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

игра-кликер “Pull the rope!”

Курсовая работа
по дисциплине «Введение в специальность»
студента 1 курса группы ПИ-б-о-201(2)
Абибулаева Эмине Ридван къызы

направления подготовки 09.03.04 «Программная инженерия»
(код и наименование)

старший преподаватель кафедры компьютерной инженерии и моделирования _____ (оценка) Чабанов В. В.

_____ (подпись, дата)

Симферополь, 2021

Реферат

Абибулаева Э. Р. Разработка игры-кликера «Pull the rope!» // Курсовая работа (уровень бакалавриат) по специальности 09.03.04 Программная инженерия / Кафедра компьютерной инженерии и моделирования Физико-технического института Крымского федерального университета им. В. И. Вернадского. – Симферополь, 2021. - 25с., 22ил., 6ист.

В современном мире игровая индустрия играет ключевую роль. Она связана с разработкой, продвижением и продажей игр, в неё входит большое количество специальностей. Особой популярностью пользуются игры с клиент-серверной архитектурой. Именно по этой причине темой курсовой работы стало создание онлайн-кликера.

В процессе разработки планируется получить навыки, которые позволят реализовать игру.

ПРОГРАММИРОВАНИЕ, ИГРА, КЛИКЕР, КЛИЕНТ-СЕРВЕР, АВТОРИЗАЦИЯ, МНОГОПОТОЧНОСТЬ, БАЗА ДАННЫХ

Оглавление

Список сокращений и условных обозначений

JSON	JavaScript Object Notation

Введение

Игровая индустрия является одной из наиболее крупных сегментов индустрии развлечений. По скорости роста она опережает остальные индустрии. По этой причине тема курсовой работы и была нацелена на это направление, учитывая ещё и популярность игр на Android.

Idle-игры — новая разновидность игр, которая очень быстро набирает популярность на рынке мобильных игр. Но и сам по себе это весьма непривычный жанр.

Что так сильно привлекает игроков в этих играх? Причин множество, но две самые важные заключаются в том, как инкрементные игры используют уникальные грани человеческой психологии.
Первая, часто упоминаемая в обсуждении инкрементных игр, известна под названием "ящик_Скиннера". Эти экспериментальные камеры, названные (к его разочарованию) в честь бихевиориста Б.Ф. Скиннера, были созданы для изучения поведенческого обусловливания животных. В «камере оперантного обусловливания» обычно содержится животное, которое может получить награду (например, пищу) как реакцию на выполнение действия (например, нажатия на кнопку). Примечательно, что после освоения механизма реагирования наблюдаемые животные продолжали выполнять действия, даже если они давали награду только после длительных интервалов или случайным образом.
По аналогии, системы, периодически награждающие пользователей или игроков за повторяющиеся действия часто иронически называют «ящиками Скиннера», потому что создаваемый ими неврологический цикл обратной связи может быть невероятно аддиктивным. Эта структура наиболее очевидна в инкрементных играх: игрок выполняет действие (нажимает или ждёт), и периодически награждается за свои усилия растущими числами. Само по себе это не всегда плохо, и эта механика используется часто. Многим играм приходится обучать игроков выполнению определённых действий в системе

игры и использовать положительные награды (очки, опыт) и отрицательные результаты (смерть), чтобы показать игроку правильный способ прохождения. В инкрементных играх использование этой механики просто гораздо очевиднее.

Вторая психологическая опора инкрементных игр — это наша *страсть к* накоплению и боязнь потерь. Наш мозг устроен так, что не любит терять, и наоборот, он даёт нам сильное желание накапливать.
В инкрементных играх используются обе эти стороны.

Целью курсовой работы является реализация онлайн-кликера. Что значит, что в функционал игры будет входить регистрация и авторизация пользователей, а также и синхронизация игровых действий между клиентами. Проект будет разрабатываться на двух языках программирования C++ и Python.

Для достижения целей были выдвинуты следующие задачи:

1. Определение методов и инструментов разработки по удобству и эффективности.
2. Определение форматов хранения данных и их архитектуру.
3. Реализация логики регистрации и авторизации с рассмотрением всех возможных действий пользователя.
4. Разработка внутриигрового магазина для более интересного развития хода игры.
5. Разработка кода, который отвечает за взаимодействие двух пользователей между собой в ходе игры.
6. Разработка и реализация логики игрового процесса.
7. Сборка под Android.

Также целью данной курсовой работы является изучение методик разработки архитектуры клиент-серверных приложений и применение их на практике.

В работе используются такие методы исследования как: тестирование, программирование, синтез и анализ.

Глава 1

Постановка задачи

1.1 Цель проекта

Создание рабочего проекта, получение опыта настройки сборки под Android. Получение опыта работы с изображениями, изучение сторонних библиотек, подключение и использование их, способов взаимодействия клиентов и сервера. Закрепление знаний и навыков.

1.2 Существующие аналоги

Существует много различных онлайн-кликеров, но именно для такого типа приложения аналогов нет.

1.3 Техническое задание

Программа должна соответствовать следующему техническому заданию:

1. Сервер должен корректно принимать, обрабатывать поступающие запросы и отправлять ответы на клиент.
2. Сервер должен поддерживать работу с базой данных SQLite.
3. Сервер должен содержать несколько таблиц данных:
   - Таблица данных Shows для хранения данных о пользователях, которые зарегистрировались
   - Таблица данных Shop для хранения приобретённых товаров во внутриигровом магазине
4. Клиент должен содержать окна:
   - Окно авторизации
   - Окно меню
   - Окно внутриигрового магазина
   - Окно с самим процессом игры
5. Программа должна функционировать под операционной системой для смартфонов Android.
6. Клиент должен отправлять корректные запросы и принимать ответы сервера.

Глава 2

Программная реализация приложения

2.1 Анализ инструментальных средств

- Среда разработки Qt Creator 4.14.1 (Community) была выбрана в связи со знакомым интерфейсом и возможностями по разработке программ на языке C++
- Среда разработки Visual Studio Code была выбрана из-за удобства разработки и её приспособленности разработки на языке Python
- C++ и Python – как изучаемые языки программирования
- Для соединения сервера и клиента были использованы библиотеки socket на Python и QTcpSocket на C++. Были выбраны из-за отсутствия потери времени на установку соединения, потому что это не сессионный канал связи, а постоянный.
- Библиотека SQLite была выбрана из-за высокой производительности и простоты использования.
- SQL - информационно-логический язык, предназначенный для описания, изменения и извлечения данных, хранимых в реляционных базах данных.

2.2 Описание алгоритмов

2.2.1 Алгоритм работы игры

При запуске игры игроки подсоединяются друг к другу и начинается сам процесс игры (рисунок 2.1), который заключён в том, чтобы набрать больше очков, чем соперник. За победу даётся вознаграждение.

```
void CountWidget::on_pushButton_clicked()
{
    self_counter++;
    ui->label->setText( QString( "%1" ).arg(self_counter));
    sender->sendTo(otherID, R"({"msg":"pushed"})");
    k = self_counter - other_counter;
    if(self_counter < 0)
        self_counter = 0;
    if(other_counter < 0)
        other_counter = 0;
    ui->label_3->setText(QString("%1").arg(k));
    ui->label_4->move(ui->label_4->pos().x(),ui->label_4->pos().y()+1);
    if(k > 10)
    {
        QMessageBox msgBox;
        msgBox.setText("Вы Победили. Выйти в главное меню?");
        ui->label->clear();
        ui->label_3->clear();
        ui->label_2->clear();
        other_counter = 0;
        self_counter = 0;
        coin = coin + 150;
        msgBox.setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);
        int res = msgBox.exec();
        if ( res == QMessageBox::Ok)
        {
            menu->show();
            this->close();
            emit win();
        }
        else
        exit(0);
    }
    if(k < -10)
    {
        QMessageBox msgBox;
        msgBox.setText("Вы Проиграли. Выйти в главное меню?");
        coin = coin - 100;
        ui->label->clear();
        ui->label_3->clear();
        ui->label_2->clear();
        other_counter = 0;
        self_counter = 0;
        if (coin < 0)
            coin = 0;
        msgBox.setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);
        int res = msgBox.exec();
        if ( res == QMessageBox::Ok)
        {
            menu->show();
            this->close();
            emit win();
        }
        else
        exit(0);
    }
}
```

Рисунок 2.1. Алгоритм игрового процесса

2.2.2 Алгоритм работы сервера

При регистрации/авторизации на сервер приходит запрос в формате json. В нём содержится информация о действиях пользователя: авторизация это или регистрация; а также содержатся пароль и логин. Если это регистрация, то для начала сервер подключается к базе данных SQLlite и проверяет есть ли такой пользователь. Нет – логин, пароль добавляются в таблицу, Да – сервер отправляет на клиент сообщение об ошибке. Если это авторизация, то сервер подключается к базе данных и проверяет корректность введённых данных на клиенте. В случае несовпадения сервер отправляет на клиент сообщение об ошибке (рисунок 2.2).

```
def listenClients(connection, clientAddress):
    while 1:
        try:
            data = connection.recv(dataPackageSize)
        except ConnectionError:
            break
        if not data:
            break
        print("Принято сообщение: ", data.decode('utf-8'))
        data_bytes = data
        data = json.loads(data_bytes.decode('utf-8'))
        # sendall - принимает массив байт
        if(data["action"] == "register"):
            # connection.sendall(data_bytes)
            cursor.execute("SELECT COUNT(ID) FROM shows WHERE name = ?", (data["name"],))
            count = cursor.fetchall()
            print(count)
            if(count == [(0, )]):
                cursor.execute("INSERT INTO shows (name, password) VALUES (?, ?)", (data["name"], data["password"]))
                conn.commit()
            else:
                print("This user already exist")
                connection.sendall("no".encode())
        elif (data["action"] == "login"):
            cursor.execute("SELECT ID FROM shows WHERE name = ? AND password = ?", (data["name"], data["password"]))
            users = cursor.fetchall()
            print(users)
            if(users != []):
                data1 = ", ".join([str(i) for i in users])
                connection.sendall(data1.encode())
            else:
                print("This user does not exist")
                connection.sendall("login_fail".encode())

    print(f"{clientAddress} has disconnect")
```

Рисунок 2.2. Проверка данных, запись в таблицу и ответ сервера

На сервере также хранится информация о покупках (рисунок 2.3). В таблице хранятся идентификатор, количество монет на момент покупки и количество объектов.

```
import socket as Socket
import json
import sqlite3

conn = sqlite3.connect('shows.db')
cursor = conn.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS Shop
                  (id text, coin TEXT, obj TEXT,  FOREIGN KEY (id) REFERENCES Shows (ID))''')
address = "localhost"
port = 3300
dataPackageSize = 1024


def listenClients(connection, clientAddress):
    while 1:
        try:
            data = connection.recv(dataPackageSize)
        except ConnectionError:
            break
        if not data:
            break
        print("Принято сообщение: ", data.decode('utf-8'))
        data_bytes = data
        data = json.loads(data_bytes.decode('utf-8'))
        # cursor.execute("SELECT COUNT(ID) FROM Shop WHERE id = ? AND obj = '1'", (data["id"],))
        # count = cursor.fetchall()
        # print(count)
        # f(count == [(0, )]):
        cursor.execute("INSERT INTO Shop (id, coin, obj) VALUES (?, ?, ?)", (data["id"], data["coin"], data["obj"]))
        conn.commit()
        # else:
            # print("This user already exist")
            # connection.sendall("no".encode())

    print(f"{clientAddress} has disconnect")
```

Рисунок 2.3. Создание таблицы данных, проверка данных, запись в таблицу, ответ сервера

Также сервер выполняет подключение клиентов друг к другу с помощью библиотеки Flask (рисунок 2.4).

```
messages = {'any': []}


@app.route('/post', methods=['post'])
def post():
    global messages
    req = request.get_json()
    print('Msg from:', req['sender'], 'to:', req['recipient'])
    print('Msg body:', req['msg'])
    if req['recipient'] != 'any':
        if req['recipient'] in messages:
            messages[req['recipient']].append({'sender': req['sender'], 'msg': req['msg']})
        else:
            messages[req['recipient']] = [{'sender': req['sender'], 'msg': req['msg']}]
    else:
        if len(messages['any']) == 0:
            messages['any'].append({'sender': req['sender'], 'msg': req['msg']})
        else:
            first = messages['any'][0]['sender']
            second = req['sender']
            if first in messages:
                messages[first].append({'sender': second, 'msg': '{"msg":"challenge_ok"}'})
            else:
                messages[first] = [{'sender': second, 'msg': '{"msg":"challenge_ok"}'}]
            if second in messages:
                messages[second].append({'sender': first, 'msg': '{"msg":"challenge_ok"}'})
            else:
                messages[second] = [{'sender': first, 'msg': '{"msg":"challenge_ok"}'}]
                del messages['any'][0]
    return 'OK'
```

Рисунок 2.4. Подключение двух клиентов.

2.3 Описание структур данных

2.3.1 Клиент

В качестве основного класса можно привести класс для реализации игрового процесса. В данном классе находятся функции, предназначенные для исполнения самой игровой логики и использования дополнений (рисунок 2.5).

```
void CountWidget::use_obj()
{
    if (b == 1)
    {
        other_counter=other_counter - 5;
        if(other_counter < 0)
            other_counter = 0;
        ui->label_2->setText(QString("%1").arg(other_counter));
        b = 0;
        obj = 0;
        emit hide();
    }
    if (ba == 1)
    {
        other_counter = 0;
        ui->label_2->setText(QString("%1").arg(other_counter));
        ba = 0;
        obj = 0;
        emit hide();
    }
    if (l == 1)
    {
        other_counter =  other_counter - 2;
        if(other_counter < 0)
            other_counter = 0;
        ui->label_2->setText(QString("%1").arg(other_counter));
        l = 0;
        obj = 0;
        emit hide();
    }
    if (n == 1)
    {
        other_counter = other_counter - 7;
        if(other_counter < 0)
            other_counter = 0;
        ui->label_2->setText(QString("%1").arg(other_counter));
        n = 0;
        obj = 0;
        emit hide();
    }
}
```

Рисунок 2.5. Использование усилителей

2.3.2 Сервер

Для хранения зарегистрированных пользователей была создана таблица «Shows» в базе данных SQLite. Данная таблица (рисунок 2.6) содержит

базовую информацию об аккаунтах: то есть логин, пароль и идентификатор пользователя.

	ID	name	password
	Фи...	Фил...	Фильтр
1	1	admin	admin
2	2	admin1	admin1

Рисунок 2.6. – Таблица данных пользователей

Для хранения данных о приобретении «усилителей» для более интересной игры во внутриигровом магазине используется таблица «Shop» (рисунок 2.7). Данная таблица содержит идентификатор пользователя, количество игровых монет и количество объектов.

	id	coin	obj
	Фи...	Фи...	Фи...
1	(1,)	400	1

Рисунок 2.7. – Таблица данных о приобретении «усилителей»

2.4 Описание основных модулей

2.4.1 Клиент

После запуска клиента пользователь сразу попадает на страницу авторизации и регистрации (рисунок 2.8), то есть он может либо создать новый аккаунт, либо войти в существующий. Если данный пользователь существует при регистрации, то будет выведено соответствующее сообщение. Если пользователь ввёл неверный логин или пароль, то также высветится сообщение об ошибке.

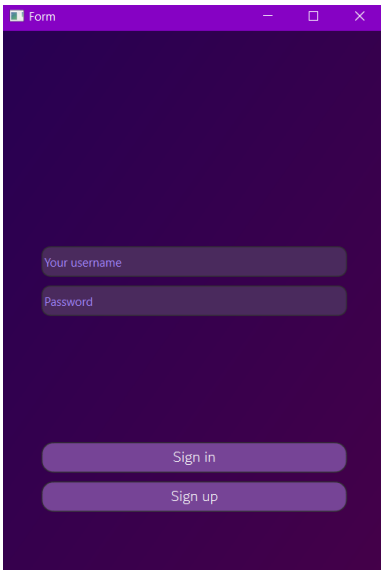

Рисунок 2.8. Окно авторизации/регистрации

После авторизации появляется главное меню (рисунок 2.9). Справа в верхнем углу отображается количество монет пользователя. С помощью кнопки в левом верхнем углу можно попасть во внутриигровой магазин, а с помощью кнопки по середине можно начать саму игру.

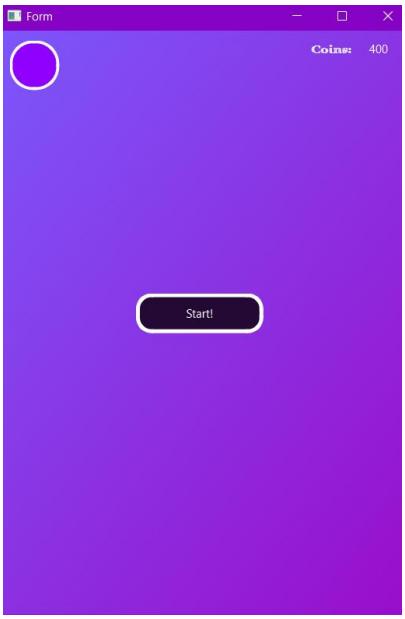

Рисунок 2.9. Главное меню

2.4.2 Сервер

Сервер исполняет функцию хранения информации о пользователях и работы с ней. Для регистрации пользователя происходит проверка существующих пользователей, чтобы не было двух одинаковых аккаунтов, а для авторизации происходит проверка на верность введённых данных, которые приходят от клиента (рисунок 2.10).

```
You, a month ago | 1 author (You)
import socket as Socket      You, a month ago • curs
import json
import sqlite3

conn = sqlite3.connect('shows.db')
cursor = conn.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS Shows
                  (ID INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, password TEXT)''')
address = "localhost"
port = 3000
dataPackageSize = 1024


def listenClients(connection, clientAddress):
    while 1:
        try:
            data = connection.recv(dataPackageSize)
        except ConnectionError:
            break
        if not data:
            break
        print("Принято сообщение: ", data.decode('utf-8'))
        data_bytes = data
        data = json.loads(data_bytes.decode('utf-8'))
        # sendall - принимает массив байт
        if(data["action"] == "register"):
            # connection.sendall(data_bytes)
            cursor.execute("SELECT COUNT(ID) FROM shows WHERE name = ?", (data["name"],))
            count = cursor.fetchall()
            print(count)
            if(count == [(0, )]):
                cursor.execute("INSERT INTO shows (name, password) VALUES (?, ?)", (data["name"], data["password"]))
                conn.commit()
            else:
                print("This user already exist")
                connection.sendall("no".encode())
        elif (data["action"] == "login"):
            cursor.execute("SELECT ID FROM shows WHERE name = ? AND password = ?", (data["name"], data["password"]))
            users = cursor.fetchall()
            print(users)
            if(users != []):
                data1 = ", ".join([str(i) for i in users])
                connection.sendall(data1.encode())
            else:
                print("This user does not exist")
                connection.sendall("login_fail".encode())

    print(f"{clientAddress} has disconnect")
```

Рисунок 2.10. Функции сервера на примере авторизации/регистрации

Глава 3

Тестирование программы

3.1 Тестирование исходного кода

Тесты исходного кода сервера покрывают работу с базой данных и составлением ответа сервера на запрос клиента.

Тесты исходного кода клиента покрывают работу алгоритма игрового процесса и корректность отправляемых запросов.

3.2 Тестирование интерфейса пользователя

Проект разделён на сервер и клиент (рисунок 3). Тесты были разделены по частям: 3 теста на сервер и 3 теста на клиент.

Содержание тестов(сценариев)/Test Cases (Scenarios Summary)

Идентификатор Test ID	Цель теста/Purpose of test
Тест 1 - Сервер	Запуск сервера
Тест 2 - Сервер	Создание базы данных
Тест 3 - Сервер	Проверка работы базы данных и сервера
Тест 1 - Клиент	Запуск клиента
Тест 2 - Клиент	Проверка работы формы авторизации и регистрации
Тест 3 - Клиент	Проверка работы самой игры

Рисунок 3. Содержание тестов

Первый тест направлен на проверку корректного запуска серверной части (рисунок 3.1).

Тест 1 - Сервер

Среда тестирования/Environment	Visual Studio Code
Предварительные действия/Pre Requisites	Запуск среды разработки Visual Studio Code
Комментарии/Comments	-

Шаг/ Step №.	Действие (операция)/Process (Actions)	Ожидаемый результат/Expected Result	Результат/Actual Result	Пройден /не пройден/ не доступен*	Комментарии / Notes
1	Запуск программы	Программа запускается без ошибок. В консоли высвечивается сообщение: “Server Started”			

Рисунок 3.1. Тест 1 - Сервер

Второй тест направлен на проверку принятия запросов от клиента (рисунок 3.2).

Шаг/ Step №.	Действие (операция)/Process (Actions)	Ожидаемый результат/Expected Result	Результат/Actual Result	Пройден /не пройден/ не доступен*	Комментарии / Notes
1	Запустить клиент в QT Creator	При введение данных в поля и нажатии кнопки Sign up, в консоль сервера выводится: «{порт} has connected Принять сообщение: {“action”: “register”, “name”: {name}, “password”:{password} } [(0,)] {порт} has disconnect», где {name} и {password} данные, которые вводились на клиенте.			
2	Открыть папку с файлами сервера после выполнения Шага №1	Появилась база данных shows с данными, которые были введены в клиенте.			

Рисунок 3.2. Тест 2 - Сервер

Третий тест нацелен на проверку данных, которые были получены сервером, и отправка ответа на клиент (рисунок 3.3 – 3.3.2).

Шаг/ Step №.	Действие (операция)/Process (Actions)	Ожидаемый результат/Expected Result	Результат/Actual Result	Пройден /не пройден/ не доступен*	Комментарии / Notes
1	Запустить клиент в QT Creator	При введение данных в поля и нажатии кнопки Sign up, в консоль сервера выводится: «{порт} has connected Принять сообщение: {“action”: “register”, “name”: {name}, “password”:{password} } [(0,)] {порт} has disconnect», где {name} и {password} данные, которые вводились на клиенте. В базе данных shows появится новая запись.			
2	Ввести те же данные, что и в	В консоль сервера выводится:			

Рисунок 3.3. Тест 3 - Сервер

	прошлом тесте и нажать кнопку Sign up	«{порт} has connected Принять сообщение: {“action”: “register”, “name”: {name}, “password”:{password} } [(1,)] This user already exist», где {name} и {password} данные, которые вводились на клиенте. В базе данных новых записей не появляется			
3	Ввести новые данные в текстовые поля и нажать кнопку Sign in	В консоль сервера выводится: «{порт} has connected Принять сообщение: {“action”: “register”, “name”: {name}, “password”:{password} } [] This user does not exist», где {name} и {password} данные, которые вводились на клиенте. В базе данных новых записей не появляется			
4	Ввести данные, которые	В консоль сервера выводится:			

Рисунок 3.3.1. Тест 3 - Сервер

	вводились в шаге №1	«{порт} has connected Принять сообщение: {“action”: “register”, “name”: {name}, “password”:{password} } [(id,)] {порт} has disconnect», где {name} и {password} данные, которые вводились на клиенте.[(id,)] – номер в базе данных, который соответсвует данным. В базе данных новых записей не появляется.			

Рисунок 3.3.2. Тест 3 - Сервер

Далее идут тесты, ответственные за клиент. Первый тест отвечает за запуск клиента (рисунок 3.4).

Тест 1 - Клиент

Среда тестирования/Environment	QT Creator 4.14.1 (Community)
Предварительные действия/Pre Requisites	Запуск среды QT Creator 4.14.1 (Community)
Комментарии/Comments	-

Шаг/ Step №.	Действие (операция)/Process (Actions)	Ожидаемый результат/Expected Result	Результат/Actual Result	Пройден /не пройден/ не доступен*	Комментарии / Notes
1	Запуск программы	Программа запускается без ошибок. Появляется окно с формой регистрации.			

Рисунок 3.4. Тест 1 - Клиент

Второй тест отвечает за проверку введённых данных на клиенте и создание запросов для сервера (рисунок 3.5 – 3.5.2).

Шаг/ Step №.	Действие (операция)/Process (Actions)	Ожидаемый результат/Expected Result	Результат/Actual Result	Пройден /не пройден/ не доступен*	Комментарии / Notes
1	Запустить сервер в VS Code	При введение данных в поля и нажатии кнопки Sign up, в консоль клиента выводится:			
2	Нажать на кнопку Sign up без ввода данных.	Появляется сообщение с содержанием: «Пожалуйста, введите данные».			
3	Нажать на кнопку Sign in без ввода данных.	Появляется сообщение с содержанием: «Пожалуйста, введите данные».			
4	Ввести данные в поля и нажать на кнопку Sign up	В консоль клиента выводится: “Connected to server Data sent successfully "" Disconnect from server”			После того как пользователь зарегистрировался, ему будет необходимо

Рисунок 3.5. Тест 2 - Клиент

					войти на свой аккаунт.
5	Ввести данные из шага №4 и нажать кнопку Sign Up	В консоль клиента выводится: “Connected to server Data sent successfully. Disconnect from server” Появляется сообщение с содержанием: «Такой пользователь уже существует»			
6	Ввести новые данные в поля и нажать кнопку Sign in	В консоль клиента выводится: “Connected to server Data sent successfully. Disconnect from server” Появляется сообщение с содержанием: «Неверный логин или пароль»			
7	Ввести данные из шага №4 и нажать кнопку Sign In	В консоль клиента выводится:” Connected to server Data sent successfully "({id},)" Disconnect from server”, где {id} – номер пользователя из базы данных, который клиент получил из			

Рисунок 3.5.1. Тест 2 - Клиент

		сервера. Открывается окно с игрой.			

Рисунок 3.5.2. Тест 2 - Клиент

Третий тест проверяет правильное выполнение логики игры (рисунок 3.6 – 3.6.1).

Шаг/ Step №.	Действие (операция)/Process (Actions)	Ожидаемый результат/Expected Result	Результат/Actual Result	Пройден /не пройден/ не доступен*	Комментарии / Notes
1	Нажать на нижнюю кнопку	Картинка двигается вниз. Счётчик справа считает количество кликов. Счётчик по середине считает разницу двух счётчиков справа от кнопок. Когда он становится равен 10, высвечивается сообщение о выигрыше. При нажатии на кнопку Cancel игра завершается.			При сборке программы картинка может потеряться из-за того, что на компьютере будет создана другая папка сборки, поэтому необходимо будет добавить туда картинку, которая находится в папке build в репозитории.
2	Нажать на верхнюю кнопку	Картинка двигается вверх. Счётчик справа считает количество кликов. Счётчик по середине считает разницу двух счётчиков справа от кнопок. Когда он становится равен 10, высвечивается сообщение о выигрыше. При нажатии на кнопку Cancel игра завершается.			

Рисунок 3.6. Тест 3 - Клиент

Рисунок 3.6.1. Тест 3 - Клиент

Глава 4

Программная реализация приложения

4.1 Перспективы технического развития

За время создания проекта было реализовано практически всё из того, что ставилось в качестве цели. Это говорит о том, что проект является рабочим. Однако остаются и некоторые вещи, которые можно доработать в процессе дальнейшего технического развития проекта. К примеру, нужно добавить поиск игроков по логину. На данный момент эта функция находится на стадии разработки. Также не реализована возможность смены или восстановления пароля пользователя, к примеру, в случае его утери.

Можно выделить дополнительные, но не критичные на данный момент недостатки:

1. Уязвимость сервера к DDoS атакам.
2. Недостаточная оптимизация сервера в случае высокой нагрузки на сервер. Скажем, если на сервере будет одновременно 1000 клиентов, то может возникнуть задержка передаваемых данных.

4.2 Перспективы монетизации

Данный проект является открытым с отсутствием монетизации. При желании, конечно, можно добавить медийную или контекстную рекламу. Это самый популярный метод. Либо это можно осуществить с помощью партнерских программ (по сути, косвенная реклама).

Также возможна отдельная версия клиента, расширенная. С другими функциями и несколькими мини-играми и событиями, чтобы играть было ещё интереснее.

Заключение

Idle-игры очень популярны в настоящее время, хоть и кажутся очень простыми и неинтересными. Но прогресс ради прогресса тоже приносит удовольствие, как и дух соперничества в данной игре. Одним из преимуществ игры является то, что игроку будет легко выйти из игры и вернуться в неё. То есть, если в других играх отсутствие игрока будет заметно тем, что будут пропущены какие-то события, ограниченные по времени, или придётся восстанавливать ресурсы. Например, в FarmVille вянут растения. Если вы вовремя не вернётесь в игру, ваши культуры могут погибнуть. В Clash of Clans у вас могут украсть ресурсы. Чем больше времени вы отсутствуете, тем больше риск. Кроме того, ваш ранг может понизиться, а ваш клан расстраивается из-за того, что вы не вводите в битву свои войска. Это может сподвигнуть игрока заходить в игру чаще, а может заставить его и вовсе забросить игру. Здесь же такого нет.

Во время выполнения данного проекта был получен новый опыт в разработке относительно крупных проектов, распределения времени, планирования, выставления приоритетов с учётов дедлайнов. Так же был получен опыт в поиске информации, её анализе.

Некоторые части продукта нуждаются в совершенстве, но весь важный функционал программы является рабочим и полностью соответствует техническому заданию.

Литература

1. Клиент-сервер [электронный ресурс] - режим доступа: https://galtsystems.com/blog/start/klient_server_o_tekhnologii_prostymi_slovami/;
2. Клиент-сервер [электронный ресурс] - режим доступа: https://it-black.ru/tekhnologiya-kliyent-server/;
3. Инкрементальная игра [электронный ресурс] - режим доступа: https://ru.wikipedia.org/wiki/Инкрементальная_игра;
4. Idle-игры [электронный ресурс] - режим доступа: https://vc.ru/flood/8083-idle;
5. Idle-игры [электронный ресурс] - режим доступа: https://pdalife.ru/search/idle-games/;
6. Idle-игры [электронный ресурс] - режим доступа: https://ru.y8.com/tags/idle_game;