Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Институт информационных технологий

Факультет компьютерных технологий

Кафедра информационных систем и технологий

К защите допустить:

Заведующий кафедрой ИСиТ
_____ А.И. Парамонов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ПРОВЕДЕНИЯ НАСТОЛЬНЫХ РОЛЕВЫХ ИГР

БГУИР ДП 1-40 01 01 07 100 ПЗ

Студент	И.Г. Малашкевич
Руководитель	В.А. Сицко
Консультанты:	
от кафедры	В.А. Сицко
по технико-экономическому разделу	В.Г. Горовой
Нормоконтроллёр	А.С. Шелягович
Рецензент	

Минск 2024

Р Е Ф Е Р А Т

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ПРОВЕДЕНИЯ НАСТОЛЬНЫХ РОЛЕВЫХ ИГР: дипломный проект / И.Г. Малашкевич. – Минск: БГУИР, 2024, – п.з. – 121 с., чертежей (плакатов) – 6 л. формата А1.

Объектом проектирования является мобильное приложение для проведения настольных ролевых игр.

Целью дипломного проекта является разработка и реализация универсального мобильного приложения для проведения разнообразных настольных ролевых игр.

В рамках выполнения проекта был реализован комплексный подход, включающий: систему заметок и журналов для фиксации событий и информации о персонажах, генераторы игровых элементов, таких как персонажи, предметы, события и места, инструменты для автоматизации расчётов бросков игральных костей и их модификаций, средства управления игровыми ресурсами и характеристиками персонажей.

Для разработки приложения использовались современные технологии, такие как C# и платформа .NET MAUI, что обеспечило кроссплатформенную совместимость, а также возможность масштабирования и расширения функционала.

Проведён анализ аналогичных программных средств, на основании которого сформирована спецификация требований и обоснован выбор инструментов и подходов к разработке. В рамках проектирования были разработаны архитектура, интерфейс и алгоритмы приложения, а также проведено тестирование, показавшее соответствие программного средства заявленным требованиям.

Технико-экономическое обоснование показало целесообразность разработки: приложение способно удовлетворить растущий спрос на инструменты для настольных ролевых игр и занять устойчивую нишу нарынке.

В целом, данное приложение улучшает качество игрового процесса в настольные ролевые игры, делая его более организованным, мобильным и доступным для широкой аудитории.

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Институт информационных технологий

Факультет КТ Кафедра ИСиТ
Специальность 1-40 01 01 Специализация 01

УТВЕРЖДАЮ
А.И.Парамонов
« 24 » сентября 2024 г.

ЗАДАНИЕ
по дипломному проекту студента

Малашкевич Илья Георгиевич
(фамилия, имя, отчество)

1. Тема проекта: **Мобильное приложение для проведения настольных ролевых игр**

утверждена приказом по университету от « 24 » сентября 2024 г. № 112-и

2. Срок сдачи студентом законченной работы 23 декабря 2024 года

3. Исходные данные к проекту Тип операционной системы – Android; Язык программирования – C#; объектно-ориентированное приложение. Перечень выполняемых функций: хранение заметок и игровой статистики, генерация игровых событий и объектов, генерация бросков игральных костей. Назначение разработки: предоставление широкого спектра инструментов для оптимизации и автоматизации процесса проведения партий в настольные ролевые игры.

4. Содержание пояснительной записки (перечень подлежащих разработке вопросов):

Введение
1 Анализ предметной области
2 Моделирование предметной области
3 Проектирование и разработка программного средства
4 Тестирование программного средства
5 Руководство по эксплуатации программного средства
6 Технико-экономическое обоснование разработки программного средства
Заключение
Список использованных источников
Приложение А. Исходный код программного средства

5. Перечень графического материала (с точным указанием наименования) и обозначения вида и типа материала):

Диаграмма вариантов использования. Плакат – формат А1, лист 1.

Диаграмма деятельности. Плакат – формат А1, лист 1.

Общий алгоритм работы приложения. Схема алгоритма – формат А1, лист 1.

Модуль заметок и журнала. Схема алгоритма – формат А1, лист 1.

Генерация бросков игральных костей. Схема алгоритма – формат А1, лист 1.

Модуль генерации. Схема алгоритма – формат А1, лист 1.

6. Содержание задания по технико-экономическому обоснованию

Технико-экономическое обоснование разработки мобильного приложения для проведения настольных ролевых игр

Консультанты по дипломному проекту (с указанием разделов, по которому они консультируют):

Сицко А.В. – разделы 1-5;

Горовой В.Г. – раздел 6 (технико-экономическое обоснование);

Шелягович А.С. – нормоконтроль.

ПРИМЕРНЫЙ КАЛЕНДАРНЫЙ ГРАФИК ВЫПОЛНЕНИЯ ДИПЛОМНОГО ПРОЕКТА

Наименование этапов дипломного проекта	Объём готовности проекта в %	Срок выполнения этапа	Примечание
Первая опроцентовка (введение, разделы 1-3)	30%	16.09.24 – 25.10.24	Консультант от кафедры
Вторая опроцентовка (разделы 3-5)	60%	26.10.24 – 30.11.24	Руководитель проекта
Третья опроцентовка (разделы 5-6, заключение, список использованных источников)	90%	01.12.24 – 18.12.24	Руководитель проекта
Консультации по оформлению графического материала и пояснительной записки, нормоконтроль	–	С 01.10.24 (согласно графику)	Нормо-контролёр
Итоговая проверка готовности дипломного проекта на заседании рабочей комиссии кафедры ИСиТ и допуск к защите в ГЭК	100%	С 23.12.24 (согласно графику)	Председатель рабочей комиссии
Рецензирование дипломного проекта	100%	С 26.12.24 (согласно распоряжению)	Рецензент
Защита дипломного проекта	100%	С 18.01.25 (согласно приказу)	ГЭК

Дата выдачи задания 16 сентября 2024 г. Руководитель _____ /В.А. Сицко/

Задание принял к исполнению _____ / И.Г. Малашкевич /

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Тема дипломного проекта – мобильное приложение для проведения настольных ролевых игр.

Мобильное приложение для проведения настольных ролевых игр используется для улучшения качества и оптимизации игрового процесса. Оно помогает игрокам и ведущему управлять множеством аспектов игры, таких как отслеживание характеристик персонажей, расчёт очков здоровья, управление инвентарём и автоматизация сложных игровых механик, что позволяет участникам, не распылятся на разнообразные расчеты и избавляет от необходимости в больших наборах игровых принадлежностей. Приложение делает игру более организованной и позволяет уделять больше времени на взаимодействие и творчество, а не на рутинные расчёты или поиск информации в правилах. Например, ведущий может мгновенно генерировать события, а игроки – в несколько нажатий сгенерировать необходимый бросок игральной кости и быстро внести изменения в характеристики своих персонажей, после чего создать и сохранить заметку о игровом событии. Такое приложение позволяют проводить партии в любом удобном месте, даже если физическое взаимодействовать с книгами и записями невозможно, так как приложения-помощники позволяют хранить все необходимые материалы под рукой, делая процесс игры плавным, более динамичным и независимым от места проведения.

Целью дипломного проекта является разработка и реализация универсального мобильного приложения для проведения разнообразных настольных ролевых игр.

Разрабатываемое приложение должно предоставлять пользователю удобные инструменты для проведения партий в разнообразные настольные ролевые игры. Приложение должно способствовать оптимизации и упрощению подготовки к проведению партий в настольные ролевую игру для всех участников. Разработка такого приложения позволит собрать широкий спектр инструментов для настольных ролевых игр в одном месте, что позволит проводит партии имея при себе только мобильный телефон, что повысить мобильность игроков.

Пояснительная записка к дипломному проекту содержит шесть разделов.

Первый раздел – «Анализ предметной области» – включает в себя описание предметной области, обоснование актуальности разработки, анализ методов, способов, подходов, методик и технологий, существующих в рассматриваемой предметной области, анализ существующих аналогов и

разработка первичных спецификации требований к разрабатываемому приложению.

Второй раздел – «Моделирование предметной области» – включает в себя функциональные модели предметной области и разработку укрупненной подробной функциональной спецификации требований к разрабатываемому приложению.

Третий раздел – «Проектирование и разработка программного средства» – включает в себя проектирование и разработку архитектуры приложения, проектирование алгоритмов, разработка общего дизайна и структуры приложения.

Четвертый раздел – «Тестирование программного средства» – содержит описание процесса проверки работоспособности приложения, разработанные для данной цели тесты и сведения о результатах, проведенного в процессе разработки тестирования, выявленных по результатам тестирования ошибках и предпринятых действиях позволивших их устранить.

Пятый раздел – «Руководство по эксплуатации программного средства» – содержит иллюстрированное руководство пользователя по эксплуатации приложения с описанием использования основных функций и инструкцией по запуску приложения.

Шестой раздел – «Технико-экономическое обоснование разработки программного средства» – содержит расчеты технико-экономическое обоснования разработки приложения.

Данный дипломный проект выполнен мной лично, проверен на заимствования в системе «Антиплагиат», оригинальность пояснительной записки составляет 83%

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

Предметная область – это часть реального мира, которая подлежит изучению с целью определения объектов и их отношений, явно характеризующих исследуемое. Предметной областью приложения является совокупность объектов, свойства которых и отношения, между которыми представляют интерес для пользователей информационной системы.

Разрабатываемое мобильное приложение предназначено для проведения настольных ролевых игр. Следовательно, изучаемой предметной областью являются настольные игры, а именно инструменты необходимые для проведения партий в настольные ролевые игры.

Мобильное приложение для проведения настольных ролевых игр решает множество проблем, с которыми сталкиваются как ведущие, так и игроки. Оно призвано автоматизировать рутинные процессы: управление персонажами, отслеживание характеристик, расчёты бросков кубиков, ведение журнала сессий и управление игровыми ресурсами. Благодаря этому приложение помогает ускорить игру, минимизировать ошибки при расчётах и сократить время на подготовку к сессиям.

Настольная ролевая игра (tabletop role-playing game, TRPG), представляет собой классификацию ролевой игры (role-playing game, RPG) и относится к классу коллективных игр, в которой участники, взаимодействуя в рамках вымышленного мира, берут на себя роли вымышленных персонажей, описывают действия своих персонажей с помощью речи и движений [1].

Основой любой настольной ролевой игры является сочетание воображения, нарратива и системных правил. Воображение позволяет игрокам погружаться в различные сценарии, будь то фэнтезийные миры, научная фантастика, ужасы или детективы. Игровые правила структурируют это воображение, определяя, как именно взаимодействовать с окружающей средой, как действуют персонажи, каким образом происходят бои или события, и как рассчитываются их результаты [2].

Основные элементы настольных ролевых игр включают создание персонажей, проведение сражений, управление инвентарём и прогрессирование сюжетной линии.

Процесс игры строится на нарративе, творческом воображении и принятии решений, а также на чётко установленных правилах, которые определяют, как именно происходят события в игровом мире. Участники определяют действия своих персонажей на основе их характеристик, и

действия достигают успеха или терпят неудачу в соответствии с установленной формальной системой правил и руководств, обычно включающих использование игральных костей. В рамках правил игроки имеют свободу импровизировать, их выбор определяет направление и исход игры. [3]

Для проведения настольных ролевых игр необходим определенный набор разнообразных инструментов.

Стандартный набор инструментов для проведения настольной ролевой игры состоит из:

– Блокнот, ежедневник или тетрадь – данный инструмент предназначен для ведения журнала игровой сессии, создания пометок и памяток игрока, расчетов значений, записи значимых событий и их результатов.

– Ручка и карандаш – предназначены для ведения записей.

– Стирка – предназначена для удаления пометок, оставленных карандашом.

– Лист персонажа – предназначен для ведения учета характеристик, имущества и способностей персонажа игрока.

– Набор игральных костей – состоит из шестигранных, восьмигранных, двенадцатигранных и двадцатигранных игральных костей. В большинстве ролевых систем игральные кости используется для добавления эффекта случайности в проверки характеристик персонажей игроков.

– Книга правил – содержит правила ролевой системы, обязательный атрибут для решения спорных вопросов, уточнения нюансов в правилах и поиска подходящей для игровой ситуации проверки.

– Книга приключений – книга, содержащая одно или несколько готовые приключения для определенной ролевой системе в одном или нескольких сеттингах.

– Атлас – сборник карт, пейзажей, портретов и прочего разнообразного графического материала объединённого общим сеттингом предназначенных для визуального отображения мест, которые посещают персонажи игроков в игровом мире.

– Ширма – позволяет ведущему скрыть от игроков свои броски игральных костей и заметки. Обычно на ширме написаны краткие выдержки из основных правил определенной ролевой системы.

Стандартный набор инструментов обычно не зависит от ролевой системы и почти не меняется в разных настольных ролевых играх. Инструменты из набора не обязательны для проведения сессии в настольную ролевую игру, но без любого из них данный процесс будет существенно осложнен [4].

Ведущий, так же известный как мастер игры (Game Master) или чаще просто мастер – в настольных ролевых играх ведущий, это тот, кто организует игровой процесс, придумывает сюжет, следит за соблюдением правил, решает возникающие спорные вопросы, описывает ситуацию, взаимодействует с игроками и управляет игровым миром, реагируя на действия персонажей игроков.

В различных ролевых системах ведущего могут называть по-разному, например, в настольной ролевой игре Dungeons&Dragons мастер называется DM (Dungeon Master). Некоторые ролевые системы заменяют ведущего сводом жестким сводом игровых правил.

Игрок – человек, участник текущей ролевой игры. В распоряжении игрока находится какой-либо персонаж, представляющий его в игровом мире. У каждого персонажа есть уникальные характеристики, способности и предметы, которые влияют на его действия. Для расчёта успешности тех или иных действий обычно используется система игральных костей, задающая элемент случайности. Чаще всего у игрока под управлением один персонаж, так как это позволяет игроку не рассеивать внимание, сконцентрироваться на отыгрыше и управлении. Но есть ролевые системы, в которых под управлением игрока несколько персонажей или целые отряды. Игрок чаще всего противопоставляется ведущему.

Ролевая система – это совокупность правил конкретной настольной ролевой игры.

Существует несколько трактовок определения ролевая система. Чаще всего за ролевую систему принято считать четко определенный набор правил, по которым ведущий определяет результаты действий персонажей игроков, возможность персонажа произвести заявленное действие и расчет статистики. В более широкой трактовке к системе относят всю совокупность правил, требований и рекомендаций. Например, правила создания персонаже, требования к развитию персонажа, описание стандартов игрового сеттинга, обязательные игровые модули, обязательная тематика для игровых сессий. Коммерческие ролевые системы для настольных ролевых игр чаще всего не используются в чистом виде. Игроки и ведущий так или иначе дополняют существующие ролевые системы своими домашними правилами.

Сеттинг – это среда, в которой происходят действия ролевой игры.

Игровая сессия – это отрезок времени, используемый для проведения настольной ролевой игры. Обычно игровая сессия длится от двух до десяти часов. Длинный сюжет, который требует от восьми и более часов разбивается на несколько сессий.

Неигровой персонаж (Non-player character, сокращенно NPC) – персонаж в играх, который не находится под контролем ведущего).

Домашние правила (Homebrew) – это особый набор правил, создаваемый игроками в настольную ролевую игру для изменения ролевой системы с целью создания комфортных игровых условий для всех участников [5].

1.2 Обоснование актуальности разработки ПС

До пандемии настольные ролевые игры находились на периферии массового развлечения. Их аудитория, хотя и была предана жанру, оставалась относительно небольшой и состояла в основном из энтузиастов. Настольные ролевые игры требовали времени, организованности и личного участия, что затрудняло их распространение среди широкой аудитории. В то время многие предпочитали более доступные и быстрые формы досуга, такие как видеоигры или мобильные приложения, которые не требовали сложной подготовки [6].

Однако пандемия и вызванные ею меры социальной дистанции значительно изменили картину. Ограниченные возможности для живого общения и встречи с друзьями спровоцировали поиск новых форм взаимодействия. Люди стали искать способы сохранить контакт с близкими и знакомыми, даже находясь на расстоянии. В этих условиях настольные ролевые игры оказались идеальной площадкой для «живого» общения. Они позволяли игрокам погружаться в миры, полные взаимодействий, общения и кооперации, приятно проводить время в компании и получить позитивные эмоции [7].

Процесс подготовки и проведения сессии в настольную ролевую игру связан с необходимостью обработки, анализа и хранения большого количества информации. Игроки создают персонажей с уникальными характеристиками, способностями, навыками и предметами, которые необходимо постоянно отслеживать. Ведущий управляет не только событиями и сюжетом, но и множеством неигровых персонажей (NPC), врагов, окружающей средой и динамикой игрового мира. Все это требует высокой внимательности, контроля, учета и правильных расчетов, что может сильно усложнять игровой процесс, особенно в кампаниях растягивающихся на большое количество сессий, где игра может длиться несколько месяцев или даже лет.

Мобильные приложения для настольных ролевых игр призваны упростить управление всеми этими аспектами, предоставив удобные инструменты как для игроков, так и для ведущих. Такие приложения автоматизируют рутинные задачи, уменьшая объем работы, который нужно выполнять вручную, и делая игру более плавной и организованной. Основные функции приложений-помощников могут включать создание и управление персонажами, проведение расчётов бросков кубиков, ведение журнала

игровых событий, управление инвентарём и мониторинг здоровья персонажей.

Одной из ключевых проблем, с которыми сталкиваются игроки в настольные ролевые игры, является необходимость управлять множеством материалов. Например, игроку нужно иметь под рукой информацию о своем персонаже – его характеристиках, навыках, заклинаниях, инвентаре и текущем состоянии. Ведущему необходимо отслеживать не только персонажей игроков, но и NPC, состояние мира, прогрессирование сюжета, а также случайные события и встречи. Мобильное приложение может помочь сократить время на выполнение этих задач, предложив пользователям удобный набор разнообразных инструментов, автоматизирующих данные игровые процессы.

Кроме того, в настольных ролевых играх обычно используется механика бросков игральных костей для определения исхода каких-либо действий. В классических условиях игроки кидают реальные многогранные кубики, однако в мобильном приложении можно реализовать встроенный генератор случайных чисел, который автоматически рассчитывает результаты бросков и применяет соответствующие модификаторы. Это значительно ускоряет игровой процесс и устраняет вероятность ошибок при подсчете, что часто случается при использовании традиционных методов.

Еще одной важной функцией мобильных приложений для настольных ролевых игр можно считать управление инвентарём персонажей игроков. В процессе игры персонажи собирают множество предметов, от оружия и доспехов до волшебных артефактов и полезных ресурсов. Отслеживание всего этого может стать проблемой, особенно в длительных кампаниях, когда количество ресурсов игроков увеличивается до десятков, а то и сотен разнообразных позиций. Мобильное приложение позволит игрокам легко управлять инвентарём своих персонажей, легко обновляя данные о количестве предметов, их характеристиках и влиянии на игру.

Для ведущего приложение может предоставить широкий набор инструментов для управления сюжетом и игровым миром. Приложение позволит генерировать случайные события и встречи, отслеживать состояние NPC, мониторить здоровье врагов и даже управлять игровыми картами. Приложения также могут включать базы данных с предустановленными шаблонами монстров, ловушек и других элементов мира, что помогает ведущему создавать разнообразные и захватывающие ситуации для игроков.

С ростом популярности настольных ролевых игр многие игроки начали искать удобные способы организации партий, что вызвало всплеск интереса к приложениям, помогающим в проведении игровых сессий.

Однако, несмотря на очевидную востребованность таких приложений, на рынке существует дефицит комплексных решений. Особенно данный дефицит ощущается в СНГ так как большинство решений на рынке произведены вне региона и имеют низкую или в принципе отсутствующую локализацию. [8]

Кроме того, большинство существующих мобильных приложений для настольных ролевых игр предоставляют лишь отдельные функции, например, генераторы персонажей, инструменты для расчета бросков кубиков или менеджеры инвентаря, но они не представляют собой комплексных инструментов, которые могли бы охватить все аспекты игрового процесса. Это приводит к тому, что игрокам и ведущим приходится использовать сразу несколько приложений, каждое для решения узкого спектра задач, что затрудняет проведение партии, отвлекает от основного игрового процесса и замедляет процесс обработки игровой информации.

В связи с низкой заполненностью рынка и растущим спросом на приложения для проведения настольных ролевых игр, разработка нового конкурентноспособного приложения, которое будет охватывать все нужды игроков и ведущих в одной платформе, является своевременным и перспективной проектом, имеющим высокие шансы окупится и занять существенную нишу на рынке. Новое конкурентноспособное приложение может привлечь как новых пользователей, так и опытных игроков, стремящихся к удобству и эффективности в организации игровых сессий, что сделает настольные ролевые игры ещё более привлекательными для широкой аудитории, а приложение востребованным на рынке.

1.3 Анализ существующих методов, подходов и технологий

Анализ методов, подходов и технологий, используемых при разработке мобильных приложений для проведения настольных ролевых игр, требует комплексного подхода, охватывающего аспекты разработки интерфейсов, функциональности, пользовательского опыта и интеграции с различными платформами. [9]

Одним из ключевых аспектов разработки любых мобильных приложения для проведения настольных ролевых игр является пользовательский интерфейс (UI).

Основная задача – создание интерфейс, который будет максимально упрощать взаимодействие с различным функционалом, обеспечивая удобный доступ к инструментам предоставляемых приложением.

Подход к разработке UI в приложениях для настольных ролевых игр основывается на принципах UX-дизайна (user experience design). Этот метод

предполагает проектирование интерфейса таким образом, чтобы он был воспринимаемым, удобным, простым и давал игрокам и ведущим доступ ко всем необходимым инструментам без перегрузки экрана ненужной информацией. Методы проектирования включают:

– Пользовательские исследования (User Research). Использование опросов, интервью и анализа поведения пользователей позволяет понять, как игроки взаимодействуют с приложением. Этот метод помогает выявить важные функции и улучшить UX.

– Прототипирование. Создание прототипов интерфейсов для тестирования различных вариантов представления информации, таких как инвентарь, характеристики персонажей, механика боя и управление картами. Этот способ позволяет быстро вносить изменения и тестировать новые идеи до полноценной разработки.

– Адаптивный дизайн (Responsive Design). Приложения должны быть адаптированы под различные устройства (смартфоны, планшеты) с разными размерами экранов. Это особенно важно для отображения сложных данных.

– При разработке мобильных приложений используются различные технологии программирования, в зависимости от целевой платформы и выбранной архитектуры приложения. Выбор технологии зависит от требований к кроссплатформенности или производительности:

– Нативная разработка. Этот метод предполагает создание приложений непосредственно под конкретную операционную систему (iOS или Android). Нативная разработка обеспечивает максимальную производительность и доступ ко всем возможностям платформы, таким как работа с графикой, уведомлениями и памятью устройства.

– Кроссплатформенная разработка. Этот подход позволяет создавать приложения, которые работают сразу на нескольких платформах. Такой метод особенно актуален для приложений для проведения настольных ролевых игр, поскольку он снижает затраты на разработку и позволяет охватить большую аудиторию.

Основные функции приложения для проведения настольных ролевых игр можно разделить на несколько категорий, каждая из которых требует своего подхода к реализации:

– Генерация и управление персонажами. Одним из ключевых элементов в настольных ролевых играх является создание персонажа с уникальными характеристиками, навыками и инвентарём. Для этого используются специальные алгоритмы генерации персонажей с возможностью настраивать параметры. Современные технологии позволяют автоматизировать создание персонажа на основе предустановленных правил игры (например, системы D&D или Pathfinder).

– Инвентарь и снаряжение: В настольных ролевых играх игроки часто управляют большим количеством предметов и ресурсов. Приложение должно предоставлять простой и быстрый способ доступа к инвентарю, управлению предметами и их характеристиками. Для этого используются базы данных с предметами, которые связаны с персонажами игроков и автоматически обновляются в процессе игры.

– Броски кубиков и расчеты: Механика бросков кубиков играет важную роль в настольных ролевых играх для определения исхода действий персонажей. Современные технологии позволяют реализовать генераторы случайных чисел, которые имитируют броски кубиков. Эти генераторы могут быть связаны с другими системами (например, модификаторами способностей), что автоматически упрощает расчеты для игроков и ведущего.

Для обеспечения стабильной работы приложения и удовлетворения ожиданий пользователей применяются различные методики тестирования:

– Юзабилити-тестирование. Оценка удобства использования интерфейсов приложения. Это помогает выявить потенциальные проблемы, связанные с взаимодействием пользователей с приложением.

– Тестирование производительности. Приложение для проведения настольных ролевых игр должно быть способно обрабатывать большие объемы данных и предоставлять быстрый доступ к ним. Тестирование производительности проверяет, насколько приложение устойчиво к нагрузкам.

Монетизация мобильных приложений для настольных ролевых игр также является важным аспектом. Возможны различные стратегии:

– Модель freemium. Основной функционал приложения предоставляется бесплатно, но доступ к дополнительным возможностям (например, расширенные базы данных персонажей, карты или предметы) может быть платным.

– Подписочная модель. Пользователи могут платить ежемесячную или годовую подписку за доступ к премиум-функциям или облачным сервисам.

– Реклама. В бесплатных версиях приложение может содержать рекламу, которая позволяет разработчикам получать доход без необходимости прямых покупок от пользователей.

Разработка мобильного приложения для проведения настольных ролевых игр – это сложный, требующий использования различных методов и технологий. Ключевыми аспектами являются удобство интерфейса, кроссплатформенность, стабильность работы, а также возможность автоматизации рутинных процессов проведения настольных ролевых игр. Комплексное решение должно объединять в себе удобный интерфейс и автоматизацию расчётов и управления данными.

1.4 Обзор существующих аналогов

Аналог – объект (техническое решение) того же назначения, близкий по совокупности существенных признаков.

Dice – популярное мобильное приложение предоставляющие инструменты для генерации бросков игральных костей и расчётом результатов бросков.

Dice позволяет игрокам в настольные игры произвести бросок игральных костей виртуально, предоставляя при этом анимацию броска игральных костей и реалистичное звуковое сопровождение. Пример броска костей представлен на рисунке 1.1.

Рисунок 1.1 – Пример генерации броска игральных костей в Dice

Дополнительно к обычному генератору бросков в приложение есть альтернативный режим, с контрастным оформлением и отдельным отображением результатов бросков каждой игральной кости. Данный режим проиллюстрирован на рисунке 1.2.

Рисунок 1.2 – Контрастный режим в приложении Dice

Преимущества:

– Простота использования.

– Гибкая настройка характеристик генерируемого броска игральных костей.

– Кастомизация игральных костей.

– Контрастный режим отображения.

Недостатки:

– Узкоспециализированный функционал.

– Высокие требования к техническим характеристикам устройства при генерации бросков большого количества игральных костей.

Spells 5e – удобное приложение справочник заклинаний для игровых персонажей игроков. Предоставляет инструменты фильтрации и поиска игровых заклинаний, позволяет создавать книгу заклинаний игрового персонажа, что позволяет игроку или ведущему быстро найти необходимое заклинание у определенного игрового или неигрового персонажа и в удобном формате прочитать его правила.

Интерфейс приложения представлен на рисунке 1.3.

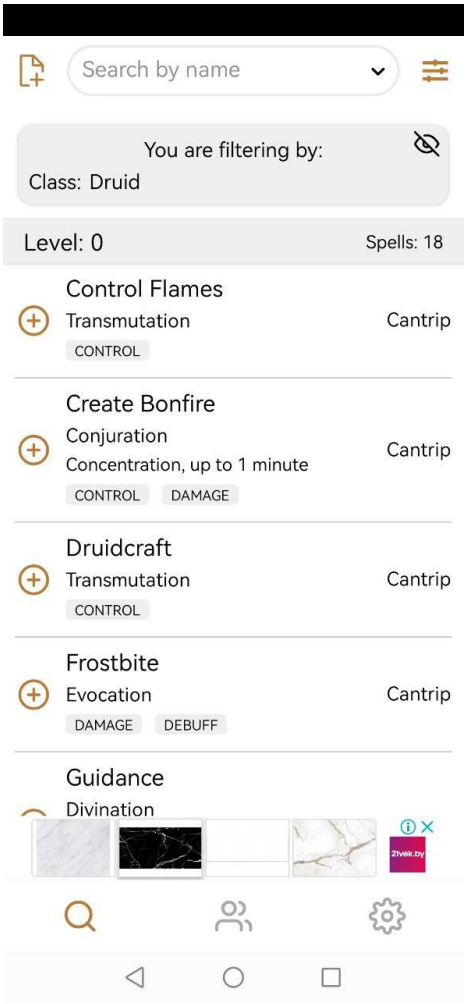

Рисунок 1.3 – Интерфейс приложения Spells 5e

Но, к сожалению, рассматриваемое приложение содержит информацию только для настольной ролевой игры Dungeons & Dragons в пятой редакции.

Кроме того, стоит отметить, что текст в приложение исключительно английский, а в тексте заклинаний отдельные блоки текста не имеют явного разграничения, что усложняет прочтение.

Преимущества:

– Удобная система фильтрации.

– Поисковая система.

– Возможность собрать необходимые заклинания в одну коллекцию для быстрого поиска.

Недостатки:

– Узкоспециализированный функционал.

– Низкое качества разделения текстовых блоков, что приводит к сложности в поиске необходимого текстового раздела.

– Отсутствие локализации.

Бестиарий DnD – приложение справочник содержащие сборники информации по противникам, сокровищам, зельям, заклинаниям и калькулятор для расчёта сложности боев. Интерфейс приложения представлен на рисунке 1.4.

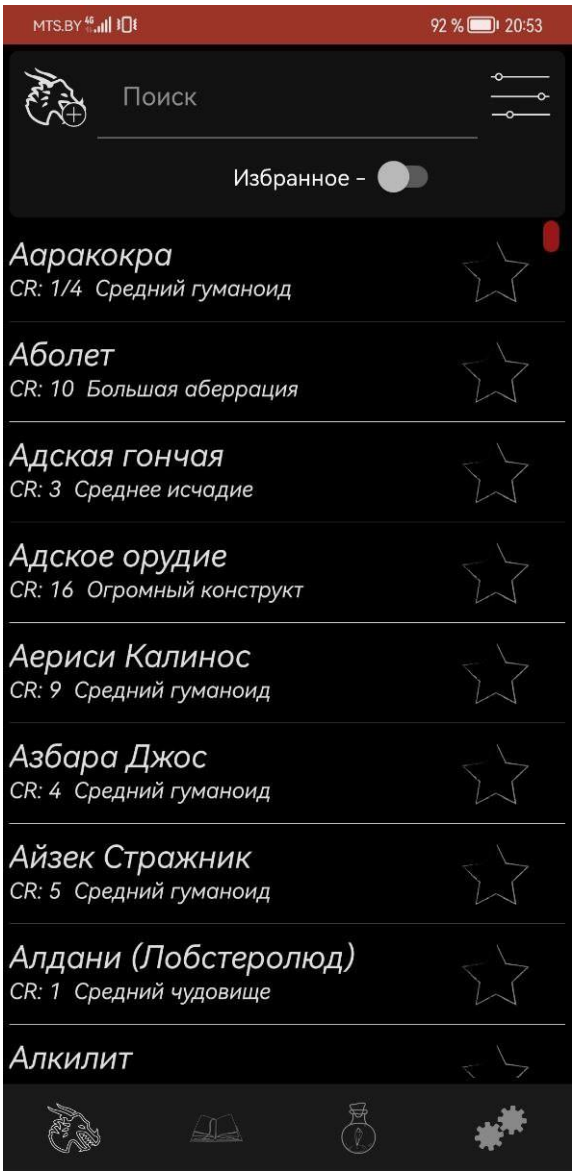

Рисунок 1.4 – Интерфейс приложения Бестиарий DnD

В разделах приложения реализован поиск по заголовку. Приложение обладает приятным интерфейсом, хорошо сгруппированной текстовой информацией. Страница справочника представлена на рисунке 1.5.

Рисунок 1.5 – Страница справочника в приложении Бестиарий DnD

Преимущества:

– Поисковая система.

– Качественная разметка текста.

– Возможность собрать необходимые заклинания в одну коллекцию для быстрого поиска.

– Русская локализация.

Недостатки:

– Специализация исключительно под одну ролевую систему Dungeon & Dragons в пятой редакции.

Compendium – приложение справочник содержащая информационные статьи о противниках в различных ролевых системах. Позволяет загружать новые и обновлять существующие информационные статьи из сети интернет Интерфейс приложения представлен на рисунке 1.6.

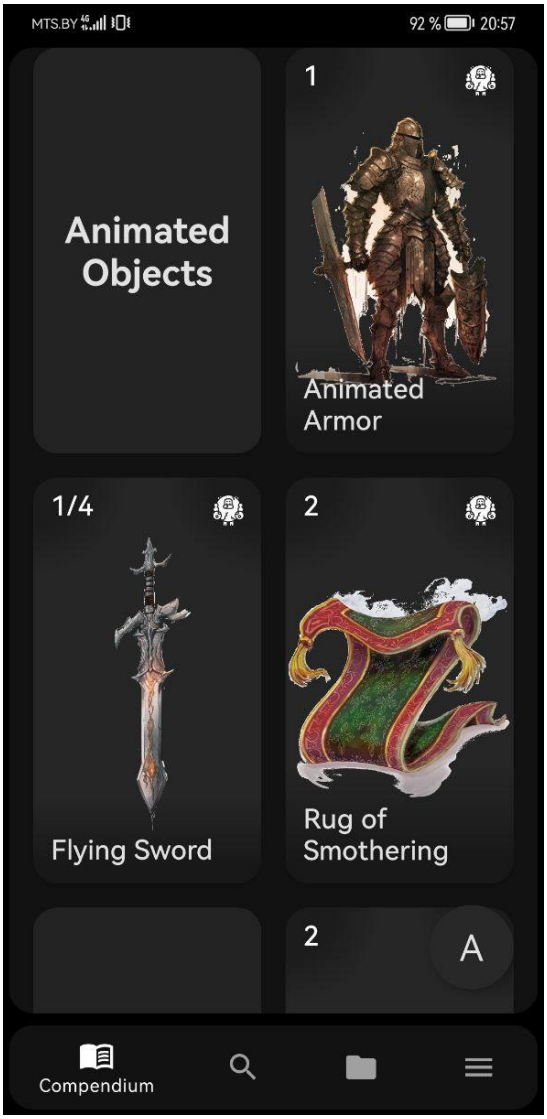

Рисунок 1.6 – Интерфейс приложения Compendium

Информационные статьи содержать текст с описанием противников для персонажей игроков в настольной ролевой игре. Текст статьи разбит на блоки характеристик, описания и иллюстрационный материал. Пример информационной статьи представлен на картинке 1.7.

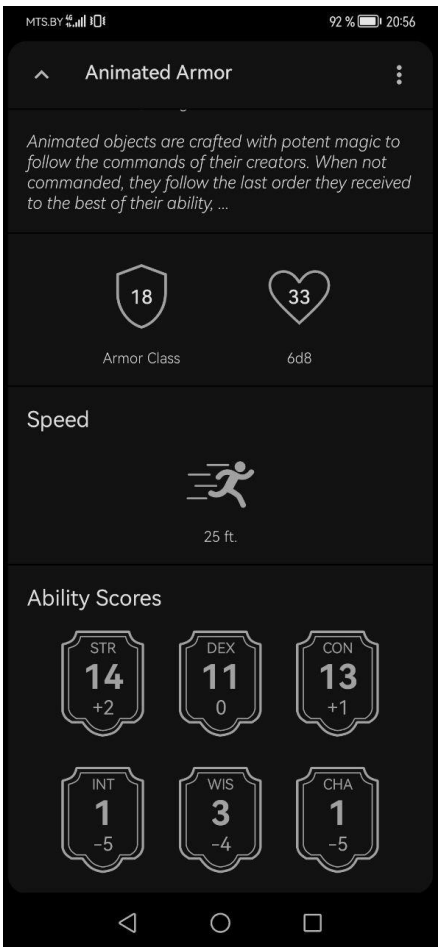

Рисунок 1.7 – Информационная статья в приложении Compendium

Преимущества:

– Удобно отформатированные информационные статьи.
– Иллюстрационный материал.
– Возможность обновления и добавления новых статей.

Недостатки:

– Узкая специализация приложения.
– Отсутствие русской локализации.
– Потребность в подключении к сети интернет для корректной работы.

RPG Notes – приложение для ведения заметок и журналов партий в настольные ролевые игры.

Приложение позволяет вести заметки по разным ролевым системам и десяткам партий. Особенно полезно для ведущих, так как позволяет оцифровать текст заметок, сюжета, событий и персонажей проводимых партий. А благодаря системе навигации легко найти необходимую запись. Интерфейс приложения представлен на рисунке 1.8.

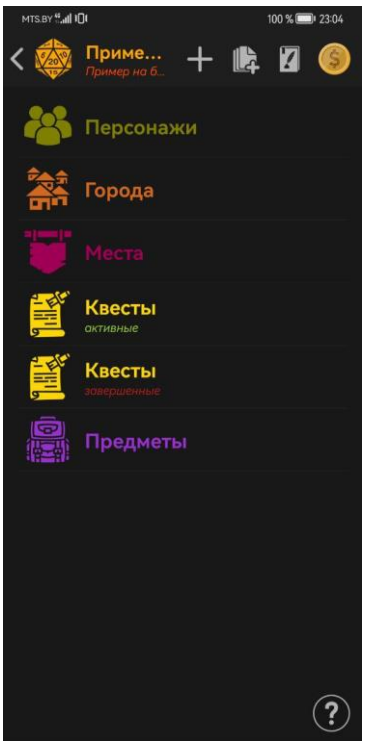

Рисунок 1.8 – Интерфейс приложения RPG Notes

Заметки в приложении поддерживают систему тегов, что позволяет быстро отфильтровать и найти необходимые записи. Интерфейс заметки представлен на рисунке 1.9.

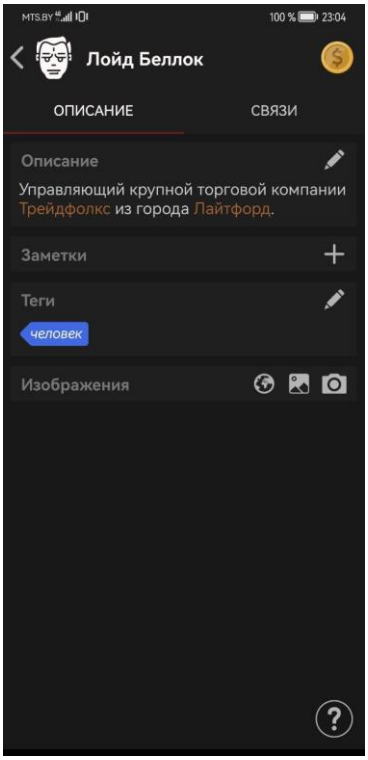

Рисунок 1.9 – Интерфейс заметки в приложении RPG Notes

Преимущества:

– Удобная структура каталогов позволяющая разделить заметки и записи по подходящим категориям.

– Поддержка системы тегов для быстрого поиска.

– Возможность применения с любой ролевой системой.

– Русская локализация.

Недостатки:

– Отсутствие поиска записи сразу по всем каталогам.

– Недостаточно широкая специализация.

Проанализировав существующие приложения, имеющие схожую задачу с проектом, был сделан вывод о том, что на данный момент существует слишком малое количество универсальных приложений для проведения настольных ролевых игр. Данные приложения должны предлагать комплексный подход, объединяя в себе все необходимые инструменты. Необходимо предоставлять игрокам возможность быстро обновлять информацию о персонажах, проводить броски кубиков, а также управлять инвентарём и ресурсами. Ведущему предоставляется инструмент для ведения сюжета, генерации событий, мониторинга здоровья и состояния противников, что делает его работу менее трудоёмкой. Такие приложения способы не только автоматизировать игровой процесс, но и создать более комфортные условия для погружения в игровой мир и сделать настольные ролевые игры доступнее для широкой аудитории, предложив более простой порог вхождения.

1.5 Выбор и обоснование инструментов разработки

Для разработки приложения выбран язык программирования C# и фреймворк .NET MAUI (Multi-platform App UI). C# и MAUI представляют собой хорошее сочетание для разработки кроссплатформенных мобильных приложений.

C# является объектно-ориентированным языком программирования, разработанным компанией Microsoft. Он предоставляет высокую производительность и имеет гибкую структуру, что делает его подходящим для создания различных типов приложений, включая мобильные, десктопные и серверные решения [10].

Использование C# в контексте разработки мобильных приложений через .NET MAUI позволит создать приложение для различных платформ (Android, iOS, Windows, macOS) с использованием единой кодовой базы. Данный подход существенно упростит процесс разработки, так как позволит задействовать одну и ту же логику для применения на разных платформах [11].

MAUI является дальнейшим развитием Xamarin.Forms и предлагает более современную архитектуру. Ключевым преимуществ выбора MAUI для разработки является поддержка C# и возможности глубокого взаимодействия с нативными API каждой из целевых платформ. Это позволяет использовать функциональность каждой платформы на уровне, сопоставимом с нативной разработкой, при этом сохраняя преимущества кроссплатформенного подхода. Кроссплатформенная разработка с использованием MAUI обеспечит уменьшение затрат на разработку и поддержку. Поскольку нет необходимости создавать отдельные версии для каждой платформы, что значительно снижает время разработки. Этот подход также снизит вероятность ошибок, связанных с тем, что отдельные версии приложений могут содержать различия в логике или функционале.

C# и MAUI обеспечит хорошую производительность, так как C# компилируется в высокоэффективный машинный код, позволяя приложению работать быстро и эффективно, что важно для мобильных приложений. Использование .NET MAUI и C# предоставит возможность оптимизации приложения для каждой платформы, что также положительно скажется на его производительности.

Важным аспектом в выборе данных инструментов является поддержка MVVM (Model-View-ViewModel) паттерна. Этот паттерн помогает разделить логику приложения и пользовательский интерфейс, что делает код более структурированным и удобным для поддержки. Применение MVVM позволяет упростить тестирование отдельных частей приложения и повысить его гибкость. Это особенно важно при работе с кроссплатформенными приложениями, где важно, чтобы интерфейс и бизнес-логика были максимально изолированы друг от друга.

Развитая экосистема .NET и наличие огромного числа библиотек также являются важными факторами при выборе языка C#. Можно использовать существующие библиотеки и фреймворки, что значительно ускоряет процесс разработки. Кроме того, важным аспектом выбора C# является интеграция с существующими инструментами и сервисами Microsoft, такими как Azure и Visual Studio [12].

Использование Visual Studio как основной среды разработки предоставляет полный набор инструментов для создания, отладки и профилирования приложений на всех платформах. Встроенные инструменты для работы с XAML, отладки и анализа производительности позволяют эффективно разрабатывать и улучшать приложение [13].

На основании вышеизложенного, выбор C# и .NET MAUI для разработки мобильного приложения позволит создать производительное, гибкое и универсальное решение, которое будет работать на разных

платформах с минимизированными затратами на поддержку и разработку. Высокая производительность, строгая типизация, поддержка нативных API и современных архитектурных паттернов делают это сочетание оптимальным для мобильной и кроссплатформенной разработки.

1.6 Спецификация требований

Проанализировав предметную область, существующие методы и технологии и существующие аналоги были сформирована спецификация требования для мобильного приложения для проведения настольных ролевых игр.

Назначение разработки – приложение предназначено для автоматизации и упрощения процесса подготовки и проведения партий в настольные ролевые игры.

Перечень основных выполняемых функций:

- Предоставление системы заметок.
- Генератор игровых элементов.
- Генератор бросков игральных костей.
- Менеджер листа персонажа.
- Журнал игровых сессий.

Входные данные для ПС:

- Текст для заметок пользователей.
- Параметры для расчета броска игральной кости.
- Параметры для генерации игровых событий.
- Характеристики игровых персонажей пользователей.

Выходные данные ПС:

- Результаты расчетов броска игральной кости.
- Результаты генерации игровых событий.

Разрабатываемое приложение должно быть совместимо с мобильными устройствами с операционной системой Android и иметь кроссплатформенный потенциал для последующего выхода на iOC и Windows. Для установки приложение не должно требовать дополнительных сторонних программных средств.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Моделирование предметной области – процесс, направленный на изучение и представление ключевых аспектов изучаемой предметной области, которые необходимо учитывать при разработке программного обеспечения. Цель моделирования – это создание абстрактного представление системы, позволяющее понять структуру, связи и процессы в рамках конкретной задачи. Данный этап позволяет лучше понять требования к приложению и определить его структуру и функциональность. При моделировании предметной области разрабатываются концептуальные модели, которые формируют основу для архитектуры программного обеспечения, пользовательских интерфейсов и механики взаимодействия. [14]

Предметная область приложения для проведения настольных ролевых игр является сложным взаимодействием множества элементов: игроков, игровой системы, игровых персонажей, предметов, событий и сюжета.

Моделирование предметной области приложения для проведения настольных ролевых игр требует учёта множества взаимодействующих элементов. Оно служит основой для проектирования и разработки системы, которая позволит эффективно управлять всеми аспектами настольной игры, делая процесс более удобным для пользователей.

Основная поставленная перед приложением задача – упростить ведение игры как для игроков, так и для ведущего, обеспечивая возможность эффективного управления всеми компонентами настольной игры через цифровой интерфейс.

Для начала моделирования предметной области определяются базовые сущности. Ключевыми элементами в этой области являются игроки, игровые персонажи, игровой мир, сюжет, правила и механики настольной игры. Каждая из этих сущностей играет важную роль игровом процессе, и их взаимодействие необходимо корректно смоделировать.

Игрок является основной сущностью. Именно игрок взаимодействует со всеми остальными сущностями и именно для него создается разрабатываемое приложение.

Игровые персонажи являются ключевой сущностью, так как через них игроки взаимодействуют с игровым миром. Персонажи обладают определёнными характеристиками, навыками, способностями и инвентарем, которые влияют на их взаимодействие с игровым миром. Для моделирования персонажей необходимо учесть возможность создания, редактирования и отслеживания изменений в их состоянии на протяжении игры.

Мир игры является ещё одной ключевой сущностью. Он состоит из различных локаций, неигровых персонажей, противников и событий, которые происходят в процессе игровой кампании. Моделирование мира предполагает взаимодействие между различными сущностями, такими как персонажи и неигровой персонаж, боевые сцены, квесты и случайные встречи.

Сюжет в настольных ролевых играх строится на заранее подготовленных или генерируемых событиях. Моделирование сюжетной линии требует создания системы управления событиями, которая позволяет ведущему контролировать развитие истории. Данные события могут включать квесты, взаимодействие с неигровыми персонажами, встречу с противниками или решение задач.

Одним из ключевых компонентов в моделировании предметной области приложения для проведения настольных ролевых игр являются правила и механики игры. Правила определяют, как именно игроки взаимодействуют с миром и как их действия влияют на происходящее. Это включает систему бросков кубиков для определения успеха или неудачи действий, боевые механики, применение заклинаний или навыков. Моделирование правил требует разработки системы контроля и поддержки правил и механик, которая позволит ускорить расчеты. Например, система должна уметь рассчитывать результаты действий персонажей на основе их характеристик или результаты необходимого броска игральных костей.

Кроме того, в игре важную роль играют предметы и инвентарь персонажей. Эти элементы требуют моделирования, которое позволит игрокам управлять инвентарём своих персонажей, а также взаимодействовать с предметами, находящимися в мире. Приложение должно предоставлять удобный интерфейс для управления предметами, и инструменты для генерации новых предметов ведущим. При этом необходимо учитывать как количественные ограничения, так и уникальные характеристики предметов, которые могут влиять на игровой процесс.

После первичного изучения предметной области, выделения первичных сущностей и создания базовых обобщенных моделей начинается процесс углубленного моделирования взаимодействия сущностей и данных внутри изучаемой предметной области. Данный процесс называется логическим моделированием.

Логическое моделирование – это процесс представления структуры данных, который помогает разработчикам и аналитикам понять, как информация организована и связана в рамках системы. Данный тип моделирования направлен на создание абстрактной модели данных, которая отражает ключевые сущности предметной области, их атрибуты и

взаимосвязи. В отличие от физического моделирования, где акцент делается на конкретные технические детали хранения и обработки данных, логическое моделирование фокусируется на концептуальной структуре данных, не привязываясь к конкретным технологиям или платформам.

Основная цель логического моделирования – создать универсальную и понятную модель, которая будет описывать информацию таким образом, чтобы её можно было эффективно использовать при разработке. Процесс включает определение сущностей, таких как объекты, события или идеи, которые являются важными для предметной области. Эти сущности имеют атрибуты, которые описывают их характеристики. Например, если мы моделируем приложение для управления книгами в библиотеке, то сущностью будет «Книга», а её атрибутами – название, автор и год издания.

Логическое моделирование также включает определение взаимосвязей между сущностями. Такие связи показывают, как одни объекты системы зависят от других или взаимодействуют с ними. Например, если мы продолжаем модель библиотеки, «Книга» может быть связана с сущностью «Стеллаж» через отношение «Располагается», которое отображает, на каком стеллаже расположена данная книга. Такие связи помогают понять, как данные будут взаимодействовать друг с другом в системе.

Для визуализации результатов логического моделирования обычно используются диаграммы на языке UML (Unified Modeling Language). Первично происходит разработка диаграммы вариантов использования, которая представляет из себя графическое представление модели данных.

Диаграмма вариантов использования (use case) – диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Актор (актёр) – множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Актором может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности.

Прецедент – спецификация последовательностей действий в унифицированном языке моделирования, которые может осуществлять система, подсистема или класс, взаимодействуя с внешними действующими лицами.

Основное назначение диаграммы – описание функциональности и поведения, позволяющие заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую систему [15].

Диаграмма вариантов использования представлена на рисунке 2.1, в более удобном виде см. ГУИР.181071.001 ПЛ

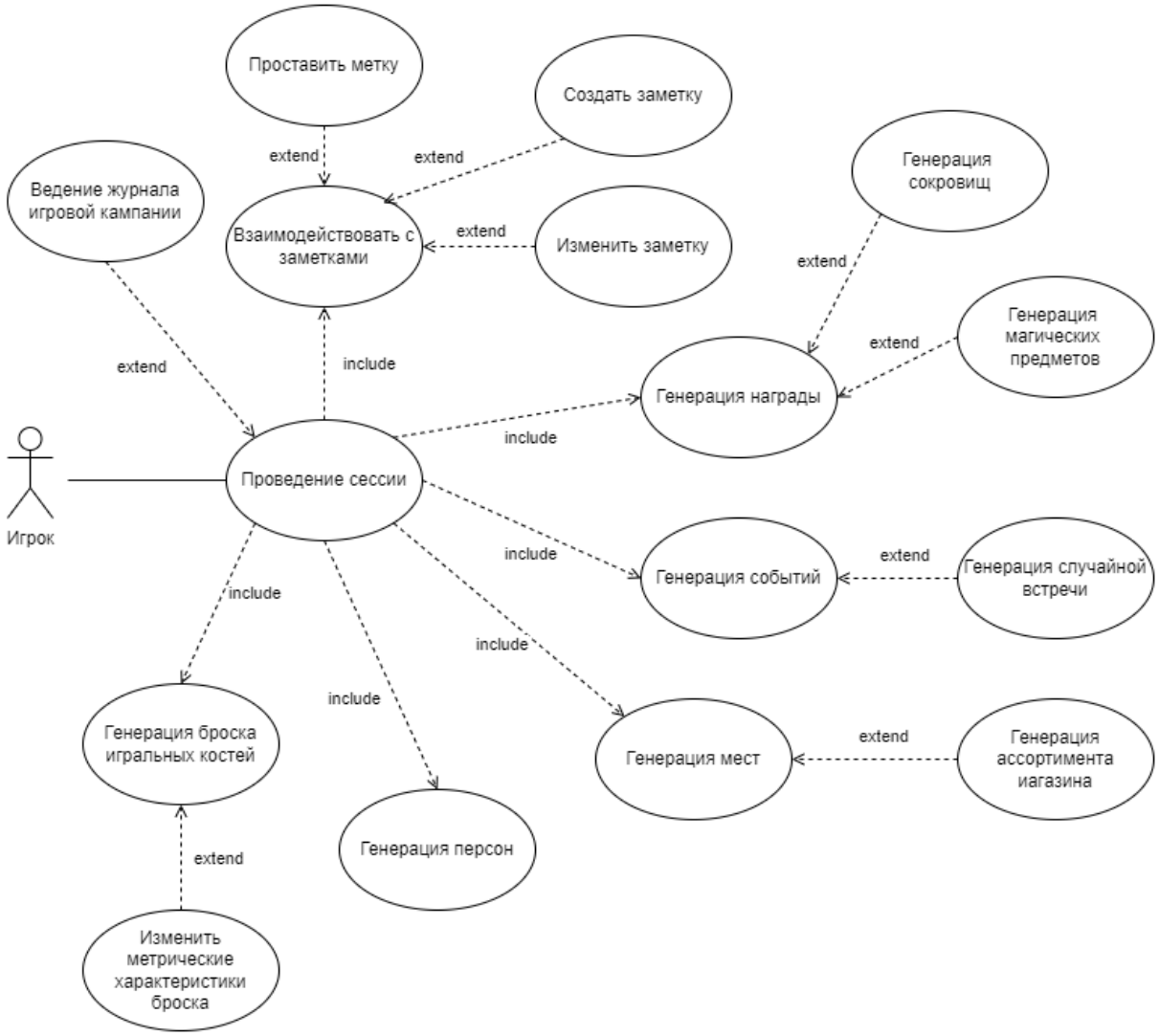

Рисунок 2.1 – Диаграмма вариантов использования

На диаграмме представлен один актор – «Игрок». «Игрок» связан с «Проведением сессии», так как основная цель приложения – это упростить игроку проведение игровой сессии. Данная связь является ключевой.

Выделенные прецеденты:

– «Проведение сессии» – представляет непосредственный процесс игровой партии и все необходимые манипуляции.

– «Ведение журнала игровой кампании» – представляет систему последовательных записей, распределенных в хронологическом порядке и предназначенных для хранения последовательности ключевых событий игрового сюжета.

– «Взаимодействовать с заметками» – система для записи чтения пометок игрока.

– «Проставить метку» – система отметки важных пометок игрока.

– «Создать заметку» – инструмент создания новой пометки.

– «Изменить заметку» – инструмент редактирования пометок игрока.

– «Генерация награды» – инструменты создания награды для персонажей игроков за выполненное задание или проведенный бой.

– «Генерация сокровищ» – инструменты для определения содержимого сокровищниц, кладов и тайников в игровом мире.

– «Генерация магических предметов» – инструменты определения особенностей магического или необычного предмета.

– «Генерация событий» – инструменты для создания случайного внутриигрового события.

– «Генерация случайной встречи» – инструмент для создания случайного взаимодействия персонажей игроков с нейтральными или враждебными неигровыми персонажами.

– «Генерация мест» – инструменты для создания описания нового случайного места в игровом мире.

– «Генерация ассортимента магазина» – инструмент для заполнения магазина в игровом мире случайными игровыми предметами.

– «Генерация персон» – инструменты для создания нового случайного неигрового персонажа.

– «Генерация броска игральных костей» – инструмент для расчета случайного броска двадцатигранной игральной кости.

– «Изменить метрические характеристики броска» – инструмент позволяющий настроить количество граней игральных костей и их общее количество в генерируемом броске.

Диаграмма деятельности (Activity Diagram) широко применяется для моделирования различных аспектов программных систем. Эта диаграмма используется для визуализации динамических аспектов системы, отражая поток работ и процессов, происходящих внутри неё. В отличие от диаграмм классов или объектов, которые показывают статическую структуру системы, диаграмма деятельности фокусируется на последовательности действий и их взаимосвязи, что делает её полезным инструментом при проектировании программного средства.

Диаграмма деятельности позволяет моделировать различные сценарии и процессы, происходящие как внутри разрабатываемого программного средства, так и на стыке взаимодействия с внешними элементами. Что позволяет легко рассматривать различные варианты выполнения задач, включая разветвления логики, параллельные процессы и точки

синхронизации. Таким образом диаграмма деятельности позволяет улучшить понимание проектируемого программного средства как среди разработчиков, так и среди бизнес-аналитиков, заказчиков и других заинтересованных сторон. Поскольку диаграмма наглядно показывает поток действий, включая точки принятия решений и условия перехода между этапами, она становится удобным инструментом для общения и согласования различных аспектов проектируемого программного средства. Благодаря чему снижается риск недопонимания между заказчиком проекта и исполнителями. [16]

Диаграмма деятельности позволяет выявить возможные узкие места, потенциальные точки отказа и уточнить требования к программному обеспечению. Моделирование потоков деятельности помогает разработчикам определить, как различные блоки системы должны взаимодействовать друг с другом и с внешними системами, таким образом можно создать более целостную согласованную архитектуру.

Часто возникают ситуации, когда несколько задач должны выполняться одновременно или асинхронно. Диаграммы деятельности позволяет моделировать процессы с параллельным выполнением, демонстрируя, как параллельные процессы взаимодействуют и как происходит их синхронизация на определенных этапах. Это помогает заранее учесть возможные сложности, связанные с параллелизмом, и спланировать механизмы синхронизации, блокировки или координации потоков.

Пошаговое изображение всех действий и их последовательности в разрабатываемом приложении позволяет легче увидеть логические несоответствия или недостатки в структуре системы, позволяя обнаружить и исправить логические ошибки в приложении ещё на этапе проектирования, не перенося их на этап программной реализации.

Диаграмма деятельности играет важную роль в создании тестовой документации и подготовке сценариев тестирования, так как отражает все возможные пути выполнения процессов. Диаграмма служит хорошей основой для разработки тестов, которые будут охватывать различные сценарии использования системы, включая как стандартные, так и нестандартные ситуации.

Применение диаграммы деятельности на этапе проектирования помогает избежать ошибок, выявить узкие места и создать эффективную архитектуру программной системы.

Разработанная для приложения диаграмма деятельности представлена на рисунке 2.2, в более удобном виде см. ГУИР.181071.002 ПЛ.

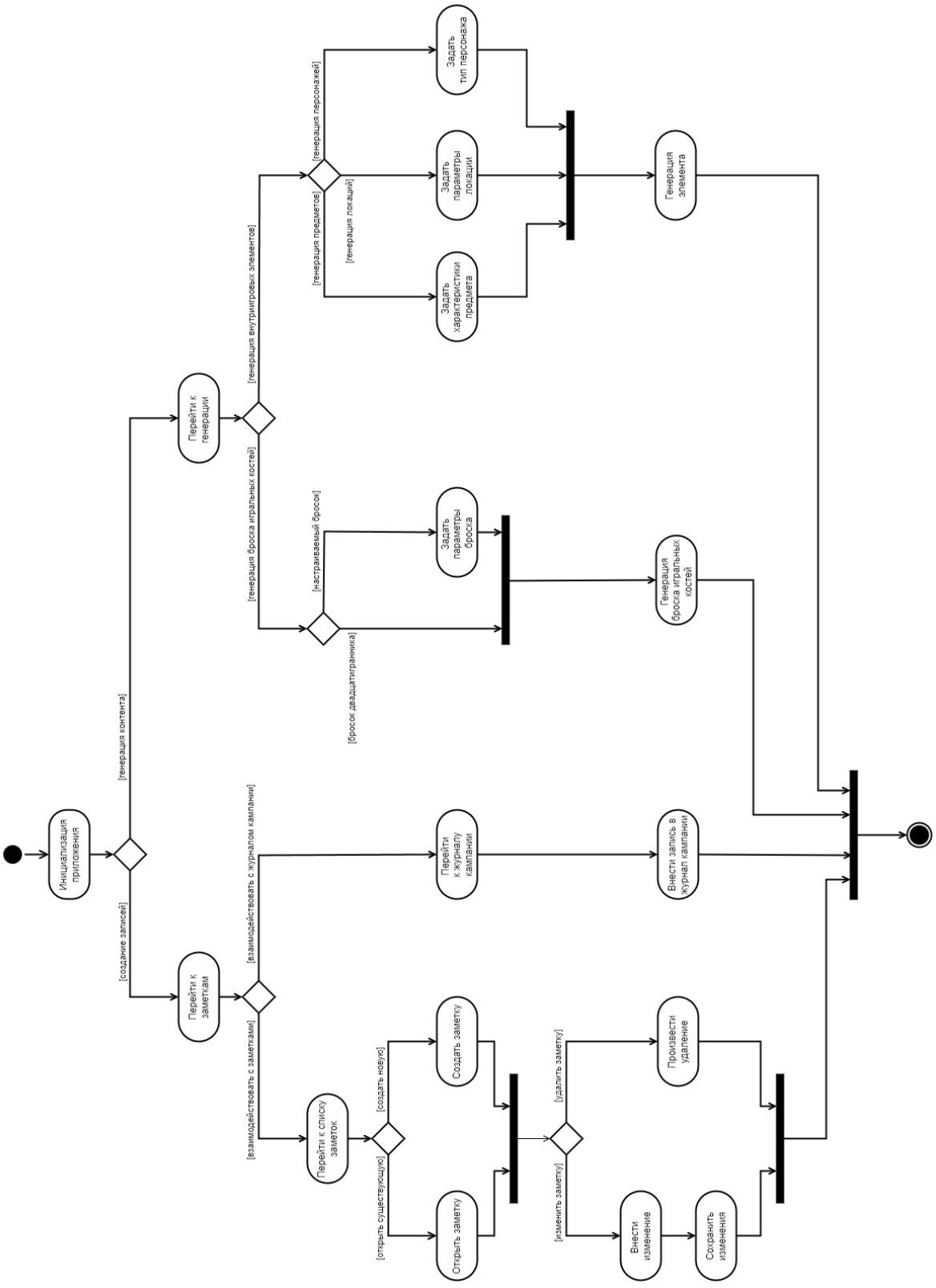

Рисунок 2.2 – Диаграмма деятельности

Изначально, на этапе запуска приложения происходит инициализация и загрузка всех необходимых компонентов. После инициализации, по запросу

пользователя, поток действия переходит либо к инициализации системы заметок, либо к инициализации генеративных инструментов.

Система заметок разделена на заметки и журнал кампании.

Переход к журналу кампании инициализирует окно журнала, после чего появляется возможность внести новую веху в журнал и закончить цикл использования.

Переход к заметкам позволяет отобразить список заметок, и пользователь выбирает либо открыть существующую заметку, либо создать новую заметку. В независимости от предыдущего действия поток действий переходит либо к процедуре удаления заметки, либо к процедуре редактирования заметки с последующим сохранением внесенных изменений. Затем заканчивается цикл использования приложения.

В случае, если пользователь после завершения инициализации приложения выбрал перейти к генеративным инструментам, происходит их инициализация с последующим выбором перехода либо к расчету результатов бросков игральных костей, либо к генерации внутриигровых элементов.

При переходе к расчету результатов бросков игральных костей пользователь производит генерацию броска стандартного двадцатигранника или переходит к настройкам параметров генерируемого броска (количеству игральных костей, количеству их ребер и модификатор, добавляемый к результату). В любом из случаев поток действия переходит к генерации броска и расчету результата после чего цикл использования приложения завершается.

При переходе потока действий к генерации внутриигровых элементов пользователь выбирает перейти к заданию требуемых характеристик к игровому предмету, или к заданию требований к локации, или к введению типа необходимого внутриигрового персонажа. После выбора и введения параметров поток действий переходит к генерации внутриигрового элемента, соответствующего заданным требованиям. По завершению генерации выводится результат и завершается цикл использования приложения.

По завершению моделирования предметной области необходимо выделить функциональные требования, предъявляемые к разрабатываемому программному средству.

Функциональные требования (functional requirements) – описание требуемого поведения системы в определенных условиях.

Функциональные требования определяют, каким должно быть поведение продукта в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи (пользовательские требования) в рамках бизнес-требований [17].

Мобильное приложение для проведения настольных ролевых игр предназначено для автоматизации и упрощения процесса проведения партий в настольные ролевые игры как для ведущих, так и для игроков.

Мобильное приложение должно предоставлять широкий спектр инструментов для автоматизации и оптимизации игрового процесса в настольные ролевые игры, иметь достаточно удобное управление и не перегруженную рабочую область.

Для оценивания готовности разрабатываемого приложения произведена разработка специализированных функциональных требований. Разработанная спецификация охватывает все главные функциональные требования, определенные на этапе анализа бизнес-процессов, моделирования предметной области и проектирования системы.

Спецификация функциональных требований к приложению для проведения настольных ролевых игр:

1. Журналирование – приложение должно иметь инструменты для создания цепочки последовательных записок пользователя, помеченных датой создания для ведения хронологически истории ключевых событий в игровой партии.

2. Журналирование броска игральной кости – приложение должно вести запись генераций броска игральных костей пользователем с указанием даты, так как в настольных ролевых играх бросок игральной кости является ключевой механикой и возможность восстановить значение предыдущего броска является критически необходимым.

3. Система заметок – пользователю должны быть предоставлены инструменты для ведения собственных произвольных заметок, так как многие кампании в настольные ролевые игры могут растягиваться на множество партий и большой промежуток времени из-за чего пользователю необходимо иметь возможность хранить пометки по ходу кампании.

4. Генератор бросков игральной кости – бросок игральной кости играет ключевое значение в системе настольных ролевых игр поэтому приложение должно предоставлять пользователю удобный инструмент для генерации необходимого броска игральной кости с функцией настройки параметров броска.

5. Система контроля персонажа – пользователю необходима возможность помечать характеристики игрового персонажа и их изменения в процесс партии в настольную ролевую игру.

6. Система генерации неигровых персонажей – для проведения партии ведущему необходимо большое количество случайных неигровых персонажей

поэтому приложение должно предоставить инструменты позволяющие автоматизировать процесс их генерации.

7. Система генерации мест – для проведения партии в настольную ролевую игру ведущему необходимо описывать разнообразные внутриигровые места, где происходят события игры поэтому приложение должно предоставить инструменты для создания типовых случайных внутриигровых мест.

8. Система генерации событий – во время парти внутри игры происходят типовые случайные события, которые надо создавать ведущему. Для оптимизации и разгрузки задач ведущего приложение должно предоставлять удобный инструмент позволяющий легко и быстро сгенерировать типовое случайное событие.

9. Система генерации награды – во время партии ведущему необходимо награждать персонажей игроков разнообразными сокровищами, для этого ведущему надо регулярно создавать сокровища, предметы и артефакты. Приложение должно предоставить инструменты автоматизации данного процесса.

Данные функциональные требования обеспечивают основной функционал приложения для проведения настольных ролевых игр, и позволяет четко определить готовность и качество разрабатываемого приложения.

3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

3.1 Проектирование и разработка алгоритмов приложения

Проектирование и разработка алгоритмов работы приложения для проведения настольных ролевых игр представляет из себя ключевой этапам создания качественного программного продукта. В основе любого приложения лежит сложная логика, которая должна обеспечивать корректное взаимодействие между пользователями, системой и предоставляемым функционалом.

Проектирование алгоритмов обеспечивает достижение поставленных задач, проектируя и моделируя структуру, которая контролирует все процессы в приложении. [18]

Изначально был разработан общий алгоритм работы приложения (см. рисунок 3.1 или ГУИР.181071.003 СА).

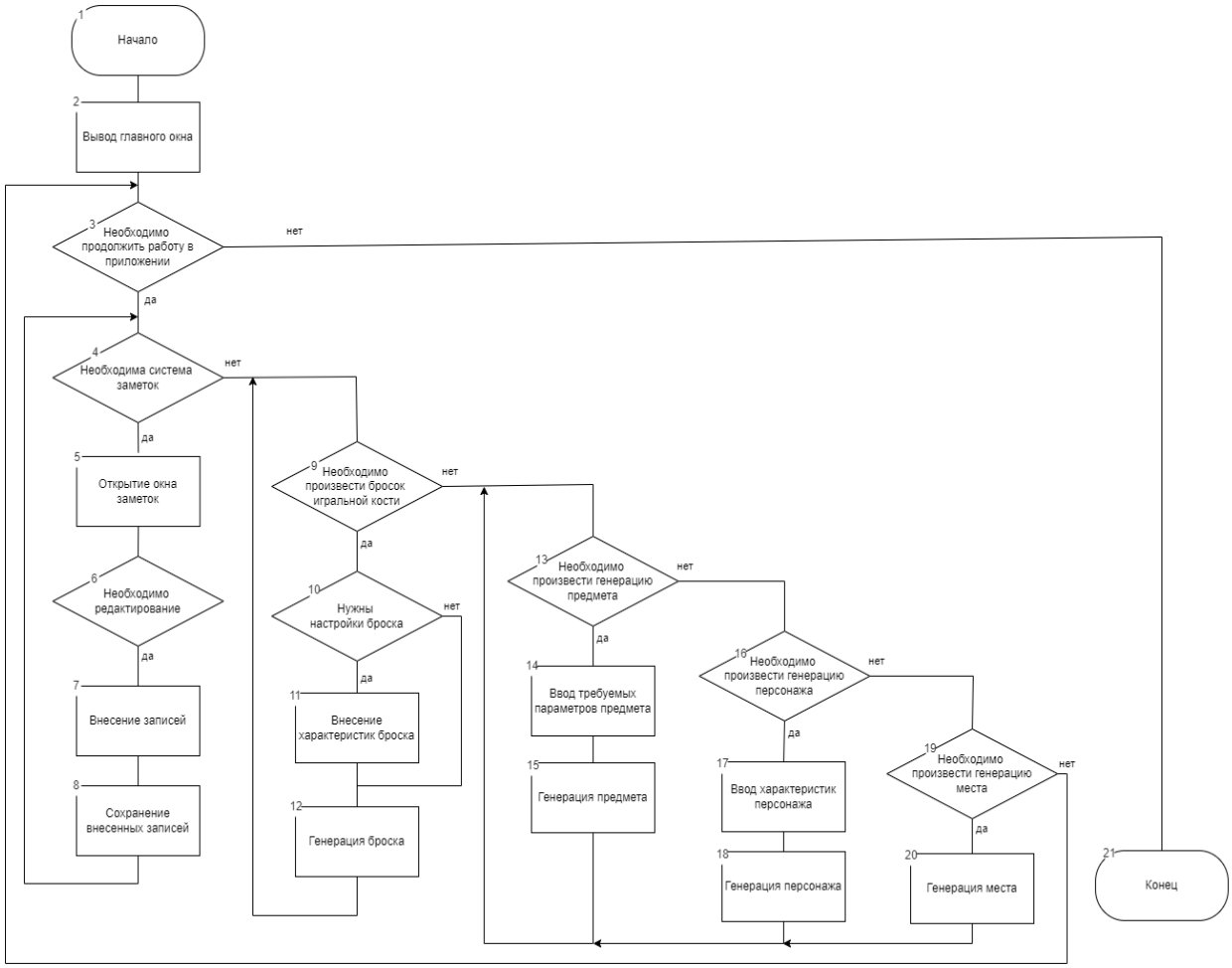

Рисунок 3.1 – Общий алгоритм работы приложения

Данный алгоритм позволяет получить общее представление о последовательности выполняемых в приложении операций и общей логике работы приложения. Изначально при запуске мобильного приложения для проведения настольных ролевых игр происходит инициализация компонентов и отображение главного окна.

Инструменты приложения разбиты на два раздела – система «Заметок и журналов» и «Генераторы внутриигровых элементов». После инициализации вывода главного окна приложения ожидает выбора группы функциональности пользователем.

При переходе к системе заметок и журналирования приложение ожидает какой из инструментов выберет пользователь, систему заметок или систему журналирования.

При выборе системы заметок отобразится окно содержащие список существующих заметок и кнопку добавить. Пользователь может выбрать и необходимую заметку и прочитать или изменить, также нажатием кнопки добавить пользователь может создать новую заметку.

При выборе системы журналирования отобразится журнал игровой кампании. Журнал представляет из себя последовательный набор записей о важных вехах и события игроков в кампании. В журнал можно вносить новые записи.

Если пользователь переходит к генераторам внутриигровых элементов, то отображается окно со списком доступных генераторов. Пользователю необходимо выбрать необходимый генератор.

Если необходимо сгенерировать бросок игральных костей тол пользователю надо выбрать «Генератор броска игральных костей» ввести параметры броска и рассчитать результат.

При необходимости в создании внутриигровых предметов пользователю надо выбрать «Генератор внутриигровых предметов» ввести желаемые параметры предмета и сгенерировать новый предмет.

Когда необходимо создать нового случайного неигрового персонажа игроку необходимы перейти к «Генератору персонажей» ввести желаемые характеристики неигрового персонажа и сгенерировать его.

А если игроку надо сгенерировать случайное место, то необходимо выбрать «Генератор локаций» и сгенерировать описание случайного внутриигрового места.

Теперь обладая общим представлением о логике работы приложения можно перейти к проектированию отдельных функциональности приложения. Данное проектирование необходимо для получения понимания о принципах

работы каждого отдельного элемента что позволит упростить процесс разработки данных элементов и избежать возможных ошибок.

В первую очередь был спроектирован алгоритм работы модуля заметок и журнала (см. рисунок 3.2 или ГУИР.181071.004 СА).

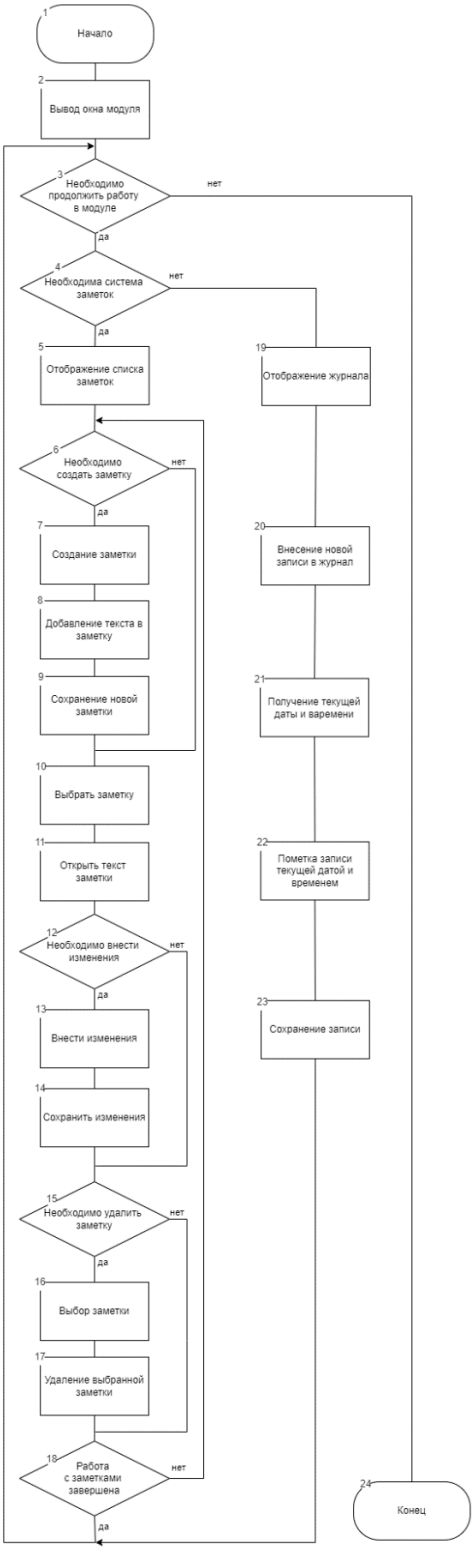

Рисунок 3.2 – Алгоритм работы модуля заметок и журнала

Первоначально происходит инициализация окна модуля. После если необходимо продолжить работу модуля, то пользователь переходит к выбору между журналом кампании и заметками.

При переходе к журналу компании пользователю отобразится список последовательных записей журнала, и он сможет добавить новую запись, которая будет помечена датой и временем своего создания.

При переходе к системе заметок приложение отображает список всех существующих заметок.

Если пользователю необходимо создать новую заметку ему необходима нажать кнопку «Создать» ввести текст заметки и нажать «Ок». Затем приложение сохранит созданную заметку и отобразит список всех заметок с добавлением только что созданной пользователем заметки.

В окне со списком заметок пользователь может выбрать заметку нажав по ней. Выбранная заметка «раскроется» и пользователь получит возможность прочесть текст заметки или при необходимости внести изменения.

В списке заметок через функционал «удаление» при необходимости можно произвести удаление не нужных заметок.

Следующим были разработаны алгоритмы генераторов внутриигровых элементов (общий алгоритм модуля см. ГУИР.181071.006 СА). Алгоритм работы генератора внутриигровых мест представлен на рисунке 3.3.

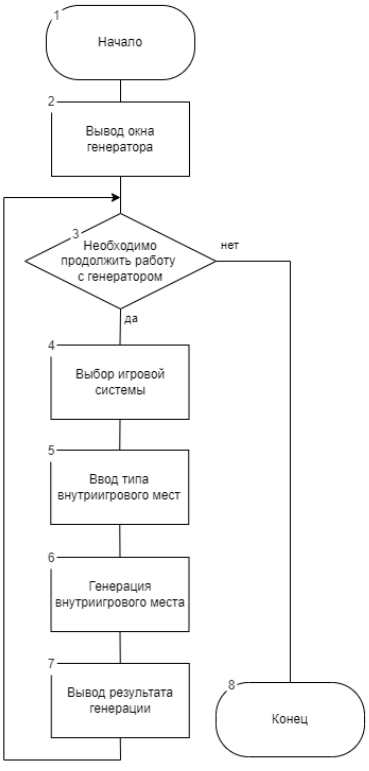

Рисунок 3.3 – Алгоритм работы генератора внутриигровых мест

Пользователь заходит в генератор внутриигрового места и выбирает желаемую ролевую систему. После чего пользователю необходимо ввести тип генерируемого места. После выбора типа генерируемого места и нажатия кнопки «Сгенерировать» приложение произведет генерацию и выведет результат. Если продолжать генерацию нет необходимости, то пользователь закрывает генератор.

На рисунке 3.4 рассмотрен алгоритм работы генератора неигровых персонажей.

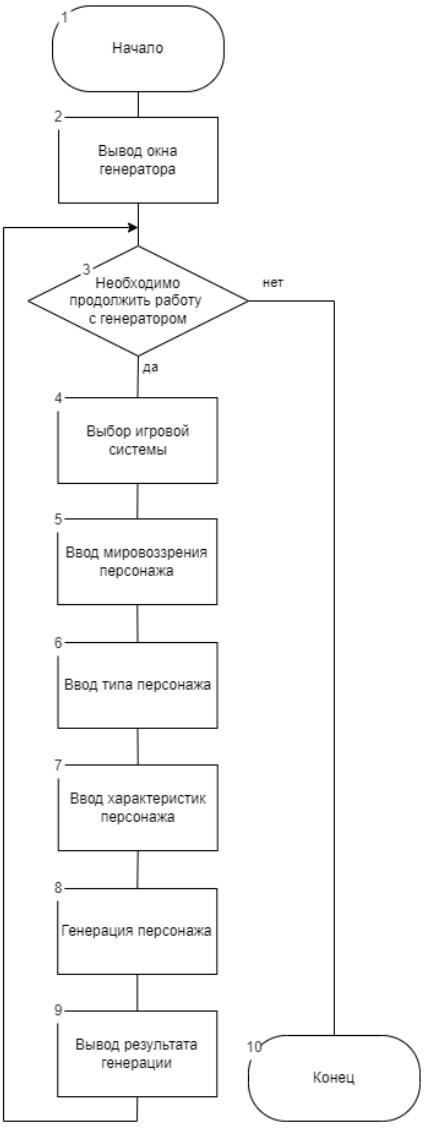

Рисунок 3.4 – Алгоритм работы генератора неигровых персонажей

После инициализации генератора необходимо выбрать ролевую систему для генерации неигрового персонажа, после производится ввод мировоззрения персонажа (враждебен к персонажам игроков или нет), его типа (например

торговец или наемник), его характеристик. Когда все параметры введены можно произвести генерацию, результаты которой будут сразу же выведены пользователю. Если продолжать генерацию нет необходимости, то пользователь закрывает генератор. Схема алгоритма работы генератора предметов изображена на рисунке 3.5.

Рисунок 3.5. – Алгоритм работы генератора предметов

Изначально происходит инициализация генератора. После пользователь выбирает игровую систему, вводит тип предмета (например щит), вводит редкость – характеристику, отвечающую за качество предмета. После ввода параметров происходит генерация предмета по заданным характеристикам с последующим выводом результата.

Последним был разработан алгоритм для самого важного генератора – генератора бросков игральных костей. Разработанный алгоритм представлен на рисунке 3.6 и в ГУИР.181071.005 СА.

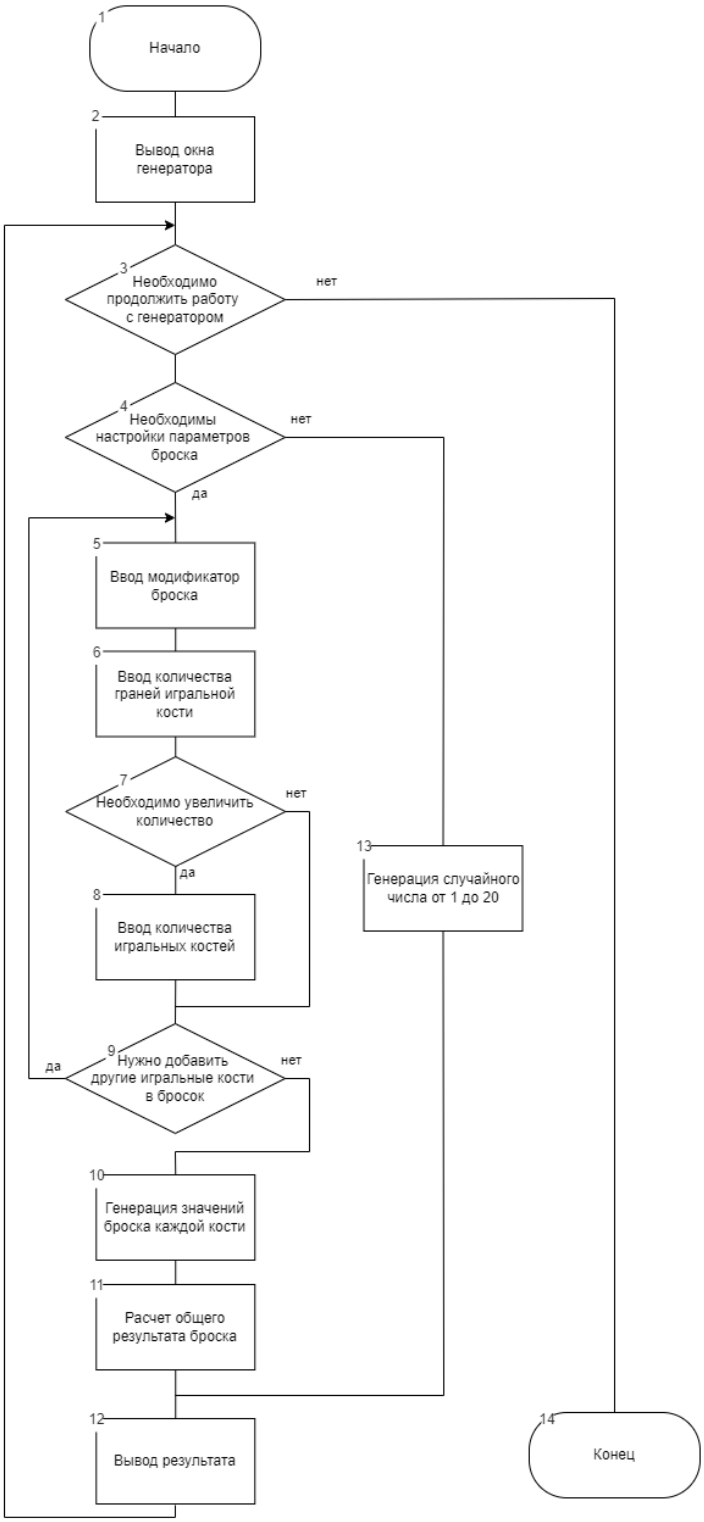

Рисунок 3.6 – Алгоритм работы генератора броска игральных костей

После выбора нужного генератора происходит инициализация окна генератора. Если нет необходимости в нестандартном броске, то пользователь сразу же нажимает «Сгенерировать» и программа сразу же сгенерирует бросок двадцатигранной игральной кости. Если пользователю необходимо настраиваемый бросок, то необходимо внести параметры модификатора броска (значения, на которое будет уменьшен или увеличен результат броска). Затем пользователь вводит количество граней игральной кости, после чего при необходимости количество самих костей. Когда ввод параметров завершен пользователь нажимает «Сгенерировать», по нажатию приложение производит генерацию значений костей и производит математические операции учитывая модификатор. После выполнения расчетов приложение выводит результат броска пользователю.

Проектирование алгоритмов работы приложения позволяет структурированно подходит к непосредственной реализации приложения. Это позволяет создать логически выверенную основу, которая гарантирует стабильную работу приложения. Спроектированные алгоритмы дают возможность заранее учесть различные сценарии использования, что снижает вероятность ошибок и сбоев. Это помогает лучше понять системные требования и сделать архитектуру приложения более качественной что в свою очередь упрощает поддержку и развитие приложения, уменьшает трудоёмкими, дальнейшее улучшение функционала может происходить с минимальными рисками для его работы.

3.2 Проектирование интерфейса приложения

Проектирование интерфейса является важным шагом, который позволяет заложить основы для создания качественного продукта.

Интерфейс – это мост между пользователем и функционалом приложения. Именно через него происходит взаимодействие с программой, и его качество во многом определяет успех приложения на рынке. Если интерфейс продуман правильно, пользователи понимают, как использовать приложение, и могут спокойно выполнять необходимые задачи, что сразу же существенно повышает привлекательность продукта на рынке.

Прежде чем переходить к технической реализации, важно глубоко понять потребности и ожидания целевой аудитории, изучит существующие аналоги и общие тенденции на рынке. Понимание позволяет создать качественный интерфейс, который будет удобным и для конечных пользователей. Если этап проектирования интерфейса пропустить или проработать поверхностно, конечный продукт может оказаться неудобным,

что, в свою очередь, приведёт к негативному восприятию пользователей и низкой оценке приложения.

Проектирование интерфейса позволяет визуализировать структуру приложения, что помогает понять, как пользователи будут перемещаться между различными экранами и разделами, какие элементы будут важными, а какие можно сократить или упростить. Это снижает риск того, что уже в процессе разработки возникнут ситуации, когда потребуется енять логику работы приложения, что может привести к потерям времени и ресурсов.

Проектирование интерфейса позволяет протестировать взаимодействие с приложением ещё до его создания. Разработка прототипов и макетов даёт возможность выявить потенциальные проблемы в пользовательском опыте на ранних стадиях. Благодаря этому можно внести изменения в структуру или логику интерфейса до того, как разработка зайдёт слишком далеко. Это помогает не только сэкономить ресурсы и улучшить качество продукта. [19]

Проектирование интерфейса перед разработкой мобильного приложения – это важный шаг, который обеспечивает более эффективный процесс создания итогового продукта. Данный процесс снижает риски возникновения серьёзных ошибок на поздних этапах разработки и помогает создать удобное, функциональное и востребованное мобильное приложение. Проектирование интерфейса перед программированием позволяет разработчику избежать множества проблем и повысить общий уровень качества конечного продукта.

Проанализировав существующие аналоги и общие концепции разработки и оформления мобильных приложений были разработаны требования к интерфейсу разрабатываемого мобильного приложения для проведения настольных ролевых игр.

Для интерфейса мобильного приложения для проведения настольных ролевых игр предъявляются следующие требования:

1. Компактность. В связи с тем, что целевая платформа мобильные устройства, а экраны у данного типа устройств относительно небольшие для разрабатываемого приложения особенно важно чтобы интерфейс мог хорошо уместится в малом пространстве.

2. Отсутствие перегруженности. Рабочая область не должна быть переполнена кнопками, ползунками и прочими элементами управления. Пользователь должен легко найти нужную кнопку, а не перебирать десятки элементов в поисках нужного.

3. Управление одной рукой. Зачастую пользователи мобильных телефонов управляют приложениями одно рукой, поэтому интерфейс

приложения должен позволять управляться с функционалом приложения только одной рукой.

После составления требований к интерфейсу приложения была проведена его разработка.

На рисунке 3.7 изображен макет интерфейса системы заметок.

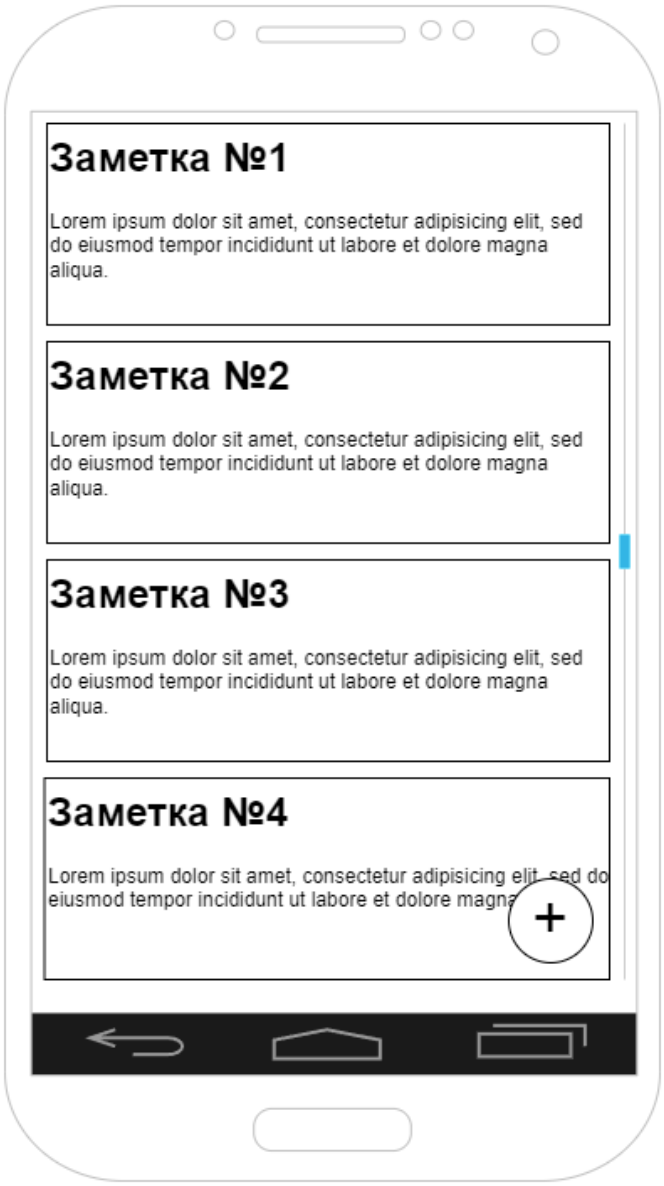

Рисунок 3.7 – Макет интерфейса системы заметок

Окно системы заметок будет состоять из заметок, кнопки «+» и скролл бара. По нажатию на кнопку добавить будет создаваться новая заметка. По нажатию на заметку откроется окно заметки для прочтения текста.

На рисунке 3.8 представлен макет интерфейса окна генерации бросков игральных костей.

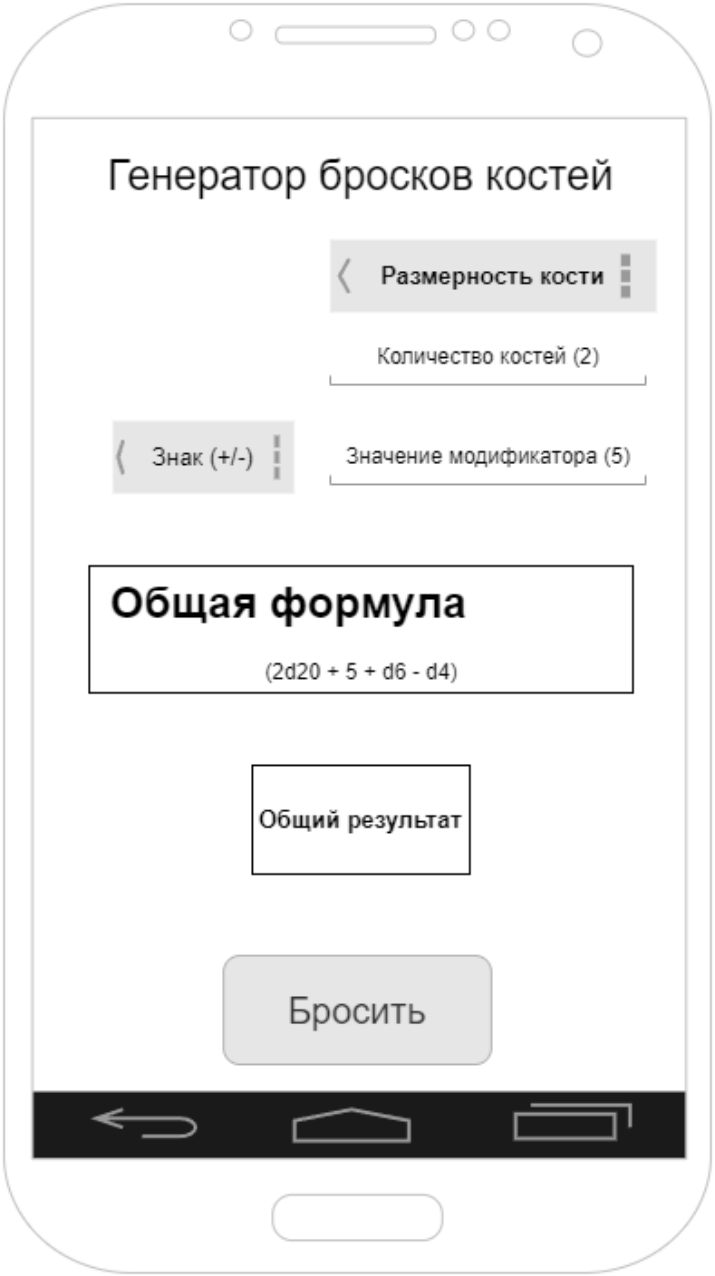

Рисунок 3.8 – Макет интерфейса генератора бросков игральных костей

Для удобства элементы данного макета и последующих макетов будут рассматриваться сверху-вниз.

Первым элементом является меню выбора размерности (количества граней) игральной кости. Следом пользователь выбирает количество игральных костей. После выбирается знак модификатора и значение модификатора. После находится поле, где выводится общая формула калькуляции броска.

Ниже поля калькуляции расположено поле общего итога, где отображается результат броска.

В самом низу расположена кнопка сгенерировать которая отвечает за начало расчета броска.

На рисунке 3.9 изображен макет генератора внутриигровых персонажей.

Рисунок 3.9 – Макет интерфейса генератора внутриигровых персонажей

Первое меню позволяет пользователю выбрать ролевую систему. Второе меню позволяет выбрать мировоззрение для генерируемого персонажа. Третье меню позволяет определить тип, к которому будет относится персонаж (например торговец).

Под тремя меню расположен блок, где выводятся итоги генерации персонажа, а именно имя персонажа и его описание.

В самом низу расположена кнопка сгенерировать которая отвечает за начало расчета броска.

На рисунке 3.10 изображен макет окна генератора внутриигровых предметов.

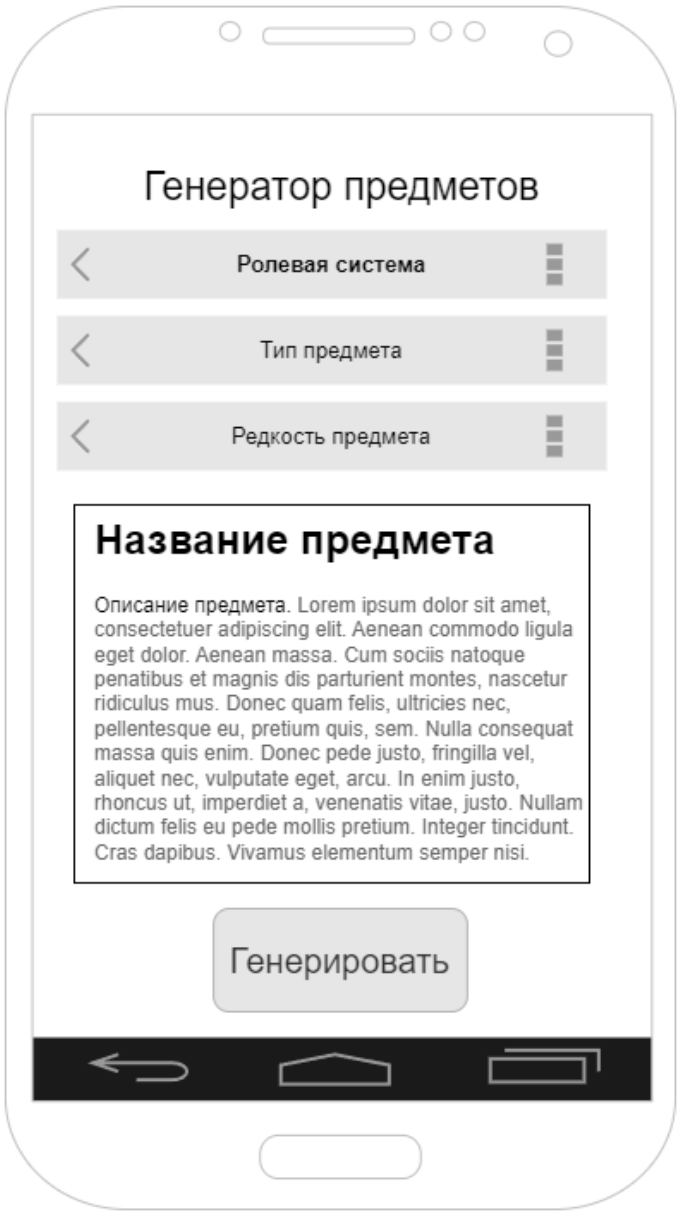

Рисунок 3.10 – Макет интерфейса генератора внутриигровых предметов

Первое меню позволяет пользователю выбрать ролевую систему. Второе меню позволяет выбрать тип предмета (например посох). Третье меню позволяет определить редкость предмета.

Под тремя меню расположен блок, где выводятся итоги генерации предмета – название предмета и описание его способностей.

В самом низу расположена кнопка «Генерировать» которая отвечает за запуск генерации предмета.

На рисунке 3.11 изображен макет генератора внутриигровых мест (локаций).

Рисунок 3.11 – Макет интерфейса генератора внутриигровых мест (локаций)

Сверху расположено меню выбора ролевой системы. Под меня выбора расположено поле для вывода результата генерации в виде названия локации и его краткого описания. В самом низу расположена кнопка «Генерировать» которая отвечает за запуск генерации новой локации.

Проектирование интерфейса играет ключевую роль в успешной разработке программного обеспечения, закладывая фундамент для дальнейших этапов работы. Этот процесс помогает создать чёткую и структурированную основу, которая значительно облегчает реализацию функциональности. Это позволяет избежать ошибок, которые могли бы возникнуть при реализации программного обеспечения без чёткого видения конечного продукта. Проектирование интерфейса позволяет выявить потенциальные проблемы или сложности на ранней стадии и внести необходимые изменения ещё до начала программирования, когда это не требует значительных затрат времени и ресурсов.

3.3 Описание разработанных модулей

Для приложение спроектировано три программных модуля. Код спроектированных модулей представлен в приложении А.

Спроектированные модули приложения:

1. Модуль заметок – состоит из двух ключевых MAUI окон и трех классов. Отвечает за систему заметок в приложении.
2. Модуль журнала – состоит из одного MAUI окна и одного класса. Отвечает за систему журналирования.
3. Модуль генеративных инструментов – состоит из пяти MAUI окон и пяти классов. Отвечает за систему генераторов внутриигровых элементов.

Кроме трех модулей стоит выделить основное окно программы и специальный класс, реализующий основную логику главного окна приложения.

4 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

4.1 Выбор и обоснование видов тестирования

Тестирование приложения – важный этап разработки представляющий из себя процесс исследования, испытания продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом. Он направлено на выявление и устранение ошибок, которые могут повлиять на работу приложения и охватывает множество аспектов, начиная от функциональности и производительности и заканчивая удобством интерфейса.

Основная задача тестирования – выявление ошибок и несовершенств, в работе приложения.

Тестирование включает несколько направлений, каждое из которых имеет свои задачи:

– Функциональное тестирование проверяет, выполняет ли приложение все заявленные функции. Оно охватывает основные сценарии использования, такие как регистрация, авторизация, выполнение операций, взаимодействие с интерфейсом и корректность обработки ошибок, возникающих при неверном вводе данных или непредвиденных действиях.

– Регрессионное тестирование проверяет внесённые изменения в код приложения на совместимость с существующим функционалом. Помогает убедиться, что обновления или исправления не повлияли на уже работающие функции. Это критически важно в условиях регулярных обновлений, когда каждое новое изменение должно быть протестировано на предмет совместимости с существующим функционалом.

– Тестированию пользовательского интерфейса (UI) и взаимодействия с пользователем (UX). Проверяет отображение элементов интерфейса приложения и корректность их расположения. Позволяет выявить «артефакты» в интерфейсе и неудобную компоновку элементов.

– Тестирование производительности оценивает, как приложение работает в условиях нагрузок. Для этого анализируется скорость отклика, использование системных ресурсов и общая стабильность. Такие проверки позволяют выявить возможные узкие места, способные замедлить работу программы или вызвать её сбой.

Тестирования базируются на систематичности и цикличности. Каждый этап разработки сопровождается проверкой, которая охватывает как технические аспекты, так и графические. Важно тестировать не только отдельные компоненты, но и их взаимодействие в рамках целого

приложения. Регулярность тестирования позволяет своевременно обнаруживать проблемы и устранять их сразу же в момент появления.

Используя соответствующее тестирование на каждом этапе разработке, можно достаточно просто избежать множества ошибок и глобальных проблем в разрабатываемого приложении, что значительно сэкономит время.

Кроме того, после завершения разработки необходимо провести итоговое тестирование готового к релизу продукта для выявления неявных ошибок. Проверка всей системы в целом позволяет выявить ошибки, происходящие при комплексном взаимодействии всех модулей приложения.

Для тестирования приложения разработаны чек-лист и тест-кейсы. Чек-лист и тест-кейс важными и полезными инструментами тестирования, которые помогают упорядочить процесс проверки.

Чек-лист (Check List) – это список, содержащий ряд необходимых проверок во время тестирования программного продукта. Отмечая пункты списка, команда или один тестировщик могут узнать о текущем состоянии выполненной работы и о качестве продукта. Это простой и компактный инструмент, содержащий ключевые аспекты, на которые следует обратить внимание.

Чек-листы полезны своей лаконичностью, позволяя быстро охватить основные задачи и отследить, что уже было выполнено, а что ещё требует проверки.

Работая над проектом по чек-листу, исключена вероятность повторной проверки по тем же кейсам, а также повышается качество тестирования, так как вероятность оставить без внимания какой-то функционал существенно снижается.

Тест-кейс (Test Case), напротив, представляет из себя более детализированный документ, описывающий конкретный сценарий тестирования. Он включает описание начальных условий, пошаговую последовательность действий и ожидаемый результат.

Тест-кейсы, собранные в последовательность для достижения некоторой цели, образуют набор тестов (Test suite).

Тест-кейсы разделяются по ожидаемому результату на позитивные и негативные.

Позитивный тест-кейс использует только корректные данные и проверяет, что приложение правильно выполнило вызываемую функцию.

Негативный тест-кейс оперирует как корректными, так и некорректными данными (минимум 1 некорректный параметр) и ставит целью проверку исключительных ситуаций (срабатывание валидаторов), а также проверяет, что вызываемая приложением функция не выполняется при срабатывании валидатора.

Чек-лист и тест-кейс дополняют друг друга и являются удобными инструментами тестирования, которые позволяют охватить и структурировать весь процесс тестирования приложения при этом разделив его на последовательные легко обрабатываемые блоки тестов.

4.2 Результаты тестирования

Тестирование программы проводилось в процесс разработки и по завершению разработки приложения было проведено итоговое тестирование.

Для разрабатываемого приложения был создан чек-лист, представленный в таблице 4.1.

Таблица 4.1 – Чек-лист

Корректность работы главного окна
Корректность работы модуля журнала
Корректность работы модуля заметок
Корректность работы генератора бросков игральных костей
Корректность работы генераторов внутриигровых элементов

На основании разработанного чек-листа были разработаны тест-кейсы для каждого модуля приложения.

В таблице 4.2 представлен тест-кейс для главного окна приложения.

Таблица 4.2 – Тест-кейс для главного окна приложения

Тест	Ожидаемый результат	Результат
Корректность работы главного окна 1. Запустить программу 2. Проверить корректность отображение элементов управления 3. Проверить корректность адресации кнопок меню	1. Открывается главное меню. 2. Отображаются кнопки вызова модулей приложения. 3. При нажатии на кнопку: – «Журнал» открывается модуль журналов, – «Заметки» открывается модуль заметок, – «Генераторы» открывается модуль генерации элементов и бросков игральных костей, – «Лист персонажа» открывается модуль редактирования листа персонажа.	Успех

После успешного тестирования главного окна приложения был проведено тестирование модуля журналов. Тест-кейс для модуля журнала представлен в таблице 4.3.

Таблица 4.3 – Тест-кейс для модуля журналов

Тест	Ожидаемый результат	Результат
Корректность работы модуля журнала 1. Перейти в модуль журнала 2. Проверить корректность отображение элементов управления 3. Проверить загрузку сохраненных записей 4. Удалить запись 5. Создать новую запись	1. Открывается модуль журнала 2. Корректно отображаются следующие элементы: – поле для ввода текста новой записи в журнале, – кнопка, добавляющая новую запись в журнал, – список существующих записей в журнале. 3. При открытии модуля все ранее созданные записи корректно отображаются в виде списка. 4. Запись удалена и больше не отображается при последующих возвращениях в модуль. 5. После нажатия кнопки «Сохранить запись» появилась новая запись с пометкой даты и времени создания, которая сохраняется при выходе из модуля и отображается после возвращения в модуль вместе с остальными записями журнала в виде списка.	Успех

Следующим был разработан тест-кейс для модуля заметок (см. таблица 4.4). И при помощи разработанного тест-кейса был протестирован модуль.

Таблица 4.4 – Тест-кейс для модуля заметок

Тест	Ожидаемый результат	Результат
Корректность работы модуля заметок 1. Перейти в модуль заметок 2. Проверить корректность отображение элементов управления 3. Создать новую заметку 4. Изменить существующую заметку 5. Удалить существующую заметку	1. Открывается модуль заметок. 2. Корректно отображаются следующие элементы: – кнопка для создания заметки, – поле для ввода ключевых слов для поиска заметки, – список заметок. 3. При нажатии на кнопку «Добавить заметку» отображается интерфейс редактора заметок. При внесении параметров и нажатии кнопки сохранить создается новая заметка, которая сохраняется после закрытий модуля и отображается в общем списке заметок. 4. При нажатии на заметку отображается интерфейс редактора заметок. При внесении изменений в заметку через интерфейс изменения сразу же применяются к заметке. 5. В интерфейсе изменения заметки при нажатии на кнопку удалить заметка удаляется с устройства и не отображается в списке заметок.	Успех

После был разработан тест-кейс для тестирования генератора бросков игральных костей. Разработанный тест-кейс представлен в таблице 4.5.

Таблица 4.5 – Тест-кейс для генератора бросков игральных костей

Тест	Ожидаемый результат	Результат
Корректность генератора бросков 1. Перейти в генератор бросков 2. Проверить корректность отображение элементов управления 3. Сгенерировать бросок игральных костей 4. Отчистить историю бросков	1. Открывается генератор бросков игральных костей. 2. Корректно отображаются следующие элементы управления: – поле для ввода комбинации броска, – кнопка «Бросить кубики», – кнопка «Отчистить историю», – поле для вывода результатов броска, – список, содержащий историю бросков. 3. После ввода в специализированное поле комбинации броска (например 4d6 + d2+ 3) и последующем нажатии кнопки «Бросить кубики» в поле для результата броска должно отобразится полученное значение, а в списке с историей бросков появится новая строка, содержащая результат сгенерированного броска. 4. При нажатии на кнопку «Отчистить историю» список, содержащий историю последних десяти бросков, полностью отчищается от всего содержимого.	Успех

Для проведения тестирования инструментов генерации внутриигровых элементов в приложении был разработан универсальный тест-кейс для генераторов. Данный тест-кейс представлен в таблице 4.6.

Таблица 4.6 – Тест-кейс для генераторов внутриигровых элементов

Тест	Ожидаемый результат	Результат
Корректность генератора бросков 1. Перейти в тестируемый генератор 2. Проверить корректность отображение элементов управления тестируемого генератора 3. Произвести генерацию	1. Открывается генератор. 2. Корректно отображаются элементы управления генератора: – поле для выбора ролевой системы, – поля для настройки конкретных параметров генерации, – текстовое поле для вывода результатов генерации внутриигровых элементов, – кнопка «Сгенерировать» отвечающая за запуск процесса генерации с последующим выводом результата, – кнопка «Отчистить» производящая сброс введенных в соответствующие поля параметров генерации. 3. Выбрать ролевую систему, затем настроить параметры генерации и нажать кнопку «Сгенерировать». В поле для вывода должен появится результат генерации в виде текста с заголовком.	Успех

В ходе тестирования использовались разработанные тест-кейсы, которые помогли выявить ряд ошибок в работе приложения. Выявленные ошибки были исправлены сразу же после обнаружения.

4.3 Вывод тестирования

В процессе разработки и создания приложения важно не только реализовать необходимый спектр функций, но и сделать это качественно. Приложение должно выполнять поставленные перед ним задачи безукоризненно и полно. Для достижения этой цели в процессе разработки по завершению разработки были проведены этапы тестирования.

В ходе тестирования были разработаны решения и методы позволяющие корректно отследить ошибки, проблемы, недочеты и неточности в разрабатываемому приложении.

Тестирования отдельных модулей производилось во время их разработки что позволило выявить ошибки на ранних этапах. По завершению разработки было проведено комплексное тестирование всей системы и ее элементов что позволило выявить ряд интеграционных ошибок и «отшлифовать» модули приложения.

В ходе тестирования был выявлен и исправлен ряд ошибок:

– Неправильная вложенность элементов управления в модуле журнала. При создании окна работы с журналом была допущена ошибка и вместо создания записей в области скроллинга каждая запись создавалась как отдельная область скроллинга и находилась в общем статичном контейнере из-за чего не работал скроллинг.

– Не корректный интерфейс генератора бросков игральных костей. На раннем этапе разработки генератора для удобства разработки оформление окна было временно сделано через связанный класс позже, когда для приложения задавался общий интерфейс данное оформление перекрывала общие правила интерфейса что вызвало некорректное отображение страницы.

– Ошибка генерации внутриигровых элементов. При генерации элемент, подготавливаемый для вывода, передавался не в виде набора значений, а в виде коллекции с одним элементом с нужными полями из-за чего значения не выводились в поле результата.

По завершению всех этапов тестирования и исправления всех выявленных ошибок было заключено о готовности приложения к релизу.

5 РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ ПРИЛОЖЕНИЯ

При запуске приложения открывается главное меню приложения. Меню содержит 4 кнопки каждая отвечает за переход в соответствующий модуль приложения.

– Кнопка «Журналы» открывает модуль журнала.

– Кнопка «Заметки» открывает модуль заметок.

– Кнопка «Генераторы» открывает модуль инструментов генерации

– Кнопка «Лист персонажа» открывает модуль хранения данных о персонаже игрока.

Далее в инструкции модули будут рассматриваться в приведенном выше порядке.

Главное меню отображено на рисунке 5.1.

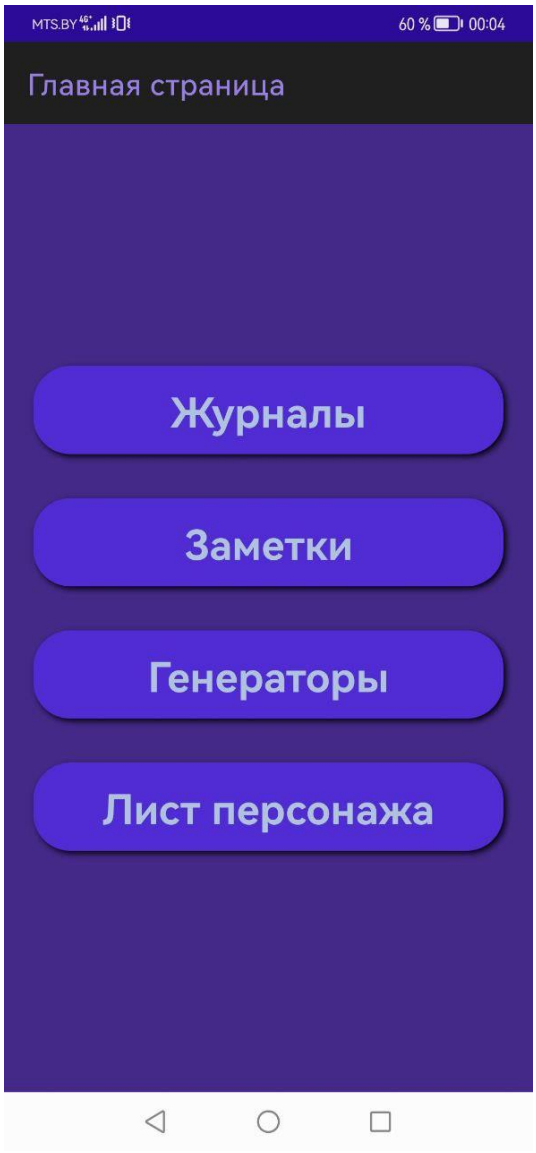

Рисунок 5.1 – Главное меню приложения

Модуль журнала позволяет пользователю вести в хронологическом порядке запись ключевых моментов в проводимых ролевых партиях.

Для создания новой записи в журнал необходимо ввести текст записи в поле «Введите текст…» и нажать сохранить после чего в начале списка записей отобразится новая запись с точной датой и временем создания.

Для удаления существующей записи необходимо свайпнуть по ней после чего отобразится кнопка «Удалить» нажатие на которую позволит удалить невостребованную запись.

Модуль журнала представлен на рисунке 5.2.

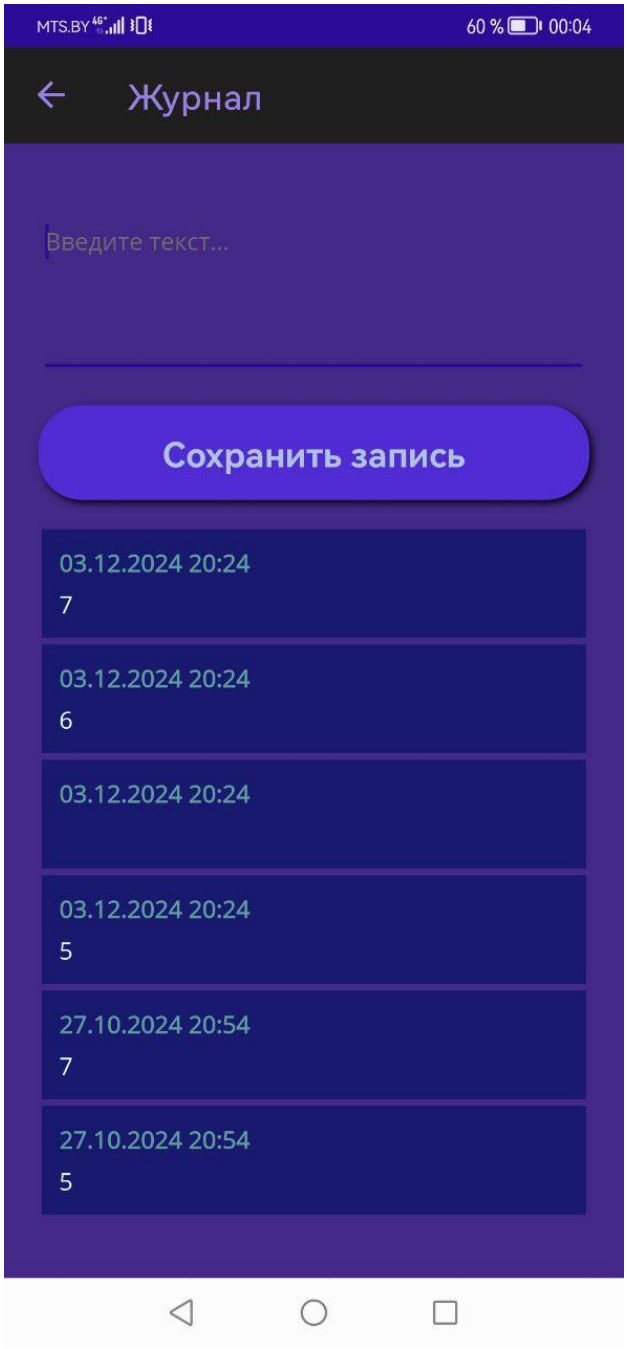

Рисунок 5.2 – Модуль журнала

Модуль заметок позволяет пользователю создавать заметки и управлять ими. Представлен на рисунке 5.3.

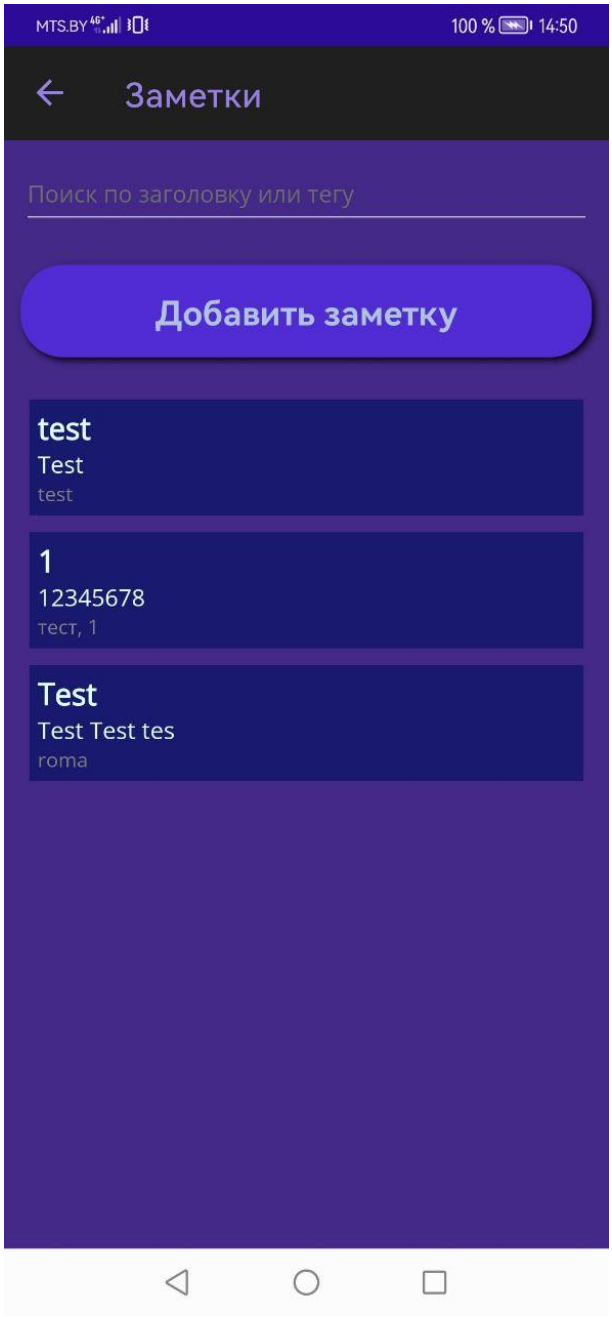

Рисунок 5.3 – Модуль заметок

Для поиска заметки необходимо ввести заголовок или тег искомой заметки в поле «Поиск по заголовку или тегу» после ввода отображается будут только заметки, подходящие под введенные параметры.

Ниже расположен список заметок. Нажав на заметку, откроется окно редактора заметок. Так же для создания заметки необходимо нажать на кнопку «Добавить заметку» это откроет пустой редактор заметок.

Окно редактирования заметок представлено на рисунке 5.4.

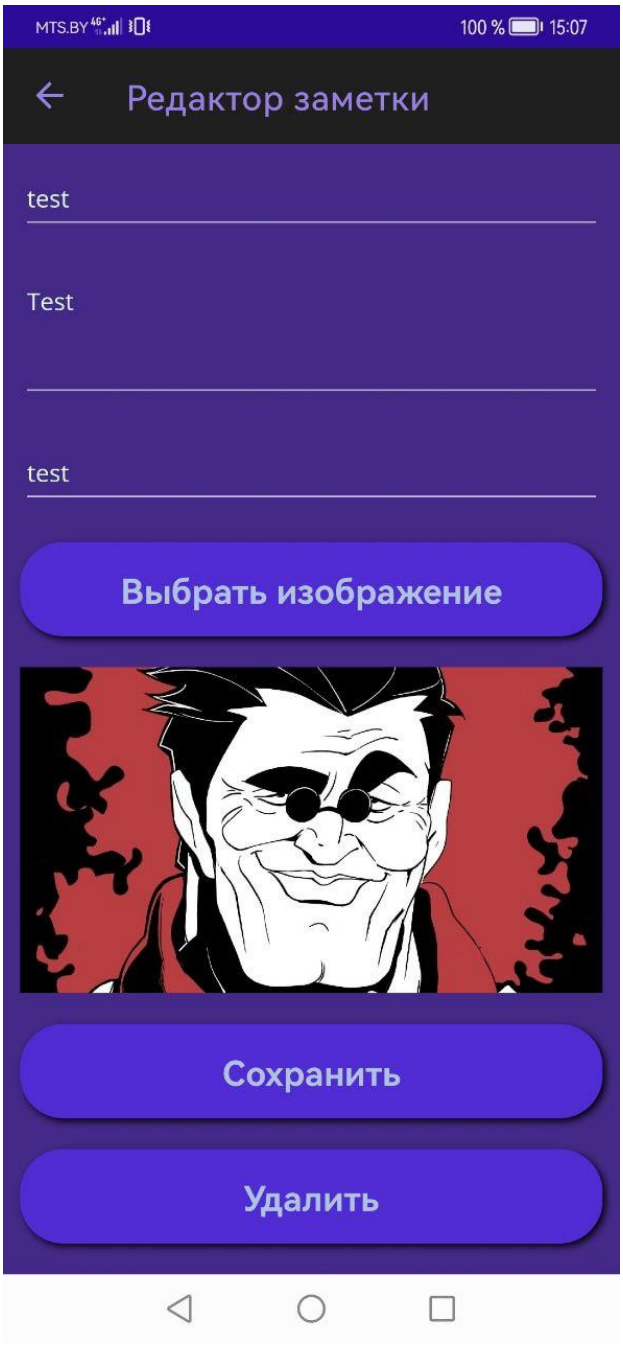

Рисунок 5.3 – Окно редактирования заметок

Для создания изменения заметок предназначено окно редактирования. Оно состоит из:

– «Заголовок» – отвечает за ввод текста для названия заметки.

– «Тело» – текстовое поле для ввода основного текста заметки.

– «Поле тегов» – текстовое поле для ввода тегов к заметке (ввод нескольких тегов производится через запятую).

– Кнопка «Выбрать изображение» – открывает фотогалерею на устройстве для выбора изображения, которое надо прикрепить к заметке.

– Кнопка «Сохранить» – сохраняет изменения в заметке, если это новая заметка, то создает новую заметку.

– Кнопка «Удалить» – доступно только при редактировании существующей заметки. По нажатию удаляет редактируемую заметку.

По завершению редактирования заметки необходимо нажать «Сохранить» (при удалении «Удалить») и приложение вернется на окно модуля заметок.

При переходе к модулю генераторов внутриигровых элементов открывается меню выбора генератора (представлено на рисунке 5.4).

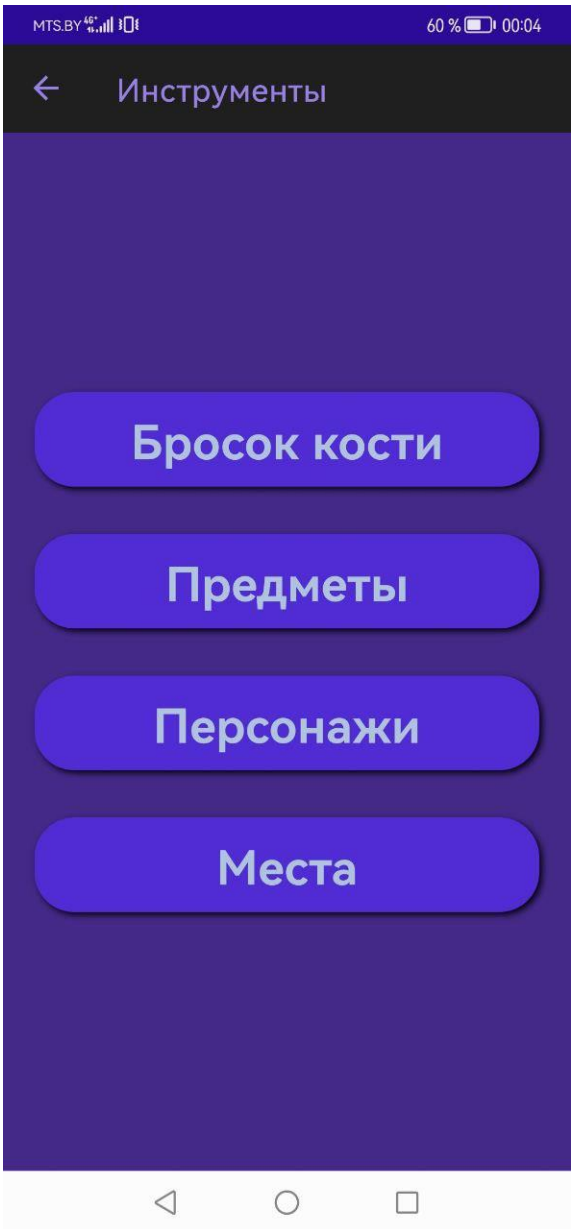

Рисунок 5.4 – Меню выбора генераторов

Меню выбора генераторов содержит четыре кнопки:

– Кнопка «Бросок кости» – открывает генератор бросков игральных костей.

– Кнопка «Предметы» – открывает генератор предметов.

– Кнопка «Персонажи» – открывает генератор неигровых персонажей.

– Кнопка «Места» – открывает генератор внутриигровых локаций.

Генератор бросков игральных костей представлен на рисунке 5.5.

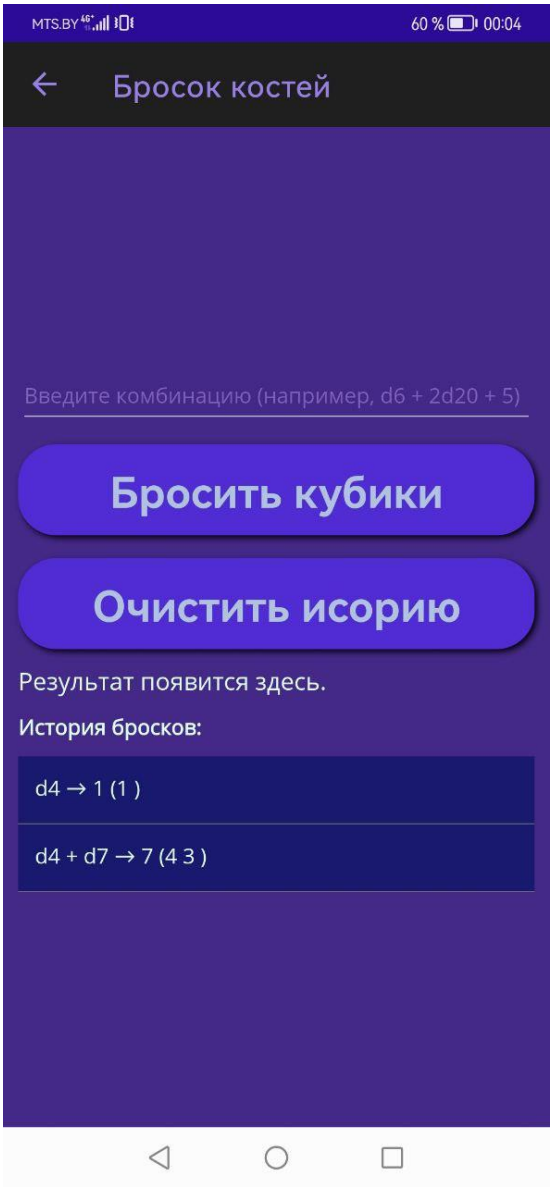

Рисунок 5.5 – Генератор бросков игральных костей

Для генерации броска игральных костей необходимо ввести комбинацию броска в специальное поле сверху окна. Вводимая комбинация должна соответствовать нормам описания бросков игральных костей.

Примеры комбинаций:

– 1d6 – бросок одного шестигранника;

– 5d6 – бросок пяти шестигранников;

– 5d6+5 – бросок пяти шестигранников и к результату прибавить пять;

– 5d6+5+2d10 – бросок пяти шестигранников к результату прибавить пять и прибавить значение броска двух десятигранников.

После ввода комбинации необходимо нажать «Бросить кубики», после нажатия результат броска будет выведен в поле «Результат появится здесь».

Последние десять бросков записываются в виде списка внизу окна. Кнопка «Очистить историю» производит отчистку истории бросков.

Далее рассмотрим использование генератора внутриигровых предметов. Сам генератор представлен на рисунке 5.6.

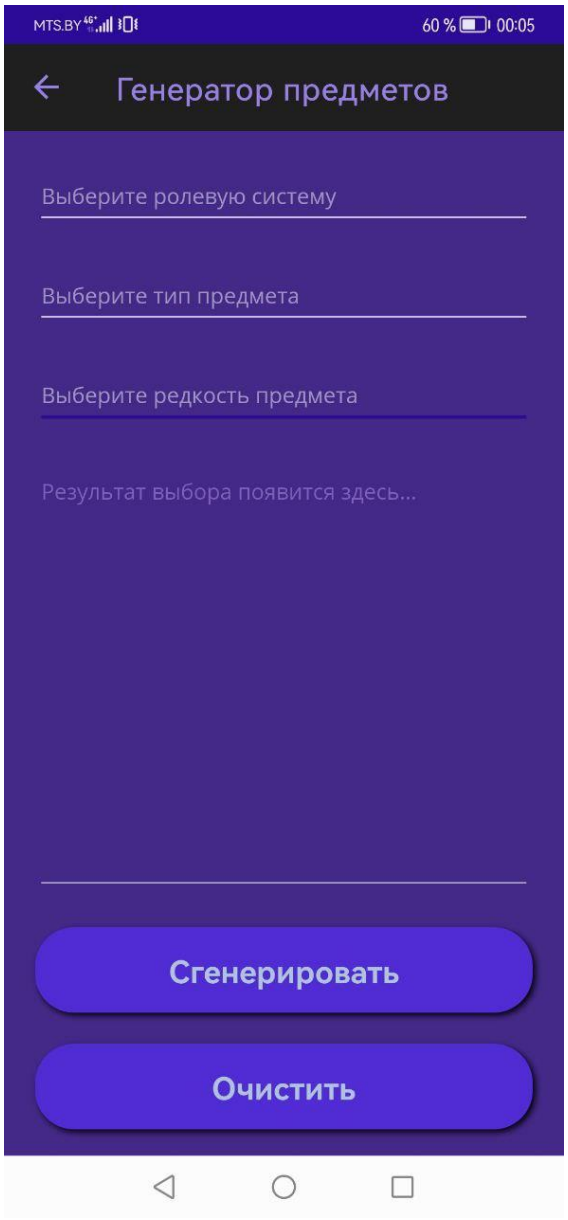

Рисунок 5.6 – Генератор внутриигровых предметов

Для начала работы необходимо выбрать ролевую систему через специальное меню. Далее при необходимости сузить спецификацию генерируемых внутриигровых предметов необходимо выбрать тип предмета и его редкость. Все меню выбора в генераторах выглядят однотипно и для примера на рисунке 5.7 изображено меню выбора типа внутриигрового предмета.

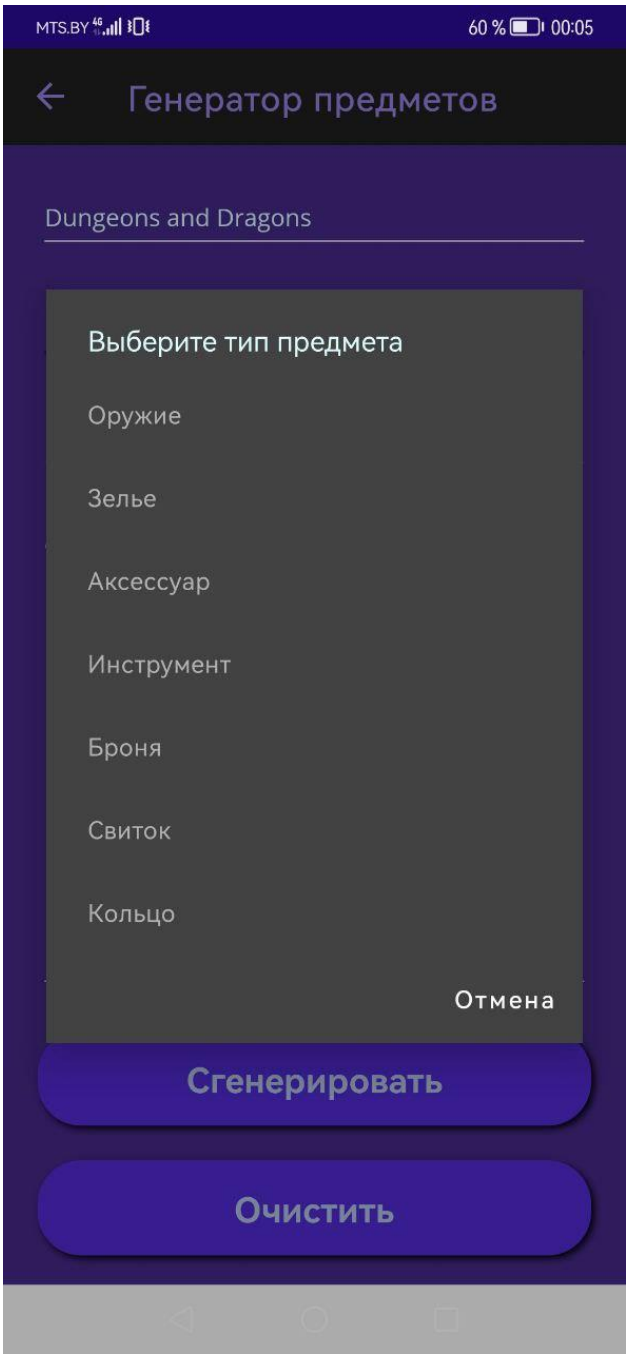

Рисунок 5.7 – Меню выбора типа внутриигрового предмета

Для произведения выбора необходима нажать на нужный вариант из предложенных и поле автоматически им заполнится.

Пример генератора внутриигровых с заполненными полями и сгенерированным предметом представлен на рисунке 5.8.

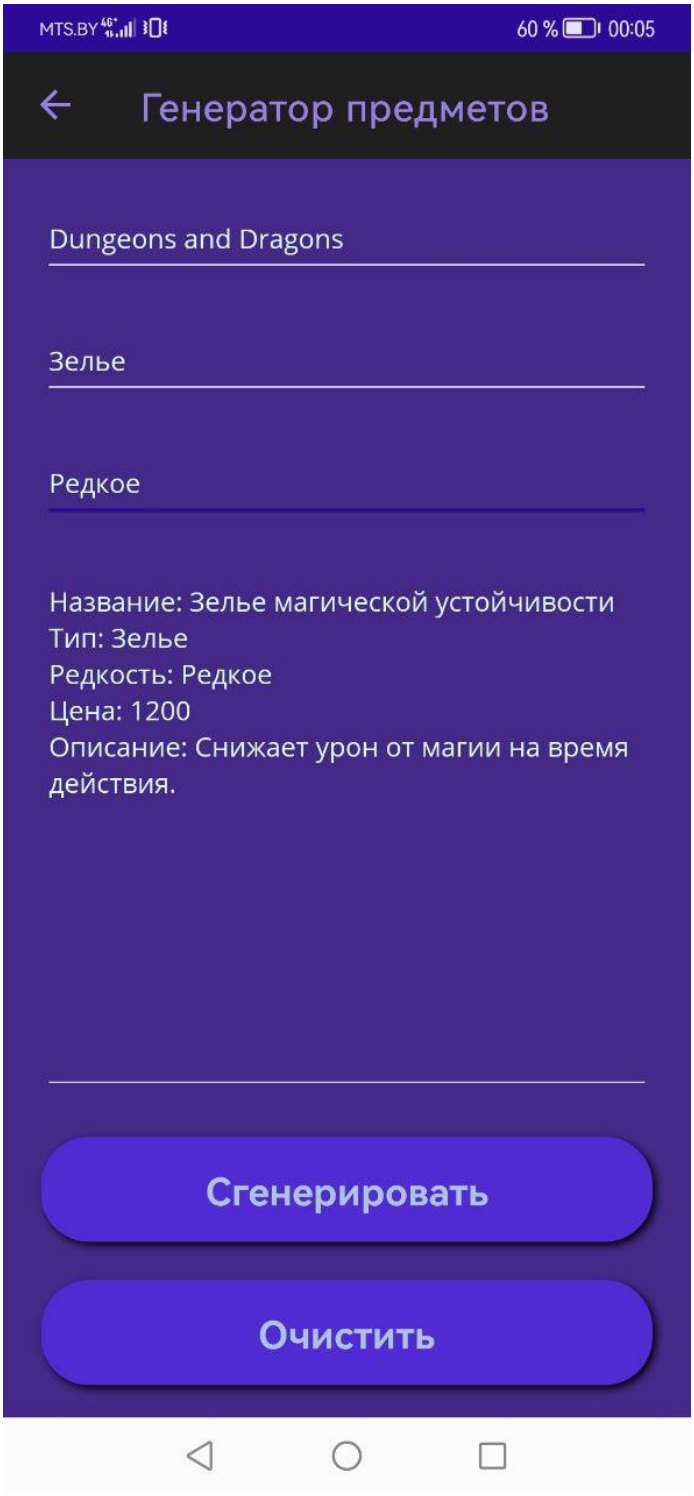

Рисунок 5.8 – Генератор внутриигровых предметов с заполненными полями

Кнопка «Очистить» производит отчистку всех полей генератора.

Генератор неигровых персонажей представлен на рисунке 5.9.

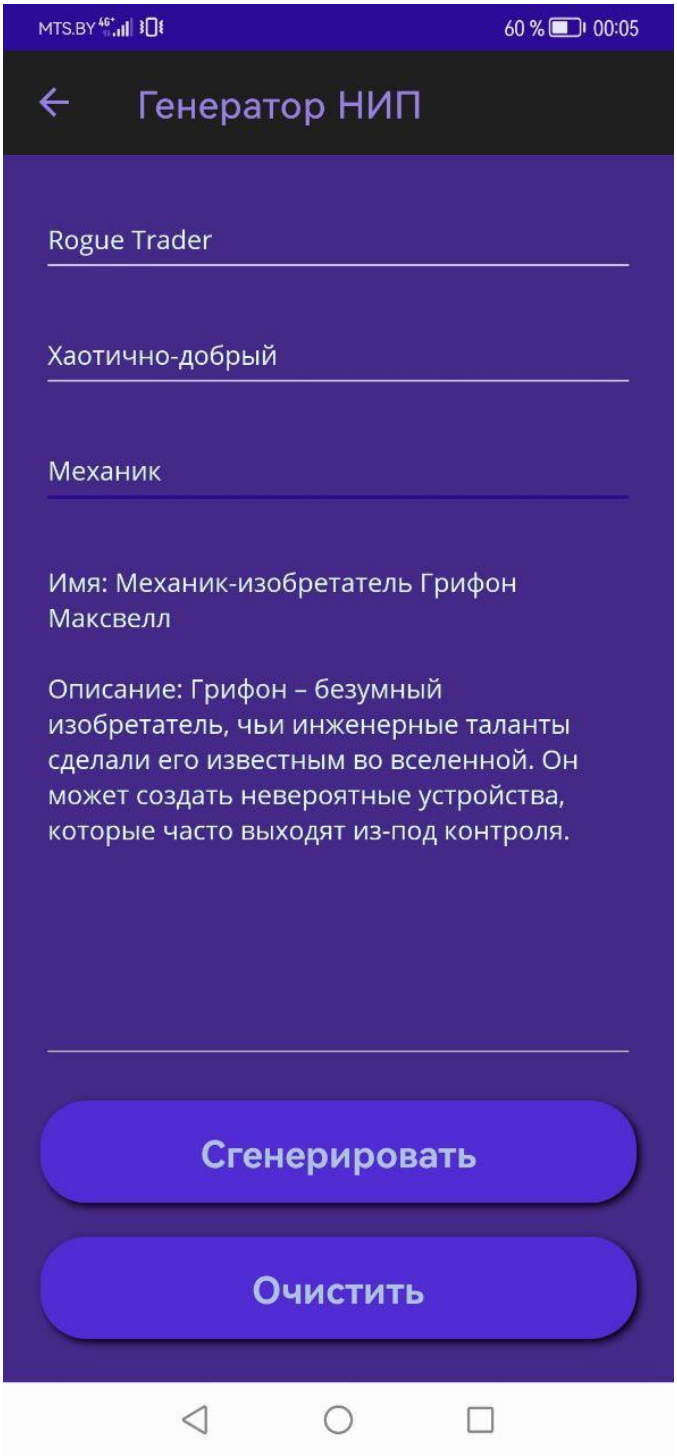

Рисунок 5.9 – Генератор неигровых персонажей

Для начала работы необходимо выбрать ролевую систему через специальное меню. Далее при необходимости сузить спецификацию генерируемых неигровых персонажей необходимо его мировоззрение и при необходимости профессию. Выбор параметров производится через специальные меню (пример меню выбора представлен на рисунке 5.7).

После необходимой настройки параметров пользователь должен нажать кнопку «Сгенерировать». Результат генерации отобразится в специально блоке над кнопкой. Внизу окна предусмотрена кнопка «Очистить» производящая сброс значения всех полей параметров.

Далее рассмотрим использование генератора внутриигровых локаций. Генератор изображен на рисунке 5.10.

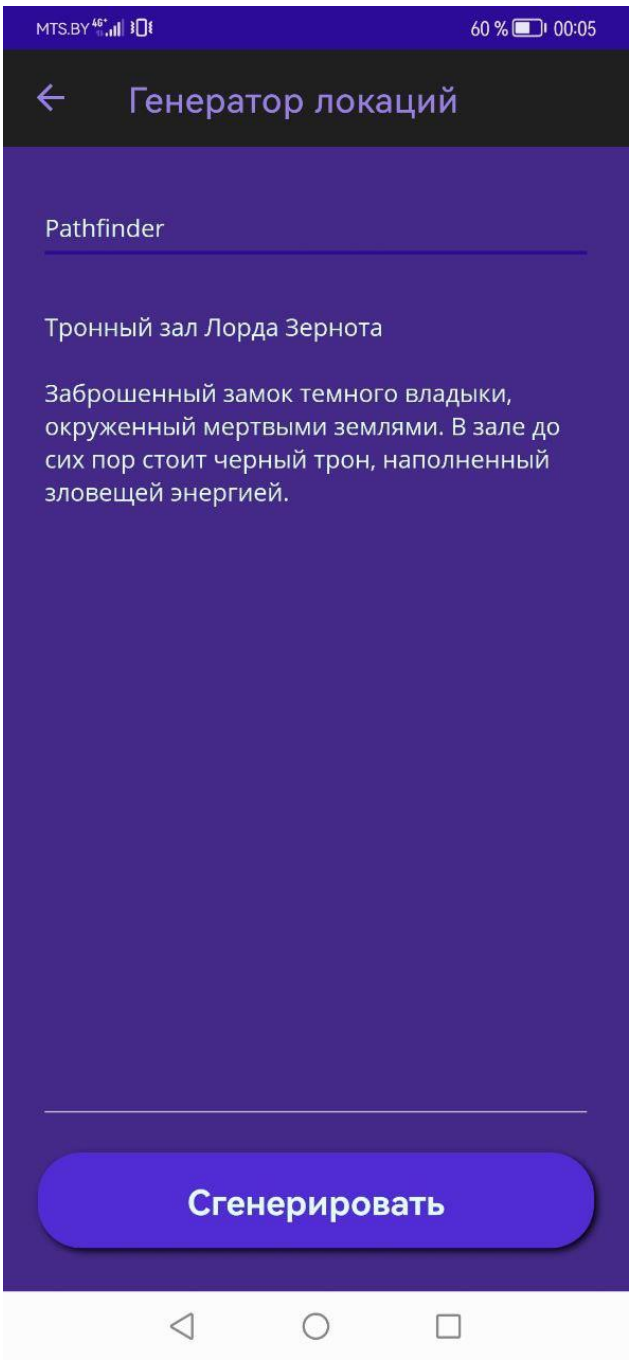

Рисунок 5.10 – Генератор внутриигровых локаций

Для начала необходимо через специальное меню выбора вверху страницы выбрать целевую ролевую систему. После выбора ролевой системы необходимо нажать кнопку «Сгенерировать». Результат генерации отобразится в специальном поле над кнопкой.

Далее рассмотрим последний модуль приложения. Модуль листа персонажа, который предназначен для хранения характеристик персонажа игрока. Первая половина окна модуля представлено на рисунке 5.11.

Рисунок 5.11 – Первая половина окна характеристик персонажа игрока

Окно состоит из набора редактируемых полей предназначенных для хранения параметров персонажа игрока.

На рисунке 5.12 изображена вторая половина окна.

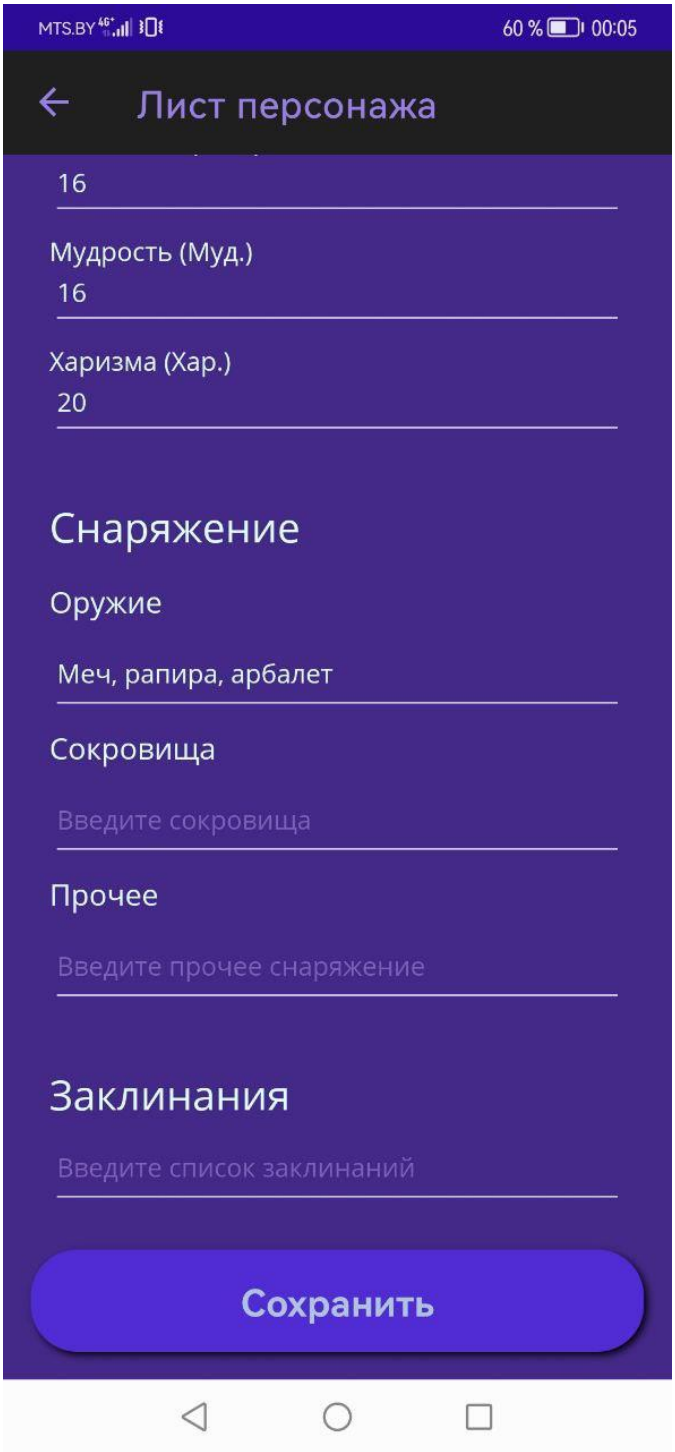

Рисунок 5.11 – Вторая половина окна характеристик персонажа игрока

Список полей сверху-вниз слева направо окна с их предназначением представлен ниже:

– «Имя персонажа» – содержит имя персонажа игрока.

– «Класс» – содержит класс (специализацию) персонажа в соответствии с ролевой системой.

– «Уровень» – содержит значение уровня персонажа.

– «Здоровье, текущее/максимальное» – содержит значения здоровья персонажа на данный момент и максимально возможного здоровья.

– «Защита» – численная характеристика параметра защиты персонажа.

– Раздел «Характеристики» – содержит численные значения основных характеристик персонажа, а именно его силы, ловкости, телосложения, интеллекта, мудрости и харизмы.

– Раздел «Снаряжение» – содержит текстовые поля для ввода списков оружия, сокровищ и прочих внутриигровых предметов, находящихся во владении персонажа.

– «Заклинания» – текстовое поля для хранения списка заклинаний (способностей) известных персонажу игрока.

Внизу окна находится кнопка «Сохранить» которая отвечает за процесс сохранения внесенных в поля модуля изменений.

Навигация между окнами происходит посредством кнопок меню, которые перенаправляют на соответствующие им окна и системных кнопок мобильного устройства (а именно кнопки назад, которая позволяет вернутся к предыдущему окну).

Для установки приложения необходимо запустить APK файл установки.

Приложение запускается через ярлык «TTGHelper».

6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПРОВЕДЕНИЯ НАСТОЛЬНЫХ РОЛЕВЫХ ИГР

6.1 Общая характеристика разрабатываемого приложения

Целью дипломного проекта является разработка приложения, предназначенного для помощи в проведении настольных ролевых игр.

Цель разработки приложения – создание программного средства способного обеспечить потребность разнообразных инструментах автоматизации и ведения заметок для упрощения и автоматизации процесса проведения партий в настольные ролевые игры.

Выполняемые функции:

- Предоставление системы заметок.
- Генератор игровых элементов.
- Генератор бросков игральных костей.
- Менеджер листа персонажа.
- Журнал игровых сессий.

Организация-разработчик: независимый разработчик (или небольшая студия), ориентированный на создание инструментов для сообщества настольных ролевых игр.

Целевая аудитория включает любителей настольных ролевых игр, мастеров, стремящихся к удобной и эффективной организации игры, а также новичков, которым важно иметь доступные и наглядные инструменты для освоения игры. Разрабатываемое приложение применяется игроками в настольные ролевые игры до, после и во время проведения партии для автоматизации игровых процессов.

Процесс подготовки и проведения сессии в настольную ролевую игру связан с необходимостью обработки, анализа и хранения большого количества информации. Игроки создают персонажей с уникальными характеристиками, способностями, навыками и предметами, которые необходимо постоянно отслеживать. Ведущий управляет не только событиями и сюжетом, но и множеством неигровых персонажей (NPC), врагов, окружающей средой и динамикой игрового мира. Все это требует высокой внимательности, контроля, учета и правильных расчетов, что может сильно усложнять игровой процесс, особенно в кампаниях растягивающихся на большое количество сессий, где игра может длиться несколько месяцев или даже лет.

Мобильные приложения для настольных ролевых игр призваны упростить управление всеми этими аспектами, предоставив удобные инструменты как для игроков, так и для ведущих. Такие приложения автоматизируют рутинные задачи, уменьшая объем работы, который нужно

выполнять вручную, и делая игру более плавной и организованной. Основные функции приложений-помощников могут включать создание и управление персонажами, проведение расчётов бросков кубиков, ведение журнала игровых событий, управление инвентарём и мониторинг здоровья персонажей.

С ростом популярности настольных ролевых игр многие игроки начали искать удобные способы организации партий, что вызвало всплеск интереса к приложениям, помогающим в проведении игровых сессий.

Однако, несмотря на очевидную востребованность таких приложений, на рынке существует дефицит комплексных решений. Особенно данный дефицит ощущается в СНГ так как большинство решений на рынке произведены вне региона и имеют низкую или в принципе отсутствующую локализацию.

Кроме того, большинство существующих мобильных приложений для настольных ролевых игр предоставляют лишь отдельные функции, например, генераторы персонажей, инструменты для расчета бросков кубиков или менеджеры инвентаря, но они не представляют собой комплексных инструментов, которые могли бы охватить все аспекты игрового процесса. Это приводит к тому, что игрокам и ведущим приходится использовать сразу несколько приложений, каждое для решения узкого спектра задач, что затрудняет проведение партии, отвлекает от основного игрового процесса и замедляет процесс обработки игровой информации.

В связи с низкой заполненностью рынка и растущим спросом на приложения для проведения настольных ролевых игр, разработка нового конкурентноспособного приложения, которое будет охватывать все нужды игроков и ведущих в одной платформе, является своевременным и перспективной проектом, имеющим высокие шансы окупится и занять существенную нишу на рынке. Новое конкурентноспособное приложение может привлечь как новых пользователей, так и опытных игроков, стремящихся к удобству и эффективности в организации игровых сессий, что сделает настольные ролевые игры ещё более привлекательными для широкой аудитории, а приложение востребованным на рынке.

Конкурирующие аналоги: Dice, Spells 5e, Compendium, RPG Notes, Бестиарий DnD.

Разрабатываемое приложение ориентировано под мобильные телефоны с ОС Android. В перспективе после достижения окупаемости количество поддерживаем платформ увеличится до Windows и iOS.

Приложение будет реализовываться по системе Freemium. Пользователь бесплатно скачивает приложение с базовым функционалом, а продвинутые функции (например инструменты генерации внутриигрового контента) получает на платной основе по мере необходимости. Следовательно, прибыль

будет извлекаться при последующей после релиза реализации подписок на дополнительный функционал. Данная система позволит прорекламировать особенности приложения большому числу пользователей которые будут использовать базовый функционал и в ходе эксплуатации прорекламируют приложение другим игрокам и с определенной вероятностью перейдут в категорию платных пользователей.

Проанализировав отчет по отрасли настольных ролевых игр с маркетинговыми исследованиями и прогнозами от GMI и исследование рынка издательством Hobby World от 18 мая 2023 года были сделаны следующие заключения:

– Стабильный рост продаж НРИ в СНГ на 22 - 40% в год.

– В СНГ на сегодняшний день 1% населения играет в настольные игры, а в странах Европы – 30%, поэтому у российского рынка настольных игр есть большие перспективы роста рынка.

– В СНГ отмечается растущая потребность в инфраструктуре (в том числе и приложениях помощниках) для настольных игр.

На базе исследования можно заключить что общее количество потребителей на рынке составляет ≈ 3194764 чел. Следовательно при плановом получении клиентской базы в размере 5% от общего числа потребителей клиентская база ≈ 159738 чел. При 10% норме покупке премиума пользователями ориентировочное количество премиум пользователей ≈ 15973 чел. При цене премиума в 11,8 рублей годовая прибыль с учетом сбора платформы составит ориентировочно 188481,4 руб. до вычета налогов.

Реализация приложения будет производится через платформы Google Play и в перспективе AppStore.

По плану продукт будет приносить прибыль в течении продолжительного времени, но не с самого начала реализации. При активном добавлении нового функционала срок получения прибыли от приложения можно увеличить.

План развития приложения после релиза:

1. Начало рекламной кампании.
2. Сбор первичного фидбека.
3. Исправления недочетов, выявленных через анализ фидбека.
4. Сбор статистики и опрос потребителей.
5. Добавление нового функционала, который определится путем анализа полученных результатов опросов и статистики.
6. Расширение поддерживаемых платформ (ОС Windows и iOS)
7. Сбор вторичного фидбека и первичного фидбека с новых платформ.
8. Запуск цикла окупаемой модернизации приложения на базе получаемого фидбека от пользователей.

В результате по завершению разработки организация-разработчик получит экономический эффект в виде прироста чистой прибыли.

6.2 Расчет инвестиций в разработку приложения

Первично необходимо произвести расчет затрат на основную заработную плату. Для данного расчета используется формула 6.1.

$$З_0 = К_{пр} \sum_{i=1}^{n} З_{чi} \times t_i, \quad (6.1)$$

где $З_0$ – затраты на основную заработную плату, руб.;

$К_{пр}$ – коэффициент премий;

t_i – трудоемкость работ, ч;

$З_{чi}$ – часовой оклад плата исполнителя i-ой категории, руб.

Коэффициент премий заложен в размере учтен в месячном окладе поэтому приравнивается к единице.

Зарплатные медианы специалистов:

– Программист – 6222,48 руб.
– Тестировщик – 2690,8 руб.
– Дизайнер – 3195,33 руб.
– Бизнес-аналитик – 4782,9 руб.

Данные сформированы на базе аналитики вакансий за 2024 год [20].

Часовой оклад вычисляется путем деления месячного оклада на 168 (среднемесячное количество рабочих часов).

Трудоемкость работ:

– Программист – 47 ч.
– Тестировщик – 12 ч.
– Дизайнер – 12 ч.
– Бизнес-аналитик – 10 ч.

Результаты расчетов затрат на основную заработную для сотрудников сведены в таблицу 6.1.

Таблица 6.1 – Расчет затрат на основную заработную плату

Категория исполнителя	Месячный оклад, руб.	Часовой оклад, руб.	Трудоемкость работ, ч	Итого, руб.
Программист	6222,48	37	47	1739
Тестировщик	2690,8	16	16	256
Дизайнер	3195,33	19	12	228
Бизнес-аналитик	4782,9	28,47	22	626,34

Продолжение таблицы 6.1

Итого	2849,34
Премия и иные стимулирующие выплаты	1424,67
Всего затрат на основную заработную плату разработчиков	4274,01

Далее необходимо рассчитать дополнительную заработную плату сотрудников. Для этого применяется формула 6.2.

$$З_д = \frac{З_о \times Н_д}{100}, \quad (6.2)$$

где $З_д$ – затраты на дополнительную заработную плату, руб.;

$З_о$ – затраты на основную заработную плату, руб.;

$Н_д$ – норматив дополнительной заработной платы.

Для организации принимается норматив дополнительной заработной платы в размеры 20%. Следовательно дополнительная заработная плата сотрудников: 4274,01 × 0,2 = 854,80 руб.

После расчета основной и дополнительной заработной платы необходимо рассчитать отчисления на социальные нужды. Для этого используется формула 6.3.

$$Р_{соц} = \frac{(З_о + З_д) \times Н_{соц}}{100}, \quad (6.3)$$

где $Р_{соц}$ – отчисления на социальные нужды, руб.;

$З_о$ – затраты на основную заработную плату, руб.;

$З_д$ – затраты на дополнительную заработную плату, руб.;

$Н_{соц}$ – норматив отчислений в ФСЗН и Белгосстрах.

Актуальный норматив отчислений равен 34,6%, следовательно, для сотрудников отчисления на социальные нужды, следующие: (4274,01 + 854,80) × 0,346 = 1774,57.

Далее необходимо рассчитать прочие расходы на разработку приложения по формуле 6.4.

$$Р_{пр} = \frac{З_о \times Н_{пр}}{100}, \quad (6.4)$$

где $Р_{пр}$ – прочие расходы, руб.;

$З_о$ – затраты на основную заработную плату, руб.;

$Н_{пр}$ – норматив прочих расходов.

Так как $Н_{пр}$ принят в размере 30%, то прочие расходы: 4274,01 × 0,3 = 1282,20 руб.

Осталось рассчитать расходы на реализацию приложения по формуле 6.5.

$$Р_р = \frac{З_о \times Н_р}{100}, \quad (6.5)$$

где $Р_р$ – расходы на реализацию, руб.;

$З_о$ – затраты на основную заработную плату, руб.;

$Н_р$ – норматив расходов на реализацию.

Норматив расходов на реализацию принят в размере 5%, то прочие расходы: 4274,01 × 0,05 = 213,70 руб.

Для расчета общей суммы затрат результаты были сведены в таблицу 6.2.

Таблица 6.2 – Расчет затрат на разработку приложения

Наименование статьи затрат	Формула/таблица для расчета	Значение, руб.
Основная заработная плата ($З_о$)	Расчеты представлены в таблице 6.1.	4274,01
Дополнительная заработная плата ($З_д$)	$З_д = \frac{4274,01 \times 20}{100}$	854,80
Отчисления на социальные нужды ($Р_{соц}$)	$Р_{соц} = \frac{(4274,01 + 854,80) \times 34,6}{100}$	1774,57
Прочие расходы ($Р_{пр}$)	$Р_{пр} = \frac{4274,01 \times 30}{100}$	1282,20
Расходы на реализацию ($Р_р$)	$Р_р = \frac{4274,01 \times 0,05}{100}$	213,70
Общая сумма затрат на разработку и реализацию ($З_р$)	$З_0 = 4274,01 + 854,80 + 1774,57 + 1282,20 + 213,70 =$	8399,28

По результатам расчетов общий размер инвестиций в разработку приложения равняется 8399,28 рублей.

6.3 Расчет экономического эффекта от реализации программного средства на рынке

Планируемый объём продаж 3194 подписок. Данный объём рассчитан путем анализов рынка. Из исследования Hobby Games и анализа рынка от GMI TAM (Total Addressable Market) для изучаемого сегмента рынка (рынок настольных ролевых игр в СНГ) равняется 31947640 человек. Из-них SAM (Serviceable Available Market) равняется 3194764 человек. С учетом того, что на рынке уже существуют схожие решения, но в ограниченном количестве то прогнозируемая доля рынка для приложения равняется от примерно от 5% до 8%. Для объективности расчетов будет выбрано наименьшее значение – 5%, следовательно, SOM (Serviceable & Obtainable Market) приложения – 159738 человека. Для приложений с подпиской общемировая статистика указывает на то, что в среднем 10% от общего числа пользователей приобретают премиум, следовательно, количество платных пользователей равняется 15973 человек.

Ценна подписки рассчитана на базе общемировых тенденций стоимости (представлены на рисунке 6.1) и равняется 14 долларам США что примерно равно 47,25 рублей.

Цена в США, $

	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Netflix	7,99	9,99	9,99	9,99	10,99	10,99	12,99	13,99	13,99	15,49
Disney+	—	—	—	—	—	—	6,99	6,99	7,99	10,99
Amazon Prime	6,58	8,25	10,99	10,99	10,99	12,99	12,99	12,99	12,99	14,99
HBO Max	—	—	14,99	14,99	14,99	14,99	14,99	11,99	7,49	14,99
Hulu	—	—	11,99	11,99	11,99	11,99	11,99	11,99	12,99	14,99
ESPN+	—	—	—	—	—	4,99	4,99	4,99	6,99	9,99
Spotify Family	—	14,99	14,99	14,99	14,99	14,99	14,99	14,99	15,99	15,99
Apple One	—	—	—	—	—	—	—	14,95	14,95	14,95
Эквивалент $ 10 с учетом инфляции	10,00	10,16	10,17	10,30	10,52	10,78	10,97	11,11	11,63	12,72

Источники: Netflix – The Verge, Disney+ – TechCrunch, Amazon Prime – Business Insider, HBO Max – Reuters, Hulu – The Verge, ESPN+ – The Motley Fool, Spotify Family – TechCrunch, Apple One – Apple.

Рисунок 6.1 – Стоимость подписок на популярные сервисы.

Но стоит учесть, что приложение не может предоставить функционала эквивалентного популярным сервисам из-за чего цена подписки равняется 25% и равняется 11,8 рублей.

Проанализировав рынок, рассчитав стоимость подписки и количество покупателей можно перейти к расчету прироста чистой прибыли. Для расчета чистой прибыли используется формула 6.6.

$$\Delta П_{ч}^{р} = (Ц_{отп} \times N - НДС) \times Р_{пр} \times \left(1 - \frac{Н_{п}}{100}\right), \quad (6.6)$$

где $\Delta П_{ч}^{р}$ – прирост чистой прибыли, руб.;

$Ц_{отп}$ – стоимость подписок, руб.;

N – количество оформляемых в год подписок, шт.;

НДС – сумма налога на добавочную стоимость, руб.;

$Р_{пр}$ – рентабельность продаж, руб.;

$Н_{п}$ – ставка налога на прибыль.

Для разрабатываемого приложения стоимость подписки – 11,8 руб., количество оформляемых в год подписок – 15973 шт., НДС рассчитывается по формуле 6.7, рентабельность продаж для приложения находится на уровне 30%, ставка налога на прибыль – 20%.

$$НДС = \frac{Ц_{отп} \times N \times Н_{д.с.}}{100\% + Н_{д.с.}}, \quad (6.7)$$

где НДС – сумма налога на добавочную стоимость, руб.;

$Ц_{отп}$ – стоимость подписок, руб.;

N – количество оформляемых в год подписок, шт.;

$Н_{д.с.}$ – ставка налога на добавленную стоимость.

Следовательно сумма налога на добавочную стоимость (НДС) равняется: 11,8 × 15973 × 20% / 120% = 31413,57 руб.

Определив НДС можно перейти к расчетам прироста чистой прибыли: (11,8 × 15973 – 31413,57) × 30% × (1 – 0,2) = 37696,27 руб.

6.4 Расчет показателей экономической эффективности разработки и реализации приложения

Для расчета экономической эффективности разработки и реализации приложения необходимо рассчитать рентабельность инвестиций ROI (Return of Investment) для этого используется формула 6.8.

$$ROI = \frac{\Delta П_{ч}^{р} - З_{р}}{З_{р}} \times 100\%, \quad (6.8)$$

где ROI – рентабельность инвестиций;

$З_{р}$ – общая сумма затрат на разработку и реализацию, руб.;

$\Delta П_{ч}^{р}$ – прирост чистой прибыли, руб.

Для разрабатываемого приложения общая сумма затрат на разработку и реализацию равняется 8399,28 руб., а прирост чистой прибыли – 37696,27 руб. Следовательно ROI равняется: (37696,27 – 8399,28) / 8399,28 = 3,48 = 34,8%.

Для того чтобы определить рентабельность приложения необходимо сравнить рентабельность инвестиций ROI с рентабельностью долгосрочного вклада той же суммы (т.е. сравнить ставку долгосрочного депозита с полученным значением ROI).

На ноябрь 2024 года в Республике Беларусь самая высокая ставка по долгосрочным депозитам по результатам анализа предложений банков Республики Беларусь и информации с сайта MYFIN (представлена на рисунке 6.2) равняется – 14,5%.

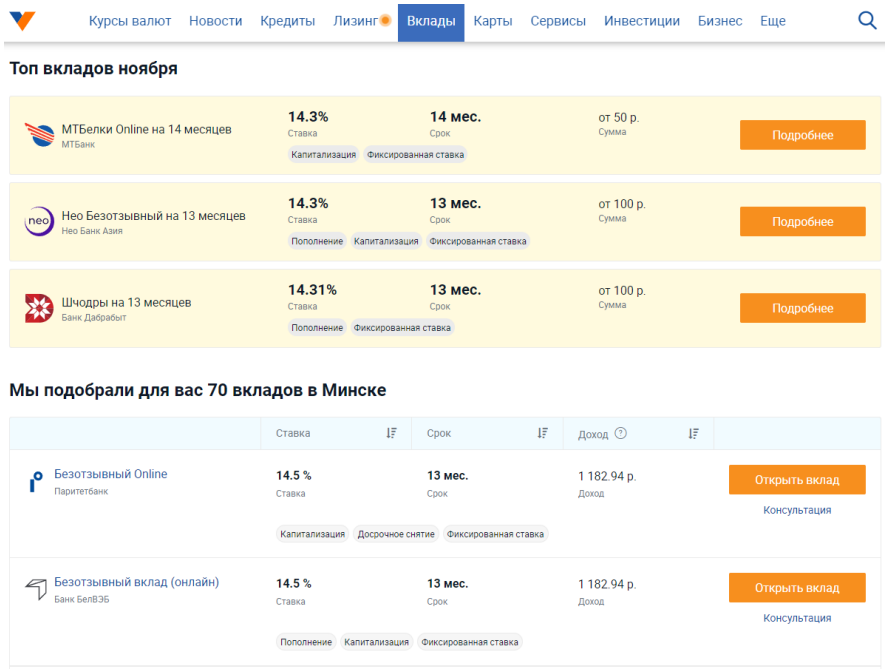

Рисунок 6.1 – Информация о ставках долгосрочных депозитов.

Ставка в размере 14,5% меньше полученного значения рентабельности инвестиций (34,8%), следовательно разработка данного приложения является экономически эффективной и принесёт организации существенную прибыль.

ЗАКЛЮЧЕНИЕ

В ходе проекта разработано мобильное приложение для проведения настольных ролевых игр.

В начале разработки было проведено исследование предметной области с целью изучения рынка, существующих аналогов и запросов потенциальной аудитории. На основе полученных данных были сформированы конкретные функциональные требования предъявляемы к приложению и спектр решаемых им задач.

В ходе исследования требований рынка и существующих инструментов реализации были выбраны современные инструменты разработки – язык программирования C# с фреймворком MAUI и языком разметки XAML. Данный выбор позволяет создать производительное, гибкое и универсальное решение, которое будет работать на разных платформах с минимизированными затратами на поддержку и разработку, что в перспективе позволяет легко добавлять новую функциональность и портировать приложение на новые платформы. Высокая производительность, строгая типизация, поддержка нативных API и современных архитектурных паттернов делают это сочетание оптимальным для мобильной и кроссплатформенной разработки.

Разработанные в приложении алгоритмы охватывают ключевые аспекты необходимые для проведения партии в настольные ролевые игры. К примеру приложение реализует систему автоматизации сложных бросков игральных костей. В приложении реализованная система заметок с возможность поиска по ключевым значениям и система записей в журнал с хронологическим порядком. Кроме того приложения укомплектовано генераторами позволяющими автоматизировать и значительно ускорить процесс создания случайных игровых событий.

Интерфейс приложения спроектирован специально для удобного использования на мобильных устройствах с расчетом на управление одной рукой.

По завершению разработки была проведена серия разноплановых тестов с целью выявления и дальнейшего исправления ошибок и несоответствий в приложении.

По результатам выполнения проекта было спроектировано, разработано и протестировано приложение, полностью выполняющее поставленные перед ним задачи, позволяющее упростить и автоматизировать разнообразные аспекты проведения партий в настольные ролевые игры и обладающее хорошим потенциалом для дальнейшего развития.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] WHAT IS A ROLE – PLAYING GAME? [Электронный ресурс]. – Режим доступа: https://web.archive.org/web/20080914081206/http://www.darkshire.net/~jhkim/rpg/whatis/.

[2] "Narrative" or "Tabletop" RPG's [Электронный ресурс]. – Режим доступа: https://web.archive.org/web/20080829174633/http://www.darkshire.net/~jhkim/rpg/whatis/tabletop.html.

[3] Как устроены настольные игры [Электронный ресурс]. – Режим доступа: https://club.dns-shop.ru/blog/t-275-nastolnyie-igryi/61675-kak-ustroenyi nastolnyie-igryi-kluchevyie-mehaniki-elementyi-i-sett/.

[4] How to make a roleplaying game [Электронный ресурс]. – Режим доступа: https://www.dicebreaker.com/categories/roleplaying-game/how-to/how-to-make-rpg.

[5] Basic Rules for Dungeons & Dragons [Электронный ресурс]. – Режим доступа: https://web.archive.org/web/20140731124943/http://dnd.wizards.com/articles/features/basicrules?x=dnd/basicrules.

[6] How COVID helped tabletop RPGs go mainstream [Электронный ресурс]. – Режим доступа: https://www.gameshub.com/news/features/how-covid-helped-tabletop-rpgs-go-mainstream-6028/.

[7] Tabletop roleplaying has given players comfort, connection and control in a world that’s taken them away [Электронный ресурс]. – Режим доступа: https://www.dicebreaker.com/categories/roleplaying-game/feature/tabletop-roleplaying-comfort-connection-control-covid-19.

[8] Леонид Мойжес 200 лет российских настолок [Электронный ресурс]. – Режим доступа: https://sreda.v-a-c.org/ru/.

[9] Выявление и сбор требований к ПО [Электронный ресурс]. – Режим доступа: https://tproger.ru/articles/vyjavlenie-i-sbor-trebovanij-k-po-ultimate-guide.

[10] Вагнер, Б. C# Эффективное программирование / Билл Вагнер. – М.: ЛОРИ, 2017.

[11] .NET Goes Cross-Platform with .NET Core [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/archive/msdn-magazine/2016/april /net-core-net-goes-cross-platform-with-net-core.

[12] Стиллмен Э., Грин Дж. Изучаем C#. 3-е изд. – СПб.: Питер, 2015.

[13] XAML Services [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/dotnet/desktop/xaml-services /?redirectedfrom=MSDN

[14] Моделирование предметной области [Электронный ресурс]. – Режим доступа: https://studme.org/174167/informatika/modelirovanie_predmetnoy_ oblasti/.

[15] Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/566218/.

[16] Диаграмма деятельности [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма_деятельности.

[17] Что такое функциональные требования: примеры, определение, полное руководство [Электронный ресурс]. – Режим доступа: https://visuresolutions.com/ru/Блог/функциональные-требования/.

[18] Алгоритмы в пограммировании [Электронный ресурс]. – Режим доступа: https://www.interface.ru/home.asp?artId=35216.

[19] Проектирование интерфейса: 8 принципов, которые должен знать каждый UX-дизайнер [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/proektirovanie-interfejsov/.

[20] Зарплата в IT [Электронный ресурс]. – Режим доступа: https://salaries.devby.io.

Приложение А
(обязательное)
Исходный код программы

```
namespace MauiApp1
{
   public partial class MainPage : ContentPage
   {
      public MainPage()
      {
         InitializeComponent();
      }

      private async void OnButtonClicked(object sender, EventArgs e)
      {
         await Navigation.PushAsync(new NotePage());
      }

      private async void OnButtonJurnal(object sender, EventArgs e)
      {
         await Navigation.PushAsync(new JournalsPage());
      }

      private async void OnButtonGen(object sender, EventArgs e)
      {
         await Navigation.PushAsync(new GenMenu());
      }

      private async void OnButtonChar(object sender, EventArgs e)
      {
         await Navigation.PushAsync(new CharList());
      }
   }

}
```

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MauiApp1.MainPage"
```

```
            Title="Главная страница">

        <ScrollView>

            <VerticalStackLayout>

                <Button Text="Журналы" Clicked="OnButtonJurnal" />
                <Button Text="Заметки" Clicked="OnButtonClicked" />
                <Button Text="Генераторы" Clicked="OnButtonGen" />
                <Button Text="Лист персонажа" Clicked="OnButtonChar" />

            </VerticalStackLayout>

        </ScrollView>

    </ContentPage>

    public partial class JournalsPage : ContentPage
    {
        private const string FileName = "journals.json";
        private string FilePath =>
Path.Combine(FileSystem.Current.AppDataDirectory, FileName);

        public ObservableCollection<JournalEntry> JournalEntries { get; set; }

        public JournalsPage()
        {
            InitializeComponent();
            JournalEntries = new ObservableCollection<JournalEntry>();
            JournalsList.ItemsSource = JournalEntries;

            LoadEntriesAsync();
        }

        private async void OnSaveButtonClicked(object sender, EventArgs e)
        {
            // Новая запись с текущей датой и временем
            var newEntry = new JournalEntry
            {
```

```
            Date = DateTime.Now.ToString("g"),
            Text = EntryText.Text
        };

        JournalEntries.Insert(0, newEntry);
        EntryText.Text = string.Empty;
        await SaveEntriesAsync();
    }

    // Чтение файла
    private async Task LoadEntriesAsync()
    {
        if (File.Exists(FilePath))
        {
            var json = await File.ReadAllTextAsync(FilePath);
            var entries =
JsonSerializer.Deserialize<ObservableCollection<JournalEntry>>(json);
            if (entries != null)
            {
                foreach (var entry in entries)
                {
                    JournalEntries.Add(entry);
                }
            }
        }
    }

    // Сохранение в файл
    private async Task SaveEntriesAsync()
    {
        var json = JsonSerializer.Serialize(JournalEntries);
        await File.WriteAllTextAsync(FilePath, json);
    }

    private async void OnDeleteSwipeItemInvoked(object sender, EventArgs
e)
    {
        if (sender is SwipeItem swipeItem && swipeItem.CommandParameter
is JournalEntry entryToDelete)
```

```
            {
                bool confirm = await DisplayAlert("Подтверждение", "Вы действительно хотите удалить эту запись?", "Да", "Нет");
                if (confirm)
                {
                    JournalEntries.Remove(entryToDelete);
                    await SaveEntriesAsync();
                }
            }
        }
    }

    public class JournalEntry
    {
        public string Date { get; set; }
        public string Text { get; set; }
    }

    <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 x:Class="MauiApp1.JournalsPage"
                 Title="Журнал">

        <ScrollView>

            <VerticalStackLayout Spacing="15">

                <!-- Поле для ввода -->
                <Editor x:Name="EntryText" Placeholder="Введите текст..." AutoSize="TextChanges" HeightRequest="100" TextColor = "#E0FFFF"/>

                <!-- Кнопка сохранения -->
                <Button Text="Сохранить запись" Clicked="OnSaveButtonClicked" Style="{StaticResource SmallButtonStyle}"/>

                <!-- Список записей -->
                <CollectionView x:Name="JournalsList" ItemsSource="{Binding JournalEntries}" HeightRequest="400">
```

```
                <CollectionView.ItemTemplate>

                    <DataTemplate>

                        <SwipeView BackgroundColor="#191970" Margin="2">

                            <!-- Контекстное меню удаления -->
                            <SwipeView.RightItems>
                                <SwipeItem                                Text="Удалить"
BackgroundColor="#8B0000"
                                    IconImageSource="delete_icon.png"
                                    CommandParameter="{Binding .}"
                                    Invoked="OnDeleteSwipeItemInvoked"/>
                            </SwipeView.RightItems>

                            <StackLayout Padding="10" Spacing="5">
                                <Label  Text="{Binding  Date}"  FontAttributes="Bold"
TextColor="#5F9EA0"/>
                                <Label Text="{Binding Text}" TextColor = "#E0FFFF"
/>
                            </StackLayout>

                        </SwipeView>

                    </DataTemplate>

                </CollectionView.ItemTemplate>

            </CollectionView>

        </VerticalStackLayout>

    </ScrollView>

</ContentPage>

public partial class NotePage : ContentPage
{
```

```
        private ObservableCollection<Note> _notes = new
ObservableCollection<Note>();

        public NotePage()
        {
            InitializeComponent();
            _notes = NoteStorage.LoadNotes();
            NotesCollectionView.ItemsSource = _notes;
        }

        private async void OnAddNoteClicked(object sender, EventArgs e)
        {
            var newNote = new Note();
            await Navigation.PushAsync(new NoteEditor(newNote, _notes));
        }

        private void OnSearchTextChanged(object sender,
TextChangedEventArgs e)
        {
            string searchText = e.NewTextValue.ToLower();
            NotesCollectionView.ItemsSource = _notes.Where(note =>
                note.Title.ToLower().Contains(searchText) ||
                note.Tags.Any(tag => tag.ToLower().Contains(searchText))
            ).ToList();
        }

        private async void OnNoteSelected(object sender,
SelectionChangedEventArgs e)
        {
            if (e.CurrentSelection.FirstOrDefault() is Note selectedNote)
            {
                NotesCollectionView.SelectedItem = null;
                await Navigation.PushAsync(new NoteEditor(selectedNote, _notes));
            }
        }
        protected override void OnDisappearing()
        {
            base.OnDisappearing();
            NoteStorage.SaveNotes(_notes);
```

```
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
            NotesCollectionView.ItemsSource = null;
            NotesCollectionView.ItemsSource = _notes;
        }
    }

    <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 xmlns:local="clr-namespace:MauiApp1"
                 x:Class="MauiApp1.NotePage"
                 Title="Заметки">

        <ContentPage.Content>

            <StackLayout Padding="10" BackgroundColor="#452988" Spacing="20">

                <!-- Поиск -->
                <Entry x:Name="SearchEntry" Placeholder="Поиск по заголовку или тегу" TextChanged="OnSearchTextChanged" TextColor = "#E0FFFF"/>
                <Button Text="Добавить заметку" Clicked="OnAddNoteClicked" Style="{StaticResource SmallButtonStyle}" />

                <!-- Список заметок -->
                <CollectionView x:Name="NotesCollectionView" SelectionMode="Single" SelectionChanged="OnNoteSelected">

                    <CollectionView.ItemTemplate>

                        <DataTemplate>

                            <StackLayout Padding="5" Margin="5" BackgroundColor="#191970">
```

```
                        <Label Text="{Binding Title}" FontSize="18"
FontAttributes="Bold" TextColor = "#E0FFFF" />
                        <Label Text="{Binding Body}" FontSize="14"
LineBreakMode="TailTruncation" TextColor = "#E0FFFF" />
                        <Label Text="{Binding TagsString}" FontSize="12"
TextColor="Gray" />

                    </StackLayout>

                </DataTemplate>

            </CollectionView.ItemTemplate>

        </CollectionView>

    </StackLayout>

</ContentPage.Content>

</ContentPage>

namespace MauiApp1
{
    public partial class NoteEditor : ContentPage
    {
        private Note _note;
        private ObservableCollection<Note> _notes;

        public NoteEditor(Note note, ObservableCollection<Note> notes)
        {
            InitializeComponent();
            _note = note ?? new Note();
            _notes = notes;

            if (_note != null)
            {
                TitleEntry.Text = _note.Title;
                BodyEditor.Text = _note.Body;
                TagsEntry.Text = string.Join(", ", _note.Tags);
```

```
            if (!string.IsNullOrEmpty(_note.ImagePath))
            {
                NoteImage.Source = _note.ImagePath;
                NoteImage.IsVisible = true;
            }

            if (_notes.Contains(_note))
                DeleteButton.IsVisible = true;
        }
    }

    private async void OnPickImageClicked(object sender, EventArgs e)
    {
        var result = await FilePicker.PickAsync(new PickOptions
        {
            FileTypes = FilePickerFileType.Images,
            PickerTitle = "Выберите изображение"
        });

        if (result != null)
        {
            string localPath =
Path.Combine(FileSystem.Current.AppDataDirectory, result.FileName);
            using (var stream = await result.OpenReadAsync())
            {
                using (var fileStream = File.OpenWrite(localPath))
                {
                    await stream.CopyToAsync(fileStream);
                }
            }
            _note.ImagePath = localPath;
            NoteImage.Source = localPath;
            NoteImage.IsVisible = true;
        }
    }

    private void OnSaveClicked(object sender, EventArgs e)
    {
```

```
            _note.Title = TitleEntry.Text?.Trim();
            _note.Body = BodyEditor.Text?.Trim();
            _note.Tags = TagsEntry.Text?.Split(',')?.Select(tag => tag.Trim()).ToList();

            if (!_notes.Contains(_note))
            {
                _notes.Add(_note);
            }

            NoteStorage.SaveNotes(_notes);
            Navigation.PopAsync();
        }

        private void OnDeleteClicked(object sender, EventArgs e)
        {
            if (_notes.Contains(_note))
            {
                _notes.Remove(_note);
            }

            Navigation.PopAsync();
        }

        // Обработчик тапа по изображению
        private void OnImageTapped(object sender, EventArgs e)
        {
            if (!string.IsNullOrEmpty(_note.ImagePath))
            {
                FullscreenImage.Source = _note.ImagePath;
                FullscreenImage.IsVisible = true;
            }
        }

        // Обработчик тапа по полноэкранному изображению
        private void OnFullscreenImageTapped(object sender, EventArgs e)
        {
            FullscreenImage.IsVisible = false;
        }
```

```
    }
}

    <ContentPage.Content>

        <Grid>

            <!-- Основной вид -->
            <ScrollView>

                <StackLayout Padding="10" Spacing="18">

                    <Entry x:Name="TitleEntry" Placeholder="Заголовок" TextColor = "#E0FFFF" />
                    <Editor x:Name="BodyEditor" Placeholder="Текст заметки" HeightRequest="80" TextColor = "#E0FFFF" />
                    <Entry x:Name="TagsEntry" Placeholder="Теги (через запятую)" TextColor = "#E0FFFF" />

                    <Button Text="Выбрать изображение" Clicked="OnPickImageClicked" Style="{StaticResource SmallButtonStyle}" />
                    <Image x:Name="NoteImage"
                        HeightRequest="190"
                        Aspect="AspectFill"
                        IsVisible="False">
                      <Image.GestureRecognizers>
                        <TapGestureRecognizer Tapped="OnImageTapped" />
                      </Image.GestureRecognizers>
                    </Image>

                    <Button Text="Сохранить" Clicked="OnSaveClicked" Style="{StaticResource SmallButtonStyle}" />
                    <Button x:Name="DeleteButton" Text="Удалить" Clicked="OnDeleteClicked" IsVisible="False" Style="{StaticResource SmallButtonStyle}"/>

                </StackLayout>

            </ScrollView>
```

```
            <!-- Полноэкранное изображение -->
            <Image x:Name="FullscreenImage"
                   Aspect="AspectFit"
                   IsVisible="False"
                   BackgroundColor="Black"
                   Opacity="0.9">
                <Image.GestureRecognizers>
                    <TapGestureRecognizer  Tapped="OnFullscreenImageTapped" />
                </Image.GestureRecognizers>
            </Image>

        </Grid>

    </ContentPage.Content>
</ContentPage>

namespace MauiApp1.Models
{
    public class Note
    {
        public string Title { get; set; }
        public List<string> Tags { get; set; } = new List<string>();
        public string Body { get; set; }
        public string ImagePath { get; set; }

        public string TagsString => string.Join(", ", Tags);
    }

    public static class NoteStorage
    {
        private static readonly string FilePath = Path.Combine(FileSystem.Current.AppDataDirectory, "notes.json");

        // Сохранение заметок в JSON
        public static void SaveNotes(ObservableCollection<Note> notes)
        {
```

```
            var json = JsonSerializer.Serialize(notes, new JsonSerializerOptions { WriteIndented = true });
            File.WriteAllText(FilePath, json);
        }

        // Загрузка заметок из JSON
        public static ObservableCollection<Note> LoadNotes()
        {
            if (File.Exists(FilePath))
            {
                var json = File.ReadAllText(FilePath);
                return JsonSerializer.Deserialize<ObservableCollection<Note>>(json) ?? new ObservableCollection<Note>();
            }

            return new ObservableCollection<Note>();
        }
    }

    public static class PremState
    {
        public static bool IsPremiumPurchased { get; set; } = false;
    }
}

public partial class PremiumPurchasePage : ContentPage
{
    private readonly Action _onPremiumPurchased;

    public PremiumPurchasePage(Action onPremiumPurchased)
    {
        InitializeComponent();
        _onPremiumPurchased = onPremiumPurchased;
    }

    private async void OnBuyPremiumClicked(object sender, EventArgs e)
    {
        PremState.IsPremiumPurchased = true;
```

```
            _onPremiumPurchased?.Invoke();
            await Navigation.PopAsync();
        }
    }

    <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 x:Class="MauiApp1.PremiumPurchasePage"
                 Title="Оформление премиума"
                 BackgroundColor="#8062bc">

        <VerticalStackLayout Padding="20" Spacing="20"
VerticalOptions="Center">
            <Label Text="Купить премиум — 11,8 BYN"
                   FontSize="Large"
                   TextColor = "#E0FFFF"
                   HorizontalOptions="Center" />
            <Button Text="Купить" Clicked="OnBuyPremiumClicked"/>
        </VerticalStackLayout>
    </ContentPage>

    public partial class GenMenu : ContentPage
    {
          public GenMenu()
          {
                InitializeComponent();
          }

        private async void OnButtonClickedDice(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new Dice());
        }

        private async void OnButtonClickedItem(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ItemGen());
        }

        private async void OnButtonClickedCharacter(object sender, EventArgs e)
```

```
        {
            await Navigation.PushAsync(new CharacterGen());
        }

        private async void OnButtonClickedPlace(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PlaceGen());
        }
    }

    <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 x:Class="MauiApp1.GenMenu"
                 Title="Инструменты">

        <ScrollView>

            <VerticalStackLayout>

                <Button Text="Бросок кости" Clicked="OnButtonClickedDice"
CommandParameter="Журнал" FontSize="30"/>
                <Button Text="Предметы" Clicked="OnButtonClickedItem"
CommandParameter="Заметки" FontSize="30"/>
                <Button Text="Персонажи" Clicked="OnButtonClickedCharacter"
CommandParameter="Генераторы" FontSize="30"/>
                <Button Text="Места" Clicked="OnButtonClickedPlace"
CommandParameter="Генераторы" FontSize="30"/>

            </VerticalStackLayout>

        </ScrollView>
    </ContentPage>

    public partial class Dice : ContentPage
    {
        private readonly List<string> _rollHistory;
        private const string HistoryKey = "DiceRollHistory";

        public Dice()
```

```
        {
            InitializeComponent();
            _rollHistory = LoadHistory();
            HistoryListView.ItemsSource = _rollHistory;
        }

        private void OnRollButtonClicked(object sender, EventArgs e)
        {
            string input = DiceInput.Text;
            if (string.IsNullOrWhiteSpace(input))
            {
                ResultLabel.Text = "Введите комбинацию кубиков.";
                return;
            }

            try
            {
                int result = RollDice(input, out string breakdown);
                ResultLabel.Text = $"Результат: {result}\n({breakdown})";
                AddToHistory($"{input} → {result} ({breakdown})");
            }
            catch (Exception ex)
            {
                ResultLabel.Text = $"Ошибка: {ex.Message}";
            }
        }

        private int RollDice(string input, out string breakdown)
        {
            var random = new Random();
            int total = 0;
            var components = input.Split(new[] { '+', '-' },
StringSplitOptions.RemoveEmptyEntries)
                                  .Select(x => x.Trim());

            breakdown = "";

            foreach (var component in components)
            {
```

```
        int sign = input.Contains($"-{component}") ? -1 : 1;

        if (component.Contains("d"))
        {
            var parts = component.Split('d');
            int count = parts[0] == "" ? 1 : int.Parse(parts[0]);
            int sides = int.Parse(parts[1]);

            for (int i = 0; i < count; i++)
            {
                int roll = random.Next(1, sides + 1);
                total += roll * sign;
                breakdown += $"{roll} ";
            }
        }
        else
        {
            int modifier = int.Parse(component);
            total += modifier * sign;
            breakdown += $"{modifier * sign} ";
        }
    }

    return total;
}

private void AddToHistory(string rollResult)
{
    if (_rollHistory.Count >= 10)
    {
        _rollHistory.RemoveAt(0);
    }

    _rollHistory.Add(rollResult);
    SaveHistory();
    HistoryListView.ItemsSource = null;
    HistoryListView.ItemsSource = _rollHistory;
}
```

```
    private void SaveHistory()
    {
        Preferences.Set(HistoryKey, string.Join(";", _rollHistory));
    }

    private List<string> LoadHistory()
    {
        string savedHistory = Preferences.Get(HistoryKey, string.Empty);
        return string.IsNullOrEmpty(savedHistory)
            ? new List<string>()
            : savedHistory.Split(';').ToList();
    }

    private void OnClearHistoryClicked(object sender, EventArgs e)
    {
        _rollHistory.Clear();
        HistoryListView.ItemsSource = null;
        Preferences.Remove(HistoryKey);
        ResultLabel.Text = "История очищена.";
    }
}
```

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MauiApp1.Dice"
             Title="Бросок костей">

    <Grid BackgroundColor="#452988">

        <ScrollView  VerticalOptions="Start">

            <VerticalStackLayout Padding="10" Spacing="10">

                <!-- Поле ввода -->
                <Entry x:Name="DiceInput" Placeholder="Введите комбинацию (например, d6 + 2d20 + 5)" PlaceholderColor="#8062bc" TextColor="#E0FFFF" Keyboard="Text" />
```

```
            <Button x:Name="RollButton" Text="Бросить кубики"
Clicked="OnRollButtonClicked" />
            <Button Text="Очистить историю"
Clicked="OnClearHistoryClicked" Margin="0,5,0,0" />

            <!-- Результат -->
            <Label x:Name="ResultLabel" Text="Результат появится здесь."
FontSize="16" TextColor="#E0FFFF" HorizontalOptions="Start" />

            <!-- История -->
            <HorizontalStackLayout>

              <Label Text="История бросков:" FontAttributes="Bold"
TextColor="#E0FFFF" HorizontalOptions="StartAndExpand" />

            </HorizontalStackLayout>

            <ListView x:Name="HistoryListView">

              <ListView.ItemTemplate>

                <DataTemplate>

                  <ViewCell>

                    <StackLayout Padding="5"
BackgroundColor="#191970">

                      <Label Text="{Binding .}" FontSize="14"
Padding="6" TextColor="#E0FFFF"/>

                    </StackLayout>

                  </ViewCell>

                </DataTemplate>

              </ListView.ItemTemplate>
```

```
            </ListView>

        </VerticalStackLayout>

    </ScrollView>

</Grid>

</ContentPage>

    private readonly string _filePath;

    public CharList()
    {
        InitializeComponent();

        // Путь к файлу для сохранения данных
        _filePath = Path.Combine(FileSystem.AppDataDirectory,
"character.json");

        // Загрузка данных при открытии страницы
        LoadCharacterData();
    }

    private async void OnSaveClicked(object sender, EventArgs e)
    {
        // Сбор данных из полей
        var characterData = new
        {
            Name = CharacterNameEntry.Text,
            Class = ClassEntry.Text,
            Level = LevelEntry.Text,
            Race = RaceEntry.Text,
            Health = new
            {
                Current = CurrentHealthEntry.Text,
                Max = MaxHealthEntry.Text
            },
            ArmorClass = ArmorClassEntry.Text,
```

```
            Stats = new
            {
                Strength = StrengthEntry.Text,
                Dexterity = DexterityEntry.Text,
                Constitution = ConstitutionEntry.Text,
                Intelligence = IntelligenceEntry.Text,
                Wisdom = WisdomEntry.Text,
                Charisma = CharismaEntry.Text
            },
            Equipment = new
            {
                Weapons = WeaponsEditor.Text,
                Treasures = TreasuresEditor.Text,
                Misc = MiscEquipmentEditor.Text
            },
            Spells = SpellsEditor.Text
        };

        try
        {
            // Сохранение данных в файл
            string json = JsonSerializer.Serialize(characterData, new JsonSerializerOptions { WriteIndented = true });
            File.WriteAllText(_filePath, json);

            await DisplayAlert("Сохранено", "Лист персонажа успешно сохранён!", "OK");
        }
        catch (Exception ex)
        {
            await DisplayAlert("Ошибка", $"Не удалось сохранить данные: {ex.Message}", "OK");
        }
    }


    private void LoadCharacterData()
    {
        if (!File.Exists(_filePath))
```

```
            return;

        try
        {
            string json = File.ReadAllText(_filePath);
            var characterData = JsonSerializer.Deserialize<CharacterData>(json);

            if (characterData != null)
            {
                CharacterNameEntry.Text = characterData.Name;
                ClassEntry.Text = characterData.Class;
                LevelEntry.Text = characterData.Level;
                RaceEntry.Text = characterData.Race;

                CurrentHealthEntry.Text = characterData.Health?.Current;
                MaxHealthEntry.Text = characterData.Health?.Max;

                ArmorClassEntry.Text = characterData.ArmorClass;

                StrengthEntry.Text = characterData.Stats?.Strength;
                DexterityEntry.Text = characterData.Stats?.Dexterity;
                ConstitutionEntry.Text = characterData.Stats?.Constitution;
                IntelligenceEntry.Text = characterData.Stats?.Intelligence;
                WisdomEntry.Text = characterData.Stats?.Wisdom;
                CharismaEntry.Text = characterData.Stats?.Charisma;

                WeaponsEditor.Text = characterData.Equipment?.Weapons;
                TreasuresEditor.Text = characterData.Equipment?.Treasures;
                MiscEquipmentEditor.Text = characterData.Equipment?.Misc;

                SpellsEditor.Text = characterData.Spells;
            }
        }
        catch (Exception ex)
        {
            DisplayAlert("Ошибка", $"Не удалось загрузить данные: {ex.Message}", "OK");
        }
    }
```

```
    private class CharacterData
    {
        public string Name { get; set; }
        public string Class { get; set; }
        public string Level { get; set; }
        public string Race { get; set; }
        public HealthData Health { get; set; }
        public string ArmorClass { get; set; }
        public StatsData Stats { get; set; }
        public EquipmentData Equipment { get; set; }
        public string Spells { get; set; }
    }

    private class HealthData
    {
        public string Current { get; set; }
        public string Max { get; set; }
    }

    private class EquipmentData
    {
        public string Weapons { get; set; }
        public string Treasures { get; set; }
        public string Misc { get; set; }
    }

    private class StatsData
    {
        public string Strength { get; set; }
        public string Dexterity { get; set; }
        public string Constitution { get; set; }
        public string Intelligence { get; set; }
        public string Wisdom { get; set; }
        public string Charisma { get; set; }
    }

public partial class PlaceGen : ContentPage
{
```

```
        private Dictionary<string, string> _objectDescriptions;
        private const string PremiumVisitedKey = "PremiumVisited";
        private const string PremiumOverlayVisibleKey = "PremiumOverlayVisible";
        private int buttonClickCount = 0;

        public PlaceGen()
        {
            InitializeComponent();
            _objectDescriptions = new Dictionary<string, string>();
        }

        private async void OnRoleSystemSelected(object sender, EventArgs e)
        {
            var selectedRoleSystem = RoleSystemPicker.SelectedItem?.ToString();

            if (string.IsNullOrEmpty(selectedRoleSystem))
            {
                ResultText.Text = "Выберите ролевую систему.";
                return;
            }

            // Загрузка JSON-файл
            await LoadJsonDataAsync(selectedRoleSystem);
            ResultText.Text = "Файл загружен. Нажмите 'Сгенерировать' для случайного выбора.";
        }

        private async Task LoadJsonDataAsync(string roleSystem)
        {
            string fileName = roleSystem switch
            {
                "Dungeons and Dragons" => "DnDPlace.json",
                "Rogue Trader" => "RougeTraderPlace.json",
                "Pathfinder" => "PathfindePlace.json"
            };

            _objectDescriptions.Clear();
```

```
            // Открытие JSON
            using var stream = await FileSystem.Current.OpenAppPackageFileAsync(Path.Combine(fileName));
            using var reader = new StreamReader(stream);
            var json = await reader.ReadToEndAsync();

            // Десериализация JSON в словарь
            var data = JsonSerializer.Deserialize<JsonData>(json);
            if (data?.Object != null)
            {
                _objectDescriptions = data.Object;
            }
        }

        private void OnGenerateButtonClicked(object sender, EventArgs e)
        {
            if (_objectDescriptions.Count == 0)
            {
                ResultText.Text = "Данные не загружены. Выберите ролевую систему и загрузите данные.";
                return;
            }

            var random = new Random();
            var randomEntry = _objectDescriptions.ElementAt(random.Next(_objectDescriptions.Count));
            ResultText.Text = $"{randomEntry.Key}\n\n{randomEntry.Value}";

            if (buttonClickCount < 3 && !PremState.IsPremiumPurchased)
            {
                buttonClickCount++;

                if (buttonClickCount == 3)
                {
                    PremiumOverlay.IsVisible = true;
                    Preferences.Set(PremiumOverlayVisibleKey, true);
                }
            }
        }
```

```
        private async void OnPurchasePremiumClicked(object sender, EventArgs
e)
        {
            await                                    Navigation.PushAsync(new
PremiumPurchasePage(OnPremiumPurchased));
        }

        private void OnPremiumPurchased()
        {
            Preferences.Set(PremiumVisitedKey, true);
            PremiumOverlay.IsVisible = false;
            Preferences.Set(PremiumOverlayVisibleKey, false);
            DisplayAlert("Премиум активирован", "Спасибо за приобретение
премиум!", "OK");
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();

            bool                    isPremiumOverlayVisible                =
Preferences.Get(PremiumOverlayVisibleKey, false);
            PremiumOverlay.IsVisible = isPremiumOverlayVisible;
        }

        public class JsonData
        {
            public Dictionary<string, string> Object { get; set; }
        }
    }

    <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 x:Class="MauiApp1.PlaceGen"
                 Title="Генератор локаций">

        <Grid BackgroundColor="#452988">
            <VerticalStackLayout Spacing="15">
```

```
            <!-- Выбор ролевой системы -->
            <Picker x:Name="RoleSystemPicker" Title="Выберите ролевую систему" SelectedIndexChanged="OnRoleSystemSelected" TextColor = "#E0FFFF" TitleColor="#E0FFFF">
                <Picker.ItemsSource>
                    <x:Array Type="{x:Type x:String}">
                        <x:String>Dungeons and Dragons</x:String>
                        <x:String>Rogue Trader</x:String>
                        <x:String>Pathfinder</x:String>
                    </x:Array>
                </Picker.ItemsSource>
            </Picker>

            <!-- Поле вывода -->
            <Editor x:Name="ResultText" IsReadOnly="True" HeightRequest="480" Placeholder="Здесь появится описание объекта" TextColor = "#E0FFFF" PlaceholderColor="#8062bc"/>

            <!-- Кнопка генерации -->
            <Button Text="Сгенерировать" Clicked="OnGenerateButtonClicked" Style="{StaticResource SmallButtonStyle}" TextColor = "#E0FFFF"/>

        </VerticalStackLayout>

        <!-- Окно премиума -->
        <Grid x:Name="PremiumOverlay" BackgroundColor="#80000000" IsVisible="False">
            <VerticalStackLayout VerticalOptions="Center" HorizontalOptions="Center" BackgroundColor="#6A5ACD" Padding="20" Spacing="20">
                <Label Text="Тестовый доступ завершен. Для дальнейшей работы необходимо оформить премиум"
                       FontSize="Medium"
                       HorizontalOptions="Center"
                       HorizontalTextAlignment="Center"
                       TextColor="#E0FFFF"/>
```

```
                <Button Text="Приобрести премиум" Clicked="OnPurchasePremiumClicked" Style="{StaticResource SmallButtonStyle}"/>
            </VerticalStackLayout>
        </Grid>

    </Grid>

</ContentPage>

public partial class ItemGen : ContentPage
{
    private Dictionary<string, Dictionary<string, List<Item>>> _itemsData;
    private const string PremiumVisitedKey = "PremiumVisited";
    private const string PremiumOverlayVisibleKey = "PremiumOverlayVisible";
    private int buttonClickCount = 0;

    public ItemGen()
    {
        InitializeComponent();
        _itemsData = new Dictionary<string, Dictionary<string, List<Item>>>();
    }

    private async void OnRoleSystemSelected(object sender, EventArgs e)
    {
        var selectedRoleSystem = RoleSystemPicker.SelectedItem?.ToString();

        if (string.IsNullOrEmpty(selectedRoleSystem))
        {
            ResultText.Text = "Выберите ролевую систему.";
            return;
        }

        await LoadJsonDataAsync(selectedRoleSystem);
        ResultText.Text = "Файл загружен. Выберите тип предмета.";
        TypePicker.IsEnabled = true;
        RarityPicker.IsEnabled = false;
```

```
            GenerateButton.IsEnabled = true;
            TypePicker.SelectedItem = null;
            RarityPicker.SelectedItem = null;
        }

        private async Task LoadJsonDataAsync(string roleSystem)
        {
            string fileName = roleSystem switch
            {
                "Dungeons and Dragons" => "DnDItems.json",
                "Pathfinder" => "PathfinderItem.json",
                "Rogue Trader" => "RogueTraderItems.json",
            };

            if (fileName == null)
            {
                ResultText.Text = "Файл не найден для выбранной ролевой системы.";
                return;
            }

            try
            {
                _itemsData.Clear();

                // Открытие JSON
                using var stream = await FileSystem.Current.OpenAppPackageFileAsync(fileName);
                using var reader = new StreamReader(stream);
                var json = await reader.ReadToEndAsync();

                // Десериализация в список предметов
                var items = JsonSerializer.Deserialize<List<Item>>(json);

                if (items != null)
                {
                    // Группировка по типу и редкости
                    _itemsData = items
                        .GroupBy(i => i.Type)
```

```
                .ToDictionary(
                    g => g.Key,
                    g => g.GroupBy(item => item.Rarity)
                        .ToDictionary(
                            rg => rg.Key,
                            rg => rg.ToList()
                        )
                );

                TypePicker.ItemsSource = _itemsData.Keys.ToList();
                ResultText.Text = "Данные успешно загружены. Выберите тип
предмета.";
            }
        }
        catch (Exception ex)
        {
            ResultText.Text = $"Ошибка загрузки файла: {ex.Message}";
        }
    }

    private void OnTypeSelected(object sender, EventArgs e)
    {
        if (TypePicker.SelectedItem is string selectedType &&
            _itemsData.TryGetValue(selectedType, out var rarityData))
        {
            RarityPicker.ItemsSource = rarityData.Keys.ToList();
            RarityPicker.IsEnabled = true;
            RarityPicker.SelectedItem = null;
            ResultText.Text = "Выберите редкость предмета.";
        }
    }

    private void OnRaritySelected(object sender, EventArgs e)
    {
        if (RarityPicker.SelectedItem != null)
        {
            ResultText.Text = "Нажмите 'Сгенерировать' для создания
предмета.";
        }
```

```
        }

        private void OnGenerateButtonClicked(object sender, EventArgs e)
        {
            if (_itemsData.Count == 0)
            {
                ResultText.Text = "Данные не загружены. Выберите ролевую систему и загрузите данные.";
                return;
            }

            // Проверка, выбран ли Alignment и Rarity
            string selectedType = TypePicker.SelectedItem?.ToString();
            string selectedRarity = RarityPicker.SelectedItem?.ToString();

            List<Item> availableItems;

            if (selectedType == null && selectedRarity == null)
            {
                // Если не выбран Type и Rarity
                availableItems = _itemsData.Values.SelectMany(rarityDict => rarityDict.Values).SelectMany(items => items).ToList();
            }
            else if (selectedType != null && selectedRarity == null)
            {
                // Если выбран только Type
                if (_itemsData.TryGetValue(selectedType, out var rarityData))
                {
                    availableItems = rarityData.Values.SelectMany(items => items).ToList();
                }
                else
                {
                    ResultText.Text = "Нет данных для выбранного типа.";
                    return;
                }
            }
            else if (selectedType != null && selectedRarity != null)
            {
```

```
            // Если выбраны Rarity и Type
            if (_itemsData.TryGetValue(selectedType, out var rarityData) &&
                rarityData.TryGetValue(selectedRarity, out var items) &&
                items.Any())
            {
                availableItems = items;
            }
            else
            {
                ResultText.Text = "Нет предметов, соответствующих выбранным критериям.";
                return;
            }
        }
        else
        {
            ResultText.Text = "Выберите параметры (тип и/или редкость).";
            return;
        }

        if (availableItems.Any())
        {
            var random = new Random();
            var randomItem = availableItems[random.Next(availableItems.Count)];

            ResultText.Text = $"Название: {randomItem.Name}\n" +
                              $"Тип: {randomItem.Type}\n" +
                              $"Редкость: {randomItem.Rarity}\n" +
                              $"Цена: {randomItem.Cost}\n" +
                              $"Описание: {randomItem.Description}";
        }
        else
        {
            ResultText.Text = "Нет предметов, соответствующих выбранным критериям.";
        }

        if (buttonClickCount <= 3 && PremState.IsPremiumPurchased == false)
```

```
            {
                buttonClickCount++;

                if (buttonClickCount == 3)
                {
                    PremiumOverlay.IsVisible = true;
                    Preferences.Set(PremiumOverlayVisibleKey, true);
                }
            }
        }
        private async void OnPurchasePremiumClicked(object sender, EventArgs
e)
        {
            await                                    Navigation.PushAsync(new
PremiumPurchasePage(OnPremiumPurchased));
        }

        private void OnPremiumPurchased()
        {
            Preferences.Set(PremiumVisitedKey, true);
            PremiumOverlay.IsVisible = false;
            Preferences.Set(PremiumOverlayVisibleKey, false);
            DisplayAlert("Премиум активирован", "Спасибо за приобретение
премиум!", "OK");
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();

            bool                    isPremiumOverlayVisible                =
Preferences.Get(PremiumOverlayVisibleKey, false);
            PremiumOverlay.IsVisible = isPremiumOverlayVisible;
        }

        private void OnClearButtonClicked(object sender, EventArgs e)
        {
            RoleSystemPicker.SelectedItem = null;
            TypePicker.SelectedItem = null;
```

```
            RarityPicker.SelectedItem = null;
            ResultText.Text = string.Empty;
        }

        public class Item
        {
            public string Type { get; set; }
            public string Rarity { get; set; }
            public int Cost { get; set; }
            public string Name { get; set; }
            public string Description { get; set; }
        }
    }

    <Grid BackgroundColor="#452988">
        <VerticalStackLayout Padding="20" Spacing="20">

            <!-- Ролевая система -->
            <Picker x:Name="RoleSystemPicker" Title="Выберите ролевую
систему" SelectedIndexChanged="OnRoleSystemSelected"
TextColor="#E0FFFF" TitleColor="#E0FFFF">
                <Picker.ItemsSource>
                    <x:Array Type="{x:Type x:String}">
                        <x:String>Dungeons and Dragons</x:String>
                        <x:String>Rogue Trader</x:String>
                        <x:String>Pathfinder</x:String>
                    </x:Array>
                </Picker.ItemsSource>
            </Picker>

            <!-- Параметры -->
            <Picker x:Name="TypePicker" Title="Выберите тип предмета"
IsEnabled="False" SelectedIndexChanged ="OnTypeSelected" TextColor =
"#E0FFFF" TitleColor="#E0FFFF"/>
            <Picker x:Name="RarityPicker" Title="Выберите редкость предмета"
IsEnabled="False" SelectedIndexChanged ="OnRaritySelected" TextColor =
"#E0FFFF" TitleColor="#E0FFFF"/>

            <!-- Поле вывода -->
```

```
            <Editor x:Name="ResultText" IsReadOnly="True" HeightRequest="280" Placeholder="Результат выбора появится здесь..." TextColor = "#E0FFFF" PlaceholderColor="#8062bc" />

            <!-- Кнопка генерации -->
            <Button x:Name="GenerateButton" Text="Сгенерировать" IsEnabled="False" Clicked="OnGenerateButtonClicked" Style="{StaticResource SmallButtonStyle}" />

            <!-- Кнопка отчистки -->
            <Button x:Name="ClearButton" Text="Очистить" Clicked="OnClearButtonClicked" Style="{StaticResource SmallButtonStyle}" />

        </VerticalStackLayout>

        <!-- Окно премиума -->
        <Grid x:Name="PremiumOverlay" BackgroundColor="#80000000" IsVisible="False">
            <VerticalStackLayout VerticalOptions="Center" HorizontalOptions="Center" BackgroundColor="#6A5ACD" Padding="20" Spacing="20">
                <Label Text="Тестовый доступ завершен. Для дальнейшей работы необходимо оформить премиум"
                       FontSize="Medium"
                       HorizontalOptions="Center"
                       HorizontalTextAlignment="Center"
                       TextColor="#E0FFFF"/>
                <Button Text="Приобрести премиум" Clicked="OnPurchasePremiumClicked" Style="{StaticResource SmallButtonStyle}"/>
            </VerticalStackLayout>
        </Grid>

    </Grid>
```

Обозначение	Наименование	Примечание
	Текстовые документы	
БГУИР ДП 1-40 01 01 01 100 ПЗ	Пояснительная записка	121 с
	Отзыв руководителя	1 с
	Рецензия	1 с
	Отчёт о проверке на заимствования	1 с
	Графические документы	
ГУИР.181071.001 ПЛ	Диаграмма вариантов использования	Формат А1
	Плакат	
ГУИР.181071.002 ПЛ	Диаграмма деятельности	Формат А1
	Плакат	
ГУИР.181071.003 СА	Общий алгоритм работы программы	Формат А1
	Схема алгоритма	
ГУИР.181071.004 СА	Модуль заметок и журнала	Формат А1
	Схема алгоритма	
ГУИР.181071.005 СА	Генератора бросков игральных костей	Формат А1
	Схема алгоритма	
ГУИР.181071.006 СА	Модуль генерации	Формат А1
	Схема алгоритма	

					БГУИР ДП 1-40 01 01 01 100 Д1
Изм	Лист	№ докум.	Подп.	Дата	
Разраб.		Малашкевич И Г			Мобильное приложение для проведения настольных ролевых игр Ведомость дипломного проекта
Пров.		Сицко В А			
Т.контр.		Сицко В А			
Н.контр.		Шелягович А С			
Утв.					

Лит.	Лист	Листов
Т	121	121

Кафедра ИСиТ,
гр. 181071