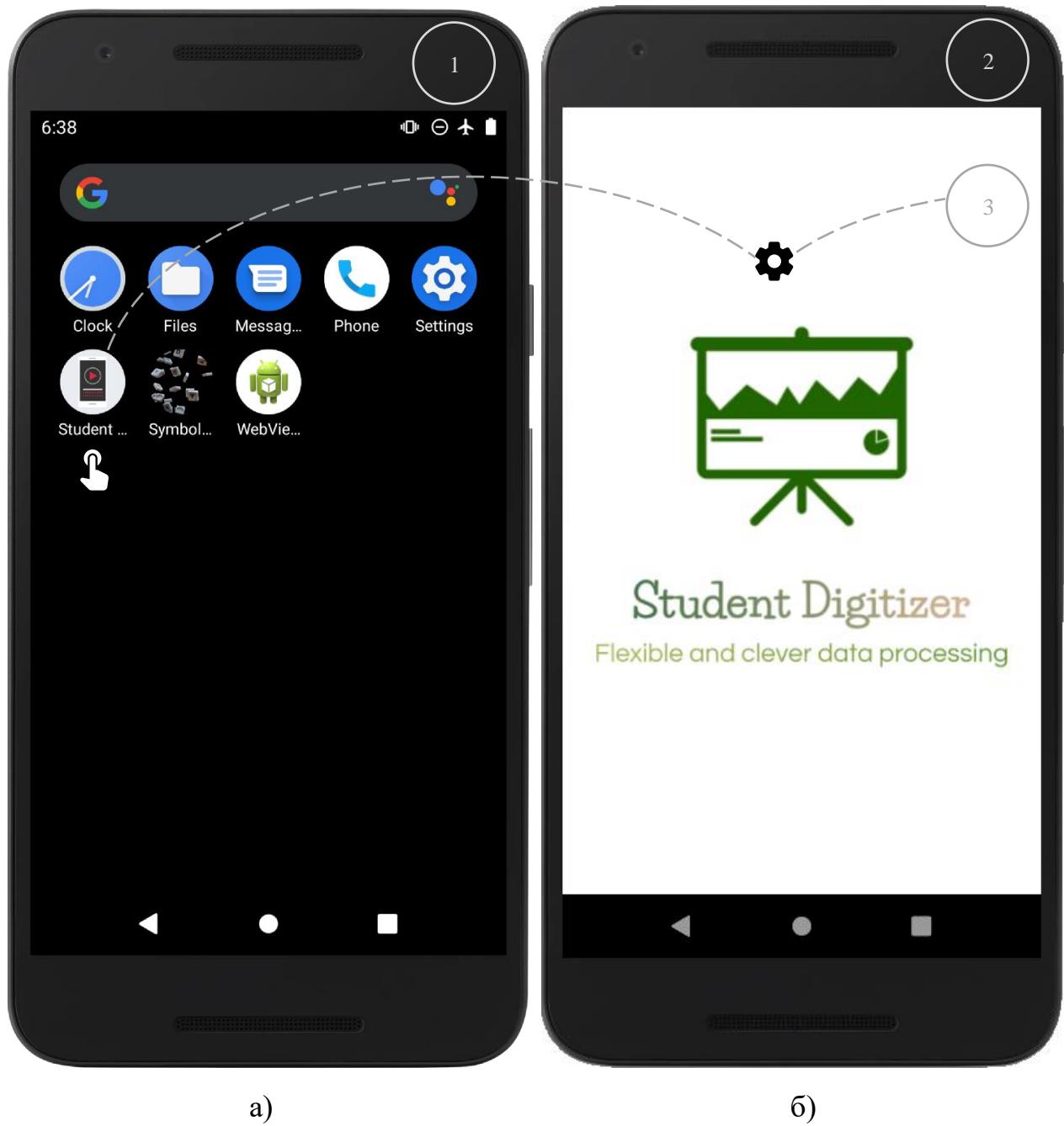


Рисунок 39 – Приложение на рабочем экране (a) и заставка (б)

Стартовый экран приложения, с проверкой введенных данных, если данные не введены, то некоторые элементы интерфейса недоступны (рисунок 40).



Рисунок 40 – Стартовый экран приложения (а),  
инструкция и лицензионное соглашение (б)

В приложении предусмотрена авторизация через логин и пароль (рисунок 41).

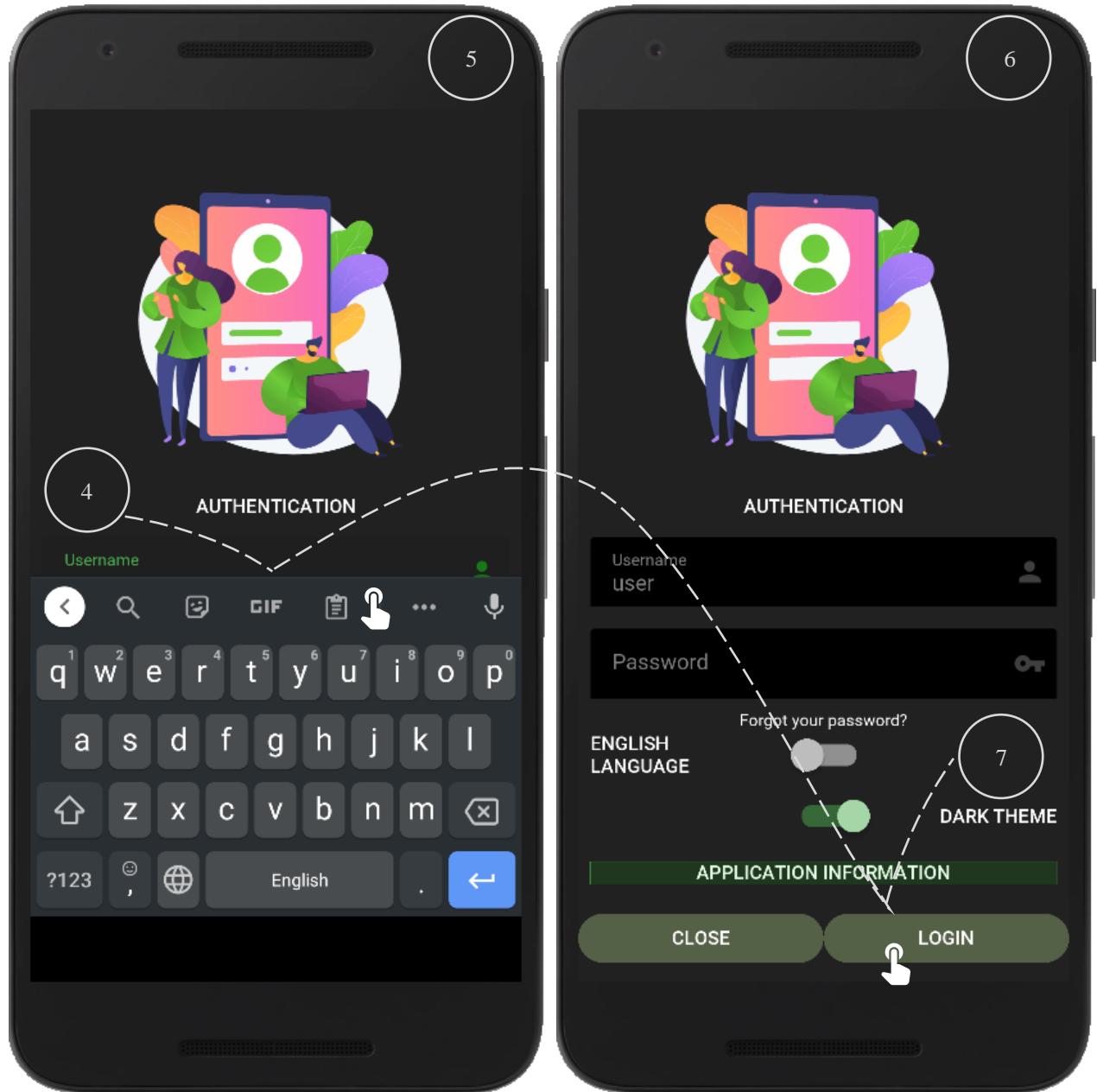


Рисунок 41 – Процесс авторизации в приложении

После авторизации в приложении отображаются параметры модели оптического распознавания символов (рисунок 42).

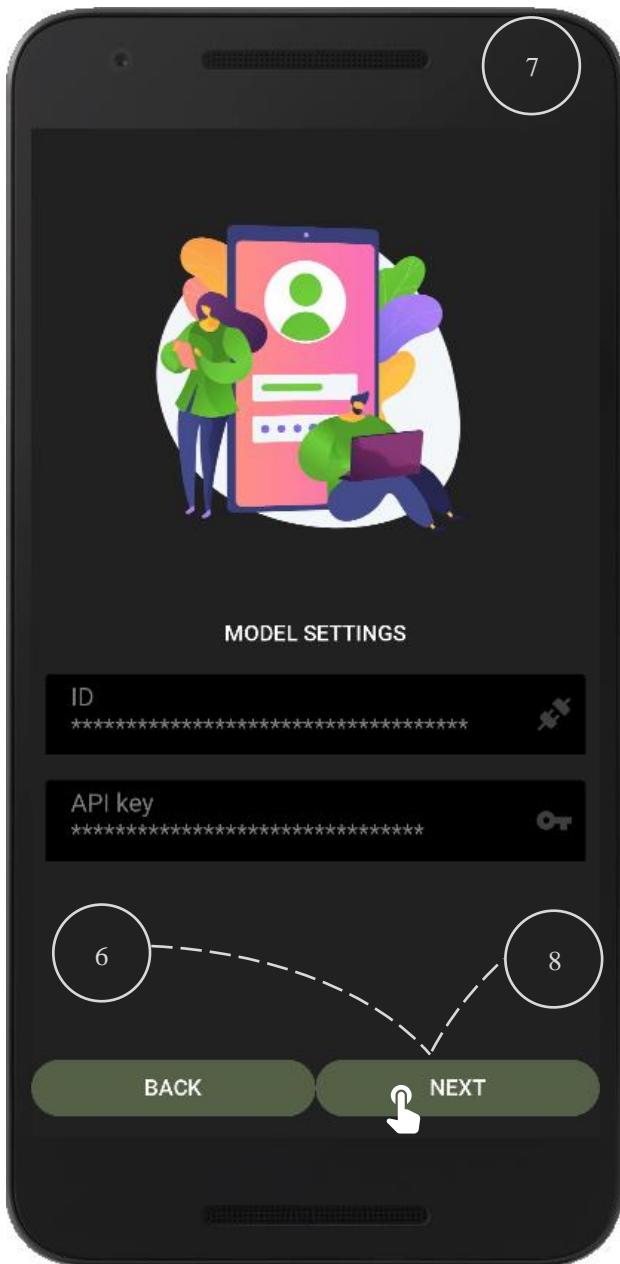


Рисунок 42 – Проверка доступности модели для обработки данных

В приложении “*Student Digitizer*” реализованы онлайн и офлайн варианты получения данных. Возможна загрузка студенческого журнала за другой номер недели с соответствующим расписанием занятий, например, если пользователь загружает несколько отчетов сразу в конце семестра.

Онлайн вариант: данные загружаются с сайта расписания университета (рисунок 43).

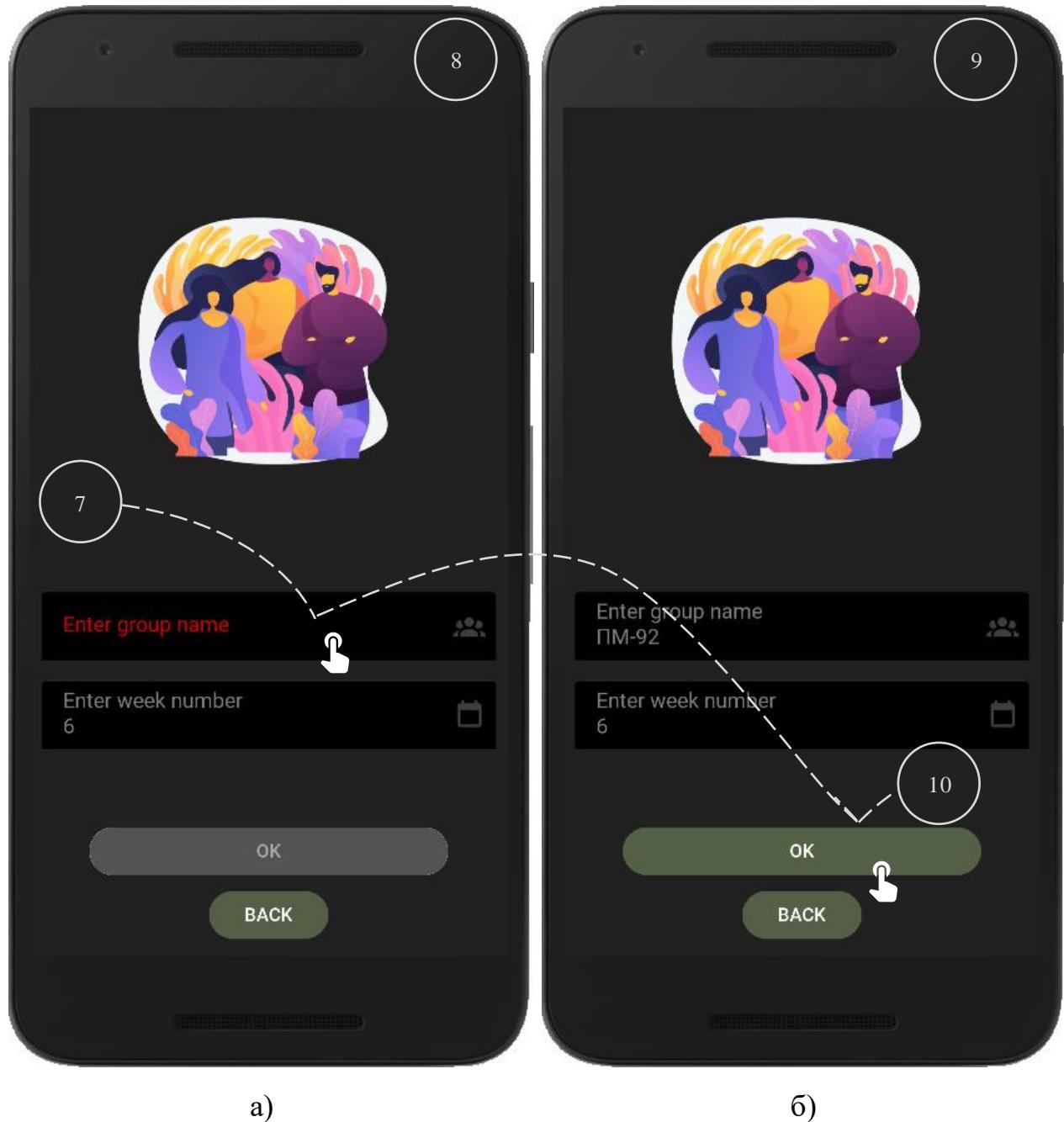


Рисунок 43 – Онлайн версия сбора данных с вводом группы студентов (а) и нажатием кнопки “OK” для загрузки учебных предметов (б)

Аббревиатуры предметов (рисунок 44) используются для записи в таблицу данных предметов как показано на рисунках 27–28.

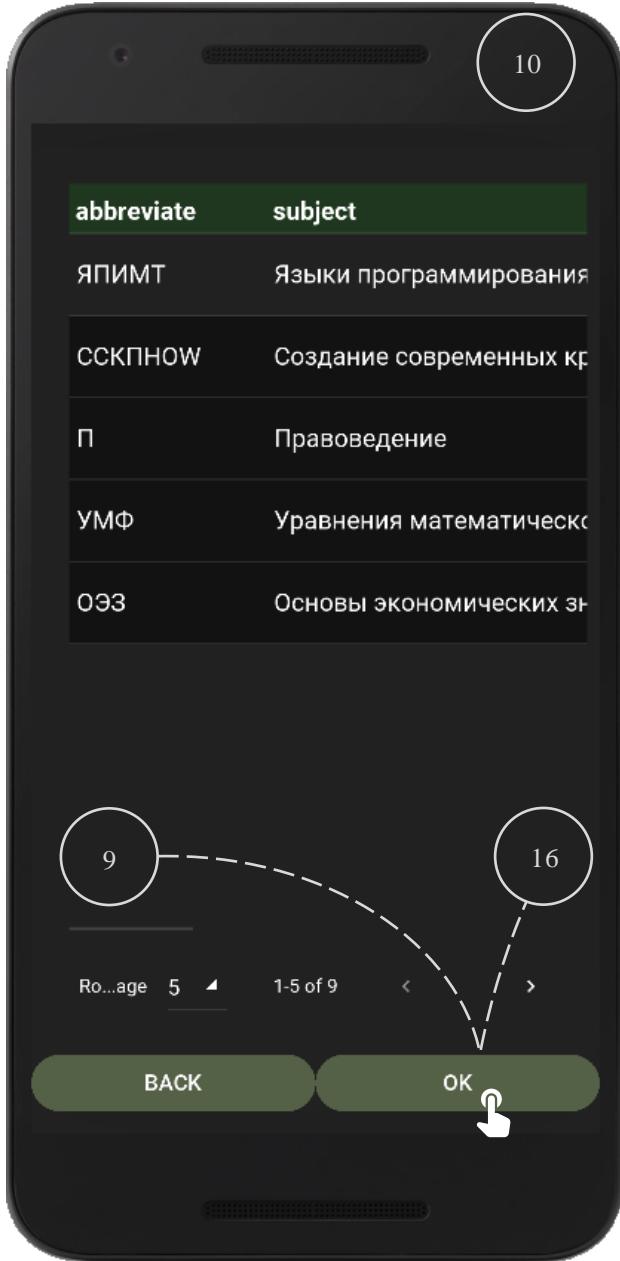


Рисунок 44 – Результат загрузки данных расписания занятий онлайн и переход к загрузке страниц студенческого журнала нажатием кнопки “OK”

Оффлайн вариант: данные загружаются из HTML файла с расписанием занятий (рисунок 45).

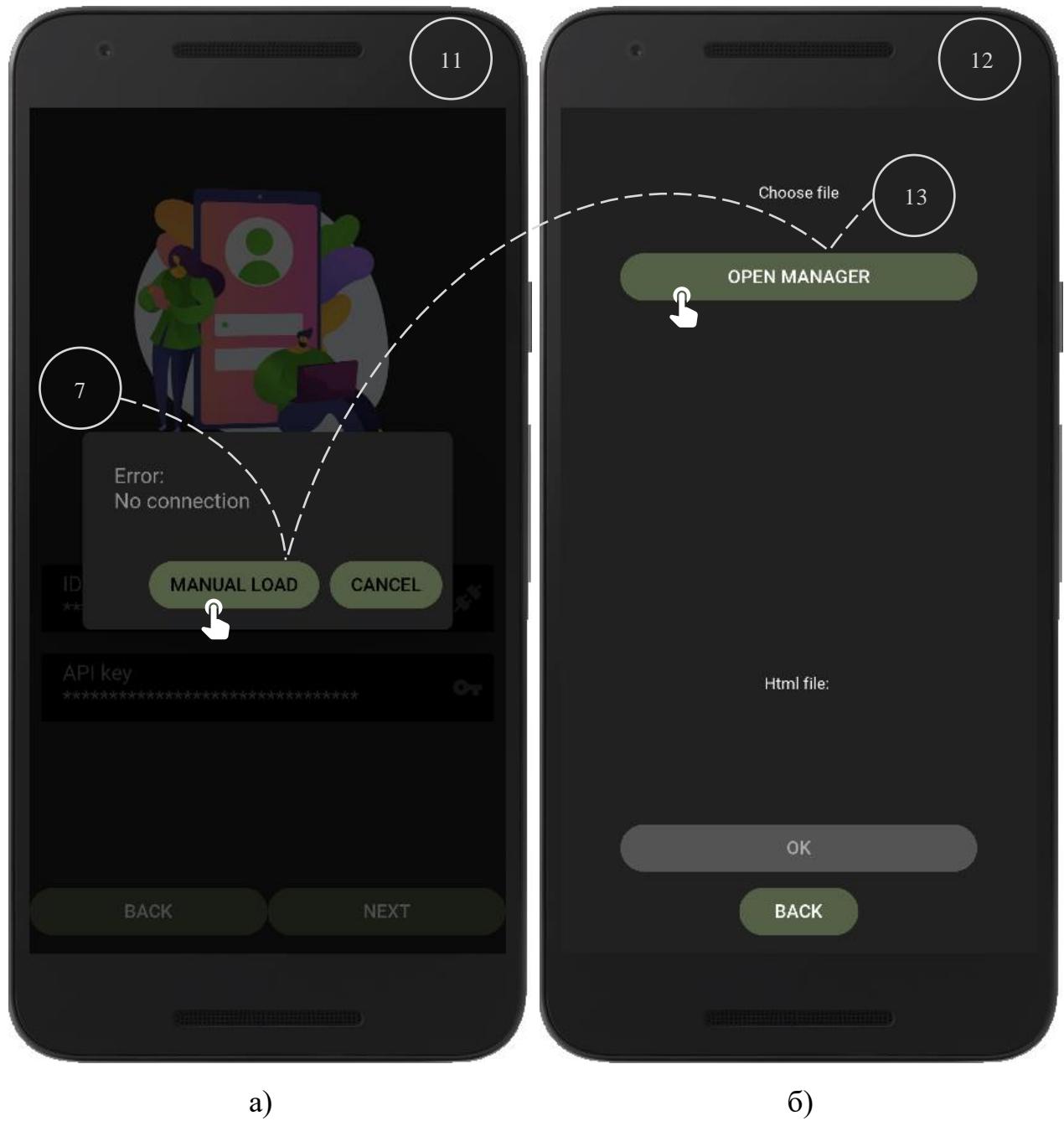


Рисунок 45 – Оффлайн версия сбора данных при недоступности данных с сайта (а) и открытие файлового менеджера (б)

Выбор HTML файла для обработки данных расписания занятий (рисунок 46).

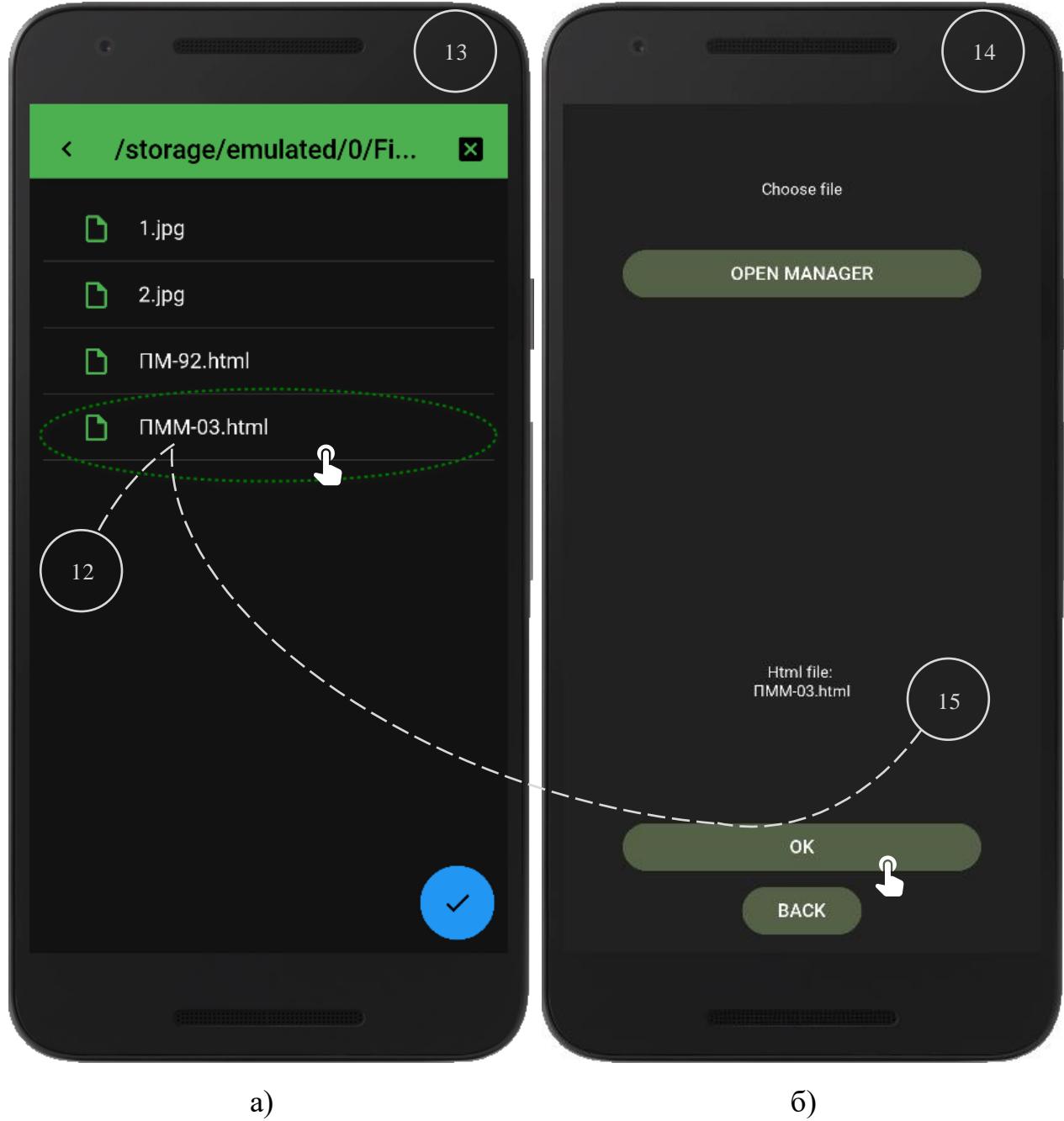


Рисунок 46 – Выбор HTML файла расписания группы студентов (а) и нажатие кнопки “OK” для загрузки учебных предметов (б)

Аббревиатуры предметов для записи в базу данных (рисунок 47).

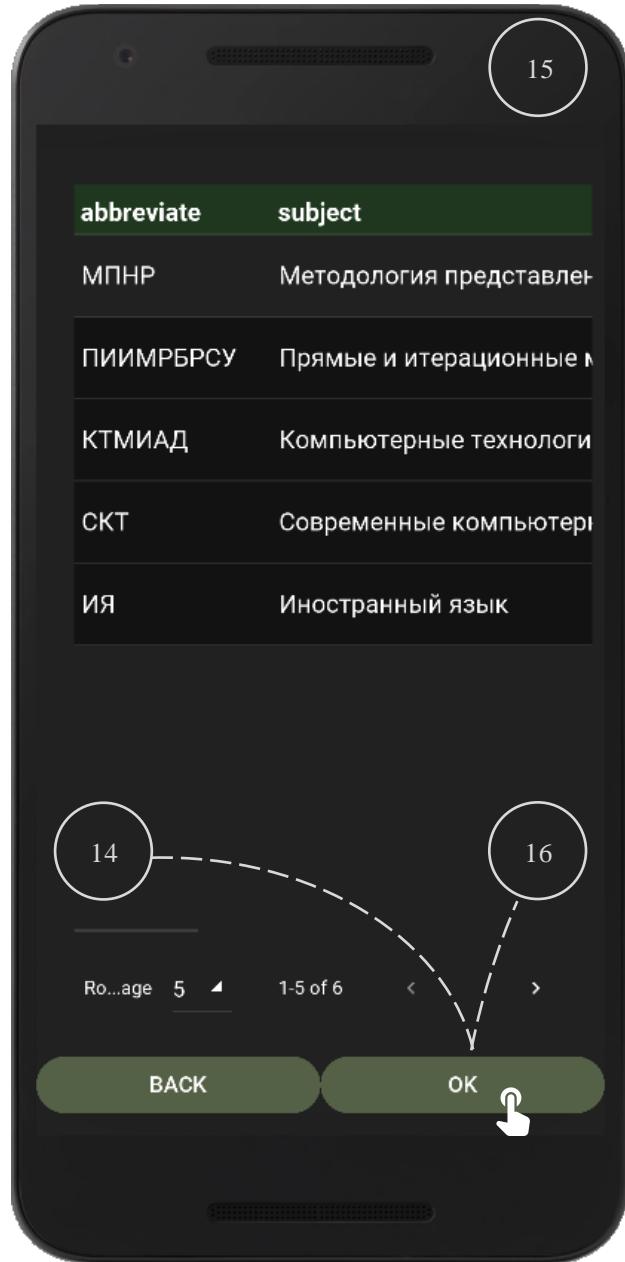


Рисунок 47 – Результат загрузки данных расписания занятий онлайн и переход к загрузке страниц студенческого журнала нажатием кнопки “OK”

На следующем этапе предлагается выбрать 2 страницы разворота студенческого журнала для обработки данных (рисунок 48).

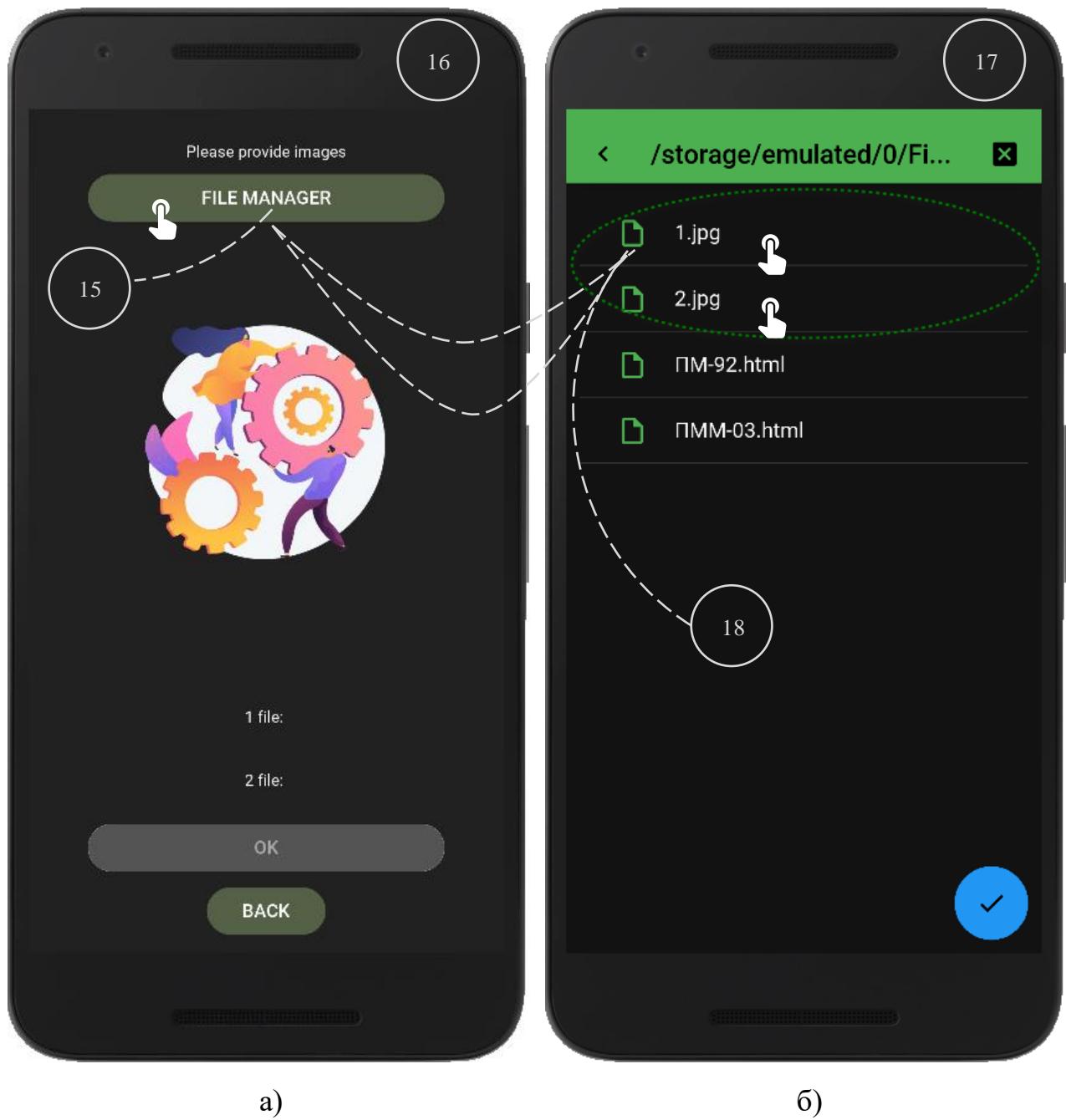


Рисунок 48 – Открытие файлового менеджера (а)  
и выбор изображений студенческого журнала (б)

Обработка страниц студенческого журнала (рисунок 49).



Рисунок 49 – Нажатие кнопки “OK” (а) и обработка изображений (б)

Таблица для записи в базу данных BigQuery (рисунок 50).

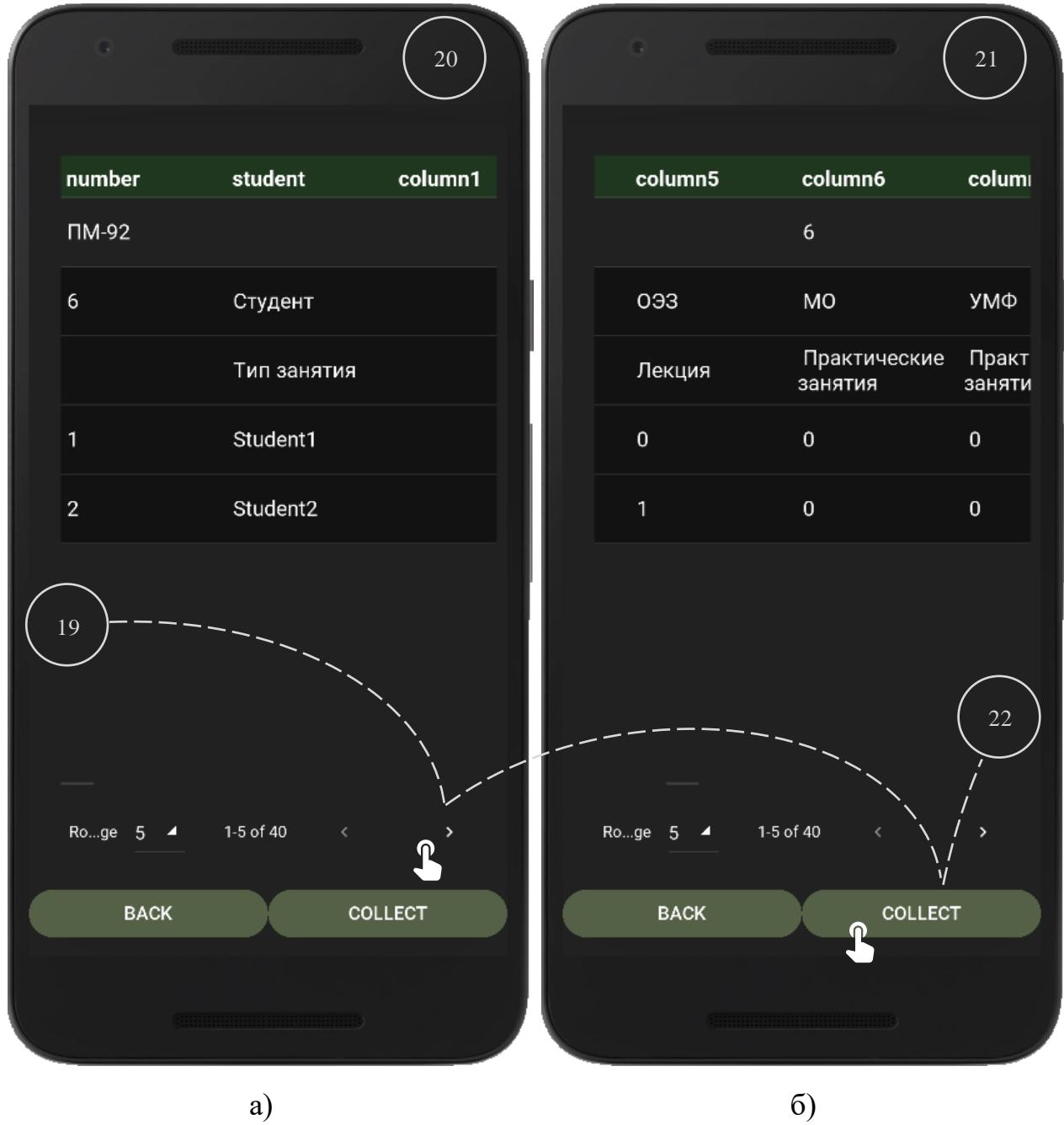


Рисунок 50 – Результаты обработки данных студенческого журнала (a, б) и нажатие кнопки “Collect” для записи в базу данных (б)

Запись в базу данных BigQuery (рисунок 51).

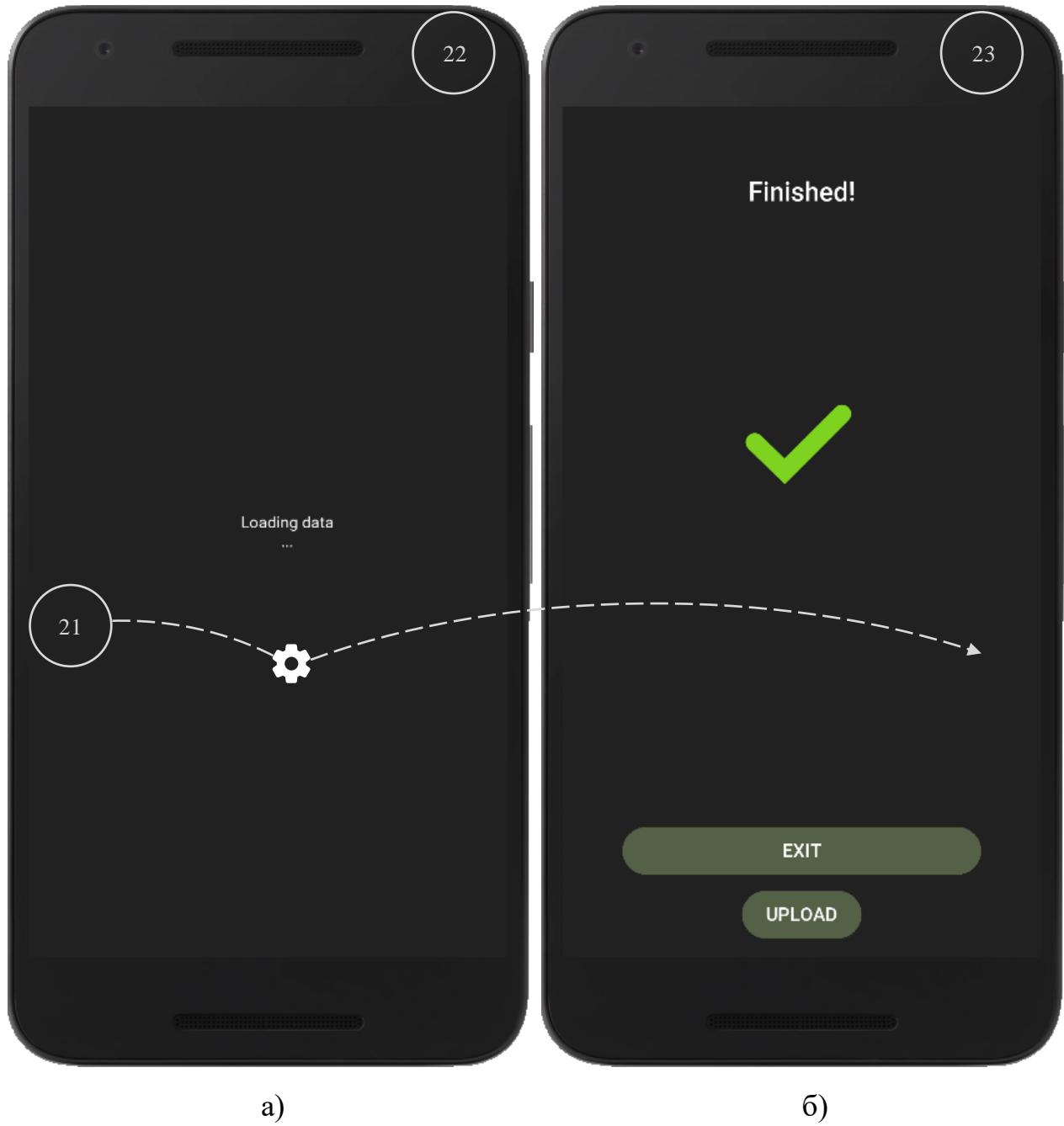


Рисунок 51 – Процесс записи данных в базу данных (а)  
и сообщение об успешном окончании обработки файлов (б)

Русская версия приложения, стандартная тема (рисунок 52).

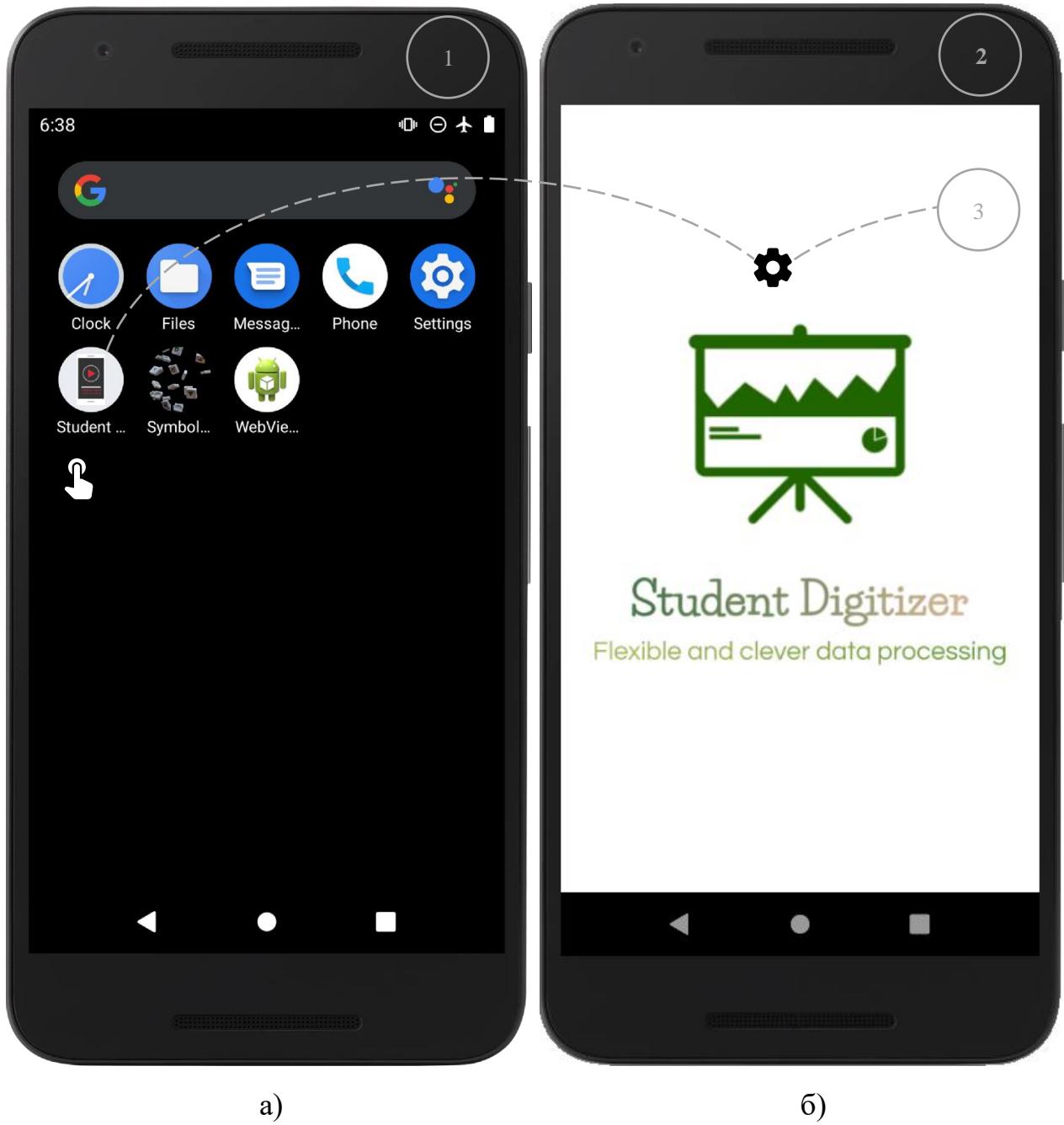


Рисунок 52 – Приложение на рабочем экране (*a*) и заставка (*b*)

Стартовый экран приложения, проверка ввода данных (рисунок 53).



а)

б)

Рисунок 53 – Стартовый экран приложения (а),  
инструкция и лицензионное соглашение (б)

Авторизация в приложении (рисунок 54).

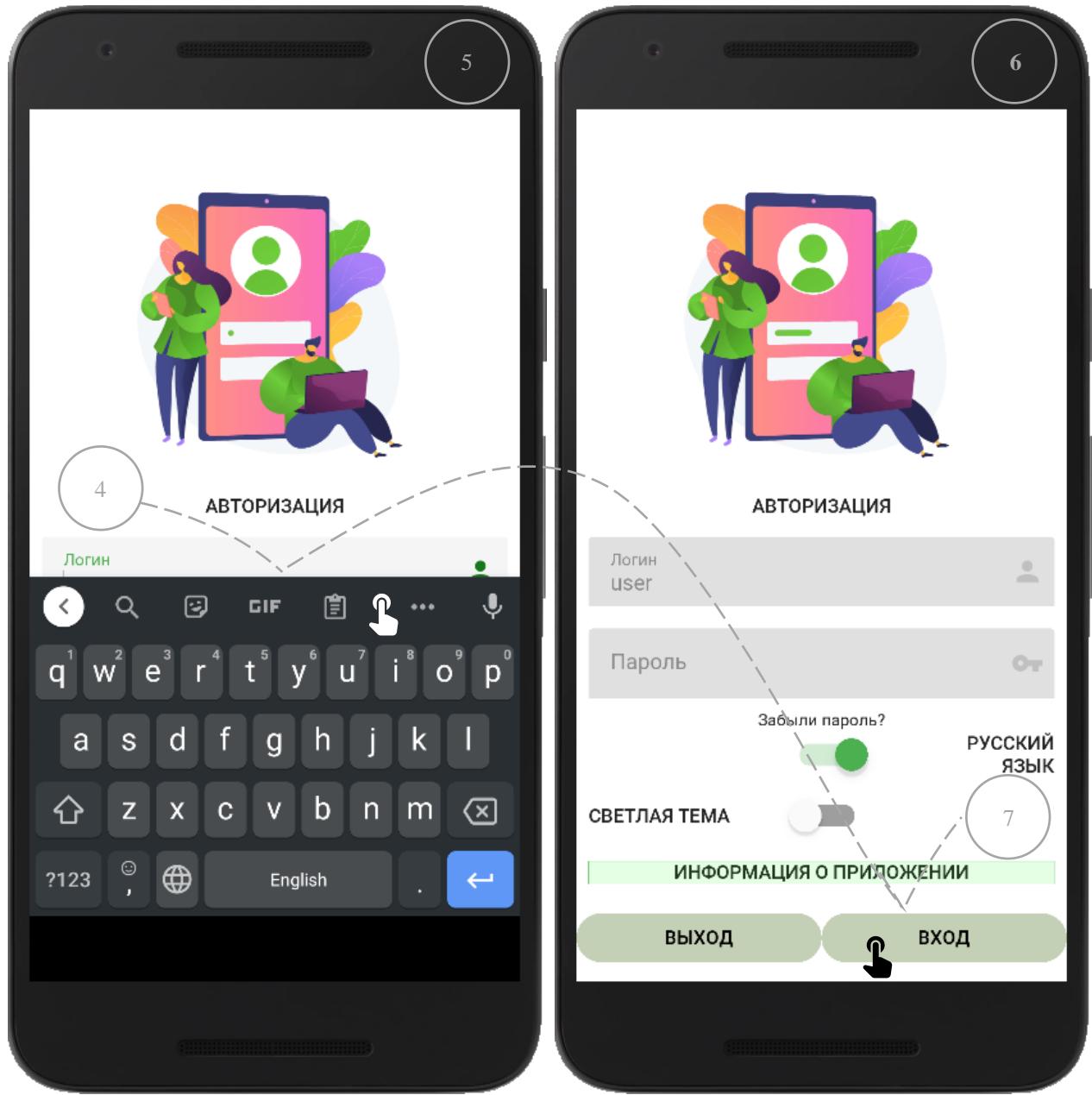


Рисунок 54 – Процесс авторизации в приложении

В таблице “authorization” содержатся данные OCR модели для каждого пользователя (рисунок 55).

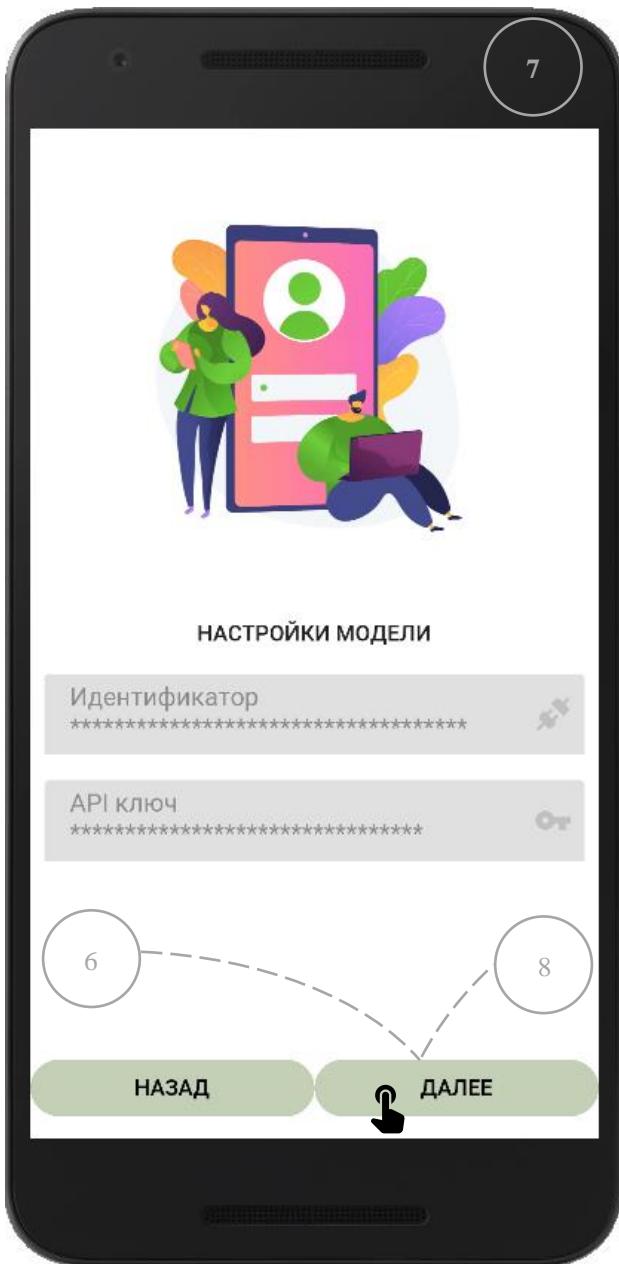


Рисунок 55 – Проверка доступности модели для обработки данных

Онлайн вариант: данные загружаются с сайта расписания университета (рисунок 56).

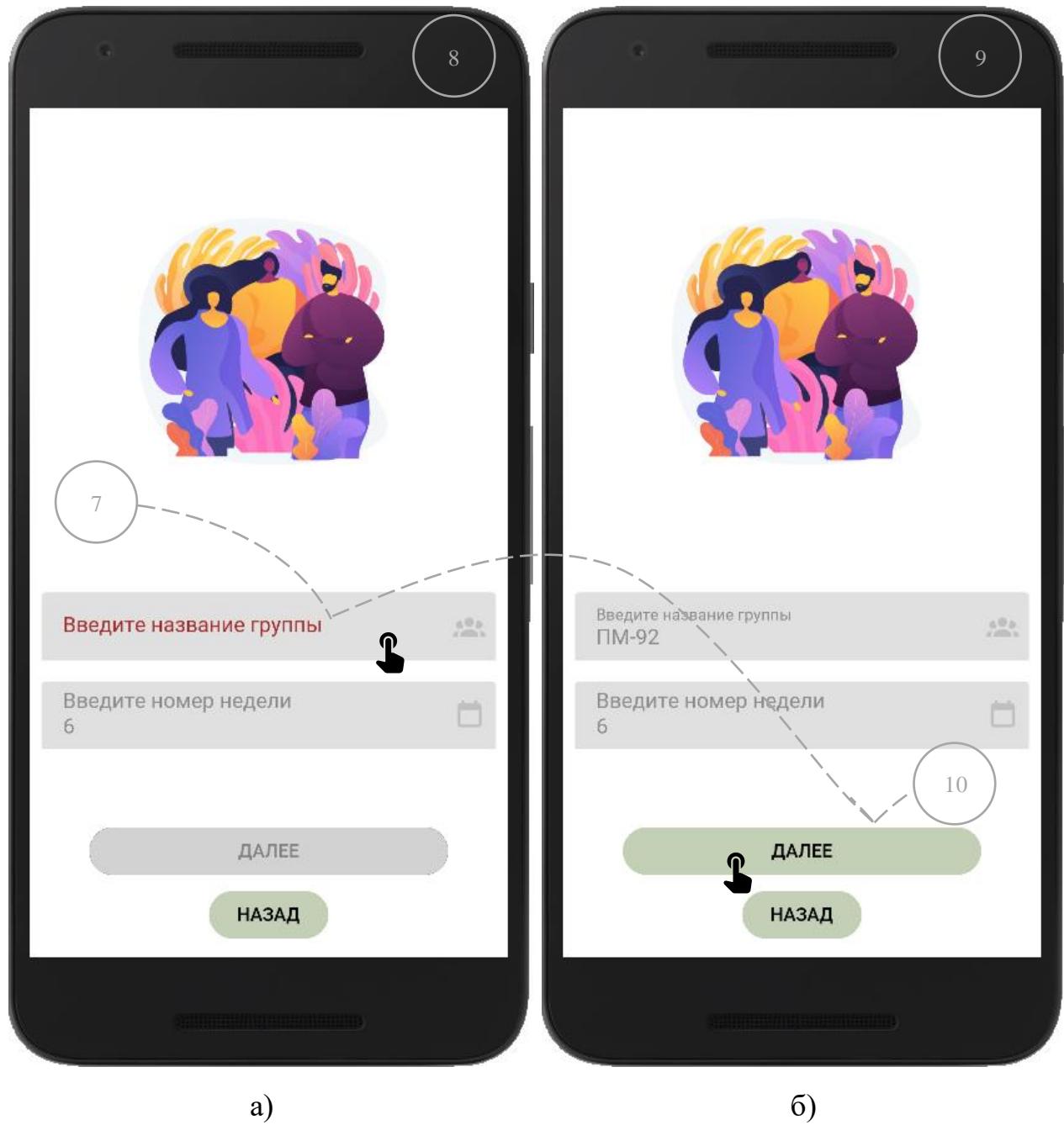


Рисунок 56 – Онлайн версия сбора данных с вводом группы студентов (а) и нажатием кнопки “OK” для загрузки учебных предметов (б)

Аббревиатуры предметов (рисунок 57) позволяют стандартизировать названия предметов и упростить отображение данных. Если в названии предмета какое-либо слово начинается с английского алфавита, то аббревиатура может состоять из букв нескольких алфавитов.

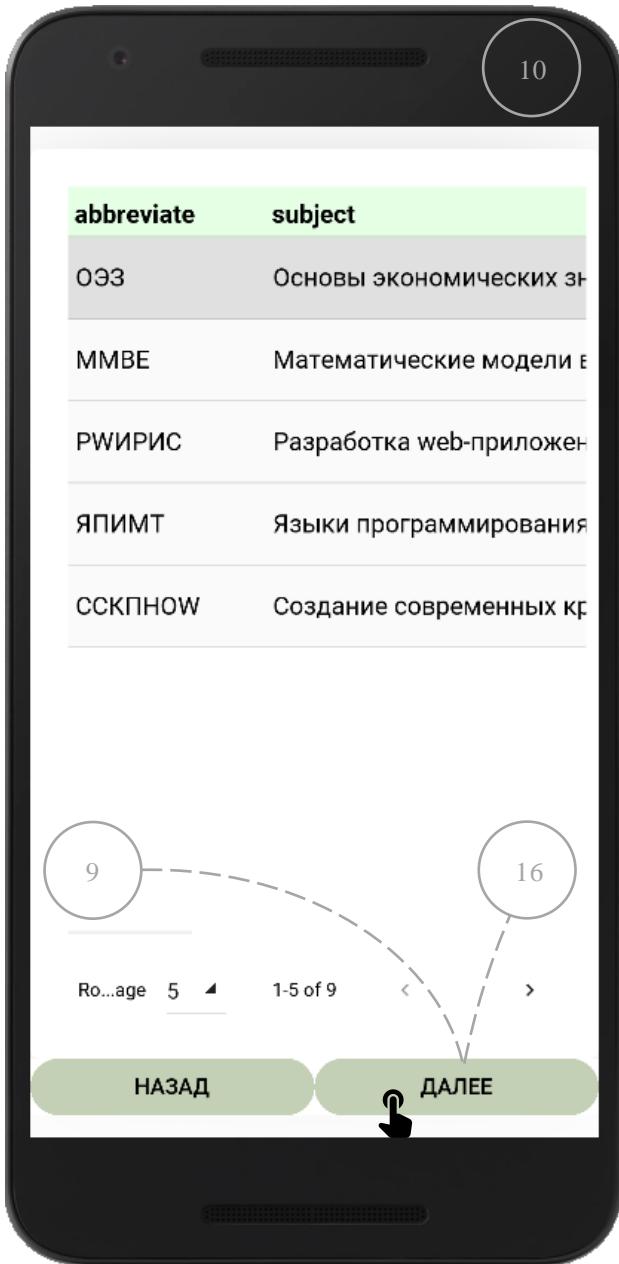


Рисунок 57 – Результат загрузки данных расписания занятий онлайн и переход к загрузке страниц студенческого журнала нажатием кнопки “OK”

Оффлайн вариант: данные загружаются из HTML файла с расписанием занятий (рисунок 58).



Рисунок 58 – Оффлайн версия сбора данных при недоступности данных с сайта (а) и открытие файлового менеджера (б)

Выбор файла для офлайн обработки данных (рисунок 59).

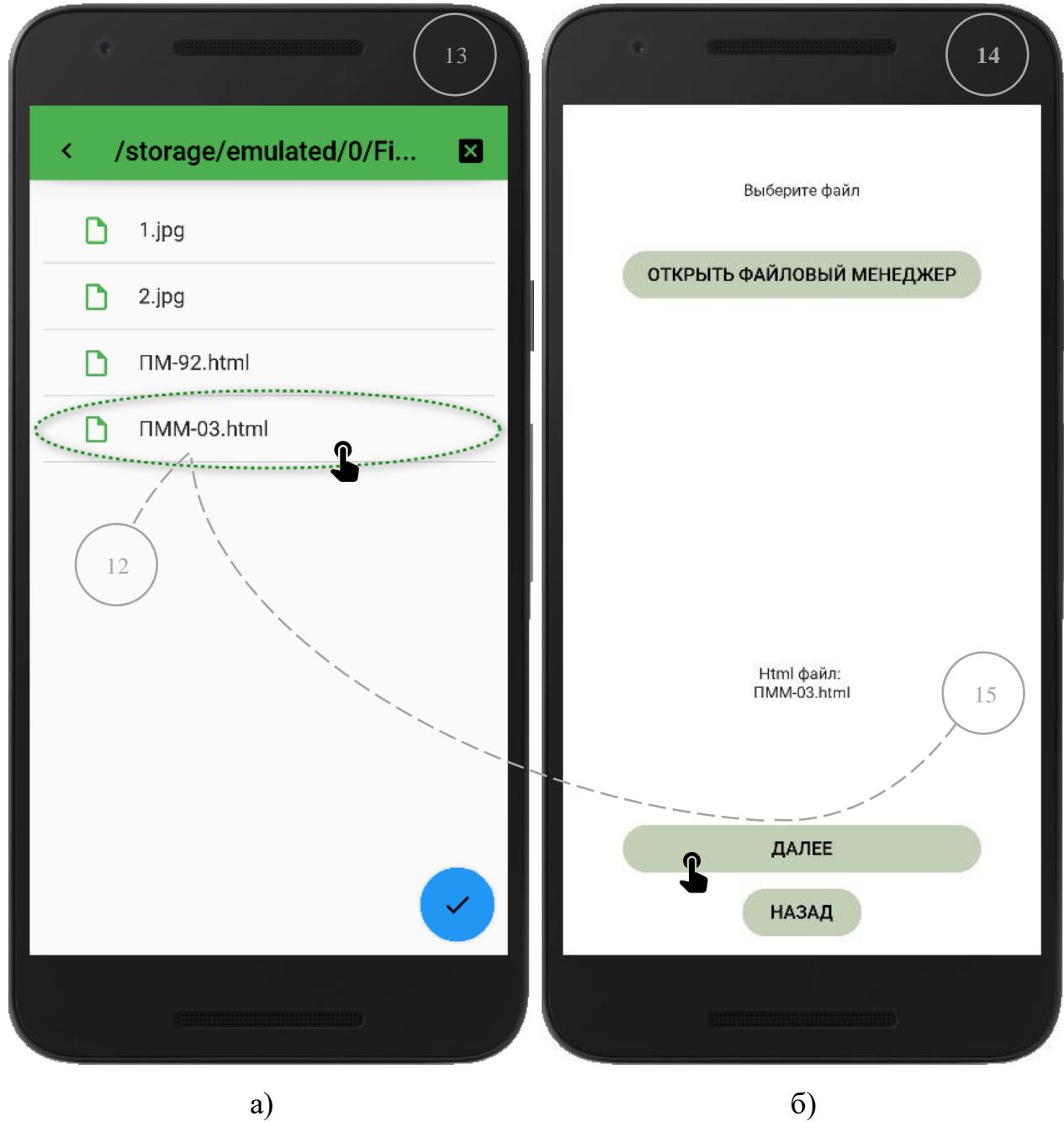


Рисунок 59 – Выбор HTML файла расписания группы

студентов (а) и нажатие кнопки “OK” для загрузки учебных предметов (б)

Результат онлайн загрузки данных (рисунок 60).

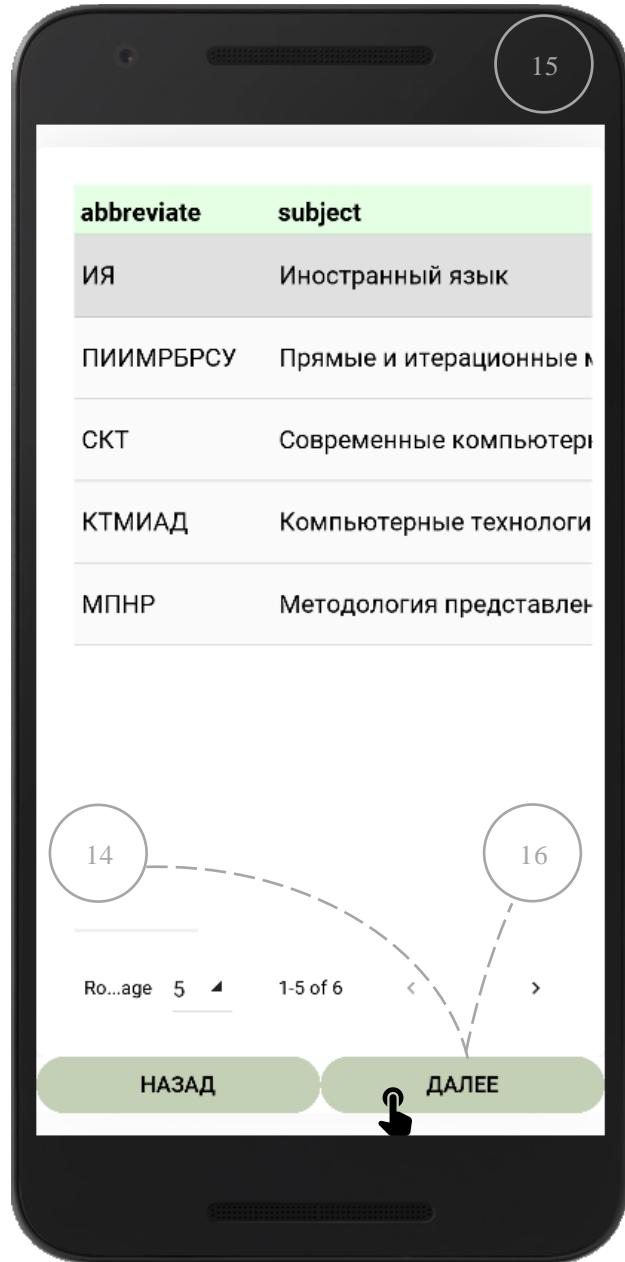


Рисунок 60 – Результат загрузки данных расписания занятий онлайн и переход к загрузке страниц студенческого журнала нажатием кнопки “OK”

На следующем этапе предлагается выбрать 2 страницы разворота студенческого журнала для обработки данных (рисунок 61).

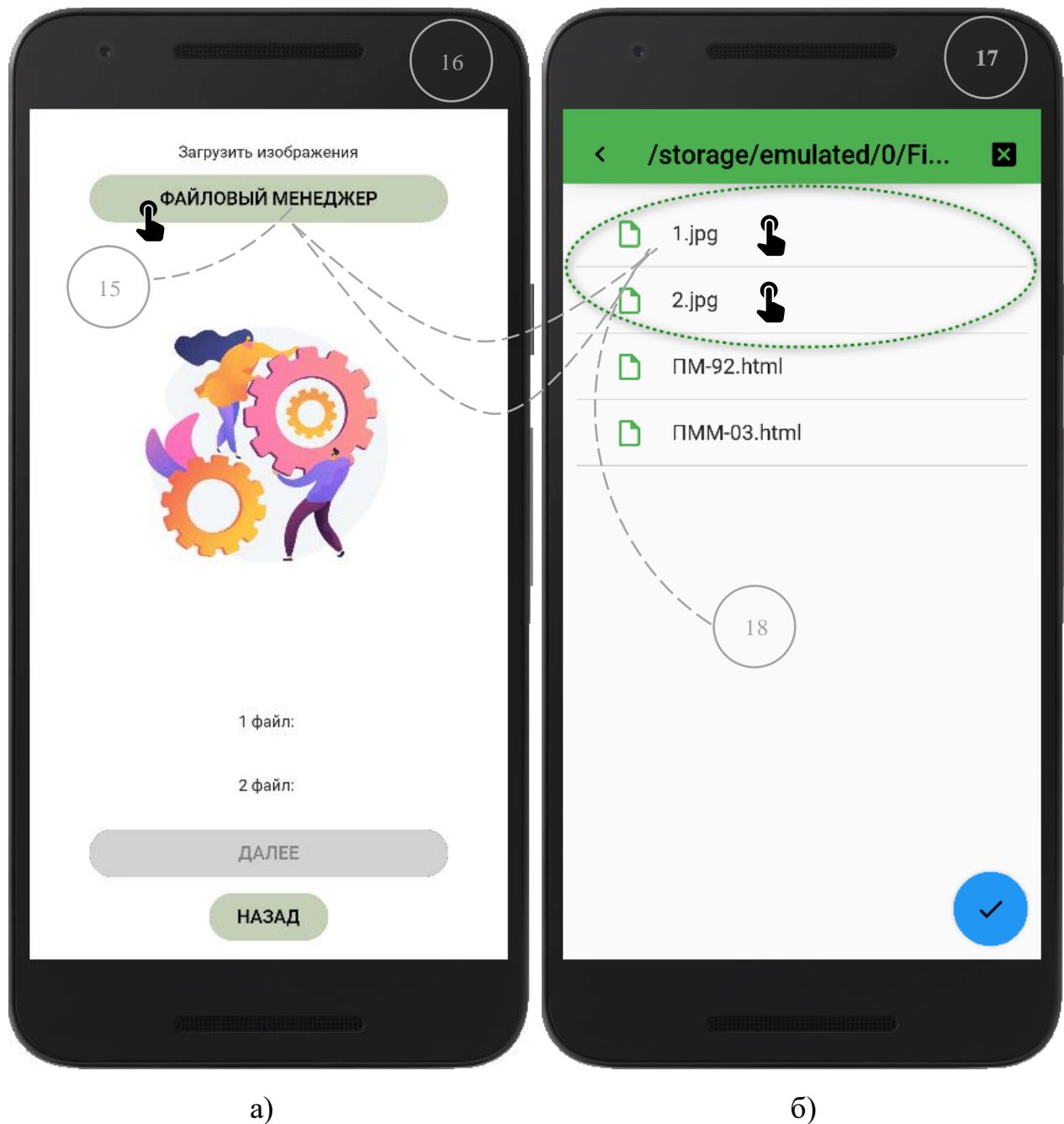


Рисунок 61 – Открытие файлового менеджера (а)  
и выбор изображений студенческого журнала (б)

Обработка файлов студенческого журнала (рисунок 62).



Рисунок 62 – Нажатие кнопки “OK” (а) и обработка изображений (б)

Получившаяся таблица с данными из студенческого журнала (рисунок 63).

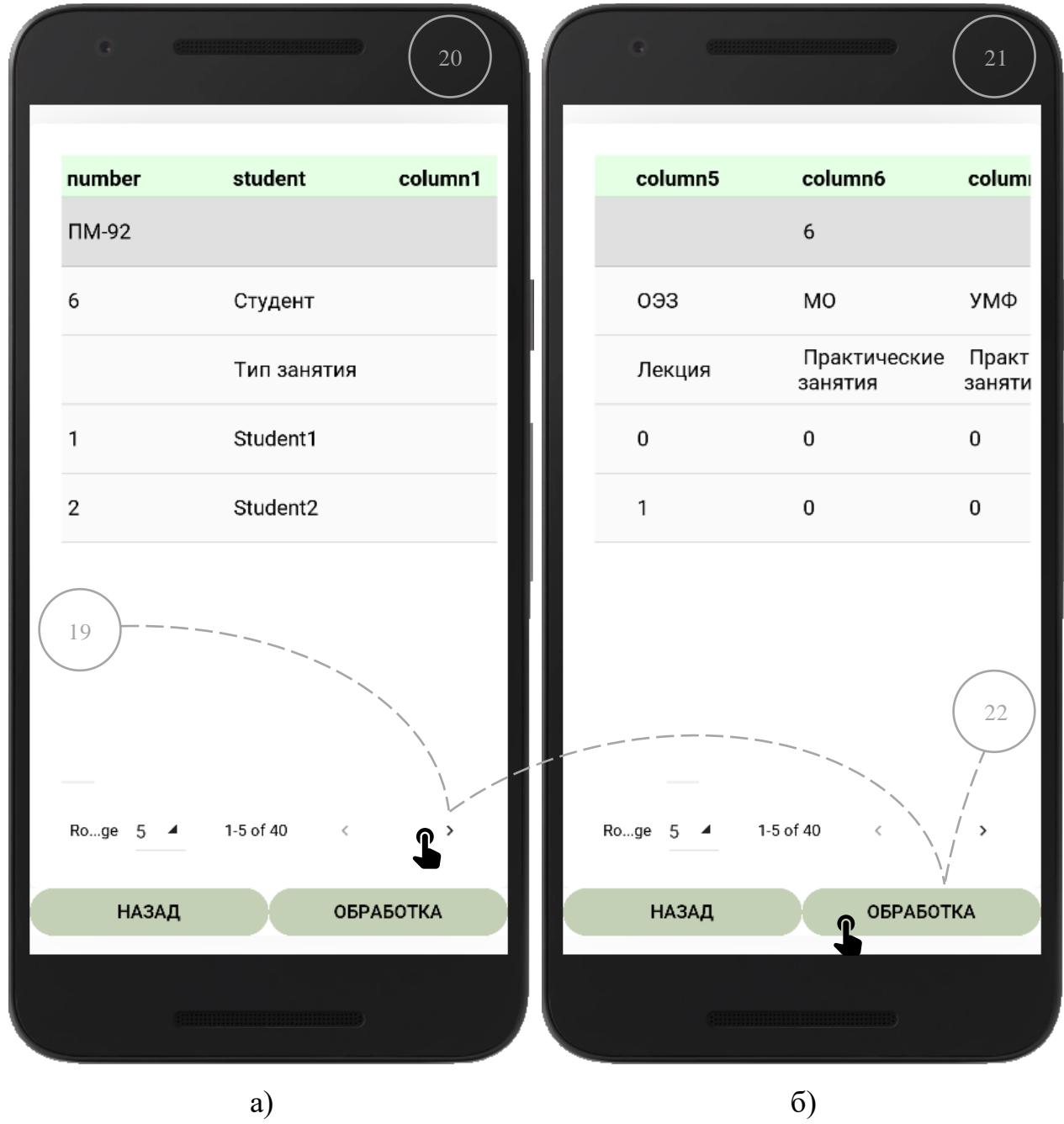


Рисунок 63 – Результаты обработки данных студенческого журнала (а, б) и нажатие кнопки “Collect” для записи в базу данных (б)

Запись в базу данных (рисунок 64).

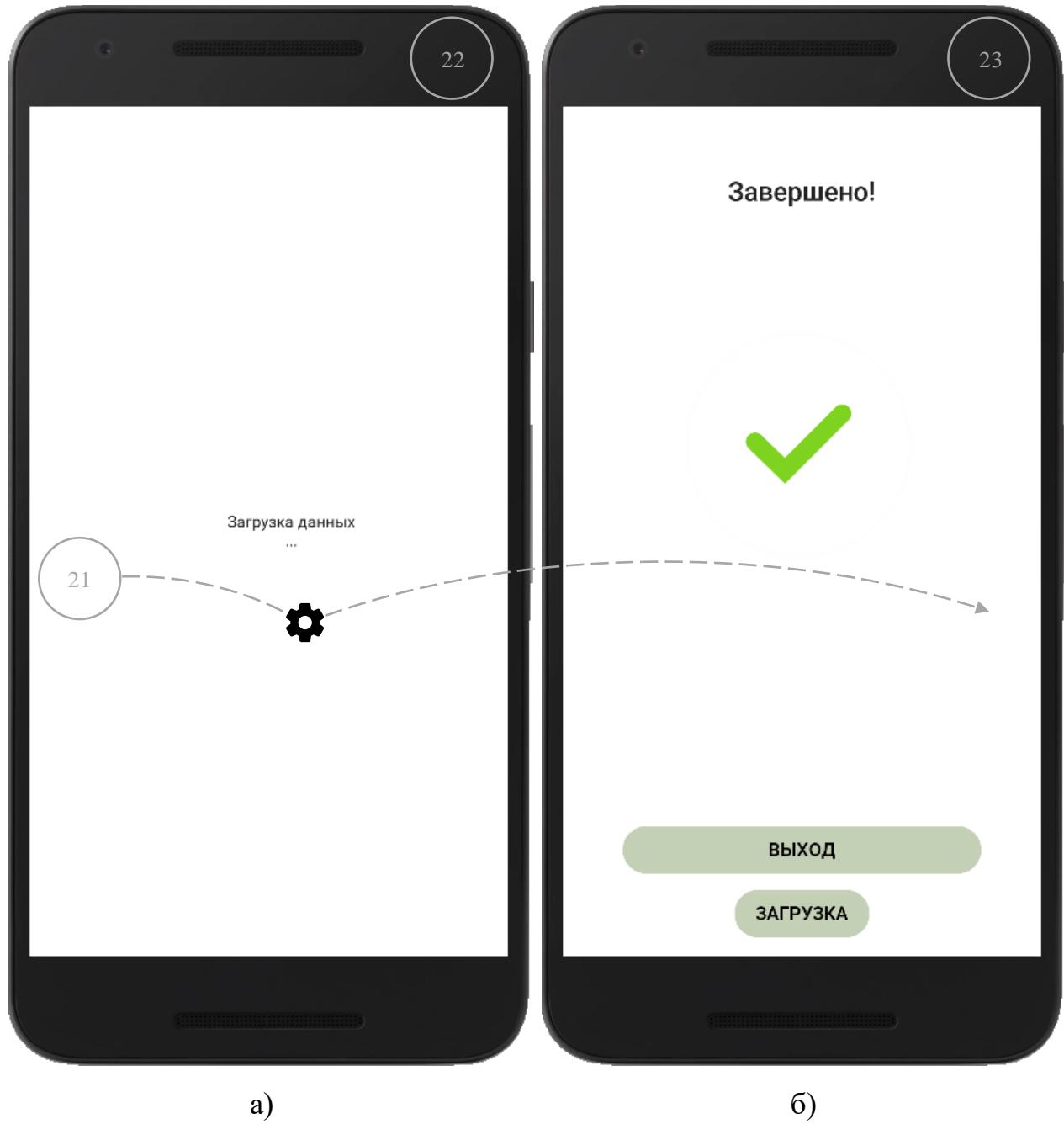


Рисунок 64 – Процесс записи данных в базу данных (а)  
и сообщение об успешном окончании обработки файлов (б)

## 5.2. ВИЗУАЛИЗАЦИЯ ПОЛУЧЕННЫХ ДАННЫХ

Для визуализации данных цифрового следа студентов использовалось программное обеспечение Tableau и фреймворк Streamlit (рисунки 65–84).

### 5.2.1. Дашборд

Дашборд представляет собой набор визуализаций данных (объектов) с автоматическим изменением отображения данных, в зависимости от выбора фильтров.

1 объект (“Groups”): Выбор группы или периода.

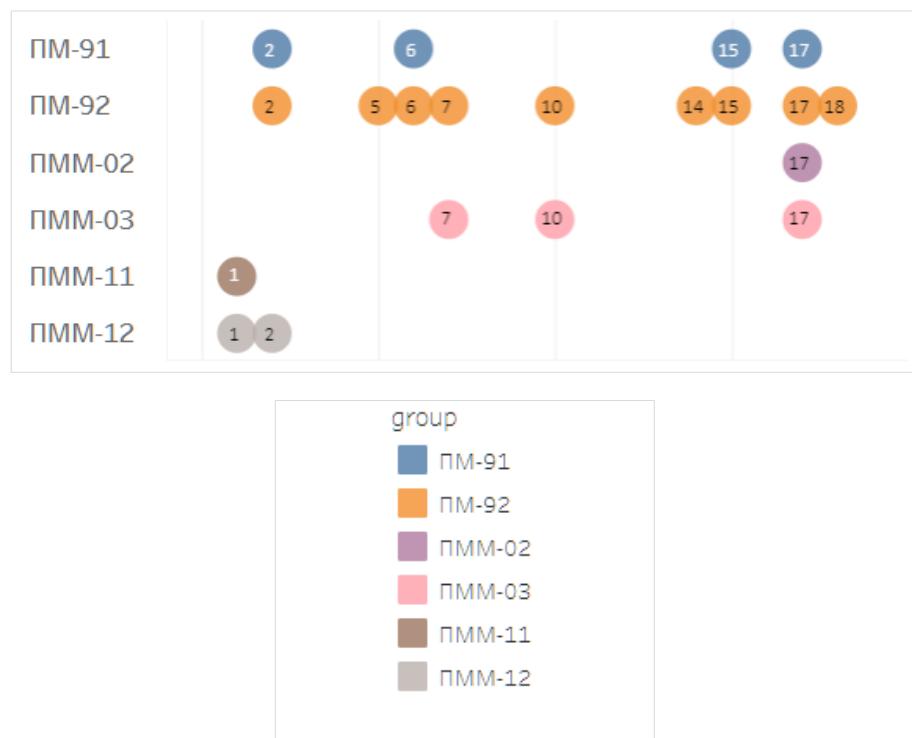


Рисунок 65 – Объект “Groups”

2 объект (“Subjects”): Предметная посещаемость студенческими группами.

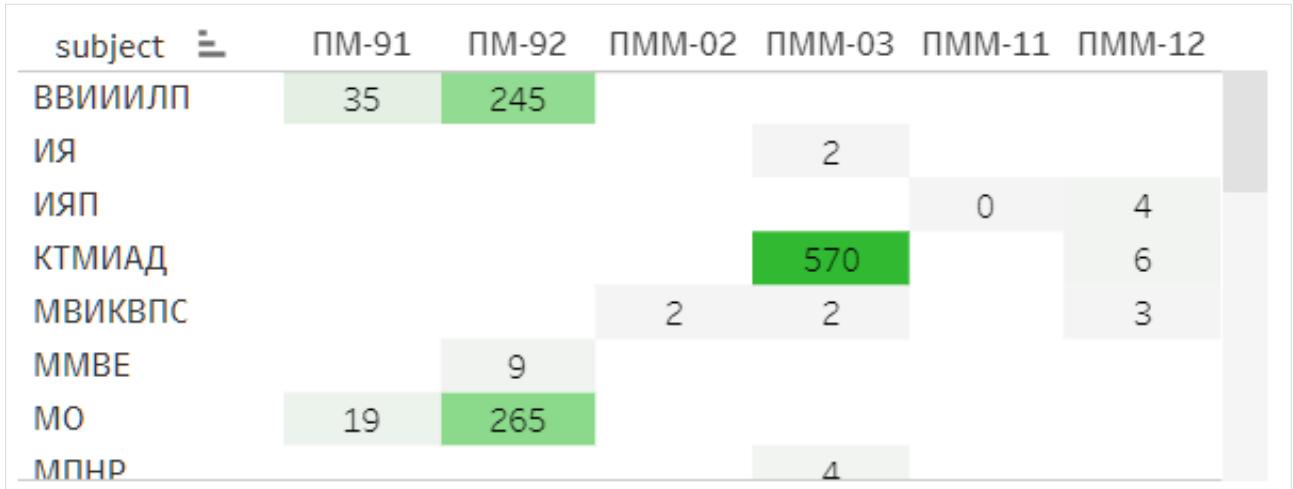


Рисунок 66 – Объект “Subjects”

3 объект (“Students”): Пропуски занятий студентами.

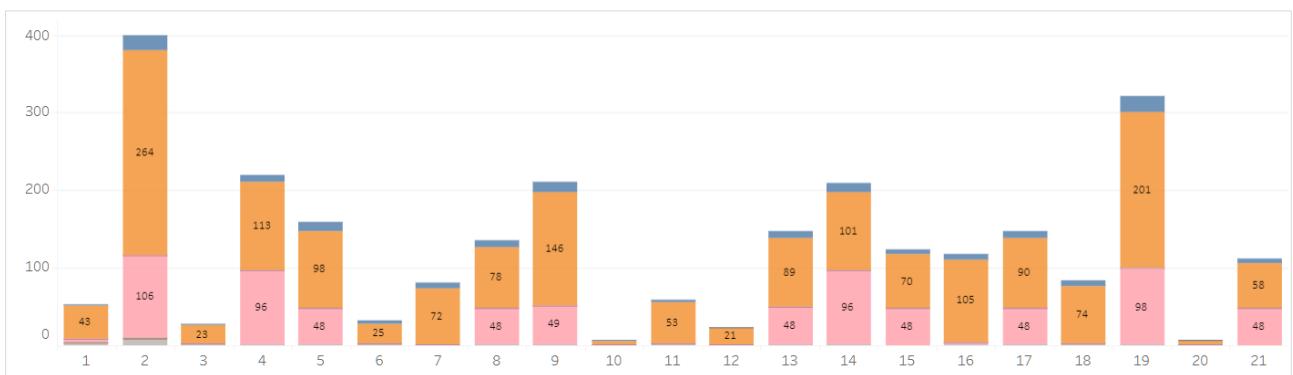


Рисунок 67 – Объект “Students”

4 объект (“Dynamics”): Пропуски занятий.

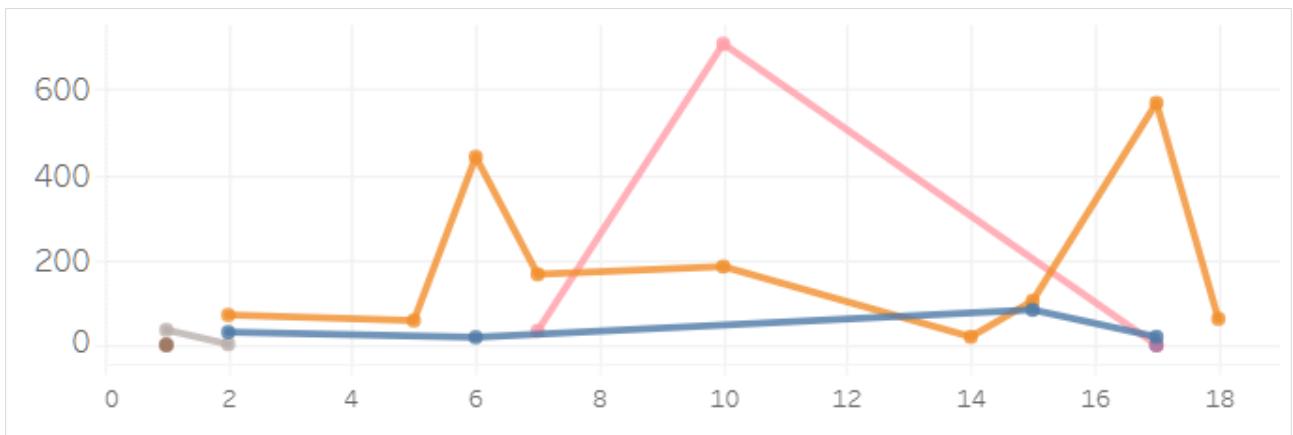


Рисунок 68 – Объект “Dynamics”

5 объект (“Dynamics\_detail”): Подробные пропуски занятий.

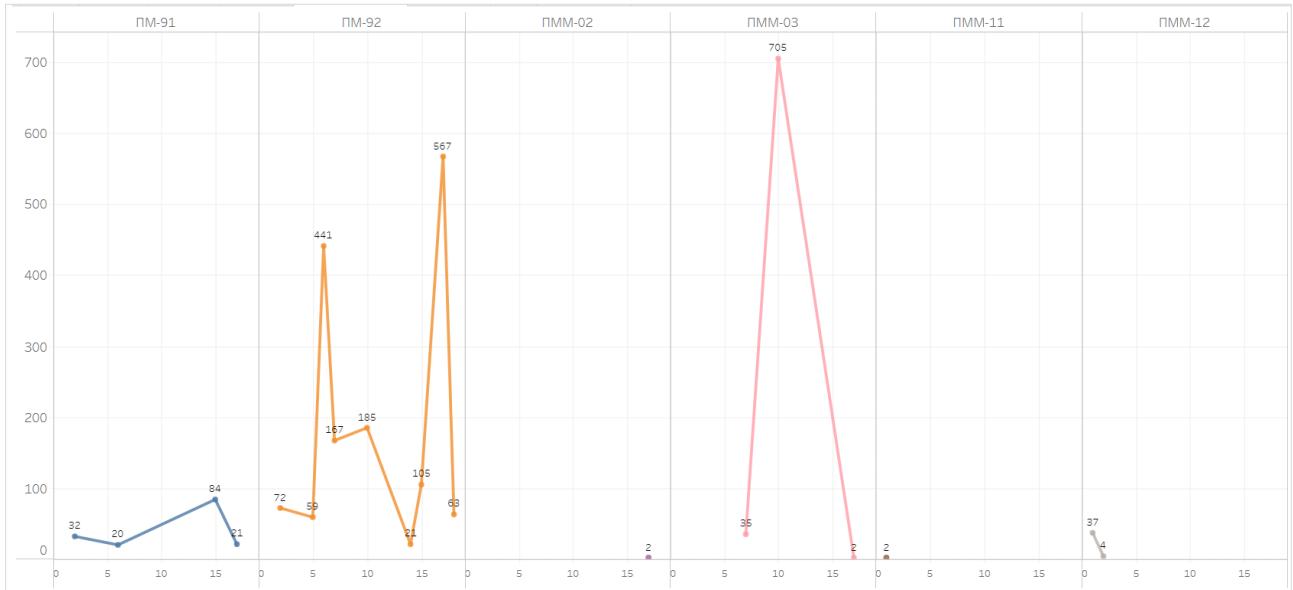


Рисунок 69 – Объект “Dynamics\_detail”

6 объект (“Percentage”): Процентное соотношение пропусков занятий.

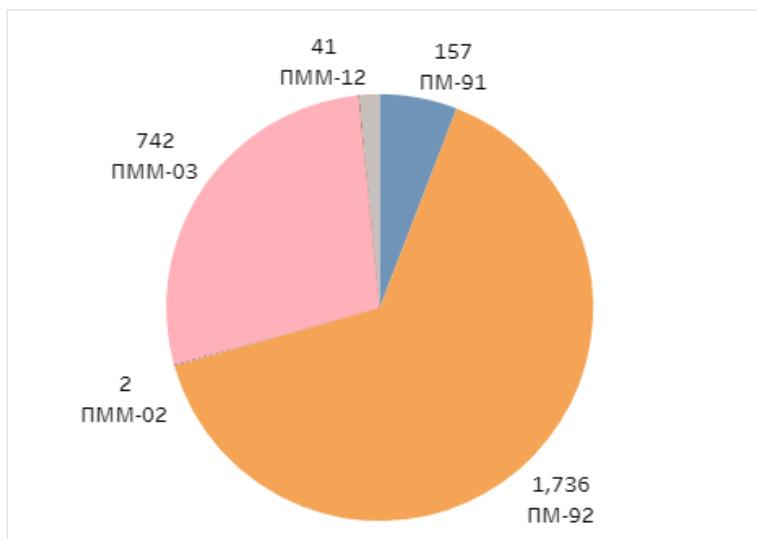


Рисунок 70 – Объект “Percentage”

7 объект (“Total”): Общее количество пропусков.

All missing lectures count 2,680

Рисунок 71 – Объект “Total”

## Дашборд (“DASHBOARD”): Анализ данных цифрового следа студентов.

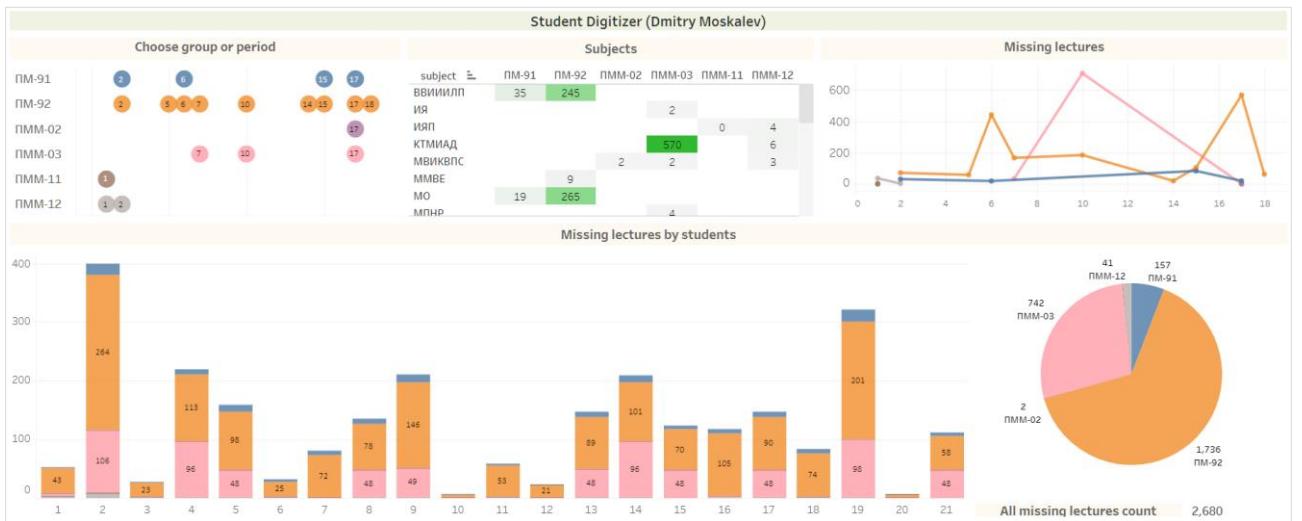


Рисунок 72 – Получившийся дашборд (“DASHBOARD”)

Пример заполнения дашборда для группы ПММ–03.

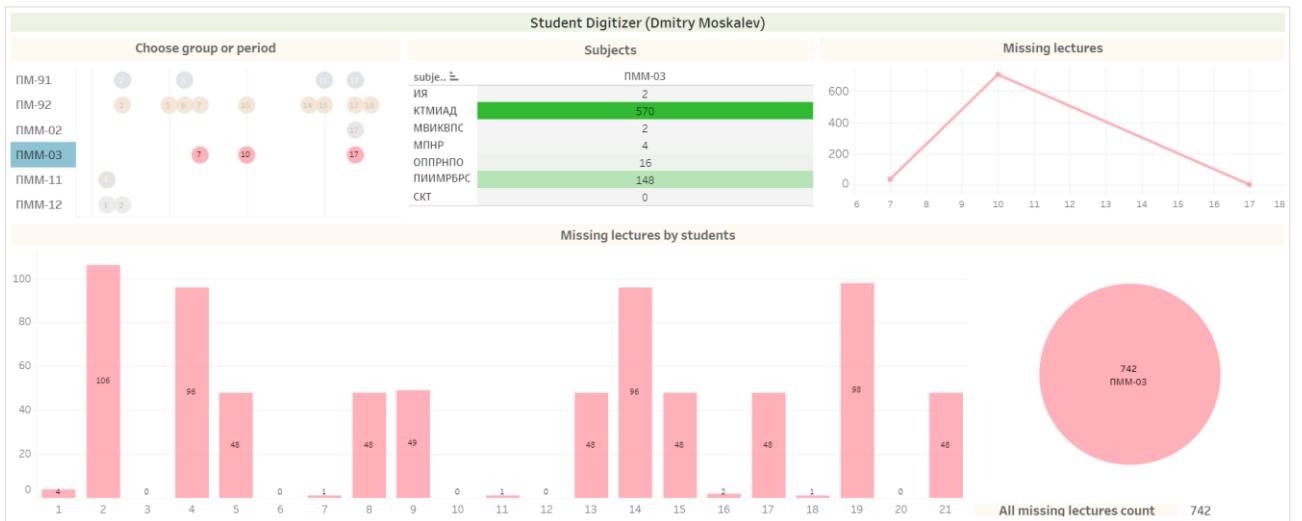


Рисунок 73 – Дашборд для группы ПММ-03

Пример заполнения дашборда для групп ПМ-92 и ПММ-03.

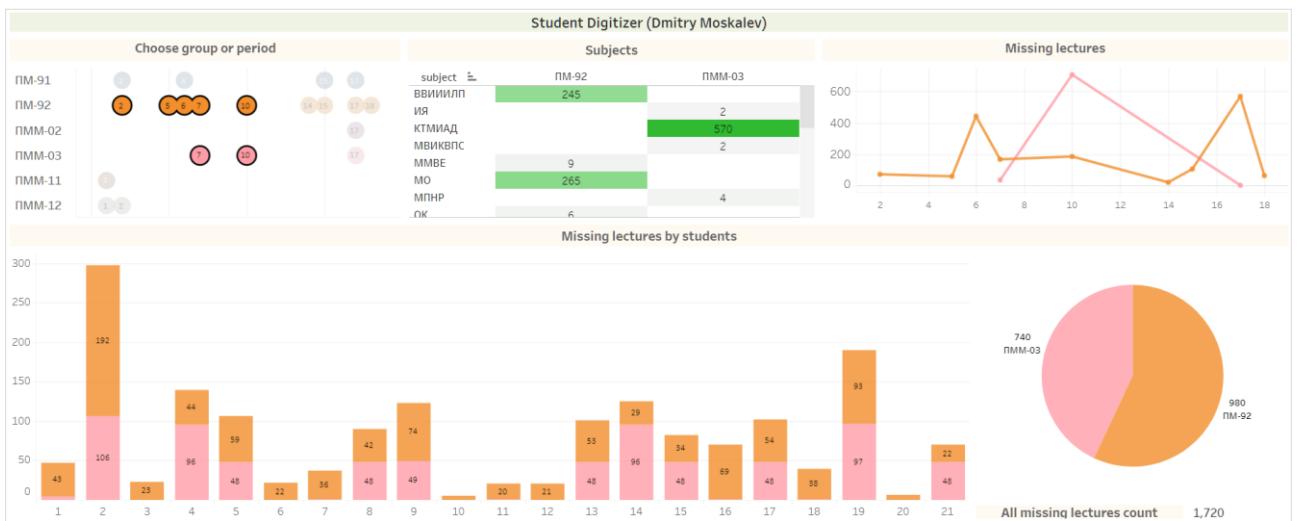


Рисунок 74 – Дашборд для групп ПМ-92 и ПММ-03

### 5.2.2. Веб-сервис

Веб-сервис позволяет быстро получить основную визуализацию данных по выбранной студенческой группе и удобен для просмотра с мобильного телефона.

На рисунке 75 показан логотип веб-сервиса.

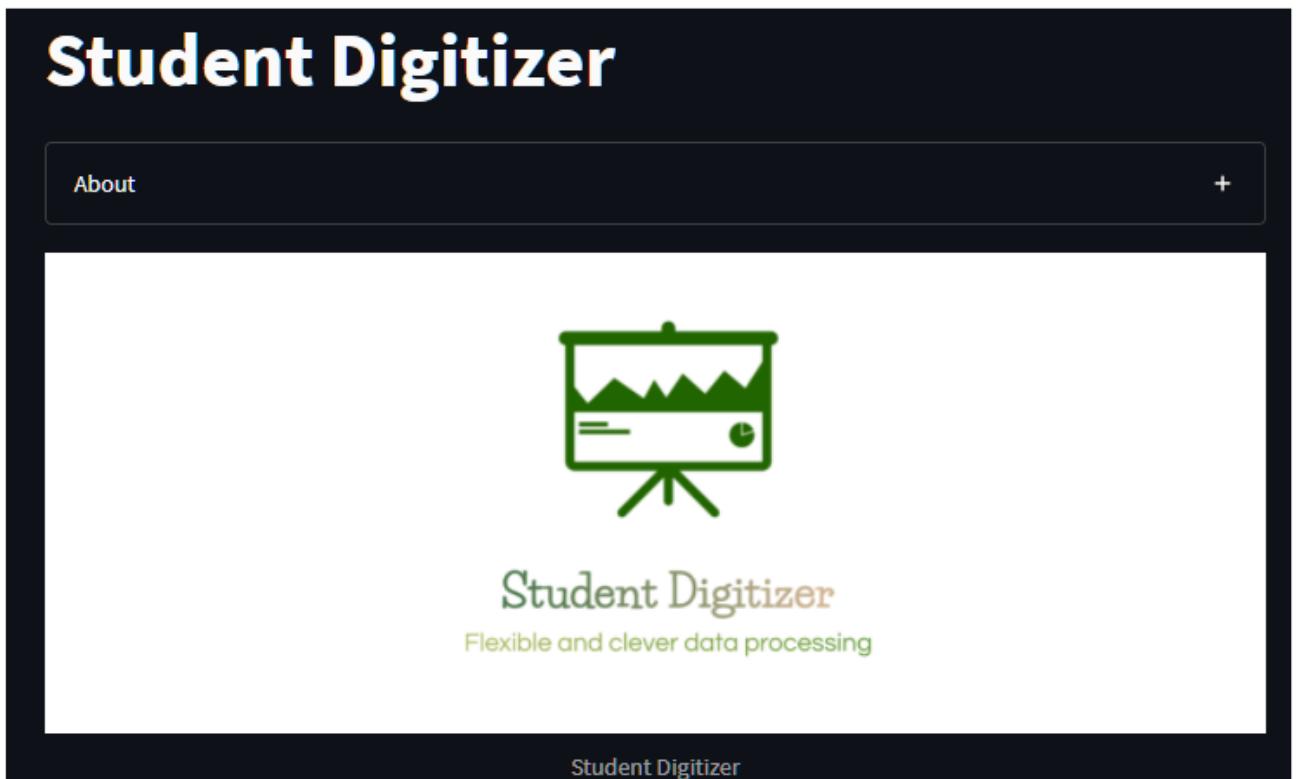


Рисунок 75 – Логотип веб-сервиса

На рисунке 76 показана реализация выбора студенческой группы.

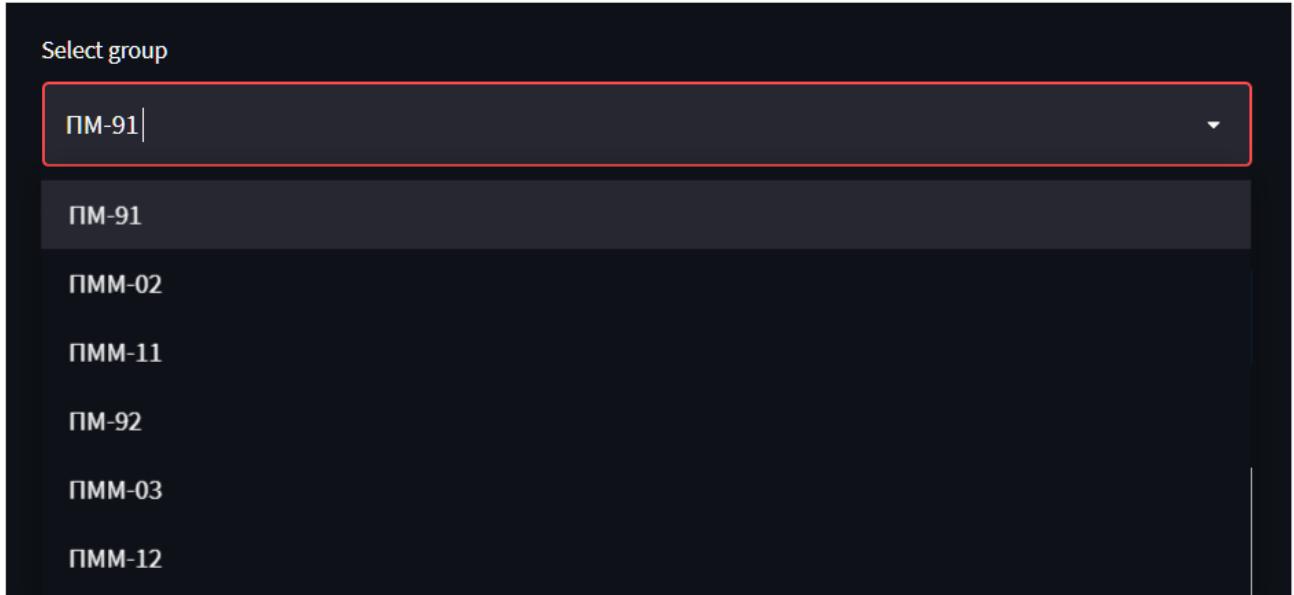


Рисунок 76 – Выбор группы для визуализации данных

Пример данных для ПММ–03 (Рисунки 77–80).



Рисунок 77 – Пример данных для группы ПММ–03

Распределение общего количества пропущенных занятий каждым студентом по всем предметам в неделю (рисунок 78).

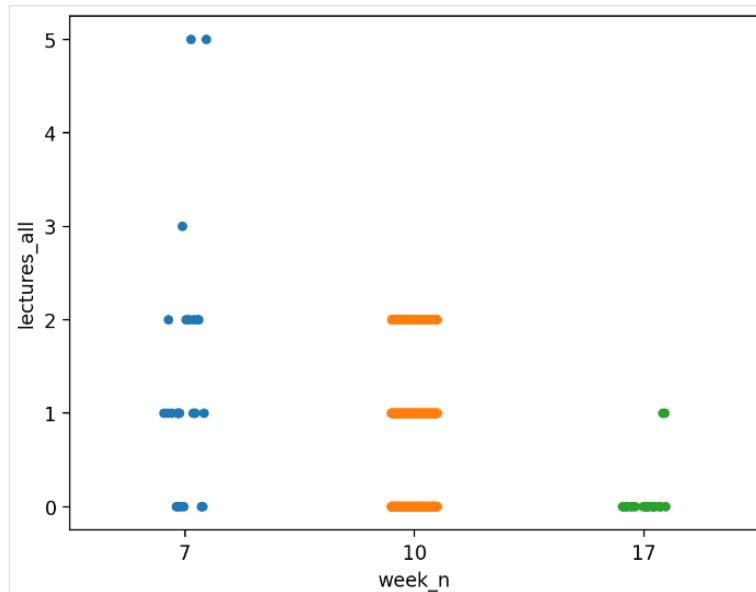


Рисунок 78 – Распределение количества пропущенных занятий студентом

Пропуски каждого из предметов студенческой группой. Отмечены уникальные предметы (рисунок 79).

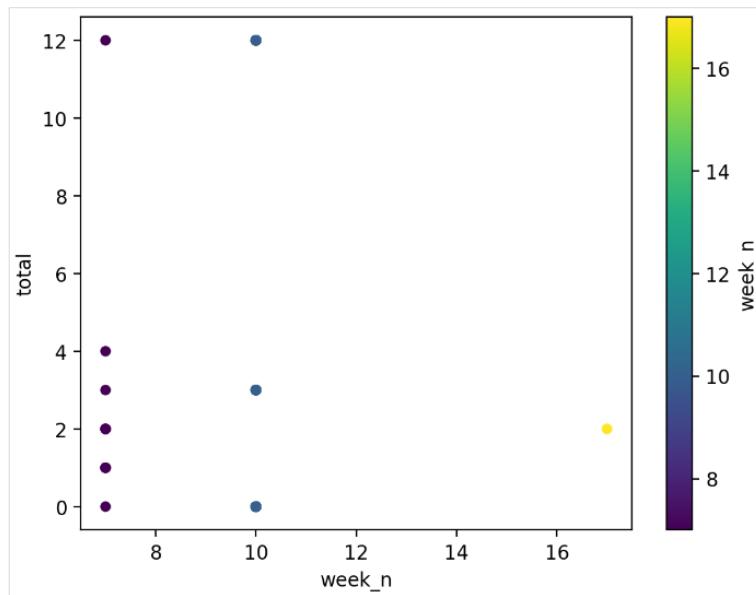


Рисунок 79 – Пропуски каждого из предметов студенческой группой

Процентное соотношение суммарного количества пропущенных занятий студентами и количества пропущенных предметов студентами (рисунок 80).

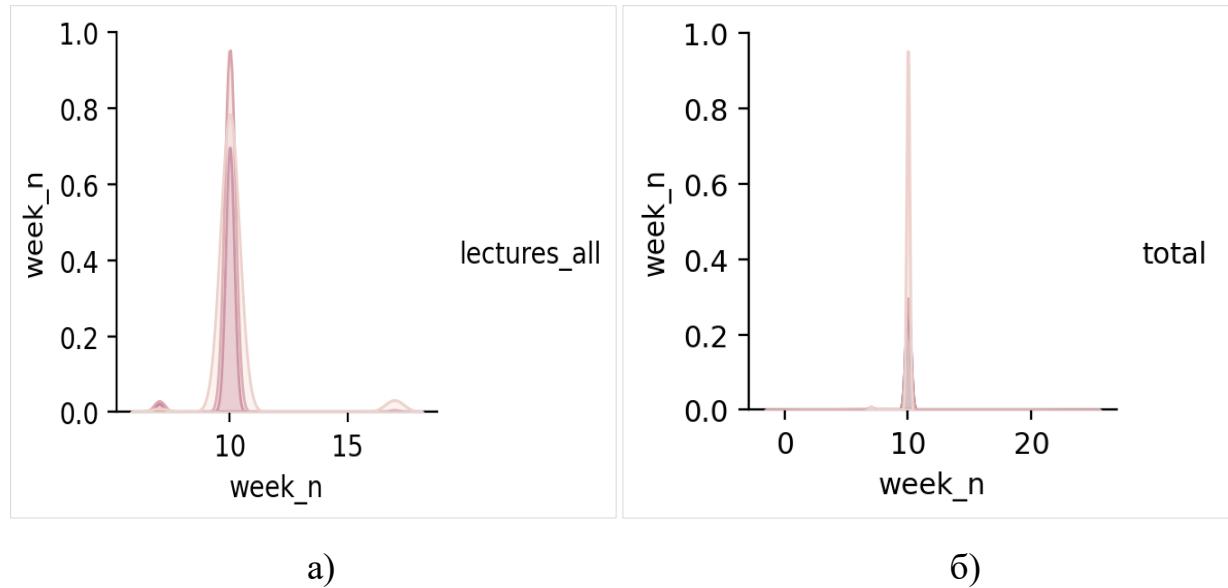


Рисунок 80 – Соотношение пропусков занятий студентами (а) и общего количества пропущенных предметов (б)

Пример данных для ПМ-92 (Рисунки 81–84).

Select group

ПМ-92

Values for ПМ-92: [2, 5, 6, 7, 10, 14, 15, 17, 18]

Students and subjects distributions

Рисунок 81 – Пример данных для группы ПМ-92

Распределение общего количества пропущенных занятий каждым студентом по всем предметам в неделю (рисунок 82).

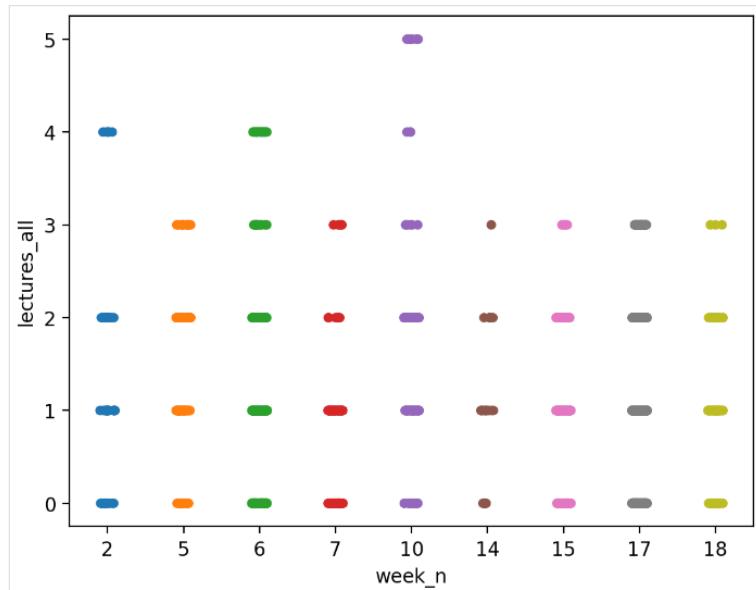


Рисунок 82 – Распределение количества пропущенных занятий студентом

Пропуски каждого из предметов студенческой группой. Отмечены уникальные предметы (рисунок 83).

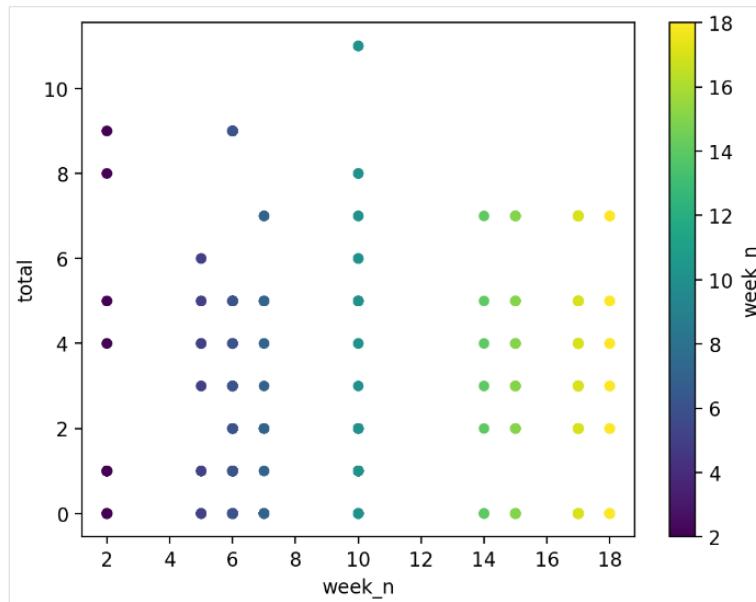


Рисунок 83 – Пропуски каждого из предметов студенческой группой

Процентное соотношение суммарного количества пропущенных занятий студентами и количества пропущенных предметов студентами (рисунок 84).

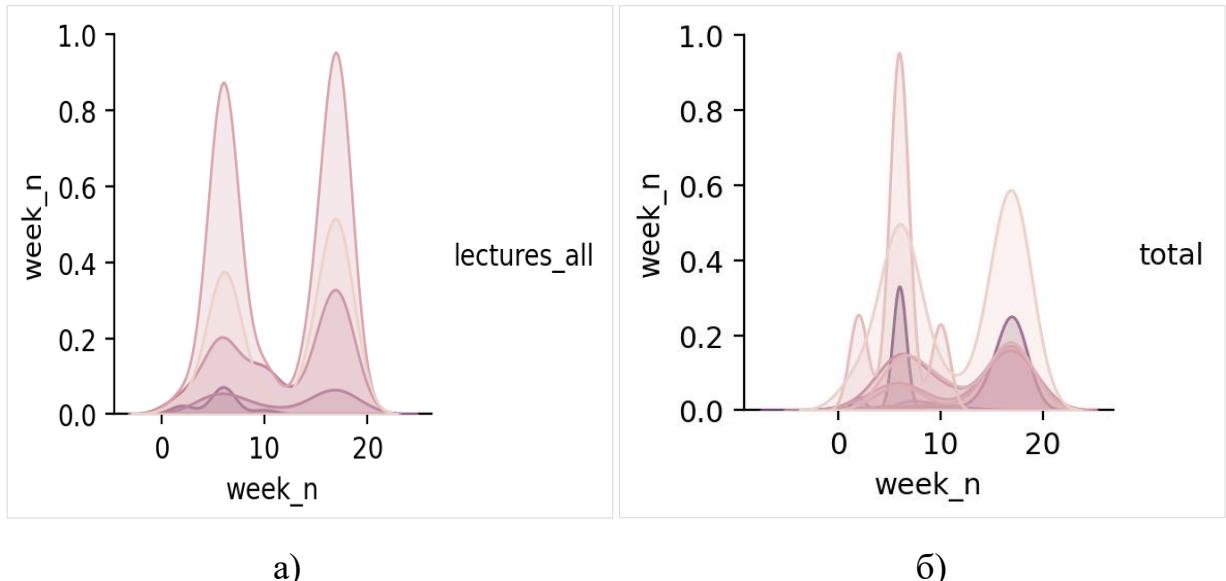


Рисунок 84 – Соотношение пропусков занятий студентами (а)  
и общего количества пропущенных предметов (б)

## **ЗАКЛЮЧЕНИЕ**

В данной магистерской диссертации была разработана информационная система, включающая в себя ETL процессы в мобильном приложении для решения задачи сбора и анализа цифрового следа студентов. Выполнена визуализация полученных результатов и сравнение метрик точности классификации символов в модели данных, сделанной с помощью свёрточных нейронных сетей.

Наиболее универсальным подходом к решению задачи было использование OCR сервиса Nanonets используя технологию Rest API. Реализованные технологии позволяют анализировать входные данные журнала и визуализировать полученные данные в виде адаптивного дашборда и веб-сервиса.

Визуализация результатов работы программы показало, что ETL процессы выполняются и позволяют автоматизировать задачи персонализации учебного процесса как для отдельного студента или предмета, так и для группы в целом. При этом результаты анализа данных после загрузки страниц журнала появляются автоматически и позволяют понять текущую активность по группам и предметам в течение недели, не ожидая окончания семестра или сессии. Таким образом, полученное решение можно использовать для быстрого и эффективного сбора и анализа цифрового следа студентов.

## СПИСОК ЛИТЕРАТУРЫ

1. ДАШБОРД - Краткий словарь информационных технологий: [Электронный ресурс] // Human Technologies, 2022. URL: <https://ht-lab.ru/knowledge/dictionaries/kratkiy-slovar-it/4372/> (дата обращения: 14.01.2022).
2. Классификация: [Электронный ресурс] // MachineLearning, 2011. URL: <http://www.machinelearning.ru/wiki/index.php?title=Классификация> (дата обращения: 27.10.2021).
3. The official home of the Python Programming Language: [Электронный ресурс] // Python Software Foundation, 2022. URL: <https://www.python.org/> (дата обращения: 02.10.2021).
4. Kotlin Programming Language: [Электронный ресурс] // JetBrains, 2022. URL: <https://kotlinlang.org/> (дата обращения: 02.09.2021).
5. Valliappa Lakshmanan, Jordan Tigani. Google BigQuery: The Definitive Guide: Data Warehousing, Analytics, and Machine Learning at Scale. – U.S.: O'Reilly Media, Inc., 2020. – 522 с.
6. Lindy Ryan. Visual Data Storytelling with Tableau. – U.S.: Addison-Wesley Professional, 2018. – 272 с.
7. База рукописных символов русского алфавита – Google Диск: [Электронный ресурс] // Google, 2022. URL: <https://drive.google.com/drive/folders/0B0EQUc5HmgcGS0l2RD1KenlpNnc> (дата обращения: 02.10.2021).
8. Understanding ReLU: The Most Popular Activation Function in 5 Minutes!: [Электронный ресурс] // Towards Data Science, 2022. URL: <https://towardsdatascience.com/understanding-relu-the-most-popular-activation-function-in-5-minutes-459e3a2124f> (дата обращения: 10.12.2021).
9. Softmax Activation Function – How It Actually Works: [Электронный ресурс] // Towards Data Science, 2022. URL: <https://towardsdatascience.com/softmax-activation-function-how-it-actually-works-d292d335bd78> (дата обращения: 10.12.2021).

10. Intelligent document processing with AI: [Электронный ресурс] // Nano Net Technologies Inc., 2021. URL: <https://nanonets.com/> (дата обращения: 02.09.2021).
11. Android's Kotlin-first approach: [Электронный ресурс] // Android Developers, 2022. URL: <https://developer.android.com/kotlin/first> (дата обращения: 10.09.2021).
12. Kivy: Cross-platform Python Framework for NUI Development: [Электронный ресурс] // Kivy, 2022. URL: <https://kivy.org/> (дата обращения: 02.09.2021).
13. Welcome to KivyMD's documentation!: [Электронный ресурс] // KivyMD, 2022. URL: <https://kivymd.readthedocs.io/en/latest/> (дата обращения: 02.09.2021).
14. Student Digitizer: [Электронный ресурс] // Tableau Software, LLC, 2022. URL: [https://public.tableau.com/app/profile/dmitry.moskalev/viz/StudentDigitizer/Dashboard\\_1](https://public.tableau.com/app/profile/dmitry.moskalev/viz/StudentDigitizer/Dashboard_1) (дата обращения: 21.03.2022).
15. Product icons for architectural diagrams: [Электронный ресурс] // Google, 2022. URL: <https://cloud.google.com/icons> (дата обращения: 10.01.2021).
16. Media Kit: [Электронный ресурс] // Tableau Software, LLC, 2022. URL: <https://www.tableau.com/about/media-kit> (дата обращения: 12.10.2021).
17. Brand • Streamlit: [Электронный ресурс] // Streamlit Inc., 2022. URL: <https://streamlit.io/brand> (дата обращения: 12.10.2021).
18. Энциклопедический словарь: [Электронный ресурс] // Академик, 2022. URL: <https://dic.academic.ru/contents.nsf/es/> (дата обращения: 02.09.2021).
19. Welcome To Colaboratory - Google Research: [Электронный ресурс] // Google, 2022. URL: <https://research.google.com/colaboratory/> (дата обращения: 02.09.2021).
20. Material Symbols and Icons - Google Fonts: [Электронный ресурс] // Google, 2022. URL: <https://fonts.google.com/icons> (дата обращения: 02.10.2021).
21. Free Lottie Animation Files, Tools & Plugins – LottieFiles: [Электронный ресурс] // Design Barn Inc., 2022. URL: <https://lottiefiles.com/> (дата обращения: 02.10.2021).

# ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ

## ПРЕДОБРАБОТКИ ДАННЫХ

**File: Processing.ipynb**

```
import os, random, shutil, cv2
from tensorflow.keras.utils import Sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D,
GlobalAveragePooling2D, Dropout
from tensorflow.keras.callbacks import ModelCheckpoint, LearningRateScheduler
from os import listdir
import numpy as np
import tensorflow as tf

# 0 - Б
# 1 - Н
# 2 - О

for i in range(3): # Count of classes
    index_number = i
    data = f'{os.getcwd()}//DATA//{i}'
    input_files = os.listdir(data)
    for index, file in enumerate(input_files):
        try:
            os.rename(os.path.join(data, file), os.path.join(data,
f'{index_number}({index}).png'))
        except FileExistsError:
            pass

# Image channels
for i in range(3): # Count of classes
    index_number = i
    data = f'{os.getcwd()}//DATA//{i}'
    input_files = os.listdir(data)
    for index, file in enumerate(input_files):
        img = cv2.imread(os.path.join(data, file), 0)
        cv2.imwrite(os.path.join(data, file), img)

# Process files to train, test and validation sub-folders
for i in ['TEST', 'TRAIN', 'VALIDATION']:
    try:
        os.mkdir(i)
    except FileExistsError:
        pass

for i in range(3):
    for j in ['TEST', 'TRAIN', 'VALIDATION']:
        try:
            os.mkdir(os.path.join(j, str(i)))
        except FileExistsError:
            pass


def calculate():
    for i in range(3): # Count of classes
        index_number = i
        train = fr'{os.getcwd()}//TRAIN//{i}'
        test = fr'{os.getcwd()}//TEST//{i}'
        validation = fr'{os.getcwd()}//VALIDATION//{i}'
```

```

        onlyfiles = [f for f in os.listdir(train) if
os.path.isfile(os.path.join(train, f))]
        no_of_files = round((len(onlyfiles) / 8)) # 12,5%
        print(f'Class: {i}, Train: {len(onlyfiles)}, Moving: {no_of_files}')

def to_test():
    for count, i in enumerate(range(no_of_files), 1):
        files = [filenames for (filenames) in os.listdir(train)]
        random_file = random.choice(files)
        try:
            shutil.move(os.path.join(train, random_file), test)
        except Exception:
            pass

    files_test = os.listdir(test)
    for index, file in enumerate(files_test):
        try:
            os.rename(os.path.join(test, file), os.path.join(test,
f'{index_number}({index}).png'))
        except FileExistsError:
            pass
    return count, len(onlyfiles)

to_test()

def to_validation():
    for count, i in enumerate(range(no_of_files), 1):
        files = [filenames for (filenames) in os.listdir(train)]
        random_file = random.choice(files)
        try:
            shutil.move(os.path.join(train, random_file), validation)
        except Exception:
            pass

    files_validation = os.listdir(validation)
    for index, file in enumerate(files_validation):
        try:
            os.rename(os.path.join(validation, file),
os.path.join(validation, f'{index_number}({index}).png'))
        except FileExistsError:
            pass

    files_train = os.listdir(train)
    for index, file in enumerate(files_train):
        try:
            os.rename(os.path.join(train, file), os.path.join(train,
f'{index_number}({index}).png'))
        except FileExistsError:
            pass
    return count, len(onlyfiles)

to_validation()

calculate()

for i in range(3): # Count of classes
    index_number = i
    for j in ['TRAIN', 'TEST', 'VALIDATION']:
        data = fr'{os.getcwd()}/{j}/{i}'
        input_files = os.listdir(data)
        for index, file in enumerate(input_files):

```

```

        try:
            os.rename(os.path.join(data, file), os.path.join(data,
f'{index_number}({index}).png'))
        except FileExistsError:
            pass

# Count minimum of files in sub-folder
def count_minimum_files():
    onlyfiles_train, onlyfiles_test, min_files_validate = 0, 0, 0
    min_files_train, min_files_test, min_files_validate = [], [], []
    for count, i in enumerate(range(3), 1): # Count of classes
        train = fr'{os.getcwd()}/TRAIN//{i}'
        test = fr'{os.getcwd()}/TEST//{i}'
        validate = fr'{os.getcwd()}/VALIDATION//{i}'
        try:
            onlyfiles_train = [f for f in os.listdir(train) if
os.path.isfile(os.path.join(train, f))]
        except Exception:
            pass

        try:
            onlyfiles_test = [f for f in os.listdir(test) if
os.path.isfile(os.path.join(test, f))]
        except Exception:
            pass

        try:
            onlyfiles_validate = [f for f in os.listdir(validate) if
os.path.isfile(os.path.join(validate, f))]
        except Exception:
            pass

        try:
            min_files_train.append(len(onlyfiles_train))
        except Exception:
            min_files_train.append(0)

        try:
            min_files_test.append(len(onlyfiles_test))
        except Exception:
            min_files_test.append(0)

        try:
            min_files_validate.append(len(onlyfiles_validate))
        except Exception:
            min_files_validate.append(0)
    print(f'Train: {min(min_files_train)}, Test: {min(min_files_test)},\nValidation: {min(min_files_validate)}')
    print()
    for count, (i, j, k) in enumerate(zip(min_files_train, min_files_test,
min_files_validate), 0):
        print(f'Class: {count}, Trains: {i}, Tests: {j}, Validations: {k}')
    return i, j, k

COUNT_TRAIN_IMAGE, COUNT_VALID_IMAGE, COUNT_TEST_IMAGE = count_minimum_files()

directory_train_data_base = 'TRAIN'
directory_valid_data_base = 'VALIDATION'
directory_test_data_base = 'TEST'

COUNT_CLASS = 3 # Count of classes
COUNT_EPOCH = 20 # Epoch count

```

```

class GeneratorImage(Sequence):
    """Class for generating packages of test images"""

    def __init__(self):
        self.Y = np.eye(COUNT_CLASS)

    def __len__(self):
        return COUNT_TRAIN_IMAGE

    def __getitem__(self, index):
        index += 1
        list_array_images = []
        list_file =.listdir(directory_train_data_base)
        for file in list_file:
            im =
        cv2.imread(f'{directory_train_data_base}\\{file}\\{file}({index}).png', 0)
            im = np.float32(im)
            im /= 255
            list_array_images.append(np.array(im))

        return np.array(list_array_images, dtype=np.float32), self.Y

class GeneratorValidImage(Sequence):
    """Class for generating packages of test images"""

    def __init__(self):
        self.Y = np.eye(COUNT_CLASS)

    def __len__(self):
        return COUNT_VALID_IMAGE

    def __getitem__(self, index):
        index += 1
        list_array_images = []
        list_file =.listdir(directory_valid_data_base)
        for file in list_file:
            im =
        cv2.imread(f'{directory_valid_data_base}\\{file}\\{file}({index}).png', 0)
            im = np.float32(im)
            im /= 255
            list_array_images.append(np.array(im))

        return np.array(list_array_images, dtype=np.float32), self.Y

class GeneratorTestImage(Sequence):
    """Class for generating packages of test images"""

    def __init__(self):
        self.Y = np.eye(COUNT_CLASS)

    def __len__(self):
        return COUNT_TEST_IMAGE

    def __getitem__(self, index):
        index += 1
        list_array_images = []
        list_file =.listdir(directory_test_data_base)
        for file in list_file:
            im =
        cv2.imread(f'{directory_test_data_base}\\{file}\\{file}({index}).png', 0)

```

```

        im = np.float32(im)
        im /= 255
        list_array_images.append(np.array(im))

    return np.array(list_array_images, dtype=np.float32), self.Y

def create_model():
    """Function for creating a model"""

    # Creating sequential model
    new_model = Sequential()

    # Add layers to the model
    new_model.add(Conv2D(32, kernel_size=2, strides=2, activation='relu',
input_shape=(145, 145, 1)))
    new_model.add(Conv2D(128, kernel_size=2, strides=2, activation='relu'))

    new_model.add(MaxPool2D((2, 2)))

    new_model.add(Conv2D(512, kernel_size=3, activation='relu'))

    new_model.add(Flatten())
    new_model.add(Dense(3, activation='softmax')) # Count of classes

    new_model.summary()

    new_model.compile('adam', 'categorical_crossentropy', metrics=['accuracy'])

    generator = GeneratorImage()

    generator_valid = GeneratorValidImage()

    new_model.fit(generator, epochs=COUNT_EPOCH,
validation_data=generator_valid, steps_per_epoch=COUNT_TRAIN_IMAGE,
shuffle=True,
callbacks=[ModelCheckpoint('MODELS\\model_example.h5', save_best_only=True)])

    return new_model

# Creating model
new_model = create_model()

tf.keras.utils.plot_model(
    new_model,
    to_file="model.png",
    show_shapes=True,
    show_dtype=False,
    show_layer_names=True,
    rankdir="TB",
    expand_nested=False,
    dpi=96,
    layer_range=None,
    show_layer_activations=True,
)

# Export model
keras_file = 'MODELS\\model.h5'
tf.keras.models.save_model(model, keras_file)
model = tf.keras.models.load_model('MODELS\\model.h5')
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.experimental_new_converter = True

```

```
tflite_model = converter.convert()  
open("MODELS\\converted_model.tflite", "wb").write(tflite_model)
```

# ПРИЛОЖЕНИЕ Б. ТЕКСТ ПРОГРАММЫ КЛАССИФИКАЦИИ ДАННЫХ

*File: Classifier.ipynb*

```
import requests
import json

API_KEY = "****"
model_id = "****"
url = f'***/{model_id}/**'
image = "****"

data = {'file': open(image, 'rb')}
response = requests.post(url, auth=requests.auth.HTTPBasicAuth(API_KEY, ' '),
files=data)
data = json.loads(response.text)

for i in data['result']:
    for j in i['prediction']:
        for k in j['cells']:
            if k['score'] >= 0 and not k['text'].isdigit():
                print(k['text'], k['score'])
```

# ПРИЛОЖЕНИЕ В. ТЕКСТ ПРИЛОЖЕНИЯ

## SYMBOL CLASSIFIER

### File: MainActivity.kt

```
/* The fundamental package from The TensorFlow Authors */
package org.tensorflow.lite.codelabs.digitclassifier
import android.annotation.SuppressLint
import android.graphics.Color
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.util.Log
import android.view.MotionEvent
import android.widget.Button
import android.widget.TextView
import com.divyanshu.draw.widget.DrawView

class MainActivity : AppCompatActivity() {
    private var drawView: DrawView? = null
    private var clearButton: Button? = null
    private var predictedTextView: TextView? = null
    private var symbolClassifier = SymbolDigitizer(this)

    @SuppressLint("ClickableViewAccessibility")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Setup view instances.
        drawView = findViewById(R.id.draw_view)
        drawView?.setStrokeWidth(70.0f)
        drawView?.setColor(Color.BLACK)
        drawView?.setBackgroundColor(Color.WHITE)
        clearButton = findViewById(R.id.clear_button)
        predictedTextView = findViewById(R.id.predicted_text)
        // Setup clear drawing button.
        clearButton?.setOnClickListener {
            drawView?.clearCanvas()
            predictedTextView?.text = getString(R.string.prediction_text_placeholder)
        }

        // Setup classification trigger so that it classify after every stroke drew.
        drawView?.setOnTouchListener { _, event ->
            drawView?.onTouchEvent(event)
            // Then if user finished a touch event, run classification
            if (event.action == MotionEvent.ACTION_UP) {
                classifyDrawing()
            }
            true
        }

        // Setup symbol classifier.
        symbolClassifier
            .initialize()
            .addOnFailureListener { e -> Log.e(TAG, "Error to setting up symbol
classifier.", e) }
    }

    override fun onDestroy() {
        symbolClassifier.close()
```

```

        super.
    onDestroy()
}

private fun classifyDrawing() {
    val bitmap = drawView?.getBitmap()
    if ((bitmap != null) && (symbolClassifier.isInitialized)) {
        symbolClassifier
            .classifyAsync(bitmap)
            .addOnSuccessListener { resultText -> predictedTextView?.text =
resultText }
            .addOnFailureListener { e ->
                predictedTextView?.text = getString(
                    R.string.classification_error_message,
                    e.localizedMessage
                )
                Log.e(TAG, "Error in classifying drawing.", e)
            }
    }
}

companion object {
    private const val TAG = "MainActivity"
}
}

```

### **File: SymbolClassifier.kt**

```

/* The fundamental package from The TensorFlow Authors */
package org.tensorflow.lite.codelabs.digitclassifier
import android.content.Context
import android.content.res.AssetManager
import android.graphics.Bitmap
import android.util.Log
import com.google.android.gms.tasks.Task
import com.google.android.gms.tasks.TaskCompletionSource
import java.io.FileInputStream
import java.io.IOException
import java.nio.ByteBuffer
import java.nio.ByteOrder
import java.nio.channels.FileChannel
import java.util.concurrent.ExecutorService
import java.util.concurrent.Executors
import org.tensorflow.lite.Interpreter

class SymbolDigitizer(private val context: Context) {
    private var interpreter: Interpreter? = null
    var isInitialized = false
        private set

    private val executorService: ExecutorService = Executors.newCachedThreadPool()
    private var inputImageWidth: Int = 0
    private var inputImageHeight: Int = 0
    private var modelInputSize: Int = 0

    fun initialize(): Task<Void> {
        val task = TaskCompletionSource<Void>()
        executorService.execute {
            try {
                initializeInterpreter()
                task.setResult(null)
            } catch (e: IOException) {
                task.setException(e)
            }
        }
    }
}

```

```

        }
        return task.task
    }

    @Throws(IOException::class)
    private fun initializeInterpreter() {
        val assetManager = context.assets
        val model = loadModelFile(assetManager, "converted_model.tflite") // Load
the TF Lite model from asset folder
        val options = Interpreter.Options()
        options.setUseNNAPI(true)
        val interpreter = Interpreter(model, options)
        // Read input shape from model file.
        val inputShape = interpreter.getInputTensor(0).shape()
        inputImageWidth = inputShape[1]
        inputImageHeight = inputShape[2]
        modelInputSize = FLOAT_TYPE_SIZE * inputImageWidth *
            inputImageHeight * PIXEL_SIZE
        // Finish interpreter initialization.
        this.interpreter = interpreter
        isInitialized = true
        Log.d(TAG, "Initialized TFLite interpreter.")
    }

    @Throws(IOException::class)
    private fun loadModelFile(assetManager: AssetManager, filename: String): ByteBuffer {
        val fileDescriptor = assetManager.openFd(filename)
        val inputStream = FileInputStream(fileDescriptor.fileDescriptor)
        val fileChannel = inputStream.channel
        val startOffset = fileDescriptor.startOffset
        val declaredLength = fileDescriptor.declaredLength
        return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset,
declaredLength)
    }

    private fun classify(bitmap: Bitmap): String {
        check(isInitialized) { "TF Lite Interpreter is not initialized yet." }

        // Pre-processing: resize the input image to match the model input shape.
        val resizedImage = Bitmap.createScaledBitmap(
            bitmap,
            inputImageWidth,
            inputImageHeight,
            true
        )
        val byteBuffer = convertBitmapToByteBuffer(resizedImage)
        // Define an array to store the model output.
        val output = Array(1) { FloatArray(OUTPUT_CLASSES_COUNT) }
        // Run inference with the input data.
        interpreter?.run(byteBuffer, output)
        // Post-processing: find the digit that has the highest probability like
string.
        val result = output[0]
        var class_output = "Nothing"
        val maxIndex = result.indices.maxByOrNull { result[it] } ?: -1

        if (maxIndex == 0 && result[maxIndex].toDouble() >= 0.75)
            class_output = "B"
        if (maxIndex == 1 && result[maxIndex].toDouble() >= 0.75)
            class_output = "H"
        if (maxIndex == 2 && result[maxIndex].toDouble() >= 0.75)
            class_output = "O"
    }
}

```

```

    val resultString =
        "Result: %s\nPrecision: %2f"
        .format(class_output, result[maxIndex])

    return resultString
}

fun classifyAsync(bitmap: Bitmap): Task<String> {
    val task = TaskCompletionSource<String>()
    executorService.execute {
        val result = classify(bitmap)
        task.setResult(result)
    }
    return task.task
}

fun close() {
    executorService.execute {
        interpreter?.close()
        Log.d(TAG, "Closed TFLite interpreter.")
    }
}

private fun convertBitmapToByteBuffer(bitmap: Bitmap): ByteBuffer {
    val byteBuffer = ByteBuffer.allocateDirect(modelInputSize)
    byteBuffer.order(ByteOrder.nativeOrder())
    val pixels = IntArray(inputImageWidth * inputImageHeight)
    bitmap.getPixels(pixels, 0, bitmap.width, 0, 0, bitmap.width, bitmap.height)

    for (pixelValue in pixels) {
        val r = (pixelValue shr 16 and 0xFF)
        val g = (pixelValue shr 8 and 0xFF)
        val b = (pixelValue and 0xFF)
        // Convert RGB to grayscale and normalize pixel value to [0..1].
        val normalizedPixelValue = (r + g + b) / 3.0f / 255.0f
        byteBuffer.putFloat(normalizedPixelValue)
    }
    return byteBuffer
}

companion object {
    private const val TAG = "SymbolClassifier"
    private const val FLOAT_TYPE_SIZE = 4
    private const val PIXEL_SIZE = 1
    private const val OUTPUT_CLASSES_COUNT = 3 // Count of classes
}
}

```

# ПРИЛОЖЕНИЕ Г. ТЕКСТ ПРИЛОЖЕНИЯ

## STUDENT DIGITIZER

*File: main.py*

```
import os # Currently, os module is only for PC version
from kivy.core.window import Window
import requests
import pandas as pd
import json
import numpy as np
import re
import bs4
import pandas_gbq
from google.oauth2 import service_account
from datetime import date
from kivymd.app import MDApp
from kivy.lang import Builder
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFillRoundFlatButton
from kivymd.uix.filemanager import MDFFileManager
from kivy.metrics import dp
from kivymd.uix.datatables import MDDDataTable
#from android.storage import primary_external_storage_path
#SD_CARD = primary_external_storage_path()

Window.size = (360, 640)

class Student_Digitizer(MDApp):

    project_id = os.environ['PROJECT_ID']
    table_id_authorization = os.environ['TABLE_ID_AUTHORIZATION']
    credentials =
    service_account.Credentials.from_service_account_file('bigquery_key.json')

    sql = f"""SELECT * FROM `table_id_authorization`"""
    df_authorization = pd.read_gbq(query=sql,
                                    project_id=project_id,
                                    dialect='standard',
                                    credentials=credentials)

    logins = list(df_authorization['login'])
    passwords = list(df_authorization['password'])
    model_id = list(df_authorization['id_model'])
    model_key = list(df_authorization['key_model'])
    access = list(df_authorization['access_type'])

    table_id_1 = os.environ['TABLE_ID_1']
    table_id_2 = os.environ['TABLE_ID_2']

    header_label = os.environ['SITE_HEADER_LABEL']
    SITE_TITLE = os.environ['SITE_TITLE']
    ITEM = os.environ['SITE_ITEM']
    BODY = os.environ['SITE_BODY']
    ROW = os.environ['SITE_ROW']
    TIME = os.environ['SITE_TIME']
    LABEL = os.environ['SITE_LABEL']
    DAY = os.environ['SITE_DAY']
    CLASS_N = os.environ['CLASS_N']
    GROUP_NAME = os.environ['GROUP_NAME']
```

```

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        Window.bind(on_keyboard=self.events)
        self.just_file = ''
        self.list_file_path = []
        self.list_path = []
        self.dialog = None
        self.manager_open = False
        self.file_manager = MDFileManager(
            exit_manager=self.exit_manager,
            select_path=self.select_path,
            preview=False,
        )
        self.data_tables = None
        self.subjects_table = None

    def authorization(self):
        for i, j, n, m, l in zip(self.logins, self.passwords, self.model_id,
        self.model_key, self.access):
            if self.root.ids.user.text == i and self.root.ids.password.text == j:
                self.root.ids.id_model.text = n
                self.root.ids.key_model.text = m
                self.root.current = 'model_login'
            else:
                self.root.ids.user.text = ''
                self.root.ids.password.text = ''

    def week_schedule(self):
        schedule_url = os.environ['SITE_NAME']
        try:
            req = requests.get(schedule_url)
            parser = bs4.BeautifulSoup(req.text, 'lxml')
        except Exception:
            value_schedule = "No connection"
        else:
            try:
                value_schedule = parser.find(class_=self.header_label).text
                value_schedule = re.findall(r'\d+', value_schedule)
                value_schedule = value_schedule[0]
            except Exception:
                value_schedule = "No subjects in university schedule"
        return value_schedule

    def load_page(self, link):
        url = os.environ['URL_TO_FILE'] + self.root.ids.id_model.text +
        os.environ['URL_TYPE']
        data = {'file': open(link, 'rb')}
        response = requests.post(url,
        auth=requests.auth.HTTPBasicAuth(self.root.ids.key_model.text, ''),
        files=data)

        col_max, row_max = 0, 0
        table = []
        data = json.loads(response.text)

        for i in data['result']:
            for j in i['prediction']:
                for k in j['cells']:
                    table.append(k['text'])
                    #table.append(k['score'])
                    if k['col'] > 0:
                        col_max = k['col']
                    if k['row'] > 0:

```

```

        row_max = k['row']

    k = np.array(table).reshape(row_max, col_max * len(table) // (col_max * row_max))
    return k

def page_left(self):
    k_1 = self.load_page(link=self.list_path[0])
    df_1 = pd.DataFrame(k_1)
    df_1.is_copy = False
    df_columns = ["number", "student"]

    for i in df_1.columns[:len(df_1.columns) - len(df_columns)]:
        df_columns.append(f"column{i + 1}")

    df_1 = pd.DataFrame(k_1, columns=df_columns)

    if df_1.loc[1, 'student'] != "Предмет":
        value = "Предмет"
        df_1.loc[1, 'student'] = value
    if df_1.loc[2, 'student'] != "Тип занятия":
        value = "Тип занятия"
        df_1.loc[2, 'student'] = value

    df_1.loc[37, 'student'] = "Всего отсутствовало"
    df_1.loc[38, 'student'] = "Подпись преподавателя"
    df_1.loc[39, 'student'] = "Подпись старосты"

    for count, i in enumerate(df_1['number'][1:df_1.shape[0] - 5], 1):
        df_1.loc[count + 2, 'number'] = count

    for count, i in enumerate(df_1['student'][3:df_1.shape[0] - 3], 3):
        if i:
            df_1.loc[count, 'student'] = f"Student{df_1.loc[count, 'number']}"

    return df_1, k_1

def page_right(self):
    k_2 = self.load_page(link=self.list_path[1])
    df_columns = []
    df_2 = pd.DataFrame(k_2)
    df_2.is_copy = False

    for i in df_2.columns:
        df_columns.append(f"column{i + 13}")

    df_2 = pd.DataFrame(k_2, columns=df_columns)

    # Insert row just with numbers (or students names) from previous df
    df_2 = df_2.rename({'column25': 'lectures_all', 'column26': 'lectures', 'column27': 'message'}, axis=1)

    df_2.loc[0, 'lectures_all':'lectures'] = 'Пропущено часов занят.'
    df_2.loc[1:2, 'lectures all'] = 'всего'
    df_2.loc[1:2, 'lectures'] = 'по уважит. прич.'
    df_2.loc[0:2, 'message'] = 'Замечания деканата и преподавателей'
    return df_2, k_2

def show_alert_dialog(self):
    self.dialog = ""
    if not self.dialog:
        self.dialog = MDDialog(
            text=f"Ошибка:\n{self.root.ids.textbox_week_number.text}" if
self.root.ids.lang.active

```

```

        else f"Error:\n{self.root.ids.textbox_week_number.text}",
        buttons=[

            MDFillRoundFlatButton(
                font_style="Button",
                md_bg_color=(138.0 / 255.0, 161.0 / 255.0, 107.0 /
255.0, .5),
                text="Загрузка файла" if self.root.ids.lang.active else
"Manual load",
                on_release=lambda _: screen_update()),

            MDFillRoundFlatButton(
                font_style="Button",
                md_bg_color=(138.0 / 255.0, 161.0 / 255.0, 107.0 / 255.0,
.5),
                text="Закрыть" if self.root.ids.lang.active else "Cancel",
                on_release=lambda _: self.dialog.dismiss())
        ])
    self.dialog.open()

    def screen_update():
        self.dialog.dismiss()
        screen = self.root.current = 'manual_load'
        return screen

    def show_information(self):
        self.dialog = ""
        if not self.dialog:
            self.dialog = MDDialog(
                text=f"Для более подробной информации напишите по контактам" \
                     if self.root.ids.lang.active else f"For more information
write to contact list",
                buttons=[MDFillRoundFlatButton(
                    font_style="Button",
                    md_bg_color=(138.0 / 255.0, 161.0 / 255.0, 107.0 / 255.0,
.5),
                    text="Закрыть" if self.root.ids.lang.active else "Cancel",
                    on_release=lambda _: self.dialog.dismiss())])
        self.dialog.open()

    def subjects_schedule(self, group_name):
        schedule_url_full = os.environ['SITE_NAME'] + group_name +
os.environ['SITE_TYPE']
        table_list, list_all_table, list_all_table_2, list_all_table_data, d, d2
= [], [], [], [], []
        if self.root.ids.textbox_week_number.text != "No connection" \
            and self.root.ids.textbox_week_number.text != "No subjects in
university schedule":
            req = requests.get(schedule_url_full)
            parser = bs4.BeautifulSoup(req.text, 'lxml')
        else:
            self.just_file = self.list_file_path[0]
            with open(self.just_file, encoding='utf-8') as file:
                src = file.read()
                parser = bs4.BeautifulSoup(src, "lxml")
        try:
            self.value_schedule = ''
            value_schedule = int(self.week_schedule())
            value_schedule = int(self.root.ids.textbox_week_number.text)
        except Exception:
            value_schedule = parser.find(class_=self.SITE_TITLE).text
            value_schedule = re.findall(r'\d+', value_schedule)
            value_schedule = int(str(value_schedule[0]))
            group_name =

```

```

parser.find(class_=self.GROUP_NAME).text.split('&middot')[1].replace(" ", "")

        table = parser.findAll(class_=self.BODY)
        rows_ = [r for r in table[0].findAll(class_=self.ROW) if
r.findAll(class_=self.DAY)]
        for r in rows_:
            day = r.findAll(class_=self.DAY)[0].text
            time_rows = [dr for dr in r.findAll(class_=self.ROW) if
dr.findAll(class_=self.TIME)]
            for tr in time_rows:
                time = tr.findAll(class_=self.TIME)[0].text
                lessons = [lesson for lesson in tr.findAll(class_=self.ROW) if
lesson.findAll(class_=self.ITEM)]
                lessons_list = []
                for lesson in lessons:
                    week_type =
lesson.findAll(class_=self.LABEL)[0].findChild('span').text \
                        if lesson.findAll(class_=self.LABEL) else ''
                    item = lesson.findAll(class_=self.ITEM)[0] \
                        if lesson.findAll(class_=self.LABEL) else
lesson.findAll(class_=self.ITEM)[0]
                    class_number =
item.findAll(class_=self.CLASS_N)[0].text.replace('&nbs', ' ')
                    item_label_reg = re.search(r'\t[\w\s,-.]+', item.text if
item else None)
                    item_label = re.sub(r'(\s+\cdot)|[\n\t]| \s{2}', '', item_label_reg.group(0)) if item_label_reg \
                        else None
                    item_label = item_label.replace(class_number, '').strip()
                    lessons_list.append({'w': week_type, 'n': item_label, 'c':
class_number})
                if lessons_list:
                    table_list.append({'d': day, 't': time, 'i': lessons_list})

        for i in table_list:
            list_all_table.append([i['d'], i['t'], i['i']])

        for t, i, j in list_all_table:
            list_all_table_2.append([t, i, j])

        for i in list_all_table_2:
            for j in i[2]:
                list_all_table_data.append([i[0], i[1], j['w'], j['n']])

all_subjects_full = []

for t, i, j, k in list_all_table_data:
    k_split = ''.join([k[0].upper() for k in k.split()])
    if k and k_split != "ФКИС":
        all_subjects_full.append(k)
        d.append(
            {
                'week_n': t,
                'class_t': i,
                'week_t': j,
                'subject': k_split
            }
        )
df = pd.DataFrame(d)

list_values_subjects = []
for count, i in enumerate(df['week_t'], 0):
    if "недели" in i:
        list_values_subjects.append(count)

```

```

for i in list_values_subjects:
    k = df.loc[i, 'week_t']
    k = re.findall(r'\d.*', k)
    l = k[0].split(' ')
    df.loc[i, 'week_t'] = list(map(int, l))

for count, i in enumerate(df['week_t'], 0):
    if i == "по чётным":
        df.loc[count, 'week_t'] = [i for i in range(1, 19) if i % 2 ==
0]
    elif i == "по нечётным":
        df.loc[count, 'week_t'] = [i for i in range(1, 19) if i % 2 ==
1]
    elif not i:
        df.loc[count, 'week_t'] = [i for i in range(1, 19)]

for count, i in enumerate(df['week_n'], 0):
    if df.loc[count, 'week_n'] == 'пн':
        df.loc[count, 'week_n'] = 1
    elif df.loc[count, 'week_n'] == 'вт':
        df.loc[count, 'week_n'] = 2
    elif df.loc[count, 'week_n'] == 'ср':
        df.loc[count, 'week_n'] = 3
    elif df.loc[count, 'week_n'] == 'чт':
        df.loc[count, 'week_n'] = 4
    elif df.loc[count, 'week_n'] == 'пт':
        df.loc[count, 'week_n'] = 5
    elif df.loc[count, 'week_n'] == 'сб':
        df.loc[count, 'week_n'] = 6

for count, i in enumerate(df['class_t'], 0):
    if df.loc[count, 'class_t'] == '08:30-10:00':
        df.loc[count, 'class_t'] = 1
    elif df.loc[count, 'class_t'] == '10:15-11:45':
        df.loc[count, 'class_t'] = 2
    elif df.loc[count, 'class_t'] == '12:00-13:30':
        df.loc[count, 'class_t'] = 3
    elif df.loc[count, 'class_t'] == '14:00-15:30':
        df.loc[count, 'class_t'] = 4
    elif df.loc[count, 'class_t'] == '15:45-17:15':
        df.loc[count, 'class_t'] = 5
    elif df.loc[count, 'class_t'] == '17:30-19:00':
        df.loc[count, 'class_t'] = 6
    elif df.loc[count, 'class_t'] == '19:15-20:45':
        df.loc[count, 'class_t'] = 7
    elif df.loc[count, 'class_t'] == '21:00-22:30':
        df.loc[count, 'class_t'] = 8

for count, i in enumerate(df['week_t'], 0):
    if value_schedule in i and len(df['subject'][count]) and
df['subject'][count] != 'ФКМС':
        d2.append(
            {
                'week_n': df.loc[count][0],
                'class_t': df.loc[count][1],
                'subject': df.loc[count][3]
            }
        )

df_current = pd.DataFrame(d2)
if df_current.empty:
    l = ''
else:

```

```

l = df_current

try:
    if self.subjects_df:
        self.subjects_df = self.subjects_df[0:0]
except Exception:
    all_subjects_full = set(all_subjects_full)
    subject_values_sorted = []

for j in all_subjects_full:
    k_split = "".join([j[0].upper() for j in j.split()])
    subject_values_sorted.append(k_split)

list_subjects = []

for i, j in zip(subject_values_sorted, all_subjects_full):
    list_subjects.append(
        {
            'abbreviate': i,
            'subject': j
        }
    )
subjects_df = pd.DataFrame(list_subjects)
column_subjects = list(subjects_df.columns)
row_data_subjects = subjects_df.to_records(index=False)
column_subjects = [(x, dp(25)) if _ % 2 == 0 else (x, dp(250)) for _, x in enumerate(column_subjects, 0)]

if not self.subjects_table:
    self.subjects_table = MDDataTable(
        background_color_header=(0, 1, 0, .1),
        pos_hint={"top": 1, "center_y": 0.5},
        size_hint_x=1,
        size_hint_y=None,
        height=Window.height * 0.9,
        use_pagination=True,
        column_data=column_subjects,
        row_data=row_data_subjects,
    )
    self.root.ids.subjects_box.add_widget(self.subjects_table)
else:
    self.subjects_table.update_row_data(self.subjects_table,
row_data_subjects)
return group_name, df_current, l, value_schedule

# Screen 1
def start(self):
    if self.root.ids.id_model.text != "" and self.root.ids.key_model.text != "":
        try:
            self.root.ids.textbox_week_number.text = self.week_schedule()
        except Exception:
            pass
        if self.root.ids.textbox_week_number.text == "No connection" \
            or self.root.ids.textbox_week_number.text == "No subjects in university schedule":
            self.show_alert_dialog()
        else:
            self.root.current = 'menu'
    else:
        self.root.ids.id_model.text = ""
        self.root.ids.key_model.text = ""

# Screen 2

```

```

def show_data(self):
    func = self.subjects_schedule(self.root.ids.textbox.text)
    text = func[2] # 1
    value_schedule = func[3]

    if len(text) > 0: # If name is not empty do:
        try:
            if self.df:
                self.df = self.df[0:0]
        except Exception:
            pass
        self.root.current = 'subjects' # Move to the Screen3
    else:
        self.root.ids.textbox.text = ''
    textbox = self.root.ids.textbox.text
    return textbox, text

def collect(self):
    func = self.subjects_schedule(self.root.ids.textbox.text)
    df_current = func[1]
    value_schedule = func[3]
    df_1, k_1 = self.page_left()
    #df_1_shape = k_1
    df_2, k_2 = self.page_right()
    #df_2_shape = k_2

    if len(self.root.ids.textbox.text) > 0:
        df_1.loc[0, 'number'] = self.root.ids.textbox.text
    else:
        group_name = self.subjects_schedule(self.root.ids.textbox.text)[0]
        df_1.loc[0, 'number'] = group_name

    df_1.loc[1, 'number'] = value_schedule
    df = df_1.join(df_2)

    list_signature_lecturer = []
    list_signature = []
    subject_index_list = []
    subject_index_schedule = []
    list2 = []
    subject_indexes_schedule_to_df = []
    values_subject = []
    values_not_subject = []
    list_values_number = []
    list_value_sum = []
    values_rows = []
    items_subjects = []

    for count, i in enumerate(df.loc[2][2:26], 2):
        if len(i) > 1:
            value = "Практические занятия"
            df.iloc[2, count] = value
        elif len(i) == 1:
            value = "Лекция"
            df.iloc[2, count] = value
        elif len(i) == 0:
            pass

    for count, i in enumerate(df.loc[38], 0):
        if i:
            if count >= 2:
                list_signature_lecturer.append(count)
    df.loc[38][list_signature_lecturer] = "Yes"

```

```

for count, i in enumerate(df.loc[39], 0):
    if i:
        if count >= 2:
            list_signature.append(count)
df.loc[39][list_signature] = "Yes"

i = 0
for j, _ in enumerate(df.loc[1][2:26], 2):
    if j % 4 == 2:
        i += 1
    subject_index_list.append([i, j])

df.iloc[1:2, 2:26] = ''

# Index of subjects from schedule to import to df
for i in df_current['week_n']:
    subject_index_schedule.append(i)

list1 = list(df_current['week_n'])

n = 1
for count, i in enumerate(list1):
    list2.append(n)
    if count + 1 < len(list1):
        if list1[count + 1] == list1[count]:
            n += 1
        else:
            n = 1

items = subject_index_schedule + list2
items_week_n = items[:len(items) // 2]
items_number_column = items[len(items) // 2:]

for i, j in zip(items_week_n, items_number_column):
    if i == 2:
        j += 4
    elif i == 3:
        j += 8
    elif i == 4:
        j += 12
    elif i == 5:
        j += 16
    elif i == 6:
        j += 20
    j += 1 # Because we compare with slice of df from 2 column
subject_indexes_schedule_to_df.append([i, j])

for count, (i, j) in enumerate(subject_indexes_schedule_to_df, 0):
    df.loc[1][j] = df_current['subject'][count]

for count, i in enumerate(df.loc[1][2:26], 2):
    if i:
        values_subject.append(count)
    elif not i:
        values_not_subject.append(count)

for i in values_not_subject: # Clear values in all row where is no
subject
    df.iloc[:, i] = ''

# Remove values where is no student
for count, i in enumerate(df['student'][3:df.shape[0] - 3], 1):
    if not i:
        df.loc[count + 2, 'number'] = ''

```

```

for count, i in enumerate(df['number'][3:37], 1):
    if not i:
        list_values_number.append(count + 2)

for i in list_values_number:
    df.iloc[i][2:27] = ''

# Sum values
for i in values_subject: # Fill only columns with some subject
    df.iloc[3:37, i:i + 1] = df.iloc[3:37, i:i + 1].apply(
        lambda x: x.str.extract(r'([а-яА-Яа-зА-З0-9])',
expand=False)).replace('None', np.nan).notnull().astype(
    int)
    list_value_sum.append(sum(df.iloc[3:37, i]))

# Remove values where is no student
for i in list_values_number:
    df.iloc[i][2:27] = ''

# Summary by student
for count, i in enumerate(df['student'][3:37], 3):
    if i:
        values_rows.append(count)

for i in values_rows:
    k = 0
    for j in df.loc[i][2:26]:
        if j:
            k += j
    df.loc[i, 'lectures_all'] = k

summary = 0
for i in df['lectures_all'][3:df.shape[0] - 3]:
    if i:
        summary += i

df.loc[37, 'lectures_all'] = summary

for i in values_subject:
    k = 0
    for j in df.iloc[3:37, i]:
        if j:
            k += j
    df.loc[37][i] = k

# Df only: number; student; lectures_all; group_name; week_number
df_students = df.iloc[1:37, :27]
df_students.loc[1, 'student'] = "Студент"
df_students = df_students[df_students["number"] != '']
df_students = df_students.reset_index(drop=True)

# Insert 2 values in 2 columns
df_students.loc[df_students.shape[0], 'number'] = 'total'
df_students.loc[df_students.shape[0] - 1, 'student'] = 'all'
df_students = df_students[1:df_students.shape[0] - 1]
df_students = df_students[['number', "student", "lectures_all"]]
df_students['group'] = df.loc[0, 'number']
df_students['week_n'] = df.loc[1, 'number']
df_students['date'] = date.today()
df_students = df_students.fillna('')
df_students = df_students.reset_index(drop=True)

# Df only: column{i}; group_name; week_number

```

```

        for count, i in enumerate(values_subject, 0):
            items_subjects.append(
                {
                    'subject': df.loc[1][i],
                    'total': df.loc[37][i],
                    'group': df.loc[0, 'number'],
                    'week_n': df.loc[1, 'number'],
                    'date': date.today()
                }
            )
    )

    df_subjects = pd.DataFrame(items_subjects)
    df_subjects = df_subjects.groupby(['subject', 'group', 'date',
    'week_n'])['total'].sum().reset_index()
    return df, df_students, df_subjects

# Screen 3
def file_manager_open(self):
    self.manual_file_path = []
    self.file_path = []
    self.file_manager.show('/') # Output manager to the screen
    #self.file_manager.show(SD_CARD)
    self.manager_open = True

def select_path(self, path):
    if self.root.ids.lang.active:
        self.root.ids.file.text = "Html файл: "
        self.root.ids.file1.text = "1 файл: "
        self.root.ids.file2.text = "2 файл: "
    else:
        self.root.ids.file.text = "Html file: "
        self.root.ids.file1.text = "1 file: "
        self.root.ids.file2.text = "2 file: "

    if self.root.current == 'manual_load':
        if not self.list_file_path:
            self.list_file_path.append(path)
        else:
            self.list_file_path.clear()
            self.list_file_path.append(path)

    if self.root.current == 'processing':
        if len(self.list_path) < 2:
            self.list_path.append(path)
        else:
            self.list_path.clear()
            self.list_path.append(path)
    self.exit_manager()

    for count, path in enumerate(self.list_file_path, 0):
        manual_path_update = list(path)[::-1]
        for i in iter(manual_path_update):
            if i in ['\\', '/']:
                break
            self.manual_file_path.append(i)
        if count < 1:
            self.manual_file_path.append(';')
    manual_file = ''.join(self.manual_file_path)[::-1].split(';')[::-1]

    for count, path in enumerate(self.list_path, 0):
        path_update = list(path)[::-1]
        for i in iter(path_update):
            if i in ['\\', '/']:
                break

```

```

        self.file_path.append(i)
    if count < 1:
        self.file_path.append(';')
file = ''.join(self.file_path)[::-1].split(';;')[::-1]

#manual_load
try:
    self.root.ids.file.text =
f'{self.root.ids.file.text}\n{manual_file[0]}'
except Exception:
    pass

#processing
try:
    self.root.ids.file1.text = f'{self.root.ids.file1.text}\n{file[0]}'
    self.root.ids.file2.text = f'{self.root.ids.file2.text}\n{file[1]}'
except Exception:
    pass
return self.list_file_path, self.list_path

def exit_manager(self, *args):
    self.manager_open = False
    self.file_manager.close()

def events(self, instance, keyboard, keycode, text, modifiers):
    if keyboard in (1001, 27):
        if self.manager_open:
            self.file_manager.back()
    return True

def callback_button_collect(self):
    # Screen 5
    self.df_students['date'] = pd.to_datetime(self.df_students['date'])
    self.df_subjects['date'] = pd.to_datetime(self.df_subjects['date'])
    pandas_gbq.to_gbq(self.df_students, destination_table=self.table_id_1,
                      project_id=self.project_id, if_exists='append',
                      credentials=self.credentials)
    pandas_gbq.to_gbq(self.df_subjects, project_id=self.project_id,
                      destination_table=self.table_id_2,
                      if_exists='append', credentials=self.credentials)
    self.root.current = 'check'

def show_table(self):
    self.df, self.df_students, self.df_subjects = self.collect()
    self.list_path.clear()
    if self.df.shape[0] == 40 and self.df.shape[1] == 29: # Input table (40
rows x 29 columns)
        # Screen 4
        column_data = list(self.df.columns)
        row_data = self.df.to_records(index=False)
        column_data = [(x, dp(75)) if (_ >= 1 and (_ % 26 == 0 or _ % 27 ==
0 or _ % 28 == 0)) \
                        else (x, dp(25)) for _, x in
enumerate(column_data, 0)]
        if not self.data_tables:
            self.data_tables = MDDataTable(
                background_color_header=(0, 1, 0, .1),
                pos_hint={"top": 1, "center_y": 0.5},
                size_hint_x=1,
                size_hint_y=None,
                height=Window.height * 0.9,
                use_pagination=True,
                column_data=column_data,
                row_data=row_data,

```

```

        )
        self.root.ids.table_box.add_widget(self.data_tables)
    else:
        self.data_tables.update_row_data(self.data_tables, row_data)
        self.root.current = 'table'
else:
    self.root.current = 'processing' # Move to the Screen3

def uploading(self):
    if self.root.ids.textbox.text == '':
        self.root.current = 'manual_load'
    else:
        self.root.current = 'menu'

def build(self):
    self.theme_cls.primary_palette = "Green"
    return Builder.load_file('main.kv')

Student_Digitizer().run()

```

### **File: main.kv**

```

#:import webbrowser webbrowser
ScreenManager:
    MDScreen:
        name: 'login'
        md_bg_color: app.theme_cls.bg_light
        on_pre_enter:
            image2.anim_delay = -1
            image3.anim_delay = -1

        MDFloatLayout:
            size_hint_x: .75
            size_hint_y: .75
            Image:
                id: image1
                source: "assets/login_another.gif" if theme.active else
"assets/login.gif"
                anim_delay: 1/24
                pos_hint: {"x": .15, "y": .5}

        MDFloatLayout:

            MDLabel:
                font_style: "Button"
                text: "Авторизация" if lang.active else "Authentication"
                pos_hint: {"center_x": .5, "center_y": .545}
                valign: 'middle'
                halign: 'center'

            MDFloatLayout:
                size_hint_x: .95
                size_hint_y: None
                pos_hint: {"center_x": .5, "center_y": .47}

            MDTTextField:
                id: user
                text_color_focus: (1,1,1,1) if theme.active else (0,0,0,1)
                color_mode: "custom"
                mode: "fill"
                hint_text: "Логин" if lang.active else "Username"
                size_hint_x: 1
                size_hint_y: None

```

```

        icon_right: "account"
        pos_hint: {"center_x": .5, "center_y": .5}
        height: self.minimum_height
        multiline: False

    MDFloatLayout:
        size_hint_x: .95
        size_hint_y: None
        pos_hint: {"center_x": .5, "center_y": .365}

    MDTextField:
        id: password
        text_color_focus: (1,1,1,1) if theme.active else (0,0,0,1)
        color_mode: "custom"
        mode: "fill"
        hint_text: "Пароль" if lang.active else "Password"
        password: True
        size_hint_x: 1
        size_hint_y: None
        icon_right: "key"
        pos_hint: {"center_x": .5, "center_y": .5}
        height: self.minimum_height
        multiline: False

    MDTextButton:
        size_hint_x: .95
        size_hint_y: None
        font_style: "Caption"
        text: "Забыли пароль?" if lang.active else "Forgot your
password?"
        pos_hint: {"center_x": .5, "center_y": .3}
        on_press: app.show_information()
        valign: 'middle'
        halign: 'center'

    MDLabel:
        size_hint_x: .95
        size_hint_y: None
        font_style: "Button"
        text: f"Русский\\язык" if lang.active else f"English\\nlanguage"
        pos_hint: {"center_x": .5, "center_y": .26}
        valign: 'middle'
        halign: 'right' if lang.active else 'left'

    MDSwitch:
        id: lang
        width: dp(48)
        pos_hint: {"center_x": .5, "center_y": .26}
        valign: 'middle'
        halign: 'center'

    MDLabel:
        size_hint_x: .95
        size_hint_y: None
        font_style: "Button"
        text: "Темная тема" if lang.active and theme.active else
        "Светлая тема" if lang.active and not theme.active else "Light Theme" if not
        lang.active and not theme.active else "Dark Theme"
        pos_hint: {"center_x": .5, "center_y": .19}
        valign: 'middle'
        halign: 'right' if theme.active else 'left'

    MDSwitch:
        id: theme

```

```

width: dp(48)
pos_hint: {'center_x': .5, 'center_y': .19}
valign: 'middle'
halign: 'center'
on_active:
    app.theme_cls.theme_style = "Dark" if self.active else
"Light"

MDTextButton:
    id: information
    size_hint_x: .95
    size_hint_y: None
    font_style: "Button"
    text: "Информация о приложении" if lang.active else "Application
information"
    pos_hint: {"center_x": .5, "center_y": .125}
    valign: 'middle'
    halign: 'center'
    line_color: app.theme_cls.primary_color
    md_bg_color: 0, 1, 0, .1
    on_press:
        root.transition.direction = 'left'
        root.current = 'information'

MDFillRoundFlatButton:
    size_hint_x: .5
    size_hint_y: None
    font_style: "Button"
    text: "Выход" if lang.active else "Close"
    pos_hint: {"center_x": .25, "center_y": .05}
    md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
    text_color: (1,1,1,1) if theme.active else (0,0,0,1)
    on_press: app.stop()

MDFillRoundFlatButton:
    size_hint_x: .5
    size_hint_y: None
    font_style: "Button"
    disabled: False if user.text else True
    text: "Вход" if lang.active else "Login"
    pos_hint: {"center_x": .75, "center_y": .05}
    md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
    text_color: (1,1,1,1) if theme.active else (0,0,0,1)
    on_press:
        root.transition.direction = 'left'
        app.authorization()

MDScreen:
    name: 'information'
    md_bg_color: app.theme_cls.bg_light

MDLabel:
    size_hint_x: .95
    size_hint_y: None
    font_style: "Caption"
    text: "Текущая версия: 1.0" if lang.active else "Current version:
1.0"
    pos_hint: {"center_x": .5, "center_y": .95}
    valign: 'top'
    halign: 'center'

MDLabel:
    size_hint_x: .95
    size_hint_y: None

```

```

        font_style: "Caption"
        text: "Инструкция:\n1. Авторизируйтесь в приложении\n2. Выберите
группу и другие параметры\n3. Загрузите изображения студенческого журнала\n4.
Проверьте правильность данных\n5. Загрузите отчет\n\nЛицензия & Подробнее:" if
lang.active else "FAQ:\n1. Log in to the application\n2. Choose group and other
parametres\n3. Load images of student journal\n4. Check input and output
data\n5. Load data\n\nLicense & Details:"
            pos_hint: {"center_x": .5, "center_y": .775}
            valign: 'center'
            halign: 'center'

MDIconButton:
        size_hint_x: .1
        size_hint_y: None
        markup: True
        pos_hint: {"center_x": .5, "center_y": .605} if lang.active else
{"center_x": .5, "center_y": .62}
        icon: "file-pdf-box"
        on_press:
            webbrowser.open('***')

MDLabel:
        size_hint_x: .95
        size_hint_y: None
        font_style: "Caption"
        text: "\nПродолжая использование приложения Вы соглашаетесь с тем,
что:\n1. Используете файлы приложения в учебных некоммерческих целях\n2. Автор
не несет ответственность за данные, загружаемые другими пользователями,
работоспособность отдельных компонентов или приложения в целом\n3. Права на
приложение принадлежат автору" if lang.active else "By continue using the
application, You agree that:\n1. You are using application files for educational
non-commercial purposes\n 2. The author is not responsible for uploaded data by
other users, the performance of individual components or the application as
well\n3. Rights to the application belong to the author"
            pos_hint: {"center_x": .5, "center_y": .455} if lang.active else
{"center_x": .5, "center_y": .47}
            valign: 'bottom'
            halign: 'center'

MDLabel:
        size_hint_x: .95
        size_hint_y: None
        font_style: "Button"
        text: "Автор:\nДмитрий Москалев" if lang.active else
"Author:\nDmitry Moskalev"
            pos_hint: {"center_x": .5, "center_y": .27} if lang.active else
{"center_x": .5, "center_y": .3}
            valign: 'middle'
            halign: 'center'

MDLabel:
        size_hint_x: .95
        size_hint_y: None
        font_style: "Caption"
        text: "Контакты:" if lang.active else "Contacts:"
            pos_hint: {"center_x": .5, "center_y": .205} if lang.active else
{"center_x": .5, "center_y": .23}
            valign: 'middle'
            halign: 'center'

MDIconButton:
        size_hint_x: .1
        size_hint_y: None
        markup: True

```

```

        pos_hint: {"center_x": .2, "center_y": .15} if lang.active else
{"center_x": .2, "center_y": .175}
        icon: "***"
        on_press:
            webbrowser.open('***')

MDIconButton:
    size_hint_x: .1
    size_hint_y: None
    markup: True
    pos_hint: {"center_x": .5, "center_y": .15} if lang.active else
{"center_x": .5, "center_y": .175}
    icon: "***"
    on_press:
        webbrowser.open('***')

MDIconButton:
    size_hint_x: .1
    size_hint_y: None
    markup: True
    pos_hint: {"center_x": .8, "center_y": .15} if lang.active else
{"center_x": .8, "center_y": .175}
    icon: "***"
    on_press:
        webbrowser.open('***')

MDFillRoundFlatButton:
    size_hint_x: .5
    size_hint_y: None
    font_style: "Button"
    text: "Назад" if lang.active else "Back"
    pos_hint: {"center_x": .5, "center_y": .05}
    md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
    text_color: (1,1,1,1) if theme.active else (0,0,0,1)
    on_press:
        root.transition.direction = 'right'
        root.current = 'login'

MDScreen:
    name: 'model_login'
    md_bg_color: app.theme_cls.bg_light

    MDFloatLayout:
        size_hint_x: .75
        size_hint_y: .75
        Image:
            id: image1
            source: "assets/login_another.gif" if theme.active else
"assets/login.gif"
            anim_delay: 1/24
            pos_hint: {"x": .15, "y": .5}

    MDFloatLayout:

        MDLabel:
            font_style: "Button"
            text: "Настройки модели" if lang.active else "Model settings"
            pos_hint: {"center_x": .5, "center_y": .5}
            valign: 'middle'
            halign: 'center'

        MDFloatLayout:
            size_hint_x: .95
            size_hint_y: None

```

```

pos_hint: {"center_x": .5, "center_y": .42}

MDTextField:
    id: id_model
    text_color_focus: (1,1,1,1) if theme.active else (0,0,0,1)
    color_mode: "custom"
    mode: "fill"
    hint_text: "Идентификатор" if lang.active else "ID"
    password: True
    size_hint_x: 1
    size_hint_y: None
    icon_right: "connection"
    pos_hint: {"center_x": .5, "center_y": .5}
    height: self.minimum_height
    multiline: False

MDFloatLayout:
    size_hint_x: .95
    size_hint_y: None
    pos_hint: {"center_x": .5, "center_y": .315}

MDTextField:
    id: key_model
    text_color_focus: (1,1,1,1) if theme.active else (0,0,0,1)
    color_mode: "custom"
    mode: "fill"
    hint_text: "API ключ" if lang.active else "API key"
    password: True
    size_hint_x: 1
    size_hint_y: None
    icon_right: "key"
    pos_hint: {"center_x": .5, "center_y": .5}
    height: self.minimum_height
    multiline: False

MDFillRoundFlatButton:
    size_hint_x: .5
    size_hint_y: None
    font_style: "Button"
    text: "Назад" if lang.active else "Back"
    pos_hint: {"center_x": .25, "center_y": .05}
    md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
    text_color: (1,1,1,1) if theme.active else (0,0,0,1)
    on_press:
        root.transition.direction = 'right'
        root.current = 'login'

MDFillRoundFlatButton:
    size_hint_x: .5
    size_hint_y: None
    font_style: "Button"
    text: "Далее" if lang.active else "Next"
    pos_hint: {"center_x": .75, "center_y": .05}
    md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
    text_color: (1,1,1,1) if theme.active else (0,0,0,1)
    on_press:
        root.transition.direction = 'left'
        app.start()

MDScreen:
    name: 'manual_load'
    md_bg_color: app.theme_cls.bg_light
    on_pre_enter: root.ids.file.text = "Html файл: " if lang.active else
    "Html file: "

```

```

MDLabel:
    font_style: "Caption"
    text: "Выберите файл" if lang.active else "Choose file"
    valign: 'middle'
    halign: 'center'
    pos_hint: {"center_x": .5, "center_y": .9}

MDFillRoundFlatButton:
    font_style: "Button"
    icon: "file"
    text: "Открыть файловый менеджер" if lang.active else "Open manager"
    size_hint_x: .75
    size_hint_y: None
    pos_hint: {"center_x": .5, "center_y": .8}
    md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
    text_color: (1,1,1,1) if theme.active else (0,0,0,1)
    on_press:
        app.file_manager_open()

MDLabel:
    id: file
    text: "Html файл: " if lang.active else "Html file: "
    font_style: "Caption"
    valign: 'middle'
    halign: 'center'
    pos_hint: {"center_x": .5, "center_y": .32}

MDFillRoundFlatButton:
    font_style: "Button"
    disabled: False if len(file.text) > 11 else True
    text: "Далее" if lang.active else "OK"
    size_hint_x: .75
    size_hint_y: None
    pos_hint: {"center_x": .5, "center_y": .125}
    md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
    text_color: (1,1,1,1) if theme.active else (0,0,0,1)
    on_press:
        root.transition.direction = 'left'
        app.show_data()
        root.current = 'subjects'

MDFillRoundFlatButton:
    font_style: "Button"
    text: "Назад" if lang.active else "Back"
    size_hint_x: .25
    size_hint_y: None
    pos_hint: {"center_x": .5, "center_y": .05}
    md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
    text_color: (1,1,1,1) if theme.active else (0,0,0,1)
    on_press:
        root.transition.direction = 'right'
        root.current = 'login'

MDScreen:
    name: 'menu'
    md_bg_color: app.theme_cls.bg_light
    on_pre_enter:
        image2.anim_delay = 1/24

MDFloatLayout:
    Image:
        id: image2

```

```

        source: "assets/process_another.gif" if theme.active else
"assets/process.gif"
        pos_hint: {"x": 0, "y": .23}

    MDFloatLayout:
        size_hint_x: .95
        size_hint_y: None
        pos_hint: {"center_x": .5, "center_y": .39}

    MDTextField:
        id: textbox
        text_color_focus: (1,1,1,1) if theme.active else (0,0,0,1)
        color_mode: "custom"
        mode: "fill"
        hint_text: "Введите название группы" if lang.active else
"Enter group name"
        required: True
        error_color: 'brown'
        icon_right: "account-group"
        pos_hint: {'center_x': .5, 'center_y': .5}
        size_hint_x: 1
        size_hint_y: None
        multiline: False

    MDFloatLayout:
        size_hint_x: .95
        size_hint_y: None
        pos_hint: {"center_x": .5, "center_y": .285}

    MDTextField:
        id: textbox_week_number
        text_color_focus: (1,1,1,1) if theme.active else (0,0,0,1)
        color_mode: "custom"
        mode: "fill"
        hint_text: "Введите номер недели" if lang.active else "Enter
week number"
        required: False
        error_color: 'brown'
        icon_right: "calendar-blank"
        pos_hint: {'center_x': .5, 'center_y': .5}
        size_hint_x: 1
        size_hint_y: None
        multiline: False
        max_text_length: 2

    MDFillRoundFlatButton:
        font_style: "Button"
        disabled: False if textbox.text and textbox_week_number.text
else True
        text: "Далее" if lang.active else "OK"
        size_hint_x: .75
        size_hint_y: None
        pos_hint: {'center_x': .5, 'center_y': .125}
        md_bg_color: (138.0/255.0, 161.0/255.0, 107.0/255.0, .5)
        text_color: (1,1,1,1) if theme.active else (0,0,0,1)
        on_press:
            root.transition.direction = 'left'
            app.show_data()
            root.current = 'subjects'

    MDFillRoundFlatButton:
        font_style: "Button"
        text: "Назад" if lang.active else "Back"
        size_hint_x: .25

```

```

        size_hint_y: None
        pos_hint: {'center_x': .5, 'center_y': .05}
        md_bg_color: (138.0/255.0, 161.0/255.0, 107.0/255.0, .5)
        text_color: (1,1,1,1) if theme.active else (0,0,0,1)
        on_press:
            root.transition.direction = 'right'
            root.current = 'login'

MDScreen:
    id: data_subjects
    name: 'subjects'
    md_bg_color: app.theme_cls.bg_light
    on_pre_enter:
        image2.anim_delay = -1
        image4.anim_delay = -1
        image4.anim_loop = 0

MDBBoxLayout:
    orientation: "vertical"
    size_hint_x: 1
    adaptive_height: True
    pos_hint: {"center_x": .5, "center_y": .5}
    md_bg_color: app.theme_cls.bg_light

    MDBBoxLayout:
        id: subjects_box
        size_hint_x: 1
        adaptive_height: True
        pos_hint: {"center_x": .5, "center_y": .5}
        md_bg_color: app.theme_cls.bg_light

    MDBBoxLayout:
        orientation: "vertical"
        adaptive_height: True
        pos_hint: {"center_x": .5, "center_y": .5}
        md_bg_color: app.theme_cls.bg_light

    MDBBoxLayout:
        orientation: "horizontal"
        adaptive_height: True
        pos_hint: {"center_x": .5, "center_y": .5}
        md_bg_color: app.theme_cls.bg_light

    MDFillRoundFlatButton:
        font_style: "Button"
        text: "Назад" if lang.active else "Back"
        size_hint_x: .5
        size_hint_y: None
        pos_hint: {"center_x": .25, "center_y": .5}
        md_bg_color: (138.0/255.0, 161.0/255.0, 107.0/255.0, .5)
        text_color: (1,1,1,1) if theme.active else (0,0,0,1)
        on_press:
            root.transition.direction = 'right'
            root.current = 'model_login'

    MDFillRoundFlatButton:
        font_style: "Button"
        text: "Далее" if lang.active else "OK"
        size_hint_x: .5
        size_hint_y: None
        pos_hint: {"center_x": .75, "center_y": .5}
        md_bg_color: (138.0/255.0, 161.0/255.0, 107.0/255.0, .5)
        text_color: (1,1,1,1) if theme.active else (0,0,0,1)
        on_press:

```

```

        root.transition.direction = 'left'
        root.current = 'processing'

MDScreen:
    name: 'spinner_screen'
    md_bg_color: app.theme_cls.bg_light
    on_enter:
        app.show_table()

MDLabel:
    id: spinner_label
    font_style: "Caption"
    text: "Обработка изображений\n..." if lang.active else "Image
processing\n..."
    valign: 'top'
    halign: 'center'
    pos_hint: {"center_x": .5, "center_y": .5}

MDScreen:
    name: 'processing'
    md_bg_color: app.theme_cls.bg_light
    on_pre_enter:
        image3.anim_delay = 1/24
        root.ids.file1.text = "1 файл: " if lang.active else "1 file: "
        root.ids.file2.text = "2 файл: " if lang.active else "2 file: "

MDFloatLayout:

    Image:
        id: image3
        source: "assets/manager_another.gif" if theme.active else
"assets/manager.gif"
        pos_hint: {"x": 0, "y": 0.1}

    MDLabel:
        font_style: "Caption"
        text: "Загрузить изображения" if lang.active else "Please
provide images"
        valign: 'top'
        halign: 'center'
        pos_hint: {"center_x": .5, "center_y": .95}

    MDFillRoundFlatButton:
        font_style: "Button"
        text: "Файловый менеджер" if lang.active else "File manager"
        size_hint_x: .75
        size_hint_y: None
        pos_hint: {'center_x': .5, 'center_y': .895}
        md_bg_color: (138.0/255.0, 161.0/255.0, 107.0/255.0, .5)
        text_color: (1,1,1,1) if theme.active else (0,0,0,1)
        on_press: app.file_manager_open()

    MDLabel:
        id: file1
        text: "1 файл: " if lang.active else "1 file: "
        font_style: "Caption"
        valign: 'middle'
        halign: 'center'
        pos_hint: {"center_x": .5, "center_y": .28}

    MDLabel:
        id: file2
        text: "2 файл: " if lang.active else "2 file: "

```

```

        font_style: "Caption"
        valign: 'middle'
        halign: 'center'
        pos_hint: {"center_x": .5, "center_y": .205}

    MDFlatButton:
        id: button_processing
        font_style: "Button"
        disabled: False if len(file1.text) >= 10 and len(file2.text) >=
10 else True
        text: "Далее" if lang.active else "OK"
        size_hint_x: .75
        size_hint_y: None
        pos_hint: {'center_x': .5,'center_y': .125}
        md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
        text_color: (1,1,1,1) if theme.active else (0,0,0,1)
        on_press:
            root.transition.direction = 'left'
            root.current = 'spinner_screen'

    MDFlatButton:
        font_style: "Button"
        text: "Назад" if lang.active else "Back"
        size_hint_x: .25
        size_hint_y: None
        pos_hint: {'center_x': .5,'center_y': .05}
        md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
        text_color: (1,1,1,1) if theme.active else (0,0,0,1)
        on_press:
            root.transition.direction = 'right'
            root.current = 'subjects'

MDScreen:
    name: 'spinner_screen_database'
    md_bg_color: app.theme_cls.bg_light
    on_enter:
        app.callback_button_collect()

MDLabel:
    id: spinner_label_database
    font_style: "Caption"
    text: "Загрузка данных\n..." if lang.active else "Loading data\n..."
    valign: 'top'
    halign: 'center'
    pos_hint: {"center_x": .5, "center_y": .5}

MDScreen:
    id: data_scr
    name: 'table'
    md_bg_color: app.theme_cls.bg_light
    on_pre_enter:
        image3.anim_delay = -1

MDBBoxLayout:
    orientation: "vertical"
    size_hint_x: 1
    adaptive_height: True
    pos_hint: {"center_x": .5, "center_y": .5}
    md_bg_color: app.theme_cls.bg_light

MDBBoxLayout:
    id: table_box
    size_hint_x: 1
    adaptive_height: True

```

```

    pos_hint: {"center_x": .5, "center_y": .5}
    md_bg_color: app.theme_cls.bg_light

    MDBBoxLayout:
        orientation: "vertical"
        adaptive_height: True
        pos_hint: {"center_x": .5, "center_y": .5}
        md_bg_color: app.theme_cls.bg_light

    MDBBoxLayout:
        orientation: "horizontal"
        adaptive_height: True
        pos_hint: {"center_x": .5, "center_y": .5}
        md_bg_color: app.theme_cls.bg_light

    MDFillRoundFlatButton:
        font_style: "Button"
        text: "Назад" if lang.active else "Back"
        size_hint_x: .5
        size_hint_y: None
        pos_hint: {"center_x": .25, "center_y": .5}
        md_bg_color: (138.0/255.0, 161.0/255.0, 107.0/255.0, .5)
        text_color: (1,1,1,1) if theme.active else (0,0,0,1)
        on_press:
            root.transition.direction = 'right'
            root.current = 'processing'

    MDFillRoundFlatButton:
        font_style: "Button"
        text: "Обработка" if lang.active else "Collect"
        size_hint_x: .5
        size_hint_y: None
        pos_hint: {"center_x": .75, "center_y": .5}
        md_bg_color: (138.0/255.0, 161.0/255.0, 107.0/255.0, .5)
        text_color: (1,1,1,1) if theme.active else (0,0,0,1)
        on_press:
            root.transition.direction = 'left'
            root.current = 'spinner_screen_database'

    MDScreen:
        name: 'check'
        md_bg_color: app.theme_cls.bg_light
        on_enter:
            image4.anim_delay = 1/30
            image4.anim_loop = 1

    MDFloatLayout:

        Image:
            id: image4
            source: "assets/check_another.gif" if theme.active else
"assets/check.gif"
            pos_hint: {"x": 0, "y": 0.1}

        MDLabel:
            font_style: "H6"
            text: "Завершено!" if lang.active else "Finished!"
            pos_hint: {"center_x": .5, "center_y": .9}
            valign: 'middle'
            halign: 'center'

        MDFillRoundFlatButton:
            font_style: "Button"
            text: "Выход" if lang.active else "Exit"

```

```
size_hint_x: .75
size_hint_y: None
pos_hint: {"center_x": .5, "center_y": .125}
md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
text_color: (1,1,1,1) if theme.active else (0,0,0,1)
on_press: app.stop()

MDFillRoundFlatButton:
    font_style: "Button"
    text: "Загрузка" if lang.active else "Upload"
    size_hint_x: .25
    size_hint_y: None
    pos_hint: {"center_x": .5, "center_y": .05}
    md_bg_color: (138.0/255.0,161.0/255.0,107.0/255.0,.5)
    text_color: (1,1,1,1) if theme.active else (0,0,0,1)
    on_press:
        root.transition.direction = 'right'
        app.uploading()
```

# ПРИЛОЖЕНИЕ Д. ТЕКСТ ВЕБ-СЕРВИСА ВИЗУАЛИЗАЦИИ ДАННЫХ

*File: website.py*

```
import streamlit as st
from google.oauth2 import service_account
from google.cloud import bigquery
import pandas as pd
import seaborn as sns

st.set_page_config(page_title="Student Digitizer",
                    page_icon='⚙',
                    layout="centered",
                    initial_sidebar_state="collapsed",
                    menu_items=None)

st.title("Student Digitizer")

with st.expander("About"):
    st.text("""
        Project for developing program system for analyzing digital student's
        footprint.

        FAQ:
        1. Select item from list of available groups
        2. The website will show values that chosen group contains
        3. After that results will be printed as visualization
        4. Left column indicates about students activity,
           right column shows attendance per subjects
        5. For better efficiency between BigQuery and StreamLit
           current query saved in cache for 1 hour

        Developer:
        Dmitry Moskalev
    """)

    st.image("assets/logo.png", caption='Student Digitizer')

# Create API client.
credentials = service_account.Credentials.from_service_account_info(
    st.secrets.gcp_service_account
)
client = bigquery.Client(credentials=credentials)

# Using st.cache to only rerun when the query changes or after 60 min.
@st.cache(ttl=3600, suppress_st_warning=True, allow_output_mutation=True)
def df():
    query_df_students = client.query(f"SELECT lectures_all, `group`, week_n FROM
`{st.secrets.table_id_1.table_1}`")
    rows_raw_students = query_df_students.result()
    table_df_students = [dict(row) for row in rows_raw_students]
    df_students = pd.DataFrame()
    df_students = df_students.append(table_df_students)

    query_job_subjects = client.query(f"SELECT subject, `group`, week_n, total
FROM `'{st.secrets.table_id_2.table_2}``")
    rows_raw_subjects = query_job_subjects.result()
    table_df_subjects = [dict(row) for row in rows_raw_subjects]
    df_subjects = pd.DataFrame()
    df_subjects = df_subjects.append(table_df_subjects)
    return df_students, df_subjects
```

```

df_students, df_subjects = df()

group = st.selectbox('Select group', set(df_subjects['group']))
st.write(f"Values for {group}: {sorted(set(df_students[df_students['group'] == group]['week_n']))}")

df_students_stripplot = sns.stripplot(x=df_students[df_students['group'] == group]["week_n"],
                                         y=df_students[df_students['group'] == group]["lectures_all"],
                                         data=df_students[df_students['group'] == group])

df_students_pairplot = sns.pairplot(df_students[df_students['group'] == group],
                                     hue="lectures_all")
df_subjects_pairplot = sns.pairplot(df_subjects[df_subjects['group'] == group],
                                     hue="total")

graphic = df_subjects[df_subjects['group'] == group].plot.scatter(x='week_n',
                                                                    y='total',
                                                                    c='week_n',
                                                                    colormap='viridis')

st.info("Students and subjects distributions")

try:
    st.write('Distribution of missing classes by each student for all subjects per week')
    st.pyplot(df_students_stripplot.figure)

    st.write("""
Count of students that missed each subject in chosen week.\n
Each point means different subject.""")
    st.pyplot(graphic.figure)

    col1, col2 = st.columns([1.0, 1.0])

    with col1:
        st.write('Missing class percentage by students')
        st.pyplot(df_students_pairplot.figure)

    with col2:
        st.write("Percentage of subjects count missed by students")
        st.pyplot(df_subjects_pairplot.figure)
except ValueError:
    st.info("Error")

```