Министерство образования Новосибирской области ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

**Разработка мобильного приложения «Облачное хранение фотографий и их создание»**

Пояснительная записка к курсовому проекту

ПМ.01Разработка модулей программного обеспечения для компьютерных систем

МДК 01.03 Разработка мобильных приложений

НАТКиГ.200900.010.000ПЗ

Выполнил: Плахотный Г.А.

Новосибирск

2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc117529920)

[1 Исследовательский раздел 5](#_Toc117529921)

[1.1 Описание предметной области 5](#_Toc117529922)

[1.2 Образ клиента 5](#_Toc117529923)

[1.3 Сценарии 5](#_Toc117529924)

[1.4 Сбор и анализ прототипов 6](#_Toc117529925)

[2 Проектирование приложения 8](#_Toc117529926)

[2.1 UI/UX дизайн проекта 8](#_Toc117529927)

[3 Разработка мобильного приложения 18](#_Toc117529928)

[3.1 Разработка базы данных 19](#_Toc117529929)

[3.2 Разработка мультимедийного контента 19](#_Toc117529930)

[3.3 Описание используемых плагинов 30](#_Toc117529931)

[3.4 Описание разработанных процедур и функций 31](#_Toc117529932)

[4 Тестирование 37](#_Toc117529933)

[4.1 Протокол тестирования дизайна приложения 37](#_Toc117529934)

[4.2 Протокол тестирования функционала приложения 37](#_Toc117529935)

[ЗАКЛЮЧЕНИЕ 40](#_Toc117529936)

[СПИСОК ИСТОЧНИКОВ 41](#_Toc117529937)

## ВВЕДЕНИЕ

Целью курсового проекта является создание мобильно приложения для ведения дневника здоровья.

Задачами курсового проекта являются:

* изучение предметной области темы;
* определение целевой аудитории;
* определения назначения приложения путем создания сценариев использования;
* рассмотрение прототипов для выявления необходимых функций;
* проектирование приложения;
* разработка мобильного приложения;
* тестирование приложения.

Объект исследования – приложения-дневники.

Предмет исследования – изучение функционала и принципов функционирования приложений-дневников.

# Исследовательский раздел

## 1.1 Описание предметной области

«Платформер» – один из популярных игровых жанров, основной чертой игрового процесса является прыгание по платформам. Противники в данном жанре обладают примитивным искусственным интеллектом, стремясь максимально приблизиться к игроку, либо не обладают им вовсе, перемещаясь по круговой дистанции, совершая повторяющиеся действия или не совершая их. Соприкосновение с противником обычно отнимает жизненные силы героя или вовсе убивает его. Иногда противник может быть нейтрализован или прыжком на голову, или из оружия, если им обладает герой.

Игра представляет собой мобильную игру для одного человека, цель игрока пройти как можно дальше по уровню, уворачиваясь от врагов. Поле игры представляет собой холст, на котором генерируются платформы в виде облаков. При попадании на платформу персонаж отскакивает, таким образом игрок продвигается выше. Так же на уровне появляются радужные и серые облака, которые могут служить как помощью игроку, так и вредом для него. При попадании на радужное облако игрок отскакивает выше, чем от обычного, а серое облако двигается в произвольном порядке и другими особыми свойствами не обладает. Для усложнения достижения желаемого результата появляются враги в виде призраков, цель игрока как можно быстрее увернуться от них. При касании с врагом игрок умирает и возвращается в самое начало уровня. В игре представлена механика достижений, при выполнении определённого действия в отдельном окне появляется метка о том, что это достижение выполнено и получено. В этом же окне можно посмотреть свой лучший результат по прохождению уровня. Результат измеряется в очках, получаемых при прохождении уровня.

## 1.2 Образ клиента

Данный проект будет полезен для тех, кто хочет немного отдохнуть и приятно провести время, за быстрой и динамичной игрой. Для игры не требуются какие-то специфические знания компьютера, поэтому играть в неё сможет абсолютно кто угодно.

## 1.3 Сценарии

### 1.3.1 Первый сценарий

Екатерина - офисный работник, которая проводит большую часть своего времени за компьютером. Чтобы взять небольшой перерыв и расслабиться, она запускает игру на своем телефоне. После небольшой игровой сессии она возвращается к работе с более ясной и сосредоточенной умственной активностью.

### 1.3.2 Второй сценарий

Андрей - путешественник, ожидающий свой рейс в аэропорту. Чтобы скоротать время до посадки, он запускает платформер на своем планшете. Побитие своего рекорда помогает Андрею отвлечься от ожидания.

## 1.4 Сбор и анализ прототипов

### 1.4.1 Первый прототип

Название игры: Doodle jump.

Описание: Doodle jump – серия видеоигр, комиксов и игрушек от Lima Sky. Для платформы IOS игра вышла 6 апреля 2009 года, а для платформ Android и BlackBerry — 2 марта 2010 года. В Doodle Jump необходимо подняться как можно выше по области, похожей на лист тетради в клетку, постоянно перепрыгивая с одной платформы на другую, по пути подбирая реактивные ранцы, избегая чёрных дыр и стреляя в монстров снарядами из носа.

Игра «Doodle jump» изображена на рисунке 1.1.

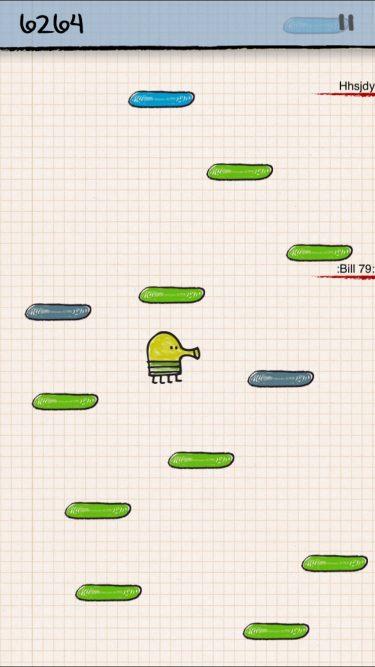


Рисунок 1.1 — Скриншот игры Doodle Jum»

### 1.4.2 Второй прототип

Название игры: Santa jump.

Описание: аркада-платформер с видом сбоку для мобильных платформ. В роли добродушного Санты вам предстоит прыгать по крышам и разносить детям подарки.

Игра «Santa jump» изображена на рисунке 1.2.



Рисунок 1.2 — Скриншот игры «Santa jump»

# Проектирование приложения

## 2.1 UI/UX дизайн проекта

Выбор приложения для разработки дизайн-проекта будущего приложения пал на веб-сервис Figma. Выбор обоснован тем, что Figma является веб-сервисом и работает из браузера, также очевидным плюсом является приятный и понятный дизайн интерфейса, а также наличием плагинов, облегчающих перенос разметки из дизайн-проекта в будущее приложение.

Для проекта были определены основные экраны:

* главное меню;
* экран выбора персонажа;
* экран достижений;
* главный экран игры;
* экран выхода;

При открытии игры пользователя встречает главный экран приложения с изображёнными кнопками взаимодействия, а именно перехода в магазин, меню и начала игры. Главное меню представлено на рисунке 2.1.

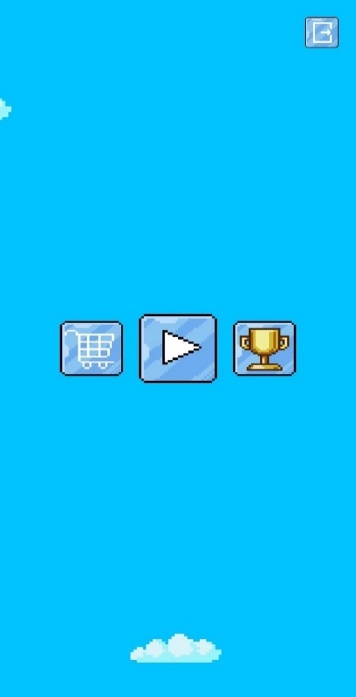


Рисунок 2.1 – главный экран игры

В магазине представлен весь ряд доступных игровых персонажей. Магазин представлен на рисунке 2.2

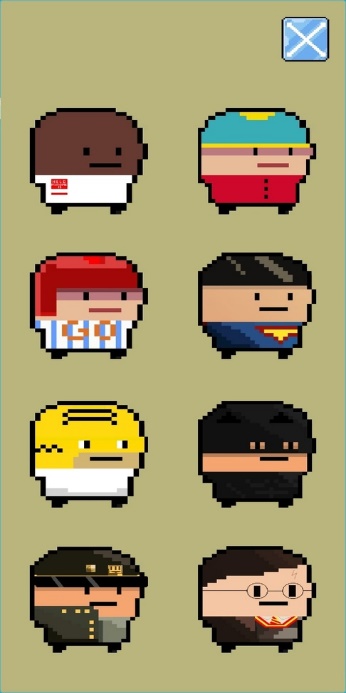


Рисунок 2.2 — магазин игры

При нажатии на «кубок» игрока переносит в окно с достижением и его лучшим счётом. Представлено на рисунке 2.3.



Рисунок 2.3 — экран достижений

При нажатии на кнопку играть игрока встречает окно с основным игровым процессом, представленным на рисунке 2.4.

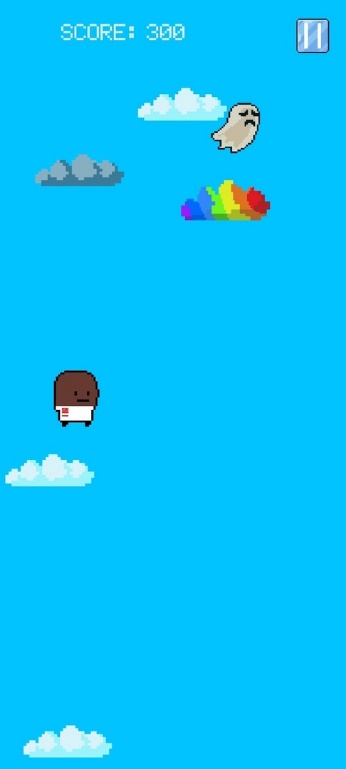


Рисунок 2.4 – игровой процесс

При нажатии на кнопку паузы игра останавливается и игроку предоставляется выбор продолжить игру или выйти из неё. Свой выбор игрок может осуществить путём нажатия на одну из двух кнопок. Это окно представлено на рисунке 2.5.



Рисунок 2.5 – экран выхода

## 2.2 Выбор технологии, языка и среды программирования

Для разработки моей игры была выбрана среда программирования Unity. Преимущества данной среды разработки, такие как ее широкое использование и наличие множества ресурсов и документации, сделали ее предпочтительным выбором.

Unity поддерживает два основных языка программирования: C# и UnityScript (базирующийся на JavaScript). В качестве языка разработки был выбран C#, так как у нас есть опыт работы с этим языком программирования и он широко используется в индустрии разработки игр

Для создания графики и спрайтов в игре использовалась программа Aseprite. Ее простота использования и возможность создания пиксельной графики делают ее идеальным инструментом для разработки игровых спрайтов и анимаций.

В процессе разработки использовались также различные ресурсы и сообщества Unity, такие как Unity Asset Store и Unity Community, где можно найти готовые ассеты, скрипты и советы от других разработчиков.

# Разработка мобильного приложения

## 3.2 Разработка мультимедийного контента

Дизайн-проект приложения разрабатывался в веб-сервисе Figma.

Рабочий интерфейс веб-сервиса представлен на рисунке 3.2.1.

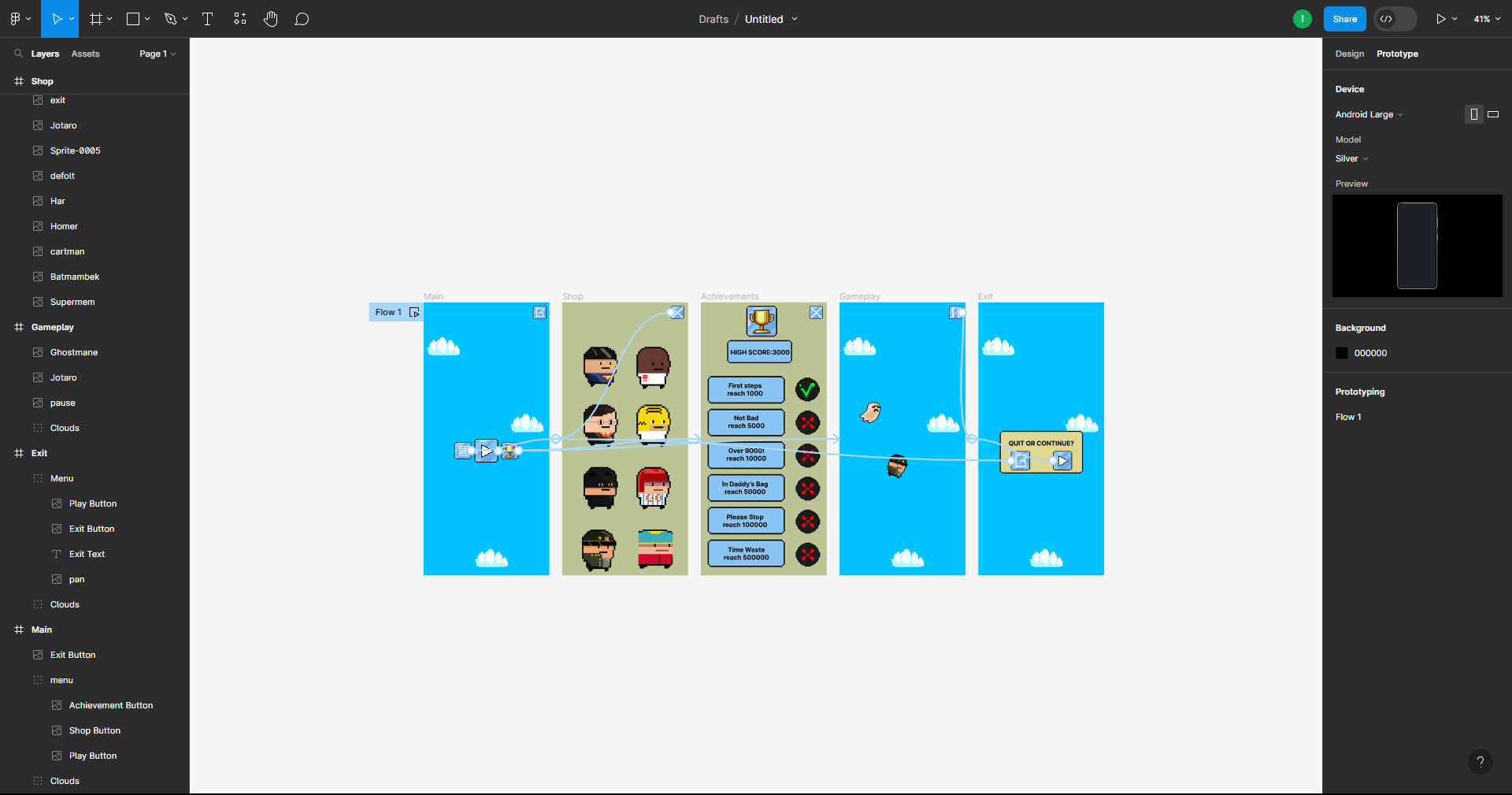


Рисунок 3.1.1 — рабочий интерфейс Figma

В приложении Aseprite были созданы 8 игровых персонажей (рис 3.1.1)

иконка приложения (рис. 3.1.2), галочка и крестик для достижений (рис 3.1.3), кнопки играть, выход, достижений, магазина, паузы (рис.3.1.4), облака и призрак (рис 3.1.5)

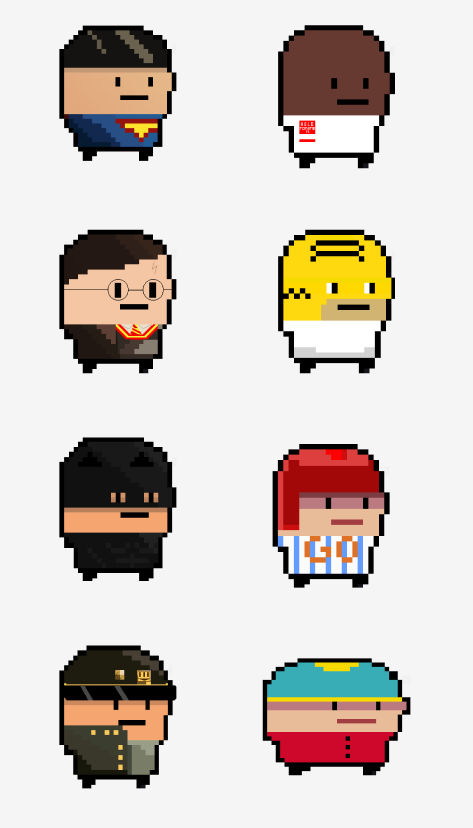


Рисунок 3.1.1 — игровые персонажи

****

Рисунок 3.1.2 – иконка игры

****

Рисунок 3.1.3 – иконка статуса достижений

****

Рисунок 3.1.3 – кнопки управления игрой



Рисунок 3.1.4 – облака и призрак

## 3.3 Описание используемых плагинов

На рисунке 3.3.1 изображен эмулятор для запуска и тестирования приложения в Android Studio.

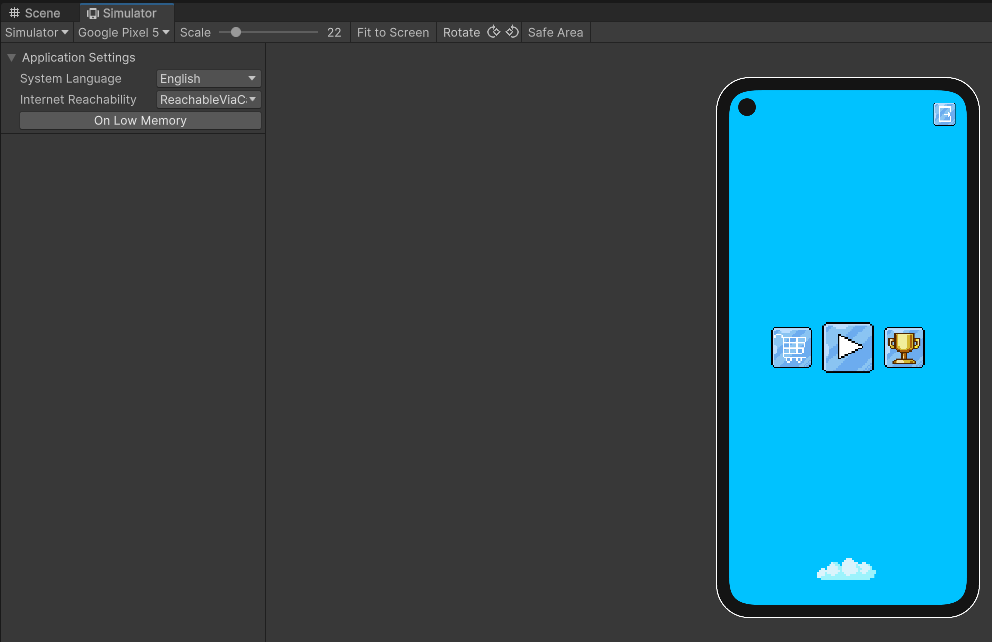


Рисунок 3.3.1 — эмулятор Android Studio

На рисунке 3.3.2 изображены используемые в проекте плагины, большая часть из них — плагины, установленные по умолчанию.

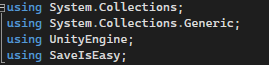
****

Рисунок 3.3.2 — используемые плагины

Добавленным является только последний плагин «SaveIsEasy», он взят из магазина ассетов Unity и отвечает за сохрание прогресса.

## 3.4 Описание разработанных процедур и функций

В приложении разработаны следующие функции:

* слежка камеры за персонажем;
* перемещение призраков;
* перемещение серых движущихся облаков;
* пауза и продолжить;
* перемещение белых облаков;
* генерация облаков;
* выход из игры;
* счетчик набранных очков;
* движение и смерть персонажа.
* сохранение и загрузка данных

Листинг кода, отвечающего за слежку камеры за персонажем представлен в листинге 1.

public class CameraFollow : MonoBehaviour

{

public Transform doodlePos;

void Update()

{

if (doodlePos.position.y > transform.position.y)

{

transform.position = new Vector3(transform.position.x, doodlePos.position.y, transform.position.z);

}

}

}

Листинг 1 — код, отвечающий слежку камеры

Данный скрипт реализует логику слежения камеры за объектом. Когда персонаж поднимается выше, чем камера, камера изменяет свою позицию по вертикали, чтобы оставаться выше объекта и следить за ним.

Следующая функция — это перемещение призраков. Скрипт реализует возможность призрака перемещаться наверх со случайной координатой Х после того, как он ушел за пределы экрана. Код метода представлен на листинге 2.

public class Ghoust : MonoBehaviour

{

public GameObject ghoust;

public void OnCollisionExit2D(Collision2D collision)

{

if (collision.collider.name == "DeadZone")

{

float RandX = Random.Range(-3.16f, 1.16f);

float RandY = Random.Range(transform.position.y + 25f, transform.position.y + 27f);

transform.position = new Vector3(RandX, RandY, 0);

}

}

}

Листинг 2 — код, отвечающий за перемещение призраков

На листинге 3 представлен метод перемещения серых облаков.

public class Graycloud : MonoBehaviour

{

public float forceJump;

public void StartMoving()

{

InvokeRepeating("Moving", 0, 1);

}

public void Moving()

{

float RandX1 = Random.Range(-3.16f, 1.16f);

float Y = transform.position.y;

transform.position = new Vector3(RandX1, Y, 0);

}

public void OnCollisionEnter2D(Collision2D collision)

{

if (collision.relativeVelocity.y < 0)

{

Dude.instance.DudeRigid.velocity = Vector2.up \* forceJump;

}

}

public void OnCollisionExit2D(Collision2D collision)

{

if (collision.collider.name == "DeadZone")

{

float RandX = Random.Range(-3.16f, 1.16f);

float RandY = Random.Range(transform.position.y + 20f, transform.position.y + 22f);

transform.position = new Vector3(RandX, RandY, 0);

}

}

}

Листинг 3 — код, отвечающий за перемещения серых облаков

Данный скрипт перемещает серое облако как и призрака, так же он двигает облако по координате Х, обрабатывает коллизию и отталкивает персонажа от облака, закручивая его.

Ниже на листинге 4 представлен метод отправки сообщения при восстановлении пароля.

public class Pause : MonoBehaviour

{

public void PauseGM()

{

Time.timeScale = 0f;

}

public void ContinueGM()

{

Time.timeScale = 1f;

}

}

Листинг 4 — код, отвечающий функции пауза и продолжить

На листинге 5 представлен метод перемещения обычного облака.

public class Platform : MonoBehaviour

{

public float forceJump;

public void OnCollisionEnter2D(Collision2D collision)

{

if (collision.relativeVelocity.y < 0)

{

Dude.instance.DudeRigid.velocity = Vector2.up \* forceJump;

}

}

public void OnCollisionExit2D(Collision2D collision)

{

if (collision.collider.name == "DeadZone")

{

float RandX = Random.Range(-3.16f, 1.16f);

float RandY = Random.Range(transform.position.y + 20f, transform.position.y + 22f);

transform.position = new Vector3(RandX, RandY, 0);

}

}

}

Листинг 5 — код, отвечающий за перемещение облака

Этот скрипт отвечает за отталкивание от облака игрока и перемещение его наверх и на случайную координату Х после выхода за пределы экрана.

На листинге 6 находится код, ответственный за генерацию облаков. Данный код создает 5 облаков и располагает их на случайных координатах XY.

public class PlatformGenerate : MonoBehaviour

{

public GameObject Cloud;

void Start()

{

Vector3 SpawnerPosition = new Vector3();

for (int i = 0; i < 5; i++)

{

SpawnerPosition.x = Random.Range(-4f, -0.2f);

SpawnerPosition.y += Random.Range(2f, 3f);

Instantiate(Cloud, SpawnerPosition, Quaternion.identity);

}

}

}

Листинг 6 — код, отвечающий за генерацию облаков

Ниже, на листинге 7, представлен код выхода из игры.

public class Quit : MonoBehaviour

{

public void ExitGame()

{

Application.Quit();

}

}

Листинг 7 — код, отвечающий за выход из игры.

Этот скрипт реализует функционал подсчета и отображения счета, обновления рекорда, а также отображения достижений в зависимости от достигнутого рекорда.

На листинге 8 находится код перемещения персонажа.

public class Dude : MonoBehaviour

{

public static Dude instance;

float horizontal;

public Rigidbody2D DudeRigid;

void Start()

{

if (instance == null)

{

instance = this;

}

}

void FixedUpdate()

{

if (Application.platform == RuntimePlatform.Android)

{

horizontal = Input.acceleration.x;

}

if (Input.acceleration.x < 0)

{

gameObject.GetComponent<SpriteRenderer>().flipX = false;

}

if (Input.acceleration.x > 0)

{

gameObject.GetComponent<SpriteRenderer>().flipX = true;

}

DudeRigid.velocity = new Vector2(Input.acceleration.x \* 10f, DudeRigid.velocity.y);

}

public void OnCollisionEnter2D(Collision2D collision)

{

if (collision.collider.name == "DeadZone")

{

SceneManager.LoadScene(0);

}

}

}

Листинг 8 — код, отвечающий за перемещение персонажа.

При помощи акселерометра определяется наклон телефона, и персонаж перемещается по оси х.

Ниже, на листинге 9 расположен код, ответственный за сохранение и загрузку данных

public class LoadScene : MonoBehaviour

{

public void LoadGM()

{

SaveIsEasyAPI.LoadAll();

}

}

public class Save : MonoBehaviour

{

public void SaveGM()

{

SaveIsEasyAPI.SaveAll();

}

}

Листинг 8 — код, отвечающий за сохранение и загрузку данных.

# Тестирование

## 4.1 Протокол тестирования дизайна приложения

Для проведения тестирования дизайна приложение было выбрано устройство Nothing Phone 1 c разрешением экрана 2340×1080.

Проверка была проведена на:

* оптимальный размер кнопок;
* читабельный размер шрифта;
* приятное глазу цветовое сочетание;
* корректное расположение элементов.

## 4.2 Протокол тестирования функционала приложения

Для проверки правильности ввода электронной почты при регистрации были проведены UnitTests, которые изображены ниже на листинге 9:

public class EmailUtils {  
  
 private static final String *VALID\_EMAIL\_REGEX* = "(^[A-Za-z0-9+\_.-]+@[A-Za-z0-9.-]+$);  
  
 @Test  
 public void validateEmail\_EmptyString() {  
 *// setup* String email = "";  
  
 *// execute* boolean actual = EmailUtils.*validateEmail*(email);  
  
 *// assert  
 assertFalse*(actual);  
 }  
@Test  
 public void validateEmail\_Missing\_Symbol() {  
 *// setup* String password = "emailmail.ru";  
  
 *// execute* boolean actual = EmailUtils.*validatePassword*(email);  
  
 *// assert  
 assertFalse*(actual);

Листинг 9 — UnitTests

Также для тестирования функционала были разработаны TestCase.

TestCase 1:

Название — регистрация.

Предусловия — приложение запущено.

Шаги:

* открыть страницу регистрации;
* заполнить все поля;
* нажать на кнопку «регистрация».

Ожидаемый результат тест кейса: пользователь зарегистрирован в системе.

Статус кейса — выполнено.

TestCase 2:

Название — авторизация.

Предусловия — приложение запущено, пользователь зарегестрирован.

Шаги:

* открыть страницу авторизации;
* заполнить все поля;
* нажать на кнопку «авторизоваться».

Ожидаемый результат тест кейса: пользователь вошел в систему.

Статус кейса — выполнено.

TestCase 3:

Название — переход на другое окно.

Предусловия —пользователь авторизован в системе.

Шаги:

* нажать на желаемую кнопку.

Ожидаемый результат тест кейса: пользователь перешел на необходимую страницу.

Статус кейса — выполнено.

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы было разработано мобильное приложение, позволяющее пользователю взаимодействовать с облачным хранилищем фотографий, а именно — добавлять фотографии и просматривать их.

Однако, разработка мобильного приложения является одной из нескольких частей курсовой работы. Важную роль сыграло взаимодействие клиентского приложения с сервером, находящимся на веб-хостинге. Были созданы классы на php, которые являлись “интерфейсом” и получали данные из http-get запроса, исходящего от приложения и записывали информацию в базу данных.

Во время разработки возникали трудности во время работы с изображениями — их передаче на сервер, отображение в приложении. Большую часть трудностей можно связать с малым количеством информации по теме, а также ее заметному устареванию.

СПИСОК ИСТОЧНИКОВ

1. ГОСТ 2.105–95 ЕСКД. Общие требования к текстовым документам. — М.: Изд-во стандартов, 1996. — 37 с.
2. ГОСТ Р 7.05–2008 СИБИД. Библиографическая ссылка. — М.: Стандартинформ, 2008 — 22 с.
3. ГОСТ 19.101–77 ЕСПД. Виды программ и программных документов. — М.: Стандартинформ, 2010 — 4 с.
4. ГОСТ 19.105–78 ЕСПД. Общие требования к программным документам. — М.: Изд-во стандартов, 1987. — 2 с.
5. ГОСТ 19.404–79 ЕСПД. Пояснительная записка. Требования к содержанию. — М.: Изд-во стандартов, 1987. — 2 с.
6. ГОСТ 2.106–96 ЕСКД. Требования к программным документам, выполненным печатным способом. — М.: Изд-во стандартов, 1996. — 37 с.
7. ГОСТ 19.401–78 ЕСПД. Текст программы. Требования к содержанию. — М.: Стандартинформ, 2010 — 4 с.
8. ГОСТ 19.402–78 ЕСПД. Описание программы. — М.: Стандартинформ, 2010 — 3 с.
9. ГОСТ 19.503–79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению. — М.: Стандартинформ, 2010 — 4 с
10. phpMyAdmin [Электронный ресурс]: Документация phpMyAdmin – Режим доступа к руководству: <https://www.phpmyadmin.net/>
11. Android Developers [электронный ресурс]: Документация Android Stusio – https://developer.android.com/docs/
12. Help Center [электронный ресурс]: Документация Java: https://docs.oracle.com/en/java/
13. Git [электронный ресурс]: Документация Git: https://git-scm.com/
14. Figma [электронный ресурс]: Прототипирование проектов https://www.figma.com/

**Приложение А**

Техническое задание

Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж

имени Б.С. Галущака»

**разработка мобильного приложения для ХРАНЕНИЯ ФОТОГРАФИЙ**

НАТКиГ.202100.010.000ПЗ

Выполнил:

студент группы

ПР-20.102к

Плахотный Г.А.

2022

ОГЛАВЛЕНИЕ

[Введение ………………………………………………………………………….23](#_Toc121334558)

[1 Назначение разработки ………………………………………………………..24](#_Toc121334559)

[2 Требования к мобильному приложению……………………………………..25](#_Toc121334560)

[2.1 Требования к функциональным характеристикам…………………………25](#_Toc121334561)

[2.2 Требования к надёжности …………...………………………………………25](#_Toc121334562)

[2.3 Условия эксплуатации……………………………………………………….26](#_Toc121334563)

[2.4 Требования к составу и параметрам технических средств………………..26](#_Toc121334564)

[2.5 Требования к информационной и программной совместимости…………26](#_Toc121334565)

[2.6 Требования к защите информации………………………………………….26](#_Toc121334566)

[2.7 Требования к маркировке и упаковке………………………………………26](#_Toc121334567)

[3 Требования к программной документации 27](#_Toc121334568)

[4 Технико-экономические показатели 28](#_Toc121334569)

[5 Стадии и этапы разработки 29](#_Toc121334570)

[6 Порядок контроля и приёмки 30](#_Toc121334571)

Введение

Настоящее техническое задание распространяется на разработку информационной системы «Разработка мобильного приложения для сохранения фотографий», используемого для отправки фотографий в облачное хранилище из галереи мобильного устройства, просмотра фотографий облачного хранилища в мобильном приложении, создании фотографий напрямую из мобильного приложения и отправки их в облачное хранилище.

Наименование приложения: «SMPhoto».

Краткая характеристика области применения: приложение для облачного хранения фотографий – полезное оснащение телефона любого пользователя. Пользователь может просмотреть фотографии, находящиеся в облачном хранилище, создавать фотографии в мобильном приложении и отправлять их в облачное хранилище.

Условные обозначения и сокращения:

БД – База данных;

ИС – Информационная система;

ОС – Операционная система.

Основанием для проведения разработки является Протокол № от месяца 202 года.

Наименование темы разработки – «Разработка мобильного приложения для облачного хранения фотографий».

Условное обозначение темы разработки – «Облако – SMPhoto».

1 Назначение разработки

Основное назначение мобильного приложения заключается в:

* просмотре фотографий, сохраненных на сервере;
* возможности клиента создавать фотографии и отправлять их на сервер;
* возможности клиента загружать фотографии из галереи мобильного устройства.

Лица, которые могут работать с данной Системой:

* пользователь – просматривает фотографии, регистрируется, авторизуется, отправляет фотографии на сервер.

2 Требования к мобильному приложению

## 2.1 Требования к функциональным характеристикам

Требования к составу выполняемых функций:

* регистрация и авторизация пользователей;
* просмотр имеющихся на сервере фотографий;
* добавление фотографий;
* создание фотографий в приложении.

Входные данные должны быть организованы в виде вводимого, в специальную форму, текста, соответствующего определённому шаблону. Данные, вводимые вручную, проверяются на корректность.

## 2.2 Требования к надёжности

Обеспечение устойчивого функционирования должно выполняться несколькими действиями:

* организация бесперебойного питания оборудования пользователя;
* использование лицензионного программного обеспечения;
* организация стабильного интернет-соединения.

Приложение должно контролировать входную информацию:

* соблюдение типов данных при заполнении полей;
* операции изменения, удаления и сохранения.

Время восстановления после отказа, вызванного сбоем электропитания технических средств, не фатальным сбоем ОС или файловой системы, не должно превышать 15 минут при соблюдении условий эксплуатации технических и программных средств и правильной настройки операционной системы.

## 2.3 **Условия эксплуатации**

Обслуживание ИС включает в себя:

1. информационное обслуживание – ввод и редактирование информации БД;
2. системное администрирование БД ИС.

## 2.4 Требования к составу и параметрам технических средств

Для работы приложения на устройстве требуется: объем свободной памяти не менее 3 ГБ, версия устройства Android 10 и выше.

## 2.5 Требования к информационной и программной совместимости

Проектирование взаимодействия с файловой системой должно быть выполнено в рамках разработки курсового проекта. При разработке взаимодействия с файловой системой должен быть использован язык программирования Kotlin.

## 2.6 Требования к защите информации

Доступ к информации БД предоставляется только администратору базы данных.

## 2.7 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

3 Требования к программной документации

Состав программной документации должен включать в себя:

* техническое задание;
* проектную документацию.

4 Технико-экономические показатели

Экономические преимущества разработки и ориентировочная экономическая эффективность не рассчитывается.

5 Стадии и этапы разработки

Стадии и этапы разработки указаны в таблице 1.

Таблица 1 – Стадии разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название этапа | Срок, даты | Отчётность |
| 1 | Исследование предметной области |  | Пояснительная записка |
| 2 | Выбор моделей, описывающих предметную область |  | Пояснительная записка |
| 3 | Разработка технического задания |  | Техническое задание |
| 4 | Изучение Kotlin и Firebase |  | Пояснительная записка |
| 5 | Анализ требований и уточнение спецификаций |  | Спецификации программного обеспечения |
| 6 | Проектирование структуры мобильного приложения, проектирование компонентов (технический проект) |  | Схема структурная системы и спецификации компонентов |
| 8 | Кодирование клиентской части |  | Программный  Продукт |
| 9 | Тестирование компонентов.  Сборка и комплексное тестирование |  | Тексты программных компонентов |
| 10 | Разработка программной  документации |  | Программная  документация |
| 12 | Защита |  |  |

6 Порядок контроля и приёмки

Виды испытаний – защита курсового проекта.

Общее требования к приёмке:

* техническое задание;
* пояснительная записка;
* программный продукт;
* презентация.