Министерство науки и высшего образования

Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

Утверждаю

Зам. директора по УПР

С.Ю. Назаров

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**ОТЧЕТ**

**по производственной практике (по профилю специальности)**

по профессиональному модулю

ПМ.02 Разработка и администрирование баз данных

(код и наименование)

Специальность09.02.03 Программирование в КС .

(код и наименование специальности)

Студент III курса 32928/1 группы

Григорьев Александр Сергеевич

(Фамилия, имя, отчество)

Место прохождения практики ООО «ИАЦ», ул. Черняховского, 59, СПб

(наименование и адрес организации)

Период прохождения практики

с «09» мая 2022 г. по «26» июня 2022 г.

Руководители практики:

От колледжа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Костин Ю.Н.

(подпись) (Ф.И.О.)

От предприятия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_.

(подпись) (Ф.И.О.)

М.П.

Итоговая оценка по практике (от колледжа): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2022

Утверждаю

Зам. директора по УПР

С.Ю. Назаров

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**Задание на производственную практику**

по профессиональному модулю

ПМ.02 Разработка и администрирование баз данных

(код и наименование)

Специальность09.02.03 Программирование в КС .

(код и наименование специальности)

Студент III курса 32928/1 группы

Григорьев Александр Сергеевич

(Фамилия, имя, отчество)

Место прохождения практики ООО «ИАЦ», ул. Черняховского 59, СПб

(наименование и адрес организации)

Период прохождения практики

с «09» мая 2022 г. по «26» июня 2022 г.

**ЧАСТЬ I. Индивидуальное задание по созданию телеграмм бота, с использованием веб-приложения:**

1) Подключить веб-приложение к хостингу

2) Детальное изучение механик и функционала веб-приложений

**ЧАСТЬ II. Виды работ, обязательных для выполнения:**

1) Создание проекта и разработка плана выполнения работ.

2) Разработка структуры и стилей веб-приложения.

3) Разработка основного функционала веб-приложения.

4) Подключение веб-приложения к телеграмм боту и установка веб-приложения на хостинг.

5) Проверка функционала.

6) Финальный код программы.

**ПРИЛОЖЕНИЯ:**

Техническое задание

Задание выдал «09» мая 2022 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Костин Ю.Н.

(подпись) (Ф.И.О.)

С заданием ознакомлен:

«09» мая 2022 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Григорьев А.С.

(подпись) (Ф.И.О. студента)

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ ПП.02.01**

по профессиональному модулю

ПМ.02 «Разработка и администрирование баз данных»

(код и наименование)

Специальность 09.02.03 «Программирование в компьютерных системах»

(код и наименование специальности)

Студент 3 курса 32928/1 группы

Григорьев Александр Сергеевич

(фамилия, имя, отчество)

Место прохождения практики

Общество с ограниченной ответственностью «ИАЦ», ул. Черняховского 59, СПб

(наименование и адрес организации)

Период прохождения практики

с «09» мая 2022 г. по «26» июня 2022 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| Виды выполненных работ обучающимся  во время практики | Объем работ, час. | Качество выполнения работ (оценка по пятибалльной системе) |
| Тема 1. Анализ предметной области | 24 |  |
| Тема 2. Проектирование схемы БД | 36 |  |
| Тема 3. Проектирование интерфейсов | 24 |  |
| Тема 4. Разработка базы данных и приложения | 40 |  |
| Тема 5. Тестирование и выпуск приложения | 20 |  |
| Тема 6. Индивидуальное задание с места производства (проект) | 144 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время производственной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены/не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Костин Ю.Н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «09» мая 2022 г.

Министерство науки и высшего образования

Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики**

по профессиональному модулю

ПМ.02 Разработка и администрирование баз данных

(код и наименование)

Специальность09.02.03 Программирование в КС .

(код и наименование специальности)

Студент III курса 32928/1 группы

Григорьев Александр Сергеевич

(Фамилия, имя, отчество)

Место прохождения практики ООО «ИАЦ», ул. Черняховского, 59, СПб

(наименование и адрес организации)

Период прохождения практики

с «09» мая 2022 г. по «26» июня 2022 г.

Руководитель(и) практики:

\_\_\_\_\_\_\_\_\_\_\_\_ Костин Ю.Н.

(подпись) (Ф.И.О.)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2022

**Содержание дневника**

| Дата | Виды выполненных работ и заданий по программе практики | Подпись ответственного лица по практической подготовке от организации |
| --- | --- | --- |
| 09.05 | Изучение функционала и работы с веб-приложениями для бота в телеграмм |  |
| 12.05 | Регистрация хостинга. |  |
| 15.05 | Создание бота телеграмм через @BotFather и подключение его к хостингу. |  |
| 21.05 | Создание базовых инструментов для взаимодействия с ботом с помощью Python. |  |
| 25.05 | Промежуточный день, сдача работ на проверку куратору практики. |  |
| 29.05 | Разработка структуры и стилей сайта . |  |
| 06.06 | Разработка функционала игры сапер с помощью JavaScript. |  |
| 09.06 | Проверка работы программы, исправление ошибок. |  |
| 14.06 | Работа над функционированием определения типа документа, проверка работы с сложными документами типа: Акт, накладная. |  |
| 20.06 | Завершение работы над программой, исправление ошибок, составление финальной версии. |  |
| 26.06 | Составление отчета о проделанной работе |  |

**ХАРАКТЕРИСТИКА**

Студент-практикант Григорьев Александр Сергеевич в период с 9.05.2022 г. по 26.06.2022 г. проходил производственную практику в ООО «ИАЦ», расположенном по адресу: ул.Черняховского, 59, СПб в качестве помощника канцелярии.

В течение всего периода практики, Григорьев Александр Сергеевич внимательно и ответственно относился к выполняемой работе. Изучал методы управления и основные методы мотивации персонала, должностные обязанности персонала, пробовал проводить анализ хозяйственной деятельности организации за последние три года, ознакомился с порядком ведения различных распорядительных документов, учетной документации, изучал организационную структуру предприятия, участвовал в различной повседневной работе.

Всю порученную работу выполнял добросовестно и в срок. Стремился приобретать новые знания, чтобы быть ещё более полезным на месте практики. Неоднократно оказывал помощь сотрудникам организации.

Замечаний по прохождению практики к Григорьев Александр Сергеевич нет.

20 июля 2022 г.

*.*

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Место работы и должность руководителя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Ф.И.О. руководителя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2022 г. Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_

М.П.

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 8](#_Toc107204080)

[ВВЕДЕНИЕ 9](#_Toc107204081)

[Виды работ, обязательных для выполнения 10](#_Toc107204082)

[Создание проекта и разработка плана выполнения работ 10](#_Toc107204083)

[Разработка структуры и стилей веб-приложения 10](#_Toc107204084)

[Разработка основного функционала веб-приложения. 19](#_Toc107204085)

[Подключение веб-приложения к телеграмм боту и установка веб-приложения на хостинг. 27](#_Toc107204086)

[Проверка функционала 28](#_Toc107204087)

[Финальный код 33](#_Toc107204088)

[CSS 33](#_Toc107204089)

[HTML 41](#_Toc107204090)

[JAVA SCRIPT 43](#_Toc107204091)

[PYTHON 52](#_Toc107204092)

[ЗАКЛЮЧЕНИЕ 53](#_Toc107204093)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 54](#_Toc107204094)

# ВВЕДЕНИЕ

Предприятие, на котором мы проходили производственную практику называется ООО «ИАЦ», на данном предприятии занимаются информатизацией информационного обеспечения органов государственной власти Санкт-Петербурга и других организаций, а также предоставления услуг в сфере создания и использования современных информационных и телекоммуникационных систем, средств и технологий.

Автоматизация процессов повышает качество и эффективность взаимодействия между всеми отделами предприятия. В этом случае конечная цель деятельности предприятия достигается быстрее.

Во время практики нам выдали задание по работе с телеграмм ботом.

# Виды работ, обязательных для выполнения

## Создание проекта и разработка плана выполнения работ

Для работы мною была выбрана самая передовая и комфортная IDE в области разработки на языке Python – PyCharm Community Edition 2022.1.1 , а также был создан проект с использованием IDE для web-разработки Visual Studio Code. Я создал проект в соответствии с моими нуждами и приступил к разработке плана выполнения работ.

План выполнения работ:

1. Я решил начать с так называемого “фронтэнда”, поэтому первым делом надо разработать структуру и стили самого веб-приложения.
2. После “фронтэнда”, как и полагается я занялся “бэкэндом” – наполнял веб-приложение основными функциями для работы.
3. После завершения работы над веб-приложением, нужно поставить его на хостинг, а затем подключить все к Telegram-Боту, которого я создам позднее.
4. Подводя итоги главное проверить работу функционала и всей проделанной работы, поэтому 5 пунктом проведу финальные тесты.

## Разработка структуры и стилей веб-приложения

Начинать я буду со структуры. Создадим файл с расширением .html , который и будет являться основой веб-приложения.

В head’e нашего документа разберемся с кодировкой, оставим место, чтобы в дальнейшем подключить туда нужные нам скрипты – я уже планирую подключать jquery для удобного пользования запросами, telegram-web-app.js для работы с Телеграмом, а также мой собственный скрипт, для работы веб-приложения. Также оставим место под стили для нашего документа. (о них далее после разметки)

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Telegram Web App</title>

    <script src="https://telegram.org/js/telegram-web-app.js"></script>

    <link rel="stylesheet" href="css/index.css">

    <script src="scripts/minesweeper.js" defer></script>

    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

</head>

Далее приступаем к самой верстке – расстановке блоков, внутри которых должны быть селекционно вмещены: radio-кнопки для выбора сложности, переключать режима для постановки флажков и прочее, связанное с нашим приложением.

<body>

    <div class="columns">

        <div class="col-left">

            <div class="settings">

                <label for="boardSize">Размер поля:</label>

                <input id="boardSize" type="range" value="10" min="6" max="15">

                <fieldset>

                    <input type="radio" id="easy" name="difficulty" value="0.1" class="difficulty">

                    <label for="easy">Легко</label>

                    <input type="radio" id="normal" name="difficulty" value="0.2" class="difficulty" checked>

                    <label for="normal">Нормально</label>

                    <input type="radio" id="hard" name="difficulty" value="0.4" class="difficulty">

                    <label for="hard">Сложно</label>

                </fieldset>

            </div>

        </div>

        <div class="col-right">

            <label class="switch">

                <div class="slider-text">

                    <p>Поставить флаг</p>

                </div>

                <div class="slider-visual">

                    <input class="slider-checkbox" type="checkbox">

                    <span class="slider round"></span>

                </div>

            </label>

            <a href="#!" class="minesweeper-btn">Новая игра</a>

        </div>

    </div>

    <div class="board-wrap">

        <div class="board"></div>

    </div>

    <div class="endscreen"></div>

</body>

</html>

После подготовки структуры займемся стилями.

Вернемся к уже ранее созданному пустому файлу со стилями.

Начну прописывать стили с сообщения, которое будет сигнализировать о конечном результате игры – поражение или выигрыш.

.endscreen {

background: var(--tg-theme-bg-color);

padding: 20px;

text-align: center;

font-size: 30px;

display: none;

color: red;

font-family: 'Roboto Condensed', sans-serif;

}

.endscreen.show {

display: block;

}

.endscreen span {

color: green;

}

Идя далее по интерфейсу – распишем стиль и анимации для слайдера (для установки размера поля) и переключателя (для постановки флажков)

.columns {

margin-top: -10px;

}

.col-left {

display: inline-block;

width: 48%;

}

.col-right {

display: inline-block;

position: relative;

top: -60px;

width:48%;

text-align: right;

}

.switch {

position: relative;

display: inline-block;

width: 30px;

height: 17px;

}

.slider-text {

position: relative;

top: -15px;

left: -155px;

width: 150px;

font-size: 14px;

font-family: Tahoma, sans-serif;

color:var(--tg-theme-button-color);

font-weight: bold;

}

.switch input {

display:none;

}

.slider {

position: absolute;

cursor: pointer;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: #ccc;

-webkit-transition: .4s;

transition: .4s;

}

.slider:before {

position: absolute;

content: "";

height: 13px;

width: 13px;

left: 2px;

bottom: 2px;

background-color: var(--tg-theme-bg-color);

-webkit-transition: .4s;

transition: .4s;

}

input:checked + .slider {

background-color: var(--tg-theme-button-color);

}

input:focus + .slider {

box-shadow: 0 0 1px var(--tg-theme-button-color);

}

input:checked + .slider:before {

-webkit-transform: translateX(13px);

-ms-transform: translateX(13px);

transform: translateX(13px);

}

.slider.round {

border-radius: 17px;

}

.slider.round:before {

border-radius: 50%;

}

После основы настроек можно приступить к стилизации борда и поля для игры (классы board и minesweeper). Также пропишем аккуратные цвета и анимации, воспользуемся удобным сортировщиком justify - ничего сложного.

.board {

background: #e2e2e2;

position: relative;

display: flex;

flex-direction: row;

flex-wrap: wrap;

margin: 0px auto;

border: 8px solid var(--tg-theme-button-color);

box-sizing: content-box;

box-shadow: 0 4px 12px rgba(0,0,0,0.2), 0 16px 20px rgba(0,0,0,0.2);

}

.board .tile {

background: #e2e2e2;

width: var(--tileSize);

height: var(--tileSize);

position: relative;

display: flex;

justify-content: center;

align-items: center;

font-size: calc(var(--tileSize) \* .8);

cursor: pointer;

user-select: none;

transition: background .2s ease;

font-family: Tahoma, sans-serif;

box-shadow: inset 0 2px 0 0 #fcfcfc, inset 2px 0 0 0 #fcfcfc, inset 0 -2px 0 0 #c9c9c9, inset -2px 0 0 0 #c9c9c9;

}

.board .tile:hover:not(.tile--checked) {

background: #efefef;

}

.board .tile--checked {

background: #d5d5d5;

box-shadow: inset 0 1px 0 0 #c9c9c9, inset 1px 0 0 0 #c9c9c9, inset 0 -1px 0 0 #c9c9c9, inset -1px 0 0 0 #c9c9c9;

}

.board .tile--checked.tile--bomb:before, .board .tile--checked.tile--bomb:after {

opacity: 1;

}

.board .tile--checked:hover {

cursor: default;

}

.board .tile--bomb {

font-size: calc(var(--tileSize) \* .5);

}

.board .tile--flagged {

font-size: calc(var(--tileSize) \* .5);

}

.minesweeper-btn {

display: inline-block;

margin-top: -100px;

padding: 0 40px;

height: 25px;

text-align: center;

font-family: 'Roboto Condensed', sans-serif;

background-color: var(--tg-theme-button-color);

border-radius: 2px;

font-size: 12px;

font-weight: bold;

color: var(--tg-theme-bg-color);

text-transform: uppercase;

line-height: 25px;

transition: all 0.4s;

outline: none !important;

border: none;

text-decoration: none;

}

.minesweeper-btn:hover {

background-color: var(--tg-theme-hint-color);

color: var(--tg-theme-bg-color);

}

Ну и для закрепления стиля нашего веб-приложения – разберемся с основными настройками, radio-кнопками.

.settings fieldset {

border: none;

padding-bottom: 0px;

}

.settings input[type=radio] {

background: var(--tg-theme-button-color);

display:none;

}

.settings label {

margin: 0px 0;

display: inline-block;

position: relative;

cursor:pointer;

font-size: 13px;

line-height:18px;

transition: .4s all;

color:var(--tg-theme-button-color);

font-weight: bold!important;

}

.settings input[type=radio] + label {

margin: 0px 14px 12px 22px;

}

.settings input[type=radio] + label:hover {

color: var(--tg-theme-hint-color);

}

.settings input[type=radio] + label:before {

position: absolute;

content: '';

width: 18px;

height: 18px;

left:-24px;

border: 2px solid var(--tg-theme-button-color);

border-radius: 2px;

transition: background .4s;

}

.settings input[type=radio] + label:before {

border-radius: 100%;

}

.settings input[type=radio]:checked + label:before {

background: var(--tg-theme-button-color);

}

## Разработка основного функционала веб-приложения.

Основной функционал – игра в Сапёра. Открываем созданный ранее ***minesweeper.js*** и начем с обработки html-документа: получения блоков (div), обозначения переменных, а также параметров для дальнейшей игры.

console.clear();

let size = 10;

let bombFrequency = 0.2;

let tileSize = 25;

let max\_size = 250;

const board = document.querySelectorAll('.board')[0];

let tiles;

let boardSize;

const restartBtn = document.querySelectorAll('.minesweeper-btn')[0];

const endscreen = document.querySelectorAll('.endscreen')[0]

const boardSizeBtn = document.getElementById('boardSize');

const tileSizeBtn = document.getElementById('tileSize');

const difficultyBtns = document.querySelectorAll('.difficulty');

let bombs = [];

let numbers = [];

let numberColors = ['#3498db', '#2ecc71', '#e74c3c', '#9b59b6', '#f1c40f', '#1abc9c', '#34495e', '#7f8c8d',];

let endscreenContent = {win: '<span>Поздавляю, победа!</span>', loose: 'Увы, проигрыш.'};

let gameOver = false;

Теперь самое трудное – опишем две большие функции в нашем приложении.

1. Clear() – функция для отчистки поля, будет использоваться в случае если пользователь проиграл или только попал на страницу.
2. Setup() – функция для подготовки поля к игре – подсчитает нужное кол-во tile’ов для игры, учитывая все настройки пользователя и на основе подсчетов сгенерирует поле для игры.

const clear = () => {

    gameOver = false;

    bombs = [];

    numbers = [];

    endscreen.innerHTML = '';

    endscreen.classList.remove('show');

    tiles.forEach(tile => {

        tile.remove();

    });

    setup();

}

const setup = () => {

    for (let i = 0; i < Math.pow(size, 2); i++) {

        const tile = document.createElement('div');

        tile.classList.add('tile');

        board.appendChild(tile);

    }

    tiles = document.querySelectorAll('.tile');

    boardSize = Math.sqrt(tiles.length);

    board.style.width = boardSize \* tileSize + 'px';

    console.log(boardSize, tileSize);

    document.documentElement.style.setProperty('--tileSize', `${tileSize}px`);

    document.documentElement.style.setProperty('--boardSize', `${boardSize \* tileSize}px`);

    let x = 0;

    let y = 0;

    tiles.forEach((tile, i) => {

        tile.setAttribute('data-tile', `${x},${y}`);

        let random\_boolean = Math.random() < bombFrequency;

        if (random\_boolean) {

            bombs.push(`${x},${y}`);

            if (x > 0) numbers.push(`${x-1},${y}`);

            if (x < boardSize - 1) numbers.push(`${x+1},${y}`);

            if (y > 0) numbers.push(`${x},${y-1}`);

            if (y < boardSize - 1) numbers.push(`${x},${y+1}`);

            if (x > 0 && y > 0) numbers.push(`${x-1},${y-1}`);

            if (x < boardSize - 1 && y < boardSize - 1) numbers.push(`${x+1},${y+1}`);

            if (y > 0 && x < boardSize - 1) numbers.push(`${x+1},${y-1}`);

            if (x > 0 && y < boardSize - 1) numbers.push(`${x-1},${y+1}`);

        }

        x++;

        if (x >= boardSize) {

            x = 0;

            y++;

        }

        tile.addEventListener('click', function(e) {

            var check\_status = $(".slider-checkbox").prop("checked");

            if (check\_status)

                flag(tile);

            else

                clickTile(tile);

        });

    });

    numbers.forEach(num => {

        let coords = num.split(',');

        let tile = document.querySelectorAll(`[data-tile="${parseInt(coords[0])},${parseInt(coords[1])}"]`)[0];

        let dataNum = parseInt(tile.getAttribute('data-num'));

        if (!dataNum) dataNum = 0;

        tile.setAttribute('data-num', dataNum + 1);

    });

}

Далее опишем функцию для постановки флажков flag()

const flag = (tile) => {

    if (gameOver) return;

    if (!tile.classList.contains('tile--checked')) {

        if (!tile.classList.contains('tile--flagged')) {

            tile.innerHTML = '🚩';

            tile.classList.add('tile--flagged');

            } else {

            tile.innerHTML = '';

            tile.classList.remove('tile--flagged');

        }

    }

}

Теперь напишем функции обработчики для нажатий на клетки, а также пропишем одну из основных механик проверки свободных клеток

const clickTile = (tile) => {

    if (gameOver) return;

    if (tile.classList.contains('tile--checked') || tile.classList.contains('tile--flagged')) return;

    let coordinate = tile.getAttribute('data-tile');

    if (bombs.includes(coordinate)) {

        endGame(tile);

        } else {

        let num = tile.getAttribute('data-num');

        if (num != null) {

            tile.classList.add('tile--checked');

            tile.innerHTML = num;

            tile.style.color = numberColors[num-1];

            setTimeout(() => {

                checkVictory();

            }, 100);

            return;

        }

        checkTile(tile, coordinate);

    }

    tile.classList.add('tile--checked');

}

const checkTile = (tile, coordinate) => {

    console.log('вњ”');

    let coords = coordinate.split(',');

    let x = parseInt(coords[0]);

    let y = parseInt(coords[1]);

    setTimeout(() => {

        if (x > 0) {

            let targetW = document.querySelectorAll(`[data-tile="${x-1},${y}"`)[0];

            clickTile(targetW, `${x-1},${y}`);

        }

        if (x < boardSize - 1) {

            let targetE = document.querySelectorAll(`[data-tile="${x+1},${y}"`)[0];

            clickTile(targetE, `${x+1},${y}`);

        }

        if (y > 0) {

            let targetN = document.querySelectorAll(`[data-tile="${x},${y-1}"]`)[0];

            clickTile(targetN, `${x},${y-1}`);

        }

        if (y < boardSize - 1) {

            let targetS = document.querySelectorAll(`[data-tile="${x},${y+1}"]`)[0];

            clickTile(targetS, `${x},${y+1}`);

        }

        if (x > 0 && y > 0) {

            let targetNW = document.querySelectorAll(`[data-tile="${x-1},${y-1}"`)[0];

            clickTile(targetNW, `${x-1},${y-1}`);

        }

        if (x < boardSize - 1 && y < boardSize - 1) {

            let targetSE = document.querySelectorAll(`[data-tile="${x+1},${y+1}"`)[0];

            clickTile(targetSE, `${x+1},${y+1}`);

        }

        if (y > 0 && x < boardSize - 1) {

            let targetNE = document.querySelectorAll(`[data-tile="${x+1},${y-1}"]`)[0];

            clickTile(targetNE, `${x+1},${y-1}`);

        }

        if (x > 0 && y < boardSize - 1) {

            let targetSW = document.querySelectorAll(`[data-tile="${x-1},${y+1}"`)[0];

            clickTile(targetSW, `${x-1},${y+1}`);

        }

    }, 10);

}

Последние из основных функций игры – ее завершение

Функция для завершения игры, а также для проверки осуществлена ли победа.

const endGame = (tile) => {

    console.log('Booom! Game over.');

    endscreen.innerHTML = endscreenContent.loose;

    endscreen.classList.add('show');

    gameOver = true;

    tiles.forEach(tile => {

        let coordinate = tile.getAttribute('data-tile');

        if (bombs.includes(coordinate)) {

            tile.classList.remove('tile--flagged');

            tile.classList.add('tile--checked', 'tile--bomb');

            tile.innerHTML = '💣';

        }

    });

}

const checkVictory = () => {

    let win = true;

    tiles.forEach(tile => {

        let coordinate = tile.getAttribute('data-tile');

        if (!tile.classList.contains('tile--checked') && !bombs.includes(coordinate)) win = false;

    });

    if (win) {

        endscreen.innerHTML = endscreenContent.win;

        endscreen.classList.add('show');

        gameOver = true;

    }

}

Ну и наконец-то добавим обработчики событий для отслеживания настроек пользователя и др.

/\* Перезапуск \*/

restartBtn.addEventListener('click', function(e) {

    e.preventDefault();

    clear();

});

/\* Изменение размера \*/

boardSizeBtn.addEventListener('change', function(e) {

    console.log(this.value);

    size = this.value;

    tileSize = max\_size / size;

    clear();

});

difficultyBtns.forEach(btn => {

    btn.addEventListener('click', function() {

        console.log(this.value);

        bombFrequency = this.value;

        clear();

    });

});

## Подключение веб-приложения к телеграмм боту и установка веб-приложения на хостинг.

С python-программой все оказалось куда проще чем с самим веб-приложением. Изучив, нужную для взаимодействия с Telegram’ом, библиотеку – создал проект в PyCharm Community.

Подключил telebot для работы.

from telebot import types  
import telebot  
  
from key import \*

Создал класс бота, предварительно создав самого бота в Telegram’е.

Создавал через @BotFather, получил Token для работы и приступил к обработчикам сообщений.

bot = telebot.TeleBot(BOT\_TOKEN)  
  
  
@bot.message\_handler(commands=['start']) # обрабатываем команду старт  
def start\_fun(message):  
 bot.send\_message(message.chat.id,  
 'Привет, я бот для проверки телеграмм webapps!)\nЗапустить тестовые страницы можно нажав на '  
 'кнопки.',  
 parse\_mode="Markdown", reply\_markup=None) # отправляем сообщение с нужной клавиатурой  
  
  
@bot.message\_handler(content\_types="text")  
def new\_mes(message):  
 start\_fun(message)  
  
  
@bot.message\_handler(content\_types="web\_app\_data") # получаем отправленные данные  
def answer(webAppMes):  
 print(webAppMes) # вся информация о сообщении  
 print(webAppMes.web\_app\_data.data) # конкретно то что мы передали в бота  
 bot.send\_message(webAppMes.chat.id, f"получили инофрмацию из веб-приложения: {webAppMes.web\_app\_data.data}")  
 # отправляем сообщение в ответ на отправку данных из веб-приложения  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bot.infinity\_polling()

## Проверка функционала



Рис.1 запуск бота

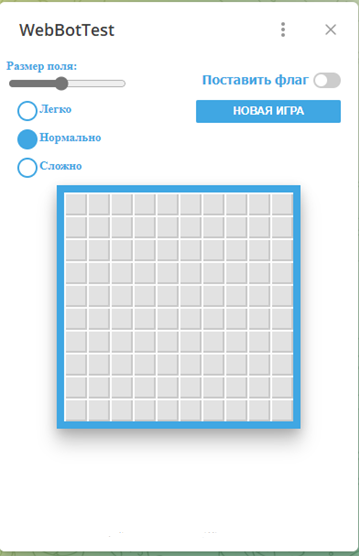


Рис.2 Запуск интерфейса веб-приложения

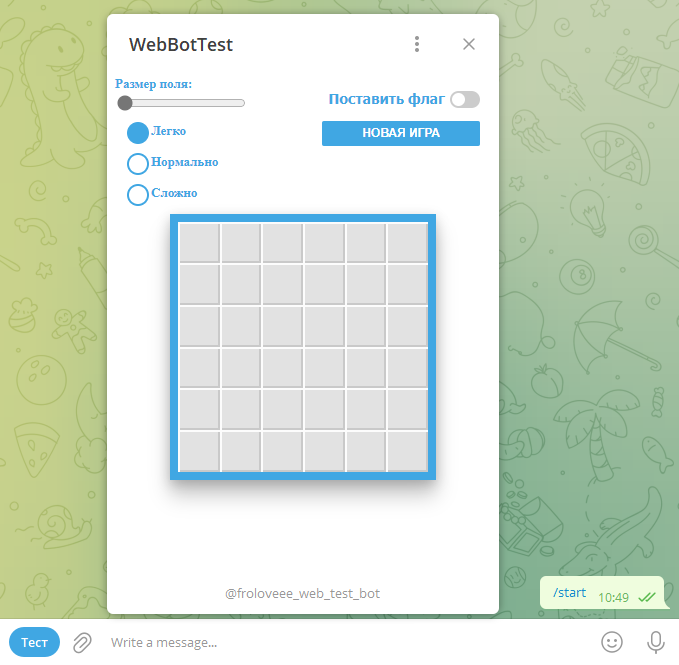
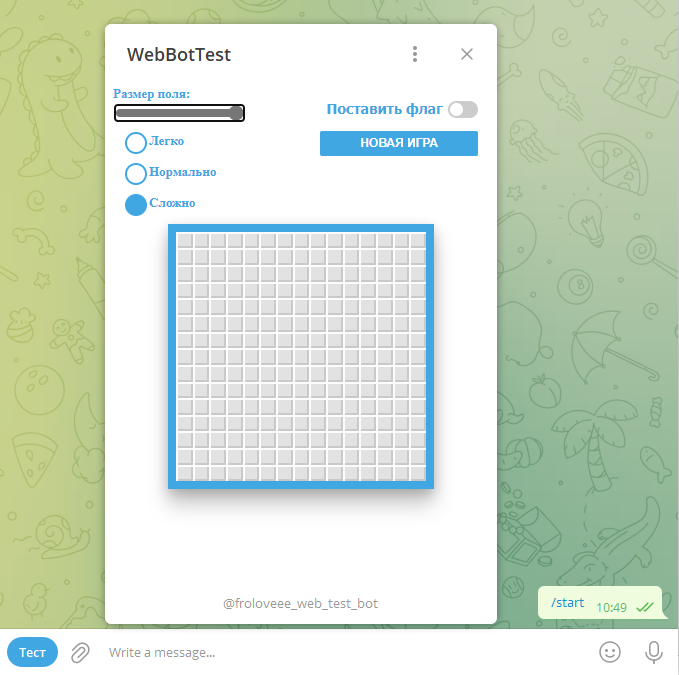


Рис. 3,4 Взаимодействие с интерфейсом

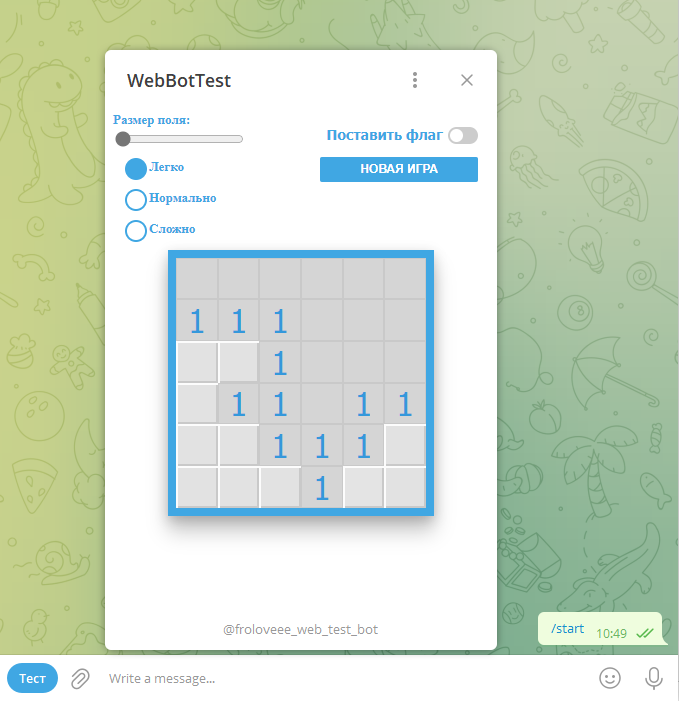


Рис. 5 Игровой процесс

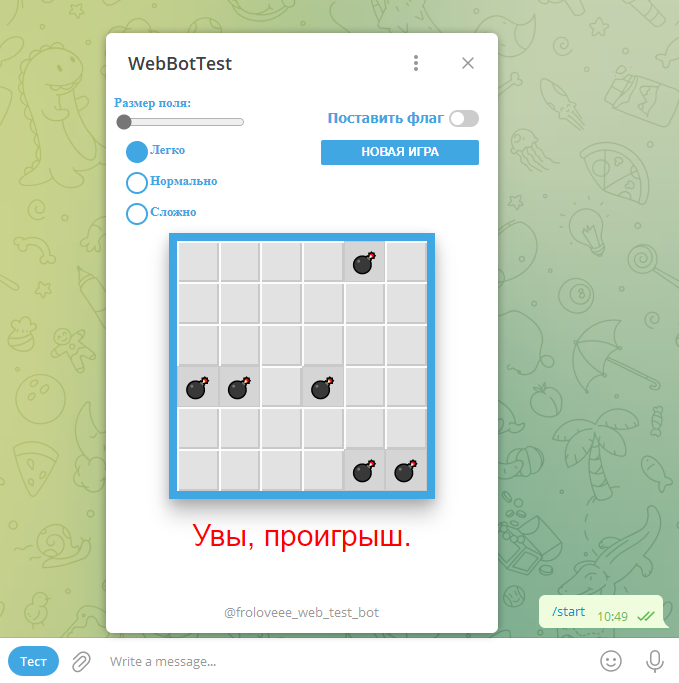


Рис. 6 Один из исходов окончания игры - проигрыш

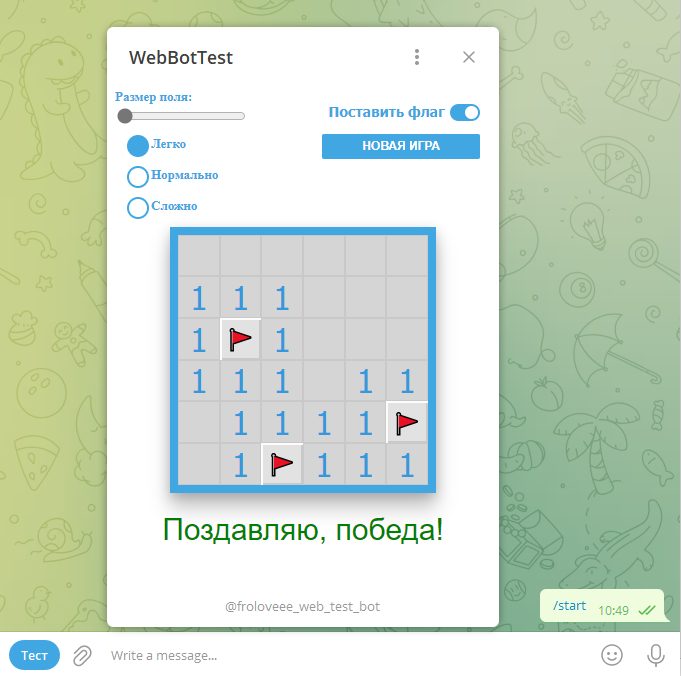


Рис. 7 Один из исходов окончания игры - победа

## Финальный код

### CSS

:root {

--tileSize: 25px;

--boardSize: 0px;

}

body {

background: var(--tg-theme-bg-color);

}

.endscreen {

background: var(--tg-theme-bg-color);

padding: 20px;

text-align: center;

font-size: 30px;

display: none;

color: red;

font-family: 'Roboto Condensed', sans-serif;

}

.endscreen.show {

display: block;

}

.endscreen span {

color: green;

}

.columns {

margin-top: -10px;

}

.col-left {

display: inline-block;

width: 48%;

}

.col-right {

display: inline-block;

position: relative;

top: -60px;

width:48%;

text-align: right;

}

.switch {

position: relative;

display: inline-block;

width: 30px;

height: 17px;

}

.slider-text {

position: relative;

top: -15px;

left: -155px;

width: 150px;

font-size: 14px;

font-family: Tahoma, sans-serif;

color:var(--tg-theme-button-color);

font-weight: bold;

}

.switch input {

display:none;

}

.slider {

position: absolute;

cursor: pointer;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: #ccc;

-webkit-transition: .4s;

transition: .4s;

}

.slider:before {

position: absolute;

content: "";

height: 13px;

width: 13px;

left: 2px;

bottom: 2px;

background-color: var(--tg-theme-bg-color);

-webkit-transition: .4s;

transition: .4s;

}

input:checked + .slider {

background-color: var(--tg-theme-button-color);

}

input:focus + .slider {

box-shadow: 0 0 1px var(--tg-theme-button-color);

}

input:checked + .slider:before {

-webkit-transform: translateX(13px);

-ms-transform: translateX(13px);

transform: translateX(13px);

}

.slider.round {

border-radius: 17px;

}

.slider.round:before {

border-radius: 50%;

}

.board {

background: #e2e2e2;

position: relative;

display: flex;

flex-direction: row;

flex-wrap: wrap;

margin: 0px auto;

border: 8px solid var(--tg-theme-button-color);

box-sizing: content-box;

box-shadow: 0 4px 12px rgba(0,0,0,0.2), 0 16px 20px rgba(0,0,0,0.2);

}

.board .tile {

background: #e2e2e2;

width: var(--tileSize);

height: var(--tileSize);

position: relative;

display: flex;

justify-content: center;

align-items: center;

font-size: calc(var(--tileSize) \* .8);

cursor: pointer;

user-select: none;

transition: background .2s ease;

font-family: Tahoma, sans-serif;

box-shadow: inset 0 2px 0 0 #fcfcfc, inset 2px 0 0 0 #fcfcfc, inset 0 -2px 0 0 #c9c9c9, inset -2px 0 0 0 #c9c9c9;

}

.board .tile:hover:not(.tile--checked) {

background: #efefef;

}

.board .tile--checked {

background: #d5d5d5;

box-shadow: inset 0 1px 0 0 #c9c9c9, inset 1px 0 0 0 #c9c9c9, inset 0 -1px 0 0 #c9c9c9, inset -1px 0 0 0 #c9c9c9;

}

.board .tile--checked.tile--bomb:before, .board .tile--checked.tile--bomb:after {

opacity: 1;

}

.board .tile--checked:hover {

cursor: default;

}

.board .tile--bomb {

font-size: calc(var(--tileSize) \* .5);

}

.board .tile--flagged {

font-size: calc(var(--tileSize) \* .5);

}

.minesweeper-btn {

display: inline-block;

margin-top: -100px;

padding: 0 40px;

height: 25px;

text-align: center;

font-family: 'Roboto Condensed', sans-serif;

background-color: var(--tg-theme-button-color);

border-radius: 2px;

font-size: 12px;

font-weight: bold;

color: var(--tg-theme-bg-color);

text-transform: uppercase;

line-height: 25px;

transition: all 0.4s;

outline: none !important;

border: none;

text-decoration: none;

}

.minesweeper-btn:hover {

background-color: var(--tg-theme-hint-color);

color: var(--tg-theme-bg-color);

}

.settings fieldset {

border: none;

padding-bottom: 0px;

}

.settings input[type=radio] {

background: var(--tg-theme-button-color);

display:none;

}

.settings label {

margin: 0px 0;

display: inline-block;

position: relative;

cursor:pointer;

font-size: 13px;

line-height:18px;

transition: .4s all;

color:var(--tg-theme-button-color);

font-weight: bold!important;

}

.settings input[type=radio] + label {

margin: 0px 14px 12px 22px;

}

.settings input[type=radio] + label:hover {

color: var(--tg-theme-hint-color);

}

.settings input[type=radio] + label:before {

position: absolute;

content: '';

width: 18px;

height: 18px;

left:-24px;

border: 2px solid var(--tg-theme-button-color);

border-radius: 2px;

transition: background .4s;

}

.settings input[type=radio] + label:before {

border-radius: 100%;

}

.settings input[type=radio]:checked + label:before {

background: var(--tg-theme-button-color);

}

### HTML

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Telegram Web App</title>

    <script src="https://telegram.org/js/telegram-web-app.js"></script>

    <link rel="stylesheet" href="css/index.css">

    <script src="scripts/minesweeper.js" defer></script>

    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

</head>

<body>

    <div class="columns">

        <div class="col-left">

            <div class="settings">

                <label for="boardSize">Размер поля:</label>

                <input id="boardSize" type="range" value="10" min="6" max="15">

                <fieldset>

                    <input type="radio" id="easy" name="difficulty" value="0.1" class="difficulty">

                    <label for="easy">Легко</label>

                    <input type="radio" id="normal" name="difficulty" value="0.2" class="difficulty" checked>

                    <label for="normal">Нормально</label>

                    <input type="radio" id="hard" name="difficulty" value="0.4" class="difficulty">

                    <label for="hard">Сложно</label>

                </fieldset>

            </div>

        </div>

        <div class="col-right">

            <label class="switch">

                <div class="slider-text">

                    <p>Поставить флаг</p>

                </div>

                <div class="slider-visual">

                    <input class="slider-checkbox" type="checkbox">

                    <span class="slider round"></span>

                </div>

            </label>

            <a href="#!" class="minesweeper-btn">Новая игра</a>

        </div>

    </div>

    <div class="board-wrap">

        <div class="board"></div>

    </div>

    <div class="endscreen"></div>

</body>

</html>

### JAVA SCRIPT

console.clear();

let size = 10;

let bombFrequency = 0.2;

let tileSize = 25;

let max\_size = 250;

const board = document.querySelectorAll('.board')[0];

let tiles;

let boardSize;

const restartBtn = document.querySelectorAll('.minesweeper-btn')[0];

const endscreen = document.querySelectorAll('.endscreen')[0]

const boardSizeBtn = document.getElementById('boardSize');

const tileSizeBtn = document.getElementById('tileSize');

const difficultyBtns = document.querySelectorAll('.difficulty');

let bombs = [];

let numbers = [];

let numberColors = ['#3498db', '#2ecc71', '#e74c3c', '#9b59b6', '#f1c40f', '#1abc9c', '#34495e', '#7f8c8d',];

let endscreenContent = {win: '<span>Поздавляю, победа!</span>', loose: 'Увы, проигрыш.'};

let gameOver = false;

const clear = () => {

    gameOver = false;

    bombs = [];

    numbers = [];

    endscreen.innerHTML = '';

    endscreen.classList.remove('show');

    tiles.forEach(tile => {

        tile.remove();

    });

    setup();

}

const setup = () => {

    for (let i = 0; i < Math.pow(size, 2); i++) {

        const tile = document.createElement('div');

        tile.classList.add('tile');

        board.appendChild(tile);

    }

    tiles = document.querySelectorAll('.tile');

    boardSize = Math.sqrt(tiles.length);

    board.style.width = boardSize \* tileSize + 'px';

    console.log(boardSize, tileSize);

    document.documentElement.style.setProperty('--tileSize', `${tileSize}px`);

    document.documentElement.style.setProperty('--boardSize', `${boardSize \* tileSize}px`);

    let x = 0;

    let y = 0;

    tiles.forEach((tile, i) => {

        tile.setAttribute('data-tile', `${x},${y}`);

        let random\_boolean = Math.random() < bombFrequency;

        if (random\_boolean) {

            bombs.push(`${x},${y}`);

            if (x > 0) numbers.push(`${x-1},${y}`);

            if (x < boardSize - 1) numbers.push(`${x+1},${y}`);

            if (y > 0) numbers.push(`${x},${y-1}`);

            if (y < boardSize - 1) numbers.push(`${x},${y+1}`);

            if (x > 0 && y > 0) numbers.push(`${x-1},${y-1}`);

            if (x < boardSize - 1 && y < boardSize - 1) numbers.push(`${x+1},${y+1}`);

            if (y > 0 && x < boardSize - 1) numbers.push(`${x+1},${y-1}`);

            if (x > 0 && y < boardSize - 1) numbers.push(`${x-1},${y+1}`);

        }

        x++;

        if (x >= boardSize) {

            x = 0;

            y++;

        }

        tile.addEventListener('click', function(e) {

            var check\_status = $(".slider-checkbox").prop("checked");

            if (check\_status)

                flag(tile);

            else

                clickTile(tile);

        });

    });

    numbers.forEach(num => {

        let coords = num.split(',');

        let tile = document.querySelectorAll(`[data-tile="${parseInt(coords[0])},${parseInt(coords[1])}"]`)[0];

        let dataNum = parseInt(tile.getAttribute('data-num'));

        if (!dataNum) dataNum = 0;

        tile.setAttribute('data-num', dataNum + 1);

    });

}

const flag = (tile) => {

    if (gameOver) return;

    if (!tile.classList.contains('tile--checked')) {

        if (!tile.classList.contains('tile--flagged')) {

            tile.innerHTML = '🚩';

            tile.classList.add('tile--flagged');

            } else {

            tile.innerHTML = '';

            tile.classList.remove('tile--flagged');

        }

    }

}

const clickTile = (tile) => {

    if (gameOver) return;

    if (tile.classList.contains('tile--checked') || tile.classList.contains('tile--flagged')) return;

    let coordinate = tile.getAttribute('data-tile');

    if (bombs.includes(coordinate)) {

        endGame(tile);

        } else {

        let num = tile.getAttribute('data-num');

        if (num != null) {

            tile.classList.add('tile--checked');

            tile.innerHTML = num;

            tile.style.color = numberColors[num-1];

            setTimeout(() => {

                checkVictory();

            }, 100);

            return;

        }

        checkTile(tile, coordinate);

    }

    tile.classList.add('tile--checked');

}

const checkTile = (tile, coordinate) => {

    console.log('вњ”');

    let coords = coordinate.split(',');

    let x = parseInt(coords[0]);

    let y = parseInt(coords[1]);

    setTimeout(() => {

        if (x > 0) {

            let targetW = document.querySelectorAll(`[data-tile="${x-1},${y}"`)[0];

            clickTile(targetW, `${x-1},${y}`);

        }

        if (x < boardSize - 1) {

            let targetE = document.querySelectorAll(`[data-tile="${x+1},${y}"`)[0];

            clickTile(targetE, `${x+1},${y}`);

        }

        if (y > 0) {

            let targetN = document.querySelectorAll(`[data-tile="${x},${y-1}"]`)[0];

            clickTile(targetN, `${x},${y-1}`);

        }

        if (y < boardSize - 1) {

            let targetS = document.querySelectorAll(`[data-tile="${x},${y+1}"]`)[0];

            clickTile(targetS, `${x},${y+1}`);

        }

        if (x > 0 && y > 0) {

            let targetNW = document.querySelectorAll(`[data-tile="${x-1},${y-1}"`)[0];

            clickTile(targetNW, `${x-1},${y-1}`);

        }

        if (x < boardSize - 1 && y < boardSize - 1) {

            let targetSE = document.querySelectorAll(`[data-tile="${x+1},${y+1}"`)[0];

            clickTile(targetSE, `${x+1},${y+1}`);

        }

        if (y > 0 && x < boardSize - 1) {

            let targetNE = document.querySelectorAll(`[data-tile="${x+1},${y-1}"]`)[0];

            clickTile(targetNE, `${x+1},${y-1}`);

        }

        if (x > 0 && y < boardSize - 1) {

            let targetSW = document.querySelectorAll(`[data-tile="${x-1},${y+1}"`)[0];

            clickTile(targetSW, `${x-1},${y+1}`);

        }

    }, 10);

}

/\* Конец игры \*/

const endGame = (tile) => {

    console.log('Booom! Game over.');

    endscreen.innerHTML = endscreenContent.loose;

    endscreen.classList.add('show');

    gameOver = true;

    tiles.forEach(tile => {

        let coordinate = tile.getAttribute('data-tile');

        if (bombs.includes(coordinate)) {

            tile.classList.remove('tile--flagged');

            tile.classList.add('tile--checked', 'tile--bomb');

            tile.innerHTML = '💣';

        }

    });

}

const checkVictory = () => {

    let win = true;

    tiles.forEach(tile => {

        let coordinate = tile.getAttribute('data-tile');

        if (!tile.classList.contains('tile--checked') && !bombs.includes(coordinate)) win = false;

    });

    if (win) {

        endscreen.innerHTML = endscreenContent.win;

        endscreen.classList.add('show');

        gameOver = true;

    }

}

/\* Запуск \*/

setup();

/\* Перезапуск \*/

restartBtn.addEventListener('click', function(e) {

    e.preventDefault();

    clear();

});

/\* Изменение размера \*/

boardSizeBtn.addEventListener('change', function(e) {

    console.log(this.value);

    size = this.value;

    tileSize = max\_size / size;

    clear();

});

difficultyBtns.forEach(btn => {

    btn.addEventListener('click', function() {

        console.log(this.value);

        bombFrequency = this.value;

        clear();

    });

});

### PYTHON

from telebot import types  
import telebot  
  
from key import \*  
  
bot = telebot.TeleBot(BOT\_TOKEN)  
  
  
@bot.message\_handler(commands=['start']) # обрабатываем команду старт  
def start\_fun(message):  
 bot.send\_message(message.chat.id,  
 'Привет, я бот для проверки телеграмм webapps!)\nЗапустить тестовые страницы можно нажав на '  
 'кнопки.',  
 parse\_mode="Markdown", reply\_markup=None) # отправляем сообщение с нужной клавиатурой  
  
  
@bot.message\_handler(content\_types="text")  
def new\_mes(message):  
 start\_fun(message)  
  
  
@bot.message\_handler(content\_types="web\_app\_data") # получаем отправленные данные  
def answer(webAppMes):  
 print(webAppMes) # вся информация о сообщении  
 print(webAppMes.web\_app\_data.data) # конкретно то что мы передали в бота  
 bot.send\_message(webAppMes.chat.id, f"получили инофрмацию из веб-приложения: {webAppMes.web\_app\_data.data}")  
 # отправляем сообщение в ответ на отправку данных из веб-приложения  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bot.infinity\_polling()

# ЗАКЛЮЧЕНИЕ

В процессе выполнения практики я проанализировал и изучил функции работы с веб-приложением для телеграмм бота, а также разработал программу, благодаря которой я изучил множество нового для себя и сделал пример работы с веб-приложением на примере игры «сапер».

Также для меня было сюрпризом писать на языке программирования, на котором мне пришлось вести работу, до этого я никогда не работал с Python, поэтому за время прохождения практики мне пришлось многому обучиться и изучить новую для меня среду программирования.

По итогам выполнения можно сделать вывод, что мною был изучено и проанализировано веб-приложение для телеграмм бота, также при работе с которым я хоть и не полностью, но погрузился в изучение нового для себя языка программирования Python и средств разработки, таких как PyCharm.

Также могу подчеркнуть для себя несколько трудностей с которыми я столкнулся в процессе выполнения задания по практике, такие как:

1) Установка веб-приложения на хостинг.

2) Непосредственно разработка игры «сапер».

Для меня это была очень сложная, но интересная производственная практика, в результате которой я многому научилися.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. pythonworld.ru Самоучитель и курсы Python: [Электронный ресурс]- https://pythonworld.ru/
2. w3schools.com Уроки и мини-упражнения по изучению языка программирования Python: [Электронный ресурс]- https://www.w3schools.com/python/