

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: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ «ОНЛАЙН –
АВТОШКОЛА» НА БАЗЕ ОС ANDROID

Студент

Алтыбаев М.К.
(подпись, дата)

Руководитель

Гурин Е.И.
(подпись, дата)

Нормоконтролёр

А.С. Бычков
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025г. № 16)

Заведующий кафедрой

М.А. Митрохина
(подпись)

Работа защищена с отметкой удовлет (протокол заседания ГЭК от

20 июня 2025г. № 16)

Секретарь ГЭК

Е.И. Гурин
(подпись)

Пенза 2025

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
06.05.2025г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Алтыбаев Мыстыбек Камчыбекович гр.21BBB1 факультета вычислительной техники

2. Тема работы: «Мобильное приложение «Онлайн-автошкола» на базе ОС Android.

Тема утверждена приказом ПГУ №601 от "06" мая 2025г.

3. Руководитель бакалаврской работы д.т.н., профессор, ВТ ПГУ Гурин Е.И.

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

Разработать приложение «Онлайн – автошкола» на базе ОС Android для сочетающего теоретические материалы, практические задания. Анализировать рынок существующих решений и определение требований к функционалу. Проектирование архитектуры приложения с использованием современных технологий. Разработка интуитивно-понятного интерфейса и системы взаимодействия с пользователем. Реализация модулей обучения, тестирования и отслеживания прогресса. Тестирование на различных устройствах и платформах для обеспечения стабильной работы. Использовать RSS-агент и API новостных порталов.

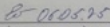
5. Перечень подлежащих разработке вопросов

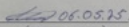
Использование современных принципов UX/UI дизайна, проведение тестирования с участием целевой аудитории для сбора обратной связи и внесение необходимых коррективов. Обеспечить регулярное обновление учебных материалов в соответствии с изменениями в законодательстве. Также необходимо изучить функционалы языка программирования Flutter и Dart. Освоить база данных на Firebase.

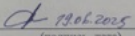
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работ	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	06.05.2025 08.05.2025	
2.	Анализ требований	10	09.05.2025 10.05.2025	
3.	Выбор технологий, применяемых при проектировании и разработке	50	11.05.2025 13.05.2025	
4.	Описание модулей классов	40	13.05.2025 15.05.2025	
5.	Разработка приложения «Онлайн-автошкола» на базе ОС Android	150	16.05.2025 27.05.2025	
6.	Оформление пояснительной записки	24	27.05.2025 30.05.2025	

Дата выдачи задания "06" мая 2025г.

Руководитель бакалаврской работы  06.05.25 Гурин Е.И.
(подпись, дата)

Задание к исполнению принял студент  06.05.25 Алтыбаев М.К.
(подпись, дата)

Работу к защите допустить
Декан факультета  19.06.2025 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 128 листов, 25 рисунков, 3 приложение.
ОС ANDROID, ИНТЕРФЕЙС, VISUAL STUDIO CODE

Цель проекта: создание кроссплатформенного мобильного приложения для обучения вождению, сочетающего теоретические материалы, практические задания и систему мотивации.

Задачи проекта: Анализ рынка существующих решений и определение требований к функционалу. Проектирование архитектуры приложения с использованием современных технологий Разработка интуитивно понятного интерфейса и системы взаимодействия с пользователем. Реализация модулей обучения, тестирования и отслеживания прогресса. Тестирование на различных устройствах и платформах для обеспечения стабильной работы

Разработано приложение с модульной структурой, включающее:
Интерактивные уроки по ПДД. Тесты с автоматической проверкой и обратной связью. Систему рейтинга и достижений для мотивации пользователей. – Административную панель для управления контентом. Обеспечена поддержка различных устройств благодаря тестированию на эмуляторах и реальных смартфонах. Реализована интеграция с облачными сервисами Firebase для хранения данных и медиа файлов.

Практическая значимость приложение позволяет:

- упростить процесс подготовки к экзаменам в ГИБДД за счет доступности материалов в любое время;
- повысить качество обучения благодаря интерактивным элементам и адаптивным тестам;
- понизить нагрузку на преподавателей за счет автоматизации проверки.

					ПГУ 1.090301.1.001 ПЗ			
Изм	Лист	№ докум	Подп	Дата	Мобильное приложение "Онлайн-вождение" на базе ОС Android Пояснительная записка	Литера	Лист	Листов
Разработ		Алибалиев И.К.					2	
Провер		Григорьев Е.И.						
И. экзам		Бухарин А.С.						
Изд		Митрофанов М.А.						
						ФВТ, каф. ВТ гр. 218ВВВ1		

Содержание

Введение.....	5
1 Анализ требований и актуальности задачи	6
1.1 Цель работы	6
1.2 Значимость разработки.....	7
1.3 Постановка задачи.....	8
1.4 Выбор технологий и средства разработки.....	10
1.4.1 Visual Studio Code (VS Code)	10
1.4.2 Dart и Flutter	14
1.5 Выбор СУБД.....	18
1.6 Эмуляторы и устройства для тестирования	24
1.7 Проектирование и разработка приложения.....	26
1.7.1 Архитектура приложения.....	26
1.7.1 Разработка модулей и классов	31
1.7.3 Разработка СУБД	35
1.7.4 Разработка интерфейса	37
1.8 Административная панель	44
1.9 Тестирование приложения	44
2 Охрана труда.....	47
Заключение	49
Список использованных источников	50
Приложение А – UML–диаграммы	51
Приложение Б – Листинг программы	57
Приложение В – Презентация.....	113

Перечень сокращений и обозначений

ГИБДД – государственная инспекция безопасности дорожного движения

СУБД – система управления базами данных

ПДД – правила дорожного движения

API (Application Programming Interface) – программный интерфейс приложения

FAQ (Frequently Asked Questions) – часто задаваемые вопросы

HTML (Hyper Text Markup Language) – язык гипертекстовой разметки

IDE (Integrated Development Environment) – интегрированная среда разработки

UML (Unified Modeling Language) – унифицирован язык моделирования.

VS Code (Visual Studio Code) – код визуальной студий, редактор кода

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

Введение

В современном обществе мобильные приложения все активнее интегрируются в нашу повседневную жизнь, предоставляя пользователям широкий выбор возможностей и сервисов. Одной из нужных сфер разработки мобильных программ является образование и самостоятельное обучение. Особенно популярной становится область обучения вождению и получения водительских удостоверений, что особенно актуально после карантина, и для повышения своей мобильности.

В рамках данной выпускной работы рассматривается создание мобильного приложения “Онлайн автошкола” для платформы Android. Данное приложение предназначено для тех, кто хочет удобно и эффективно изучать теоретическую часть вождения, подготовиться к экзаменам на права и повысить свои водительские навыки.

Основной целью проекта является разработка современного мобильного продукта, способного значительно упростить процесс обучения вождению и сделать его более доступным для широкой аудитории. “Онлайн автошкола” предлагает не только чтение стандартных теоретических материалов, но интерактивные задания и различные тесты, что способствует глубокому пониманию правил дорожного движения и их практическому применению. Такие упражнения разнообразны и адаптированы под разные стили обучения, что делает обучение не только полезным, но и увлекательным. В результате пользователи смогут более уверенно оценить свою подготовленности к экзаменам и успешно их сдать.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

1 Анализ требований и актуальности задачи

1.1 Цель работы

Основной целью этого проекта является разработка мобильного приложения, сочетающего в себе достоинства дистанционного обучения, интерактивные занятия, проверочные тесты и практические упражнения, а также мгновенную обратную связь от программы. Приложение рассчитано на начинающих водителей и тех, кто желает повторить или углубить свои знания ПДД, и направлено на решение следующих вопросов:

- обеспечение удобства;
- соединение теории и практики;
- улучшение качества учебного процесса;
- увеличение заинтересованности;
- поддержание постоянного обучения.

Разработка образовательного мобильного приложения для подготовки к экзамену по вождению направлена на решение нескольких ключевых задач. В первую очередь, мы стремимся создать удобный и понятный интерфейс, который будет доступен пользователям с разным уровнем технической подготовки. Это позволит сделать процесс обучения комфортным и эффективным для широкого круга будущих водителей.

Особое внимание уделяется комплексному подходу к обучению. Наше приложение органично сочетает теоретическую подготовку по правилам дорожного движения с практическими заданиями, формируя единую образовательную среду. Такой подход помогает учащимся не только запоминать правила, но и сразу применять их в смоделированных ситуациях.

Важным аспектом является обеспечение актуальности учебных материалов. Мы регулярно обновляем контент, чтобы предоставлять пользователям только достоверную информацию, соответствующую последним изменениям в ПДД. Все тестовые задания проходят тщательную проверку на соответствие современным требованиям.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

Для повышения вовлеченности пользователей в приложении реализована система мотивации. Она включает различные механизмы поощрения, визуализацию прогресса и достижений, что помогает поддерживать интерес к обучению на протяжении всего курса подготовки.

Особая методика повторения материала, реализованная через систему "попыток", способствует лучшему усвоению информации. Этот механизм побуждает учащихся регулярно возвращаться к пройденным темам, что значительно повышает эффективность подготовки и качество запоминания правил дорожного движения.

1.2 Значимость разработки

В последние годы заметно вырос рынок мобильных приложений, предназначенных для обучения вождению. Это связано с увеличением числа владельцев смартфонов и планшетов, а также с растущим спросом на удобные и доступные образовательные сервисы.

По данным отчета ASOMobile за 2024 год, рынок мобильных решений продолжает активно развиваться, несмотря на общемировые экономические трудности, растет спрос на приложения, предлагающие комплексный подход к обучению, объединяющий теорию и практику [1].

Пользовательские ожидания: водители, использующие мобильные приложения для подготовки, ожидают удобных, интерактивных и полных решений. Они ценят подробные объяснения, реалистичные симуляции и практические рекомендации. Важным фактором являются актуальные тесты и материалы, соответствующие последним нормативным актам.

Соревновательная среда: рынок приложений для автошкол отличается высокой конкуренцией. Чтобы занять свою нишу, новые приложения должны предлагать уникальные возможности или превосходный пользовательский опыт. Для успешного развития мобильного приложения онлайн автошколы необходимо учитывать современные тенденции и

запросы пользователей. Ключевым фактором является предоставление качественного контента, сочетающего теоретические знания и практические упражнения, а также своевременное обновление учебных материалов в соответствии с изменениями в ПДД.

1.3 Постановка задачи

Целью данного проекта является создание мобильного приложения, которое объединит в себе удобство онлайн-обучения с интерактивными уроками, тестами и практическими заданиями, а также возможностью получения реального обратного отклика от системы. Приложение будет ориентировано как на начинающих водителей, так и на тех, кто желает обновить свои знания и подготовиться к сдаче экзаменов по ПДД. Для реализации образовательной платформы предусматривается решение следующих ключевых задач:

- проектирование пользовательского интерфейса;
- разработка образовательного контента;
- организация информационного раздела;
- внедрение мотивационной системы;
- создание сервисного модуля;
- поддержание актуальности контента;
- организация пользовательской поддержки.

Разрабатываемая образовательная платформа для подготовки водителей представляет собой комплексное решение, объединяющее несколько взаимосвязанных компонентов.

Основой приложения станет интуитивно понятный интерфейс, обеспечивающий доступ к полному спектру учебных материалов по правилам дорожного движения, включая дорожные знаки, разметку и правовые аспекты.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

Содержание платформы формируется при участии экспертов в области дорожной безопасности и включает как теоретические разделы, так и практические задания, максимально приближенные к экзаменационным билетам.

Важной составляющей станет автоматизированная система оповещения о законодательных изменениях, работающая через интеграцию с новостными источниками.

Для повышения вовлеченности пользователей предусмотрена многоуровневая система мотивации с визуализацией прогресса и элементами геймификации. Дополнительные сервисные возможности включают внутреннюю экономическую модель с виртуальной валютой.

Особое внимание уделяется постоянному обновлению контента в соответствии с изменениями законодательства через налаженные каналы взаимодействия с регулирующими органами.

Завершает функционал платформы комплексная система пользовательской поддержки с различными форматами обратной связи.

Совокупность этих элементов создает современную образовательную среду, обеспечивающую эффективную подготовку будущих водителей как в теоретическом, так и в практическом аспектах.

Ожидаемые характеристики платформы:

- оптимизированная система навигации и управления;
- полноценный учебно-методический комплекс;
- механизмы оперативного информирования;
- эффективные инструменты мотивации;
- сбалансированная сервисная модель;
- гарантия соответствия нормативным требованиям;
- комплексная пользовательская поддержка.

Реализация проекта позволит создать современную образовательную среду для подготовки водителей, сочетающую теоретическую базу и практическую направленность.

1.4 Выбор технологий и средства разработки

1.4.1 Visual Studio Code (VS Code)

Visual Studio Code (VS Code) — это бесплатный, кроссплатформенный редактор исходного кода, разработанный компанией Microsoft. С момента своего выпуска в 2015 году, VS Code стал одним из самых популярных инструментов для разработчиков по всему миру, благодаря своей гибкости, производительности и богатому набору функций. В данной статье мы подробно рассмотрим историю создания VS Code, ключевые этапы его развития, влияние на сообщество разработчиков и перспективы будущего развития.

История VS Code началась в начале 2010-х годов, когда Microsoft осознала необходимость создания легковесного, но мощного редактора кода, который бы работал на всех основных платформах (Windows, macOS и Linux). Компания стремилась предоставить разработчикам современный инструмент, который сочетал бы в себе возможности полнофункциональной среды разработки (IDE) и быстроту и простоту текстового редактора.

Мотивация. Ключевой мотивацией для создания VS Code было желание Microsoft предложить инструмент, который удовлетворял бы потребности разработчиков в современном редакторе кода, отвечающем следующим критериям.:

- поддержка всех основных операционных систем;
- быстрая работа и малое потребление ресурсов;
- возможность добавления новых функций и поддержка большого числа языков программирования;
- поддержка контейнеров, облачных сервисов и систем контроля версий.

Создание и первоначальный выпуск.

Работа над Visual Studio Code началась в 2013 году в компании Microsoft, а первая публичная версия была анонсирована на мероприятии Microsoft Build в апреле 2015 года. Изначально редактор был создан на базе Electron – фреймворка, предназначенного для разработки настольных приложений с использованием веб-технологий, таких как JavaScript, HTML и CSS. Такой подход позволил разработчикам создавать кроссплатформенные решения с единым исходным кодом.

Основные характеристики первой версии включали.

Редактор кода: наличие подсветки синтаксиса для различных языков программирования, а также возможность установки расширений для добавления новых функций.

Интегрированная система контроля версий (Git): встроенная поддержка Git-операций, что позволяло пользователям управлять репозиториями прямо из редактора.

Инструмент отладки: возможность отладки программного кода непосредственно в редакторе для популярных языков программирования.

Поддержка IntelliSense: интеллектуальные подсказки и автодополнение кода для повышения продуктивности.

Эволюция и расширение возможностей. С момента появления первая версия VS Code неоднократно обновлялась и получала новые функции. Microsoft активно занимается развитием редактора, постоянно улучшая его и дополняя существующий функционал.

Ключевые этапы развития. Расширения и Marketplace: в 2016 году был запущен Visual Studio Code Marketplace – платформа, предназначенная для установки расширений. Это дало возможность разработчикам добавлять новые языки программирования, инструменты отладки, темы оформления и другие компоненты, значительно расширяя возможности редактора.

Активное развитие и поддержка сообществом. Магазин расширений VS Code постоянно пополняется новыми плагинами и дополнениями, созданными сообществом разработчиков для удовлетворения самых разнообразных потребностей пользователей.

Интеграция с облачной платформой Azure. Microsoft внедрила поддержку Azure в VS Code, что дает возможность легко создавать, запускать и управлять облачными приложениями прямо из редактора, значительно упрощая работу с облачными сервисами.

Работа с контейнерами и оркестрацией. Были добавлены инструменты для работы с Docker и Kubernetes, что позволяет разработчикам создавать, тестировать и управлять контейнеризированными проектами и инфраструктурой прямо внутри VS Code.

Мощные средства для отладки и тестирования. В редакторе реализованы расширенные средства для отладки и тестирования, поддерживающие широкий спектр языков программирования и фреймворков, что повышает эффективность разработки.

Функция совместной работы Live Share. В 2017 году Microsoft представила Live Share – инструмент, который позволяет нескольким разработчикам одновременно работать над одним проектом в реальном времени, значительно упрощая командное взаимодействие.

Глубокая интеграция с GitHub. После приобретения платформы GitHub компанией Microsoft, VS Code получил расширенные возможности для работы с репозиториями и задачами на GitHub, а также интеграцию с GitHub Actions для автоматизации процессов разработки.

Поддержка Windows Subsystem for Linux (WSL). Добавлена возможность использования WSL, что позволяет разработчикам запускать Linux-инструменты и окружения прямо на Windows, что особенно удобно для кроссплатформенной работы.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

Внедрение удаленной разработки. Функционал Remote Development был добавлен для поддержки работы с кодом на удаленных серверах, в контейнерах и виртуальных машинах. Это позволяет разработчикам выполнять редактирование, запуск и отладку приложений без необходимости переноса данных на локальный компьютер.

Влияние на сообщество и индустрию. VS Code стал невероятно популярным среди программистов благодаря ряду своих преимуществ:

- кроссплатформенность;
- легкость и высокая производительность;
- гибкая настройка;
- активное сообщество.

Редактор работает на операционных системах Windows, macOS и Linux, что обеспечивает единое рабочее пространство для разработчиков независимо от используемой платформы.

Он быстро работает даже на менее мощных машинах, что делает его универсальным инструментом для самых разных проектов.

Огромное количество расширений и возможностей для глубокой кастомизации позволяют адаптировать редактор под любые требования разработчика.

Пользователи и разработчики активно делятся опытом, создают новые плагины и расширения, а также участвуют в обсуждениях на форумах, блогах и в соцсетях.

Признание и награды. VS Code неоднократно получал высокие оценки и престижные награды за свою функциональность и интерфейс. Red Dot Award 2016: награда за выдающийся дизайн пользовательского интерфейса, подтверждающая высокое качество и удобство использования редактора. Stack Overflow Developer Survey: многочисленные опросы, включая ежегодный опрос Stack Overflow Developer Survey, подтверждают, что Visual Studio Code является самым популярным редактором кода среди

разработчиков по всему миру. Его предпочтение объясняется богатым функционалом, гибкостью и активным сообществом. [2]

Будущее Visual Studio Code. Microsoft продолжает активно развивать VS Code, внедряя новые функции и совершенствуя существующие возможности. В ближайшие годы ожидается усиление интеграции с облачными платформами, улучшение инструментов работы с контейнерами, расширение функций для командной совместной работы и поддержка новых языков программирования и технологий.

В целом, Visual Studio Code – это мощный и универсальный редактор, подходящий как новичкам, так и профессионалам. Его гибкость, расширяемость и богатый функционал делают его идеальной средой для написания, отладки и управления проектами. Поддержка множества языков программирования и возможность расширения через плагины делают VS Code незаменимым инструментом, особенно в разработке мобильных приложений на Dart и Flutter.

1.4.2 Dart и Flutter

Dart – это язык программирования, разработанный компанией Google. Его история началась в 2011 году, когда Google представила первый релиз языка. Dart изначально создавался как замена JavaScript для написания клиентских веб-приложений. [3]

Цели и Мотивация. Основной целью создания Dart было устранение недостатков, присущих JavaScript, таких как медленная производительность, ограниченные возможности для масштабирования и сложность поддержания больших кодовых баз.

Высокая производительность. Dart компилируется в нативный код для мобильных и настольных приложений, что обеспечивает высокую производительность.

Простота и читаемость. Синтаксис Dart напоминает C-подобные языки (например, Java и JavaScript), что облегчает его изучение для разработчиков, знакомых с этими языками.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		14

Мощные инструменты. Dart предоставляет развитые инструменты для разработки, включая IDE, дебаггер и средства профилирования.

Flutter – это фреймворк для разработки кроссплатформенных приложений, также созданный компанией Google. Его история началась в 2015 году, когда Google анонсировала первый прототип Flutter. [4]

Цели и Мотивация. Flutter был разработан с целью облегчить создание высококачественных кроссплатформенных приложений с нативным интерфейсом.

Высокая производительность. Использование нативных графических API (OpenGL и Metal) для рендеринга интерфейсов обеспечивает плавную работу приложений.

Единая кодовая база. Возможность написания единого кода для Android и iOS, что сокращает время и затраты на разработку.

Красивый UI. Поддержка богатого набора виджетов и гибких анимаций для создания привлекательных пользовательских интерфейсов.

Flutter изначально был разработан для работы с языком программирования Dart, что обеспечило тесную интеграцию между языком и фреймворком. Dart предоставляет высокую производительность, эффективную работу с асинхронными операциями и развитые инструменты для разработки, что делает его идеальным выбором для создания кроссплатформенных приложений с использованием Flutter.

Использование языка программирования Dart вместе с фреймворком Flutter предлагает разработчикам ряд существенных преимуществ, которые делают эту связку особенно привлекательной для создания кроссплатформенных приложений. Одним из главных достоинств является возможность компиляции в нативный машинный код, что обеспечивает высокий уровень производительности конечного продукта, сравнимый с решениями, написанными на platform-specific языках. Не менее важной особенностью является функция Hot Reload, позволяющая мгновенно

применять изменения в коде без необходимости полной перезагрузки приложения, что значительно ускоряет итерационный процесс разработки и упрощает отладку. Универсальность технологии проявляется в её способности создавать приложения для всех популярных платформ, включая мобильные операционные системы Android и iOS, веб-приложения и десктопные решения для различных операционных систем, что обеспечивает истинную кроссплатформенность.

Google как создатель и основной поддерживающий Dart и Flutter обеспечивает постоянное развитие этих технологий, что гарантирует их актуальность и соответствие современным требованиям. Активное сообщество разработчиков, объединенное вокруг этих технологий, способствует быстрому распространению знаний и лучших практик через различные каналы коммуникации. Регулярные специализированные мероприятия, такие как Flutter Engage и Dart Summit, служат площадками для обмена опытом и демонстрации новых возможностей. Обилие обучающих ресурсов, включая онлайн-курсы, документацию, форумы и тематические сообщества, делает освоение этих технологий доступным для разработчиков разного уровня подготовки. Такая комплексная поддержка со стороны как компании-разработчика, так и сообщества значительно снижает барьеры входа и способствует популяризации Dart и Flutter в профессиональной среде.

Будущее Dart и Flutter. На сегодняшний день Dart и Flutter продолжают активно развиваться. Google инвестирует значительные ресурсы в улучшение и расширение возможностей обоих проектов. В будущем можно ожидать дальнейших улучшений производительности, новых функций и расширения возможностей для разработки на различных платформах.

Основные особенности Darta.

Строгая типизация. Dart поддерживает строгую типизацию, что позволяет выявлять ошибки на этапе компиляции и улучшает надежность кода.

Асинхронное программирование. Dart имеет встроенную поддержку асинхронного программирования, что делает его отличным выбором для работы с сетевыми запросами и асинхронными операциями.

Удобный синтаксис. Синтаксис Dart прост в изучении и понимании, что делает его доступным для новичков и приятным для опытных разработчиков.

Компиляция в JavaScript. Dart может компилироваться в JavaScript, что позволяет использовать его для разработки веб-приложений.

Основные особенности Флаттера.

Гибкий интерфейс. Флаттер предлагает гибкий и мощный интерфейс для создания пользовательских интерфейсов, который позволяет разработчикам легко создавать красивые и интуитивно понятные приложения.

Горячая перезагрузка. Функция «горячей перезагрузки» в Флаттере позволяет разработчикам мгновенно видеть изменения, вносимые в код, что значительно ускоряет процесс разработки и отладки.

Высокая производительность. Благодаря использованию собственного движка рендеринга, Флаттер обеспечивает высокую производительность и плавность анимаций в приложениях.

Единый код для iOS и Android. Флаттер использует один и тот же код для создания приложений под iOS и Android, что упрощает разработку и поддержку приложений на разных платформах.

Таким образом, Dart и Флаттер – это мощные инструменты для разработки кроссплатформенных мобильных приложений, которые предлагают разработчикам удобный и эффективный способ создания высококачественных приложений для различных платформ. Фреймворк:

Flutter Flutter – это кроссплатформенный фреймворк для разработки мобильных приложений, созданный компанией Google. Он позволяет разработчикам создавать красивые и высокопроизводительные приложения, которые работают как на Android, так и на iOS, используя общий код на Dart. Flutter предоставляет богатый набор виджетов и инструментов для разработки интерфейса и обеспечивает быструю разработку и гибкость в создании пользовательского опыта.

1.5 Выбор СУБД

Firebase – это платформа для разработки мобильных и веб-приложений, предоставляющая инструменты и инфраструктуру для создания, улучшения и роста приложений. История Firebase начинается в 2011 году, когда она была основана Джеймсом Темплином и Эндрю Ли как стартап, предоставляющий облачную инфраструктуру для приложений. [5]

Первоначально Firebase была известна как Envolvе и представляла собой службу чата, которая позволяла добавлять функции реального времени в веб-приложения. Envolvе предлагала разработчикам простые API для встраивания чатов на свои сайты, обеспечивая мгновенную коммуникацию между пользователями. Вскоре стало очевидно, что основной интерес разработчиков вызывала не столько сама функция чата, сколько технология синхронизации данных в реальном времени, лежащая в его основе.

В 2011 году Темплин и Ли решили переориентировать свой старт-ап, сосредоточившись на предоставлении инструментария для синхронизации данных в реальном времени. Это привело к появлению Firebase как платформы для разработки приложений. Firebase предлагала облачную базу данных в реальном времени, позволяющую синхронизировать данные между клиентами через интернет мгновенно, что было особенно полезно для создания чатов, игр и других интерактивных приложений.

В 2014 году Google приобрела Firebase, что позволило платформе значительно расширить свои возможности и интегрироваться с другими продуктами Google. Под руководством Google Firebase стала частью Google Cloud Platform, что обеспечило ей стабильную инфраструктуру и возможность масштабирования. Это слияние дало Firebase доступ к ресурсам и технологиям Google, что позволило ускорить развитие и внедрение новых функций.

После присоединения к Google Firebase начала быстро развиваться, добавляя новые функции и сервисы. В 2016 году на конференции Google I/O был представлен новый обновленный Firebase, включающий множество новых продуктов и сервисов.

Firebase Authentication. Сервис для управления аутентификацией пользователей, поддерживающий вход через электронную почту, телефон и социальные сети. Это упрощает процесс регистрации и входа для пользователей, обеспечивая при этом безопасность их данных.

Firebase Analytics (ныне Google Analytics для Firebase). Мощный инструмент для аналитики, позволяющий отслеживать взаимодействие пользователей с приложением. Это помогает разработчикам понять, как пользователи взаимодействуют с приложением, и принимать обоснованные решения для его улучшения.

Firebase Cloud Messaging (FCM). Сервис для отправки push-уведомлений на устройства пользователей. FCM поддерживает отправку уведомлений на Android и iOS устройства, а также на веб-приложения.

Firebase Crashlytics. Инструмент для мониторинга и отчета об авариях приложений. Crashlytics помогает разработчикам быстро обнаруживать и исправлять ошибки в приложении, улучшая его стабильность.

Firebase Hosting. Услуга для хостинга веб-приложений. Firebase Hosting предоставляет быструю и безопасную инфраструктуру для развертывания веб-приложений.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

Firebase Cloud Storage. Хранилище для файлов, интегрированное с Google Cloud Storage. Firebase Cloud Storage предоставляет безопасное и масштабируемое хранилище для пользовательских данных, таких как фотографии, видео и документы.

Одним из ключевых преимуществ Firebase стало её тесное взаимодействие с другими сервисами Google, такими как Google Cloud Platform, Google Ads и Google Play. Это позволило разработчикам более эффективно использовать экосистему Google для продвижения, анализа и масштабирования своих приложений.

С каждым годом Firebase продолжает расширять свои возможности и улучшать существующие сервисы. В 2017 году была добавлена поддержка Firestore — новой облачной базы данных, обеспечивающей более гибкую и мощную структуру данных по сравнению с Firebase Realtime Database. Firestore отличается улучшенной производительностью и поддержкой сложных запросов, а также синхронизацией данных в реальном времени и оффлайн-режимом.

Firebase также интегрировала функции машинного обучения, предоставляя разработчикам доступ к моделям TensorFlow через Firebase ML Kit. Это позволяет добавлять интеллектуальные функции в приложения, такие как распознавание текста, лица и объектов.

В последние годы Firebase расширила свой инструментарий для DevOps и CI/CD (непрерывной интеграции и непрерывного развертывания), добавив такие инструменты, как Firebase Performance Monitoring и Firebase Test Lab. Эти инструменты помогают разработчикам тестировать и оптимизировать свои приложения для достижения максимальной производительности и стабильности.

Firebase активно поддерживает сообщество разработчиков, предоставляя обширную документацию, учебные материалы и примеры кода. Регулярные мероприятия, такие как Firebase Summit и участие в Google

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

I/O, позволяют разработчикам узнавать о новейших функциях и лучших практиках использования платформы. Сообщество Firebase включает миллионы разработчиков по всему миру, которые обмениваются опытом и помогают друг другу через форумы, блоги и социальные сети.

Будущее Firebase. На сегодняшний день Firebase стала одной из самых популярных платформ для разработки мобильных и веб-приложений. Она используется миллионами разработчиков по всему миру благодаря своим богатым функциональным возможностям, надежности и поддержке от Google. Firebase продолжает играть ключевую роль в экосистеме Google Cloud, предлагая все необходимые инструменты для создания, тестирования, развертывания и масштабирования современных приложений.

В будущем Firebase планирует продолжать интеграцию с новыми технологиями и расширение своих возможностей. Это включает улучшение поддержки для разработки на различных платформах, интеграцию с новыми сервисами Google и внедрение инновационных функций для более эффективного управления и масштабирования приложений.

Firebase Realtime Database — это гибкая и масштабируемая облачная база данных, предназначенная для хранения и синхронизации данных в реальном времени между вашими клиентскими приложениями. Она основана на модели данных JSON и обеспечивает быстрый и надежный доступ к данным как в реальном времени, так и в автономном режиме.

Основные особенности Firebase Realtime Database.

Изменения данных в базе данных мгновенно отображаются на всех подключенных клиентах в реальном времени без необходимости обновления страницы или приложения.

Синхронизация данных. База данных Firebase автоматически синхронизирует данные между всеми подключенными клиентами, обеспечивая одинаковое состояние данных для всех пользователей.

Гибкая структура данных. Модель данных JSON позволяет легко организовывать данные и создавать гибкие структуры для вашего приложения.

Высокая доступность и масштабируемость. Firebase обеспечивает высокую доступность и масштабируемость вашей базы данных, что позволяет обрабатывать как небольшие, так и крупные нагрузки данных.

Безопасность данных. Firebase предоставляет механизмы аутентификации и авторизации для защиты данных от несанкционированного доступа.

Firebase Realtime Database интегрируется легко и непрерывно с мобильными и веб-приложениями, что делает ее отличным выбором для хранения и синхронизации данных в реальном времени для широкого спектра приложений, включая чаты, игры, приложения для социальных сетей и многое другое.

Помимо базы данных Firebase, Firebase также предоставляет инструменты для хранения файлов, таких как изображения, и для аутентификации пользователей. Давайте рассмотрим их более подробно:

Firebase Storage – это облачное хранилище, которое обеспечивает простой и надежный способ хранения файлов, таких как изображения, видео, аудио и другие медиафайлы. Firebase Storage интегрируется с другими сервисами Firebase, такими как Firebase Realtime Database и Firebase Authentication, обеспечивая безопасный и эффективный способ хранения и обмена файлами для вашего приложения.

Ключевые характеристики Firebase Storage.

Firebase Storage представляет собой мощное облачное хранилище файлов, обладающее рядом важных преимуществ для разработчиков. Сервис отличается продуманным и удобным интерфейсом API, который значительно упрощает процессы загрузки и скачивания файлов различного типа, что сокращает время на интеграцию и снижает порог входа для новых

разработчиков. Особое внимание уделено вопросам информационной безопасности – система предоставляет гибкие инструменты управления доступом, позволяющие тонко настраивать права доступа к файлам в зависимости от различных параметров, включая аутентификацию пользователей. Технологическая платформа сервиса построена с учетом требований к масштабируемости, гарантируя стабильную работу как с небольшими объемами данных, так и при обработке значительных потоков информации. Важным конкурентным преимуществом является глубокая интеграция с другими компонентами Firebase–экосистемы, включая Realtime Database и Authentication, что обеспечивает комплексное решение для хранения и управления файлами, связанными с пользовательскими данными и другими элементами приложения.

Аутентификация пользователей: Firebase Authentication.

Firebase Authentication – это сервис, предоставляемый Firebase, который обеспечивает простой и безопасный способ аутентификации пользователей в вашем приложении. С помощью Firebase Authentication вы можете предложить пользователям различные методы аутентификации, такие как адрес электронной почты, номер телефона, социальные сети и многое другое, а также управлять их аутентификационными данными и безопасностью.

Firebase Authentication представляет собой мощное решение для управления пользовательской аутентификацией, сочетающее в себе гибкость, безопасность и простоту интеграции. Одним из ключевых преимуществ сервиса является поддержка множества методов аутентификации, включая вход по email и паролю, через социальные сети (Google, Facebook, Twitter), телефонные номера (SMS) и даже систему OAuth–провайдеров. Это позволяет пользователям выбирать наиболее удобный для них способ авторизации, повышая конверсию и улучшая пользовательский опыт.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

Безопасность данных является приоритетом Firebase Authentication – сервис использует современные алгоритмы шифрования, защиту от брутфорс–атак и автоматическое хеширование паролей, что минимизирует риски утечки конфиденциальной информации. Кроме того, разработчики получают встроенные инструменты для управления учетными записями: блокировка подозрительных аккаунтов, сброс паролей, верификация email и телефонов, а также настройка прав доступа через кастомные claims.

Firebase Authentication легко интегрируется с другими сервисами платформы, такими как Firebase Realtime Database, Cloud Firestore и Storage, обеспечивая безопасный доступ к данным на основе ролей пользователей. Например, можно настроить правила безопасности так, чтобы аутентифицированные пользователи могли загружать файлы в Storage или читать определенные узлы базы данных.

Таким образом, Firebase Authentication не только упрощает процесс внедрения авторизации в приложение, но и обеспечивает высокий уровень безопасности, масштабируемости и интеграции с другими инструментами Firebase, что делает его идеальным выбором для проектов любого масштаба.

1.6 Эмуляторы и устройства для тестирования

Android Studio – это интегрированная среда разработки (IDE) от Google, предназначенная для создания приложений под операционную систему Android. В состав Android Studio входит множество инструментов, включая редактор кода, средства отладки, симуляторы устройств и многое другое. Один из ключевых инструментов Android Studio – это эмулятор Android, который позволяет разработчикам запускать и тестировать свои приложения на виртуальных устройствах, эмулирующих различные конфигурации и версии операционной системы Android.

Основные особенности эмулятора Android в Android Studio. Широкий выбор устройств. Эмулятор Android предоставляет доступ к широкому

выбору виртуальных устройств, включая различные размеры экранов, разрешения и версии операционной системы Android, что позволяет разработчикам тестировать свои приложения на различных устройствах.

Эмулятор Android позволяет настраивать различные параметры виртуальных устройств, такие как количество памяти, разрешение экрана, версия операционной системы и другие параметры, что делает его удобным инструментом для тестирования приложений в различных условиях.

Быстрая загрузка и запуск. Эмулятор Android обеспечивает быструю загрузку и запуск виртуальных устройств, что позволяет разработчикам мгновенно начинать тестирование своих приложений без необходимости настройки физического устройства.

Интеграция с Android Studio. Эмулятор Android интегрируется непосредственно в Android Studio, что обеспечивает единый и удобный интерфейс для разработки и тестирования приложений.

Эмулятор Android позволяет использовать сенсорные устройства, такие как сенсорный экран и акселерометр, для более реалистичного тестирования приложений.

В целом, эмулятор Android в Android Studio является мощным и удобным инструментом для разработки и тестирования приложений под операционную систему Android, который позволяет разработчикам быстро и эффективно создавать высококачественные приложения для широкого спектра устройств и конфигураций.

Физические устройства. В дополнение к эмуляторам, разработчики также могут использовать реальные физические устройства для тестирования своих приложений. Это предоставляет ряд преимуществ и возможностей, которые эмуляторы не могут полностью воспроизвести.

Реалистичное тестирование. Использование физических устройств позволяет более точно оценить производительность, поведение и пользовательский опыт приложения на реальном оборудовании. Эмуляторы

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		25

могут приблизиться к реальным условиям, но они не могут полностью воссоздать все аспекты использования приложения на реальном устройстве.

Тестирование аппаратных особенностей. Физические устройства позволяют проверять работу приложения с различными аппаратными особенностями, такими как камера, датчики, GPS и т. д.

Это особенно важно для приложений, зависящих от конкретных аппаратных функций.

Реальные условия использования. При использовании физических устройств разработчики могут тестировать приложение в реальных условиях использования, таких как с разной силой сигнала, разным уровнем яркости экрана и т. д. Это позволяет обнаруживать потенциальные проблемы и улучшать пользовательский опыт.

Отладка на реальном устройстве. Использование физических устройств обеспечивает возможность отладки приложения на реальном оборудовании, что может быть полезно для выявления и исправления проблем, которые могут возникнуть только на определенных устройствах.

Хотя использование физических устройств требует дополнительных затрат на приобретение и обслуживание оборудования, оно является необходимым этапом в процессе разработки приложений для обеспечения высокого качества и удовлетворения потребностей пользователей.

1.7 Проектирование и разработка приложения

1.7.1 Архитектура приложения

Архитектурный шаблон MVC (Model–View–Controller) предназначен для разделения бизнес-логики и её визуального представления. Это разделение способствует повышению возможности повторного использования кода. Данная концепция особенно полезна, когда пользователю необходимо одновременно видеть одни и те же данные в различных контекстах или с разных точек зрения.

Шаблон состоит из трех основных компонентов: модели, представления и контроллера. Модель отвечает за хранение данных и выполнение логики приложения. Представление служит интерфейсом для взаимодействия пользователя с приложением, отображая информацию и позволяя пользователю взаимодействовать с системой. Контроллер выполняет роль связующего звена между моделью и представлением, обрабатывая запросы пользователя, работая с моделью и получая данные от представления.

Логическая схема MVC-приложения представлена на рисунке 1.

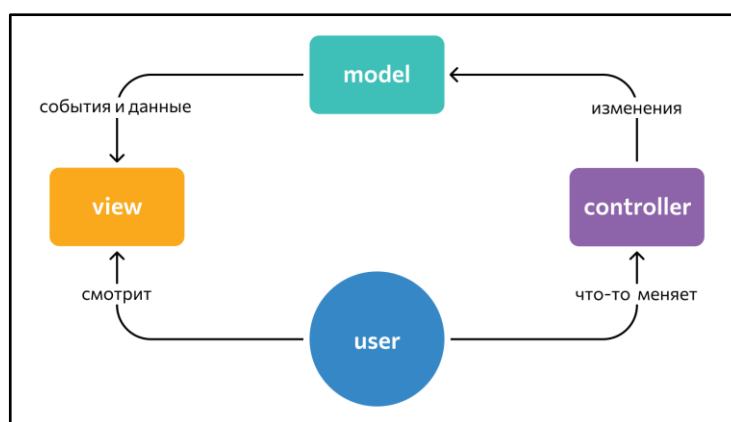


Рисунок 1 – Логическая схема MVC-приложения

После того как пользователь взаимодействует с представлением, начинается обмен данными. Когда модель обновляется, контроллер обновляет представление, записывая актуальные данные, что позволяет пользователю видеть обновленную информацию.

Основным преимуществом архитектуры MVC является четкое распределение задач, что упрощает организацию кода. Это позволяет тестировать каждый компонент отдельно, что помогает выявлять и устранять ошибки на этапе разработки. Каждый компонент может быть повторно использован в других проектах, что экономит время разработчиков. Архитектура MVC также обеспечивает масштабируемость, позволяя легко добавлять или удалять компоненты для адаптации приложения. Этот шаблон

часто используется для разработки веб-приложений, где браузер выполняет роль представления, а сервер содержит модель и контроллер.

Примером работы архитектуры MVC может служить любое веб-приложение в сфере электронной коммерции. Данные о товарах, пользователях и заказах хранятся в модели. Страницы оформления заказа, каталога товаров и информации о пользователе составляют представление. Контроллер отвечает за обработку запросов, получение данных и их отображение в представлении, а также за действия пользователей по добавлению товаров в корзину или оформлению заказа.

На рисунке 2 представлена UML-диаграмма классов.

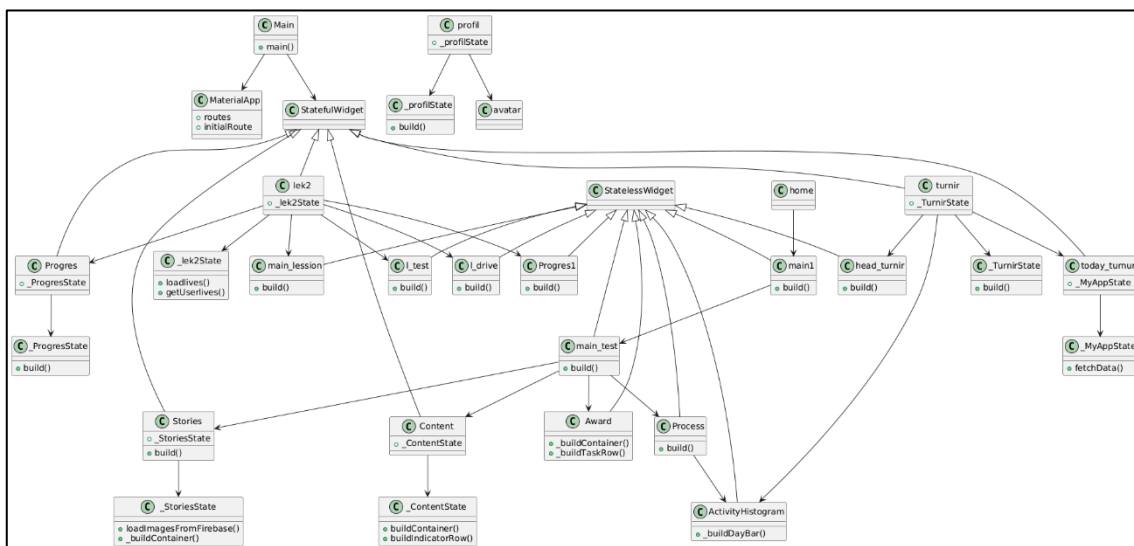


Рисунок 2 – UML-диаграмма классов

На рисунке 3 представлена UML-диаграмма развертывания.

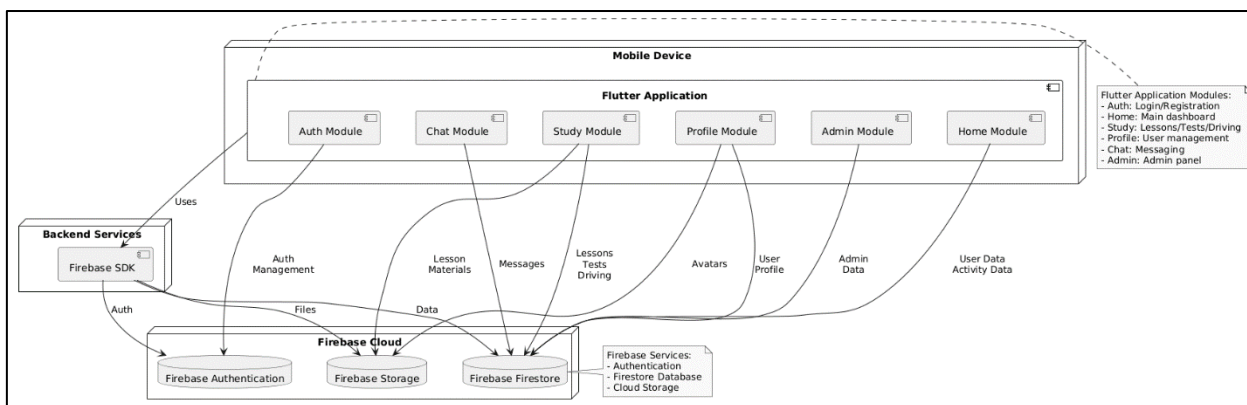


Рисунок 3 – UML-диаграмма развертывания

На рисунке 4 представлена UML-диаграмма прецедентов.

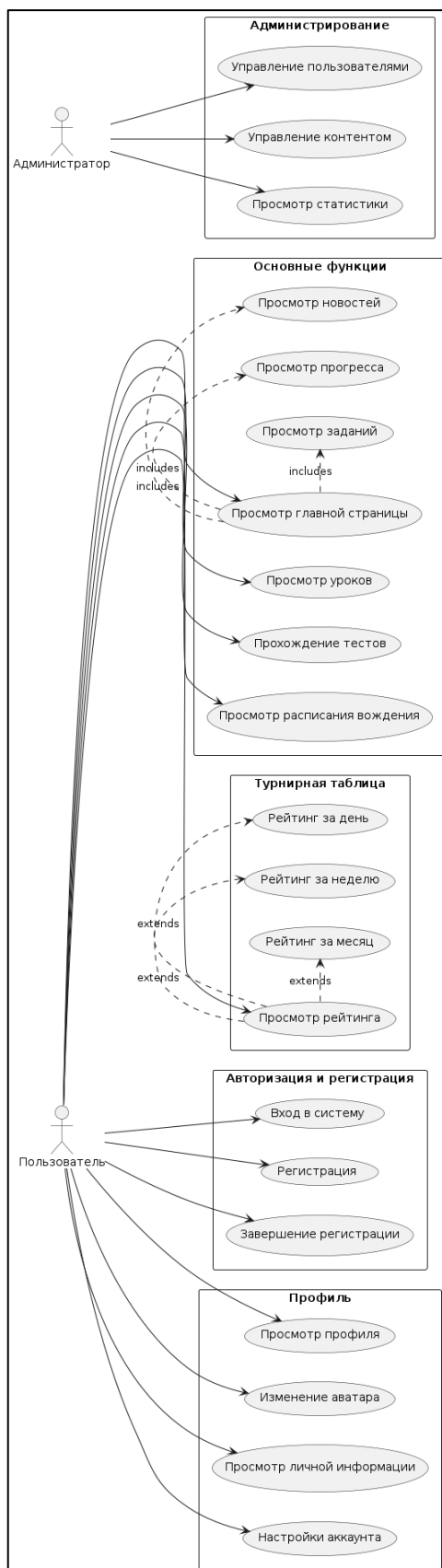


Рисунок 4 – UML-диаграмма прецедентов

На рисунке 5 представлена UML-диаграмма деятельности.

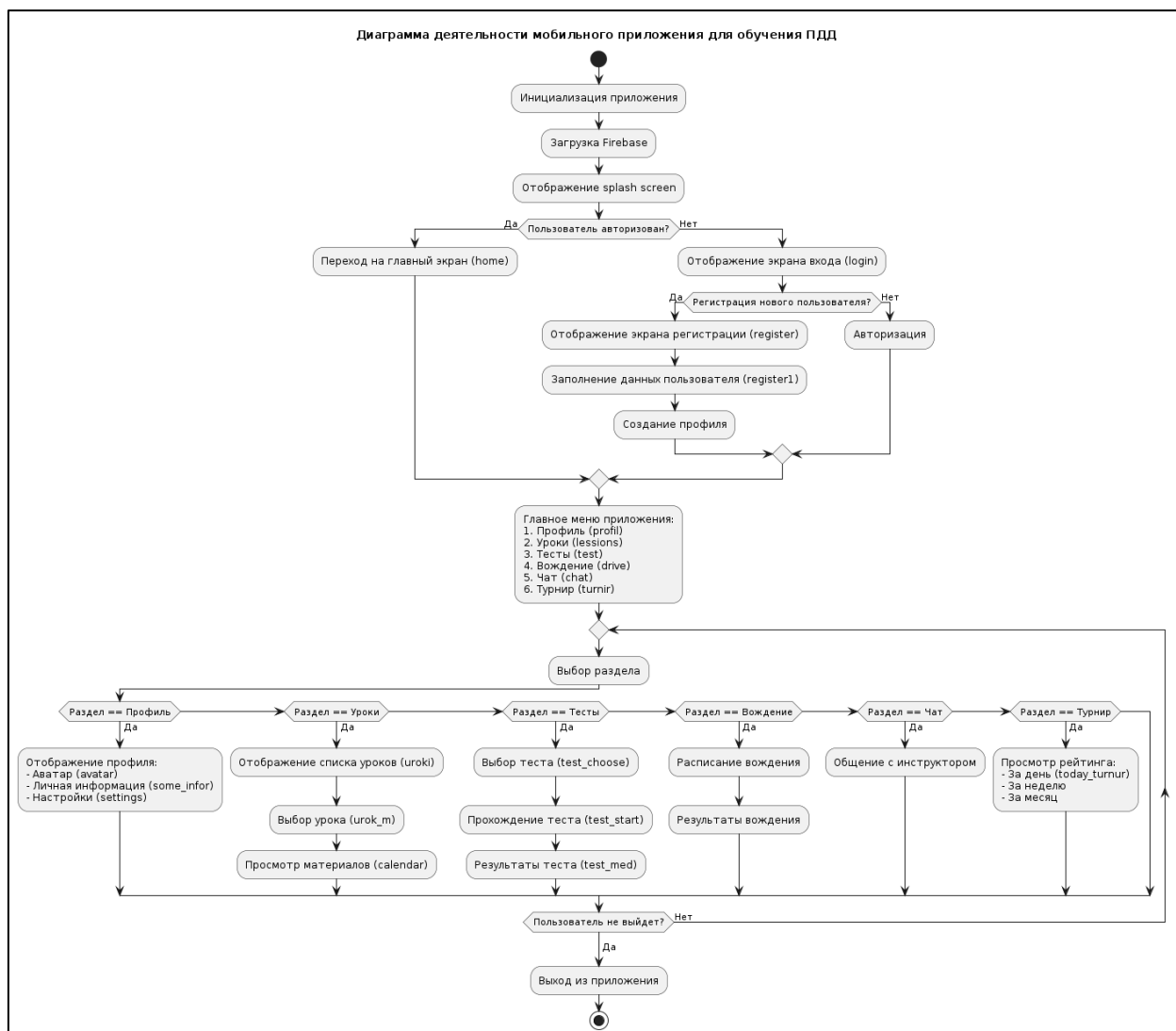


Рисунок 5 – UML-диаграмма деятельности

На рисунке 6 представлена UML-диаграмма последовательностей.

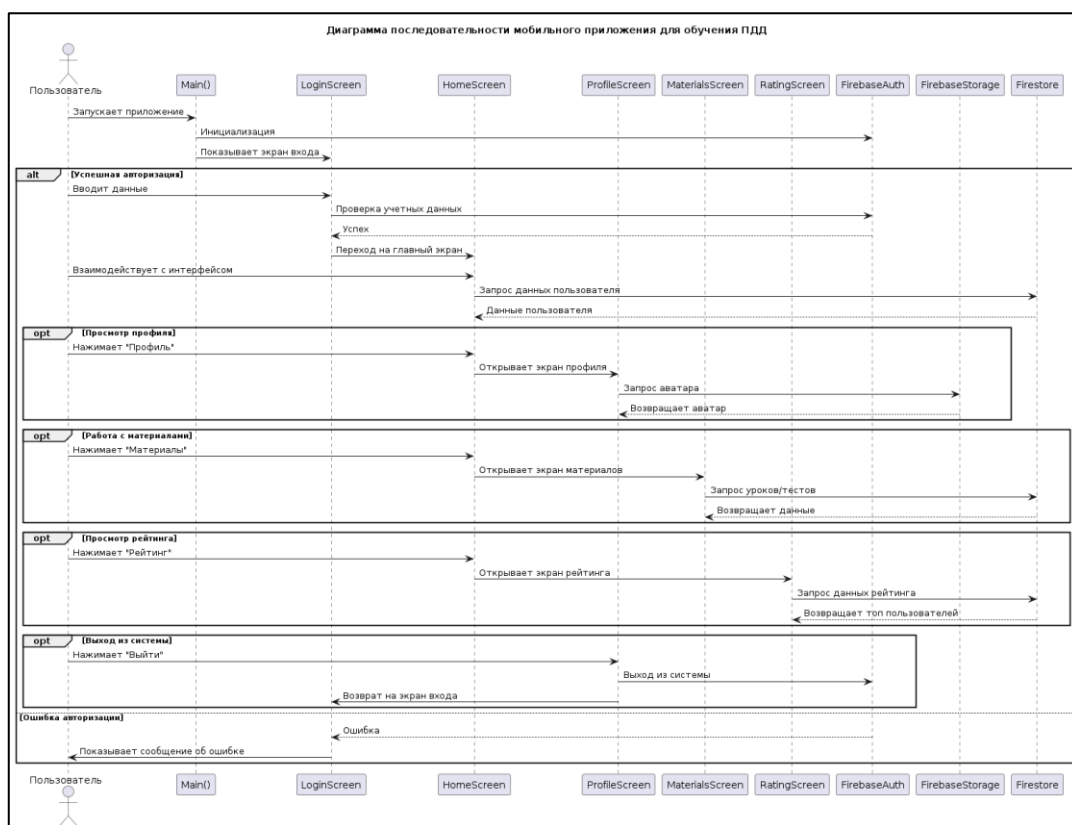


Рисунок 6 – UML-диаграмма последовательностей

1.7.1 Разработка модулей и классов

Класс User представляет собой основную сущность, идентифицирующую каждого участника платформы. Он содержит уникальный идентификатор, имя пользователя, электронную почту и процент завершения курса. Также реализован метод для получения текущего прогресса, который может использоваться в аналитике и визуализации данных.

ProfileScreen отвечает за отображение информации о пользователе в интерфейсе. Содержит: аватар, статистику прохождения материала и функции редактирования профиля. Класс связан с пользовательским интерфейсом и используется для взаимодействия с персональными данными.

Achievement хранит информацию о достижениях, полученных пользователем за активность. Каждое достижение имеет название, иконку и

флаг, определяющий, было ли оно открыто. Класс может использоваться в мотивационных механиках (рисунок 7).

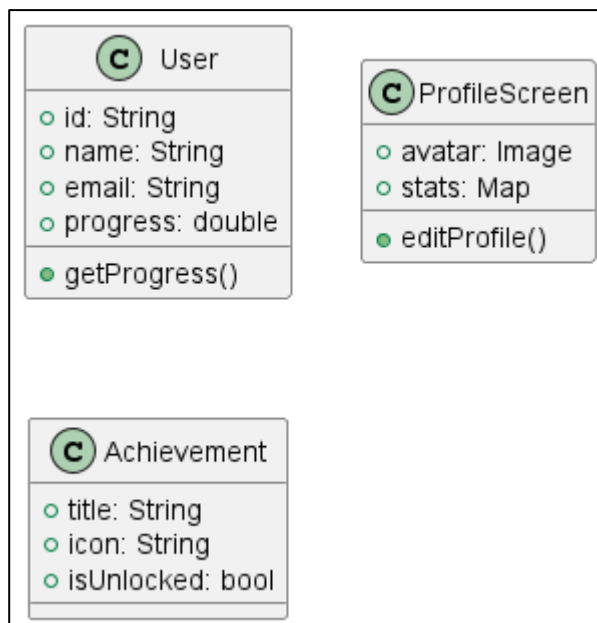


Рисунок 7 – UML-диаграмма классов User, ProfileScreen, Achievement

Lesson – основной элемент обучающего контента. Включает уникальный идентификатор, заголовок, продолжительность и ссылку на обучающее видео. Метод markAsCompleted() позволяет зафиксировать завершение урока пользователем.

Quiz представляет собой модуль для самопроверки. Содержит список вопросов, ограничение по времени и метод отправки ответов. Используется после прохождения теоретического материала.

RoadSign – класс описывает дорожные знаки: изображение, краткое описание и категорию. Эти данные применяются для теоретической части обучения и тестов.

PracticalDriving -отражает практическую часть. Содержит информацию о месте проведения, назначенном инструкторе и функцию для планирования практических занятий (рисунок 8).

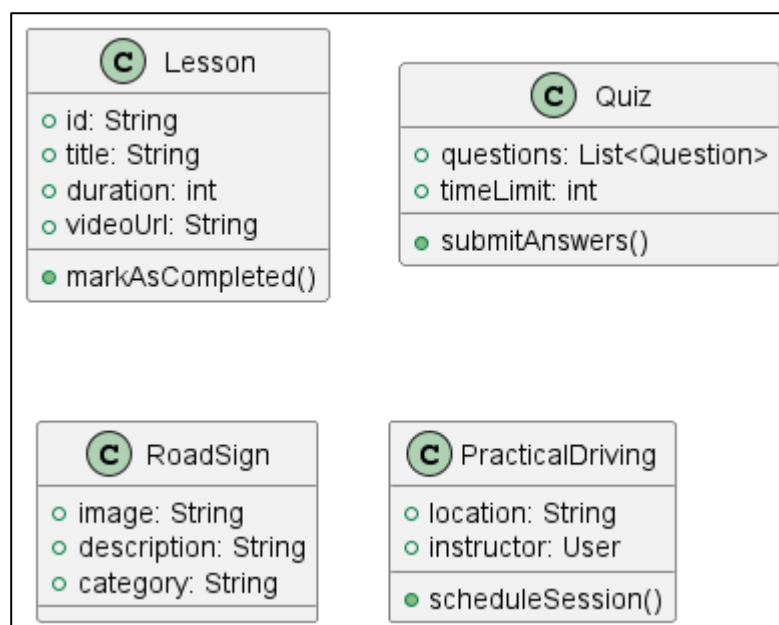


Рисунок 8 – UML-диаграмма классов Lesson, Quiz, RoadSign, PracticalDriving

ExamSimulator – моделирует сдачу экзамена в условиях, приближенных к реальным. Хранит список вопросов, устанавливает таймер и генерирует отчёт по завершении теста.

Question – базовая структура для тестовых заданий. Содержит текст вопроса, варианты ответов и правильный индекс. Метод explainAnswer() позволяет отобразить разъяснение для выбранного ответа.

TestResult – фиксирует результат тестирования: итоговый балл, список ошибок и предоставляет метод для сохранения результата в историю пользователя (рисунок 9).

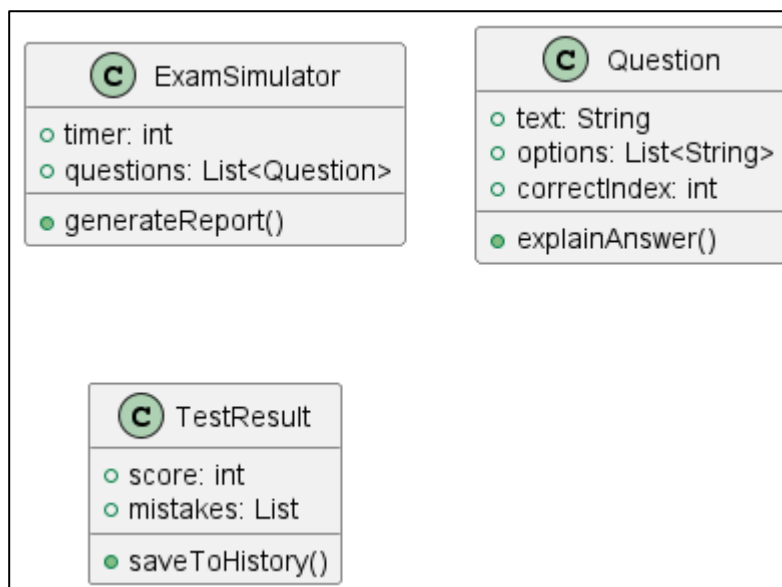


Рисунок 9 – UML-диаграмма классов ExamSimulator, Question, TestResult

ProgressTracker – служит для отслеживания динамики обучения. Сохраняет количество завершённых уроков и среднюю оценку, а также предоставляет метод анализа слабых тем.

AnalyticsDashboard – интерфейсный компонент, визуализирующий данные: прогресс в виде графиков, общий рейтинг, позволяет формировать отчёты в формате PDF (рисунок 10).

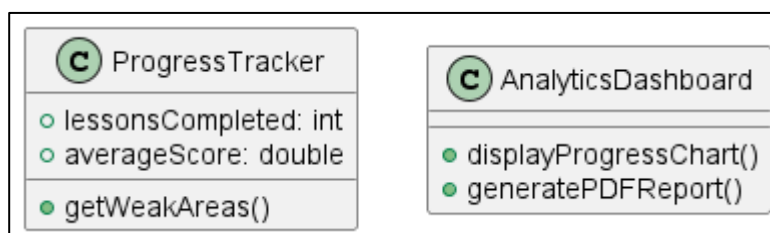


Рисунок 10 – UML-диаграмма классов ProgressTracker, AnalyticsDashboard

PaymentGateway – абстракция над платёжной системой. Позволяет проводить транзакции, а также получать историю покупок пользователя. Интегрируется с внешними сервисами.

VirtualGoods – представляет товары, которые можно приобрести: их название, цена и метод покупки. Может использоваться для разблокировки контента или персонализации (рисунок 11).

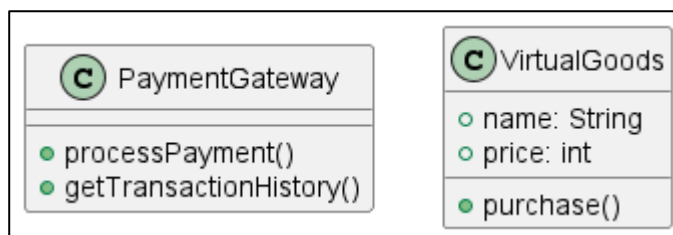


Рисунок 11 – UML-диаграмма классов PaymentGateway, VirtualGoods

NotificationService – реализует механизм отправки уведомлений: напоминаний о тестах, новых занятиях или достижениях. Может работать как с локальными, так и с push–уведомлениями.

Settings – хранит пользовательские настройки: тему интерфейса, предпочтения по уведомлениям и прочие параметры. Обеспечивает сохранение и восстановление конфигурации (рисунок 12).

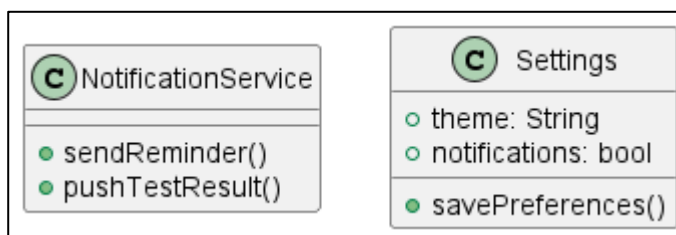


Рисунок 12 – UML-диаграмма классов NotificationService, Settings

1.7.3 Разработка СУБД

База данных является важным элементом для хранения данных о пользователях, уроках, тестовых вопросах и их результатах. В нашем приложении мы выбрали Firebase Realtime Database благодаря ее простой интеграции с Android и возможностями работы в реальном времени.

Структура базы данных включает следующие основные таблицы (или коллекции):

Users – хранит информацию о пользователях (имя, электронная почта, прогресс и т.д.). А также хранит выбранную категорию, аватар и процесс обучение (рисунок 13 схема базы данных users).

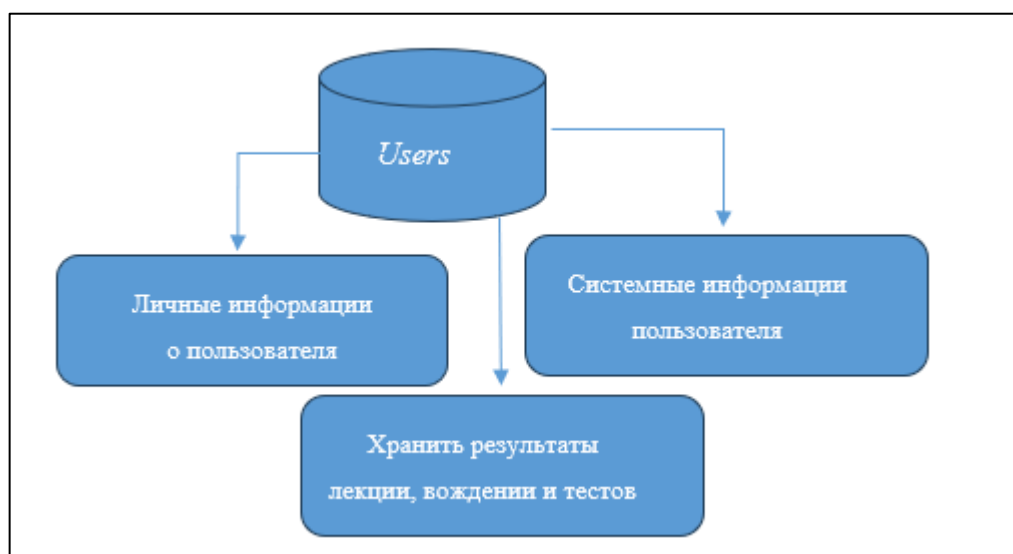


Рисунок 13 – Схема users

Lessions – хранит все информации всего категории материалы, тесты, а также информация о вождении (рисунок 14 схема базы данных Lessons).

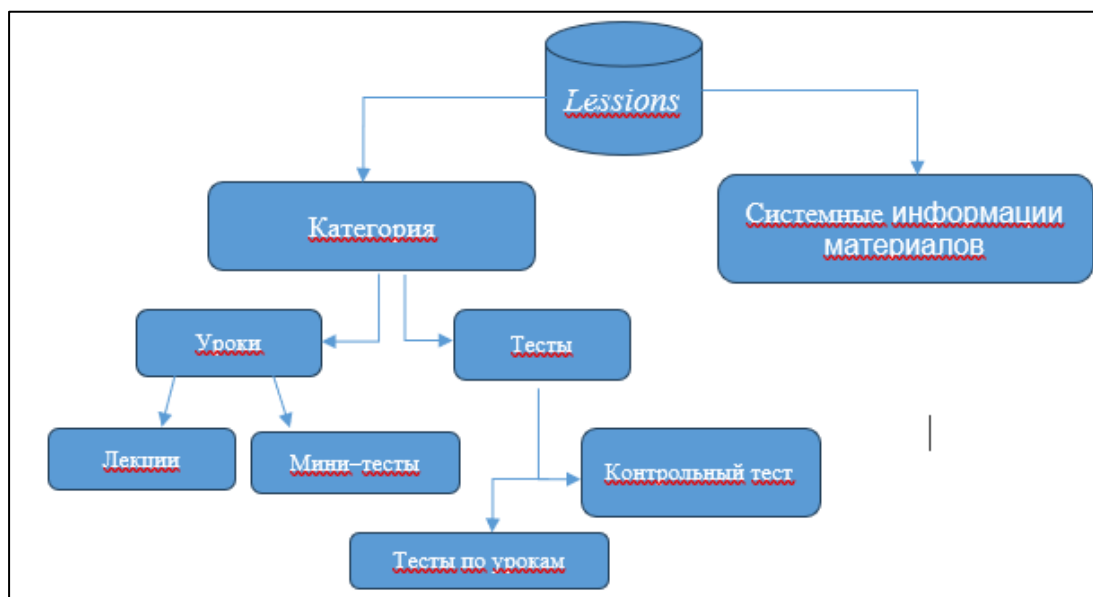


Рисунок 14 – Схема lessons

Admin – данные для административной части.

Это таблица для админов и хранить логин и пароль. Админ может войти через сайт и может добавлять, редактировать, изменять и удалять:

новости, лекции, тесты, аватары. А также может получить информацию о пользователях и удалять.

Systems – информация о картинках и системные настройки

Все текстовые данные хранятся в Firebase Database, а медиа файлы (картинки, видео и. т. д.) хранятся в Firebase Storage.

1.7.4 Разработка интерфейса

Интерфейс – критический важный компонент приложения, выступающий посредником между пользователем и программой. В нашем проекте для создания UI применяется фреймворк Jetpack Compose – это современное решение для построения гибких и адаптивных интерфейсов.

Процесс работы с приложением начинается с экрана аутентификации. Для существующих пользователей предусмотрена стандартная форма ввода учетных данных (логина и пароля), после успешной проверки которых открывается основной функционал (рисунок 15). Новым пользователям необходимо сначала пройти процедуру регистрации, введя требуемую информацию в соответствующие поля визуализация процесса представлена на рисунках 16 и 17.

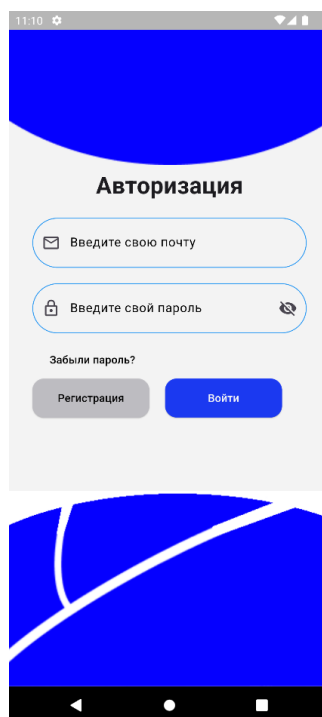


Рисунок 15 – Основной функционал приложения

Рисунок 16 – Поле визуализации

Рисунок 17 – Поле визуализации

После успешной авторизации пользователь получает доступ к основному интерфейсу приложения.

Верхняя панель содержит ленту актуальных новостей, связанных с вождением и дорожным движением (рисунок 18).



Рисунок 18 – Лента новостей

Включает кнопку уведомлений для просмотра пропущенных оповещений (рисунок 19)



Рисунок 19 – Кнопка уведомлений

Центральный контентный блок представляет собой интуитивно понятный интерфейс для доступа к учебным материалам. Позволяет продолжить обучение с последней остановленной точки. Оптимизирует временные затраты пользователя на поиск нужных уроков

Панель прогресса визуализирует активность пользователя за последнюю неделю. Отображает статистику выполнения заданий.

Боковая панель навигации (рисунок 20).

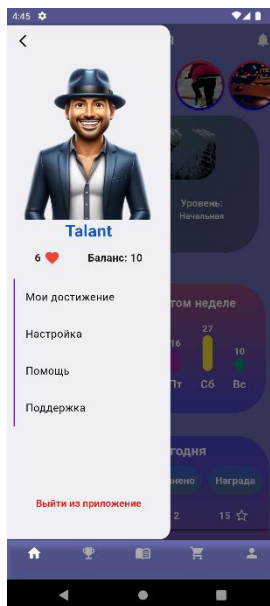


Рисунок 20 – Боковая панель навигации

Обеспечивает быстрый доступ к ключевым разделам:

- достижения и результаты;
- параметры приложения;
- справочный раздел;
- техническая поддержка;
- выход из системы.

Основное навигационное меню содержит пять ключевых разделов:

- главная страница;
- рейтинги пользователей;
- учебные материалы;
- внутри приложенный магазин;

- личный кабинет.

Рейтинг обновляется каждую неделю и в конце недели награждает пользователей за определенные занятое места. Небольшая мотивация для пользователя, чтобы больше учился (рисунок 21)

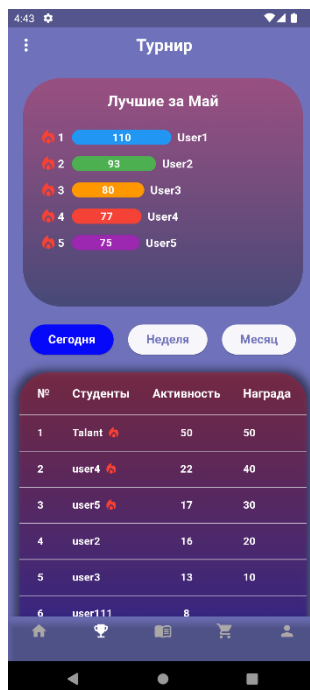


Рисунок 21 – Рейтинг

В разделе находится все нужные материалы, вождение, и тесты по курсу (рисунок 22).

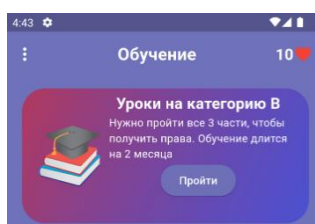


Рисунок 22 – Материалы по курсу

Когда нажимает на кнопку «Пройти» появляется выбор:

- уроки;
- тесты;
- вождение.

Уроки – основные материалы: теории, правила дорожной движения, знаки, видео материалы и. так далее (рисунок 23).

Тесты – тесты по урокам и контрольный тест.

Вождение – информация и место сдачи практическую вождению.

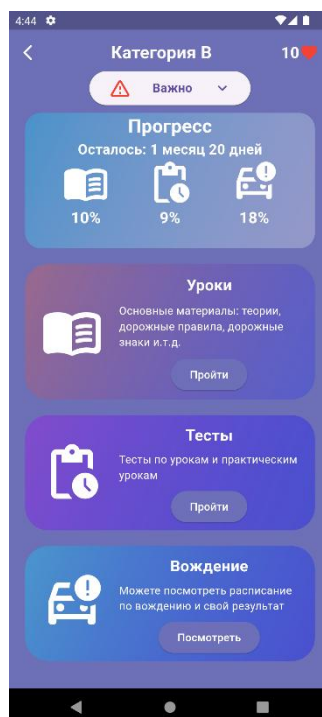


Рисунок 23 – Основной материал

В этом приложение есть внутри валютная монета. За монету можно купить жизнь.

Жизнь – попытка пересдать проваливший урок или тест (не считается контрольный тест). Добавил жизнь, чтобы замотивировать пользователя для внимательного прохождения уроков и тестов.

Монеты можно получить таким образом: получение определенного места в рейтинге, достижение, за отличного прохождения некоторых уроков, либо тестов (рисунок 24).

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		42

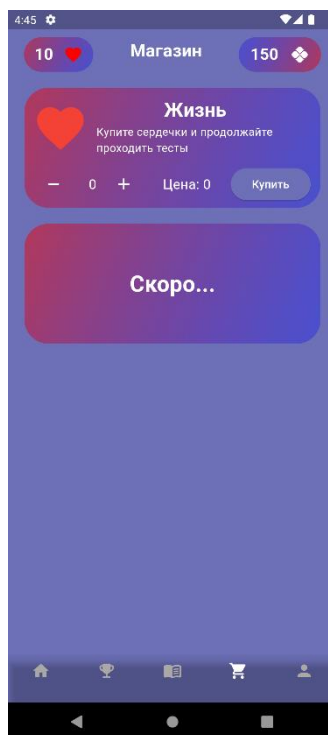


Рисунок 24 – Магазин

Личная информация пользователя, пользователь может посмотреть и изменять данные. Например, поменять изображения аватара, почту и пароль.

Настройка. Можно настроить системные настройки: уведомление, цвет фона.

Поддержка. Если у пользователя появились проблемы или ошибки в приложении или в учебе, он может обратиться в поддержку отправив сообщение и поддержка поможет решить данную проблему (рисунок 25).



Рисунок 25 – Профиль

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		43

1.8 Административная панель

В системе предусмотрена возможность управления пользователями, которая позволяет осуществлять просмотр и редактирование их данных, а также при необходимости блокировать аккаунты. Кроме того, панель предоставляет инструменты для работы с уроками, включая добавление новых материалов, внесение изменений и удаление уроков или видеоконтента.

Также реализовано управление тестами, что включает создание вопросов, а также редактирование и удаление уже существующих позиций. Для удобства администраторов доступна функция просмотра статистики, позволяющая анализировать результаты прохождения тестов и активность пользователей на платформе.

Техническая реализация административной панели выполнена в виде веб-приложения с использованием Python и облачной базы данных Firebase Firestore. Серверная часть реализована на основе Firebase Functions.

1.9 Тестирование приложения

Для обеспечения стабильной работы мобильного приложения "Онлайн автошкола" проводится комплексное тестирование на различных платформах.

Использование эмуляторов Android Studio позволяет: проверять работу приложения на разных версиях ОС, тестировать на устройствах с различными характеристиками, выявлять проблемы совместимости.

Основные этапы тестирования:

- настройка виртуальных устройств;
- проведение автоматизированных тестов;
- выполнение ручной проверки интерфейса.

Тестирование на реальных устройствах

Помимо тестирования на эмуляторах, также важно проводить тестирование на реальных физических устройствах. Это позволяет получить более точные данные о производительности, поведении приложения и пользовательском опыте. Разработчики тестируют приложение на различных моделях устройств с разными характеристиками, чтобы убедиться в его корректной работе на разных аппаратных платформах. Основные этапы включают: различные модели смартфонов, устройства с разными техническими характеристиками проверка в реальных условиях использования.

Анализ результатов тестирования

По результатам тестирования выявляются возможные проблемы, ошибки и недочеты, которые требуют исправления. Разработчики анализируют отчеты о тестировании, идентифицируют причины возникновения проблем и предпринимают меры для их устранения. Процесс анализа включает:

- сбор логов и отчетов о сбоях, предоставленных автоматическими тестами и обратной связью от пользователей;
- анализ данных для выявления основных причин проблем, таких как утечки памяти, сбои приложения или низкая производительность;
- определение приоритетности исправления выявленных проблем в зависимости от их критичности и влияния на пользователей.

Оптимизация и улучшение производительности

На основе результатов тестирования и обратной связи от пользователей разработчики вносят оптимизации и улучшения в приложение. Это может включать в себя оптимизацию работы приложения, улучшение интерфейса пользователя, исправление ошибок и добавление нового функционала.

Тестирование на различных платформах, включая эмуляторы Android Studio и реальные физические устройства, является важным этапом в

разработке мобильного приложения "Онлайн–автошкола". Этот процесс позволяет обеспечить надежную работу приложения, выявить и устранить проблемы, обеспечить высокое качество и удобство использования для пользователей.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		46

2 Охрана труда

В соответствии со статьей 25 Федерального закона №52-ФЗ от 30.03.2015 "О санитарно-эпидемиологическом благополучии населения" условия труда должны исключать вредное воздействие на здоровье работников, что также подтверждается статьей 209 ТК РФ, определяющей условия труда как комплекс факторов производственной среды, влияющих на работоспособность и здоровье сотрудников.

Вредными признаются факторы, способные привести к профессиональным заболеваниям. Основные требования к организации рабочих мест включают соблюдение норм освещенности, что подразумевает допустимость использования персональных электронных вычислительных машин в помещениях без естественного освещения при подтверждении соответствия нормам безопасности, а также рекомендуется ориентировать оконные проемы на север или северо-восток с установкой регулируемых светозащитных систем. Нормы освещенности должны составлять 300-500 люкс для рабочей поверхности и до 300 люкс для поверхности экрана, при этом обязательно использование антибликовых защитных элементов.

Организация рабочего пространства требует, чтобы ширина столешницы составляла 800-1400 мм, глубина — 800-1000 мм, а высота — 725 мм, при этом монитор должен располагаться на расстоянии 50-70 см от глаз пользователя, а клавиатура — на расстоянии 10-30 см от края стола. Важным аспектом является эргономика, которая подразумевает использование регулируемых кресел с возможностью изменения высоты и угла наклона, а также применение подставок для ног с размерами не менее 300 мм в ширину и 400 мм в глубину.

Все параметры рабочего места должны регулироваться в соответствии с антропометрическими данными сотрудника. Электробезопасность требует обязательного оборудования помещений

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		47

системами защитного заземления и соблюдения норм эксплуатации электрооборудования. Обязанности работодателя включают разработку и регулярное обновление инструкций по охране труда не реже одного раза в пять лет, проведение санитарно-противоэпидемических мероприятий, обеспечение соответствия рабочих мест требованиям санитарных правил и нормативных актов, а также предупреждение профессиональных заболеваний и травматизма.

Все указанные требования основаны на положениях Трудового кодекса РФ (ст. 209, 212), Санитарных правилах и методических рекомендациях Минтруда России.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		48

Заключение

В условиях стремительной цифровизации и технологического прогресса мобильное приложение «Онлайн авто – школа» воплощает в себе инновационный подход к обучению вождению. Данный проект не только соответствует актуальным трендам в сфере мобильного образования, но и задает новые ориентиры в области удобного и интерактивного обучения.

Важным этапом разработки стало тестирование приложения на разных платформах. Проверка на эмуляторах Android Studio и реальных устройствах помогла выявить и исправить потенциальные ошибки, обеспечив стабильную работу программы в различных условиях.

Особый акцент был сделан на дизайне и удобстве интерфейса. Применение лаконичной цветовой гаммы на основе синих оттенков способствует созданию спокойной и доверительной атмосферы, что особенно важно для пользователей, осваивающих вождение. Простота навигации и интуитивно понятный интерфейс повышают эффективность обучения, делая его комфортным для учеников.

Проект «Онлайн авто – школа» продолжает развиваться. В ближайших планах расширение функциональных возможностей, обновление учебных материалов и интеграция передовых технологий. Среди перспективных направлений внедрение искусственного интеллекта для адаптивного обучения, добавление новых образовательных модулей и улучшение пользовательского опыта.

В итоге была успешно разработана мобильная платформа «Онлайн авто – школа», которая служит эффективным инструментом для подготовки водителей и способствует повышению безопасности дорожного движения. Этот проект наглядно демонстрирует потенциал цифровых технологий в образовательной сфере и подчеркивает значимость инноваций в области водительской подготовки.

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		49

Список использованных источников

1. Смирнов, П.А. Руководство по разработке мобильных приложений на платформе Android. СПб.: Издательство "БХВ–Петербург", 2019. 320 с.
2. Лебедев, Г.М. Программирование мобильных устройств на примере Android. М.: Издательство "Лори", 2018. 280 с.
3. Куликов, Е.Д. Основы тестирования программного обеспечения. М.: Издательство "Бином", 2020. 220 с.
4. Горшкова, Л.Н. Firebase для мобильных разработчиков: практическое руководство. Киев: Издательство "Технологии", 2017. 190 с.
5. Жуков, В.И. Основы архитектуры мобильных приложений. М.: Издательство "Питер", 2021. 240 с.

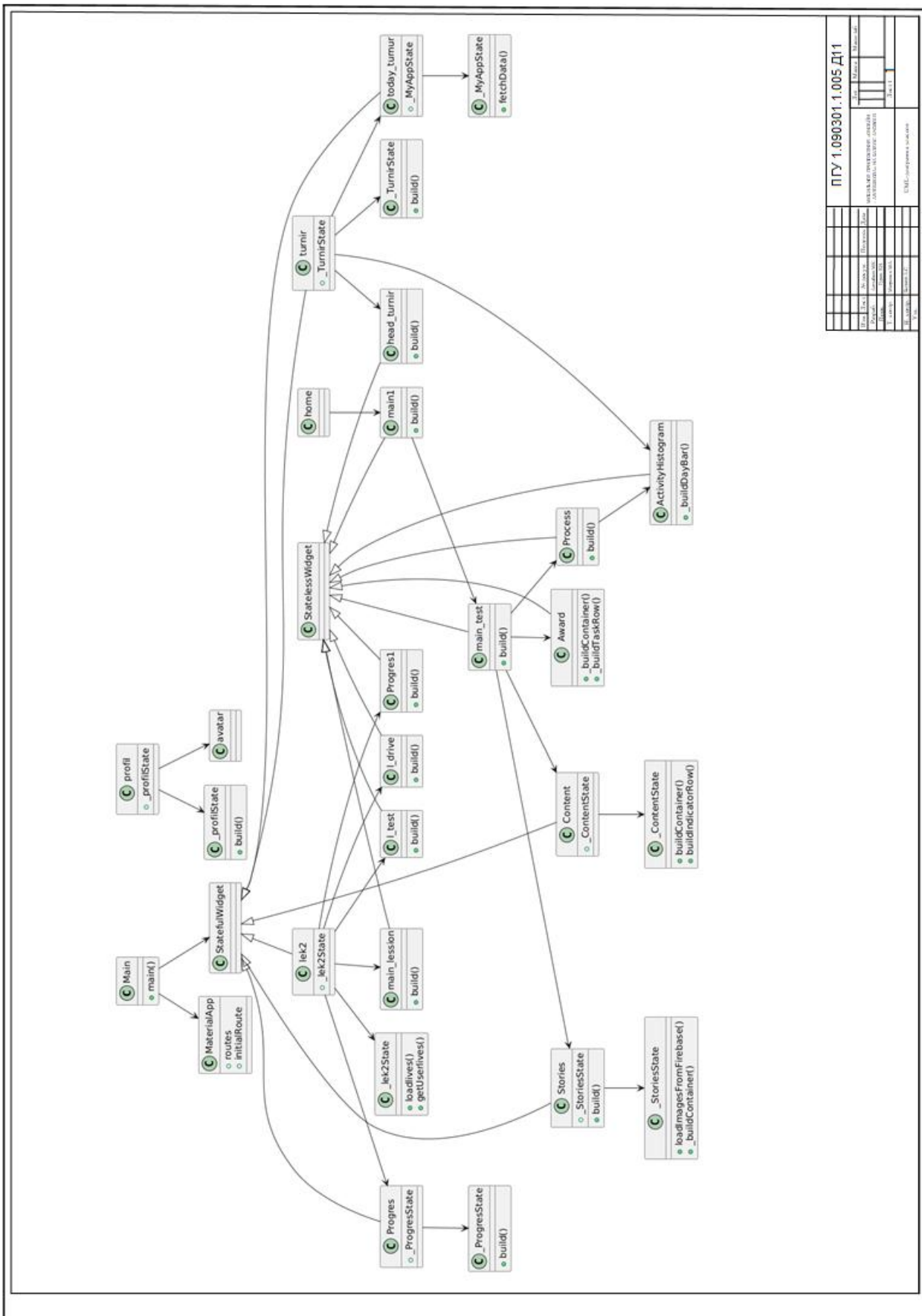
Приложение А

(обязательное)

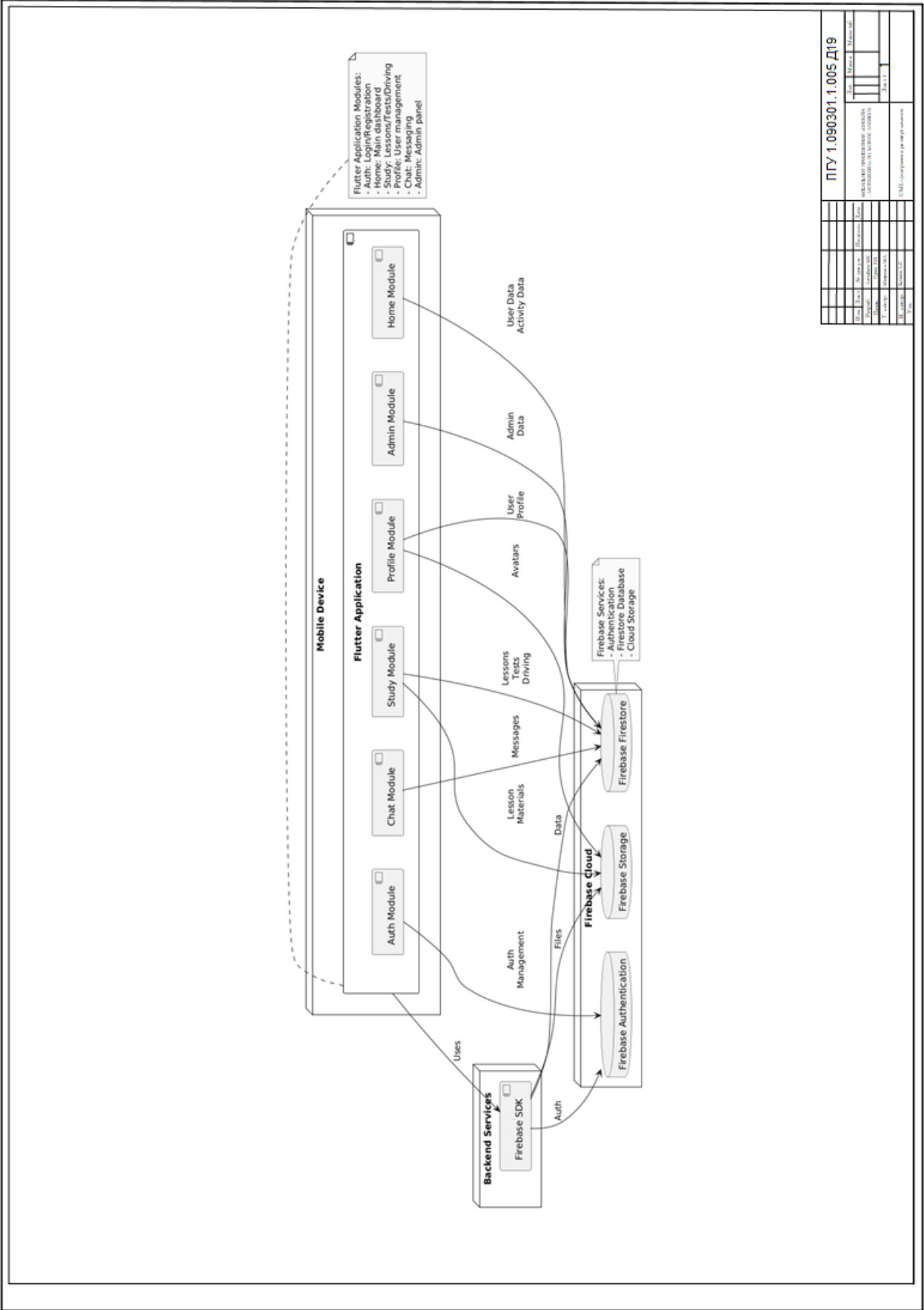
Мобильное приложение «Онлайн–автошкола» на базе ОС Android

UML диаграммы

					ПГУ 1.090301.1.001 ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

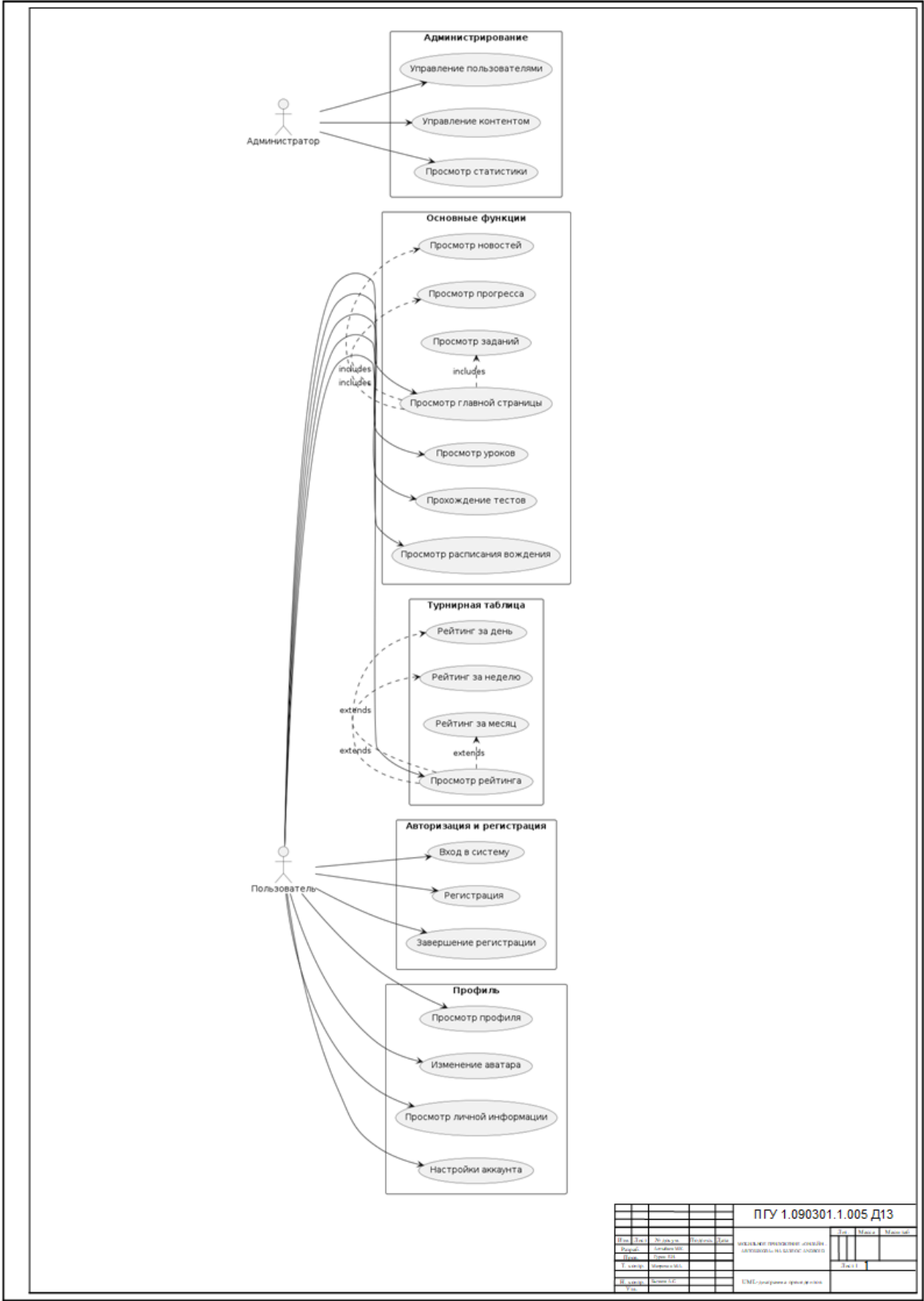


ПГУ 1.090301.1.005 Д11									
№ п/п	№ докум.	Дата	Исполнитель	Проверенный	Согласованный	Согласованный	Согласованный	Согласованный	Согласованный
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									



ПГУ 1.090301.1.005 Д19									
Изм.	Лист	№ докум.	Подпись	Дата	Исполнитель	Проверка	Утверждение	Внесение	Модификация

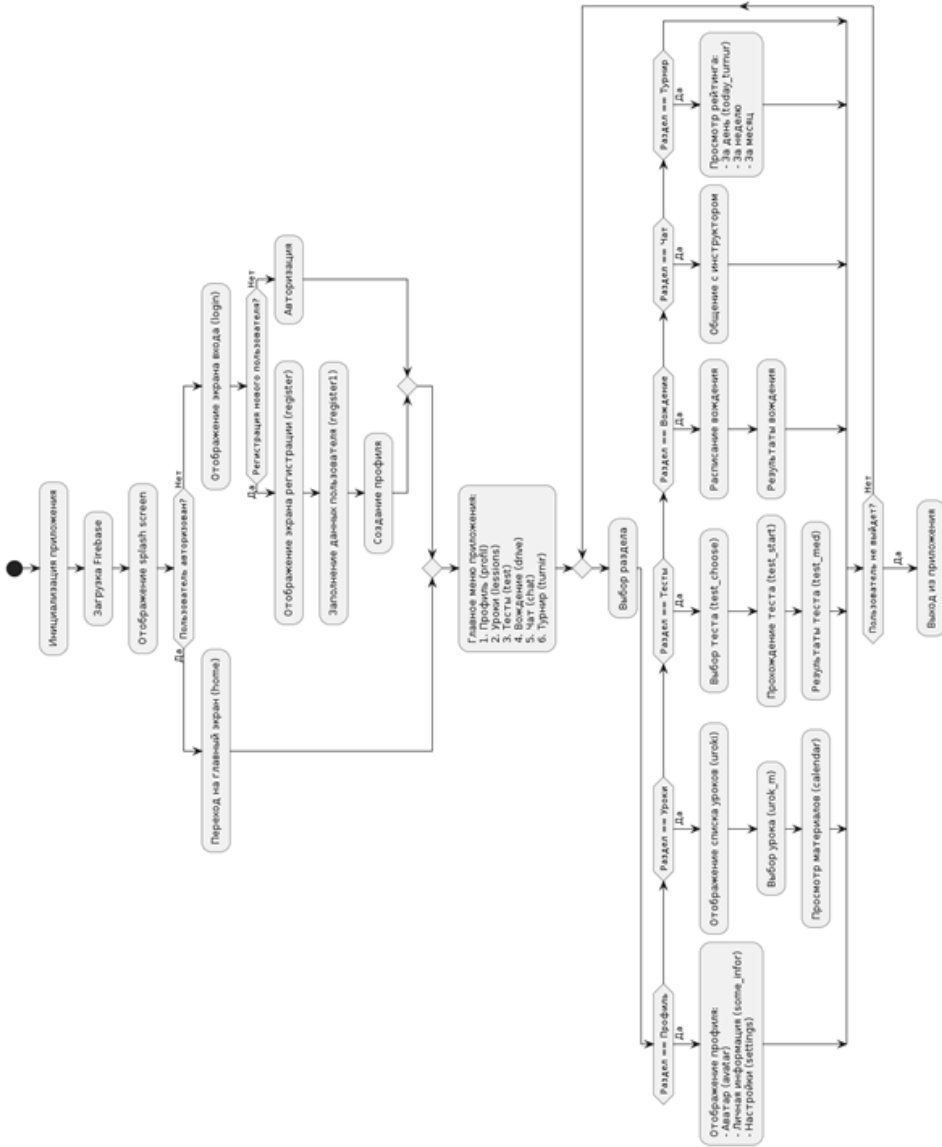
Изм.	Лист	№ докум.	Подпись	Дата



					ПГУ 1.090301.1.005 Д13		
Вариант	Экз.	Результат	Подпись	Дата	Итого: _____		
Решено	_____	_____	_____	_____	_____		
Получено	_____	_____	_____	_____	_____		
Всего	_____	_____	_____	_____	_____		
					Итого: _____		

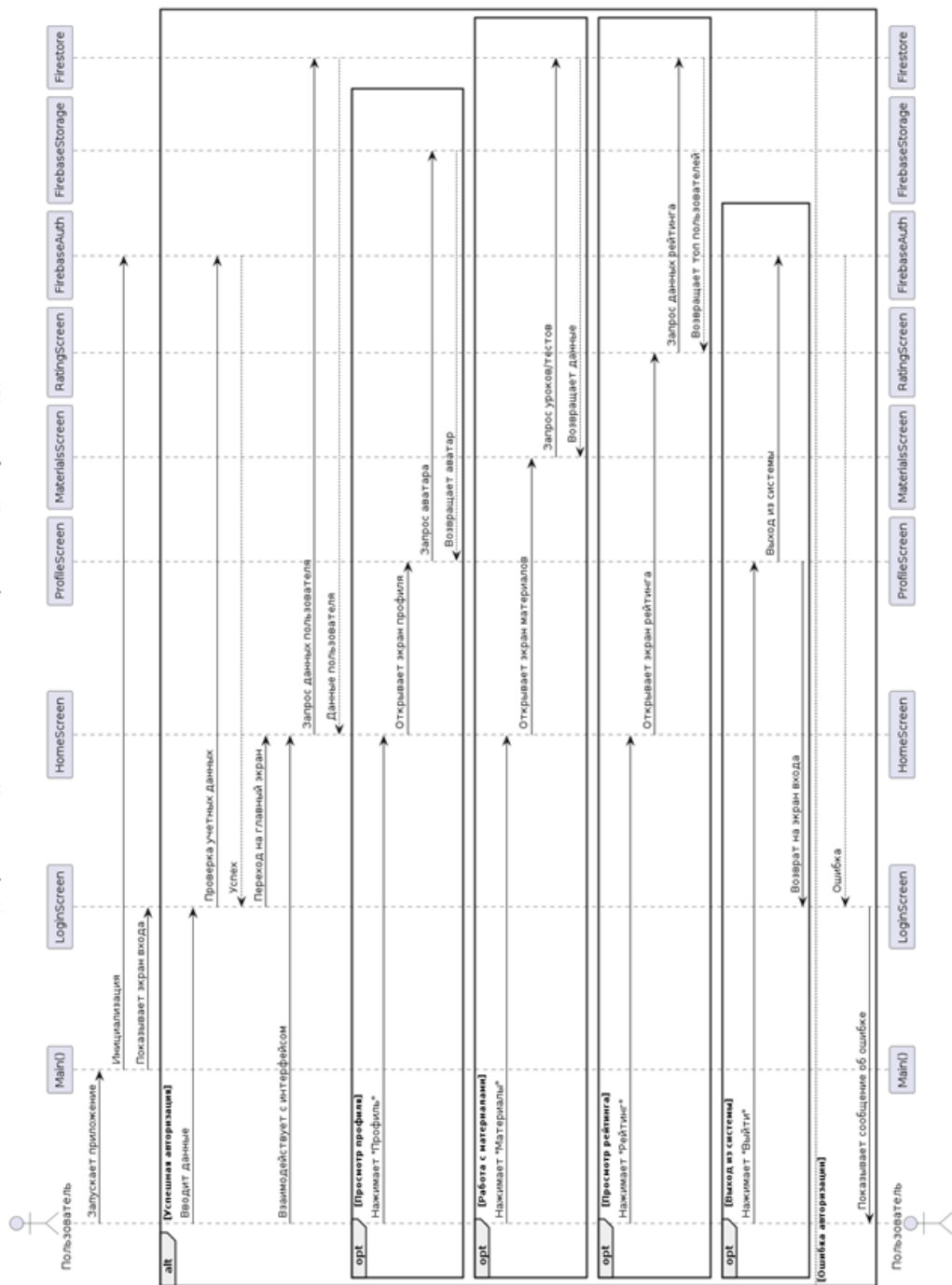
Изм.	Лист	№ докум.	Подпись	Дата

Диаграмма деятельности мобильного приложения для обучения ПДД



ПГУ 1.090301.1.005 Д17									
№ п/п	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
1	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
2	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
3	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
4	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
5	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
6	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
7	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
8	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
9	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата
10	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата	И.И.О.	Подпись	Дата

Диаграмма последовательности мобильного приложения для обучения ПДД



ПУ 1.090301.1.005 Д14									
Имя	Фамилия	Степень	Должность	Подпись	Место	Дата	Подпись руководителя		
Иван	Иванов	Инженер	Инженер	Иванов	Иванов	Иванов	Иванов		
Иван	Иванов	Инженер	Инженер	Иванов	Иванов	Иванов	Иванов		
Иван	Иванов	Инженер	Инженер	Иванов	Иванов	Иванов	Иванов		
Иван	Иванов	Инженер	Инженер	Иванов	Иванов	Иванов	Иванов		
Иван	Иванов	Инженер	Инженер	Иванов	Иванов	Иванов	Иванов		
Иван	Иванов	Инженер	Инженер	Иванов	Иванов	Иванов	Иванов		
Иван	Иванов	Инженер	Инженер	Иванов	Иванов	Иванов	Иванов		
Иван	Иванов	Инженер	Инженер	Иванов	Иванов	Иванов	Иванов		

Приложение Б

(обязательное)

Мобильное приложение «Онлайн–автошкола» на базе ОС Android

Листинг-программы

					ПГУ 1.090301.1.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

main.dart

```
import 'package:diplom/admin/a_home.dart';
import 'package:diplom/load/load.dart';
import 'package:diplom/pages/chat.dart';
import 'package:diplom/pages/home.dart';
import 'package:diplom/Auth/login.dart';
import 'package:diplom/profil/avatar.dart';
import 'package:diplom/profil/profil.dart';
import 'package:diplom/Auth/register.dart';
import 'package:diplom/Auth/register1.dart';
import 'package:diplom/study/drive/drive.dart';
import 'package:diplom/study/lec/lessons.dart';
import 'package:diplom/study/lec/urok/calendar.dart';
import 'package:diplom/study/lec/urok/urok_m.dart';
import 'package:diplom/study/lec/urok/uroki.dart';
import 'package:diplom/study/test/test.dart';
import 'package:diplom/study/test/test_choos.dart';
import 'package:diplom/study/test/test_med.dart';
import 'package:diplom/study/test/test_start.dart';
import 'package:flutter/material.dart';
import
'package:flutter_native_splash/flutter_native_splash.dart';

import 'package:firebase_core/firebase_core.dart';
import 'dart:io';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Future.delayed(const Duration(seconds: 5));
  FlutterNativeSplash.remove();

  Platform.isAndroid
    ? await Firebase.initializeApp(
        options: const FirebaseOptions(
          apiKey: 'AIzaSyAF1hBdle3lPR02NikCTKLbVVU9d27rrrk',
```

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		58

```

        appId:
'1:996321945027:android:63fe27b0ea8f74d6524976',
        messagingSenderId: '996321945027',

        projectId: 'diplom-9afb1',
        storageBucket: 'diplom-9afb1.appspot.com',
    ))
    : await Firebase.initializeApp();

runApp(MaterialApp(
  debugShowCheckedModeBanner: false,
  theme: ThemeData(primaryColor: Colors.amber),
  initialRoute: '/',
  routes: {
    '/': (context) => login(), //Начальный экран авторизации
    '/todo': (context) => register(),
    '/todo1': (context) => register1(), //Права картинка
регистрации
    '/home': (context) => home(), //Нижный меню
    '/profil': (context) => profil(),
    '/load': (context) => load(),
    '/lessions': (context) => lessions(),
    '/avatar': (context) => avatar(),
    '/Chat': (context) => Chat(),
    '/uroki': (context) => uroki(),
    '/urok_m': (context) => urok_m(),
    '/calendar': (context) => Calendar(),
    '/test': (context) => Test(),
    '/Test_choose': (context) => Test_choose(),
    '/Test_strat': (context) => Test_strat(),
    '/Test_med': (context) => Test_med(),
    '/Drive': (context) => Drive(),
  },
));
}

pages.dart

import 'package:diplom/pages/stories.dart';

```

```

import 'package:flutter/material.dart';
import 'package:firebase_storage/firebase_storage.dart';
import
'package:cached_network_image/cached_network_image.dart';
import 'dart:async';

class main1 extends StatelessWidget {

  const main1({super.key});
  @override
  Widget build(BuildContext context) {
    return const Scaffold(
      backgroundColor: Color.fromARGB(250, 111, 114, 187),
      body: SafeArea(
        child: main_test(),
      ),
    );
  }
}

//Контенты для задания, которые находится первым
class main_test extends StatelessWidget {
  const main_test({Key? key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      //backgroundColor: Color.fromARGB(250, 111, 114, 187),
      backgroundColor: Color.fromARGB(250, 111, 114, 187),
      body: SafeArea(
        child: ListView.builder(
          scrollDirection: Axis.vertical,
          itemCount: 1,
          itemBuilder: (BuildContext context, int index) {
            return Column(
              children: [
                Stories(),
                Content(),
              ],
            );
          },
        ),
      ),
    );
  }
}

```

```

        Process(),
        Award(),
        SizedBox(
            height: 20,
        ),
    ],
);
},
),

),
);
}
}

//-----
//-----История новостей-----
//-----
//Индекс картинок после нажатии
class ii {
    static int ii1 = 1;
}

//Для сториса. Присваивает и начинает там.
class ppi {
    static int pp1 = 1;
}

class Stories extends StatefulWidget {
    const Stories({super.key});

    @override
    State<Stories> createState() => _StoriesState();
}

class _StoriesState extends State<Stories> {
    int pageCount = 3;

```

```

@override
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.all(8.0),
    child: FutureBuilder(
      future: loadImagesFromFirebase(),
      builder: (context, snapshot) {
        if (snapshot.connectionState ==
ConnectionState.waiting) {
          return CircularProgressIndicator();
        } else if (snapshot.hasError) {
          return Text('Error: ${snapshot.error}');
        } else {

          List<Map<String, dynamic>> imageUrlsWithIndex =
            snapshot.data as List<Map<String, dynamic>>;
          return Container(
            height: 75,
            child: ListView.builder(
              scrollDirection: Axis.horizontal,
              itemCount: imageUrlsWithIndex.length,
              itemBuilder: (context, index) {
                return _buildContainer(
                  imageUrlsWithIndex[index]['url'],
                  imageUrlsWithIndex[index]['index'],
                );
              },
            ),
          );
        }
      },
    ),
  );
}

Future<List<Map<String, dynamic>>> loadImagesFromFirebase()
async {

```

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

```

List<Map<String, dynamic>> imageUrlsWithIndex = [];

var storage = FirebaseStorage.instance;
var listResult = await storage.ref('News/').listAll();

for (int i = 0; i < listResult.items.length; i++) {
  var item = listResult.items[i];
  var url = await item.getDownloadURL();
  imageUrlsWithIndex.add({'url': url, 'index': i});
}

return imageUrlsWithIndex;
}

Widget _buildContainer(String imageUrl, int index) {
  return Container(
    margin: EdgeInsets.symmetric(horizontal: 3.0),

    width: 75,
    height: 75,
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(100),
      gradient: const LinearGradient(
        begin: Alignment.topRight,
        end: Alignment.bottomRight,
        colors: [
          Color.fromARGB(171, 242, 9, 9),
          Color.fromARGB(254, 0, 4, 255),
        ],
      ),
    ),
    child: Center(
      child: Container(
        width: 68,
        height: 68,
        decoration: BoxDecoration(

```

```

        borderRadius: BorderRadius.circular(100),
        color: Colors.black,
    ),
    child: ElevatedButton(
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) =>
GeneratedPage(pageCount),
          ),
        );

        print(index);
        ii.iil = index;
        ppi.pp1 = ii.iil + 1;
      },
      style: ElevatedButton.styleFrom(
        padding: EdgeInsets.zero,
      ),
      child: ConstrainedBox(

        constraints: BoxConstraints.tightFor(width: 90,
height: 90),

        child: ClipRRect(
          borderRadius: BorderRadius.circular(100.0),
          child: CachedNetworkImage(
            imageUrl: imageUrl,
            fit: BoxFit.cover,
            placeholder: (context, url) =>
CircularProgressIndicator(),
            errorWidget: (context, url, error) =>
Icon(Icons.error),
          ),
        ),
      ),
    ),
  ),

```

```

        ),
        ),
        ),
    );
}
}

//-----
//-----Тест контент-----
//-----

class Content extends StatefulWidget {
    const Content({Key? key}) : super(key: key);

    @override
    _ContentState createState() => _ContentState();
}

class _ContentState extends State<Content> {
    late PageController _pageController;
    final int _totalPages = 4;
    int _currentPage = 0;

    late Timer _timer;

    @override

    void initState() {
        super.initState();
        _pageController = PageController(
            viewportFraction: 0.9,
        )..addListener(() {
            setState(() {
                _currentPage = _pageController.page!.round();
            });
        });
    }
}

```

```

        _timer = Timer.periodic(Duration(seconds: 4), (Timer
timer) {
    _currentPage = (_currentPage + 1) % _totalPages;
    _pageController.animateToPage(
        _currentPage,
        duration: Duration(seconds: 1),
        curve: Curves.easeInOut,
    );
});
}

@override
Widget build(BuildContext context) {
    return Column(
        children: [
            SizedBox(
                height: MediaQuery.of(context).size.height * 0.24,
                child: PageView.builder(
                    controller: _pageController,
                    scrollDirection: Axis.horizontal,
                    itemCount: _totalPages,
                    itemBuilder: (context, index) {
                        return buildContainer(index);
                    },
                ),
            ),
            SizedBox(height: 10),
            buildIndicatorRow(),
        ],
    );
}

Widget buildContainer(int index) {
    List<List<Color>> containerColors = [
        [
            Color.fromARGB(72, 211, 31, 31),

```

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		66

```

        Color.fromARGB(150, 8, 16, 23),

],
[
    Color.fromARGB(179, 29, 144, 47),
    Color.fromARGB(80, 18, 17, 32),
],
[
    Color.fromARGB(146, 10, 84, 91),
    Color.fromARGB(141, 27, 28, 29),
],
[
    Color.fromARGB(111, 197, 150, 22),
    Color.fromARGB(133, 165, 22, 115),
],
];

return Container(
    margin: const EdgeInsets.symmetric(horizontal: 10),
    decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(50),
        gradient: LinearGradient(
            begin: Alignment.topLeft,
            end: Alignment.bottomRight,
            colors: containerColors[index % _totalPages],
        ),
    ),
    child: ElevatedButton(
        onPressed: () {},
        style: ElevatedButton.styleFrom(
            fixedSize: Size(370, 200),
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(50.0),
            ),

            backgroundColor: Colors.transparent,
            shadowColor: Colors.transparent,
        ),
    ),

```

```

child: Column(
  children: [

    Row(
      children: [
        Container(
          margin: EdgeInsets.only(top: 20),
          width: 150,
          child: const Text(
            "Правила дорожного движение",
            style: TextStyle(fontSize: 20, color:
Colors.white),
          ),
        ),
        Image.asset(
          "assets/img/menu_img/home_work1.png",
          width: MediaQuery.of(context).size.width *
0.27,
          fit: BoxFit.cover,
        ),
      ],
    ),
    const SizedBox(
      height: 15,
    ),
    const Row(
      mainAxisAlignment:
MainAxisAlignment.spaceAround,
      children: [
        Column(
          children: [
            Text("Время обучение:",
              style: TextStyle(
                fontSize: 15,
                color: Colors.white,
              )),
            Text("2 часа",
              style: TextStyle(

```

```

        fontSize: 13,

        color: Colors.white,
    )),
    ],
),
Column(
    children: [
        Text("Уровень:",
            style: TextStyle(
                fontSize: 15,
                color: Colors.white,
            )),
        Text("Начальная",
            style: TextStyle(
                fontSize: 13,
                color: Colors.white,
            )),
    ],
),
],
),
],
),
],
),
),
);
}

Widget buildIndicatorRow() {
    return Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: List.generate(
            _totalPages,
            (index) => buildIndicator(index),
        ),
    );
}

```

```

    }

Widget buildIndicator(int index) {
    return Container(

        width: MediaQuery.of(context).size.width * 0.02,
        height: MediaQuery.of(context).size.height * 0.02,
        margin: EdgeInsets.symmetric(horizontal: 5),
        decoration: BoxDecoration(
            shape: BoxShape.circle,
            color: index == _currentPage
                ? const Color.fromARGB(
                    255, 255, 255, 255) // Цвет активного
индикатора
                : Colors.grey.withOpacity(0.6), // Цвет
неактивного индикатора
        ),
    );
}

@override
void dispose() {
    _pageController.dispose();
    _timer.cancel();
    super.dispose();
}
}

//-----
//-----График-----
//-----

class Process extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return ActivityHistogram();
    }
}

```

```

class ActivityHistogram extends StatelessWidget {
  final Map<String, double> activityValues = {
    'Пн': 3,
    'Вт': 5,
    'Ср': 9,

    'Чт': 12,
    'Пт': 16,
    'Сб': 27,

    'Вс': 10,
  };

  final double maxHeight = 200.0; // Максимальная
  высота контейнера

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(20.0),
      child: Container(
        decoration: BoxDecoration(
          gradient: const LinearGradient(
            begin: Alignment.topRight,
            end: Alignment.bottomRight,
            colors: [
              Color.fromARGB(82, 242, 9, 9),
              Color.fromARGB(91, 0, 4, 255),
            ],
          ),
        borderRadius: BorderRadius.circular(50),
      ),
      padding: EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          const Center(
            child: Text(

```



```

    }

    Widget _buildDayBar(String day, double heightPercentage,
    Color color) {
        double normalizedHeight = heightPercentage / 100;
        final double containerHeight = normalizedHeight *
            maxContainerHeight; // Нормализация и установка
            МАКСИМАЛЬНОЙ ВЫСОТЫ

        return Column(
            children: [
                Text(
                    '${(heightPercentage).toInt()}',
                    style: TextStyle(fontWeight: FontWeight.bold, color:
Colors.white),

                ),
                Container(
                    width: 15.0,
                    height: containerHeight,
                    decoration: BoxDecoration(
                        color: color,
                        borderRadius: BorderRadius.circular(20),
                    ),
                ),
                SizedBox(height: 8.0),
                Text(
                    day,
                    style: TextStyle(
                        fontWeight: FontWeight.bold,
                        color: Colors.white,
                        fontSize: 17,
                    ),
                ),
            ],
        );
    }

```

```

Widget _buildDayLabel(String day) {
  return Text(
    day,
    style: TextStyle(fontWeight: FontWeight.bold),
  );
}

//-----
//-----Награда-----
//-----

class Award extends StatelessWidget {
  const Award({super.key});

  @override
  Widget build(BuildContext context) {

    final double screenWidth =
MediaQuery.of(context).size.width;
    final double screenHeight =
MediaQuery.of(context).size.height;

    return Column(
      children: [
        Padding(
          padding: const EdgeInsets.all(15.0),
          child: Container(
            width: screenWidth,
            height: screenHeight * 0.4,
            decoration: BoxDecoration(
              gradient: const LinearGradient(
                begin: Alignment.topRight,
                end: Alignment.bottomRight,
                colors: [
                  Color.fromARGB(91, 0, 4, 255),

```

```

        Color.fromARGB(82, 242, 9, 9)
      ],
    ),
    borderRadius: BorderRadius.circular(50),
  ),
  child: Column(
    children: [
      const Padding(
        padding: EdgeInsets.all(8.0),
        child: Text(
          "Задания на сегодня",
          style: TextStyle(
            color: Colors.white,
            fontSize: 20,
            fontWeight: FontWeight.bold),
        ),
      ),
      Row(
        children: [
          _buildContainer(context, "Задания",
screenWidth * 0.38),
          _buildContainer(context, "Выполнено",
screenWidth * 0.25),
          _buildContainer(context, "Награда",
screenWidth * 0.2),
        ],
      ),
      _buildTaskRow(context, "Пройти 2 новые
лекции", "0 | 2", "15",
Icons.star_outline),
      Container(
        decoration: BoxDecoration(
          border: Border(
            top: BorderSide(width: 1, color:
Colors.black),
            bottom: BorderSide(width: 1))),

```

```

        child: _buildTaskRow(context, "Пройти 3
теста подряд",
        "3 | 3", "20", Icons.star),
      ),
      _buildTaskRow(context, "Поиграть и выиграть 5
раз в играх",
        "0 | 5", "30", Icons.star_outline),
    ],
  ),
),
)
],
);
}

```

```

Widget _buildContainer(BuildContext context, String text,
double width) {
  return Container(
    margin: EdgeInsets.only(left: 10),
    width: width,
    height: MediaQuery.of(context).size.height * 0.05,
    decoration: BoxDecoration(
      gradient: const LinearGradient(
        begin: Alignment.topRight,
        end: Alignment.bottomRight,
        colors: [

          Color.fromARGB(47, 0, 187, 255),
          Color.fromARGB(56, 32, 242, 9)
        ],
      ),
      borderRadius: BorderRadius.circular(50),
    ),
    child: Center(
      child: Text(
        text,
        style: const TextStyle(

```

```

        color: Colors.white, fontSize: 15, fontWeight:
FontWeight.bold),
    ),
),
);
}

Widget _buildTaskRow(BuildContext context, String task,
String progress,
    String reward, IconData icon) {
return Padding(
padding: const EdgeInsets.only(top: 10),
child: Row(
children: [
Container(
width: MediaQuery.of(context).size.width * 0.4,
height: MediaQuery.of(context).size.height * 0.05,
margin: const EdgeInsets.only(left: 10),
child: Center(
child: Text(
task,
textAlign: TextAlign.center,
style: const TextStyle(
color: Colors.white,
fontSize: 15,
fontWeight: FontWeight.bold),
),
),
),
Container(
margin: const EdgeInsets.only(left: 5),
width: MediaQuery.of(context).size.width * 0.25,
height: MediaQuery.of(context).size.height * 0.05,
child: Center(
child: Text(
progress,

```

```

        textAlign: TextAlign.center,
        style: const TextStyle(
            color: Colors.white,
            fontSize: 15,
            fontWeight: FontWeight.bold),
    ),
),
),
Container(
    margin: const EdgeInsets.only(left: 5),
    width: MediaQuery.of(context).size.width * 0.2,
    height: MediaQuery.of(context).size.height * 0.05,
    child: Center(
        child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
                Text(
                    reward,
                    textAlign: TextAlign.center,
                    style: const TextStyle(
                        color: Colors.white,
                        fontSize: 15,
                        fontWeight: FontWeight.bold),
                ),
                const SizedBox(width: 5),
                Icon(icon, color: Colors.white),
            ],
        ),
    ),
),
],
),
);
}
}

```

study/drive.dart

```
import 'package:diplom/study/drive/drive.dart';
import 'package:diplom/study/lec/urok/urok_m.dart';
import 'package:diplom/study/test/test.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

class lek2 extends StatefulWidget {
  const lek2({super.key});

  @override
  State<lek2> createState() => _lek2State();
}

class _lek2State extends State<lek2> {
  int lives = 0;

  @override
  void initState() {
    super.initState();
    loadlives();
  }

  Future<void> loadlives() async {
    int userlives = await getUserlives();
    setState(() {
      lives = userlives;
    });
  }

  Future<int> getUserlives() async {
    DocumentSnapshot<Map<String, dynamic>> userDoc =
      await FirebaseFirestore.instance
        .collection('Users')

        .doc('19x87sZVIref3vggPCgawBuahRA2') //uui.uuid
    для других студентов
```

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		79

```

        .get();

        return userDoc.data()?['lives'] ?? 0;
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            backgroundColor: Color.fromARGB(250, 111, 114, 187),
            body: SafeArea(
                child: ListView(
                    children: [
                        Column(
                            children: [
                                Padding(
                                    padding: const EdgeInsets.only(right: 10, top:
5),

                                    child: Row(
                                        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                                        children: [
                                            IconButton(
                                                onPressed: () {
                                                    Navigator.pop(context);
                                                },
                                                icon: const Icon(
                                                    Icons.arrow_back_ios_new_rounded,
                                                    color: Colors.white,
                                                )),
                                            const Text(
                                                "Категория В",
                                                style: TextStyle(
                                                    fontSize: 22,
                                                    color: Colors.white,
                                                    fontWeight: FontWeight.bold),
                                            ),
                                            Container(
                                                width: 60,

```

```

        height: 40,

        child: Padding(
          padding: EdgeInsets.all(6.0),
          child: Row(
            mainAxisAlignment:
MainAxisAlignment.spaceAround,
            children: [
              Text(
                lives.toString(),
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 20,
                  fontWeight:
FontWeight.bold),
              ),
              Icon(
                Icons.favorite,
                color: Colors.red,
              ),
            ],
          ),
        ),
      ),
    ),
  ],
),
Progres(),
Progres1(),
const main_lesson(),
l_test(),
l_drive(),
const SizedBox(
  height: 50,
)
],
),

```

```

        ],
      )),
    );
  }
}

//-----
//-----Уведомление важно-----
//-----

class Progres extends StatefulWidget {
  const Progres({super.key});

  @override
  State<Progres> createState() => _ProgresState();
}

bool invis = false;

class _ProgresState extends State<Progres> {
  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Container(
          width: MediaQuery.of(context).size.width * 0.5,
          height: MediaQuery.of(context).size.height * 0.05,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(30), color:
Colors.amber),
          child: ElevatedButton(
            onPressed: () {
              setState(() {
                if (invis == true)
                  invis = false;
                else
                  invis = true;
              });
            },
          ),
        ),
      ],
    );
  }
}

```

```

        },
        child: Row(
          mainAxisAlignment:
MainAxisAlignment.spaceBetween,
          children: [
            Icon(
              Icons.warning_amber_rounded,
              color: Colors.red,
              size: 30,

            ),
            Text(
              "Важно",
              style: TextStyle(fontSize: 16, fontWeight:
FontWeight.bold),
            ),
            Icon(
              invis == true
                ? Icons.keyboard_arrow_up_rounded
                : Icons.keyboard_arrow_down_rounded,
            )
          ],
        ),
      ),
    ),
    if (invis == true)
      Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(20),
          gradient: const LinearGradient(
            begin: Alignment.topLeft,
            end: Alignment.bottomRight,
            colors: [
              Color.fromARGB(128, 40, 101, 223),
              Color.fromARGB(187, 22, 22, 22),
            ],
          ),
        ),

```

```

    ),
    width: MediaQuery.of(context).size.width * 0.95,
    height: MediaQuery.of(context).size.height * 0.2,
    child: const Padding(
      padding: const EdgeInsets.all(8.0),
      child: Text(
        "Обучение на получение права длится на
определенное время и нужно войти в урок по расписанию, чтобы не
пропустить уроки (за пропущенные уроки будете проходите
дополнительные тесты). Если вы За определенный срок не сдадите все
части то все прогрессы будут потеряны.",
        style:
          TextStyle(fontWeight: FontWeight.bold,
color: Colors.white),
      ),
    ),
  ),
],
);
}
}

//-----
//-----Прогресс-----
//-----

class Progres1 extends StatelessWidget {
  const Progres1({super.key});

  @override
  Widget build(BuildContext context) {
    return Container(
      margin: EdgeInsets.only(top: 10),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(20),
        gradient: const LinearGradient(
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,

```

```

        colors: [
            Color.fromARGB(128, 40, 180, 223),
            Color.fromARGB(80, 238, 239, 241),
        ],
    ),
),
width: MediaQuery.of(context).size.width * 0.9,
height: MediaQuery.of(context).size.height * 0.2,
child: const Column(
    children: [
        Text(
            "Прорпеcc",
            style: TextStyle(
                color: Colors.white,
                fontSize: 24,
                fontWeight: FontWeight.bold),
        ),
        Text(
            "Осталось: 1 месяц 20 дней",

            style: TextStyle(
                color: Colors.white,
                fontSize: 18,
                fontWeight: FontWeight.bold),
        ),
        Row(
            mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
            children: [
                Row(
                    children: [
                        Column(
                            children: [
                                Icon(
                                    Icons.menu_book_rounded,
                                    size: 60,
                                    color: Colors.white,
                                ),

```

```

        Text(
          "10%",
          style: TextStyle(
            color: Colors.white,
            fontSize: 20,
            fontWeight: FontWeight.bold),
        ),
      ],
    ),
  ],
),
Row(
  children: [
    Column(
      children: [
        Icon(
          Icons.pending_actions_outlined,
          size: 60,
          color: Colors.white,
        ),
        Text(
          "9%",
          style: TextStyle(
            color: Colors.white,
            fontSize: 20,
            fontWeight: FontWeight.bold),
        ),
      ],
    ),
  ],
),
Row(
  children: [
    Column(
      children: [
        Icon(
          Icons.car_crash_outlined,
          size: 60,

```

```

        color: Colors.white,
      ),
      Text(
        "18%",
        style: TextStyle(
          color: Colors.white,
          fontSize: 20,
          fontWeight: FontWeight.bold),
      )
    ],
  ),
],
),
],
),
],
));
}

```

```

}

```

```

//-----

```

```

//-----Уроки-----

```

```

//-----

```

```

class main_lesson extends StatelessWidget {

```

```

  const main_lesson({super.key});

```

```

  @override

```

```

  Widget build(BuildContext context) {

```

```

    return Container(

```

```

      margin: EdgeInsets.only(top: 20),

```

```

      width: MediaQuery.of(context).size.width * 0.9,

```

```

      height: MediaQuery.of(context).size.height * 0.2,

```

```

      decoration: BoxDecoration(

```

```

        borderRadius: BorderRadius.circular(30),

```

```

        gradient: const LinearGradient(

```

```

          begin: Alignment.topLeft,

```

```

end: Alignment.bottomRight,
colors: [
    Color.fromARGB(128, 199, 100, 100),
    Color.fromARGB(86, 49, 59, 240),
],
),
),
child: Row(
    children: [
        const Padding(
            padding: const EdgeInsets.all(20.0),
            child: Icon(
                Icons.menu_book_rounded,
                size: 80,
                color: Colors.white,
            ),
        ),
    ],
    Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
            Container(

                margin: const EdgeInsets.only(top: 5),
                child: const Text(
                    "Уроки",
                    style: TextStyle(
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                        color: Colors.white),
                ),
            ),
            SizedBox(
                width: MediaQuery.of(context).size.width *
0.56,

                child: Center(
                    child: const Text(

```

```

        "Основные материалы: теории, дорожные
правила, дорожные знаки и.т.д.",
        style: TextStyle(color: Colors.white),
    ),
),
),
ElevatedButton(
    onPressed: () {
        Navigator.push(context,
            MaterialPageRoute(builder: (context) =>
urok_m()));
    },
    style: ElevatedButton.styleFrom(
        backgroundColor: Color.fromARGB(250, 111,
114, 187),
    ),
    child: const Text(
        "Пройти",
        style: TextStyle(color: Colors.white),
    ),
),
],
),
],
),
);

}
}

//-----
//-----Тесты-----
//-----

class l_test extends StatelessWidget {
    const l_test({super.key});

    @override
    Widget build(BuildContext context) {

```

```

return Container(
  margin: EdgeInsets.only(top: 20),
  width: MediaQuery.of(context).size.width * 0.9,
  height: MediaQuery.of(context).size.height * 0.17,
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(30),
    gradient: const LinearGradient(
      begin: Alignment.topLeft,
      end: Alignment.bottomRight,
      colors: [
        Color.fromARGB(128, 152, 41, 226),
        Color.fromARGB(87, 9, 21, 253),
      ],
    ),
  ),
  child: Row(
    children: [
      const Padding(
        padding: const EdgeInsets.all(20.0),
        child: Icon(
          Icons.pending_actions_outlined,
          size: 80,
          color: Colors.white,
        ),
      ),
      Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [

          Container(
            margin: const EdgeInsets.only(top: 5),
            child: const Text(
              "Тесты",
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
                color: Colors.white),
            ),
          ),

```

```

        ),
    ),
    SizedBox(
      width: MediaQuery.of(context).size.width *
0.56,

      child: const Text(
        "Тесты по урокам и практическим урокам",
        style: TextStyle(color: Colors.white),
      ),
    ),
    ElevatedButton(
      onPressed: () {
        Navigator.push(
          context, MaterialPageRoute(builder:
(context) => Test()));
      },
      style: ElevatedButton.styleFrom(
        backgroundColor: Color.fromARGB(250, 111,
114, 187),

      ),
      child: const Text(
        "Пройти",
        style: TextStyle(color: Colors.white),
      ),
    ),
  ],
),
],
),
);
}
}

```

//-----

//-----Вожделение-----

//-----

```
class l_drive extends StatelessWidget {
```

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		91

```

const l_drive({super.key});

@override
Widget build(BuildContext context) {
  return Container(
    margin: EdgeInsets.only(top: 20),
    width: MediaQuery.of(context).size.width * 0.9,
    height: MediaQuery.of(context).size.height * 0.17,
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(30),
      gradient: const LinearGradient(
        begin: Alignment.topLeft,
        end: Alignment.bottomRight,
        colors: [
          Color.fromARGB(128, 41, 189, 226),
          Color.fromARGB(87, 9, 21, 253),
        ],
      ),
    ),
    child: Row(
      children: [
        const Padding(
          padding: const EdgeInsets.all(20.0),
          child: Icon(
            Icons.car_crash_outlined,
            size: 80,
            color: Colors.white,
          ),
        ),
        Column(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            Container(
              margin: const EdgeInsets.only(top: 5),
              child: const Text(
                "Вожделение",

```

```

        style: TextStyle(
          fontSize: 20,
          fontWeight: FontWeight.bold,
          color: Colors.white),
      ),
    ),
    SizedBox(
      width: MediaQuery.of(context).size.width *
0.56,

      child: const Text(
        "Можете посмотреть расписание по вождению и
свой результат",

        style: TextStyle(color: Colors.white),
      ),
    ),
    ElevatedButton(
      onPressed: () {
        Navigator.push(context,
          MaterialPageRoute(builder: (context) =>
Drive())));

      },
      style: ElevatedButton.styleFrom(
        backgroundColor: Color.fromARGB(250, 111,
114, 187),

      ),
      child: const Text(
        "Посмотреть",
        style: TextStyle(color: Colors.white),
      ),
    ),
  ],
),
],
),
);
}
}

```

profil.dart

```
import 'package:diplom/load/load.dart';
import 'package:diplom/profil/avatar.dart';
import 'package:diplom/profil/settings.dart';
import 'package:diplom/profil/some_infor.dart';
import 'package:flutter/material.dart';

class profil extends StatefulWidget {
  const profil({super.key});

  @override
  State<profil> createState() => _profilState();
}

class _profilState extends State<profil> {
  String ava_prof = "";
  void initState() {
    super.initState();
    ava_prof = avatar_1.avatar_1l;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Color.fromARGB(250, 111, 114, 187),
      body: SafeArea(
        child: ListView(children: [
          const Center(
            child: Text(
              "Профиль",
              style: TextStyle(
                fontSize: 22, color: Colors.white, fontWeight:
FontWeight.bold),
            ),
          ),
        ]),
      ),
    ),
  ),
}
```

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		94

```

//Картинка аватар
Container(
  margin: const EdgeInsets.only(top: 20, left: 10),
  child: Transform.scale(

    scale: 1.3,
    child: Center(
      child: Container(
        width: MediaQuery.of(context).size.width * 0.5,
        height: MediaQuery.of(context).size.height *
0.25,

        child: Image.network(
          ava_prof,
          fit: BoxFit.cover,
        ),
      )),
    ),
  ),
// Кнопка изменить картинку
Row(
  mainAxisAlignment: MainAxisAlignment.end,
  children: [
    Container(
      margin: EdgeInsets.only(top: 30),
      width: 140.0,
      height: 34.0,
      child: TextButton(
        onPressed: () {
          Navigator.push(context,
            MaterialPageRoute(builder: (context) =>
avatar())));

        },
        style: TextButton.styleFrom(
          backgroundColor: Colors.transparent, // Цвет
фона кнопки

        ),
        child: const Text(

```

```

        'Изменить аватар', // Текст кнопки
        style: TextStyle(color: Color.fromARGB(255,
225, 225, 225)),
    ),
),
),
],
),

// Имя пользователя
Center(
  child: Text(
    student.name_student,
    style: TextStyle(
      color: Colors.white, fontSize: 30, fontWeight:
FontWeight.bold),
  ),
),
const SizedBox(
  height: 10,
),
// Кнопки
Container(
  alignment: Alignment.center,
  child: Column(
    children: [
      SizedBox(
        width: 250,
        height: 50,
        child: ElevatedButton(
          onPressed: () {
            print(Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) =>
some_infor())));
          },

```

```

        child: const Text(
          "Личная информация",
          style:
            TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
        )),
      ),
      const SizedBox(
        height: 10,
      ),
      SizedBox(
        width: 250,

        height: 50,
        child: ElevatedButton(
          onPressed: () {},
          child: const Text(
            "Безопасность",
            style:
              TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
          )),
      ),
      const SizedBox(
        height: 10,
      ),
      SizedBox(
        width: 250,
        height: 50,
        child: ElevatedButton(
          onPressed: () {
            Navigator.push(context,
              MaterialPageRoute(builder: (context)
=> settings()));
          },
          child: const Text(
            "Настройка",

```

```

        style:
            TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
        )),
    ),
    const SizedBox(
        height: 10,
    ),
    SizedBox(
        width: 250,
        height: 50,
        child: ElevatedButton(
            onPressed: () {
                print(student.name_student);
                print(uui.uuid);

            },
            child: const Text(
                "Поддержка",
                style:
                    TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
            )),
    ),
    const SizedBox(
        height: 10,
    ),
    SizedBox(
        width: 250,
        height: 50,
        child: ElevatedButton(
            onPressed: () {},
            child: const Text(
                "Выйти из аккаунта",
                style:
                    TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),

```

```

        )),
    ),
],
),
),
// Подпись
Center(
    child: Column(
        children: [
            Container(
                margin: const EdgeInsets.only(top: 40),
                child: const Text(
                    "Версия: 1.0",
                    style: TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.bold,
                        color: Colors.white),
                ),
            ),
        ],
    ),
    Container(
        margin: const EdgeInsets.only(top: 0),
        child: const Text(
            "Автор: Талантбек уулу Олжобай",
            style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
                color: Colors.white),
        ),
    ),
],
),
)
]))),
);
}
}

```

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		99

turnir.dart

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

class turnir extends StatefulWidget {
  const turnir({super.key});

  @override
  State<turnir> createState() => _TurnirState();
}

class _TurnirState extends State<turnir> {
  int selectedTab = 0;

  void initState() {
    super.initState();
  }

  @override

  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Color.fromARGB(250, 111, 114, 187),
      body: SafeArea(
        child: ListView(
          scrollDirection: Axis.vertical,
          children: [
            head_turnir(),
            ActivityHistogram(),
            Padding(
              padding: const EdgeInsets.only(bottom: 20, left:
10, right: 10),
              child: Row(
                mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
```

```

children: [
  ElevatedButton(
    style: ElevatedButton.styleFrom(
      backgroundColor: selectedTab == 0
        ? Color.fromARGB(255, 1, 5, 255)
        : Colors.white,
    ),
    onPressed: () {
      setState(() {
        selectedTab = 0;
      });
    },
    child: Text(
      "Сегодня",
      style: TextStyle(
        color: selectedTab == 0
          ? Colors.white
          : Color.fromARGB(250, 111, 114,
187),

        fontWeight: FontWeight.bold,
        fontSize: 16,
      ),
    ),
  ),
  ElevatedButton(
    style: ElevatedButton.styleFrom(

      backgroundColor: selectedTab == 1
        ? Color.fromARGB(255, 1, 5, 255)
        : Colors.white,
    ),
    onPressed: () {
      setState(() {
        selectedTab = 1;
      });
    },
    child: Text(

```

```

        "Неделя",
        style: TextStyle(
          color: selectedTab == 1
            ? Colors.white
            : Color.fromARGB(250, 111, 114,
187),

          fontWeight: FontWeight.bold,
          fontSize: 16,
        ),
      ),
    ),
    ElevatedButton(
      style: ElevatedButton.styleFrom(
        backgroundColor: selectedTab == 2
          ? Color.fromARGB(255, 1, 5, 255)
          : Colors.white,
      ),
      onPressed: () {
        setState(() {
          selectedTab = 2;
        });
      },
      child: Text(
        "Месяц",
        style: TextStyle(
          color: selectedTab == 2
            ? Colors.white
            : Color.fromARGB(250, 111, 114,
187),

          fontWeight: FontWeight.bold,
          fontSize: 16,

        ),
      ),
    ),
  ],
),

```

```

        ),
        // Отображение контейнера в зависимости от выбора
        if (selectedTab == 0) ...[
            today_turnur(),
        ] else if (selectedTab == 1) ...[
            today_turnur(),
        ] else if (selectedTab == 2) ...[
            today_turnur(),
        ],
    ],
),
),
);
}
}

////////////////////////////////////
////////////////////////////////////Верхний часть
экрана////////////////////////////////////
////////////////////////////////////

class head_turnir extends StatelessWidget {
  const head_turnir({super.key});

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        IconButton(
          onPressed: () {},
          icon: const Icon(
            Icons.more_vert_rounded,
            color: Colors.white,
            size: 26,
          )),

        const Text(

```

```

        "Турнир",
        style: TextStyle(
            color: Colors.white, fontSize: 22, fontWeight:
FontWeight.bold),
    ),
    const Text("
    ")
  ],
);
}
}

```

```

////////////////////////////////////
////////////////////////////////////Топ контент////////////////////////////////////
////////////////////////////////////

```

```

class ActivityHistogram extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(20.0),
      child: Container(
        height: MediaQuery.of(context).size.height * 0.35,
        decoration: BoxDecoration(
          gradient: const LinearGradient(
            begin: Alignment.topRight,
            end: Alignment.bottomRight,
            colors: [
              Color.fromARGB(82, 242, 9, 9),
              Color.fromARGB(91, 0, 0, 0),
            ],
          ),
          borderRadius: BorderRadius.circular(50)),
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            const Center(
              child: Text(
                'Лучшие за Май',

```

```

        textAlign: TextAlign.center,
        style: TextStyle(
          fontSize: 20.0,
          fontWeight: FontWeight.bold,
          color: Colors.white),
      ),
    ),
    SizedBox(height: 16.0),
    Column(
      mainAxisAlignment:
MainAxisAlignment.spaceAround,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        _buildDayBar(1, 'User1', 110, Colors.blue),
        _buildDayBar(2, 'User2', 93, Colors.green),
        _buildDayBar(3, 'User3', 80, Colors.orange),
        _buildDayBar(4, 'User4', 77, Colors.red),
        _buildDayBar(5, 'User5', 75, Colors.purple),
      ],
    ),
  ],
),
),
);
}

```

```

Widget _buildDayBar(
  int ii, String day, double heightPercentage, Color
color) {
  return Padding(
    padding: const EdgeInsets.all(5.0),
    child: Row(
      children: [
        Row(
          children: [
            Icon(

```

```

Icons.local_fire_department_rounded,
color: Colors.red,
size: 25,
),
Text(

    ii.toString(),
    style: TextStyle(
        fontWeight: FontWeight.bold,
        color: Colors.white,
        fontSize: 15),
    ),
],
),
const SizedBox(
    width: 10,
),
Container(
    width: 1.2 * heightPercentage,
    height: 20,
    decoration: BoxDecoration(
        color: color, borderRadius:
BorderRadius.circular(50)),
    child: Center(
        child: Text(
            '${(heightPercentage).toInt()}',
            style:
                TextStyle(fontWeight: FontWeight.bold,
color: Colors.white),
        ),
    ),
),
SizedBox(width: 8.0),
Text(
    day,
    style: TextStyle(

```

```

fontWeight: FontWeight.bold, color:
Colors.white, fontSize: 15),
    ),
  ],
),
);
}

Widget _buildDayLabel(String day) {

  return Text(
    day,
    style: TextStyle(fontWeight: FontWeight.bold),
  );
}

}

////////////////////////////////////
////////////////////////////////////Сегодняшний
рейтинг////////////////////////////////////
////////////////////////////////////

class User {
  final String login;
  final int activity;

  User({required this.login, required this.activity});
}

class today_turnur extends StatefulWidget {
  today_turnur({Key? key}) : super(key: key);

  @override
  State<today_turnur> createState() => _MyAppState();
}

int award = 0;

```

```

class _MyAppState extends State<today_turnur> {
  List<User> users = [];

  @override
  void initState() {
    super.initState();
    fetchData();
  }

  Future<void> fetchData() async {
    CollectionReference usersCollection =
      FirebaseFirestore.instance.collection('Users');

    // Получение данных с использованием get()
    QuerySnapshot querySnapshot = await usersCollection.get();
    // Обработка каждого документа
    querySnapshot.docs.forEach((doc) {
      Map<String, dynamic> data = doc.data() as Map<String,
dynamic>;
      users.add(User(login: data['Login'], activity:
data['Activity']));
    });

    // Сортировка по убыванию рейтинга (Activity)
    users.sort((a, b) => b.activity.compareTo(a.activity));

    setState(() {});
  }

  @override
  Widget build(BuildContext context) {
    return Container(
      height: MediaQuery.of(context).size.height * 0.5,
      margin: EdgeInsets.only(left: 15, right: 15, bottom:
10),
      decoration: BoxDecoration(

```

```

        boxShadow: [
            BoxShadow(
                color: Color.fromARGB(255, 0, 0,
0).withOpacity(0.5), // цвет тени
                spreadRadius: 7, // распределение тени по
контур контейнера
                blurRadius: 9, // радиус размытия тени
                offset: Offset(0, 3), // смещение тени по x и y
            ),
        ],
        gradient: const LinearGradient(
            begin: Alignment.topRight,
            end: Alignment.bottomRight,
            colors: [
                Color.fromARGB(82, 242, 9, 9),
                Color.fromARGB(91, 0, 4, 255),
            ],
        ),

        borderRadius: BorderRadius.circular(40)),
child: SingleChildScrollView(
    scrollDirection: Axis.vertical,
    child: DataTable(
        columnSpacing: 30, // Установка расстояния между
столбцами

        columns: const [
            DataColumn(
                label: Text(
                    '№',
                    style: TextStyle(
                        color: Colors.white,
                        fontWeight: FontWeight.bold,
                        fontSize: 16),
                )),
            DataColumn(
                label: Text(
                    'Студенты',

```

```

        style: TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.bold,
            fontSize: 16),
    )),
    DataColumn(
        label: Text(
            'Активность',
            style: TextStyle(
                color: Colors.white,
                fontWeight: FontWeight.bold,
                fontSize: 16),
        )),
    DataColumn(
        label: Text(
            'Награда',
            style: TextStyle(
                color: Colors.white,
                fontWeight: FontWeight.bold,
                fontSize: 16),
        )),
    ],

    rows: List<DataRow>.generate(
        users.length,
        (index) {
            return DataRow(
                cells: [
                    DataCell(Text(
                        (index + 1).toString(),
                        style: const TextStyle(
                            color: Colors.white,
                            fontWeight: FontWeight.bold,
                            fontSize: 14),
                    )),
                    DataCell(Row(
                        children: [

```

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		110

```

        Text(
            users[index].login,
            style: const TextStyle(
                color: Colors.white,
                fontWeight: FontWeight.bold,
                fontSize: 14),
        ),
        if (index < 3)
            const Padding(
                padding: const EdgeInsets.only(left:
5),

                child: Icon(
                    Icons.local_fire_department_rounded,

                    size: 19,
                    color: Colors.red,
                ),
            ),
    ],
)),
DataCell(Center(
    child: Text(
        users[index].activity.toString(),
        style: const TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.bold,
            fontSize: 14),

    ),
)),
DataCell(
    index < 5
        ? Text(
            (50 - ((index) * 10)).toString(),
            style: const TextStyle(
                color: Colors.white,
                fontWeight: FontWeight.bold,

```

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		111

```
fontSize: 14),  
    )  
    : Text(''),  
    ),  
    ],  
    );  
    },  
    ),  
    ),  
    ),  
    );  
    }  
}
```

					ПГУ 1.090301.1.001 ПЗ	Лист
						112
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(обязательное)

Мобильное приложение «Онлайн–автошкола» на базе ОС Android

Презентация

					ПГУ 1.090301.1.001 ПЗ	Лист
						113
Изм.	Лист	№ докум.	Подпись	Дата		

Тема: Мобильное приложения «Онлайн авто-
школа» на базе ОС Android

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		114

Цель проекта

Целью данного проекта является создание мобильного приложения, которое объединит в себе удобство онлайн-обучения с интерактивными уроками, тестами, а также возможностью получения реального обратного отклика от системы. Приложение будет ориентировано как на начинающих водителей, так и на тех, кто желает обновить свои знания и подготовиться к сдаче экзаменов

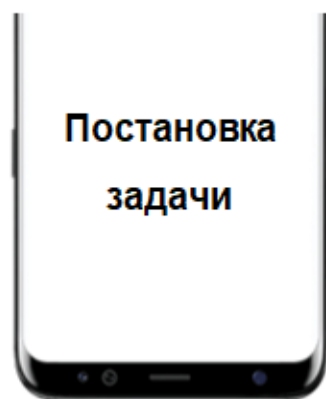
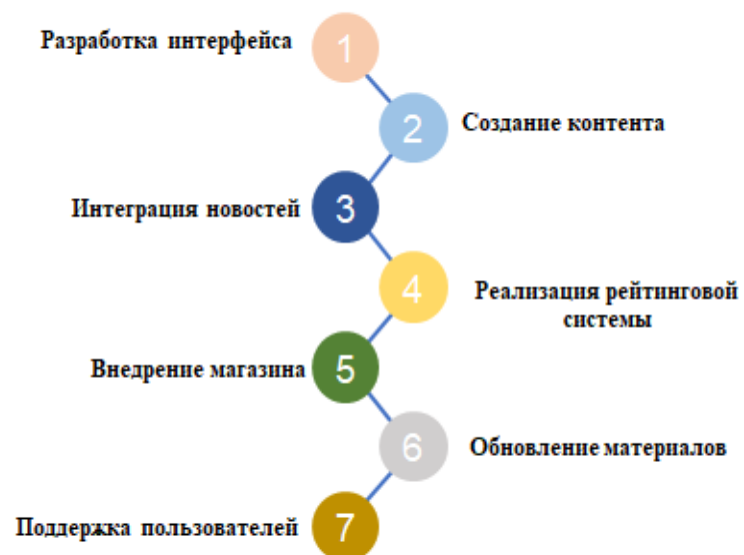
2

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.1.001 ПЗ

Лист

115



3

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.1.001 ПЗ

Лист
116

Используемые средства, инструменты

Среда разработки:  1 Visual Studio Code (VS Code)

Язык программирования:  2 Dart

Фреймворк:  3 Flutter

База данных :  4 Firebase

Эмуляторы и устройства для тестирования:  5 Android Studio – Emulator
Физические устройства

4

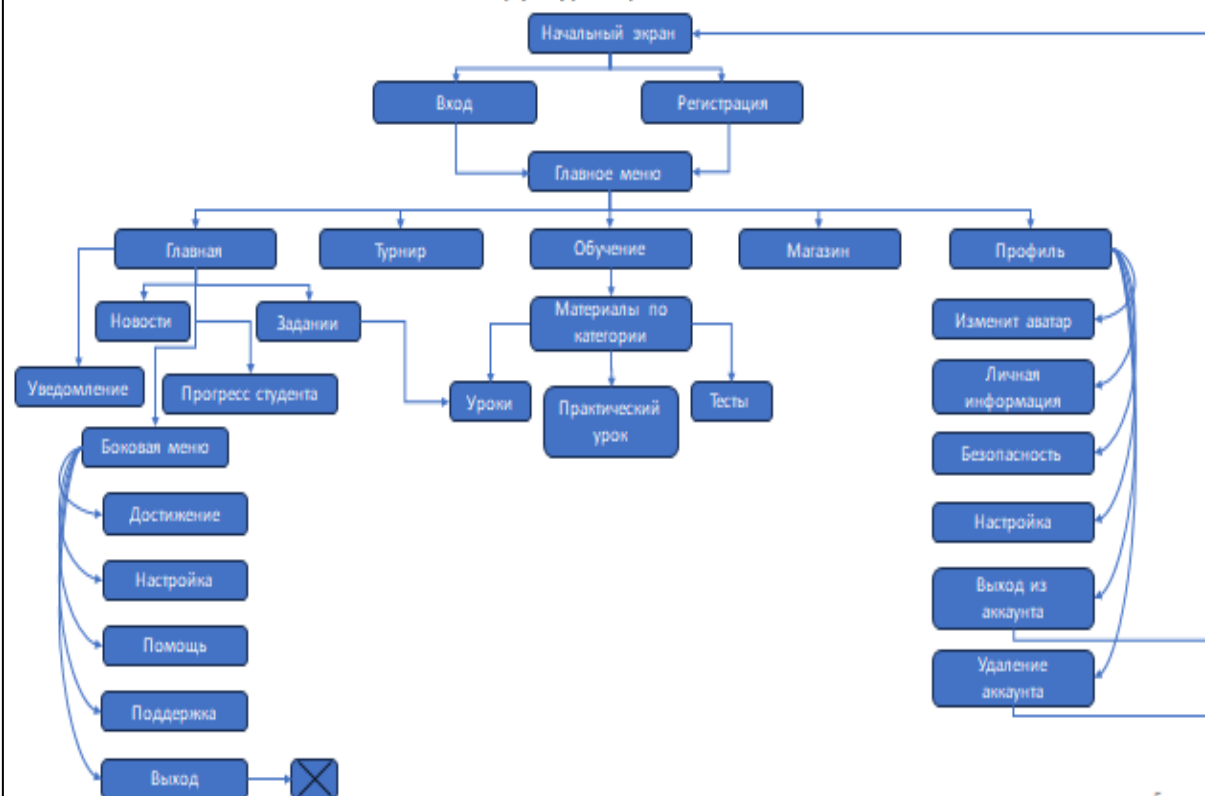
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.1.001 ПЗ

Лист

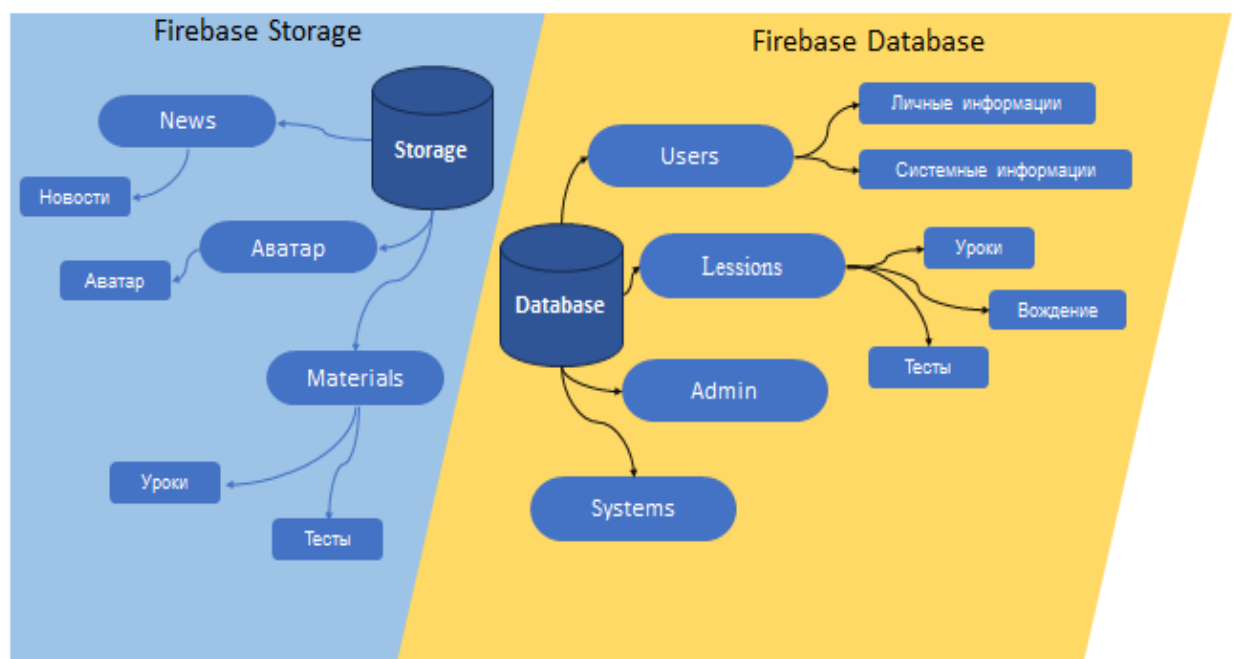
117

Структура приложения



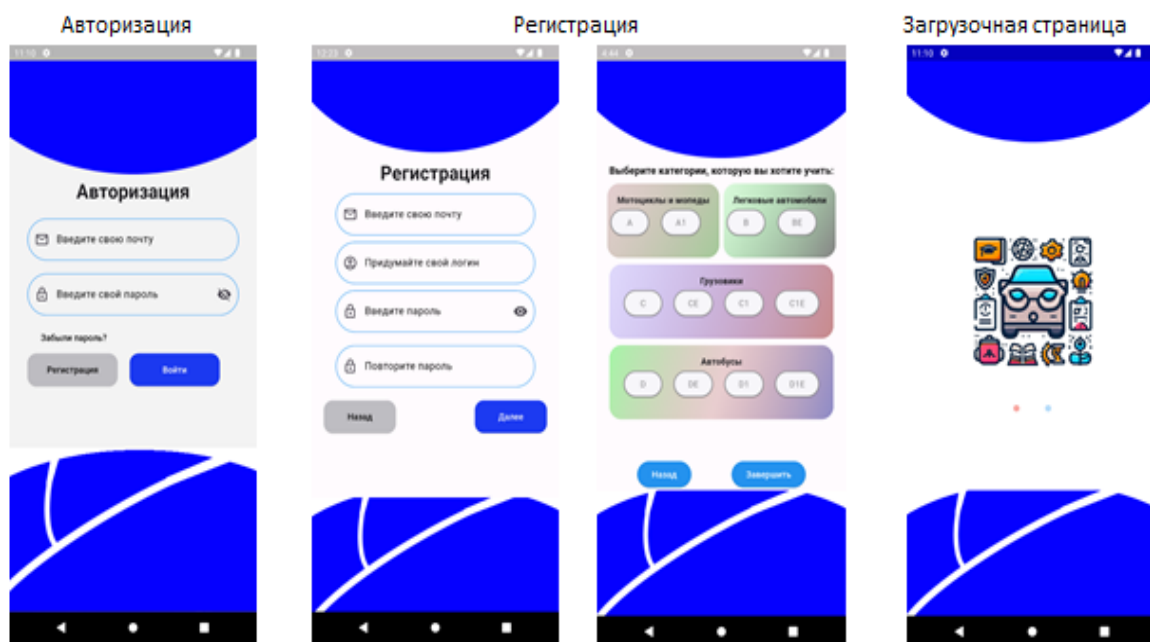
5

Схема базы данных



6

Скриншот приложения



7

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.1.001 ПЗ

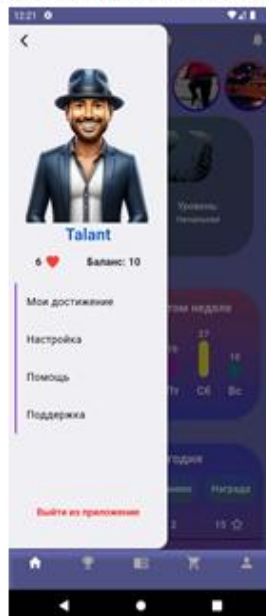
Лист

120

Главная страница



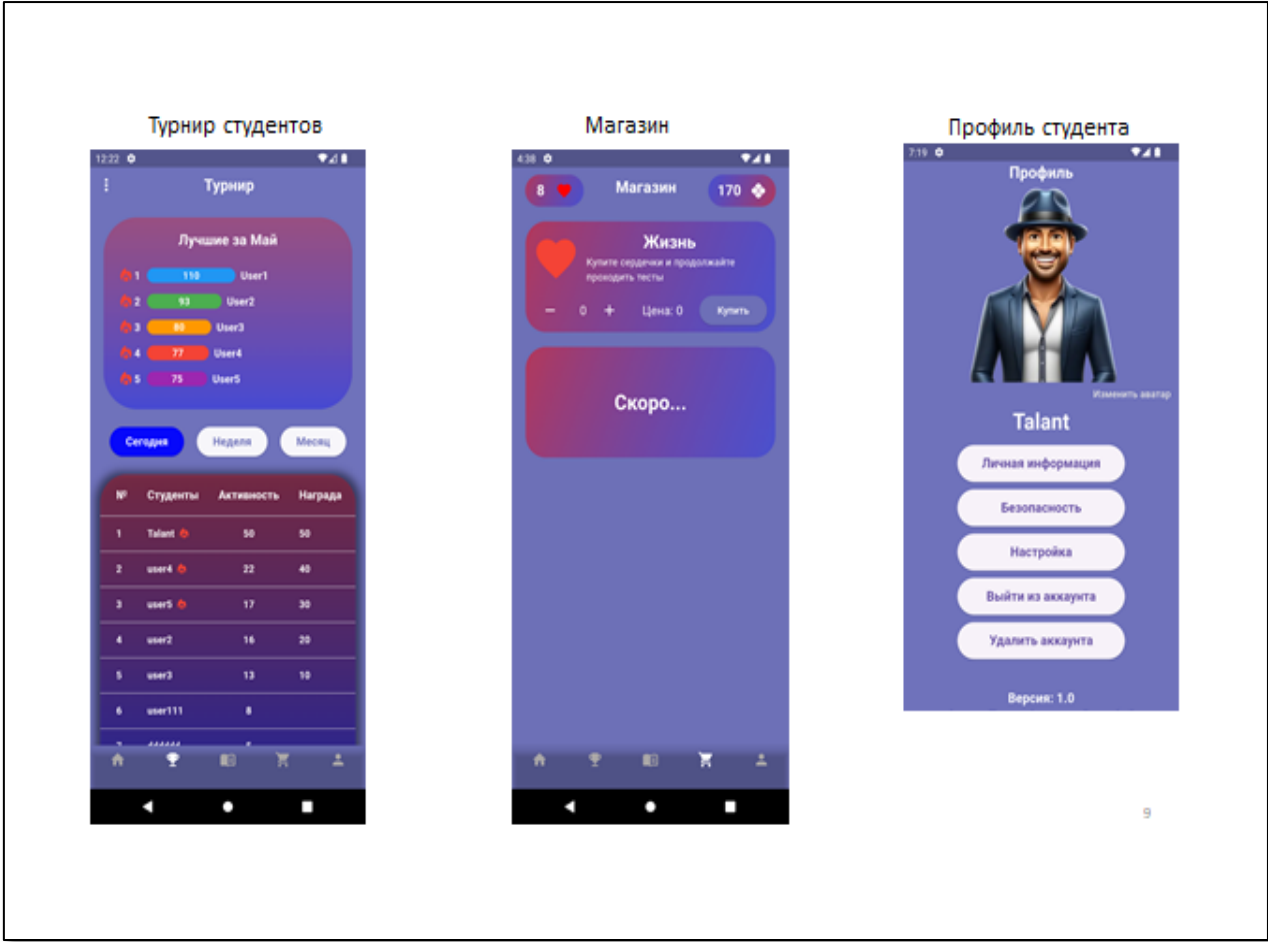
Главная страница
Под меню главной страницы



Новости

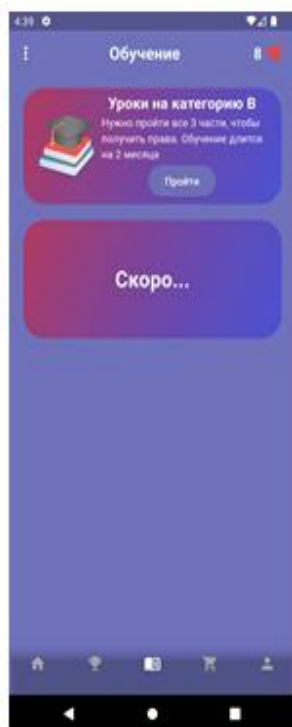


86

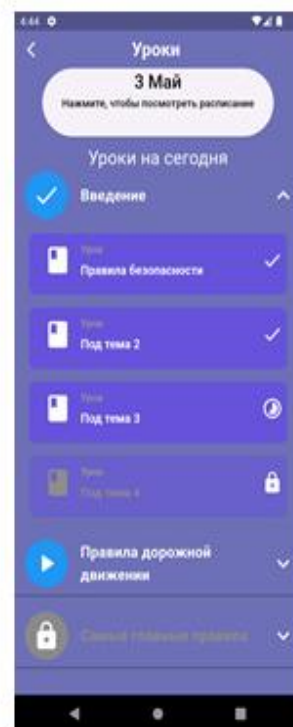


Изм.	Лист	№ докум.	Подпись	Дата

Страница материалов



Уроки



Изм.	Лист	№ докум.	Подпись	Дата

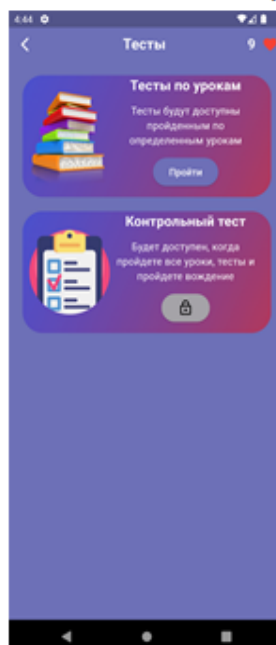
ПГУ 1.090301.1.001 ПЗ

Лист
123

Уроки



Тесты



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.1.001 ПЗ

Лист
124

Заключение

В целом, разработка и внедрение мобильного приложения "Автошкола" представляет собой значимый шаг в сфере образования и подготовки водителей.

Мобильное приложение "Онлайн авто-школа" дает возможность ученикам учиться в любом месте и в любое время, обеспечивая гибкий и удобный график обучения. Это особенно важно в современном мире, где люди часто заняты и не могут уделить достаточно времени для посещения обычных автошкол.

12

					ПГУ 1.090301.1.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		125