Учреждение образования

«Гродненский государственный политехнический колледж»

Специальность: 2 – 40 01 01 «Программное обеспечение информационных

технологий»

Специализация: 2 – 40 01 01 35 «Программное обеспечение обработки

экономической и деловой информации»

Дисциплина: «Технология разработки программного обеспечения»

Группа: ПЗТ – 39

Техническое задание на разработку десктопного приложения «Собери ПК»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Разработал А.С. Смирнов

Руководитель проекта О.И. Кизер

2023

**Содержание**

Изм.

Кол

Лист №док

Подпись

Дата

Лист

2

ТЗ 2-40 01 01.33.39.21.23 ТРПО

Разраб.

Сминров

Пров.

Кизер

Н. контр.

Утв.

Техническое задание на разработку десктопного приложения «Собери ПК»

Стадия

Листов

29

УО ГГПК

1 Постановка задачи 3

1.1Организационно-экономическая сущность задачи 3

1.2.Функциональные требования 3

1.3 Эксплуатационные требования 4

1.4 Инструменты разработки 4

[2 Проектирование задачи](#_Достоинства_и_недостатки) 5

[2.1](#_Toc406878086) Разработка структуры сайта, системы меню, навигации 5

[2.2](#_Toc406878088) Организация данных 5

2.3 Разработка UML-диаграмм 5

2.3.1 Диаграмма последовательности 5

2.3.2 Модель данных 5

2.4 Диаграмма вариантов использования 6

2.5 Диаграмма классов 6

2.6 Диаграмма деятельности 6

2.7 Диаграмма коопирации (объектов) 6

2.8 Моделирование бизнес-процессов 6

[3 Реализация](#_Достоинства_и_недостатки) 7

3.1 Руководство программиста 7

4 Тестирование 8

[4.1](#_Toc406878088) Тесты на использование 8

[4.2](#_Toc406878086) Отчет о результатах тестирования 9

[5](#_Toc406878086) Руководство пользователя 10

Заключение 11

Список использованных источников 12

[Приложение А Диаграмма последовательности](#_Toc406878088)  14

Приложение Б Модель данных 16

Приложение В Диаграмма вариантов использования 18

Приложение Г Диаграмма классов 20

Приложение Д Диаграмма деятельности 22

Приложение Е Диаграмма коопирации (объектов) 24

Приложение Ж Моделирование бизнес-процессов 26

Приложение 3 Листинг программы ……28

**1 Постановка задачи**  
 **1.1 Организационно-экономическая сущность задачи**  
Наименование задачи: Десктопное приложение «Собери ПК»  
Цель разработки: показать, как устроен ПК;

Назначение: данная игра разрабатывается для желающих, кто хочет немного больше узнать о компьютере.

Периодичность использования: по мере необходимости.   
Источники и способы получения данных: в описании вещей, геймплее.  
 **1.2 Функциональные требования**Описание перечня функций и задач, которые должен выполнять будущий ПП:

Игрок:

1. Регистрация
2. Авторизация
3. Взаимодействия с покупателями
4. Выбор предмета
5. Выполнение заказов
6. Изменение настроек

Описание процессов с входной, выходной и условно-постоянной информацией представлены в таблице 1

Таблица 1 – описание процессов.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Категория пользователей | Наименование процесса | Краткое описание алгоритма выполнения процесса | Входная информация | Выходная информация | Условно-постоянная  информация |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | Игрок | Регистрация | При входе в игру, пользователь должен зарегистрироваться, чтобы начать играть | Почта, пароль, никнейм в игре | Появится табличка «пользователь успешно зарегистрирован» | Данные регистрации |
| 2 | Игрок | Авторизация | При выходе из аккаунта, игрок может заново авторизоваться, с помощью почты и пароля | Почта, пароль | После авторизации игрок может зайти в свой аккаунт | Данные авторизации |
| 3 | Игрок | Взаимодействия с покупателями | При нажатии на список с заказами, выбрать покупателя | Отсутствует | После нажатия на список заказов, игрок может выбрать покупателя | Данные игры |
| 4 | Игрок | Выбор предмета | При нажатии на магазин, игрок выбирает различные предметы, которые требуются в заказе | Отсутствует | При нажатии на предмет, появляется иконка с информацией о выбранном предмете | Данные игры |
| 5 | Игрок | Выполнение заказов | При нажатии на покупателя, игрок может принять или отклонить его заказ | Отсутствует | На иконке с заказом, игрок может прочитать, что требуется ему сделать в заказе | Данные игры |
| 6 | Игрок | Настройки | Игрок может изменить настройки игры | Звук, музыка | Применение новых настроек | Варианты настроек |

**1.3.Эксплуатационные требования**

Требования к применению: Помогает игроку больше узнать о компьютере

Требования к реализации: Для разработки данного ПП должен использоваться unity, основанный на С#

Требования к надёжности: дальнейшие обновления ПО

Требования к интерфейсу:брать в пример сборку ПК из реальной жизни

**1.4 Инструменты разработки**

Для разработки данного приложения будет выбран unity.

Иные инструменты, используемые при разработке и написании сопутствующей документации:

- WEB-ресурс DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм;

- Microsoft Office Word – для написания документации к программному продукту;

- Microsoft Office PowerPoint – для создания презентации, которая будет использована на защите проекта;

- WEB-ресурс Figma– будет использоваться для создания UX, UI ;

Разработка проекта будет происходить на компьютере со следующими параметрами:

- Intel(R) Core(TM) i7-11400H @ 2.30GHz 2.30 GHz;

- объем ОЗУ 16Гб;

- объем места на жестком диске 500 GB;

- графический адаптер nvidia gforce rtx3060;

[**2 Проектирование задачи**](#_Достоинства_и_недостатки)

[**2.1**](#_Toc406878086) **Разработка структуры приложения, системы меню, навигации**

Разработка макета проекта. Чтобы увидеть UX, UI прототипы приложения необходимо перейти по ссылке:

UI:https://www.figma.com/file/lkp69AvPVjjb2AoSWbJmps/Untitled?type=design&node-id=14-76&mode=design&t=nZUrJjLX8CxNAu0B-0

UX:https://www.figma.com/file/1Wsg0nq7IKt4gY6zjW96ti/Untitled?type=design&mode=design&t=Pi5G5P83Y9qdXTrJ-0

[**2.2**](#_Toc406878088) **Организация данных**

Данные находятся в приложениях.

**2.3 Разработка UML-диаграмм**

**2.3.1 Диаграмма последовательности**

Диаграмма последовательности UML — такая диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления. Основные элементы диаграммы последовательности это: обозначения объектов (прямоугольники), вертикальные линии, отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами.

При разработке программного продукта была поставлена задача разработать диаграмму последовательности, которая показывает порядок взаимодействия пользователя с системой при оформлении заказа. С данной диаграммой можно ознакомиться в Приложении А.

**2.3.2 Модель данных**

Модель данных – это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. С данной моделью можно ознакомиться в Приложении Б.

**2.4 Диаграмма вариантов использования**

Диаграмма вариантов использования – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. С прототипом вы можете ознакомиться в Приложении В.

**2.5 Диаграмма классов**

**Диаграмма классов** определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации). С данной схемой вы можете ознакомиться в Приложении Г.

**2.6 Диаграмма деятельности**

**Диаграмма деятельности** — это графическое представление процессов и задач, выполняемых в рамках определенного проекта или деятельности. Эта диаграмма используется для анализа и оптимизации бизнес-процессов, планирования работы, принятия решений и управления проектами. Диаграмма деятельности состоит из набора символов и линий, каждый из которых имеет свое значение и назначение. С данной схемой вы можете ознакомиться в Приложении Д.

**2.7 Диаграмма кооперации(объектов)**

Диаграмма кооперации предназначена для описания поведения системы на уровне отдельных объектов, которые обмениваются между собой сообщениями, чтобы достичь нужной цели или реализовать некоторый вариант использования. С данной схемой вы можете ознакомиться в Приложении Е.

**2.8 Моделирование бизнес-процессов**

Моделирование бизнес-процессов (BPM) в [управлении бизнес-процессами](https://ru.wikipedia.org/wiki/BPM_(%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%86%D0%B8%D1%8F)) и [системной инженерии](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F) — это деятельность по представлению процессов предприятия, позволяющая анализировать, улучшать и автоматизировать текущие [бизнес-процессы](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81). С данным моделированием вы можете ознакомиться в Приложении Ж.

**3 Реализация**

**3.1 Руководство программиста**

Данный программный продукт был разработан с помощью unity.

Для создания сцены нам понадобится создать саму сцену (File–new Scene). Чтобы создать что-либо на сцене, для начала нужно создать Canvas (пкм в иерархии-UI-Сanvas), и дальше создаем кнопки, путь такой же как для канвас, только выбрать то, что нужно. Чтобы создать картинку, в Canvas создаем image и в sprites перетаскиваем картинку, которую надо вставить, и изменить размеры. Чтобы кнопки работали нужно для начала создать кнопку (Canvas-пкм-UI-button) потом создаем С# script, называем его и в C# пишем код для работы кнопки.

[**4 Тестирование**](#_Достоинства_и_недостатки)

**4.1 Тесты на использование**

При разработке сайта «3D модели» многие возникающие ошибки и недоработки были исправлены на этапе реализации программного продукта. После завершения испытания реализации ресурса было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программного продукта в автономном режиме. Тесты на использование представлены в таблице 4.

Таблица4 –Тесты на использование

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № теста | | Тест | | | Краткое описание алгоритма  выполнения процесса | | Ожидаемый результат | | | Физический результат | | Результат тестирования | |
| 1 | | 2 | | | 3 | | 4 | | | 5 | | 6 | |
| 1 | | Проверяем открытие приложения | | | Нажать на иконку приложения | | Открытие приложения | | | Произошло открытие приложения | | Выполнено | |
| 2 | | Проверка закрытия игры | | | Нажать на кнопку «Выход» | | Закрытие приложения | | | Произошло закрытие приложение | | Выполнено | |
| 3 | | Проверка кнопок, переходящих на другие сцены | | | Нажать на кнопку «Старт» | | Переход на другую сцену | | | Произошел переход на сцену с меню | | Выполнено | |
| 4 | | Проверка музыки | | | При открытии приложения должна играть музыка | | Музыка играет на всех сценах | | | Музыка играет на всех сценах | | Выполнено | |
| 5 | | Проверка изменения громкости музыки | | | Пользователь редактирует громкость музыки с помощью slider | | Музыка должна играть тише(громче) | | | Музыка играет тише(громче) | | Выполнено | |
|  | |  |  | |  | |  |  | |  | |  | |
|  | |  |  | |  | |  |  | |  | |  | |

**4.2 Отчет о результатах тестирования**

В результате проведения тестирования выяснилось, что не все ранее оговоренные функции и требования, были разработаны, а также протестированы. Тесты показали, что все написанные функции работают правильно.

В ходе тестирования программного обеспечения продукта на устройстве не было выявлено каких-либо ошибок, так как адаптивность ресурса была проведена на всех стадиях разработки.+

**5 Руководство пользователя**

Разрабатываемое приложение «Собери ПК» будет является информационным, для помощи подросткам, юным программистам, и людям заинтересованных в компьютерах чтобы они чуть лучше разбирались в составляющих компьютера. Проект будет включать в себя картинки с компонентами, описание к компонентам, чтобы лучше понять для чего есть данное составляющее и схема компьютера, чтобы лучше запомнить расположение комплектующих.

[**Заключение**](#_Toc406878088)

Целью данного проекта заключалась в том, чтобы показать, как устроен ПК;

В целом при реализации программного продукта, были выполнены не все условия, поставленные на начальном этапе разработки. Из чего мы можем сделать вывод, что ресурс нужно будет доработать и проверить остальные новые функции.

Так же в процессе создания программного продукта была подготовлена программная документация. Был разработан перечень диаграмм и проведены тесты для проверки правильности функционирования ПО.

**Список использованных источников**

1. Руководство по программированию на C# [Электронный ресурс]. – Электронные данные. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/.
2. Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/.
3. YouTube [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=xAtoU-FnC7o&list=PLDyJYA6aTY1k\_-3fFiMVoYY04jCr-QY55.

***Приложение А*** (Диаграмма последовательности)

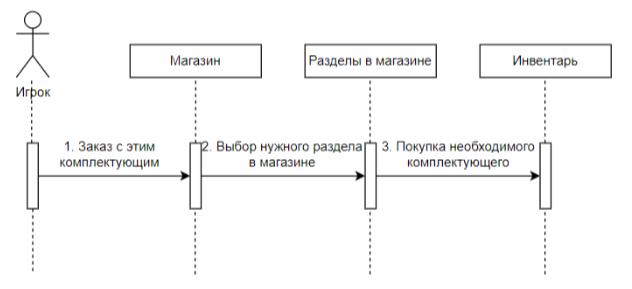


Рисунок 1- Диаграмма последовательности

***Приложение Б*** (Модель данных)

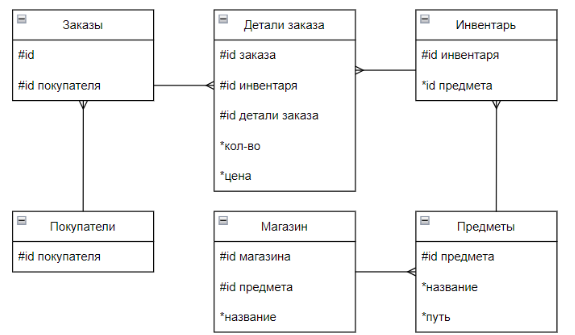


Рисунок 2- Модель данных

***Приложение В*** (Диаграмма вариантов использования)

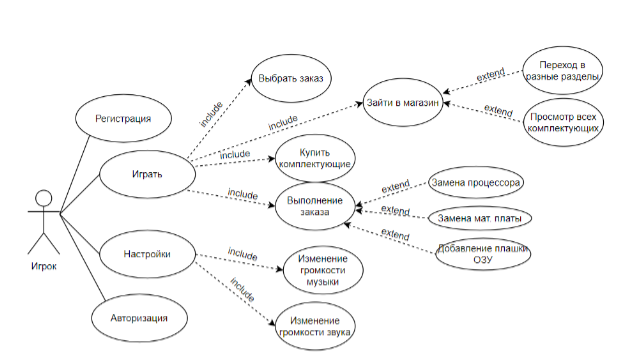


Рисунок 3- Диаграмма вариантов использования

***Приложение Г*** (Диаграмма классов)

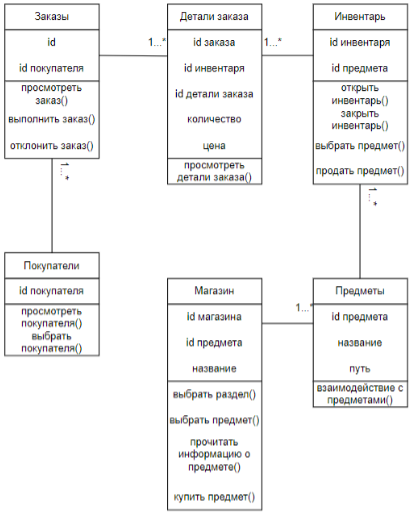


Рисунок 4- Диаграмма классов

***Приложение Д*** (Диаграмма деятельности)

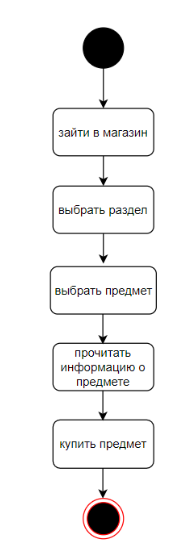


Рисунок 5- Диаграмма деятельности

***Приложение Е*** (Диаграмма кооперации(объектов))

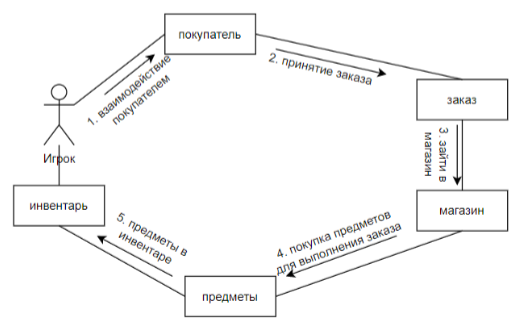
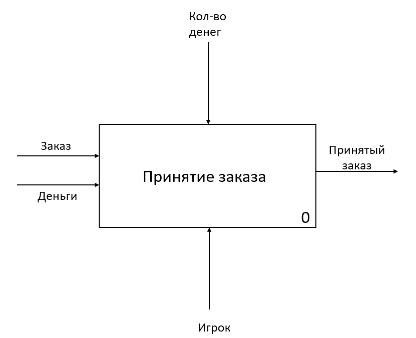


Рисунок 6- Диаграмма кооперации(объектов)

***Приложение Ж*** (Моделирование бизнес-процессов)



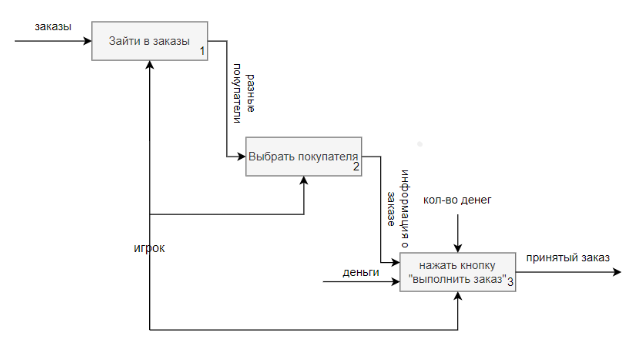


Рисунок 7- Моделирование бизнес-процессов

***Приложение З*** (Листинг программы)

using UnityEngine;

public class BackgroundInstanceControllerComponent : MonoBehaviour

{

[Header("Tags")]

[SerializeField] private string createdTag;

private void Awake()

{

GameObject obj = GameObject.FindWithTag(this.createdTag);

if (obj != null)

{

Destroy(this.gameObject);

}

else

{

this.gameObject.tag = this .createdTag;

DontDestroyOnLoad(this.gameObject);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class SoundVolueControl : MonoBehaviour

{

[Header("Components")]

[SerializeField] AudioSource audio;

[SerializeField] Slider slider;

[SerializeField] Text text;

[Header("Key")]

[SerializeField] private string saveVolumeKey;

[Header("tags")]

[SerializeField] private string sliderTag;

[SerializeField] private string textVolumeTag;

[Header("parameters")]

[SerializeField] private float volume;

private void Awake()

{

if (PlayerPrefs.HasKey(this.saveVolumeKey))

{

this.volume = PlayerPrefs.GetFloat(this.saveVolumeKey);

this.audio.volume = this.volume;

GameObject sliderObj = GameObject.FindWithTag(this.sliderTag);

if (sliderObj != null)

{

this.slider = sliderObj.GetComponent<Slider>();

this.slider.value = this.volume;

}

}

else

{

this.volume = 0.5f;

PlayerPrefs.SetFloat(this.saveVolumeKey, this.volume);

this.audio.volume = this.volume;

}

}

private void LateUpdate()

{

GameObject sliderObj = GameObject.FindWithTag(this.sliderTag);

if (sliderObj != null)

{

this.slider = sliderObj.GetComponent<Slider>();

this.volume = slider.value;

if (this.audio.volume != this.volume)

{

PlayerPrefs.SetFloat(this.saveVolumeKey, this.volume);

}

GameObject textObj = GameObject.FindWithTag(this.textVolumeTag);

if (textObj != null)

{

this.text = textObj.GetComponent<Text>();

this.text.text = Mathf.Round(f: this.volume \* 100) + "%";

}

}

this.audio.volume = this.volume;

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class SceneTransition : MonoBehaviour

{

public int SceneNumber;

public void Transition()

{

SceneManager.LoadScene(SceneNumber);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Menu : MonoBehaviour

{

public void ExitGame()

{

Application.Quit();

}

}