Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледжимени Б.С. Галущака»

**разработка** **мобильного приложения для автосервиса**

Пояснительная записка к курсовому проекту

ПМ.01Разработка модулей программного обеспечения для компьютерных систем

МДК.01.03 Разработка мобильных приложений

НАТКиГ.201900.010.000ПЗ

Разработал:

студент группы ПР-21.106

Бушин А.Н.

2023

**Содержание**

[Введение 3](#_Toc136171097)

[1 Исследовательский раздел 5](#_Toc136171098)

[1.1 Описание предметной области 5](#_Toc136171099)

[1.2 Образ клиента 6](#_Toc136171100)

[1.3 Сценарии 6](#_Toc136171101)

[1.4 Сбор и анализ прототипов 7](#_Toc136171102)

[2 Проектирование приложения 12](#_Toc136171103)

[2.1 UI/UX дизайн проекта 12](#_Toc136171104)

[2.2 Выбор технологии, языка и среды программирования 17](#_Toc136171105)

[2.3 Описание элементов игры 17](#_Toc136171106)

[3 Тестирование 18](#_Toc136171107)

[3.1 Протокол тестирования дизайна приложения 18](#_Toc136171108)

[3.2 Протокол тестирования функционала приложения 18](#_Toc136171109)

[Заключение 19](#_Toc136171110)

[Библиография 20](#_Toc136171111)

[Приложение А 21](#_Toc136171112)

Введение

Руководитель исследования доктор Анна Корбетт говорит, что чем регулярнее люди разгадывают кроссворды или судоку, тем лучше работает мозг в области памяти, внимания и логики. Особенно явные улучшения были видны в скорости и аккуратности выполнения заданий. По оценке способности решать задачи ученые получили впечатляющие результаты, когда показатели мозговой деятельности людей с данным увлечением соотносились с оценочными мозговыми показателями человека, в среднем на 8 лет моложе биологического возраста, чего не наблюдалось у участников без подобного увлечения.

А также головоломки могут являться профилактикой по защите от развития болезни Альцгеймера или других форм деменции. Изучение о воздействиях головоломок на организм человека продолжаются на протяжении длительного времени и, хотя утверждать ещё рано, но на данном этапе исследования полученные результаты поддерживают предыдущие научные изыскания, указывающие на положительное влияние регулярной практики в решении задач со словами и цифрами в поддержании качественного функционирования мозга.

Есть множество традиционных игр, в которые играют «на бумаге», однако это ведет к вырубке лесов, а значит – вредит экологии. При этом мобильное устройство являются обязательным спутником человека в современном мире.

Также преимущество этой игры заключается в том, что в судоку хорошо играть в транспорте по дороге на работу или домой работать мозгу в области памяти, внимания и логики., а другое преимущество можно выделить как развитие различных качеств человека, например собранность, внимательность, а также игра помогает лучше

В связи со всем вышесказанным создание мобильных версий классических игр является актуальной задачей.

Целью курсового проекта является создание мобильной игры, которая основывается на развитии логического мышления, памяти и концентрации.

Задачами курсового проекта в связи с указанной целью являются:

* изучение предметной области темы;
* сбор и анализ прототипов;
* проектирование дизайна;
* разработка функционала приложения;
* тестирование.

Объект исследования – жанр игры головоломки.

Предмет исследования – особенности функционирования мобильных головоломок на примере игры «судоку».

# Исследовательский раздел

## Описание предметной области

Прообраз современного судоку, так называемые «магические квадраты» знали еще в древнем Китае. В Европе вспоминается что-то подобное в XVIII веке, когда слепой швейцарский математик Леонард Эйлер (Leonhard Euler) выяснил, что в матрице размером 9х9 каждый ряд и каждую колонку можно заполнить цифрами от 1 до 9 в определенном порядке и без повторения.

Особенностью является то, что головоломка – это маленькая проекция жизненных проблем, ведь если есть возможность справиться с головоломкой, то и решение из любой ситуации в жизни можно найти. Иногда играя простыми головоломками, можно понять, что все вокруг нас строится на тех же законах логики, поэтому с помощью таких заданий можно укрепить знания и улучшить жизнь.

Если говорить о правилах самой головоломки, то они совершенно просты. Игровое поле судоку разделено на 9 малых квадратов и состоит из восемьдесят одной клетки. Размеры поля составляют 9 столбцов и 9 строк. Некоторые клетки уже содержат числа. Задача игрока – заполнить остальные клетки, используя цифры от 1 до 9. Цель головоломки – заполнить свободные ячейки эти цифрами так, чтобы в каждой строке, в каждом столбце и в каждом малом квадрате 3х3, каждая цифра встречалась только один раз. Правильная головоломка имеет только одно решение.

При игре на бумаге нередко возникают ситуации, когда игрок поставил неправильную цифру, но не заметил это и продолжил играть, из-за чего в конце концов решение у него не получилось, а выяснить на каком моменте он ошибся очень сложно. Или при заполнении ручкой игрок поставил 2 одинаковые цифры в одну строку, а исправить не имеется возможности.

## Образ клиента

Клиентами являются все, кто интересуется головоломками и имеющие мобильное устройство.

Сама игра ориентирована на аудиторию людей старше десяти лет и подходит для русскоговорящих людей. Игра будет интересна для людей, которым нравится мыслить нестандартно и искать наиболее правильные решения, даже в самых сложных ситуациях.

## Сценарии

Молодой человек, сидя в очереди, решил скоротать время и взял почитать газету, в которой кроме новостей был раздел головоломок, перейдя к которому его особо увлекла игра судоку, что помогло ему скоротать время, решая её молодому человеку пришлось потрудиться, но в конце концов он справился с задачей и почувствовал удовольствие от решенной головоломки. И она ему так понравилось, что он решил установить мобильную версию этой игры на свой смартфон.

При игре в мобильную игру «PUBG» игрок Валера часто проигрывал и начинал нервничать. Но после очередного неудачного раунда ему выплыла реклама игры судоку, и он решил ее скачать. Поиграв в нее, он получил спокойствие, порядок и умиротворение путем решения головоломки. Это помогло ему освежить сознание и облегчить выполнение других задач с новой силой и энергией.

Во время обеденного перерыва Кирилл быстро поел и оставшееся свободное время он решил поиграть в судоку, в которое часто играл в детстве с отцом. Не имея возможности купить газету, Кирилл скачал мобильное приложение.

## Сбор и анализ прототипов

У мобильной версии судоку конкурентов достаточно много, однако многие имеют аналогичную структуру, логотип и даже дизайн. Поэтому в качестве прототипов были выбраны наиболее уникальные по дизайну приложения. В качестве платформы для поиска прототипов был выбран магазин Play market.

Судоку имеет не особо яркое цветовое решение приложения, что помогает сосредоточиться на самой игре. Фоновое изображение всех экранов приложения сделаны в едином стиле.

На рисунке 1 представлено изображение главной страницы «Судоку».

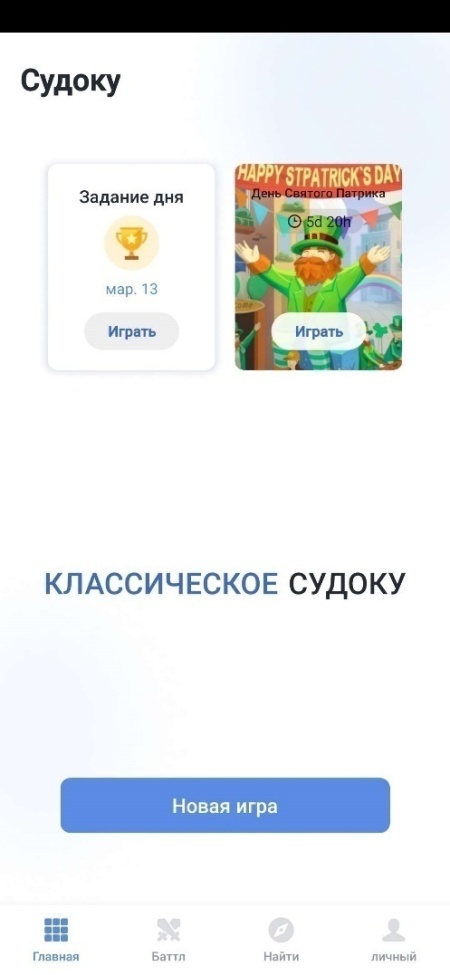


Рисунок 1 – Главная страница

У приложения есть шапка, информационная часть и навигационное меню в нижней части экрана. Шапку приложения составляет название судоку. В навигационном меню имеются кнопки для перехода между экранами «Главная», «Баттл», «Найти», «Личный». Внутри информационной части приложение предлагает выполнить ежедневное или тематическое задание.

На рисунке 2 представлена страница «Баттл»

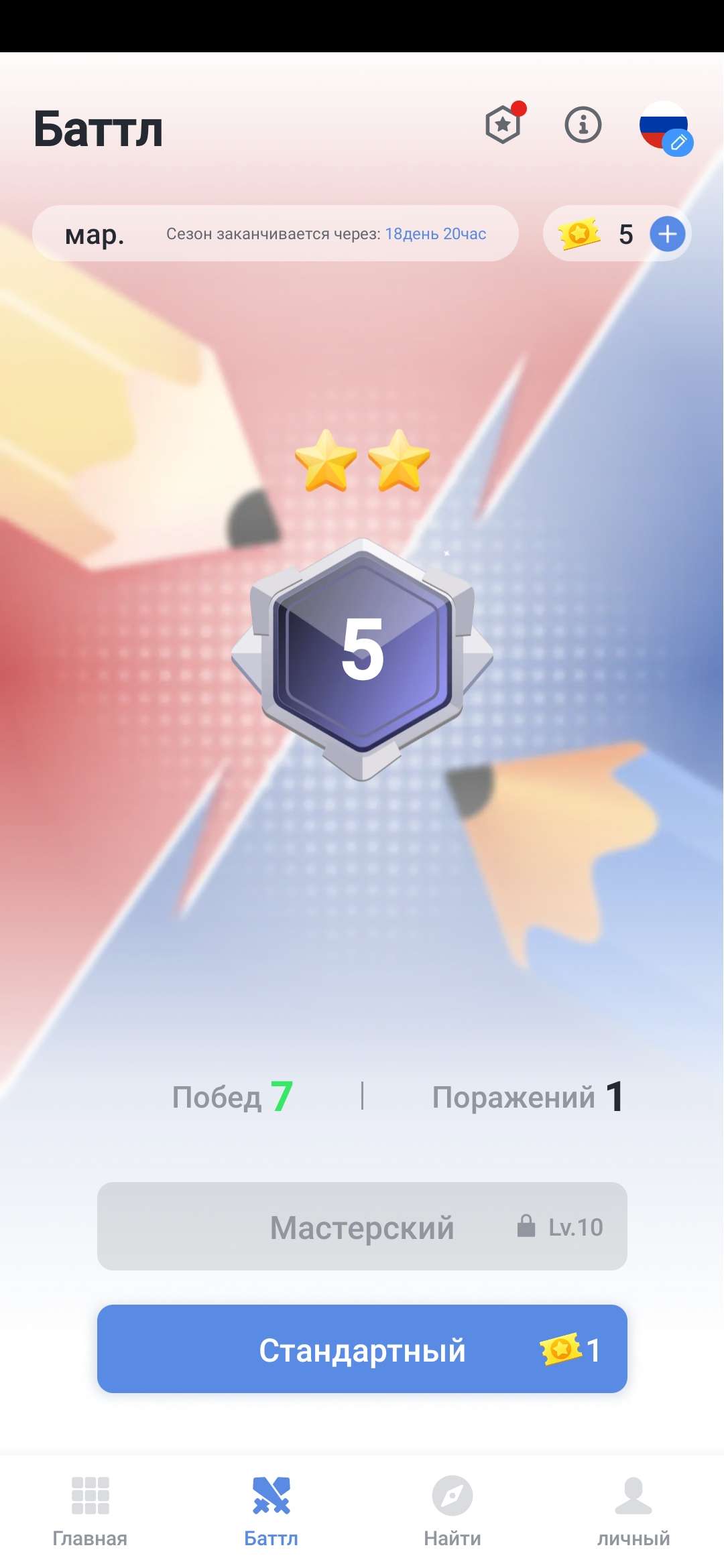


Рисунок 2 – Страница «Баттл»

На данной экране расположены кнопка «Стандартный», которая позволяет начать соревнование с другим игроком. Кнопка «Мастерский» позволяет посостязаться с более сильным противником. Так же есть отображение статистики с количеством побед и поражений.

На рисунке 3 представлена страница «Найти».

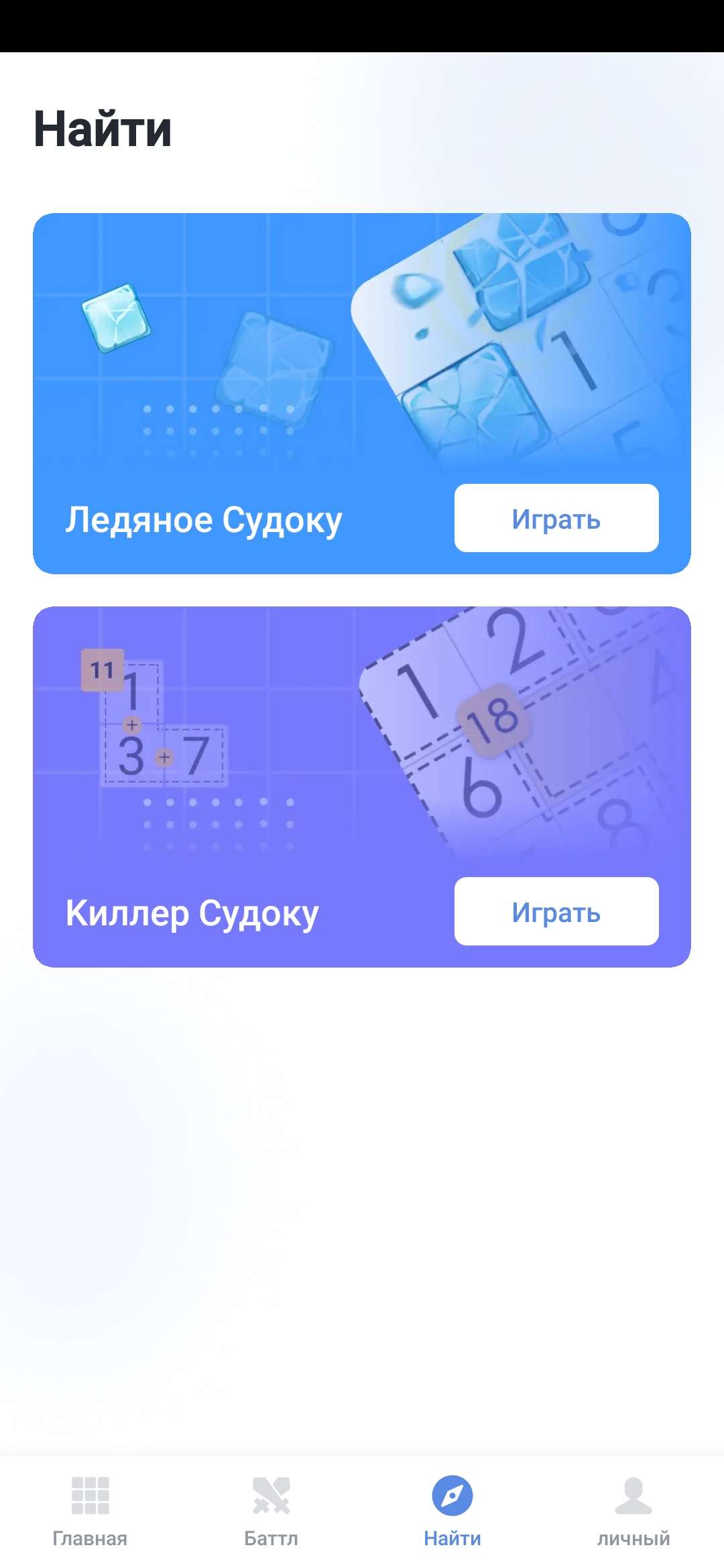


Рисунок 3 – Страница «Найти»

Внутри информационной части данной страницы можно найти несколько игр, что хоть и похожи на классическую судоку, но всё ещё являющиеся чем-то другим, что помогает разнообразить процесс решения головоломки.

На рисунке 4 представлена страница «Личный»

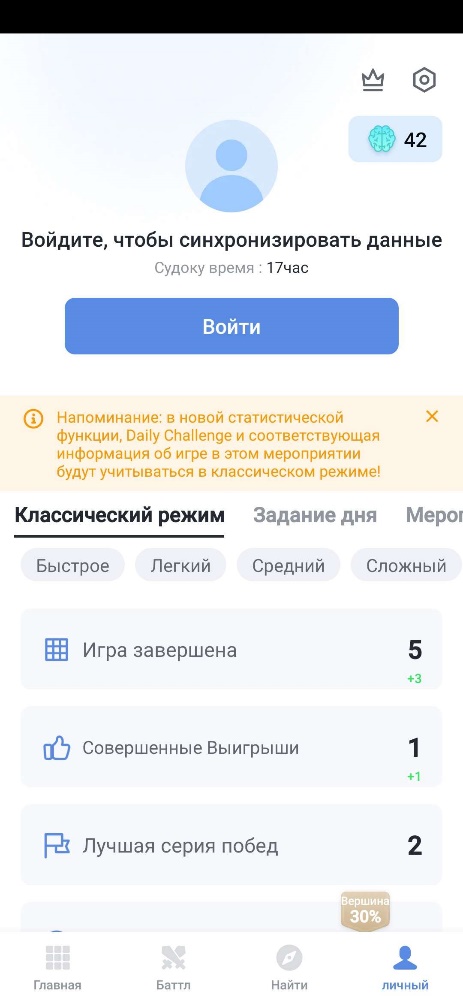


Рисунок 4 – Страница «Личный»

На данной странице можно войти к себе в личный кабинет, чтобы сохранить свои результаты и иметь возможность играть на других мобильных устройствах. Так же имеется возможность просмотреть свои результаты в разных режимах и на разных уровнях сложности.

На рисунке 5 представлено основное окно игры.

Изображение выглядит как календарь

Автоматически созданное описание

Рисунок 5 – Окно игры

В данном окне присутствуют поле для заполнения размером 9 на 9, цифры от 1 до 9, различный элементы игрового функционала, такие как «стереть», «быстрый карандаш», «намек», а также дополнительный функционал. Так же есть отображение счета, которое позволяет узнать свой текущей результат

Рассмотрев примеры данного приложения, можно сделать вывод:

Плюсы приложения:

* оснащенность главными функциями;
* соблюдена цветовая гамма, которая состоит из трёх основных цветов;
* простой и понятный интерфейс.

Минусы:

* при игре в разделе «Баттл» нельзя обойтись без интернета;
* большая загруженность различными функциями;
* нельзя отключить ежедневные задания, что мешают игре.

# Проектирование приложения

## UI/UX дизайн проекта

Дизайн проекта был разработан в программе Figma.

Для проекта были определены основные экраны:

* главное меню;
* игровое поле;
* меню с выбором сложности;
* настройки игры.

Цветовая гамма выбрана посредством изучения информации о значении цветов и восприятии их людьми. При игре в судоку человек должен расслабляться, для этого лучше всего подобрать спокойный цвет, который являлся бы не только успокаивающим, но и способствующий физическому и умственному расслаблению, что в дополнении создаёт атмосферу безопасности и доверия. Это помогло бы игроку сосредоточиться на игровом процессе и с комфортом играть.

Потому принято решение в разработке макета отдать предпочтение спокойным цветам: голубой, белый и темно синий. Но стоит заметить, что данные цвета не столь просты, ведь голубой цвет известен своим мягким успокаивающим эффектом, но при этом не вызывает усталости или торможения, как другие холодные цвета. Он снимает умственное напряжение, понижает температуру, расслабляет мышцы. Таким образом голубой цвет повышает сопротивляемость организма к различным стрессам.

Ниже на рисунке (Рисунок 5) представлен логотип судоку.

☜(ﾟヮﾟ☜)

Рисунок 5 – Логотип судоку

На логотипе изображена тематическая приложению иллюстрация.

На рисунке 6 представлена страница «Start menu» приложения.

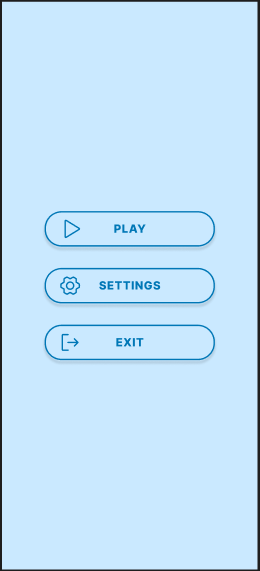


Рисунок 6 – Страница «Startmenu»

На данной странице мобильного приложения предоставлены кнопки «Play», при нажатии на которую переходит на экран выбора уровня сложности, «Setting», переходящий при нажатии в окно настроек игры, «Exit», позволяющий выйти из приложения.

На рисунке 7 представлена страница «Game» у приложения.

Изображение выглядит как календарь

Автоматически созданное описание

Рисунок 7 – Страница «Game»

На данной странице присутствует поле для заполнения, в ячейки которого вводятся цифры, клавиши от 1 до 9, с помощью которых происходит заполнение поля, а также стрелка для возврата в главное меню.

На рисунке 8 представлена страница «Settings».

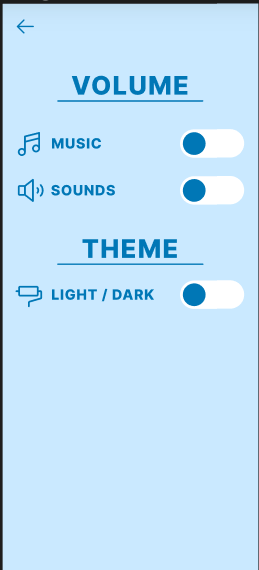


Рисунок 8 – Страница «Settings»

На данной странице находятся настройки приложения, такие как переключатели для музыки, звука и темы приложения.

На рисунке 9 представлена страница выбора сложности.

На этой странице отображается все возможные уровни сложности.

## Выбор технологии, языка и среды программирования

Средой программирования была выбрана платформа для разработки игр Unity, так как лучше всего подходит для разработки небольших проектов, таких как судоку. Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек. На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров.

Языком программирования является C#, так как он является единственным языком, которое поддерживает Unity. C# — объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумотакак язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

# Разработка мобильного приложения

## Разработка мультимедийного контента

В игре мультимедийный контент представлен анимацией и звуковыми эффектами, которые играют важную роль в создании атмосферы и взаимодействии с игроком.

Анимации воспроизводятся при определенных действиях игрока, например, при нажатии на ячейку или допущении ошибки. При нажатии на ячейку она выделяется, так же, как и строка со столбцом, на котором эта ячейка находится. Если в данной ячейки находится цифра, то она все цифры такого же значения подчеркиваются синим цветом. При заполнении пустой ячейки цифрой происходит звуковой эффект. При допущении ошибки ячейка выделяется красным, а так же воспроизводится соответствующий звуковой сигнал. Все это помогаем игроку лучше ориентриоваться в игре.

## Описание процедур и функций

В приложении разработаны следующие функции:

* переход из главного меню
* навигация по приложению
* выбор уровня сложности и запуск игры
* управление состоянием игры

На листинге 1 представлен код, отвечающий за заполнение поля, а также подсчет допущенных ошибок игроком.

Листинг 1 – Заполнение поля

private void Awake()

{

GameActive = true;

Global = this; // Устанавливаем глобальную ссылку на текущий экземпляр fillingUser

}

// Метод, вызываемый при старте игры

private void Start()

{

\_timer = GetComponent<Timer>(); // Получаем компонент Timer, присоединенный к этому объекту

\_timer.StartTImer(); // Запускаем таймер

}

// Метод выбора ячейки для заполнения

public void ChooseCell(Cell cell)

{

if (\_currentFilling != null)

CrossChange(\_currentFilling.X, \_currentFilling.Y, false); // Отменяем выбор предыдущей ячейки

\_currentFilling = cell; // Устанавливаем текущую выбранную ячейку

CrossChange(\_currentFilling.X, \_currentFilling.Y, true); // Вызываем метод для отметки выбранной ячейки

}

// Метод установки значения в выбранную ячейку

public void PutValue()

{

if (\_currentFilling != null)

{

if (\_currentFilling.IsHade)

{

if (\_currentFilling.Value == PutCell.CurrentValue.Value)

{

\_currentFilling.Active(); // Делаем ячейку активной

ChooseCell(\_currentFilling); // Вызываем метод для отметки выбранной ячейки

PutCell.CurrentValue.Decrease(); // Уменьшаем значение текущего заполняемого числа

\_activeCells[\_currentFilling.X, \_currentFilling.Y] = true; // Устанавливаем флаг активной ячейки

FinisdCheck(); // Проверяем, завершена ли игра

}

else

{

\_currentFilling.WrongNumber(PutCell.CurrentValue.Value); // Помечаем неправильное число в ячейке

\_countError++; // Увеличиваем счетчик ошибок

\_errors.text = $"Ошибки {\_countError}/3"; // Обновляем текст ошибок в пользовательском интерфейсе

if (\_countError == 3)

{

\_gameUi.GameOver(); // Завершаем игру

GameActive = false; // Устанавливаем флаг активности игры в false

}

}

}

}

}

// Метод проверки завершения игры

private void FinisdCheck()

{

foreach (var item in \_activeCells)

{

if (item == false)

{

print(item.ToString());

return;

}

}

\_gameUi.Win(); // Вызываем метод для победы в игре

GameActive = false; // Устанавливаем флаг активности игры в false

\_timer.StopTimer(); // Останавливаем таймер

}

// Метод для изменения визуального отображения ячеек при выборе

private void CrossChange(int x, int y, bool isChooise)

{

stateCell stateSomeValues = isChooise == true ? stateCell.Values : stateCell.NotChoose;

stateCell line = isChooise == true ? stateCell.Choose : stateCell.NotChoose;

for (int i = 0; i < 9; i++)

{

\_grid[i, y].ChangeBackGround(line); // Изменяем фон ячеек в строке

\_grid[x, i].ChangeBackGround(line); // Изменяем фон ячеек в столбце

}

if (\_currentFilling.IsHade == false)

{

for (int i = 0; i < 9; i++)

{

\_grid[x, y].\_cellsThisValues[i].ChangeBackGround(stateSomeValues); // Изменяем фон ячеек с возможными значениями

}

}

}

// Метод для сброса игры

public void ResetGame()

{

\_countError = 0; // Сбрасываем счетчик ошибок

\_errors.text = $"Ошибки {\_countError}/3"; // Обновляем текст ошибок в пользовательском интерфейсе

}

// Метод для установки сетки ячеек

public void SetGrid(Cell[,] grid, bool[,] active)

{

\_grid = grid; // Устанавливаем сетку ячеек

\_activeCells = active; // Устанавливаем флаги активности ячеек

}

На листинге 2 представлен код для хранения значения ячейки и его состояния.

Листинг 2 – Функционал ячейки

public void SetCellsValue(Cell[] newValues)

{

\_cellsThisValues = newValues; // Устанавливаем массив ячеек с возможными значениями

}

private void Start()

{

\_filling = fillingUser.Global; // Получаем ссылку на экземпляр fillingUser

}

public virtual void SetValue(int value, int x, int y)

{

X = x; // Устанавливаем координату X ячейки

Y = y; // Устанавливаем координату Y ячейки

Value = value; // Устанавливаем значение ячейки

LogValue(); // Выводим значение ячейки в лог

}

public void Hide()

{

IsHade = true; // Устанавливаем флаг, указывающий на то, что ячейка скрыта

\_valueText.gameObject.SetActive(false); // Делаем текстовый объект неактивным

}

public void Active()

{

IsHade = false; // Устанавливаем флаг, указывающий на то, что ячейка активна

\_valueText.gameObject.SetActive(true); // Делаем текстовый объект активным

\_valueText.text = Value.ToString(); // Устанавливаем текстовому объекту значение ячейки

\_isError = false; // Сбрасываем флаг ошибки

ChangeBackGround(stateCell.Choose); // Изменяем фон ячейки на состояние выбранной

}

public void WrongNumber(int value)

{

\_valueText.gameObject.SetActive(true); // Делаем текстовый объект активным

\_valueText.text = value.ToString(); // Устанавливаем текстовому объекту значение

\_isError = true; // Устанавливаем флаг ошибки

ChangeBackGround(stateCell.Error); // Изменяем фон ячейки на состояние ошибки

}

public void ChangeBackGround(stateCell state)

{

if (state == stateCell.Choose && !\_isError)

\_backGround.color = \_colorChooise; // Изменяем цвет фона на цвет выбранной ячейки

else if (state == stateCell.NotChoose && !\_isError)

\_backGround.color = Color.clear; // Изменяем цвет фона на прозрачный

else if (state == stateCell.Values && IsHade == false && !\_isError)

{

\_backGround.color = \_colorValues; // Изменяем цвет фона на цвет ячейки с возможными значениями

\_valueText.gameObject.SetActive(true); // Делаем текстовый объект активным

}

else if(state == stateCell.Error)

{

\_backGround.color = \_colorError; // Изменяем цвет фона на цвет ячейки с ошибкой

}

}

public void OnMouseDown()

{

if (\_filling.GameActive)

{

\_filling.ChooseCell(this); // Вызываем метод выбора ячейки в fillingUser

}

}

private void LogValue()

{

\_valueText.text = Value.ToString(); // Выводим значение ячейки в текстовый объект

}

На листинге 3 представлен код, который отвечает за создание поля и заполнение его случайными числами, придерживаясь определенных правил.

Листинг 3 – Генерация поля

[SerializeField] private GameObject \_cellPrefab; // Префаб ячейки судоку

[SerializeField] private GameObject \_cellPutPrefab;

const int n = 3;

public static int[,] \_map = new int[n \* n, n \* n]; // Карта чисел судоку

private static Cell[,] \_board = new Cell[9, 9]; // Двумерный массив ячеек судоку

private bool[,] \_cellActive = new bool[9, 9]; // Массив, указывающий, активна ли каждая ячейка судоку

private List<Cell> \_hideCell = new List<Cell>(); // Список скрытых ячеек

private List<Cell> \_hideCellSort = new List<Cell>(); // Отсортированный список скрытых ячеек

private Vector2 \_position; // Позиция начала сетки

private fillingUser \_filling; // Класс для заполнения пользовательскими значениями

private void Start()

{

\_filling = fillingUser.Global; // Получаем ссылку на объект fillingUser

\_position -= Vector2.right \* (((float)Mathf.Sqrt(\_board.Length) \* \_cellPrefab.transform.localScale) / 2); // Позиция начала сетки по горизонтали

\_position += Vector2.up \* (((float)Mathf.Sqrt(\_board.Length) \* \_cellPrefab.transform.localScale) / 2); // Позиция начала сетки по вертикали

\_position += Vector2.one / 2; // Смещение позиции

GenerateMap(); // Генерируем карту чисел судоку

\_filling.SetGrid(\_board, \_cellActive); // Устанавливаем сетку и активность ячеек в fillingUser

}

private void GenerateMap()

{

for (int i = 0; i < n \* n; i++)

{

for (int j = 0; j < n \* n; j++)

{

\_cellActive[i, j] = true; // Устанавливаем активность ячейки в true

\_map[i, j] = (i \* n + i / n + j) % (n \* n) + 1; // Заполняем карту числами судоку

}

}

for (int i = 0; i < 40; i++)

{

ShuffleMap(Random.Range(0, 5)); // Перемешиваем карту чисел судоку

}

Print(); // Выводим сгенерированную сетку на экран

HideCells(); // Скрываем некоторые ячейки судоку

}

private void Print()

{

Vector2 lovalPosition = \_position;

for (int i = 0; i < 9; i++)

{

lovalPosition.x = \_position.x;

for (int y = 0; y < 9; y++)

{

Cell cell = Instantiate(\_cellPrefab).GetComponent<Cell>(); // Создаём ячейку судоку

\_board[i, y] = cell; // Добавляем ячейку в двумерный массив

\_hideCellSort.Add(cell); // Добавляем ячейку в список скрытых ячеек

cell.transform.position = lovalPosition; // Устанавливаем позицию ячейки

cell.SetValue(\_map[i, y], i, y); // Устанавливаем значение и позицию ячейки

lovalPosition += Vector2.right \* (Vector2)\_cellPrefab.transform.localScale; // Смещаем позицию по горизонтали

}

lovalPosition += Vector2.down \* (Vector2)\_cellPrefab.transform.localScale; // Смещаем позицию по вертикали

}

\_hideCellSort = \_hideCellSort.OrderBy(p => p.Value).ToList(); // Сортируем список скрытых ячеек по значению

for (int i = 0; i < \_hideCellSort.Count;)

{

Cell[] values = new Cell[9];

for (int y = 0; y < 9; y++)

{

values[y] = \_hideCellSort[i + y];

}

for (int x = i; x < 9 + i; x++)

{

\_hideCellSort[x].SetCellsValue(values);

}

i += 9;

}

}

private void ShuffleMap(int i)

{

switch (i)

{

case 0:

MatrixTransposition();

break;

case 1:

SwapRowsInBlock();

break;

case 2:

SwapColumnsInBlock();

break;

case 3:

SwapBlocksInRow();

break;

case 4:

SwapBlocksInColumn();

break;

default:

MatrixTransposition();

break;

}

}

private void SwapBlocksInColumn()

{

var block1 = Random.Range(0, n);

var block2 = Random.Range(0, n);

while (block1 == block2)

block2 = Random.Range(0, n);

block1 \*= n;

block2 \*= n;

for (int i = 0; i < n \* n; i++)

{

var k = block2;

for (int j = block1; j < block1 + n; j++)

{

var temp = \_map[i, j];

\_map[i, j] = \_map[i, k];

\_map[i, k] = temp;

k++;

}

}

}

// Остальные методы SwapRowsInBlock(), SwapColumnsInBlock(), SwapBlocksInRow(), SwapBlocksInColumn() и MatrixTransposition() аналогичны SwapBlocksInColumn()

// Они отвечают за различные операции перемешивания чисел на карте судоку

private void HideCells()

{

for (int i = 0; i < \_map.Length / 2;)

{

var x = Random.Range(0, 9);

var y = Random.Range(0, 9);

if (\_cellActive[x, y] == true)

{

\_cellActive[x, y] = false;

\_board[x, y].Hide();

\_hideCell.Add(\_board[x, y]);

i++;

}

}

\_hideCell = \_hideCell.OrderBy(p => p.Value).ToList(); // Сортируем список скрытых ячеек по значению

int localValue = 0;

int countValue = 0;

Vector2 localPosition = \_position;

localPosition += Vector2.down \* 12;

for (int i = 0; i < \_hideCell.Count; i++)

{

if (localValue == 0)

{

localValue = \_hideCell[i].Value;

}

if (localValue == \_hideCell[i].Value)

{

countValue++;

}

else

{

CreatePutCell(localValue, countValue, localPosition);

countValue = 1;

localValue = \_hideCell[i].Value;

localPosition += Vector2.right \* (Vector2)\_cellPrefab.transform.localScale;

}

}

CreatePutCell(localValue, countValue, localPosition);

}

private void CreatePutCell(int value, int count, Vector2 position)

{

PutCell put = Instantiate(\_cellPutPrefab).GetComponent<PutCell>(); // Создаём пустую ячейку для заполнения пользователем

put.Inicialize(value, count);

put.transform.position = position; // Устанавливаем позицию пустой ячейки

}

}

На листинге 4 представлен код, отвечающий за функционал меню игры.

Листинг 4 - Меню

[SerializeField] private AudioSource \_music; // Ссылка на компонент AudioSource для управления музыкой

[SerializeField] private Sprite \_OffSound, \_OnSound; // Ссылки на спрайты для отображения состояния звука (вкл/выкл)

[SerializeField] private GameObject \_windowSettings; // Ссылка на игровой объект окна настроек

[SerializeField] private Toggle \_musicToggle, \_soundToggle, \_darkThemeToggle; // Ссылки на компоненты Toggle для управления настройками звука и темы

private const int \_level = 1; // Уровень, который будет загружаться при нажатии кнопки "Play"

private float \_soundVolum; // Громкость звука

private Theme \_theme; // Ссылка на скрипт Theme для управления темой

private void Awake()

{

var oneMusic = FindObjectsOfType<AudioSource>(); // Поиск всех компонентов AudioSource на сцене

\_music = oneMusic[0]; // Присваиваем первый найденный компонент AudioSource переменной \_music

\_soundVolum = \_music.volume; // Запоминаем текущую громкость звука

if (oneMusic.Length > 1)

{

Destroy(oneMusic[oneMusic.Length - 1].gameObject); // Удаляем лишние компоненты AudioSource

}

DontDestroyOnLoad(\_music); // Запрещаем уничтожение компонента AudioSource при загрузке новой сцены

}

private void Start()

{

\_theme = Theme.Global; // Получаем ссылку на скрипт Theme из глобального объекта

\_darkThemeToggle.isOn = PlayerPrefs.GetInt("Theme", 0) == 0 ? false : true; // Устанавливаем состояние переключателя темы на основе сохраненного значения в PlayerPrefs

}

public void Sound()

{

if (\_soundVolum == 0)

{

\_soundVolum = 1; // Включаем звук, если он был выключен

}

else

{

\_soundVolum = 0; // Выключаем звук, если он был включен

}

\_music.volume = \_soundVolum; // Применяем новую громкость звука

PlayerPrefs.SetFloat("Music", \_soundVolum); // Сохраняем новую громкость звука в PlayerPrefs

}

public void Play()

{

SceneManager.LoadScene(\_level); // Загружаем указанный уровень

}

public void Exit()

{

Application.Quit(); // Закрываем приложение

}

public void OpenSettings()

{

\_windowSettings.SetActive(true); // Открываем окно настроек

}

public void ThemeChange()

{

PlayerPrefs.SetInt("Theme", \_darkThemeToggle.isOn ? 1 : 0); // Сохраняем состояние темы в PlayerPrefs

\_theme.SetTheme(\_darkThemeToggle.isOn); // Применяем новую тему

}

public void Back()

{

\_windowSettings.SetActive(false); // Закрываем окно настроек

}

На листинге 5 представлен код, отвечающий за функционал клавиш, позволяющих заполнять игровое поле.

Листинг 5 – Функционал клавиш

public static PutCell CurrentValue { private set; get; } // Ссылка на текущую ячейку PutCell, доступная только для чтения

public int Value { private set; get; } // Значение ячейки

[SerializeField] private TextMesh \_valueText; // Ссылка на компонент TextMesh для отображения значения ячейки

[SerializeField] private Color \_valueZero; // Цвет для значения ноль

[SerializeField] private SpriteRenderer \_spriteRenderer; // Ссылка на компонент SpriteRenderer для отображения спрайта ячейки

[SerializeField] private TextMesh \_countText; // Ссылка на компонент TextMesh для отображения количества ячеек

private fillingUser \_filling; // Ссылка на скрипт fillingUser для управления заполнением ячеек

private int \_count; // Количество ячеек

private void Start()

{

\_filling = fillingUser.Global; // Получаем ссылку на скрипт fillingUser из глобального объекта

}

public virtual void Inicialize(int value, int count)

{

Value = value; // Устанавливаем значение ячейки

\_count = count; // Устанавливаем количество ячеек

LogValue(); // Обновляем отображение значения и количества ячеек

}

public void Decrease()

{

\_count--; // Уменьшаем количество ячеек

if(\_count == 0)

{

gameObject.SetActive(false); // Если количество ячеек стало нулевым, деактивируем объект ячейки

}

\_countText.text = \_count.ToString(); // Обновляем отображение количества ячеек

}

public void OnMouseDown()

{

if (\_filling.GameActive)

{

CurrentValue = this; // Устанавливаем текущую ячейку как выбранную ячейку PutCell

\_filling.PutValue(); // Вызываем метод PutValue() из скрипта fillingUser для размещения значения ячейки

}

}

private void LogValue()

{

\_countText.text = \_count.ToString(); // Обновляем отображение количества ячеек

\_valueText.text = Value.ToString(); // Обновляем отображение значения ячейки

}

# Тестирование

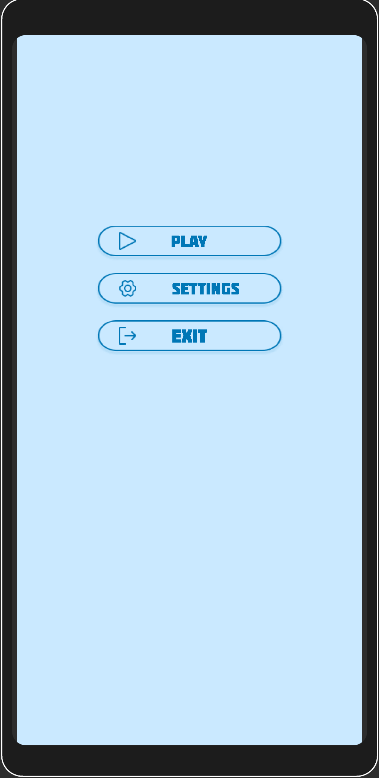
## Протокол тестирования дизайна приложения

Тестирование дизайна приложения проводится на самом минимальном (Android SDK 24) и на более позднем (Android SDK 33) с различной диагональю экранов для проверки разметки страниц и вёрстки приложения.

Примеры проверок отображения элементов на экране представлены на рисунках 9–12

Проверка была проведена на:

* оптимальный размер кнопок;
* читабельный размер шрифта;
* приятное глазу цветовое сочетание;
* корректное расположение элементов;
* отсутствие графических ошибок;
* устойчивость к функциональным ошибкам;

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Экраны главного меню

Главный экран отображается правильно. Масштаб элементов сохранен. Сами элементы остаются на своем месте.

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Экраны выбора уровня сложности

На экране выбора сложности так же все корректно отображается. Все элементы расположены на своем месте и имеет нормальные размеры.

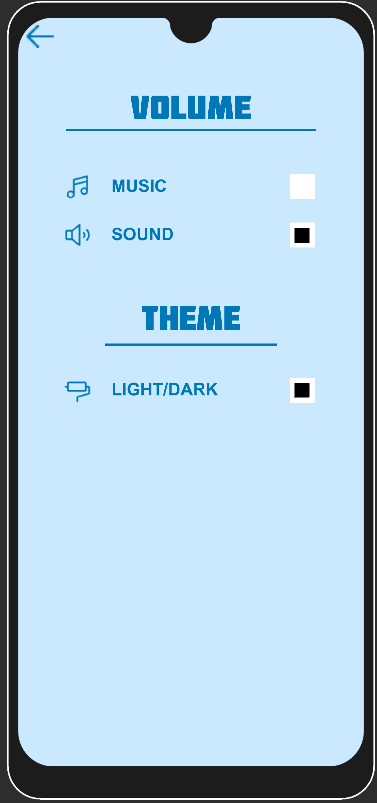
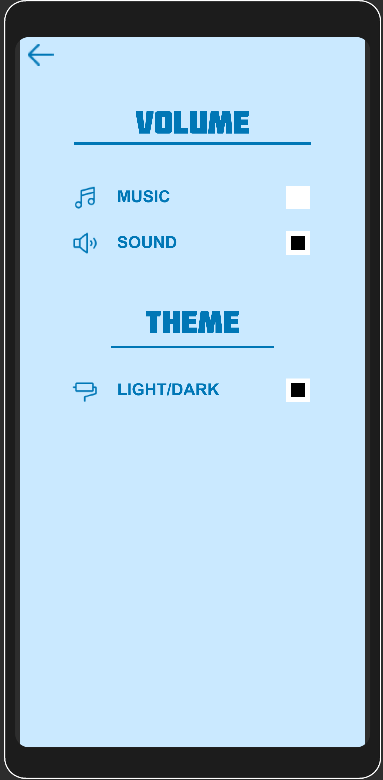


Рисунок 11 – Экраны настроек игры

Экрана настроек отображаются так же корректно. Его элементы остаются на своих местах и не растягиваются или сужаются. Текст не растянут и не выходит за пределы экрана.

Изображение выглядит как календарь

Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описание

Рисунок 12 – Экраны игрового процесса приложения

Экран игрового процесса так же не обнаружено ошибок. Игровое поле отображается одинаково на обоих экранах. Элементы поля выглядят корректно, без смещений.

## Протокол тестирования функционала приложения

Так же необходимо проверить функционал приложения. Для этого, для каждой функции были разработаны TestCase.

В таблице 1 представлено тестирование функции вывода видеоролика на экран.

Таблица 1 -

Заключение

В ходе разработки мобильного приложения для видеохостинга учтены предпочтения и потребности пользователей, в формате коротких видеороликов. Функция ведения списка просмотренного контента успешно реализована, позволяя пользователям отслеживать свои предпочтения и легко вернуться к уже просмотренному контенту.

Интерфейс приложения разработан с учетом удобства использования и минимального количества действий для достижения пользовательской цели. Основные экраны спроектированы таким образом, чтобы не перегружать пользователей лишней информацией, и добавлены полезные функции, такие как просмотр трейлеров и возможность создания собственных списков просмотренного и избранного контента.

Важным элементом концепции приложения является возможность просмотра коротких видеороликов, которая позволяет пользователям сократить время в дороге на работу, учебу или в перерывах между делами. Это удобный способ получать информацию на ходу.

Поддержка списка избранного контента также оказалась значимой для пользователей, предоставляя удобный способ отслеживания и доступа к контенту, который им особенно нравится.

Важным этапом в разработке приложения было тестирование дизайна, которое подтвердило его корректность и успешное функционирование. Это гарантирует, что разработанное приложение соответствует поставленным целям и требованиям.

В целом разработка мобильной версии видеохостинга позволила создать функциональное и удобное приложение, учитывающее потребности пользователей. Проделанная работа успешно достигла поставленной цели курсового проекта и является основой для дальнейшего развития и улучшения приложения.

Библиография

1. AndroidDevelopers [электронный ресурс]: ДокументацияAndroidStusio – https://developer.android.com/docs/
2. HelpCenter [электронныйресурс]: ДокументацияJava: https://docs.oracle.com/en/java/
3. Автостат[электронный ресурс]:Структура российских автовладельцев по возрасту и маркамhttps://www.autostat.ru/infographics/40710/
4. Firebase [электронныйресурс]: Документация Firebase - https://firebase.google.com/docs?hl=uk
5. Material Design [электронныйресурс]: BottomNavigation: https://material.io/components/bottom-navigation/android
6. Git[электронныйресурс]: ДокументацияGit: https://git-scm.com/
7. Figma[электронныйресурс]: Прототипирование проектовhttps://www.figma.com/