

Реферат

Дипломная работа: 93 страницы, 68 рисунков, 6 таблиц, 2 приложения, 6 использованных источников.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ИГРА, АРКАДА, GALOS, ANDROID, MOBILE APP.

Цель работы: разработать игру жанра аркада для мобильных устройств.

Задачи работы:

* выбрать технологию разработки мобильного приложения;
* выполнить анализ и разработку требований к разрабатываемой игре;
* выполнить проектирование игры;
* выполнить тестирование и отладку игры;
* выполнить ввод в эксплуатацию разработанной игры.

Дипломная работа включает в себя разделы: «Разработка и анализ требований», «Проектирование», «Конструирование», «Тестирование и отладка», «Ввод в эксплуатацию».

В разделе «Разработка и анализ требований» описана актуальность разработки мобильных игр, приведена информация о наиболее актуальной операционной системе и её версии для разработки мобильных приложений, а также показана разработка требований к мобильной игре.

В разделе «Проектирование» описаны вид приложения, его архитектура, архитектурный шаблон, информационное обеспечение, программное и техническое обеспечение, проект взаимодействия элементов программного обеспечения.

В разделе «Конструирование» описано обоснование выбора среды программирования и языка реализации, описана организация файлов хранилища исходных кодов, исполняемых файлов и внешних компонентов, описан стиль кодирования и правила оформления кода, описаны способы поддержки версий файлов и сборок в процессе разработки.

В разделе «Тестирование и отладка» описано обоснование выбора и анализ возможностей используемого инструмента тестирования, описана методика и результаты модульного и функционального тестирования, описана методика и результаты тестирования пользовательского интерфейса и показана разработка тестовых сценариев.

В разделе «Ввод в эксплуатацию» описан ввод в эксплуатацию разработанной мобильной игры.

Результатом работы стала игра «Galos» жанра аркада для мобильных устройств на Androidверсии 4.1 JellyBean и выше, а также необходимая документация к ней.

Содержание

ВВЕДЕНИЕ 9

1 Разработка и анализ требований 10

1.1 Актуальность разработки 10

1.2 Анализ популярных мобильных операционных систем 11

1.3 Анализ преимуществ и недостатков аналогов 12

1.3.1 RGBCircles 12

1.3.2 Agar.io 13

1.4 Цель разработки 15

1.5 Основные бизнес процессы и бизнес правила 15

1.6 Границы системы 16

1.7 Варианты использования системы 19

1.8 Спецификация требований 20

1.8.1 Пользовательские истории 20

1.8.2 Общее описание системы 20

1.8.3 Функции системы 21

1.8.4 Нефункциональные требования 24

1.9 Концептуальная модель предметной области 25

2 Проектирование 27

2.1 Вид приложения 27

2.2 Архитектура приложения 27

2.3 Архитектурный шаблон 28

2.4 Информационное обеспечение 29

2.4.1 Логическая модель БД 29

2.4.2 Физическая модель БД 30

2.4.3 SQL скрипты для создания таблиц БД 31

2.5 Решения по программному обеспечению 33

2.5.1 Диаграмма классов 34

2.5.2 Диаграммы взаимодействия программных модулей 49

2.5.3 Проект окон и их взаимодействие 51

2.6 Техническое обеспечение 52

3 Конструирование 53

3.1 Обоснование выбора среды программирования и языка реализации 53

3.2 Стиль кодирования и правила оформления кода 54

3.2.1 Имена файлов, пакетов 54

3.2.2 Имена методов, переменных 54

3.2.3 Отступы, длина строки, переносы строк 54

3.2.4 Расположение блоков, операторов, пробелов, скобок 55

3.2.5 Структурирование кода 55

3.3 Структура проекта 55

3.4 Способ поддержки версий файлов и сборок в процессе разработки 59

4 Тестирование и отладка 61

4.1 Выбор инструментов тестирования 61

4.2 Методы и результаты модульного тестирования 61

4.2.1 Тестирование базового пути 63

4.2.2 Тестирование ветвей и операторов отношений 64

4.2.3 Построение информационного графа 66

4.2.4 Автоматизация тестирования 66

4.3 Тестирование пользовательского интерфейса 68

5 Ввод в эксплуатацию 71

5.1 Установка приложения на мобильное устройство 71

5.2 Руководство пользователя 74

5.2.1 Введение 74

5.2.2 Запуск приложения 75

ЗАКЛЮЧЕНИЕ 81

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 82

Приложение А. Графические материалы доклада 83

Приложение В. Отчёт о проверке ВКР на плагиат 93

Определения, обозначения и сокращения

|  |  |
| --- | --- |
| Аркада | * это жанр компьютерных игр, характеризующийся коротким по времени, но интенсивным игровым процессом. |
| Android | - это операционная система для смартфонов, планшетов, электронных книг, цифровых проигрывателей, наручных часов, фитнес-браслетов, приставок, ноутбуков, нетбуков, смартбуков, очков GoogleGlas, телевизоров и других устройств. |
| Galos | - название разрабатываемой системы, которая является игрой жанра аркада для мобильных устройств под операционную систему Android |
| «таймкиллер» | - название мобильных игр, нацеленных на то, чтобы игрок быстро и незаметно провёл время. |
| PlayMarket | - магазин приложений, игр, книг, музыки и фильмов компании [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)) и других компаний, позволяющий владельцам устройств с операционной системой [Android](https://ru.wikipedia.org/wiki/Android) устанавливать и приобретать различные приложения. |
| DU-цепь | -  структура данных, используемая в [анализе потоков данных](https://en.wikipedia.org/wiki/Data-flow_analysis) для нахождения мест использования определяемой переменной или нахождения всех мест определений значений переменных, входящих в выражение. |
| DU-тестирование | - тестирование всех DU-цепочек приложения. |
| База данных (БД) | - представленная в [объективной](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D1%84%D0%B8%D0%BB%D0%BE%D1%81%D0%BE%D1%84%D0%B8%D1%8F)) форме [совокупность](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) самостоятельных материалов (статей, [расчётов](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5), [нормативных актов](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%B0%D0%B2%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B0%D0%BA%D1%82), [судебных решений](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%B4%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5) и иных подобных материалов), [систематизированных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) таким образом, чтобы эти материалы могли быть [найдены](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и обработаны с помощью [электронной вычислительной машины (ЭВМ)](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80). |
| Активити (Activity) | - это компонент приложения, который выдает экран, и с которым пользователи могут взаимодействовать для выполнения каких-либо действий, например набрать номер телефона, сделать фото, отправить письмо или просмотреть карту |
| UI | * это интерфейс пользователя, он же пользовательский интерфейс (от английского user interface) |
| Операционная система (ОС) | (англ. operating system, OS) – комплекс взаимосвязанных программ, предназначенных для управления ресурсами [компьютера](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) и организации взаимодействия с пользователем. |

#### D:\ISPU\ВКР\Сканы\2.jpg

#### D:\ISPU\ВКР\Сканы\3.jpg

#### D:\ISPU\ВКР\Сканы\4.jpg

#### ВВЕДЕНИЕ

В каждом современном мобильном устройстве есть разнообразные приложения, целью которых является упрощение жизни пользователя и удовлетворение его потребностей. Среди таких мобильных приложений есть игры, которые чаще всего являются «таймкиллерами» – это приложения, созданные для того, чтобы скоротать несколько минут в общественном транспорте или ожидая своей очереди на приеме у врача, занять себя чем-то в перерыве или просто развлечься, отвлекаясь от повседневных забот. Такая игра рассчитана на то, чтобы пользователь смог быстро ее запустить и так же быстро из нее выйти при необходимости.

Появление мощных мобильных устройств в качестве замены простым и менее функциональным мобильным телефонам вызвало своеобразный бум игровой индустрии. И это то место, где могут проявить себя производители как с большим штатом и долгой историей работы, так и всего один человек, при наличии нужного количества знаний и времени, но это не значит, что такие игры станут менее качественными. На оценку игры не влияет то, кем она разрабатывается, так как сейчас на рынке мобильных игровых приложений побеждает тот, кто создает небольшую, но интересную игру.

Одними из самых ярких примеров таких игр можно назвать мобильные игры жанра аркада, где весь игровой процесс заключается в простом нажатии по экрану. Сам жанр многими любим ещё с детства, а простота игрового процесса затягивает игрока на долгое время, а благодаря тому, что их дистрибутив занимает мало места на телефоне, такие игры можно обнаружить даже в новых телефонах с прилавков магазинов сотовой связи.

В данной научно исследовательской работе представлены разработка и анализ требований для мобильной игры жанра аркада для мобильных устройств под названием «Galos». На данный момент времени актуальность этой системы обусловлена малым количеством её аналогов, и тем, что игры являются неотъемлемой частью почти каждого современного человека.

#### 1 Разработка и анализ требований

1.1 Актуальность разработки

На протяжении последних  лет  показатель, характеризующий уровень спроса  на  мобильные  устройства, постоянно растет.  Такая  статистика  позволяет сделать вывод о том, что  разработка  мобильных приложений актуальна и целесообразна. Главное  грамотно  оценить,  для кого  и  зачем  создается софт. Только  полезная разработка  получит  достойное  признание со стороны пользователей.

На рис. 1 представлен отчёт о глобальном рынке игр от Newzoo (глобальный партнер по маркетингу на рынке для любой компании, заинтересованной в играх, смартфонах и киберспорте). Данный отчёт был размещён 20 апреля 2017 года с прогнозом дальнейшего развития глобального рынка игр до 2020 года. На нём можно видеть подтверждение тому, как игры для мобильных устройств постепенно будут вытеснять другие устройства из рынка игр, тем самым приобретая всё большую актуальность.

#### Newzoo_Global_Games_Market_Revenue_Growth_2016-2020_April_2017

Рисунок 1. Отчёт о глобальном рынке игр от экспертов Newzoo

30 апреля 2018 года эксперты из Newzoo выпустили последнее ежеквартальное обновление своего отчёта о состоянии глобального рынка игр, которое представлено на рис. 2. В этом отчёте видно, что на начало 2018 года более половины всех доходов от игр будет поступать именно из мобильного сегмента, что ещё раз подтверждает актуальность разработки мобильных игр, так как на момент написания этой работы именно они сейчас занимают большую часть глобального рынка игр.

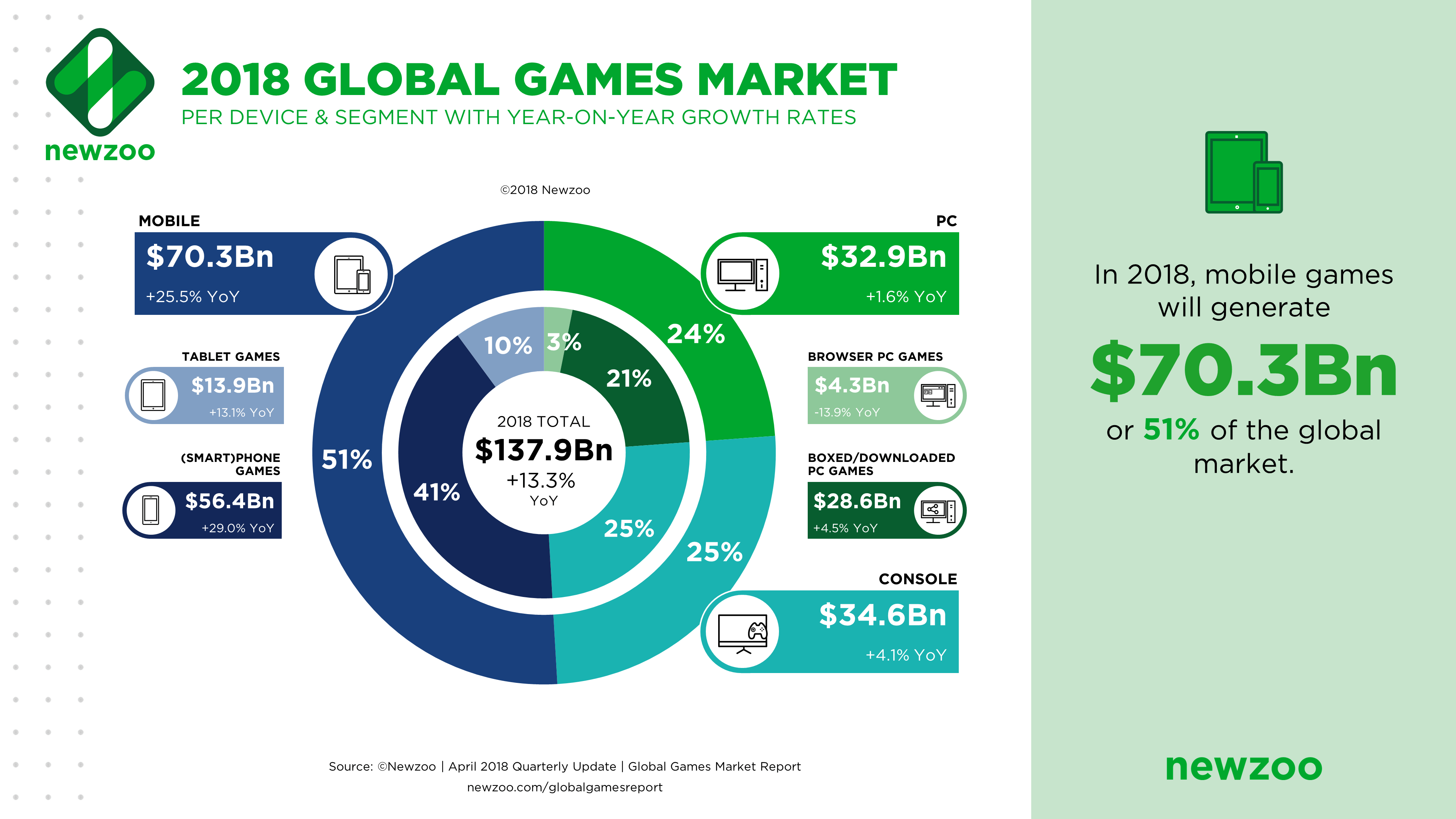


Рисунок 2. Обновлённый отчёт о глобальном рынке игр от экспертов Newzoo

1.2 Анализ популярных мобильных операционных систем

В январе 2018 года порталом NetMarketShare (портал, предоставляющий статистику использования веб-ресурсов для реальных пользователей) была выложена статистика, актуальная на момент написания этой работы, показывающая популярность операционных систем смартфонов. Данная статистика представлена на рис. 3.



Рисунок 3. Статистика операционных систем от портала NetMarketShare

Основываясь на этой статистике можно сделать вывод, что большая часть смартфонов имеет операционную систему Android, а значит, разработка приложений для этой системы имеет больший потенциал в количестве пользователей и возможной прибыли.

На основании приведенных сведений можно сделать вывод о целесообразности разработки мобильной игры на базе ОС Android.

Также стоит отметить проблему выбора версии мобильной операционной системы, актуальных для разрабатываемой игры. На рис. 4 представлен скриншот из Android Studio, на котором показана статистика о том, сколько пользователей смартфонов с OS Android смогут использовать приложение, если выбрать ту или иную версию Android.

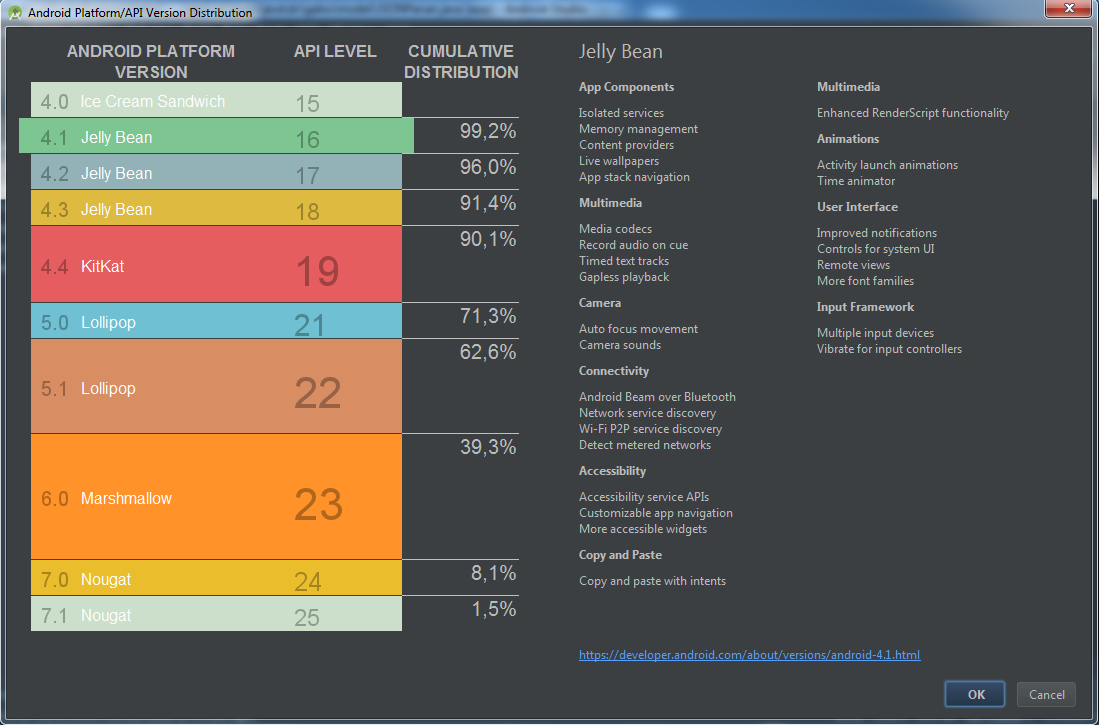


Рисунок 4. Статистика распределения версий OS Android

Можно видеть то, что при разработке приложения для OS Android версии 4.1 JellyBean можно охватить 99.2% мобильных устройств, поэтому для разработки игры в качестве минимальной версии OS Android будет использоваться 4.1 JellyBean.

1.3 Анализ преимуществ и недостатков аналогов

В PlayMarket найдены два наиболее близких по игровому процессу аналога: «RGBCircles» и «Agar.io». Выявленные преимущества и недостатки у аналогов, будут использованы для разработки мобильной игры.

1.3.1 RGBCircles

Данная игра представлена на рис. 5 и на момент написания этой работы имеет 45 скачиваний и среднюю оценку 4,4.

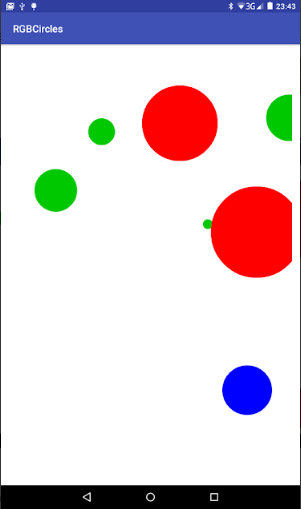


Рисунок 5. Скриншот игры «RGBCircles»

Поиграв в эту игру можно выделить следующие преимущества и недостатки:

1. Преимущества:
   * простой и оригинальный игровой процесс
   * простое управление
2. Недостатки:
   * однообразный игровой процесс без какого-либо развития
   * отсутствует меню и какие-либо настройки
   * нет никакого фиксирования игрового результата

Для разработки мобильной игры можно использовать простой и оригинальный игровой процесс, который представлен в игре «RGBCircles» и доработать его с учётом недостатков.

1.3.2 Agar.io

Данная игра представлена на рис. 6 и на момент написания этой работы имеет 3772483 скачивания и среднюю оценку 4,2. В отличие от предыдущего аналога, данная игра достаточно популярна и имеет стабильный доход за счёт встроенной рекламы и внутриигровых покупок

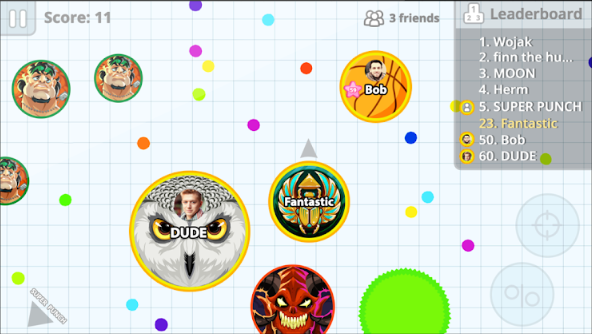


Рисунок 6. Скриншот игры «Agar.io»

Поиграв в эту игру и прочитав отзывы пользователей, можно выделить следующие преимущества и недостатки:

1. Преимущества:
   * простой и оригинальный игровой процесс
   * интуитивно понятный интерфейс
   * конкуренция среди игроков
   * простое управление
   * есть фиксация игрового результата
   * игровой процесс с реальными игроками в режиме реального времени
2. Недостатки:
   * очень сильно зависит от скорости и стабильности интернета
   * неудобно играть на экранах смартфона из-за очень больших кругов
   * на экране смартфона сложно контролировать игровой процесс, так как пальцем постоянно загораживается часть игрового пространства
   * нет возможности играть без подключения к интернету

Для разработки мобильной игры можно использовать идею конкуренции среди игроков с фиксацией игрового результата, которая добавляет игровой азарт, а также идею возможности игры, как с интернетом, так и без него. Также в игре должен быть интуитивно понятный интерфейс.

1.4 Цель разработки

Целью разработки является создание востребованного продукта на рынке мобильных игр с перспективой на получение прибыли.

Для достижения данной цели необходимо разработать простой и интересный игровой процесс с интуитивно понятным интерфейсом, а также системой рейтинга для дополнительного игрового азарта.

1.5 Основные бизнес процессы и бизнес-правила

Разрабатываемая игра для OS Android называется «Galos».

Целью игры «Galos» является прохождение одного уровня за другим. В начале уровня, игровое пространство заполняется кружками разных цветов, одним из которых управляет игрок. При контакте круга игрока с другим кругом меньшего размера, круг игрока его поглощает, увеличиваясь в размере, а поглощенный круг исчезает. Уровень считается пройденным, если круг игрока поглотит все остальные круги, которые можно поглотить. Игрок проиграет, если столкнется с кругом большего размера или с кругом, который нельзя поглотить.

Особенность мобильного приложения «Galos» в сравнении с его аналогами будет заключаться в увеличенном функционале, а именно будет добавлена система достижений, рекорды, новые враги и игровая валюта, на которую можно покупать бонусы, влияющие на игровой процесс при прохождении игры.

Игра проходит на всём рабочем экране мобильного устройства.

Из особенностей игрового процесса является то, что вражеские круги передвигаются по определенной траектории, а ударяясь об края экрана, немного изменяют свою траекторию. Круг игрока может передвигаться в любом направлении, в зависимости от того, какой точки экрана коснулся игрок. Все круги на игровом пространстве начинают своё движение в тот момент, когда игрок касается экрана, если же касания нет, то все круги замирают на своих позициях.

При запуске нового уровня на игровом пространстве хаотическим образом размещаются круги в зависимости от игрового режима:

1. Смешанный:
   1. Круг игрока (1 шт.);
   2. Неуязвимые кружки (0-5 шт.);
   3. Исчезающие кружки (0-5 шт.);
   4. Обычные кружки (5-8 шт.).
2. Очередь:
   1. Круг игрока (1 шт.);
   2. Неуязвимые кружки (0-5 шт.);
   3. Обычные кружки (5-8 шт.).
3. Догонялки:
   1. Круг игрока (1 шт.);
   2. Неуязвимые кружки (30-35 шт.);
   3. Обычный кружки (1 шт.).
4. Невидимки:
   1. Круг игрока (1 шт.);
   2. Неуязвимые кружки (0-5 шт.);
   3. Исчезающие кружки (5-8 шт.).
5. Рейтинговый режим – в этом режиме поочередно генерируется каждый из режимов (сначала «Смешанный», затем «Очередь», «Догонялки», «Невидимки», затем снова «Смешанный» и т.д., пока игрок не проиграет).

В процессе игры можно нажать на паузу и купить за игровую валюту один из трех бонусов:

1. Жизнь – позволяет сохранить рекорд и пройти уровень заново, в случае проигрыша на текущем уровне;
2. Замедление – уменьшает скорость движения вражеских кругов в 2 раза;
3. Рост – немного увеличивает размер круга игрока;

При прохождении уровня, игрок зарабатывает игровую валюту.

При входе в игру пользователь должен ввести данные аккаунта, чтобы перейти в главное меню игры, или он может создать новый аккаунт, восстановить пароль от аккаунта или войти в игру как «Гость» без возможности просмотра рейтинга игроков.

В главном меню игрок может посмотреть игровые достижения, которые он может получить или уже получил, рейтинг игроков, помощь по игре, вкл./выкл. звук, изменить данные аккаунта или вернуться к авторизации.

Разработка игры будет происходить для OS Android4.1 JellyBean и выше.

1.6 Границы системы

Диаграмма потоков данных (Data Flow Diagram**) –** основной инструмент структурного анализа. DFD позволяет определить преобразовательные процессы системы, перемещение (хранение) данных или материалов между процессами, хранилищами и внешним миром.

Для начала рассмотрим контекстную диаграмму (рис. 7) - наивысший уровень абстракции диаграммы потоков данных. Контекстная диаграмма представляет всю систему как единый процесс, изображенный в виде одного блока.

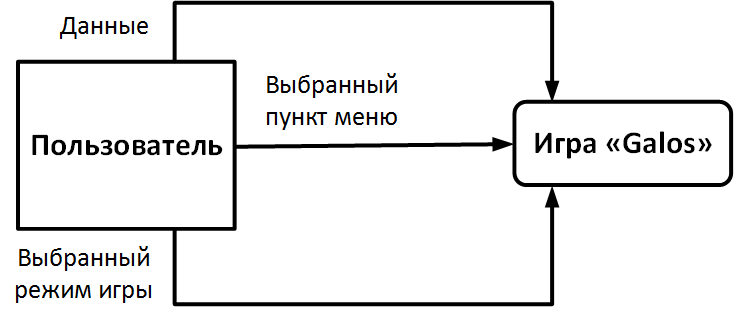


Рисунок 7. Контекстная диаграмма потоков данных

На диаграмме представлена внешняя сущность «Пользователь», связанная с системой информационными потоками.

Так как в игру можно играть с интернетом или без него, то рассмотрим две диаграммы декомпозиции контекстной диаграммы потоков данных, одна из которых будет показывать потоки данных, когда пользователь не авторизован в системе, а другая будет показывать потоки данных, когда пользователь имеет доступ в интернет.

Диаграмма декомпозиции контекстной диаграммы потоков данных конкретизирует контекстную диаграмму, выделяя важнейшие процессы системы (рис. 8). На диаграмме декомпозиции отображаются все внешние сущности и потоки данных контекстной диаграммы. Кроме того, здесь также показываются хранилища данных для моделирования данных, которые должны сохраняться в промежутках между процессами.

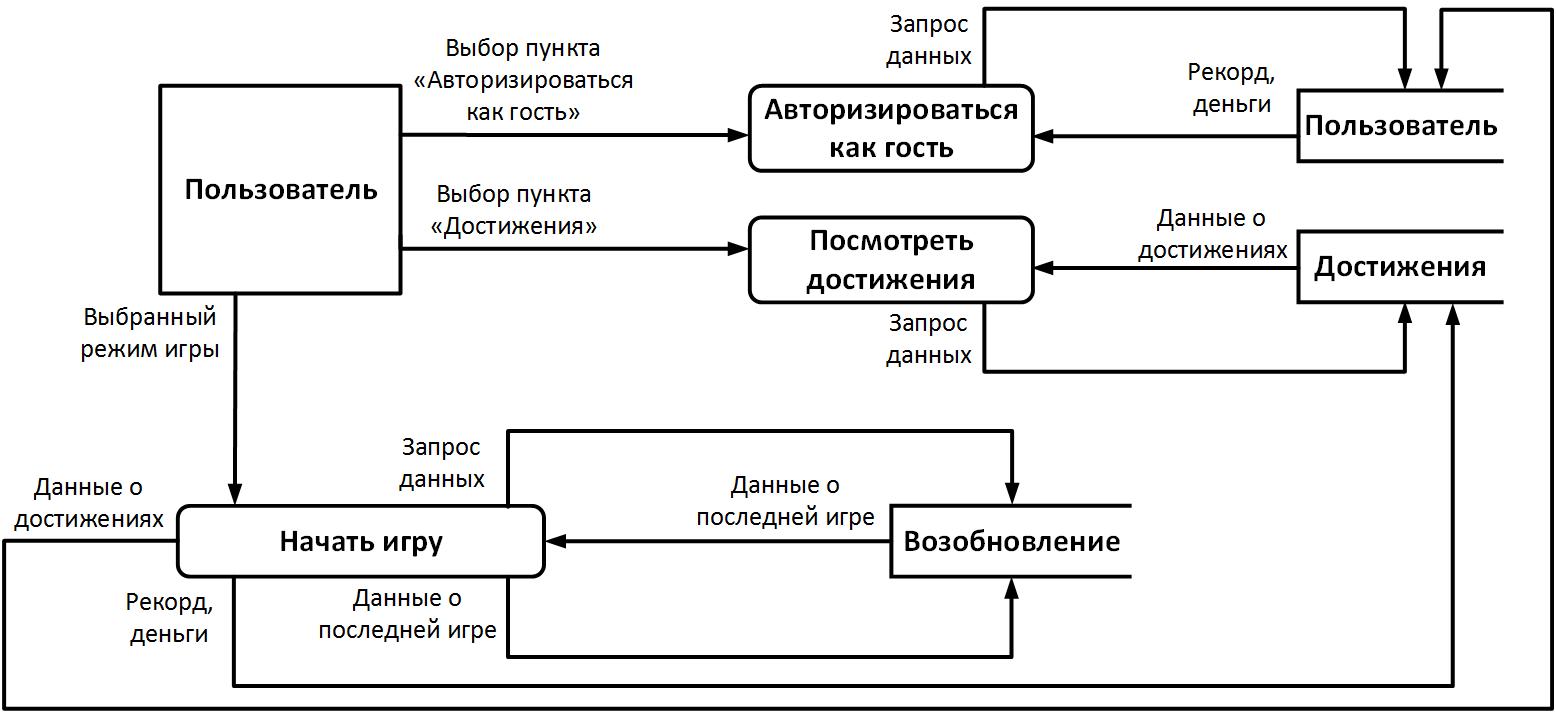


Рисунок 8. Диаграмма декомпозиции для неавторизованного пользователя

Проанализируем полученную диаграмму:

* сущность «Пользователь», выбрав в игре пункт «Авторизироваться как гость», запускает процесс, который совершает запрос данных о пользователе из хранилища «Пользователь»;
* сущность «Пользователь», выбрав в игре пункт «Достижения», запускает процесс «Посмотреть достижения», который совершает запрос данных о достижениях из хранилища «Достижения»;
* сущность «Пользователь», выбрав определенный игровой режим, запускает игровой процесс «Начать игру», который прежде всего посылает запрос на получение данных о последней игре из хранилища «Возобновление», на основе которых запускается тот или иной уровень, а также в процессе игры передаются данные о последней сыгранной игре в хранилище «Возобновление» и данные о полученных достижениях в ходе игры в хранилище «Достижения».

Диаграмма декомпозиции для пользователя, имеющего доступ в интернет, описывает дополнительные потоки данных между сущностью «Пользователь» и системой «Galos» (рис. 9). Пользователь с доступом в интернет, также имеет те потоки данных, которые описаны в диаграмме декомпозиции на рис. 8, но разница будет заключаться лишь в том, что хранилища расположены не на локальной базе данных, а на базе данных сервера.

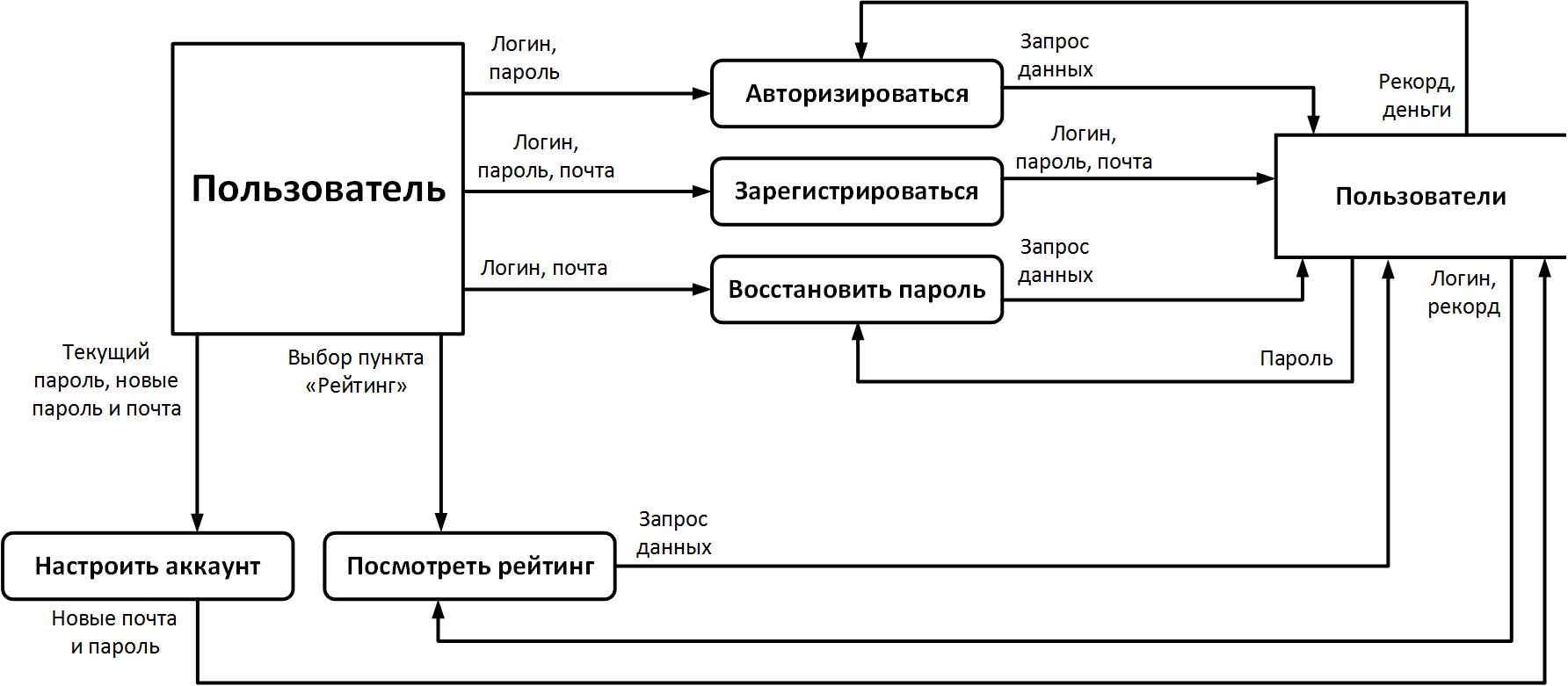


Рисунок 9. Диаграмма декомпозициидля пользователя с доступом в интернет

Проанализируем полученную диаграмму:

* сущность «Пользователь», передает логин и пароль процессу «Авторизироваться», который передает эти данные хранилищу «Пользователи», после чего с сервера приходят ответные данные в виде рекорда и денег пользователя;
* сущность «Пользователь», передает логин, пароль и почту процессу «Зарегистрироваться», который передает эти данные хранилищу «Пользователи» для создания нового аккаунта;
* сущность «Пользователь», логин, почту процессу «Восстановить пароль», который делает запрос данных хранилищу «Пользователи» для получения пароля от аккаунта пользователя;
* сущность «Пользователь», передает текущий пароль, новый пароль и почту процессу «Изменить аккаунт», который передает эти данные хранилищу «Пользователи» для изменения данных аккаунта;
* сущность «Пользователь», выбрав в игре пункт «Рейтинг», запускает процесс «Посмотреть рейтинг», который совершает запрос данных о рейтинге хранилищу «Пользователи», который в ответ присылает данные о рейтинге;

1.7 Варианты использования системы

Для того чтобы описать все функциональные возможности системы «Galos» воспользуемся диаграммой вариантов использования [5].

Диаграмма вариантов использования иллюстрирует актера системы и все варианты ее использования.

Диаграмма вариантов использования для разрабатываемой мобильной игры продемонстрирована на рис. 10.

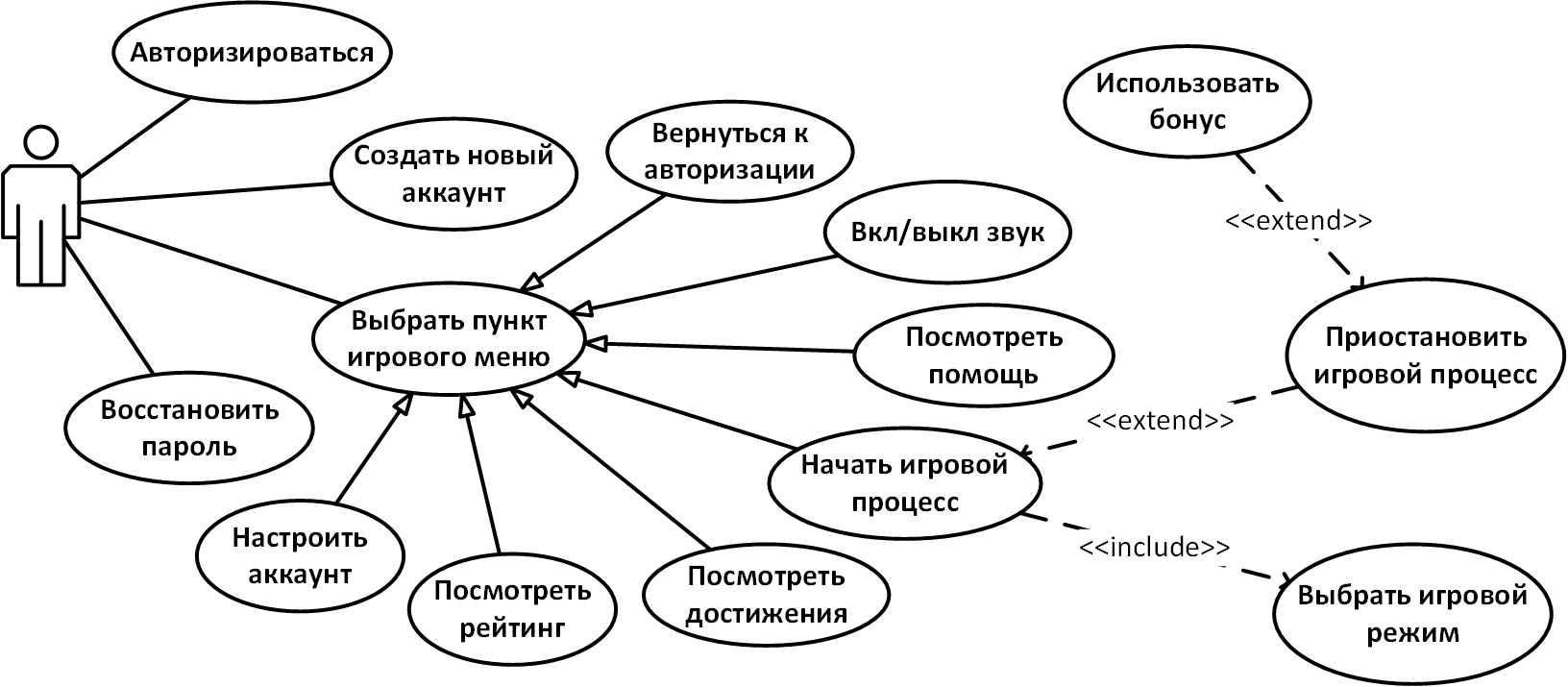


Рисунок 10. Use Case диаграмма игры «Galos»

На диаграмме представлен актер «Пользователь», который может авторизироваться в системе, создать новый аккаунт, восстановить пароль, посмотреть помощь, рейтинг, достижения, может настроить звук и аккаунт, вернуться к авторизации, выбрать игровой режим и начать игровой процесс, который можно приостановить, чтобы использовать какой-то бонус.

1.8 Спецификация требований

1.8.1 Пользовательские истории

Пользовательские истории (User story) **-** способ описания требований к разрабатываемой системе, сформулированных как одно или более предложений на ежедневном или деловом языке пользователя.

Я, как Пользователь приложения, могу просмотреть помощь для того, чтобы получить информацию по игре.

Я, как Пользователь приложения, могу зарегистрироваться в системе, если у меня нет аккаунта или восстановить пароль, если предыдущий по каким-либо причинам был утерян, чтобы в дальнейшем можно было авторизироваться под своим аккаунтом для сохранения игрового прогресса.

Я, как Пользователь приложения, могу посмотреть достижения, чтобы получить информацию о том, как их заработать, чтобы получить за них вознаграждение.

Я, как Пользователь приложения, могу включить, или выключить звук, чтобы полностью погрузиться в игровую атмосферу или наоборот, чтобы не отвлекаться на звуки или не нарушать окружающей тишины.

Я, как Пользователь приложения, могу посмотреть таблицу рекордов, чтобы знать на каком месте в списке я нахожусь и к какому рекорду нужно стремиться, чтобы стать первым.

Я, как Пользователь приложения, могу начать игру, чтобы получить положительные эмоции или приятно скоротать время.

Я, как Пользователь приложения, могу приостановить игровой процесс, чтобы передохнуть, выйти в главное меню игры или купить за игровую валюту бонусы, которые помогут мне при прохождении игрового уровня.

1.8.2 Общее описание системы

Система «Galos» - это мобильная игра жанра аркада, с интуитивно понятным интерфейсом, простым игровым процессом, небольшим размером дистрибутива и низкими требованиями к мобильному устройству, целью, которой является развлечение пользователя.

1.8.3 Функции системы

Авторизация

Программный продукт должен предоставлять возможность авторизации в системе для неавторизированного пользователя.

* Идентификация пользователей должна осуществляться посредством логина и пароля, вводимых в стартовом окне программы
* При корректном вводе авторизационных данных должен осуществляться переход в главное меню игры с игровым прогрессом данного пользователя
* При неверном вводе, появляется сообщение о просьбе проверить правильность ввода данных

Регистрация

Пользователь имеет возможность создать аккаунт в системе для того, чтобы в дальнейшем в ней авторизироваться.

* В окне авторизации можно выбрать «Регистрация»
* В окне регистрации должны быть все необходимые поля, которые должен заполнить пользователь, а именно логин, пароль, почта.
* При корректном вводе регистрационных данных, осуществляется оповещение пользователя о успешном создании аккаунта
* В случае неверного ввода, появляется сообщение о просьбе проверить правильность ввода данных

Восстановление пароля

Пользователь может восстановить пароль, если по каким-либо причинам он его утратил, чтобы он мог снова авторизироваться под своим аккаунтом.

* В окне авторизации можно выбрать «Восстановление пароля»
* В окне восстановления пароля имеются поля для ввода почты и логина, одно из которых (или оба) должен корректно заполнить пользователь
* При корректном вводе данных для восстановления пароля, появляется сообщение, с просьбой проверить почту, указанную при регистрации, на наличие письма с подтверждением о восстановлении пароля, в котором должна быть ссылка для подтверждения восстановления пароля
* Если пользователь переходит по ссылке, указанной в письме, то на его почту должно прийти письмо с паролем от его аккаунта

Просмотр рейтинга

У пользователя есть возможность посмотреть рейтинг игроков, чтобы знать свое положение в списке.

* В окне главного меню можно выбрать пункт, отвечающий за просмотр рекордов, при нажатии на который, откроется окно с рекордами
* Окно с рекордами должно отображать информацию об игроках в порядке убывания, а также позицию пользователя в этом списке

Настройка звука

У пользователя есть возможность включить или выключить звук по своему усмотрению.

* В окне главного меню можно выбрать пункт, отвечающий за включение/выключение звука, при нажатии на который, произойдет изменение иконки, которая проинформирует о состоянии звука (перечеркнутая иконка – нет звука, обычная иконка – есть звук)

Посмотреть достижения

Пользователь может посмотреть свои достижения, чтобы знать, как их достичь и какую награду можно получить за их достижение.

* В окне главного меню можно выбрать пункт, отвечающий за просмотр достижений, при нажатии на который, откроется окно с достижениями
* Окно с достижениями должно отображать информацию обо всех достижениях, информацию о том, как их заработать, статус (получено достижение или нет) и награду, которую можно получить.
* Если пользователь заработал достижение, то это достижение должно сменить статус с «Не получено» на «Получено» и автоматически начислить указанную награду на счёт пользователя.

Начать игровой процесс

Пользователь имеет возможность начать игровой процесс, для получения положительных эмоций или приятного времяпрепровождения.

* В окне главного меню можно выбрать пункт, отвечающий за начало игрового процесса, при нажатии на который, откроется окно с игровым уровнем в зависимости от того, какой игровой режим был выбран игроком
* Окно с игровым уровнем должно содержать в себе случайно сгенерированные вражеские кружки и круг самого игрока
* При изменении положения круга игрока, должен проверяться результат этих действий и происходить соответствующее изменение игрового поля
* Если круг игрока соприкоснется с врагом меньшего размера, то этот враг исчезнет с игрового поля, а круг игрока увеличится на размер равный размеру врага
* Если круг игрока соприкоснется с врагом большего размера или с неуязвимым кругом, то должно появиться сообщение, информирующее о проигрыше игрока и с возможностью выбора дальнейших действий «Выйти в главное меню» или «Начать заново»
* Если все возможные круги будут поглощены, то должно выйти сообщение, информирующее о победе игрока и с выбором дальнейших действий «Продолжить игру» или «Выйти в главное меню»

Приостановить игровой процесс

Пользователь может вызвать паузу во время игры, чтобы передохнуть, использовать бонусы или выйти в главное меню.

* Если в процессе игры пользователь нажмет на кнопку паузы, то игровой процесс приостановится и откроется окно паузы
* Окно паузы содержит кнопку «Выйти в главное меню», «Продолжить» и 3 кнопки с бонусами «Жизнь», «Замедление», «Рост» и их цена
* При нажатии на кнопку «Выйти в главное меню» пользователь перенаправляется в главное меню приложения
* При нажатии на кнопку «Продолжить» происходит возобновление игрового процесса, а окно с паузой закрывается
* При нажатии на одну из кнопок с бонусами, то с игрового счёта пользователя должна сняться соответствующая денежная сумма, а в игре должен появиться соответствующий эффект от бонуса
* При выборе бонуса должно появиться сообщение о том, успешно ли произведена покупка этого бонуса
* При выборе бонуса, который уже активирован, нужно оповестить пользователя об этом

Посмотреть помощь

Пользователь может посмотреть помощь, чтобы получить информацию об игре.

* В окне главного меню нужно выбрать пункт «Помощь», при нажатии на который, откроется окно с информацией по игре
* В помощи должна быть информация о целях игры, управлении, о игровых режимах и бонусах.
* В окне помощи должна быть кнопка «Вернуться», которая возвращает пользователя в главное меню игры

Настроить аккаунт

Авторизованный пользователь может посмотреть настроить аккаунт чтобы изменить пароль или почту.

* В окне главного меню нужно выбрать пункт «Настроить аккаунт», при нажатии на который, откроется окно с изменением аккаунта
* Окно должно содержать 3 поля для ввода текущего пароля, нового пароля и новой почты
* При вводе корректных данных, пользователь должен быть оповещен об успешном изменении данных
* В случае ввода пользователем некорректных данных пользователь должен быть об этом оповещен

Вернуться к авторизации

Авторизованный пользователь может вернуться к окну авторизации, чтобы войти под другим аккаунтом, создать новый аккаунт, восстановить пароль, войти как гость или чтобы просто выйти из текущего аккаунта.

* В окне главного меню нужно выбрать пункт «Вернуться к окну авторизации», при нажатии на который, откроется начальное окно авторизации, которое появляется при входе в игру

1.8.4 Нефункциональные требования

* Производительность. Несмотря на постоянное увеличение информации на сервере, время отклика системы при получении информации из базы данных не должно превышать 3 секунд.
* Надежность. При каждой транзакции в базе данных должна производиться проверка целостности данных: отсутствие недействительных ссылок, ключей, противоречивого содержимого полей.
* Стабильность. Восстановление мобильного приложения после сбоя не должно вызвать затруднений у его пользователя. Восстановление серверного компонента системы не должно превышать по времени один рабочий день.
* Удобство использования. Пользователь, никогда ранее не пользовавшийся системой должен освоить ее не более чем за 5 минут. Интуитивный интерфейс позволяет быстро разобраться в управлении системой.
* Ограничения:
  + Данная система реализуется с помощью интегрированной среды разработки Android Studio для работы с платформой Androidна языке Java.
  + Для реализации базы данных используется MySQL версии 5.0 или выше.
  + Необходимо соединение с интернетом со скоростью не менее 60 кбит/с.
* Системные требования:
  + Операционные системы: Android 4.1 JellyBeanи выше.
  + Для установки системы на мобильное устройство требуется не менее 20 Мб свободного места на карте памяти.
  + Видеочип не слабее Mali-400MP, Adreno 320, PowerVR SGX544, Tegra 3.
  + Минимальная частота процессора: 1000 МГц
  + Минимальное количество ядер процессора: 2
  + Минимальный объем оперативной памяти: 500 МБ

1.9 Концептуальная модель предметной области

Концептуальная модель для системы «Galos» представлена на рис. 11.

В системе есть 17 сущностей. Авторизованные пользователи «Игрок» имеют свои достижения, которых всего 26 («Достижение») и свою информацию о последних играх в разных игровых режимах «Возобновление». Также каждый «Игрок» находится в рейтинге игроков, за который отвечает сущность «Рейтинг». «Игрок» может выбрать один из пяти игровых режимов («Игра»), которые содержат уровни («Уровень»), проходя которые пользователь набирает счёт («Рекорд») и получает награду («Награда»). Также уровни состоят из случайно сгенерированных кругов («Вражеский круг», «Неуязвимый круг», «Исчезающий круг»), одним из которых является круг игрока («Круг игрока»), а также в процессе прохождения уровня можно использовать бонусы («Бонус»), которых в игре всего 3: «Жизнь», «Замедление» и «Рост». Все круги наследуют характеристики и методы от базового круга «Базовый круг».

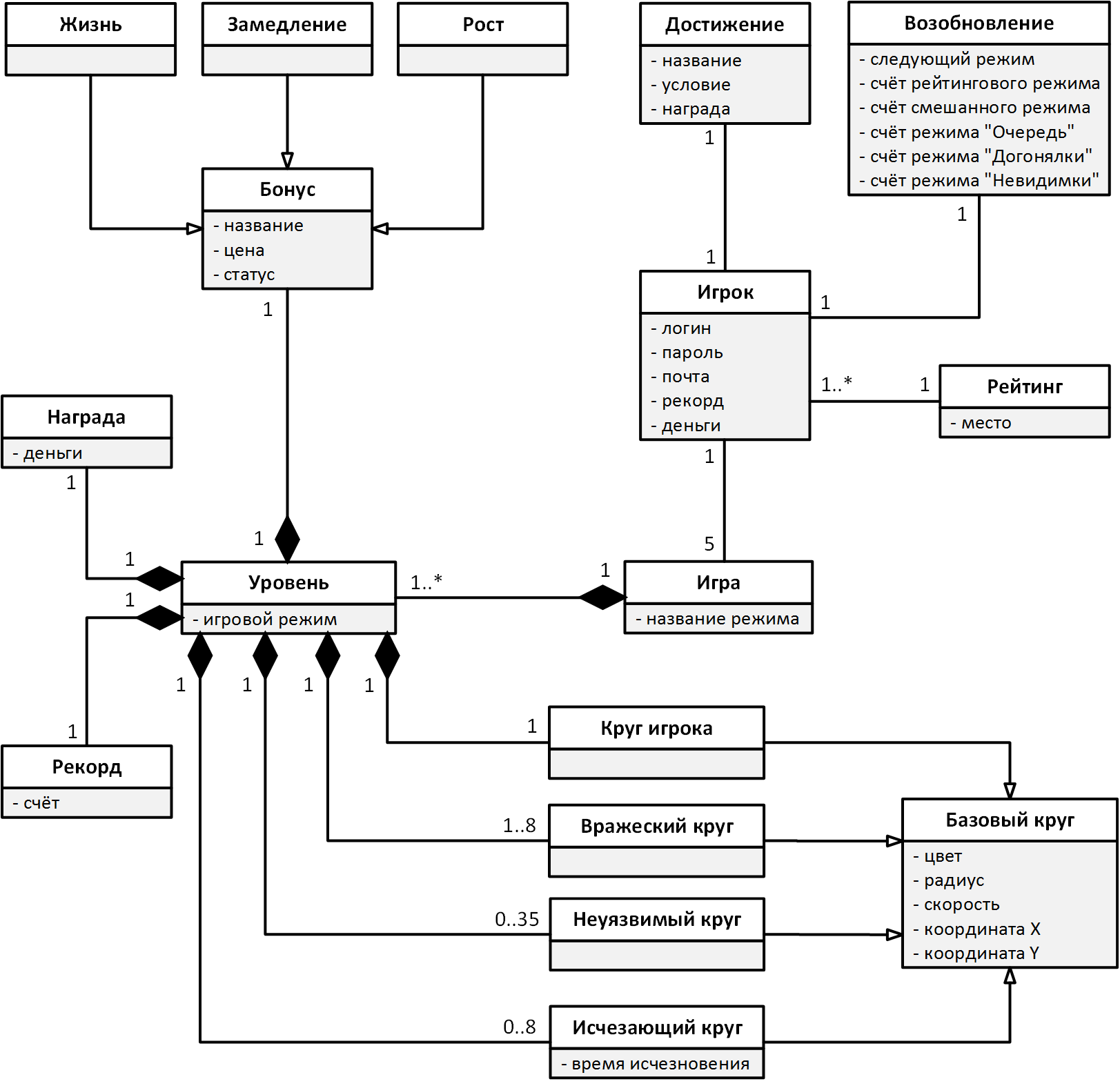


Рисунок 11. Концептуальная модель

#### Проектирование

2.1 Вид приложения

Мобильное приложение для операционной системы Android 5.0 lolipop и ниже.

2.2 Архитектура приложения

Приложение использует встраиваемую СУБД SQLite, предназначенную для хранения игровых данных пользователя и возможности игры без авторизации, а также использует 3-ех уровневую клиент-серверную архитектуру (рис. 12) для возможности отображения рейтинга игроков и хранения пользовательских данных в БД сервера.

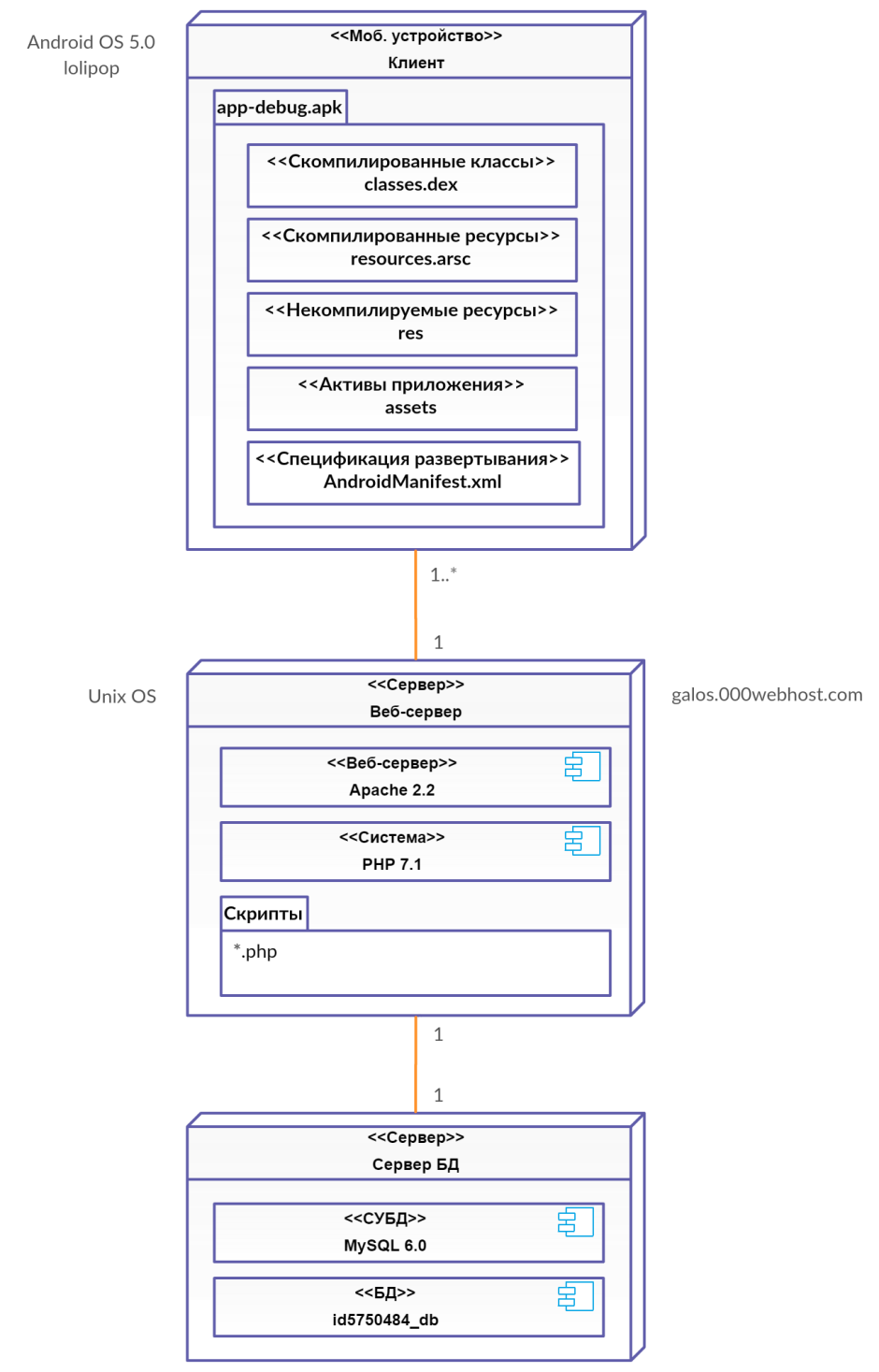


Рисунок 12. Диаграмма развертывания приложения

Содержимым папки «Скрипты» являются файлы с исходными текстами, написанными на языке PHP7.1, каждый из которых реализует в себе тот или иной метод для взаимодействия с серверной БД.

Компонентами исходных текстов являются:

1. create\_user.php – реализует регистрацию пользователя;
2. db\_config.php – содержит конфигурацию БД;
3. get\_records.php – реализует получение данных о рейтинге;
4. get\_user\_data.php – реализует получение всех данных пользователя;
5. restore\_password.php – реализует процесс отправки письма с новым паролем;
6. update\_data.php – обновление данных пользователя;
7. update\_resume.php – обновление данных о последней игре пользователя;
8. update\_user.php – обновление данных аккаунта пользователя.

Таким образом, приложение разворачивается на мобильном устройстве под управлением операционной системы Android, которая в свою очередь предоставляет библиотеку SQLite для работы с локальной базой данных. Также приложение связано с сервером, который получает от приложения запросы на изменение/добавление пользовательских данных в БД сервера.

2.3 Архитектурный шаблон

Функциональное деление компонентов приложений Android по слоям выполняется в соответствии со структурным паттерном MVC (Model-View-Controller, см. рис. 13). Согласно канонам MVC, каждый объект приложения должен быть объектом модели, объектом представления или объектом контроллера [3].

Моделью (Model) в терминологии MVC является уровень бизнес-логики, ответственный за хранение и обработку информации. Данный слой не должен делать никаких предположений относительно представления информации или взаимодействия пользователя с приложением. В приложениях Android классы моделей обычно создаются разработчиком для конкретной задачи.

Представление (View) – это уровень, ответственный за отображение данных модели. Классы, принадлежащие данному уровню, как правило, являются виджетами платформы либо унаследованы от них. Вид может обращаться к модели для получения данных, однако модель не должна непосредственно зависеть от вида.

Контроллер (Controller) – это уровень, ответственный за взаимодействие с пользователем. В зависимости от конкретной реализации MVC уровни вида и контроллера могут быть отдельными классами или совмещены. Контроллер, как правило, зависит от модели, однако, как и в случае вида, модель не должна хранить ссылок на конкретные классы контроллера. В Android контроллер обычно представляется субклассом Activity, Fragment или Service.



Рисунок 13. Архитектурный шаблон MVC

2.4 Информационное обеспечение

Компонентом информационного обеспечения является встраиваемая БД «galos.db». Реализация текущей версии приложения предполагает, что БД существует заранее и заполняется данными в процессе игры. Кроме локальной БД, существует БД, которая находится на сервере и также заполняется данными в процессе игры [4].

**2.4.1 Логическая модель БД**

На основе анализа концептуальной модели предметной области, была составлена логическая модель локальной и серверной реляционных баз данных (см. рис. 14-15).

Таким образом, в базе данных будет иметь место набор следующих таблиц:

* таблица «Пользователи» - предназначена для хранения основных данных аккаунта пользователя;
* таблица «Возобновление» - предназначена для хранения игрового режима и счёта разных игровых режимов, чтобы можно было возобновить игру без потери счёта;
* таблица «Достижения» - предназначена для хранения данных, необходимых для проверки выполнения достижений;
* таблица «Запоминание» - предназначена для хранения логина и пароля последнего вошедшего пользователя, для автоматического входа в игру;

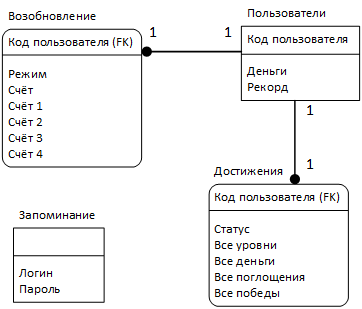


Рисунок 14. Логическая модель локальной базы данных

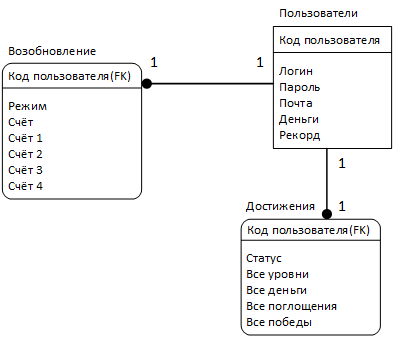


Рисунок 15. Логическая модель базы данных на сервере

2.4.2 Физическая модель БД

Рисунки 16-17 представляют физические модели баз данных, полученных из логических моделей, изображенных на рис. 14-15.

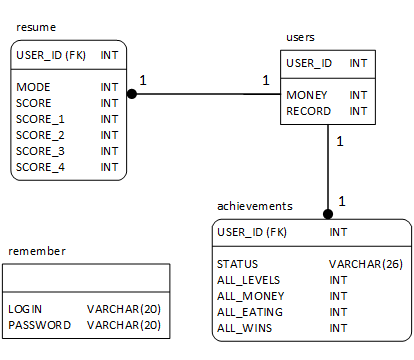


Рисунок 16. Физическая модель локальной базы данных

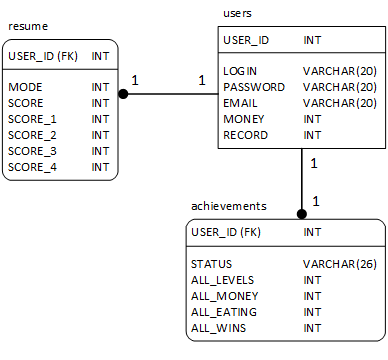


Рисунок 17. Физическая модель базы данных на сервере

Модель определяет физические характеристики (индексы) и использует типы данных конкретной СУБД: SQLite для локальной и MySQL для серверной.

2.4.3 SQL скрипты для создания таблиц БД

Скрипты для создания таблиц БД:

1. Локальная БД:

CREATE TABLE IF NOT EXISTS `users` (

`\_id` INTEGER NOT NULL PRIMARY KEY,

`money` INTEGER NOT NULL DEFAULT 0,

`record` INTEGER NOT NULL DEFAULT 0

);

CREATE TABLE IF NOT EXISTS `resume` (

`mode` INTEGER NOT NULL DEFAULT 1,

`score` INTEGER NOT NULL DEFAULT 0,

`score\_1` INTEGER NOT NULL DEFAULT 0,

`score\_2` INTEGER NOT NULL DEFAULT 0,

`score\_3` INTEGER NOT NULL DEFAULT 0,

`score\_4` INTEGER NOT NULL DEFAULT 0,

`id\_user` INTEGER,

FOREIGN KEY(`id\_user`) REFERENCES `users`(`\_id`) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS `remember` (

`login` VARCHAR(20),

`password` VARCHAR(20)

);

CREATE TABLE IF NOT EXISTS `achievements` (

`status` TEXT NOT NULL DEFAULT '00000000000000000000000000',

`all\_levels` INTEGER NOT NULL DEFAULT 0,

`all\_money` INTEGER NOT NULL DEFAULT 0,

`all\_eating` INTEGER NOT NULL DEFAULT 0,

`all\_wins` INTEGER NOT NULL DEFAULT 0,

`id\_user` INTEGER,

FOREIGN KEY(`id\_user`) REFERENCES `users`(`\_id`) ON DELETE CASCADE

);

1. СервернаяБД:

CREATE TABLE `achievements` (

`\_id` int(11) NOT NULL,

`status` varchar(26) COLLATE utf8\_unicode\_ci NOT NULL DEFAULT '00000000000000000000000000',

`all\_levels` int(11) NOT NULL DEFAULT '0',

`all\_money` int(11) NOT NULL DEFAULT '0',

`all\_eating` int(11) NOT NULL DEFAULT '0',

`all\_wins` int(11) NOT NULL DEFAULT '0',

`id\_user` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE `resume` (

`\_id` int(11) NOT NULL,

`mode` int(11) NOT NULL DEFAULT '1',

`score` int(11) NOT NULL DEFAULT '0',

`score\_1` int(11) DEFAULT '0',

`score\_2` int(11) DEFAULT '0',

`score\_3` int(11) DEFAULT '0',

`score\_4` int(11) DEFAULT '0',

`id\_user` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE `users` (

`\_id` int(11) NOT NULL,

`login` varchar(20) COLLATE utf8\_unicode\_ci NOT NULL,

`password` varchar(20) COLLATE utf8\_unicode\_ci NOT NULL,

`email` varchar(20) COLLATE utf8\_unicode\_ci NOT NULL,

`money` int(11) NOT NULL DEFAULT '0',

`record` int(11) NOT NULL DEFAULT '0'

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

ALTER TABLE `achievements`

ADD PRIMARY KEY (`\_id`),

ADD UNIQUE KEY `\_id\_UNIQUE` (`\_id`),

ADD KEY `id\_user\_idx` (`id\_user`);

ALTER TABLE `resume`

ADD PRIMARY KEY (`\_id`),

ADD UNIQUE KEY `\_id\_UNIQUE` (`\_id`),

ADD KEY `id\_user\_idx` (`id\_user`);

ALTER TABLE `users`

ADD PRIMARY KEY (`\_id`),

ADD UNIQUE KEY `idtable1\_UNIQUE` (`\_id`),

ADD UNIQUE KEY `login\_UNIQUE` (`login`),

ADD UNIQUE KEY `email\_UNIQUE` (`email`);

ALTER TABLE `achievements`

MODIFY `\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=34;

ALTER TABLE `resume`

MODIFY `\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=34;

ALTER TABLE `users`

MODIFY `\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=74;

ALTER TABLE `achievements`

ADD CONSTRAINT `achievementsUserFK` FOREIGN KEY (`id\_user`) REFERENCES `users` (`\_id`) ON DELETE CASCADE ON UPDATE NO ACTION;

ALTER TABLE `resume`

ADD CONSTRAINT `resumeUserFK` FOREIGN KEY (`id\_user`) REFERENCES `users` (`\_id`) ON DELETE CASCADE ON UPDATE NO ACTION;

2.5 Решения по программному обеспечению

На данном этапе работы решения технического проектирования представлены проектом классов, диаграммами взаимодействия программных модулей и проектом пользовательского интерфейса.

2.5.1 Диаграмма классов

Диаграмма классов для приложения «Galos» представлена на рис. 18 и состоит из 20 классов с расширением java: Registration, RestorePassword, Authorization, Sound, Menu, CanvasView, Game, GameManager, GameModes, SimpleCircle, MainCircle, EnemyCircle, ImmortalCircle, VanishingCircle, Achievements, Rating, Help, EditAccInf, JSONParser, DatabaseHelper [1].

В зависимости от того к какому слою относится класс, он окрашен в определенный цвет:

1. Model (красный цвет) – к нему относятся классы, которые предоставляют данные или содержат методы для взаимодействия с БД;
2. Controller (синий цвет) – к нему относятся классы, которые отвечают за логику приложения;
3. View + Controller (жёлтый цвет) – так как в Android Studio такой компонент как Activity, объединяет в себе представление в виде xmlфайла и контроллер в виде javaкласса, то отдельных классов, которые бы отвечали за представление в приложении нет.

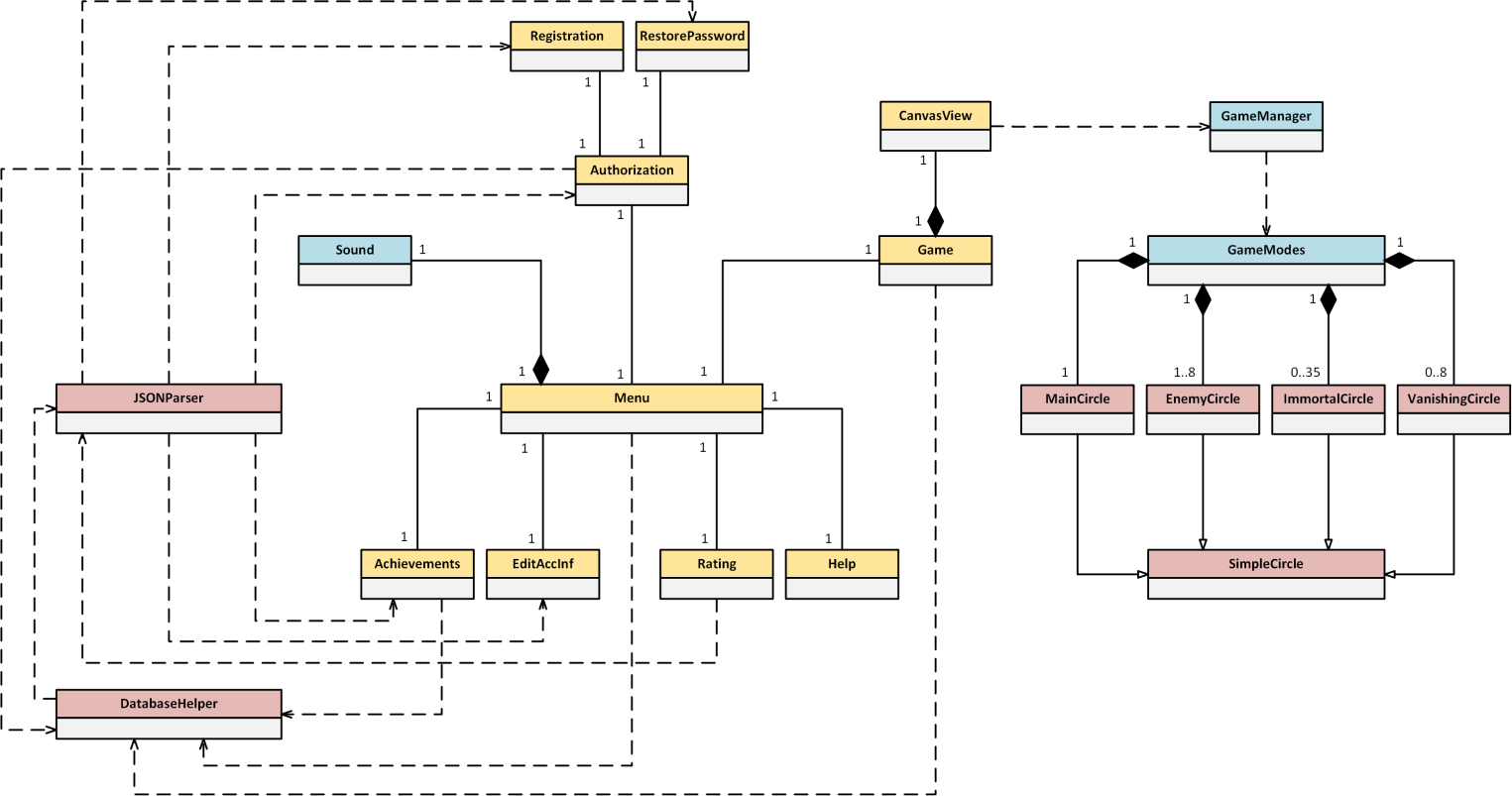


Рисунок 18. Диаграмма классов приложения

Для избегания сложного восприятия диаграммы классов, было решено выделить атрибуты каждого из классов в отдельную таблицу. В таблице 1 не только перечислены атрибуты классов, но также дано краткое пояснение их функциональной роли.

Таблица 1

Атрибуты классов приложения

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
| *Achievements.java* | - *backButton* :button (кнопка возврата)  - *listView* :ListView (лист с достижениями)  - *progressTextView* :TextView (надпись с количеством выполненных достижений)  - *achievementsList*:ArrayList<HashMap<String, Object>> (массив с данными о достижениях)  - *adapter* :SimpleAdapter (адаптер для управления листом достижений) | # *onCreate*(Bundles) :void (стандартный метод Activity для инициализации объектов)  + *onBackPressed*() :void (переопределение действий кнопки «Back» смартфона) |
| *Authorization.java* | - *loginAsGuestButton* :Button (кнопка авторизации гостем)  - *registrationButton* :Button (кнопка регистрации)  - *restorePasswordButton* :Button (кнопка восстановления пароля)  - *loginButton* :Button (кнопка авторизации)  - *loginEditText* :EditText (поле ввода логина)  - *passwordEditText* :EditText (поле ввода пароля)  - *rememberCheckBox* :CheckBox (флажок запоминания данных авторизации)  - *progress* :LinearLayout (разметка с прогрессом загрузки)  - *intent* :Intent (переменная для взаимодействия с отдельными частями приложения) | # *onCreate*(Bundle s) :void |
| *EditAccInf.java* | - *newEmailEditText* :EditText (поле ввода новой почты)  -*newPasswordEditText* :EditText (полевводановогопароля)  - *currentPasswordEditText* :EditText (поле ввода текущего пароля)  - *backButton* :Button  - *acceptButton* :Button (кнопка подтверждения) | # *onCreate*(Bundle s) :void  + *onBackPressed*() :void |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *progress* :LinearLayout  - *intent* :Intent |  |
| *Game.java* | + *PAUSE* :boolean (переменная с состоянием true, если в игре стоит пауза и false, если паузы нет)  - *intent* :Intent  - *timer* :Timer (таймер появления/исчезновения исчезающих кругов)  - *dialog* :boolean (переменная с состоянием true, если открыт диалог с результатом игры и false, если диалога нет)  - *PRICE*\_*LIFE* :int (стоимость бонуса с жизнью)  - *PRICE*\_*DECELERATION* :int (стоимость бонуса с замедлением)  - *PRICE*\_*GROWTH* :int (стоимость бонуса с ростом)  -*dialogConstraintLayout* :ConstraintLayout (диалог с результатом игры)  - *continueButton* :Button  - *menuButton* :Button (кнопка возврата в главное меню)  - *countScoreTextView* :TextView (поле с информацией о счёте)  - *titleTextView* :TextView (поле с информацией победы или поражения игрока)  - *countRewardTextView* :TextView (поле с общей наградой за игру)  - *rewardTextView* :TextView (поле с наградой за уровень)  - *pauseMenuConstraintLayout*:ConstraintLayout (менюпаузы)  - *exitButton* :Button (кнопка с возвратом в главное меню)  - *pauseImageButton* :ImageButton (кнопка вызова меню паузы)  - *countMoneyTextView* :TextView (поле с количеством денег игрока в меню паузы) | # *onCreate*(Bundle s) :void  + *showDialog*(int score, int reward, int sum, int flag) :void (вызов диалога с результатом по окончанию игры)  - *getData*() :void (получение данных из DatabaseHelper.java)  - *onBackPressed* :void |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *lifeBonusImageButton* :ImageButton (кнопка бонуса жизни)  - *decelerationBonusImageButton* :ImageButton (кнопка бонуса замедления)  - *growthBonusImageButton* :ImageButton (кнопка бонуса роста)  - *priceLifeBonusTextView* :TextView (поле с ценой бонуса жизни)  - *priceDecelerationBonusTextView* :TextView (поле с ценой бонуса замедления)  - *priceGrowthBonusTextView* :TextView (поле с ценой бонуса роста)  - *money* :int (количество денег игрока)  - *record* :int (рекорд игрока)  - *all\_levels* :int (общее количество пройденных уровней)  - *all\_money* :int (общая награда за игру)  - *all\_eating* :int (все поглощения за игру)  - *all\_wins* :int (все победы за игру)  - *upAnimation* :Animation (анимация выхода виджета снизу вверх)  - *downAnimation* :Animation (анимация ухода виджета сверху вниз)  - *alphaAnimation* :Animation (анимация появления кругов на игроком поле) |  |
| *Help.java* | - *intent* :Intent  - *backButoon* :Button | # *onCreate*(Bundle s) :void  + *onBackPressed*() :void |
| *Menu.java* | - *volumeImageButton* :ImageButton (кнопка вкл./выкл. музыки)  - *blackImageButton* :ImageButton (кнопка рейтинговой игры) | # *onCreate*(Bundle s) :void  + *onBackPressed*() :void |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *logoutImageButton* :ImageButton (кнопка выхода к окну авторизации)  - *settingsImageButton* :ImageButton (кнопка настроек аккаунта)  - *ratingImageButton* :ImageButton (кнопка рейтинга)  - *achievementsImageButton* :ImageButton (кнопка достижений)  - *helpImageButton* :ImageButton (кнопка помощи)  - *darkGreyImageButton* :ImageButton (кнопка смешанной игры)  - *lightGreyImageButton* :ImageButton (кнопка игры «Очередь»)  - *redImageButton* :ImageButton (кнопка игры «Догонялки»)  -*blueImageButton* :ImageButton (кнопка игры «Невидимки»)  - *loginTextView* :TextView (поле с логином игрока)  - *countMoneyTextView* :TextView (поле с деньгами игрока)  - *registrationButton* :Button (кнопка регистрации для неавторизированного игрока)  - *laterButton*: Button (кнопка отмены регистрации для неавторизированного игрока)  -*registration* :LinearLayout (диалог с просьбой зарегистрироваться для неавторизированного игрока)  - *intent* :Intent  - *upAnimation* :Animation  - *downAmination* :Animation |  |
| *Rating.java* | - *intent* :Intent  - *context* :Context (контексттекущейActivity)  - *backButton* :Button | # *onCreate*(Bundle s) :void  + *onBackPressed*() :void |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *listView* :ListView (листсрейтингом)  - *goldUserTextView* :TextView (поле с логином пользователя, который в рейтинге на 1-ом месте)  - *silverUserTextView* :TextView (поле с логином пользователя, который в рейтинге на 2-ом месте)  - *bronzeUserTextView* :TextView (поле с логином пользователя, который в рейтинге на 3-ем месте)  - *positionTextView* :TextView (поле с местом в рейтинге игрока)  - *progress* :LinearLayout |  |
| *Registration.java* | - *acceptButton* :Button  - *backButton* :Button  - *loginEditText* :EditText  - *passwordEditText* :EditText  - *confirmPasswordEditText* :EditText (поле с подтверждением пароля)  - *emailEditText* :EditText (поле с вводом почты)  - *progress* :LinearLayout  - intent :Intent  - *flag* :boolean (содержит false, если создается новый аккаунт и true, если переносятся данные из гостя в новый аккаунт) | # *onCreate*(Bundle s) :void  + *onBackPressed*() :void |
| *RestorePassword.java* | - *acceptButton* :Button  - *backButton* :Button  -*enterLoginEditText* :EditText (поле с вводом логина)  - *enterEmailEditText* :EditText (поле с вводом почты) | # *onCreate*(Bundle s) :void  + *onBackPressed*() :void |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *progress* :LinearLayout  - *intent* :Intent  - *flag* :int (содержит 1, если восстановление пароля по почте, содержит 2, если восстановление пароля по логину, содержит 3, если восстановление пароля по логину и почте) |  |
| *CanvasView.java* | - *width* :int (ширина экрана)  - *height* :int (высота экрана)  - *gameManager* :GameManager (экземпляр класса GameManager)  - *paint* :Paint (для рисования кругов)  - *canvas* :Canvas (для отображения кругов)  - *x* :int (координата Х)  - *y* :int (координата Y) | - *initPaint*() :void (инициализация инструмента рисования paint)  - *initWidthAndHeight*(Context context) :void(инициализация размеров экрана и центрирование объектов на нём)  # *onDraw*(Canvas canvas) :void (рисованиекругов)  + *drawCircle*(SimpleCircle circle) :void (рисование круга)  + *redraw*() :void (перерисовка игрового поля)  + *onTouchEvent*(MotionEvent event) :boolean (обработка нажатия на экран)  + *performClick*() :boolean (постоянное нажатие на экран) |
| *DatabaseHelper.java* | - *toast* :Toast (вывод уведомления)  -*databaseHelper* :DatabaseHelper (управление БД)  - *db* :SQLiteDatabase (БД)  - *cursor* :Cursor (перемещение между записями БД) | - *create*\_db() :void (создание БД)  - *open*() :SQLiteDatabase (открытие БД)  +*setMyContext*(Contextcontext) :void (установка контекста для вывода уведомлений) |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *myContext* :Context (контекст текущей Activity)  - *DB\_PATH* :String (полный путь к БД)  - *DB*\_*NAME* :String (имя БД) - *SCHEMA* :int (версия БД)  - *COLUMN*\_*ID* :String (столбец «\_id» в таблицах БД)  - *TABLE\_USERS* :String (название таблицы «users»)  - *COLUMN*\_*LOGIN* :String (название столбца таблицы «users»)  - *COLUMN\_ PASSWORD* :String (название столбца таблицы «users»)  - *COLUMN\_EMAIL* :String (название столбца таблицы «users»)  - *COLUMN\_MONEY* :String (название столбца таблицы «users»)  - *COLUMN\_RECORD* :String (название столбца таблицы «users»)  - *TABLE\_ACHIEVEMENTS* :String (название таблицы «achievements»)  - *COLUMN\_STATUS* :String (название столбца таблицы «achievements»)  -*COLUMN\_ALL\_LEVELS* :String (название столбца таблицы «achievements»)  - *COLUMN\_ALL\_MONEY* :String (название столбца таблицы «achievements»)  - *COLUMN\_ALL\_EATING* :String (название столбца таблицы «achievements»)  - *COLUMN\_ALL\_WINS* :String (название столбца таблицы «achievements») | + *showInformation*(String text, int y) :void (отображение уведомления при покупке бонусов)  - *showInformation*(String text) :void (отображение уведомления)  - *showAchievements*(String text) :void (отображение уведомления при получении достижения)  - *getGuestData*() :void (получение данных гостя)  - *getAchievementsGuest* :void (получение достижений гостя)  - *getResumeGuest* :void (получение возобновления гостя)  + *zeroGuest*() :void (обнуление всех данных гостя)  - *zeroData*() :void (обнуление данных гостя внутри приложения)  + *getUserData*() :void (получение данных пользователя)  + *getAchievementsUser*() :void (получение достижений пользователя)  + *getResumeUser*() :void (получение возобновления пользователя)  + *rememberOrForgetUser*(String login, String password) :void (запоминание/забывание данных авторизации пользователя)  + *searchRemember*() :boolean (поиск запомненных данных авторизации) |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *COLUMN\_ID\_USER* :String (название столбца таблицы «achievements»)  - *TABLE\_RESUME* :String (название таблицы «resume»)  - *COLUMN\_MODE* :String (название столбца таблицы « resume»)  - *COLUMN\_SCORE* :String (название столбца таблицы « resume»)  - *COLUMN\_SCORE*\_*1* :String (название столбца таблицы « resume»)  - *COLUMN\_SCORE\_2* :String (название столбца таблицы « resume»)  - *COLUMN\_SCORE\_3* :String (название столбца таблицы « resume»)  - *COLUMN\_SCORE\_4* :String (название столбца таблицы « resume»)  - *TABLE\_REMEMBER* (название таблицы «remember»)  - *titleAchievements* :String[] (названия достижений)  - *descriptionAchievements* :String[] (описание достижений)  - *rewardsAchievements* :String[] (награда за выполнение достижений)  - *id* :int (idпользователя)  - *login* :String (логин пользователя)  - *password* :String (пароль пользователя)  - *email* :String (почта пользователя)  - *money* :String (деньги пользователя)  - *record* :int (рекорд пользователя)  - *status* :String (статус достижений) | + *getAchievements*() :ArrayList<HashMap<String, Object>> (получение списка достижений)  - *updateStatus*(int position) :void (обновление статуса достижения)  - *checkRecord*() :void (проверка получения достижения)  - *checkAllLevels*() :void (проверка получения достижения)  - *checkAllEating*() :void (проверка получения достижения)  - *checkAllMoney*() :void (проверка получения достижения)  - *checkMoney*() :void (проверка получения достижения)  - *checkAllWins*() :void (проверка получения достижения)  - *checkGod*() :void (проверка получения достижения)  - *checkAchievements*() :void (проверка получения достижения)  + *updateData*(int money, int record, int all\_levels, int all\_money, int all\_eating, int\_all\_wins) :void (обновление всех данных)  + *updateResume*(int score) :void (обновление возобновления)  + *isOnline*() :void (проверка подключения пользователя к интернету)  + *getId*() :int (получение id)  + *getLogin*() :String (получение логина) |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *all\_levels* :int (все пройденные уровни пользователя)  - *all*\_*money* :int (все деньги пользователя)  - *all*\_*eating* :int (все поедания пользователя)  - *all\_wins* :int (все победы пользователя)  - *mode* :int (режим игры)  - *score* :int (счёт рейтинговой игры)  - *score*\_*1* :int (счёт смешанной игры)  - *score*\_*2* :int (счёт игры «Очередь»)  - *score*\_*3* :int (счёт игры «Догонялки»)  - *score*\_*4* :int (счёт игры «Невидимки») | + *getPassword*() :String(получение пароля)  + *getEmail*() :String (получение почты)  + *getRecord*() :int (получение рекорда)  + *getMoney*() :int (получение денег)  + *getMode*() :int (получение режима игры)  + *getScore*() :int (получение счёта рейтинговой игры)  + *getScore*\_1() :int (получение счёта смешанной игры)  + *getScore*\_2() :int (получение счёта игры «Очередь»)  + *getScore*\_3() :int (получение счёта игры «Догонялки»)  + *getScore*\_4() :int (получение счёта игры «Невидимки»)  + *getStatus*() :String (получение статуса достижений)  + *setId*() :void (установкаid)  + *setLogin*() :void (установка логина)  + *setPassword*() :void (установка пароля)  + *setEmail*() :void (установка почты)  + *setMoney*() :void (установка денег) |
| *EnemyCircle.java* | - *FROM*\_*RADIUS* :int (начальный радиус)  - *TO*\_*RADIUS* :int (конечный радиус)  - *ENEMY\_COLOR* :int (цвет вражеского круга) | + *getRandomCircle*() :EnemyCircle :void (создание круга со случайным радиусом)  + *getRandomCircle*(int radius) :void (создание круга определенного радиуса) |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *FOOD\_COLOR* :int (цвет круга-еды)  - *RANDOM\_SPEED* :int (скорость)  - *dx* :float (приращение по Х)  - *dy* :float (приращение по Y) | + *setEnemyOrFoodColor*(MainCircle mainCircle) :void (установка цвета кругу)  + *isSmallerThan*(SimpleCircle circle) :boolean (сравнение радиусов)  + *moveOneStep*() :void (перемещение на 1 шаг)  -*checkBounds*() (проверка захода за пределы экрана)  +*decelerationSpeed*() (уменьшение скорости в 2 раза) |
| *ImmortalCircle.java* | *- FROM\_RADIUS* :int  - *TO\_RADIUS* :int  - *CIRCLE\_COLOR* :int (цвет круга)  - *RANDOM\_SPEED* :int  - *dx* :int  - *dy* :int | + *getRandomCircle*() :ImmortalCircle (создание круга со случайным радиусом)  + *moveOneStep*() :void  -*checkBounds*() :void  + *decelerationSpeed*(): void |
| *SimpleCircle.java* | # *x* :int (координата Х)  # *y* :int (координата Y)  # *radius* :int (радиус)  -*color* :int (цвет) | + *getX*() :int (получение координаты Х)  + *getY*() :int (получение координаты Y)  + *getRadius*() :int (получение радиуса)  + *getColor*() :int (получение цвета)  + *setColor*(int color) :void (установкацвета)  + *getCircleArea*() :SimpleCircle (получение области круга)  + *isIntersect*(SimpleCircle circle) :boolean (сравнение областей круга, чтобы при инициализации на игровом поле они не соприкасались) |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
| *VanishingCircle.java* | - *FROM*\_*RADIUS* :int  - *TO\_RADIUS* :int  - *ENEMY\_COLOR* :int  - *FOOD\_COLOR* :int  - *VANISH\_COLOR* :int (цвет исчезнувшего круга)  - *RANDOM\_SPEED* :int  - *dx* :int  - *dy* :int | + *getRandomCircle*() :VanishingCircle (создание круга со случайным радиусом)  + *getRandomCircle*(int radius) :VanishingCircle (создание круга с определенным радиусом)  + *setEnemyOrFoodColor*(MainCircle mainCircle) :void  +*goVanish*() :void (исчезновение круга)  + *isSmallerThan*(SimpleCircle circle) :boolean  + *moveOnStep*() :void  - *checkBounds*() :void  + *decelerationSpeed*() :void |
| *Sound.java* | - *mediaPlayer* :MediaPlayer (управление медиа-плеером)  - *volume* :boolean (переменная содержит true, если звук есть иfalse, если звука нет) | + *onCreate*() :void (инициализация медиа-плеера)  + *onDestroy*() :void (остановка медиа-плеера)  + *onStartCommand*() :int (запуск медиа-плеера)  + *isVolume*() :boolean (получение значения переменной volume)  + *setVolume*(booleanvolume) :void (установка значения переменной volume) |
| *JSONParser.java* | - *url* :String (ссылка на интерфейс сервера)  - *param\_1..param\_8* :String (параметры для ссылки)  - *value*\_*1*..*value\_8* :String (параметры для ссылки) | + *updateData*(String money, String record, String status, String all\_levels, String all\_money, String all\_eating, String all\_wins) :void (обновление всех данных пользователя)  + *updateResume*(String score) :void (обновление данных о возобновлении)  + *getAchievements*() :void (получение данных о достижениях) |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  |  | + *getRating*() :void (получение данных о рейтинге)  + *editAccInf*() :void (изменение данных аккаунта пользователя) |
| *GameModes.java* | - *FROM\_ENEMY\_CIRCLE* :int (начальное количество вражеских кругов)  - *TO\_ENEMY\_CIRCLE* :int (конечное количество вражеских кругов)  - *MAX\_FOOD\_CIRCLE* :int (количество кругов-еды)  - *INITIAL\_RADIUS* :int (радиус вражеского круга)  - *SMALL\_ENEMY\_RADIUS* :int (радиус круга-еды)  - *FROM\_IMMORTAL\_CIRCLES* :int (начальное количество неуязвимых кругов)  - *TO\_IMMORTAL\_CIRCLES* :int (конечное количество неуязвимых кругов)  - *MAX\_IMMORTAL\_CIRCLES* :int (максимальное количество кругов для определенных режимов игры)  - *mainCircle* :MainCircle (круг игрока)  - *enemyCircles* :ArrayList<EnemyCircle> (массив вражеских кругов)  - *immortalCircles* :ArrayList<ImmortalCircle> (массивнеуязвимыхкругов)  - *vanishingCircles* :ArrayList<VanishingCircle> (массивисчезающихкругов) | + *generateMode*(int mode) :void (генерация уровня)  + *initEnemyCircles*(int mode) :void (генерация вражеских кругов)  + *initImmortalCircles*(int mode) :void (генерация неуязвимых кругов)  + *initVanishingCircles*(int mode) :void (генерация исчезающих кругов)  - *setEnemyCircleColor*() :void (установка цвета вражеским кружкам)  - *setVanishingCircleColor*() :void (установка цвета исчезающим кружкам)  + *getEnemy\_circles*() :ArrayList<EnemyCircle> (получение массива вражеских кругов)  + *getImmortal\_circles*() :ArrayList<ImmortalCircle> (получение массива неуязвимых кружков)  + *getVanishing*\_*circles*() :ArrayList<VanishingCircle> (получение массива исчезающих кругов) |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
| *GameManager.java* | - *PERIOD* :int (период исчезновения исчезающих кругов)  - *mainCircle* :MainCircle (круг игрока)  - *enemy*\_circles :ArrayList<EnemyCircle> (массиввражескихкружков)  - *immortal*\_circles :ArrayList<ImmortalCircle> (массив неуязвимых кругов)  - *vanishing*\_circles :ArrayList<VanishingCircle> (массив исчезающих кругов)  - *canvasView* :CanvasView (холст для рисования)  - *timer* :Timer (таймер исчезновения исчезающих кружков)  - *width* :int (ширина экрана)  - *height* :int (высота экрана)  - *vanish* :boolean (переменная содержит true, если исчезающие кружки невидимы и false, если видимы)  - *reward* :int (награда за победу в уровне)  - *gameMode*: int (игровой режим: 0 – рейтинговый, 1 – смешанный, 2 – очередь, 3 – догонялки, 4- невидимки)  - *score* :int (игровой счёт)  - *all\_rewards* :int (награда за все уровни)  - *rate* :int (коэффициент награды) | - *initGameMode*() :void (инициализация игрового режима)  - *setEnemyCircleColor*() :void  - *setVanishingCircleColor*() :void  + *getWidth*() :int (получение ширины экрана)  + *getHeight*() :int (получение высоты экрана)  - *initMainCircle*() :void (инициализация круга игрока)  + *onDraw*() :void  + *onTouchEvent*(int x, int y) :void (проверка коллизий и перемещение кружков)  - *startTimer*() :void (старт таймера исчезновения исчезающих кружков)  - *setCirclesColor*() :void(установка цвета кругов)  - *checkEnemyCirclesCollision*() :void (проверка коллизий вражеских кругов)  - *checkImmortalCirclesCollision*() :void (проверка коллизий неуязвимых кругов)  - *checkVanishingCirclesCollision*() :void (проверка коллизий исчезающих кругов)  - *checkEmptyCircles*() :void (проверка условия победы игрока)  - *lifeLoser*() :void (оповещает игрока о поражении) |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Класс** | **Переменные** | **Методы** |
|  | - *life* :boolean (переменная содержит true, если бонус жизни активирован и false, если не активирован)  - *deceleration* :boolean (переменная содержит true, если бонус замедления активирован и false, если не активирован)  - *money* :int (количество денег игрока)  - *record* :int (рекорд игрока)  - *all\_levels* :int (количество сыгранных уровней игроком)  - *all\_money* :int (количество всех заработанных за игру денег игроком)  - *all\_eating* :int (количество поглощенных кругов за всю игру игроком)  - *all\_wins* :int (количество всех пройденных уровней игроком) | - *gameEnd*() :void (запуск нового уровня)  - *moveCircles*() :void (перемещение кругов)  - *setRewards*() :void (установка награды за уровень)  - *zeroScoreAndRewards*() :void (обнуление счёта и награды)  - *switchRate*() :void (изменение коэффициента награды)  + *useLifeBonus*() :void (использование бонуса жизни)  + *useDecelerationBonus*() :void (использование бонуса замедления)  + *useGrowthBonus*() :void (использование бонуса роста)  - *getData*() :void  + *getMode*() :int (получение режима игры)  + *getScore*() :int (получение счёта игры)  + *setMode*(intmode) :void (установка режима игры)  + *setScore*(int score) :void (установка счёта игры)  + *setAll\_rewards*(int all\_rewards) :void (установка общей награды за игру)  + *getAll*\_*rewards*() :int (получение общей награды за игру)  + *isLife* :boolean (получение значения переменной «life»)  + *isDeceleration* :boolean (получение значения переменной «deceleration»)  +*setLife*(booleanlife) :void (установка значения переменной «life»)  + *setDeceleration*(boolean deceleration) :void (установка значения переменной «deceleration»)  + *setTimer*(Timer timer) :void (установка таймера)  + *getTimer*() :Timer (получение таймера) |

2.5.2 Диаграммы взаимодействия программных модулей

На рисунке 19 изображена диаграмма последовательности для варианта использования «Выбрать игровой режим».

Пользователь выбирает один из игровых режимов нажатием на соответствующую кнопку в главном меню, за которое отвечает активити *«Menu»*. Затем из активити *«Menu»* запускается активити *«Game»* с параметром *«mode»*, в котором хранится номер игрового режима, который выбрал пользователь. Из активити *«Game»* этот игровой режим передается в класс *«GameManager»* методом *setGameMode(mode)*, после чего в классе *«GameManager»* вызывается метод *initMainCircle()* для инициализации круга игрока и метод *initGameMode(mode)* класса *«GameModes»* для генерации остальных кругов в зависимости от выбранного пользователем игрового режима. После того как класс *«GameManager»* получит коллекцию сгенерированных кругов для уровня, вызывается метод *onDraw()*, в котором методом *drawCircle()* класса *«CanvasView»* прорисовываются все сгенерированные круги на игровом поле, за который отвечает класс *«CanvasView»*.

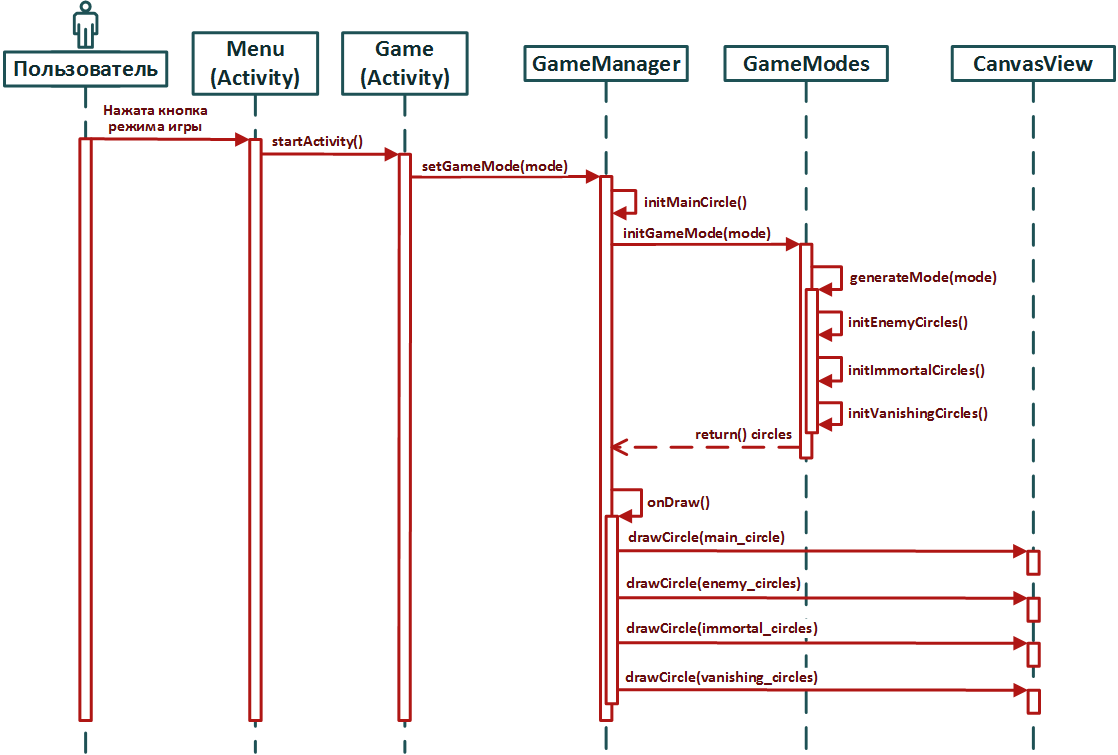


Рисунок 19. Диаграмма последовательности «Выбрать игровой режим»

На рисунке 20 изображена диаграмма последовательности для варианта использования «Использовать бонус» при выборе бонуса замедления.

После того, как пользователь нажимает кнопку с бонусом замедления, активити *«Game»* обращается к классу *«GameManager»* методом *isDeceleration()* чтобы получить переменную *«deceleration»*, которая имеет *true*, если бонус уже используется или *false*, если бонус не используется. Затем, если переменная *«deceleration»* имеет значение *false*, то методом *showInformation(“Бонус уже активирован”)* выведется сообщение о том, что бонус уже активирован, иначе, если переменная *«deceleration»* имеет значение *true*, то запросом к классу *«DatabaseHelper»* с помощью метода *getMoney()* получаем переменную *«money»*, которая хранит текущий баланс денег игрока. Затем, если его денег достаточно для покупки бонуса, то методом *useDeceleration()* активируем эффект замедления, а методом *showInformation(“Бонус активирован”)* оповещаем пользователя о том, что бонус успешно куплен и активирован. Иначе, если денег пользователя недостаточно для покупки, методом *showInformation(“Недостаточно денег”)* оповещаем пользователя о том, что ему не хватает денег на покупку бонуса.

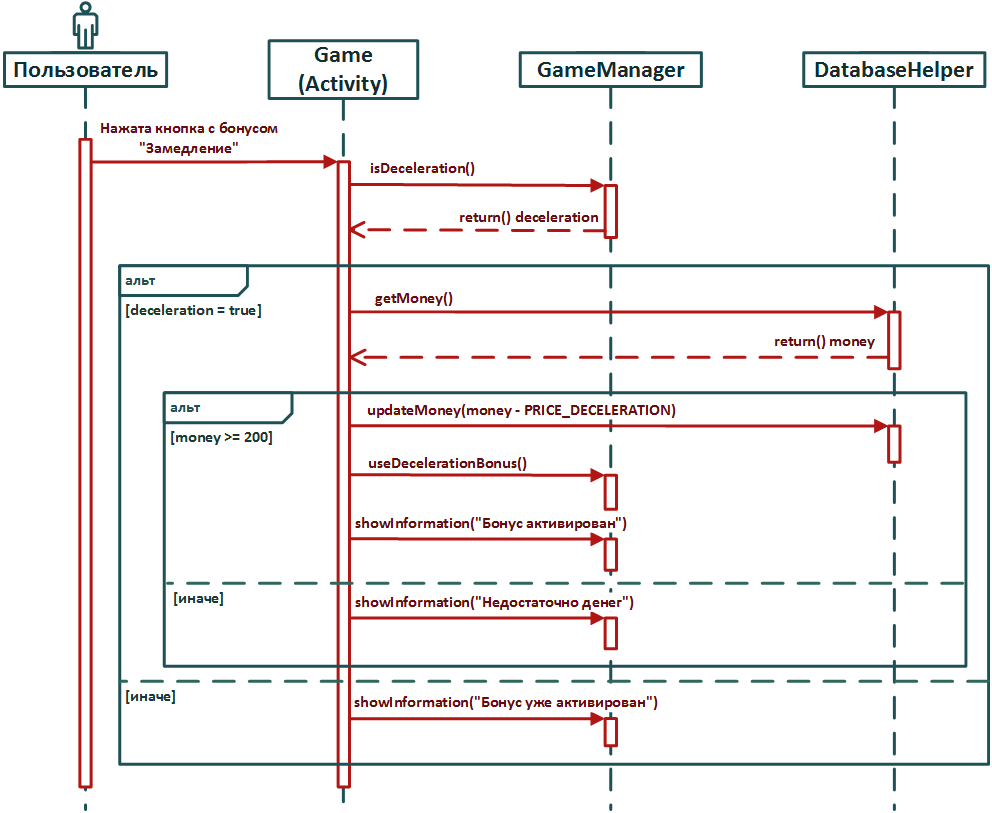


Рисунок 20. Диаграмма последовательности «Использовать бонус»

2.5.3 Проект окон и их взаимодействие

Приложение «Galos» включает в себя 19 экранов, взаимодействие которых показано на рис. 21 (стрелками указаны перемещение между активностями по нажатию кнопок) [2].

Стартовый экран с авторизацией, с которого можно перейти на экраны с регистрацией, восстановления пароля и на экран с главным меню в роли «Гость» или авторизированным пользователем. Каждая кнопка в главном меню открывает соответствующее ей окно. Кнопки «Вернуться», «Вернуться в главное меню» или кнопка «Back» на мобильном устройстве закрывают текущее окно и возвращают пользователя к предыдущему.

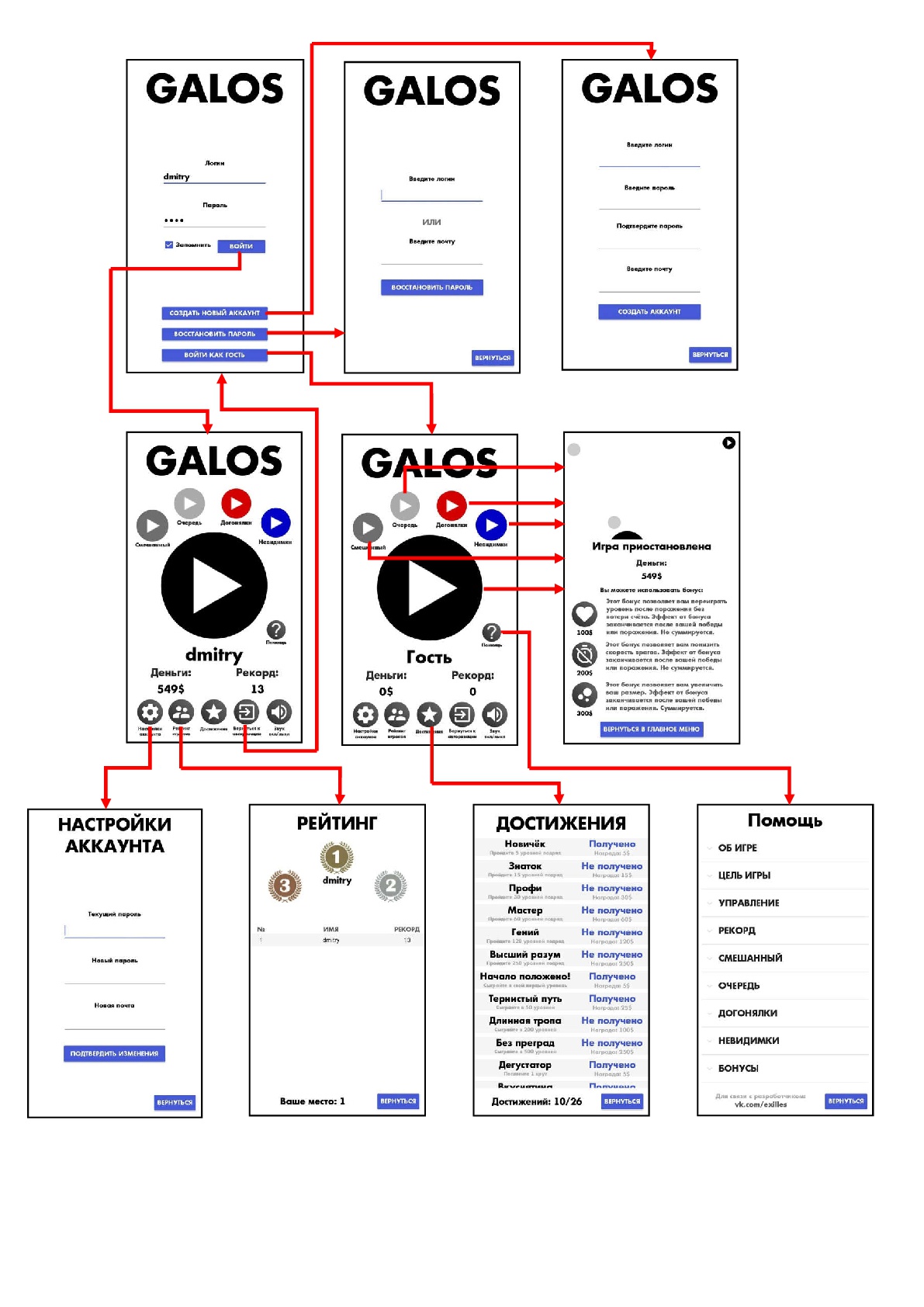


Рисунок 21. Взаимодействие экранов приложения

2.6 Техническое обеспечение

Для работы приложения необходимо мобильное устройство на ОС Android версии не ниже 4.1 JellyBean (API Level 16). Другие строгие требования к компонентам технического обеспечения отсутствуют.

#### 3 Конструирование

3.1 Обоснование выбора среды программирования и языка реализации

Для разработки приложений под операционную систему Android можно использовать множество IDE, рассмотрим самые заметные из них: Eclipse, IntelliJ IDEA, Android Studio.

**Eclipse** - открытая интегрированная среда разработки приложений, изначально разрабатываемая фирмой IBM. Наиболее важными достоинствами Eclipse являются:

* *Кроссплатформенность*. Eclipse выполняется на всех распространенных платформах;
* *Универсальность и расширяемость*. В Eclipse предусмотрена возможность использования различных дополнений и плагинов, разработанных сторонними программистами;
* *Открытость и бесплатность*. Eclipse является OpenSource-проектом и имеет активное коммьюнити, постоянно работающее над улучшением программы и расширением ее возможностей.

**IntelliJ IDEA** - коммерческая интегрированная среда разработки приложений. Разработана фирмой JetBrains и позиционируется как «IDE, которая понимает код». Основным преимуществом данной IDE перед Eclipse является понимание контекста. IDE индексирует весь проект, анализирует все, что в нем есть и строит синтаксическое дерево. Самым заметным недостатком IntelliJ IDEA служит ее коммерческая направленность и урезание функций в бесплатной версии.

**Android Studio** - IDE, созданная компанией Google и изначально направленная на разработку приложений для операционной системы Android. Стоит отметить, что в Android Studio присутствует достаточное количество преимуществ по сравнению с Eclipce, например, интеграция всех необходимых функций для новых версий Android и удобный дизайн. Android Studio регулярно обновляется и идет в ногу с последними версиями самой операционной системы, что позволяет разработчикам и любителям быть в курсе самых свежих изменений в ОС Android.

Основным критерием выбора интегрированной среды разработки является удобство разработки и отладки приложения. Учитывая преимущества Android Studio. указанные выше, и тот факт, что среда создана специально для разработки под Android и нацелена на развитие платформы, выбор был сделан в ее пользу.

По умолчанию разработка под Android ведется на языке Java. Соответственно, кроме IDE был установлен специальный инструментарий - пакет JDK (Java Development Kit).

3.2 Стиль кодирования и правила оформления кода

Использование определенного стиля кодирования и правил оформления кода позволяет делать код читаемым, аккуратным, избегать ошибок, связанных с разными стилями написания кода у людей, работающих над проектом, и в будущем легко редактировать его.

В целом, при форматировании кода использовались стандартные правила Java- кодирования.

3.2.1 Имена файлов, пакетов

* В именах пакетов используются только строчные буквы;
* Имена Java-классам даются согласно стандартной нотации Java;
* Имена классов должны быть существительными, первые буквы всех слов — заглавные;
* В именах web-папок и файлов используются только строчные буквы. Слова в многословных названиях разделяются подчеркиванием.

3.2.2 Имена методов, переменных

* Названия методов должны быть глаголами, первая буква должна быть строчной, первые буквы внутренних слов — заглавные;
* Имена переменных должны начинаться со строчной буквы, внутренние слова — с заглавной;
* Имена констант составляются из всех заглавных букв, разделенных на слова символом подчеркивания.

3.2.3 Отступы, длина строки, переносы строк

* Отступы должны составлять строго 4 пробела (не знак табуляции);
* Длина строки не должна превышать 80 символов;
* Если длина выражения превышает длину строки, то необходимо разбить его на несколько строк согласно следующим правилам:

1. перенос после запятой;
2. перенос перед оператором;
3. необходимо использовать отступ 8 пробелов для обозначения второй строки разделенного выражения. Последующие строки выравниваются по второй строке либо добавляются новые 8 пробелов для обозначения вложенности.

3.2.4 Расположение блоков, операторов, пробелов, скобок

* Определение переменных нужно располагать в начале блока, а не «ждать» первого использования переменной. Инициализация должна производиться, по возможности, сразу;
* Между именем метода и скобками для списка параметров нет пробела.
* Открывающаяся скобка { располагается на той же строке, что и сигнатура метода/заголовок if, while-блока и т.п;
* Закрывающаяся скобка } выровнена по строке начала данного блока;
* Методы разделяются пустой строкой, объявления свойств класса располагаются по одному на строку;
* На строке располагается только один оператор.

3.2.5 Структурирование кода

* Методы должны быть короткими, и выполнять только одну задачу (к примеру, почти любой цикл уже достоин того, чтобы вынести его в особый метод);
* Имена методов должны быть самодокументированными;
* Шаблоны ООП должны применяться для структурирования и облегчения восприятия.

3.3 Структура проекта

Рассмотрим структуру проекта приложения «Galos» под ОС Android:

* app
* manifest
  1. AndroidManifest.xml
* java
* dc.galos
* model
  1. DarabaseHelper.java
  2. EnemyCircle.java
  3. GameManager.java
  4. GameModes.java
  5. ImmortalCircle.java
  6. JSONParser.java
  7. MainCircle.java
  8. SimpleCircle.java
  9. Sound.java
  10. VanishingCircle.java
* controller
  1. Achievements.java
  2. Authorization.java
  3. CanvasView.java
  4. EditAccInf.java
  5. Game.java
  6. Help.java
  7. Menu.java
  8. Rating.java
  9. Registration.java
  10. RestorePassword.java
* assets
  1. a\_futurica\_extrabold.tff
  2. galos.db
* res
* anim
  1. alpha.xml
  2. down.xml
  3. list\_layout\_controller.xml
  4. list\_view\_animation.xml
  5. scale\_play.xml
  6. scale\_play\_black.xml
  7. up.xml
* drawable
  1. button\_states\_black.xml
  2. button\_states\_blue.xml
  3. button\_states\_bonus.xml
  4. button\_states\_dark\_grey.xml
  5. button\_states\_grey.xml
  6. button\_states\_light\_grey.xml
  7. button\_states\_red.xml
  8. ic\_bronze.png
  9. ic\_buble\_chart\_white\_48dp.png
  10. ic\_exit\_to\_app\_white\_48dp.png
  11. ic\_favorite\_white\_48dp.png
  12. ic\_gold.png
  13. ic\_help.png
  14. ic\_launcher\_background.xml
  15. ic\_launcher\_foreground.xml
  16. ic\_pause\_white\_24dp.png
  17. ic\_play\_1.png
  18. ic\_play\_2.png
  19. ic\_play\_arrow\_white\_24dp.png
  20. ic\_settings\_white\_48dp.png
  21. ic\_silver.png
  22. ic\_star\_white\_48dp.png
  23. ic\_supervisor\_account\_white\_48dp.png
  24. ic\_timer\_off\_white\_48dp.png
  25. ic\_volume\_off\_white\_48dp.png
  26. ic\_volume\_up\_white\_48dp.png
  27. toast\_border.xml
* font
  1. a\_futurica\_extrabold.ttf
  2. a\_futurica\_extrabolditalica.ttf
* layout
  1. activity\_achievements.xml
  2. activity\_authorization.xml
  3. activity\_edit\_acc.xml
  4. activity\_game.xml
  5. activity\_help.xml
  6. activity\_menu.xml
  7. activity\_rating.xml
  8. activity\_registration.xml
  9. activity\_restore\_password.xml
  10. exp\_list\_item\_help.xml
  11. exp\_list\_title\_help.xml
  12. list\_item\_achievements.xml
  13. list\_item\_records.xml
* menu
  1. menu\_achievements.xml
  2. menu\_rating.xml
* mipmap
* ic\_launcher
  1. ic\_launcher.png
  2. ic\_launcher.xml
* ic\_launcher\_round
  1. ic\_launcher\_round.png
  2. ic\_launcher\_round.xml
* raw
  1. background\_music.ogg
* values
  1. colors.xml
  2. dimens.xml
  3. font\_certs.xml
  4. preloaded\_fonts.xml
  5. strings.xml
  6. styles.xml
* Gradle Scripts
  1. build.gradle (Project: Galos)
  2. build.gradle (Modile: app)
  3. gradle-wrapper.properties (Gradle Version)
  4. proguard-rules.pro (ProGuard Rules for app)
  5. gradle.properties (Project Properties)
  6. settings.gradle (Project Settings)
  7. local.properties (SDK Location)

Проект Android представлен двумя корневыми папками: app и Gradle Scripts. В свою очередь папка app включает в себя 4 подпапки:

1. Папка manifests содержит файлы конфигураций или файлы манифеста приложения: AndroidManifest.xml.Файл AndroidManifest.xml является одним из самых важных в Android проекте, содержит информацию о пакетах приложения, компонентах типа Activity, Service и выполняет следующие задачи: предоставляет разрешения приложению на использование или доступ к другим компонентам системы; определяет, как будут запускаться, например, Activity (какие фильтры использовать);
2. Папка **java** содержит исходный код приложения. Классы могут быть расположены в различных пакетах (в данном приложении существуют два пакета *game* и *bd*, которые также содержат исходный код), но обязательно внутри папки java;
3. Папка res содержит файлы используемых в Android приложении ресурсов. В папке *res* расположены все используемые приложением ресурсы, включая изображения, различные xml файлы, анимации, звуковые файлы и многие другие. Внутри папки *res* эти все ресурсы распределены по своим папкам:

* Папка **anim** содержит файлы анимации;
* Папка **drawable** содержит файлы с изображениями, которые будет использоваться в приложении;
* Папка **font** содержит шрифты;
* Папка **layout**содержит xml файлы, которые используются для построения пользовательского интерфейса Android приложения;
* В папке **menu** находятся xml файлы, используемые только для создания меню.
* В **mipmap** папке хранят только значки приложения. Любые другие drawable элементы должны быть размещены в своей папке;
* Если разместить файлы, в том числе написанные на XML, в каталоге  **raw**, они не будут скомпилированы в двоичном формате, а попадают в пакет прикладных программ как есть. Для считывания таких фалов нужно использовать явные API с поддержкой потоков. К категории raw относятся аудио- и видеофайлы;
* **values** хранит те xml файлы, в которых определяются простые значения типа строк, массивов, целых чисел, размерностей, цветов и стилей;

1. Каталог assets, в котором могут храниться файлы, предназначенные для включения в пакет приложения, например, файлы БД, собственные шрифты;
2. Папка **GradleScripts** содержит в себе все файлы автоматической сборки приложения. Основной файл, с которым производилась работа - build.gradle (Module: app), в котором содержатся настройки сборки, а также подключаемые зависимости.

3.4 Способ поддержки версий файлов и сборок в процессе разработки

Система управления версиями (VCS) — программное обеспечение для облегчения работы с изменяющейся информацией. Софт регистрирует изменения в одном или нескольких файлах, чтобы в дальнейшем была возможность вернуться к определённым предыдущим версиям этих файлов. Программируемый код доступен в едином репозитории.

Один из самых популярных и хорошо зарекомендованных VCS является Git, поэтому он и был выбран для поддержки версий файлов и сборок в процессе разработки приложения.

**Git** — это распределенная система контроля версиями. В отличие от централизованных и локальных систем, Git не просто выгружает последние версии файлов, но и полностью копируют репозиторий локально. В случае, когда «умирает» главный сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных.

#### 4 Тестирование и отладка

4.1 Выбор инструментов тестирования

В IDE Android Studio есть возможность подключить необходимый инструментарий для тестирования приложений: наиболее популярный и широко используемый фрейморк JUnit и фреймфорк Espresso, являющийся частью репозитория Android Support Repository, что позволяет быстро и удобно добавлять поддержку тестов в проект [6].

В Android Studio все исходные коды локальных unit-тестов хранятся в специально отведенной директории: (src/test/java).

Исходя из вышесказанного, было решено использовать уже встроенные средства тестирования, такие как JUnit для функционального тестирования и Espresso для тестирования интерфейса приложения.

4.2 Методика и результаты модульного тестирования

В качестве примера модульного тестирования рассмотрим совместную работу методов*getAchievement()*, возвращающего строку с информацией о полученном достижении вида «Получено достижение: *<название достижения>*»*,* и *checkAllEating()*, который вызывает метод getAchievement() и возвращает сформированное сообщение. В таблице 2 показан листинг данных методов, где справа красными цифрами указаны узлы будущего потокового графа.

Таблица 2

Листинг тестируемых методов

|  |  |  |
| --- | --- | --- |
| 1 | public String checkAllEating(int all\_eating){ |  |
| 2 | String message = ""; | **1** |
| 3 | switch (all\_eating){ |  |
| 4 | case 1: | **2** |
| 5 | message = getAhievement(10); | **3** |
| 6 | return message; |  |
| 7 | case 50; | **4** |
| 8 | message = getAhievement(11); | **5** |
| 9 | return message; |  |
| 10 | case 300: | **6** |
| 11 | message = getAhievement(12); | **7** |
| 12 | return message; |  |
| 13 | case 700: | **8** |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| 14 | message = getAhievement(13); | **9** |
| 15 | return message; |  |
| 16 | default: | **10** |
| 17 | message = getAhievement(-1); | **11** |
| 18 | return message; |  |
| 19 | } |  |
| 20 | } |  |
| 21 | public String getAhievement(int position){ |  |
| 22 | if (position != -1) { | **12** |
| 23 | StringBuilder newStatus = new StringBuilder(status); | **13** |
| 24 | new Status.setCharAt(position, '1'); |  |
| 25 | status = String.valueOf(newStatus); |  |
| 26 | money += Integer.parseInt(rewardAchievements[position]); |  |
| 27 | return "Получено достижение: " + titleAchievements[position]; |  |
| 28 | } |  |
| 29 | else return "Достижение не получено"; | **14** |
| 30 | } | **15** |

На основе данных из таблицы 2 был построен потоковый граф, который представлен на рис. 22.

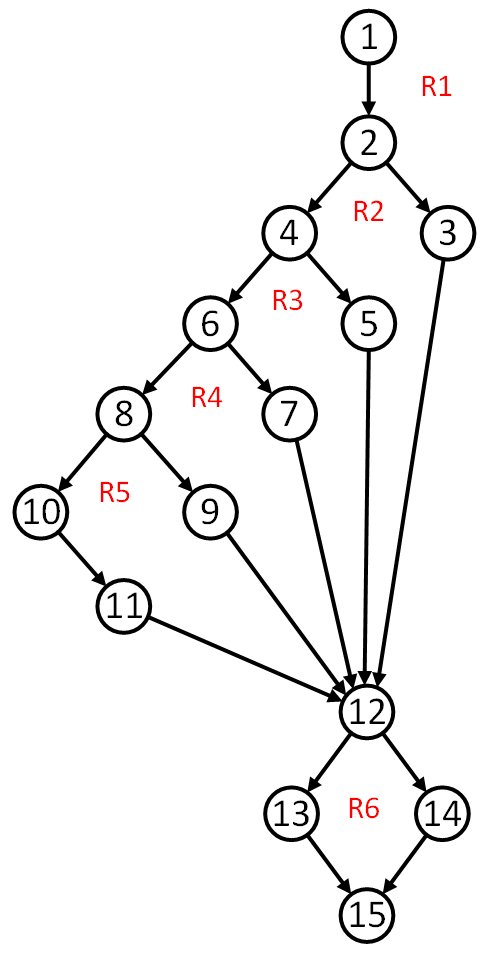


Рисунок 22. Потоковый граф тестируемых методов

4.2.1 Тестирование базового пути

Данный потоковый граф имеет 6 регионов, 15 узлов, 19 дуг и 5 предикатных узлов. Независимые пути начинаются в начальном узле, а заканчиваются в конечном узле графа и формируются в порядке от самого короткого к самому длинному. Такими путями будут следующие:

Путь 1: 1-2-3-12-13-15;

Путь 2: 1-2-3-12-14-15;

Путь 3: 1-2-4-5-12-13-15;

Путь 4: 1-2-4-5-12-14-15;

Путь 5: 1-2-4-6-7-12-13-15;

Путь 6: 1-2-4-6-7-12-14-15;

Путь 7: 1-2-4-6-8-9-12-13-15;

Путь 8: 1-2-4-6-8-9-12-14-15;

Путь 9: 1-2-4-6-8-10-11-12-13-15;

Путь 10: 1-2-4-6-8-10-11-12-14-15;

Цикломатическая сложность графа, вычисленная тремя различными способами равна:

1. Потоковый граф имеет 6 регионов = Цикломатическая сложность = 6;
2. Цикломатическая сложность = 19 дуг - 15 узлов + 2 = 6;
3. Цикломатическая сложность = 5 предикатных узлов + 1 = 6.

Таким образом, уровень сложности потокового графа равен 6. Следовательно, необходимо составить 6 тестовых вариантов для прохождения всех путей.

Так как всего возможных путей на потоковом графе 10, а для тестов достаточно протестировать лишь 6 из них, то необходимо выделить те пути, которые не нужно тестировать. Такими путями являются: путь 1, путь 3, путь 5 и путь 7. Проходя эти пути невозможно попасть в узел 13, следовательно, тестировать их не нужно. Также важно отметить, что путь 10 также невозможно протестировать, так как невозможно попасть в узел 14, значит достаточно написать всего 5 тестов для того, чтобы полностью протестировать работу этих методов.

В таблице 3 показаны исходные данные для теста каждого из путей с их ожидаемым результатом.

Таблица 3

Исходные данные для теста каждого из путей с их ожидаемым результатом

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Путь** | **Исходные данные** | **Ожидаемый результат** |
| 1 | 1-2-3-12-14-15 | all\_eating = 1 | “Получено достижение: Дегустатор” |
| 2 | 1-2-4-5-12-14-15 | all\_eating = 50 | “Получено достижение: Вкуснятина” |

Продолжение таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Путь** | **Исходные данные** | **Ожидаемый результат** |
| 3 | 1-2-4-6-7-12-14-15 | all\_eating = 300 | “Получено достижение: Обжора” |
| 4 | 1-2-4-6-7-12-14-15 | all\_eating = 700 | “Получено достижение: Чёрная дыра” |
| 5 | 1-2-4-6-8-10-11-12-13-15 | all\_eating = 1000 | “Достижение не получено” |

4.2.2 Тестирование ветвей и операторов отношений

Вновь рассмотрим листинг тестируемых методов и выделим в нем все условия, которые обозначены красным в правой части таблицы 4.

Таблица 4

Листинг тестируемых методов с выделенными условиями

|  |  |  |
| --- | --- | --- |
| 1 | public void checkAllEating(int all\_eating){ |  |
| 2 | String message = ""; |  |
| 3 | switch (all\_eating){ |  |
| 4 | case 1: | **С1** |
| 5 | message = getAhievement(10); |  |
| 6 | showAchievement(message); |  |
| 7 | break; |  |
| 8 | case 50; | **С2** |
| 9 | message = getAhievement(11); |  |
| 10 | showAchievement(message); |  |
| 11 | break; |  |
| 12 | case 300: | **С3** |
| 13 | message = getAhievement(12); |  |
| 14 | showAchievement(message); |  |
| 15 | break; |  |
| 16 | case 700: | **С4** |
| 17 | message = getAhievement(13); |  |
| 18 | showAchievement(message); |  |
| 19 | break; |  |
| 20 | default: | **С5** |
| 21 | message = getAhievement(-1); |  |
| 22 | showAchievement(message); |  |
| 23 | break; |  |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| 24 | } |  |
| 25 | } |  |
| 26 | public String getAhievement(int position){ |  |
| 27 | if (position != -1) { | **С6** |
| 28 | StringBuilder newStatus = new StringBuilder(status); |  |
| 29 | new Status.setCharAt(position, '1'); |  |
| 30 | status = String.valueOf(newStatus); |  |
| 31 | money += Integer.parseInt(rewardAchievements[position]); |  |
| 32 | return "Получено достижение: " + titleAchievements[position]; |  |
| 33 | } |  |
| 34 | else return "Достижение не получено"; |  |
| 35 | } |  |

Для каждого условия в методе выполним следующие действия:

1. Построим ограничения условий (ОУ):
2. Выделим ограничения результата по каждому простому условию
3. Построим ограничивающее множество (ОМ) для каждого условия:
4. Для каждого элемента ОМ разрабатывается тестовый вариант:
   * Тестовый вариант 1:  
     Исходные данные:all\_eating = 1;  
     Ожидаемый результат: “Получено достижение: Дегустатор”
   * Тестовый вариант 2:  
     Исходные данные:all\_eating = 50;  
     Ожидаемый результат: “Получено достижение: Вкуснятина ”

Аналогично составляются тестовые варианты для остальных множеств, но уже на данном этапе, очевидно, что тестовые варианты полностью покрываются тестами базового пути, указанными (см. п. 4.2.1), поэтому проводить тестирование ветвей и операторов отношений нецелесообразно.

4.2.3 Построение информационного графа и накладывание его на потоковый граф

На рисунке 23 показан граф с управляющими и информационными связями. В вершине 1 формируется переменная «message», и далее ее значение используется в вершинах 3, 5, 7, 9, 11.

Можно также заметить, что переменная «message» определяется непосредственно перед ее использованием, поэтому в DU-тестировании нет необходимости. Кроме этого, существующие DU-цепочки покроют тесты для базового пути (см. п. 4.2.1).

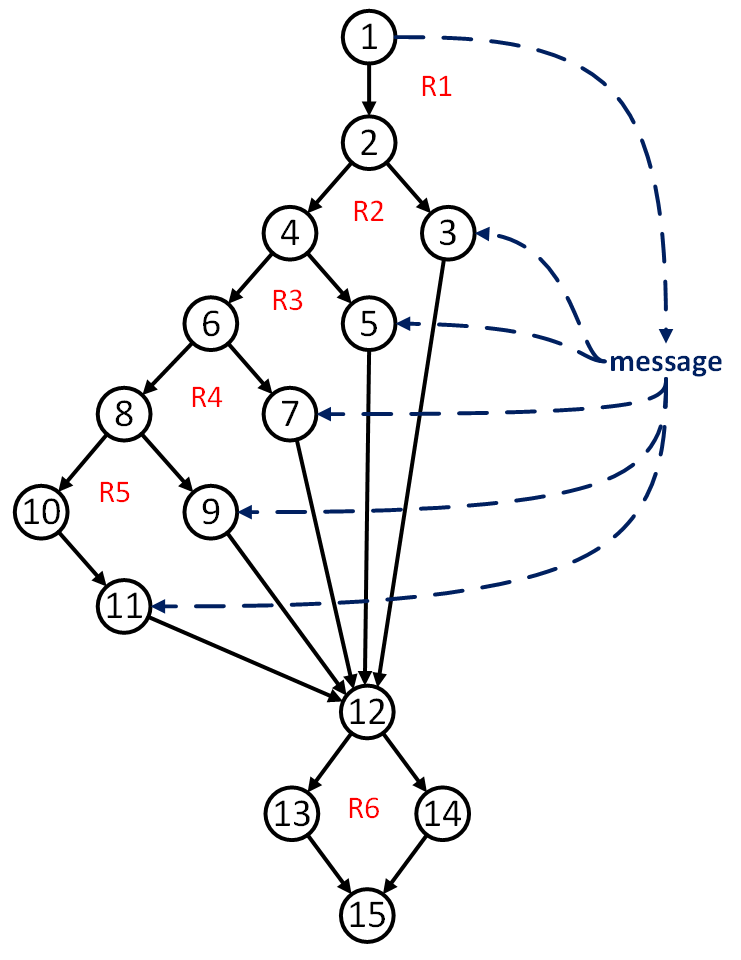


Рисунок 23. Потоковый граф с информационными связями тестируемых методов

4.2.4 Автоматизация тестирования

В таблице 5 показаны тестовые методы для базового пути, которые содержатся в классе «getAchievementUnitTest».

Таблица 5

Тестовые методы для базового пути

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Тест** | **Описание** | **Результат** |
| 1 | @Test  public void test() throws Exception {  assertEquals("Получено достижение: Дегустатор", | Покрытие первого базового пути | Тест успешно пройден |

Продолжение таблицы 5

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Тест** | **Описание** | **Результат** |
|  | gameManager.checkAllEating(1)); |  |  |
| 2 | @Test  public void test() throws Exception {  assertEquals("Получено достижение: Вкуснятина", gameManager.checkAllEating(50)); | Покрытие второго базового пути | Тест успешно пройден |
| 3 | @Test  public void test() throws Exception {  assertEquals("Получено достижение: Обжора", gameManager.checkAllEating(300)); | Покрытие третьего базового пути | Тест успешно пройден |
| 4 | @Test  public void test() throws Exception {  assertEquals("Получено достижение: Чёрнаядыра", gameManager.checkAllEating(700)); | Покрытие четвертого базового пути | Тест успешно пройден |
| 5 | @Test  public void test() throws Exception {  assertEquals("Достижениенеполучено", gameManager.checkAllEating(1000)); | Покрытие пятого базового пути | Тест успешно пройден |

На рисунке 24 показан скриншот работы модульных тестов в Android Studio. Для того, чтобы быть полностью уверенным в работе модульных тестов, был изменён ожидаемый результат в каждом из тестов таким образом, чтобы они не сработали. На рисунке 25 показан скриншот работы таких модульных тестов. Ни один из тестов не сработал, так как ожидаемый результат не совпал с реальным, а значит можно утверждать, что модульные тесты работают корректно.

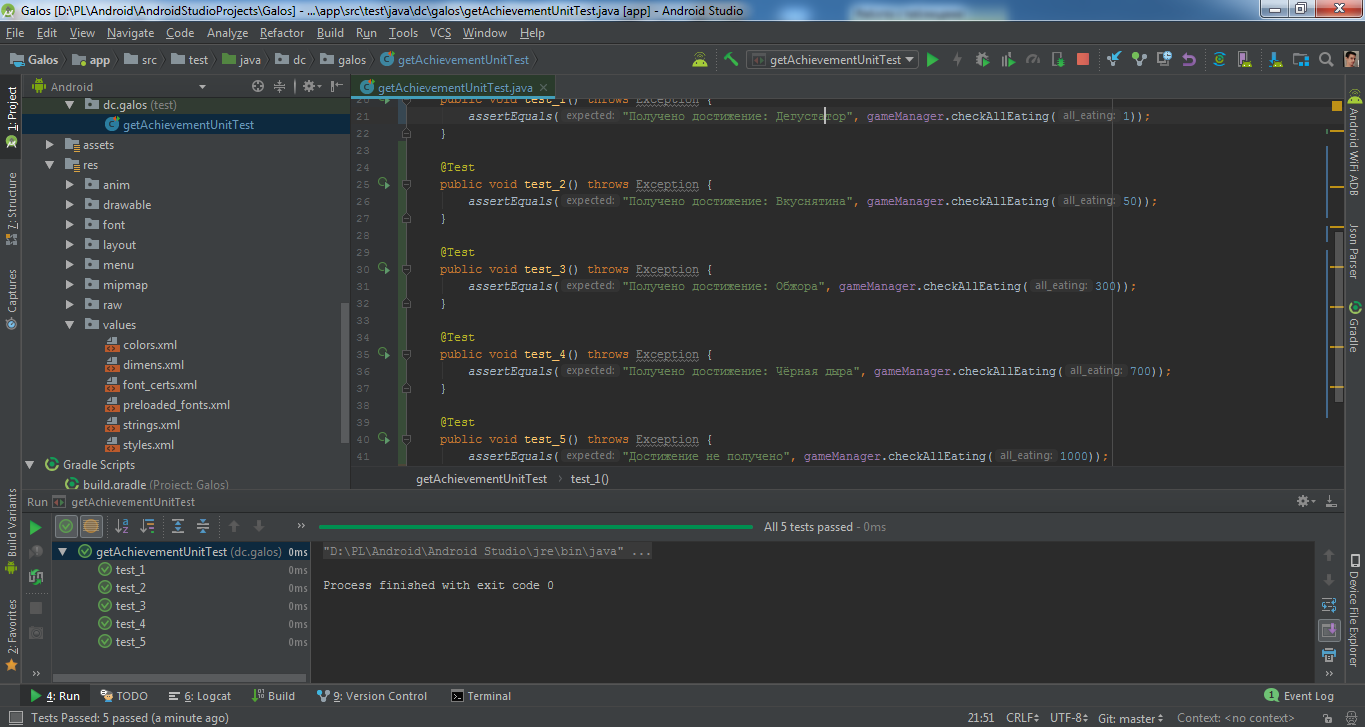


Рисунок 24. Успешная работа модульных тестов

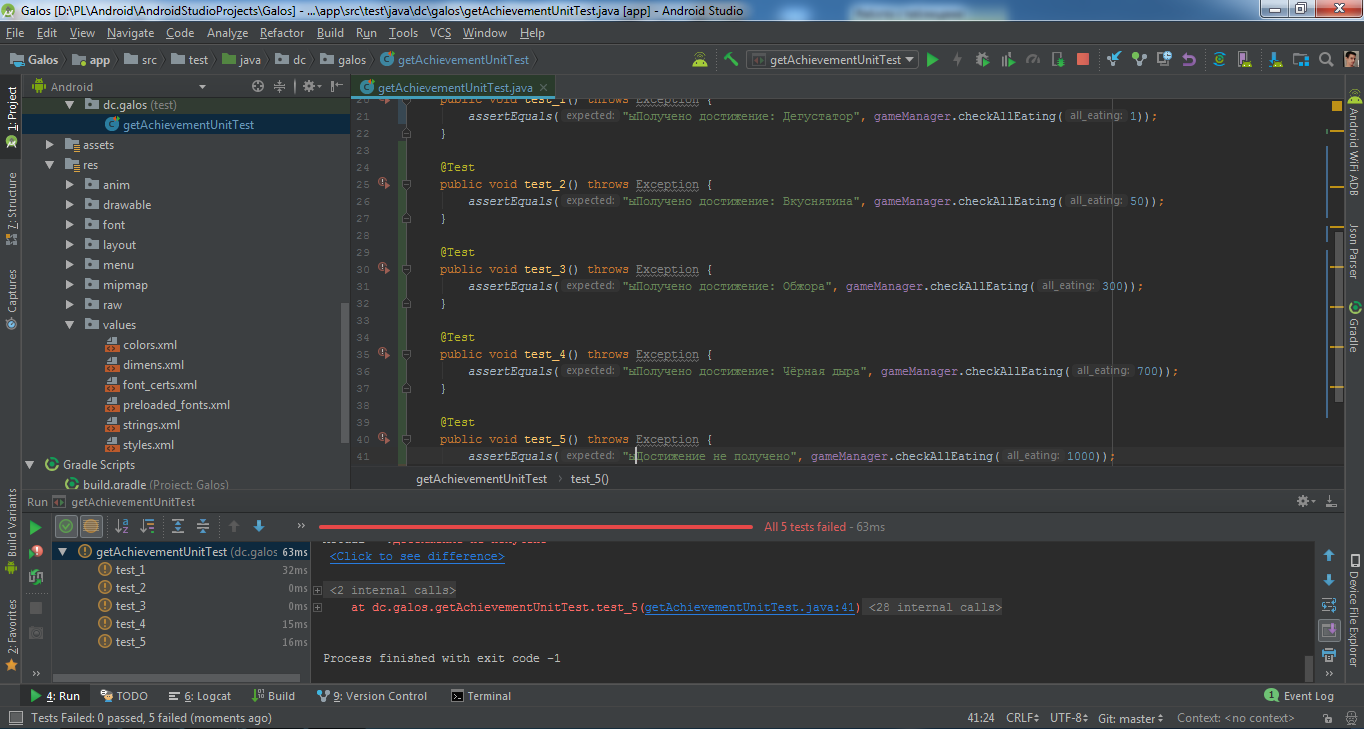


Рисунок 25. Проверка корректности работы модульных тестов

4.3 Тестирование пользовательского интерфейса

Тестирование пользовательского интерфейса проводится при помощи библиотеки *Espresso*, которая предоставляет интерфейсы API для моделирования взаимодействия с пользователем в пределах одного целевого приложения, что в свою очередь необходимо для проведения UI-тестирования.

Espresso имеет несколько основных классов для тестирования:

* *Espresso* – основной класс. Содержит в себе статические методы, такие как нажатия на системные кнопки (*Back*, *Home*), вызвать/спрятать клавиатуру, открыть меню, обратится к компоненту;
* *ViewMatchers* – позволяет найти компонент на экране в текущей иерархии;
* *ViewActions* – позволяет взаимодействовать с компонентом (*click*, *longClick*, *doubleClick*, *swipe*, *scroll* и т.д.);
* *ViewAssertions* – позволяет утвердить состояние компонента.

Для тестирования в студии уже есть пакет «androidTest», предназначенный для хранения тестов пользовательского интерфейса.

Был написан тест, в котором происходит переход в роли гостя из активити *«Authorization»*по кнопке *«loginAsGuestButton»* к активити *«Menu»,* которое должно иметь поле *«loginTextView»* с текстом *«Гость».*Содержание этого текстового поля и проверяется в тесте, листинг которого представлен в таблице 6.

Таблица 6

Листинг теста пользовательского интерфейса

|  |  |
| --- | --- |
| 1 | public class AuthorizationTest { |
| 2 | @Rule |
| 3 | public ActivityTestRule<Authorization> mActivityRule = new |
| 4 | ActivityTestRule<>(Authorization.class); |
| 5 | @Test |
| 6 | public void onClick() throws Exception { |
| 7 | onView(withId(R.id.loginAsGuestButton)).perform(click()); |
| 8 | onView(withId(R.id.loginTextView)).check(matches(withText(“Гость”))); |
| 9 | } |
| 10 | } |

Так как UI-тестирование должно происходить на эмуляторе или на реальном мобильном устройстве, для теста использовался реальный смартфон LeMobileLEX626 (Android 6.0, API 23). На рис. 26 представлен пример теста и результат его выполнения.

Так как успешное выполнение теста не гарантирует корректную работу самого теста, его ожидаемый результат был изменен, чтобы проверить, что выполнение теста будет не успешным. Результат работы такого теста представлен на рис. 27. Так как тест не был выполнен, то можно считать его работу корректной.

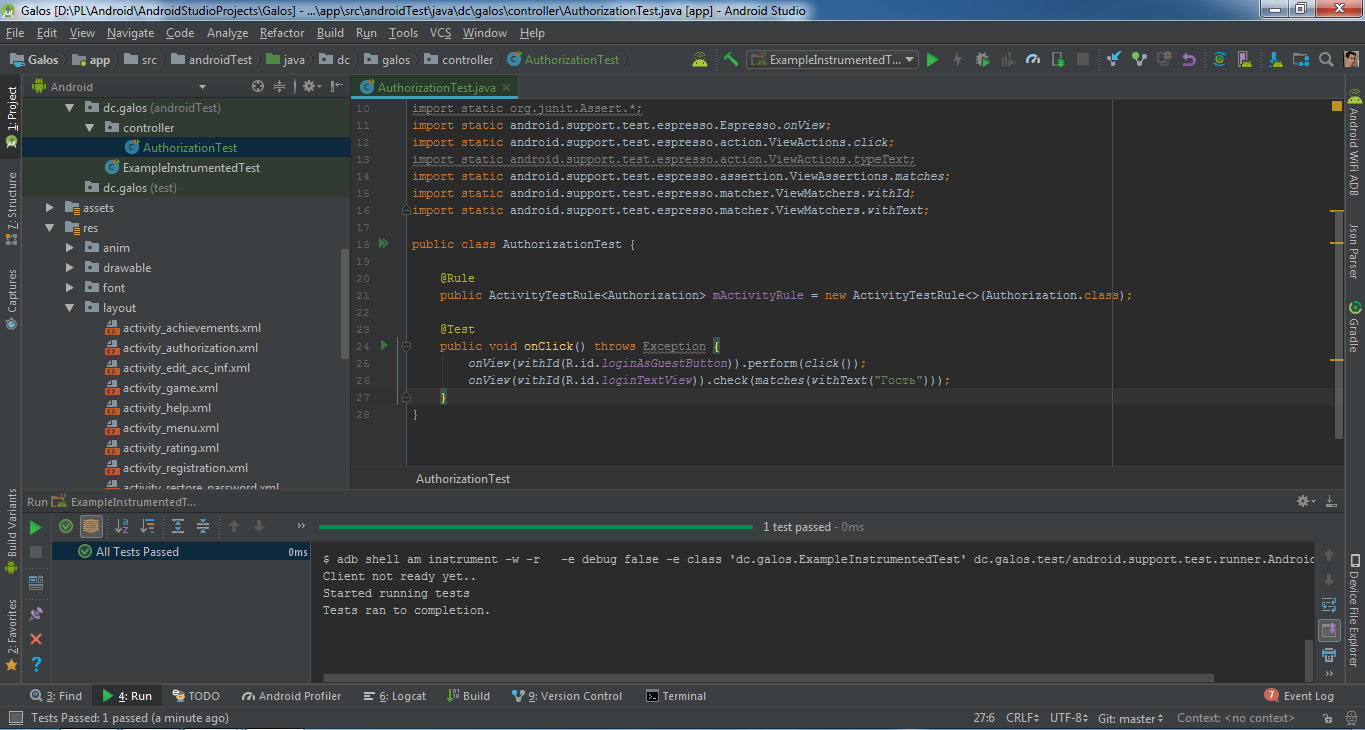


Рисунок 26. Успешное выполнение теста пользовательского интерфейса

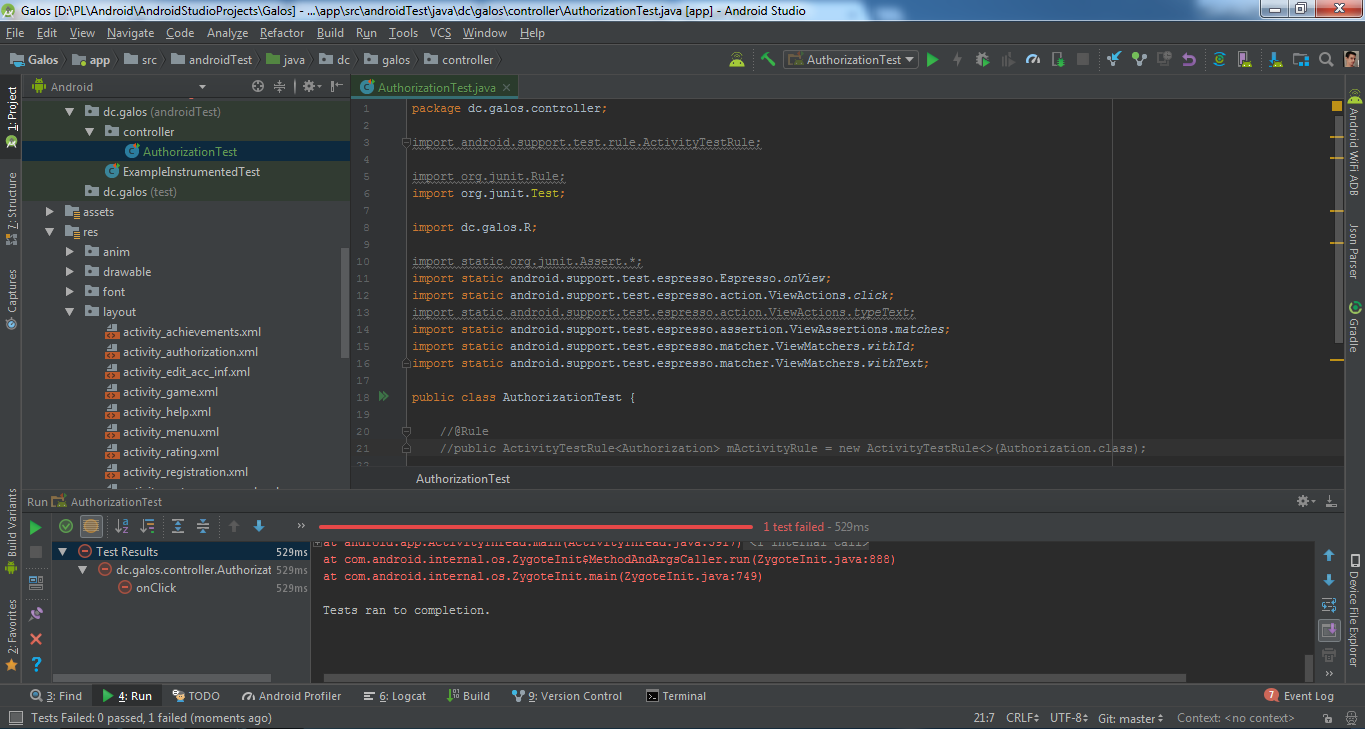


Рисунок 27. Проверка корректности работы теста пользовательского интерфейса

#### 5 Ввод в эксплуатацию

5.1 Установка приложения на мобильное устройство

IDE Android Studio поддерживает установку, запуск и тестирование android-приложений на реальном устройстве через USB-кабель. Однако, прежде всего, необходимо подготовить само мобильное устройство: для проведения испытаний было выбранное следующее мобильное устройство – LeMobile LEX626 (Android 6.0, API 23), так как оно удовлетворяет всем необходимым требованиям для успешной работы приложения, а также имеет подходящий размер экрана (5.5” дюймов) для корректного отображения пользовательского интерфейса.

На устройстве был включен режим разработчика и разрешена отладка по USB (рис.28).

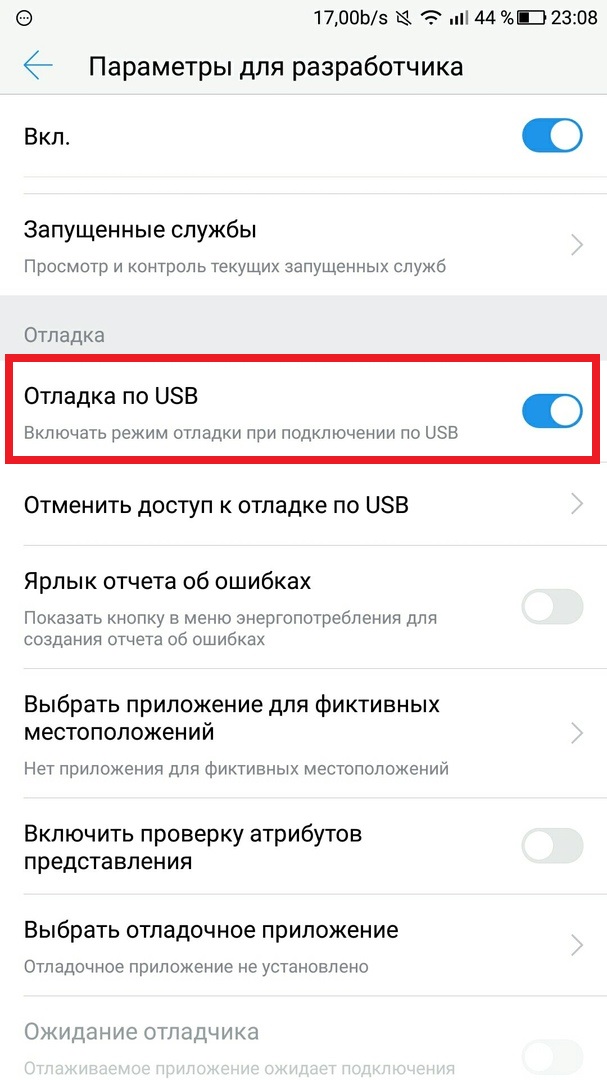


Рисунок 28. Включение отладки по USBна мобильном устройстве

Затем мобильное устройство было подключено физически к ноутбуку, на котором велась разработка приложения, и стало доступным в списке устройств в IDE Android Studio (рис.29).

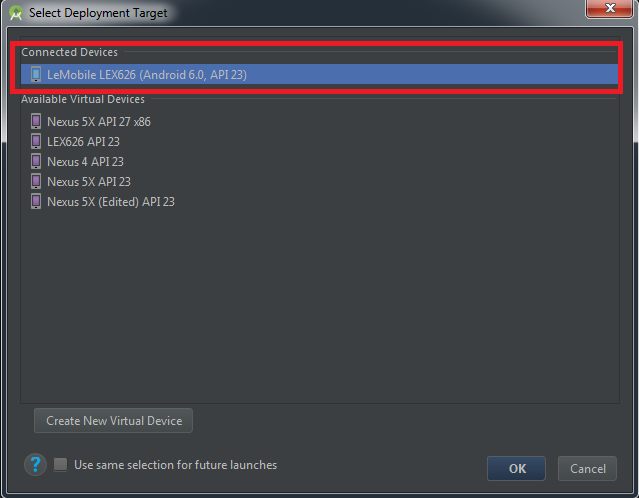


Рисунок 29. Подключение мобильного устройства к IDE Android Studio

Таким образом, через запуск не при помощи эмулятора, а на реальном устройстве, приложение «Galos» было установлено на LeMobile LEX626 (см. рис. 30).

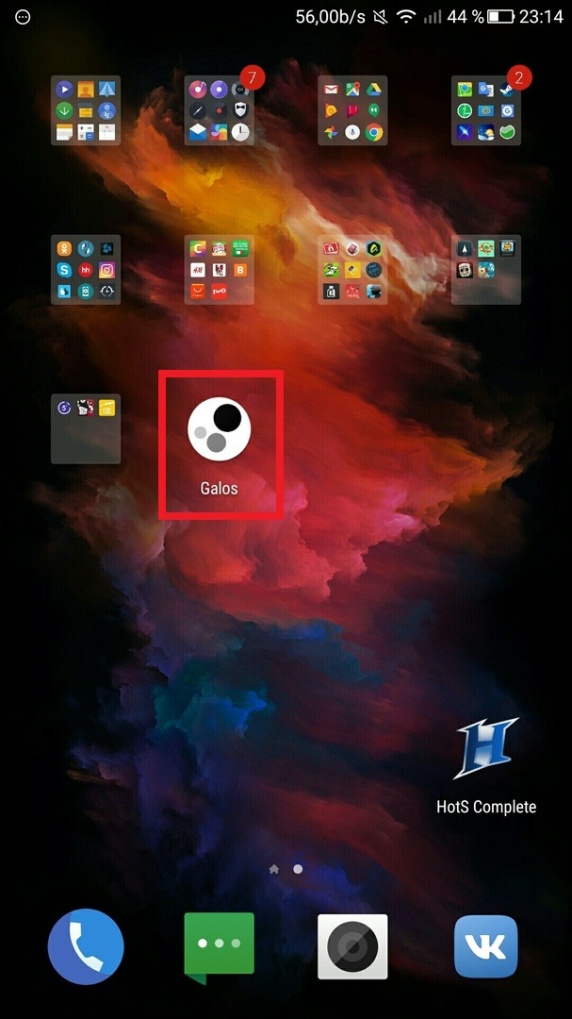


Рисунок 30. Установленное приложение «Galos»

Аналогичным образом игра была установлена на такое же мобильное устройство LeMobile LEX626 (Android 6.0, API 23) другого пользователя (рис 31).

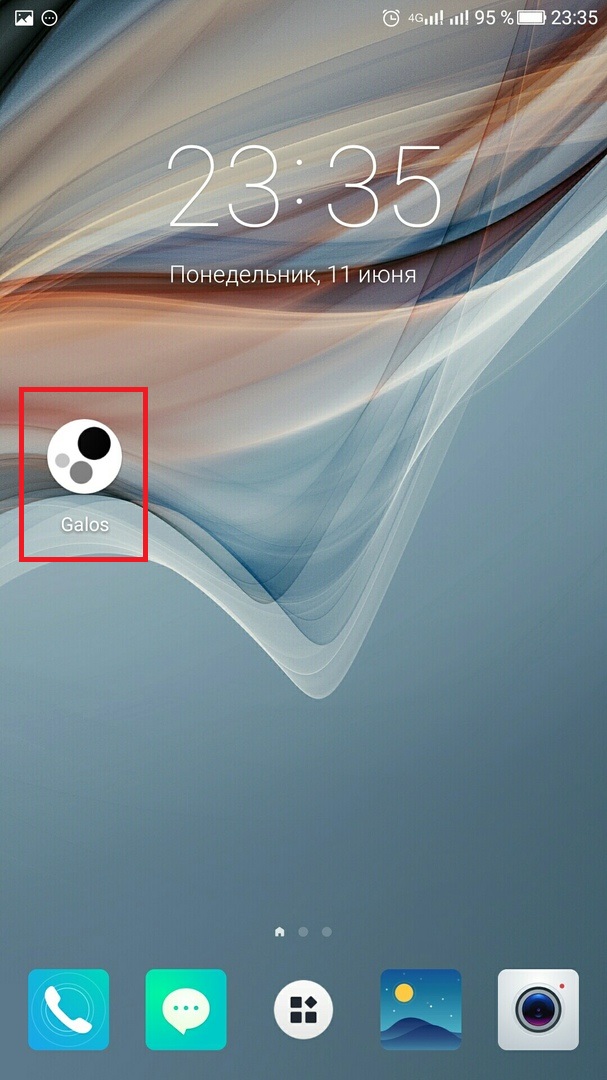


Рисунок 31. Установленное приложение «Galos» на другое мобильное устройство

Один из пользователей поделился своим отзывом об этой игре, который показан на рисунке 32.

Оба пользователя успешно прошли регистрацию в игре и уже сделали первые рекорды, которые можно видеть в разделе «Рейтинг игроков» игры «Galos» (рис. 33).

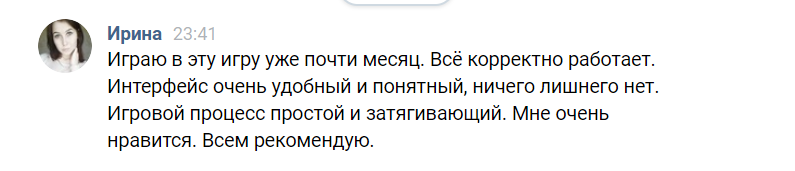


Рисунок 32. Отзыв пользователя об игре «Galos»



Рисунок 33. Рейтинг игроков игры «Galos»

5.2 Руководство пользователя

5.2.1 Введение

Мобильное приложение «Galos» представляет собой игру жанра аркада, позволяющую пользователям приятно провести время и посоревноваться с другими игроками.

Данное руководство предназначено для пользователей мобильного приложения «Galos», установленного на устройствах, работающих под управлением мобильной операционной системы *Android версии 4.1 JellyBean* и выше. В нем содержатся сведения, необходимые пользователям для эксплуатации приложения.

Интерфейс Приложения разработан с учетом особенностей мобильных устройств:

* основные разделы приложения вынесены на главный экран;
* запустив приложение, пользователь может создать новый аккаунт, восстановить пароль от аккаунта, авторизироваться под своим аккаунтом или авторизироваться как гость с ограничением игрового функционала (невозможно просматривать и участвовать в рейтинге), после авторизации пользователь может просмотреть помощь по игре, включить или выключить фоновую музыку, настроить аккаунт, посмотреть рейтинг игроков и список достижений, а также сыграть в один из пяти режимов игры, которые будут представлены в главном меню;
* в приложении предусмотрены простые переходы и понятная навигация.

Для управления данными в приложении используется типовое действие: «*tap»* – нажатие на элемент интерфейса,«*click»* — жест прикосновения пальца к поверхности экрана, и «*move»* – жест прикосновения пальца к поверхности экрана с его постоянным перемещением, не отрывая от экрана.

5.2.2 Запуск приложения

Работа приложения начинается с окна авторизации, представленного на рисунке 34.

В окне авторизации можно ввести логин и пароль от аккаунта в соответствующие поля и нажать кнопку «Войти» чтобы перейти к главному меню, представленному на рисунке 35. Также пользователь может поставить галочку в чекбоксе «Запомнить», чтобы при следующем входе в игру сразу войти под этим аккаунтом и перейти к главному меню.

Рисунок 34. Окно авторизации Рисунок 35. Главное меню

При нажатии на кнопку «Создать новый аккаунт» в окне авторизации, откроется окно с регистрацией, представленное на рисунке 36, в котором необходимо в соответствующие поля ввести логин, пароль, подтверждение пароля и почту, а затем нажать на кнопку «Создать аккаунт» для создания игрового аккаунта.

При нажатии на кнопку «Вернуться», пользователь всегда будет возвращаться к предыдущему окну.

Также в окне авторизации можно нажать на кнопку «Восстановить пароль», тогда откроется окно с восстановлением пароля (рис. 37), куда можно ввести логин и/или пароль, а затем нажать на кнопку «Восстановить пароль», после этого система пришлёт письмо на указанную при регистрации почту с подтверждением восстановления пароля, в котором после перехода по указанной ссылке, на почту пользователя придёт повторное письмо с паролем от его аккаунта.

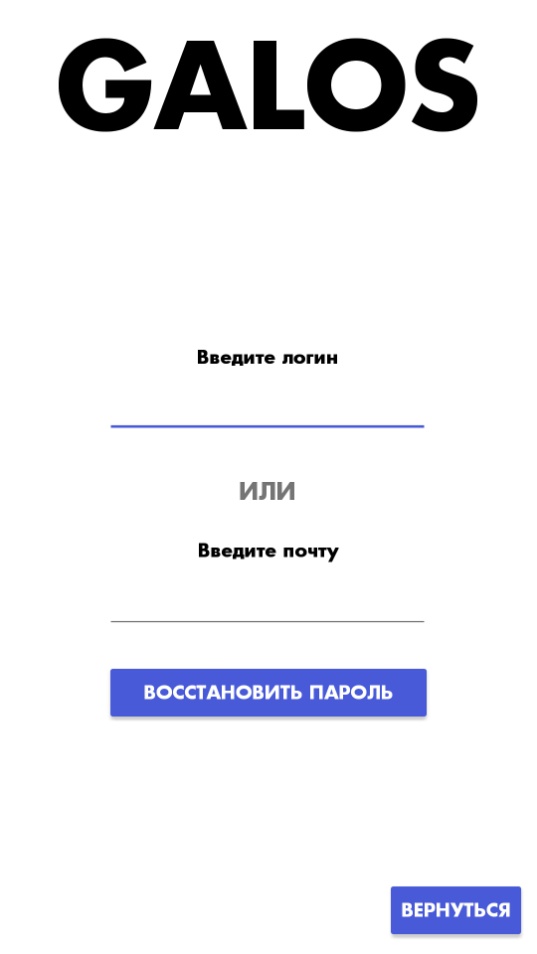
 

Рисунок 36. Окно регистрации Рисунок 37. Окно восстановления пароля

При нажатии на кнопку «Войти как гость» в окне авторизации, после чего откроется окно главного меню (рис. 38), в котором при нажатии на кнопки «Настройки аккаунта» и «Рейтинг игроков» появиться всплывающее сообщение, показанное на рисунке 39, в котором сказано, что данная функция недоступная для неавторизированного пользователя и при нажатии на кнопку «Создать аккаунт», откроется окно регистрации, показанное на рисунке 36. После создания аккаунта именно таким способом, все игровые данные из аккаунта в роли гостя, перенесутся на вновь созданный аккаунт, а игровые данные аккаунта гостя обнулятся. При нажатии кнопки «Позже» во всплывающем сообщении, данное сообщение исчезнет.

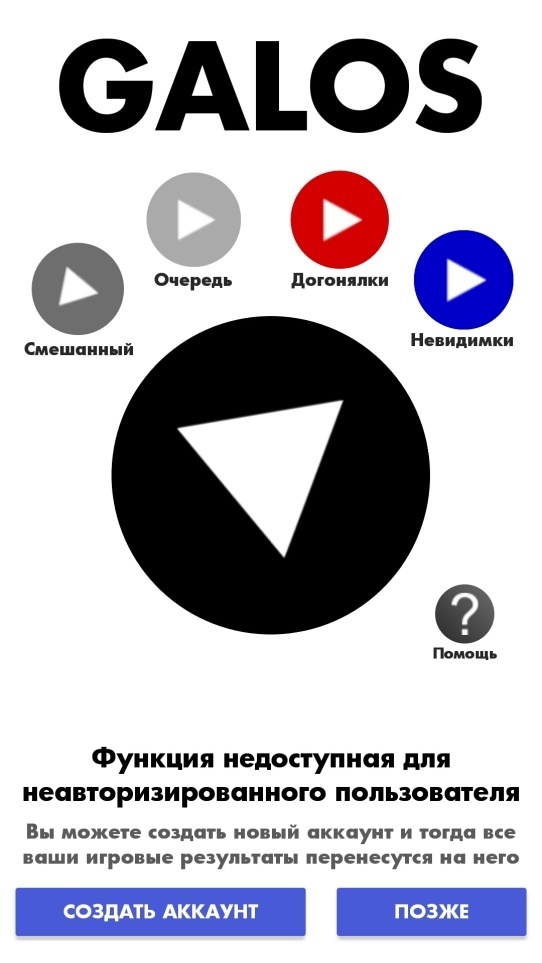
 

Рисунок 38. Главное меню (Гость) Рисунок 39. Всплывающее сообщение

В главном меню, представленном на рисунках 35 и 38, пользователь может наблюдать в нижней части окна количество его денег, наибольший игровой рекорд и его логин (имя) (при авторизации за гостя, его именем считается «Гость»), в верхней части экрана название игры «Galos», в центральной части экрана 5 кнопок с запуском того или иного игрового режима, а также кнопку «Помощь» и в самом низу экрана 5 кнопок: «Настройки аккаунта», «Рейтинг игроков», «Достижения», «Вернуться к авторизации» и «Звук вкл/выкл».

При нажатии на кнопку «Звук вкл/выкл» будет включена/выключена фоновая музыка приложения.

При нажатии на кнопку «Вернуться к авторизации» откроется окно авторизации.

При нажатии на кнопку «Настройки аккаунта» откроется окно изменения данных аккаунта (рис. 40), где можно изменить текущий пароль или почту.

При нажатии на кнопку «Рейтинг игроков» откроется окно с рейтингом игроков (рис. 41), в котором в верхней части показаны первые 3 игрока из рейтинга, ниже представлен список игроков в порядке убывания рейтинга, а в самом низу показана информация о текущем месте пользователя в этом рейтинге.

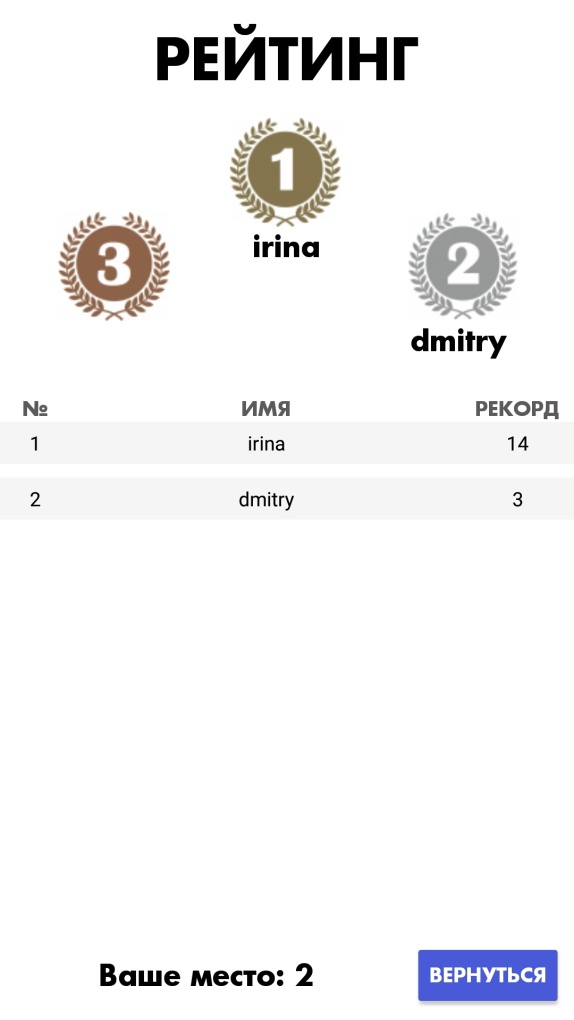
 

Рисунок 40. Настройки аккаунта Рисунок 41. Рейтинг игроков

При нажатии на кнопку «Достижения» в главном меню, откроется окно с достижениями (рис. 42), где показан лист из 26 достижений с их названием, условием достижения, наградой, а также статусом (получено достижения или нет). В нижней части экрана показана информация о том, сколько достижений получено пользователем.

При нажатии на кнопку «Помощь» в главном меню, откроется окно с информацией по игре (рис. 43), в котором есть разные пункты, при нажатии на которые откроется информация соответствующая данному пункту. В нижней части экрана отображается информация о том, как связаться с разработчиком.

При нажатии на одну из кнопок начала игры, откроется окно со сгенерированным уровнем, представленное на рисунке 44.

При нажатии на кнопку в виде паузы в верхнем правом углу во время игрового процесса, откроется дополнительное игровое меню (рис. 45), в котором будет показано количество денег игрока, а также 3 кнопки с разными бонусами, их описание и цена. При нажатии на одну из кнопок будет активирован соответствующий эффект. При нажатии на кнопку «Вернуться в главное меню», откроется окно главного меню приложения. Чтобы скрыть данное меню, нужно снова нажать на кнопку в верхнем правом углу, только на этот раз она будет в виде небольшого треугольника в чёрном круге (до этого эта кнопка была в виде паузы).

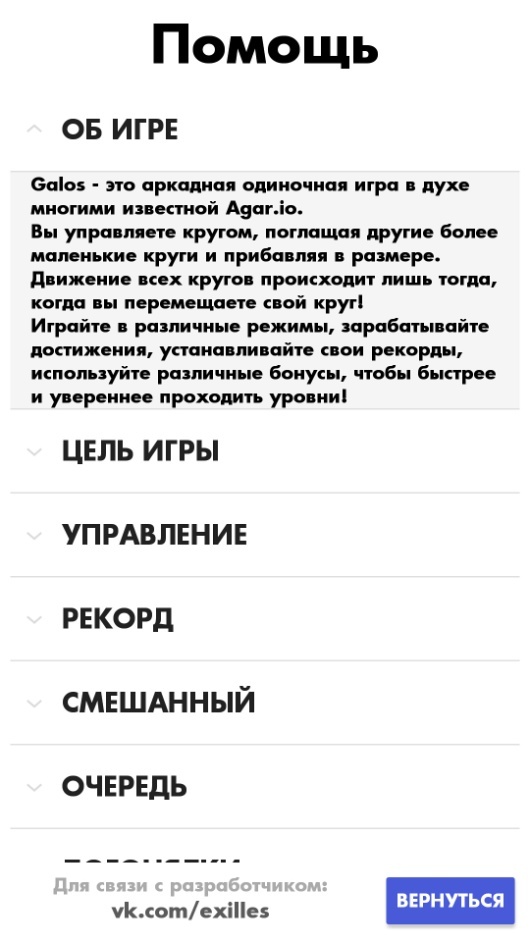
 

Рисунок 42. Окно с достижениями Рисунок 43. Окно с информацией по игре

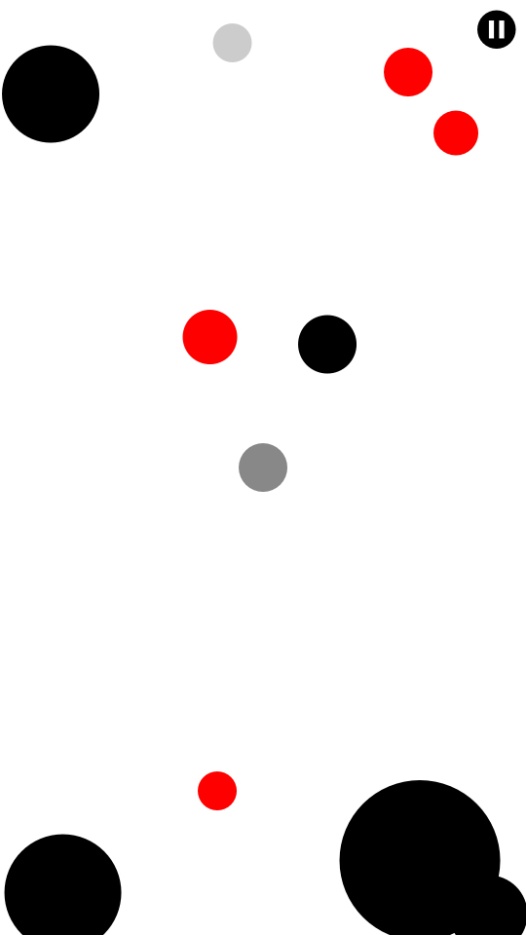
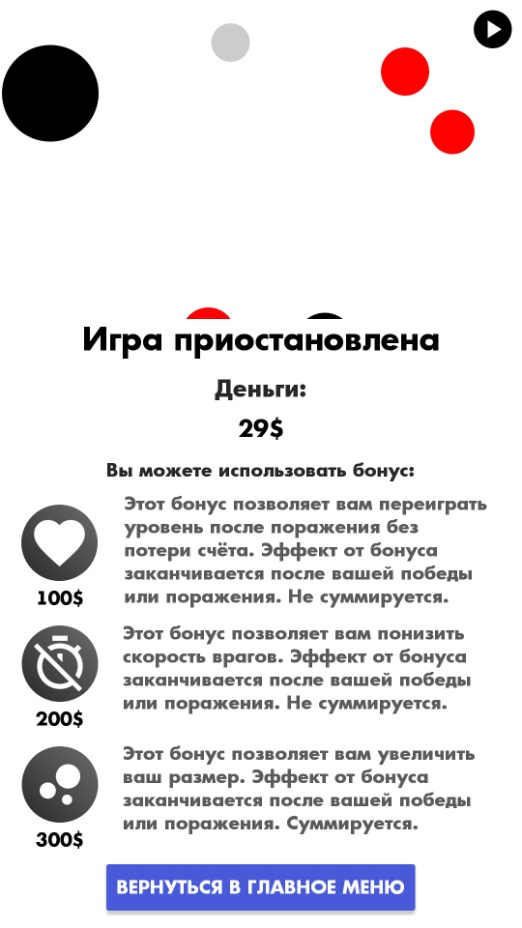
 

Рисунок 44. Игровой процесс Рисунок 45. Меню во время игры

После победы или поражения, появится сообщение, показанное на рисунках 46 и 47, с информацией о текущем счёте игрока, его наградах, а также с кнопками продолжения игры (в случае победы), начала новой игры (в случае поражения), возвратом в главное меню.

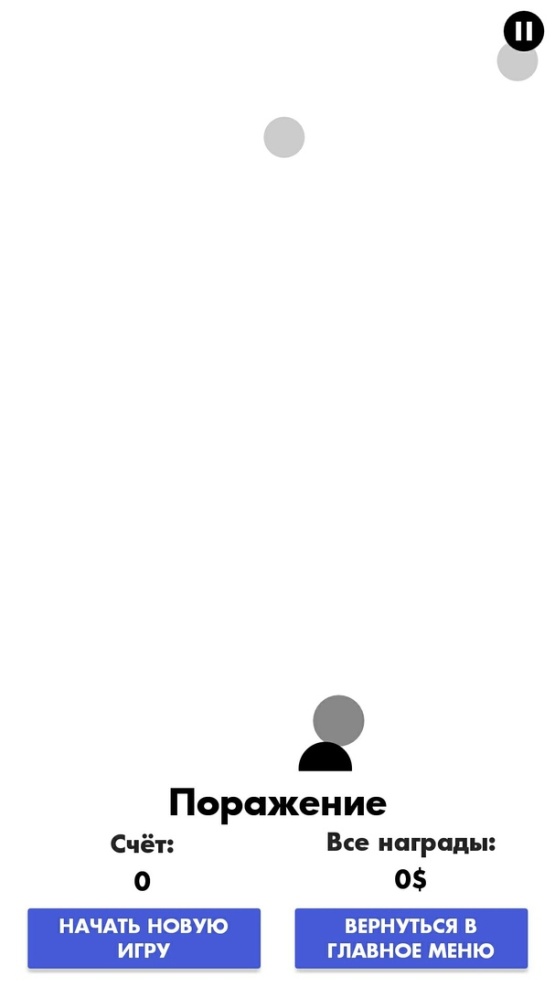
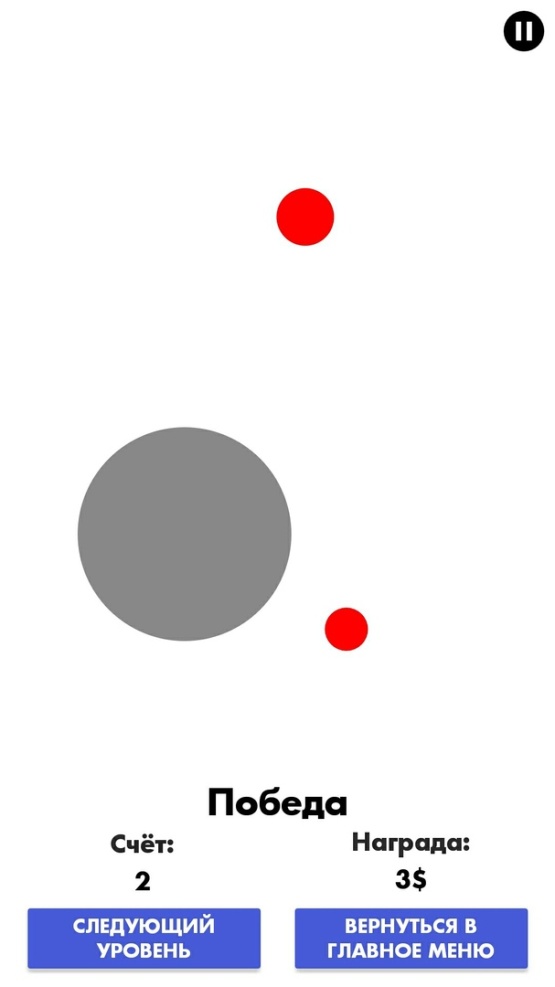
 

Рисунок 46. Сообщение при поражении Рисунок 47. Сообщение при победе

Заключение

Задачи и цели, поставленные в начале работы над мобильным приложением под ОС Android, были достигнуты в полном объеме, поскольку разработка прошла все этапы жизненного цикла ПО.

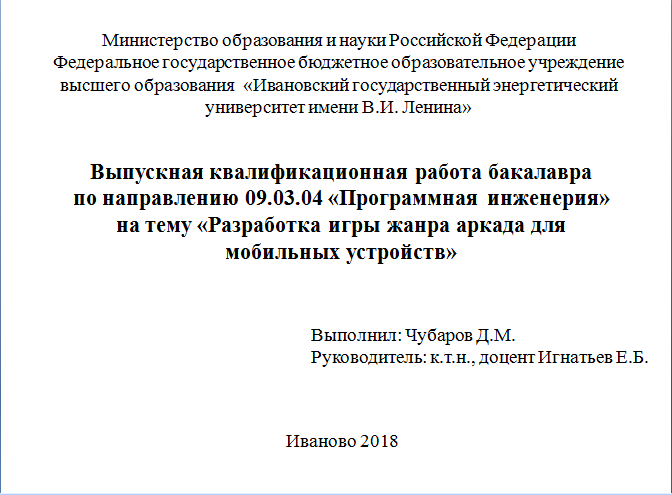
Также в процессе работы были получены практические навыки работы с IDE Android Studio и внешними библиотеками, расширяющими ее возможности, изучены особенности мобильной разработки и тестирования.

В дальнейшем вероятно расширение функционала и модернизация интерфейса данного приложения для всех экранов мобильных устройств и планшетов, а также его последующая публикация в PlayMarket.

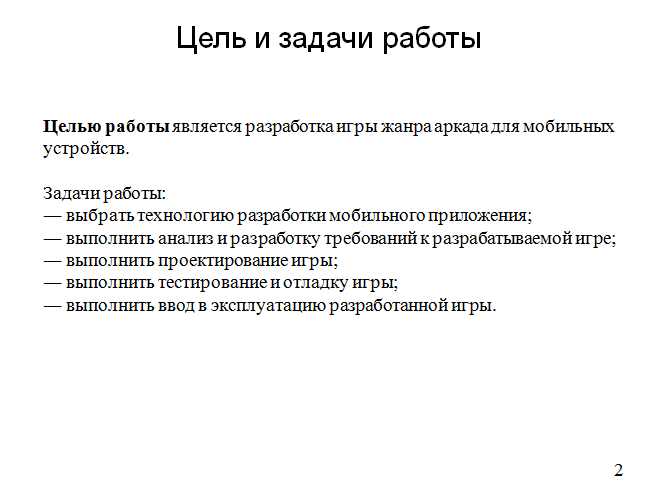
Список использованных источников

1. Цехнер Марио Программирование игр под Android; Питер - Москва, 2012. - 688 c.
2. Гультяев, А. К. Проектирование и дизайн пользовательского интерфейса / А.К. Гультяев, В.А. Машин. - М.: Корона-Принт, 2010. - 350 c.
3. Ларман, Крэг Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / Крэг Ларман. - М.: Вильямс, 2013. - 736 c.
4. Мюллер, Роберт Дж. Проектирование баз данных и UML / Мюллер Роберт Дж.. - М.: ЛОРИ, 2013. - **422** c.
5. Коберн, Алистер Современные методы описания функциональных требований к системам / Алистер Коберн. - Москва: **Машиностроение**, 2012. - 264 c.
6. Месарош, Джерард Шаблоны тестирования xUnit. Рефакторинг кода тестов / Джерард Месарош. - М.: Вильямс, 2016. - 832 c.

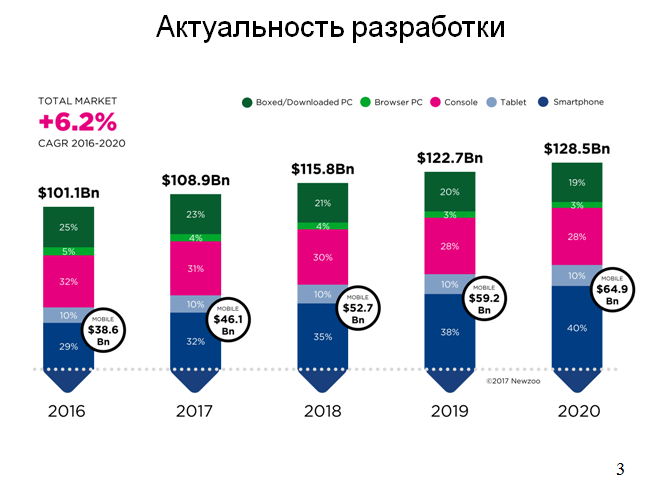
#### Приложение А. Графический материал доклада



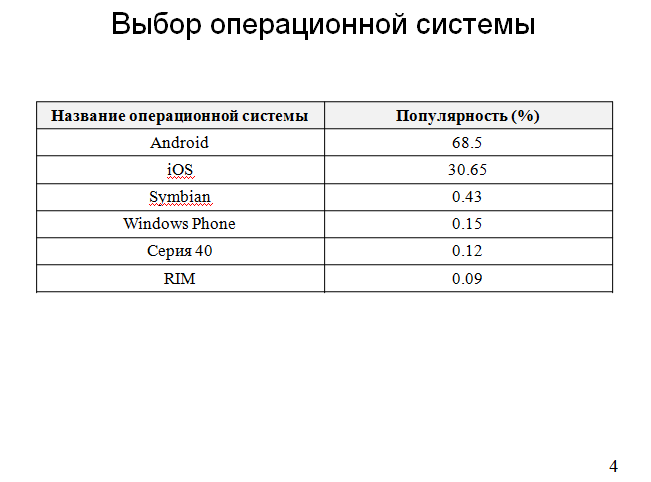
Слайд А1. Титульный лист



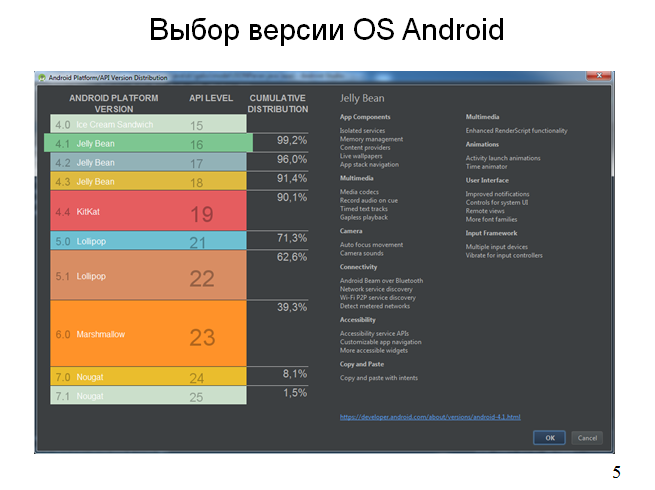
Слайд А2. Цели и задачи работы



Слайд А3. Актуальность разработки



Слайд А4. Выбор операционной системы



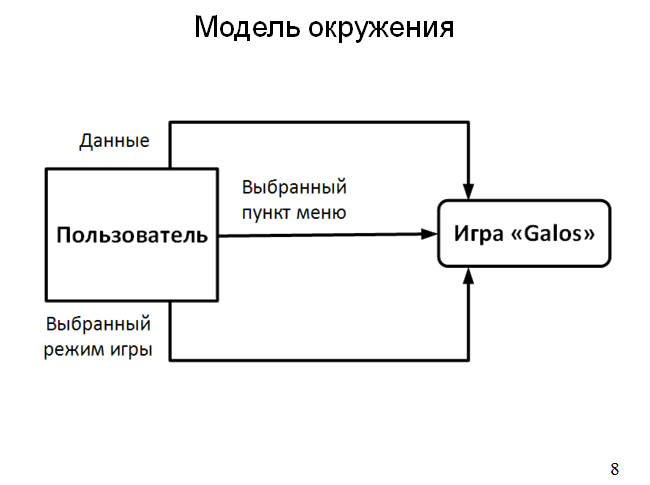
Слайд А5. Выбор версии OS Android



Слайд А6. Варианты использования



Слайд А7. Концептуальная модель



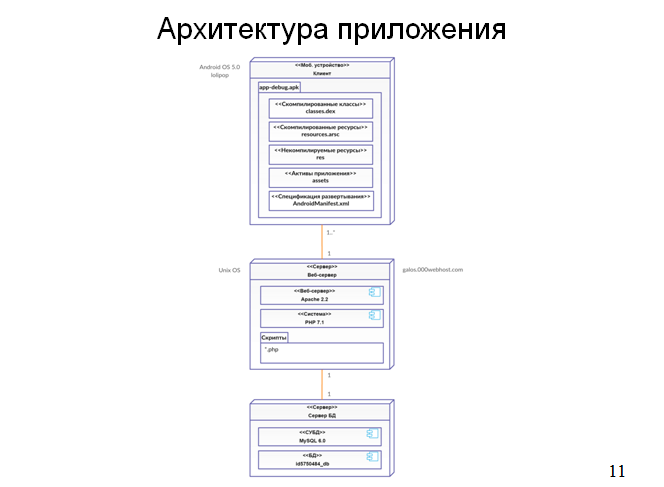
Слайд А8. Модель окружения



Слайд А9. Декомпозиция модели окружения (часть 1)



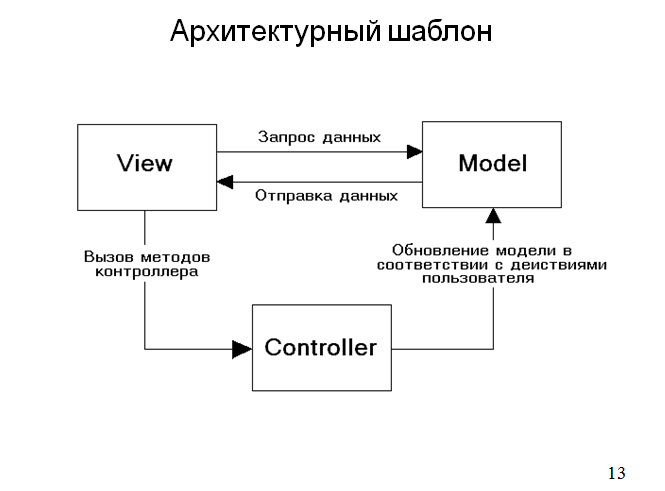
Слайд А10. Декомпозиция модели окружения (часть 2)



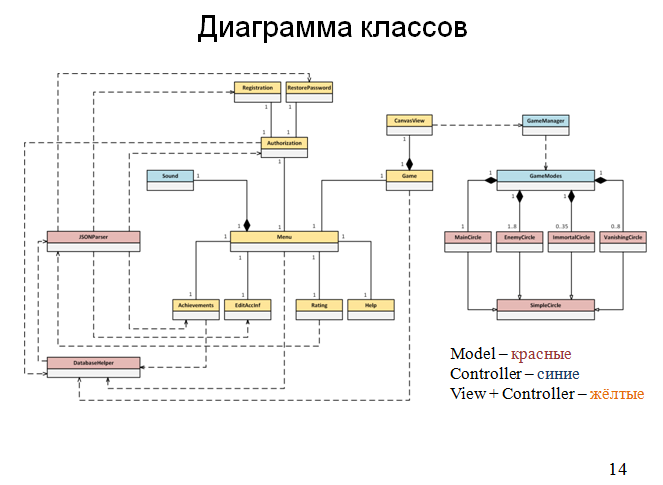
Слайд А11. Архитектура приложения



Слайд А12. Информационное обеспечение



Слайд А13. Архитектурный шаблон



Слайд А14. Диаграмма классов



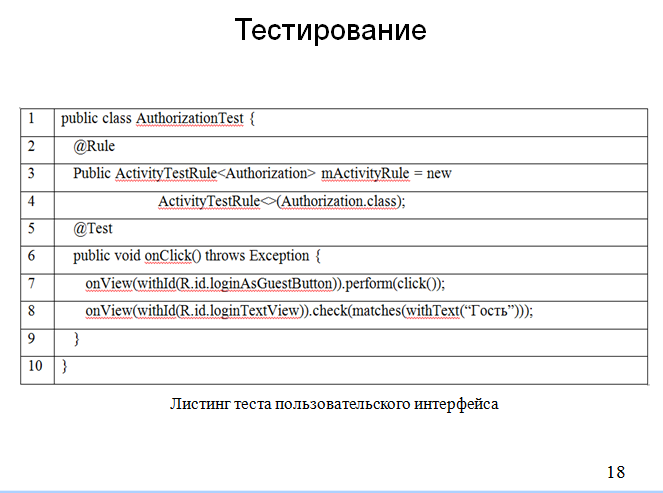
Слайд А15. Игровой процесс



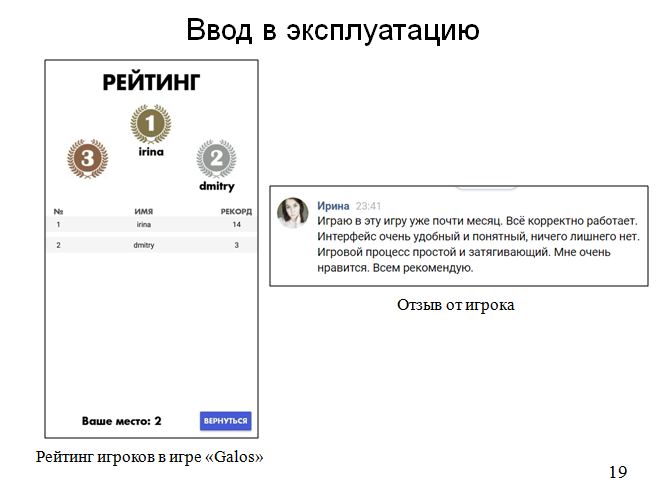
Слайд А16. Проект окон и их взаимодействие



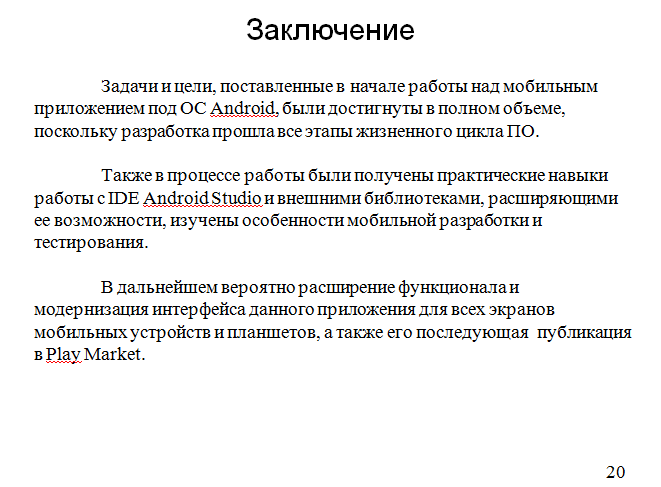
Слайд А17. Тестирование (часть 1)



Слайд А18. Тестирование (часть 2)



Слайд А19. Ввод в эксплуатацию



Слайд А20. Заключение

#### Приложение В. Отчёт о проверке ВКР на плагиат

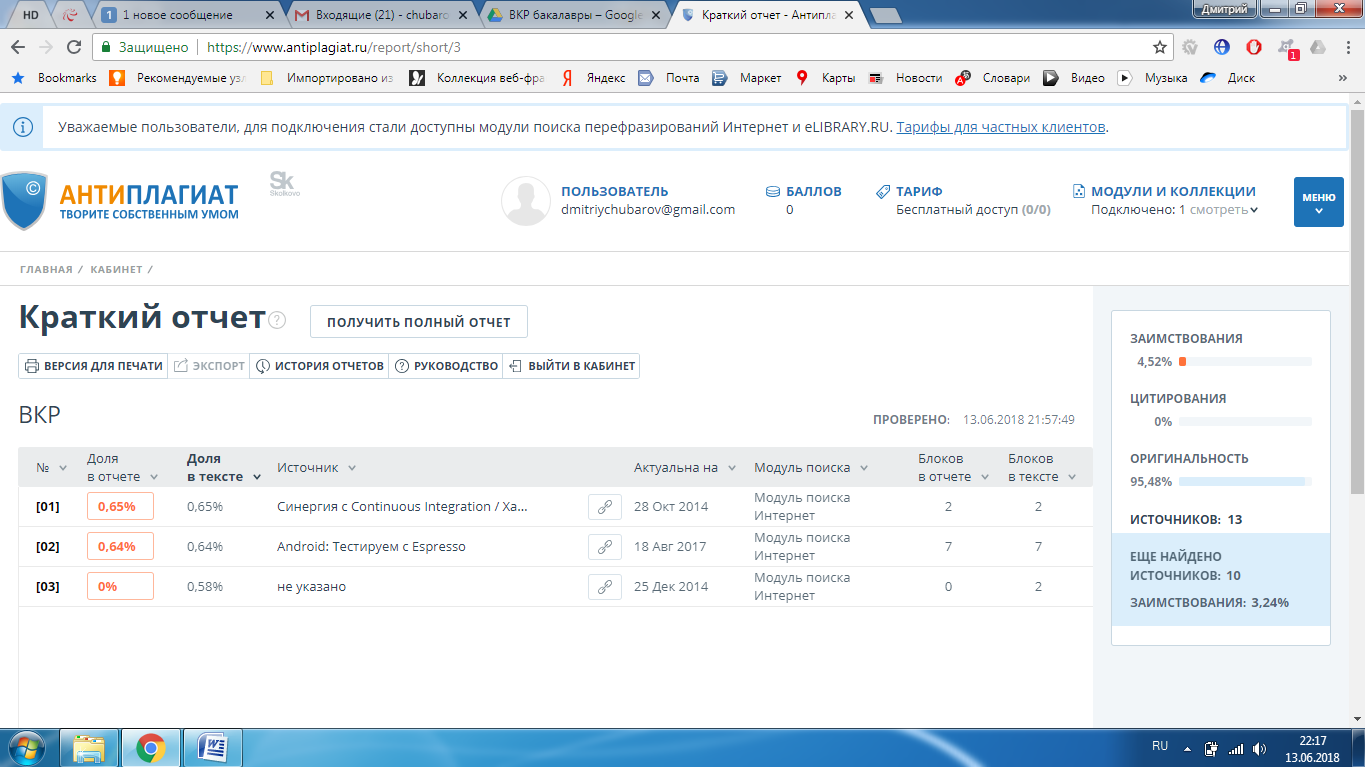


Рисунок В1. Отчёт о проверке ВКР на плагиат