

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МО и ПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

Интернет портал по обучению школьников
алгоритмизации с использованием визуального
программирования

Студент _____ Подгорнов Данил Владимирович
(подпись, дата) (ФИО полностью)

Руководитель _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролер _____ Такташкин Д.В
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 202_ г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Подгорнов Данил Владимирович гр. 20ВИ2 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Интернет портал по обучению школьников алгоритмизации с
использованием визуального программирования

Тема утверждена приказом ПГУ № _____ от "___" _____ 202_ г.

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Такташкин Д.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

*Назначение: Создание интернет портала, который будет представлять
использоваться для обучения школьников основам алгоритмизации
с помощью визуального программирования.*

Среда выполнения: Windows

Языковые средства разработки: JavaScript, Java

Инструментальные средства: NetBeans, IntelliJ IDEA, Google Chrome

Технологии разработки ООП, клиент-ориентированный подход.

Основные функции разрабатываемого программного обеспечения:

а) создание визуальных алгоритмов для обучения школьников;

б) предоставление теории по командам, которые есть в уровне;

в) проверка корректности и оценивания решения пользователей;

г) визуализация процесса выполнения команд персонажем;

д) взаимодействие с сервером для хранения и обработки данных;

е) управление прогрессом пользователей.

5. Перечень подлежащих разработке вопросов:

1) Анализ предметной области и постановка задачи;
2) План разработки бюджета;
3) Разработка требований и проектирование;
4) Реализация;
5) Контроль качества программного обеспечения;
6) Оформление пояснительной записки.

6. Календарный график выполнения работы

п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
	Анализ предметной области и требований к системе	10%	12.04.2024 – 16.04.2024	
	Проектирование программных средств	20%	17.04.2024 – 26.04.2024	
	Реализация программных средств	30%	27.04.2024 – 11.05.2024	
	Тестирование программных средств	20%	12.05.2024 – 21.05.2024	
	Оформление пояснительной записки	20%	22.05.2024 – 01.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 87 листов, 18 рисунков, 17 таблиц, 13 использованных источников и 2 приложения.

ОБУЧАЮЩИЙ ПОРТАЛ, ИНТЕРНЕТ-ПОРТАЛ, ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ, БЛОЧНОЕ ПРОГРАММИРОВАНИЕ, UML, JAVA, INTELLIJ IDEA, NETBEANS, POSTGRESQL.

Объект исследования – интернет-портал по обучению школьников алгоритмизации с использованием визуального программирования.

Целью выпускной квалификационной работы является разработка интернет-портала по обучению школьников алгоритмизации с использованием визуального программирования.

Технологии разработки – программные средства написаны с использованием языков программирования JavaScript и Java, фреймворка Spring в среде разработки NetBeans, IntelliJ IDEA.

Среда исполнения – Windows 10, веб-браузер.

Результаты работы – по итогу выполнения выпускной квалификационной работы разработан интернет-портал по обучению школьников алгоритмизации с использованием визуального программирования.

Все требования к разработке приложения соблюдены.

					ПГУ 09.03.02 – 08BP202.06 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Интернет портал по обучению школьников алгоритмизации с использованием визуального программирования Пояснительная записка	Лит.	Лист	Листов
Разраб.		Подгорнов Д.В.						
Пров.		Такташкин Д.В.					4	87
Н. Контр.		Такташкин Д.В.				Группа 20ВИ2		
Утв.		Козлов А.Ю.						

Содержание

Введение	6
1 Анализ обучающих порталов	7
1.1 Анализ предметной области	7
1.2 Анализ аналогичных обучающих порталов.....	8
1.3 Анализ требований к обучающему portalу	11
1.4 Постановка задачи.....	16
2 Проектирование обучающего портала	18
2.1 Выбор технологий проектирования	18
2.2 Проектирование архитектуры	18
2.3 Проектирование графического интерфейса.....	21
2.4 Оценка бюджета разработки обучающего портала	25
3 Разработка и реализация интернет-портала по обучению школьников алгоритмизации при помощи визуального программирования	30
3.1 Выбор технологий разработки.....	30
3.2 Разработка архитектуры обучающего портала.....	31
3.3 Разработка слоя представления приложения.....	37
4 Тестирование программных средств.....	40
4.1 Тестирование белым ящиком	40
4.1.1 Покрытие операторов.....	40
4.1.2 Покрытие решений.....	41
4.1.3 Покрытие маршрутов	42
4.2 Автоматизированное тестирование.....	43
4.3 Метрика SLOC.....	46
4.4 Метрика Холстеда	46
Заключение	51
Список использованных источников	52
Приложение А (рекомендуемое) Варианты использования	54
Приложение Б (рекомендуемое) Листинг программы	57

Введение

В эпоху стремительного развития цифровых технологий интернет-порталы стали неотъемлемой частью образовательного процесса, предоставляя доступ к разнообразным и качественным образовательным ресурсам. Большую значимость они приобретают в контексте обучения алгоритмизации, потому что программирование является самой перспективной областью знаний в современном мире. В особенности такие порталы полезны для школьников, которые только ищут свою будущую профессию.

Интернет-порталы стали такими популярными из-за доступности использования. Они позволяют учащимся изучать программирование в удобное для них время и в своем собственном темпе, что особенно важно для школьников, у которых занятости хватает. На мой взгляд, самые увлекательные платформы это те, в которых используется визуальное программирование. В таком виде программирования процесс изучения материала напоминает игру, в которой надо пользователю выполнить разные задачи для достижения цели. Это делает процесс обучения более интересным и запоминающимся.

Не менее важным аспектом является подготовка школьников к будущей профессии. Платформы помогают получить и закрепить начальные знания, а дальнейшее изучение программирования школьники могут продолжить в университетах, а благодаря уже приобретённым знаниям, процесс обучения будет значительно проще. Также стоит отметить, что интернет-порталы не только обучают основам программирования, но и помогают развивать критическое мышление, аналитические способности и умение решать сложные задачи, что тоже пригодится в будущей профессии.

Исходя из вышеизложенного, принято решение создать собственный интернет-портал по обучению школьников алгоритмизации при помощи визуального программирования.

1 Анализ обучающих порталов

1.1 Анализ предметной области

В современном мире интернет-порталы играют ключевую роль в образовании школьников, предоставляя доступ к разнообразным образовательным ресурсам и методикам. Эти платформы не только расширяют возможности для обучения, но и делают процесс более гибким и доступным. Проанализируем различные виды образовательных интернет-порталов для понимания, каким именно навыкам обучают школьников:

- существуют порталы для обучения школьным предметам, которые предоставляют материалы и задания по математике, физике, химии, истории, биологии и литературе;
- платформы для изучения иностранных языков предлагают интерактивные упражнения на чтение, письмо, аудирование и говорение, а также игровые элементы для мотивации учащихся;
- платформы для развития навыков программирования и ИТ обучают школьников основам программирования и информационным технологиям через проекты и задания по созданию вебсайтов и приложений;
- порталы для развития креативных навыков, такие как рисование, музыка, писательство и дизайн, предлагающие онлайн-уроки и мастер-классы.

В каждой сфере интернет-порталов хватает, но в теме образовательных платформ по изучению алгоритмизации не хватает. Потому что некоторые из них имеют высокую цену за подписку, другие представляют из себя конечный продукт, который дальше не развивается, а последние предоставляют недостаточную теорию для объяснения самого процесса алгоритмизации. Все это наталкивает на идею о необходимости разработки собственного интернет-портала по обучению школьников алгоритмизации, который не будет иметь вышеперечисленных недостатков.

1.2 Анализ аналогичных обучающих порталов

В ходе разработки обучающего портала был проведён анализ похожих обучающих платформ. Далее приведён обзор некоторых обучающих порталов.

Образовательная среда «ПиктоМир», ориентированная на использование визуального языка программирования, предлагает решения как в виде вебсайта, так и в виде кроссплатформенного приложения, что несомненно является плюсом с точки зрения удобства использования. Также стоит выделить ещё одно достоинство – это полностью бесплатный доступ к обучающей среде. В «ПиктоМир» пользователям предлагается выполнять задачи с помощью различных роботов, каждый из которых обладает уникальным функционалом и для которых разработаны свои виды уровней.

Но протестировав «ПиктоМир» в глаза сразу бросаются следующие очевидные недостатки:

- бестекстовый интерфейс;
- неудобный, неприветливый в использовании интерфейс: команды робота перетаскиваются очень медленно;
- медленная работа вебсайта: если приложение работает плавно, то вебсайт делает изображение экрана дерганным;
- невозможность повернуть изометрическую карту: одни объекты могут перекрывать другие;
- однообразные уровни: роботы имеют разный функционал и оформление, а вот сами уровни отличаются только расположением и количеством объектов взаимодействия на них;
- всего 1 музыкальная композиция, которая начинает раздражать после двух минут прослушивания.

Ниже на рисунке 1 будет представлен интерфейс, в котором можно увидеть вышеперечисленные недостатки «ПиктоМира».

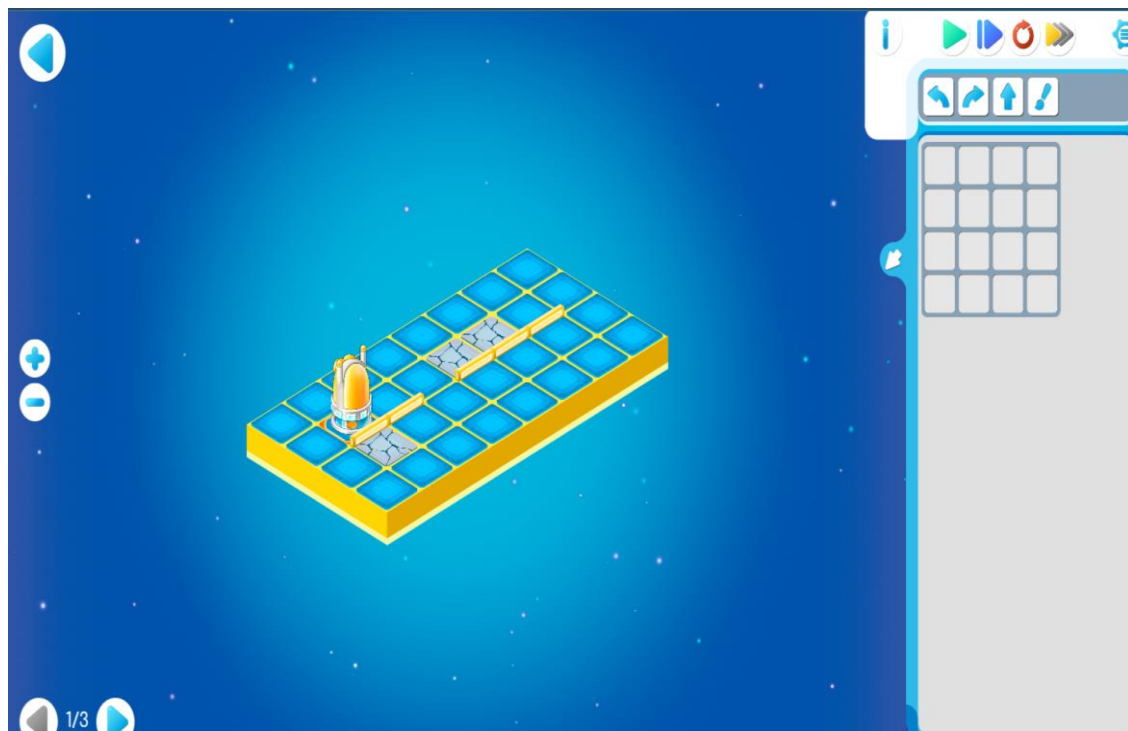


Рисунок 1 – Интерфейс «ПиктоМира»

Подробнее разберем недостатки интерфейса «ПиктоМира»:

- задания на уровнях не сопровождаются текстовыми описаниями, что затрудняет понимание цели каждого уровня;
- текстовое описание отсутствует даже у редактора кода, с которым взаимодействует пользователь для управления роботом.

Из-за этого пользователи вынуждены тратить значительное время на то, чтобы разобраться, что именно требуется выполнить для успешного прохождения уровня. Это может приводить к снижению интереса и отказа от данного обучающего портала.

«Code.org» – это обучающая некоммерческая платформа, которая предлагает бесплатные онлайн-курсы интерактивные уроки и учебные планы для школ. Уроки на платформе представляют из себя некий уровень. На нем есть рабочая область, где с помощью блоков или набора с клавиатуры можно составлять код программы. Интерактивная среда, где какой-либо персонаж производит действия компилируя код программы. Текстовая консоль, в которой

выводится пояснение к данному уроку и цели. Ниже на рисунке 2 представлен интерфейс одного из уроков «Code.org».

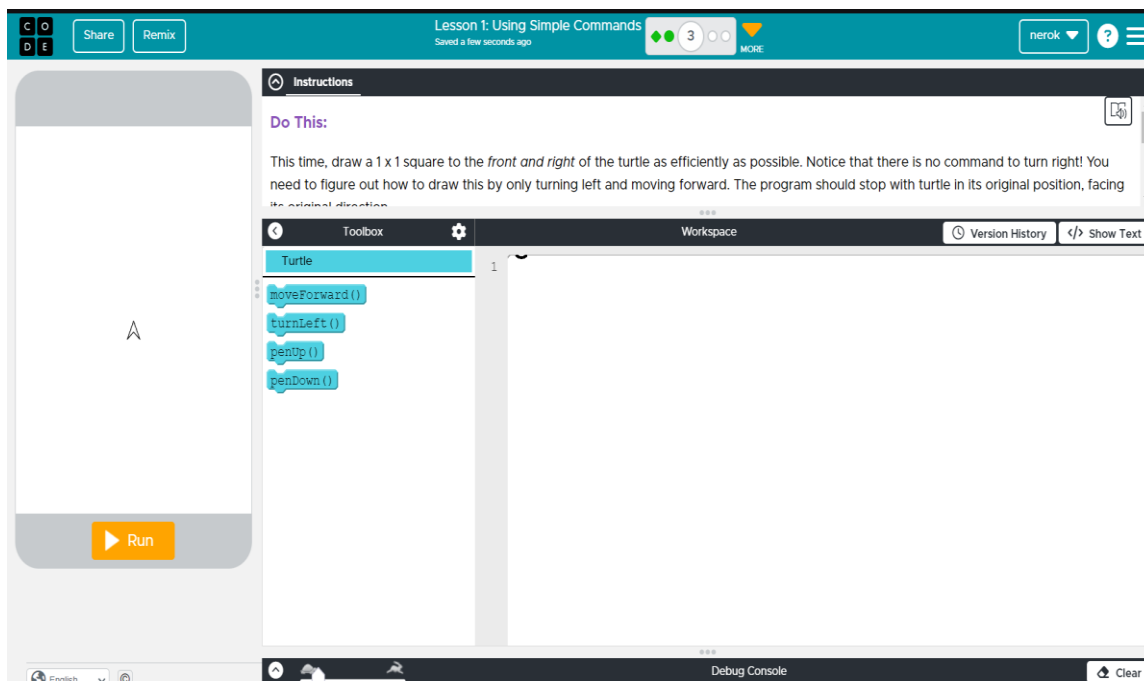


Рисунок 2 – Интерфейс урока «Code.org»

Подробнее изучив интерфейс, выявляются следующие недостатки:

- отсутствие полноценного компилятора: на платформе нельзя отследить пошагово результат программы;
- задания начального уровня: большинство уроков не подходят для учеников с продвинутыми навыками;
- абстрактное отношение к программированию: нет возможности для более глубокого изучения из-за ограниченного набора функций и команд.

Следующий обучающий портал – это «CodeMonkey», созданный специально для обучения детей программированию с помощью интерактивных игр и заданий. Платформа ориентирована на детей дошкольного возраста и учеников младших классов. К преимуществам можно отнести игровой формат, минималистичный интуитивно понятный и удобный интерфейс, поддержку инструментов для родителей и учителей, которые выступают в роли администратора по урокам. Ниже на рисунке 3 представлен интерфейс обучающей среды «CodeMonkey».

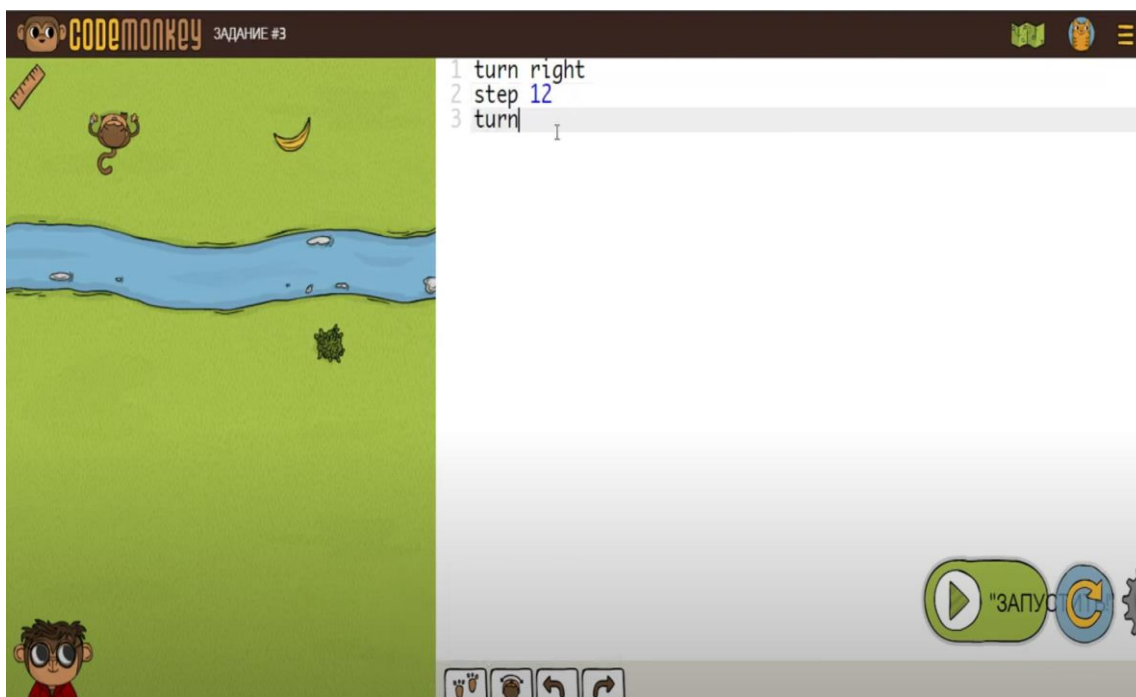


Рисунок 3 – Интерфейс «CodeMonkey»

Внимательно изучив обучающую среду, становятся видны следующие недостатки платформы:

- ограничение бесплатной версии: для получения доступа к платформе нужно подключить достаточно дорогую подписку;
- возрастные ограничения: платформа больше подходит для учеников младших классов, для старших же и средних обучение будет слишком легким;
- минималистичный интерфейс: достаточно мало инструментов для взаимодействия с редактором кода.

Разрабатываемый обучающий портал будет отличаться от аналогичных продуктов доступностью в использовании, достаточно информативным описанием каждой команды и операции, с которой будет взаимодействовать пользователь, дружелюбным интерфейсом и постоянной поддержкой для дальнейшего развития проекта.

1.3 Анализ требований к обучающему portalу

В обучающем портале должна быть возможность:

- создать аккаунт;

- авторизоваться через email и пароль;
- выбрать уровень;
- вести диалог;
- взаимодействовать с редактором кода;
- взаимодействовать с компилятором;
- выйти из аккаунта;
- сбросить пароль.

Создание аккаунта осуществляется на странице регистрации, на которой нужно будет ввести свой email и придуманный пароль в соответствующие текстовые поля. Вход зарегистрированного пользователя осуществляется на странице авторизации, где нужно ввести свои email и пароль, если они верны, то происходит переход на страницу выбора уровней с сохраненным прогрессом пользователя.

На странице выбранного уровня пользователь может взаимодействовать с редактором кода, компилятором, а также с персонажем-кораблем для ведения диалога. В диалоге пользователь может узнать всю интересующую его информацию о текущем уровне, например задание, которое необходимо выполнить. В компиляторе может пошагово отследить действия скомпилированной программы. В редакторе кода пользователь может добавить команду, выделить ее, удалить, перенести на другую строчку или узнать в подсказке, что данная команда из себя представляет, а также полностью очистить поле с кодом.

Диаграмма вариантов использования (англ. use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей (актеров) [1].

На основе сформулированных требований к приложению была разработана диаграмма вариантов использования, которая представлена ниже на рисунке 4.



Рисунок 4 – Диаграмма вариантов использования

Актер, представленный на диаграмме вариантов использования – пользователь приложения.

Далее описаны основные варианты использования для актёра «Пользователь приложения».

В таблице 1 приведено описание варианта использования «Выбрать уровень».

Таблица 1 – Вариант использования «Выбрать уровень»

Вариант использования	Выбрать уровень
ID	1
Краткое описание	Пользователь выбирает уровень, который он хочет пройти
Главные актёры	Пользователь
Второстепенные актёры	Нет
Предусловия	Нет
Основной поток	Пользователь выбирает уровень
Постусловия	Пользователь перешел в конкретный уровень

В таблице 2 приведено описание следующего варианта использования «Вести диалог».

Таблица 2 – Вариант использования «Вести диалог»

Вариант использования	Вести диалог
ID	2
Краткое описание	Пользователь ведет диалог с персонажем
Главные актёры	Пользователь
Второстепенные актёры	Нет
Предусловия	Нет
Основной поток	1. Пользователь кликает на корабль; 2. Пользователь может узнать задание; 3. Пользователь может узнать доп информацию; 4. Пользователь может завершить диалог.
Постусловия	Пользователь провел диалог

В таблице 4 приведено описание следующего варианта использования «Взаимодействовать с редактором кода».

В таблице 3 приведено описание следующего варианта использования «Взаимодействовать с компилятором».

Таблица 3 – Вариант использования «Взаимодействовать с компилятором»

Вариант использования	Взаимодействовать с компилятором
ID	3
Краткое описание	Пользователь использует компилятор
Главные актёры	Пользователь
Второстепенные актёры	Нет
Предусловия	Нет
Основной поток	1. Пользователь может скомпилировать код программы; 2. Пользователь может сделать шаг назад; 3. Пользователь может сделать шаг вперед; 4. Пользователь может перезапустить программу;
Постусловия	Пользователь использовал компилятор

Таблица 4 – Вариант использования «Взаимодействовать с редактором кода»

Вариант использования	Взаимодействовать с редактором кода
ID	4
Краткое описание	Пользователь использует редактор кода
Главные актёры	Пользователь
Второстепенные актёры	Нет
Предусловия	Нет
Основной поток	1. Пользователь может добавить команду; 2. Пользователь может удалить команду; 3. Пользователь может выделить команду; 4. Пользователь может перенести строчку кода; 5. Пользователь может очистить редактор кода; 6. Пользователь может посмотреть подсказку о команде.
Постусловия	Пользователь использовал редактор кода

Итак, выше были рассмотрены все основные виды вариантов использования, необходимые для проектирования обучающего интернет-портала. С остальными вариантами использования можно ознакомиться в приложении А.

Нефункциональные требования описывают характеристики системы, которые не связаны непосредственно с ее функциональностью, но влияют на ее общую производительность, безопасность, надежность, поддерживаемость и другие аспекты [2].

В таблице 5 будут подробно приведены минимальные требования к системе.

Таблица 5 – Системные требования

Параметр	Требование
ОС	Windows 10 и выше. Ubuntu 14.04 и старше
Процессор	Intel Pentium 4 или более поздней версии
Свободное место на диске	2 Гб
Оперативная память	2 Гб

Исходя из вышеперечисленных требования к системе можно сказать, что будущий обучающий интернет-портал будет довольно не требовательным, а значит позволить использовать его для обучения алгоритмизации может почти каждый современный школьник.

1.4 Постановка задачи

В результате анализа приложений и веб-сайтов, обучающих алгоритмизации, было принято решение о необходимости разработать обучающий интернет-портал для обучения алгоритмизации при помощи визуально-блочного программирования. У пользователя будет доступно несколько уровней с разными задачами, командами, операциями и с уникальным визуальным оформлением.

Каждый последующий уровень будет открываться путем успешного прохождения предыдущего. В каждом уровне пользователю будет предоставлен компилятор, редактор кода и игровое поле с видом сверху, на котором будет изображен персонаж, выполняющий команды из редактора кода, и различные предметы и объекты, с которыми пользователю можно взаимодействовать в ходе прохождения уровня.

Обучающий портал должен иметь интуитивно понятный пользовательский интерфейс, обновляющийся при действиях пользователя, а также текстовые подсказки, поясняющие отдельные его части. Текстовые подсказки помогут пользователям быстро ориентироваться в задачах конкретного уровня и доступных командах и операциях. Все данные учетной записи должны сохраняться в базу данных для того, чтобы пользователь мог

видеть свой прогресс и продолжать обучение с того уровня, на котором он остановился.

Целью данной работы является разработка обучающего портала, который поможет в обучении алгоритмизации при помощи визуально-блочного программирования.

Для достижения поставленной цели требуется решить следующие задачи:

- выполнить анализ предметной области;
- выполнить анализ приложений с аналогичной функциональностью;
- определить требования к разрабатываемой информационной системе;
- разработать диаграмму вариантов использования;
- выбрать технологии для разработки;
- спроектировать архитектуру приложения;
- разработать диаграмму компонентов;
- разработать диаграммы классов;
- спроектировать графический интерфейс приложения;
- выполнить оценку бюджета разработки;
- разработать диаграмму Ганта;
- реализовать спроектированное приложение;
- протестировать разработанное приложение.

В результате выполнения вышеперечисленных задач должен быть выполнен анализ по проведённой работе и сделаны выводы о достижении поставленной цели.

2 Проектирование обучающего портала

2.1 Выбор технологий проектирования

В ходе проектирования обучающего портала были использованы Visual Paradigm CE, Draw.io, Online Gantt, потому что все эти программы доступны бесплатно и позволяют создавать диаграммы нотаций UML. Visual Paradigm привлекает своей привычностью и простотой установки. Draw.io – простотой и удобностью интерфейса. Online Gantt обеспечивает функциональность для построения диаграммы, что позволяет эффективно спланировать весь цикл разработки ПО.

Для проектирования интерфейса была выбрана онлайн версия FigJam из-за ее доступности и наличия необходимых инструментов для построения макетов интерфейса. Также FigJam позволяет совместно работать над проектом в реальном времени, что способствует лучшей командной работе.

Таким образом, для проектирования были выбраны доступные версии инструментов, которые позволяют строить UML-диаграммы и проектировать будущей интерфейс обучающего портала.

2.2 Проектирование архитектуры

Согласно поставленным задачам, становится понятно, что необходимо разработать микросервисную архитектуру, чтобы система была легко масштабируемой и каждый ее уровень не зависел от предыдущего. Например, сам портал можно будет расширить до обучения конкретного языка программирования. Не потребуются сложное встраивание его в единую архитектуру.

Для визуализации компонентов, входящих в состав приложения, можно использовать диаграмму компонентов, построенную с помощью UML-средств. Она позволяет более четко представить структуру системы и ее архитектуру.

Диаграмма компонентов представляет архитектуру веб-приложения, отображая связи и зависимости между различными файлами и модулями.

Файлы HTML представляют из себя конкретные страницы, которые включают множество своих индивидуальных модулей JavaScript, CSS, файлов.

На рисунке 5 изображена диаграмма компонентов обучающего портала.

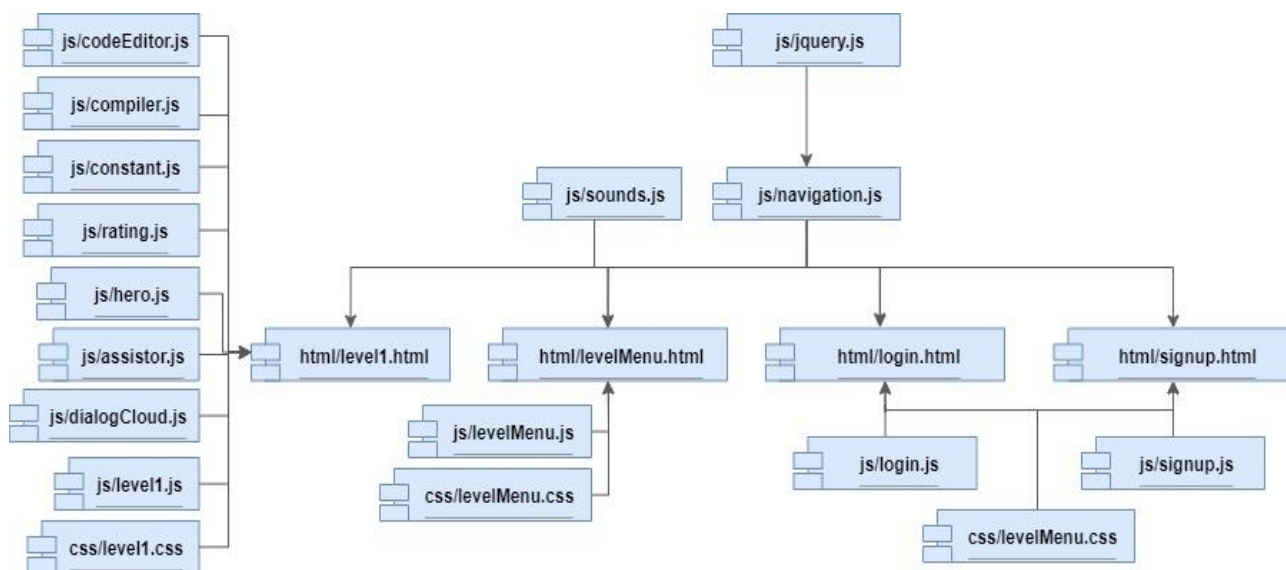


Рисунок 5 – Диаграмма компонентов

То есть можно сказать, что диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода [3].

Далее разберем компоненты серверной части. На рисунке 6 представлена диаграмма компонентов серверной части обучающего портала.

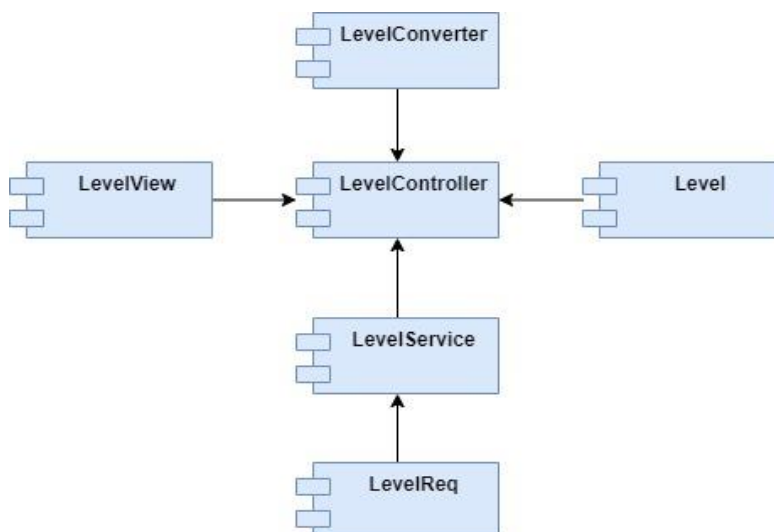


Рисунок 6 – Диаграмма компонентов

Диаграмма компонентов серверной части интернет-портала отображает взаимодействие между различными модулями, организованными по принципам MVC (Model-View-Controller). «LevelView» отвечает за представление данных пользователю и отправляет запросы к контроллеру. «LevelController» обрабатывает эти запросы, взаимодействуя с моделью «Level» для получения или изменения данных, и передает их в представление. «LevelService» обрабатывает бизнес-логику приложения, используя данные, полученные от «LevelReq», который взаимодействует с внешними сервисами или базой данных. «LevelConverter» преобразует данные между различными форматами, обеспечивая правильное представление информации.

Использование диаграммы компонентов в процессе проектирования архитектуры системы позволяет наглядно представить структуру приложения, выявить и описать взаимосвязи между его компонентами, что значительно упрощает процесс разработки и поддержки программного обеспечения.

Следующая диаграмма, построенная для визуализации процесса составления кода программы, компиляции и вывода результата оценивания, – это диаграмма деятельности. Данная диаграмма представлена на рисунке 7.

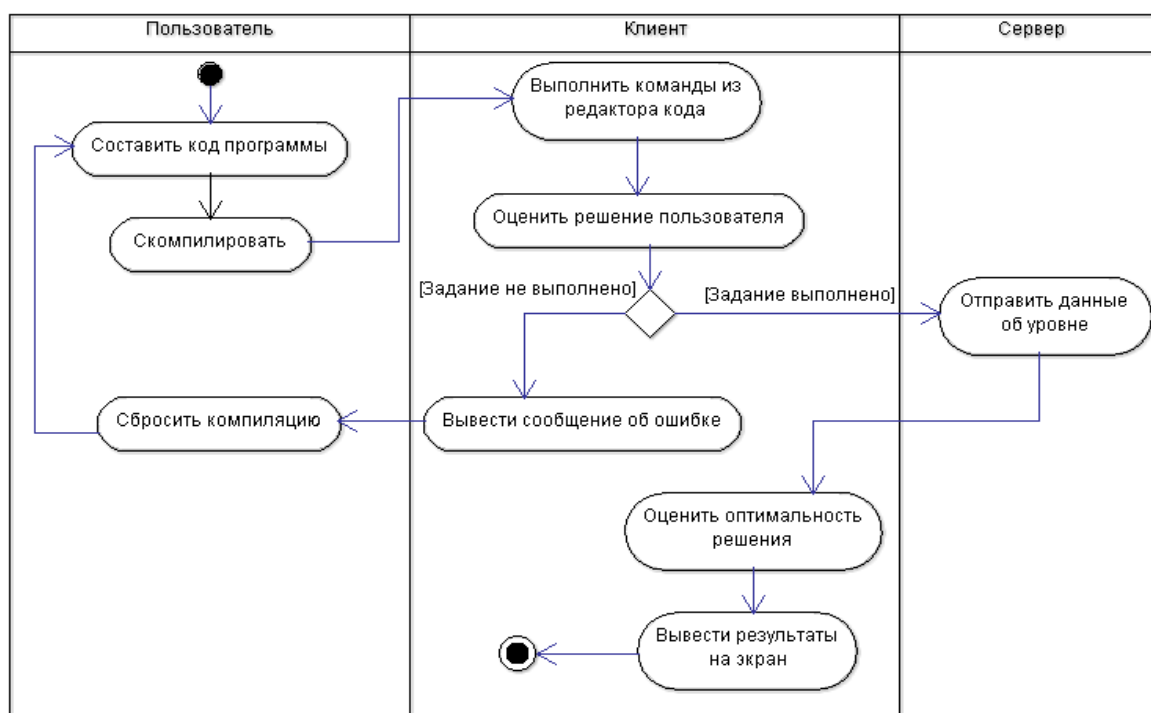


Рисунок 7 – Диаграмма деятельности

На диаграмме изображен процесс выполнения программного кода пользователем и проверка его сервером через клиентское приложение.

Пользователь составляет и компилирует код программы. Клиентское приложение выполняет команды из редактора кода и оценивает решение пользователя. Если задание не выполнено, выводится сообщение об ошибке, и пользователь может сбросить компиляцию, чтобы попробовать снова. Если задание выполнено, данные об уровне отправляются на сервер. Затем оценивается оптимальность решения, и результаты выводятся на экран.

Таким образом, визуализация архитектуры с помощью UML-диаграмм позволяет четко представить структуру приложения, выявить взаимосвязи между компонентами и упростить процесс масштабирования и интеграции новых функций.

2.3 Проектирование графического интерфейса

Графический интерфейс пользователя – одна из задач при создании обучающего интернет-портала. Он является частью пользовательского интерфейса и определяет взаимодействие с пользователем на уровне визуализированной информации. Поэтому важно спроектировать интерфейс интуитивно понятным и современным для того, чтобы пользователь мог быстро в нем ориентироваться.

Проектирование графического интерфейса – создание способа взаимодействия пользователя с элементами системы.

Общие принципы проектирования пользовательских интерфейсов:

- интерфейс должен быть интуитивно понятным, таким, чтобы пользователю не требовалось объяснять, как им пользоваться;
- чаще всего, пользователи в интерфейсе сначала ищут сущность (существительное), а затем действие (глагол) к ней;
- чем быстрее человек увидит результат – тем лучше;
- следует с осторожностью предоставлять пользователю возможность, по установке личных настроек;

– чем больше пользователь работает с какой-то конкретной задачей, тем больше он на ней концентрируется и тем меньше перестает замечать подсказки и сообщения, выводимые программой [4].

Все элементы графического интерфейса должны быть выполнены в современном стиле и иметь интуитивно понятные и привлекательные элементы дизайна, что позволит пользователям легко и быстро ориентироваться в образовательном портале.

В результате проектирования были созданы следующие макеты экранов.

На рисунке 8 изображён макет экрана входа.

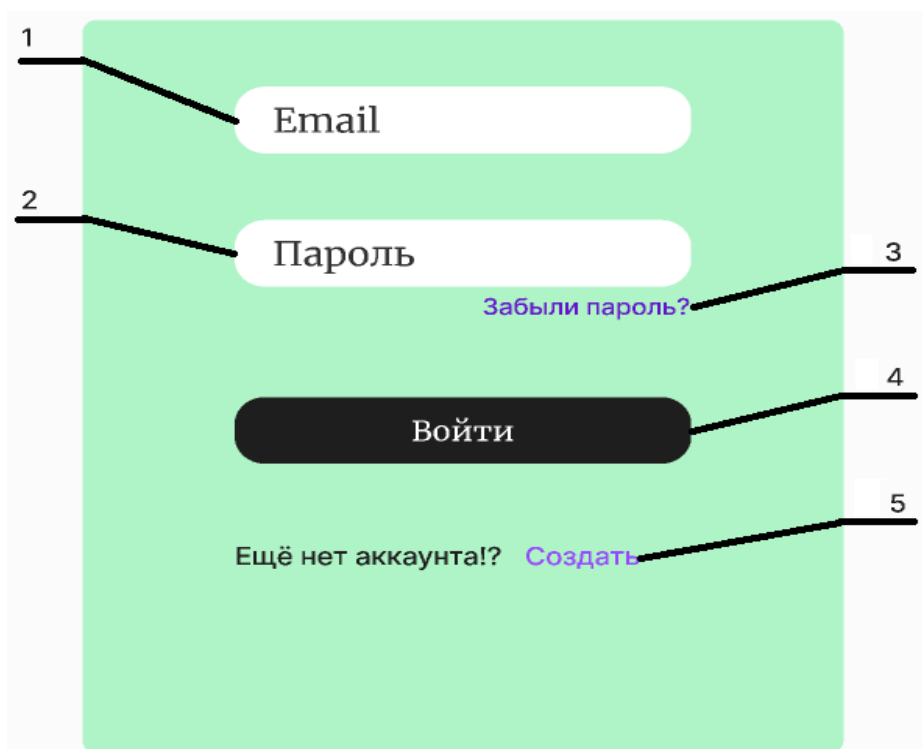


Рисунок 8 – Макет окна авторизации

Макет экрана входа содержит следующие элементы:

- 1 – поле ввода email;
- 2 – поле ввода пароля;
- 3 – кнопку «Забыли пароль?» для перехода на экран сброса пароля;
- 4 – кнопку «Войти» для входа в приложения;
- 5 – кнопку «Создать» для перехода на экран создания аккаунта;

На рисунке 9 изображён макет экрана регистрации.

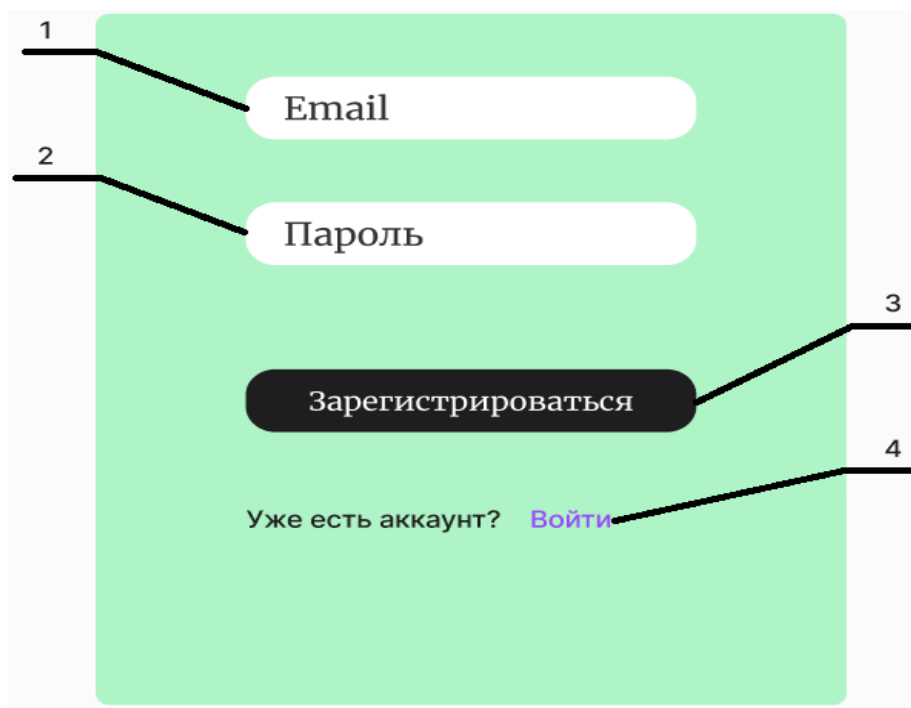


Рисунок 9 – Макет окна регистрации

Макет экрана регистрации содержит следующие элементы:

- 1 – поле ввода email;
- 2 – поле ввода пароля;
- 3 – кнопку «Зарегистрироваться» для создания аккаунта;
- 4 – кнопку «Войти» для перехода на экран входа.

На рисунке 10 изображён макет экрана создания тренировки.



Рисунок 10 – Макет окна выбора уровня

Макет экрана создания тренировки содержит уровни (уроки), которые может выбирать пользователь.

На рисунке 11 изображён макет экрана создания упражнения.

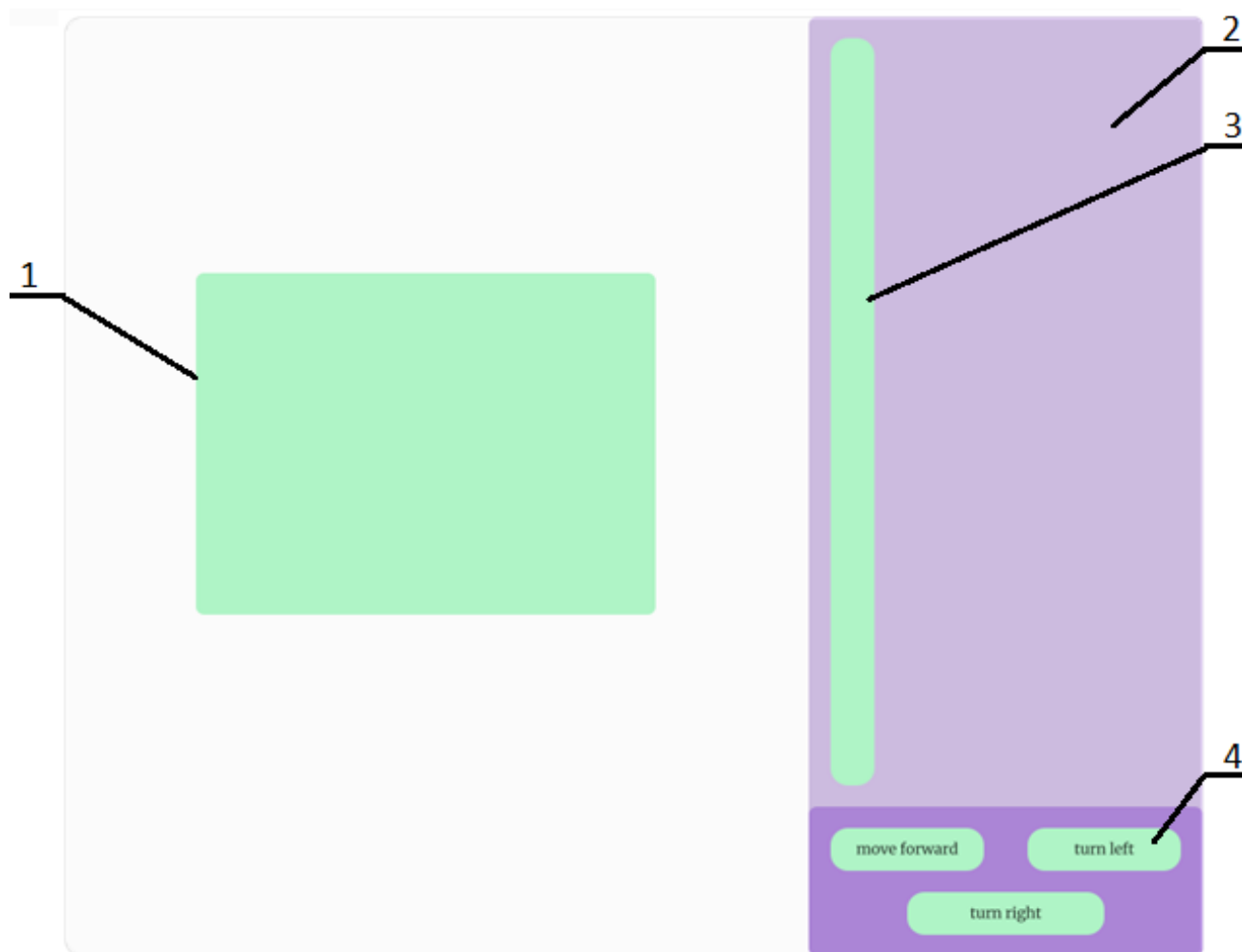


Рисунок 11 – Макет окна уровня

Макет экрана создания тренировки содержит следующие элементы:

- 1 – игровое поле, на котором персонаж будет визуально выполнять команды после компиляции кода;
- 2 – область редактора кода;
- 3 – область нумерации строчек кода;
- 4 – кнопки «turn left», «turn right», «move forward», при нажатии на которые будут добавляться строчки кода в редактор.

Таким образом, были спроектированы все страницы обучающего портала. В дальнейшем на основе этих макетов будет разработан пользовательский интерфейс.

2.4 Оценка бюджета разработки обучающего портала

Одним из ключевых аспектов успешного проектирования и дальнейшей разработки проекта, а особенно обучающего интернет-портала, является грамотное распределение бюджета. Бюджет проекта влияет на все стадии создания интернет-портала – от анализа требований к нему до финальных этапов тестирования и внедрения. Поэтому рассмотрим более подробно от чего зависит бюджет:

- масштабность проекта: от этого зависят время и ресурсы, требуемые для разработки;
- опыт разработчиков: опытные специалисты требуют больше финансовых затрат;
- дизайн: более современный, стилистически приятный дизайн требуют вложений в опытных дизайнеров;
- тестирование: проведение своевременного и качественного тестирования дополнительно требует финансовых и временных затрат;
- техническая поддержка и обслуживание: после запуска портала потребуются ресурсы на его дальнейшую поддержку [5].

Средняя стоимость разработки обучающего интернет-портала может изменяться в зависимости от вышеперечисленных аспектов. Но для каждого проекта проводится индивидуальную оценку бюджета для точных расчетов.

Для визуализации всех этапов разработки приложения и оценки затрат времени и ресурсов на каждый этап было принято решение разработать диаграмму Ганта при помощи вебсайта «Free online gantt». В таблице 6 указаны средние зарплаты специалистов за месяц и за час.

Таблица 6 – Доходы работников

Профессия	Средняя зарплата за месяц	Средняя зарплата за час (округлённо до 50 Р/ч)
Аналитик	80 000 Р	500 Р/ч
Дизайнер	70 000 Р	450 Р/ч

Продолжение таблицы 6

Профессия	Средняя зарплата за месяц	Средняя зарплата за час (округлённо до 50 Р/ч)
Программист	100 000 Р	650 Р/ч
Тестировщик	90 000 Р	600 Р/ч

На основании стоимости необходимого оборудования и доходов работников была составлена таблица 7.

Таблица 7 – Ресурсы

Название ресурса	Тип	Стандартная ставка	Начисление
Аналитик	Трудовой	500 Р/ч	Пропорциональное
Дизайнер	Трудовой	450 Р/ч	Пропорциональное
Программист	Трудовой	650 Р/ч	Пропорциональное
Тестировщик	Трудовой	600 Р/ч	Пропорциональное
Компьютер	Материальный	50 000,00 Р	В начале

На основании ресурсов и оценки сложности планируемых при проектировании работ были выделены задачи и время на их выполнение. Все это указано в таблице 8.

Таблица 8 – Задачи проекта

Название задачи	Длительность
Начало проекта	-
Анализ обучающих порталов	9 дней
Анализ предметной области «Интернет порталы для обучения школьников»	3 дня
Анализ аналогичных обучающих интернет порталов	3 дня

Продолжение таблицы 8

Название задачи	Длительность
Анализ требований к обучающему portalу	2 дня
Постановка задачи	1 день
Проектирование обучающего portalа	5 дней
Проектирование архитектуры portalа	3 дня
Проектирование графического интерфейса	1 день
Оценка бюджета разработки обучающего portalа	1 день
Разработка и реализация интернет portalа по обучению школьников алгоритмизации с использованием визуального программирования	14 дней
Разработка слоя модели данных интернет portalа	3 дня
Разработка слоя бизнес-логики интернет portalа	6 дней
Разработка слоя представления интернет portalа	5 дней
Тестирование программных средств	6 дней
Тестирование методом белого ящика	2 дня
Тестирование метрикой Холстеда	1 день
Тестирование метрикой SLOC	1 день
Автоматизированное тестирование	2 дня
Подготовка документации	4 дня
Написание документации	2 дня
Оформление и печать документации	2 дня
Конец проекта	-

На основании задач и затрачиваемого на них времени была разработана диаграмма «Ганта», приведённая на рисунке 12.

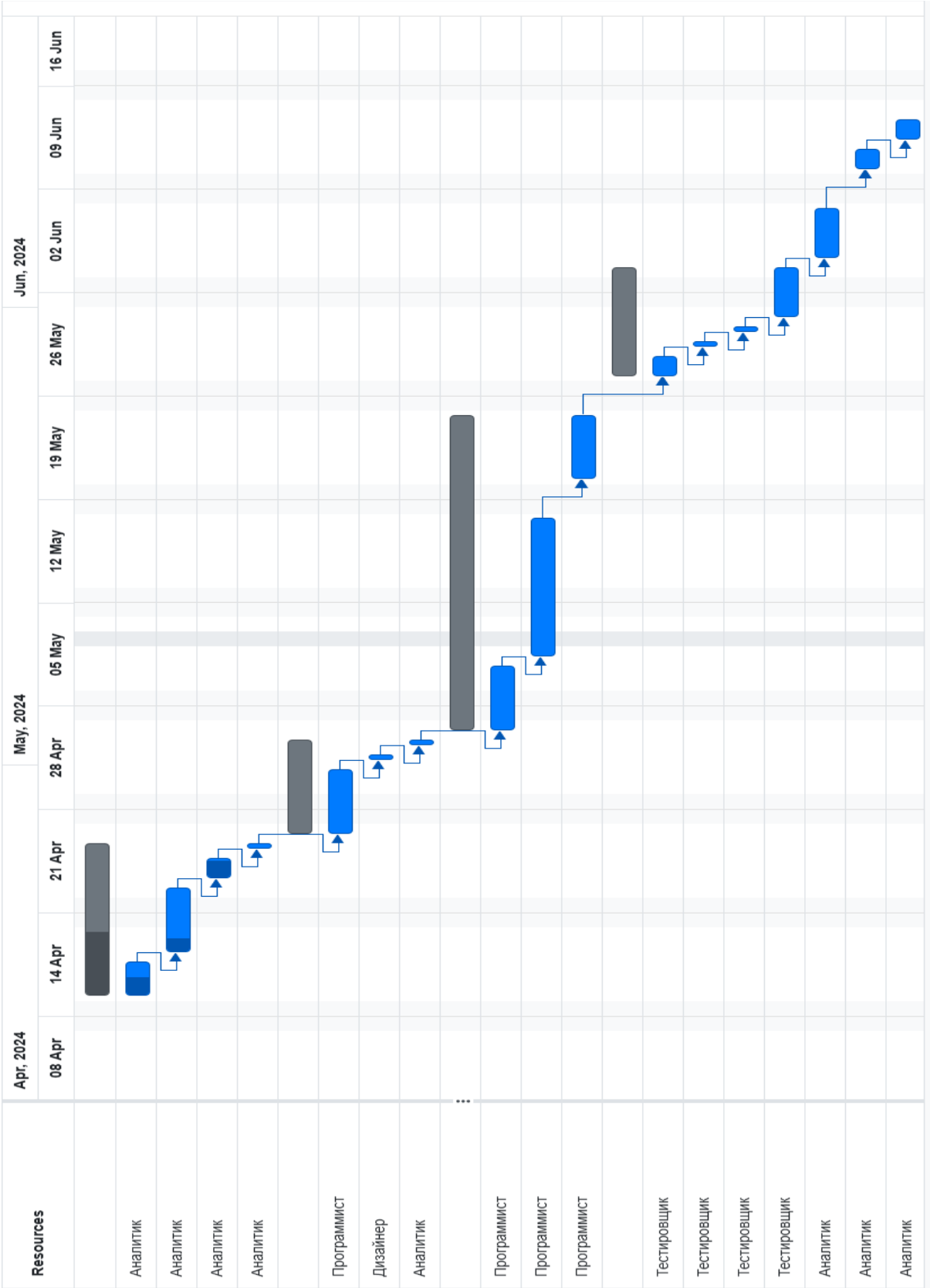


Рисунок 12 – Диаграмма «Ганта»

Таким образом, по результатам оценки бюджета разработки мобильного приложения была составлена диаграмма «Ганта», рассчитано время необходимое на полный цикл разработки проекта, которое составляет 42 дня, включая праздничные дни и выходные. А на основании необходимых ресурсов рассчитан бюджет проекта, который равен 220 900 рублей.

3 Разработка и реализация интернет-портала по обучению школьников алгоритмизации при помощи визуального программирования

3.1 Выбор технологий разработки

Для разработки интернет-портала был выбран язык программирования JavaScript, который позволяет создать динамичные и интерактивные веб-страницы. Это преимущество упрощает разработку, так как JavaScript широко поддерживается всеми современными веб-браузерами и имеет богатую экосистему библиотек. Использование чистого JavaScript без дополнительных фреймворков даёт возможность гибко и точно контролировать весь процесс разработки и адаптировать код под специфические требования проекта [6].

Для локального хранения данных портала была выбрана реляционная база данных PostgreSQL из-за её высокой производительности, масштабируемости и поддержки сложных запросов и транзакций. PostgreSQL также известна своей надёжностью и соответствием стандартам SQL, что делает её идеальным выбором для веб-приложений с большими объемами данных и сложными бизнес-логиками [7].

Для серверной части портала был выбран язык программирования Java с использованием фреймворка Spring. Java обеспечивает высокую производительность, безопасность и возможность масштабирования приложений, в частности серверных. Благодаря JVM (Java Virtual Machine) приложения на Java могут быть запущены на любой операционной системе, что делает её кроссплатформенной и гибкой для различных инфраструктур. Использование Spring ускоряет разработку и упрощает управление зависимостями и конфигурациями серверных приложений, предоставляя мощные инструменты для создания надёжных и масштабируемых серверных решений [8].

3.2 Разработка архитектуры обучающего портала

В рамках разработки серверной части интернет-портала было проведено следующее:

- реализована архитектура «MVC» (Model-View-Controller), что позволяет эффективно организовать взаимодействие между различными компонентами;
- написаны классы `LevelView`, `LevelController`, `LevelService`, `LevelReq` и `LevelConverter` и аналогичные классы для микросервиса аутентификации с учетом требований функциональности портала;
- реализованы методы в каждом из этих классов для выполнения конкретных задач, таких как обработка запросов от пользователя, взаимодействие с базой данных, обработка бизнес-логики и преобразование данных;
- внедрена возможность взаимодействия с сервисами между собой, например, чтобы получать данные об авторизованном пользователе;
- проведено тестирование каждого компонента на соответствие требованиям и правильную работу в рамках всей системы.

Поподробнее рассмотрим файлы, которые из себя представляют принцип «Model», «View», «Controller», на примере микросервиса `Level` [9].

Файл `Level`, представляющий из себя «Model», хранит описание методов для работы с данными. Фрагмент кода файла представлен на листинге 1.

Листинг 1 – Фрагмент кода файла `Level`

```
package com.core.level;
import org.hibernate.annotations.GenericGenerator;
import javax.persistence.*;

@Entity
@Table(name = "my_level")
public class Level {
    @Id
    @Column(name = "id")
    @GenericGenerator(
        name = "level_id_seq",
```

```

        strategy =
"org.hibernate.id.enhanced.SequenceStyleGenerator",
        parameters = {
            @org.hibernate.annotations.Parameter(name =
"sequence_name", value = "level_id_seq"),
            @org.hibernate.annotations.Parameter(name=
"INCREMENT", value = "1"),
            @org.hibernate.annotations.Parameter(name =
"MINVALUE", value = "1"),
            @org.hibernate.annotations.Parameter(name =
"MAXVALUE", value = "2147483647"),
            @org.hibernate.annotations.Parameter(name = "CACHE",
value = "1")
        }
    )
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"level_id_seq")
    private Integer id;
    @Column(name="name")
    private String name;
    @Column(name="optimal_length")
    private int optimalLength;

    public Integer getId() {return id;}
    public void setId(Integer id) {this.id = id;}

    public String getName() {return name;}
    public void setName(String name) {this.name = name;}
    public int getOptimalLength() {return optimalLength;}
    public void setOptimalLength(int optimalLength) {this.optimalLength =
optimalLength;}
}

```

Класс LevelView, предназначен для отображения данных пользователю.

На листинге 2 представлен код данного файла.

Листинг 2 – Фрагмент кода файла LevelView

```

package com.core.level.web;

public class LevelView {
    private Integer id;
    private String name;
    private int optimalLength;

    public Integer getId() {return id;}
    public void setId(Integer id) {this.id = id;}

    public String getName() {return name;}
    public void setName(String name) {this.name = name;}

    public int getOptimalLength() {return optimalLength;}
    public void setOptimalLength(int optimalLength) {this.optimalLength =
optimalLength;}
}

```

Класс `LevelController`, отвечает за обработку HTTP-запросов, связанных с сущностью «Level» включая создание, получение, обновление и удаление уровней, а также преобразование данных и локализацию сообщений об ошибках. Код данного файла представлен на листинге 3.

Листинг 3 – Фрагмент кода файла `LevelController`

```
package com.core.level.web;
import com.core.level.Level;
import com.core.level.LevelConverter;
import com.core.level.LevelService;
import javax.servlet.http.HttpSession;
import javax.validation.Valid;
import java.util.Optional;

@RestController
@RequestMapping(value = "/level")
@CrossOrigin(origins = "http://localhost:8383")
public class LevelController {
    private final LevelService service;
    private final LevelConverter converter;
    private final MessageSource messageSource;

    @Autowired
    public LevelController(LevelService service,
                           MessageSource messageSource,
                           LevelConverter converter) {
        this.service = service;
        this.messageSource = messageSource;
        this.converter = converter;
    }

    @GetMapping("/{id}")
    @ResponseBody
    public LevelView getLevelById(@PathVariable Integer id) {
        Level level = service.getLevelById(id);
        checkNull(level, id);
        return converter.convert(level);
    }

    @GetMapping
    @ResponseBody
    public Page <LevelView> getAllLevel(@PageableDefault(sort = "id",
        direction = Sort.Direction.ASC) Pageable pageable) {
        return service.findAllLevel(pageable)
            .map(converter::convert);
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    @ResponseBody
    public LevelView createLevel(@RequestBody @Valid LevelReq req) {
        return converter.convert(service.createLevel(req));
    }
}
```

```

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteLevel(@PathVariable Integer id) {
        Level level = service.getLevelById(id);
        checkNull(level, id);
        service.deleteLevel(id);
    }

    @PutMapping("/{id}")
    public LevelView updateLevel(@PathVariable(name = "id") Integer id,
                                @RequestBody @Valid LevelReq req) {
        Level level = service.getLevelById(id);
        checkNull(level, id);
        return converter.convert(service.updateLevel(level, req));
    }

    private void checkNull (Level level, Integer id) {
        if (level == null) {
            String message = messageSource.getMessage(
                "com.core.level.web.LevelWithIdNotExists", new
Object[]{id},
                LocaleContextHolder.getLocale());
        }
    }
}

```

Класс `LevelReq` представляет собой DTO (Data Transfer Object) для передачи данных при создании или обновлении сущности «Level», содержащий обязательные поля `name` и `optimalLength` с соответствующими геттерами и сеттерами. Код данного файла представлен на листинге 4.

Листинг 4 – Фрагмент кода файла `LevelReq`

```

package com.core.level.web;
import javax.validation.constraints.NotEmpty;

public class LevelReq {
    @NotEmpty
    private String name;
    @NotEmpty
    private int optimalLength;
    public int getOptimalLength() {return optimalLength;}
    public void setOptimalLength(int optimalLength) {this.optimalLength =
optimalLength;}
    public String getName() {return name;}
    public void setName(String name) {this.name = name;}
}

```

Класс `LevelConverter` преобразует объекты `Level` в объекты `LevelView`, используя для этого метод `convert`, который заполняет поля `id`, `name` и

optimalLength в LevelView из соответствующих полей объекта Level. Код данного файла представлен на листинге 5.

Листинг 5 – Фрагмент кода файла LevelConverter

```
package com.core.level;
import com.core.level.web.LevelView;
import org.springframework.core.convert.converter.Converter;
import org.springframework.lang.NonNull;
import org.springframework.stereotype.Component;

@Component
public class LevelConverter implements Converter<Level, LevelView> {
    @Override
    public LevelView convert(@NonNull Level level) {
        LevelView view = new LevelView();
        view.setId(level.getId());
        view.setName(level.getName());
        view.setOptimalLength(level.getOptimalLength());
        return view;
    }
}
```

Класс LevelService управляет бизнес-логикой для сущности «Level», выполняя операции создания, получения, обновления и удаления уровней, а также обрабатывает пагинацию и локализацию сообщений об ошибках. Код данного файла представлен на листинге 6.

Листинг 6 – Фрагмент кода файла LevelService

```
package com.core.level;
import com.core.level.web.LevelReq;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

import javax.persistence.EntityNotFoundException;
import javax.transaction.Transactional;

@Service
public class LevelService {

    private final LevelRepo levelRepo;

    private final MessageSource messageSource;

    public LevelService(LevelRepo levelRepo, MessageSource
messageSource) {
        this.levelRepo = levelRepo;
        this.messageSource = messageSource;
    }
}
```

```

    }

    public Level getLevelById(Integer id) {
        return levelRepo.findById(id)
            .orElseThrow(() -> new
EntityNotFoundException(messageSource
.getMessage("com.core.level.web.LevelWithIdNotExists", new Object[]{id},
            LocaleContextHolder.getLocale())));
    }

    public Page<Level> findAllLevel(Pageable pageable) {
        return levelRepo.findAll(pageable);
    }

    public Level createLevel(LevelReq req) {
        Level level = new Level();
        this.prepareLevel(level, req);
        levelRepo.save(level);
        return level;
    }

    @Transactional
    public void deleteLevel(Integer id) {
        try {
            levelRepo.deleteById(id);
        } catch (EmptyResultDataAccessException e) {
            throw new EntityNotFoundException(messageSource
.getMessage("com.core.level.web.LevelWithIdNotExists", new Object[]{id},
            LocaleContextHolder.getLocale())));
        }
    }

    public Level updateLevel(Level level, LevelReq req) {
        this.prepareLevel(level, req);
        levelRepo.save(level);
        return level;
    }

    public void prepareLevel(Level level, LevelReq levelBaseReq) {
        level.setName(levelBaseReq.getName());
        level.setOptimalLength(levelBaseReq.getOptimalLength());
    }
}

```

Интерфейс `LevelRepo` является репозиторием Spring Data JPA для сущности `Level`, предоставляя методы для выполнения CRUD операций в базе данные. Код данного файла представлен на листинге 7.

Листинг 7 – Фрагмент кода файла `LevelRepo`

```

package com.core.level;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

```

```
@Repository
public interface LevelRepo extends JpaRepository<Level, Integer> {
}
```

Таким образом, разработанная серверная часть интернет-портала обеспечивает эффективное и надежное функционирование системы, а также удовлетворяет потребностям пользователей.

3.3 Разработка слоя представления приложения

При разработке графической части интернет-портала использовался язык JavaScript, который позволяет создавать динамические и интерактивные элементы на веб-страницах. JavaScript обеспечивает широкие возможности для создания различных эффектов, анимаций, обработки пользовательских действий и взаимодействия с сервером без необходимости перезагрузки страницы.

Таким образом, были разработаны несколько страниц, отображающие процессы регистрации, авторизации, выбора уровня и самого уровня.

При первом запуске интернет-портала пользователь видит страницу авторизации (рисунок 13).

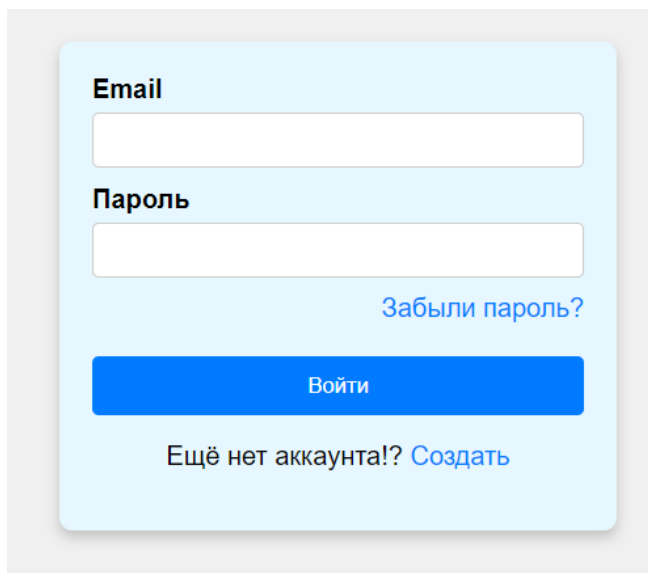
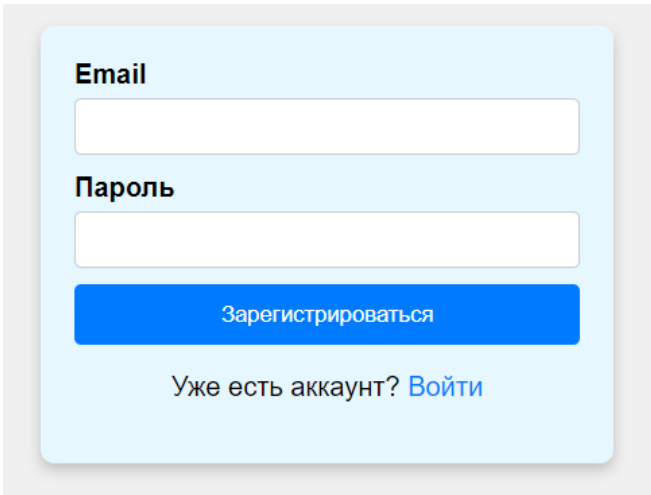


Рисунок 13 – Страница входа

Если пользователь ещё не имеет аккаунта, то он может перейти на страницу регистрации и создать его (рисунок 14).



The image shows a registration form on a light blue background. It contains two input fields: 'Email' and 'Пароль' (Password). Below the password field is a blue button with the text 'Зарегистрироваться' (Register). At the bottom, there is a link that says 'Уже есть аккаунт? Войти' (Already have an account? Log in).

Рисунок 14 – Страница регистрации

После авторизации загружается страница выбора уровней. Пользователь имеет возможность выбрать один из доступных ему уровней, представленные здесь в виде островов, выйти из аккаунта, щелкнув по значку профиля, и включить или выключить музыкальное сопровождение (рисунок 15).



Рисунок 15 – Страница выбора уровней

После выбора острова пользователь переходит на страницу конкретного уровня, в котором ему требуется составить код программы с помощью набора команд для выполнения задания интерактивным персонажем (рисунок 16).



Рисунок 16 – Страница уровня

Таким образом, при реализации слоя представления использовались встроенные виджеты, предоставляемые фреймворком Flutter, в соответствии с методологией Material, используемой в Android приложениях. Кроме того, были использованы сторонние графические библиотеки для реализации изогнутой панели навигации, анимированных иконок и расширяемой всплывающей кнопки.

4 Тестирование программных средств

4.1 Тестирование белым ящиком

В рамках ВКР было проведено тестирование методом белого ящика алгоритма оценивания пройденного уровня по следующим критериям покрытия кода:

- покрытие операторов;
- покрытие решений;
- покрытие маршрутов.

Ход и результаты тестирования представлены ниже [10].

4.1.1 Покрытие операторов

Покрытие операторов – относится к метрике, измеряющей покрытие операторов в тест-кейсах. То есть отвечает на вопрос, каждая ли строка кода была протестирована или нет, каждый ли оператор программы был выполнен хотя бы один раз [11].

Блок-схема алгоритма оценивания пройденного уровня представлена ниже на рисунке 17.

Для достижения выполнения каждого оператора в тестах не менее одного раза, было составлено три тест-кейса со значениями входных, выходных данных, указанными в таблице 9.

Таблица 9 – Тест-кейсы покрытия операторов

Операторы	Входные данные	Выходные данные
a, c, h, i.	Уровень не пройден	Сообщение: «Попробуйте ещё раз»
a, d, f, h, i.	Уровень пройден Количество строк кода	Сообщение: «Цель не достигнута!»
a, d, g, h, i.	Уровень пройден Количество строк кода	Сообщение: «Цель достигнута!»

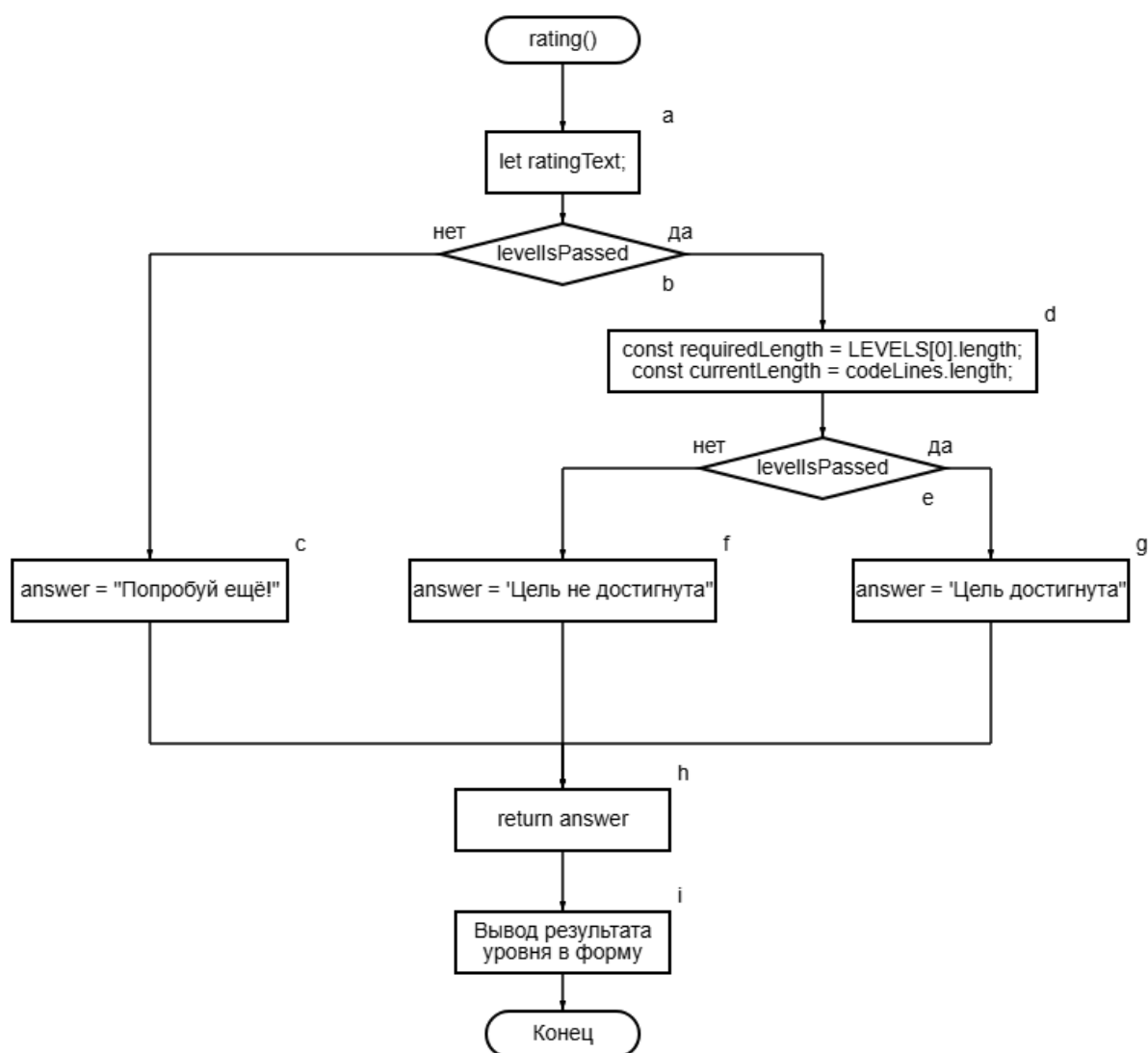


Рисунок 17 – Блок-схема алгоритма оценивания пройденного уровня

Таким образом, тестирование критерия покрытия операторов привело к положительным результатам. Также можно понять, что для покрытия всех операторов требуется несколько тест-кейсов.

4.1.2 Покрытие решений

Покрытие решений измеряет, насколько хорошо тесты охватывают все возможные решения, то есть ветвления, в программном коде. Покрытие решений же означает, что каждая логическая ветвь программы была выполнена в ходе тестов не менее одного раза [11].

В ходе тестирования этого критерия стояла задача написать достаточное количество тестов для получения всех возможных значений у ветвлений.

Чтобы выполнить каждую ветвь из каждой точки решения не менее одного раза, было составлено три тест-кейса со значениями входных, выходных данных, указанными в таблице 10.

Таблица 10 – Тест-кейсы покрытия решений

Операторы	Входные данные	Выходные данные
b – нет	Уровень не пройден	Сообщение: «Попробуйте ещё раз»
b – да, e – нет	Уровень пройден Количество строк кода	Сообщение: «Цель не достигнута!»
b – да, e – да	Уровень пройден Количество строк кода	Сообщение: «Цель достигнута!»

Таким образом можно понять, что тестирование критерия покрытия решений привело к положительным результатам и были пройдены все возможные ветвления у решений в исходном коде.

4.1.3 Покрытие маршрутов

Покрытие маршрутов критерий кода, который измеряет покрытие всех возможных ветвей из каждой точки решения и всех возможных операторов в исходном коде, которые были выполнены и протестированы [11].

В ходе тестирования этого критерия необходимо было написать достаточное количество тестов для получения всех возможных значений у ветвлений.

Чтобы выполнить каждую ветвь из каждой точки решения и каждый оператор хотя бы один раз, было составлено три тест-кейса со значениями входных данных, указанными в таблице 11.

Таблица 11 – Тест-кейсы покрытия маршрутов

Операторы	Входные данные	Выходные данные
a, b (нет), c, h, i.	Уровень не пройден	Сообщение: «Попробуйте ещё раз»
a, b (да), d, e (нет), f, h, i.	Уровень пройден Количество строк кода > n	Сообщение: «Цель не достигнута!»

Продолжение таблицы 11

Операторы	Входные данные	Выходные данные
a, b (да), d, e (да), g, h, i.	Уровень пройден Количество строк кода $\leq n$	Сообщение: «Цель достигнута!»

Таким образом, тестирование критерия покрытия маршрутов привело к положительным результатам и были покрыты все возможные ветви каждой точки решения и все возможные операторы в исходном коде.

4.2 Автоматизированное тестирование

Для автоматического тестирования были написаны автотесты, которые проверяют работу всего интернет-портала. При написании тестов использовался фреймворк Selenide, который предоставляет инструменты для создания тестов и проверки результатов [12].

Сборка и управление зависимостями выполнялись с помощью Gradle, что обеспечило удобство и гибкость в управлении проектом. Selenide в сочетании с Gradle позволили создать надежную и легко поддерживаемую систему автоматического тестирования для моего интернет-портала.

Автоматизированное тестирование проведено на примере работы всего интернет-портала, то есть выполнение авторизации и проверка работы уровня. Код файла уровня представлен в приложении Б. Код тест-кейсов представлен на листинге 8.

Листинг 8 – Код тест-кейсов проверки аутентификации и работы уровня

```
public class Tests extends BaseTest {
    @Test
    void checkAuth() {
        AuthPage authPage = new AuthPage();
        openAuthPage();

        authPage
            .checkElements()
            .signIn()
            .checkElements();
    }

    @Test
    void checkLevel() {
```

```

AuthPage authPage = new AuthPage();
openAuthPage();
authPage
    .checkElements()
    .signIn()
    .checkElements()
    .clickVolume(false)
    .clickVolume(true)
    .clickIsland(0)
    .checkElements()
    .checkCompilerBtns()
    .checkCommandsBtns();
}
}

```

Ниже на листинге 9 будет представлен код страницы уровня.

Листинг 9 – Код страницы автотеста уровня

```

package pages;
import base.BaseActions;
import com.codeborne.selenide.CollectionCondition;
import com.codeborne.selenide.ElementsCollection;
import com.codeborne.selenide.SelenideElement;
import static com.codeborne.selenide.Selenide.$;

public class Level1Page extends BaseActions {
    public static final String grampId = "gramp";
    public static final String robotId = "robot";
    public static final String mapId = "mapCanvas";
    public static final String compilerClass = "compiler";
    public static final String editorBlockId = "editorBlock";
    public static final String clearBtn = "clearButton";
    public static final String commandsId = "commads";
    public static final SelenideElement editor = $("#editor");
    public static final int countOfBtns = 5;

    public Level1Page checkElements() {
        search.findById(grampId);
        search.findById(robotId);
        search.findById(mapId);
        search.findByClass(compilerClass);
        search.findById(editorBlockId);
        return this;
    }

    public Level1Page checkCompilerBtns() {
        SelenideElement compiler = $(".compiler");
        ElementsCollection buttons = compiler.findAll("button");
        buttons.shouldHave(CollectionCondition.size(5));
        new MenuPage().clickVolume(false);
        new MenuPage().clickVolume(true);
        return this;
    }

    public Level1Page checkCommandsBtns() {

```

```

SelenideElement compiler = $("#commands");
ElementsCollection buttons = compiler.findAll("button");
buttons.shouldHave(CollectionCondition.size(5));

if (!buttons.get(0).getText().equals("move forward") ||
    !buttons.get(1).getText().equals("turn left") ||
    !buttons.get(2).getText().equals("turn right"))
    System.out.println("Error, commands name");

checkCommand(buttons.get(0));
checkCommand(buttons.get(1));
checkCommand(buttons.get(2));
return this;
}

public void checkCommand(SelenideElement btn) {
    btn.click();
    if (!btn.getText().equals(editor.getText()))
        System.out.println("Wrong text - " + btn.getText());

    $("#clearButton").click();
    if (!editor.getText().isEmpty())
        System.out.println("Error with clearBtn");
}
}

```

Исходный код других классов автотестов приведен в приложении Б. Результаты тестирования всех тест-кейсов представлены на рисунке 18.

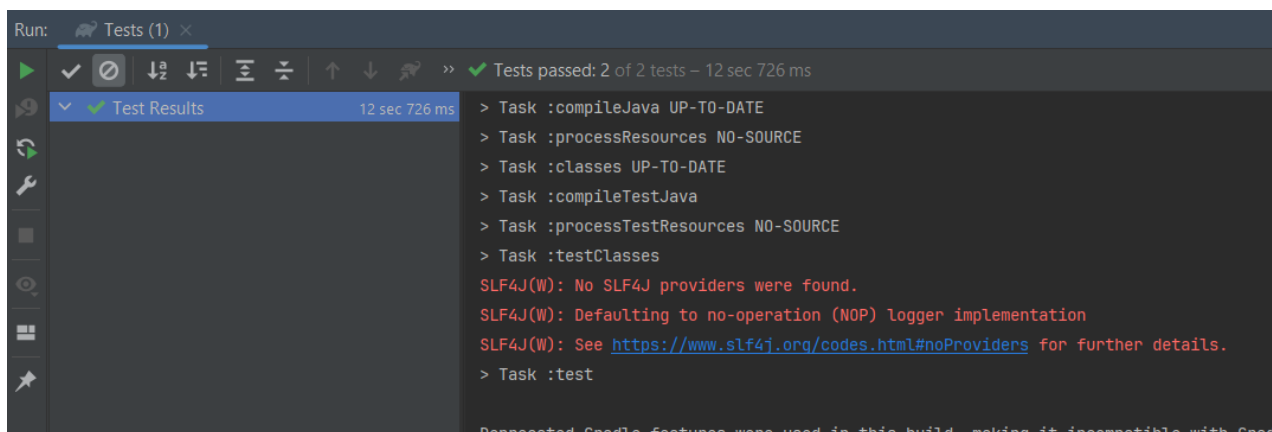


Рисунок 18 – Результат автоматизированного тестирования интернет-портала

Исходя из положительных результатов прохождения всех автотестов, можно сделать вывод о корректности работы обучающего портала.

4.3 Метрика SLOC

Метрика SLOC представляет собой метрику количества строк кода. Подсчет данной метрики был проведен для модуля, отвечающего за оценивание пройденного уровня [13]. Код модуля приведен в приложении Б.

В данной программе имеем:

- 95 физических строк кода;
- 68 логических строк кода;
- 4 строки комментариев.

Таким образом, 68 логических строк кода более полно отражают размер программы, так как исключают пустые строки и комментарии.

4.4 Метрика Холстеда

Для страницы отображения рейтинга были подсчитаны метрики Холстеда. Метрики Холстеда относятся к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы. То есть была произведена оценка характеристик программы [13].

Для расчёта метрик Холстеда были взяты функции, отвечающего за оценку пройденного уровня. Код данного модуля представлен в приложении Б.

Количество используемых операндов и операторов, а также их общее число в программе приведены в таблицах 12 и 13 соответственно.

Таблица 12 – Операторы

Операторы	Число вхождений
=	49
>=	1
+=	6
document.getElementById	1
document.querySelector()	2
document.querySelectorAll()	1
appendChild()	1
removeChild()	2

Продолжение таблицы 12

Операторы	Число вхождений
style.left	1
style.background	1
style.zIndex	2
style.pointerEvents	1
addEventListener	2
.	53
if	2
else	1
function()	2
cells.find()	1
;	47
document.createElement	1
id	1
return	4
length	2
style.width	3
style.height	1
style.position	1
style.top	1
n1=27	N1=190

Таблица 13 – Операнды

Операнды	Число вхождений
widgetBlock	9
rating()	1
CreateDiv	6
«»	15
«widget-container»	1
«h2»	1
levelsNames	1
sectionVolume	4
volumeTitle	1

Продолжение таблицы 13

Операнды	Число вхождений
volumeText	1
rateSolution()	2
CreateLabel	3
«р»	1
«Испытание Объёма»	8
currentLength	4
requiredLength	4
ratingText	8
«editor»	1
«.code-line»	1
getRequiredText	2
getCurrentText	2
getYESText	1
getNOText	1
LEVELS_LENGTH	1
progressBlock	3
currentRate	2
requiredRate	2
currProcent	1
reqProcent	1
buttonsBlock	3
backButton	2
«result-bar»	1
«progress-bar»	1
«remaining-bar»	1
«button-container»	1
«back-button»	1
«click»	2
«#overlay»	1
nextButton	2
«Пройти к островам»	1

Окончание таблицы 13

Операнды	Число вхождений
«next-button»	1
ToIslands()	1
displayOff()	2
getCellType()	3
isExit()	1
isDeath()	1
Cell	5
Cells	1
Row	5
robot.position	4
overlay	11
«div»	1
«fixed»	1
«100%»	2
«rgba(0, 0, 0, 0.5) »	1
«1»	1
«#widget-container»	1
«auto»	1
col	5
n2=62	N2=155

Длина программы рассчитывается по формуле:

$$N = N1 + N2 = 190 + 155 = 345.$$

Словарь программы рассчитывается по формуле:

$$n = n1 + n2 = 27 + 62 = 89.$$

Объем программы рассчитывается по формуле:

$$H_V = N * \log_2(n) = 345 * 6,48 = 2234,13.$$

Сложность программы рассчитывается по формуле:

$$HD = (n1 / 2) * (N2 / n2) = (27 / 2) * (155/62) = 33,75.$$

Полученные показатели объема и сложности программы показывают, что код данного класса имеет среднюю сложность из-за большого количества уникальных операторов и операндов.

В рамках проведенного тестирования белым ящиком алгоритма оценивания пройденного уровня и тестирования автотестами на основе фреймворка Selenide, метрикой Холстеда и метрикой SLOC были успешно проведены. Отсутствие выявленных недостатков в процессе тестирования свидетельствует о правильности работы программы.

Заключение

В ходе выполнения преддипломной практики был проанализирован и спроектирован интернет-портал по обучению школьников алгоритмизации при помощи визуального программирования. Были выполнены следующие задачи:

- анализ предметной области;
- анализ аналогичных обучающих порталов;
- анализ требований к обучающему portalу;
- постановка задачи;
- проектирование архитектуры;
- проектирование графического интерфейса;
- оценка бюджета разработки;
- выбор технологии разработки;
- разработка архитектуры обучающего портала;
- разработка слоя представления приложения;
- тестирование белым ящиком;
- автоматизированное тестирование;
- метрика SLOC;
- метрика Холстеда.

В результате преддипломной практики был разработан и протестирован интернет-портал для обучения школьников алгоритмизации с помощью визуального программирования. Проведен анализ предметной области и аналогичных порталов, сформулированы требования, спроектирована архитектура и графический интерфейс, выбраны технологии разработки. Реализованы архитектура и слой представления приложения, проведено тестирование белым ящиком и автоматизированное тестирование. Применены метрики SLOC и Холстеда для оценки кода.

Список использованных источников

1. Использование диаграмма вариантов использования при проектировании программного обеспечения // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/566218/> (дата обращения: 10.05.2024).
2. Нефункциональные требования // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/companies/simbirsoft/articles/688428/> (дата обращения: 10.05.2024).
3. Диаграмма компонентов // НОУ ИНТУИТ [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/32/32/lecture/1022/> (дата обращения 11.05.2024).
4. Проектирование графического интерфейса // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/208966/> (дата обращения 11.05.2024).
5. Как управлять бюджетом проекта // SkillBox [Электронный ресурс]. URL: <https://skillbox.ru/media/management/kak-upravlyat-byudzhetom-proekta-poshagovoe-rukovodstvo/> (дата обращения 11.05.2024).
6. Что нужно знать о популярных JS-фреймворках // Хабр [Электронный ресурс]. URL: https://habr.com/ru/companies/yandex_praktikum/articles/533702/ (дата обращения 09.06.2024).
7. Шумиха вокруг PostgreSQL // Server gate [Электронный ресурс]. URL: <https://servergate.ru/articles/shumikha-vokrug-postgresql/> (09.06.2024).
8. Spring // SkillFactory [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/spring/> (07.06.2024).
9. Что такое MVC // Хекслет [Электронный ресурс]. URL: <https://ru.hexlet.io/blog/posts/cto-takoe-mvc-rasskazyvaem-prostymi-slovami> (07.06.2024).
10. Тестирование методом белого ящика // QA_Bible [Электронный ресурс]. URL: https://vladislavermeev.gitbook.io/qa_bible/vidy-metody-urovni-testirovaniya/testirovanie-metodom-belogo-yashika-white-box-testing (09.06.2024).

11. Что такое покрытие кода // ATLASSIAN [Электронный ресурс]. URL: <https://www.atlassian.com/ru/continuous-delivery/software-testing/code-coverage> (09.06.2024).
12. Фреймворки для тестирования // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/579032/> (09.06.2024).
13. Программный код и его метрики // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/companies/intel/articles/106082/> (09.06.2024).

Приложение А
(рекомендуемое)
Варианты использования

Таблица А.1 – Вариант использования «Создать аккаунт»

Вариант использования	Создать аккаунт
ID	1
Краткое описание	Пользователь указывает email и пароль
Главные актёры	Пользователь
Второстепенные актёры	Нет
Предусловия	1. Email не занят; 2. Email проходит валидацию; 3. Пароль проходит валидацию.
Основной поток	1. Пользователь переходит страницу регистрации; 2. Пользователь указывает email; 3. Пользователь указывает пароль.
Постусловия	Аккаунт создан

Таблица А.2 – Вариант использования «Авторизоваться через email и пароль»

Вариант использования	Авторизоваться через email и пароль
ID	2
Краткое описание	Пользователь указывает email и пароль, которые он использовал при регистрации
Главные актёры	Пользователь
Второстепенные актёры	Нет
Предусловия	1. Email корректный; 2. Пароль корректный.
Основной поток	1. Пользователь указывает email; 2. Пользователь указывает пароль.
Постусловия	Пользователь авторизуется

Таблица А.3 – Вариант использования «Выйти из аккаунта»

Вариант использования	Выйти из аккаунта
ID	3
Краткое описание	Пользователь выходит из аккаунта
Главные актёры	Пользователь
Второстепенные актёры	Нет
Предусловия	Пользователь был авторизован
Основной поток	Пользователь выходит из аккаунта
Постусловия	Пользователь переходит на страницу авторизации

Таблица А.4 – Вариант использования «Сбросить пароль»

Вариант использования	Сбросить пароль
ID	4
Краткое описание	Пользователю отправляется ссылка на подтверждение сброса пароля на почту
Главные актёры	Пользователь
Второстепенные актёры	Нет
Предусловия	Нет
Основной поток	<ol style="list-style-type: none"> 1. Пользователь переходит на страницу сброса пароля; 2. Пользователь вводит email; 3. Пользователю отправляется ссылка на подтверждение.
Постусловия	На почту пришла ссылка для подтверждения

Приложение Б
(рекомендуемое)
Листинг программы

Листинг Б.1 – Код файла Level

```

const bgCanvas = document.getElementById("bgCanvas");
const mapCanvas = document.getElementById("mapCanvas");
const tileSize = 80;
const mapHeight = rows * tileSize;
const mapWidth = cols * tileSize;

// Устанавливаем контексты рисования
const bgCtx = bgCanvas.getContext("2d");
const mapCtx = mapCanvas.getContext("2d");

// Устанавливаем размеры canvas
bgCanvas.width = mapWidth;
bgCanvas.height = mapHeight;
mapCanvas.width = mapWidth;
mapCanvas.height = mapHeight;

const waterPaths = [];
for (let i = 1; i <= 39; i++) {
    let imagePath = `../img/water/${i.toString().padStart(4, '0')}.png`;
    waterPaths.push(imagePath);
}

let currentWaterFrameIndex = 0;
const waterFrames = [];

function loadWaterFrames() {
    let loadedCount = 0;
    waterPaths.forEach((path, index) => {
        const img = new Image();
        img.onload = function() {
            waterFrames[index] = img;
            loadedCount++;
            if (loadedCount === waterPaths.length) {
                // Запуск анимации после загрузки всех изображений
                startWaterAnimation();
            }
        };
        img.src = path;
    });
}

let waterInterval;
function startWaterAnimation() {
    waterInterval = setInterval(() => {
        bgCtx.clearRect(0, 0, bgCanvas.width, bgCanvas.height);
        const pattern =
bgCtx.createPattern(waterFrames[currentWaterFrameIndex], 'repeat');
        bgCtx.fillStyle = pattern;
        bgCtx.fillRect(0, 0, bgCanvas.width, bgCanvas.height);
        currentWaterFrameIndex = (currentWaterFrameIndex + 1) %
waterFrames.length;
    }, 80);

    // const skyImage = new Image();
    // skyImage.src = "../img/sky2.png";

```

```

//      skyImage.onload = function() {
//          bgCtx.drawImage(skyImage, 0, 0, mapWidth, mapHeight / 3.7); //
Разместить небо в верхней части холста
//      };
}

loadWaterFrames();

const loadImage = (src) => {
    return new Promise((resolve, reject) => {
        const img = new Image();
        img.onload = () => resolve(img);
        img.onerror = (err) => reject(err);
        img.src = src;
    });
};

async function loadImages() {
    try {
        const [islandImage, island2Image, sandImage] = await
Promise.all([
            loadImage("../img/island/greenField.jpg"),
            loadImage("../img/island/greenField2.jpg")
        ]);
        drawMap(islandImage, island2Image);
    } catch (error) {
        console.error("Ошибка загрузки изображений:", error);
    }
}

loadImages();

function drawMap(field1, field2) {
    for (let row = 0; row < rows; row++) {
        for (let col = 0; col < cols; col++) {
            const x = col * tileSize;
            const y = row * tileSize;

            // Проверяем, есть ли текущая клетка в списке координат дыр
            const isHole = holes.some(hole => hole.row === row &&
hole.col === col);
            const isCactus = cactus.some(cactus => cactus.row === row &&
cactus.col === col);
            const isStone = flowers.some(flowers => flowers.row === row
&& flowers.col === col);
            const isExit = exit.some(exit => exit.row === row && exit.col
=== col);
            const isBone = bones.some(bone => bone.row === row &&
bone.col === col);
            const isItem = items.some(item => item.row === row &&
item.col === col);

            if (isHole) {
                // Если текущая клетка - дыра, отрисовываем изображение с
ямой
                mapCtx.clearRect(x, y, tileSize, tileSize);

                const holeImage = new Image();

```

```

holeImage.src = "../img/island/hole1.png";
holeImage.onload = function() {
    mapCtx.drawImage(holeImage, x, y, tileSize,
tileSize);
};
cells.push({ row: row, col: col, type: "hole" });
}
else if(isCactus) {
    mapCtx.clearRect(x, y, tileSize, tileSize);
    mapCtx.drawImage(field2, x, y, tileSize, tileSize);

    const image = new Image();
    image.src = "../img/island/grass1.png";
    image.onload = function() {
        mapCtx.drawImage(image, x, y, tileSize, tileSize+5);
    };
    cells.push({ row: row, col: col, type: "grass" });
}
else if (isExit) {
    mapCtx.clearRect(x, y, tileSize, tileSize);
    mapCtx.drawImage(field2, x, y, tileSize, tileSize);

    const image = new Image();
    image.src = "../img/island/sand1.jpg";
    image.onload = function() {
        mapCtx.drawImage(image, x, y, tileSize, tileSize);
    };
    cells.push({ row: row, col: col, type: "exit" });
}
else if (isBone) {
    mapCtx.clearRect(x, y, tileSize, tileSize);
    mapCtx.drawImage(field2, x, y, tileSize, tileSize);

    const bone = bones.find(bone => bone.row === row &&
bone.col === col);
    const image = new Image();
    image.src = "../img/island/cactus1.png";

    image.onload = function() {
        mapCtx.drawImage(image, x+20, y+15, tileSize - 40,
tileSize- 35);
    };
    cells.push({ row: row, col: col, type: bone.color});
}
else if (isItem) {
    mapCtx.clearRect(x, y, tileSize, tileSize);
    mapCtx.drawImage(field2, x, y, tileSize, tileSize);

    const item = items.find(bone => bone.row === row &&
bone.col === col);
    const image = new Image();
    if (item.color.includes("red"))
        image.src = "../img/island/exit1.png";
    else if (item.color.includes("white"))
        image.src = "../img/island/exit2.png";

    image.onload = function() {

```

```

        mapCtx.drawImage(image, x+17, y+13, tileSize - 32,
tileSize - 30);
    };
    cells.push({ row: row, col: col, type: item.color });
}
else if (isStone) {
    mapCtx.clearRect(x, y, tileSize, tileSize);
    mapCtx.drawImage(field2, x, y, tileSize, tileSize);
    cells.push({ row: row, col: col, type: "free" });
}
else {
    const pattern = mapCtx.createPattern(field1, 'repeat');
    mapCtx.fillStyle = pattern;
    mapCtx.fillRect(x, y, tileSize, tileSize);
    cells.push({ row: row, col: col, type: "free" });
}

// Отрисовываем контур клетки
mapCtx.strokeStyle = "rgba(0, 0, 0, 0.5)";
mapCtx.lineWidth = 1;
mapCtx.strokeRect(x, y, tileSize, tileSize);
}
}
}

// Создаем элемент div
const imageDiv = document.createElement("div");
imageDiv.id = "island1";
imageDiv.style.backgroundImage = "url(../img/island/sendBeach.png)";

// Устанавливаем стили для div
imageDiv.style.position = "absolute";
imageDiv.style.backgroundColor = "contain";
imageDiv.style.backgroundPosition = "center center";
imageDiv.style.backgroundRepeat = "no-repeat";
imageDiv.style.left = "60%";
imageDiv.style.top = "300px";
imageDiv.style.width = "135px";
imageDiv.style.height = "140px";

document.body.appendChild(imageDiv);

//////////
// Создаем элемент div
const imageDiv2 = document.createElement("div");
imageDiv2.id = "island2";
imageDiv2.style.backgroundImage = "url(../img/island/sendBeach2.png)";

// Устанавливаем стили для div
imageDiv2.style.position = "absolute";
imageDiv2.style.backgroundColor = "contain";
imageDiv2.style.backgroundPosition = "center center";
imageDiv2.style.backgroundRepeat = "no-repeat";
imageDiv2.style.left = "4%";
imageDiv2.style.top = "300px";
imageDiv2.style.width = "135px";

```

```
imageDiv2.style.height = "140px";

document.body.appendChild(imageDiv2);
```

Листинг Б.2 – Код файла Rating

```
function rating() {
    const widgetBlock = CreateDiv("", "", document.body, "widget-
container");
    const title = CreateLabel("h2", "Звезда", "", "", widgetBlock);

    /** VolumeText */

    const sectionVolume = CreateDiv("", "section", widgetBlock);

    const volumeTitle = CreateLabel("h4", "Испытание Объёма", "", "",
sectionVolume);

    const volumeText = CreateLabel("p", rateSolution(), "", "",
sectionVolume);
    function rateSolution() {
        const currentLength = document.getElementById("editor")
            .querySelectorAll('.code-line').length;
        const requiredLength = LEVELS_LENGTH[0].length;
        let ratingText = "";

        if (requiredLength >= currentLength) {
            ratingText += getRequiredText(requiredLength);
            ratingText += getCurrentText(currentLength);
            ratingText += getYESText();
        }
        else {
            ratingText += getRequiredText(requiredLength);
            ratingText += getCurrentText(currentLength);
            ratingText += getNOText();
        }

        return ratingText;
    }

    /** Volume's ProgressBar */

    const progressBlock = CreateDiv("", "result-bar", sectionVolume, "");

    const currentRate = CreateDiv("5", "progress-bar", progressBlock,
"");
    currentRate.style.width = `${(100 / 6) * 5}%`;

    const requiredRate = CreateDiv('6', 'remaining-bar', progressBlock);
    requiredRate.style.width = `${(100 / 6)}%`;

    /** button's block */

    const buttonsBlock = CreateDiv('', 'button-container', widgetBlock);
```

```

    const backButton = CreateButton('Попробовать ещё', "", "back-button",
buttonsBlock);
    backButton.addEventListener("click", function() {
        document.body.removeChild(document.querySelector('#overlay'));
        document.body.removeChild(widgetBlock);
        resetRobot();
    });

    const nextButton = CreateButton('Пройти к островам', "", "next-
button", buttonsBlock);
    nextButton.addEventListener("click", function() {
        ToIslands();
    });

    displayOff();
}

/*----- Get coords of hero -----*/

function getCellType(row, col) {
    const cell = cells.find(cell => cell.row === row && cell.col ===
col);
    return cell.type;
}

function isExit() {
    return getCellType(robot.position.row, robot.position.col) ===
"exit";
}

function isDeath() {
    return getCellType(robot.position.row, robot.position.col) ===
"hole";
}

function displayOff() {
    const overlay = document.createElement('div');
    overlay.id = 'overlay';

    overlay.style.position = 'fixed';
    overlay.style.top = 0;
    overlay.style.left = 0;
    overlay.style.width = '100%';
    overlay.style.height = '100%';
    overlay.style.background = "rgba(0, 0, 0, 0.5)";
    overlay.style.zIndex = "1";

    document.body.appendChild(overlay);

    const widgetBlock = document.querySelector('#widget-container');
    if (widgetBlock) {
        widgetBlock.style.zIndex = 2;
        widgetBlock.style.pointerEvents = 'auto';
    }
}

```

Листинг Б.3 – Код файла assistor

```

const bodyHTML = document.body;

/** Ship */

const gramp = document.createElement("div");
gramp.id = "gramp";
bodyHTML.appendChild(gramp);
gramp.direction = "right";
gramp.coords = "topLeft";
gramp.shift = [parseInt(getComputedStyle(gramp).width) - 30,
               parseInt(getComputedStyle(gramp).height) - 30
              ];
gramp.style.backgroundImage = 'url(../img/parrot/ship.png)';

let rect = mapCanvas.getBoundingClientRect();
let islandShift = 50;

let waypoints = {
  topLeft: {
    x: rect.left - gramp.shift[0] - islandShift,
    y: rect.top - gramp.shift[1] - islandShift
  },
  bottomLeft: {
    x: rect.left - gramp.shift[0] - islandShift,
    y: rect.top + rect.height - gramp.shift[1] + islandShift
  },
  topRight: {
    x: rect.left + rect.width - gramp.shift[0] + islandShift,
    y: rect.top - gramp.shift[1] - islandShift
  },
  bottomRight: {
    x: rect.left + rect.width - gramp.shift[0] + islandShift,
    y: rect.top + rect.height - gramp.shift[1] + islandShift
  }
};

gramp.style.left = waypoints.topLeft.x + "px";
gramp.style.top = waypoints.topLeft.y + "px";

//***** Animation *****//

let currentAnimation = null;
let isPaused = false;
let animTimer;

class Animation {
  constructor(element, targetX, targetY, duration) {
    this.element = element;
    this.startX = parseInt(element.style.left, 10) || 0;
    this.startY = parseInt(element.style.top, 10) || 0;
    this.targetX = targetX;
    this.targetY = targetY;
    this.deltaX = targetX - this.startX;
    this.deltaY = targetY - this.startY;
    this.duration = duration;
  }
}

```

```

        this.startTime = null;
        this.animationFrame = null;
        this.isPaused = false;
        this.elapsed = 0;
        this.resolve = null;
    }

    animate() {
        return new Promise((resolve) => {
            this.resolve = resolve;
            const step = (currentTime) => {
                if (!this.startTime) {
                    this.startTime = currentTime;
                }

                const elapsed = currentTime - this.startTime +
this.elapsed;
                const progress = Math.min(elapsed / this.duration, 1);

                this.element.style.left = this.startX + this.deltaX *
progress + 'px';
                this.element.style.top = this.startY + this.deltaY *
progress + 'px';

                if (progress < 1) {
                    this.animationFrame = requestAnimationFrame(step);
                } else {
                    this.resolve();
                }
            };

            this.animationFrame = requestAnimationFrame(step);
        });
    }

    pause() {
        this.isPaused = true;
        clearTimeout(animTimer);
        cancelAnimationFrame(this.animationFrame);
        this.elapsed += performance.now() - this.startTime;
    }

    resume() {
        if (this.isPaused) {
            this.isPaused = false;
            this.startTime = null;
            this.animate().then(() => {
                if (this.resolve) {
                    this.resolve();
                }
            });
            turnGramp();
            animTimer = setTimeout(moveGramp, 43000);
        }
    }
}

```

```

async function moveGramp() {
  let x, y;
  switch (gramp.coords) {
    case 'topLeft':
      ({ x, y } = waypoints.topRight);
      gramp.coords = 'topRight';
      break;
    case 'topRight':
      ({ x, y } = waypoints.bottomRight);
      gramp.coords = 'bottomRight';
      break;
    case 'bottomRight':
      ({ x, y } = waypoints.bottomLeft);
      gramp.coords = 'bottomLeft';
      break;
    case 'bottomLeft':
      ({ x, y } = waypoints.topLeft);
      gramp.coords = 'topLeft';
      break;
  }

  currentAnimation = new Animation(gramp, x, y, 8000);

  currentAnimation.animate().then(() => {
    turnGramp();
    animTimer = setTimeout(moveGramp, 43000);
  });
}

function turnGramp() {
  if ((gramp.coords === 'topRight' && gramp.direction === 'down') ||
      (gramp.coords === 'topLeft' && gramp.direction === 'right')
  ||
      (gramp.coords === 'bottomRight' && gramp.direction ===
'left') ||
      (gramp.coords === 'bottomLeft' && gramp.direction === 'up'))
    return;

  const directions = ["up", "right", "down", "left"];
  let currentIndex = directions.indexOf(gramp.direction);
  let newIndex = (currentIndex + 1) % directions.length;
  gramp.direction = directions[newIndex];

  let rotation = newIndex * 90;
  gramp.style.transform = `rotate(${rotation}deg)`;
}

function animateMovement(element, targetX, targetY, duration) {
  return new Promise(resolve => {
    const startX = parseInt(element.style.left, 10) || 0;
    const startY = parseInt(element.style.top, 10) || 0;
    const deltaX = targetX - startX;
    const deltaY = targetY - startY;
    const startTime = performance.now();

```

```

function animate(currentTime) {
    const elapsed = currentTime - startTime;
    const progress = Math.min(elapsed / duration, 1);

    element.style.left = startX + deltaX * progress + 'px';
    element.style.top = startY + deltaY * progress + 'px';

    if (progress < 1) {
        requestAnimationFrame(animate);
    } else {
        resolve();
    }
}

requestAnimationFrame(animate);
});
}

moveGramp();

```

Листинг Б.4 – Код файла levelMenu

```

const screen = document.querySelector('#screen');
const levels = document.querySelector('#levels');

//const level1 = CreateDiv("", "level", levels);

//const staticCanvas1 = document.createElement('canvas');
//const staticCtx1 = staticCanvas1.getContext('2d');
//staticCanvas1.width = level1.clientWidth;
//staticCanvas1.height = level1.clientHeight;
//staticCanvas1.classList.add('levelCanvas');
//level1.appendChild(staticCanvas1);
//
//const animatedCanvas1 = document.createElement('canvas');
//const animatedCtx1 = animatedCanvas1.getContext('2d');
//animatedCanvas1.width = level1.clientWidth;
//animatedCanvas1.height = level1.clientHeight;
//animatedCanvas1.classList.add('levelCanvas');
//level1.appendChild(animatedCanvas1);

function createLevel(parentElement) {
    const level = CreateDiv("", "level", parentElement);

    const staticCanvas = document.createElement('canvas');
    const staticCtx = staticCanvas.getContext('2d');
    staticCanvas.width = level.clientWidth;
    staticCanvas.height = level.clientHeight;
    staticCanvas.classList.add('levelCanvas');
    level.appendChild(staticCanvas);

    const animatedCanvas = document.createElement('canvas');
    const animatedCtx = animatedCanvas.getContext('2d');
    animatedCanvas.width = level.clientWidth;
    animatedCanvas.height = level.clientHeight;
    animatedCanvas.classList.add('levelCanvas');
}

```

```

    level.appendChild(animatedCanvas);

    return { level, staticCtx, animatedCtx, animatedCanvas };
}

let levelsArr = [];
levelsArr.push(createLevel(levels));
levelsArr[0].level.addEventListener('click', () => {
    ToLevel(1);
});
levelsArr.push(createLevel(levels));
levelsArr[1].level.addEventListener('click', () => {
    ToLevel(2);
});
levelsArr.push(createLevel(levels));
levelsArr[2].level.addEventListener('click', () => {
    ToLevel(3);
});

/////    Scrolling    /////
let isMouseDown = false;
let startX, startScrollLeft;

document.addEventListener('mousedown', (e) => {
    if (e.target.closest('#screen')) {
        isMouseDown = true;
        startX = e.pageX;
        startScrollLeft = screen.scrollLeft;
    }
});

document.addEventListener('mouseleave', () => {
    isMouseDown = false;
});

document.addEventListener('mouseup', () => {
    isMouseDown = false;
});

document.addEventListener('mousemove', (e) => {
    if (!isMouseDown) return;
    e.preventDefault();
    const x = e.pageX;
    const walk = (x - startX) * 2;
    screen.scrollLeft = startScrollLeft - walk;
});

levels.addEventListener('wheel', (e) => {
    e.preventDefault();
    levels.scrollLeft += e.deltaY;
});

document.addEventListener('wheel', (e) => {
    if (e.target.closest('#screen')) {
        e.preventDefault();
        screen.scrollLeft += e.deltaY;
    }
});

```

```

    }
  });

  /*----- Animations -----*/

  const loadImage = (src) => {
    return new Promise((resolve, reject) => {
      const img = new Image();
      img.onload = () => resolve(img);
      img.onerror = (err) => reject(err);
      img.src = src;
    });
  };

  let isHovering = [];
  let cloudX = [];
  let cloudY = [];
  let cloud = [];
  let cloudIndex = [];

  async function loadImages() {
    try {
      const palm1 = await loadImage("../img/island/palm1.png");
      const exit1 = await loadImage("../img/island/exit1.png");
      const exit2 = await loadImage("../img/island/exit2.png");
      const redBones = await loadImage("../img/island/redBones.png");
      const whiteBones = await
loadImage("../img/island/whiteBones.png");
      const cloud1 = await loadImage("../img/island/cloud3.png");
      const cloud2 = await loadImage("../img/island/cloud2.png");

      const imagesData = [
        [
          { image: exit1, x: 10, y: 30, width: 27, height: 30 },
          { image: palm1, x: 40, y: 1, width: 115, height: 120 }
        ],
        [
          { image: exit1, x: 10, y: 30, width: 27, height: 30 },
          { image: redBones, x: 110, y: 30, width: 40, height: 35 }
        ],
        [
          { image: palm1, x: 40, y: 1, width: 115, height: 120 }
        ],
        [
          { image: exit1, x: 120, y: 40, width: 27, height: 30 },
          { image: exit2, x: 10, y: 60, width: 23, height: 25 },
          { image: redBones, x: 100, y: 20, width: 40, height: 35 }
        ],
        [
          { image: whiteBones, x: 10, y: 30, width: 40, height: 35 }
        ],
        [
          { image: palm1, x: 40, y: 1, width: 115, height: 120 }
        ]
      ];
    }
  };
  const clouds = [cloud1, cloud2];

  levelsArr.forEach((level, index) => {
    isHovering[index] = false;
    cloudX[index] = 0;
    cloudY[index] = 0;
    cloudIndex[index] = 0;
  });

```

```

drawMap(level.staticCtx, imagesData[index]);

level.animatedCanvas.addEventListener('mouseenter', () => {
    isHovering[index] = true;
    moveCloud(level.animatedCtx, clouds, index);
});

level.animatedCanvas.addEventListener('mouseleave', () => {
    isHovering[index] = false;
    cloudX[index] = cloud[index].x;
    cloudY[index] = cloud[index].y;
});
});
} catch (error) {
    console.error("Ошибка загрузки изображений:", error);
}
}

function moveCloud(ctx, clouds, levelIndex) {
    cloud[levelIndex] = {
        image: clouds[cloudIndex[levelIndex]],
        x: cloudX[levelIndex],
        y: cloudY[levelIndex],
        width: 90,
        height: 50,
        speed: 0.3
    };

    function animate() {
        if (isHovering[levelIndex]) {
            cloud[levelIndex].x += cloud[levelIndex].speed;

            if (cloud[levelIndex].x >
levelsArr[levelIndex].animatedCanvas.width) {
                cloud[levelIndex].x = -cloud[levelIndex].width;
                cloudIndex[levelIndex] = (cloudIndex[levelIndex] + 1) %
clouds.length;
                cloud[levelIndex].image = clouds[cloudIndex[levelIndex]];
            }

            ctx.clearRect(0, 0,
levelsArr[levelIndex].animatedCanvas.width,
levelsArr[levelIndex].animatedCanvas.height);
            ctx.drawImage(cloud[levelIndex].image, cloud[levelIndex].x,
cloud[levelIndex].y, cloud[levelIndex].width,
cloud[levelIndex].height);
            requestAnimationFrame(animate);
        }
    }

    animate();
}

function drawMap(ctx, imagesData) {
    imagesData.forEach(({ image, x, y, width, height }) => {
        ctx.drawImage(image, x, y, width, height);
    });
}

```

```

    });
}

/*----- Water's Animation -----*/

const bgCanvas = document.getElementById("bgCanvas");
const bgCtx = bgCanvas.getContext("2d");

const tileSize = 80;
const rows = 5;
const cols = 6;
const mapHeight = rows * tileSize;
const mapWidth = cols * tileSize;
bgCanvas.width = mapWidth;
bgCanvas.height = mapHeight;

const waterPaths = [];
let currentWaterFrameIndex = 0;
const waterFrames = [];

for (let i = 1; i <= 39; i++) {
    let imagePath = `../img/water/${i.toString().padStart(4, '0')}.png`;
    waterPaths.push(imagePath);
}

function loadWaterFrames() {
    let loadedCount = 0;
    waterPaths.forEach((path, index) => {
        const img = new Image();
        img.onload = function() {
            waterFrames[index] = img;
            loadedCount++;
            if (loadedCount === waterPaths.length) {
                startWaterAnimation();
            }
        };
        img.src = path;
    });
}

let waterInterval;
function startWaterAnimation() {
    waterInterval = setInterval(() => {

        bgCtx.clearRect(0, 0, bgCanvas.width, bgCanvas.height);
        const pattern =
bgCtx.createPattern(waterFrames[currentWaterFrameIndex], 'repeat');
        bgCtx.fillStyle = pattern;
        bgCtx.fillRect(0, 0, bgCanvas.width, bgCanvas.height);
        currentWaterFrameIndex = (currentWaterFrameIndex + 1) %
waterFrames.length;
    }, 80);
}

loadWaterFrames();
loadImages();
const volumeBtn = CreateDiv("", "", document.body, "volume");

```

Листинг Б.5 – Код файла hero

```

/***** Character
*****/

const robot = {
  element: document.getElementById("robot"),
  shiftBorder: 2.5,
  position: { row: 0, col: 0 },
  direction: "down"
};

let mapCanvasRect = mapCanvas.getBoundingClientRect();
robot.element.style.left = getCoords()[0];
robot.element.style.top = getCoords()[1];

window.addEventListener('resize', function() {
  robot.element.style.left = getCoords()[0];
  robot.element.style.top = getCoords()[1];
});

function getCoords() {
  let mapCoords = mapCanvas.getBoundingClientRect();

  let axisX = mapCoords.left + tileSize * robot.position.col +
robot.shiftBorder + "px";
  let axisY = mapCoords.top + tileSize * robot.position.row +
robot.shiftBorder + "px";

  return [axisX, axisY];
}

/* Character's actions */

function moveRobot(forward = true) {
  const rob = robot.element;
  const newPosition = { ...robot.position };
  let direct = robot.direction;

  if (!forward) {
    direct = robot.direction === "up" ? "down"
      : robot.direction === "down" ? "up"
      : robot.direction === "left" ? "right"
      : "left";
  }

  switch(direct) {
    case "up":
      newPosition.row--;
      break;
    case "down":
      newPosition.row++;
      break;
    case "left":
      newPosition.col--;
      break;
    case "right":

```

```

        newPosition.col++;
        break;
    }

    if (isValidPosition(newPosition)) {
        robot.position = newPosition;
        updateRobotPosition();
    }
}

async function take() {
    const cell = cells.find(cell => cell.row === robot.position.row &&
cell.col === robot.position.col);
    if (cell && cell.type.includes("Item")) {

        const field = await loadImage("../img/island/greenField2.jpg");
        const x = cell.col * tileSize;
        const y = cell.row * tileSize;
        mapCtx.clearRect(x, y, tileSize, tileSize);
        mapCtx.drawImage(field, x, y, tileSize, tileSize);

        const index = cells.indexOf(cell);
        if (index > -1) {
            cells.splice(index, 1);
        }
        return false;
    }
    else if (cell && cell.type.includes("cactus")) {
        playDeathAnimation();
        return true;
    }
}

function updateRobotPosition() {
    const mapCanvasRect = mapCanvas.getBoundingClientRect();

    const currentPosition = {
        row: parseInt(robot.element.style.top, 10),
        col: parseInt(robot.element.style.left, 10)
    };

    const targetPosition = {
        row: mapCanvasRect.top + tileSize * robot.position.row +
robot.shiftBorder,
        col: mapCanvasRect.left + tileSize * robot.position.col +
robot.shiftBorder
    };

    const steps = 4;
    const stepDistance = {
        row: (targetPosition.row - currentPosition.row) / steps,
        col: (targetPosition.col - currentPosition.col) / steps
    };

    let currentIndex = 1;
    let currentStep = 0;

```

```

const intervalId = setInterval(() => {
    currentPosition.row += stepDistance.row;
    currentPosition.col += stepDistance.col;

    robot.element.style.top = `${currentPosition.row}px`;
    robot.element.style.left = `${currentPosition.col}px`;

    robot.element.style.backgroundImage =
`url('../img/bird/Fly${currentIndex}.png')`;
    currentIndex = currentIndex % 4 + 1;

    currentStep++;
    if (currentStep >= steps) {
        clearInterval(intervalId);
        setTimeout(() => {
            robot.element.style.backgroundImage =
`url('../img/bird/dragonF.png')`;
            }, 10);
        }
    }, 500 / steps);
}

function turnRobot(direct) {
    const directions = ["up", "right", "down", "left"];
    const currentIndex = directions.indexOf(robot.direction);
    const newIndex = direct === "left" ? (currentIndex + 3) % 4 :
(currentIndex + 1) % 4;
    robot.direction = directions[newIndex];

    const rotation = newIndex * 90;
    document.getElementById("robot").style.transform =
`rotate(${rotation}deg)`;
}

function playDeathAnimation(callback) {
    let currentIndex = 1;
    const maxIndex = 8;
    const animationInterval = 100; // Интервал между кадрами в
миллисекундах

    const intervalId = setInterval(() => {
        robot.element.style.backgroundImage =
`url('../img/bird/Death${currentIndex}.png')`;
        currentIndex++;

        if (currentIndex > maxIndex) {
            clearInterval(intervalId);
            if (callback) {
                callback();
                setTimeout(() => {
                    robot.element.style.backgroundImage =
`url('../img/bird/dragonF.png')`;
                    }, 500);
                }
            }
        }, animationInterval);
}

```

Листинг Б.6 – Код файла codeEditor

```

const editor = document.getElementById("editor");
const commands = document.getElementById("commands");
const editorBlock = document.getElementById("editorBlock");

//***** Command's buttons *****//

let commandsBtns = [];
commandsBtns.push(CreateButton("move forward", "", "gradient-button",
commands));
commandsBtns.push(CreateButton("turn left", "", "gradient-button",
commands));
commandsBtns.push(CreateButton("turn right", "", "gradient-button",
commands));
commandsBtns.push(CreateButton("for", "", "gradient-button", commands));
commandsBtns.push(CreateButton("take", "", "gradient-button", commands));

commandsBtns.forEach((button) => {
    button.addEventListener("click", function() {
        addCodeLine(button.textContent);
    });
});

document.addEventListener("DOMContentLoaded", function () {
    var commandsHeight =
document.getElementById("commands").offsetHeight;
    var commandsWidth = document.getElementById("row").offsetWidth;;
    commandsHeight += 5;
    commandsWidth += 15;
    editor.style.bottom = commandsHeight + "px";
    document.getElementById("row").style.bottom = commandsHeight + "px";
    editor.style.left = commandsWidth + "px";
});

//***** addCodeLine *****//

async function addCodeLine(instruction) {
    let newLine = document.createElement("div");
    let row;

    if (instruction === 'for') {
        newLine = await showForForm();
        row = addRowNumber(2);
    } else {
        row = addRowNumber();
        newLine.appendChild(setTextBGColor(instruction));
        newLine.classList.add('code-line');
    }

    const selectedLine = editor.querySelector(".selected");
    if (selectedLine) {
        editor.insertBefore(newLine, selectedLine.nextSibling);
    } else {
        editor.appendChild(newLine);
    }
}

```

```

newLine.addEventListener("click", function() {
  if (commandsBtns.some(btn => !btn.disabled)) {
    const previousSelectedLine = editor.querySelector(".selected");

    if (previousSelectedLine) {
      const deleteButton =
previousSelectedLine.querySelector(".delete-button");
      if (deleteButton) {
        previousSelectedLine.removeChild(deleteButton);
      }
      previousSelectedLine.classList.remove("selected");
    }
    newLine.classList.add("selected");

    const deleteButton = document.createElement("span");
    deleteButton.textContent = "✕";
    deleteButton.classList.add("delete-button");

    deleteButton.addEventListener("click", function(event) {
      event.stopPropagation();

      const forBlock = newLine.closest('.for-block');
      if (forBlock) {
        const forBody = forBlock.querySelector('.for-body');
        const forLines = 2 + forBody.children.length; // For
header, footer, and body lines
        for (let i = 0; i < forLines; i++) {
          row.removeChild(row.lastChild);
        }
        editor.removeChild(forBlock);
      } else {
        row.removeChild(row.lastChild);
        editor.removeChild(newLine);
      }
    });
    newLine.appendChild(deleteButton);
  }
});

enableDragAndDrop();
}

function showForForm() {
  // Create the form elements
  const form = document.createElement("div");
  form.classList.add("for-form");

  const label = document.createElement("label");
  label.textContent = "for 1 to";

  const input = document.createElement("input");
  input.type = "number";
  input.min = "1";
  input.placeholder = "[число]";

  const okButton = document.createElement("button");

```

```

    okButton.textContent = "OK";

    form.appendChild(label);
    form.appendChild(input);
    form.appendChild(okButton);
    commands.appendChild(form);

    return new Promise((resolve) => {
        okButton.addEventListener("click", function() {
            const value = input.value;
            if (value) {
                const block = addForBlock(value);
                commands.removeChild(form);
                resolve(block);
            }
        });
    });
}

function addForBlock(value) {
    const forBlock = document.createElement("div");
    forBlock.classList.add('for-block');
    forBlock.counter = value;
    forBlock.value = 1;

    const forHeader = document.createElement("div");
    forHeader.appendChild(setTextBGColor(`for 1 to ${value} {`,
'rgba(200, 255, 235, 0.85)'));
    forHeader.classList.add('code-line');
    forHeader.style.pointerEvents = 'none';

    const forBody = document.createElement("div");
    forBody.classList.add('for-body');

    const forFooter = document.createElement("div");
    forFooter.appendChild(setTextBGColor(`}`, 'rgba(200, 255, 235,
0.85)'));
    forFooter.classList.add('code-line');
    forFooter.style.pointerEvents = 'none';

    forBlock.appendChild(forHeader);
    forBlock.appendChild(forBody);
    forBlock.appendChild(forFooter);
    return forBlock;
}

//***** addCodeLineAndClearBtn *****/

const clearButton = document.getElementById("clearButton");
clearButton.className = "gradient-button";
clearButton.addEventListener("click", clearAllCode);
function clearAllCode() {
    const row = document.getElementById("row");
    editor.innerHTML = "";
    row.innerHTML = "";
    enableDragAndDrop();
}

```

```

        resetRobot();
    }

function addRowNumber(count = 1) {
    const row = document.getElementById("row");
    for (i=0; i<count; i++) {
        const lineNumber = document.createElement("div");
        lineNumber.textContent = row.children.length + 1;
        row.appendChild(lineNumber);
    }
    return row;
}

function setTextBGColor(text, color = 'rgba(240, 200, 255, 0.85)') {
    const span = document.createElement("span");
    span.textContent = text;
    span.style.backgroundColor = color;
    span.style.pointerEvents = 'none';
    return span;
}

//***** dragAndDrop *****/

let draggedElement;

function dragStart(event) {
    if (commandsBtns.some(btn => !btn.disabled)) {
        draggedElement = event.target;
    } else {
        event.preventDefault();
    }
}

function dragOver(event) {
    if (commandsBtns.some(btn => !btn.disabled)) {
        event.preventDefault();
    }
}

function dragEnter(event) {
    if (commandsBtns.some(btn => !btn.disabled)) {
        event.preventDefault();
        if (event.target.classList.contains('code-line')
            || event.target.classList.contains('for-block')) {
            event.target.classList.add('drag-over');
        }
    }
}

function dragLeave(event) {
    if (commandsBtns.some(btn => !btn.disabled)) {
        if (event.target.className.includes('drag-over')) {
            event.target.classList.remove('drag-over');
        }
    }
}

```

```

function dragDrop(event) {
    if (commandsBtns.some(btn => !btn.disabled)) {
        event.preventDefault();
        //event.target.closest('.for-block')?.classList.add('drag-over');

        const dragOverElement = document.querySelector('.drag-over');

        if (dragOverElement) {
            if (dragOverElement.classList.contains('for-block')) {
                const forBodyElement = dragOverElement.querySelector('.for-
body');
                forBodyElement.appendChild(draggedElement);}
            else {
                const targetIndex =
Array.from(dragOverElement.parentElement.children).indexOf(dragOverElemen
t);
                const draggedIndex =
Array.from(draggedElement.parentElement.children).indexOf(draggedElement)
;
                if (targetIndex !== -1 && draggedIndex !== -1)
                    if (targetIndex > draggedIndex)

dragOverElement.parentElement.insertBefore(draggedElement,
dragOverElement.nextSibling);
                    else

dragOverElement.parentElement.insertBefore(draggedElement,
dragOverElement);

                dragOverElement.classList.remove('drag-over');
            }
        }
    }
}

function enableDragAndDrop() {
    const codeLines = document.querySelectorAll('.code-line, .for-
block');
    codeLines.forEach(function(codeLine) {
        codeLine.setAttribute('draggable', true);
        codeLine.addEventListener('dragstart', dragStart);
        codeLine.addEventListener('dragover', dragOver);
        codeLine.addEventListener('dragenter', dragEnter);
        codeLine.addEventListener('dragleave', dragLeave);
        codeLine.addEventListener('drop', dragDrop);
    });
}

```

Листинг Б.7 – Код файла compiler

```

/*----- Compiler's block -----*/
const compilerBlock = CreateDiv("", "compiler", document.body);
const volumeBtn = CreateButton("", "", "", compilerBlock);
volumeBtn.style.backgroundImage = "url('../img/volumeOff.png')";
volumeBtn.id = "volume";

const resetBtn = CreateButton("", " ", "", compilerBlock);

```

```

resetBtn.style.backgroundImage = "url('../img/compiler/resetGray.png')";
resetBtn.disabled = true;
resetBtn.addEventListener("click", function () {resetRobot();});

const backBtn = CreateButton("", "" , "", compilerBlock);
backBtn.style.backgroundImage = "url('../img/compiler/backGray.png')";
backBtn.disabled = true;
backBtn.addEventListener("click", function ()
{executeCommands("backward");});

const playBtn = CreateButton("", "" , "", compilerBlock);
playBtn.style.height = "60px";
playBtn.style.width = "60px";
playBtn.style.top = "10px";
playBtn.style.backgroundImage = "url('../img/compiler/playGreen.png')";
playBtn.addEventListener("click", function () {
    executeCommands("run");
});

const toBtn = CreateButton("", "" , "", compilerBlock);
toBtn.style.backgroundImage = "url('../img/compiler/toGreen.png')";
toBtn.addEventListener("click", function ()
{executeCommands("forward");});

/*----- Debug's arrow block -----*/

const arrow = CreateDiv("", "arrow", editorBlock);
arrow.style.visibility = "hidden";
arrow.id = "arrow";
arrow.style.backgroundImage = "url('../img/play.png')";
let previousCommandLine = null;

function resetRobot() {
    robot.position = { row: 0, col: 0 };
    robot.direction = "down";
    robot.element.style.left = getCoords()[0];
    robot.element.style.top = getCoords()[1];
    robot.element.style.backgroundImage =
`url('../img/bird/dragonF.png')`;
    document.getElementById("robot").style.transform = 'rotate(180deg)';

    commandIndex = 0;
    resetFor();
    switchButtons();
    switchCompilerBtns("reset");
    updateArrowPosition("backward");
    reloadMap();
}

function resetFor() {
    const editor = document.getElementById("editor");
    const commands = editor.querySelectorAll('.code-line');
    const result = Array.from(commands).filter(element =>
element.textContent.includes("for 1 to"));
    result.forEach(command => {
        command.counter = command.value;
    });
}

```

```

function clearCanvas() {
    mapCtx.clearRect(0, 0, mapCanvas.width, mapCanvas.height);
    cells = [];
}
async function reloadMap() {
    clearCanvas();
    await loadImages();
}

function isValidPosition(position) {
    if (
        position.row >= 0 &&
        position.row < rows &&
        position.col >= 0 &&
        position.col < cols
    ) {
        const cell = cells.find(cell => cell.row === position.row &&
cell.col === position.col);
        return !(cell && cell.type === 'grass');
    }
    return false;
}

/**/ Execute's command ***/

let commandIndex = 0;
function executeCommands(commandType) {
    const editor = document.getElementById("editor");
    const commands = editor.querySelectorAll('.code-line');

    if (commands.length !== 0 && commandIndex === 0) {
        switchButtons(true);
        switchCompilerBtns("toStep");
        if (commandType === "run")
            switchCompilerBtns(commandType);
    }
    else if (commands.length === 0)
        return;

    if (commandType === "run") {
        let isReturn = false;
        const executeNextCommand = () => {
            if (commandIndex < commands.length) {
                const command =
commands[commandIndex].textContent.trim();
                if (command.includes("{}"))
                    forCommand(commands);

                executeCommand(command, isReturn);
                updateArrowPosition();

                if (isReturn)
                    return;

                if(isDeath()) {
                    setTimeout(() => {
                        deathMessage();

```

```

        playDeathAnimation();
    }, 500);
    return;
}

if (commandIndex === commands.length) {
    switchCompilerBtns("endPoint");
    if (isExit()) {
        rating();
    }
    else {
        setTimeout(() => {
            errorMessage();
        }, 800);
    }
} else {
    setTimeout(executeNextCommand, 500);
}
};
executeNextCommand();
}

else if (commandType === "forward") {
    const executeNextCommand = () => {
        if (commandIndex < commands.length) {

            const command =
commands[commandIndex].textContent.trim();
            if (command.includes("{}"))
                forCommand(commands);

            executeCommand(command);
            updateArrowPosition();

            if (commandIndex === commands.length)
                switchCompilerBtns("endPoint");
        }
    };
    executeNextCommand();
}

else if (commandType === "backward") {
    const executeNextCommand = () => {
        if (commandIndex >= 0) {

            switchCompilerBtns("backStep");
            updateArrowPosition("backward");

            const command =
commands[commandIndex].textContent.trim();
            if (command.includes("for 1 to"))
                forCommand(commands, "forBack");

            if (command === "turn left") {
                executeCommand("turn right");
            }
            else if (command === "turn right") {

```

```

        executeCommand("turn left");
    }
    else if (command === "move forward") {
        executeCommand("move backward");
    }

    if (commandIndex === 0)
        switchCompilerBtns("startPoint");
    }
};
executeNextCommand();
}
}

function executeCommand(command, isReturn = false) {
    switch(command) {
        case "move forward":
            moveRobot();
            break;
        case "turn left":
            turnRobot("left");
            break;
        case "turn right":
            turnRobot("right");
            break;
        case "move backward":
            moveRobot(false);
            break;
        case "take":
            take(isReturn);
            break;
    }
}

function forCommand(commands, command = "forTo") {
    if (commands[commandIndex].parentElement.querySelector('.for-
body').children.length > 0) {

        if (command === "forTo" &&
commands[commandIndex].parentElement.counter > 1) {
            commands[commandIndex].parentElement.counter--;
            commands[commandIndex].parentElement.value++;
            updateArrowPosition(command);
        }
        else if (command === "forBack" &&
commands[commandIndex].parentElement.value > 1){
            commands[commandIndex].parentElement.counter++;
            commands[commandIndex].parentElement.value--;
            updateArrowPosition(command);
        }
    }
}

function switchButtons(disable) {
    commandsBtns.forEach(btn => {
        btn.disabled = disable;
    });
}

```

```

}

function switchCompilerBtns(typeBtn) {
  const buttonConfigs = {
    run: [
      { btn: playBtn, bg: 'playGray', disabled: true }
    ],
    toStep: [
      { btn: resetBtn, bg: 'resetGreen', disabled: false },
      { btn: backBtn, bg: 'backGreen', disabled: false }
    ],
    backStep: [
      { btn: toBtn, bg: 'toGreen', disabled: false }
    ],
    startPoint: [
      { btn: backBtn, bg: 'backGray', disabled: true },
      { btn: playBtn, bg: 'playGreen', disabled: false }
    ],
    endPoint: [
      { btn: playBtn, bg: 'playGray', disabled: true },
      { btn: toBtn, bg: 'toGray', disabled: true }
    ],
    reset: [
      { btn: resetBtn, bg: 'resetGray', disabled: true },
      { btn: backBtn, bg: 'backGray', disabled: true },
      { btn: playBtn, bg: 'playGreen', disabled: false },
      { btn: toBtn, bg: 'toGreen', disabled: false }
    ]
  };

  if (buttonConfigs[typeBtn]) {
    buttonConfigs[typeBtn].forEach(config => {
      updateButtonState(config.btn, config.bg, config.disabled);
    });
  }
}

function updateButtonState(button, bgSuffix, isDisabled) {
  const currentBg = button.style.backgroundImage;
  const newBg = `url('../img/compiler/${bgSuffix}.png')`;

  if (button.disabled !== isDisabled) {
    button.disabled = isDisabled;
    button.style.backgroundImage = newBg;
  }
}

/*----- Debug arrow -----*/

function updateArrowPosition(forward = "forward") {
  const editor = document.getElementById("editor");
  const arrow = document.getElementById("arrow");
  const commands = editor.querySelectorAll('.code-line');

  if (previousCommandLine) {
    previousCommandLine.style.backgroundColor = "";
  }
}

```

```

    if (forward === "forward") {
        if (commandIndex < commands.length) {
            updatePosition(commands, commandIndex);
            commandIndex++;
        } else {
            arrow.style.visibility = 'hidden';
        }
    }
    else if (forward === "backward") {
        if (commandIndex > 0) {
            commandIndex--;
            updatePosition(commands, commandIndex);
        } else {
            arrow.style.visibility = 'hidden';
        }
    }
}

else if (forward === "forTo") {

    let command = commands[commandIndex].textContent.trim();
    while (!command.includes("for 1 to")) {
        commandIndex--;
        command = commands[commandIndex].textContent.trim();
    }
    updatePosition(commands, commandIndex);
}

else if (forward === "forBack") {

    let command = commands[commandIndex].textContent.trim();
    while (!command.includes("")) {
        commandIndex++;
        command = commands[commandIndex].textContent.trim();
    }

    updatePosition(commands, commandIndex);
}
}

function updatePosition(commands, index) {
    const editor = document.getElementById("editor");
    const arrow = document.getElementById("arrow");
    const currentElement = commands[index];
    const rect = currentElement.getBoundingClientRect();
    const editorRect = editor.getBoundingClientRect();

    arrow.style.top = `${rect.top - editorRect.top +
currentElement.offsetHeight / 2 - 5}px`;
    arrow.style.left = `${rect.left - editorRect.left - 1}px`;

    arrow.style.visibility = 'visible';
    currentElement.style.backgroundColor = "lightblue";
    previousCommandLine = currentElement;
}

```

Листинг Б.8 – Код файла navigation

```
document.addEventListener('DOMContentLoaded', function () {
    const isLoggedIn = localStorage.getItem('isLoggedIn');
    if (isLoggedIn !== 'true') {
        window.location.href = '../html/auth/login.html';
        return;
    }
});

/** Functions of creating elements */
function CreateElement(param)
{
    return document.createElement(param);
}

function CreateDiv(text, bClass, doc, id = "") {
    const div = document.createElement('div');
    div.className = bClass;
    div.id = id;
    div.innerText = text;
    doc.appendChild(div);

    return div;
}

function CreateButton(title, id, bClass, object) {
    let button = document.createElement('button');
    button.textContent = title;
    button.className = bClass;
    button.id = id;

    if (object === null || object === "")
        document.body.appendChild(button);
    else
        object.appendChild(button);

    return button;
}

function CreateLabel(type, title, id, className, object = "") {
    let tableLabel = CreateElement(type);
    tableLabel.innerHTML = title;
    tableLabel.id = id;
    tableLabel.className = className;

    if (object === null || object === "")
        document.body.appendChild(tableLabel);
    else
        object.appendChild(tableLabel);

    return tableLabel;
}

function CreateInputText(title, id, bClass) {
    let search = document.createElement('input');
    search.placeholder = title;
}
```

```

        search.type = "text";
        search.value = "";
        search.className = bClass;
        search.id = id;
        search.autofocus = true;
        document.body.appendChild(search);

        return search;
    }

function CreateInputBtnOrSubmit(title, type, id, bClass) {
    let search = document.createElement('input');
    search.placeholder = title;
    search.type = type;
    search.value = "";
    search.title = title;
    search.className = bClass;
    search.id = id;
    search.autofocus = true;
    document.body.appendChild(search);

    return search;
}

function CreateImg(src, alt, id) {
    let portrait = CreateElement('img');
    portrait.src = src;
    portrait.alt = alt;
    portrait.id = id;

    return portrait;
}

function CreateSelect(arr, id, className) {
    const createCharacterUrl = document.createElement('select');
    createCharacterUrl.id = id;
    createCharacterUrl.className = className;

    arr.forEach(item => {
        const optionElement = document.createElement('option');
        optionElement.value = item;
        optionElement.text = item.charAt(0) + item.slice(1);
        createCharacterUrl.add(optionElement);
    });

    return createCharacterUrl;
}

function ToIslands() {
    window.location.href =
"http://localhost:8383/Frontend/html/levelMenu.html";
}

function ToLevel(index) {
    window.location.href =
`http://localhost:8383/Frontend/html/level${index}.html`;
}

```