Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новгородский государственный университет имени Ярослава Мудрого»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ

УТВЕРЖДАЮ
Зам. директора по УМ и ВР
_____Л.Н. Иванова
«___» _____ 2025 г.

РАЗРАБОТКА ВЕБ-СЕРВИСА ДЛЯ ТРЕНИРОВКИ СЛЕПОЙ ПЕЧАТИ В ИГРОВОЙ ФОРМЕ

Пояснительная записка к дипломному проекту по специальности
09.02.07 Информационные системы и программирование
ПТК. ДП 2994 06. 000ПЗ

Согласовано:
Консультант по спец. части
_____ Л.Н. Цымбалюк
«____» _____ 2025 года

Консультант по экон. части
_____ Г.В. Лебедева
«____» _____ 2025 года

Нормоконтроль
_____ А.М. Чернега
«____» _____ 2025 года

Заместитель директора по УПР
_____ А.М. Чернега
«____» _____ 2025 года

Руководитель
_____ В.В. Бурбах
«____» _____ 2025 года

Выполнил:
обучающийся группы 2994
_____ В.В. Иванов
«____» _____ 2025 года

Содержание

Введение 3
1 Общая часть 5
1.1 Постановка задачи 5
1.2 Обоснование проектных решений 9
1.3 Обзор и анализ существующих программных систем 14
2 Специальная часть 16
2.1 Анализ задачи 16
2.2 Описание логической структуры 19
2.3 Описание работы программы 23
2.4 Руководство оператора 31
3 Экономическая часть 39
3.1 Расчёт основной и дополнительной заработной платы с отчислениями на социальное страхование 39
3.2 Расчёт стоимости материалов и лицензионного обеспечения 42
3.3 Расчёт накладных расходов 43
3.4 Составление и расчёт цены реализации программного продукта 44
Заключение 46
Список литературы 47
Список сокращений и обозначений 49
Приложение А 50
Приложение Б 53
Приложение В 54

Введение

Освоение новых навыков – то, что помогает расти как в личном, так и в профессиональном плане. Если раньше работа с компьютером требовала специальных знаний и усилий, то сегодня использование цифровых технологий стало повседневной рутиной. Современные технологии прочно вошли в нашу жизнь, и умение эффективно ими пользоваться превратилось в обязательный навык. Одним из таких навыков является слепая печать – способность быстро набирать текст, не глядя на клавиатуру. Этот навык экономит время и улучшает сосредоточенность на содержании работы, что важно для тех, кто ежедневно работает с текстами.

Для людей, которые проводят за компьютером большую часть дня, владение слепой печатью становится практически необходимым. Быстрый набор текста без необходимости постоянно смотреть на клавиатуру помогает сосредоточиться на задачах, а не на технических деталях. Однако процесс освоения этого навыка часто оказывается скучным и однообразным. Традиционные методы, такие как повторение упражнений или использование специализированных тренажеров, не всегда способны удержать интерес пользователя. В результате большинство пользователей бросают тренировки, так и не достигнув желаемого уровня.

Главная трудность в освоении слепой печати – это монотонность, которая может быстро вызывать у ученика уныние и снижать интерес к процессу. Решением этой проблемы может стать геймификация – использование игровых приёмов в любом неигровом контексте. Геймификация способна превратить рутинные упражнения в захватывающий опыт, где каждый шаг приносит не только новые навыки, но и удовольствие.

В данной работе предлагается подход к обучению слепой печати через использование веб-сервиса, который объединяет образовательный процесс с игровыми элементами. Пользователь управляет персонажем, вводя символы на клавиатуре, и получает очки за правильные действия. Такой формат делает

тренировки вовлекающими и даёт увидеть прогресс непосредственно в процессе.

Проект рассчитан на аудиторию школьников, студентов, офисных работников и всех, кто заинтересован в повышении эффективности при работе с текстами. Для школьников и студентов тренажер может стать полезным инструментом для развития навыков, которые пригодятся в учебе. Для офисных работников освоение слепой печати позволит ускорить выполнение повседневных задач.

Цель проекта – создание интерактивного веб-сервиса для освоения слепой печати, использующего элементы геймификации для повышения мотивации пользователей.

Объект исследования – веб-сервис для тренировки слепой печати с игровыми элементами «Typing Quest».

Предмет исследования – процесс разработки эффективной системы тренировки слепой печати через игровую механику, с повышением вовлеченности пользователей в процесс освоения навыка.

Задачами данной работы, являются:

— изучить существующие методы тренировки слепой печати, преимущества и недостатки;

— разработать концепцию веб-сервиса, сочетающего обучение и игровые элементы;

— реализовать функционал тренажера с внедрением игровых элементов.

1 Общая часть

1.1 Постановка задачи

1.1.1 Обоснование необходимости разработки

В соответствии с заданием на выпускную квалификационную работу требуется разработать веб-сервис для обучения слепой печати с элементами игровой механики. Актуальность разработки обусловлена возрастающей потребностью в эффективном освоении навыка слепой печати, который позволяет:

— значительно повысить продуктивность работы с текстами;

— снизить нагрузку на зрение за счет отсутствия необходимости переключать внимание между клавиатурой и экраном;

— минимизировать количество опечаток при наборе.

Традиционные методы обучения слепой печати часто обладают недостаточной вовлекающей составляющей, что приводит к снижению мотивации пользователей. Предлагаемое решение интегрирует обучающий процесс в игровую среду, что способствует:

— повышению вовлеченности за счет визуализации прогресса;

— формированию устойчивой мотивации через систему постепенного усложнения заданий;

— созданию положительного эмоционального фона обучения.

Проект ориентирован на широкий круг пользователей, которым необходимо освоить или улучшить навык слепой печати:

— офисные сотрудники и IT-специалисты, работающие с большими объёмами текста;

— студенты и школьники, желающие повысить эффективность работы за компьютером;

— все, кто заинтересован в развитии полезного навыка для профессионального или личного использования.

Сервис рассчитан на пользователей любого возраста с базовыми навыками работы на компьютере. Игровая составляющая реализована как

дополнительный мотивационный инструмент, а не как основная функция, что отличает проект от чисто игровых продуктов.

1.1.2 Технико-математическое описание задачи

В основе проекта лежат следующие математические методы, алгоритмы и технологии:

— математические методы:

1) средняя скорость печати рассчитывается путем деления общего количества введенных символов на общее время прохождения уровня в минутах;

2) точность печати определяется как отношение правильно введённых символов к общему количеству нажатий, выраженное в процентах;

3) количество очков за уровень зависит от скорости печати, точности и времени прохождения: чем выше скорость и точность, тем больше очков. Дополнительные баллы начисляются за быстрое выполнение, а ошибки снижают итоговый результат.

— алгоритмы:

1) сравнение введённого символа с ожидаемым;

2) подсчёт статистики;

3) перемещение персонажа на определенную дистанцию при верном вводе;

4) зависимость скорости перемещения персонажа от скорости набора текста.

— технологии разработки:

1) протокол HTTP* для клиент-серверного взаимодействия;

2) язык программирования TypeScript для строгой типизации кода [3];

3) реляционная система управления базой данных PostgreSQL для хранения данных;

4) REST API архитектура серверной части.

---

* Предоставлено в списке сокращений и обозначений

1.1.3 Характеристика бизнес-процессов

Проект представляет собой игровой тренажёр слепой печати, сочетающий:

— обучающую функцию – тренировка скоростной печати;

— игровую механику – уровни, бои, достижения;

— соревновательный аспект – таблицы лидеров, статистика.

Группы пользователей и функционал:

— неавторизованные пользователи:

1) просмотр информационной страницы о проекте;
2) регистрация в системе;
3) авторизация в системе.

— авторизованные пользователи:

1) полноценное прохождение уровней;
2) просмотр личной статистики;
3) участие в рейтинговой системе;
4) выход из аккаунта;
5) получение достижений.

Важным элементом является мотивационный процесс, включающий систему достижений за выполнение определенных нормативов, обновляемую таблицу лидеров, где пользователи могут видеть свои позиции относительно других участников, а также возможность сравнивать свои текущие результаты с предыдущими попытками.

Контекстная диаграмма и диаграмма декомпозиции IDEF0, демонстрирующие основной процесс использования сервиса, представлены на Рисунке 1 и Рисунке 2.

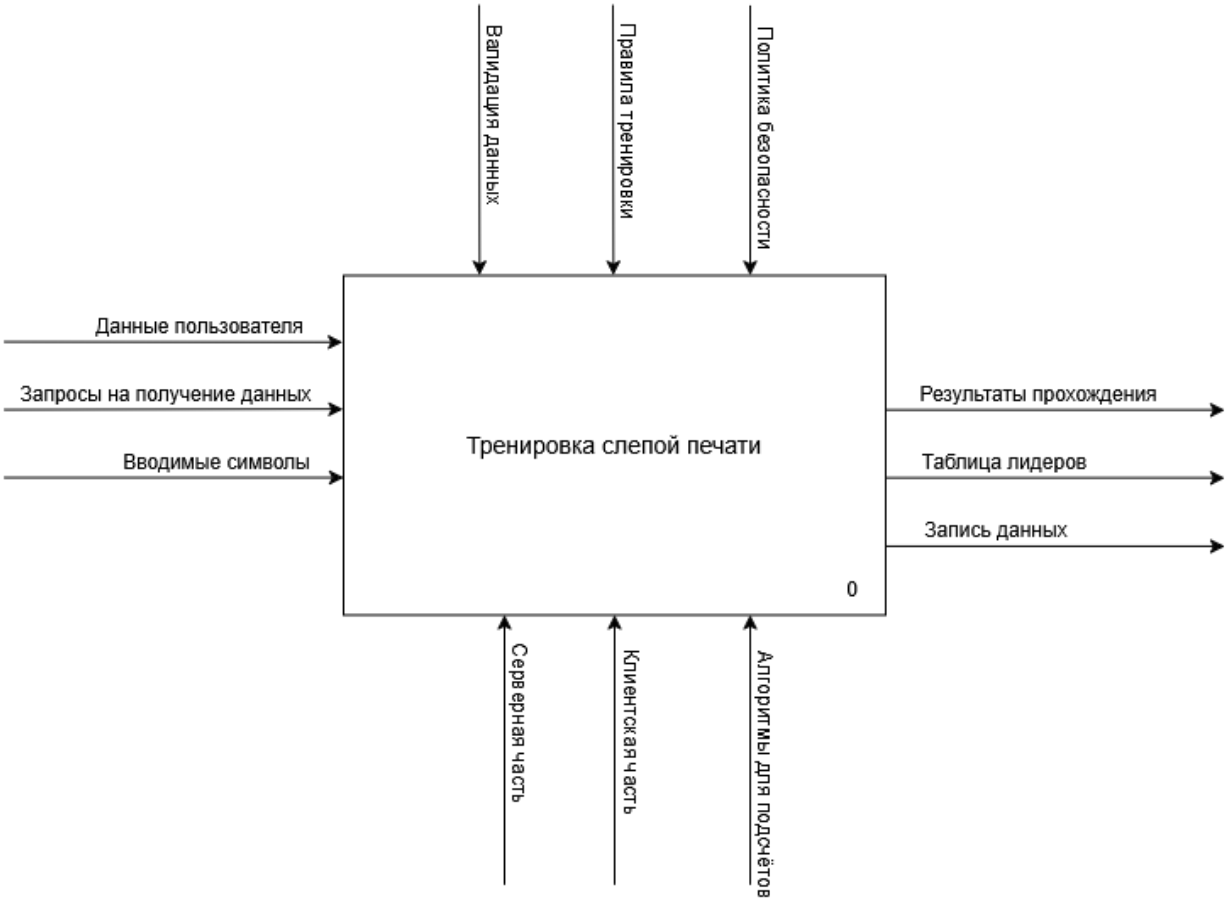

Рисунок 1 – Контекстная диаграмма IDEF0

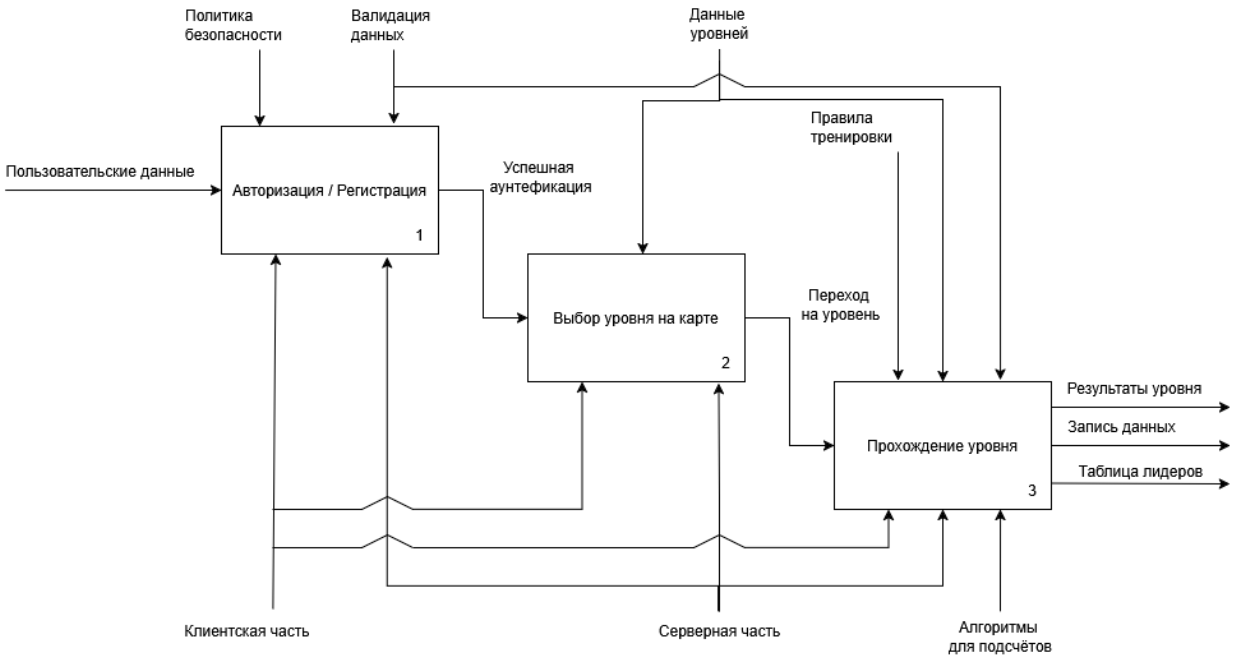

Рисунок 2 – Диаграмма декомпозиции IDEF0

1.1.4 Требования к программе

Для клиентской части веб-интерфейса необходимо наличие современного веб-браузера последних версий, включая Chrome, Firefox, Edge или Safari. Минимальное разрешение экрана должно составлять 1280 на 720 пикселей. Для стабильной работы требуется интернет-соединение со скоростью не менее 5 Мбит/с.

Серверная часть требует процессора с двумя или более ядрами и тактовой частотой от 2.4 ГГц. Минимальный объем оперативной памяти составляет 2 ГБ, а дискового пространства – 20 ГБ, при этом рекомендуется использовать SSD-накопители. Сетевой интерфейс должен поддерживать скорость передачи данных не менее 100 Мбит/с.

Клиентская часть требует поддержки JavaScript стандарта ES6 и выше, а также разрешения на использование cookies в браузере.

Для серверной части необходима операционная система Linux Ubuntu. Система использует систему управления базой данных PostgreSQL, среду выполнения Node.js и веб-сервер Nginx.

Клиентская часть разработан на фреймворке Next.js с использованием игрового веб-движка Phaser. Серверная часть реализована на NestJS, для работы с базой данных применяется TypeORM [8].

Приложение представляет собой веб-решение, не требующее установки на клиентские устройства. Доступ осуществляется через стандартные веб-браузеры.

Обязательно наличие SSL-сертификата для обеспечения безопасного HTTPS-соединения.

1.2 Обоснование проектных решений

1.2.1 Обоснование выбора языков программирования

Основным языком разработки проекта выбран TypeScript, который сочетает преимущества JavaScript с дополнительной надежностью за счет строгой типизации. Этот выбор особенно важен для тренажера слепой печати,

где необходимо точно обрабатывать пользовательский ввод и взаимодействовать с различными компонентами системы. Статическая типизация позволяет на этапе компиляции обнаруживать потенциальные ошибки взаимодействия с DOM-элементами и обработки пользовательского ввода, что критически важно для системы обучения слепой печати [5].

HTML выбран в качестве основы для построения пользовательского интерфейса благодаря своей универсальности и широкой поддержке всеми современными веб-браузерами. Этот язык разметки обеспечивает четкую структуру и логическую организацию контента на страницах тренажера, включая интерфейс статистики, профиля пользователя и информационные разделы. HTML позволяет корректно отображать текстовое содержимое, формы ввода данных и другие элементы интерфейса.

Для стилизации интерфейса используется CSS через фреймворк TailwindCSS, который был выбран за свою практичность и эффективность. TailwindCSS предоставляет готовые утилитарные классы, позволяя быстро создавать адаптивные интерфейсы без необходимости написания собственных стилей. Это значительно ускоряет процесс разработки и обеспечивает единообразие визуального оформления всех компонентов системы. Особенно важно, что TailwindCSS хорошо интегрируется с компонентным подходом, используемым в Next.js, сохраняя при этом высокую производительность.

Такой подход к выбору языков программирования и разметки позволяет создать устойчивую основу для дальнейшего развития тренажера, сохраняя баланс между производительностью и удобством разработки.

1.2.2 Инструментальные средства

Для сборки проекта и управления зависимостями применяется npm – Node Package Manager. Этот инструмент позволяет эффективно работать с более чем 40 зависимостями проекта, включая основные фреймворки и вспомогательные библиотеки. npm обеспечивает выполнение скриптов сборки, тестирования и развертывания.

Next.js выступает основным фреймворком для фронтенд-разработки, предоставляя серверный рендеринг, статическую генерацию страниц и

встроенную маршрутизацию. Для серверной части выбран NestJS, который предлагает модульную архитектуру, систему внедрения зависимостей и готовые решения для построения REST API. Игровая механика реализована с помощью Phaser – специализированного движка для создания браузерных игр. Стилизация интерфейсов выполнена с использованием TailwindCSS, что позволяет быстро создавать адаптивные компоненты с минимальным объемом кастомного CSS [13].

Тестирование API осуществляется с помощью Postman – мощного инструмента для работы с HTTP-запросами. Postman позволяет не только отправлять тестовые запросы и проверять ответы сервера, но и сохранять коллекции запросов, автоматизировать рутинные проверки и документировать endpoints API. Это значительно упрощает процесс отладки и валидации серверной части приложения.

Проект предусматривает возможность развертывания с использованием Docker и Nginx. Docker обеспечивает контейнеризацию приложения, создавая изолированную среду со всеми необходимыми зависимостями. Nginx может быть использован в качестве обратного прокси и веб-сервера, оптимизируя обработку запросов и обслуживание статических ресурсов.

Выбранный набор инструментов обеспечивает комплексный подход к разработке, охватывая все этапы – от проектирования до развертывания. Каждый инструмент был отобран с учетом его соответствия требованиям проекта, производительности и удобства использования, что в совокупности создает комфортные условия для разработки веб-сервиса.

1.2.3 Обоснование выбора среды программирования

В качестве основной среды разработки для проекта был выбран Visual Studio Code – современный редактор кода с широкими возможностями настройки и богатой экосистемой расширений. Этот выбор обусловлен рядом ключевых преимуществ, которые особенно важны для разработки веб-приложений.

VS Code обеспечивает превосходную поддержку TypeScript – основного языка разработки в проекте. Редактор предоставляет автодополнение кода,

навигацию по типам, мгновенную проверку ошибок и рефакторинг, что значительно ускоряет процесс написания кода. Для Next.js и NestJS доступны специализированные расширения, упрощающие работу с этими фреймворками.

Редактор имеет встроенную интеграцию с Git, что позволяет выполнять основные операции с репозиторием прямо из интерфейса. Для работы с Docker доступны удобные инструменты управления контейнерами. Поддержка ESLint и Prettier обеспечивает соблюдение стиля кодирования и выявление потенциальных проблем.

VS Code отличается высокой производительностью даже при работе с крупными проектами. Редактор потребляет умеренное количество системных ресурсов, что важно для комфортной разработки. Возможность тонкой настройки интерфейса и горячих клавиш позволяет адаптировать среду под индивидуальные предпочтения.

VS Code доступен для всех основных операционных систем, что обеспечивает согласованность среды разработки независимо от используемой платформы.

Встроенный отладчик поддерживает как клиентский, так и серверный код. Для тестирования API через Postman доступно специальное расширение, позволяющее отправлять запросы и просматривать ответы прямо из редактора.

Богатая библиотека расширений позволяет добавлять в редактор только необходимые функции, избегая избыточности.

1.2.4 Информационное обеспечение

В проекте активно используется система Git в сочетании с платформой GitHub. Git обеспечивает надежный контроль версий, позволяя фиксировать изменения на разных этапах разработки. Это дает возможность отслеживать историю модификаций, возвращаться к предыдущим состояниям проекта и создавать изолированные ветки для реализации новых функций. GitHub служит облачным хранилищем для репозитория, предоставляя удобный интерфейс для управления кодом и дополнительными возможностями [6].

В качестве основного хранилища данных применяется реляционная система управления базой данных PostgreSQL. Эта система обеспечивает надежное хранение информации проекта, включая пользовательские данные и статистику тренировок. Для администрирования базы данных и выполнения запросов используется графический интерфейс pgAdmin, который предоставляет удобные инструменты для работы со структурой базы, выполнения SQL-запросов и анализа производительности [4].

Для работы с документацией и сопроводительными материалами использовались облачные офисные решения Google Workspace. Google Docs применялся для создания и совместного редактирования технической документации, обеспечивая удобный доступ к актуальным версиям документов с любого устройства. Для подготовки презентационных материалов и демонстрации возможностей системы применялся Google Slides, позволяющий создавать интерактивные презентации с возможностью онлайн-демонстрации.

В процессе разработки игровых элементов проекта использовались специализированные инструменты для работы с графикой. TexturePacker применялся для создания оптимальных текстурных атласов из отдельных графических ресурсов. Этот инструмент позволил значительно улучшить производительность рендеринга игровых сцен за счет:

— объединения множества отдельных изображений в единые текстуры;

— автоматической оптимизации пространства текстур;

— генерации соответствующих JSON-файлов с координатами спрайтов.

Tiled Map Editor использовался для создания и редактирования игровых карт и уровней. Этот редактор предоставил удобный визуальный интерфейс для [14]:

— проектирования структуры игровых уровней;

— размещения объектов и декоративных элементов;

— настройки слоев;

— экспорта карт в формате JSON для последующего использования в Phaser.

1.3 Обзор и анализ существующих программных систем

Stamina:

— достоинства:

1) сервис представляет собой классический тренажер слепой печати, доступный как в виде отдельной программы, так и в онлайн-версии;

2) главным преимуществом является гибкость настроек – пользователь может самостоятельно регулировать сложность упражнений, выбирать тексты для тренировки и настраивать интерфейс под свои предпочтения;

3) программа отличается оригинальным юмористическим подходом, что делает процесс обучения менее напряженным;

4) наличие оффлайн-версии позволяет заниматься без подключения к интернету.

— недостатки:

1) интерфейс Stamina выглядит устаревшим и не соответствует современным стандартам UX/UI;

2) функционал ограничен базовыми упражнениями без продвинутой аналитики результатов.

Typing Study:

— достоинства:

1) сервис предлагает четко структурированную программу обучения;

2) поддерживается множество языковых раскладок, включая специализированные варианты;

3) модуль для тренировки работы с цифровой клавиатурой.

— недостатки:

1) функционал анализа результатов ограничен базовой статистикой по скорости и точности без персонализированных рекомендаций по улучшению навыков;

2) обучение построено на традиционной схеме выполнения упражнений без дополнительных мотивационных элементов.

Клавогонки:

— достоинства:

1) Клавогонки реализуют уникальный игровой подход к обучению через систему многопользовательских соревнований в реальном времени;

2) разнообразные режимы позволяют отрабатывать разные аспекты печати;

3) рейтинговая система и достижения создают сильную мотивационную составляющую.

— недостатки:

1) сервис требует стабильного интернет-соединения для полноценной работы;

2) обучающий контент представлен в минимальном объеме, основной акцент сделан на соревновательную составляющую;

3) отсутствует структурированная программа для начинающих.

2 Специальная часть

2.1 Анализ задачи

2.1.1 Информационное моделирование предметной области

На Рисунке 3 представлена реляционная модель базы данных, включающая сущности пользователей, сессии, достижения и результаты с описанием атрибутов и связей.

Рисунок 3 – Реляционная модель базы данных

Связи между сущностями:

— один пользователь может иметь множество результатов;

— один уровень может быть связан с множеством результатов;

— много пользователей могут получать много достижений. Реализовано через дополнительную таблицу user_achievements;

— один пользователь может иметь множество сессий.

Файловая структура:

— графические объекты хранятся на сервере;

— конфигурация уровней хранится в таблице levels;

— логи ошибок отправляются в файлы на сервере.

2.1.2 Проектирование пользовательского интерфейса

Стартовый экран:

— цель: первичное знакомство пользователя с сервисом, навигация к основным функциям;

— элементы: фоновое изображение в фэнтезийном стиле, крупное название «Typing Quest», центрально расположенная кнопка «Начать», второстепенная кнопка «О проекте»;

— обоснование: минималистичный дизайн без лишних элементов фокусирует внимание на основном действии. Крупная кнопка «Начать» соответствует принципам UX для целевых действий.

Стартовый экран представлен в Приложении А.1.

Экран авторизации:

— цель: авторизация существующих и регистрация новых пользователей;

— элементы: форма ввода с полями «Username/Email» и «Пароль», кнопки «Войти» или «Зарегистрироваться»;

— обоснование: простая форма без отвлекающих элементов ускоряет процесс входа. Стандартное расположение полей соответствует пользовательским ожиданиям.

Экран авторизации представлен в Приложении А.2.

Карта уровней:

— цель: визуализация доступных уровней и ознакомление с информацией об уровнях;

— элементы: стилизованная карта с маркерами уровней, всплывающая подсказка с названием при наведении, а также модальное окно с информацией об уровне при нажатии. Кнопка профиля и кнопка выхода в правом верхнем углу;

— обоснование: игровая стилизация мотивирует к прохождению уровней. Интуитивно понятные маркеры показывают название каждого уровня.

Карта уровней представлена в приложении А.3.

Игровой экран:

— цель: основной интерфейс тренировки печати;

— элементы: персонаж и противники на локации, текст для ввода внизу экрана, индикаторы здоровья и статистики ввода рядом с текстом для ввода, визуальная подсветка вводимых символов;

— обоснование: разделение экрана на игровую зону и зону ввода соответствует принципу «Один экран - одна задача».

Экран боя:

— цель: создание динамичной среды для тренировки слепой печати через игровую механику боя;

— элементы: персонаж игрока и противник, последовательность символов для атаки, индикаторы здоровья и таймер атаки, статистика ввода, визуальная подсветка вводимых символов;

обоснование: игровая механика боя мотивирует пользователя повышать скорость и точность печати, создавая эффект срочности.

Экран боя представлен в Приложении А.4.

Экран результатов:

— цель: отображение статистики после завершения уровня;

— элементы: крупные цифры с основными метриками (скорость, точность), таблица рекордов, а также график средней скорости печати, кнопка «Вернуться на карту»;

— обоснование: акцент на ключевых показателях помогает пользователю оценить прогресс. Крупные кнопки упрощают навигацию.

Экран результатов представлен в Приложении А.5.

Экран профиля:

— цель: отображение персональной статистики и достижений;

— элементы: блок с основной статистикой, список результатов уровней и список полученных достижений;

— обоснование: оценка результатов прошлых уровней помогает оценить прогресс. Список достижений реализует игровую механику коллекционирования.

Экран профиля представлен в Приложении А.6.

2.2 Описание логической структуры

2.2.1 Алгоритм программы

Основная механика программы связывает ввод текста пользователем с игровым процессом: каждый правильно введенный символ продвигает персонажа вперед, а в режиме боя позволяет наносить урон противникам. Алгоритм работы программы можно разделить на следующие этапы.

Загрузка и инициализация:

— инициализация сцены Level:

1) загрузка конфигурации уровня и токена доступа пользователя;

2) создание игрового мира, включая фон, тайловую карту и коллизии;

3) инициализация систем: ScoreManager, BattleSystem, InputSystem, UISystem.

Навигация по уровням:

— создание карты уровня:

1) загрузка тайловой карты, масштабирование под разрешение экрана;

2) создание слоев для фона, коллизий и специальных зон;

3) инициализация физического движка с границами уровня.

Игровой процесс:

— обработка ввода:

1) генерация последовательности символов для ввода на основе доступных символов уровня;

2) сравнение ввода пользователя с ожидаемым символом, обновление интерфейса;

3) перемещение персонажа на заданное расстояние при правильном вводе.

— движение персонажа:

1) физическое взаимодействие с окружением;

2) автоматические прыжки при пересечении специальных зон.

— система боя:

1) проверка дистанции до врагов для начала боя;

2) генерация последовательности символов для атаки;

3) обработка ввода в режиме боя, нанесение урона противнику при правильном вводе;

4) таймер атаки, отслеживающий время для ввода последовательности.

— управление врагами:

1) создание врагов на основе данных из тайловой карты;

2) управление поведением врагов (перемещение, атака);

3) обработка столкновений с окружением и персонажем.

Системы интерфейса:

— отображение статистики:

1) точность ввода, количество правильных/неправильных символов;

2) динамическое обновление данных.

— элементы управления:

1) кнопка паузы с вызовом модального окна;

2) отображение результатов уровня с таблицей лидеров.

Завершение уровня:

— проверка пересечения с финишной зоной:

1) сбор статистики: скорость печати, точность, время прохождения;

2) расчет итогового счета;

3) отправка результатов на сервер;

4) показ окна результатов с возможностью возврата на карту уровней.

Особенности реализации:

— интеграция React и Phaser:

1) использование React-компонентов для модальных окон поверх игрового холста;

2) динамическое создание и управление DOM-элементами интерфейса.

— оптимизация:

1) ленивая загрузка ресурсов уровней;

2) очистка объектов и систем при завершении уровня.

— сетевое взаимодействие:

1) сохранение данных через REST API;

2) Загрузка таблицы лидеров для текущего уровня.

2.2.2 Используемые методы

Разработка базы данных:

— проектирование структуры: создана нормализованная реляционная схема с таблицами users, levels, results, achievements. Реализованы связи 1:N и M:N через внешние ключи.

Методы защиты данных:

— аутентификация: refresh токены с httpOnly флагом и сроком жизни 30 дней, а также access токены со сроком жизни в 15 мин.;

— хеширование паролей;

— защита API: валидация входящих данных через class-validator.

Готовые методы NestJS:

— Встроенные механизмы Phaser:

1) Physics.Arcade для коллизий;

2) KeyboardEvent для обработки ввода.

— NestJS-модули:

1) ConfigService для управления переменными окружения;

2) TypeOrmModule для работы с PostgreSQL.

Собственный метод обработки ввода:

— комбинирует нативные события клавиатуры с программной эмуляцией для плавного игрового процесса. Основной метод handleInput проверяет совпадение введенного символа с текущим в последовательности, при успехе вызывает processCorrectInput, который обновляет интерфейс и двигает персонажа;

— особенности реализации:

1) работает с двумя типами событий: реальные (KeyboardEvent) и сгенерированные ({ key: string });

2) синхронизирует ввод с движением персонажа через параметр distancePerKey.

— преимущества решения:

1) обеспечивает плавный геймплей с эффектом «бегущей строки»;

2) повышает отзывчивость интерфейса за счет единого обработчика событий.

2.2.3 Составные части программы и связи между ними

Основные модули системы:

— клиентская часть:

1) Boot: инициализация ядра игры;
2) Preloader: загрузка ресурсов;
3) Intro: стартовый экран с меню;
4) Map: карта уровней;
5) Level: основная игровая сцена;
6) Character: управление персонажем;
7) EnemyManager: контроль врагов.

— Серверная часть:

1) AuthModule: аутентификация;
2) LevelsModule: аправление уровнями;
3) ResultsModule: обработка статистики;
4) AchievementsModule: система достижений.

Ключевые связи:

— Phaser-сцены: линейный переход между состояниями игры через this.scene.start();

— данные уровней: клиент запрашивает конфигурацию через LevelsModule (REST API);

— статистика: отправка результатов после завершения уровня в ResultsModule;

— персонаж: взаимодействует с Physics и EnemyManager.

2.3 Описание работы программы

2.3.1 Общие сведения

Программа под названием Typing Quest представляет собой игровой тренажёр для обучения слепой печати, реализованный в формате веб-приложения. Основная концепция проекта заключается в совмещении обучающей функции по развитию навыков быстрого и точного набора текста с увлекательным игровым процессом, где пользователь управляет рыцарем, продвигающимся по уровням. Особенностью программы является прямая зависимость игрового процесса от скорости и точности ввода текста: персонаж движется быстрее при корректном наборе символов, а при встрече с противниками игроку необходимо вводить определённые последовательности символов в ограниченное время для нанесения ударов. В случае замедления или допущения ошибок персонаж получает урон, что добавляет игровому процессу элемент напряжённости и мотивирует пользователя совершенствовать свои навыки.

Каждый уровень имеет чётко обозначенный финиш, при достижении которого система анализирует результаты прохождения: подсчитывает количество набранных символов, определяет скорость ввода (в символах в минуту), фиксирует допущенные ошибки и на основе этих данных выставляет итоговые очки. Особое внимание уделено системе поощрений за быстрое прохождение – если игрок завершает уровень раньше контрольного времени,

Пользователь получает дополнительные баллы, пропорциональные сэкономленным секундам. Это создаёт дополнительный стимул для улучшения скорости печати.

Важной составляющей программы является система статистики и достижений. В личном профиле пользователь может отслеживать свой прогресс: общее количество введённых символов, среднюю скорость печати, процент точности и лучшие результаты по уровням. Игровые достижения, такие как «Завершите первый уровень» или «Наберите 100% точности», служат дополнительными мотивационными элементами. Для создания соревновательной атмосферы реализована таблица лидеров, позволяющая сравнивать свои результаты с достижениями других игроков.

Проект ориентирован на широкую целевую аудиторию, включающую школьников, студентов и офисных работников, для которых навык слепой печати является профессионально значимым. Техническая реализация в виде веб-приложения обеспечивает доступность без необходимости установки дополнительного программного обеспечения, а адаптивная система сложности, основанная на прогрессе пользователя, позволяет постепенно осваивать технику быстрой печати. Визуальные элементы, такие как анимации боев и подсветка вводимых символов, обеспечивают интуитивно понятную обратную связь.

2.3.2 Функциональное назначение

Основное функциональное назначение системы заключается в эффективном обучении пользователей технике быстрого и точного набора текста без необходимости визуального контроля клавиатуры, посредством применения методов геймификации для повышения мотивации и вовлечённости в учебный процесс.

Ключевые функциональные возможности системы включают:

— обучение слепой печати через интерактивный игровой процесс, где скорость и точность ввода текста напрямую влияют на продвижение игрового персонажа (рыцаря) по уровням;

— игровая механика состоит из:

1) управление персонажем, скорость движения которого зависит от скорости и точности ввода текста;

2) систему боевых взаимодействий, где успешность атаки определяется способностью пользователя быстро и правильно вводить заданные последовательности символов;

3) прогрессивную систему уровней сложности, адаптирующуюся под навыки игрока.

— систему оценки и анализа результатов, предоставляющую пользователю детальную статистику по:

1) количеству введённых символов;

2) скорости печати (в символах в минуту);

3) проценту допущенных ошибок;

4) времени прохождения уровней.

— мотивационные механизмы, включающие:

1) систему начисления очков за успешное выполнение заданий;

2) бонусы за быстрое прохождение уровней;

3) таблицу лидеров для сравнения результатов с другими игроками;

4) систему достижений за выполнение определённых условий.

— персонализированный профиль пользователя, позволяющий:

1) отслеживать индивидуальный прогресс в обучении;

2) анализировать историю результатов.

Функциональное назначение программы также включает создание комфортной обучающей среды за счёт:

— интуитивно понятного интерфейса;

— визуальной обратной связи;

— анимационных эффектов, усиливающих восприятие игрового процесса.

2.3.3 Связи с другими программами

Веб-сервис Typing Quest является самостоятельным приложением и не требует взаимодействия с внешними программами для своего основного функционала.

2.3.4 Входные данные

Программа Typing Quest обрабатывает следующие категории входных данных:

— данные аутентификации пользователя:

1) учетные данные при входе (логин/email и пароль);

2) регистрационные данные нового пользователя (имя, email, пароль и его подтверждение).

— игровые данные в реальном времени:

1) нажатия клавиш клавиатуры (символьные и служебные);

2) временные параметры ввода (точное время нажатия и отпускания клавиш);

3) последовательности вводимых символов во время боевых сцен.

— данные игрового процесса:

1) состояние персонажа (текущее здоровье, позиция на уровне);

2) параметры противников;

3) прогресс прохождения уровня (пройденное расстояние, оставшееся время).

— системные данные:

1) настройки интерфейса (язык, размер шрифта);

2) аудиовизуальные параметры;

3) данные синхронизации с сервером.

— статистические данные:

1) результаты завершенных уровней (точность, скорость печати);

2) количество ошибок и распределение ошибок по клавишам;

3) время прохождения каждого уровня.

— особенности обработки входных данных:

1) все символьные данные обрабатываются с учетом регистра;

2) временные параметры фиксируются с миллисекундной точностью;

3) игровые события регистрируются в хронологическом порядке;

4) вводимые символы проверяются на соответствие ожидаемым значениям.

— система обеспечивает:

1) валидацию входных данных на корректность;

2) защиту от некорректного ввода;

3) логирование критических событий;

4) формирование статистики на основе входных данных.

— все входные данные используются для:

1) управления игровым процессом в реальном времени;

2) оценки навыков пользователя;

3) формирования игровой статистики;

4) обновления рейтинговых таблиц.

2.3.5 Выходные данные

Программа Typing Quest формирует три основных типа выходных данных, которые сохраняются как на клиентской стороне, так и передаются на сервер для дальнейшей обработки.

Статистика прохождения уровней. После завершения каждого уровня система генерирует детальный отчет, включающий: скорость набора, процент точности, количество допущенных ошибок, общее время прохождения. Эти данные сохраняются в базе данных PostgreSQL в таблице результатов и используются для формирования персональной статистики пользователя.

Достижения и прогресс. При выполнении определенных условий (например, достижение целевой скорости или прохождение уровня без ошибок) система присваивает пользователю соответствующие достижения. Информация о полученных достижениях сохраняется в таблице и отображается в профиле игрока.

Таблицы лидеров. На основе собранной статистики система формирует рейтинговые таблицы для каждого уровня и общий рейтинг игроков. Данные обновляются в реальном времени и включают: позицию в рейтинге, имя пользователя, лучший результат по скорости и точности.

Все выходные данные доступны пользователю через интерфейс профиля, где они визуализированы в виде таблиц. Данные передаются в зашифрованном виде через HTTPS-соединение, что обеспечивает конфиденциальность и целостность.

2.3.6 Тестирование

Для проверки корректности работы веб-сервиса были разработаны тест-кейсы, охватывающие ключевые сценарии взаимодействия пользователя с системой. Результаты тестирования представлены в следующих таблицах:

Таблица 1 – Тест-кейс на проверку регистрации пользователя

Поле	Описание
Название тест-кейса	Проверка регистрации пользователя.
Краткое изложение тест-кейса	Проверка на успешную регистрацию пользователя на платформе.
Алгоритм тестирования	Нажатие на кнопку «Начать». Ввод данные в форму регистрации. Перенаправление пользователя на карту уровней
Тестовые данные	Имя пользователя: Иван, email: ivan@mail.ru, пароль: 12345678.
Ожидаемый результат	Успешная регистрация пользователя.
Фактический результат	Успешная регистрация пользователя в системе.
Статус тест-кейса	Зачет.
Предварительное условие	Отсутствие зарегистрированного аккаунта на вводимую почту.

Окончание таблицы 1

Поле	Описание
Постусловие	Переход на карту уровней.

Таблица 2 – Тест-кейс на проверку перехода на уровень

Поле	Описание
Название тест-кейса	Проверка перехода на уровень.
Краткое изложение тест-кейса	Проверка на успешный переход с общей карты на игровой уровень.
Алгоритм тестирования	Нажатие на маркер уровня. Нажатие на кнопку «Начать уровень». Запуск загрузки уровня.
Тестовые данные	Отсутствуют.
Ожидаемый результат	Успешный переход на уровень.
Фактический результат	Успешное перенаправление пользователя на игровой уровень.
Статус тест-кейса	Зачет.
Предварительное условие	Авторизованный пользователь, отображение уровней на карте.
Постусловие	Перенаправление на уровень.

Таблица 3 – Тест-кейс на просмотр статистики в профиле

Поле	Описание
Название тест-кейса	Проверка просмотра пользователем статистики в профиле.
Краткое изложение тест-кейса	Проверка на успешный просмотр статистики в профиле пользователя.
Алгоритм тестирования	Нажатие на кнопку «Профиль». Загрузка профиля со статистикой.

Окончание таблицы 3

Поле	Описание
Тестовые данные	Отсутствуют.
Ожидаемый результат	Успешное отображение данных статистики в профиле.
Фактический результат	Данные были выгружены при открытии профиля.
Статус тест-кейса	Зачет.
Предварительное условие	Авторизованный пользователь, отображение карты уровней.
Постусловие	Открытие профиля пользователя с загруженными данными.

Тестирование с помощью Unit-тестов представлено в Приложении В.1.

2.3.7 Вызов и загрузка

Процесс запуска веб-сервиса:

— инициализация клиентской части:

1) пользователь вводит URL сервиса в адресную строку браузера;
2) браузер отправляет HTTP-запрос к серверу приложения;
3) сервер возвращает HTML-страницу с базовой структурой и ссылками на ресурсы.

— загрузка основных ресурсов:

1) CSS-стили интерфейса;
2) JavaScript-код приложения;
3) графические ресурсы и шрифты;
4) игровые объекты.

— инициализация игрового движка:

1) создается основной холст для рендеринга;
2) инициализируются подсистемы ввода и обработки событий;
3) загружаются конфигурационные данные уровней.

— проверка аутентификации:

1) приложение проверяет наличие токена доступа;
2) при отсутствии - перенаправляет на стартовый экран;
3) при наличии - загружает данные пользователя.

— отображение интерфейса в зависимости от состояния:

1) стартовый экран (для новых пользователей);
2) карта уровней (для авторизованных);
3) игровая сцена (при прямом переходе на уровень).

Особенности загрузки:

— используется прогрессивная загрузка ресурсов;

— критически важные компоненты загружаются в первую очередь;

— дополнительные объекты подгружаются по мере необходимости;

— реализован механизм предзагрузки для часто используемых ресурсов.

Время загрузки:

— основной интерфейс – до трёх секунд;

— полная загрузка игровых уровней – до семи секунд.

2.4 Руководство оператора

2.4.1 Назначение программы

Основная задача программы – помочь пользователям улучшить навыки работы с клавиатурой через систему последовательных упражнений, представленных в виде уровней с возрастающей сложностью.

Программа ориентирована на всех, кому важно быстро и безошибочно набирать текст. В первую очередь это:

— студенты и школьники, которым нужно печатать рефераты, конспекты и другие учебные материалы. Сервис поможет им увеличить скорость набора и снизить количество опечаток;

— офисные работники, ежедневно работающие с документами, отчётами и электронной почтой. Регулярные тренировки позволят им повысить

продуктивность и уменьшить усталость при длительной работе за компьютером;

— программисты и IT-специалисты, которым важно быстро вводить код и специальные символы. Система уровней включает последовательности символов, характерные для программирования;

— все, кто хочет освоить слепой метод печати или улучшить свои текущие показатели. Игровая форма обучения делает процесс более увлекательным по сравнению с традиционными тренажёрами.

Программа не требует специальных знаний и подходит для пользователей с любым уровнем подготовки. Простота интерфейса и понятная механика позволяют сразу приступить к тренировкам без длительного освоения.

2.4.2 Условия выполнения

Для корректной работы веб-сервиса необходимо соблюдение следующих технических требований:

— клиентские требования:

1) устройство с процессором не менее 2 ядер и тактовой частотой от 1.5 ГГц;
2) минимум 2 ГБ оперативной памяти (рекомендуется 4 ГБ);
3) видеоадаптер с поддержкой современных графических стандартов;
4) физическая клавиатура для ввода текста;
5) одна из современных операционных систем (Windows, macOS или Linux);
6) актуальная версия веб-браузера;
7) минимальное разрешение экрана 1024 на 768 пикселей;
8) стабильное интернет-соединение со скоростью от 1 Мбит/с.

— серверные требования:

1) серверное окружение с поддержкой современных веб-технологий;
2) реляционная система управления базами данных;

3) кэширующий сервер для повышения производительности;

4) выделенный IP-адрес и доменное имя;

5) SSL-сертификат для безопасного соединения.

Программа реализована как веб-приложение и не требует установки дополнительного программного обеспечения на устройства пользователей. Доступ осуществляется через защищенное соединение по стандартному веб-протоколу.

Для организаций могут потребоваться дополнительные настройки сетевого оборудования:

— открытие портов для веб-соединений;

— настройка исключений для веб-сокетов;

— разрешение доступа к API-интерфейсам.

Рекомендуется использовать устройства с современными характеристиками для обеспечения плавной работы графических элементов и быстрой обработки вводимого текста.

2.4.3 Выполнение программы

— инициализация приложения:

1) загрузка клиентской части. При обращении пользователя к веб-сервису загружается клиентское приложение, построенное на Next.js. Инициализируется Phaser 3 как игровой движок, создается базовый контекст рендеринга [9];

2) проверка состояния аутентификации. Приложение проверяет наличие JWT-токена в localStorage/cookies для автоматической авторизации пользователя. При отсутствии данных открывает окно для авторизации или регистрации.

— загрузка игровых ресурсов:

1) получение статических объектов. Загружаются графические ресурсы (спрайты персонажей, фоны уровней), звуковые файлы и шрифты из публичной директории Next.js. Для больших файлов реализовано прогрессивное отображение с прелоадерами;

2) запрос динамических данных. Через API-эндпоинты запрашиваются: конфигурация уровней, таблицы лидеров, статистика пользователя (если авторизован).

— основной игровой цикл:

1) создание игровой сцены. Phaser инициализирует сцену с настройками физики (Arcade Physics), камерой и слоями отображения. Создаются игровые объекты: персонаж, враги, элементы окружения [10];

2) обработка ввода. Реализована система обработки клавиатурного ввод. Каждое нажатие клавиш анализируется на соответствие текущему заданию;

3) игровая логика. В update-цикле Phaser обрабатывается: движение персонажа, столкновения, таймеры боев, проверка условий завершения уровня. Скорость анимаций привязана к FPS.

— интеграция с бэкендом:

1) синхронизация прогресса. При завершении уровня отправляется POST-запрос с результатами (время прохождения, ошибки) на сервер для обновления статистики;

2) работа с API. Используются защищенные HTTPS-эндпоинты Next.js API Routes для: получения конфигурации уровней, обновления таблиц лидеров, регистрации достижений.

— обработка ошибок:

1) отслеживание игровых сбоев. Реализованы fallback-состояния при ошибках загрузки ресурсов. Критические ошибки Phaser перехватываются Sentry;

2) восстановление сессии. При повторном подключении предлагается продолжить с последнего checkpoint.

— оптимизации производительности

1) ленивая загрузка. Игровые сцены загружаются динамически через next/dynamic для уменьшения первоначального bundle;

2) рендеринг интерфейса. React-компоненты (меню, модальные окна) отрисовываются поверх Phaser Canvas;

3) управление памятью. Phaser автоматически выгружает неиспользуемые текстуры. Реализован ручной cleanup при переходе между сценами.

2.4.3.1 Работа пользователя

— навигация по интерфейсу сервиса

1) пользователь видит главную страницу с тематическим фэнтезийным фоном, где представлены: название сервиса «Typing Quest», краткое описание проекта, кнопка «Начать» для перехода к обучению и кнопка «О проекте» с дополнительной информацией;

2) при первом входе система предлагает создать аккаунт (ввод email/логина и пароля) или авторизоваться (для зарегистрированных пользователей). После успешного входа происходит автоматический переход на карту уровней.

— взаимодействие с уровнями:

1) на интерактивной карте представлены все доступные уровни с визуальными отметками о прогрессе (пройден/не пройден). При наведении на уровень отображается его название и сложность;

2) после выбора конкретного уровня открывается окно с детальной информацией: описание сюжета уровня, рекомендуемые символы для ввода, среднее время прохождения, таблица лидеров и личные результаты пользователя (если уровень пройден ранее);

3) для начала прохождения необходимо нажать кнопку «Начать печатать». Система предоставляет трёхсекундный отсчет перед стартом игрового процесса.

— игровой процесс:

1) скорость движения рыцаря напрямую зависит от скорости ввода текста пользователем. Чем быстрее и точнее ввод, тем быстрее персонаж продвигается по уровню;

2) при встрече с противником активируется режим боя, где необходимо ввести определенную последовательность символов за

ограниченное время. Успешный ввод наносит урон противнику, ошибки или медленный ввод приводят к потере здоровья персонажа;

3) при достижении финиша открывается окно результатов с подробной статистикой: количество набранных очков, скорость ввода (знаков в минуту), количество допущенных ошибок, время прохождения. За быстрое завершение начисляются бонусные очки.

— работа с профилем и статистикой:

1) в профиле пользователь может просматривать: общее количество напечатанных символов, среднюю скорость ввода, прогресс по всем уровням, полученные достижения;

2) сервис награждает пользователя виртуальными достижениями за выполнение определенных условий;

3) визуализация результатов обучения в виде графиков, показывающих динамику улучшения скорости и точности печати за выбранный период.

— соревновательные функции:

1) для каждого уровня существует рейтинговая таблица, отображающая ТОП-5 игроков по количеству набранных очков. Пользователь может видеть свою позицию в общем рейтинге;

2) начисление дополнительных очков за быстрое прохождение уровня: чем быстрее завершен уровень относительно среднего времени, тем больше бонусных очков получает пользователь.

2.4.3.2 Техника безопасности при работе на компьютере

Работа за компьютером требует соблюдения определенных правил безопасности, направленных на сохранение здоровья пользователя и предотвращение несчастных случаев. Длительное взаимодействие с электронно-вычислительной техникой сопряжено с рядом рисков, включая зрительное перенапряжение, статическую нагрузку на позвоночник и суставы, а также потенциальные угрозы, связанные с электробезопасностью и пожарной опасностью. В связи с этим организация безопасного рабочего

процесса приобретает особую значимость как с точки зрения охраны труда, так и с позиции повышения эффективности деятельности.

Важнейшим аспектом является правильная организация рабочего пространства. Монитор должен располагаться на расстоянии от 50 см до глаз пользователя, при этом его верхний край должен находиться на уровне или чуть ниже прямой линии взгляда. Это позволяет минимизировать нагрузку на шейный отдел позвоночника и снизить утомляемость глаз. Освещение в помещении должно быть равномерным, без резких бликов на экране, для чего рекомендуется использовать рассеянный свет и антибликовые фильтры. Рабочий стол и кресло необходимо подбирать с учетом антропометрических данных пользователя: стул должен обеспечивать поддержку поясницы и регулироваться по высоте, а стол – иметь достаточную площадь для комфортного размещения периферийных устройств.

Особого внимания заслуживает электробезопасность, поскольку компьютерное оборудование работает от сети переменного тока, представляющего потенциальную опасность. Все кабели и разъемы должны находиться в исправном состоянии, без повреждений изоляции. Запрещается подключать к одной розетке несколько мощных устройств во избежание перегрузки и перегрева проводки. Корпус системного блока и монитора должен быть надежно заземлен, а в случае обнаружения неисправностей (посторонний шум, запах гари, искрение) оборудование следует немедленно отключить от сети и обратиться к специалисту.

Гигиена труда при работе за компьютером предполагает соблюдение режима труда и отдыха. Оптимальным считается цикл, при котором после каждого часа непрерывной работы следует делать перерыв на 10 минут. В это время рекомендуется выполнять простые физические упражнения для улучшения кровообращения, а также гимнастику для глаз. Микроклимат в помещении также играет существенную роль: температура воздуха должна поддерживаться в пределах от 20 до 24°C, а влажность – не опускаться ниже 40%, для чего целесообразно использовать увлажнители и системы вентиляции.

Не менее важна профилактика профессиональных заболеваний, связанных с длительной работой за компьютером. Постоянное использование клавиатуры и мыши может приводить к развитию туннельного синдрома запястья, поэтому рекомендуется применять эргономичные модели устройств и специальные коврики с поддержкой кисти. Для предотвращения болей в спине и шее следует избегать статичных поз, периодически менять положение тела и использовать кресла с анатомической спинкой.

Пожарная безопасность также является неотъемлемой частью техники безопасности при эксплуатации компьютерной техники. Запрещается оставлять включенное оборудование без присмотра на длительное время, особенно в ночные часы. Вблизи рабочих мест не должно находиться легковоспламеняющихся материалов, а в помещении необходимо наличие огнетушителя класса ABC, предназначенного для тушения электрооборудования.

Таким образом, соблюдение правил техники безопасности при работе на компьютере позволяет не только снизить риск травматизма и профессиональных заболеваний, но и создать комфортные условия для продуктивной деятельности.

3 Экономическая часть

3.1 Расчёт основной и дополнительной заработной платы с отчислениями на социальное страхование

Смета затрат на производство – это общий свод плановых затрат (в стоимостном выражении) на производство продукции, выполнение работ и услуг в соответствии с производственной программой предприятия. Расчет сметы затрат на производство осуществляется на основе бизнес-плана в части производственной программы на будущий финансовый год, а также анализа фактических данных прошедшего периода и расчетных нормативов по статьям затрат, принятых в организации.

Расчетные нормативы по статьям затрат планируются на основе анализа действующих рыночных цен на аналогичную выпускаемую продукцию, фактических цен реализации продукции прошедшего финансового года.

По этим данным определяется предельный норматив затрат на производство при существующем уровне рыночной (договорной) цены. Рассчитанный таким образом предельный норматив затрат на единицу продукции с учетом принятых на предприятии нормативов по статьям затрат позволяет определить плановую смету в разрезе статей и элементов.

С учетом специфики отраслевой деятельности организации составление сметы затрат на производство ведется по сегментам (видам) деятельности, видам продукции, работ, услуг; дальнейшая группировка возможна по структурным подразделениям.

На заключительном этапе планирования сметы затрат на производство составляется сводная смета под плановую производственную программу с учетом объемов выпуска.

На первом этапе разработки веб-сервиса необходимо рассчитать смету затрат, а также определить этапы разработки веб-сервиса и трудоёмкость выполнения каждого этапа. Все показатели представлены в Таблице 4.

Таблица 4 – Этапы и трудоёмкость выполнения работ В часах

Этапы разработки	Руководитель	Техник
Постановка задачи	4	10
Выбор инструментальных средств	2	8
Разработка мат. модели	2	16
Построение алгоритма	1	18
Программирование	3	52
Тестирование	2	15
Отладка	1	12
Написание пояснительной записки	2	10
Итого	17	141

Месячный оклад руководителя составляет 34000 руб., месячный оклад техника составляет 28000 руб.

При расчёте основной заработной платы учитывается заработная плата всех категорий работников, непосредственно занятых разработкой программного продукта. Размер заработной платы определяется исходя из количества исполнителей и квалификационного уровня, а также затраченного ими времени в целом на разработку программного продукта и отдельных его этапов. Так как разработка происходила в апреле, то количество рабочих часов равно 175.

Основная заработная плата руководителя составляет 34000 * 17 / 175 = 3302,86 руб., основная заработная плата техника составляет 28000 * 141 / 175 = 22560 руб. В сумме основная заработная плата руководителя и техника составила 25862,86 руб.

Таблица 5 – Основная заработная плата работников В рублях

Должность	Заработная плата
Руководитель проекта	3302,86
Техник	22560
Итого	25862,86

Дополнительная заработная плата включает различные виды доплат сверх основной заработной платы: премии, доплату за работу в сверхурочное время, надбавки за профессиональное мастерство, оплату очередного и учебных отпусков и прочие виды доплат.

Дополнительная заработная плата устанавливается на предприятии в процентах от суммы основной заработной платы и составляет от 16 до 20%.

Таким образом, дополнительная заработная плата будет определяться по следующей формуле:

$$ЗП_{доп} = ЗП_{осн} \times 20\% / 100\% , \quad (3.1)$$

где $ЗП_{осн}$ – основная заработная плата, руб.

Значение $ЗП_{осн}$ возьмём из Таблицы 4 и рассчитаем дополнительную заработную плату.

$$ЗП_{доп} = 25862,86 \times 20\% / 100\% = 5172,57 \text{ руб.}$$

На основании вышеприведенных расчетов, делаем вывод, что основная и дополнительная заработная плата работников по создания программного продукта составит 31035,43 руб.

Всего отчисления на социальное страхование составляют 30% от суммы основной и дополнительной заработной платы. Отчисления на социальное страхование рассчитываются по формуле 3.2.

$$З_{стр} = (ЗП_{осн} + ЗП_{доп}) \times 30\% / 100\% , \quad (3.2)$$

где $ЗП_{осн}$ – основная заработная плата, руб.;

$ЗП_{доп}$ – дополнительная заработная плата, руб.

Значение $ЗП_{осн}$ возьмём из Таблицы 5, а $ЗП_{доп}$ рассчитывается по формуле 3.1. Вычислим величину отчислений с помощью формулы 3.2.

$$ЗП_{стр} = (31035,43) \times 30\% / 100\% = 9310,63 \text{ руб.}$$

Таким образом, из расчетов видно, что отчисления на социальное страхование составят 9310,63 руб.

В сумме основная и дополнительная заработная плата с отчислением на социальное страхование равна 40346,06 руб.

3.2 Расчёт стоимости материалов и лицензионного обеспечения

Расчёт стоимости необходимых материалов для разработки программного продукта рассчитывается на основе норм расхода материальных ресурсов и оптовых цен на приобретение по формуле 3.3.

$$С_м = Н_р \times Ц, \quad (3.3)$$

где $Н_р$ – норма расхода материальных ресурсов в натуральных единицах;

Ц – цена приобретения за единицу, руб.

Расчёт стоимости материалов представлен в Таблице 6.

Таблица 6 – Расчёт стоимости материалов В рублях

Наименование материала	Норма расхода	Цена за единицу	Сумма
USB-накопитель, шт.	1	299	299
Бумага, уп.	1	515	515
Папка, шт.	1	370	370
Итого			1184

Общая стоимость материалов составит 1184 руб.

При создании программного продукта используется лицензионное программное обеспечение, поэтому необходимо включить стоимость лицензионных программ, используемых при разработке программного продукта, в смету затрат.

Сначала определим перечень лицензионных программ, срок действия и норму установки в таблице 7.

Таблица 7 – Перечень программного обеспечения для выполнения работ

Наименование лицензионных программ	Норма установки, шт.	Цена, руб.
Microsoft Windows 11 Home	1	15500
Хостинг	1	1000
Visual Studio Code	1	В свободном доступе
Google Docs	1	В свободном доступе
Итого		16500

Стоимость лицензионного программного обеспечения для разработки программного продукта составит 16500 руб.

Расходы на эксплуатацию составляют:

— основная и дополнительная заработная плата;

— отчисления на социальное страхование;

— стоимость материалов;

— стоимость лицензионного обеспечения.

В итоге расходы на эксплуатацию равны 58030 руб.

3.3 Расчёт накладных расходов

Накладные расходы представляют собой дополнительные к основным расходам затраты на управление, организацию и обслуживание производства.

Не связаны напрямую с основным производством товаров или предоставлением услуг, не входят в стоимость материалов и оплату труда.

Накладные расходы закладываются в себестоимость товара, издержки его производства и обращения, но не прямо, а косвенно, пропорционально стоимости материалов и сырья, сумме заработной платы и так далее.

Накладные расходы составляют 10% от расходов на эксплуатацию. и рассчитываются по формуле 3.4.

$$Н_р = (Р_{эксп} \times 10\%) / 100\% , \quad (3.4)$$

где $Р_{эксп}$ – величина расходов на эксплуатацию, руб.

$$Н_р = (58030 \times 10\%) / 100\% = 5803 \text{ руб.}$$

3.4 Составление и расчёт цены реализации программного продукта

На основании выше рассчитанных данных составим смету затрат на программный продукт в Таблице 8.

Таблица 8 – Смета затрат на программный продукт

Статьи затрат	Сумма, руб.	Структура затрат, %
Основная и дополнительная заработная плата	31035,43	48,62
Отчисления на социальное страхование	9310,63	14,59
Стоимость материалов	1184	1,85
Стоимость лицензионного обеспечения	16500	25,85
Итого расходы на эксплуатацию	58030	90.91
Накладные расходы	5803	9,09
Итого	63833	100

Таким образом, общая сумма затрат на разработку программного продукта составит 63833 руб.

Под структурой затрат понимается удельный вес (в процентах) каждой статьи в общей величине затрат. По результатам расчета построим диаграмму структуры затрат на программный продукт и представим на Рисунке Б.1.

Цена является одним из самых важных экономических показателей. Основная функция состоит в обеспечении выручки от реализации

программного продукта, поэтому цена определяет прибыль и финансовую стабильность предприятия, его жизнеспособность.

Так как целью предпринимательской деятельности является получение прибыли, то цена реализации программного продукта будет представлять собой сумму затрат на его разработку и запланированную величину прибыли. Примем уровень рентабельности Р = 35%. Величину общих затрат на разработку программного продукта возьмём из таблицы 3.5, а стоимости материалов из таблицы 3.3.

Расчёт оптовой цены производится по формуле 3.5.

$$Ц_{опт} = З \times (1 + Р), \quad (3.5)$$

где З – общие затраты на разработку программного продукта, руб.;

Р – уровень рентабельности проекта, коэффициент.

Произведём расчет оптовой цены используя формулу 3.5.

$$Ц_{опт} = 63833 \times (1 + 0{,}35) = 86175 \text{ руб.}$$

Расчёт прибыли от продажи продукта производится по формуле 3.6.

$$П = Ц_{опт} - З, \quad (3.6)$$

$$П = 86175 - 63833 = 22342 \text{ руб.}$$

Таким образом, можно сделать вывод, что данный продукт прибыльный и может быть реализован. Прибыль от продажи программного продукта составит 22342 руб.

Заключение

В рамках выполнения выпускной квалификационной работы была достигнута поставленная цель – разработан интерактивный веб-сервис для тренировки слепой печати в игровой форме. В ходе работы проведён анализ существующих решений и выделены основные недостатки.

Предложенное решение направлено на устранение этих проблем за счёт внедрения игровых элементов, делающих процесс тренировки более интересным и мотивирующим. Разработанная система позволяет пользователю управлять персонажем, вводя символы на клавиатуре, и получать очки за правильные действия. Такой подход способствует не только формированию устойчивого навыка слепой печати, но и поддерживает высокий уровень интереса к обучению на протяжении всего процесса.

В проекте реализованы основные функциональные модули веб-сервиса и проведено тестирование. Также произведена оценка экономической эффективности разработки.

Список литературы

1 ГОСТ Р 7.0.100-2018. Библиографическая запись. Библиографическое описание. Общие требования и правила составления : национальный стандарт Российской Федерации : дата введения 2019-07-01 / Федеральное агентство по техническому регулированию. – Москва : Стандартинформ, 2018. – 124 с.

2 ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Москва : Издательство стандартов, 2010. – 24 с.

3 Баумгартнер, С. Рецепты TypeScript. Программирование на уровне типов для реальных задач / С. Баумгартнер. – Москва: Sprint Book, 2025. – 432с. – ISBN 978-601-08-4355-4.

4 Рогова, А. PostgreSQL 17 изнутри / Рогова А. – Москва: ДМК Пресс, 2025. – 668с. – ISBN 978-5-93700-372-0.

5 Черный, Б. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений / Б. Черный. – Питер: Издательский дом «Питер» – 352с. – ISBN 978-5-4461-1651-5.

6 Официальная документация Git на английском языке: [Электронный ресурс]. – URL: https://git-scm.com/doc (дата обращения: 28.04.2025).

7 Официальная документация Nest.js на английском языке: [Электронный ресурс]. – URL: https://docs.nestjs.com/ (дата обращения: 10.04.2025).

8 Официальная документация Next.js на английском языке: [Электронный ресурс]. – URL: https://nextjs.org/docs (дата обращения: 30.03.2025).

9 Официальная документация Phaser на английском языке: [Электронный ресурс]. – URL: https://docs.phaser.io/ (дата обращения: 15.04.2025).

10 Официальная документация Postman на английском языке: [Электронный ресурс]. – URL: https://learning.postman.com/docs/ (дата обращения: 25.04.2025).

11 Официальная документация PostgreSQL на английском языке: [Электронный ресурс]. – URL: https://www.postgresql.org/docs/ (дата обращения: 10.04.2025).

12 Официальная документация TailwindCSS на английском языке: [Электронный ресурс]. – URL: https://tailwindcss.com/docs (дата обращения: 27.03.2025).

13 Официальная документация Tiled на английском языке: [Электронный ресурс]. – URL: https://doc.mapeditor.org/en/stable/ (дата обращения: 12.04.2025).

Список сокращений и обозначений

HTTP – HyperText Transfer Protocol (протокол передачи гипертекста).

IT – Information Technology (информационные технологии).

DOM – Document Object Model (объектная модель документа).

CSS – Cascading Style Sheets (каскадные таблицы стилей).

HTML – HyperText Markup Language (язык гипертекстовой разметки).

REST API – Representational State Transfer Application Programming Interface (программный интерфейс передачи состояния представления).

JWT – JSON Web Token (токен веб-доступа в формате JSON).

SSL – Secure Sockets Layer (уровень защищённых сокетов).

HTTPS – HyperText Transfer Protocol Secure (безопасный протокол передачи гипертекста).

FPS – Frames Per Second (кадров в секунду).

JSON – JavaScript Object Notation (формат обмена данными).

SQL – Structured Query Language (язык структурированных запросов).

ES6 – ECMAScript 6 (стандарт JavaScript).

API – Application Programming Interface (программный интерфейс приложения).

IDEF0 – Integrated DEFinition for Function Modeling (методология функционального моделирования).

ER – Entity-Relationship (модель «сущность-связь»).

npm – Node Package Manager (менеджер пакетов Node.js).

SSD – Solid State Drive (твердотельный накопитель).

RAM – Random Access Memory (оперативная память).

CPU – Central Processing Unit (центральный процессор).

URL – Uniform Resource Locator (унифицированный указатель ресурса).

UI – User Interface (пользовательский интерфейс).

UX – User Experience (пользовательский опыт).

Приложение А

(обязательное)

Рисунок А.1 – Стартовый экран

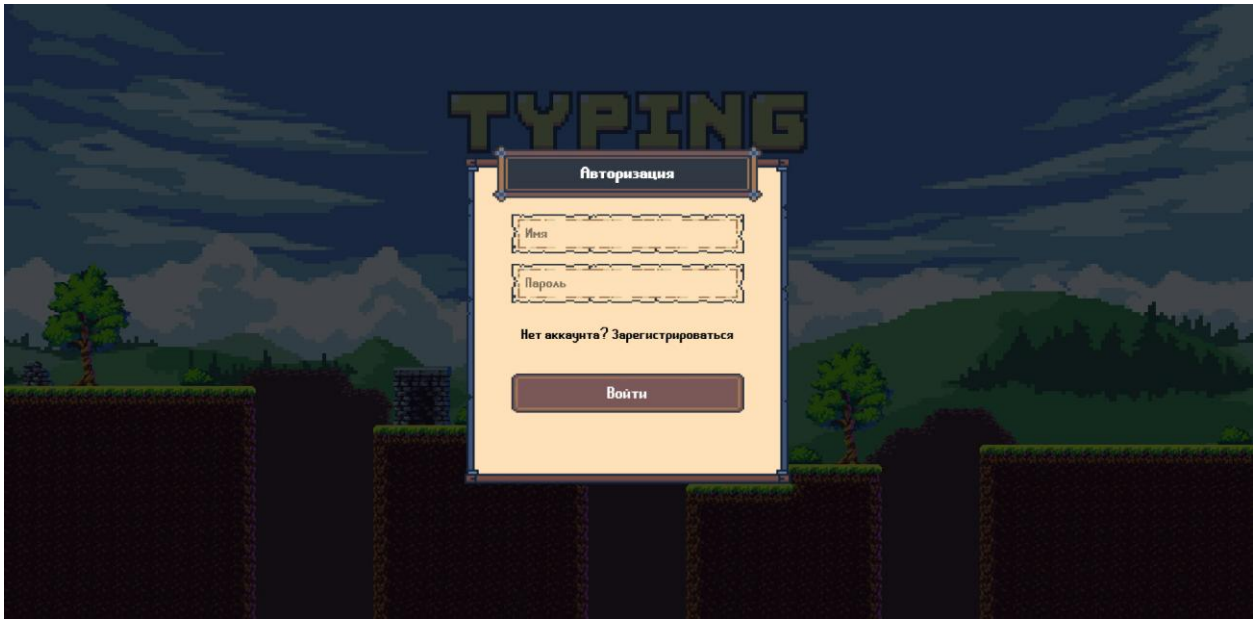

Рисунок А.2 – Экран авторизации

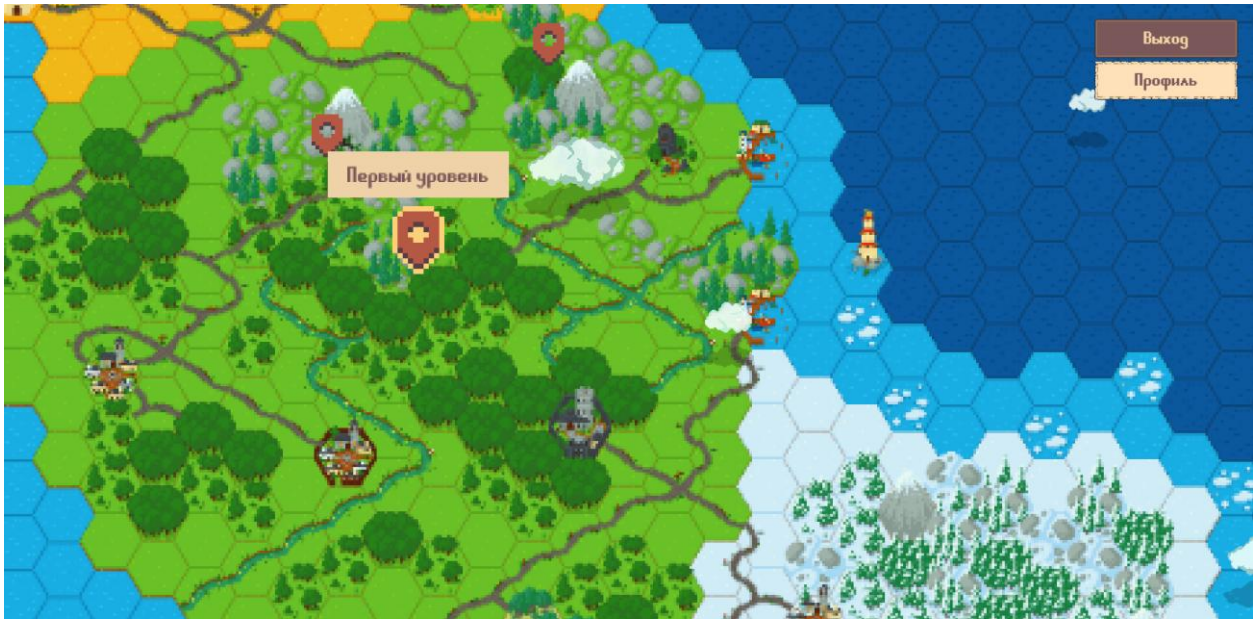

Рисунок А.3 – Карта уровней

Рисунок А.4 – Экран боя

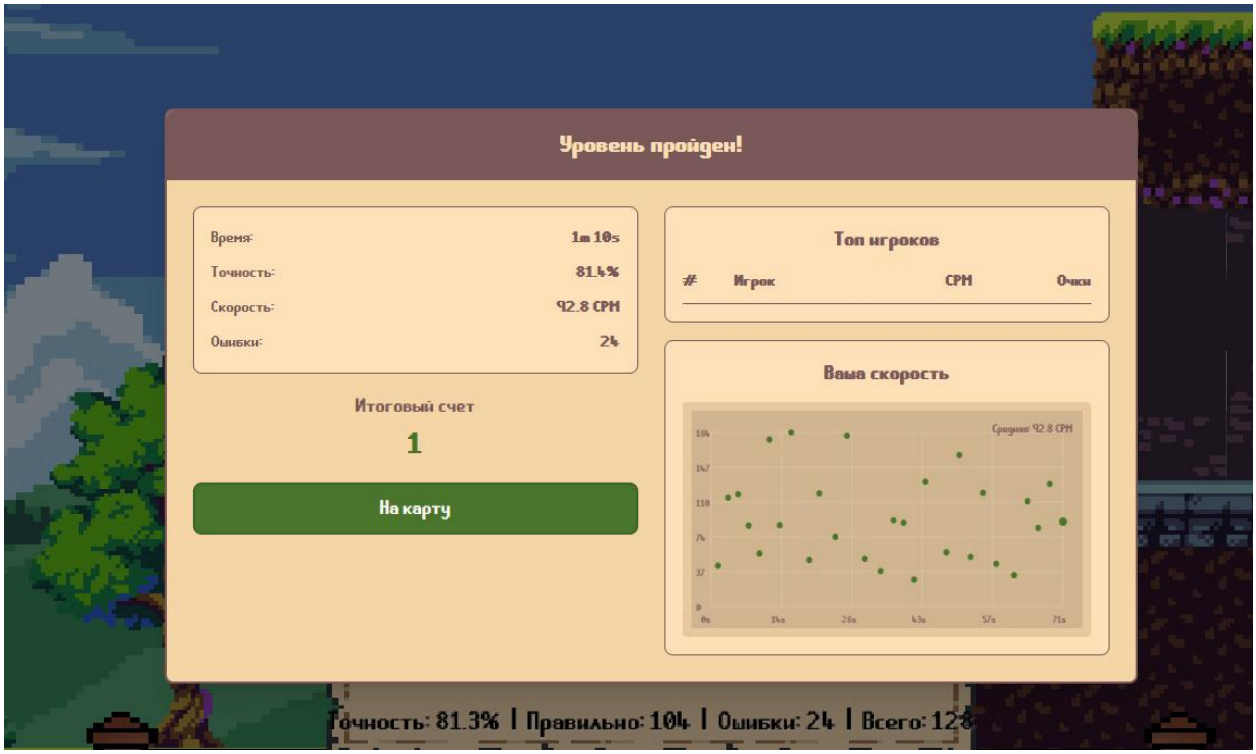

Рисунок А.5 – Экран результатов

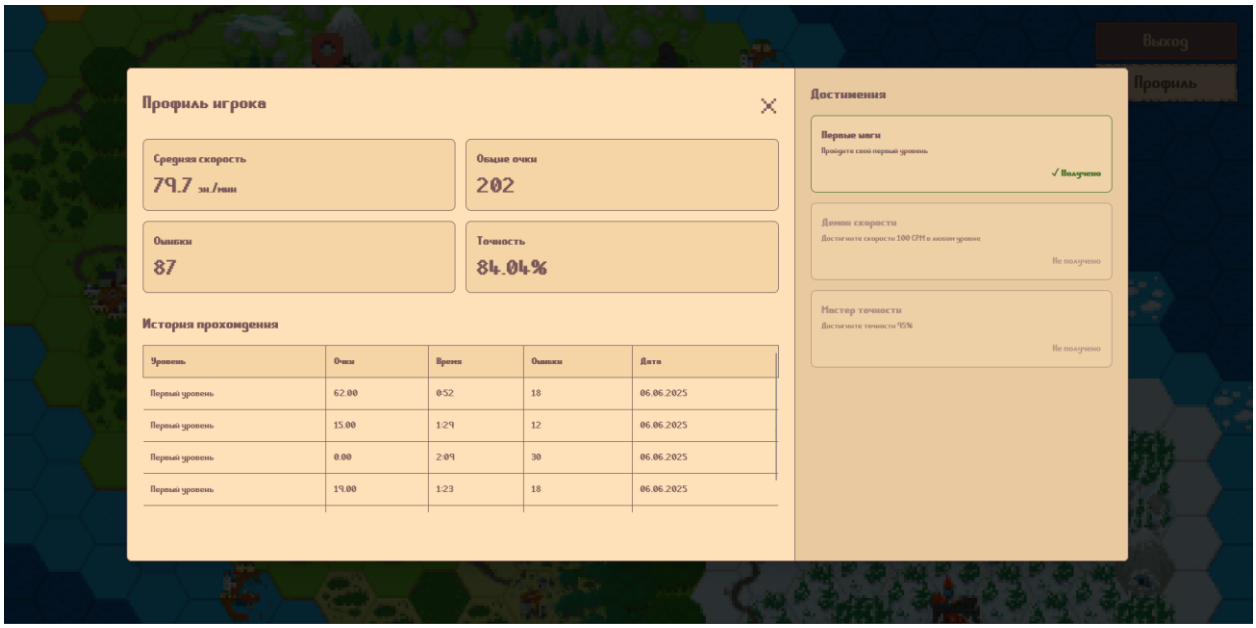

Рисунок А.6 – Экран профиля

Приложение Б

(обязательное)

Рисунок Б.1 – Структура затрат на программный продукт

Приложение В

(обязательное)

В.1 Листинг unit-тестирования регистрации пользователя

```
    describe('register', () => {
        it('должен создавать пользователя и возвращать токены',
async () => {
          const mockUser = {
            user_id: 1,
            username: 'newuser',
            email: 'new@example.com',
            password_hash: 'hashed_password',
            created_at: new Date(),
            last_login: null,
            sessions: [],
            results: [],
            achievements: [],
          } as User;

          usersService.createUser.mockResolvedValue(mockUser);
          jest.spyOn(service, 'login').mockResolvedValue({
            access_token: 'access_token',
            refresh_token: 'refresh_token',
            session_id: 'session_id',
            user: {
              id: 1,
              username: 'newuser',
              email: 'new@example.com',
            },
          });
             const result = await service.register('newuser',
'new@example.com', 'Password123!');
          expect(usersService.createUser).toHaveBeenCalledWith('
newuser', 'new@example.com', 'Password123!');
          expect(result).toEqual({
            access_token: 'access_token',
            refresh_token: 'refresh_token',
            session_id: 'session_id',
            user: {
              id: 1,
              username: 'newuser',
              email: 'new@example.com',
            },
          });
        });
      });
```

В.2 Листинг unit-тестирования авторизации пользователя

```
    describe('login', () => {
        it('должен возвращать токены и создавать сессию', async
() => {
          const mockUser = {
            user_id: 1,
            username: 'testuser',
            email: 'test@example.com',
          };
           jwtService.sign.mockImplementation((payload, options)
=> {
                  if (options?.expiresIn === '15m') return
'access_token';
                  if (options?.expiresIn === '30d') return
'refresh_token';
            return '';
          });

          sessionService.createSession.mockResolvedValue({
            session_id: 'session_id',
            user_id: 1,
            refresh_token_hash: 'hashed_refresh_token',
            created_at: new Date(),
            expires_at: new Date(),
            user: mockUser as User,
          } as UserSession);
          const result = await service.login(mockUser as User);
          expect(jwtService.sign).toHaveBeenCalledWith(
            {
              username: 'testuser',
              sub: 1,
              email: 'test@example.com',
            },
            { expiresIn: '15m' },
          );
          expect(jwtService.sign).toHaveBeenCalledWith(
            {
              username: 'testuser',
              sub: 1,
              email: 'test@example.com',
            },
            { expiresIn: '30d' },
          );
          expect(sessionService.createSession).toHaveBeenCalledW
ith(1, 'refresh_token');
          expect(result).toEqual({
            access_token: 'access_token',
            refresh_token: 'refresh_token',
            session_id: 'session_id',
            user: {
              id: 1,
              username: 'testuser',
              email: 'test@example.com',
            },
```

```
      });
    });
});
```