МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем и цифровых технологий

Работа допущена к защите
_____Руководитель
«_1_»_____04_____20_23_г.

КУРСОВОЙ ПРОЕКТ

по дисциплине «Проектная деятельность»

на тему: «Разработка мобильного приложения для развития когнитивных способностей»

Студент _____ Бессонов М.П.

Шифр 191009

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 92ПГ

Руководитель _____ Конюхова О.В.

Оценка: «_отлично_» Дата _1.04.2023_

Орел 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем и цифровых технологий

УТВЕРЖДАЮ:

_____ и. о. зав. кафедрой

«14» февраля 2023 г.

ЗАДАНИЕ
на курсовой проект

по дисциплине «Проектная деятельность»

Студент Бессонов М.П. Шифр 191009

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 92ПГ

1 Тема курсового проекта

«Разработка мобильного приложения для развития когнитивных способностей»

2 Срок сдачи студентом законченной работы «20» апреля 2023

3 Исходные данные

Литературные источники, периодические издания и интернет-ресурсы по тематике курсового проекта.

4 Содержание курсового проекта

Анализ задач и формулировка требований к разработке мобильного приложения для развития когнитивных способностей

Анализ требований и определение спецификаций мобильного приложения для развития когнитивных способностей

Проектирование мобильного приложения для развития когнитивных способностей

Реализация разработанного мобильного приложения для развития когнитивных способностей

5 Отчетный материал курсового проекта

Пояснительная записка курсового проекта, приложение, презентация.

Руководитель _____ Конюхова О.В.

Задание принял к исполнению: «13» февраля 2023

Подпись студента_____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Ежедневно человек совершает привычные для него действия: общается, ходит на работу и учебу, изучает окружающий мир. За реализацию этих действий отвечает мозг человека. Современный мир устроен таким образом, что наш мозг с каждым годом должен обрабатывать все больше и больше информации.

Мозг – это мышца, то есть его, как и любую мышцу в теле человека, можно тренировать. До активного внедрения в жизнь человека информационных технологий для этого использовались головоломки, кроссворды, научные журналы с интеллектуальными задачами. Благодаря использованию компьютеров и мобильных устройств у множества людей появилась возможность развиваться при помощи огромного количества цифровых версий игр, что расширило потенциальные возможности по развитию когнитивных способностей.

Мобильные приложения для развития пластичности мозга – возможность сохранять производительность в различных делах с помощью повышения уровня мозговой активности. Главная проблема людей, занимающихся укреплением связей между нервными клетками мозга – нехватка времени и мотивации. Приложение для развития когнитивных способностей будет находиться на мобильном устройстве пользователя, что позволит уделять несколько минут каждый день на использование его функционала вне зависимости от местонахождения.

Следовательно, основная задача мобильного приложения для развития когнитивных способностей – предоставление возможности тренировать и развивать способности своего мозга в любом месте и в любое время.

Объектом исследования нашей работы является когнитивное развитие личности с помощью мобильных приложений.

Предметом исследования нашей работы являются сценарии мобильных приложений, направленные на развитие памяти, внимания, вычислительных способностей, а также средства и методы создания таких приложений.

Целью курсового проекта является помощь в организации тренировок мозга пользователей за счет использования разрабатываемого мобильного приложения.

Для достижения цели нам необходимо решить следующие задачи:

1) анализ предметной области, формирование функциональных требований к приложению для развития когнитивных способностей;

2) обзор и анализ аналогов приложения для развития когнитивных способностей;

3) анализ требований и определение спецификаций приложения для развития когнитивных способностей;

4) проектирование приложения для развития когнитивных способностей;

5) реализация и тестирование приложения для развития когнитивных способностей.

1 АНАЛИЗ ЗАДАЧИ И ФОРМУЛИРОВКА ТРЕБОВАНИЙ К РАЗРАБОТКЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ РАЗВИТИЯ КОГНИТИВНЫХ СПОСОБНОСТЕЙ

1.1 Описание предметной области мобильного приложения для развития когнитивных способностей

Наш мозг нуждается в постоянной тренировке, а одних знаний часто бывает недостаточно для успешной жизнедеятельности. В этом могут помочь различные методики развития когнитивных способностей, то есть способностей к мышлению, запоминанию данных и другим способам обработки и анализа информации [16].

Научно доказано, что любая информация лучше всего усваивается в игровой форме [5], поэтому до массового внедрения информационных технологий составлялись кроссворды, судоку, изобретались головоломки. С появлением цифровых устройств многие тренажеры такого типа были перенесены на электронные устройства. Сейчас можно легко найти целые сборники развивающих игр в своем мобильном телефоне, компьютере и на других устройствах.

Например, в разных магазинах мобильных приложений можно найти приложения под каждый конкретный навык, начиная с устного счета и заканчивая играми на скорость реакции. Некоторые разработчики объединяют их в целые сборники развивающих игр, а некоторые превращают их в один из источников дохода, начиная продавать ежемесячные подписки на доступ к определенным категориям игр или рассылая рекламу при регистрации на сайте [10].

В настоящее время, чтобы сохранять ясность ума можно за утренней чашкой кофе, в перерыве на обед или в поездке на общественном транспорте просто решать задачи определенного типа в своем телефоне. Именно создание такого приложения мы будем рассматривать в нашей работе.

Наиболее популярными операционными системами, разработанными под мобильные устройства являются iOS от компании Apple, Android от компании

Google, а также Windows Phone компании Microsoft. Лидирующие позиции уже не первый год занимает ОС Android, так как согласно проведенным исследованиям доля принадлежащего ей мирового рынка составляет 75% [7]. Это говорит о том, что мобильные приложения, написанные под названную выше операционную систему потенциально могут устанавливаться и использоваться большим количеством пользователей.

Доказательством этого является магазин мобильных приложений Google Play для системы Android. В настоящий момент он выдает более ста различных приложений, а также показывает отзывы людей по большинству запросов пользователей. Не становится исключением и рассматриваемая нами тема.

Большое количество игр на развитие памяти («СЧС», «Мнемонист» и др.), устного счета («Счет в уме», «Математические хитрости» и др.) и других мозговых процессов, участвующих в познании окружающего мира, а также количество пользователей этих приложений, которое может достигать более ста тысяч скачиваний и сотен положительных отзывов, касающихся использования приложения, в совокупности с высокими оценками говорят о популярности игровых приложений, направленных на совершенствование мозга [10].

Некоторые разработчики объединяют эти игры, создавая целые отдельные категории из развивающих игр (например, сборник игр для развития памяти «Mnemocon»). Разумеется, среди таких приложений самыми популярными будут крупные коммерческие проекты с миллионами установок (среди зарубежных: «NeuroNation», «CogniFit»; среди российских: «Викиум»), объединяющие десятки игр различной направленности [10].

Установка нескольких отдельных развивающих игр на свое мобильное устройство может оказаться не самым эффективным решением из-за трат памяти или из-за большого количества сторонних приложений среди которых созданная папка с развивающими играми может затеряться. Создание аккаунта у крупных коммерческих приложений иногда может не соответствовать цели пользователя, который просто хотел время от времени проводить разминку своего мозга, а не покупать подписку или получать рекламные рассылки после прохождения игр.

Для таких целей как раз могут подойти некоммерческие приложения, содержащие больше одной развивающей мозг игры («BrainExer», «Brain Gym» и др.) [2, 10]

В данной работе будет рассматриваться мобильное приложение для развития когнитивных способностей, благодаря которому можно расширять потенциал своего мозга при помощи набора специализированных упражнений вне зависимости от местонахождения пользователя.

Планируется создать такое приложение, чтобы оно не было привязано к какому–либо одному когнитивному навыку (например, развитие способности запоминания информации). В нашем курсовом проекте мы хотели бы расширить область применения приложений такого рода, то есть включить в него различные наборы развивающих игр, направленные на совершенствование различных когнитивных способностей человека.

Такой поход обычно включает выбор пользователем той тренировки, которая нужна игроку на данном этапе. Каждая из развивающих игр, включенная в тот или иной блок, должна быть описана, чтобы пользователь четко понимал, на что конкретно направлена данная тренировка и как ее выполнять.

Воспользоваться мобильным приложением должна быть возможность у любого желающего. Для этого, как правило, необходимо только установить его на свое мобильное устройство.

Также в приложении рассматриваемого содержания могут быть доступны следующие виды формирования тренировок:

1) тренировка в конкретной категории, которая в свою очередь включает в себя возможность попрактиковаться в любом выбранном упражнении, пройти его в специальном или в классическом режиме;

2) тренировка в режиме испытания, то есть последовательный циклический запуск нескольких игр. Это позволяет пользователям развить свою способность переключения между различными задачами. Формирование испытаний может происходить автоматически или задаваться пользователем.

Рассмотрим подробнее тренировку в конкретной категории. В таком случае в приложении представлено несколько категорий, каждая из которых

включает в себя различное количество развивающих игр. Пользователь ничем не ограничен, ведь он может сам составлять себе тренировку, включая в нее необходимые развивающие игры. А так как к каждой игре приложено описание ее прохождения, пользователь может не беспокоиться о том, что не сможет разобраться в том или ином упражнении.

Как правило, в сборниках, содержащих развивающие игры, присутствует возможность постепенного увеличения сложности. По мере прохождения той или иной тренировки прогресс игрока в них будет расти, что со временем позволит открыть дополнительные режимы, позволяющие взглянуть на процесс тренировки по–новому, или открыть новые уровни для прохождения, усложняющие текущую тренировку. Такой подход в целом будет поддерживать заинтересованность пользователя в использовании приложения.

Одним из способов реализации такого функционала, который позволит не отвлекаться от основной сути приложения, но в то же время решит вопрос обработки достигнутого прогресса в игре – введение внутриигровой валюты как платы за открытие нового функционала. Такая реализация является наиболее подходящей при разработке приложения как набора различных развивающих упражнений, так как позволит награждать пользователя за прохождение той или иной игры.

Еще один способ, который позволит мобильному приложению такого типа удерживать внимание пользователя, возможность отслеживания своего прогресса в каждом упражнении при помощи таблицы рекордов. Она может содержать все наименования развивающих игр и максимальный результат, полученный в каждой из них.

Это позволит пользователю самостоятельно решать, на какой конкретно навык ему необходимо обратить внимание в данный момент, а главное, стремиться совершенствовать свои способности, так как предложенная реализация представляет пользовательский прогресс в количественном эквиваленте.

1.2 Обзор аналогов мобильного приложения для развития когнитивных способностей

Перед началом разработки мобильного приложения, предназначенного для развития когнитивных способностей, произведем поиск различных прототипов и аналогов с целью выявления и уточнения ряда основных требований. Данная предметная область уже содержит ряд решений, среди которых есть как крупные коммерческие проекты, так и бесплатные аналоги. Но, так как большинство аналогов имеют схожую структуру, рассмотрим некоторые из них. При этом основной упор сделаем на изучение функционала некоммерческих приложений.

Первое приложение – это «BrainExer 2.0», в котором собраны различные развивающие игры, распределенные по категориям [2]. Приложение имеет положительные отзывы в магазине мобильных приложений Google Play (средняя оценка 4,8 из 5). Интерфейс приложения изображен на рисунке 1.

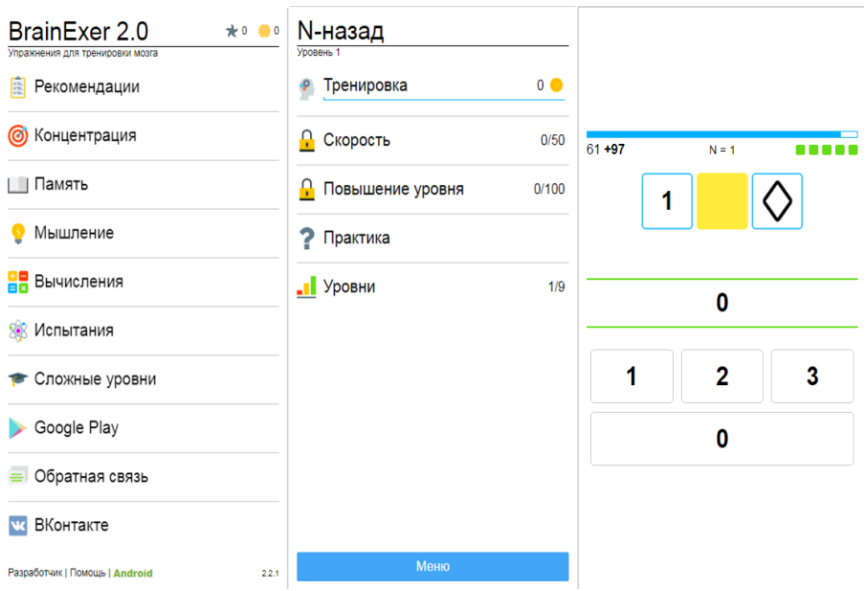

Рисунок 1 – Приложение «BrainExer 2.0»

В результате анализа были выявлены следующие достоинства приложения:

1) большое количество развивающих игр, разделенных на четыре категории;

2) простая система рекомендаций, отображающая развивающие игры, в которых прогресс достигнут меньше всего (без отправки уведомлений на мобильное устройство пользователя);

3) наличие дополнительных режимов (режим «Практика» для тренировки упражнения и др.) в каждой развивающей игре;

4) возможность повышения уровня сложности за счет достижения прогресса;

5) наличие режима автоматической тренировки типа «испытание»

6) система открытия новых игр за внутриигровую валюту.

Среди недостатков приложения можно выделить следующие:

1) отсутствие возможности создания тренировки;

2) Малое количество механик, поддерживающих интерес и удерживающих пользователя в приложении (на основе некоторой части отзывов в магазине приложений Google Play) [2];

3) содержание таблицы прогресса включает только результат пяти лучших попыток, что может мешать пользователю отслеживать свой результат в реальном времени.

Следующее приложение – это «Тренер мозга», которое включает в себя 24 игры, направленные на развитие различных когнитивных способностей [12]. Приложение также имеет положительные отзывы в магазине мобильных приложений Google Play (средняя оценка 4,2 из 5). Интерфейс приложения изображен на рисунке 2.

Рисунок 2 – Приложение «Тренер мозга»

В результате анализа были выявлены следующие достоинства приложения:

1) большое количество развивающих игр;

2) возможность повышения уровня сложности за счет достижения прогресса;

3) подробное визуальное описание каждой тренировки;

4) система открытия новых игр за внутриигровую валюту.

Среди недостатков приложения можно выделить следующие:

1) отсутствие четкой классификации упражнений по категориям;

2) прогресс можно отслеживать только по своему максимальному результату (без указания даты и времени его получения);

3) отсутствие возможности создания тренировки;

4) отсутствие дополнительных режимов (усложняющих стандартный режим тренировки для поддержания интереса [6]) в развивающих играх;

5) Малое количество механик, поддерживающих интерес и удерживающих пользователя в приложении (на основе некоторой части отзывов в магазине приложений Google Play) [12].

Как мы видим, оба аналога представляют собой сборники развивающих игр, каждую из которых можно проходить. Все эти игры направлены на совершенствование разных навыков, поэтому в таком приложении обязательно должно быть разграничение на категории.

Отзывы к обоим аналогам показывают, что наличие внутриигровой валюты является хорошим способом удержания внимания пользователя и его стремления к прогрессу в развивающих играх за счет открытия нового функционала [2, 12].

Подробное отслеживание своих результатов является важной функцией в приложениях такого рода [6], поэтому при реализации нашего приложении будем кроме максимальных результатов в каждой игре хранить последние пять результатов с указанием даты и времени их получения, а также их визуальное представление. Это позволит пользователю самостоятельно решать, какой перерыв между тренировками ему необходимо сделать.

Среди недостатков двух аналогов можно выделить общие следующие:

– отсутствие возможности создания тренировки. И хотя приложение «BrainExer 2.0» может сгенерировать тренировку из набора упражнений, оно не позволяет пользователю сформировать ее самому. Поэтому в нашем приложении будут присутствовать оба вида формирования тренировок;

– отсутствие проработанной системы мотивации пользователя к постоянному использованию приложения, то есть недостаточное количество механизмов, подталкивающих пользователя к ежедневному использованию приложения (на основе некоторой части отзывов в магазине приложений Google Play) [2, 12].

Добавление системы прогресса является интересным, но не единственным способом удержания внимания [6]. Кроме этой механики в нашем приложении будет реализована механика выполнения ежедневных процедурно–генерируемых заданий, за которые пользователь сможет получать награды, а также ежедневная награда за вход. Также реализуем в нашем приложении отображение интересных фактов о развитии мозга (после прохождения развивающей игры), так как такая механика может стимулировать человека использовать приложение дольше, чтобы ознакомиться с гораздо большим числом фактов.

Кроме этого реализуем в нашем приложении внутриигровой магазин, содержащий товары, не влияющие на сам игровой процесс, но позволяющие несколько быстрее достигать в упражнениях необходимого прогресса.

Последнее, что стоит отметить, распространение выбранных аналогов происходит совершенно бесплатно. Наше приложение будет следовать такому же принципу.

На основании сделанных нами выводов составим таблицу аналогов (Таблица 1).

Таблица 1 – Обзор аналогов

Виды приложений / Критерии	Разрабатываемое приложение	«BrainExer 2.0»	«Тренер мозга»
1	2	3	4
Возможность выбрать развивающую игру	+	+	+

Продолжение таблицы 1

1	2	3	4
Наличие рекомендательной системы	–	+	–
Наличие механик, позволяющих пользователю развиваться в приложении	+	+	+
Возможность создания своей тренировки	+	–	–
Описание развивающих игр	+ (Отсутствует видеоописание)	+ (Отсутствует видеоописание)	+
Возможность подробного отслеживания своего прогресса в каждой игре	+	–	–
Механики, поддерживающие интерес пользователя к приложению	+	–	–
Бесплатность распространения	+	+	+

Так как рекомендательная система, рассылающая уведомления, как правило, сохраняет их на мобильном устройстве пользователя, что будет тратить дополнительные ресурсы мобильного устройства пользователя, а реализация отдельного окна с отображением мало используемых игр применительно к разработке нашего приложения будет мало актуальна на данном этапе, опустим ее реализацию в нашей работе.

Дополнительное внедрение подробного описания упражнений (с помощью видео) также увеличит конечный объем реализуемого приложения, но повысит наглядность. Так как для понимания сути задания может быть достаточно одного текстового описания [2], также опустим реализацию этого функционала.

Для упрощения представления итоговой таблицы поместим следующий описанный выше функционал в один критерий «Механики, поддерживающие интерес пользователя к приложению»: ежедневные процедурно–генерируемые задания, ежедневный бонус, интересные факты после прохождения развивающей игры, внутриигровой магазин.

На основании выше приведенного анализа предметной области и аналогов, можно дать следующую краткую характеристику нашего мобильного приложения: программный продукт, представляя собой совокупность различных развивающих игр, распределенных по категориям, будет вовлекать человека в процесс использования приложения за счет наличия следующих особенностей:

1) наличие подробных описаний каждого упражнения (при этом не только на русском языке, но и на английском языках для возможного привлечения большей аудитории);

2) возможность отслеживания прогресса в каждой развивающей игре;

3) возможность создания собственной тренировки, а также прохождения автоматически сгенерированной тренировки;

4) наличие внутриигровой валюты, привносящее в игру дополнительный интерес и стимулирующее пользователя развивать свои когнитивные навыки;

5) наличие системы увеличения сложности в развивающих играх.

Подробное описание возможностей нашего приложения будет описано при формулировании функциональных требований.

1.3 Исследование метода выбора развивающих упражнений в мобильном приложении для развития когнитивных способностей

Перед началом проектирования и разработки необходимо решить, какие виды развивающих игр будут реализованы в нашем мобильном приложении и почему. Ведь как указывалось ранее, мы хотели бы создать такой программный продукт, который был бы направлен на развитие различных когнитивных способностей человека.

Виды этих развивающих игр напрямую зависят от того, чего пользователь хочет добиться. Для выявления закономерности оценим еще раз выделенные аналоги, а также ознакомимся с одним классом методик развития мозга.

Приложение «BrainExer 2.0» содержит 4 категории: концентрация, память, мышление и вычисления. В целом, можно отметить, что такое решение

классификации развивающих игр позволяет охватить большинство важных для человека когнитивных способностей (за исключением воображения) [16].

Приложение «Тренер мозга» не имеет четкой классификации, но при детальном анализе представленных игр можно понять, что большинство из них направлены на тренировку памяти и концентрации (примерно две трети от всех представленных в приложении игр).

Также стоит упомянуть один достаточно популярный комплекс методик, разработанный центром CUMON. Его исследования показали, что самое эффективное средство для поддержания мозга в здоровом состоянии – решение нетрудных задач [13]: для предотвращения старения мозга необходимо решать примеры, тренировать память при помощи запоминания текстовой и цифровой информации, стараясь хранить ее в памяти в качестве образов, а также концентрироваться на какой–то определенной задаче.

По итогу этих исследований профессор и доктор медицины Рюта Кавашима предложил методику, содержащую простые и эффективные упражнения для тренировки пластичности мозга: вычисления примеров, запоминание различных слов и задачи на концентрацию и переключение внимания [13]. Одна из программных разработок, реализованная на основе этой методики, включает в себя три игры: устный счет (на простых примерах), запоминание слов и задача на соотнесение цвета и слова, обозначающего цвет.

Так как эта методика хорошо себя зарекомендовала, запомним этот комплекс упражнений и реализуем его идеи при разработке нашего приложения.

На основе приведенного выше сравнительного анализа реализуем в нашем мобильном приложении следующие категории: «Вычисления», «Память» и «Концентрация».

1.4 Формулирование требований к мобильному приложению для развития когнитивных способностей

При разработке мобильного приложения необходимо учитывать, что в дальнейшем оно будет использоваться по своему прямому назначению, а также,

возможно, будет дополнено рядом функций и новых возможностей касательно логики и интерфейса.

На основе исходных требований проектирования мобильного приложения, сначала были сформулированы основные функциональные и нефункциональные требования. Функциональные требования описывают то, как некая система должна работать. Нефункциональные требования являются менее строгими и описывают качество итоговой системы, выражают ее характеристики [14].

К функциональным требованиям относится:

1) возможность выбора развивающей игры в определенной категории;

2) возможность выбора определенного уровня и/или режима в развивающей игре;

3) возможность разблокирования определенных уровней и/или режимов в развивающей игре;

4) возможность отображения таблицы рекордов, заполненной по итогам прохождения развивающих игр, с максимальными результатами;

5) возможность отдельного отображения последних пяти попыток в каждой развивающей игре с датой и временем прохождения в виде таблицы и графика;

6) возможность заработать внутриигровую валюту при прохождении развивающих игр и при выполнении заданий;

7) возможность покупки определенных бонусов в игровом магазине за внутриигровую валюту;

8) возможность автоматического сохранения данных игрового процесса в локальную память телефона и в базу данных;

9) возможность автоматического чтения сохраненной информации из локальной памяти телефона и из базы данных для возврата в программу;

10) возможность прохождения автоматически сгенерированных испытаний из существующих развивающих игр: псевдослучайная генерация двух номеров испытаний для формирования из них одной игры;

11) возможность создания и прохождения собственных испытаний из существующих развивающих игр;

12) возможность выполнения ежедневных процедурно–генерируемых заданий и получения за них награды;

13) возможность получения награды за ежедневный вход;

14) возможность регулирования звука и изменения языка в самой игре при помощи меню «Настройки»;

К группе нефункциональных требований можно отнести следующие:

1) Требование к надежности: программа должна иметь защиту от некорректных действий пользователей – аварийный выход из приложения с дальнейшим восстановлением состояния. Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

2) Требования к составу и параметрам технических средств:

– Мобильное устройство с ОС Android 8.0 или выше;

– Размер ОЗУ от 2 ГБ – минимальный требуемый объем ОЗУ для текущего функционирования системы Android [7];

– Размер встроенной памяти от 8 ГБ – объем для установки и использования дополнительных приложений, включая будущие возможные обновления разрабатываемого приложения;

1.5 Выбор и обоснование средств разработки мобильного приложению для развития когнитивных способностей

Для разработки приложения для развития когнитивных способностей были выбраны следующие средства разработки:

1) Android Studio в качестве среды разработки;

2) Java в качестве основного языка программирования;

3) SQLite в качестве СУБД для хранения результатов прохождения развивающих игр пользователем.

Для разработки мобильного приложения наилучшим решением является интегрированная среда разработки Android Studio. Данное решение позволяет писать программное обеспечение для операционных систем на базе Android на языках программирования Kotlin и Java. К основным преимуществам среды относятся удобная визуализированная работа с пользовательским интерфейсом, статистический анализ кода, большая скорость сборки приложений и многое другое [1].

Одним из самых популярных языков для разработки приложений под мобильные устройства является Java. Android SDK включает в себя множество стандартных Java–библиотек (математические библиотеки, библиотеки структур данных, графические библиотеки и множество других). К основным преимуществам этого языка программирования относятся его безопасность и возможность объектно–ориентированной разработки [3].

Одной из достаточно легких и открытых систем администрирования баз данных является SQLite . Она занимает небольшое количество дискового пространства и отлично подходит для группировки и хранения небольших объемов структурированной информации на мобильном устройстве пользователя без создания и использования удаленного сервера [4].

2 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ РАЗВИТИЯ КОГНИТИВНЫХ СПОСОБНОСТЕЙ

2.1 Проектирование архитектуры мобильного приложения для развития когнитивных способностей

Архитектура – это организация системы, воплощенная в ее компонентах и отношениях между собой и с окружением. Она идентифицирует компоненты системы и способы их взаимодействия [15].

Так как наше мобильное приложение хранит все необходимые данные на самом мобильном устройстве (без взаимодействия с сетью Интернет), выберем модульную архитектуру. Данный вид архитектуры подразумевает разделение всех возможностей приложения на отдельные модули, каждый из которых отвечает за определенную часть функционала.

Общая схема программы приведена на рисунке 3.

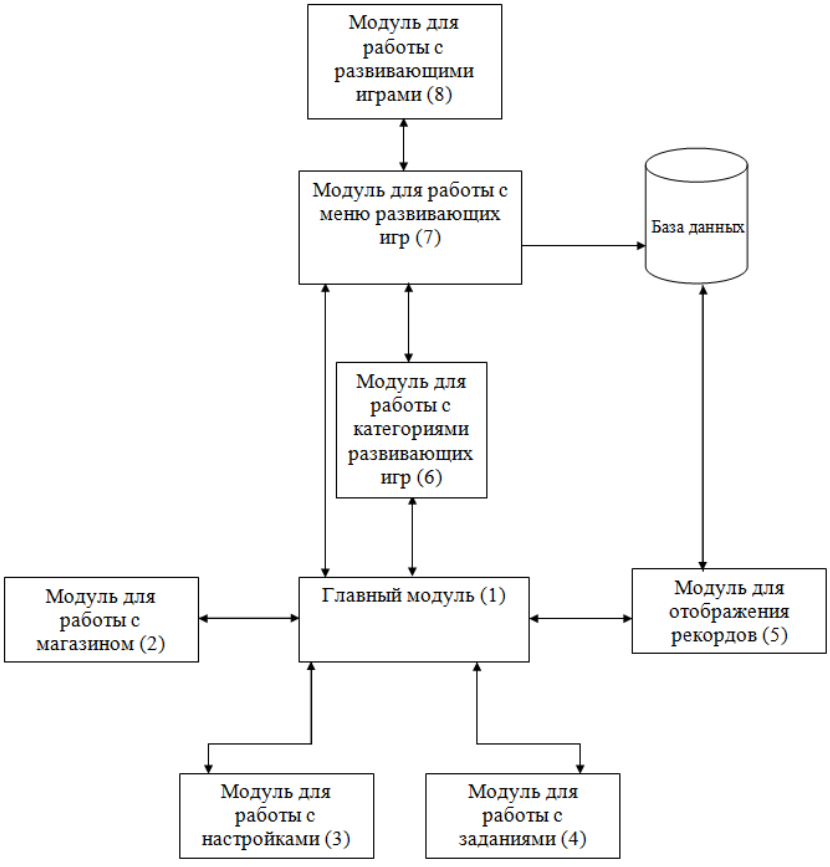

Рисунок 3 – Общая модульная схема приложения

Модуль 1 представляет собой точку входа в программу, которая включает в себя несколько функций по работе с ежедневной наградой, отображением информации, а также функции для загрузки и сохранения результатов игры.

Модуль 2 представляет собой совокупность функций и классов для взаимодействия с магазином внутри приложения: просмотр и покупка внутриигровых бонусов. Переход из модуля 1 осуществляется при выборе соответствующего режима работы.

Модуль 3 представляет собой совокупность функций и классов для взаимодействия с элементами экрана настроек внутри приложения: установка языка и включение/выключение звука. Переход из модуля 1 осуществляется при выборе соответствующего режима работы.

Модуль 4 представляет собой совокупность функций и классов для взаимодействия с элементами экрана ежедневных заданий внутри приложения: генерация заданий, изменение их состояния, получение наград за их выполнение. Переход из модуля 1 происходит после выбора соответствующего режима работы.

Модуль 5 представляет собой совокупность функций и классов для взаимодействия с элементами экрана результатов внутри приложения: получение информации о максимальном результате, а также о последних попытках пользователя в каждой развивающей игре. Информация о последних результатах берется из базы данных, хранящейся на мобильном устройстве пользователя, а информация о максимальном результате хранится отдельно в памяти устройства. Переход из модуля 1 происходит после выбора соответствующего режима работы.

Модуль 6 представляет собой совокупность функций и классов для взаимодействия со списком развивающих игр в категориях внутри приложения (кроме категории «Испытания»). Переход осуществляется при выборе определенной категории.

Модуль 7 представляет собой совокупность функций и классов для взаимодействия с элементами экрана меню развивающих игр: выбор параметров игры, открытие заблокированных режимов, смена текущего уровня. Также модуль содержит функции для записи результатов прохождения игр в базу данных и

обновления некоторых параметров состояния игры. Переход из модуля 6 осуществляется при выборе соответствующего режима работы. Также в модуле 1 автоматически может быть выбрана игра «Испытание» с последующим отображением ее меню в модуле 7.

Модуль 8 представляет собой совокупность функций и классов для взаимодействия с самими развивающими играми: возможность их прохождения, обработка результата. Модуль вернет результат прохождения игры. Для функционирования этого модуля в него необходимо передать совокупность выбранных данных (например, режим игры и уровень). Переход из модуля 7 осуществляется при установке всех нужных параметров.

База данных представляет собой файл, содержащий информацию обо всех возможных для прохождения развивающих играх, их режимах и результатах пользователя.

Для корректной работы функционала модулей 2, 3, 4, 6, 7 нужно передать в них «состояние приложения», то есть совокупность данных о состоянии настроек, имеющихся бонусах, состоянии внутриигрового счета. Это связано с тем, что приложение будет сохранять состояние всех параметров автоматически после каждого значимого действия пользователя: покупка бонуса, установка настроек, разблокирование режима, выполнение задания. Соответственно все названные выше параметры должны постоянно переходить между модулями для корректной работы программы.

Возврат в предыдущие модули преимущественно будет осуществляться при помощи выходов из соответствующих режимов. При возврате также необходимо контролировать передачу состояния приложения.

2.2 Разработка функциональной спецификации мобильного приложения для развития когнитивных способностей

Унифицированный язык моделирования (UML) – это семейство графических нотаций, в основе которого лежит единая метамодель. UML является

стандартным инструментом для создания «чертежей» программного обеспечения [8].

Проектируемая система на этой диаграмме представляется в виде актера (или множества актеров), взаимодействующего с разрабатываемой системой с помощью вариантов использования. Актером на схеме может являться любой субъект или объект, если он влияет на разрабатываемую систему извне. Вариант использования представляет собой спецификацию функций, с помощью которых осуществляется необходимое взаимодействие между актером и системой.

Актер в случае разработки нашего приложения будет один – пользователь. Он при использовании мобильного приложения для развития когнитивных способностей имеет следующие возможности:

1) Просмотр информации о своих рекордах. При этом пользователь может посмотреть на список своих максимальных результатов или последних достигнутых результатов в каждой выбранной игре.

2) Изменение текущих настроек: включить/выключить звуковое сопровождение при работе с приложением или изменить язык (русский, английский).

3) Получение ежедневной награды (за вход) или выполнение ежедневных процедурно–генерируемых заданий (выполнение определенных действий и получение награды).

4) Использование магазина. В магазине содержатся различные бонусы, ускоряющие процесс достижения прогресса в играх, и их стоимость.

5) Выбор развивающей игры. Так как все игры разбиты на категории и отображаются явно (кроме игры «Испытание», которая автоматически устанавливается при выборе категории «Испытания»), сначала необходимо выбрать ее.

6) Прохождение развивающей игры (выбранной из списка или созданной автоматически/вручную). У каждой игры есть свои собственные параметры (различные режимы, уровни или параметры создания своей игры типа «Испытание»), которые перед началом прохождения пользователю необходимо

выбрать и установить. Некоторые дополнительные игровые режимы и уровни можно открыть за внутриигровую валюту (при выполнении определенных действий, например, достижения определенного прогресса в игре).

Функциональная диаграмма для мобильного приложения для развития когнитивных способностей представлена на рисунке 4.

Рисунок 4 – Функциональная модель разрабатываемого приложения

2.3 Концептуальная модель мобильного приложения для развития когнитивных способностей

Диаграмма классов – это структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их кооперации, атрибутов (полей), методов и взаимосвязей между ними. Широко

применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования [8].

Было выделено 14 основных классов.

Для класса «Главная активность» были выделены поля, представляющие собой в совокупности общее состояние приложения: бонусы, настройки, максимальные результаты в каждой игре (рекорды), дата, необходимая для обработки состояния входа в приложение, массив ежедневных заданий, игровой прогресс (какие игры разблокированы, какой результат в процентах достигнут в каждой), количество звезд и количество очков, для представления внутриигровой валюты. Также этот класс имеет методы для загрузки и сохранения общего состояние (то есть всех перечисленных выше полей), обработки общего состояния и для взаимодействия с ежедневной наградой и заданиями.

Класс «Настройки» получает на вход общее состояние приложения и выделяет из него свойство «Настройки». Также у него есть методы, направленные на изменение состояния этих настроек и на сохранение общего состояния.

Классы «Задания» и класс «Магазин» получают на вход общее состояние приложения и выделяют из него свойства «Задания» и «Бонусы» соответственно. Также у них есть методы, направленные на изменение состояния этих данных и на сохранение общего состояния.

Так как заданий или товаров может быть достаточно много, для простоты их обработки используются классы «Поле_задание» и «Поле_товар», содержащие свойства, являющиеся структурными составляющими списка заданий или товаров. Также классы содержат методы для изменения состояния задания или бонусов соответственно.

Класс «Рекорды» получает на вход общее состояние приложения и выделяет из него свойство «Рекорды». Также у него есть метод, направленный на обработку рекордов, то есть на отображение результатов прохождения развивающих игр.

Для отображения последних результатов в каждой развивающей игре необходимо будет использовать базу данных. За ее создание отвечает класс «База

данных», который содержит поля для формирования первоначальных таблиц и методы для создания структуры данных.

Класс «Категории» получает на вход общее состояние приложения, а также хранит флаг категорий, необходимый для корректного отображения возможных развивающих игр, и флаг выбранной игры. Также у него есть методы, направленные на обработку выбранной категории и выбранной игры, а также метод, отвечающий за сохранение общего состояния.

Класс «Испытания» будет являться частью модуля для работы с меню развивающих игр и будет получать на вход общее состояние приложения. Он будет хранить различные флаги режима, которые влияют на то, как конкретно будет сформировано испытание, а также флаг уровня для выбора сложности испытания. Кроме того, присутствует метод, отвечающий за сохранение общего состояния.

Класс «Меню_игра» получает на вход общее состояние приложения, а также хранит флаги, необходимые для создания игрового процесса. Также у него есть метод, направленный на обработку общего состояния: изменение прогресса и отображение внутриигровой валюты. Кроме того присутствует метод, отвечающий за сохранение этого состояния. Для сохранения результатов прохождения развивающей игры ему необходима база данных.

Для упрощения отображения и обработки списка категорий, развивающих игр, а также параметров в меню игр, будем использовать класс «Поле_информация», содержащий метод для корректного отображения важных графических компонентов мобильного приложения.

Класс «Игра» получает на вход сформированное состояние игры, а также хранит элементы, необходимые для создания игрового процесса. Обработка этих элементов осуществляется соответствующим методом. Также присутствует метод для обработки завершения игры.

Каждая развивающая игра подчиняется какой–то логике, которая вынесена в отдельный класс «Игровая логика», содержащий методы и свойства, предназначенные для генерации заданий и обработки ответов.

Путем построения диаграммы классов мы выяснили, из каких основных классов будет состоять разрабатываемое мобильное приложение для развития когнитивных способностей. Также были описаны основные параметры и методы классов. Данная диаграмма строит четкую картину представления разрабатываемого программного обеспечения.

Основные классы системы мобильного приложения для развития когнитивных способностей представлены на рисунке 5. Для упрощения отображения информации все основные поля, относящиеся к общему состоянию приложения представлены только в классе «Главная активность». Все остальные классы содержат только те поля, которые являются значимыми для реализации их логики.

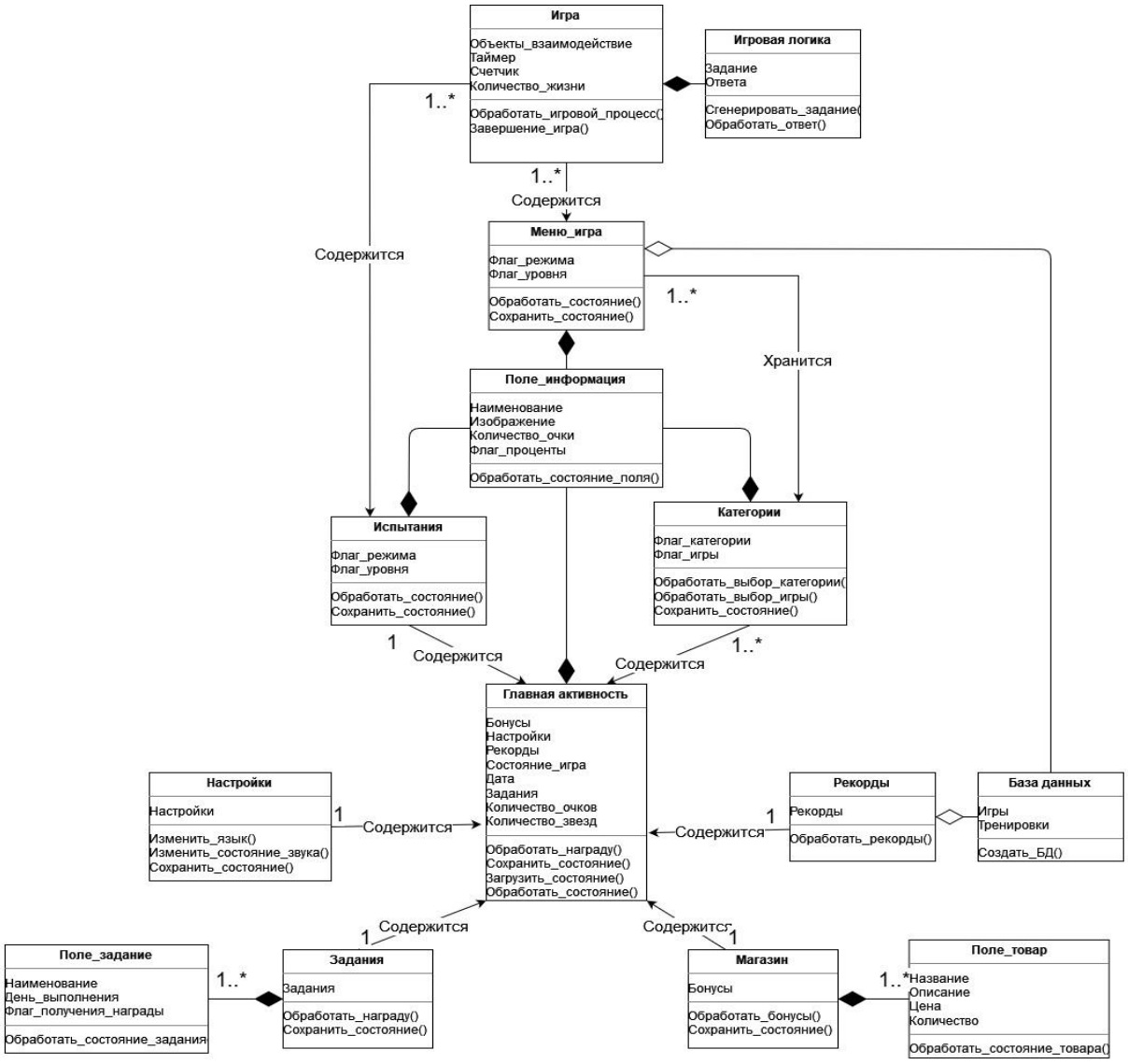

Рисунок 5 – Концептуальная модель разрабатываемого приложения

2.4 Поведенческая модель мобильного приложения для развития когнитивных способностей

Для отображения всех возможных состояний, в которых может находиться приложение, а также процесс смены состояний в результате внешнего влияния построим диаграмму переходов состояний [8]. При работе с нашим мобильным приложением можно выделить следующие состояния: ожидание действия в приложении, просмотр рекордов, просмотр последних прохождений, работа с магазином, работа с настройками, работа с заданиями, выбор игры, выбор параметров игры, выбор уровня, игровой процесс и обработка ответа на задание.

На рисунке 6 представлено поведение мобильного приложения для развития когнитивных способностей.

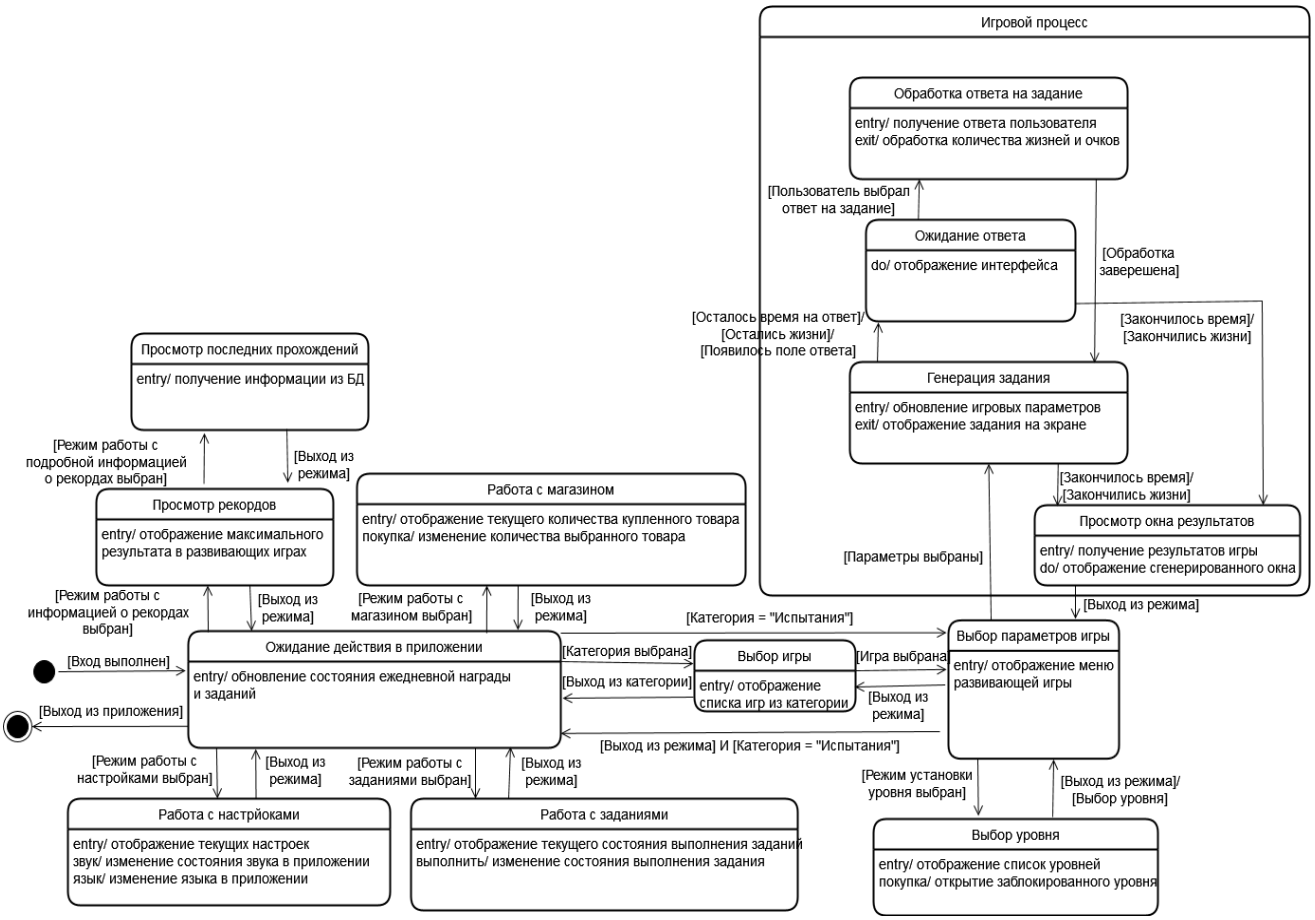

Рисунок 6 – Состояния разрабатываемого мобильного приложения

Из начального состояния выполняется переход в состояние ожидания действий в приложении. В этом состоянии происходит начальная инициализация параметров состояния и обновление значений.

При выборе режима просмотра информации о рекордах состояние изменяется на «Просмотр рекордов», где будут показаны все доступные развивающие игры с полученными в них максимальными результатами.

В каждой игре есть возможность изучить подробную информацию о последних прохождениях. При выборе этого режима состояние нашего приложения меняется на «Просмотр последних прохождений». Для отображения результатов сначала необходимо подключиться к базе данных.

При выборе пользователем режима работы с настройками состояние изменяется на «Работа с настройками», при этом сначала будут установлены последние выбранные пользователем настройки. При выборе языка основной язык приложения будет изменен. При взаимодействии со звуком будет включаться или выключаться звуковое сопровождение при прохождении.

При выборе пользователем режима работы с магазином состояние изменяется на «Работа с магазином», при этом сначала должен отобразиться весь список товаров и информация о его наличии у пользователя. При совершении покупки товара пользователем его количество изменится.

При выборе пользователем режима работы с заданиями состояние изменяется на «Работа с заданиями», при этом сначала должен отобразиться список заданий с текущим состоянием его выполнения. При попытке пользователя сдать выполненное задание, его состояние изменится.

Если же была выбрана какая–либо категория развивающих игр и эта категория не является испытаниями, состояние изменится на «Выбор игры» и будет отображен список игр в каждой категории. Если же выбрана категория «Испытания», автоматически загрузится меню тренировки («Испытание») с двумя вариантами ее создания и выбором уровня.

После выбора определенной игры состояние изменится на «Выбор параметров игры» и будут отображены различные режимы развивающих игр и текущий уровень, который пользователь может изменить.

Если, находясь в состоянии «Выбор параметров игры», выбрать режим смены уровня, состояние изменится на «Выбор уровня» и будет отображен список

уровней в развивающей игре. Некоторые уровни можно открыть за внутриигровую валюту.

При выборе нужного пользователю режима состояние изменится на «Игровой процесс». В нем с самого начала сгенерируется первое задание (состояние «Генерация задания») и обновятся количество очков, таймер и, в зависимости от выбранной игры, варианты ответов.

После этого, если у пользователя еще осталось время, жизни или, в зависимости от выбранной игры, появилось поле для ввода ответов, состояние изменится на «Ожидание ответа» в котором будет происходить обновление интерфейса до тех пор, пока пользователь не выберет ответ или у него не закончатся время или жизни.

Как только пользователь решит выбрать один из возможных ответов на задание, состояние изменится на «Обработка ответа на задание». Выбранный пользователем ответ должен быть получен и обработан (в соответствии с правильным ответом). При выходе из этого состояния количество жизней у игрока и количество очков, заработанных за игру, должно измениться. После окончания обработки состояние снова изменится на «Генерация задания» и снова будет генерироваться задание.

Как только у пользователя кончится жизни или время, состояние изменится на «Просмотр окна результатов». Пока пользователь не выйдет из этого режима, на экране будет отображаться сгенерированное окно, содержащее количество результатов и интересный факт. При подтверждении результатов состояние изменится на «Выбор параметров игры».

Основные возвратные механизмы между состояниями реализованы при помощи выходов из соответствующих режимов. Если пользователь закроет приложение состояние изменится на окончание работы приложения.

3 ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ РАЗВИТИЯ КОГНИТИВНЫХ СПОСОБНОСТЕЙ

3.1 Проектирование структуры мобильного приложения для развития когнитивных способностей

Так как в нашем приложении модульная архитектура, рассмотрим внутреннюю структуру каждого модуля в отдельности. Для этого будем использовать диаграмму классов, описанную на физическом уровне. Это необходимо, чтобы показать все важные промежуточные классы и их взаимодействие непосредственно внутри программы [8].

Опишем классы, которые необходимы для корректной работы главного модуля. Сам модуль состоит только из одного класса (MainActivity). В его свойствах перечисляются элементы, которые в общем случае можно описать составляющими состояния нашего приложения (массив для настроек, достижений, бонусов, переменных для внутриигровой валюты и т.д.).

В классе будут реализованы стандартные методы необходимые для работы: OnCreate(), OnActivityResult() и OnBackPressed().

Для заполнения списка элементов, который будет отображаться на экране, создадим метод SetInitialData().

Для сохранения информации непосредственно в память мобильного устройства и загрузки информации из нее реализуем два метода: SaveArrayList() и LoadArrayList(). Для непосредственной работы с полученной из памяти информацией реализуем метод LoadInfoForGame().

Так как на нашей главной активности находятся элементы, для работы с ежедневными заданиями и наградами, реализуем метод GetDurationBreakdown(), для проверки разницы между двумя датами, а также метод GetEveryDayBonus() для обработки состояния ежедневной награды.

Для передачи состояния нашего приложения в другие модули реализуем метод SendData().

Наш модуль взаимодействует с несколькими классами, не относящимися к какому–то конкретному модулю.

Прежде всего это класс Save. Он содержит свойства, представляющие собой в основном состояние приложения, а также методы для преобразования информации в корректную для сохранения (или загрузки) форму.

Класс Sound отвечает за работу со звуком. Он содержит необходимые для корректной загрузки и смены и воспроизведения звука набор свойств и методов.

Класс StateAdapter представляет собой простейший адаптер, который связывает массив данных с набором графических элементов. Этот класс содержит метод для корректного отображения информации.

Класс State содержит данные, которые и будут отображаться на нашей главной активности. Он содержит методы только для работы со свойствами (сеттеры и геттеры). Массив элементов этого класса заполняется в нашем главном модуле.

Диаграмма классов главного модуля приведена на рисунке 7.

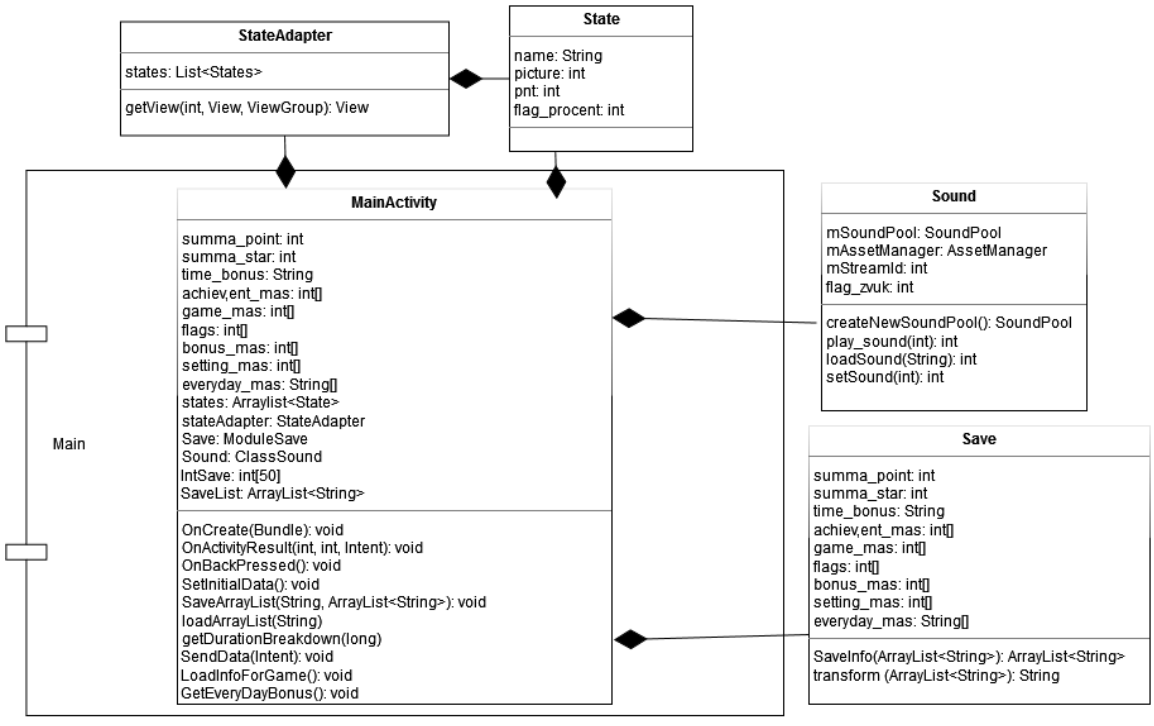

Рисунок 7 – Диаграмма классов главного модуля

Далее разберем модуль для работы с настройками. Модуль также состоит из одного класса, содержащего определенные свойства (состояние игры и переменные для работы с настройками), а также методы, где кроме стандартных будут реализованы HandleButton() для более удобного сохранения данных и воспроизведения звука и LoadInfoForSettings() для преобразования состояния в вид, удобный для работы.

Этот модуль также взаимодействует с классом Save (после смены языка или установки звука происходит автоматическое сохранение), а также с классом Sound, так как смена языка и включение/выключение звука реализованы в виде кнопок.

Диаграмма классов модуля для работы с настройками приведена на рисунке 8.

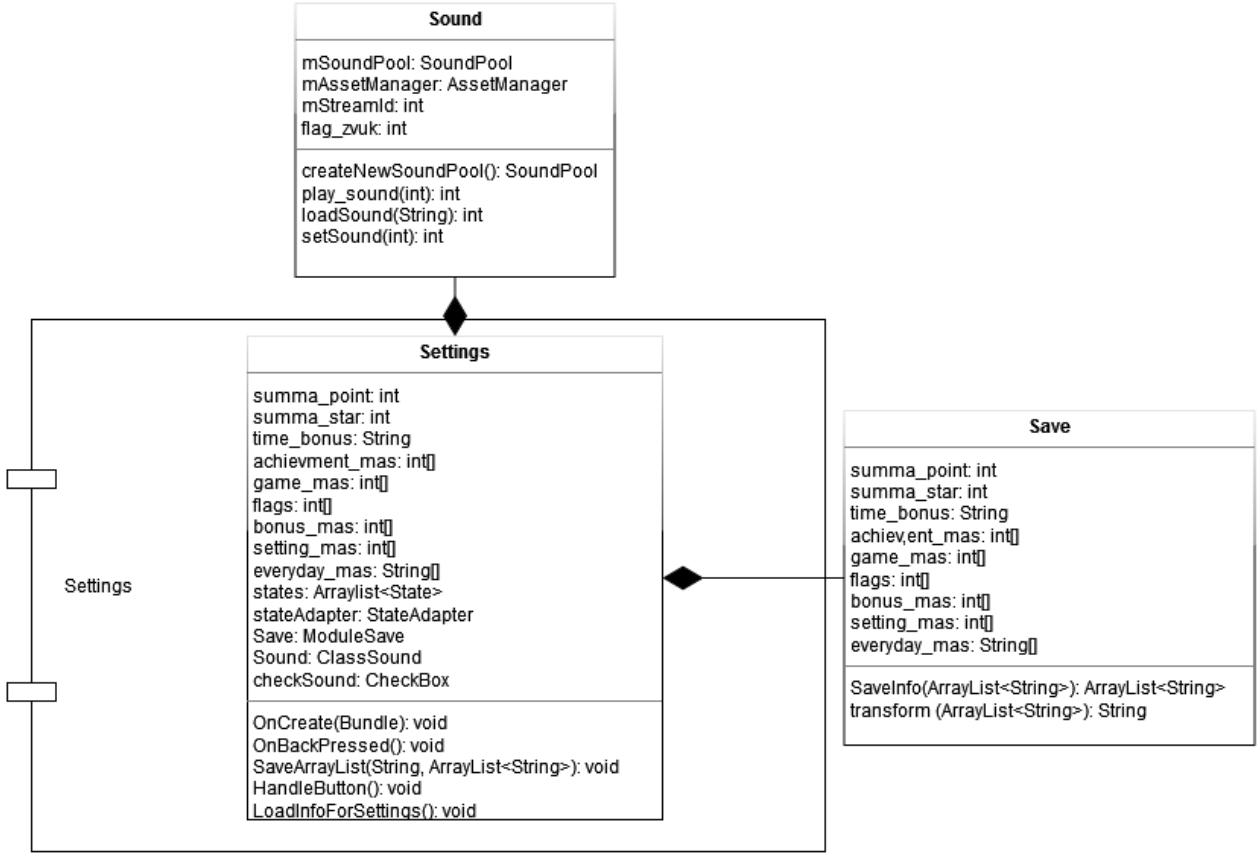

Рисунок 8 – Диаграмма классов модуля для работы с настройками

Далее опишем модуль для отображения рекордов. Он состоит из двух классов: AchievmentActivity для отображения максимального результата в

каждой игре и ShowActivity для отображения последних пяти попыток, а также графика результата в выбранной игре.

Класс AchievmentActivity содержит информацию о состоянии настроек и достижений. Инициализация содержания окна осуществляется при помощи метода InitRecords(). Рядом с каждым инициализированным режимом находится кнопка при нажатии на которую запускается активность ShowActivity (и проигрывается звук, если установлены соответствующие настройки). Для проигрывания звуков также необходим класс Sound.

Класс ShowActivity содержит методы для загрузки переданной информации (LoadInfoForShow()), инициализации запроса из базы данных (InitQuery()) и отображения графика (DrawGraph()). Среди его свойств есть переменные для работы с состоянием приложения и отображением графика и массив для записи результатов конкретной игры. Для работы этому классу необходима база данных (объект ее класса), инициализация которой происходит в классе DbHelper. Для его работы необходимы массивы, содержащие наименования всех игр и всех необходимых для отображения режимов (режимы «Тренировка» по уровням и другие необходимые режимы).

Диаграмма классов модуля для отображения рекордов приведена на рисунке 9.

Далее опишем модуль для работы с магазином. Он состоит из трех классов, взаимодействующих друг с другом по принципу паттерна «Адаптер».

Класс ShopClass содержит информацию о большинстве параметров состояния приложения, а также методы, предназначенные для инициализации товаров (SetInitProducts()), загрузки и сохранения инфорции (LoadInfoShop(), SaveArrayList()), а также отображения внутриигровой валюты после того, как пользователь купил товар (ShowResult_point(), ShowResult_star()).

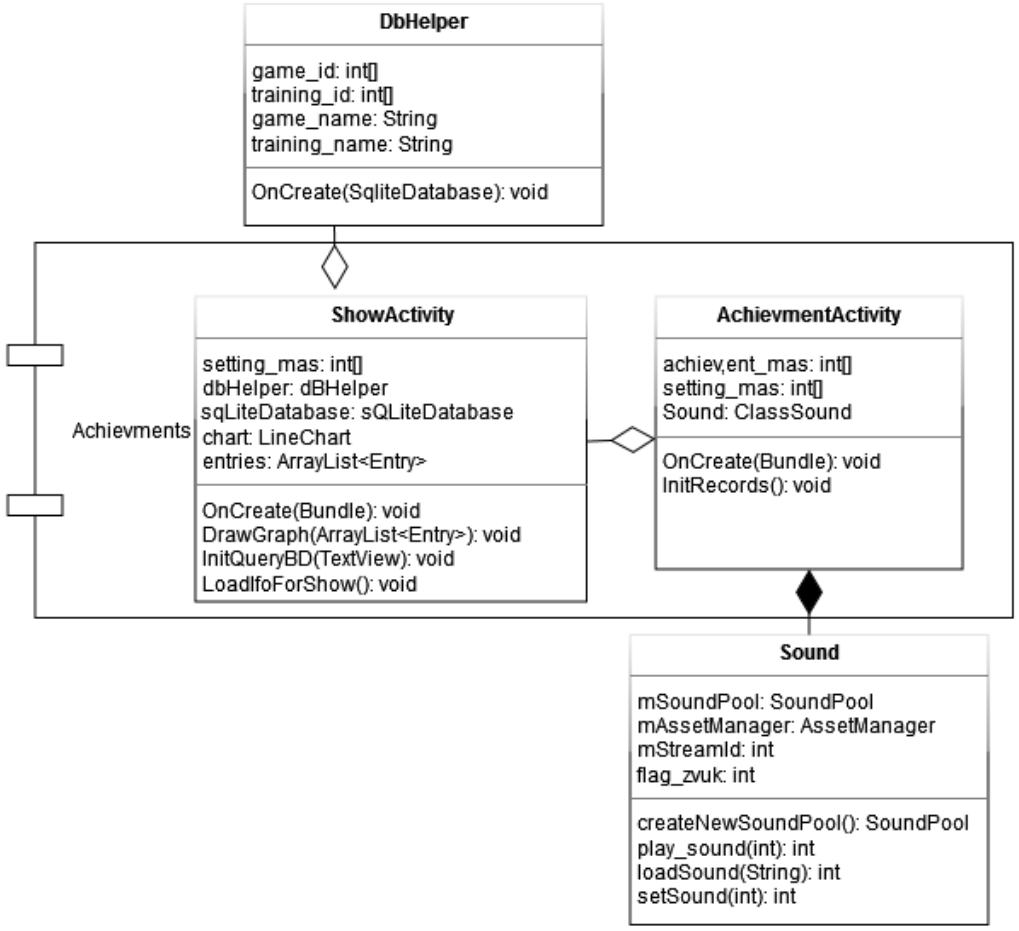

Рисунок 9 – Диаграмма классов модуля для отображения рекордов

Класс ProductAdapter представляет собой простейший адаптер, который связывает массив данных с набором графических элементов. Этот класс содержит метод для корректного отображения информации, а также методы, необходимые для реализации логики покупки (BuyForPoint(), BuyForStar()): как только пользователь нажмет на кнопку, являющуюся частью списочного элемента товаров, должна запуститься проверка на соответствие цены товара и имеющейся у пользователя валюты.

Класс Product содержит атрибуты, описывающие продукт (наименование, цена, количество и т.д.). Среди его методов только сеттеры и геттеры для получения доступа к информации.

Так как при нажатии на кнопку должен воспроизводиться звук (если выбрана соответствующая опция), объект класса Sound будет являться одним из свойств класса адаптера.

После покупки и отображения результата (изменение количества товара и внутриигровой валюты) информацию об игре необходимо сохранить (класс Save).

Диаграмма классов модуля для работы с магазином приведена на рисунке 10.

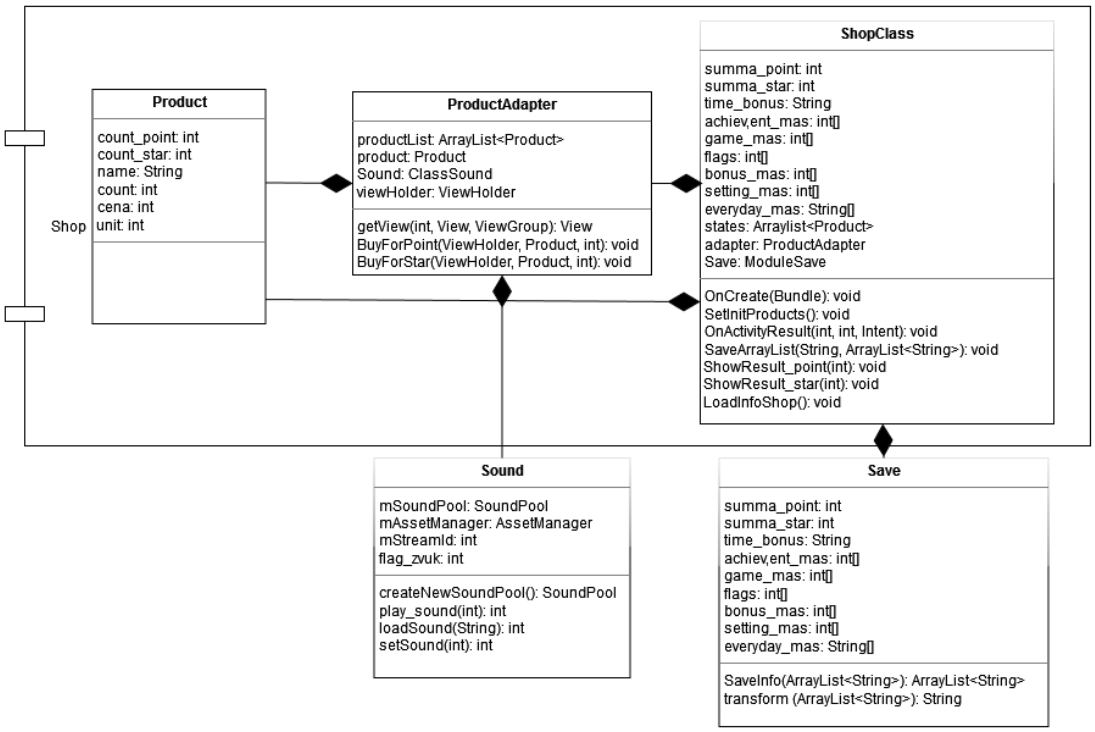

Рисунок 10 – Диаграмма классов модуля для работы с магазином

Далее опишем модуль для работы с ежедневными заданиями. Он также состоит из трех классов, взаимодействующих друг с другом по принципу паттерна «Адаптер».

Класс EveryDayTask содержит информацию о большинстве параметров состояния приложения, а также методы, предназначенные для инициализации товаров (SetInitTasks()), загрузки и сохранения инфорции (LoadInfoTasks(), SaveArrayList()), а также создания заданий (CreateTasks()) и передачи состояния (SendData()).

Класс TaskAdapter представляет собой простейший адаптер, который связывает массив данных с набором графических элементов. Этот класс содержит метод для корректного отображения информации, а также методы, необходимые для возможности выполнения задания (ChangeDataTasks(),

GetDurationBreakdown()): как только пользователь нажмет на кнопку, являющуюся частью списочного элемента заданий, должны запуститься методы проверки выполнения задания и возможности выдачи награды.

Класс Tasks содержит атрибуты, описывающие ежедневное задание (наименование, флаг выполнения, дата выполнения). Среди его методов только сеттеры и геттеры для получения доступа к информации.

Диаграмма классов модуля для работы с заданиями приведена на рисунке 11.

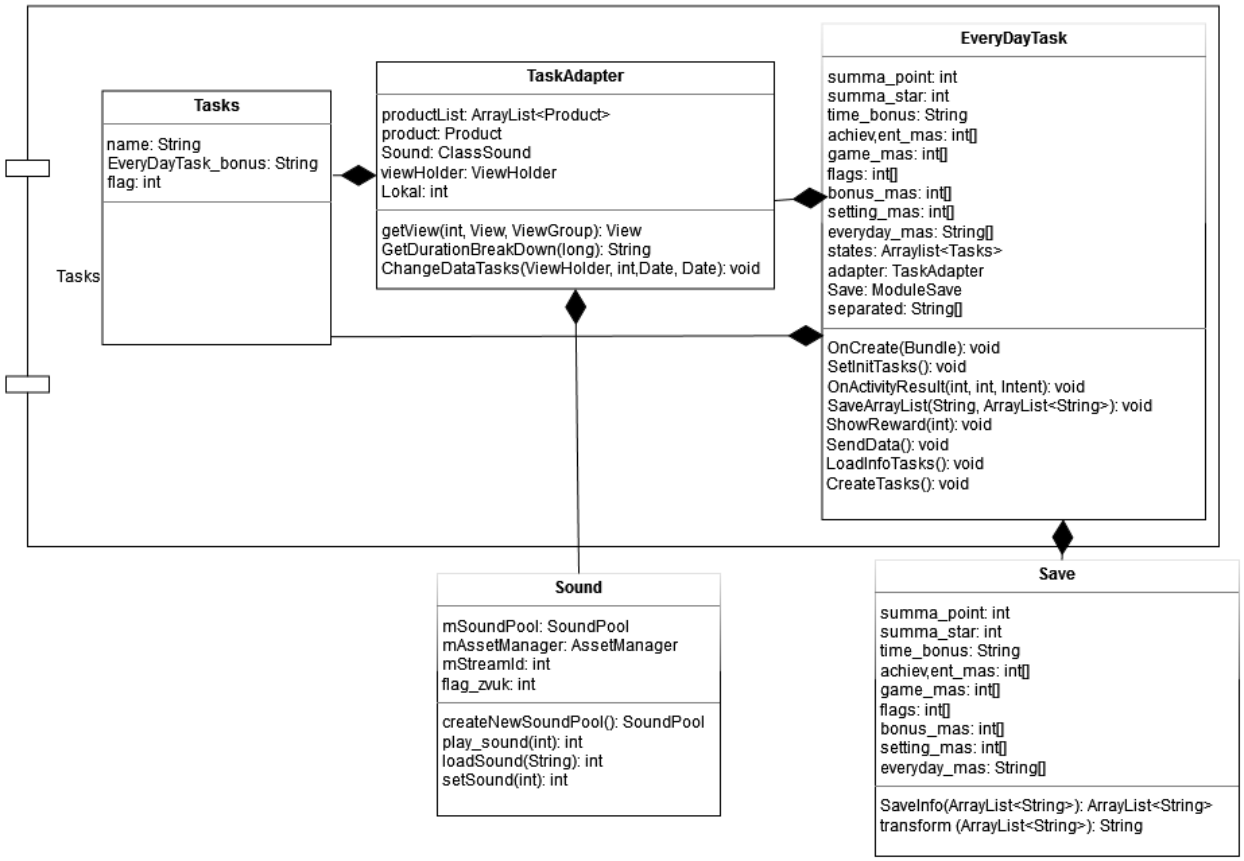

Рисунок 11 – Диаграмма классов модуля для работы с заданиями

Далее рассмотрим модуль для работы с категориями развивающих игр. Сам модуль представлен одним классом: CountList для работы с категориями «Вычисления», «Память», «Концентрация».

Класс CountList среди свойств содержит различные массивы и переменные, составляющие состояние игры, а также отдельную переменную–флаг для выбора списка развивающих игр в зависимости от выбранной категории.

Этот класс содержит методы для загрузки, сохранения и отправки информации, а также метод SetGameParametrs() для инициализации развивающих игр. Выбор любой развивающей игры из сформированного списка приведет к переходу в модуль меню развивающей игры.

При переходу в этот модуль, а также возврат в него происходит автоматическое сохранение (класс Save). При нажатии на любую развивающую игру, издается или не издается звук (класс Sound).

Так как развивающие игры представлены списком, подключим также реализованный нами ранее адаптер StateAdapter для обработки выбора развивающих игр, а также класс State для их хранения.

Диаграмма классов модуля для работы с категориями развивающих игр приведена на рисунке 12.

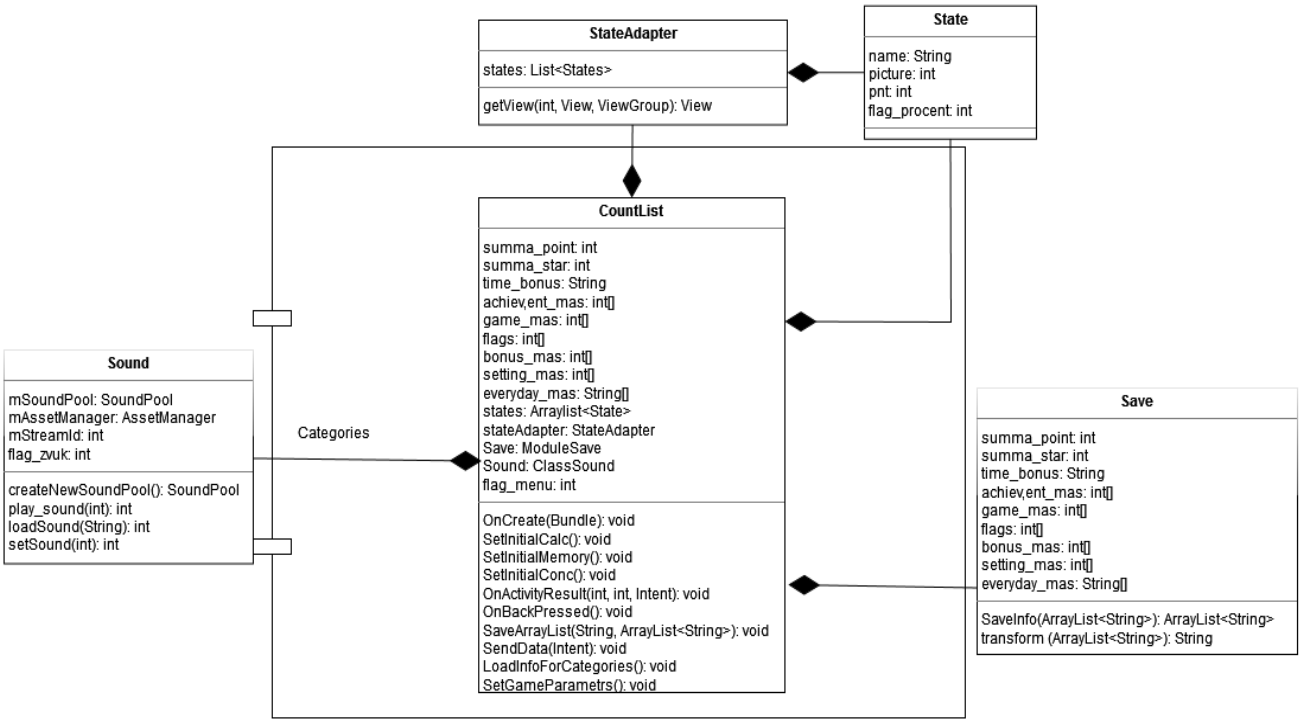

Рисунок 12 – Диаграмма классов модуля для работы с категориями развивающих игр

Далее рассмотрим модуль для работы с меню развивающих игр. Сам модуль содержит 4 класса, необходимых для реализации меню развивающих игр в трех категориях, а также меню игры «Испытание».

Классы меню развивающих игр содержат свойства, необходимые для инициализации всех важных компонентов меню (состояние игры и, возможно, дополнительные флаги, счетчики и диалоговые окна).

При запуске каждого меню информация сначала загружается, затем на основе полученной информации инициализируются ключевые флаги (звук, внутриигровая валюта и т.д.), а затем на основе полученной информации инициируется содержание меню.

Каждое меню реализовано в виде списка, каждый элемент которого содержит картинку, текст, различные счетчики. Реализуем это с помощью уже спроектированных нами классов StateAdapter для отображения и State для хранения информации.

Так как каждое меню содержит какую–либо развивающую игру, при завершении прохождения результат этой игры вернется в меню. Для сохранения этого результата используется класс DbHelper, а также методы WorkDatabase...().

Класс OperationGameLevel() предназначен для реализации функционала смены уровня. Он содержит, помимо стандартных методов, функции для обработки выбора уровня, а также их инициализации в зависимости от меню.

Для воспроизведения звука и сохранения информации о прохождении в память мобильного устройства используются уже спроектированные нами классы Sound и Save.

Диаграмма классов модуля для работы с развивающими играми приведена на рисунке 13. Для простоты отображения различные методы типа «геттеры»/»сеттеры» были опущены. Также классы для работы с адаптером, звуком и сохранением ведут к целому модулю, а не к отдельному классу. Это сделано для меньшего загромождения схемы, так как объекты названных выше классов реализованы в каждом классе меню внутри модуля.

Далее рассмотрим модуль для работы с самими развивающими играми. Так как кроме самих развивающих игр, часть из которых будет

иметь схожую модель функционирования и отображения, мы также делаем игры–испытания, представляющие собой объединение наших игр, рассмотрим этот модуль как три отдельные смысловые составляющие.

Сначала рассмотрим игры из категорий «Вычисления» и «Концентрация». Рассматривая логику их работы, можно сказать, что, в общем случае, чтобы проходить эти игры, нужно выбирать ответ на каждое генерирующееся задание до тех пор, пока не закончится время или пока не закончатся жизни.

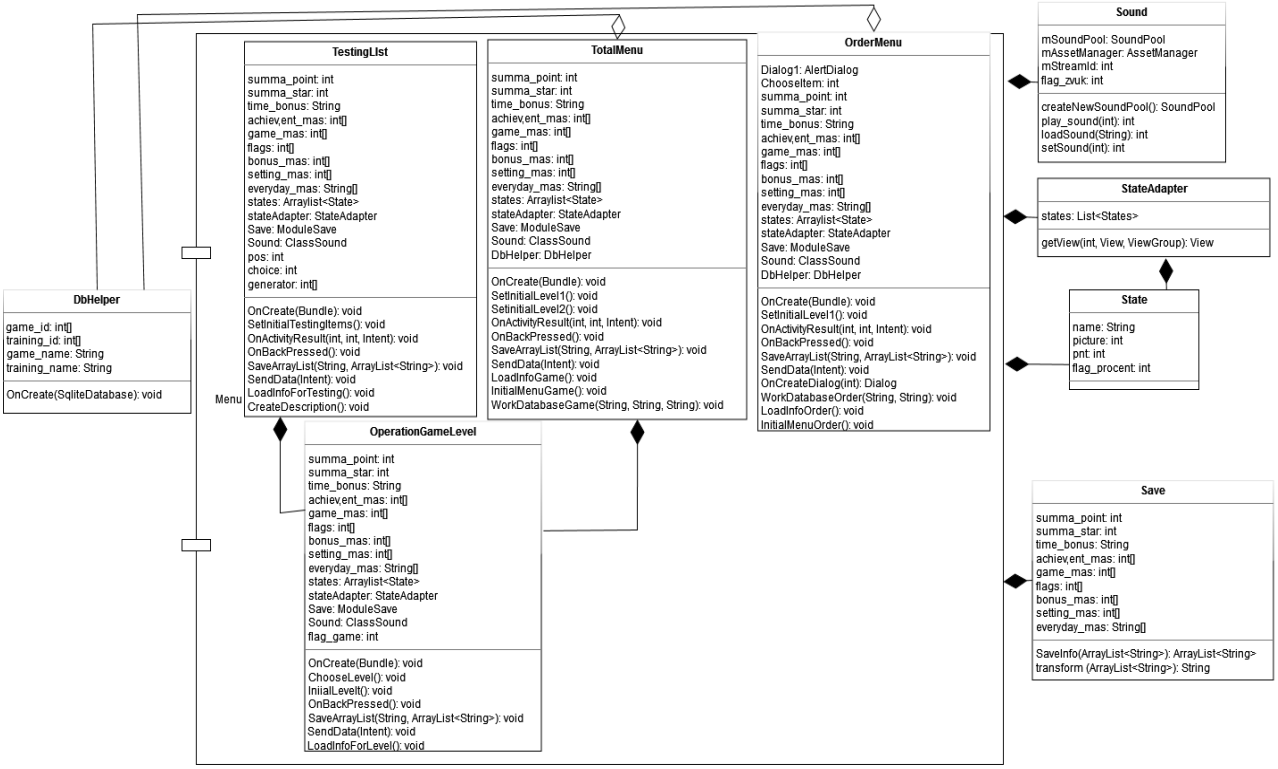

Рисунок 13 – Диаграмма классов модуля для работы с меню развивающих игр

Это позволяет нам вынести общий интерфейс у классов логики работы развивающих игр и использовать этот интерфейс в классе отображения, инициализируя его в зависимости от конкретно выбранной игры. Диаграмма классов этой части нашего модуля для работы с развивающими играми приведена на рисунке 14.

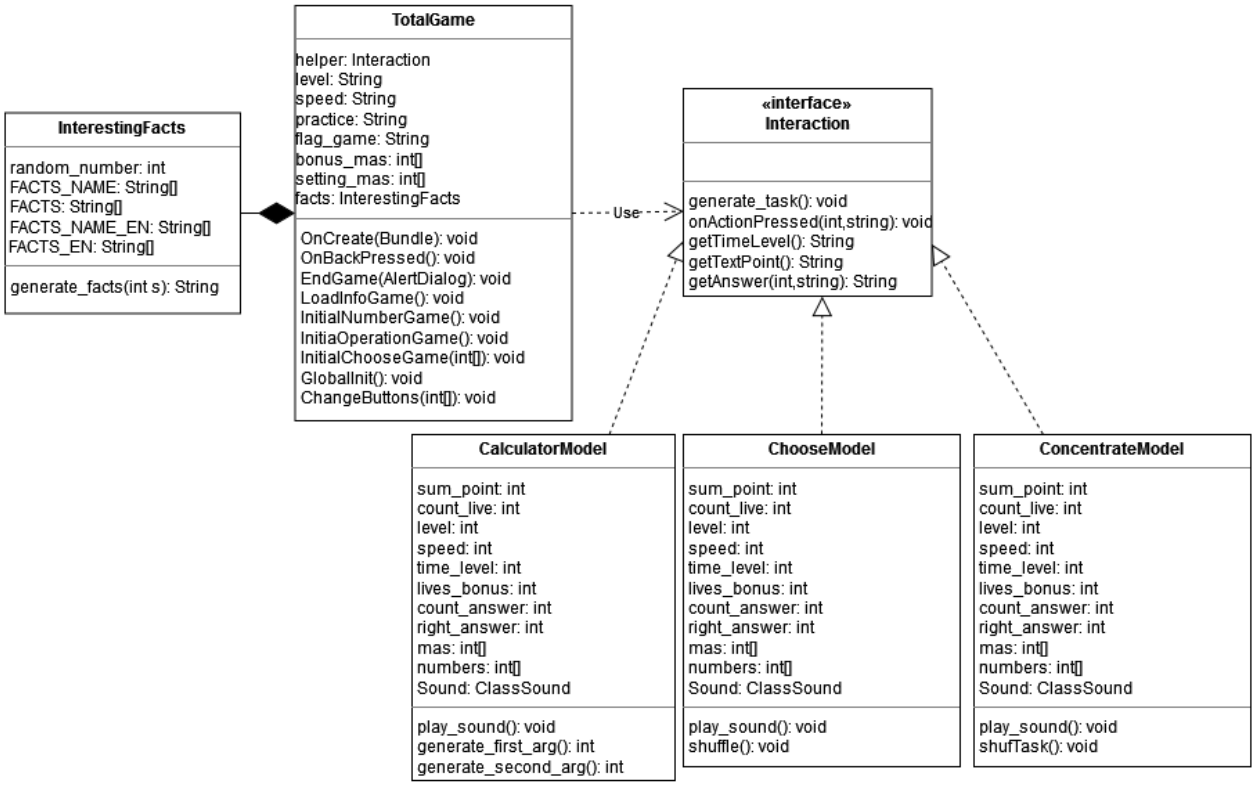

Рисунок 14 – Диаграмма классов для игр из категорий «Вычисления» и «Концентрация»

При определении необходимого нам класса логики через интерфейс, игра должна загрузить необходимые данные (режим, уровень и т.п.) и проинициализировать начальные значения и поля взаимодействия.

Классы CalculatorModel, ChooseModel, ConcenctrateModel представляю собой логику развивающих игры «Вычисления», «Найди число» и «Пропуск» соответственно.

Класс TotalGame является классом представления, отображая логику работы игр и позволяя взаимодействовать с ней пользователю. Основной обработчик игрового процесса заложен в методе OnCreate(). В нем создается объект интерфейса и при помощи соответствующих методов изменяется игровое поведение.

Классы игровых моделей, кроме методов интерфейса, содержат свои свойства и методы для генерации задания, проигрывания звука и обработки информации.

Интерфейс содержит методы, которые необходимо будет реализовать: для генерации заданий, обработки нажатия на кнопки, а также методы, связанные с передачей и обработкой количества очков, времени и ответов.

Класс InterestingFacts реализует генерацию случайного интересного факта о мозге человека. Его метод generate_facts() вызывается при отображении окна завершения прохождения развивающей игры (EndGame()).

Далее рассмотрим часть модуля, содержащую игру из категории «Память». Логика ее работы отличается от остальных игр: ответ необходимо вводить после каждого окончания таймера, а у игрока всего одна жизнь.

Так как мы разрабатываем небольшое игровое приложение, отделим логику от интерфейса при помощи реализации паттерна MVP (Model–View–Presenter).

Каждое представление должно реализовывать соответствующий интерфейс. Интерфейс представления определяет набор функций и событий, необходимых для взаимодействия с пользователем. Presenter должен иметь ссылку на реализацию соответствующей логики, которую обычно передают в конструкторе.

Логика представления должна иметь ссылку на экземпляр Presenter. Все события представления передаются для обработки в Presenter и практически никогда не обрабатываются логикой представления.

Диаграмма классов этой части модуля приведена на рисунке 15.

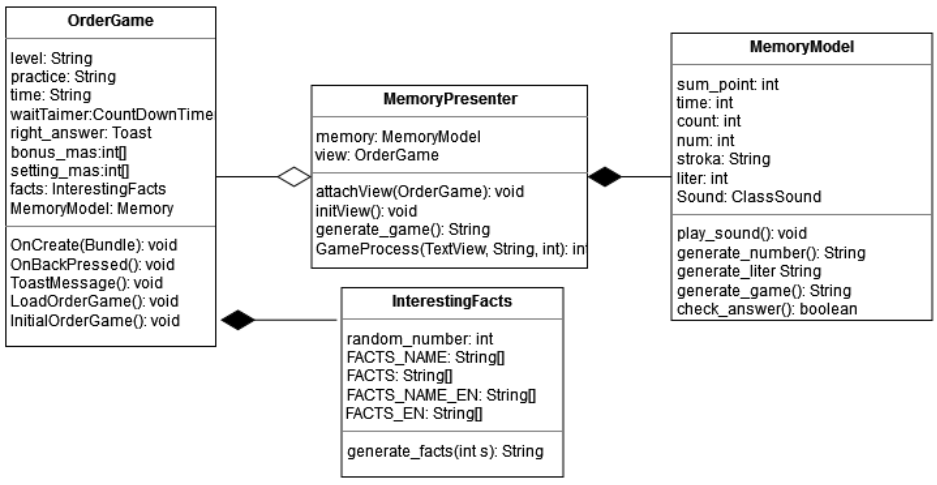

Рисунок 15 – Диаграмма классов для игры из категории «Память»

В завершении рассмотрим часть модуля, в которой содержатся игры из категории «Испытания». Так как каждая такая игра содержит в себе две игры из других категорий, спрячем логику их обработки в отдельном классе.

Для этой цели подойдет паттерн проектирования Фасад, который объединит два интерфейса двух необходимых для конкретного испытания игр, приводя все вызовы к одному объекту.

Диаграмма классов этой части модуля приведена на рисунке 16.

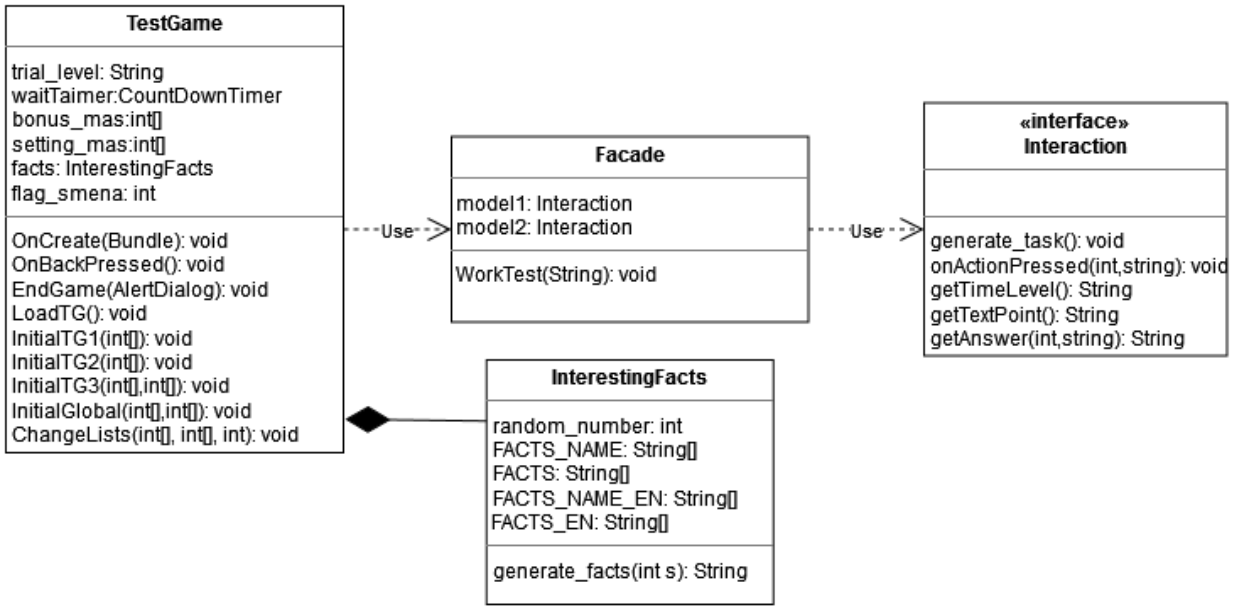

Рисунок 16 – Диаграмма классов для игр из категории «Испытания»

Классы TestGame отвечает за отображение игры типа «Испытание». Он также содержат методы для начальной загрузки и инициализации информации, а также содержит свои методы и свойства для смены заданий в развивающих играх. Так как подобные игры основаны на взаимодействии двух развивающих игр из других категорий, для реализации необходимо связать наше представление с классом фасада, в котором инициализируются через соответствующий интерфейс два класса логики.

Также при взаимодействии с игровыми элементами, в зависимости от состояния настроек будет издаваться звук или изменяться язык (класс Sound).

Для простоты отображения различные методы типа «геттеры»/»сеттеры» во всех диаграммах этого модуля были опущены. Также был опущен внешний класс Sound, взаимодействующий с классами модели.

3.2 Проектирование алгоритмов мобильного приложения для развития когнитивных способностей

Первым алгоритмом, который будет представлен, является общий алгоритм, по которому работают большинство развивающих игр (кроме игр из категории «Память»).

Прежде, чем запустить непосредственно игровой процесс, развивающая игра должна обработать переданные для дальнейшего использования значения: информацию о режиме игры, выбранном уровне, имеющихся бонусах и настройках.

После этого модель нашей игры на основе полученных значений должна сгенерировать стартовое задание. Каждая развивающая игра использует различные способы генерации. Например, случайная обработка массива при помощи перемешивания его элементов или генерация нескольких случайных величин и создание на их основе строки.

Так как среди доступных пользователю режимов есть режим «Практика», необходимо проверить, этот ли режим выбран пользователем. Если да, то в игре необходимо убрать таймер и скрыть счетчик жизней, чтобы пользователь мог разбираться с заданием столько, сколько ему необходимо и при этом не боялся ошибиться.

В этом режиме происходит обработка нажатой клавиши, которая может включать в себя обновление очков (в зависимости от игры). После этого обновляется поле с заданием (в некоторых играх оно обновляется на предыдущее задание в зависимости от обработанной клавиши). После этого в режиме всплывающего сообщения или, например, в выделении нужной кнопки другим цветом, показывается правильный ответ. Выход из этого режима осуществляется кнопкой «Назад», присутствующей на мобильном устройстве Android.

Если же пользователь играет в любом другом режиме, то запускается таймер обратного отсчета со временем, за которое пользователь должен решить как можно больше заданий.

При решении задания алгоритм анализирует, с каким игровым элементом взаимодействует пользователь, на основании этого обновляет счетчик полученных очков. При этом, если пользователь выбрал режим «Скорость», то очки будут обновляться на предыдущее значение, пока пользователь не достигнет трех верных ответов подряд. Но в любом случае, после трех ответов в этом режиме количество правильных ответов будет сбрасываться и будет происходить обновление таймера (так как на три ответа пользователю каждый раз будет даваться три–четыре секунды).

На основании выбранного пользователем ответа на задание в развивающей игре, будет обновляться количество жизней: уменьшаться или изменяться на то же самое. После этого, если у пользователя еще есть жизни, игровой процесс продолжится.

Как только закончится время пользователя на прохождение или он допустит столько ошибок, что количество его жизней достигнет нуля, выведется окно с результатами его игры. Окно результата подразумевает собой информацию о полученном количестве очков, а также случайно выбранный из списка интересный факт о мозге человека.

Схема описанного выше алгоритма представлена на рисунке 17.

Вторым алгоритмом, который мы рассмотрим, будет алгоритм генерации ежедневных заданий. Он должен каждый день формировать список из двух различных заданий. Разнообразие в этих заданиях один из способов поддержания интереса пользователя к частому использованию приложения.

Так как все значения для нашей генерации будут браться из массива, перед началом работы алгоритма необходимо загрузить нужные данные.

Если в приложении наступил следующий день (или прошло больше одного дня), сбрасываются значения на основе которых составляется список заданий.

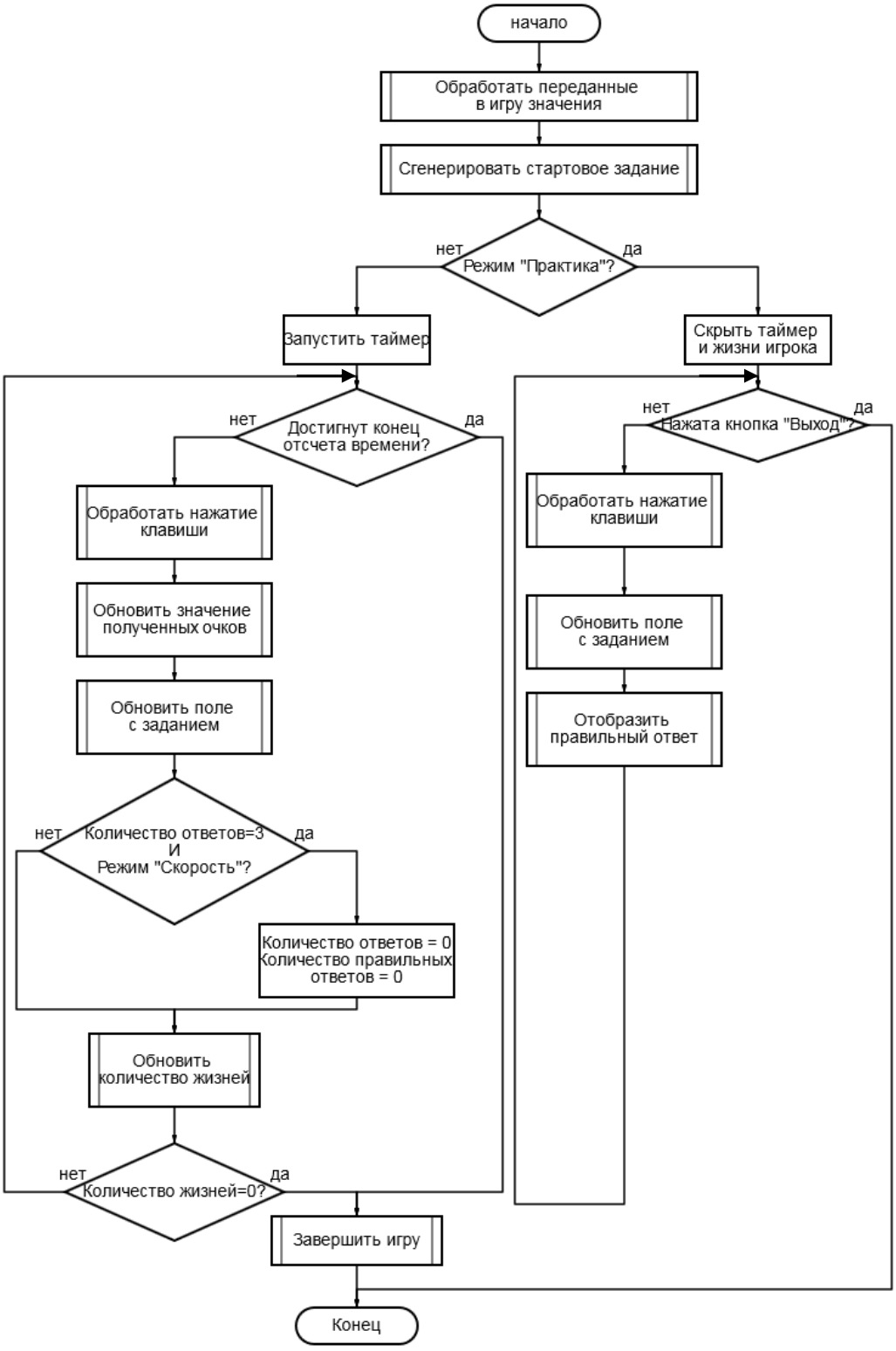

Рисунок 17 – Общий алгоритм большинства развивающих игр

Затем идет проверка на необходимость генерации флага заданий. Этот флаг необходим для дальнейшего формирования внутреннего наполнения списка всех заданий.

Затем последовательно идут проверки на необходимость генерации первого и второго флага поиска. Оба флага поиска нужны для генерации уже непосредственно самих заданий из списка.

Чтобы не было двух одинаковых заданий, после генерации второго флага запускается цикл проверки на совпадение первого и второго флага.

После генерации всех нужных параметров происходит генерация массива заданий. Таким образом, реализована двойная процедурная генерация: сначала формируются сами задания и из них собирается список, а затем, на основе двух сгенерированных флагов поиска из этого списка выбираются два задания.

Все сгенерированные для работы параметры сохраняются, чтобы при повторном запуске не генерировать новые задания, а продолжать работу со старыми.

В конце сформированный массив заданий для пользователя отображается. В своей работе процедура отображения также использует отдельный массив, на основе которого делается выбор: отображать ли само задание или информацию о его выполнении, а также может ли пользователь получить награду за задание в данный момент.

Схема описанного выше алгоритма представлена на рисунке 18.

Рассмотрим также алгоритм создания собственных заданий. Если пользователь выберет нужный пункт меню из окна «Испытания», откроется диалоговое окно, в котором необходимо выбрать две развивающие игры из предоставленного списка.

При каждом выборе развивающей игры (при добавлении или удалении игры из списка) информация об этом действии пользователя будет сохраняться в отдельный массив.

После того, как пользователь выберет необходимые ему развивающие игры, ему необходимо нажать на кнопку «ОК» для формирования испытания. При этом на основе сделанного выбора будут сформированы данные, на основе которых будет в дальнейшем формироваться испытание.

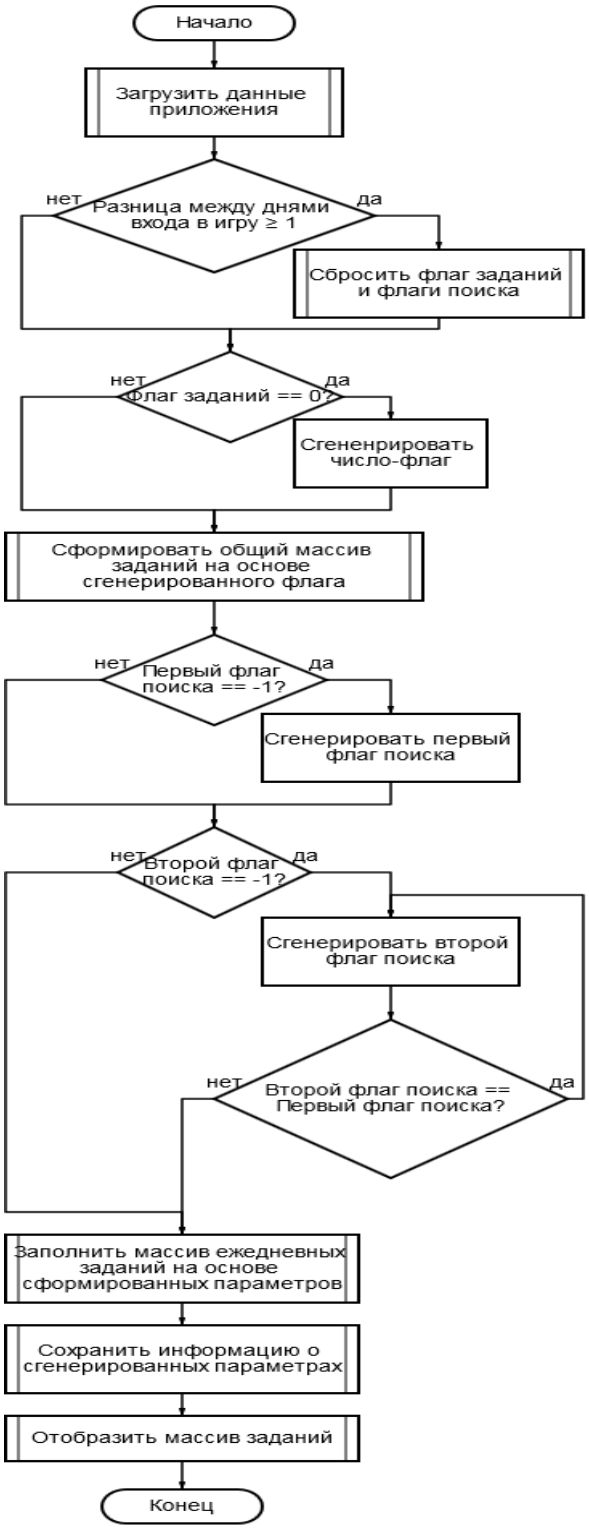

Рисунок 18 –Алгоритм генерации ежедневных заданий

Кроме того, будет посчитана контрольная сумма, на основе которой будет сделан вывод, нужное ли количество развивающих игр выбрал пользователей. Если было выбрано две игры, то на основе сформированных флагов будет сгенерировано сообщение, на основе которого будет

сформировано испытание. Иначе выведется сообщение об ошибке формирования испытания, а алгоритм завершит работу.

Схема описанного выше алгоритма представлена на рисунке 19.

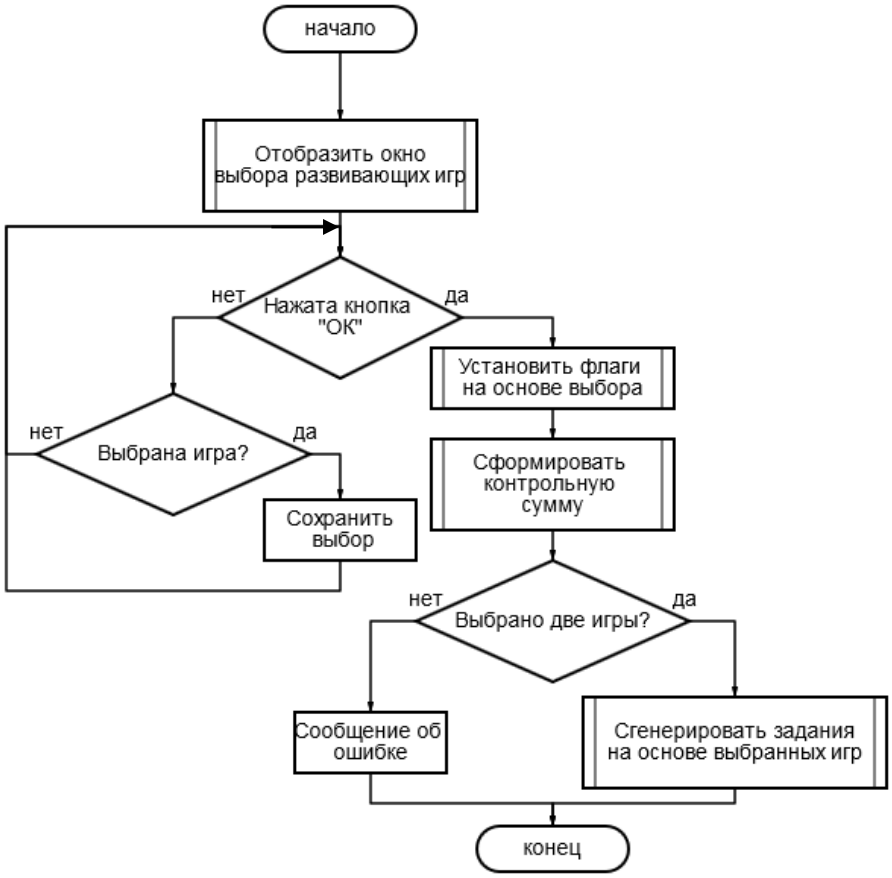

Рисунок 19 –Алгоритм самостоятельного формирования тренировки типа «Испытание»

3.3 Проектирование пользовательского интерфейса мобильного приложения для развития когнитивных способностей

Пользовательский интерфейс – набор аппаратных и программных средств, обеспечивающих взаимодействие пользователя с устройством. Это способ общения между программой и пользователем или внешний вид продукта. Его основная цель – сделать это взаимодействие максимально комфортным, то есть обеспечить полное ознакомление с возможностями программы при минимальных усилиях со стороны пользователей [9].

Проектирование интерфейса для мобильного приложения имеет особенность, которую необходимо учитывать при разработке: размер мобильного устройства, по сравнению с персональными компьютерами, достаточно небольшой, поэтому расположение на нем большого количества форм взаимодействия может вызвать некоторые трудности. Поэтому очень важно обеспечить достаточный, но не перегруженный интерфейс.

На этапе проектирования пользовательского интерфейса необходимо определить окна, наличие которых обязательно в разрабатываемом приложении, а также основные задачи, выполняемые ими. В рамках нашего курсового проекта необходимо реализовать следующие окна: «Главное меню», «Настройки», «Магазин», «Рекорды», «График рекордов», «Испытания», «Категории», «Меню игр», «Выбор уровня», «Игры», «Ежедневные задания».

«Главное меню» – начальное окно мобильного приложения, позволяющее с помощью кнопок взаимодействовать с ежедневной наградой за вход (Нагр), а также осуществлять переход к другим окнам:

1) кнопки, отвечающие за выбор категорий («Вычисления», «Память», «Концентрация») – к окну «Категории»;
2) кнопка «Испытания» – к окну «Испытания»;
3) кнопка–значок «Задания» (ЕЗ) – к окну «Ежедневные задания»;
4) кнопка «Магазин» – к окну «Магазин»;
5) кнопка «Рекорды» – к окну «Рекорды»;
6) кнопка «Настройки» – к окну «Настройки».

«Категории» – окно выбора развивающей игры из некоторого списка. Нажатие на один из элементов этого списка осуществляет переход к окну «Меню игр».

«Меню игр» – окно выбора параметров определенной развивающей игры. Оно содержит несколько кнопок, которые осуществляет переход к другим окнам:

1) кнопка «Уровень» – к окну «Выбор уровня»;

2) кнопки различных режимов («Тренировка», «Скорость», «Практика» и т.п.) – к окну «Игра».

«Выбор уровня» – окно выбора необходимого уровня из представленного списка. Нажатие на любой уровень осуществляет переход к окну «Меню игр».

«Игра» – окно непосредственно игрового процесса. С помощью специальных кнопок, представленных на экране (цифры, буквы, арифметические знаки) и полей ввода пользователь может взаимодействовать с игрой. Нажатие кнопки завершения прохождения или кнопки «Назад» осуществляет переход к окну «Меню игр».

«Достижения» – окно просмотра максимального результата, достигнутого пользователем в каждой развивающей игре. При нажатии на кнопку «Подробнее» («I») осуществляется переход к окну «График рекордов», который содержит последние пять попыток в конкретной развивающей игре (с указанием даты и времени получения), а также графическое представление результатов в виде кривой.

«Испытания» – окно, представляющее собой своеобразное меню, содержащее кнопки для автоматической и ручной генерации испытания (развивающей игры, собранной из двух других игр), а также кнопки для выбора уровня тех игр, из которых будет составляться испытание. Взаимодействие с этими кнопками осуществляет переход к окну «Игра» и «Уровень» соответственно.

«Настройки» – окно настроек приложения. С помощью кнопок выбора языка и кнопки включения/выключения звука пользователь может изменять текущие настройки приложения.

«Магазин» – окно внутриигровых товаров, покупка которых несколько упростит процесс достижения прогресса в развивающих играх.

«Ежедневные задания» – окно, содержащее список процедурно–генерируемых ежедневных заданий, за выполнение которых пользователь может получить определенную награду.

Большинство окон также будут отображать информацию о внутриигровой валюте, которую заработает пользователь (ВВ).

Схема экранов разрабатываемого программного продукта представлена на рисунке 20. Для простоты восприятия обратные переходы на схеме выделены красным цветом и соединяют экраны (так как большинство переходов осуществляется при помощи кнопки «Назад», присутствующей на всех мобильных устройствах Android).

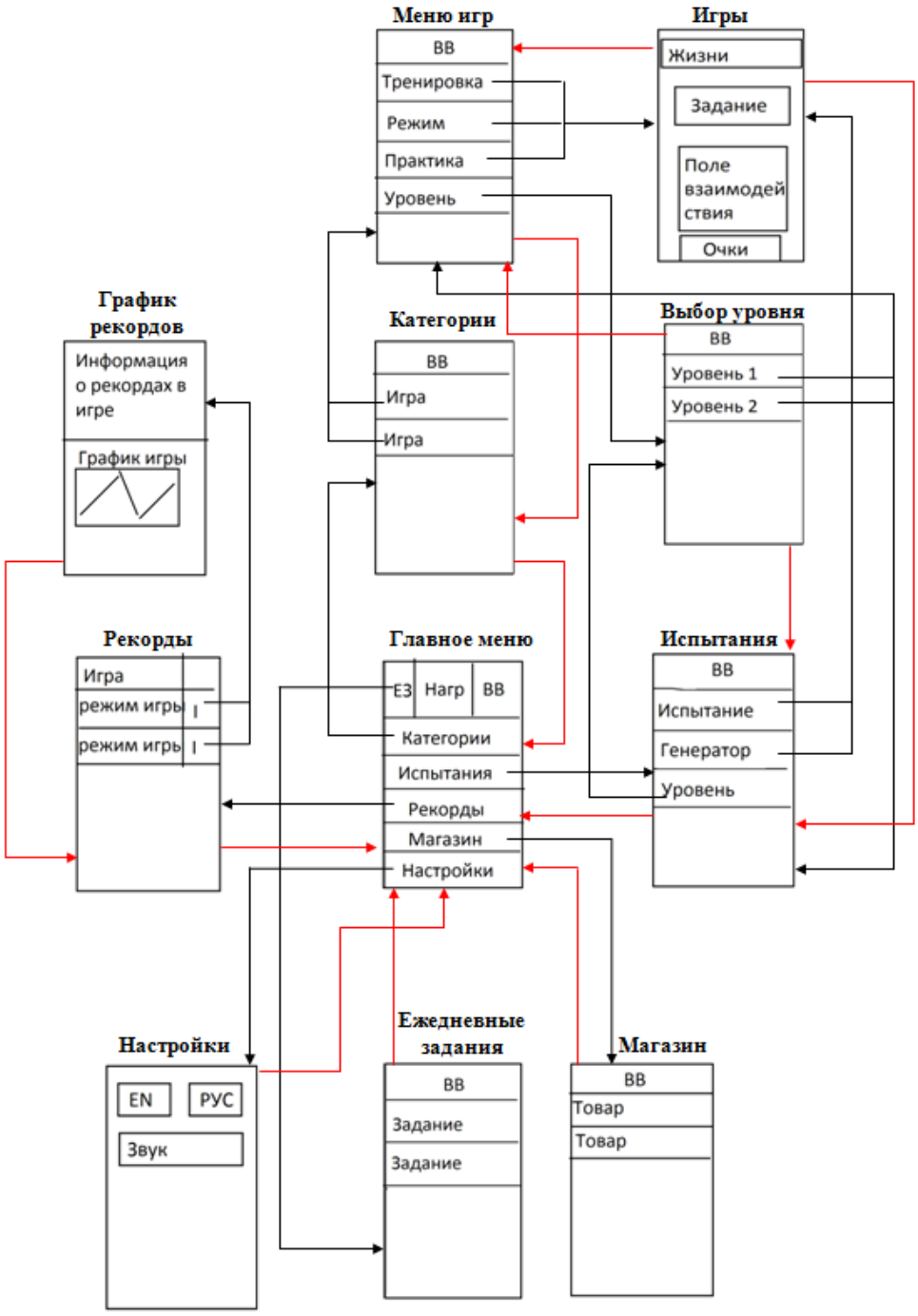

Рисунок 20 – Схема экранов разрабатываемого приложения

4 РЕАЛИЗАЦИЯ РАЗРАБОТАННОГО МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ РАЗВИТИЯ КОГНИТИВНЫХ СПОСОБНОСТЕЙ

4.1 Реализация компонентов мобильного приложения для развития когнитивных способностей

Так как для реализации нашего приложения был выбран язык Java, а большинство активностей, следуя описанной выше логике диалога с пользователем, представляют собой различные списки, воспользуемся функциями встроенного класса ArrayAdapter, которые изначально предназначены для работы с элементами списка.

Так как списки в нашем приложении представляют не только обычный массив текстов, но и дополнительные элементы (изображения, кнопки), наследуем наши классы StateAdapter, TaskAdapter и ShopAdapter от базового класса ArrayAdapter.

Для реализации класса StateAdapter нам необходимо всего лишь переопределить метод базового класса getView(), в котором указать, какие конкретно параметры будут передаваться. Описание этого метода приведено ниже.

```
public View getView(int position, View convertView, ViewGroup parent){
    View view=inflater.inflate(this.layout, parent, false);
    ImageView flagView = view.findViewById(R.id.flag);
    TextView nameView = view.findViewById(R.id.name);
    TextView pointView = view.findViewById(R.id.pnt);
    State state = states.get(position);
    flagView.setImageResource(state.getPicture());
    nameView.setText(state.getName());
    if(state.getPnt()>=0 && state.getFlag()==1){
       pointView.setText(state.getPnt()+"%");
    }else if(state.getPnt()>=0 && state.getFlag()!=1){
```

```
            pointView.setText(String.valueOf(state.getPnt()));
        }else{pointView.setText("");}
        return view;}}
```

Так как этот адаптер предназначен для отображения похожих элементов, для корректного отображения мы применяем методы объектов нашего класса State. Эти условия необходимы для отображения простого списка с картинкой, списка с процентами (необходим для прохождения развивающих игр) и списка с дополнительным текстовым полем (например, выбор уровня). Для взаимодействия с элементами полученных списков используется слушатель OnItemClickListener.

Классы для работы со списками заданий или списками товаров используют адаптеры с дополнительными кнопками. Их обработка заключается в дополнительном кодировании слушателя кнопок (setOnClickListener). Пример такого обработчика для класса взаимодействия с ежедневными заданиями приведен ниже.

```
viewHolder.addButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            task = TaskList.get(position);
            ClassSound.playSound(SoundClick);
            if(task.getFlag()==1){
                task.setFlag(2);
                task.setBonus(formatter.format(curDate));
                if (mContext instanceof EveryDayTask) {
                    ((EveryDayTask)mContext).showResult_star(position);}}}}
```

В зависимости от нажатой кнопки соответствующего элемента списка метод проверяет, выполнено ли задание (проверка соответствующего флага). Если задание выполнено, то адаптер вызывает метод класса, который содержит ссылку на адаптер.

Как отмечалось ранее, наше приложение реализовано при помощи модулей, передающих между собой информацию о состоянии игры (необходимую для автоматического сохранения состояния). Для передачи такой информации воспользуемся механизмом намерений (Intent), встроенном в язык программирования Java. Если наш модуль отправляет информацию и при этом ждет информацию в ответ, воспользуемся методом startActivityForResult() и перегрузим метод onActivityResult(), чтобы понимать, в какую активность мы отправили информацию и из какой активности ее ждем.

Далее рассмотрим часть метода генерации ежедневных заданий. Сохранение сгенерированных значений будет происходить в массиве ежедневных заданий (everyday_mas), в который мы занесем сгенерированные номера заданий и отметим выполненными все остальные виды заданий, которые не были выбраны в конкретный день. Генерация номеров заданий и формирование массива ежедневных заданий представлены ниже.

```
separated = everyday_mas[0].split(" ");
int gen = Integer.parseInt(separated[1]);
int gen1 = Integer.parseInt(separated[2]);
if(separated[1].equals("-1")) {gen = new Random().nextInt(3-0)+0; separated[1]=String.valueOf(gen);}
if(separated[2].equals("-1")) {
gen1=new Random().nextInt(tk); separated[2]=String.valueOf(gen1);
while (gen1==gen){gen1=new Random().nextInt(tk); separated[2]=String.valueOf(gen1);}}
SimpleDateFormat formatter = new SimpleDateFormat("dd.MM.yyyy");
formatter.setLenient(false); Date curDate = new Date();
for (int i = 0; i < GamesName.length; i++) {if(i!=gen && i!=gen1) {everyday_mas[i+tr] = "2";everyday_mas[i+dr]=formatter.format(curDate);}}
everyday_mas[0] = String.join(" ", separated);
```

Для сохранения информации в базу данных воспользуемся объектом специального класса для добавления строк в таблицу (contentValues). В зависимости от текущей игры будем формировантъ запрос, чтобы получить сохраненные результаты. Это необходимо, что получить номер последней попытки. На основе этой информации, а также на основе результата прохождения игры и текущей даты и времени, которые мы получаем с помощью встроенного функции Locale.getDefault(), в наш объект класса контейнера будет занесена информация, которая затем будет записана в базу данных. Фрагменты кода, отвечающие за добавление результата прохождения игры в базу данных приведены ниже.

```
String selectQuery = "SELECT num_attempt\n" +"from ACHIEVMENT \n" + "WHERE id_game=? and id_training=?;";
Cursor cursor = sqLiteDatabase.rawQuery(selectQuery, new String[]{String.valueOf(id_g),String.valueOf(id_t)});
        if (cursor != null && cursor.getCount() > 0) {
           cursor.moveToLast(); attempt = cursor.getInt(0);}
        contentValues.put("NUM_ATTEMPT", attempt + 1);
        contentValues.put("RESULT", Integer.parseInt(thiefname));
        Date currentDate = new Date();
        DateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss", Locale.getDefault());
        String dateText = dateFormat.format(currentDate);
        contentValues.put("DATARESULT", dateText);
        sqLiteDatabase.insert("ACHIEVMENT", null, contentValues);
```

В конце рассмотрим функции для генерации задания одного из классов логики, необходимый для возможности прохождения нашей развивающей игры.

Так как каждая игра реализована с использованием своих функций, здесь опишем только функции класса ConcentrateModel, реализующие логику генерации задания в игре «Пропуск».

Эта игра инициализирует первоначальный массив в зависимости от выбранного пользователем уровня, меняет его элементы местами при помощи функции shuffle(), а далее создает строку на основе перемешанного массива. Создание строки заключается в занесении в нее всех элементов полученного массива, кроме последнего. Последний элемент необходим как раз для проверки правильности выбора пользователя. Функции, необходимые для генерации задания такого типа приведены ниже.

```
private void init_mas(){
    if(level==1){ mas = new int[]{1, 2, 3, 4, 5};
    }else{mas = new int[]{1, 2, 3, 4, 5, 6};}}
  public String generate_massiv(){
    init_mas(); shuffle();
    int[] new_mas = new int[mas.length–1];
    StringBuilder str = new StringBuilder();
    for (int i=0;i<new_mas.length;i++){
      new_mas[i]=mas[i]; str.append(new_mas[i]).append("");}
    last_elem=mas[mas.length–1]; return str.toString();}
  private void shuffle() {
    Random rnd = new Random();
    for(int i = 0; i < mas.length; i++) {
      int index = rnd.nextInt(i + 1);
      int a = mas[index];  mas[index] = mas[i];  mas[i] = a;}}
```

Основные классы, реализованные в нашей программе, приведены в Приложении А.

4.2 Реализация пользовательского интерфейса мобильного приложения для развития когнитивных способностей

После запуска приложения пользователь видит главное окно, внешний вид которого отображен на рисунке 21. Данное окно содержит следующие элементы:

1) Кнопки «Вычисления», «Память» и «Концентрация», относящиеся к категориям;

2) Кнопка с картинкой «Ежедневное задание» в верхней части экрана;

3) Кнопка с картинкой «Ежедневная награда» в верхней части экрана;

4) Кнопка «Испытания»;

5) Кнопка «Рекорды»;

6) Кнопка «Магазин»;

7) Кнопка «Настройки».

Также в верхней части экрана находится внутриигровая валюта.

Рисунок 21 – Главное окно приложения

Нажатие кнопки «Настройки» в главном окне приложения позволяет перейти к окну «Настройки», вид которого представлен на рисунке 22, и управлять основными настройками приложения с помощью следующих дополнительных элементов:

1) кнопки выбора языка (EN – английский, РУС – русский);

2) кнопка–чекбокс «Звук» для управления включения и выключения звука;

Кнопка «Назад», находящаяся на всех телефонах с ОС Android и расположенная по умолчанию внизу позволяет вернуться обратно к главному окну приложения, вид которого отображен на рисунке 21.

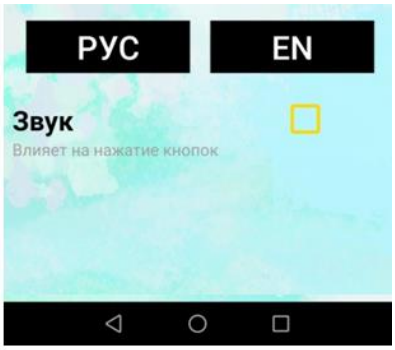

Рисунок 22 – Экран настроек приложения

Нажатие кнопки «Магазин» в главном окне приложения позволяет перейти к окну «Магазин», внешний вид которого представлен на рисунке 23, и просмотреть доступный список товаров, цену каждого и его количество, которое имеется у пользователя. Кнопка «Назад» позволяет вернуться обратно к главному окну приложения, вид которого отображен на рисунке 21.

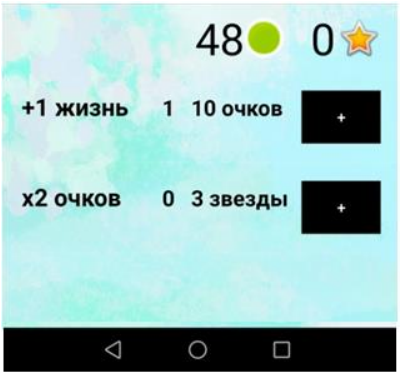

Рисунок 23 – Внутриигровой магазин приложения

Нажатие кнопки «Рекорды» в главном окне приложения позволяет перейти к окну «Рекорды», внешний вид которого представлен на рисунке 24, и просматривать информацию о максимально достигнутом результате в различных режимах и на различных уровнях развивающих игр (преимущественно в режиме «Тренировка» 1 и 2 уровня). Кнопка «Назад»

позволяет вернуться обратно к главному окну приложения, вид которого отображен на рисунке 21.

Рисунок 24 – Окно отображения рекордов

Нажатие на кнопку со знаком «I» позволяет подробно изучить свои последние попытки прохождения в конкретной развивающей игре и увидеть их представление на графике. Внешний вид окна представлен на рисунке 25. Кнопка «Назад» позволяет вернуться обратно к окну «Рекорды», вид которого отображен на рисунке 24.

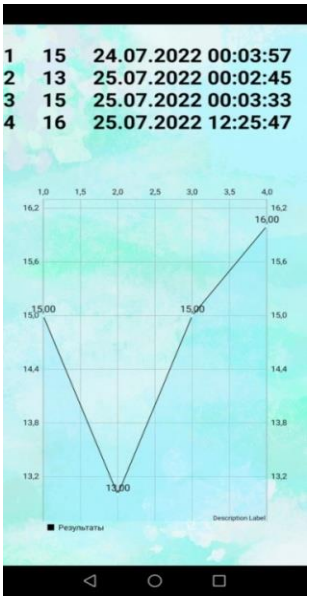

Рисунок 25 – Окно подробного отображения результатов в игре

Нажатие на кнопку «Испытания» в главном окне приложения позволяет перейти к окну «Испытания», внешний вид которого представлен

на рисунке 26. В окне представлено два режима: автоматическая и ручная генерация испытания, а также кнопка выбора конкретного уровня, которые будут принимать участие в генерации. Кнопка «Назад» позволяет вернуться обратно к главному окну приложения, вид которого отображен на рисунке 21.

Рисунок 26 – Окно категории «Испытания» в игре

Выбор категории из списка в главном окне приложения позволяет перейти к окну конкретной категории, внешние виды которых представлены на рисунке 27. Эти окна содержат список реализованных развивающих игр. Кнопка «Назад» позволяет вернуться обратно к главному окну приложения, вид которого отображен на рисунке 21.

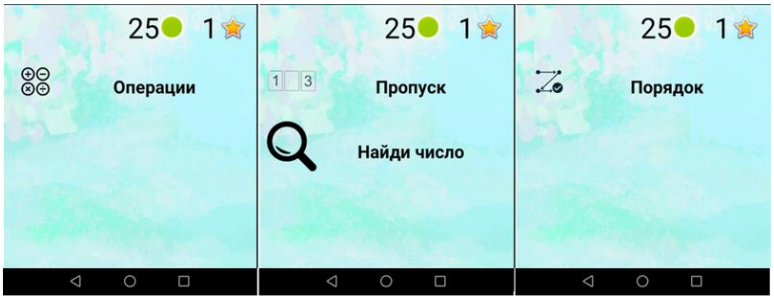

Рисунок 27 – Примеры развивающих игр в категориях «Вычисления», «Концентрация» и «Память»

Выбор конкретной развивающей игры из списка позволяет перейти к окну «Меню игры», пример внешнего вида которого представлен на рисунке 28. Окно содержит кнопки с различными режимами игры и кнопку для

выбора уровня. Кнопка «Назад» позволяет вернуться обратно к одному из окон категорий, виды которых отображены на рисунке 27.

Рисунок 28 – Пример меню развивающей игры

Нажатие на кнопку «Уровень» в окне «Испытания» или в окне конкретной развивающей игры позволяет перейти к окну «Выбор уровня», внешний вид которого представлен на рисунке 29, в котором представлены доступные для прохождения уровни. Кнопка «Назад» или выбор конкретного уровня позволяют вернуться к предыдущему отображаемому окну (рисунки 26, 27).

Рисунок 29 – Окно выбора уровня в игре

Нажатие на кнопку конкретного режима приведет к отображению диалогового окна с описанием, успешное завершение работы с которым позволяет из окна «Испытания» или «Меню игры» перейти к окну «Игра», примеры внешних видов которого приведены на рисунке 30. На экране отображены различные элементы прямого взаимодействия, таймер, в течение

которого нужно проходить игру, счетчик очков и количество жизней. Кнопка «Назад» позволяет вернуться к предыдущему отображаемому окну (рисунки 26, 27).

Рисунок 30 – Примеры игрового процесса

По достижении таймером нуля или при трате всех жизней появится одно из окон окончания игры (Рисунок 31). В окне находится интересный факт, связанный с когнитивными способностями, количество полученных очков, а также кнопка завершения игры. Нажатие кнопки позволяет вернуться к предыдущему отображаемому окну (рисунки 26, 27).

Рисунок 31 – Примеры окна завершения развивающей игры

Нажатие кнопки «Ежедневные задания» в главном окне приложения позволяет перейти к окну «Ежедневные задания», возможные варианты внешнего вида которого представлены на рисунке 32. Окно содержит список заданий с кнопкой их завершения (при выполнении определенных условий). Кнопка «Назад» позволяет вернуться обратно к главному окну приложения, вид которого отображен на рисунке 21.

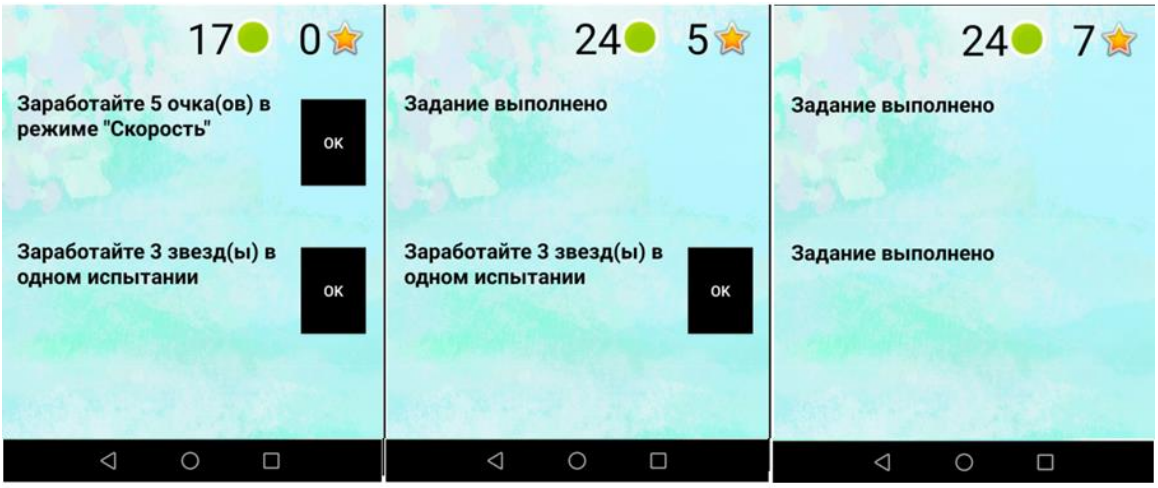

Рисунок 32 – Окно ежедневных заданий в игре

4.3 Тестирование мобильного приложения для развития когнитивных способностей

Тестирование представляет собой процесс анализа разработанного программного продукта и сопутствующей документации с целью выявления дефектов и повышения качества. Основное назначение тестирования – проверить способность программного обеспечения при заданных условиях удовлетворять установленным или предполагаемым потребностям [11].

Тестирование позволяет определить, работает ли система так, как ожидается, выявить пробелы в реализации и отличия от фактических требований.

Процесс тестирования можно проводить при помощи ручных или автоматизированных инструментов для оценки одного или нескольких интересующих свойств программного продукта.

Основным методом тестирования приложений, содержащих в себе игровые составляющие – проверка реализуемости функциональных требований, то есть определение способности программного обеспечения решать поставленные задачи. Проведение тестирования позволяет подтвердить, что система реализована в соответствии с предъявленными к ней функциональными требованиями и готова к работе.

На этапе функционального тестирования разработанного мобильного приложения для развития когнитивных способностей был сформирован набор общих входных состояний, общие действия пользователя, ожидаемые результаты и результаты проверки, представленный в таблице 2.

Таблица 2 – Результаты проведения функционального тестирования

Тест	Проверка	Ожидаемый результат	Проверка
1	2	3	4
1	Взаимодействие с настройками	Изменение состояния звука при взаимодействии с кнопкой–чекбоксом, а также языка интерфейса (кнопки «РУС» и «EN»)	Верно
2	Взаимодействие с магазином	Изменение состояния внутриигровой валюты и счетчика товара при его покупке	Верно
3	Просмотр рекордов	Отображение максимально полученного результата в каждой развивающей игре	Верно
4	Просмотр информации о рекордах в игре	Отображение последних пяти попыток с их датой и временем получения и графика результатов	Верно
5	Взаимодействие с графиком результатов	Изменение масштаба графика	Верно
6	Взаимодействие с испытаниями	Возможность проходить сгенерированное или составленное вручную (с указанием нужных параметров) испытание	Верно
7	Взаимодействие с уровнями	Отображение списка уровней и возможность выбрать нужный уровень (отображение выбора)	Верно
8	Взаимодействие с категориями	Отображение доступных развивающих игр в каждой категории	Верно

Продолжение таблицы 2

1	2	3	4
9	Взаимодействие с меню развивающей игрой	Возможность проходить развивающую игру (с указанием нужных параметров при необходимости)	Верно
10	Взаимодействие с развивающей игрой	Возможность пройти развивающую игру в выбранном режиме (взаимодействовать с игровыми элементами). Отображение результата прохождения игры в конце	Верно
11	Взаимодействие с закрытым контентом	Возможность открыть различные игровые режимы за внутриигровую валюту	Верно
12	Взаимодействие с ежедневной наградой	Изменение состояния внутриигровой валюты при наступлении нового дня	Верно
13	Взаимодействие с ежедневными заданиями	Генерация заданий при наступлении нового дня. Изменение состояния внутриигровой валюты при выполненных условиях и нажатии на кнопку выполнения задания	Верно

Проведя функциональное тестирование системы целиком, можно сделать вывод, что системные элементы реализованы верно и выполняют назначенные функции. В рамках данной выпускной квалификационной работы реализованы и протестированы описанные на этапе постановки функциональных требований механики.

ЗАКЛЮЧЕНИЕ

Так как предметная область разрабатываемого программного проекта была поставлена, функциональные требования к мобильному приложению были сформулированы, был проведен анализ требований и была определена функциональная спецификация мобильного приложения, а также проектирование и реализация программного обеспечения осуществлены, поставленную цель курсового проекта можно считать достигнутой.

Разработанное программное обеспечение позволит поддерживать активность своего мозга на достаточно высоком для выполнения повседневных действий уровне и при этом будет всегда доступно пользователю на его мобильном устройстве. Практическая значимость разработанного приложения заключается в оптимизации процесса развития памяти, концентрации, вычислительных способностей за счет прохождения различных развивающих игр в любом месте и в любое время.

Можно отметить, что на основе анализа предметной области было принято решение включить в разработку следующие основные моменты: создание тренировки как автоматически, так и вручную, система мотивации пользователя (система прогресса, бонусы, ежедневные задания, игровой магазин и внутриигровая валюта), развивающие игры с подробным описанием, распределенные по категориям (при этом были реализованы игры, опирающиеся на идеи центра CUMON – быстрый счет: определение правильного знака в примере, запоминание последовательности цифр и букв, а также соотнесение текстового представления числа с его представлением в виде цифр), система рекордов с точным указанием времени получения результата, а также возможность бесплатного использования приложением.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Android Studio: среда разработки мобильных приложений [Электронный ресурс]. – Режим доступа: https://arduinoplus.ru/android–studio/ (дата обращения: 04.03.2023).

2. BrainExer 2.0 [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=com.brxr.next&hl=ru&gl=ru (дата обращения: 16.07.2022).

3. Java для разработки для Android [Электронный ресурс]. – Режим доступа: https://code.tutsplus.com/ru/tutorials/learn–java–for–android–development–introduction–to–java—mobile–2604 (дата обращения: 04.03.2023).

4. SQLite в Android Studio: база данных [Электронный ресурс]. – Режим доступа: https://randroid.ru/dev/sqlite–v–android–studio–baza–dannykh (дата обращения: 04.03.2023).

5. Влияние игровой деятельности на уровень усвоения знаний обучающихся [Электронный ресурс]. – Режим доступа: https://www.prodlenka.org/metodicheskie-razrabotki/311733-vlijanie-igrovoj-dejatelnosti-na-uroven-usvoe (дата обращения: 04.03.2023).

6. Кейс геймификации мобильного приложения: советы и примеры [Электронный ресурс]. – Режим доступа: https://spark.ru/startup/russiangeeks/blog/72620/kejs-gejmifikatsii-mobilnogo-prilozheniya-soveti-i-primeri?ysclid=le9yjjy0e335170846 (дата обращения: 04.03.2023).

7. Мобильные операционные системы (ОС). [Электронный ресурс]. – Режим доступа: https://apptoday.ru/publication/reiting-mobilnykh-operatsionnykh-sistem?ysclid=lecx7p6mrf436561318 (дата обращения: 04.03.2023).

8. ОСНОВЫ УНИФИЦИРОВАННОГО ЯЗЫКА МОДЕЛИРОВАНИЯ [Электронный ресурс]. – Режим доступа:

https://www.sites.google.com/site/anisimovkhv/learning/pris/lecture/tema11 (дата обращения: 04.03.2023).

9. Разработка пользовательского интерфейса [Электронный ресурс]. – Режим доступа: https://studbooks.net/2039322/informatika/razrabotka_polzovatelskogo_interfeysa?ysclid=l6ajqr86js233346541 (дата обращения: 04.03.2023).

10. Сервисы и приложения, которые сделают вас умнее [Электронный ресурс]. – Режим доступа: https://lifehacker.ru/razvivaj-mozg/?ysclid=letyggthwh2759516 (дата обращения: 04.03.2023).

11. Тестирование программного обеспечения. Базовый курс. [Электронный ресурс]. – Режим доступа: https://www.bsuir.by/m/12_108786_1_98216.pdf?ysclid=l6ajnwuoaz93025741411 (дата обращения: 04.03.2023).

12. Тренер мозга 8.6.9 [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=com.raghu.braingame (дата обращения: 04.03.2023).

13. Тренируем мозг [Электронный ресурс]. – Режим доступа: https://publications.hse.ru/mirror/pubs/share/folder/c6u29h5eq3/direct/201356170?ysclid=l66dnn2iqq657384573 (дата обращения: 04.03.2023).

14. Функциональные и нефункциональные требования (Functional and Non–functionalRequirements) [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/2152457/page:4/ (дата обращения: 04.03.2023).

15. Что такое архитектура программного приложения? [Электронный ресурс]. – Режим доступа: https://yapro.ru/uploads/users/1/2021/01/22/eeles–pdf.pdf–19–48–46.pdf (дата обращения: 04.03.2023).

16. Что такое когнитивные способности и как их развить [Электронный ресурс]. – Режим доступа: https://iklife.ru/samorazvitie/chto-takoe-kognitivnye-sposobnosti?ysclid=lezfo4xbrx59279691 (дата обращения: 04.03.2023).

ПРИЛОЖЕНИЕ А

(обязательное)

РЕАЛИЗАЦИЯ ПРОГРАММЫ

```
classMemoryModel {
privateStringBuilderinputStr = new StringBuilder();
privateintsum_point;
privateint time;
privateint count=1;
privateintnum;
private String stroka;
privateint liter;

private Sound ClassSound;
privateintSoundClick;

MemoryModel(int time, int speed, AssetManagermAssetManager, intflag_zvuk){
this.time=time;
this.liter=speed;
ClassSound = new Sound(mAssetManager,flag_zvuk);
SoundClick = ClassSound.loadSound("zvuk_1.ogg");
   }

public void play_sound(){
ClassSound.playSound(SoundClick);
   }

privateintgenerate_number(){
      Random rand = new Random();
intArg=0;
Arg=rand.nextInt(8)+1;
returnArg;
   }

private String generate_liter(){
      String abc = "abcdefghijklmnopqrstuvwxyz";
      Random rand = new Random();
char letter = abc.charAt(rand.nextInt(abc.length()));
returnString.valueOf(letter);
   }

public String generate_game(){
inti=0;
while(i<count) {
if(liter==0) {
num = generate_number();
inputStr.append(num).toString();
}else{
stroka=generate_liter();
```

```
inputStr.append(stroka).toString();
          }
i+=1;
       }
returninputStr.toString();
   }

publicbooleancheck_answer(String answer) {
play_sound();
if (answer.equals(inputStr.toString())){
sum_point+=1;
return true;
       }
else{
return false;
       }
   }

public String getExpr() {
returninputStr.toString();
   }

public String getTextpoint() {
StringBuilderstr = new StringBuilder();
returnstr.append(sum_point).toString();
   }

public String getTime() {
StringBuilderstr = new StringBuilder();
returnstr.append(time).toString();
   }

public void Setcount(int count) {
this.count=count;
   }

public void ClearStringBuilder(){
inputStr.setLength(0);
}
}
```