Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

Специальность 09.02.03 Программирование в компьютерных системах

**ОТЧЕТ**

**ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**(по профилю специальности)**

Профессиональный модульПМ.02 Разработка и администрирование баз данных

*(наименование профессионального модуля)*

Выполнил:

обучающийся учебной группы № 3ПКС-120

Плюснин Максим Юрьевич

*(И.О. Фамилия)*

Проверил:

руководитель практики от колледжа:

И.В. Сибирев

*(И.О. Фамилия)*

**Москва**

**20­­­­23**

СОДЕРЖАНИЕ

[**ВВЕДЕНИЕ** 3](#_Toc139051845)

[**ОБЩАЯ ЧАСТЬ** 4](#_Toc139051846)

[**Листинги скриптов** 6](#_Toc139051847)

[**Визуальная часть проекта** 16](#_Toc139051848)

[**ВЫВОД** 19](#_Toc139051849)

# **ВВЕДЕНИЕ**

Я проходил практику в Центре Разработки Программного Обеспечения от Финансового Университета (ЦРПО), где нас распределили между разными проектами. Я работал в группе по разработке VR(Virtual Reality) приложений.

В этом отчете представлены итоги по работе над проектом.

Для выполнения задания нам дали такое ТЗ(Техническое Задание):

1. Обучаемый должен пройти в «чистое» отделение санпропускника, подойти к своему шкафчику и «раздеться», оставить личные вещи в индивидуальном шкафу
2. Надеть переходные тапочки и пройти в «грязное» отделение санпропускника (дверь, отделяющая «чистое» помещение от «грязного», открывается только в одну сторону)
3. В «грязном» отделении у дежурного в окне получить комплект спецодежды (с/о).
4. Подойти к своему шкафчику и надеть основную спецодежду (кроме спецобуви. Она должна остаться в руках)
5. Взяв в руки спецобувь подойти к дисциплинирующему барьеру
6. На выходе из «грязного» отделения за дисциплинирующим барьером с надписью: «Надеть спецобувь» - надеть спецобувь. Далее пройти через дверь к секционному шкафу (дверь открывается только в одну сторону)
7. Взять из соответствующей ячейки секционного шкафа кассетницы личный индивидуальный дозиметр и закрепить его на спецодежде
8. Подойти к окну к оперативному персоналу отдела радиационной безопасности (ОРБ) для оформления допуска в ЗКД и получить оперативный дозиметр. Закрепить его на одежде
9. Пройти в ЗКД

# **ОБЩАЯ ЧАСТЬ**

**Unity** - [кроссплатформенная](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), разработанная американской компанией [Unity Technologies](https://ru.wikipedia.org/wiki/Unity_Technologies). Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих [персональные компьютеры](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), [игровые консоли](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0), [мобильные устройства](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE), [интернет-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и другие[[4]](https://ru.wikipedia.org/wiki/Unity_(%D0%B8%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)#cite_note-_2220faf84e878ba7-4). Выпуск Unity состоялся в [2005](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4_%D0%B2_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80%D0%B0%D1%85) году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие [визуальной среды разработки](https://ru.wikipedia.org/wiki/RAD_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [межплатформенной](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних [библиотек](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).

**Ход работы:**

На рисунке 1 представлены папки проекта.

Изображение выглядит как снимок экрана, текст, дизайн

Автоматически созданное описание

Рис 1. Папки проекта

На рисунке 2 изображена папка с анимациями.

Изображение выглядит как снимок экрана, Шрифт

Автоматически созданное описание

Рис 2. Анимации

На рисунке 3 представлены файлы материалов

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рис 3. Материалы

На рисунках 4 и 4.1 продемонстрированы prefabs объектов в проекте.

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис 4. Prefabs объектов

Изображение выглядит как снимок экрана, текст, дизайн

Автоматически созданное описание

Рис 4.1. Prefabs предметов

На рисунках 5 и 5.1. изображены scriptable objects предметов и диалогов.

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Автоматически созданное описание

Рис 5. Scriptable objects предметов

Изображение выглядит как текст, снимок экрана, Шрифт, часы

Автоматически созданное описание

Рис 5.1. Scriptable objects диалогов

На рисунках 6, 6.1, 6.2, 6.3 представлены скрипты проекта.

Изображение выглядит как текст, снимок экрана, Шрифт, часы

Автоматически созданное описание

Рис 6. Общие скрипты проекта.

Изображение выглядит как текст, снимок экрана, Шрифт, символ

Автоматически созданное описание

Рис 6.1. Скрипты игрока.

Изображение выглядит как текст, символ, Шрифт, логотип

Автоматически созданное описание

Рис 6.2. Скрипты интерфейса.

Изображение выглядит как текст, Шрифт, снимок экрана, логотип

Автоматически созданное описание

Рис 6.3. Скрипты объектов.

# **Листинги скриптов**

Далее будут продемонстрированы листинги каждого скрипта:

Скрипт CameraMovement нужен для вращения игроком камеры.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraMovement : MonoBehaviour

{

[SerializeField] private float \_mouseSensivity = 2f;

[SerializeField] private Transform \_playerBody;

private float \_xRotate;

private Vector3 \_velocity = Vector3.zero;

private void Start()

{

Cursor.lockState = CursorLockMode.Locked;

\_xRotate = 0;

}

private void Update()

{

float mouseX = Input.GetAxis("Mouse X") \* \_mouseSensivity;

float mouseY = Input.GetAxis("Mouse Y") \* \_mouseSensivity;

Vector3 playerRotation = \_playerBody.rotation.eulerAngles;

if (mouseY != 0f)

{

\_xRotate -= mouseY;

\_xRotate = Mathf.Clamp(\_xRotate, -90f, 90f);

}

if (mouseX != 0f)

{

\_playerBody.Rotate(Vector3.up \* mouseX);

playerRotation = \_playerBody.rotation.eulerAngles;

}

transform.rotation = Quaternion.Euler(\_xRotate, playerRotation[1], playerRotation[2]);

}

private void LateUpdate()

{

transform.position = Vector3.SmoothDamp(transform.position,

\_playerBody.position + Vector3.up,ref \_velocity, 5,

Mathf.Infinity, Time.unscaledTime);

}

}

Скрипт PlayerMovement используется игроком для движения персонажа.

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[RequireComponent(typeof(Rigidbody))]

public class PlayerMovement : MonoBehaviour

{

[SerializeField] private float \_speed = 10f;

private Rigidbody \_rigidbody;

private Vector3 \_velocity;

private float \_x, \_z;

private void Start()

{

\_rigidbody = GetComponent<Rigidbody>();

}

private void Update()

{

\_x = Input.GetAxis("Horizontal");

\_z = Input.GetAxis("Vertical");

}

private void FixedUpdate()

{

if (\_x != 0 || \_z != 0)

{

Vector3 move = (transform.right \* \_x + transform.forward \* \_z);

float speed = \_speed;

if (Physics.Raycast(transform.position, move, out RaycastHit hit, 0.3f))

{

Vector2 oldDirection = new Vector2(move.x, move.z);

Vector2 newDirection = Vector2.Perpendicular(hit.normal);

float angle = Vector2.Angle(oldDirection, newDirection);

if (angle > 90)

{

newDirection = -newDirection;

}

if (angle > 20 && angle < 160)

{

speed \*= Mathf.Abs(angle - 90) / 70;

}

move = new Vector3(newDirection.x, 0, newDirection.y);

}

\_rigidbody.velocity = move \* Time.fixedDeltaTime \* speed;

}

}

public void Stop()

{

\_rigidbody.velocity = Vector3.zero;

gameObject.SetActive(!gameObject.activeSelf);

}

}

Скрипт PlayerInteract используется для того, чтобы игрок мог взаимодействовать с предметами и другими объектами окружения.

using UnityEngine;

using TMPro;

using UnityEngine.InputSystem;

public class PlayerInteract : MonoBehaviour

{

[SerializeField] private float \_maxUseDistance = 5f;

private PlayerInput \_playerInput;

private Transform \_camera;

private GameObject \_interactHintUI;

private MonoBehaviour \_currentObject;

private MonoBehaviour CurrentObject

{

get => \_currentObject;

set

{

if (\_currentObject != null &&

\_currentObject is IOutlineable oldOutlineable)

{

oldOutlineable.TurnHighlight();

}

\_currentObject = value;

if (\_currentObject is IOutlineable newOutlineable)

{

newOutlineable.TurnHighlight();

}

\_interactHintUI.SetActive(\_currentObject != null);

}

}

private void Start()

{

\_playerInput = Root.Instance.Player.PlayerInput;

\_camera = Camera.main.transform;

\_interactHintUI = Root.Instance.MainUI.InteractHintUI;

\_playerInput.actions["Use"].performed += ctx => OnUse();

}

private void OnUse()

{

if(CurrentObject is InteractableObject interactable)

{

interactable.Interact();

}

}

private void FixedUpdate()

{

if (Physics.Raycast(\_camera.position, \_camera.forward, out RaycastHit hit, \_maxUseDistance))

{

if (hit.collider.TryGetComponent<InteractableObject>(out InteractableObject interactable))

{

CurrentObject = interactable;

}

else

{

CurrentObject = null;

}

}

else

{

CurrentObject = null;

}

}

}

Скрипт UIController нужен для показа и скрытия to do листа

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.InputSystem;

public class UIController : MonoBehaviour

{

[SerializeField] private Animator \_questsListAnimator;

[SerializeField] private GameObject \_interactHintUI;

[SerializeField] private DialogueUI \_dialogueUI;

[SerializeField] private InventoryUI \_inventoryUI;

private PlayerInput \_playerInput;

public GameObject InteractHintUI { get => \_interactHintUI;}

public DialogueUI DialogueUI { get => \_dialogueUI;}

public InventoryUI InventoryUI { get => \_inventoryUI;}

private void Start()

{

\_playerInput = Root.Instance.Player.PlayerInput;

\_playerInput.actions["ShowQuests"].performed += ctx => OnShowQuests();

}

private void OnShowQuests()

{

bool isOpen = \_questsListAnimator.GetBool("isOpen");

\_questsListAnimator.SetBool("isOpen", !isOpen);

}

}

Скрипт DialogueUI используется для показа и скрытия диалоговых окон интерфейса.

using System;

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.UI;

public class DialogueUI : MonoBehaviour

{

public Action OnDialogueEnded;

[SerializeField] private TMP\_Text \_dialogueText;

[SerializeField] private Button \_endButton;

private void Awake()

{

\_endButton.onClick.AddListener(() => OnDialogueEnded?.Invoke());

}

public void SetDialog(Dialogue dialogue)

{

\_dialogueText.text = dialogue.TextDialogue;

}

}

Скрипт Dialogue – это скрипт для создания Scriptable Objects.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[CreateAssetMenu(fileName = "dialogue", menuName = "Dialogue")]

public class Dialogue : ScriptableObject

{

[SerializeField] private string \_textDialogue;

public string TextDialogue { get => \_textDialogue; }

}

Скрипт DocumentOpen запускает анимацию открытия и закрытия папки с документами.

using AYellowpaper;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class DocumentOpen : InteractableObject

{

[SerializeField] private InterfaceReference<ICompleteable, MonoBehaviour> \_condition;

private bool \_isOpen = false;

private Animator \_animator;

private void Awake()

{

\_animator = gameObject.GetComponent<Animator>();

}

public override void Interact()

{

if(\_condition.Value.IsCompleted)

{

\_isOpen = !\_isOpen;

\_animator.SetBool("isOpen", \_isOpen);

OnInteracted?.Invoke();

}

}

}

Скрипт DocWrite используется для подписи документа

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class DocWrite : InteractableObject, ICompleteable

{

[SerializeField] private Sprite \_sprite;

private SpriteRenderer \_objSprite;

private bool \_isCompleted;

public Action OnCompleted { get; set; }

public bool IsCompleted

{

get => \_isCompleted;

private set

{

\_isCompleted = value;

if (\_isCompleted)

{

OnCompleted?.Invoke();

}

}

}

private void Awake()

{

\_objSprite = gameObject.GetComponent<SpriteRenderer>();

}

public override void Interact()

{

\_objSprite.sprite = \_sprite;

IsCompleted = true;

OnInteracted?.Invoke();

}

}

Скрипт Quest нужен для проверки статусов заданий из списка задач.

using AYellowpaper;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data.SqlTypes;

using System.Linq;

using TMPro;

using UnityEngine;

[RequireComponent(typeof(TMP\_Text))]

public class Quest : MonoBehaviour, ICompleteable

{

[SerializeField] private Item[] \_questItems;

[SerializeField] private InterfaceReference<ICompleteable,MonoBehaviour>[] \_questSteps;

private Inventory \_inventory;

private TMP\_Text \_questPresenter;

private bool \_isCompleted;

public Action OnCompleted { get; set; }

public bool IsCompleted

{

get => \_isCompleted;

private set

{

\_isCompleted = value;

if (\_isCompleted)

{

\_inventory.OnInventoryChanged -= UpdateStatus;

OnCompleted?.Invoke();

}

}

}

private void Awake()

{

\_questPresenter = GetComponent<TMP\_Text>();

OnCompleted += () => \_questPresenter.fontStyle = FontStyles.Strikethrough;

}

public void Start()

{

\_inventory = Root.Instance.Player.Inventory;

\_inventory.OnInventoryChanged += UpdateStatus;

foreach (var \_questStep in \_questSteps)

{

\_questStep.Value.OnCompleted += UpdateStatus;

}

Root.Instance.QuestList.AddQuest(this);

}

private void UpdateStatus()

{

bool isCompleted = true;

isCompleted = isCompleted && \_questSteps.All(x => x.Value.IsCompleted == true);

isCompleted = isCompleted && \_inventory.CheckItems(\_questItems);

IsCompleted = isCompleted;

}

}

В скрипте QuestList хранятся все задания.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class QuestList : MonoBehaviour

{

private List<Quest> \_quests = new List<Quest>();

public void AddQuest(Quest quest)

{

\_quests.Add(quest);

}

}

Скрипт TalkingNPC запускает разговор с NPC при нажатии клавиши “E”.

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class TalkingNPC : InteractableObject, ICompleteable

{

[SerializeField] private Dialogue \_dialogue;

private DialogueUI \_dialogueUI;

private UIController \_mainUI;

private PlayerController \_playerController;

private bool \_isCompleted;

public Action OnCompleted { get; set; }

public bool IsCompleted {

get => \_isCompleted;

private set

{

\_isCompleted = value;

if (\_isCompleted)

{

OnCompleted?.Invoke();

}

}

}

private void Start()

{

IsCompleted = false;

\_playerController = Root.Instance.Player;

\_mainUI = Root.Instance.MainUI;

\_dialogueUI = Root.Instance.MainUI.DialogueUI;

}

public override void Interact()

{

\_playerController.TurnPlayer();

Cursor.lockState = CursorLockMode.None;

\_dialogueUI.SetDialog(\_dialogue);

\_dialogueUI.gameObject.SetActive(true);

\_dialogueUI.OnDialogueEnded += EndDialog;

\_mainUI.gameObject.SetActive(false);

OnInteracted?.Invoke();

}

public void EndDialog()

{

\_playerController.TurnPlayer();

Cursor.lockState = CursorLockMode.Locked;

\_dialogueUI.OnDialogueEnded -= EndDialog;

\_dialogueUI.gameObject.SetActive(false);

\_mainUI.gameObject.SetActive(true);

IsCompleted = true;

}

}

Скрипт Door нужен корректной работы дверей.

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[RequireComponent(typeof(Rigidbody))]

public class Door : InteractableObject

{

[SerializeField] private float \_speed = 1f;

[SerializeField] private float \_rotationAmount = 90f;

[SerializeField] private bool \_isRight = false;

private bool \_isOpen = false;

private bool \_isRotating = false;

private float \_time;

private Vector3 \_startRotation;

private Quaternion \_endRotation;

private IEnumerator \_stateCoroutine;

private Rigidbody \_rigidbody;

private void Start()

{

\_startRotation = transform.rotation.eulerAngles;

\_rigidbody = GetComponent<Rigidbody>();

}

private int RotateModifier

{

get

{

int openModifier =\_isOpen ? 1 : -1;

int positionModifier = \_isRight ? 1 : -1;

return openModifier \* positionModifier;

}

}

public override void Interact()

{

if (\_isRotating)

{

StopCoroutine(\_stateCoroutine);

\_isRotating = false;

\_time = 1 - \_time;

}

else

{

\_time = 0;

}

\_isOpen = !\_isOpen;

\_stateCoroutine = ChangeState();

StartCoroutine(\_stateCoroutine);

OnInteracted?.Invoke();

}

private IEnumerator ChangeState()

{

\_isRotating = true;

var rotationVector = \_startRotation;

float x = rotationVector[0];

float y = rotationVector[1];

float z = rotationVector[2];

if (\_isOpen)

{

y += \_rotationAmount \* RotateModifier;

}

\_endRotation = Quaternion.Euler(x, y, z);

Vector3 rotation = new Vector3(0, \_rotationAmount \* RotateModifier, 0);

while (Quaternion.Angle(\_endRotation, transform.rotation) > 1)

{

Quaternion deltaRotation = Quaternion.Euler(rotation \* Time.fixedDeltaTime \* \_speed);

\_rigidbody.MoveRotation(\_rigidbody.rotation \* deltaRotation);

yield return null;

}

transform.rotation = \_endRotation;

\_isRotating = false;

}

}

Скрипт ItemStorage используется для создания объектов в запущенном проекте.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[RequireComponent(typeof(Outline))]

public class ItemStorage : InteractableObject, IOutlineable

{

[SerializeField] private Item \_item;

private Outline \_outline;

private void Awake()

{

\_outline = GetComponent<Outline>();

}

public override void Interact()

{

var spawned = Instantiate(\_item.Presenter);

spawned.Init(\_item);

spawned.transform.position = transform.position + transform.rotation \* Vector3.forward;

spawned.transform.rotation = transform.rotation;

OnInteracted?.Invoke();

}

public void TurnHighlight()

{

\_outline.enabled = !\_outline.enabled;

}

}

ItemPresenter нужен для того, чтобы обводить предметы, с которыми можно взаимодействовать.

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[RequireComponent(typeof(Outline))]

public class ItemPresenter : InteractableObject, IOutlineable

{

private Outline \_outline;

private Item \_item;

public void Init(Item item)

{

\_item = item;

}

private void Awake()

{

\_outline = GetComponent<Outline>();

}

public override void Interact()

{

if (Root.Instance.Player.Inventory.TryPickupItem(\_item))

{

Destroy(gameObject);

}

OnInteracted?.Invoke();

}

public void TurnHighlight()

{

\_outline.enabled = !\_outline.enabled;

}

}

# **Визуальная часть проекта**

На рисунке 1 продемонстрирован интерфейс персонажа.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис 1. Интерфейс персонажа.

На рисунке 2 представлена локация.

Изображение выглядит как текст, снимок экрана, Параллельный, карта

Автоматически созданное описание

На рисунке 3 изображён интерфейс диалога с персонажем.

Изображение выглядит как снимок экрана, 3D-моделирование

Автоматически созданное описание

Рис 3. Интерфейс диалога.

На рисунке 4 представлено изменение инвентаря при подборе предметов.

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание

Рис 4. Изменение инвентаря.

На рисунке 5 изображён список задач.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис 5. Список задач.

# **ВЫВОД**

Я прошёл ПМ.02 и обучился много новому выполняя задания, которые давали в компании.