МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
Кафедра компьютерной инженерии и моделирования

РАЗРАБОТКА КЛАВИАТУРНОГО ТРЕНАЖЁРА «TYPERR»

Курсовая работа
по дисциплине «Программирование»
студента 1 курса группы ПИ-б-о-192(1)
Фамилия Имя Отчество

направления подготовки 09.03.04 «Программная инженерия»

Научный руководитель _____
(оценка)
старший преподаватель кафедры
компьютерной инженерии и моделирования _____ Чабанов В.В.
(подпись, дата)

Симферополь, 2020

РЕФЕРАТ

Разработка клавиатурного тренажёра «Typerr». – Симферополь: ФТИ КФУ им. В. И. Вернадского, 2020. – 30 с., 18 ил., 5 ист.

Объект разработки – игра клавиатурный тренажер «Typerr», сервер игры.

Цель работы – создание рабочего игрового проекта с реализацией клиент-сервер системы. Создание сервера с использованием языков PHP и SQL, реализация обмена данными приложения с сервером посредством POST запросов.

Было рассмотрено многочисленное количество вариантов реализации клиент-серверных систем, в ходе которых было принято решение использовать API подход для взаимодействия C++ клиента и PHP сервера, в виду его легкости развертывания. Это позволило проекту оторваться от ограничений локальной сети и работать более глобально с помощью сервисов бесплатного хостинга.

ПРОГРАММИРОВАНИЕ, C++, SFML, libcurl, PHP, SQL, API, игровой проект.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время очевидно, что эффективность работы человека за компьютером напрямую зависит от его скорости печати. Все больше и больше аспектов нашего современного общества затрагиваются компьютерными технологиями, а это значит, что благодаря высокой скорости и эффективности печати, человек может не только быть продуктивнее в рабочей обстановке, а так же тратить меньше времени на повседневной основе. Суть проекта заключается в создании затягивающей игры, которая поможет улучшать эффективность печати пользователя, при этом доставляя ему удовольствие.

Целями проекта с самого начала были: приятный интерфейс, звуковое сопровождение и онлайн-элементы. Для реализации этих целей понадобилось изучить стороннюю библиотеку SFML, для реализации графической составляющей приложения, разобраться в процессе работы API-систем для коммуникации приложения с сервером, написать серверную часть, используя языки PHP и SQL и изучить библиотеку libcURL для коммуникации приложения с сервером.

ГЛАВА 1
ПОСТАНОВКА ЗАДАЧИ

1.1 Цель проекта

Создание рабочего проекта, получение опыта в разработке крупных проектов, получение опыта в создании API систем. Закрепление навыков использования и внедрения сторонних C++ библиотек, их изучение. Укрепление навыков наладки работы клиент-серверных систем, верстки веб-страниц, написание сервера с использованием языка PHP, SQL.

1.2 Существующие аналоги

- Typeracer – многопользовательская онлайн-игра на основе браузера. Этот сайт предоставляет возможность многопользовательской игры в виде лобби с одним текстом и несколькими игроками. Все игроки начинают набирать текст одновременно и тот, кто доберется до конца текста первым, тот и победитель. Сайт также ведет статистику игрока (если он авторизован).
- Stamina – программа для овладения десятипальцевым набором и методом слепой печати.
- Keyhero – онлайн клавиатурный тренажер, с возможностями ведения статистики.

1.3 Основные отличия от аналогов

Данный проект отличается уникальным интерфейсом, в котором варианты пользовательского интерфейса впечатываются с клавиатуры. Визуальная оценка результата пользователя посредством системы оценивания от D до S (S, A, B, C, D). Стилистическое оформление приложения.

1.4 Техническое задание

Программа должна корректно интерпретировать и соблюдать следующие основные правила игры:

- Пользователю предоставляется выбор между онлайн или одиночным режимом игры.
- Пользователю предоставляется 10 слов для ввода.
- Все варианты меню должны вводиться с помощью клавиатуры.
- После ввода всех слов пользователю должен выдаваться результат в виде числа WPM(слов в минуту) и процента точности.
- Пользователю предоставляется возможность сохранять свою статистику после создания профиля.
- После ввода 10 слов на экран выводится результат игры.
- Два авторизованных пользователя должны иметь возможность вызвать друг друга на дуэль.
- В режиме дуэли двум пользователям должны выдаваться 10 одинаковых слов.
- Режим онлайн игры для нескольких игроков, более чем двух.
- Настройки игры, такие как количество слов, различные модификаторы, такие как сброс при ошибке и т.п.
- Программа должна предоставлять возможность сохранения результатов игры на сервер.
- Должна существовать статистика игрока в виде его среднего WPM(слов в минуту) и точности (в процентах).
- Программа должна иметь звуковое сопровождение.

ГЛАВА 2
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

2.1 Анализ инструментальных средств

- Среда Microsoft Visual Studio 2019 – была выбрана в пользу знакомого интерфейса и удобных функций интегрирования с сервисом GitHub;
- phpMyAdmin – веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL, для удобного создания, редактирования базы данных, используемой для хранения информации о пользователях (Аккаунты, результаты, статистика);
- C++ – как изучаемый язык программирования;
- Библиотека SFML – в пользу её простоты и доступности;
- Библиотека libcURL – кроссплатформенная служебная программа командной строки, позволяющая взаимодействовать с множеством различных серверов по множеству различных протоколов с синтаксисом URL, для реализации формирования запросов на сервер, и получения данных от сервера;
- Библиотека JSON for Modern C++ – header-only библиотека для работы с JSON информацией на стороне приложения.
- PHP – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений, для создания серверной части;
- SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных, для, соответственно, создания, модификации и управления данными в базе данных.

2.2 Описание алгоритмов

2.2.1 Алгоритм работы сцен.

В данном проекте присутствуют понятия сцен и перехода между ними. Каждый новый экран (к примеру: главное меню, игра, результаты) это своя сцена, со своей логикой. Активную, в данный момент времени, сцену определяет перечислитель «currentGameScene». Все, что нужно для работы сцен передается им алгоритмом на рисунке 2.1.

```
while (window.isOpen()) {
    std::chrono::steady_clock::time_point begin = std::chrono::steady_clock::now();

    sf::Event event;
    while (window.pollEvent(event)) {
        if (event.type == sf::Event::Closed) window.close();
        if (scenes[currentGameScene]->isStarted)
            scenes[currentGameScene]->EventHandle(&window, &event, &nM, &sm);

        if (event.type == sf::Event::Resized){
            sf::FloatRect visibleArea(0, 0, event.size.width, event.size.height);
            window.setView(sf::View(visibleArea));
        }

    }
    window.clear(sf::Color(50, 50, 50));

    if (!scenes[currentGameScene]->isStarted) {
        scenes[currentGameScene]->Start(&window, &nM);
    }
    else {
        if (scenes[currentGameScene]->switchScene)
            currentGameScene = scenes[currentGameScene]->switchSceneEvent();
        else
            scenes[currentGameScene]->Update(&window, &nM, fElapsedTime);
    }

    if (nM.isLoggedIn()) {
        if (!nM.polling) {
            nM.startPolling();
        }
        nM.drawServerMessages(&window, fElapsedTime);
    }

    window.display();

    std::chrono::steady_clock::time_point end = std::chrono::steady_clock::now();
    fElapsedTime = std::chrono::duration_cast<std::chrono::nanoseconds>(end - begin).count() / 1e+9;
}
```

Рис. 2.1. Алгоритм работы сцен.

Каждый кадр, пока игра не закрыта, алгоритм обрабатывает основные события, такие как закрытие окна, изменения размеров окна, а также передает событие на обработку активной сцене, которая использует это событие для обработки своей логики (например, ввод букв в игровой сцене или нажатие Esc для возврата в главное меню). Далее алгоритм очищает экран для рисовки нового кадра и определяет, запускается ли активная сцена в первый раз, или работает уже не первый кадр. Если сцена исполняется первый раз, то она выполняет функцию «Start», в которой инициализирует свои переменные и подготавливает все для стабильной работы. После выполнения функции «Start» сцена меняет свою переменную «isStarted» на значение «true», и дальше наш алгоритм вызывает функцию «Update» каждый кадр, пока переменная «switchScene» сцены не станет равной «true». Если же наследованная сценой переменная «switchScene» равна «true», то это означает, что сцена меняет активную сцену на другую сцену (совершает переход между сценами), и вызывается соответствующая функция сцены «switchSceneEvent» (которая так же меняет переменную «isStarted» обратно на «false»). Дальше идет небольшой блок кода, который принимает сообщения с сервера, и выводит их (если это нужно) независимо от активной сцены (если вам приходит приглашение на дуэль, то сообщение об этом будет выводиться на экран, независимо от активной на данный момент сцены).

Это все что нужно для правильной работы программы в файле «main.cpp», отсюда последующий рост программного кода определяется только количеством файлов, так как реализация каждой сцены содержится в своем соответствующем файле. Если требуется добавить новую сцену в игру, все что нужно изменить в этом файле – это добавить сцену в массив. Таким образом, проект и процесс его последующей разработки становится менее запутанным.

2.2.2 Алгоритм расчета средней скорости и точности пользователя на основе статистики.

После окончания ввода пользователем десяти слов в игровой сцене (дуэль или одиночная), приложение посылает результаты пользователя на сервер. На сервере эта информация добавляется в глобальную таблицу «results», в которой каждая запись содержит WPM (Words Per Minute – число, обозначающее скорость печати), ACC (точность) и идентификатор пользователя. Для таблицы «results» установлен триггер, который после добавления новой записи обновляет статистику игрока, результаты которого были записаны (Рисунок 2.2).

```
CREATE TRIGGER `updateAverages` AFTER INSERT ON `results` FOR EACH ROW BEGIN

UPDATE users
SET users.avgWPM = (SELECT AVG(wpm) FROM results WHERE userid = NEW.userid),
users.avgACC = (SELECT AVG(acc) FROM results WHERE userid = NEW.userid)
WHERE users.idUsers = NEW.userid;
```

Рис. 2.2. Обновление статистики игрока.

2.3 Описание структур данных

2.3.1 Клиент

Все сцены наследуются от одного класса «GameScene», это позволяет сгруппировать указатели всех сцен в один массив типа «GameScene*», и легко определять какую сцену мы хотим использовать с помощью перечислений с соответствующими названиями (Рисунок 2.3).

```
GameScene* scenes[7];
ProfileScene    s_profile;
MainMenuScene   s_mainmenu;
ResultScene     s_result;
PickPlayerScene s_pickplayer;
DuelResultScene s_duelresult;

DuelPlayScene   s_duelplay;
s_duelplay.setResultScenePointer(&s_duelresult);

PlayScene s_play;
s_play.loadText();
s_play.setResultScenePointer(&s_result);

scenes[MENU]        = &s_mainmenu;
scenes[PLAY]        = &s_play;
scenes[PROFILE]     = &s_profile;
scenes[RESULT]      = &s_result;
scenes[PLAYER_PICK] = &s_pickplayer;
scenes[DUEL_PLAY]   = &s_duelplay;
scenes[DUEL_RESULT] = &s_duelresult;
```

Рис. 2.3. Инициализация игровых сцен.

Для обеспечения программы базой слов в одиночном режиме игры используется обыкновенный текстовый файл с новым словом в каждой строчке. Текстовый файл, используемый в проекте, содержит более 93 тыс. слов. При запуске игры все слова загружаются в один большой вектор и, при запуске одиночной игры, в функции «Start» класса «PlayScene», выбираются 10 случайных элементов из вектора для игры (Рисунок 2.5). Если база слов не загружена в массив, то загружаем. Далее в цикле от 0 до переменной «wordAmt», которая на данный момент может равняться только 10, программой загружается по случайному слову из массива, используя выражение «`rand() % text.size()`» для генерирования номера случайного элемента из массива.

```
if (!loaded) loadText();
for (int i = 0; i < wordAmt; i++) {
    std::wstring line = text[rand() % text.size()];
    charAmt += line.length() + 1; //+1 for enter
    playtext.push_back({ 0.0f, -300.0f, line, 36, false });
}
playtext.push_back({ 0.0f, -300.0f, L"Начать!", 36, false });
```

Рис. 2.4. выбор случайных 10 слов.

2.3.2 Сервер

Для хранения информации об аккаунтах пользователей была создана таблица в базе данных сервера «users». Она содержит базовую информацию: идентификатор (порядковый номер в таблице) пользователя, имя аккаунта, электронная почта пользователя, пароль (хешированный), а так же его средняя скорость печати и точность (Рисунок 2.5).

idUsers	uidUsers	emailUsers	pwdUsers	avgWPM	avgACC
4	skamazzz	human3562@	$2y$10$96gMNLZ5rGg0Xm5RSjfzMupF5mN9YX/saTs1klB.vyw...	56	90
5	SASHA	lagut.alex1@	$2y$10$VKpDZ2KpVVvuL3AgAdtHDeAu3RcK6Q6eSCtZ.IzFXV6...	67	93
9	testacc	human3562@	$2y$10$UCg1Cjki9SDm5qw8tqVWVOGHbYvwyzZq6JEDZSEBkOP...	52	91
11	anothertestacc	test@test.tes	$2y$10$FzuAsJgmkuaoeil6MIGCP.GID4.K3xAxBz0RQBucidI...	0	0
12	thirdtestacc	test@test.tes	$2y$10$0lzJVrk3c/jp0tnIK/MNKugMKi83XxqFfwe5X.NK7XH...	0	0

Рис. 2.5. Таблица «users»

Для хранения информации о том, какие пользователи сейчас в игре, используется таблица «onlineUsers». Данная таблица содержит уникальный API ключ, выдаваемый сервером приложению после успешной авторизации, идентификатор пользователя (соответствующий его идентификатору в таблице «users») а также имя пользователя (Рисунок 2.6).

id	userKey	userID	userName
2	c6b7c01e62b047d5ffa2e3cb227fa7bb	4	skamazzz
3	15a29f9fd9d5123236132454ae405094	5	SASHA
6	ccab0107c57d03b70ea418720426d415	9	testacc
7	f731675b8260020e4cf227bb12f6b70a	13	handsomeandi
8	88b25bbc7995fb794973a93a55f25065	14	Nikikita
9	920c4f156e2d9bec8848796519dde278	15	Lenin

Рис. 2.6. Таблица «onlineUsers»

Для хранения информации о результатах игры пользователей используется таблица «results», в которой содержится идентификатор пользователя, результаты которого содержатся в строке, итоговая скорость, точность и дата записи (Рисунок 2.7).

userid	wpm	acc	created_at
4	57	97	2020-05-26 09:53:56
9	26	91	2020-05-26 09:54:21
4	61	91	2020-05-26 10:05:45
5	67	90	2020-05-26 10:07:11
4	54	86	2020-05-26 10:07:16
5	74	97	2020-05-26 10:23:14
4	55	91	2020-05-26 10:23:21
5	71	92	2020-05-26 10:31:24
4	51	90	2020-05-26 10:31:30
5	65	92	2020-05-26 10:32:44
4	61	89	2020-05-26 10:32:46
5	63	88	2020-05-26 10:34:41
4	56	90	2020-05-26 10:34:45
5	62	88	2020-05-26 10:43:24
4	50	84	2020-05-26 10:43:30
5	66	95	2020-05-26 10:44:28
4	56	88	2020-05-26 10:44:33
5	66	93	2020-05-26 10:45:43
4	57	89	2020-05-26 10:45:47
5	59	90	2020-05-26 10:46:42
9	66	95	2020-05-26 10:46:42
5	69	100	2020-05-26 10:47:37
9	56	91	2020-05-26 10:47:43
5	67	97	2020-05-26 10:49:29
9	58	89	2020-05-26 10:49:30

Рисунок 2.7. Таблица «results»

2.4 Описание основных модулей

2.4.1 Клиент

После запуска игры пользователь сразу попадает в главное меню (Рисунок 2.8). Из главного меню пользователь может перейти в игру после ввода слов «Играть», «в одиночку» (слова дописывать до конца не обязательно, после того как вы начнете вводить вариант, выбранный вами вариант окраситься зеленым цветом, после чего достаточно нажать Enter чтобы перейти). Без авторизации пользователь сможет играть только в одиночную игру, однако его статистика нигде не будет записываться. После авторизации результаты пользователя будут автоматически отправляться на сервер после каждой игры и пользователю будут видны его средние показатели скорости и точности в окне профиля (Рисунок 2.9). Так же после авторизации для пользователя станет доступен режим дуэли с другим пользователем. Пользователь может либо сам предложить другому игроку дуэль, либо получить приглашение на дуэль (Рисунок 2.10). Для того

чтобы пригласить другого пользователя на дуэль достаточно из главного меню вписать слова «Играть» и «онлайн», после чего пользователю будет показан список игроков в сети (Рисунок 2.11). Пользователю достаточно навестись курсором на нужного пользователя, и щелкнуть на него левой кнопкой мыши. Этот выбор происходит помощью мыши потом, что имена пользователей могут начинаться с одинаковых букв. К сожалению, метод, который реализует выбор элементов пользовательского интерфейса, еще не поддерживает подобные тонкости. Точнее, с его помощью можно отдельно выбирать варианты, названия которых начинаются с одинаковых букв, однако этот механизм не понятен интуитивно и может запутать пользователя. Для того чтобы принять приглашение нужно вернутся в главное меню и вписать свой вариант выбора: «принять» или «отклонить».

Рис. 2.8. Внешний вид главного меню

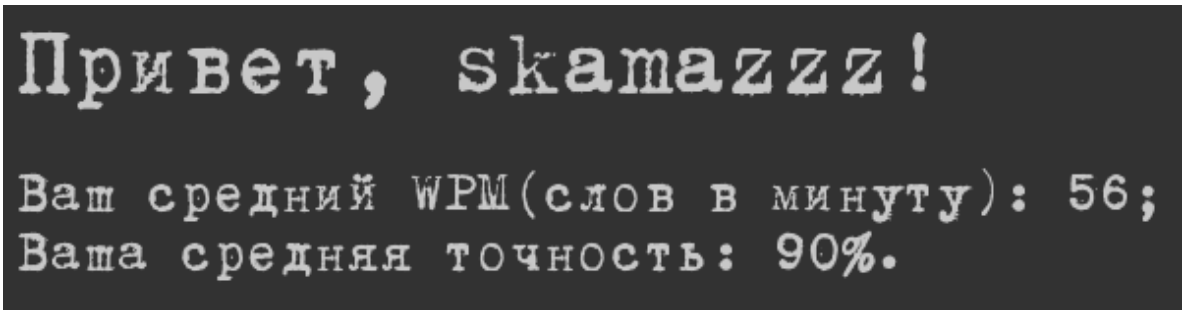

Рис. 2.9. Вывод средних показателей скорости и точности печати пользователя после авторизации.

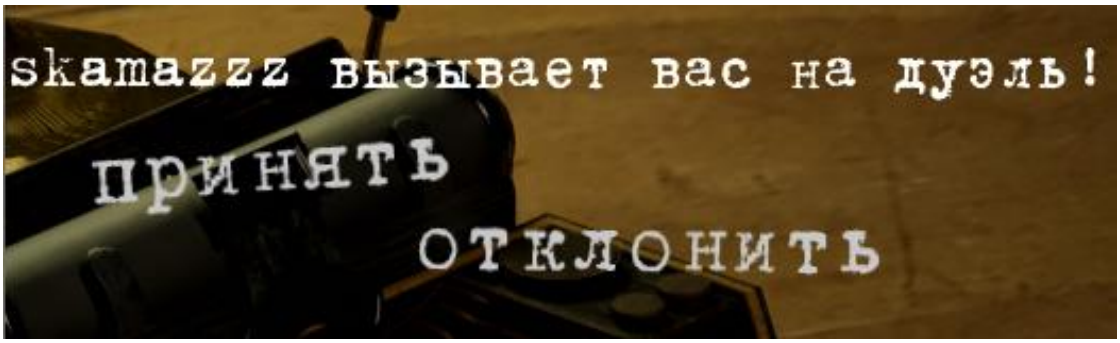

Рис. 2.10. Внешний вид приглашения на дуэль от другого пользователя.

Рис. 2.11. Список пользователей в сети.

2.4.2 Сервер

Сервер исполняет несколько функций. В первую очередь он служит для сохранения игровой статистики пользователя. Чтобы сохранять свою статистику на сервере пользователю потребуется создать аккаунт. Зарегистрировать аккаунт можно на специальной веб-странице (Рисунок 2.12). После регистрации пользователь может авторизоваться в окне профиля в игре.

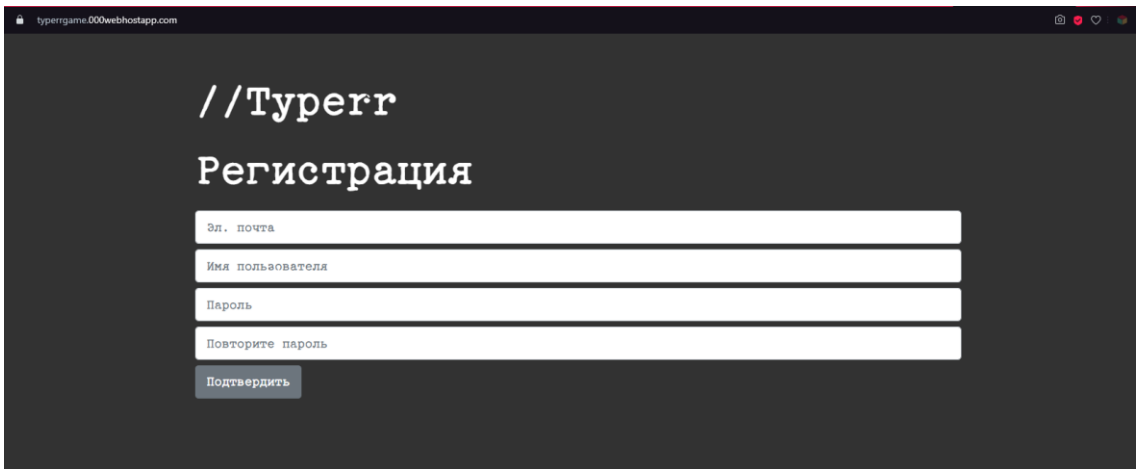

Рис. 2.12. Страница регистрации пользователя

В процессе регистрации используются: подготовленные запросы для защиты от SQL-инъекций, проверка на уже существующего пользователя, хеширование паролей, проверка на наличие пустых полей в переданной форме, проверка на действительность формата электронной почты и фильтрование нежелательных символов (Рисунок 2.13).

```php
if(empty($username) || empty($email) || empty($password) || empty($passwordRepeat)){
    header("Location: ../index.php?error=emptyfields&uid=".$username."&mail=".$email);
    exit();
} else if(!filter_var($email, FILTER_VALIDATE_EMAIL) && !preg_match('/^([a-zA-Z0-9_]+)$/u', $username)) {
    header("Location: ../index.php?error=invalidmailuid");
    exit();
} else if(!filter_var($email, FILTER_VALIDATE_EMAIL)){
    header("Location: ../index.php?error=invalidmail&uid=".$username);
    exit();
} else if(!preg_match('/^([a-яА-ЯЁёa-zA-Z0-9_]+)$/u', $username)){
    header("Location: ../index.php?error=invaliduid&mail=".$email);
    exit();
} else if($password !== $passwordRepeat){
    header("Location: ../index.php?error=passwordcheck&uid=".$username."&mail=".$email);
    exit();
} else {
    $db = new DBConnect();
    $conn = $db->getDB();
    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt,"SELECT uidUsers FROM users WHERE uidUsers=?")){
        header("Location: ../index.php?error=sqlerror");
        exit();
    } else {
        mysqli_stmt_bind_param($stmt, "s", $username);
        mysqli_stmt_execute($stmt);
        mysqli_stmt_store_result($stmt);
        $resultCheck = mysqli_stmt_num_rows($stmt);
        if($resultCheck > 0){
            header("Location: ../index.php?error=usertaken");
            exit();
        } else{
            $sql = "INSERT INTO users (uidUsers, emailUsers, pwdUsers) VALUES (?, ?, ?)";
            $stmt = mysqli_stmt_init($conn);
            if(!mysqli_stmt_prepare($stmt, $sql)) {
                header("Location: ../index.php?error=sqlerror");
                exit();
            } else{
                $hashedPwd = password_hash($password, PASSWORD_DEFAULT);
                mysqli_stmt_bind_param($stmt, "sss", $username,  $email, $hashedPwd);
                mysqli_stmt_execute($stmt);
                require 'success.html';
            }
        }
    }
}
mysqli_stmt_close($stmt);
```

Рис. 2.13. Процесс регистрации пользователя.

В процессе авторизации сервер выдает клиенту уникальный ключ, добавляет в файл пользователя новое уведомление об успешной авторизации и добавляет его в таблицу «onlineUsers». Процесс авторизации так же защищен от SQL-инъекций (Рисунок 2.14).

```
$sql = "SELECT * FROM users WHERE emailUsers = ? OR uidUsers = ?";
$db = new DBConnect();
$stmt = mysqli_stmt_init($db->getDB());
if(!mysqli_stmt_prepare($stmt,$sql)){
    echo 'error 3';
}else{
    mysqli_stmt_bind_param($stmt, "ss", $mailuid, $mailuid);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);
    if($row = mysqli_fetch_assoc($result)){
        $pwdCheck = password_verify($password, $row['pwdUsers']);
        if($pwdCheck == false){
            echo 'error 0 pwd';
        }else{
            $id = $row['idUsers'];
            $key = md5(microtime().rand());

            $userFile = fopen("userFiles/testfile".$id.".txt", "w");
            fwrite($userFile, '{"sendType":"notification", "content":"login_success"}');
            fclose($userFile);

            $checkIfUserExists = mysqli_query($db->getDB(), "SELECT * FROM onlineUsers WHERE userID = ".$id);
            if(mysqli_num_rows($checkIfUserExists)>0){
                mysqli_query($db->getDB(), 'UPDATE `onlineUsers` SET `userKey`="'.$key.'" WHERE `userID` = '.$id.';');
            }else{
                echo 'nop';
                mysqli_query($db->getDB(), "INSERT INTO `onlineUsers` (`id`, `userKey`, `userID`, `userName`)
                VALUES (NULL, '".$key."', '".$id."', '".$row['uidUsers']."');"))
            }
            echo('{ "username":"'.$row['uidUsers'].'", "userId":"'.$row['idUsers'].'",
               "userWPM":"'.$row['avgWPM'].'", "userACC":"'.$row['avgACC'].'", "uniqueKey":"'.$key.'"}');
        }
    }else{
        echo 'error 0';
    }
}
```

Рис. 2.14. Процесс авторизации пользователя.

ГЛАВА 3
ТЕСТИРОВАНИЕ ПРОГРАММЫ

3.1 Тестирование исходного кода

Проект делится на два модуля: сервер и приложение. Тестами было покрыто приложение, и страница регистрации (что по сути является частью сервера). Всего тестов было четыре. Первый тест был направлен на работоспособность самого приложения на системе тестировщика (запускается ли оно) (Рисунок 3.1). Второй тест был направлен на работоспособность элементов пользовательского интерфейса в главном меню (Рисунок 3.2). Третий тест направлен на работоспособность одиночной игры, на вводимость игровых слов, перехода в сцену результатов и отображение верных данных в сцене результатов (Рисунок 3.3). Последний тест был направлен на проверку работоспособности страницы регистрации, работоспособности авторизации в приложении и на изменение статистики пользователя после нескольких одиночных игр (Рисунок 3.4).

Тест 1/Test Name 1

Среда тестирования /Environment	ОС – Windows 10 или любая совместимая
Предварительные действия /Pre Requisites	Скачать beta-версию курсовой работы с ресурса по ссылке, указанной в «Общей информации»
Комментарии /Comments	С элементами пользовательского интерфейса пока что невозможно воздействовать мышкой, нужно вводить желаемый выбор элемента.

Шаг /Step No.	**Действие (операция) /Process (Actions)**	**Ожидаемый результат /Expected Results**	**Результат /Actual Results**	**Пройден /не пройден/ не доступен***	**Комментарии /Notes**
Запуск приложения					
1	Разархивировать скачанный архив в произвольное место	В выбранном месте появляется папка с исполняемым файлом.	Появляется папка с файлом.	Пройден.	
2	Запустить исполняемый файл с названием attempt1.exe	Открытие приложения, появление главного меню.	Появление главного меню.	Пройден.	

Рис. 3.1. Первый тест

Тест 2/Test Name 2

Среда тестирования /Environment	ОС – Windows 10 или любая совместимая
Предварительные действия /Pre Requisites	Скачать beta-версию курсовой работы с ресурса по ссылке, указанной в «Общей информации»
Комментарии /Comments	С элементами пользовательского интерфейса пока что невозможно воздействовать мышкой, нужно вводить желаемый выбор элемента.

Шаг /Step No.	Действие (операция) /Process (Actions)	Ожидаемый результат /Expected Results	Результат /Actual Results	Пройден /не пройден/ не доступен*	Комментарии /Notes
Проверка работоспособности главного меню					
1	Проверить все кнопки на работоспособность	Каждый вариант выбора в главном меню должен успешно выполнять переход между сценами.	Переход между сценами успешно осуществлён.	Пройден.	
2	Проверить, как программа справляется с изменениями разрешения экрана.	Каждый элемент пользовательского интерфейса должен пересчитывать свое положение должным образом.	Все элементы пользовательского режима работают должным образом.	Пройден.	

Рис. 3.2. Второй тест

Тест 3/Test Name 3

Среда тестирования /Environment	ОС – Windows 10 или любая совместимая
Предварительные действия /Pre Requisites	Скачать beta-версию курсовой работы с ресурса по ссылке, указанной в «Общей информации»
Комментарии /Comments	С элементами пользовательского интерфейса пока что невозможно воздействовать мышкой, нужно вводить желаемый выбор элемента.

Шаг /Step No.	Действие (операция) /Process (Actions)	Ожидаемый результат /Expected Results	Результат /Actual Results	Пройден /не пройден/ не доступен*	Комментарии /Notes
Проверка работоспособности одиночной игры					
1	Запустите программу, впишите «Играть», «в одиночку».	Должна запуститься сцена игры с 10 словами для ввода.	Сцена с 10 словами.	Пройден.	
2	Введите «Начать!» и начните вписывать слова.	Все слова должны вводиться, счетчики работать и таймер считать время должным образом.	Слова выведены, счётчики работают, таймер работает.	Пройден.	
3	Допишите все 10 слов для перехода в сцену результатов.	Сцена результатов должна показывать время, за которое вы ввели все слова, набранные вами очки, ваша скорость в словах в минуту и точность.	Вывод времени, очков, скорости и точности.	Пройден.	
4	Проверьте работоспособность кнопок «Ещё раз!» и «Назад»	Кнопка «Ещё раз!» должна перезапустить сцену игры, «Назад» должна вернуть вас в главное меню.	Перезапуск сцены. Возвращение в главное меню.	Пройден.	

Рис. 3.3. Третий тест

Тест 4/Test Name 4

Среда тестирования /Environment	ОС – Windows 10 или любая совместимая
Предварительные действия /Pre Requisites	Скачать beta-версию курсовой работы с ресурса по ссылке, указанной в «Общей информации»
Комментарии /Comments	С элементами пользовательского интерфейса пока что невозможно воздействовать мышкой, нужно вводить желаемый выбор элемента.

Шаг /Step No.	Действие (операция) /Process (Actions)	Ожидаемый результат /Expected Results	Результат /Actual Results	Пройден /не пройден/ не доступен*	Комментарии /Notes
Проверка работоспособности входа в свой аккаунт					
1	Перейдите по ссылке https://typerrgame.000webhostapp.com/ (рабочая форма регистрации, внешний вид станет лучше)	Ваш аккаунт должен быть зарегистрирован.	Аккаунт зарегистрирован.	Пройден.	
2	Запустите игру, выберите вариант «Профиль» и введите свои данные для входа.	Вы будете перенаправлены в главное меню, после чего справа сверху должно появиться имя вашего аккаунта и ваш WPM (слова в минуту)	Перенаправление в главное меню, появление имени и WPM.	Пройден.	
3	Сыграйте пару игр и убедитесь, что ваша статистика обновляется.	После каждой игры вы должны наблюдать изменения в вашей статистике, которую можно найти в сцене «Профиль» (Пока что есть только WPM и точность)	Статистика меняется в сцене «Профиль».	Пройден.	

Рис. 3.4 Четвертый тест.

3.2 Тестирование интерфейса пользователя и юзабилити

Пользовательский интерфейс довольно устойчив, однако, наличие возможности выбора элементов с помощью мыши было бы удобно. Поля ввода и кнопка «Подтвердить» в сцене профиля, могла бы быть улучшена в некоторых аспектах, например: возможность удалять текст из полей, зажав кнопку Backspace, возможность переключаться между элементами интерфейса с помощью кнопки TAB и смещение текста в поле ввода, если текст выходит за границы поля ввода. В остальном интерфейс интуитивен и прост к пониманию.

ГЛАВА 4
ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА

4.1 Перспективы технического развития

Из идей, которых не удалось реализовать в процессе разработки, можно отметить отсутствие каких либо индикаторов прогресса игры противника в режиме дуэли – узнать выиграл пользователь или проиграл можно только после окончания дуэли. Так же не удалось создать сцену общих настроек игры по причине нехватки времени. Еще одной из значимых недостатков – это малое количество многопоточной оптимизаций. На данный момент мульти поточность используется только для реализации получения информации с сервера методом «long-poll», потому что в противном случае, программа бы зависала до тех пор, пока пользователю не пришло какое-либо уведомление.

Для посыла запроса на сервер и получение ответа требуется некоторое количество времени, так как сервер проекта размещен на бесплатном хостинге, эти недостатки четко видны в процессе эксплуатации приложения. Например, после завершения одиночной игры приложение посылает результаты пользователя на сервер, и в промежуток этого времени, все приложение зависает до тех пор, пока приложение не получит ответ от сервера об успехе или ошибке. Эта проблема может быть решена с помощью понятия списка задач приложения, которые бы выполнялись на другом потоке. Например, если приложению требуется отправить результаты игрока на сервер, то эта задача добавляется в очередь процессов, которые выполняются на другом потоке параллельно с основным приложением. Таким образом, приложение сможет обмениваться информацией с сервером, не замедляя элементов, с которыми взаимодействует пользователь.

Кроме этого, в проекте сделано много из того, что планировалось сделать, однако есть некоторые идеи, которые хотелось бы реализовать или изменить. Во-первых, это возможность конфигурации настроек игры, таких как количество слов, выдаваемых игроку на ввод, различные модификаторы для усложнения или

упрощения игры и т.п. Из желаемых добавлений так же можно было бы добавить режим ввода не одиночных слов, как это реализовано сейчас, а режима ввода целых предложений или абзацев, например из книг или статей.

4.2 Перспективы монетизации

На данном этапе никаких способов монетизации не реализовано, однако идеи реализации присутствуют.

Одной из перспектив является продажа на онлайн-сервисах цифрового распространения компьютерных игр и программ, таких как «Steam», «Epic Games Store», «GOG». Выставление проекта на продажу на популярной площадке поможет ему получить хотя бы минимум внимания, и шанс заинтересовать человека, просматривающего списки игр.

После успешного добавления режима предложений в игру, перспективной идеей может стать возможность покупки тематических наборов текстов, например цитат, абзацев из литературы, стихов или даже блоков кода, для начинающих или желающих улучшить свои навыки программистов.

ЗАКЛЮЧЕНИЕ

Во все области деятельности человека стремительно входят компьютерные технологии, поэтому, в наше время, большим фактором в определении эффективности работника, который взаимодействует с компьютером в ходе своего рабочего дня, является его скорость и эффективность печати. Так же, высокая скорость печати это экономия времени, даже вне рабочей обстановки. К примеру, с высокой скоростью печати, набор поздравления для родственника или знакомого человека на его день рождения становиться намного менее затянутым по времени действием. Итогом выполненной курсовой работы является ранняя версия продукта, который может помочь пользователю улучшить свои навыки печати, делая его более эффективным и полезным работником.

Был получен опыт в разработке крупных проектов, работе с мульти-файловыми проектами в среде разработки Microsoft Visual Studio 2019, распределении времени и выставления приоритетов в процессе разработки. Так же был получен опыт в поиске и выборе библиотек, позволяющих реализовать функционал, нужный приложению. Был получен опыт работы со сторонними библиотеками, а именно SFML, libcURL, JSON for Modern C++. Освоен принцип реализации работы клиент-серверных структур.

В процессе разработки освоил принципы создания PHP серверов, научился создавать процессы регистрации и авторизации, защищать их от SQL-инъекций. Научился формировать данные в виде JSON формата, с помощью инструментария PHP, а так же принимать и десериализовывать JSON данные на стороне приложения, используя сторонние библиотеки C++.

Продукт, будучи на ранних этапах, все еще нуждается в полировке, в некоторых местах все еще можно найти грубоватые заготовки или не полностью продуманные функции, однако весь функционал в программе рабочий и отвечает целям, поставленным в начале разработки.

ЛИТЕРАТУРА

1. ГОСТ 19.002-80 Схемы алгоритмов и программ. Правила выполнения [Текст] – Введ. с 01.07. 1981 г. М.: Изд-во стандартов, 1981. – 9 с.

2. ГОСТ 19.003-80 Схемы алгоритмов и программ. Обозначение условные графические [Текст] – Введ. с 01.07. 1981 г. М.: Изд-во стандартов, 1981. – 9 с.

3. Оформление выпускной квалификационной работы на соискание квалификационного уровня «Магистр» («Бакалавр»): методические рекомендации. / сост. Бержанский В.Н., Дзедолик И.В., Полулях С.Н. – Симферополь: КФУ им. В.И.Вернадского, 2017. – 31 с.

4. Руководство конфигурации проекта для работы с библиотекой SFML в среде Microsoft Visual Studio «SFML and Visual Studio» [Электронный ресурс] URL: https://www.sfml-dev.org/tutorials/2.5/start-vc.php

5. Простая реализация long polling механизма на PHP [Электронный ресурс] URL: https://habr.com/ru/post/128535/

ПРИЛОЖЕНИЕ 1
КОД ОСНОВНЫХ МОДУЛЕЙ ПРОЕКТА

Файл «main.cpp»

```
int main() {
	srand(unsigned(time(0)));

	sf::Image icon;
	icon.loadFromFile("resources/textures/typewritericon.png");

	float fElapsedTime = 0.001f;
	e_gameState currentGameScene = MENU;

	sf::ContextSettings settings;
	settings.antialiasingLevel = 4;

	sf::RenderWindow window(sf::VideoMode(1024, 768), "jeesus", sf::Style::Default,
settings);

	window.setIcon(icon.getSize().x, icon.getSize().y, icon.getPixelsPtr());
	//window.setFramerateLimit(300);

	window.setKeyRepeatEnabled(false);

	NetworkManager nM;
	SoundMaster sm;

	sf::Text mainText;
	sf::Font font;

	if (!font.loadFromFile("resources/font2.ttf")) {
		//bad
	}

	mainText.setFont(font);
	mainText.setStyle(sf::Text::Regular);

	nM.mainText = mainText;

	GameScene* scenes[7];
	ProfileScene    s_profile;
	MainMenuScene   s_mainmenu;
	ResultScene     s_result;
	PickPlayerScene s_pickplayer;
	DuelResultScene s_duelresult;

	DuelPlayScene   s_duelplay;
	s_duelplay.setResultScenePointer(&s_duelresult);

	PlayScene s_play;
	s_play.loadText();
	s_play.setResultScenePointer(&s_result);

	scenes[MENU]        = &s_mainmenu;
	scenes[PLAY]        = &s_play;
	scenes[PROFILE]     = &s_profile;
	scenes[RESULT]      = &s_result;
	scenes[PLAYER_PICK] = &s_pickplayer;
	scenes[DUEL_PLAY]   = &s_duelplay;
	scenes[DUEL_RESULT] = &s_duelresult;

	while (window.isOpen()) {
		std::chrono::steady_clock::time_point begin =
std::chrono::steady_clock::now();
```

```
			sf::Event event;
			while (window.pollEvent(event)) {
				if (event.type == sf::Event::Closed) {
					window.close();
				}
				if (scenes[currentGameScene]->isStarted)
					scenes[currentGameScene]->EventHandle(&window, &event, &nM,
&sm);

				if (event.type == sf::Event::Resized){
					sf::FloatRect visibleArea(0, 0, event.size.width,
event.size.height);
					window.setView(sf::View(visibleArea));
				}

			}
			window.clear(sf::Color(50, 50, 50));

			if (!scenes[currentGameScene]->isStarted) {
				scenes[currentGameScene]->Start(&window, &nM);
			}
			else {
				if (scenes[currentGameScene]->switchScene)
					currentGameScene = scenes[currentGameScene]->switchSceneEvent();
				else
					scenes[currentGameScene]->Update(&window, &nM, fElapsedTime);
			}

			if (nM.isLoggedIn()) {
				if (!nM.polling) {
					nM.startPolling();
				}
				nM.drawServerMessages(&window, fElapsedTime);
			}

			window.display();

			std::chrono::steady_clock::time_point end = std::chrono::steady_clock::now();
			fElapsedTime = std::chrono::duration_cast<std::chrono::nanoseconds>(end -
begin).count() / 1e+9;
		}

		nM.stop();

		return 0;
		}
```

Рисунок П.1 Файл «main.cpp»

Файл «mainMenuScene.h»

```
#pragma once
#include "gameScene.h"
#include "inputText.h"

class MainMenuScene : public GameScene
{
public:
		MainMenuScene() {}

public:
	void Start(sf::RenderWindow* window, NetworkManager* nM) override;
		void Update(sf::RenderWindow* window, NetworkManager* nM, float fElapsedTime)
override;
		void EventHandle(sf::RenderWindow* window, sf::Event* event, NetworkManager* nM,
SoundMaster* sm) override;
		e_gameState switchSceneEvent() override;
		float lerp(float a, float b, float f);
```

```
	float SmoothApproach(float pastPosition, float pastTargetPosition, float
targetPosition, float speed, float deltaTime); //alternative for lerp

//private:
//	std::string login(std::string uid, std::string pwd);

private:
	InputText startMenu[8] = {
		{0.0f, -200.0f,L"Профиль\n", 50.0f, -0.7f },
		{0.0f, 240.0f, L"Играть\n", 70.0f, -7.0f},
		{0.0f, -200.0f,L"в одиночку\n", 50.0f, -0.6f },
		{0.0f, -200.0f,L"онлайн\n", 50.0f, -0.6f },
		{0.0f, 350.0f, L"Настройки\n", 50.0f, -6.0f},
		{0.0f, 440.0f, L"выход\n", 45.0f, -4.0f},
		{0.0f, 380.0f, L"принять\n", 45.0f, -4.0f},
		{0.0f, 410.0f, L"отклонить\n", 45.0f, 0.6f}
	};
	sf::Text mainText = {};
	sf::Text secondaryText;
	sf::Font font;
	sf::Font font2;
	std::wstring test = L"";
	bool startSelected = false;
	bool profileSelected = false;
	bool optionsSelected = false;
	bool onlineSelected = false;
	bool duelintent = false;
	float uiPositions[8] = {};

	bool loaded = false;

	sf::Texture typewriter;
	sf::Texture paper;

};
```

Рисунок П.2 Файл «mainMenuScene.h»

Файл «playScene.h»

```
#pragma once
#include "gameScene.h"
#include "inputText.h"
#include "charParticle.h"
#include "resultScene.h"
class PlayScene : public GameScene
{
public:
	PlayScene() {}

public:
	void Start(sf::RenderWindow* window, NetworkManager* nM) override;
	void Update(sf::RenderWindow* window, NetworkManager* nM, float fElapsedTime)
override;
	void EventHandle(sf::RenderWindow* window, sf::Event* event, NetworkManager* nM,
SoundMaster* sm) override;
	e_gameState switchSceneEvent() override;
	void loadText();
	void setResultScenePointer(ResultScene* ptr);

private:
	float lerp(float a, float b, float f);
	float SmoothApproach(float pastPosition, float pastTargetPosition, float
targetPosition, float speed, float deltaTime);
	void pop(InputText* t);
	std::string nf(float time);


private:
```

```
	sf::Text mainText;
	sf::Font font;
	std::vector<std::wstring> text;
	std::vector<InputText> playtext;
	std::vector<CharParticle> part;
	bool errorWhenLoading = false;
	bool back = false;
	bool loaded = false;
	bool finished = false;
	float fScreenRotation = 0.0f;

	int wordAmt = 10;
	int charAmt = 0;
	float time = 0.0f;
	int multiplier = 1;
	int score = 0;
	int mistakes = 0;
	int keyPresses = 0;

	ResultScene* ResultScenePointer = nullptr;
	/*sf::SoundBuffer keySounds[4];
	sf::Sound sound;*/
};
```

Рисунок П.3 Файл «playScene.h»

Файл «gameScene.h»

```
#pragma once
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>
#include "gameState.h"
#include "soundMaster.h"
#include "networkManager.h"

class GameScene{
public:
	GameScene() {}

public:
	virtual void Start(sf::RenderWindow* window, NetworkManager* nM) {}
	virtual void Update(sf::RenderWindow* window, NetworkManager* nM, float
fElapsedTime) {}
	virtual void EventHandle(sf::RenderWindow* window, sf::Event* event,
NetworkManager* nM, SoundMaster* sm) {}
	virtual e_gameState switchSceneEvent() { return MENU; }

public:
	bool isStarted = false;
	bool switchScene = false;
};
```

Рисунок П.4 Файл «gameScene.h»

Файл «resultScene.h»

```
#pragma once
#include "gameScene.h"
#include "inputText.h"
class ResultScene : public GameScene
{
public:
	ResultScene() {}
	~ResultScene() {}

public:
	void Start(sf::RenderWindow* window, NetworkManager* nM) override;
```

```
	void Update(sf::RenderWindow* window, NetworkManager* nM, float fElapsedTime)
override;
	void EventHandle(sf::RenderWindow* window, sf::Event* event, NetworkManager* nM,
SoundMaster* sm) override;
	e_gameState switchSceneEvent() override;

	void setupInfo(float time, int charAmt, float accuracy, int score);

private:
	std::wstring nf(float time); //formatting a float storing time in seconds into a
Minute:Second:Millisecond format string

	//stats
	int WPM = 0;
	int accuracy = 0;
	float time = 0.0f;
	int totalScore = 0;

	//scene switch flags
	bool back = false;
	bool restart = false;

	sf::Text optText;
	sf::Text mainText;
	sf::Font font;
	sf::Font font2;

	InputText options[2] = {
		{0.0f, 240.0f, L"Ещё раз!\n", 70.0f, -7.0f},
		{0.0f, 350.0f, L"Назад\n", 50.0f, -6.0f},
	};
};
```

Рисунок П.5 Файл «resultScene.h»