

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

СИСТЕМА ПРОГНОЗИРОВАНИЯ
АКАДЕМИЧЕСКИХ РИСКОВ УЧАЩИХСЯ

Студент

Бондарь 22.05.2026
(подпись, дата)

Бондарь Ксения Владимировна
(ФИО полностью)

Руководитель

М.П. Синев 22.05.2026
(подпись, дата)

М.П. Синев
(фамилия, инициалы)

Нормоконтролёр

А.С. Бычков 22.05.2026
(подпись, дата)

А.С. Бычков
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой

М.А. Митрохин
(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

02 июня 2026, № 3)

Секретарь ГЭК

Е.И. Гурин

Е.И. Гурин
(подпись)

(фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 07 » 04 20 26 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Бондарь Ксения Владимировна гр. 22ВВП1 факультета вычислительной техники

2. Тема работы «Система прогнозирования академических рисков учащихся»

Тема утверждена приказом ПГУ № 347/0 от " 07 " 04 20 26 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ Синев М.П.

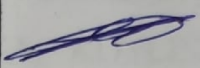
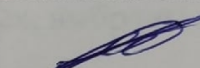
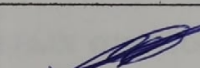
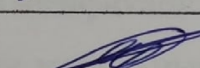
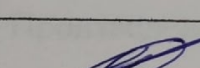
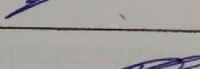
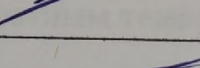
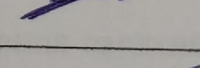
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать мобильное приложение, помогающее предупреждать академические риски учащихся на раннем этапе. Приложение должно осуществлять учет общей успеваемости ученика, а также предоставлять список конкретных тем, по которым прогнозируются ближайшие неудовлетворительные результаты.

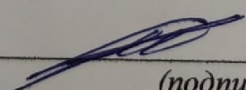
5. Перечень подлежащих разработке вопросов

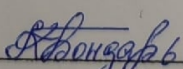
В рамках работы необходимо выполнить аналитический обзор существующих решений, обосновать выбор технологий разработки. Спроектировать архитектуру мобильного приложения и базы данных. Требуется обучить модели логистической регрессии для прогнозирования неудовлетворительных отметок по будущим темам, обучить модель KNN (к ближайших соседей) для классификации учащихся по уровню успеваемости. Также необходимо разработать мобильный интерфейс. Интегрировать ML-модели в Android-приложение, провести тестирование и отладку.

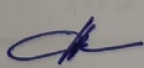
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	18	07.04.2026 – 08.04.2026	
2	Анализ аналогов	22	9.04.2026 – 10.04.2026	
3	Постановка задачи	15	11.04.2026 – 12.04.2026	
4	Выбор и настройка инструментов разработки	17	13.04.2026 – 15.04.2026	
5	Разработка архитектуры приложения	28	16.04.2026 – 18.04.2026	
6	Разработка архитектуры базы данных	35	19.04.2026 – 23.04.2026	
	Обучение ML-моделей	55	24.04.2026 – 30.04.2026	
7	Разработка пользовательского интерфейса	45	01.05.2026 – 07.05.2026	
8	Тестирование приложения	33	08.05.2026 – 13.05.2026	
9	Описание раздела БЖД	17	14.05.2026 – 17.05.2026	
10	Оформление пояснительной записки	39	18.05.2026 – 21.05.2026	

Дата выдачи задания " 07 " 04 20 26 г.

Руководитель бакалаврской работы  07.04.26 Синев М.П.
(подпись, дата)

Задание к исполнению принял студент  07.04.26 Бондарь К.В.
(подпись, дата)

Работу к защите допустить
Декан факультета  27.06.2020 Фионова Л.Р.

Реферат

Бакалаврская работа содержит 139 листов, 45 рисунков, 9 таблиц, 18 источников и 3 приложения.

АНАЛИЗ ДАННЫХ, МЕТОД К-БЛИЖАЙШИХ СОСЕДЕЙ, ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ, МОБИЛЬНАЯ РАЗРАБОТКА, KOTLIN, PYTHON, FIRESTOR

Цель работы: разработать мобильное приложение, систематизирующее информацию по учебным факторам учащихся и помогающее выявить академические риски учеников.

Для достижения цели были выполнены следующие работы: спроектирована архитектура мобильного приложения и базы данных, выбраны и обработаны датасеты, использующиеся для обучения, обучены модели логистической регрессии для прогнозирования неудовлетворительных отметок по будущим темам и модель KNN (k ближайших соседей) для классификации учащихся по уровню успеваемости. Также необходимо разработан мобильный интерфейс. ML-модели интегрированы в Android-приложение. Произведено тестирование и отладка приложения.

В результате проделанной работы разработано мобильное приложение, с функциями предсказания неудовлетворительных отметок по конкретным темам и определения общего уровня ученика на основе таких признаков, как количество сна, количество часов изучения, регулярность посещения занятий, социально-экономическое окружение.

					ПГУ1.090301.4.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Система прогнозирования академических рисков учащихся Пояснительная записка	Лит.	Лист	Листов
Разраб.		Бондарь К.В.		22.05.23				
Провер.		Синев М.П.		22.05.23				
Н. Контр		Бычков А.С.		25.05.23				
Утверд.		Митрохин М.А.		25.05.23				
						ФВТ, каф. ВТ		
						гр. 22ВВП1		

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Анализ темы	7
1.1 Анализ предметной области	7
1.2 Анализ аналогов	8
1.1.1 Цифровая образовательная платформа ЭлЖур	8
1.1.2 Московская электронная школа	11
1.1.3 Выводы	15
2 Постановка задачи	16
3. Выбор и настройка инструментов разработки	18
3.1 Выбор СУБД	18
3.2 Выбор инструментов разработки серверной части	20
3.2.1 Выбор языка разработки	20
3.2.2 Выбор среды разработки	21
3.3 Выбор инструментов разработки клиентской части	24
3.3.1 Выбор языка разработки	24
3.3.2 Выбор среды разработки	26
4 Архитектура системы	32
4.1 Архитектура приложения	32
4.2 Архитектура базы данных	34
5 Разработка приложения	37
5.1 Система прогнозирования неудовлетворительных отметок учащихся	41
5.2 Классификатор успеваемости учащихся	47
6 Разработка пользовательского интерфейса	55
7 Тестирование	67
7.1 Тест-кейс «Выставление отметки и получение прогноза»	67

7.2 Тест-кейс «Удаление отметки и получение прогноза»	67
7.3 Тест-кейс «Оценка факторов, влияющих на учебный прогресс ученика» ...	68
8 Безопасность жизнедеятельности	70
8.1 Классификация рабочих мест	70
8.2 Санитарно-гигиенические требования	71
8.3 Электробезопасность	72
8.4 Пожарная безопасность	73
8.5 Расчет искусственного освещения	73
Приложение А – UML-диаграммы	80
Приложение Б – Листинг мобильного приложения	85
Приложение В – Презентация	132

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

АИС – автоматизированная информационная система

БД – база данных

КТП – календарно-тематический план

ОС – операционная система

СУБД – система управления базами данных

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста

IDE (Integrated Development Environment) – интегрированная среда разработки

JVM (Java Virtual Machine) – виртуальная машина Java

SDK (Software Development Kit) – набор инструментов для разработки программного обеспечения

					ПГУ1.090301.4.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Цифровые технологии стремительно развиваются и используются в различных областях человеческой жизни. Образовательная сфера не стала исключением. Использование электронных систем позволило систематизировать большие объёмы информации, внести новые возможности в учебный процесс как для преподавателей, так и для учащихся и их родителей.

Помимо структурирования данных всё чаще начали применяться попытки автоматизации процесса образования. Одним из вариантов уменьшить человеческие трудозатраты стало введение систем на основе искусственного интеллекта. А именно системы прогнозирования получили свое развитие в части предсказания учебных успехов учащихся.

Система прогнозирования на основе предшествующих данных позволяет предсказать будущие тенденции. Так, на основе полученных отметок учеником можно сделать выводы о дальнейшей успеваемости учащегося. Действительно, один из важнейших показателей качества обучения являются отметки ученика, однако, помимо этого, следует учитывать различные внешние факторы, взаимосвязанные с образовательным процессом: возраст, пол, состояние здоровья, внеучебная деятельность обучающегося, качество подготовки к занятиям [1].

В учебной среде системы прогнозирования активно пытаются использовать для выстраивания индивидуальных планов обучения учащихся. Выдача персональных рекомендаций для учащихся способствует лучшему усвоению темы и повышению результатов обучения, так как принимаются во внимание личные способности обучающегося.

					ПГУ1.090301.4.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ темы

Большая нагрузка учителя складывается из количества учеников в классе, умноженное на число классов, где он преподаёт. Для того, чтобы уделить внимание каждому учащемуся требуется принимать во внимание большое количество факторов во время учебного процесса. Электронные сервисы помогают снизить нагрузку на преподавателя и повысить образовательные результаты.

1.1 Анализ предметной области

Образовательный процесс – это сложная многоэтапная система, являющаяся неотъемлемой частью жизни каждого человека. Выбранная предметная область будет всегда актуальна, что подтверждают слова мэра Москвы С.С. Собянина: «Образование – это наше настоящее, наше будущее. От того, какое образование получит молодое поколение, зависит будущее и Москвы, и нашей страны» [2].

Именно учителя вносят большой вклад в воспитание и обучение молодого поколения. В повседневной работе учитель выполняет колоссальное количество задач, к которым относятся:

- анализ и подбор учебных материалов для урока;
- подготовка интерактивных материалов к уроку;
- проведение уроков (в том числе дополнительных и внеурочных занятий);
- проверка результатов деятельности ученика различных видов (активность в классе на уроке, домашние задания, проверочные и контрольные работы, дополнительная и внеурочная деятельность);
- ведение бумажной работы;
- проведение родительских собраний.

Нередко работа учителя совмещается с классным руководством. Поэтому к вышеперечисленным задачам прибавляются

- контроль успеваемости учащихся по различным предметам;
- контроль посещения занятий учащихся;

– сохранение постоянной связи с родителями/законными представителями учащихся.

Необходимо понимать, что учитель помимо немалой учебной нагрузки должен сформировать в учащихся правильные личностные навыки. Для этого преподаватель обязан демонстрировать обучающимся такие качества как организованность, терпение, трудолюбие, аккуратность, умение владеть собой.

Кроме этого, учитель занимает наблюдательную позицию для изучения особенностей коммуникации, способностей, индивидуальных качеств ребёнка для создания комфортной среды обучения и возможности мотивации и большего вовлечения учащегося в учебный процесс.

Таким образом, внедрение в работу учителя цифровых технологий не должно увеличить нагрузку на преподавателя. Важно обеспечить удобство и лёгкость использования электронной системы для учителя. А также упростить уже имеющиеся ресурсозатратные процессы в работе учителя.

1.2 Анализ аналогов

Появление электронного документооборота повлекло развитие различных образовательных систем с целью их введения в работу учителя. Рассмотрим подобные продукты и выделим особенности каждого из них.

1.1.1 Цифровая образовательная платформа ЭлЖур

Цифровая образовательная платформа ЭлЖур [3] объединяет в единую экосистему различные сервисы: «Цифровые образовательные ресурсы», «Типовые сайты образовательных организаций», «Печать журналов и аттестатов», «ЭлЖур.Видео», «Электронный дневник» и «Электронный журнал». С 2010 года АИС «Электронный журнал ЭлЖур» остаётся цифровым ядром платформы, которое востребовано среди администраций школ и учителей.

Изучим инструкцию для учителя и классного руководителя системы «Электронный журнал ЭлЖур» [4]. Раздел «Журналы» включает в себя различный функционал от работы с журналами класса до обширного анализа успеваемости

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

учащихся. Помимо базовых возможностей раздела можно отметить реализацию функции информирования о просмотренных отметках учащегося родителями. Для журналов класса введена возможность включить/выключить индикацию для просмотренных отметок. Этот механизм осуществляется посредством нажатия на фразу «Выделить отметки, просмотренные родителями». При включении функции клетки соответствующих отметок закрашиваются в оранжевый цвет. Выключение функции журнальная сетка имеет единую серо-белую цветовую гамму. Включение подсветки отметок, просмотренных родителями, представлено на рисунке 1.

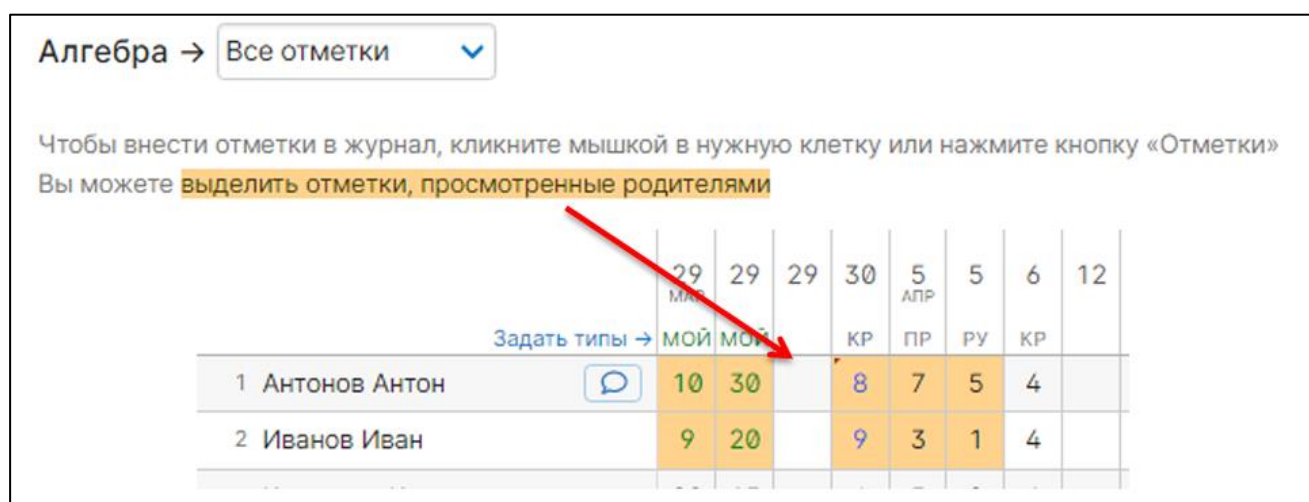


Рисунок 1 – Включенная функция подсветки отметок, просмотренных родителями

Раздел «Журналы» отличается способностью отображения различной графической аналитики, которая способствует быстрому установлению закономерностей и лёгкому восприятию информации. Диаграммы и графики строятся для отображения статистических сведений:

- по контрольным работам показана на рисунке 2;
- по успеваемости за учебные периоды показана на рисунке рисунок 3;
- по достижению целей обучающимися показана на рисунке рисунок 4.

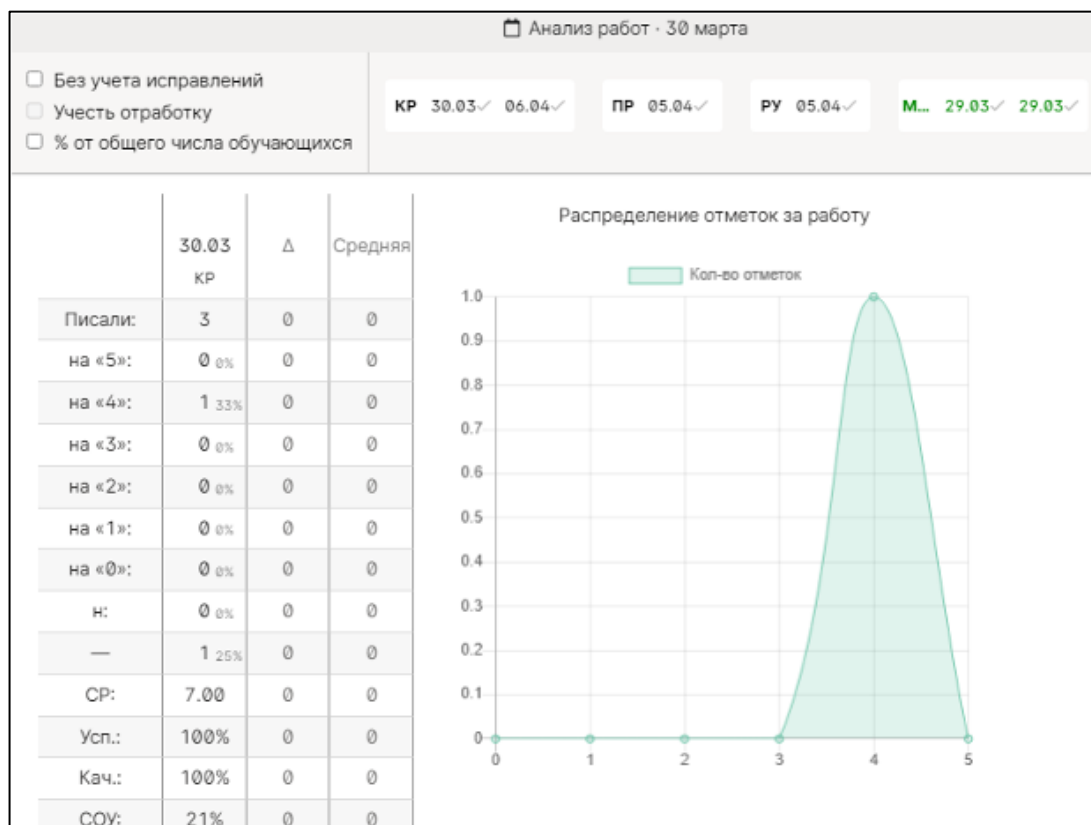


Рисунок 2 – Анализ контрольных работ учащихся

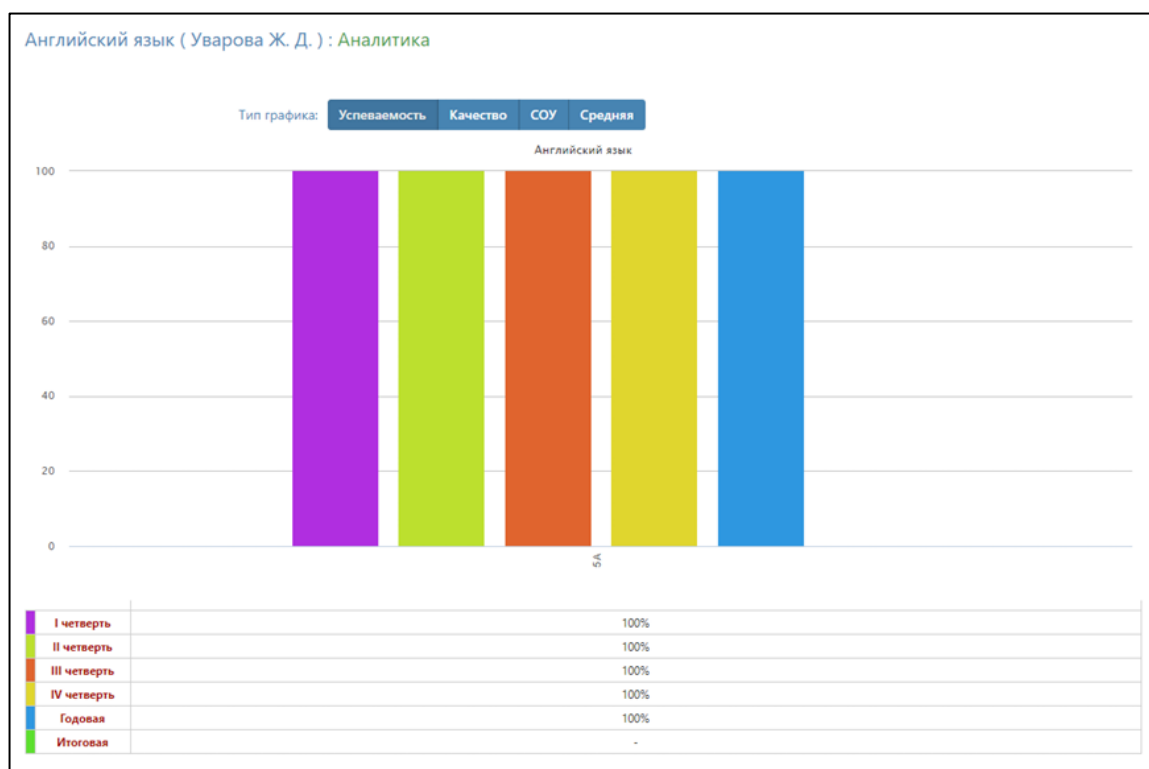


Рисунок 3 – Анализ результатов учащихся за учебные периоды

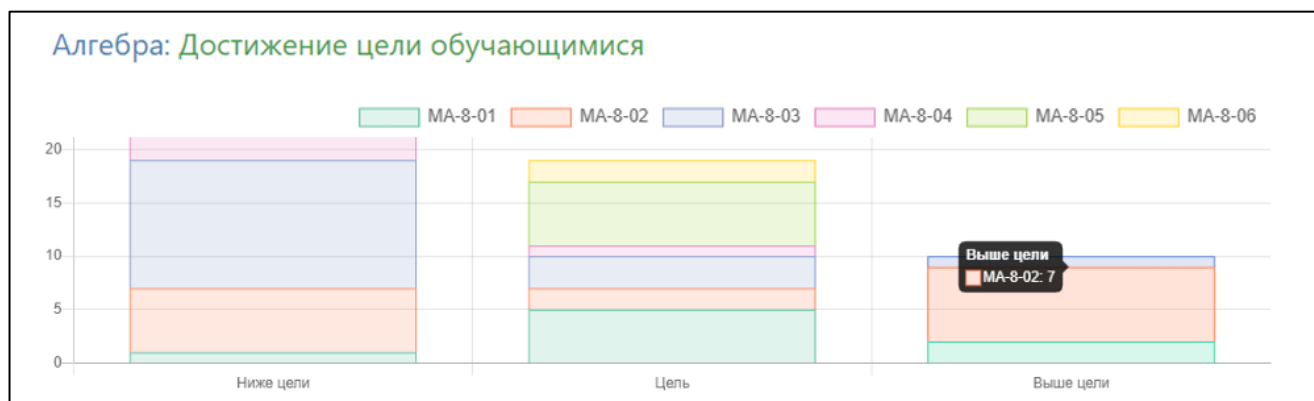


Рисунок 4 – Анализ достижения целей обучающимися

Цели по каждому предмету задаются завучами или учащимися. Это вносит дополнительную возможность сравнить предполагаемую отметку и фактическую. Однако электронная система предусматривает ручной ввод прогнозируемой отметки, что не является оптимальным вариантом отслеживания успеваемости учащегося. На формирование предполагаемой отметки учащегося может влиять субъективное мнение завуча или некорректная самооценка учащегося. Автоматизация рассматриваемой функции позволила бы наиболее точно производить аналитику успеваемости учащегося.

1.1.2 Московская электронная школа

Проект «Московская электронная школа» начал развиваться в 2018 году и на сегодняшний день охватывает большое количество сервисов» «Портфолио учителя», «Портфолио учащегося», «Библиотека МЭШ», «Олимпиады», «Москвенок», «Колледж МЭШ», «Электронный дневник», а также «Электронный журнал» [5]. Рассмотрим мобильное приложение «Электронный журнал», созданное для преподавателей школы и классных руководителей. Ссылка на скачивание приложения из RuStore: <https://www.rustore.ru/catalog/app/ru.mos.school.journal>.

Для отслеживания успеваемости учащегося в приложении предусмотрен специальный функционал. Для каждого учащегося есть возможность просмотра признаков, требующих обратить внимание на учебу ученика:

– прогнозируется академическая задолженность;

- есть задолженность по контрольным;
- низкая накапливаемость отметок;
- пограничная отметка;
- точка скоро станет отметкой;
- высокий процент пропусков;
- уменьшился средний балл учащегося;
- увеличился средний балл учащегося.

Для каждого учащегося, на которого стоит обратить особое внимание, можно просмотреть конкретную проблему, которая отображается в модальном окне «Актуальное на сегодня», показанном на рисунке 5.

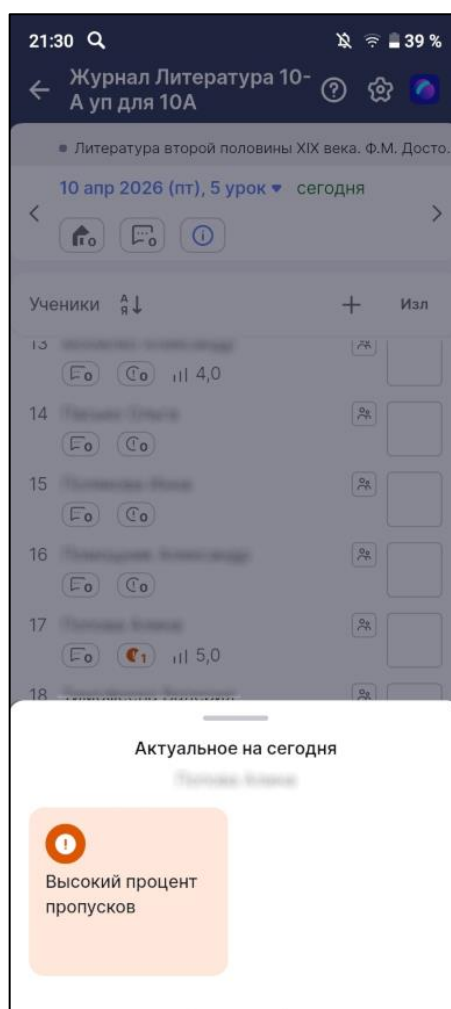


Рисунок 5 – Модальное окно «Актуальное на сегодня»

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

Также в приложении можно открыть страницу, показанную на рисунке 6, где собраны проблемы успеваемости всего класса по предмету.

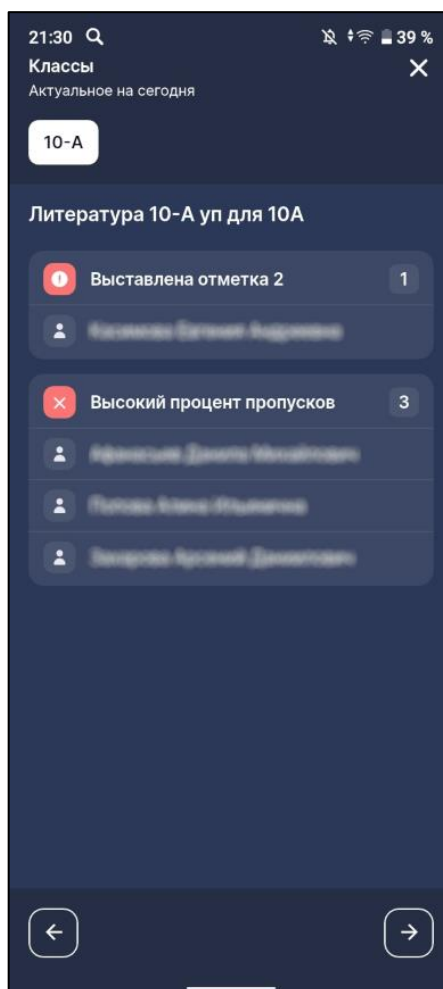


Рисунок 6 – Экранная форма «Актуальное на сегодня» для всего класса

Кроме этого, в приложении реализован отдельный раздел «Учебные умения», позволяющий оценить навыки учащегося, требуемые для успешного обучения. Раздел состоит из нескольких групп вопросов «Самоорганизация», «Самообразование», «Саморегуляция», «Коммуникация». Учитель, проводя оценку учащегося каждый учебных период, способен вести статистику по развитию учебных умений ученика, влияющих на обучение учащегося. Процесс оценивания ученика представлен на рисунке 7.

21:42 🔍 📶 37 %

← Оценка учебных умений

Иванов Иван Иванович

Саморегуляция · 3 вопроса

Работает внимательно, легко переключается между разными задачами

☐ Никогда
☐ Редко
☒ Часто
☐ Всегда
☐ Затрудняюсь ответить

Оценивает свою работу и результаты объективно

☐ Никогда
☐ Редко
☐ Часто
☒ Всегда
☐ Затрудняюсь ответить

Регулирует свое поведение, соблюдает дисциплину

← Назад Далее →

Рисунок 7 – Экранная форма «Оценка учебных умений»

Реализация индивидуальной аналитики конкретного учащегося в мобильном приложении с интуитивно понятным интерфейсом открывает доступ к информации в любой момент времени, а также позволяет оперативно провести анализ и выявить неудовлетворительные результаты учащегося. Однако можно заметить, что в приложении производится индикация общего неудовлетворительного результата обучения учащегося, конкретизация отдельных тем, по которым выявлены отрицательные показатели, отсутствует.

1.1.3 Выводы

Таким образом, после проведения анализа электронных систем, которыми пользуются преподаватели, были выявлены положительные паттерны, которые целесообразно использовать в создаваемой системе прогнозирования рисков у учащихся:

- быстрый доступ к прогнозам неудовлетворительных результатов учащегося;
- учет различных факторов, влияющих на образовательный процесс, учащегося;
- использование яркой и понятной индикации для выделения отметок;
- быстрый доступ к информации об уроке (дата, тема);

					ПГУ1.090301.4.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

2 Постановка задачи

Требуется разработать систему, которая будет включать две основные функции:

- оценка факторов, влияющих на учебный процесс – целесообразно и более эффективно использовать в начале обучения;
- выявление тем, по которым прогнозируются неудовлетворительные отметки – применять во время учебного процесса.

В начале учебного года при отсутствии отметок предлагается производить анкетирование учащихся по факторам, влияющим на их учебу (количество часов сна, количество часов обучения, регулярность посещений занятий, социально-экономический уровень учащегося). Результаты анкетирования позволят преподавателям с самого начала обучения обратить внимание на учащихся, для которых прогнозируются неудовлетворительные результаты. Аккумулировать ответы всех учащихся и понятно визуализировать позволит электронная система.

Во время образовательного процесса отслеживание учебного прогресса рекомендуется производить, принимая во внимание прогнозы неудовлетворительных отметок по темам. Система прогнозирования предупредит о возможности получения неудовлетворительной отметки по конкретной теме, следовательно, на эту тему требуется обратить особое внимание.

Диаграмма вариантов использования приложения Пользователем представлена на рисунке 8.

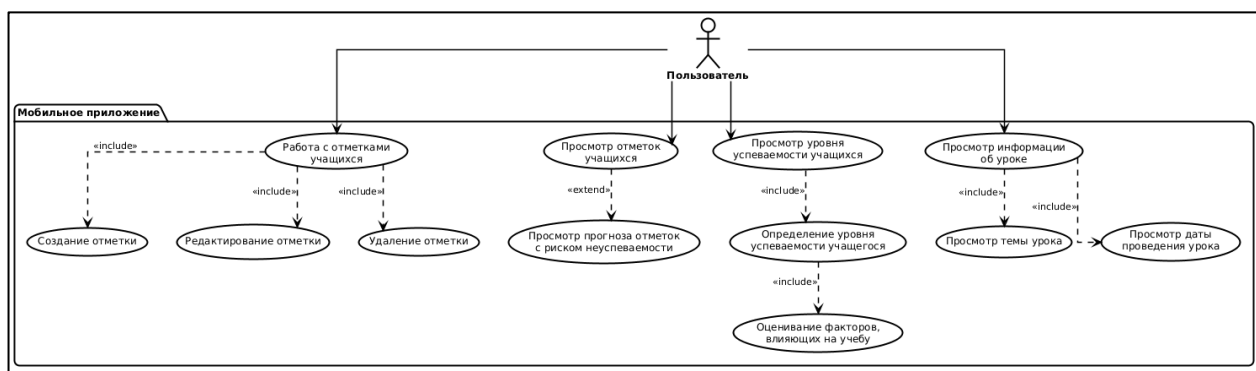


Рисунок 8 – Диаграмма вариантов использования приложения

Таким образом, требуется реализовать:

– классификатор учеников по уровню образования (низкий, средний, высокий уровень образования у учащегося) на основе факторов, влияющих на образовательный процесс;

– систему прогнозирования неудовлетворительно усвоенных тем индивидуально для учащегося с учетом оценок, полученных учеником ранее.

Технические требования к устройству:

– версия Android не ниже 7;

– наличие свободного места на устройстве в размере 40 Мб.

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

3 Выбор и настройка инструментов разработки

Электронные сервисы помогают собрать данные по каждому учащемуся в единое целое для удобства преподавателя. Более того, на сегодняшний день электронные образовательные платформы стали развиваться не только в виде десктопных версий, но и в виде мобильных приложений. Мобильные приложения отличаются своей оперативностью работы, так как остаются доступны всегда под рукой, в отличие от десктопной версии. Поэтому было принято решение о создании системы прогнозирования рисков учащихся в виде мобильного приложения.

3.1 Выбор СУБД

Для работы с данными была выбрана платформа Firebase в качестве СУБД. Firebase [6] предоставляет различные сервисы, обеспечивающие

- управление авторизацией пользователей;
- NoSQL-база данных Cloud Firestore;
- облачное хранилище Cloud Storage;
- система отслеживания падений приложения Crashlytics;
- A/B-тестирование;
- сбор статистики приложения и её аналитика.

Удаленные вычисления и хранилище, предоставляемые платформой Farebase обеспечивают беспрепятственное внедрение, развертывание, масштабирование и управление удаленной инфраструктурой приложения.

Рассмотрим более подробно сервис, обеспечивающий бессерверную базу данных NoSQL – Cloud Firestore. Cloud Firestore выполняет функции хранения данных, синхронизации в реальном времени, автоматического масштабирования, сортировки и фильтрации данных. Таким образом, Cloud Firestore хорошо подходит для проектов, требующих обновлений в реальном времени и совместной работы [7].

Помимо этого, Cloud Firestore функционирует в офлайн-режиме, продолжая работу после повторного подключения к Интернету. Мобильные устройства могут терять сетевое соединение на произвольные промежутки времени. Поведение приложения в отключенном состоянии часто может быть ключевым фактором его успеха. Сервис базы данных Cloud Firestore в сочетании с библиотеками клиентского SDK Firebase поддерживает полностью автономную работу с автоматической синхронизацией при повторном подключении.

Благодаря бессхемной структуре Cloud Firestore организует данные с помощью документов и коллекций. Каждый документ представляет собой объект JSON, содержащий поля и значения, а коллекции группируют связанные документы [8].

Для созданного проекта TeacherJournal на платформе Firebase были выданы уникальный идентификатор и номер проекта, показанные на рисунке 9. Уникальный идентификатор проекта способствует лёгкому распознаванию проекта и может появляться в URL-адресах или названиях некоторых ресурсов Firebase. Номер проекта используется внутренними системами Google.

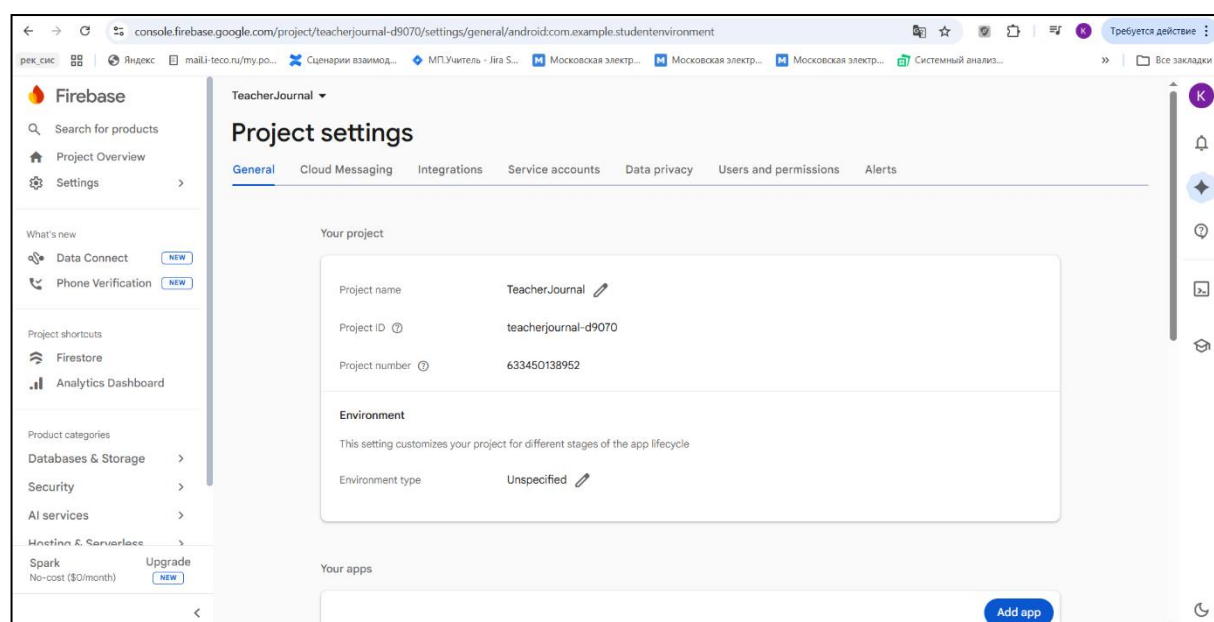


Рисунок 9 – Создание проекта на платформе Firebase

Для базы данных была создана индексация для оптимизации и повышения скорости работы приложения. Составной индекс коллекции lessons по нескольким полям позволят производить ускоренную сортировку по классу, дате занятия и имени учащихся. Активность индексации продемонстрирована на рисунке 10.

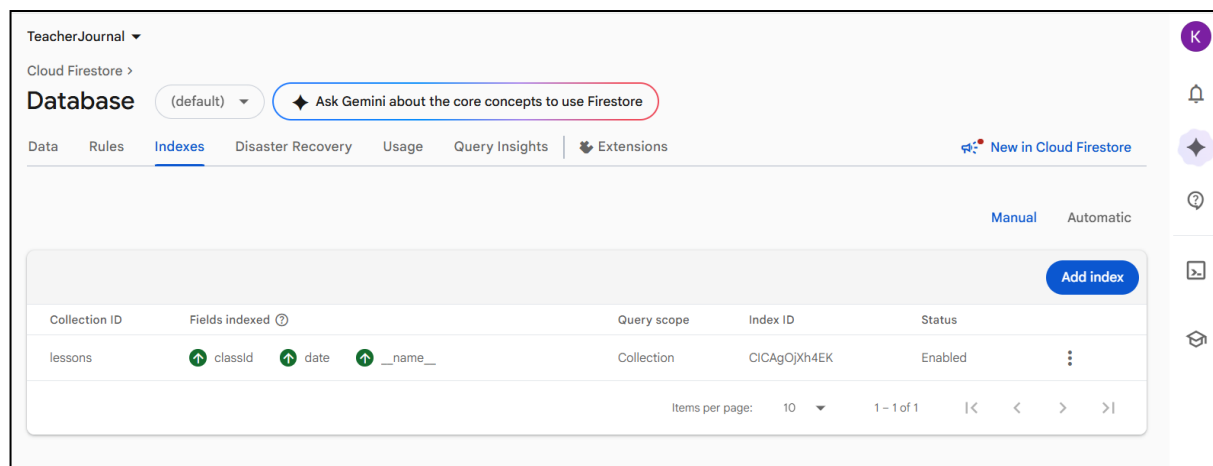


Рисунок 10 – Создание составной индексации данных

3.2 Выбор инструментов разработки серверной части

3.2.1 Выбор языка разработки

Для обработки данных на серверной стороне приложения был выбран язык Python. Python появился в 1991 году и был разработан Гвидо ван Россумом. Рассматриваемый язык программирования является объектно-ориентированным, интерактивным и интерпретируемым. Python включает в себе унифицированный набор принципов многих других языков программирования.

Отличительной чертой языка является — динамическая типизация переменных. Тип данных определяется во время выполнения программы. Важным фактором при написании кода является его форматирование при помощи отступов. Для завершения инструкции используется перенос каретки на новую строку. Такой подход имеет свои преимущества и недостатки. Несмотря на то, что код на Python легче читается из-за ясности и структурирования блоков кода, даже одна ошибка форматирования приводит к ошибке в программе.

Стандартная библиотека Python содержит большое количество встроенных пакетов и модулей, но, несмотря на это, в открытом доступе существует еще больше библиотек и модулей, позволяющих расширить его функциональность. Расширяемость и простота сделали Python распространённым и удобным в использовании [9].

Python является подходящим для системного программирования, так как язык содержит модули, поддерживающие работу со службами операционной системы. Стандартная библиотека Python содержит ряд инструментов для работы с системными ресурсами: переменные среды, файлы, сокеты, каналы, процессы, командная строка, стандартные интерфейсы потоков и т. д.

Python – язык с открытым исходным кодом, что подтверждает его свободное использование и распространение. Python применяется для создания веб-приложений, обработки данных, взаимодействия с базами данных, разработки пользовательских интерфейсов, машинного обучения, написания скриптов, визуализации и т. д.

3.2.2 Выбор среды разработки

3.2.2.1 Spyder

Spyder — это бесплатная кроссплатформенная среда разработки на Python, интегрированная в Anaconda. Она поддерживает различные функции: подсветка синтаксиса, автодополнение кода, анализ и подсветка потенциальных ошибок, отладка как с пошаговым выполнением, так и с точками останова. Помимо этого, предоставляется встроенная консоль, табличное отображение используемых переменных в программе и визуализация различной графической статистики.

Кроме того, Spyder предлагает встроенную интеграцию с различными универсальными пакетами: Pandas, SciPy, NumPy, IPython, QtConsole, Matplotlib и многими другими. Spyder можно использовать как расширение библиотеки PyQt5, что позволяет пользователям настраивать функциональность среды, а также интегрировать её компоненты в собственное программное обеспечение.

3.2.2.2 PyCharm

PyCharm – интегрированная среда разработки (IDE), позволяющая редактировать текст, отлаживать и полностью интерпретировать код Python. PyCharm подходит для работы над проектом, содержащим множество взаимосвязанных друг с другом скриптов. PyCharm решает обширный круг задач – от разработки веб-сайтов до анализа данных [10]. Однако PyCharm может запрашивать большое количество ресурсов в виде большого объема памяти и места для хранения. К отличительным особенностям PyCharm относятся:

- возможность создания различных сред, позволяющих разделять зависимости, требующиеся для разных проектов;
- возможность рефакторинга кода для безопасной реструктуризации: переименование, извлечение метода, встраивание локальной переменной и т.д.;
- удобная навигация по коду и возможность создания комментариев для напоминания о необходимости внесения изменений в дальнейших сеансах работы;
- встроенное модульное тестирование;
- возможность использования плагинов для добавления дополнительного функционала с среду разработки.

3.2.2.3 Google Colab

Google Colab — это бесплатная облачная среда разработки от компании Google. Google Colab предоставляет мощную и простую платформу для разработки прогностических моделей. Использование Google Colab совместно с TensorFlow, Keras и библиотеками машинного обучения позволяет упростить реализацию сложных систем, ускорить обучение моделей и время их выполнения. Поточковая передача данных и методы предварительной обработки, такие как очистка и нормализация данных, в Google Colab позволяют повысить качество моделирования.

Наличие бесплатных графических процессоров повышает производительность нейронных сетей и глубокого обучения, обеспечивая более высокую скорость обработки по сравнению с персональными компьютерами. Это

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

делает Google Colab оптимальным инструментом для внедрения передовых методов искусственного интеллекта без необходимости наличия мощной локальной инфраструктуры.

Google Colab обеспечивает интеграцию с Google Drive и другими облачными хранилищами, что обеспечивает простой доступ к большим наборам данных. Кроме того, использование библиотек для распределённой обработки больших наборов данных, таких как PySpark, способно производить работу с большими данными, которые традиционно требовали бы выделения ресурсов в виде кластеров [11].

После анализа сред разработки для написания внутренней логики приложения был выбран Google Colab. Для работы с данными, хранящимися на платформе Firebase, необходимо произвести интеграцию. Для возможности управления Firebase на языке Python была выполнена команда `!pip install firebase-admin` [12], результат которой представлен на рисунке 11.

```

student_env.ipynb
Файл Изменить Вид Вставка Среда выполнения Инструменты Справка
Команды + Код + Текст ▶ Выполнить все
!pip install firebase-admin
Requirement already satisfied: firebase-admin in /usr/local/lib/python3.12/dist-packages (6.9.0)
Requirement already satisfied: cachecontrol>0.12.14 in /usr/local/lib/python3.12/dist-packages (from firebase-admin) (0.14.4)
Requirement already satisfied: google-api-python-client>1.7.8 in /usr/local/lib/python3.12/dist-packages (from firebase-admin) (2.104.0)
Requirement already satisfied: google-cloud-storage>1.37.1 in /usr/local/lib/python3.12/dist-packages (from firebase-admin) (3.10.1)
Requirement already satisfied: pyjwt>2.5.0 in /usr/local/lib/python3.12/dist-packages (from http[https]>=0.28.1->firebase-admin) (2.12.1)
Requirement already satisfied: httpx>0.28.1 in /usr/local/lib/python3.12/dist-packages (from http[https]>=0.28.1->firebase-admin) (0.28.1)
Requirement already satisfied: google-api-core<3.0.0dev,>=1.22.1 in /usr/local/lib/python3.12/dist-packages (from google-api-core[grpc]<3.0.0dev,>=1.22.1; platform_python_implementation != "PyPy")
Requirement already satisfied: google-cloud-firestore>2.19.0 in /usr/local/lib/python3.12/dist-packages (from firebase-admin) (2.26.0)
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packages (from httpx>0.28.1->http[https]>=0.28.1->firebase-admin) (4.13.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from httpx>0.28.1->http[https]>=0.28.1->firebase-admin) (2026.2.25)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx>0.28.1->http[https]>=0.28.1->firebase-admin) (1.0.9)
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from httpx>0.28.1->http[https]>=0.28.1->firebase-admin) (3.11)
Requirement already satisfied: h2<5,>=3 in /usr/local/lib/python3.12/dist-packages (from httpx[https]>=0.28.1->firebase-admin) (4.3.0)
Requirement already satisfied: h11>0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx>0.28.1->http[https]>=0.28.1->firebase-admin) (0.16.0)
Requirement already satisfied: requests>2.16.0 in /usr/local/lib/python3.12/dist-packages (from cachecontrol>0.12.14->firebase-admin) (2.32.4)
Requirement already satisfied: msgpack<2.0.0,>=0.5.2 in /usr/local/lib/python3.12/dist-packages (from cachecontrol>0.12.14->firebase-admin) (1.1.2)
Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.63.2 in /usr/local/lib/python3.12/dist-packages (from google-api-core<3.0.0dev,>=1.22.1->google-api-core[grpc]<3.0.0dev,>=1.22.1; platform_python_implementation != "PyPy")
Requirement already satisfied: protobuf<8.0.0,>=4.25.8 in /usr/local/lib/python3.12/dist-packages (from google-api-core<3.0.0dev,>=1.22.1->google-api-core[grpc]<3.0.0dev,>=1.22.1; platform_python_implementation != "PyPy")
Requirement already satisfied: proto-plus<2.0.0,>=1.22.3 in /usr/local/lib/python3.12/dist-packages (from google-api-core<3.0.0dev,>=1.22.1->google-api-core[grpc]<3.0.0dev,>=1.22.1; platform_python_implementation != "PyPy")
Requirement already satisfied: google-auth<3.0.0,>=2.14.1 in /usr/local/lib/python3.12/dist-packages (from google-api-core<3.0.0dev,>=1.22.1->google-api-core[grpc]<3.0.0dev,>=1.22.1; platform_python_implementation != "PyPy")
Requirement already satisfied: grpcio<2.0.0,>=1.33.2 in /usr/local/lib/python3.12/dist-packages (from google-api-core[grpc]<3.0.0dev,>=1.22.1; platform_python_implementation != "PyPy")
Requirement already satisfied: grpcio-status<2.0.0,>=1.33.2 in /usr/local/lib/python3.12/dist-packages (from google-api-core[grpc]<3.0.0dev,>=1.22.1; platform_python_implementation != "PyPy")
Requirement already satisfied: httplib2<1.0.0,>=0.19.0 in /usr/local/lib/python3.12/dist-packages (from google-api-python-client>1.7.8->firebase-admin) (0.31.2)
Requirement already satisfied: google-auth-httplib2<1.0.0,>=0.2.0 in /usr/local/lib/python3.12/dist-packages (from google-api-python-client>1.7.8->firebase-admin) (0.3.1)
Requirement already satisfied: google-cloud-core<3.0.0,>=1.4.1 in /usr/local/lib/python3.12/dist-packages (from google-cloud-firestore>2.19.0->firebase-admin) (2.5.1)
Requirement already satisfied: google-resumable-media<3.0.0,>=2.7.2 in /usr/local/lib/python3.12/dist-packages (from google-cloud-storage>1.37.1->firebase-admin) (2.8.2)
Requirement already satisfied: google-crc32c<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from google-cloud-storage>1.37.1->firebase-admin) (1.8.0)
Requirement already satisfied: cryptography>3.4.0 in /usr/local/lib/python3.12/dist-packages (from pyjwt[crypto]>2.5.0->firebase-admin) (43.0.3)
Requirement already satisfied: cffi>1.12 in /usr/local/lib/python3.12/dist-packages (from cryptography>3.4.0->pyjwt[crypto]>2.5.0->firebase-admin) (2.0.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packages (from cffi>1.12->cryptography>3.4.0->pyjwt[crypto]>2.5.0->firebase-admin) (2.0.0)
Requirement already satisfied: pyasn1-modules>0.2.1 in /usr/local/lib/python3.12/dist-packages (from google-auth<3.0.0,>=2.14.1->google-api-core<3.0.0dev,>=1.22.1->google-api-core[grpc]<3.0.0dev,>=1.22.1)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.12/dist-packages (from google-auth<3.0.0,>=2.14.1->google-api-core<3.0.0dev,>=1.22.1->google-api-core[grpc]<3.0.0dev,>=1.22.1)
( ) Переменные Терминал 11:46 T4 (Python 3)

```

Рисунок 11 – Подключение Firebase

Firebase Admin SDK – библиотека для взаимодействия с серверными сервисами Firebase на различных языках программирования, в том числе Python. Предоставляет привилегированный доступ к Firebase-проекту с правами

администратора, позволяя выполнять операции чтения, записи, изменения и удаления данных.

Помимо этого, для реализации системы прогнозирования были подключены следующие библиотеки:

- Pandas – для управления данными и их анализа данных. Библиотека предоставляет определенные структуры данных и операции для работы с числовыми таблицами;

- NumPy – для работы с большими многомерными массивами и матрицами. Библиотека также включает в себя большое количество высокоуровневых математических функций для работы с массивами и матрицами;

- Scikit-learn – для машинного обучения на языке программирования Python. Библиотека предоставляет алгоритмы классификации, регрессии и кластеризации и разработана как расширение библиотек NumPy и SciPy.

Успешное подключение библиотек показано на рисунке 12.

```
[1] ✓ 1 сек. from firebase_admin import credentials, firestore
      from google.cloud.firestore import ArrayUnion, ArrayRemove

[11] ✓ 0 сек. import pandas as pd
      import numpy as np
      from sklearn.preprocessing import LabelEncoder, StandardScaler
      from sklearn.model_selection import train_test_split

[12] ✓ 0 сек. from sklearn.linear_model import LogisticRegression
      from sklearn.multiclass import OneVsRestClassifier

      # Оценка качества
      from sklearn.metrics import accuracy_score, precision_recall_fscore_support, classification_report
```

Рисунок 12 – Подключение библиотек

3.3 Выбор инструментов разработки клиентской части

3.3.1 Выбор языка разработки

Для разработки мобильного приложения был выбран язык программирования Kotlin, первая версия которого была выпущена в 2016 году. Kotlin — это язык программирования общего назначения со статическим типом и

открытым исходным кодом, созданный не для кроссплатформенности, а для мультиплатформенности. Это означает, что язык предназначен для создания различных проектов – от разработки мобильного приложения или сервера до скрипта и даже для создания проектов на предметно-ориентированных языках (DSL).

Как и любой язык программирования, Kotlin необходимо компилировать или интерпретировать. Как уже упоминалось, он кроссплатформенный, поэтому компиляция происходит по-разному в зависимости от того, для чего мы хотим его использовать.

В 2011 году JetBrains анонсировал язык Kotlin, главной целью которого стало упрощение разработки программного обеспечения для мобильных устройств Android в части внедрения лаконичного синтаксиса и широкого спектра функций. Android Studio обеспечивает высокую интеграцию с языком Kotlin.

Язык программирования Kotlin считается нативным языком для разработки мобильных приложений, что обеспечивает высокую эффективность и оперативную работоспособность программного обеспечения, так как его создание производилось под конкретную платформу.

Асинхронное программирование в мобильной разработке является важной задачей. Её помогают решить корутины, которые могут выполняться в любом потоке и даже переключаться между потоками. Это означает, что программист не ведет такую работу как создание потоков и их поддержка, т.е. снижается вероятность ошибок и появления уязвимостей безопасности.

К основным особенностям языка Kotlin также относятся: распознавание ошибки Null Pointer Exception на этапе компиляции; наличие функций высших порядков, анонимных функций и лямбда-выражений, перегрузки операторов [13] и другие.

Поскольку Kotlin — это язык JVM, приложения, созданные на языке Kotlin будет корректно работать и на старых устройствах Android. Совместимость со

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		25

старыми версиями Android обеспечивает лёгкое масштабирование приложения на различные группы пользователей.

Необходимо помнить, что язык Kotlin обладает следующими характерными признаками: отсутствие обратной совместимости и медленное время сборки проекта.

3.3.2 Выбор среды разработки

3.3.2.1 Eclipse IDE

Eclipse IDE – открытая, независимая от языка программирования интегрированная среда разработки, выпущенная компанией IBM в 2001 году. Хотя среда разработки включает в себя большое количество встроенных функций, большая часть из них носит общий характер. Для расширения среды разработки требуются дополнительные плагины, позволяющие работать с новыми типами данных, выполнять новые действия с существующими типами данных и фокусировать общие функции на чем-то конкретном. При запуске платформы пользователю предоставляется IDE, состоящая из набора доступных плагинов. Качество пользовательского опыта в значительной степени зависит от того, насколько хорошо инструменты интегрируются с платформой и насколько хорошо различные инструменты взаимодействуют друг с другом.

Особенности Eclipse IDE:

- поддержка создания различных инструментов для разработки приложений;
- поддержка неограниченного набора поставщиков инструментов;
- обеспечение бесшовной интеграции инструментов внутри и между различными типами данных и поставщиками инструментов;
- поддержка как графических, так и неграфических сред разработки приложений;
- работа в широком спектре операционных систем, включая Windows и Linux [14].

Доступно большое количество плагинов на сайте разработчика [15] и бесплатное распространение среды. Однако в последнее время среда стала менее

популярна из-за устаревшего интерфейса и потребления большого количества ресурсов.

3.3.2.2 IntelliJ IDEA

IntelliJ IDEA — это интегрированная среда разработки, написанная на языке Java, для разработки программного обеспечения. Она разработана компанией JetBrains (ранее известной как IntelliJ) и доступна как в виде лицензированной версии Apache 2 Community Edition, так и в виде проприетарной коммерческой версии Ultimate Edition. Обе версии могут использоваться для коммерческой разработки.

К задачам, решаемым с помощью IntelliJ IDEA, относятся: трансляция, проверка, конфигурация и логирование. Все проекты включают модули тестирования, чтобы предотвратить попадание ошибок при развертывании.

Рассматриваемую среду разработки отличает функция автодополнения, помогающая оперативно написать код. Присутствует возможность удаленной разработки, а также режим коллективной работы над проектом в режиме реального времени. Однако такая функциональность требует высокой производительности от техники, на которой ведется разработка.

3.3.2.3 Android Studio

Android Studio — это официальная интегрированная среда разработки (IDE) для создания приложений на платформе Android. Android Studio выпущена компанией Google и построена на основе IntelliJ IDEA, включает специализированных плагинов для Android.

Она предоставляет разработчикам широкий спектр инструментов и функций для эффективного управления проектом. Android Studio предлагает удобный и интуитивно понятный интерфейс, который включает редактор кода, конструктор макетов, инструменты отладки, симуляторы устройств и другие инструменты, упрощающие процесс разработки.

Рассматриваемая среда программирования поддерживает языки программирования Java, Kotlin, Dart и C++, а также доступна для таких операционных систем, как Windows, macOS и GNU/Linux.

Android Studio предоставляет следующий набор возможностей:

- разработка программного обеспечения для различных видов устройств с разнообразным соотношением сторон экрана: смартфоны и планшеты, телевизионные приставки, переносные компьютеры, умные часы;
- обширный список готовых шаблонов и компонентов для разработки;
- поддержка сборки и развертывания проекта как на встроенных эмуляторах, так и на реальных устройствах, что упрощает разработку и тестирование;
- автоматизация процесса написания кода с помощью умного редактора кода [16].

Из отрицательных сторон можно отметить значительную загрузку аппаратной части персонального компьютера, на котором ведётся разработка. Также среда разработки не предназначена для разработки серверных проектов.

После анализа средств разработки была выбрана Android Studio, которая является наиболее подходящей для гибкой разработки мобильного приложения для ОС Android.

Для начала работы над проектом был выбран шаблон Empty Activity и установлены следующие зависимости:

- AndroidX – стандартные библиотеки Android (androidx.core:core-ktx:1.17.0, androidx.appcompat.appcompat:1.7.1, androidx.constraintlayout.constraintlayout:2.2.1, androidx.lifecycle:*, androidx.activity:activity:1.11.0, androidx.fragment:fragment:1.5.4);
- библиотеки для языка Kotlin (kotlin-stdlib:2.0.21, kotlinx.coroutines-android:1.8.1);
- библиотеки для работы с Firebase (firebase-firestore:25.1.0, firebase-analytics:22.1.2, firebase-common, firebase-components, firebase-installations, play-services-base, play-services-tasks, play-services-basement, protobuf-javalite);

– библиотеки для тестирования (junit:junit:4.13.2, androidx.test:runner:1.7, androidx.test.espresso:espresso-core:3.7.0).

Зависимости были проверены с помощью дерева зависимостей, выведенного через команду `./gradlew :app:dependencies`. Частичный вывод команды показан на рисунке 13.

```

+--- project :app (*)
+--- androidx.core:core-ktx:1.17.0
|   +--- androidx.annotation:annotation:1.8.1 (*)
|   +--- androidx.core:core:1.17.0
|   |   +--- androidx.annotation:annotation:1.8.1 (*)
|   |   +--- androidx.annotation:annotation-experimental:1.4.1
|   |   |   \--- org.jetbrains.kotlin:kotlin-stdlib:1.7.10 -> 2.0.21 (*)
|   |   +--- androidx.core:core-viewtree:1.0.0
|   |   |   +--- org.jetbrains.kotlin:kotlin-stdlib -> 2.0.21 (*)
|   |   |   \--- org.jetbrains.kotlin:kotlin-stdlib:1.8.22 -> 2.0.21 (c)
|   |   +--- androidx.lifecycle:lifecycle-common:2.6.2 (*)
|   |   +--- androidx.lifecycle:lifecycle-runtime:2.6.2
|   |   |   +--- androidx.annotation:annotation:1.1.0 -> 1.8.1 (*)
|   |   |   +--- androidx.arch.core:core-common:2.2.0
|   |   |   |   \--- androidx.annotation:annotation:1.1.0 -> 1.8.1 (*)
|   |   |   +--- androidx.lifecycle:lifecycle-common:2.6.2 (*)
|   |   |   +--- org.jetbrains.kotlin:kotlin-stdlib:1.8.10 -> 2.0.21 (*)
|   |   |   +--- androidx.lifecycle:lifecycle-common:2.6.2 (c)
|   |   |   +--- androidx.lifecycle:lifecycle-livedata-core:2.6.2 (c)
|   |   |   +--- androidx.lifecycle:lifecycle-viewmodel:2.6.2 (c)
|   |   |   +--- androidx.lifecycle:lifecycle-viewmodel-savedstate:2.6.2 (c)
|   |   |   |   \--- androidx.lifecycle:lifecycle-livedata:2.6.2 (c)
|   |   +--- androidx.versionedparcelable:versionedparcelable:1.1.1
|   |   |   +--- androidx.annotation:annotation:1.1.0 -> 1.8.1 (*)
|   |   |   \--- androidx.collection:collection:1.0.0 -> 1.1.0
|   |   |       \--- androidx.annotation:annotation:1.1.0 -> 1.8.1 (*)
|   |   +--- com.google.guava:guava:1.0 -> 9999.0-empty-to-avoid-conflict-with-guava
|   |   +--- org.jetbrains.kotlin:kotlin-stdlib -> 2.0.21 (*)
|   |   +--- org.jetbrains.kotlinx:kotlinx-coroutines-core:1.8.1 (*)
|   |   +--- org.jspecify:jspecify:1.0.0
|   |   |   \--- org.jetbrains.kotlin:kotlin-stdlib:2.0.21 (c)
|   |   +--- org.jetbrains.kotlin:kotlin-stdlib -> 2.0.21 (*)
|   |   |   \--- org.jetbrains.kotlin:kotlin-stdlib:2.0.21 (c)
+--- androidx.appcompat:appcompat:1.7.1
|   +--- androidx.activity:activity:1.8.0 -> 1.11.0
|   +--- androidx.annotation:annotation:1.8.1 (*)
|   +--- androidx.core:core-ktx:1.13.0 -> 1.17.0 (*)
|   +--- androidx.core:core-viewtree:1.0.0 (*)

```

Рисунок 13 – Вывод некоторых стандартных библиотек с помощью команды `./gradlew :app:dependencies`

В созданном проекте файл `AndroidManifest.xml` описывает конфигурацию, устанавливающую настройки запуска/отладки приложения. Значение параметра `allowBackup` по умолчанию установлено как `true`, что разрешает резервное копирование и восстановление приложения. Резервным копированием данных управляют следующие два параметра `dataExtractionRules` и `fullBackupContent` для

новых и старых версий Android соответственно. Далее задаются атрибуты такие как иконка приложения и её форма, название и тема приложения, а также возможность использования RTL API.

В проекте реализуется три экрана, два из которых являются внутренними. Точкой входа в приложение установлена SplashActivity, для которой задаётся атрибут `exported = "true"`, а также устанавливается intent-фильтр для фиксации главной активности и отображения её в лаунчере. Для других двух активностей – MainActivity и EstimateFactors – отсутствует разрешение их запуска системой извне. Для возможности открыть активности только внутри самого приложения параметр `exported` задан как `false`, что проиллюстрировано на рисунке 14.

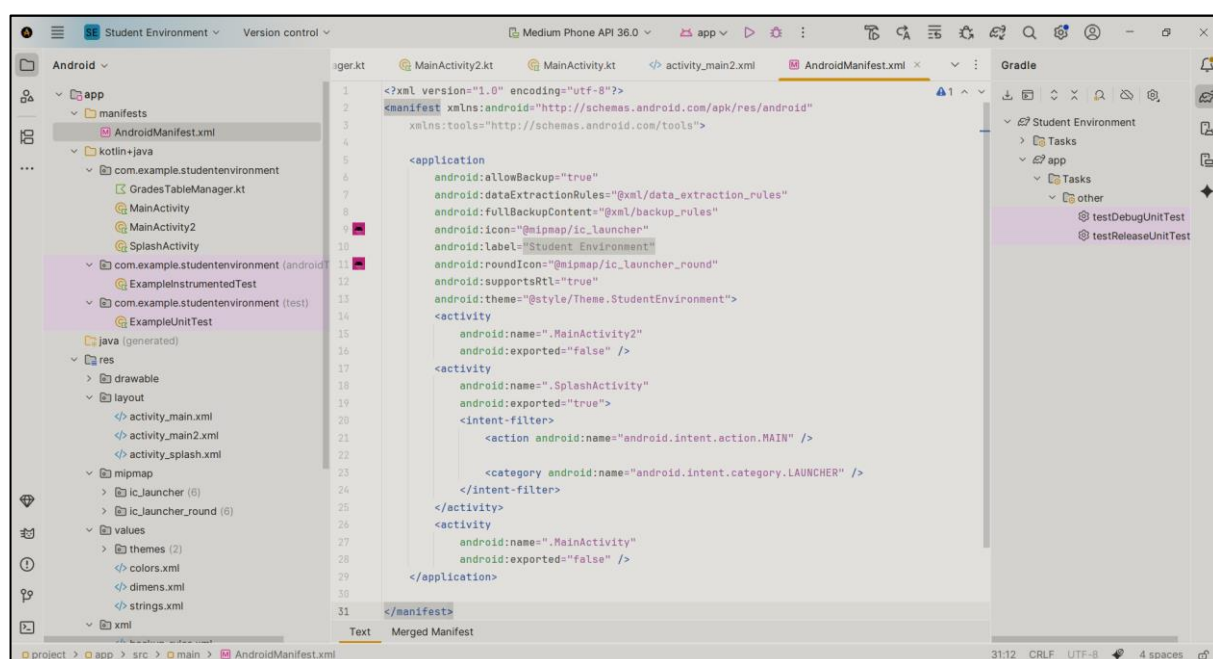


Рисунок 14 – Содержимое файла AndroidManifest.xml

Также была выполнена настройка файла `build.gradle.kts` модуля `app`. Помимо плагинов для сборки приложений и поддержки языка Kotlin был указан плагин Google Services, необходимый для интеграции с Firebase. Также были дополнительно подключены в блоке `dependencies` зависимости Firebase BOM версии 33.4.0. Параметры `minSdk`, `targetSdk`, `compileSdk` на рисунке 15 задают

минимальную версию Android 7.0, версию Android 15, для которой оптимизировано приложение, и которая используется для компиляции приложения.

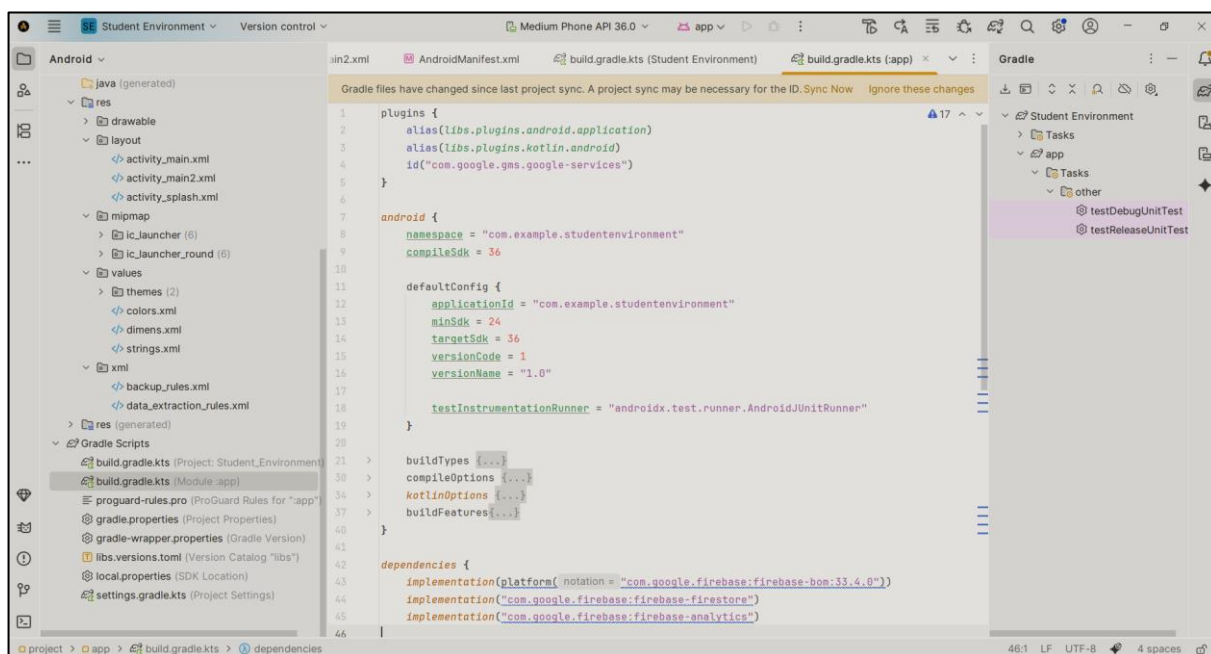


Рисунок 15 – Содержимое файла build.gradle.kts (модуль app)

Таким образом, настройка среды разработки завершена.

4 Архитектура системы

4.1 Архитектура приложения

Разрабатываемое приложение состоит из трёх основных узлов: клиентское мобильное устройство и серверная часть, содержащая логику приложения, и облачная инфраструктура Google Firebase, представляющая собой связующее звено между клиентом и сервером. Клиентская сторона включает в себя три экранные формы (SplashActivity, MainActivity и EstimateFactors), Firebase Firestore SDK и локальный кэш. Интерфейс приложения реализуется на языке Kotlin с использованием библиотеки Kotlin Runtime. Firebase Firestore SDK обеспечивает обмен данными между интерфейсом приложения и облачной базой данных. Локальный кэш позволяет сохранить работоспособность приложения без подключения к сети и обновить данные при появлении интернет-соединения.

Серверная сторона содержит две модели: классификатор уровня успеваемости (knn_model_export) и система прогнозирования неудовлетворительных отметок (grade_predictor_models_log) учащихся. На вход моделей подаются персональные данные учащихся (оценки, количество часов сна, количество часов обучения, регулярность посещений занятий, социально-экономический уровень учащегося). На выходе модели системы прогнозирования отметок получаем прогноз появления неудовлетворительной отметки для будущих 4 тем учащегося. На выходе классификатора уровня успеваемости ученика – показатель низкого, среднего или высокого уровня образования у учащегося. Полученные результаты передаются в Firestore.

Облачная база данных Firestore состоит из трёх коллекций: students, lessons, marks, которые хранят информацию об учениках, уроках и оценках соответственно. Firestore позволяет аккумулировать данные, получаемые из пользовательского интерфейса, передать их на сторону серверной части, где производится их обработка, а затем получить результат работы сервера.

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		32

Взаимодействие между клиентским и серверным узлами осуществляется через протокол HTTP. Firebase Firestore SDK обеспечивает поддержку офлайн-режима и синхронизацию данных после установления интернет-соединения. Диаграмма разворачивания приложения представлена на рисунке 8.

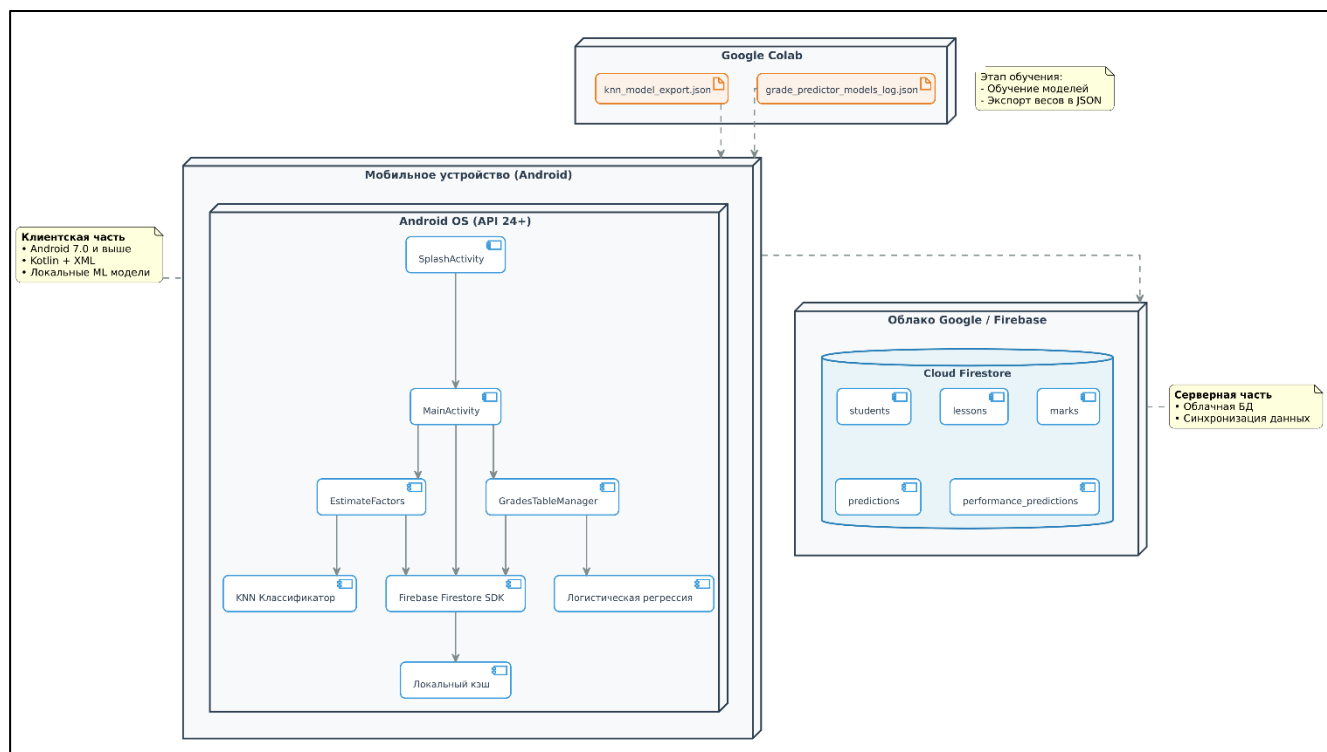


Рисунок 8 – Диаграмма разворачивания приложения

Экран-заставка – модуль `SplashActivity` – является точкой входа в приложение. Модуль предназначен для отображения приветственного экрана-заставки при запуске приложения. К функциям модуля относится перенаправление на следующий экран, содержащий журнальную сетку с учащимися – модуль `MainActivity`.

Модуль `MainActivity` загружает из Firestore

- ФИО учащихся (коллекция `students`);
- информацию об уроках – тема и дата занятия (коллекция `lessons`);
- имеющиеся отметки учащихся (коллекции `marks`);
- прогнозы неудовлетворительных отметок учащихся (коллекция `performance_predictions`).

После подгрузки указанных данных осуществляется отображение журнальной сетки учащихся: списка учеников и таблицы оценок с прогнозами. Также для работы с отметками модуль MainActivity обращается к модулю GradesTableManager. Помимо этого, модуль MainActivity осуществляет обработку нажатий кнопки напротив каждого ученика списка. При нажатии на иконку рядом с учащимся производится запуск EstimateFactors с передачей studentId для возможности дальнейшей идентификации, кому приписывать оценку факторов, влияющих на учебу.

Модуль EstimateFactors предназначен для сбора данных по учащемуся касательно учебных факторов, чтобы в дальнейшем определить уровень успеваемости ученика. После нажатия на кнопку «Получить результаты» модуль осуществляет сохранение баллов в Firestore (коллекция student_scores), а также результатов анкетирования (коллекция student_classification). Для определения результата производится вызов локальной KNN-модели (KNN Классификатор).

Модуль GradesTableManager осуществляет работу с отметками учащихся и содержит функции создания, редактирования и удаления отметки, а также управляет прогнозами неудовлетворительных отметок. Для прогноза неудовлетворительных отметок определяются три последние отметки ученика, производится вычисление их характеристик, пересчитывается прогноз отметок и результаты сохраняются в Firestore.

4.2 Архитектура базы данных

База данных состоит из пяти коллекций: lessons, marks, students, performance_predictions, predictions. Подробное описание коллекций представлено в таблицах 1-5.

Таблица 1 – Структура документа с информацией об уроке из коллекции lessons

№	Параметр	Тип данных	Описание
1	classId	String	Идентификатор класса, в учебном плане которого содержится урок
2	date	String	Дата проведения урока

№	Параметр	Тип данных	Описание
3	topic	String	Тема урока

Таблица 2 – Структура документа с информацией об отметках из коллекции marks

№	Параметр	Тип данных	Описание
1	lessonId	Integer	Идентификатор урока, по которому выставлена отметка
2	score	Integer	Вес отметки в 5-балльной шкале
3	studentId	Integer	Идентификатор учащегося

Таблица 3 – Структура документа с информацией об учащихся из коллекции students

№	Параметр	Тип данных	Описание
1	classId	String	Идентификатор класса, в котором закреплен ученика
2	item	String	Тема урока, по которой производится оценивание учащегося
3	name	String	ФИО учащегося
4	score	Integer	Вес отметки в 5-балльной шкале

Таблица 4 – Структура документа с информацией о предсказании уровня успеваемости учащегося из коллекции performance_predictions

№	Параметр	Тип данных	Описание
1	performance_level	String	Уровень успеваемости учащегося Пример: «Высокий уровень успеваемости», «Средний уровень успеваемости», «Низкий уровень успеваемости»
2	probabilities	Map	Вероятность классификации учащегося по уровню успеваемости
2.1	high	Double	Вероятность высокого уровня успеваемости учащегося
2.2	low	Double	Вероятность низкого уровня успеваемости учащегося
2.3	medium	Double	Вероятность среднего уровня успеваемости учащегося
3	scores	Map	Баллы, выставленные ученику по различным факторам, влияющим на образовательный процесс
3.1	criterion1	Integer	Баллы, выставленные за критерий «Количество часов сна»
3.2	criterion2	Integer	Баллы, выставленные за критерий «Количество часов обучения»

№	Параметр	Тип данных	Описание
3.3	criterion3	Integer	Баллы, выставленные за критерий «Регулярность посещений занятий»
3.4	criterion4	Integer	Баллы, выставленные за критерий «Социально-экономическое окружение»
4	studentId	String	Идентификатор учащегося

Таблица 5 – Структура документа с информацией об отметках из коллекции predictions

№	Параметр	Тип данных	Описание
1	lessonId	String	Идентификатор урока, по которому сделан прогноз неудовлетворительной отметки
2	predictedScore	Integer	Прогноз неудовлетворительной отметки. Принимает значения «0» или «1». При «0» – риск неудовлетворительно отметки по теме минимален. При «1» – прогнозируется неудовлетворительная отметка по теме.
3	studentId	String	Идентификатор учащегося, для которого произведен прогноз неудовлетворительной отметки
4	timestamp	String	Время выполненного анализа для прогноза неудовлетворительной отметки

5 Разработка приложения

В модуле GradesTableManager для определения уровня усвоения образовательного материала по темам были реализованы функции создания, редактирования и удаления отметок для учащихся – createGradeCell, updateGrade, deleteGrade соответственно.

```
setOnKeyListener { _, keyCode, event ->
    if (keyCode == KeyEvent.KEYCODE_ENTER && event.action ==
KeyEvent.ACTION_UP) {
        val grade = text.toString().toIntOrNull()
        if (grade != null && grade in 1..5) {
            saveGradeToFirebase(studentId, lessonId, grade)
            clearFocus() }
    }
```

После ввода отметки при нажатии на кнопку «Готово» производится сохранение оценки в соответствующую коллекцию БД с помощью функции saveGradeToFirebase:

```
private fun saveGradeToFirebase(studentId: String, lessonId:
String, score: Int) {
    val db = FirebaseFirestore.getInstance()
    db.collection("marks")
        .whereEqualTo("studentId", studentId)
        .whereEqualTo("lessonId", lessonId)
        .get()
        .addOnSuccessListener { documents ->
            if (documents.isEmpty()) {
                val gradeData = hashMapOf(
                    "studentId" to studentId,
                    "lessonId" to lessonId,
                    "score" to score,
                    "timestamp" to FieldValue.serverTimestamp()
                )
                db.collection("marks").add(gradeData)
            }
        }
```

```

        .addOnSuccessListener {
            updatePredictions(studentId) }

```

После выставления или после редактирования отметки происходит запуск пересчета прогноза неудовлетворительных оценок при условии, что у учащегося имеется хотя бы три подряд выставленные отметки. С помощью функции `updatePredictions()` производится обновление прогноза и его сохранение в Firestore. Прогноз строится на основе трёх последних отметок учащегося, а также дополнительных признаков таких как средний балл, количество неудовлетворительных отметок, последняя отметка, наличие изменения отметки (высчитывается между последней и предпоследней отметками), минимальная отметка и т.д. Подробное описание алгоритма прогноза неудовлетворительных отметок представлено в подразделе «Система прогнозирования неудовлетворительных отметок учащихся». Сохранение результатов прогноза производится через функцию `savePredictionsToFirestore()` в коллекцию `predictions`.

```

private fun savePredictionsToFirestore(studentId: String,
predictions: List<Float>, lessonIds: List<String>) {
    for (i in predictions.indices) {
        if (i < lessonIds.size) {
            val lessonId = lessonIds[i]
            val docId = "${studentId}_${lessonId}"
            val predData = hashMapOf(
                "studentId" to studentId,
                "lessonId" to lessonId,
                "predictedScore" to predictions[i],
                "timestamp" to FieldValue.serverTimestamp(),
                "type" to "grade_prediction"
            )

```

Обновление пользовательского интерфейса производится функцией `reloadPredictions()`, которая окрашивает ячейки в белый цвет (вспомогательная функция `gradeNormal()`) – риск неудовлетворительной отметки минимален, либо в

розовый цвет (вспомогательная функция gradeWarning()) – возможен плохой результат по теме.

```
private fun reloadPredictions(studentId: String) {
    db.collection("predictions")
        .whereEqualTo("studentId", studentId)
        .get()
        .addOnSuccessListener { documents ->
            for (document in documents) {
                val lessonId = document.getString("lessonId") ?:
continue
                val predictedScore =
document.getDouble("predictedScore")?.toInt() ?: continue

                if (predictedScore == 1) {
                    gradeWarning (studentId, lessonId)
                } else {
                    gradeNormal (studentId, lessonId)
                }
            }
        }
}
```

В модуле MainActivity для отображения уровня успеваемости учащегося используется кнопка напротив имени ученика. При нажатии на неё осуществляется переход к EstimateFactors для оценивания ученика по 4 различным критериям, влияющих на образовательный процесс. Изменение баллов по критерию вызывает функцию их сохранения в коллекцию student_scores:

```
private fun saveScoreToFirestore(criterionId: Int, value: Int) {
    if (studentId.isEmpty()) return
    val data = hashMapOf(
        "studentId" to studentId,
        "criterionId" to criterionId,
        "value" to value,
```

```

        "timestamp" to System.currentTimeMillis()
    )

    db.collection ("student_scores")
        .document ("${studentId}_criterion${criterionId}")
        .set (data)
    }
}

```

При нажатии на кнопку «Получить результаты» вызывается функция `saveAndPredict()`, которая осуществляет высчитывание результата уровня успеваемости ученика на основе оценённых критериев. Подробное описание методологии определения уровня успеваемости представлено в подразделе «Классификатор успеваемости учащихся»

Подробное описание классов показано на рисунке 16.

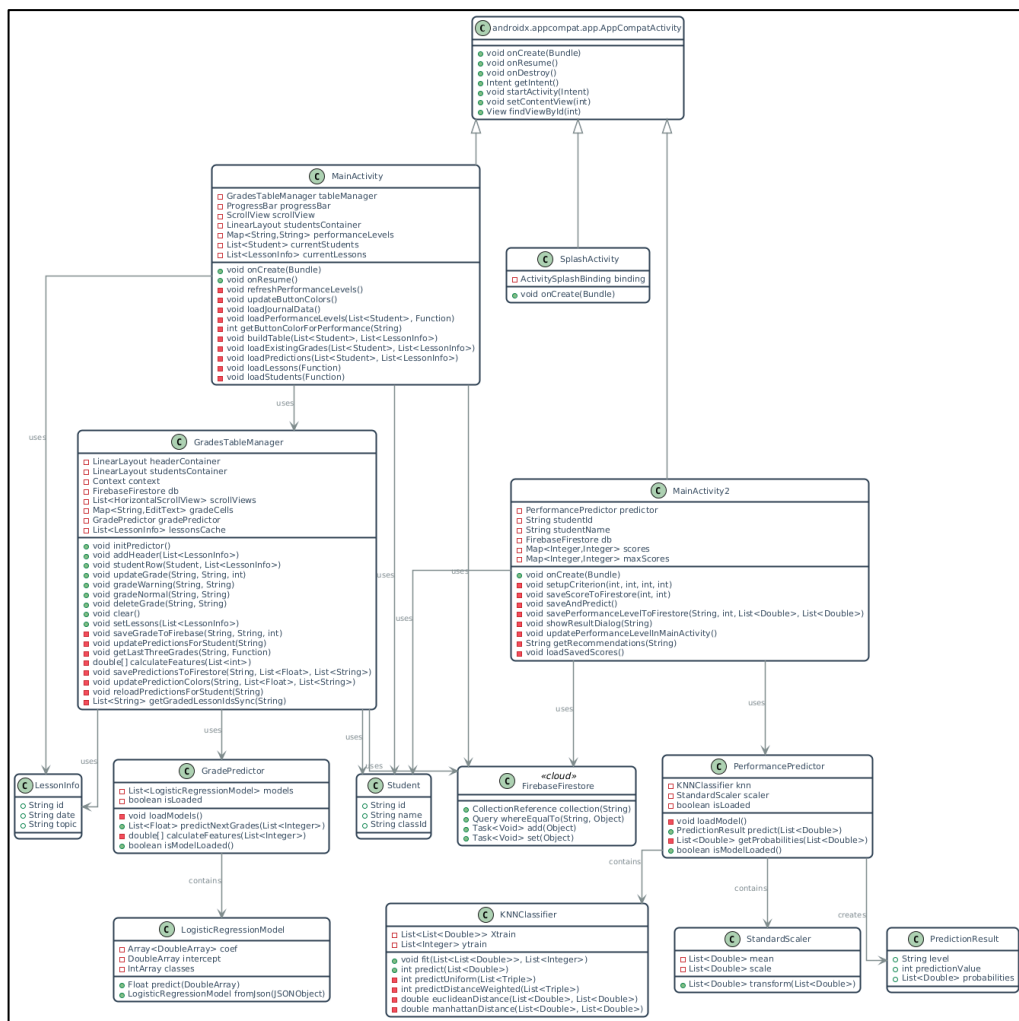


Рисунок 16 – Диаграмма классов

5.1 Система прогнозирования неудовлетворительных отметок учащихся

Система прогнозирования неудовлетворительных отметок учащихся должна принимать на вход массив отметок ученика, после обработки которого определять возможное наличие неудовлетворительных отметок за будущую тему.

Для того, чтобы результат работы системы оставался всегда актуальным было принято решение подавать на вход три оценки, полученные учеником за последние изученные темы. Таким образом система будет опираться на свежие данные успеваемости ученика, учитывая его текущие тенденции обучения.

Следует учитывать: чем дальше дата предсказываемой отметки, тем ниже точность прогноза. Поэтому прогноз был ограничен четырьмя будущими отметками, отсчитывая от последней полученной оценки.

Неудовлетворительной оценкой считается отметка «2» или «3». Целью системы не является предсказание конкретной отметки в 5-балльной шкале, система должна обнаруживать факт наличия неудовлетворительной отметки в будущем. Следовательно, задача сводится к классификации и ответу на вопрос «Ученик получит за тему N неудовлетворительную отметку или нет?», где N – это тема будущего урока, который предусмотрен по КТП максимум через 3 отметки после уже имеющейся отметки за текущую тему. Для решения рассматриваемой задачи была выбрана логистическая регрессия.

Логистическая регрессия – простой, но высокопроизводительный инструмент, используемый для задачи классификации. Логистическая регрессия включает в себя три основных шага:

- 1) суммирование признаков с учетом коэффициентов важности;
- 2) на основе вычисленного логита получение вероятности;
- 3) сравнение вычисленной вероятности с порогом, отвечающим за принятие решения.

Для обучения модели на основе логистической регрессии для рассматриваемой задачи необходим набор данных, состоящий из оценок,

полученных учениками за все темы учебного периода. Так как требуемый набор данных относится к персональным данным, поиск подобных выборок на открытых платформах результатов не дал. Поэтому набором данных для обучения послужили искусственно сгенерированные данные с учетом особенностей обучения. Для корректного обучения модели был принят во внимание календарно-тематический план учащихся за 7 класс по предмету «Физика» [17]. Из КТП были взяты часы, которые отводятся на изучение каждой темы из учебного курса. За выбранный учебный период учащиеся изучают 16, таким образом, получаем массив из 16 чисел, каждое число которого – это количество часов, требуемое для освоения темы.

```
hours_per_topic = [2, 2, 2, 1, 2, 2, 3, 4, 14, 3, 5, 6, 7, 3, 5, 4]
```

Заметим зависимость: чем больше часов определено на тему, тем сложнее тема для освоения. Этот признак учтем при составлении синтетического набора данных. Помимо длительности изучения темы учтем средний балл учащегося (чем ниже средний балл учащегося, тем выше риск неудовлетворительной отметки) и количество имеющихся неудовлетворительных отметок ученика (чем больше плохих отметок, тем выше риск неудовлетворительной отметки):

```
const = 1.2          #  $\beta_0$  – константа
coef_mean = -1.15    # коэфф-т ср. балла
coef_bad = 1.5        # коэфф-т числа неудовлетворительных отметок
coef_hours = 0.4      # коэфф-т длительности изучения темы
```

Коэффициенты были подобраны так, чтобы выборка давала оптимальный баланс классов. Баланс классов в выборке для каждой из 4 будущих тем показан на рисунке 17.

Затем организован цикл, в котором для каждого ученика генерируются первые три оценки – основные оценки, по которым в дальнейшем строятся показатели успеваемости учащегося с учетом признаков.

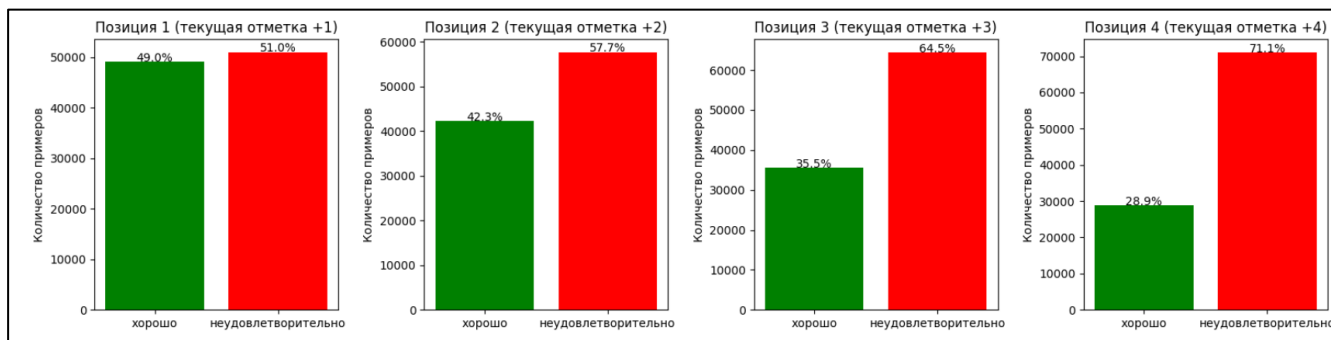


Рисунок 17 – Баланс классов для каждой будущей темы

При генерации первых трех отметок учитывается то, что вероятность получения отметок «2», «3», «4», «5» распределена примерно как 0.05, 0.15, 0.45, 0.35 соответственно.

```
for student_id in range(n_students):
    grades = []

    # генерация первых 3 тем - достаточно, чтобы сделать первые
    # предсказания

    grade1 = np.random.choice([2,3,4,5], p=[0.05, 0.15, 0.45, 0.35])
    grade2 = np.random.choice([2,3,4,5], p=[0.05, 0.15, 0.45, 0.35])
    grade3 = np.random.choice([2,3,4,5], p=[0.05, 0.15, 0.45, 0.35])
    grades = [grade1, grade2, grade3]
```

Следующие отметки генерируются на основе трёх последних оценок. По каждым трем оценкам высчитываются следующие показатели: средний балл, число неудовлетворительных отметок, а также из массива `hours_per_topic` берётся длительность изучения темы, по которой создается отметка.

```
# генерация тем 4-16 (каждая на основе 3 последних)
for topic_idx in range(3, n_topics):
    # подсчет статистики для 3 последних оценок
    last_three_items = grades[-3:]
    mean_grade = np.mean(last_three_items)
    bad_count = sum(1 for g in last_three_items if g <= 3)
    hours = hours_per_topic[topic_idx]
```

Используя формулы

$$P(Y=1) = 1 / (1 + e^{(-z)}), \quad (1)$$

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n, \quad (2)$$

где $P(Y=1)$ — вероятность положительного исхода;

e — основание натурального логарифма (≈ 2.718);

z — линейная комбинация признаков;

β_0 — константа (свободный член);

$\beta_1, \beta_2, \dots, \beta_n$ — коэффициенты регрессии;

$X_1, X_2, \dots, X_n$ — независимые переменные (признаки).

Получаем логит и преобразуем его в вероятность:

```
logit = (const
        + coef_mean * mean_grade
        + coef_bad * bad_count
        + coef_hours * hours)

probability = expit(logit) # преобразование логита в вероятность
```

Опираясь на полученную вероятность, формируем оценку учащегося с добавлением случайного шума при помощи функции `random()`:

```
if np.random.random() < probability:
    new_grade = np.random.choice([2, 3], p=[0.3, 0.7])
else:
    new_grade = np.random.choice([4, 5], p=[0.6, 0.4])
grades.append(new_grade)
all_students.append(grades)
```

Таким образом получаем набор данных, состоящий из 10000 учащихся, у каждого из которых имеется отметка по 16 темам. Первые 10 строк полученного датасета показаны на рисунке 18.

[13] ✓ 0 Cbk. df.head(10)

	topic_1	topic_2	topic_3	topic_4	topic_5	topic_6	topic_7	topic_8	topic_9	topic_10	topic_11	topic_12	topic_13	topic_14	topic_15	topic_16
0	4	5	5	4	4	5	4	5	2	3	3	3	2	3	2	3
1	2	4	3	3	5	2	3	3	3	3	3	2	3	3	3	2
2	2	4	4	3	4	4	4	5	2	5	5	3	3	3	3	3
3	4	5	4	5	4	4	4	2	3	3	2	3	2	2	3	3
4	5	3	5	5	4	4	5	4	3	2	3	3	3	5	3	2
5	2	4	4	2	4	3	4	5	3	5	4	3	2	3	2	2
6	4	3	5	5	3	3	4	3	3	2	3	3	3	3	3	2
7	5	4	2	3	2	3	2	2	3	3	3	3	2	3	3	3
8	4	4	3	4	3	3	5	2	3	5	2	3	2	3	3	3
9	5	3	4	5	5	4	5	5	3	5	5	4	4	4	4	4

Рисунок 18 – Сгенерированный набор данных

Чтобы обучить модель для требуемой задачи на вход необходимо подать признаки и целевую переменную. Список признаков сформируем из следующих показателей:

- средний балл;
- количество неудовлетворительных отметок;
- последняя отметка;
- наличие изменения отметки (высчитывается между последней и предпоследней отметками);
- минимальная отметка;
- стандартное отклонение (характеризует стабильность ученика);
- наличие отметки «2».

Применяем вышеуказанные признаки для окна из трех отметок. Таким образом, окно двигается по всем темам учащегося составляя массив признаков.

```
# определение окна
for start_id_x in range(window_size, n_topics -
prediction_range + 1):
    # статистика - признаки из окна (ср. балл и кол-во
плохих отметок)

    window = grades[start_id_x - window_size:start_id_x]
    mean_grade = np.mean(window)
    bad_count = sum(1 for g in window if g <= 3)
```

```

last_grade = window[-1]                # последняя оценка
trend = window[-1] - window[-2]        # изменилась ли оценка
min_grade = np.min(window)              # минимальная оценка
std_grade = np.std(window)              # стабильность
has_two = 1 if 2 in window else 0      # была ли двойка
X_list.append([mean_grade, bad_count, last_grade, trend,
min_grade, std_grade, has_two])

```

Прогноз неудовлетворительной отметки высчитывается отдельно для 4 будущих тем, поэтому целевая переменная y будет являться многомерной бинарной целевой переменной, для которой показатель «1» служит индикатором наличия неудовлетворительной отметки.

```

y_window = []
for offset in range(prediction_range):
    target_id_x = start_id_x + offset
    y_window.append(1 if grades[target_id_x] <= 3 else 0)
y_list.append(y_window)

```

Первые строки полученного набора данных, подаваемого на вход модели для обучения представлен на рисунке 19.

***	№	ср.балл	кол-во неуд.	последняя	отметка	тренд	мин.отметка	стандарт.отклонение	наличие"2"	тема4	тема5	тема6	тема7
	1	4.67	0.0		5.0	+0.0	4.0	0.47	0	0	0	0	0
	2	4.67	0.0		4.0	-1.0	4.0	0.47	0	0	0	0	0
	3	4.33	0.0		4.0	+0.0	4.0	0.47	0	0	0	0	1
	4	4.33	0.0		5.0	+1.0	4.0	0.47	0	0	0	1	1
	5	4.33	0.0		4.0	-1.0	4.0	0.47	0	0	1	1	1
	6	4.67	0.0		5.0	+1.0	4.0	0.47	0	1	1	1	1
	7	3.67	1.0		2.0	-3.0	2.0	1.25	1	1	1	1	1
	8	3.33	2.0		3.0	+1.0	2.0	1.25	1	1	1	1	1
	9	2.67	3.0		3.0	+0.0	2.0	0.47	1	1	1	1	1
	10	3.00	3.0		3.0	+0.0	3.0	0.00	0	1	1	1	1

Рисунок 19 – Набор данных для обучения

После обучения модели получена следующая метрика точности для каждой из будущих тем, представленная на рисунке 20.

Обучение модели для будущей отметки (текущая тема + 1)
Accuracy: 80.52%

Обучение модели для будущей отметки (текущая тема + 2)
Accuracy: 79.81%

Обучение модели для будущей отметки (текущая тема + 3)
Accuracy: 76.65%

Обучение модели для будущей отметки (текущая тема + 4)
Accuracy: 69.88%

Рисунок 20 – Результаты обучения модели

5.2 Классификатор успеваемости учащихся

Классификатор успеваемости учащихся должен определять уровень успеваемости ученика (низкий средний, высокий) основываясь на 4 признака:

- количество часов сна учащегося;
- количество часов обучения учащегося;
- регулярность посещений занятий учащимся;
- социально-экономический уровень учащегося.

Для обучения модели был подобран датасет на открытой платформе Kaggle [18]. Выбранный набор данных продемонстрирован на рисунке 21.

df.head(10)

	Socioeconomic Score	Study Hours	Sleep Hours	Attendance (%)	Grades
0	0.95822	3.4	8.2	53.0	47.0
1	0.85566	3.2	5.9	55.0	35.0
2	0.68025	3.2	9.3	41.0	32.0
3	0.25936	3.2	8.2	47.0	34.0
4	0.60447	3.8	10.0	75.0	33.0
5	0.98320	3.4	9.0	47.0	51.0
6	0.56648	7.9	8.1	63.0	54.0
7	0.93487	1.4	8.0	47.0	34.0
8	0.46660	5.4	8.8	67.0	39.0
9	0.62130	1.4	9.6	42.0	34.0

Рисунок 21 – Содержимое датасета

Датасет содержит 4 вышеперечисленных признака, а также целевую переменную – отметку учащегося по 100-балльной шкале. Для классификации на низкий, средний и высокий уровни образования требуется разделить набор данных на три категории. Наиболее оптимальный баланс выборки достигается при установлении для низкого уровня успеваемости верхнего порога, равного 34 баллам; для среднего уровня успеваемости – 45 баллов. Полученный баланс выборки представлен на рисунке 22.

На рисунке 23 представлено распределение характеристик для каждого признака (социально-экономический уровень, количество часов обучения, количество часов сна, процент посещений занятий учащимся, отметка учащегося).

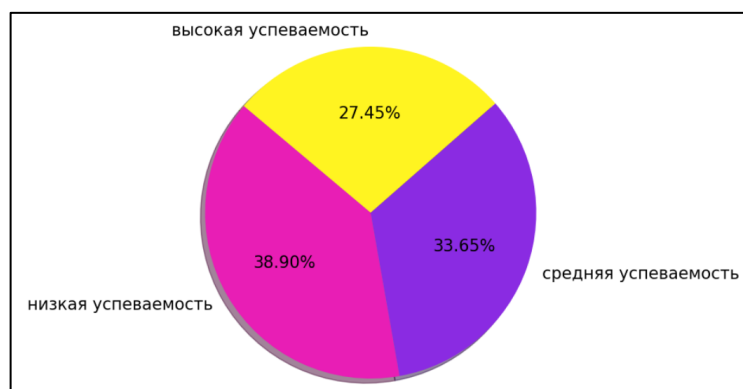


Рисунок 22 – Баланс выборки

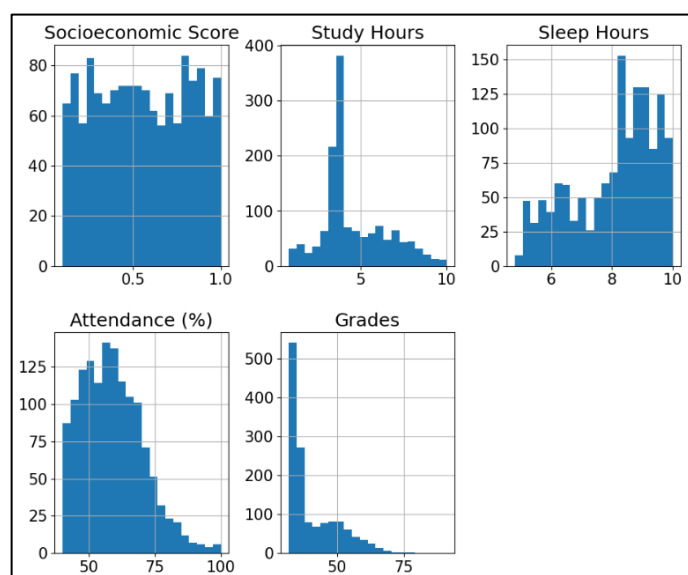


Рисунок 23 – Гистограммы, построенные по данным выбранного датасета

Сводка на рисунке 24 описательных статистик по датасету показывает, что в наборе данных отсутствуют пропуски, сильных выбросов нет, либо они симметрично распределены.

[687]
✓ 0 сек.

```
df.describe()
```

	Socioeconomic Score	Study Hours	Sleep Hours	Attendance (%)	Grades
count	1388.000000	1388.000000	1388.000000	1388.000000	1388.000000
mean	0.552274	4.560807	8.047262	58.536023	40.691643
std	0.261272	1.897581	1.370700	11.675287	9.467358
min	0.101280	0.800000	4.800000	40.000000	32.000000
25%	0.322118	3.475000	7.000000	49.000000	34.000000
50%	0.545945	3.900000	8.400000	57.000000	35.000000
75%	0.789610	5.900000	9.100000	66.000000	47.000000
max	0.999820	10.000000	10.000000	100.000000	91.000000

Рисунок 24 – Сводка статистических характеристик полного датасета

Отдельно для каждого класса успеваемости (низкий, средний, высокий уровень успеваемости) приведены статистические показатели, которые также подтверждают корректность данных для использования их в обучении модели. Данные для каждого класса представлены на рисунках 25-27.

[688]
✓ 0 сек.

```
data_low.describe()
```

	Socioeconomic Score	Study Hours	Sleep Hours	Attendance (%)	Grades
count	540.000000	540.000000	540.000000	540.000000	540.000000
mean	0.469905	3.309630	8.083704	55.594444	33.601852
std	0.217464	0.860976	1.390544	10.388929	0.573694
min	0.101550	0.800000	5.000000	40.000000	32.000000
25%	0.283445	3.100000	6.900000	47.000000	33.000000
50%	0.452275	3.600000	8.400000	54.000000	34.000000
75%	0.637350	3.800000	9.200000	63.000000	34.000000
max	0.934870	4.800000	10.000000	87.000000	34.000000

Рисунок 25 – Сводка статистических характеристик для класса «низкий уровень успеваемости»

[689]

0

сек.

✓

▼

data_mid.describe()

	Socioeconomic Score	Study Hours	Sleep Hours	Attendance (%)	Grades
count	467.000000	467.000000	467.000000	467.000000	467.000000
mean	0.555590	4.187794	8.177302	57.884368	37.899358
std	0.257788	1.256042	1.322664	10.135113	3.332848
min	0.101280	0.900000	4.800000	40.000000	35.000000
25%	0.326130	3.400000	7.400000	51.000000	35.000000
50%	0.554120	3.900000	8.500000	57.000000	36.000000
75%	0.806930	5.250000	9.200000	65.000000	40.500000
max	0.986530	6.800000	10.000000	88.000000	45.000000

Рисунок 26 – Сводка статистических характеристик для класса «средний уровень успеваемости»

[690]

0

сек.

✓

▼

data_high.describe()

	Socioeconomic Score	Study Hours	Sleep Hours	Attendance (%)	Grades
count	381.000000	381.000000	381.000000	381.000000	381.000000
mean	0.664953	6.791339	7.836220	63.503937	54.162730
std	0.279649	1.665202	1.379245	13.450616	7.078491
min	0.103850	2.900000	5.000000	40.000000	46.000000
25%	0.441180	6.100000	6.600000	54.000000	49.000000
50%	0.740710	7.000000	8.200000	61.000000	52.000000
75%	0.925570	7.900000	8.900000	71.000000	58.000000
max	0.999820	10.000000	10.000000	100.000000	91.000000

Рисунок 27 – Сводка статистических характеристик для класса «высокий уровень успеваемости»

Также на рисунке 28 показаны диаграммы выбросов в виде «ящика с усами», на которых видно наличие выбросов в признаках, отвечающих за процент посещаемости и количество часов обучения. Выбросы были удалены.

Обновленные данные для двух признаков показаны на рисунках 29-30. При этом оптимальный баланс выборки сохранился, показан на рисунке 31.

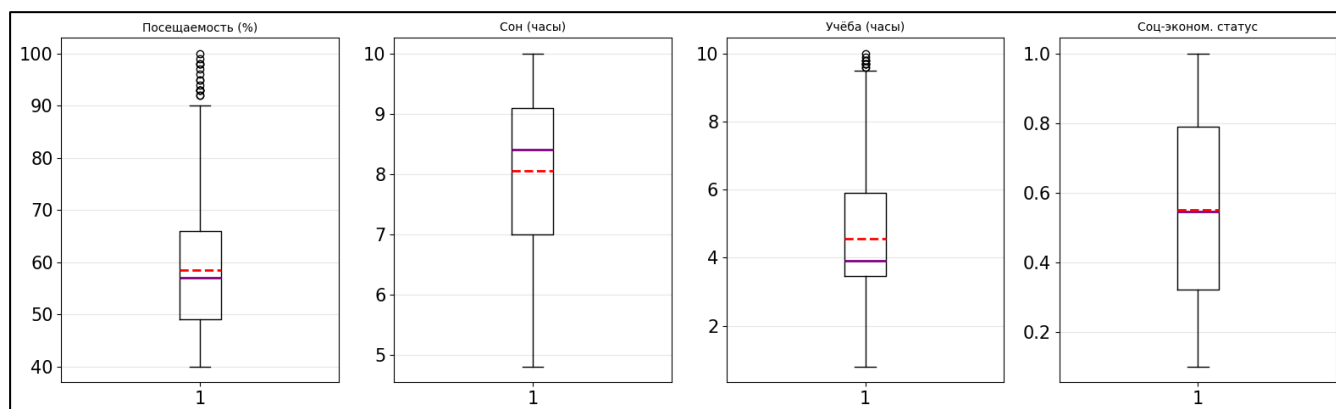


Рисунок 28 – Диаграммы выбросов

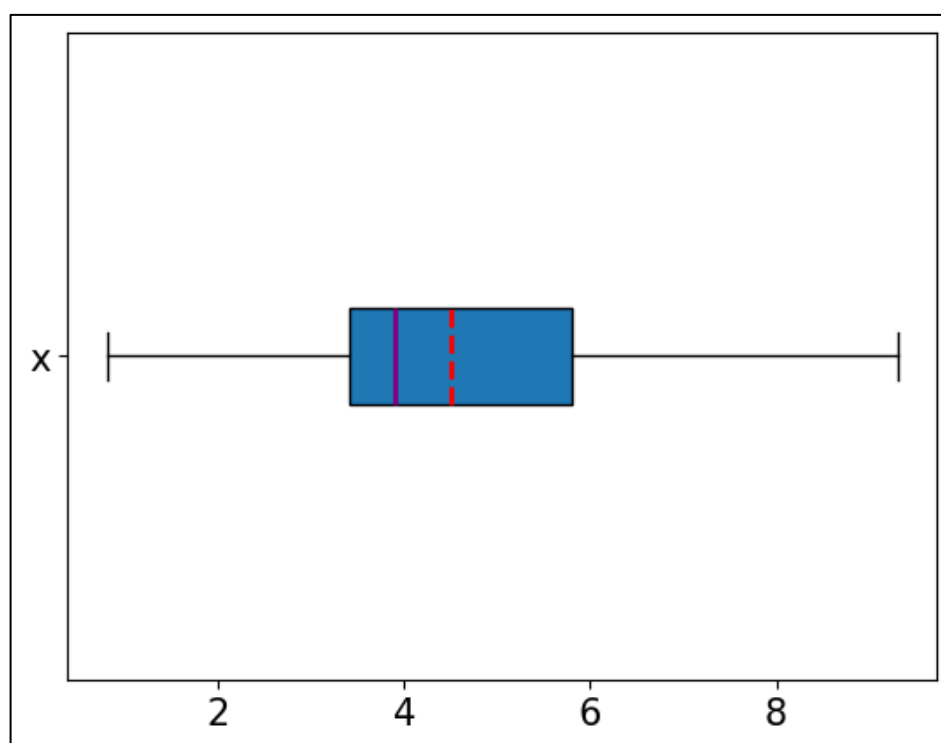


Рисунок 29 – Диаграммы выбросов для признака «количество часов обучения учащегося»

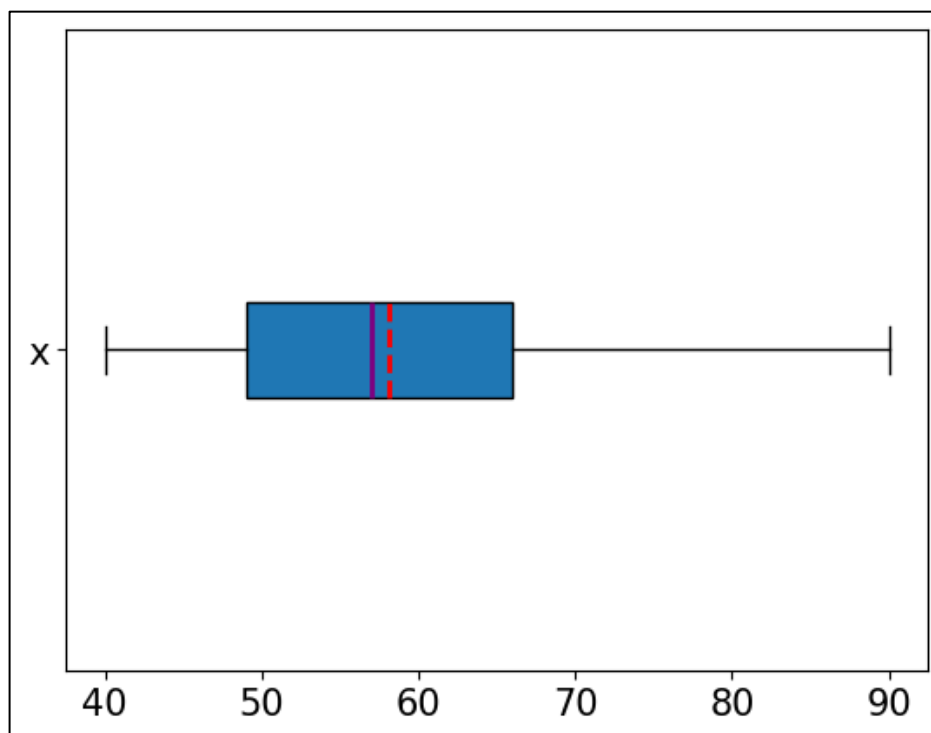


Рисунок 30 – Диаграммы выбросов для признака «посещение занятий учащимся»

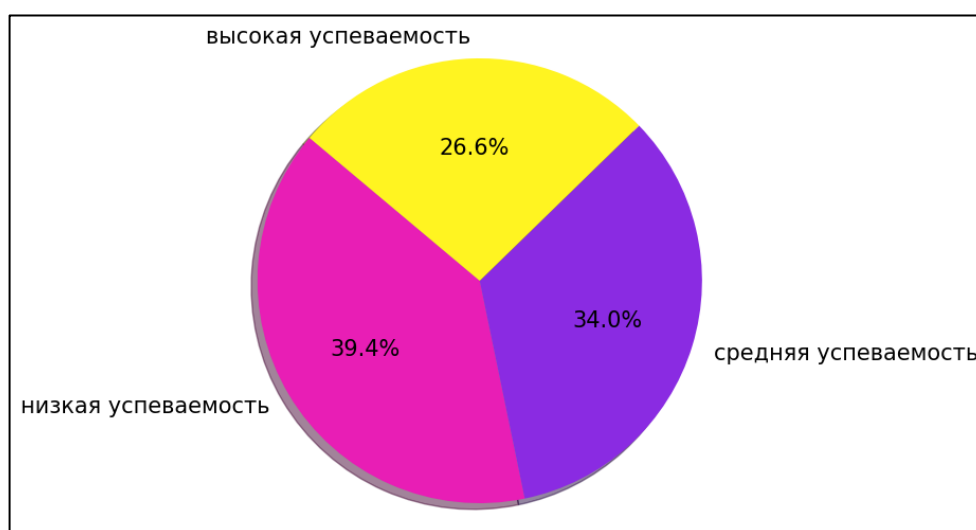


Рисунок 31– Баланс выборки после удаления выбросов

Дубликаты в наборе данных не обнаружены. Подготовка обучающих данных завершена.

Для определения к какому классу принадлежит учащийся (низкая, средняя, высокая успеваемость) использован метод k-ближайших соседей. Метод предполагает определение класса для новых точек опираясь на расстояние между

известной меткой и новой. Чем больше в окружении новой метки определенного класса, тем наиболее вероятно, что она принадлежит к тому же классу.

Для обучения данные были разделены на обучающую и тестовую выборки и стандартизированы. Наилучшие результаты были достигнуты при количестве соседей, равном 19.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
knn = KNeighborsClassifier(n_neighbors=19)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

На рисунке 32 представлена матрица ошибок, на которой низкий уровень успеваемости обозначен в виде «0», средний уровень успеваемости – «1», высокий уровень успеваемости – «2».



Рисунок 32 – Матрица ошибок по трем классам

Точность обученной модели составляет 75%.

После обучения модели были достигнуты следующие результаты. Метрика точности Precision отражает способность системы рекомендовать нужные материалы с небольшим процентом неточности. Полнота Recall характеризуется высоким значением, что свидетельствует об охвате выборки положительных рекомендаций [22]. Гармоническое среднее Precision и Recall характеризует хороший баланс между точностью и полнотой. Общая точность системы составляет 75%. Результаты обучения модели продемонстрированы на рисунках 33-34.

Точность(Accuracy): 0.75
Точность (Precision): 0.75
Полнота (Recall): 0.75
F1-мера: 0.73

Рисунок 33 – Результаты обучения

	precision	recall	f1-score	support
-1.0	0.70	0.92	0.80	109
0.0	0.67	0.48	0.56	92
1.0	0.91	0.82	0.87	74
accuracy			0.75	275
macro avg	0.76	0.74	0.74	275
weighted avg	0.75	0.75	0.73	275

Рисунок 34 – Результаты обучения по каждому классу

6 Разработка пользовательского интерфейса

Перед реализацией мобильного интерфейса построена карта экранов, представленная на рисунке 35.

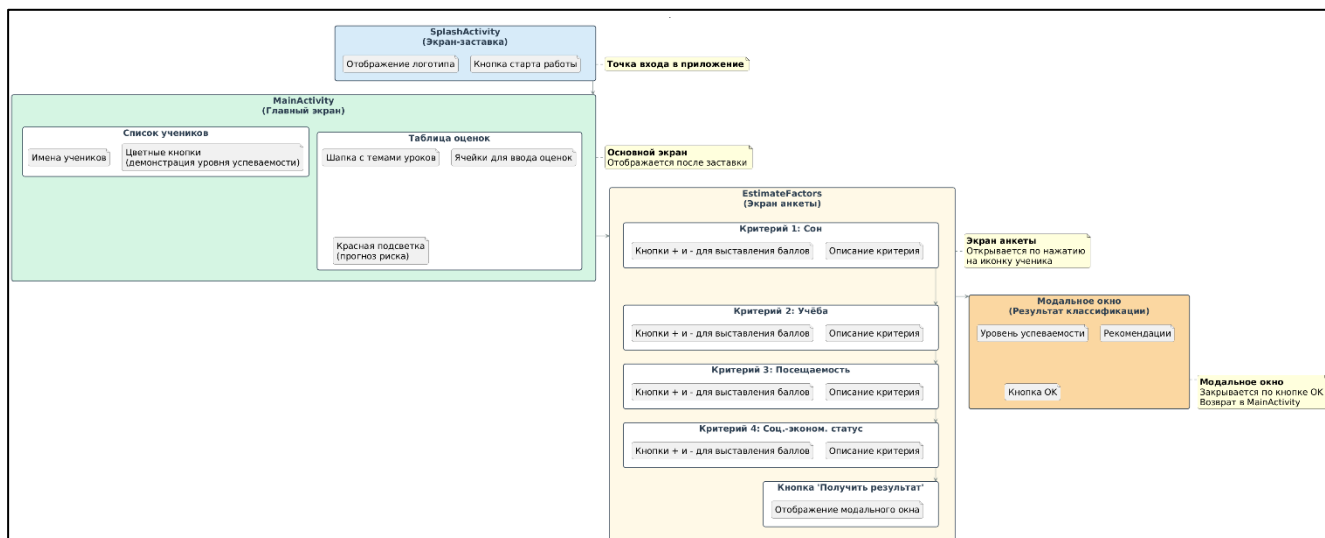


Рисунок 35 – Карта экранов приложения

При открытии приложения загружается первая экранная форма – сплеш-экран, содержащий название системы и кнопку «Перейти» для продолжения работы в системе. Для этого в файле AndroidManifest.xml были заданы настройки, разрешающие запуск активности SplashActivity другими приложениями/системой. Активность назначается главной точкой входа в приложение и устанавливается иконка разрабатываемого приложения в список всех приложений на устройстве.

```
<activity
android:name=".SplashActivity"
android:exported="true"> <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter> </activity>
```

Интерфейс сплеш-экрана разрабатываемого приложения показан на рисунке 36.



Рисунок 36 – Сплеш-экран приложения

При нажатии на кнопку «Перейти» отображается экранная форма, представляющая собой журнальную сетку и содержащая

- название класса;
- ФИО учащихся классов;
- отметки учащихся по темам;
- темы, необходимые для изучения в данном учебном периоде;
- индикатор успеваемости напротив каждого ученика;
- индикатор прогноза неудовлетворительной отметки в ближайших темах.

Экранная форма показана на рисунке 37.

17:23							
Физика 7А							
Физика – наука о природе	Физические величины	Строение вещества	Движение частиц вещества	Агрегатные состояния вещества	Механическое движение	Инерция, масса, плотность	
Алексеева Василиса Ивановна							
—	—	—	—	—	—	—	
Васильева Евгения Константиновна							
4	4	4	5	—	—	—	
Зайцева Татьяна Михайловна							
4	3	4	4	—	—	—	
Кузнецов Евгений Павлович							
4	3	3	—	—	—	—	
Никитин Александр Николаевич							
3	4	4	—	—	—	—	
Смирнова Татьяна Михайловна							
5	4	—	—	—	—	—	
Соколов Матвей Олегович							
2	5	3	—	—	—	—	
Фёдоров Александр Максимович							
5	5	5	5	—	—	—	
Яковлев Игорь Геннадьевич							
4	4	5	4	—	—	—	

Рисунок 37 – Экранная форма с информацией об отметках учащихся

При нажатии на ячейку с темой урока отображается дополнительная информация о занятии: полное название темы и дата проведения урока. Модальное окно, содержащее информационное сообщение показано на рисунке 38.

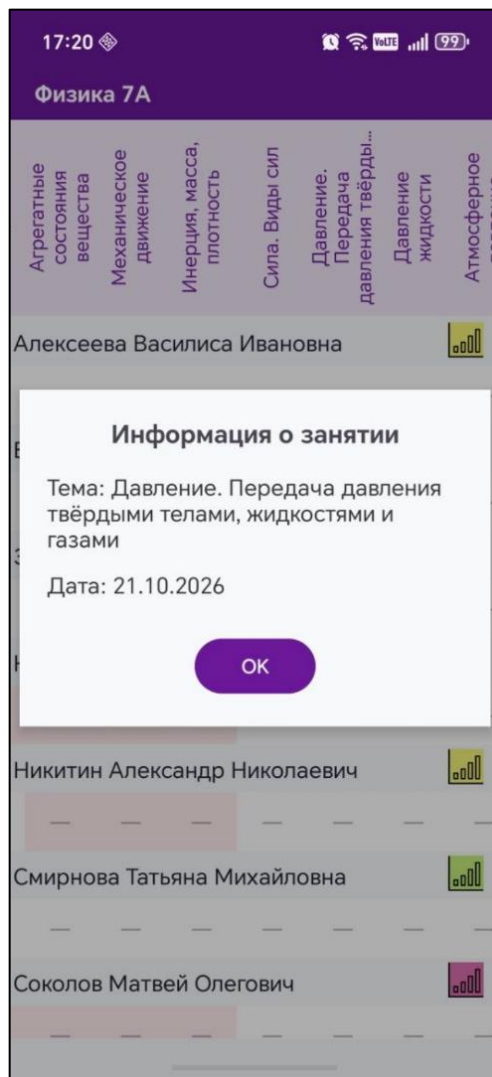


Рисунок 38 – Модальное окно с информацией об уроке

Заполнение журнальной сетки производится с помощью функции, загружающей имеющиеся отметки ученика, и функции, загружающей темы, по котором у учащегося прогнозируются неудовлетворительные отметки. Темы с плохим прогнозом выделяются розовым цветом.

```
private fun buildTable(students: List<Student>, lessons:
List<LessonInfo>) {
    tableManager.clear()
    tableManager.addHeader(lessons)
    students.forEach { student ->
        tableManager.addStudentRow(student, lessons) }
    loadExistingGrades(students, lessons)
```

```
loadPredictions(students, lessons)
}
```

Редактирование отметки осуществляется нажатием на оценку, требующую исправления, затем её стиранием и введением нового значения. Создание новой отметки осуществляется при нажатии на пустую ячейку в таблице напротив нужного учащегося и темы занятия. Создание новой отметки показан на рисунке 39.

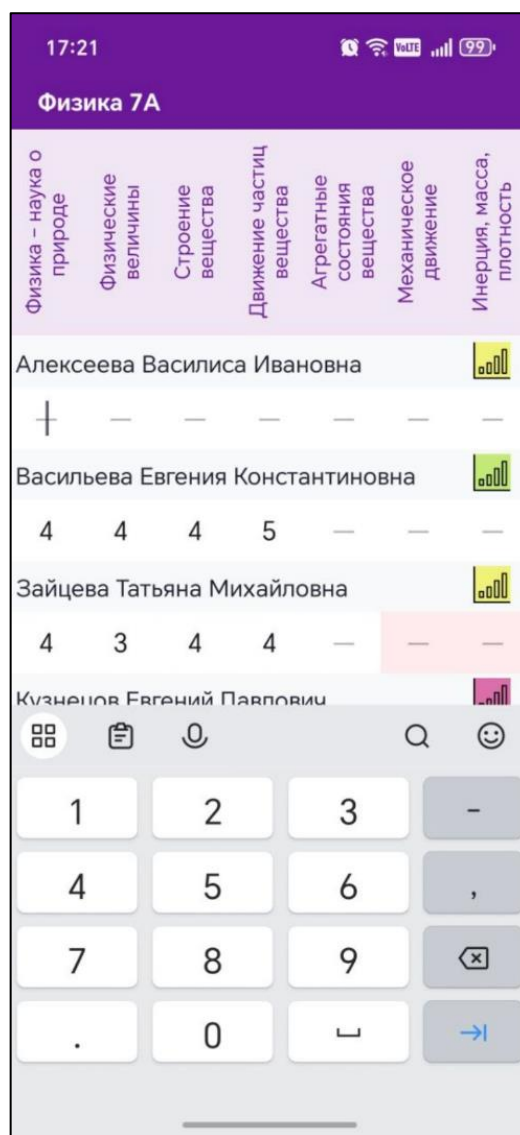


Рисунок 39 – Ввод новой отметки

Напротив каждого учащегося рисуется кнопка, выполняющая две функции: отображение уровня успеваемости учащегося и переход к оцениванию различных

факторов ученика, влияющих на его учебу. Подсветка кнопки в соответствии с уровнем успеваемости – решение, позволяющее передать Пользователю дополнительную информацию, лаконично её интегрируя в мобильный формат. При отсутствии оценки жизненные факторы учащегося кнопка отображается серым цветом. При наличии оценки – кнопка окрашивается в один из трех цветов в зависимости от результата оценки:

- красный цвет – низкая успеваемость учащегося;
- желтый цвет – средняя успеваемость учащегося;
- зеленый цвет – высокая успеваемость учащегося.

```
private fun getButtonColorForPerformance(performanceLevel: String?):
Int {
    return when {
        performanceLevel == null -> Color.parseColor("#9E9E9E")
        performanceLevel.contains("Высокий") -> Color.parseColor("#4CAF50")
        performanceLevel.contains("Средний") -> Color.parseColor("#FFC107")
        performanceLevel.contains("Низкий") -> Color.parseColor("#F44336")
        else -> Color.parseColor("#9E9E9E")
    }
}
```

При нажатии на кнопку напротив учащегося открывается экранная форма для оценки факторов, влияющих на образовательный процесс. Экран содержит следующие элементы:

- ФИО учащегося;
- 4 блока с критериями для оценки (количество часов сна, количество часов обучения, регулярность посещения занятий и социально-экономическое окружение);
- кнопка «Получить результат».

Каждый блок критерия содержит название фактора, текст с описанием влияния рассматриваемого фактора на учебный процесс и кнопок для выставления баллов за критерий. Увеличение или уменьшение баллов производится через

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		60

кнопки «+» и «-» соответственно. Причем Пользователь не может выставить баллы ниже 1 и выше 15 для критериев «Количество часов сна», «Количество часов обучения» или выше 10 для критериев «Регулярность посещений занятий», «Социально-экономическое окружение».

```
minusBtn.setOnClickListener {
    var current = scores[criterionId] ?: 1
    if (current > 1) {
        current--
        scores[criterionId] = current
        scoreTv.text = current.toString()
        saveScoreToFirestore(criterionId, current)
    }
}

plusBtn.setOnClickListener {
    var current = scores[criterionId] ?: 1
    if (current < maxValue) {
        current++
        scores[criterionId] = current
        scoreTv.text = current.toString()
        saveScoreToFirestore(criterionId, current)
    }
}
```

Интерфейс экранной формы для оценки факторов, влияющих на образовательный процесс продемонстрирован на рисунке 40.

После выставления баллов по каждому критерию необходимо нажать на кнопку «Получить результаты».

17:24

Васильева Евгения Константиновна

Количество часов сна

- 6 +

Сон помогает обрабатывать информацию, закреплять знания и улучшать способность запоминать информацию.

Количество часов обучения

- 9 +

Ученики, которые тратят больше времени на выполнение заданий, как правило, показывают более высокие результаты. Детальное изучение материала и практика способствуют лучшему пониманию предмета.

Регулярность посещений занятий

- 10 +

Высокий процент посещений обычно коррелирует с хорошей успеваемостью. Посещаемость — один из показателей, свидетельствующих о вовлечённости учеников и их заинтересованности в обучении.

Социально-экономическое окружение

- 8 +

Высокий показатель социально-экономического уровня в целом положительно сказывается на успеваемости и мотивации школьников. Материальные и образовательные ресурсы семьи способствуют доступу к качественному образованию. При этом связь между уровнем семьи и успеваемостью не всегда линейна и может опосредоваться дополнительными переменными: структурой семьи, образованием родителей, характером детско-родительских отношений, вовлечённостью в учебный процесс.

Получить результат

Рисунок 40 – Экранная форма с критериями для оценки

Система обрабатывает полученные данные и выводит результат классификации, необходимый для определения уровня успеваемости учащегося.

```
private fun showResultDialog(performanceLevel: String) {
```

```

val recommendations = getRecommendations(performanceLevel)

val dialogView =
LayoutInflater.from(this).inflate(R.layout.dialog_result, null)

val messageText =
dialogView.findViewById<TextView>(R.id.dialogMessage)
messageText.text = "$performanceLevel\n\n$recommendations"

val dialog = AlertDialog.Builder(this)
    .setView(dialogView)
    .setCancelable(false)
    .create()

```

В зависимости от уровня успеваемости (низкий, средний, высокий) отрисовывается модальное окно, содержащее полученный уровень успеваемости и рекомендации для данного уровня. Возможные варианты отображения модального окна с результатами показаны на рисунках 41-43.

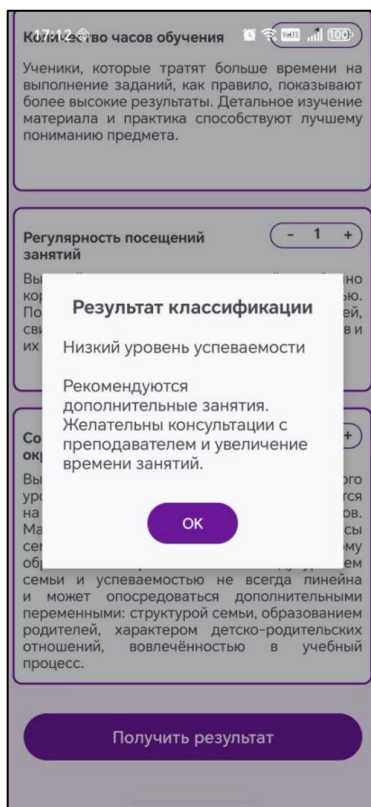


Рисунок 41 – Модальное окно при низком уровне успеваемости учащегося

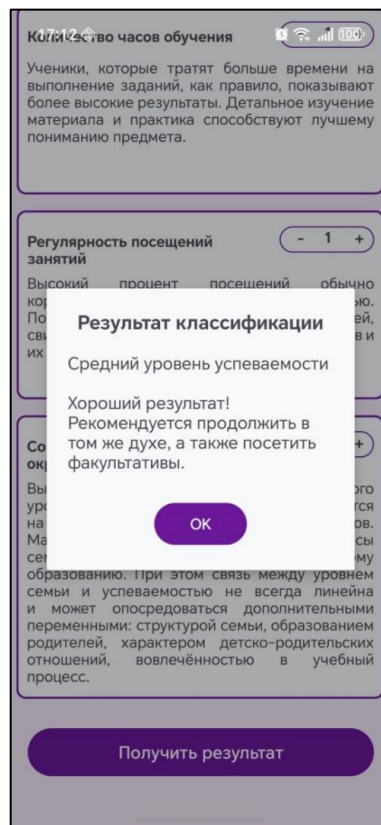


Рисунок 42 – Модальное окно при среднем уровне успеваемости учащегося

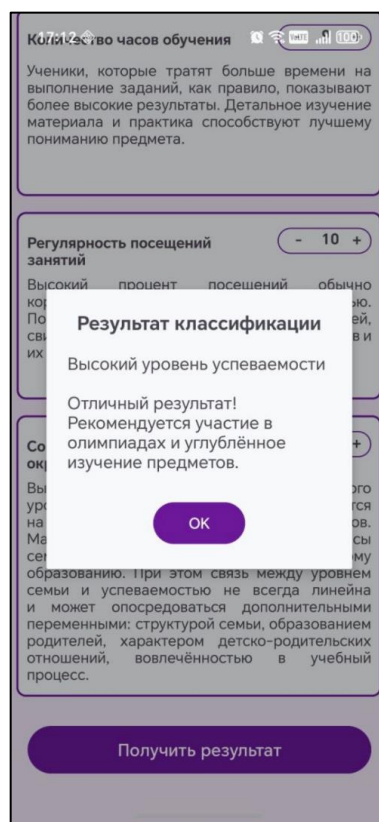


Рисунок 43 – Модальное окно при высоком уровне успеваемости учащегося

На рисунке 44 представлены все экраны приложения и установленны связи между ними.

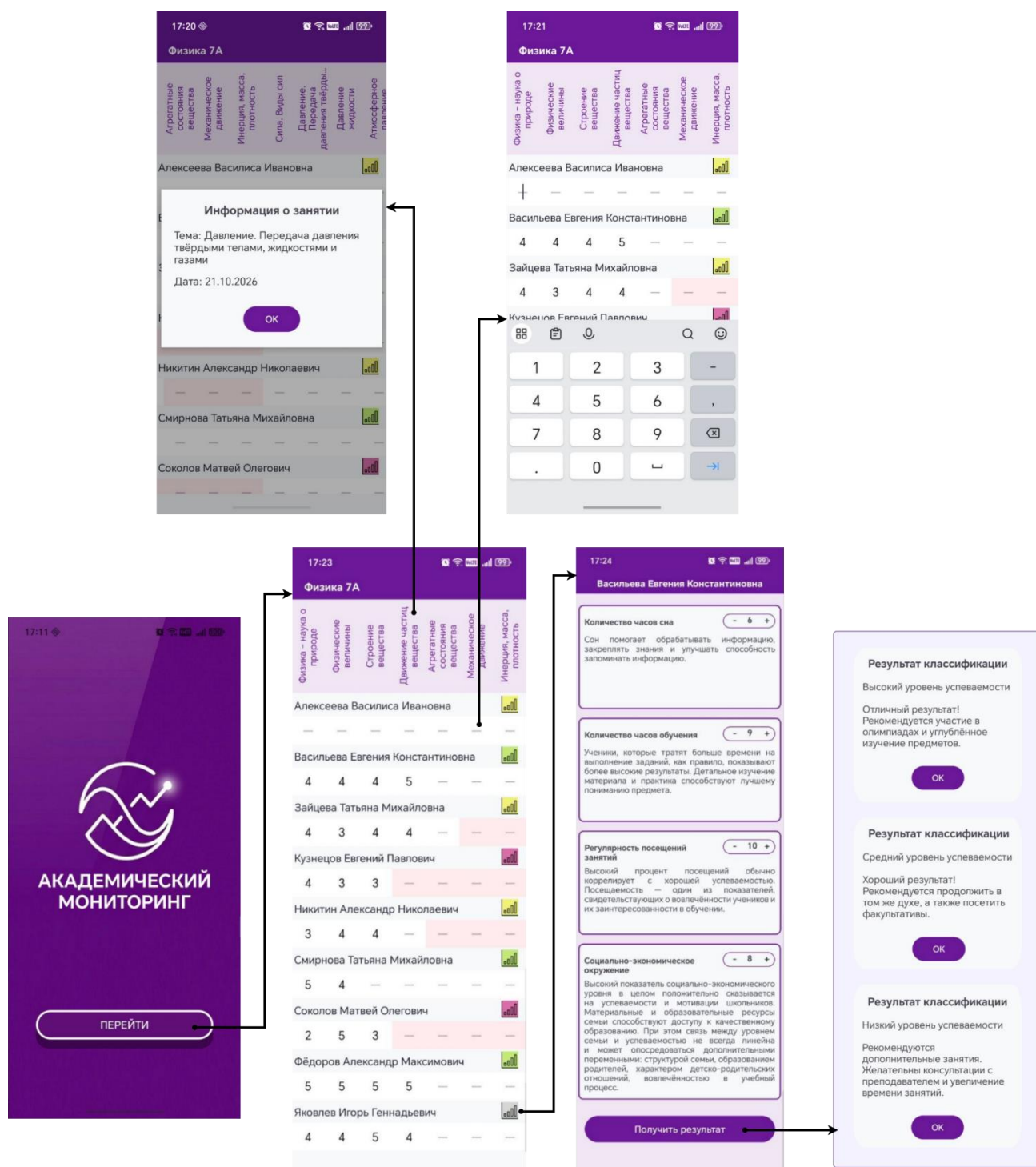


Рисунок 44 – Схема экранов приложения

Диаграмма последовательности показана на рисунке 45.

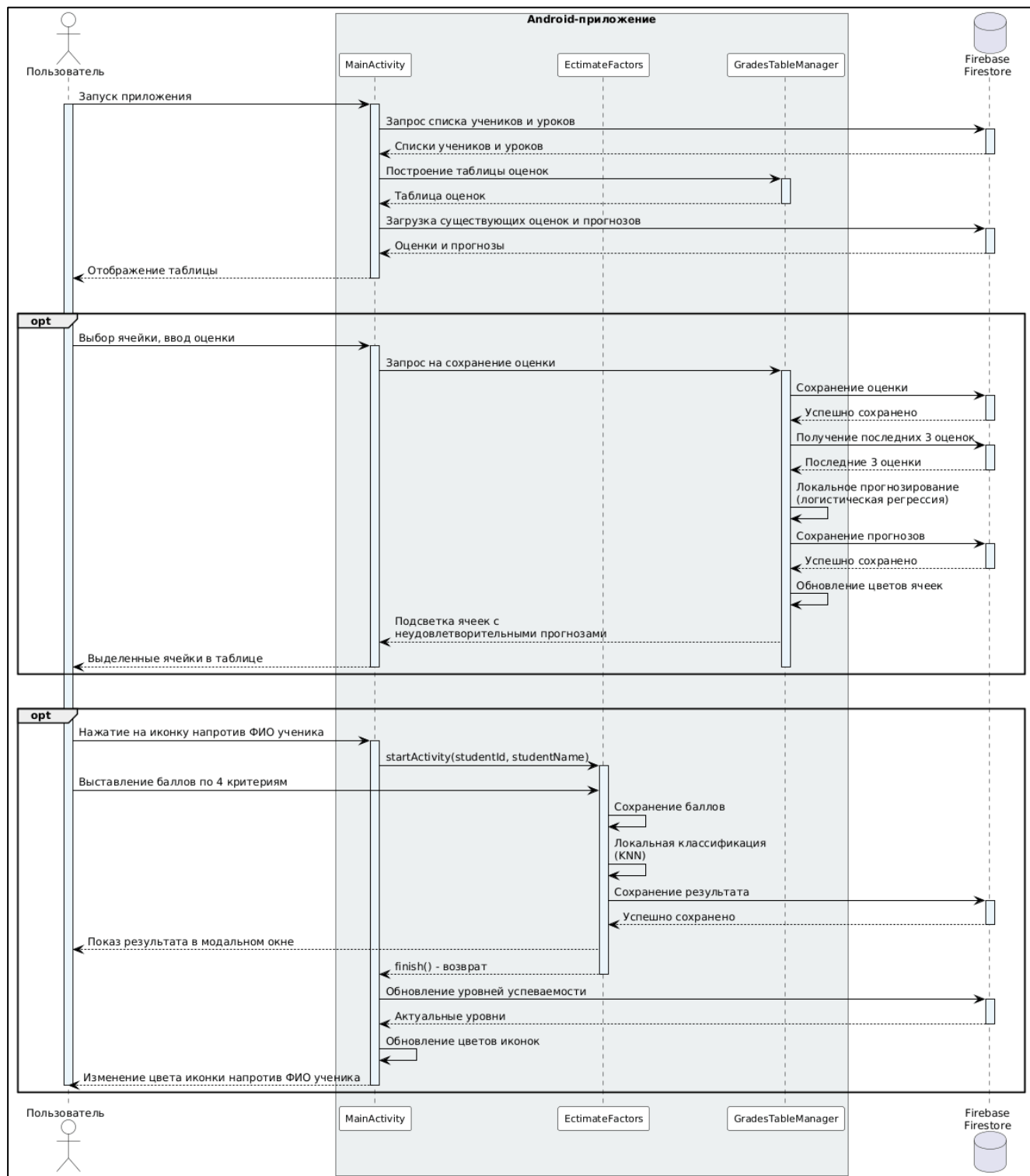


Рисунок 45 – Диаграмма последовательности

7 Тестирование

Произведено ручное тестирование приложения. Тестирование осуществлялось для следующих кейсов:

- выставление отметки и получение прогноза;
- удаление отметки и получение прогноза;
- оценка факторов, влияющих на учебный прогресс ученика.

7.1 Тест-кейс «Выставление отметки и получение прогноза»

Предусловие: Пользователь находится на экранной форме, отображающей журнальную сетку с отметками учащихся.

Требуемое условие для выполнения кейса: у учащегося, которому выставляется отметка, ранее уже имеется минимум две оценки, выставленным по последовательным темам.

Таблица 6 – Шаги тест-кейса «Выставление отметки и получение прогноза»

№	Описание шага	Ожидаемый результат	Фактический результат
1	Пользователь нажимает на пустую ячейку, следующую за имеющейся отметкой.	Система отображается курсор в выбранной ячейке и клавиатуру для ввода отметки.	Совпадает с ожидаемым.
2	Пользователь нажимает на цифру «2», «3», «4» или «5», соответствующую отметке, и нажимает на кнопку «готово» в виде «галочки», закрывает клавиатуру.	Система отображает набранную отметку в ячейке. Производится выделение цветом ячеек будущих тем, для которых прогнозируется неудовлетворительные отметки.	
	Сценарий завершен.		

7.2 Тест-кейс «Удаление отметки и получение прогноза»

Предусловие: Пользователь находится на экранной форме, отображающей журнальную сетку с отметками учащихся.

Требуемое условие для выполнения кейса: у учащегося, для которого выполняется удаление отметки, имеется не менее 4 оценок.

Таблица 7 – Шаги тест-кейса «Удаление отметки и получение прогноза»

№	Описание шага	Ожидаемый результат	Фактический результат
1	Пользователь нажимает ячейку, содержащую отметку «2», «3», «4» или «5».	Система отображается курсор в выбранной ячейке и клавиатуру для редактирования отметки.	Совпадает с ожидаемым.
2	Пользователь зажимает область отметки на 1-2 секунды.	Система отображает пустую выбранную ячейку. Производится выделение цветом ячеек будущих тем, для которых прогнозируется неудовлетворительные отметки.	
	Сценарий завершен.		

7.3 Тест-кейс «Оценка факторов, влияющих на учебный прогресс ученика»

Предусловие: Пользователь на экранной форме, отображающей журнальную сетку с отметками учащихся, нажимает на кнопку напротив ученика, для которого необходимо произвести оценку.

Требуемое условие для выполнения кейса: для учащегося, для которого выполняется оценка, ранее не проводилась оценка факторов, влияющих на образовательный процесс.

Таблица 8 – Шаги тест-кейса «Удаление отметки и получение прогноза»

№	Описание шага	Ожидаемый результат	Фактический результат
1	Пользователь нажимает на кнопки «+» или «-» для выставления баллов по критерию «Количество часов сна».	Система отображается набранные баллы Пользователем в блоке критерия «Количество часов сна».	Совпадает с ожидаемым.
2	Пользователь нажимает на кнопки «+» или «-» для выставления баллов по критерию «Количество часов обучения».	Система отображается набранные баллы Пользователем в блоке критерия «Количество часов обучения».	
3	Пользователь нажимает на кнопки «+» или «-» для выставления баллов по	Система отображается набранные баллы Пользователем в блоке критерия «Регулярность посещений занятий».	

№	Описание шага	Ожидаемый результат	Фактический результат
	критерию «Регулярность посещений занятий».		Совпадает с ожидаемым.
4	Пользователь нажимает на кнопки «+» или «-» для выставления баллов по критерию «Социально-экономическое окружение».	Система отображается набранные баллы Пользователем в блоке критерия «Социально-экономическое окружение».	
5	Пользователь нажимает на кнопку «Получить результаты».	Система отображает модальное окно с результатами, содержащее уровень успеваемости учащегося и рекомендации для данного уровня.	
6	Пользователь нажимает на кнопку «Ок».	Система закрывает модальное окно с результатами, происходит переход к экранной форме, отображающей журнальную сетку с отметками учащихся.	
	Сценарий завершен.		

8 Безопасность жизнедеятельности

Работа на персональном компьютере – привычная часть деятельности разработчиков программного обеспечения, но она связана со многими факторами, которые могут повлиять на здоровье. При написании и отладке программного обеспечения к таким факторам относятся:

- электрический ток;
- внешний шум и вибрация;
- электромагнитное излучение;
- статическое электричество;
- застоявшийся воздух;
- монотонные физические нагрузки с общим отсутствием движений рук, спины и шеи;
- постоянное напряжение для глаз.

8.1 Классификация рабочих мест

Помещения разделены на три категории по выбору мер защиты от поражения электрическим током: минимальная опасность, опасность повышена, и особенно опасно.

Минимальная опасность – сухость, отсутствие пыли, комнатная температура и антистатические полы (например, деревянные).

Повышенный риск – сырость (влажность помещения превышает более 70%), повышенная теплота (температура помещения более 35°C), токопроводящая пыль, токопроводящий пол. Возможность контакта с заземляющей конструкцией и корпусом оборудования.

Особенно опасно – влажность, близкая к 100%, химически активная среда или при наличии двух или более опасных условий, так же скопление других технических устройств, создающих свою опасность.

					ПГУ1.090301.4.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

В соответствии с пожарной и взрывопожарной опасностью дома делятся на следующие категории:

- А – взрывопожароопасность повышена;
- Б – взрывопожароопасность;
- В1-В4 – пожароопасность;
- Г – умеренная Пожароопасность;
- Д – пожароопасность понижена.

8.2 Санитарно-гигиенические требования

Организация рабочих мест и персональных компьютеров регулируется трудовым законодательством и СанПиНом. Предпочтительно, чтобы окна выходили на север или северо-восток, и были установлены жалюзи. Работа без естественного освещения допускается только по уважительным причинам. Искусственное освещение должно быть равномерным: 300-500 Люкс, индекс цветопередачи $R_a \geq 85\%$. Естественный свет должен падать слева, а окна и светильники не должны находиться в поле зрения или строго сзади. Яркость бликов на экране не превышает 40 кд/м², а высота потолка достигает 200 кд/м². Желательно, чтобы на столе стояла регулируемая настольная лампа.

Уровни шума и вибрации соответствуют государственным строительным нормам (СанПиН 1.2.3685-21). Шумное оборудование (принтеры, серверы) выносятся за пределы помещения. Параметры микроклимата для категорий (категории Ia и Ib) приведены в таблице.

Таблица 9 - Допустимые параметры микроклимата

Период года	Категория работ по уровню энергозатрат	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность в помещении, %	Скорость воздуха, м/с, не превышает
Холодный	Ia	23 - 25	22 – 26	55 - 35	0,15
	Iб	20 - 22	19 – 23	55 - 35	0,15
Теплый	Ia	24 - 26	23 – 27	55 - 35	0,15
	Iб	21 - 23	20 - 24	55 - 35	0,15

Площадь каждого рабочего места с жидкокристаллическим или плазменным дисплеем составляет не менее 4,5 квадратных метров. Расстояние между столами составляет от 2 метров, между двумя сторонами монитора - от 1,2 метра. Кресло подъемное и вращающееся, регулируется по высоте, сиденье и спинка наклонены, с неэлектризуемым покрытием.

При 8-часовых сменах программистам рекомендуется каждый час делать 15-минутный перерыв во время работы за ПК. Если вы не можете отдыхать, вам не следует работать непрерывно более 4 часов. Желательны кабинеты психологической релаксации и упражнения для глаз.

8.3 Электробезопасность

Согласно Санпин 2.2.2/2.4.1340-03, помещения с компьютерами должны иметь защитное заземление. Рабочее место не должно располагаться вблизи силовых кабелей, высоковольтных трансформаторов или оборудования, вызывающего помехи. Рекомендуется включать компьютер от отдельного источника питания.

Во время работы за ПК выполните следующие действия:

- проверьте надежность контактов трехпроводной розетки;
- при использовании двухпроводной сети подключите дисплей через соответствующее устройство;
- не ставьте системный блок на пол, под ноги, в пыльные или влажные места;
- не прикасайтесь к задней панели системного блока;
- устанавливайте компьютер на жесткую раму без вибрации;
- избегайте поверхностей, на которых скапливается статическое электричество (оргстекло, полированный лак).

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		72

8.4 Пожарная безопасность

Основными причинами пожаров в офисах являются неосторожное обращение с огнем и неисправности электрооборудования. Согласно статье 37 федерального закона "О пожарной безопасности", организация обязана проводить обучение пожарной безопасности, обслуживать системы защиты и средства пожаротушения, обеспечивать доступ к противопожарной защите и незамедлительно сообщать о пожарах и неисправностях.

Инструктажи могут быть вводными, в основном на рабочем месте, повторяющимися, незапланированными и целенаправленными. Сотрудников ознакомили с правилами содержания помещений, путями эвакуации, действиями в случае пожара, правилами вызова пожарных и использования огнетушащих средств.

В помещениях с компьютерами для тушения пожара используются холодильные огнетушители - они не повредят оборудованию. Порошок и углекислый газ нельзя использовать при напряжении выше 1 кВ и 10 кВ соответственно. В обычных условиях нельзя использовать воздушную пену и воду.

8.5 Расчет искусственного освещения

Было рассчитано помещение площадью 48 м² (длина 8 м, ширина 6 м, высота 3 м). Освещение, как правило, равномерное, с использованием коэффициента использования светового потока. Работа с ПК относится к третьему типу визуальной работы (высокоточной), с нормированной освещенностью 300-500 Люкс. Мы принимаем $E_n = 400$ лк.

Выбор светодиодной лампы-Грильято: световой поток 3756 лм, мощность 32 Вт, индекс цветопередачи 85%.

Световой поток светильника рассчитывается по формуле:

$$\Phi_{\text{л}} = \frac{E_n * S * z * k_3}{N * \eta}, \quad (3)$$

где E_n – нормативный уровень освещенности рабочей поверхности [лк];

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

S – площадь помещения [м^2];

z – коэффициент неравномерности освещения;

k_3 – коэффициент запаса;

N – количество светильников;

η – коэффициент использования светового потока.

Нормативный уровень освещенности E_n принят 400 лк. Площадь помещения – 48м^2 . Коэффициент неравномерности освещения z принимается равным 1,5. Коэффициент запаса k_3 учитывает запыленность помещения, снижение светового потока ламп в процессе эксплуатации. Для светодиодных ламп, используемых в офисных помещениях, значение коэффициента составляет 1.

Для определения коэффициента использования светового потока η находится индекс помещения:

$$i = \frac{A*B}{h_p*(A+B)}, \quad (4)$$

где A – длина помещения в плане;

B – ширина помещения в плане;

h_p – высота подвеса светильников над рабочей поверхностью.

Высота подвеса светильников составляет 2,2 м, длина помещения 8 м, ширина помещения 6 м. Индекс помещения тогда равен 1,55844.

При коэффициенте отражения: 70% от потолка (белый), 50% от стен (бежевый) и 20% от пола (светлый паркет) — коэффициент использования светового потока составляет $\eta=37\%$

Для расчета количества светильников в помещении используется формула (2), она выводится из формулы (1).

$$N = \frac{E_n*S*z*k_3}{\eta*\Phi_l}, \quad (5)$$

где E_n – уровень освещенности;

S – площадь помещения [м^2];

z – коэффициент неравномерности освещения;

					ПГУ1.090301.4.001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

k_3 – коэффициент запаса;

n – количество ламп в светильнике;

η – коэффициент использования светового потока

$\Phi_{\text{л}}$ – световой поток.

Тогда количество светильников в помещении равно: $N = 20,72 \approx 21$.

Фактический уровень освещенности $E_{\text{факт}}$ тогда равен 405,335 лк, значение находится в пределах нормы.

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		75

Заключение

В ходе выполнения выпускной квалификационной работы разработано мобильное приложение для отслеживания академических рисков учащихся. Приложение содержит две основные функции: прогноз неудовлетворительных отметок учащихся и определение уровня успеваемости учащегося.

Для реализации функционала приложения были проведены работы по выбору датасета для одной модели обучения – классификатор уровня успеваемости – и созданию синтетических данных для другой – прогноз оценок. Для полученных данных были проведены анализ и обработка для дальнейшего использования. После обучения моделей были достигнуты удовлетворительные результаты, достигающие 80% точности.

Работа приложения поддерживается облачной базой данных, что обеспечивает работу на различных устройствах с сохранением данных. А также возможно продолжение работы в офлайн-режиме.

Перспективы дальнейшего развития системы заключаются в повышении точности прогнозов и усовершенствовании пользовательского интерфейса для наиболее удобного пользования.

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		76

Список использованных источников

1. Шухман Александр Евгеньевич, Парфенов Денис Игоревич, Легашев Леонид Вячеславович, Гришина Любовь Сергеевна. Анализ и прогнозирование успеваемости обучающихся при использовании цифровой образовательной среды // Высшее образование в России. 2021. №8-9. [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://cyberleninka.ru/article/n/analiz-i-prognozirovanie-uspevaemosti-obuchayuschihsya-pri-ispolzovanii-tsifrovoy-obrazovatelnoy-sredy> (дата обращения: 21.04.2026).

2. Департамент образования и науки города Москвы. Дайджест новостей 23-24 августа 2023 год. [ЭЛЕКТРОННЫЙ РЕСУРС] URL: https://sch1359uv.mskobr.ru/attach_files/novostey-23---24-avgusta-2023.pdf (дата обращения 21.04.2026).

3. Цифровая образовательная платформа ЭлЖур. [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://eljur.ru/> (дата обращения 21.04.2026).

4. Электронный журнал ЭлЖур инструкция учителя и классного руководителя [ЭЛЕКТРОННЫЙ РЕСУРС] URL: https://shkola-gimnasia.crm.eduru.ru/media/2020/11/23/1245123492/Instrukciya_dlya_uchitelya_i_klassnogo_rukovoditelya.pdf (дата обращения 21.04.2026).

5. Московская электронная школа. [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://www.mos.ru/city/projects/mesh/> (дата обращения 21.04.2026).

6. Firebase Google's Mobile and Web App Development Platform. [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://firebase.google.com/> (дата обращения 21.04.2026).

7. Ram Kesavan, David Gay, Daniel Thevessen, Jimit Shah Firestore: The NoSQL Serverless Database for the Application Developer. [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://storage.googleapis.com/gweb-research2023-media/pubtools/7076.pdf> (дата обращения 21.04.2026).

8. Katarina Milojković, Miodrag Živković, Nebojša Bačanin, Džakula Agile multi-user android application development with firebase: authentication, authorization, and profile management. // Sinteza 2024. International scientific conference on information

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		77

technology, computer science, and data science. [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://portal.sinteza.singidunum.ac.rs/Media/files/2024/405-412.pdf> (дата обращения 21.04.2026).

9. Копытова М. А. Актуальность языка программирования Python // Экономика и социум. 2016. №10 (29). [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://cyberleninka.ru/article/n/aktualnost-yazyka-programmirovaniya-python> (дата обращения: 21.04.2026).

10. Зуенко Александр Анатольевич, Алмаматов Александр Анатольевич Программирование в ограничениях на языке Python с применением структур и алгоритмов алгебры кортежей // Труды Кольского научного центра РАН. 2014. №5 (24). [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://cyberleninka.ru/article/n/programmirovanie-v-ogranicheniyah-na-yazyke-python-s-primeneniem-struktur-i-algoritmov-algebry-kortezhey> (дата обращения: 21.04.2026).

11. Álaze Gabriel do Breviário, Jaine Marques de Souza, João Batista Lucena, Logan Faedda Rago Potentialities and challenges do google colab to machine learning and big data analytics // Current Scientific Journal. Internationally Indexed Scientific Journal. [ЭЛЕКТРОННЫЙ РЕСУРС] URL: https://www.researchgate.net/profile/Alaze-Breviario-2/publication/395915111_potentialities_and_challenges_do_google_colab_to_machine_learning_and_big_data_analytics/links/68d7ad3bffdca73694b39d7a/potentialities-and-challenges-do-google-colab-to-machine-learning-and-big-data-analytics.pdf (дата обращения 21.04.2026).

12. firebase-admin 7.4.0 [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://pypi.org/project/firebase-admin/> (дата обращения 21.04.2026).

13. Милихин А.М. Преимущества использования языка программирования Kotlin в Android разработке // наука. технологии. инновации: сборник статей по итогам Международной научнопрактической конференции (Магнитогорск, 23 мая 2019 г.). - Стерлитамак: АМИ, 2019. - 104 с.

14. Цхошвили Д. З., Иванова Н. А. Возможности среды Eclipse для разработки интерфейсов мобильных приложений // Ученые записки Брянского

государственного университета. 2016. №4. URL: <https://cyberleninka.ru/article/n/vozmozhnosti-sredy-eclipse-dlya-razrabotki-interfeysov-mobilnyh-prilozheniy> (дата обращения: 21.04.2026).

15. Eclipse IDE The Leading Open Platform for Professional Developers [ЭЛЕКТРОННЫЙ РЕСУРС] URL: <https://eclipseide.org/> (дата обращения 21.04.2026).

16. Бурзуева Наталья Новрузовна, Мостовой Яков Анатольевич Анализ надежности среды разработки Android Studio // Вестник евразийской науки. 2017. №6 (43). URL: <https://cyberleninka.ru/article/n/analiz-nadezhnosti-sredy-razrabotki-android-studio> (дата обращения: 21.04.2026).

17. Федеральная рабочая программа основного общего образования физика (базовый уровень) для 7–9 классов образовательных организаций [Электронный ресурс]. – URL: https://edsoo.ru/wp-content/uploads/2025/06/12_frp_fizika-7-9-klassy_baza_23062025_itog-na-sajt.pdf (дата обращения: 21.04.2026).

18. Конкурсная платформа по исследованию данных Kaggle Machine Learning Competition Platform (Google). 2020 [Электронный ресурс]. – URL: <https://www.kaggle.com/> (дата обращения: 21.04.2026).

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		79

Приложение А

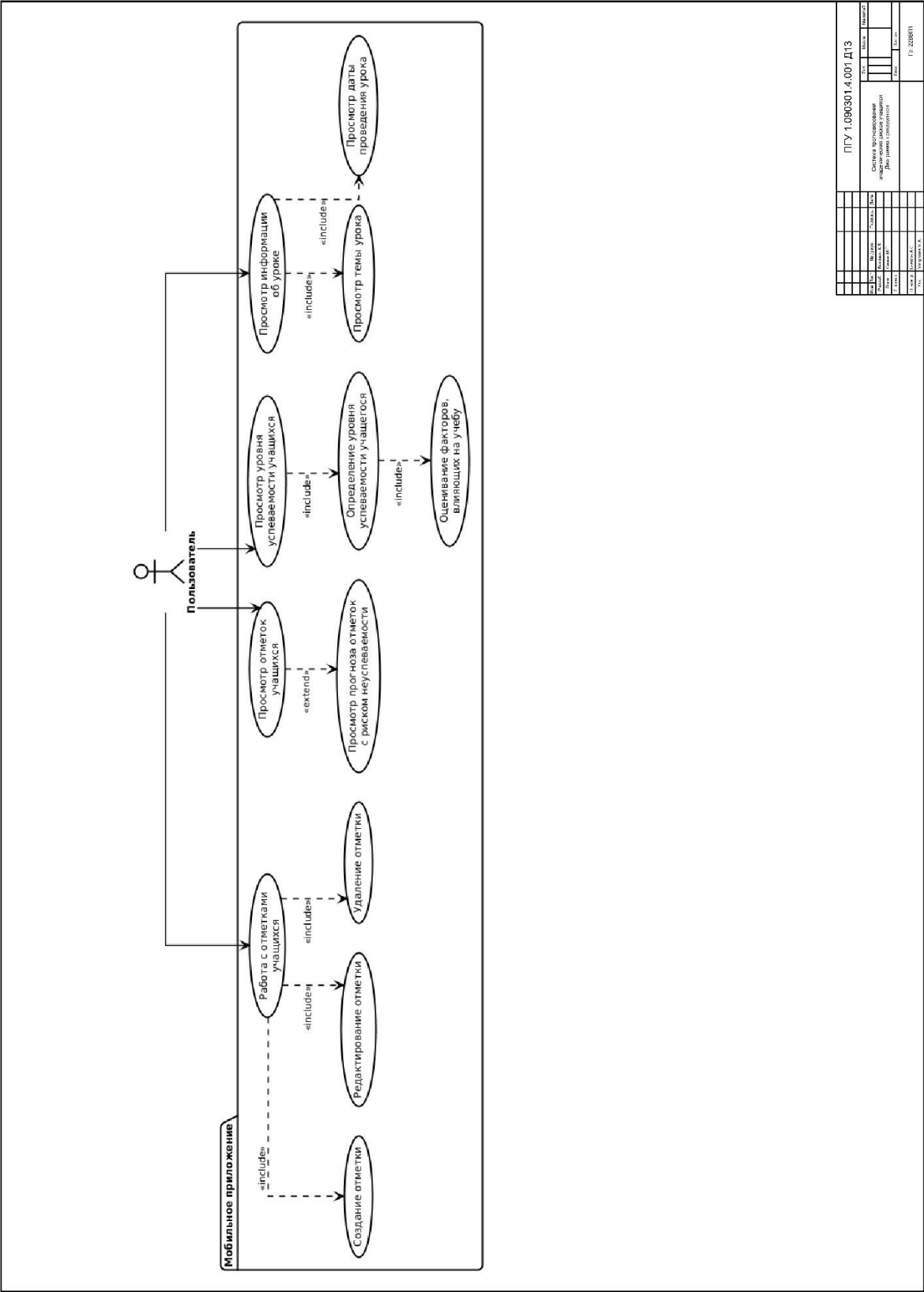
(обязательное)

Система прогнозирования академических рисков учащихся

Графическая часть

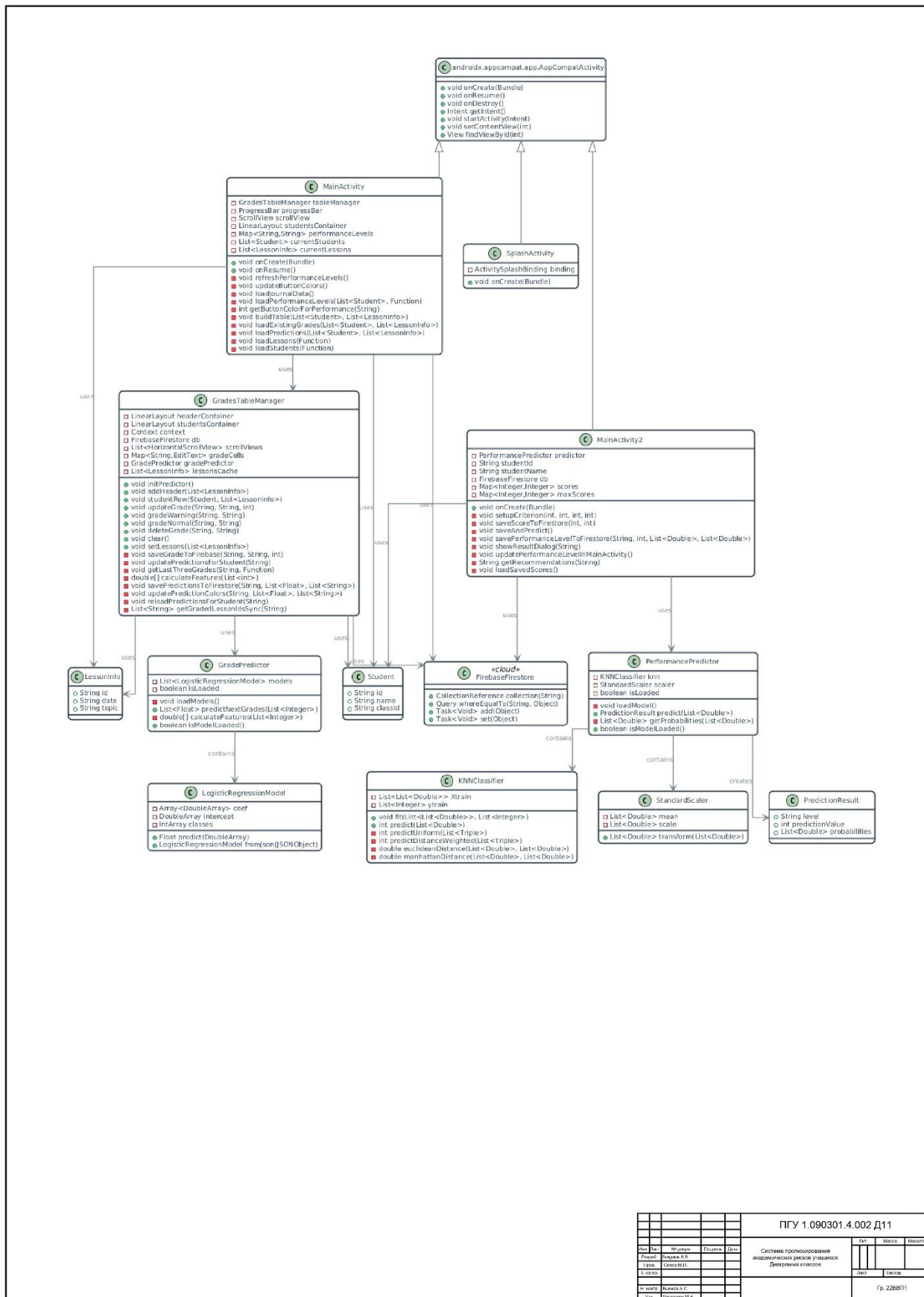
					ПГУ1.090301.4.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – Диаграмма прецедентов

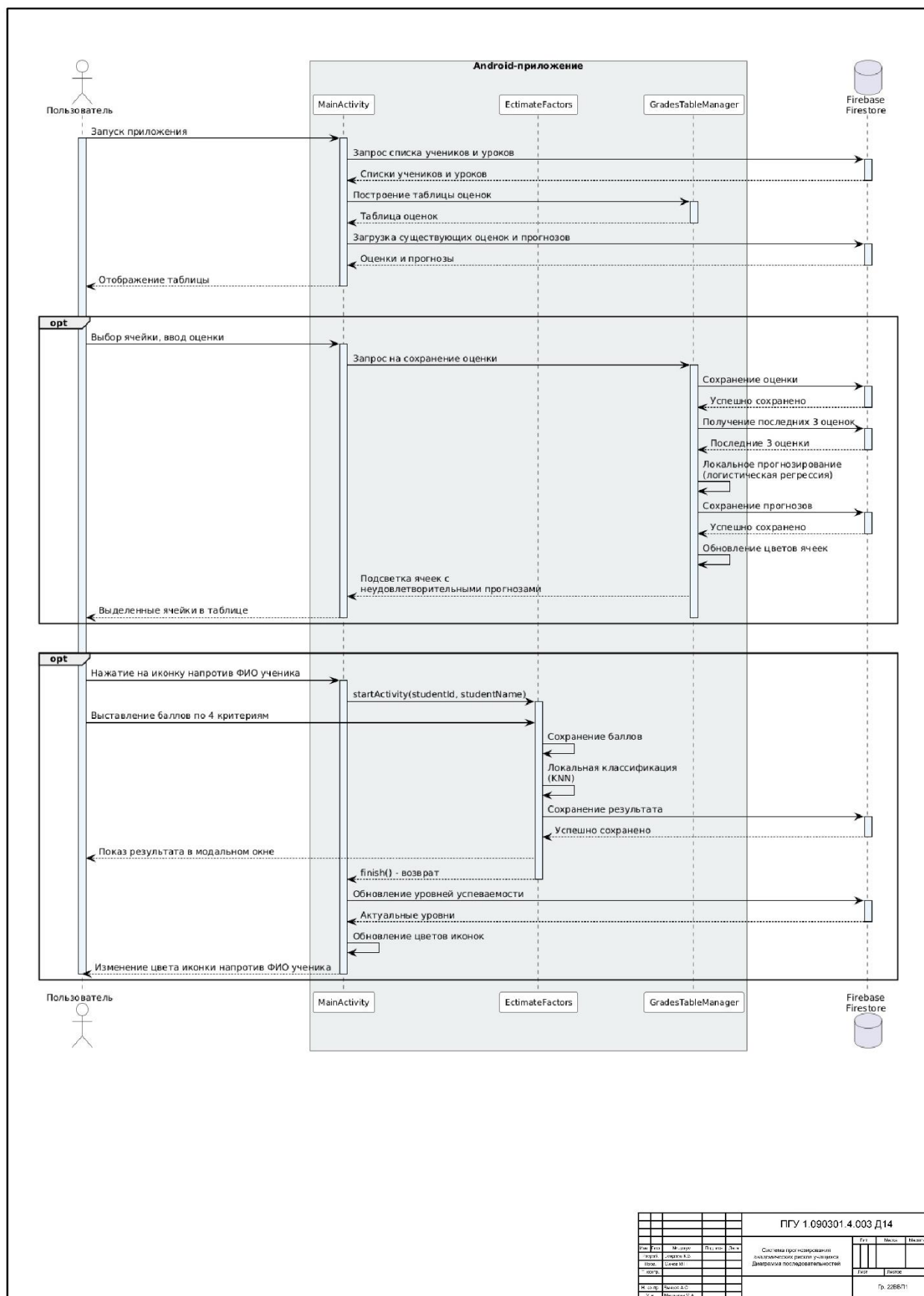


ПГУ 1.090301.4.001 Д13									
Изм.	Лист	№ докум.	Подпись	Дата	Изм.	Лист	№ докум.	Подпись	Дата
Описание приложения					История изменений				
Детальное описание					Детальное описание				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				
Изм.					Изм.				

Приложение А.2 – Диаграмма классов

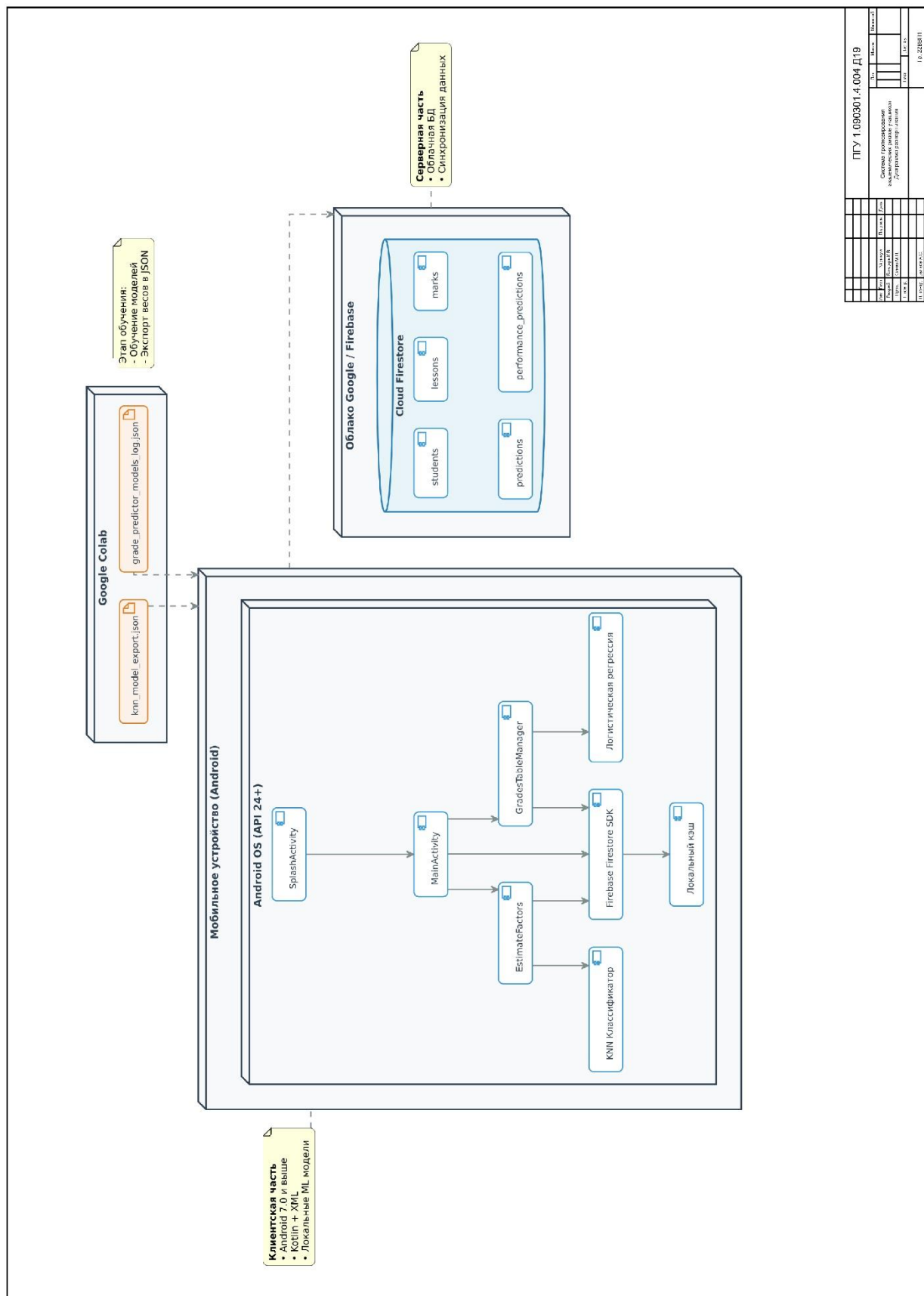


Приложение А.3 – Диаграмма последовательностей



ПГУ 1.090301.4.003 Д14									
Имя	Фамилия	Имя отчество	Пол	Дата	Подпись	Место	Дата	Подпись	Место
Имя	Фамилия	Имя отчество	Пол	Дата	Подпись	Место	Дата	Подпись	Место
Имя	Фамилия	Имя отчество	Пол	Дата	Подпись	Место	Дата	Подпись	Место
Имя	Фамилия	Имя отчество	Пол	Дата	Подпись	Место	Дата	Подпись	Место

Приложение А.4 – Диаграмма развертывания



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.4.001 ПЗ

Лист

84

Приложение Б

(обязательное)

Система прогнозирования академических рисков учащихся

Листинг мобильного приложения

					ПГУ1.090301.4.001 ПЗ	Лист
						85
Изм.	Лист	№ докум.	Подпись	Дата		

Файл MainActivity.kt

```
package com.example.studentenvironment

import android.content.Intent
import android.graphics.Color
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.ImageButton
import android.widget.LinearLayout
import android.widget.ProgressBar
import android.widget.ScrollView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.firestore.FirebaseFirestore

class MainActivity : AppCompatActivity() {

    private lateinit var tableManager: GradesTableManager
    private lateinit var progressBar: ProgressBar
    private lateinit var scrollView: ScrollView
    private lateinit var studentsContainer: LinearLayout

    private val performanceLevels = mutableMapOf<String, String>()
    private var currentStudents: List<Student> = emptyList()
    private var currentLessons: List<LessonInfo> = emptyList()

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

        val headerContainer =
            findViewById<LinearLayout>(R.id.headerContainer)
        studentsContainer = findViewById(R.id.studentsContainer)
```

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		86

```

        progressBar = findViewById(R.id.progressBar)
        scrollView = findViewById(R.id.scrollView)
        tableManager = GradesTableManager(headerContainer,
studentsContainer, this)

        tableManager.initPredictor()
        loadJournalData()
    }

    override fun onResume() {
        super.onResume()
        if (::tableManager.isInitialized &&
currentStudents.isNotEmpty()) {
            refreshPerformanceLevels()
        }
    }

    private fun refreshPerformanceLevels() {
        val studentIds = currentStudents.map { it.id }
        if (studentIds.isEmpty()) return

        val db = FirebaseFirestore.getInstance()
        db.collection("performance_predictions")
            .whereIn("studentId", studentIds)
            .get()
            .addOnSuccessListener { documents ->

                performanceLevels.clear()

                for (document in documents) {
                    val studentId = document.getString("studentId")
?: continue

                    val performanceLevel =

```

					ПГУ1.090301.4.001 ПЗ	Лист
						87
Изм.	Лист	№ докум.	Подпись	Дата		

```

document.getString("performance_level") ?: continue
        performanceLevels[studentId] = performanceLevel
    }

    //обновление UI
    runOnUiThread {
        updateButtonColors()
    }
}

private fun updateButtonColors() {
    for (i in 0 until studentsContainer.childCount) {
        val studentBlock = studentsContainer.getChildAt(i) as?
LinearLayout ?: continue
        val nameRow = studentBlock.getChildAt(0) as?
LinearLayout ?: continue
        val profileButton = nameRow.getChildAt(1) as?
ImageButton ?: continue

        val studentId = profileButton.tag as? String ?: continue

        val performanceLevel = performanceLevels[studentId]

        val color =
getColorForPerformance(performanceLevel)
        profileButton.setBackgroundColor(color)
    }
}

private fun loadJournalData() {
    progressBar.visibility = View.VISIBLE
    scrollView.visibility = View.GONE
}

```

```

loadLessons { lessons ->
    currentLessons = lessons
    loadStudents { students ->
        currentStudents = students
        loadPerformanceLevels(students) {
            buildTable(students, lessons)
            updateButtonColors()
            progressBar.visibility = View.GONE
            scrollView.visibility = View.VISIBLE
        }
    }
}

private fun loadPerformanceLevels (students: List<Student>,
onComplete: () -> Unit) {
    if (students.isEmpty()) {
        onComplete()
        return
    }

    val studentIds = students.map { it.id }
    val db = FirebaseFirestore.getInstance()

    db.collection("performance_predictions")
        .whereIn("studentId", studentIds)
        .get()

        .addOnSuccessListener { documents ->
            for (document in documents) {
                val studentId = document.getString("studentId")
                ?: continue

                val performanceLevel =

```

```

document.getString("performance_level") ?: continue
        performanceLevels[studentId] = performanceLevel
    }
    onComplete()
}
.addOnFailureListener { e ->
    onComplete()
}
}

```

```

private fun getButtonColorForPerformance(performanceLevel:
String?): Int {
    return when {
        performanceLevel == null -> Color.parseColor("#D6D6D4")
        performanceLevel.contains("Высокий") ->
Color.parseColor("#C4EA77")
        performanceLevel.contains("Средний") ->
Color.parseColor("#F2F27B")
        performanceLevel.contains("Низкий") ->
Color.parseColor("#DB6FAC")
        else -> Color.parseColor("#D6D6D4")
    }
}

```

```

private fun buildTable(students: List<Student>, lessons:
List<LessonInfo>) {
    tableManager.setLessons(lessons)
    tableManager.clear()
    tableManager.addHeader(lessons)

    students.forEach { student ->
        tableManager.studentRow(student, lessons)
    }
}

```

					ПГУ1.090301.4.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }
    loadExistingGrades(students, lessons)
    loadPredictions(students, lessons)
}

private fun loadExistingGrades(students: List<Student>, lessons:
List<LessonInfo>) {
    val studentIds = students.map { it.id }
    if (studentIds.isEmpty()) return

    val db = FirebaseFirestore.getInstance()
    db.collection("marks")
        .whereIn("studentId", studentIds)
        .get()
        .addOnSuccessListener { documents ->
            for (document in documents) {
                val studentId = document.getString("studentId")
?: continue

                val lessonId = document.getString("lessonId") ?:
continue

                val score = document.getLong("score")?.toInt()
?: continue

                tableManager.updateGrade(studentId, lessonId,
score)

            }
        }
        .addOnFailureListener { e ->
            Log.w("Marks", "Ошибка загрузки оценок", e)
        }
}

private fun loadPredictions(students: List<Student>, lessons:
List<LessonInfo>) {

```

```

        for (student in students) {
            for (lesson in lessons) {
                tableManager.gradeNormal(student.id, lesson.id)
            }
        }

val db = FirebaseFirestore.getInstance()
db.collection("predictions")
    .whereIn("studentId", students.map { it.id })
    .get()
    .addOnSuccessListener { documents ->
        for (document in documents) {
            val studentId = document.getString("studentId")
?: continue

            val lessonId = document.getString("lessonId") ?:
continue

            val predictedScore =
document.getDouble("predictedScore")?.toInt() ?: continue

            if (predictedScore == 1) {
                tableManager.gradeWarning(studentId,
lessonId)

            }
        }
    }

}

private fun loadLessons(callback: (List<LessonInfo>) -> Unit) {
    val db = FirebaseFirestore.getInstance()
    db.collection("lessons")
        .get()
        .addOnSuccessListener { documents ->
            val lessons = documents.map { doc ->
                LessonInfo(
                    id = doc.id,

```

```

        date = doc.getString("date") ?: "",
        topic = doc.getString("topic") ?: ""
    )
}
callback(lessons)
}
}

private fun loadStudents(callback: (List<Student>) -> Unit) {
    val db = FirebaseFirestore.getInstance()
    db.collection("students")
        .get()
        .addOnSuccessListener { documents ->
            val students = documents.map { doc ->
                Student(
                    id = doc.id,
                    name = doc.getString("name") ?: "",
                    classId = "10a"
                )
            }
            callback(students)
        }
}
}
}

```

Файл EstimateFactors.kt

```

package com.example.studentenvironment

import android.content.Intent
import android.os.Bundle
import android.view.LayoutInflater
import android.view.WindowManager
import android.widget.Button
import android.widget.TextView

```

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		93

```

import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.firestore.FirebaseFirestore
import
com.example.studentenvironment.modell1.PerformancePredictor

class EstimateFactors : AppCompatActivity() {
    private lateinit var predictor: PerformancePredictor
    private lateinit var studentId: String
    private lateinit var studentName: String
    private val db = FirebaseFirestore.getInstance()
    private val scores = mutableMapOf<Int, Int>()

    private val maxScores = mapOf(
        1 to 15, //сон
        2 to 15, //учеба
        3 to 10, //посещение
        4 to 10 //соц-эконом
    )

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main2)

        predictor = PerformancePredictor(this)
        //фio
        studentId = intent.getStringExtra("studentId") ?: ""
        studentName = intent.getStringExtra("studentName") ?:
""

        findViewById<TextView>(R.id.textView3).text =
studentName

```

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		94

```

        //кнопки +/-
        setupCriterion(1, R.id.textView14, R.id.textView15,
R.id.textView16)
        setupCriterion(2, R.id.textView17, R.id.textView18,
R.id.textView19)
        setupCriterion(3, R.id.textView20, R.id.textView21,
R.id.textView22)
        setupCriterion(4, R.id.textView23, R.id.textView24,
R.id.textView25)

loadSavedScores()

//"Получить результат"
val btnGetResult =
findViewById<Button>(R.id.btnGetResult)
    btnGetResult.setOnClickListener {
        saveAndPredict()
    }
}

private fun setupCriterion(criterionId: Int, minusBtnId:
Int, scoreTvId: Int, plusBtnId: Int) {
    val minusBtn = findViewById<TextView>(minusBtnId)
    val plusBtn = findViewById<TextView>(plusBtnId)
    val scoreTv = findViewById<TextView>(scoreTvId)

    scores[criterionId] = 1
    scoreTv.text = "1"

    val maxValue = maxScores[criterionId] ?: 10

    minusBtn.setOnClickListener {
        var current = scores[criterionId] ?: 1
        if (current > 1) {

```

```

        current--
        scores[criterionId] = current
        scoreTv.text = current.toString()
        saveScoreToFirestore(criterionId, current)
    }
}

plusBtn.setOnClickListener {
    var current = scores[criterionId] ?: 1
    if (current < maxValue) {
        current++
        scores[criterionId] = current
        scoreTv.text = current.toString()
        saveScoreToFirestore(criterionId, current)
    }
}

private fun saveScoreToFirestore(criterionId: Int, value:
Int) {
    if (studentId.isEmpty()) return

    val data = hashMapOf(
        "studentId" to studentId,
        "criterionId" to criterionId,
        "value" to value,
        "timestamp" to System.currentTimeMillis()
    )

    db.collection("student_scores")
        .document("${studentId}_criterion${criterionId}")
        .set(data)
}

```

```

private fun saveAndPredict() {
    if (studentId.isEmpty()) {
        return
    }

    val classificationData = hashMapOf(
        "studentId" to studentId,
        "studentName" to studentName,
        "criterion1" to (scores[1] ?: 1),
        "criterion2" to (scores[2] ?: 1),
        "criterion3" to (scores[3] ?: 1),
        "criterion4" to (scores[4] ?: 1),
        "timestamp" to System.currentTimeMillis()
    )

    db.collection("student_classification")
        .document(studentId)
        .set(classificationData)

    val scoresList = listOf(
        (scores[1] ?: 1).toDouble(),
        (scores[2] ?: 1).toDouble(),
        (scores[3] ?: 1).toDouble(),
        (scores[4] ?: 1).toDouble()
    )

    if (!predictor.isModelLoaded()) {
        return
    }

    val result = predictor.predict(scoresList)

```

```

        savePerformanceLevelToFirestore(result.level,
            result.predictionValue,
            result.probabilities, scoresList)

        showResultDialog(result.level)
    }

    private fun
savePerformanceLevelToFirestore(performanceLevel: String,
predictionValue: Int, probabilities: List<Double>, scoresList:
List<Double>) {
    val data = hashMapOf(
        "studentId" to studentId,
        "studentName" to studentName,
        "performance_level" to performanceLevel,
        "prediction" to predictionValue,
        "probabilities" to hashMapOf(
            "low" to probabilities[0],
            "medium" to probabilities[1],
            "high" to probabilities[2]
        ),
        "scores" to hashMapOf(
            "criterion1" to scoresList[0].toInt(),
            "criterion2" to scoresList[1].toInt(),
            "criterion3" to scoresList[2].toInt(),
            "criterion4" to scoresList[3].toInt()
        ),
        "timestamp" to System.currentTimeMillis(),
        "status" to "completed"
    )

    db.collection("performance_predictions")
        .document(studentId)

```

```

        .set(data)

    }

    private fun showResultDialog(performanceLevel: String) {
        val recommendations =
getRecommendations(performanceLevel)

        val dialogView =
LayoutInflater.from(this).inflate(R.layout.dialog_result, null)

        val messageText =
dialogView.findViewById<TextView>(R.id.dialogMessage)
        messageText.text =
"$performanceLevel\n\n$recommendations"

        val dialog = AlertDialog.Builder(this)
            .setView(dialogView)
            .setCancelable(false)
            .create()

        val okButton =
dialogView.findViewById<Button>(R.id.dialogButton)
        okButton.setOnClickListener {
            dialog.dismiss()
            updatePerformanceLevelInMainActivity()
            finish()
        }

        dialog.show()

        val window = dialog.window
        window?.setLayout(

```

```

        (resources.displayMetrics.widthPixels *
0.9).toInt(),
        WindowManager.LayoutParams.WRAP_CONTENT
    )
}

private fun updatePerformanceLevelInMainActivity() {
    val intent = Intent()
    intent.putExtra("updated_student_id", studentId)
    setResult(RESULT_OK, intent)
}

private fun getRecommendations(level: String): String {
    return when {
        level.contains("Низкий") -> "Рекомендуются
дополнительные занятия.\nЖелательны консультации с преподавателем и
увеличение времени занятий."
        level.contains("Средний") -> "Хороший
результат!\nРекомендуется продолжить в том же духе, а также посетить
факультативы."
        level.contains("Высокий") -> "Отличный
результат!\nРекомендуется участие в олимпиадах и углублённое
изучение предметов."
        else -> "Продолжайте заниматься!"
    }
}

private fun loadSavedScores() {
    if (studentId.isEmpty()) return

    db.collection("student_classification")
        .document(studentId)
        .get()
        .addOnSuccessListener { document ->

```

					ПГУ1.090301.4.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (document.exists()) {
            val saved1 =
document.getLong("criterion1")?.toInt() ?: 1
            val saved2 =
document.getLong("criterion2")?.toInt() ?: 1
            val saved3 =
document.getLong("criterion3")?.toInt() ?: 1
            val saved4 =
document.getLong("criterion4")?.toInt() ?: 1

            scores[1] = saved1
            scores[2] = saved2
            scores[3] = saved3
            scores[4] = saved4

findViewById<TextView>(R.id.textView15).text = saved1.toString()

findViewById<TextView>(R.id.textView18).text = saved2.toString()

findViewById<TextView>(R.id.textView21).text = saved3.toString()

findViewById<TextView>(R.id.textView24).text = saved4.toString()

        }
    }
}
}

```

Файл GradesTableManager.kt

```

package com.example.studentenvironment

import android.content.Context
import android.graphics.Color
import android.util.Log

```

					ПГУ1.090301.4.001 ПЗ	Лист
						101
Изм.	Лист	№ докум.	Подпись	Дата		

```

import android.view.Gravity
import android.widget.EditText
import android.widget.FrameLayout
import android.widget.HorizontalScrollView
import android.widget.ImageButton
import android.widget.LinearLayout
import android.widget.TableLayout
import android.widget.TableRow
import android.widget.TextView
import android.widget.Toast
import com.google.firebase.firestore.FirebaseFirestore
import android.content.Intent
import android.graphics.drawable.GradientDrawable
import android.view.KeyEvent
import android.widget.ImageView
import androidx.core.util.TypedValueCompat.dpToPx
import com.example.studentenvironment.EstimateFactors
import com.google.firebase.firestore.FieldValue
import kotlin.math.roundToInt
import com.google.firebase.firestore.Query
import com.example.studentenvironment.model1.GradePredictor
import android.view.LayoutInflater
import androidx.appcompat.app.AlertDialog
import android.widget.Button

class GradesTableManager(
    private val headerContainer: LinearLayout,    // темы
    private val studentsContainer: LinearLayout,  // ученики
    private val context: Context

) {
    private val db = FirebaseFirestore.getInstance()

```

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		102

```

private val scrollViews = mutableListOf<HorizontalScrollView>()
private val gradeCells = mutableMapOf<String, EditText>()

private lateinit var gradePredictor: GradePredictor

private fun Int.toPx(): Int = (this *
context.resources.displayMetrics.density).toInt()

fun initPredictor() {
    gradePredictor = GradePredictor(context)
}

// шапка таблицы
fun addHeader(lessons: List<LessonInfo>) {

    val headerBlock = LinearLayout(context).apply {
        orientation = LinearLayout.HORIZONTAL
        setPadding(0, 0, 0, 8)
        layoutParams = LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT
        )
    }

    // скролл горизонт
    val scrollView = HorizontalScrollView(context).apply {
        layoutParams = LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT
        )
        isHorizontalScrollBarEnabled = false
    }
}

```

```

    }

    val topicsContainer = LinearLayout(context).apply {
        orientation = LinearLayout.HORIZONTAL
        layoutParams = LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT
        )
    }

    lessons.forEach { lesson ->
        topicsContainer.addView(verticalTopics(lesson))
    }

    scrollView.addView(topicsContainer)
    headerBlock.addView(scrollView)
    headerContainer.addView(headerBlock)

    scrollViews.add(scrollView)

    scrollView.setOnScrollChangeListener { _, scrollX, _, _, _ -
>
        scrollViews.forEach { other ->
            if (other != scrollView) {
                other.scrollTo(scrollX, 0)
            }
        }
    }
}

//тема, написание снизу вверх
private fun verticalTopics(lesson: LessonInfo): FrameLayout {
    val container = FrameLayout(context).apply {
        layoutParams = LinearLayout.LayoutParams(

```

```

        180,
        500,
    )
    setBackgroundColor(Color.parseColor("#F1E6F5"))
    setPadding(4, 4, 4, 4)
}

val textView = TextView(context).apply {
    text = lesson.topic
    gravity = Gravity.CENTER
    setTextColor(Color.parseColor("#6b1799"));
    textSize = 14f

    // текст в -90 градус
    rotation = -90f
    layoutParams = FrameLayout.LayoutParams(
        500,
        180
    )

    (layoutParams as FrameLayout.LayoutParams).gravity =
Gravity.CENTER
    maxLines = 3
    ellipsize = android.text.TextUtils.TruncateAt.END
    setPadding(25, 0, 25, 0)
    includeFontPadding = false

    setOnClickListener {
        dialogTopicInfo(lesson.topic, lesson.date)
    }
}

container.addView(textView)

```

```

        return container
    }

    private fun dialogTopicInfo(topic: String, date: String) {
        val activity = context as?
androidx.appcompat.app.AppCompatActivity
        if (activity == null) {
            return
        }

        activity.runOnUiThread {
            val dialogView =
LayoutInflater.from(activity).inflate(R.layout.dialog_info, null)

            dialogView.findViewById<TextView>(R.id.dialogTopicText).text =
            "Тема: $topic"

            dialogView.findViewById<TextView>(R.id.dialogDateText).text = "Дата:
            $date"

            val dialog = AlertDialog.Builder(activity)
                .setView(dialogView)
                .setCancelable(true)
                .create()

            dialogView.findViewById<Button>(R.id.dialogTopicButton).setOnClickListener {
                dialog.dismiss()
            }

            dialog.show()

```

```

        dialog.window?.setBackgroundDrawable(

android.graphics.drawable.ColorDrawable(android.graphics.Color.parseColor
Color("#F8F9FA"))
    )
}
}

//строка ученика
fun studentRow(student: Student, lessons: List<LessonInfo>) {
    val studentBlock = LinearLayout(context).apply {
        orientation = LinearLayout.VERTICAL
        layoutParams = LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT
        )
        setPadding(0, 3, 0, 3)
    }

    val nameRow = LinearLayout(context).apply {
        orientation = LinearLayout.HORIZONTAL
        layoutParams = LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT
        )
        setBackgroundColor(Color.parseColor("#F8F9FA"))
    }

    val nameView = TextView(context).apply {
        text = student.name
        gravity = Gravity.CENTER_VERTICAL
        setPadding(16, 20, 16, 20)
        textSize = 16f
        setTextColor(Color.parseColor("#1E1E2F"))
    }
}

```

					ПГУ1.090301.4.001 ПЗ	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		

```

        layoutParams = LinearLayout.LayoutParams(
            0,
            LinearLayout.LayoutParams.WRAP_CONTENT,
            1f
        )
    }

//кнопка в анкету ученика
    val profileButton = ImageButton(context).apply {
        setImageResource(R.drawable.student_data)
        setBackgroundColor(Color.TRANSPARENT)
        setPadding(0, 0, 0, 0)
        layoutParams = LinearLayout.LayoutParams(
            90,
            90
        ).apply {
            marginEnd = 50
            topMargin = 5
        }
        scaleType = ImageView.ScaleType.CENTER_INSIDE

        tag = student.id
        setOnClickListener {
            val intent = Intent(context,
EstimateFactors::class.java).apply {
                putExtra("studentId", student.id)
                putExtra("studentName", student.name)
            }
            context.startActivity(intent)
        }
    }
}

```

```

nameRow.addView(nameView)
nameRow.addView(profileButton)
studentBlock.addView(nameRow)

// скролл оценок горизонт
val scrollView = HorizontalScrollView(context).apply {
    layoutParams = LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.WRAP_CONTENT
    )
    isHorizontalScrollBarEnabled = false
}

val gradesContainer = LinearLayout(context).apply {
    orientation = LinearLayout.HORIZONTAL
    layoutParams = LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.WRAP_CONTENT,
        LinearLayout.LayoutParams.WRAP_CONTENT
    )
}

lessons.forEach { lesson ->
    gradesContainer.addView(createGradeCell(student.id,
lesson.id))
}

scrollView.addView(gradesContainer)
studentBlock.addView(scrollView)
studentsContainer.addView(studentBlock)

scrollViews.add(scrollView)

//синхронизация скролла
scrollView.setOnScrollChangeListener { _, scrollX, _, _, _ -

```

>

```
        scrollView.forEach { other ->
            if (other != scrollView) {
                other.scrollTo(scrollX, 0)
            }
        }
    }

}

//работа с отметками (редактир-е, созд-е, уд-е)
fun updateGrade(studentId: String, lessonId: String, score: Int)
{
    val key = "${studentId}_${lessonId}"
    val cell = gradeCells[key]
    if (cell != null) {
        cell.setText(score.toString())
        Log.d("Marks", "Обновлена ячейка для $key значением
$score")
    } else {
        Log.w("Marks", "Ячейка для $key не найдена")
    }
}

private fun createGradeCell(studentId: String, lessonId:
String): EditText {
    return EditText(context).apply {
        hint = "-"
        gravity = Gravity.CENTER
        inputType = android.text.InputType.TYPE_CLASS_NUMBER
        setBackgroundColor(Color.parseColor("#F2DBFF"))
        setPadding(16, 20, 16, 20)
    }
}
```

```

        layoutParams = LinearLayout.LayoutParams(180, 150)

        // обновление по Enter
        setOnKeyListener { _, keyCode, event ->
            if (keyCode == KeyEvent.KEYCODE_ENTER &&
event.action == KeyEvent.ACTION_UP) {
                val grade = text.toString().toIntOrNull()
                if (grade != null && grade in 1..5) {
                    saveGradeToFirebase(studentId, lessonId,
grade)

                        clearFocus()
                    }
                true
            } else {
                false
            }
        }

        setOnLongClickListener { deleteGrade(studentId,
lessonId)

            true
        }

        gradeCells["${studentId}_${lessonId}"] = this
    }
}

fun deleteGrade(studentId: String, lessonId: String) {
    val db = FirebaseFirestore.getInstance()
    db.collection("marks")
        .whereEqualTo("studentId", studentId)
        .whereEqualTo("lessonId", lessonId)

```

```

.get()
.addOnSuccessListener { documents ->
    for (doc in documents) {
        db.collection("marks").document(doc.id).delete()
        .addOnSuccessListener {
            val key = "${studentId}_${lessonId}"
            gradeCells[key]?.setText("")

            updatePredictions(studentId)

        }
    }
}
}

```

```

private fun savePredictionsToFirestore(studentId: String,
predictions: List<Float>, lessonIds: List<String>) {
    for (i in predictions.indices) {
        if (i < lessonIds.size) {
            val lessonId = lessonIds[i]
            val docId = "${studentId}_${lessonId}"

            val predData = hashMapOf(
                "studentId" to studentId,
                "lessonId" to lessonId,
                "predictedScore" to predictions[i],
                "timestamp" to FieldValue.serverTimestamp(),
                "type" to "grade_prediction"
            )

            db.collection("predictions")

```

```

        .document(docId)
        .set(predData)

    }

}

//

private fun saveGradeToFirebase(studentId: String, lessonId:
String, score: Int) {
    val db = FirebaseFirestore.getInstance()
    db.collection("marks")
        .whereEqualTo("studentId", studentId)
        .whereEqualTo("lessonId", lessonId)
        .get()
        .addOnSuccessListener { documents ->
            if (documents.isEmpty()) {
                val gradeData = hashMapOf(
                    "studentId" to studentId,
                    "lessonId" to lessonId,
                    "score" to score,
                    "timestamp" to FieldValue.serverTimestamp()
                )
                db.collection("marks").add(gradeData)
                    .addOnSuccessListener {
                        updatePredictions(studentId)
                    }
            } else {
                val docId = documents.first().id
                db.collection("marks").document(docId)
                    .update("score", score)
                    .addOnSuccessListener {
                        updatePredictions(studentId)
                    }
            }
        }
    }
}

```

```

    }
}

private fun updatePredictions (studentId: String) {
    getLastThreeGrades(studentId) { lastThree ->
        if (lastThree.size >= 3 &&
::gradePredictor.isInitialized) {
            val predictions =
gradePredictor.predictNextGrades(lastThree)

            //ассинхронное получение прошедших уроков
            getGradedLessonIdsSync(studentId) { gradedLessonIds
->

                val futureLessonIds = lessonsCache
                    .filter { it.id !in gradedLessonIds }
                    .map { it.id }
                    .take(4)

                savePredictionsToFirestore(studentId,
predictions, futureLessonIds)

                reloadPredictionsForStudent(studentId)
            }
        }
    }
}

private fun reloadPredictionsForStudent(studentId: String) {
    db.collection("predictions")
        .whereEqualTo("studentId", studentId)
        .get()
}

```

```

        .addOnSuccessListener { documents ->
            for (document in documents) {
                val lessonId = document.getString("lessonId") ?:
continue

                val predictedScore =
document.getDouble("predictedScore")?.toInt() ?: continue

                if (predictedScore == 1) {
                    gradeWarning(studentId, lessonId)
                } else {
                    gradeNormal(studentId, lessonId)
                }
            }
        }
    }
}

```

```

private fun getGradedLessonIdsSync(studentId: String, callback:
(List<String>) -> Unit) {
    db.collection("marks")
        .whereEqualTo("studentId", studentId)
        .get()
        .addOnSuccessListener { documents ->
            val gradedIds = documents.mapNotNull {
it.getString("lessonId") }
            callback(gradedIds)
        }
        .addOnFailureListener {
            callback(emptyList())
        }
}

```

```

private fun getLastThreeGrades(studentId: String, callback:
(List<Int>) -> Unit) {

```

```

try {
    db.collection("marks")
        .whereEqualTo("studentId", studentId)
        .get()
        .addOnSuccessListener { documents ->
            val grades = mutableListOf<Int>()
            for (doc in documents) {
                val score = doc.getLong("score")?.toInt()
                if (score != null) {
                    grades.add(score)
                }
            }

            val lastThree = if (grades.size > 3) {
                grades.subList(grades.size - 3, grades.size)
            } else {
                grades
            }
            callback(lastThree)
        }
        .addOnFailureListener {
            callback(emptyList())
        }
} catch (e: Exception) {
    callback(emptyList())
}
}

```

```
private var lessonsCache: List<LessonInfo> = emptyList()
```

```
fun setLessons(lessons: List<LessonInfo>) {
```

					ПГУ1.090301.4.001 ПЗ	Лист
						116
Изм.	Лист	№ докум.	Подпись	Дата		

```

        this.lessonsCache = lessons
    }

    private fun getLessonsList(): List<LessonInfo> = lessonsCache

    // подсветка ячейки
    fun gradeWarning(studentId: String, lessonId: String) {
        val key = "${studentId}_${lessonId}"

        gradeCells[key]?.setBackgroundColor(Color.parseColor("#FFEBEE"))
    }

    fun gradeNormal(studentId: String, lessonId: String) {
        val key = "${studentId}_${lessonId}"
        gradeCells[key]?.setBackgroundColor(Color.WHITE)
    }

    fun clear() {
        studentsContainer.removeAllViews()
        gradeCells.clear()
    }
}

data class Student(val id: String, val name: String, val classId:
String)
data class LessonInfo(val id: String, val date: String, val topic:
String)

```

Файл StudentProgress.py

```
import pandas as pd
```

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		117

```

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sts
import statistics as st
import seaborn as sns
import plotly.offline as pyo

!git clone https://github.com/KseniaRastor/OOP.git
df=pd.read_csv('/content/OOP/data.csv', delimiter=',')

df.head(10)

print(df.shape)
print(df.info())

df['Grades'].value_counts().sort_index()

# низкая успеваемость
data_low = df[df['Grades'] <= 34].copy()

# средняя успеваемость
data_mid = df[(df['Grades'] > 34) & (df['Grades'] < 46)].copy()

# высокая успеваемость
data_high = df[df['Grades'] >= 46].copy()

labels = ['низкая успеваемость', 'средняя успеваемость',
'высокая успеваемость']
sizes = [len(data_low), len(data_mid), len(data_high)]
colors = ['#E81EB5', 'blueviolet', '#FFF421']
explode = (0, 0, 0) # Выделение первой группы

```

```

# Построение диаграммы пирог
plt.figure(figsize=(8, 6))
plt.pie(sizes, explode = explode, labels=labels, colors=colors,
        autopct='%1.2f%%', shadow=True, startangle=140,
textprops={'fontsize': 15})

plt.axis('equal') # Равные оси для правильной формы круга
plt.show()

plt.rcParams.update({'font.size': 15})
df.hist(bins=20,figsize=(11,9),layout=(2,3));

df.describe()
data_low.describe()
data_mid.describe()
data_high.describe()

plt.boxplot(x=df['Socioeconomic Score'], vert=False,
showmeans=True, meanline=True, labels=('x'), patch_artist=True,
medianprops={'linewidth': 2, 'color': 'purple'},
meanprops={'linewidth': 2, 'color': 'red'})

plt.boxplot(x=df['Study Hours'], vert=False, showmeans=True,
meanline=True, labels=('x'), patch_artist=True,
medianprops={'linewidth': 2, 'color': 'purple'},
meanprops={'linewidth': 2, 'color': 'red'})

plt.boxplot(x=df['Sleep Hours'], vert=False, showmeans=True,
meanline=True, labels=('x'), patch_artist=True,
medianprops={'linewidth': 2, 'color': 'purple'},
meanprops={'linewidth': 2, 'color': 'red'})

plt.boxplot(x=df['Attendance (%)'], vert=False, showmeans=True,
meanline=True, labels=('x'), patch_artist=True,

```

```

medianprops={'linewidth': 2, 'color': 'purple'},
meanprops={'linewidth': 2, 'color': 'red'})

fig, axes = plt.subplots(1, 4, figsize=(16, 5), sharey=False)

features = ['Attendance (%)', 'Sleep Hours', 'Study Hours',
'Socioeconomic Score']

titles = ['Посещаемость (%)', 'Сон (часы)', 'Учёба (часы)',
'Соц-эконом. статус']

for i, (ax, feature, title) in enumerate(zip(axes, features,
titles)):

    ax.boxplot(df[feature], vert=True, showmeans=True,
meanline=True,

                medianprops={'linewidth': 2, 'color': 'purple'},
                meanprops={'linewidth': 2, 'color': 'red'})

    ax.set_title(title, fontsize=10)
    #ax.set_ylabel('Значение')
    ax.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.show()

df_filt=df[df['Study Hours']<=9.3]
plt.boxplot(x=df_filt['Study Hours'], vert=False,
showmeans=True, meanline=True, labels=('x'), patch_artist=True,
medianprops={'linewidth': 2, 'color': 'purple'},
meanprops={'linewidth': 2, 'color': 'red'})

df_filt=df[df['Attendance (%)']<=91]
plt.boxplot(x=df_filt['Attendance (%)'], vert=False,
showmeans=True, meanline=True, labels=('x'), patch_artist=True,
medianprops={'linewidth': 2, 'color': 'purple'},
meanprops={'linewidth': 2, 'color': 'red'})

```

```

print(df_filt.shape)
print(df_filt.info())

# низкая успеваемость
data_low_filt = df_filt[df_filt['Grades'] <= 34].copy()

# средняя успеваемость
data_mid_filt = df_filt[(df_filt['Grades'] > 34) &
(df['Grades'] < 46)].copy()

# высокая успеваемость
data_high_filt = df_filt[df_filt['Grades'] >= 46].copy()

labels = ['низкая успеваемость', 'средняя успеваемость',
'высокая успеваемость']
sizes = [len(data_low_filt), len(data_mid_filt),
len(data_high_filt)]
colors = ['#E81EB5', 'blueviolet', '#FFF421']
explode = (0, 0, 0) # Выделение первой группы

# Построение диаграммы пирог
plt.figure(figsize=(8, 6))
plt.pie(sizes, explode = explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140,
textprops={'fontsize': 15})

plt.axis('equal') # Равные оси для правильной формы круга
plt.show()

# отбрасываем неуникальные строки
df_deduped = df.drop_duplicates()

# сравниваем формы старого и нового наборов

```

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		121

```

print(df.shape)
print(df_dedupped.shape)

# Модель машинного обучения «К ближайших соседей»
from sklearn.neighbors import KNeighborsClassifier
# Для разбивки данных на тренировочный и тестовый наборы
from sklearn.model_selection import train_test_split
#для стандартизации
from sklearn.preprocessing import StandardScaler
#для метрик
from sklearn.metrics import precision_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,
confusion_matrix
    from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix
    from sklearn.metrics import classification_report,
confusion_matrix, roc_curve, auc
    from sklearn.metrics import confusion_matrix,
ConfusionMatrixDisplay
#для выбора параметров
from sklearn.model_selection import GridSearchCV

df = df_filt

#переход к понятиям Высокий, Средний и Низкий уровень
успеваемости
df.loc[df['Grades'] <= 34, 'Grades'] = -1
df.loc[(df['Grades'] > 34) & (df['Grades'] < 46), 'Grades'] = 0
df.loc[df['Grades'] >= 46, 'Grades'] = 1

X = df.iloc[:, 0:4].values
y = df.iloc[:, 4].values

```

```

#делим на тестовую и обучающую выборку
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#Поставить нужное значение k
knn = KNeighborsClassifier(n_neighbors=19)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

# Метрики точности
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Вывод результатов
print(f'Точность (Accuracy): {accuracy:.2f}')
print(f'Точность (Precision): {precision:.2f}')
print(f'Полнота (Recall): {recall:.2f}')
print(f'F1-мера: {f1:.2f}')

print(classification_report(y_test,y_pred))

conf_matrix = confusion_matrix(y_test,y_pred)
print ("Матрица ошибок \n",conf_matrix )

disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix)
disp.plot(cmap=plt.cm.Blues)
plt.title('Матрица ошибок', fontsize = 15)
plt.xlabel('Предсказанный класс', fontsize = 15)

```

```

plt.ylabel('Истинный класс', fontsize = 15)
plt.show()

from sklearn.model_selection import train_test_split,
GridSearchCV

knn = KNeighborsClassifier()

param_grid = {
    'n_neighbors': list(range(1, 50)), # Количество соседей
    #'weights': ['uniform', 'distance'], # Весовая функция
    #'metric': ['euclidean', 'manhattan'] # Метрика расстояния
}

# Настройка GridSearchCV
grid_search = GridSearchCV(knn, param_grid, cv=10,
scoring='accuracy', return_train_score=False, verbose=1)

# CV- разрезы кросс-валидации (меньше 3 делать не стоит),
scoring метрика, по которой будет оцениваться модель

grid_search.fit(X_train, y_train)

print("Лучшие параметры:", grid_search.best_params_)

# Оценка модели на тестовой выборке
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

print(classification_report(y_test, y_pred))

knn = KNeighborsClassifier()
from sklearn.model_selection import GridSearchCV
k_range = list(range(1, 50))

```

```

param_grid = dict(n_neighbors=k_range)

grid = GridSearchCV(knn, param_grid, cv=10, scoring='accuracy',
return_train_score=False,verbose=1)

grid_search=grid.fit(X_train, y_train)

```

Файл GradePrediction.py

```

from firebase_admin import credentials, firestore
from google.cloud.firestore import ArrayUnion, ArrayRemove

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier

# Оценка качества
from sklearn.metrics import accuracy_score,
precision_recall_fscore_support, classification_report

import numpy as np
import pandas as pd
from scipy.special import expit

# часы на изучение темы
hours_per_topic = [2, 2, 2, 1, 2, 2, 3, 4, 14, 3, 5, 6, 7, 3,
5, 4]

def synthetic_data_generation(n_students, n_topics):

    const = 1.2                #  $\beta_0$  – константа

```

					ПГУ1.090301.4.001 ПЗ	Лист
						125
Изм.	Лист	№ докум.	Подпись	Дата		

```

coef_mean = -1.15      # коэфф-т ср. балла (чем ниже
средний балл учащегося, тем выше риск неудачи)
coef_bad = 1.5         # коэфф-т числа неудовлетворительных
отметок (чем больше плохих отметок, тем выше риск неудачи)
coef_hours = 0.4       # коэфф-т длительности изучения темы
(чем больше часов на изучение темы, тем выше риск неудачи)

np.random.seed(42)
all_students = []

for student_id in range(n_students):
    grades = []

    # генерация первых 3 тем
    grade1 = np.random.choice([2,3,4,5], p=[0.05, 0.15,
0.45, 0.35])
    grade2 = np.random.choice([2,3,4,5], p=[0.05, 0.15,
0.45, 0.35])
    grade3 = np.random.choice([2,3,4,5], p=[0.05, 0.15,
0.45, 0.35])
    grades = [grade1, grade2, grade3]

    # генерация тем 4-16
    for topic_idx in range(3, n_topics):
        # статистика для 3 последних оценок
        last_three_items = grades[-3:]
        mean_grade = np.mean(last_three_items)
        bad_count = sum(1 for g in last_three_items if g <=
3)

        hours = hours_per_topic[topic_idx]

        # вероятность неудачи в теме
        logit = (const

```

```

        + coef_mean * mean_grade
        + coef_bad * bad_count
        + coef_hours * hours)

    probability = expit(logit) # сырой логит в
вероятность от 0 до 1

    # выставление оценки
    if np.random.random() < probability:
        new_grade = np.random.choice([2, 3], p=[0.3,
0.7])
    else:
        new_grade = np.random.choice([4, 5], p=[0.6,
0.4])

    grades.append(new_grade)

    all_students.append(grades)

    columns = [f'topic_{i+1}' for i in range(n_topics)]
    df = pd.DataFrame(all_students, columns=columns)
    return df

df = synthetic_data_generation(10000, 16)
print(df)

df.head(10)

def instruction_for_education(df, hours_per_topic,
window_size=3, prediction_range=4):

    X_list = []
    y_list = []

```

```

for student_grades_array in df.to_numpy():
    grades = student_grades_array.tolist()
    n_topics = len(grades)

    # определение окна
    for start_id_x in range(window_size, n_topics -
prediction_range + 1):
        # статистика - признаки из окна (ср. балл и кол-во
плохих отметок)

        window = grades[start_id_x -
window_size:start_id_x]
        mean_grade = np.mean(window)
        bad_count = sum(1 for g in window if g <= 3)

        # доп признаки
        last_grade = window[-1] #
последняя оценка
        trend = window[-1] - window[-2] #
изменилась ли оценка
        min_grade = np.min(window) #
минимальная оценка
        std_grade = np.std(window) #
стабильность
        has_two = 1 if 2 in window else 0 # была
ли двойка

        X_list.append([mean_grade, bad_count, last_grade,
trend, min_grade, std_grade, has_two])

        # 4 цели у на окно - будущие 4 темы
        y_window = []
        for offset in range(prediction_range):
            target_id_x = start_id_x + offset

```

```

        y_window.append(1 if grades[target_id_x] <= 3
else 0)

        y_list.append(y_window)

X = np.array(X_list)
y = np.array(y_list)
return X, y

X, y = instruction_for_education(df, hours_per_topic)

print(f"X: {X.shape}")
print(f"y: {y.shape}")

import pandas as pd
import numpy as np

def print_examples_table(X, y, n_examples=5, topic_offset=4):

    # Названия признаков (без часов)
    feature_names = [
        'ср.балл',          # mean_grade
        'кол-во неуд.',      # bad_count
        'последняя отметка', # last_grade
        'тренд',            # trend
        'мин.отметка',       # min_grade
        'стандарт.отклонение', # std_grade
        'наличие"2"'        # has_two
    ]

    rows = []
    for i in range(min(n_examples, len(X))):
        row = {'№': i + 1}

```

```

        for feat_name, feat_idx in zip(feature_names,
range(len(feature_names))):
            value = X[i][feat_idx]

            # Форматирование в зависимости от типа признака
            if feat_name == 'ср.балл' or feat_name ==
'стандарт.отклонение':
                row[feat_name] = f"{value:.2f}"
            elif feat_name == 'тренд':
                row[feat_name] = f"{value:+.1f}"
            elif feat_name == 'наличие"2"':
                row[feat_name] = '1' if value == 1 else
'0'

            else:
                row[feat_name] = value

        # Добавляем прогнозы для каждой позиции
        for pos, t in enumerate(range(topic_offset,
topic_offset + len(y[i]))):
            row[f'тема{t}'] = '1' if y[i][pos] == 1 else '0'

        rows.append(row)

df = pd.DataFrame(rows)
print(df.to_string(index=False))

print_examples_table(X, y, n_examples=10, topic_offset=4)

for offset in range(4):
    y_offset = y[:, offset]
    unique, counts = np.unique(y_offset, return_counts=True)
    print(f"\nПозиция {offset+1} (тема +{offset+1}):")
    for cls, count in zip(unique, counts):

```

```

        print(f"    Класс {cls}: {count}
({count/len(y_offset)*100:.1f}%)")

# обучение 4 моделей - отдельно для каждой темы
models = {}

for offset in range(4):
    print(f"\nОбучение модели для будущей отметки (текущая тема
+ {offset+1})")

    y_offset = y[:, offset]

    X_train, X_test, y_train, y_test = train_test_split(
        X, y_offset, test_size=0.2, random_state=42,
stratify=y_offset
    )

    model = LogisticRegression(class_weight='balanced',
random_state=42)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)

    models[offset] = model
    print(f"    Accuracy: {acc:.2%}")

```

Приложение В

(обязательное)

Система прогнозирования академических рисков учащихся

Презентация

					ПГУ1.090301.4.001 ПЗ	Лист
						132
Изм.	Лист	№ докум.	Подпись	Дата		

Система прогнозирования академических рисков учащихся

Подготовил: студент гр. 22ВВП1 Бондарь Ксения
Руководитель: к.т.н., доцент кафедры ВТ Синев М.П.

Пенза 2026

Актуальность темы

Трудоёмкость учёта успеваемости

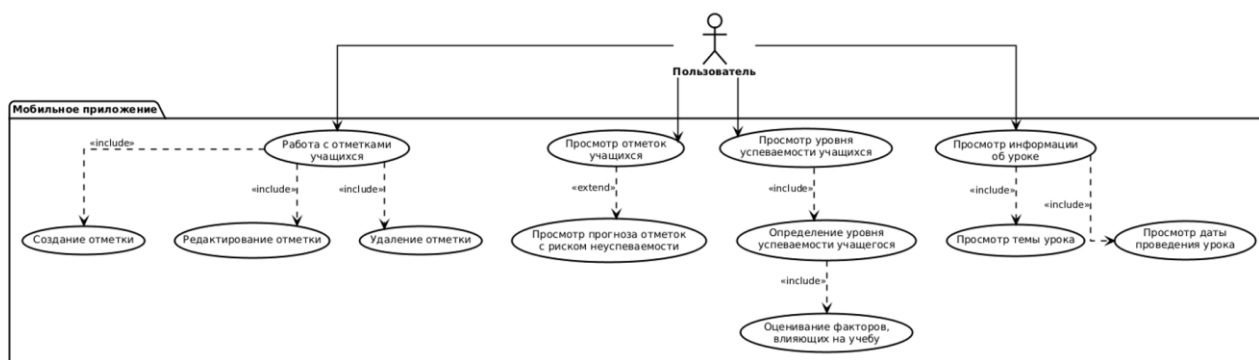
Работа с
журналом —
одна из самых
трудоёмких и
отнимающих
время задач в
работе учителя

Разрозненность данных

Информация об
успеваемости
хранится
фрагментарно и
не
интегрирована

Необходимость индивидуального подхода

Требуется
диагностика
академических
рисков на ранних
этапах обучения

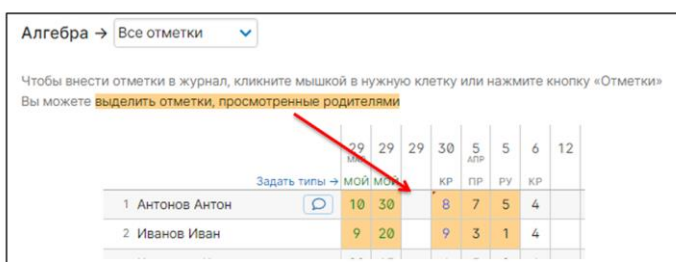


Постановка задачи

Требуется разработать мобильное приложение, которое будет включать две основные функции:

- определение успеваемости учащегося;
- выявление тем, по которым прогнозируются неудовлетворительные отметки.

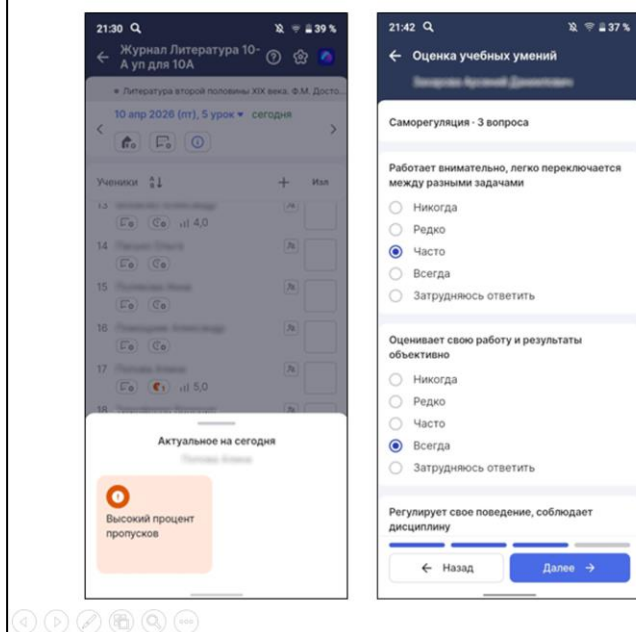
Аналоги



Электронный журнал ЭлЖур

- функция информирования о просмотренных отметках учащегося родителями
- наличие различной графической аналитики
- ручной ввод прогнозируемой отметки не является оптимальным вариантом отслеживания успеваемости

Аналоги



Московская электронная школа

- специальный функционал отслеживания успеваемости учащихся;
- сводная аналитика успеваемости для всего класса учеников
- раздел «Учебные умения» для оценки навыков учащегося, требуемые для успешного обучения

Стек технологий



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.4.001 ПЗ

Лист

135

Прогноз неудовлетворительных отметок

3 последние
отметки
учащегося



Обучение модели для будущей отметки (текущая тема + 1)
Accuracy: 80.52%

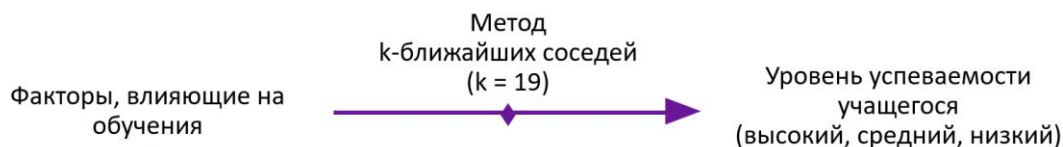
Обучение модели для будущей отметки (текущая тема + 2)
Accuracy: 79.81%

Обучение модели для будущей отметки (текущая тема + 3)
Accuracy: 76.65%

Обучение модели для будущей отметки (текущая тема + 4)
Accuracy: 69.88%

1. средний балл;
2. количество неудовлетворительных отметок;
3. последняя отметка;
4. наличие изменения отметки (высчитывается между последней и предпоследней отметками);
5. минимальная отметка;
6. стандартное отклонение (характеризует стабильность ученика);
7. наличие отметки «2».

Определение уровня успеваемости



Количество часов сна

Количество часов обучения

Регулярность посещения занятий

Социально-экономическое окружение

Точность(Accuracy): 0.75
Точность (Precision): 0.75
Полнота (Recall): 0.75
F1-мера: 0.73

	precision	recall	f1-score	support
-1.0	0.70	0.92	0.80	109
0.0	0.67	0.48	0.56	92
1.0	0.91	0.82	0.87	74
accuracy			0.75	275
macro avg	0.76	0.74	0.74	275
weighted avg	0.75	0.75	0.73	275

Диаграмма классов

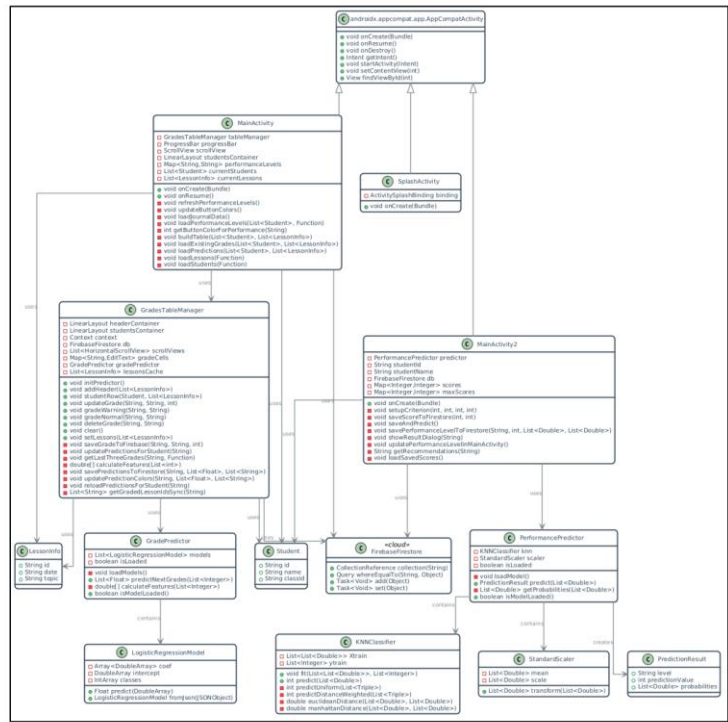


Диаграмма последовательности

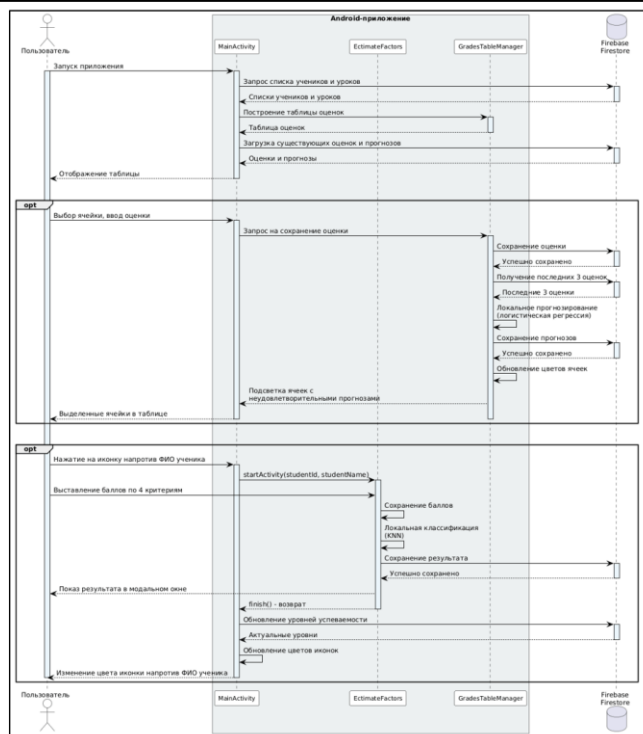
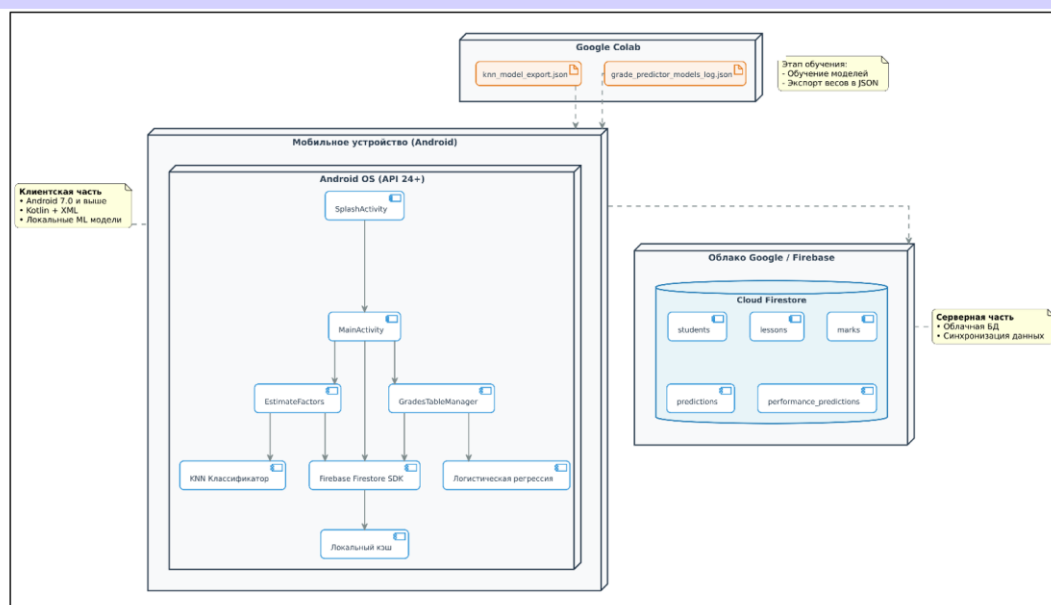


Диаграмма развертывания



Карта экранов

Основные экраны приложения

- Экран-заставка
- Журнальная сетка класса
- Анкета для оценки факторов обучения



Заключение

В ходе выполнения выпускной квалификационной работы разработано мобильное приложение для отслеживания академических рисков учащихся. Приложение содержит две основные функции:

- прогноз неудовлетворительных отметок учащихся,
- определение уровня успеваемости учащегося.

Для реализации функционала приложения были проведены работы по выбору датасета для одной модели обучения – классификатор уровня успеваемости – и созданию синтетических данных для другой – прогноз оценок. Для полученных данных были проведены анализ и обработка для дальнейшего использования. После обучения моделей были достигнуты удовлетворительные результаты, достигающие 80% точности.

					ПГУ1.090301.4.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		139

