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 БРАУЗЕРА ДЛЯ ИНТЕРАКТИВНОЙ ЛИТЕРАТУРЫ**

Курсовая работа  
по дисциплине «Программирование»

студента 1 курса группы ПИ-б-о-201(1)

Шенгелай Всеволода Михайловича

направления подготовки 09.03.04 «Программная инженерия»

(код и наименование)

|  |  |  |
| --- | --- | --- |
| Научный руководитель  старший преподаватель кафедры компьютерной инженерии и моделирования | \_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Чабанов В.В. |

Симферополь, 2021

# РЕФЕРАТ

Шенгелай Всеволод Михайлович – Проект «Разработка браузера для интерактивной литературы» – Симферополь: ФТИ КФУ им. В. И. Вернадского, 2021. – 25 с., 9 ил., 8 ист.

Объект разработки – браузер для интерактивной литературы.

Цель работы – создать сервис, позволяющий пользователям проходить текстовые квесты с возможностью возвращаться к уже пройденным главам, а авторам квестов – размещать свои работы в данном приложении; получить опыт создания многофайловых программ, передающих между собой данные по протоколу https.

Проблема работы – практически полное отсутствие сервисов, позволяющих пользователям проходить текстовые квесты (или интерактивную литературу) на компьютерах под управлением Windows.

Метод разработки – спиральная модель разработки ПО.

Актуальность работы ­ – почти все приложения с интерактивными квестами являются программами, продающими контент. То есть они стилизованы под конкретный квест и имеют целью продать его пользователю. При этом данный контент может «продаваться» по 10 раз. Наше приложение также содержит квесты, но их будет несколько десятков, интерфейс для прохождения интерактивной литературы будет един для всех квестов, постоянно будут появляться новые.

Язык С++ будем использовать для организации сервера (возможность создания высокопроизводительных многопоточных программ), а язык Python – для создания клиента (наличие огромного числа GUI - библиотек и их недооцененность).

# 

С++, PYTHON, ИНТЕРАКТИВНАЯ ЛИТЕРАТУРА, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА ПО, TKINTER, JSON, PYGAME

# ОГЛАВЛЕНИЕ

[РЕФЕРАТ 2](#_Toc76409005)

[ОГЛАВЛЕНИЕ 3](#_Toc76409006)

[ВВЕДЕНИЕ 4](#_Toc76409007)

[ГЛАВА 1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc76409008)

[1.1 Цель проекта 6](#_Toc76409009)

[1.2 Существующие аналоги 6](#_Toc76409010)

[1.3 Основные отличия от аналогов 7](#_Toc76409011)

[1.4 Техническое задание 7](#_Toc76409012)

[ГЛАВА 2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 9](#_Toc76409013)

[2.1 Анализ инструментальных средств 9](#_Toc76409014)

[2.2. Описание алгоритмов 10](#_Toc76409015)

[2.2.1. Алгоритм «чтения» текстовых квестов 10](#_Toc76409016)

[2.2.2. Алгоритм выбора текстовых квестов 13](#_Toc76409017)

[2.2.3. Серверные функции 13](#_Toc76409018)

[2.3. Описание структур данных 14](#_Toc76409019)

[2.3.1. Клиент 14](#_Toc76409020)

[2.3.1. Сервер 15](#_Toc76409021)

[2.4. Описание основных модулей 15](#_Toc76409022)

[ГЛАВА 3 ТЕСТИРОВАНИЕ ПРОГРАММЫ 19](#_Toc76409023)

[3.1 Тестирование исходного кода 19](#_Toc76409024)

[3.2 Тестирование интерфейса пользователя и юзабилити 20](#_Toc76409025)

[3.3 Исправление найденных багов 21](#_Toc76409026)

[ГЛАВА 4 ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА 22](#_Toc76409027)

[4.1 Перспективы технического развития 22](#_Toc76409028)

[4.2 Перспективы монетизации 22](#_Toc76409029)

[ЗАКЛЮЧЕНИЕ 24](#_Toc76409030)

[ЛИТЕРАТУРА 25](#_Toc76409031)

[ПРИЛОЖЕНИЕ 1 26](#_Toc76409032)

# ВВЕДЕНИЕ

Книги постепенно канут в лету, линейность и сложность ассоциирования себя с героем делают их слишком нереальными и даже немного скучными. Игры, сериалы и фильмы почти полностью уничтожили художественную литературу. В 1980х годах, когда вычислительные возможности компьютеров не позволяли запускать на них графические игры, появились текстовые квесты. Их нелинейность, погружение в мир рассказа и олицетворение себя с персонажем, давали им огромное преимущество перед обычной литературой. С появлением графических игр развитие этого жанра замедлилось. Но в настоящее время, с появлением визуальных платформ создания упаковки текстовых игры и интерактивных историй в компактный HTML-файл, их популярность значительно возросла

Задачей курсовой работы является создание полноценного программного и аппаратного продукта, который должен будет пройти тестирование пользователями.

Целью работы является повысить навыки разработки приложений с графическим интерфейсом под ОС Windows, научиться создавать консольные сервера на С++, изучить информацию о создании и продвижении программных продуктов, выбрать один из способов их создания и разработать приложение под ОС Windows, позволяющее пользователям проходить текстовые квесты с возможностью возвращаться к уже пройденным главам, а авторам квестов – размещать свои работы в данном приложении.

Для реализации поставленной цели следует решить следующие задачи:

1. Рассмотрение известных примеров текстовых игр-квестов, ознакомление с общими принципам их работы;
2. Выбор типа создаваемого приложения и выбор программных средств для его разработки;
3. Создать прототип интерфейса цифрового продукта;
4. Разработать дизайн приложения;
   1. Создать иконку приложения
   2. Выбрать/нарисовать заставки, обои, темы
5. Разработать функционирующий клиент;
6. Разработать сервер;
7. Выполнить отладку и тестирование приложения.

Объектом работы в данном проекте является создание дескотпного приложения, которым можно будет пользоваться offline и сервера на С++, где будут храниться пользовательские данные (очки и игровая валюта) и ещё не скаченные квесты.

# ГЛАВА 1 ПОСТАНОВКА ЗАДАЧИ

## 1.1 Цель проекта

Создание рабочего проекта, получение опыта в разработке крупных проектов. Укрепление навыков использования и внедрения сторонних C++ библиотек, их изучение. Получение опыта разработки программ с графическим интерфейсом на языке Python.

Стоит отметить, что в качестве модели процесса разработки использовалась спиральная модель. Спиральная модель — это модель процесса разработки программного обеспечения с учетом рисков. Это комбинация модели водопада и итеративной модели. Spiral Model помогает внедрить элементы разработки программного обеспечения из нескольких моделей процессов для программного проекта на основе уникальных шаблонов рисков, обеспечивая эффективный процесс разработки.

## 1.2 Существующие аналоги

* Zork и последующие игры в мире Zork — действие игры происходит в запутанном подземном лабиринте, занимающем часть «Великой Подземной Империи». Игрок — безымянный искатель приключений, чья цель — найти спрятанные в пещерах сокровища, выжить и вернутся с ними на поверхность. В подземельях встречается множество оригинальных существ и вещей. Одна из первых игр жанра.
* Пандемия: Дорога домой. Сюжетный квест – выживание. Приложение под Android. Яркий пример приложения, продающего контент.
* Квестоманьяк – приложение под Android. Больше всего похоже на наш проект, но его серверная часть служит только для хранения очков – пользователи не могут выбирать квесты для скачивания, и получают всю предложенную разработчиком литературу со следующим обновлением приложения.

## 1.3 Основные отличия от аналогов

Для того, чтобы пройти текстовый квест, или его отдельную главу, людям не придётся устанавливать новую программу и удалять старую (когда квест пройден – она становится бесполезной). Они смогут скачать квест любого жанра из предложенных нами. Это не исключает того, что авторы, уже разместившие свои квесты в отдельных программах, смогут написать адаптированную под наши форматы файлов его версию.

## 1.4 Техническое задание

1. Интерфейс программы должен содержать несколько окон:
   1. Окно просмотра уже имеющейся интерактивной литературы;
   2. Окно со списком квестов, которые можно скачать;
   3. Внутриигровой магазин;
2. Состав интерактивной литературы:
   1. Содержание квеста:
      1. Текст
      2. Постер
      3. Звуковая тема
   2. Описание, прилагаемое к квесту:
      1. Название, автор
      2. Категория (например, фентези, детектив)
      3. Описание
      4. Статистические данные:
         1. Количество глав
         2. Количество успешных концовок
         3. Количество проигрышных концовок
         4. Уровень сложности
         5. Протяжённость
         6. Вариативность выбора
3. Система начисления очков и внутриигровой валюты:
   1. За каждую главу квеста в зависимости от её сложности начисляется определённое количество очков, которые необходимы только для определения места в глобальной статистике самого пользователя
   2. Внутриигровая валюта позволяет пользователю покупать дополнительные возможности:
      1. Возможность оставлять закладки
      2. Возможность заглянуть вперёд (предсказывание последствий хода)
4. Для обеспечения возможности скачать квесты и узнать о своём месте в глобальной статистике по собранным очкам, будет написан сервер, на который на этапе тестирования приложения возможно будет возложена дополнительная задача приёма данных отслеживания действий пользователей в приложении

# ГЛАВА 2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

## 2.1 Анализ инструментальных средств

На этапе определения и планирования проекта мы выбирали тип будущего клиента – нативное, или кроссплатформенное, при этом выделили плюсы и минусы каждого из вариантов.

Нативное приложение – это тип приложений, разрабатываемый на «родном» для каждой платформы языке программирования.

Кроссплатформенные приложения – это решение, разработанное таким образом, чтобы иметь возможность, с минимальными усилиями, скомпилировать исходный код для исполнения на нескольких платформах, но результатом каждой отдельной компиляции будут отдельные исполняемые файлы.

Наш проект не использует типовых решений, поэтому было решено сделать нативный клиент. Сложность в написании кроссплатформенного приложения состоит также в том, что на выбранный нами язык для клиентской части существует лишь один такой фреймворк ­– Kivy, и для того, чтобы написать на нём программу не под Windows – придётся учить абстрактный API, созданный командой разработчиков фреймворка под данную платформу, который, к тому же, работает не стабильно при отхождении от шаблонных операций.

Программные средства, применяемые в процессе выполнения курсовой работы:

* Среда Microsoft Visual Studio 2019 ­­– удобный интерфейс, возможность простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня, позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кодa.
* C++ – как изучаемый язык программирования
* Библиотека JSON for Modern C++ – header-only библиотека для работы с информацией в формате json. Используется для хранения квестов, конфигурационной информации пользователей на сервере и клиенте. Json удобен в первую очередь для программиста.
* Модуль tkinter для Python – был выбран из-за опыта работы с ним разработчика во время оценки сроков проекта и учёта рисков
* Модуль pygame.mixer – используется для работы с аудио. Нам потребуется воспроизводить только фоновую музыку, а лишних методов этот модуль не содержит.
* Библиотека Pillow – обеспечивает обширную поддержку форматов файлов, эффективное внутреннее представление и довольно мощные возможности обработки изображений. Быстрая загрузка и обработка изображений.
* Модуль Requests – упрощает веб-запросы, оставаясь при этом чрезвычайно мощным и разноплановым инструментом.

## 2.2. Описание алгоритмов

### 2.2.1. Алгоритм «чтения» текстовых квестов

При клике на текстовый квест в списке уже имеющихся на компьютере – запускается функция init() модуля Quest\_viever, в котором содержится главный алгоритм программы. Она принимает 3 входных параметра:

* root – ссылка на корневое окно
* current\_quest – название выбранного квеста
* GamePointsLabel – ссылка на таблицу имающихся у игрок очков

Изменения графического интерфейса похожи на смену экрана приложения на мобильном устройстве. Поверх уже расположенных виджетов появляется поле чтения квестов, при этом остаётся открытой расположенная внизу панель навигации по программе.

Все тестовые квесты содержатся в файлах с расширением json со строго определённой структурой. Таким образом, нам нужен всего один модуль чтения для всех квестов.

После запуска квеста первым делом будет отображён список глав, из которых он состоит (рисунок 2.1). Чтобы начать главу, нужно нажать по кнопке с соответствующим названием.

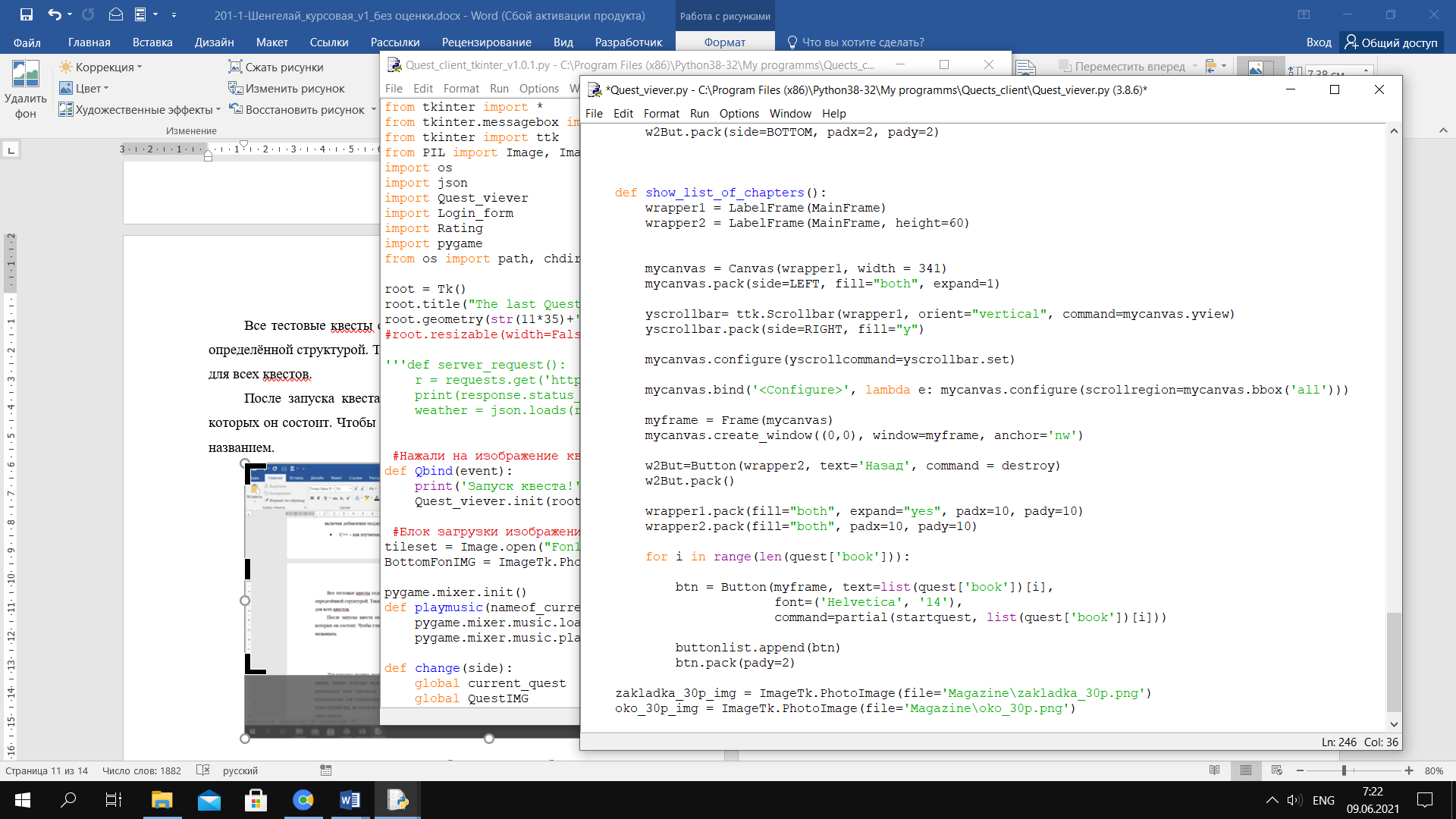


Рисунок 2.1. Цикл for создания кнопок с главами

Каждая из кнопок запускает функцию startquest, которой в качестве аргумента передаёт своё название. Главная часть функции startquest – внутренняя рекурсивно вызываемая функция Change\_buttons, которая убирает старые кнопки-действия, открывает текстовый фрагмент выбранной на предыдущем шаге кнопки и создаёт новые кнопки-действия для текущего шага (рисунок 2.2.). В файле json у каждой кнопки есть свойство goto, указывающее, к какой части главы она ведёт. Соответствующий атрибут мы создаём и у кнопок в коде.

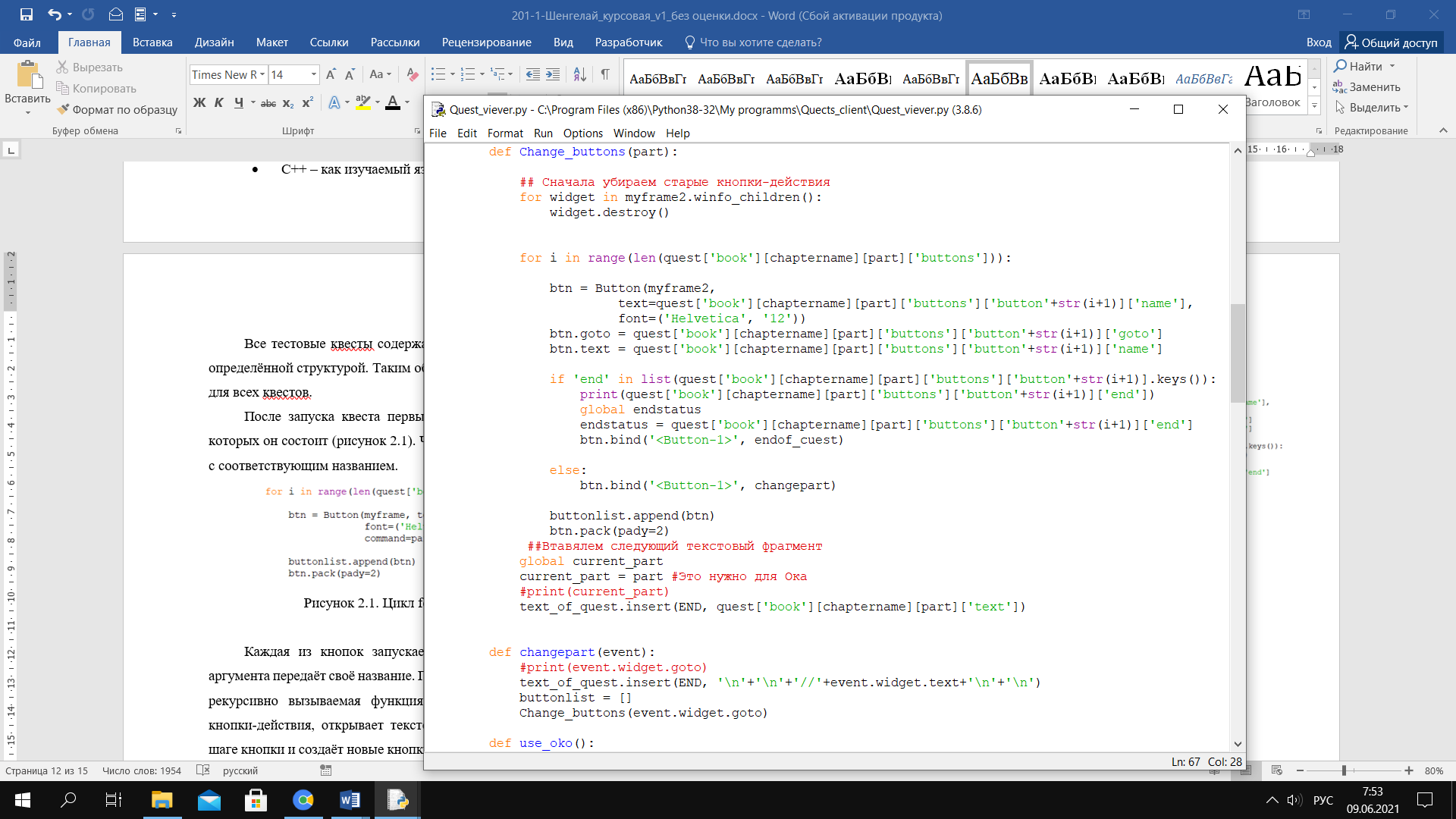


Рисунок 2.2. Код динамической смены кнопок и добавления тестовых фрагментов

В модуле также реализованы «фишки» наших квестов – возможность оставить закладку в любом месте квеста и заглянуть вперёд, узнав, какое последствие будет у каждой кнопки-действия, которые предложены нашему выбору.

Самая простая реализация у предмета «закладка». При её оставлении мы запоминаем в глобальной переменной название текущего фрагмента главы (в json-файле они именуются так: part1, part2, part3, …, и т.д.). При возвращении к закладке мы запускаем ранее описанную функцию Change\_buttons, передавая ей единственный аргумент – название фрагмента главы.

При активировании предмета «око» мы просматриваем все текстовые фрагменты, путь к которым мы находим с помощью аргумента кнопок goto. Также в коде есть проверка на «бесконечность предмета» - в таком случае в файле config.json не будет уменьшаться его количество (рисунок 2.3.).

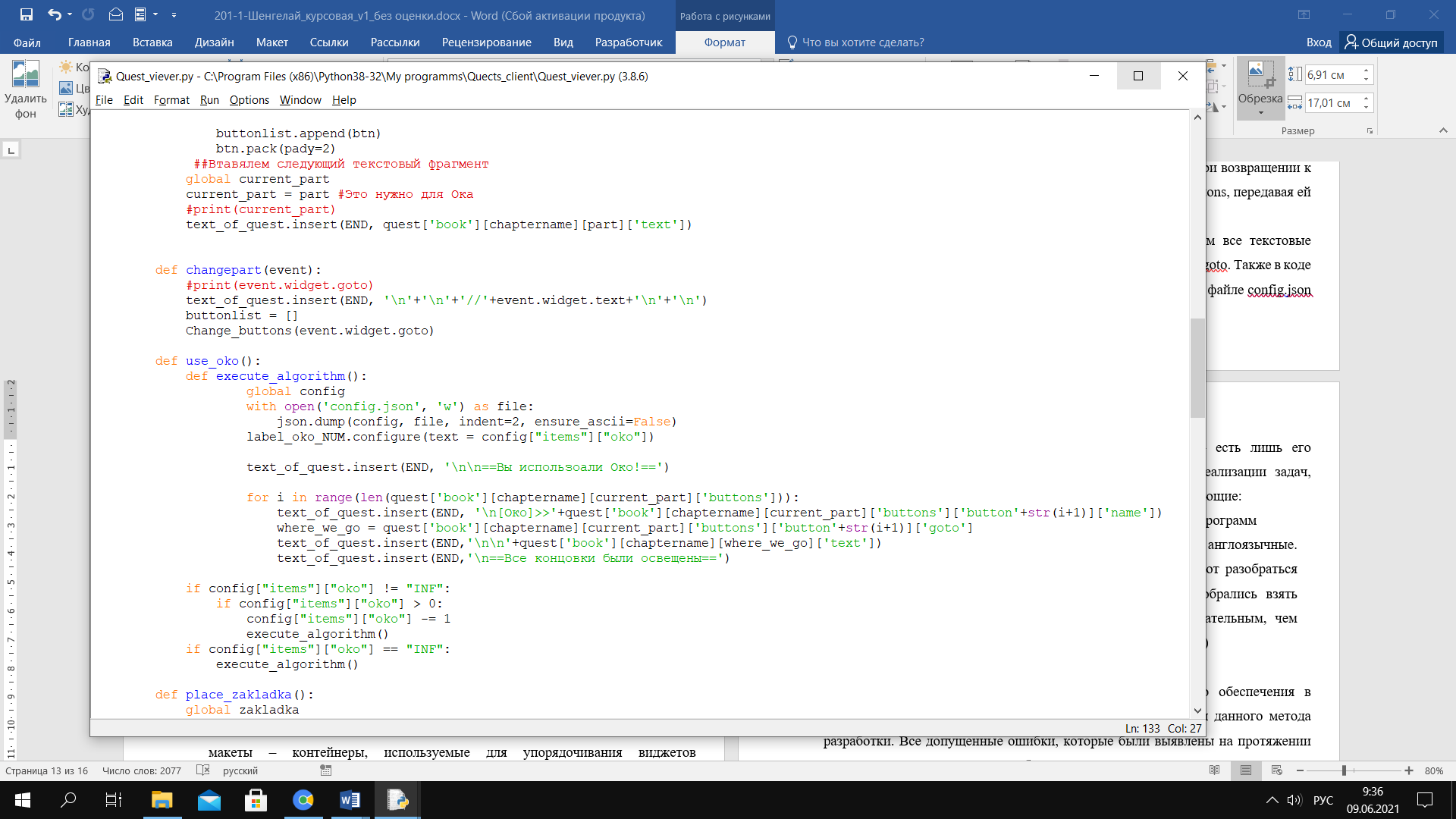


Рисунок 2.3. Функция активации предмета «око»

### 2.2.2. Алгоритм выбора текстовых квестов

Интересным является алгоритм выбора одного из скаченных текстовых квестов для игры. Квест состоит из файла json с самим текстом, постера и аудиофайла музыкальной темы. При этом в конфигурационном файле хранятся данные о когда-либо пройденных квестах и факте получения за них очков, но не о скаченных в настоящее время. Клиент узнаёт о скаченных в данный момент квестах, просматривая папку с постерами – Quest\_images. По названию файлов png программа ищет к ним текст и аудио. Как только мы доходим до границы списка с квестами, выбирая их кнопками «следующий» и «предыдущий», указатель текущего квеста перемещается к противоположной границе.

### 2.2.3. Серверные функции

Код сервера в основном состоит из двух функций: функции регистрации нового пользователя и функции отправки конфигурационного файла клиенту, если его логин и пароль есть у сервера. Код этих функций приведён в приложении 1.

## 2.3. Описание структур данных

### 2.3.1. Клиент

Текстовый квест состоит из трёх файлов, размещённых в трёх папках – в зависимости от расширения файла. Картинки к квестам сохранены в формате png в разрешении 385x584 пикселей – папка Quest\_images. Аудиофайлы хранятся в расширении mp3 – папка Audio. Сами квесты хранятся в папке Quest\_text. Теперь подробнее о внутренней структуре json – файлов с квестами. Json-файл состоит из описания к квесту (имя, автор, вводная информация, количество очков, начисляемых за прохождение) и самой «книги», которая подразделяется на главы, те, в свою очередь, на элементарные части квеста. Каждая такая часть имеет уникальное имя, содержит текстовый фрагмент и набор кнопок, у которых указаны название и фрагмент, к которому они ведут. Внутренняя структура текстового квеста на примере одного из них представлена на рисунке 2.4.

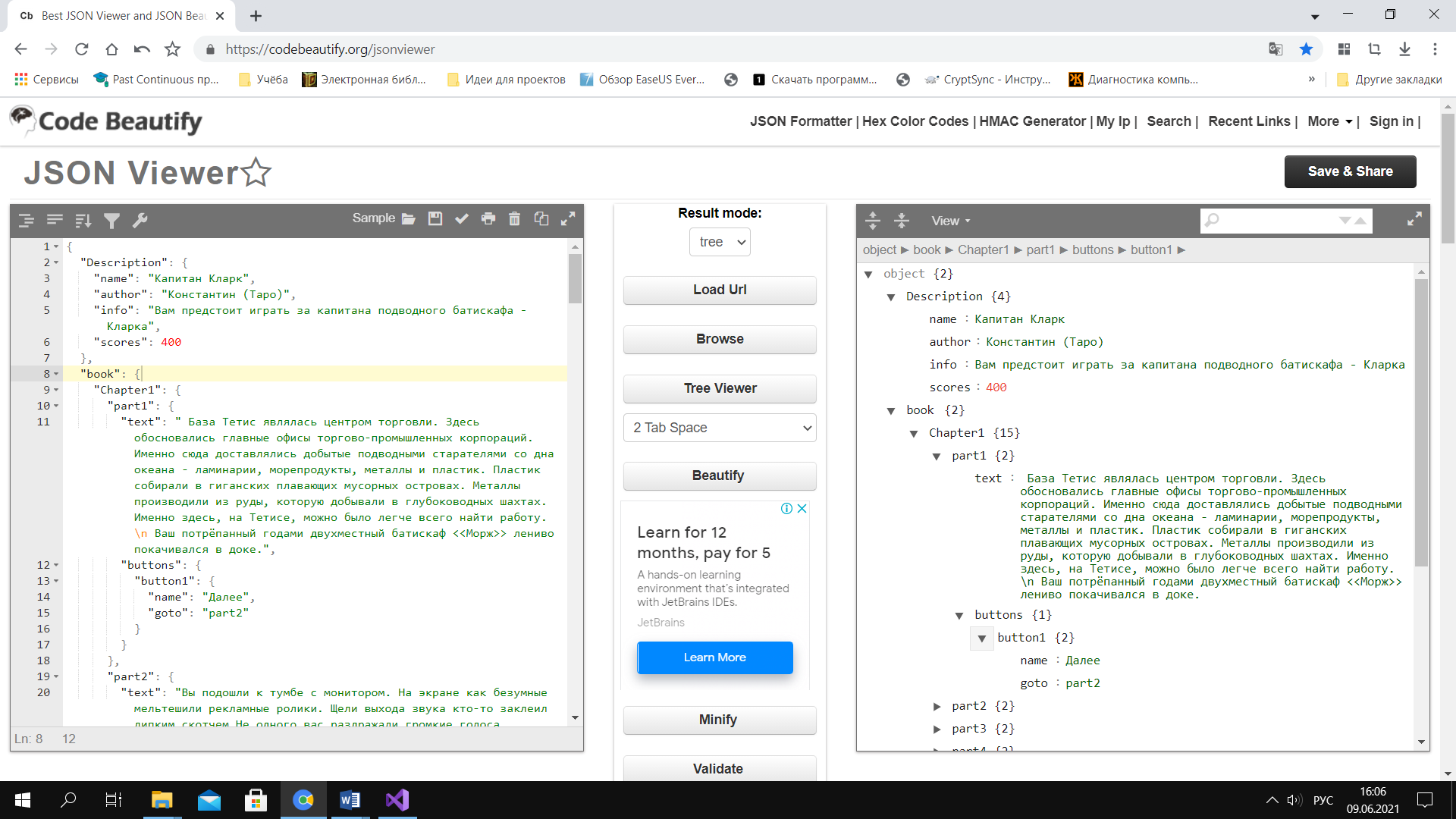


Рисунок 2.4. Демонстрационный пример организации внутренней структуры файла с текстовым квестом

Также в папке с кодом располагаются изображения, используемые в интерфейсе программы: это фон для панели навигации, фон для окна настроек, изображения предметов, покупаемых за внутриигровую валюту.

### 2.3.1. Сервер

На сервере используется только один файл для хранения данных – users.json. В нём под ключами-логинами записаны пароль, количество заработанных очков и конфигурационный файл с клиента в виде строки.

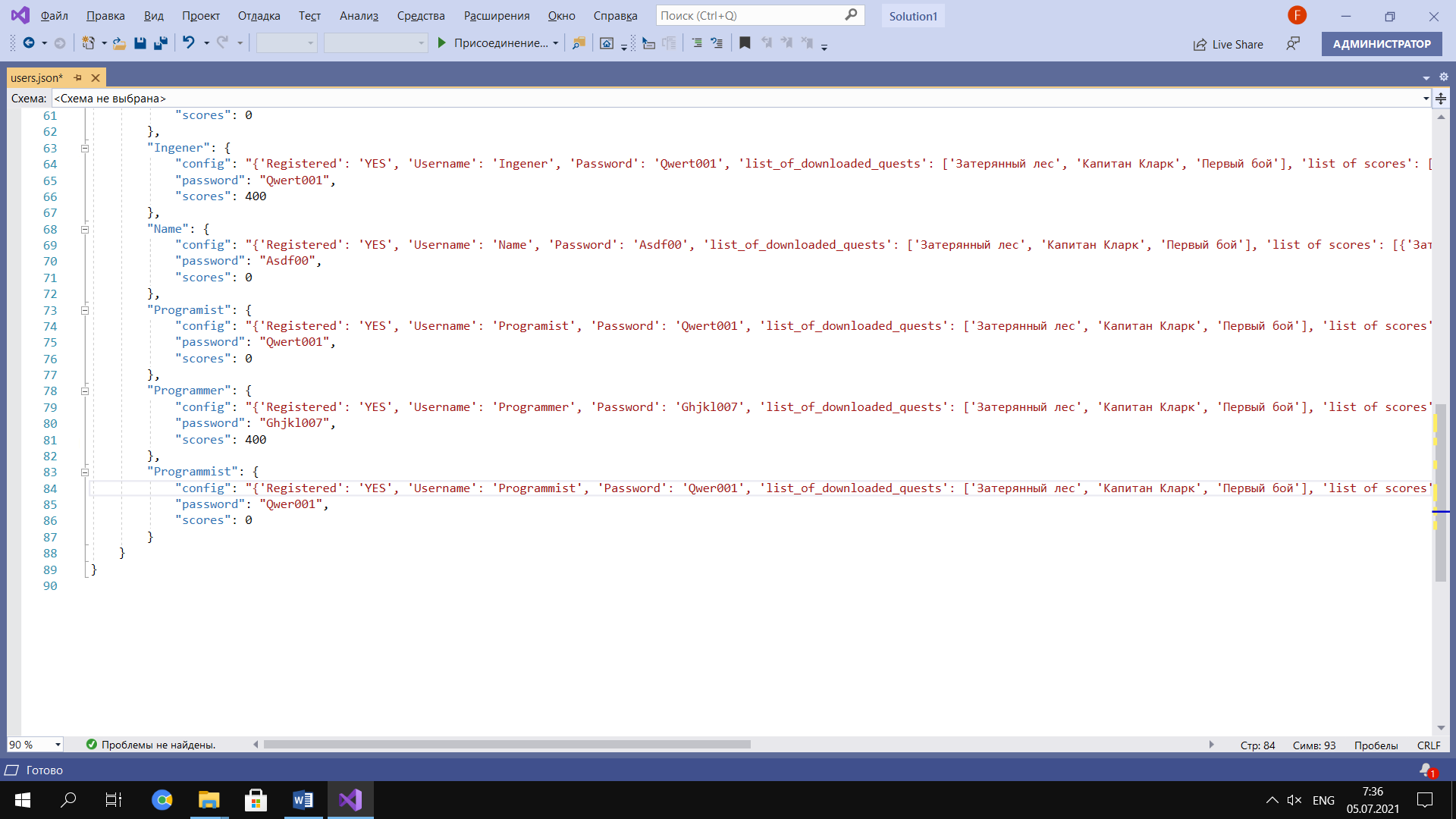


Рисунок 2.5. Тестовые пользовательские данные на сервере

## 2.4. Описание основных модулей

* Quest\_client\_tkinter\_v1.0.1.py – главный файл с расширением .py, в который импортируются все остальные модули. При запуске он проверяет, зарегистрирован ли пользователь. Если да – отображает основное окно программы, иначе – вызывает модуль Login\_form (о нём будет сказано далее). Основное окно состоит из поля квестов и нижней панели навигации по программе (рисунок 2.6). В этом модуле реализованы: просмотр директории запуска клиента для поиска текстовых квестов, смена музыкальной темы в зависимости от выбранной «книги», часть окон, открываемых с нижней панели. На ней расположены кнопки, вызывающие следующие окна:
* Окно настроек (текущий модуль + модуль Login\_Form);
* Окно рейтинга (модуль Rating);
* Окно выбора текстов для скачивания (модуль Download\_quest);
* Окно магазина (текущий модуль)
* Окно «о программе» (текущий модуль)

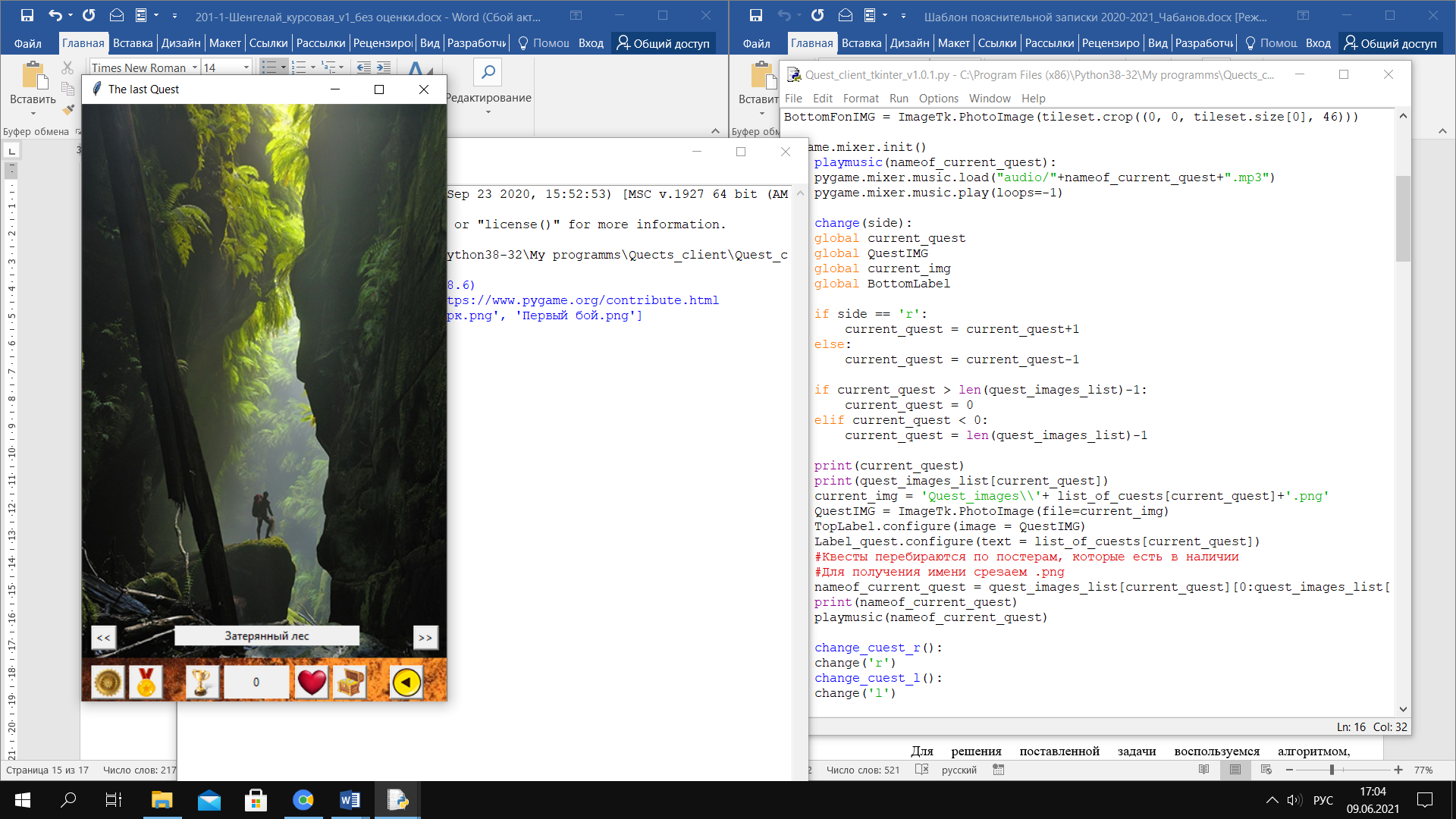
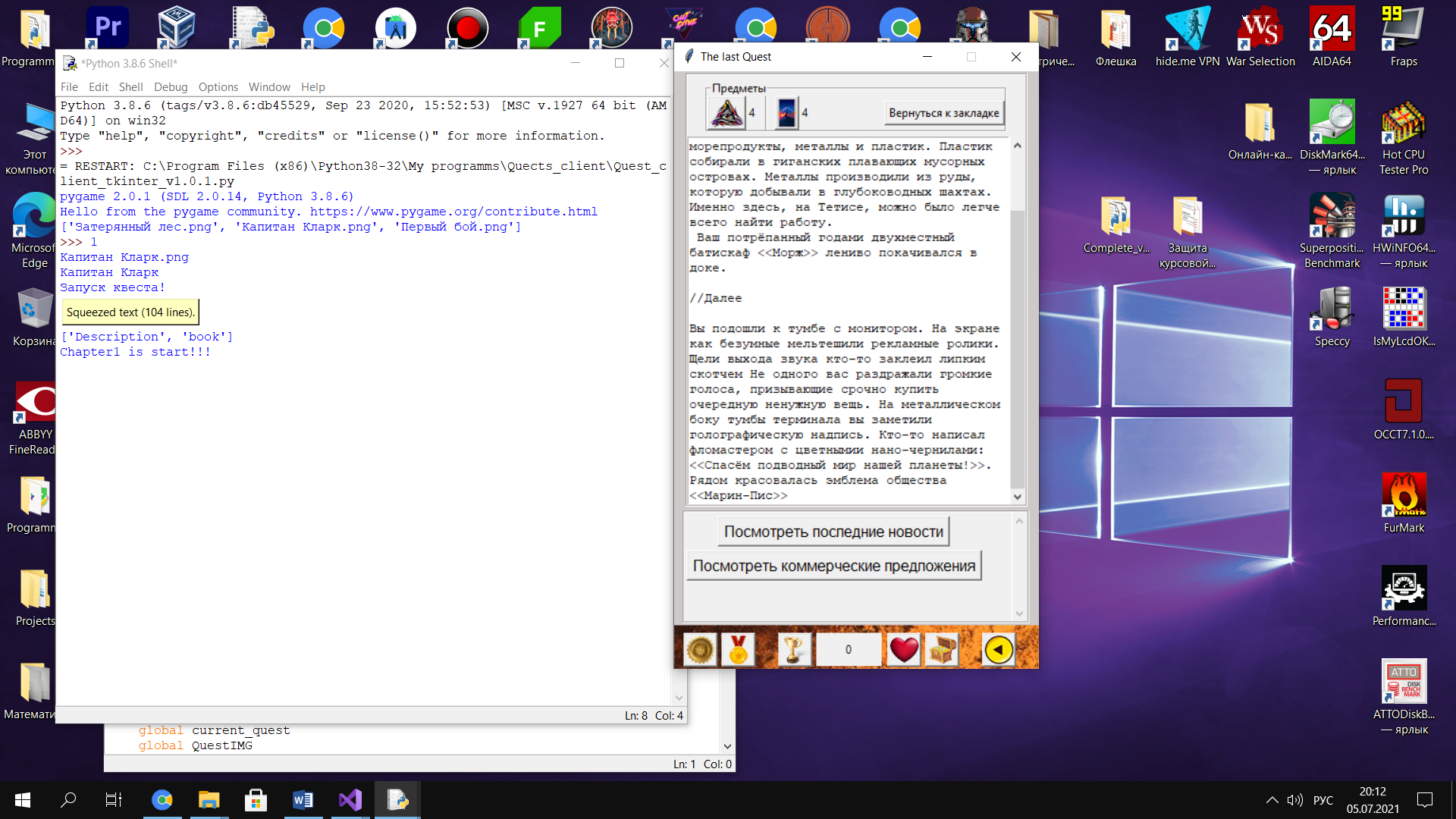
 

Рисунок 2.6. Скриншоты главного окна программы – постер квеста и уже открытая книга

Также на нижней панели расположена информация о полученных очках и кнопка «Назад», закрывающая некоторые из открытых окон. Окно магазина (представлено на рисунке 2.7.) реализует возможность приобретения уникальных предметов за внутриигровую валюту:

* Золотое око (бесконечное всевидящее око)
* Всевидящее око (+3 шт.)
* Золотая закладка (бесконечные закладки)
* Закладка (+2 шт.)

В конфигурационном файле в случае покупки бесконечного предмета его количество будет записано как строка “INF” (бесконечность), и везде, где используются эти данные, будет организована проверка, чтобы не отнять число от строки (или прибавить). В магазине при наличии золотого предмета при покупке обычного не будет вычитаться валюта, а при прохождении квестового текста для золотого предмета не будет открываться конфигурационный план.

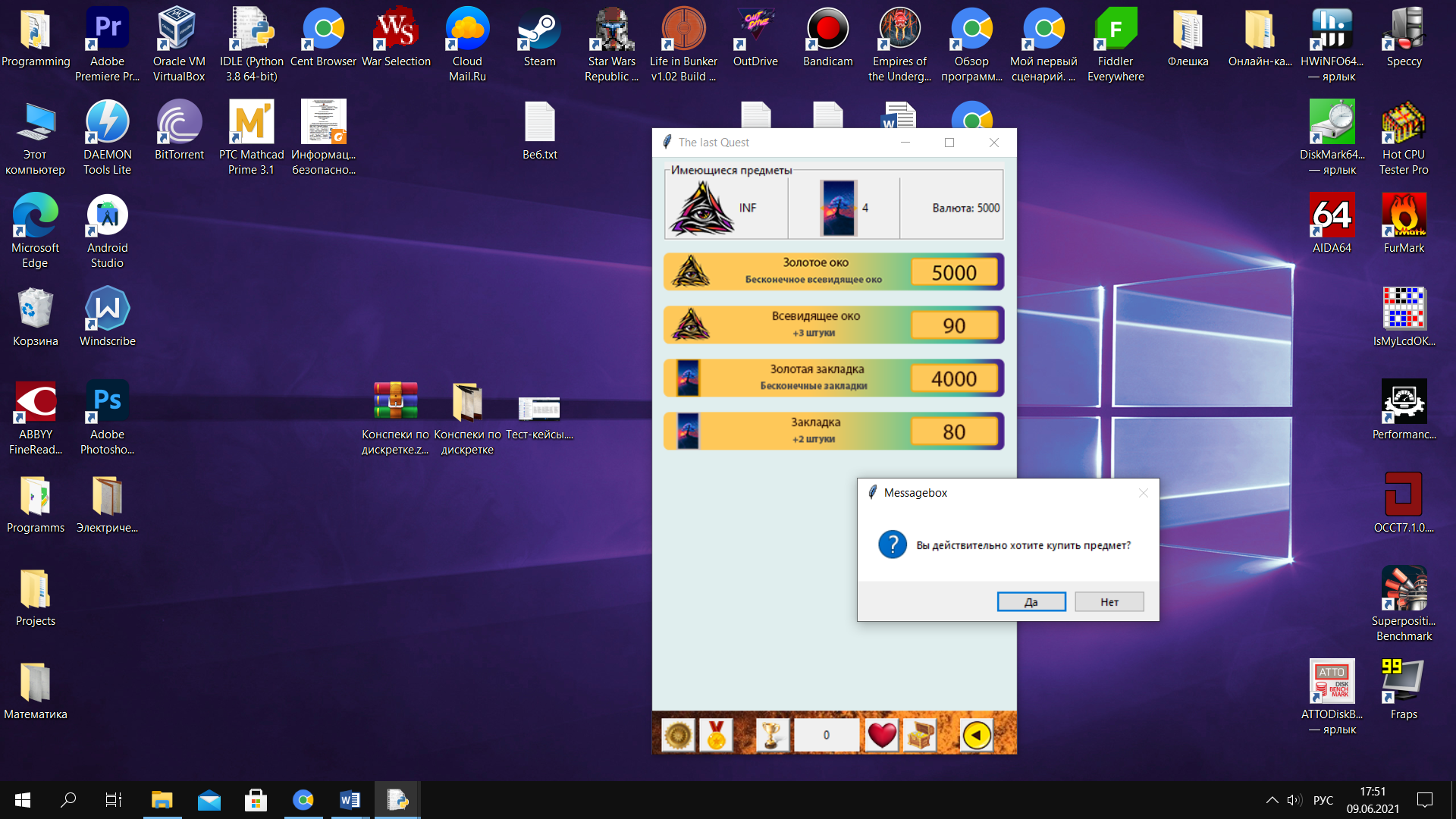


Рисунок 2.7. Покупка в магазине предмета «Золотое око»

* Login\_form – модуль, позволяющий пользователю зарегистрироваться на сервере, или войти в уже созданную учётную запись. При регистрации клиент отправляет http запрос на сервер, передавая свой логин, пароль, и конфигурационный файл (при локальной игре во время регистрации все достижения будут сохранены), и получает ответ – строку с разрешением зарегистрироваться – сервер проверит, нет ли ещё одной учётной записи с таким же логином. При входе пользователь отправляет свой логин и пароль, и, если на сервере действительно есть такая учётная запись – получает конфигурационный файл, хранящийся на сервере, при этом локальное сохранение стирается. Если человек не хочет регистрироваться – он лишиться только возможности восстановления прогресса и участия в глобальном рейтинге.
* Quest\_viever – содержит в себе алгоритм чтения текстовых квестов. (более подробно о нём – в заголовке 2.2.1.) Вызывается по клику пользователя на постер одного из скаченных квестов. По ходу квеста пользователь нажимает на одну из динамически отображаемых кнопок-действий, и видит новый текстовый фрагмент, таким образом приближаясь к концу истории. Некоторые из кнопок имеют свойство «end», которое может принимать два строковых значения: «BAD» и «GOOD». Алгоритм на каждом витке рекурсии проверяет, нет ли такого свойства у кнопки. Если пользователь нажмёт на такую – то, если концовка удачная – квест завершится закрытием просмотрщика.

# ГЛАВА 3 ТЕСТИРОВАНИЕ ПРОГРАММЫ

## 3.1 Тестирование исходного кода

Проект делится на два модуля: сервер и приложение. Тестами было покрыто приложение, и окно регистрации (уже получающее ответы с сервера).

В рамках вуза было проведено два вида тестирования – чёрное (пользовательское) и белое (тестирование кода). Также было проведено дополнительное тестирование клиента двумя пользователями, не разбирающимися в программировании.

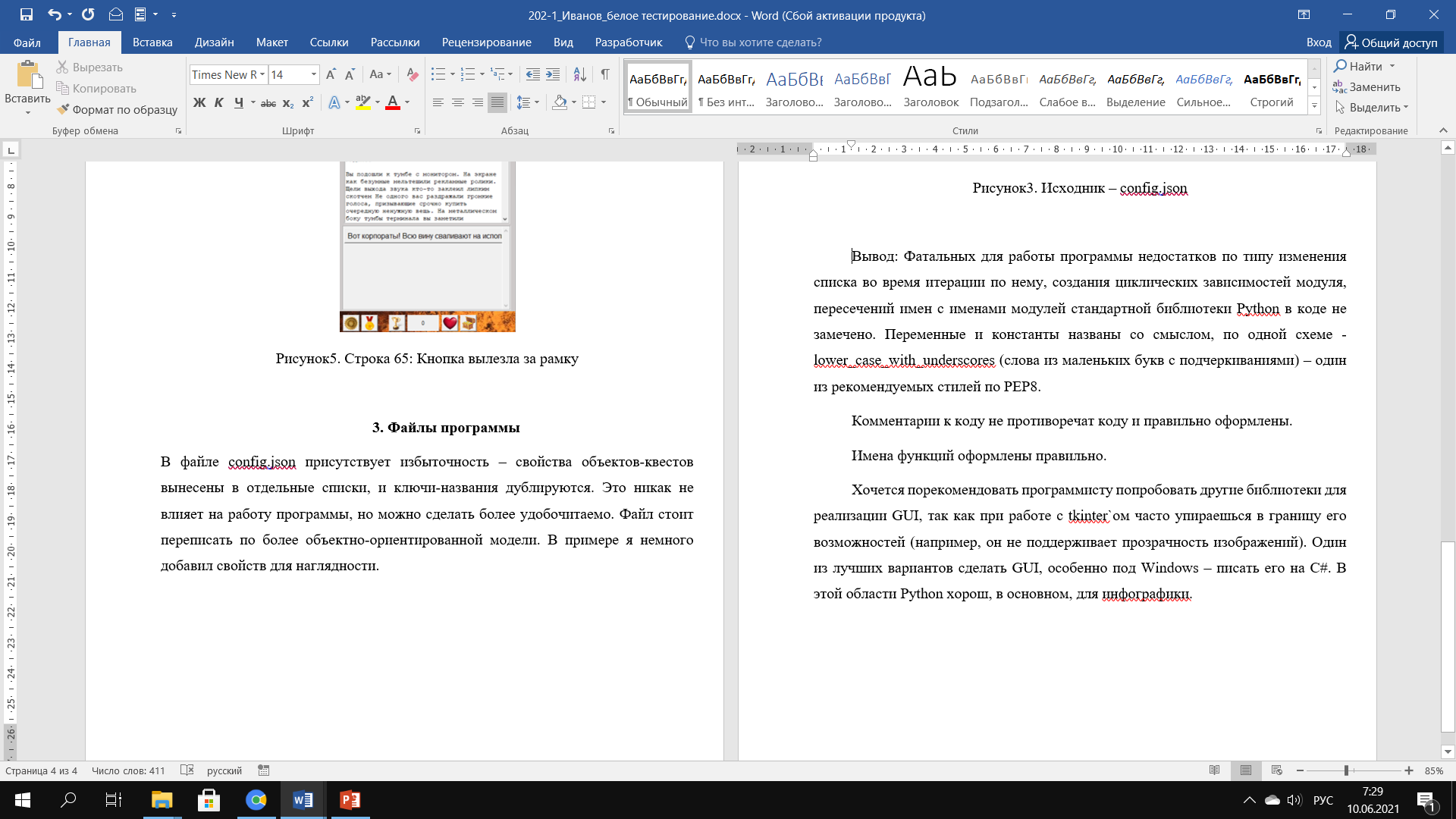


Рисунок 3.1. Вывод по чёрному тестированию

Баги, найденные при белом тестировании:

1. Функция playmusic(), включающая музыкальную тему квеста, вызывалась только после выбора следующего. Таким образом, первый квест, открывающийся при старте программы, не имел музыкального сопровождения.
2. В некоторых частях программы не были закрыты конфигурационные файлы, оставаясь висеть в памяти. При новом открытии того же файла в него не всегда записывалась информация.

## 3.2 Тестирование интерфейса пользователя и юзабилити

Баги, найденные при чёрном тестировании:

1. При чтении квеста было заметно, что некоторые из кнопок вылезали за границу рамки. Помимо этого, один из квестов, идущих с «чистым» клиентом – “Затерянный лес” – не запускался, при попытке его запустить поле с выбором квестов становилось серым, и программу приходилось перезапускать
2. Не работал скроллбар списка выбора квестов для скачивания.
3. При повторной покупке бесконечного предмета тратилась валюта.
4. При открытии формы регистрации и выполнении следующей последовательности действий: Открыть форму «Войти» –> Открыть форму «Зарегистрироваться» –> Вновь открыть форму «Войти» и ввести данные, программа вылетала с ошибкой

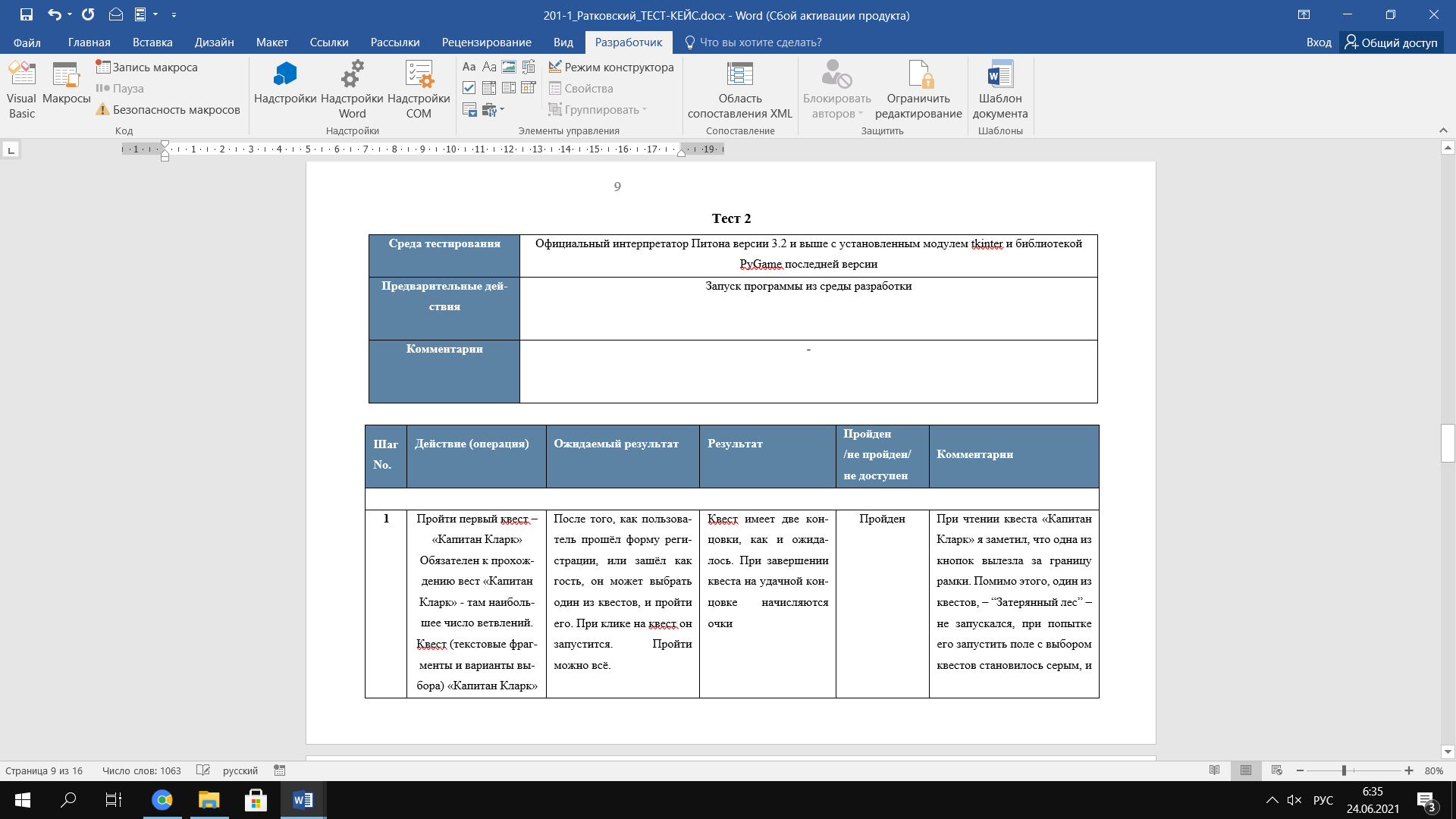


Рисунок 3.2. Тест №2 для чёрного тестирования

## 3.3 Исправление найденных багов

Действия по исправлению багов представлены в таблице 3.1.

Таблица 3.1. Отчёт по исправлению багов (номер бага формируется из номера главы, подзаголовка и порядкового номера внутри подзаголовка)

|  |  |
| --- | --- |
| Баг 3.1.1 | Функция playmusic была вызвана перед первым квсетом, чтобы и у него было музыкальное сопровождение |
| Баг 3.1.2 | При помощи функции close конфигурационные файлы, вызывающие конфликт, были закрыты |
| Баг 3.2.1 | Ошибка при запуске квеста была исправлена (в файле JSON была опечатка. В текст кнопок, выходящих за границу, были добавлены переносы строк |
| Баг 3.2.2 | Скроллбар был связан с объектом Canvas |
| Баг 3.2.3 | Были добавлены необходимые условия проверки наличия у пользователя бесконечного предмета |
| Баг 3.2.4 | Баг был оставлен – в будущем модуль будет переписан |

# ГЛАВА 4 ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА

## 4.1 Перспективы технического развития

В процессе разработки мы столкнулись с большим количеством багов, которые приходилось постоянно отлавливать. Текущую версию можно назвать стабильной, но даже мелкие баги могут испортить игровые впечатления. Один пункт технического задания, касающийся загрузки текстовых квестов с сервера, остался невыполненным – из-за переоценки программистом своих навыков.

После завершения разработки первого прототипа приложения, мы решили прекратить его поддержку на платформе Windows, начав разработку клиентского приложения под платформы с бо́льшим числом потенциальных пользователей нашего браузера текстовых квестов. При этом обязательным условием при составлении ТЗ на следующем витке проекта будет являться модернизация разработанного нами формата хранения квестов на основе JSON – чтобы оставаться конкурентоспособными, нам нужно реализовать поддержку переменных и логических условий, чтобы сделать квесты более сложными, нелинейными, организовать нечто наподобие открытого мира.

## 4.2 Перспективы монетизации

Нами была изучена информация по различным способам монетизации приложений под различные ОС, в том числе текстовых квестов.

Можно выделить 3 способа монетизации интерактивной литературы, которые подойдут для нас:

1. Показ рекламы: как за вознаграждение, так и через всплывающие баннеры;
2. Организация системы внутриигровой валюты, за которую можно получать уникальные предметы и бонусы
3. Продажа части контента: когда пользователь доходит о определённого места в квесте – разработчики предлагают ему купить оставшуюся часть.

Мы будем использовать второй способ монетизации – пользователи не будут ограничены в прохождении интерактивной литературы, но смогут получить более интересный игровой опыт.

# ЗАКЛЮЧЕНИЕ

В ходе разработки программного продукта больше времени уделялось клиенту на языке Python, из-за этого он обладает повышенными правами на изменение конфигурационных данных, что нежелательно для разработчиков, монетизирующих своё приложение – «взломать» его можно, записав при отключённом интернете необходимые значения в файл JSON. В следующих версиях клиента данные будут шифроваться, или мы используем иной формат хранения данных.

Среди причин невыполнения пункта технического задания можно выделить следующие:

* 1. Все источники информации по формату хранения данных JSON (для С++) и библиотеке httplib – англоязычные.
  2. Ограниченность проекта во времени.

Изучены основные методы разработки программного обеспечения, а также исследованы все достоинства и недостатки выбранного нами – спирального метода.

Несмотря на то, что в текущей версии программы до сих пор присутствуют незначительные баги, мы считаем проект удачно завершившимся, так как помимо задачи разработать программный продукт, мы планировали получить опыт разработки крупных проектов, укрепить навыки использования и внедрения сторонних C++ библиотек, их изучение, получить опыт разработки программ с графическим интерфейсом на языке Python.

# 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. Иконки для фотошопа [Электронный ресурс]: <http://sergeychikin.ru/365/>
4. Обработка видео в фотошопе [Электронный ресурс]: <https://www.ixbt.com/divideo/video-in-photoshop.shtml>
5. Официальный сайт с документацией Python [Электронный ресурс] <https://www.python.org/doc/>.
6. Лучано Рамальо Python. К вершинам мастерства. – М.: ДМК Пресс, 2016. – 768 с.
7. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
8. 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, 2016. – 31 с.

# ПРИЛОЖЕНИЕ 1

Функция gen\_response (регистрация)

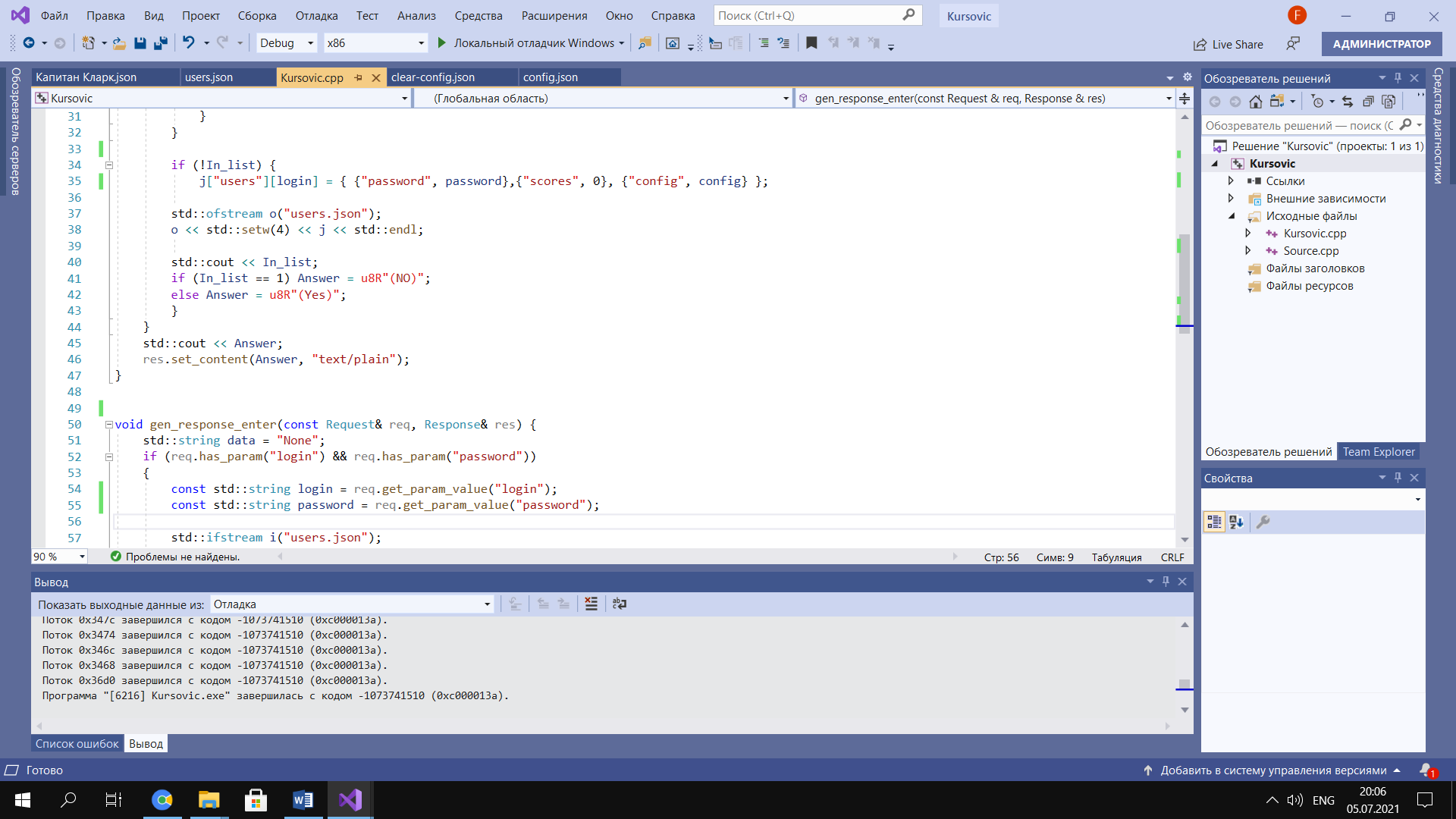
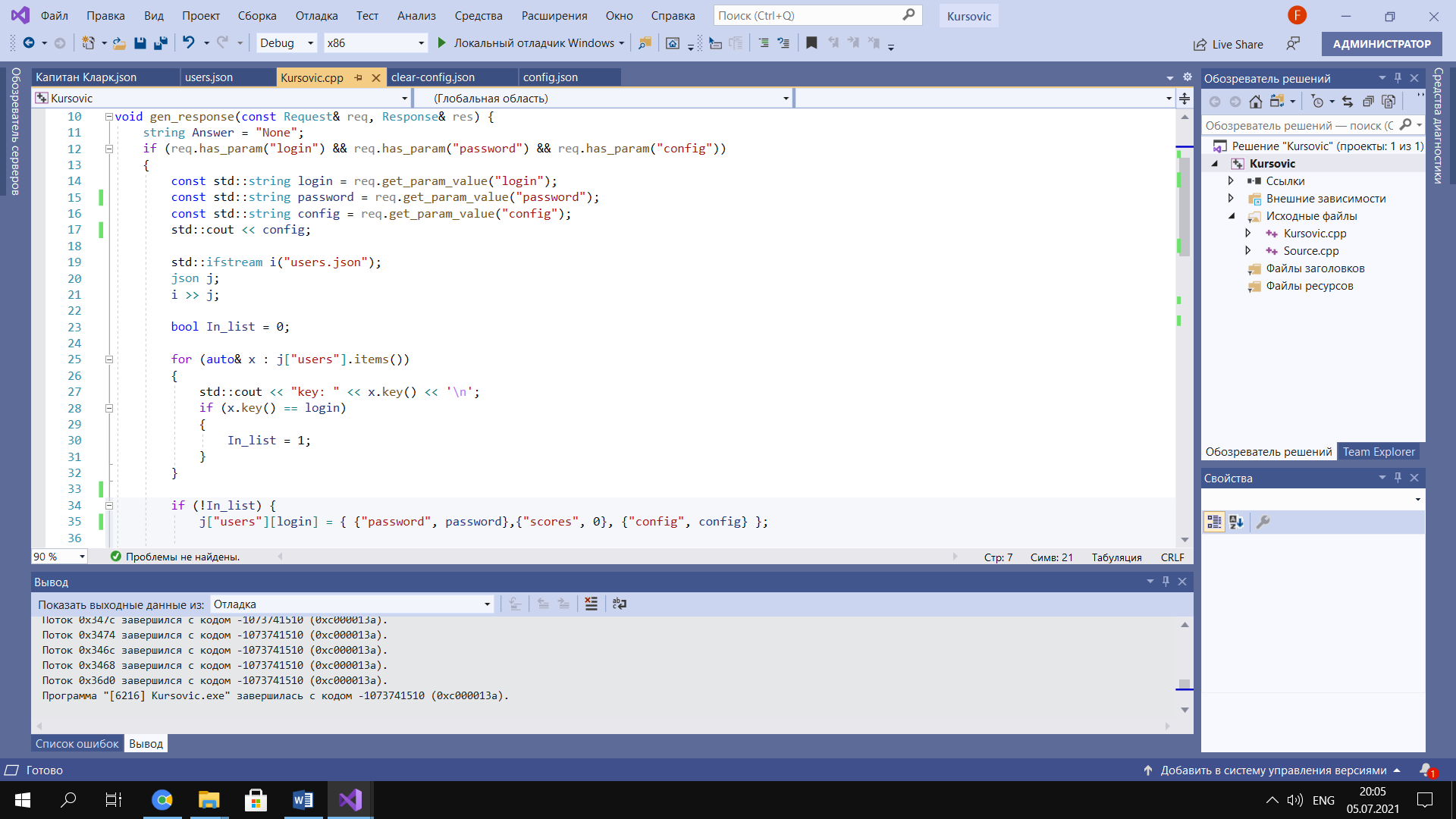


Рисунок П.1 Код функции gen\_response

Функция gen\_response\_enter (вход)

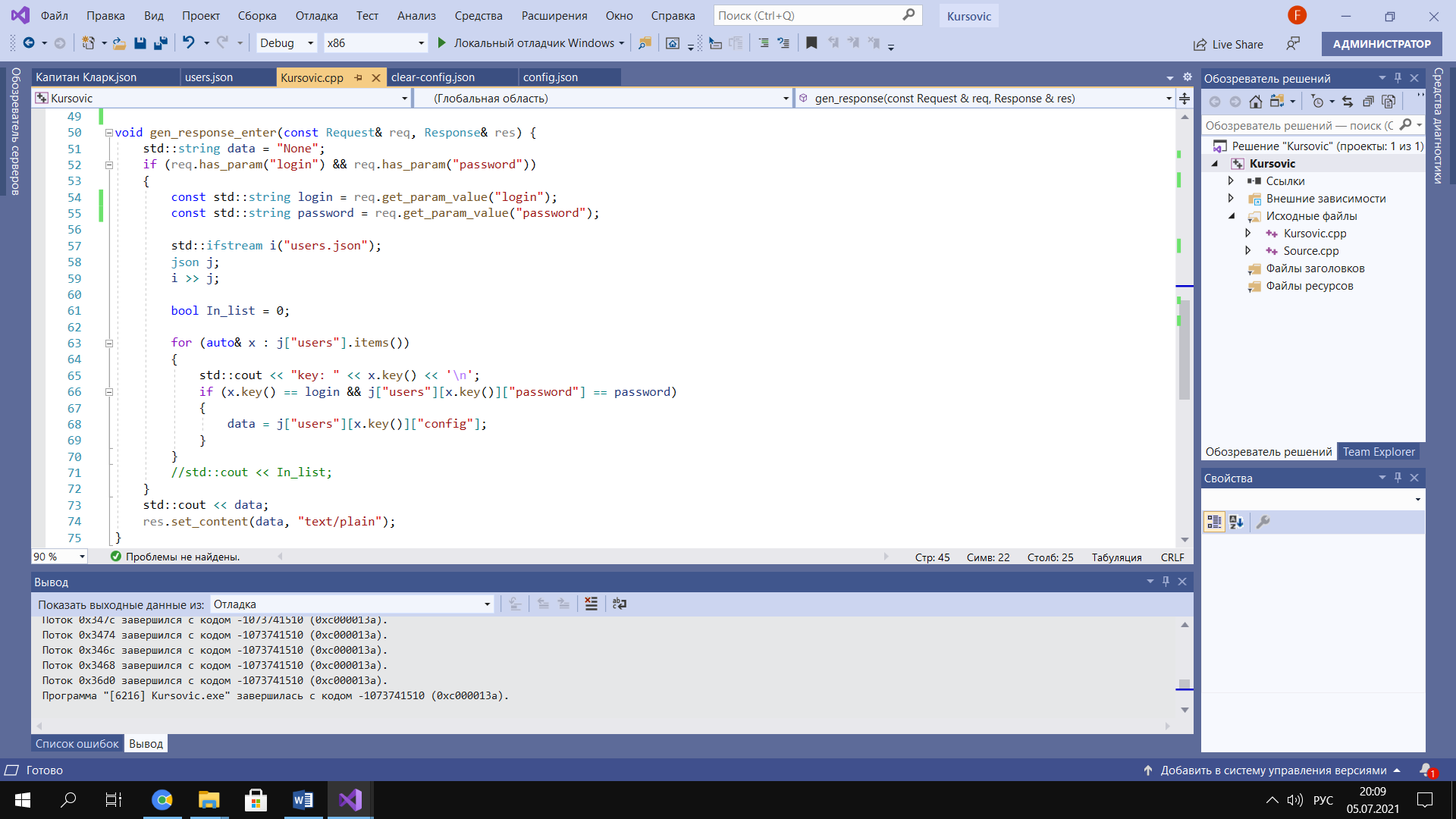


Рисунок П.2 Код функции gen\_response\_enter