

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

К защите допустить:

Заведующая кафедрой ПОИТ

_____ Н. В. Лапицкая

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту
на тему

**СЕРВИС-ФОРУМ МНОГОПОЛЬЗОВАТЕЛЬСКОЙ ИГРЫ
«ТЕТРИС» НА РАЗЛИЧНЫХ УСТРОЙСТВАХ**

БГУИР ДП 1-40 01 01 047 ПЗ

Студент

А.А. Дранкевич

Руководитель

Е.Е. Фадеева

Консультанты:
от кафедры ПОИТ
по экономической части

Е.Е. Фадеева
И.В. Смирнов

Нормоконтролер

К.Э. Рудак

Рецензент

Минск 2024

РЕФЕРАТ

СЕРВИС-ФОРУМ МНОГОПОЛЬЗОВАТЕЛЬСКОЙ ИГРЫ «ТЕТРИС»
НА РАЗЛИЧНЫХ УСТРОЙСТВАХ: дипломный проект / А.А. Дранкевич. –
Минск: БГУИР, 2024, – п.з. – 112 с., чертежей (плакатов) – 6 л. формата А1.

Цель настоящего дипломного проекта состоит в разработке сервиса, предназначенного для проведения досуга и развития реакции, внимания и стратегического мышления, обмена информацией, участия в соревнованиях.

При разработке и внедрении приложения используется следующий стек технологий: JavaScript, TypeScript, React, React Three Fiber, Node.js, Express, Socket.IO.

В первом разделе проводится анализ литературных источников, обзор аналогичных программных средств и формируются требования.

Во втором разделе проводится анализ выдвинутых требований и разрабатываются функциональные требования и их спецификации.

Третий раздел посвящен разработке архитектуры сервиса, модели базы данных, отдельных алгоритмов, протокола взаимодействия, пользовательского интерфейса.

В четвертом разделе содержится информация о тестировании сервиса, приведены тестовые случаи для проверки соответствия всех модулей выдвинутым функциональным требованиям.

Пятый раздел содержит руководство пользователя и администратора, а также руководство по развертыванию сервиса.

В шестом разделе приведено технико-экономическое обоснование разработки и внедрения сервиса. Проведен расчет затрат на разработку, а также вычисление показателей эффективности инвестиций в разработку.

Заключение содержит краткие выводы по дипломному проекту.

Дипломный проект является завершенным, поставленная задача решена в полной мере. Планируется дальнейшее развитие программного средства и расширение его функциональности. Проект выполнен самостоятельно, проведен анализ оригинальности в системе «Антиплагиат». Процент оригинальности составляет 96,39%.

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра _____ Программное обеспечение информационных технологий
(наименование кафедры)

УТВЕРЖДАЮ
Заведующий кафедрой

_____ Лапицкая Н. В.

« ____ » _____ 2024 г.

ЗАДАНИЕ
на дипломный проект (дипломную работу)

Обучающемуся _____ Дранкевичу Артёму Андреевичу

Курс 4 Учебная группа 051006 Специальность 1-40 01 01

1 Тема дипломного проекта (дипломной работы):

Сервис-форум многопользовательской игры «Тетрис» на различных устройствах
Утверждена приказом ректора от « 7 » марта 2024 г. № 512-с

2 Исходные данные к дипломному проекту (дипломной работе):

Языки программирования JavaScript, TypeScript.
Платформа Node.js, библиотека React, библиотека Three.js.
Перечень выполняемых функций: регистрация пользователей, наличие личного кабинета для
зарегистрированных пользователей, осуществление игрового процесса по базовым и
расширенным правилам игры «Тетрис», осуществление командной игры многопользовательской
игры, осуществление обмена сообщениями между пользователями.

3 Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):

Введение
1 Анализ прототипов, литературных источников и формирование требований к проектируемому
сервису
2 Анализ требований к сервису и разработка функциональных требований
3 Проектирование сервиса
4 Создание сервиса
5 Тестирование, проверка работоспособности и анализ полученных результатов
6 Руководство по установке и эксплуатации сервиса
7 Техничко-экономическое обоснование разработки многопользовательской игры «Тетрис»
Заключение
Список использованных источников
Приложения

4 Перечень графического материала (с точным указанием обязательных чертежей и графиков):

Исследование индустрии казуальных игр. Плакат – формат А1, лист 1.

Диаграмма вариантов использования сервиса. Плакат – формат А1, лист 1.

Пользовательский интерфейс. Плакат – формат А1, лист 1.

Взаимодействие клиента и сервера. Схема алгоритма – формат А1, лист 1.

Обработка событий игровой комнаты. Схема алгоритма – формат А1, лист 1.

Обработка событий игрового поля. Схема алгоритма – формат А1, лист 1.

5 Консультанты по дипломному проекту (дипломной работе):

Консультант от кафедры – Фадеева Е. Е.

Консультант по экономической части – Смирнов И. В.

Нормоконтролер – Рудак К. Э.

6 Примерный календарный график выполнения дипломного проекта (дипломной работы):

Анализ предметной области, разработка технического задания – 25.03–01.04

Разработка функциональных требований, проектирование архитектуры ПС – 02.04–13.04

Разработка схемы программы, алгоритмов, данных: 14.04–21.04

Разработка программного средства: 22.04–05.05

Тестирование и отладка: 06.05–13.05

Оформление пояснительной записки и графического материала: 14.05–31.05

Дата выдачи задания 25 марта 2024 г.

Срок сдачи законченного дипломного проекта (дипломной работы) 31 мая 2024 г.

Руководитель дипломного проекта
(дипломной работы)

(подпись)

Фадеева Е.Е.

(фамилия, инициалы)

Подпись обучающегося _____

Дата 25 марта 2024 г.

СОДЕРЖАНИЕ

Введение.....	9
1 Анализ прототипов, литературных источников и формирование требований к проектируемому сервису	10
1.1 Анализ концепции на основе существующих аналогов	10
1.2 Разработка возможных концепций игры.....	17
1.3 Требования к сервису	20
2 Анализ требований к сервису и разработка функциональных требований .	25
2.1 Описание функциональности сервиса	25
2.2 Спецификация функциональных требований	29
3 Проектирование сервиса	35
3.1 Разработка архитектуры сервиса	35
3.2 Разработка логической и физической модели базы данных	35
3.3 Разработка алгоритмов модулей сервиса	36
4 Создание сервиса.....	42
4.1 Общие принципы	42
4.2 Разработка клиента	43
4.3 Разработка сервера	49
4.4 Разработка игровой логики.....	52
5 Тестирование, проверка работоспособности и анализ полученных результатов	55
5.1 Общий принцип и инструменты тестирования	55
5.2 Тестирование основного игрового процесса	55
5.3 Тестирование базы данных.....	57
5.4 Тестирование игрового аккаунта и мессенджера.....	58
5.5 Тестирование функционала игровых комнат	61
5.6 Автоматическое тестирование	63
5.7 Вывод из тестирования сервиса	64
6 Руководство по установке и эксплуатации сервиса	65
6.1 Руководство по установке сервиса	65
6.2 Руководство по эксплуатации сервиса для администратора	65
6.3 Руководство по эксплуатации сервиса для игрока.....	65
7 Техничко-экономическое обоснование разработки многопользовательской игры «Тетрис»	76
7.1 Характеристика сервиса, разрабатываемого для реализации на рынке	76
7.2 Расчет инвестиций в разработку сервиса для его реализации на рынке	77
7.3 Расчет экономического эффекта от реализации сервиса на рынке	79
7.4 Расчет показателей экономической эффективности разработки и реализации сервиса на рынке.....	82
7.5 Вывод технико-экономического обоснования	84
Заключение	85
Список использованных источников	86
Приложение А (обязательное) Исследование индустрии казуальных игр	88

Приложение Б (обязательное) Диаграмма вариантов использования сервиса	89
Приложение В (обязательное) Обработка событий игрового поля	90
Приложение Г (обязательное) Взаимодействие клиента и сервера.....	91
Приложение Д (обязательное) Обработка событий игровой комнаты.....	92
Приложение Ж (обязательное) Текст программы модуля игрового взаимодействия с сервером.....	93
Приложение И (обязательное) Текст программы модуля игрового взаимодействия с клиентом	97
Приложение К (обязательное) Текст программы модуля игровой логики...	101

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

Авторизация – процесс предоставления пользователю или группе пользователей определенных разрешений, прав доступа и привилегий в компьютерной системе.

Алгоритм – совокупность точно заданных правил решения некоторого класса задач или набор инструкций, описывающих порядок действий исполнителя для решения определенной задачи.

Аутентификация – процедура проверки подлинности, например проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.

Клиент – это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу.

Кроссплатформенность – способность программного обеспечения работать более чем на одной аппаратной платформе и (или) операционной системе.

Платформа – группа технологий, которые используются в качестве основы для разработки других приложений, процессов или технологий.

Регистрация – запись, фиксация фактов или явлений с целью учета и придания им статуса официально признанных актов.

Сервер – компонент, содержащий основную бизнес-логику.

Сервис – идентифицируемая уникальным веб-адресом программная система со стандартизированными интерфейсами.

Спецификация программы – формализованное представление требований, предъявляемых к программе, которые должны быть удовлетворены при ее разработке, а также описание задачи, условия и эффекта действия без указания способа его достижения.

Триггер – хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных.

Фреймворк – программное обеспечение, облегчающее разработку и объединение различных компонентов большого программного проекта.

ИД – Индекс доходности.

НДС – Налог на добавленную стоимость.

СУБД – Система управления базами данных.

ФСЗН – Фонд социальной защиты населения.

ЧДД – Чистый дисконтированный доход.

3D – 3-Dimensional (трехмерный).

API – Application Programming Interface (программный интерфейс приложения).

DDoS – Distributed denial-of-service (распределенная атака «отказ в обслуживании»).

HTTP – HyperText Transfer Protocol (протокол передачи гипертекста).

JSON – JavaScript Object Notation (текстовый формат обмена данными, основанный на JavaScript).

NAT – Network Address Translation (преобразование сетевых адресов).

ORM – Object-Relational Mapping (объектно-реляционное отображение).

REST – Representational State Transfer (передача состояния представления).

TCP – Transmission Control Protocol (протокол управления передачей).

SQL – Structured Query Language (язык структурированных запросов).

UDP – User Datagram Protocol (протокол пользовательских датаграмм).

URL – Uniform Resource Locator (унифицированный указатель ресурса).

ВВЕДЕНИЕ

В эпоху цифровых технологий и инноваций, игры стали неотъемлемой частью нашей жизни. Особое место среди них занимает жанр казуальных игр. Их популярность среди пользователей обусловлена многими факторами.

Выполнение цели проектирования будет произведено на основе игры «Тетрис» – одной из самых узнаваемых и любимых игр, которая была впервые разработана в СССР в 1984 году. С тех пор, она обзавелась миллионами поклонников по всему миру. Данная игра заложила новый важный принцип в разработку игр, чем проще и понятнее механика игры, тем выше шанс, что она завоевывает рынок, став одной из основополагающих игр казуального жанра.

Актуальность данной темы обусловлена рядом факторов. Во-первых, игры, особенно классические, такие как «Тетрис», остаются востребованными и любимыми многими пользователями. Внедрение сервиса посвященного игре «Тетрис» позволит создать виртуальное пространство для обмена опытом, стратегиями, и творческими идеями. Во-вторых, общество стремится к социализации в цифровой среде, и игровые сообщества становятся важной частью этого процесса. Мой проект предоставит возможность не только общения, но и обучения, создавая уникальное сообщество, сплоченное общим интересом к игре «Тетрис».

Предметная область моей разработки охватывает не только аспекты развлечения, но и технологические вызовы, связанные с созданием и управлением сервисом. Основные требования, предъявляемые к базе данных и программному комплексу, включают в себя безопасность данных, высокую производительность, гибкость архитектуры для возможности дальнейшего развития.

В ходе работы будет проведен анализ существующих концептов казуальных игр, разработаны проект и архитектура будущего сервис-форума, а также реализован прототип данного сервиса. Данная работа будет полезна как для игроков, так и для разработчиков игр, интересующихся созданием подобных сервисов.

1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ СЕРВИСУ

1.1 Анализ концепции на основе существующих аналогов

Игра «Тетрис» пользуется огромной популярностью и имеет многочисленную фанатскую базу, в мире существует бесчисленное множество аналогов этой игры, поэтому рассмотреть их всех не представляется возможным. Основная цель – выявить наиболее успешные аналоги этой игры, а также игр, построенных на этом концепте, определить их положительные и отрицательные решения, выявить решения, которые позволили этим играм получить свою популярность. Игра «Тетрис» относится к жанру казуальных игр-головоломок, поэтому аналоги будут также выбираться из игр, принадлежащих этому жанру. Казуальные игры занимают особое место среди других игр. Их популярность среди пользователей обусловлена многими факторами [1, 2]:

- простота и доступность;
- краткосрочность игровых сессий;
- разнообразие жанров;
- социальный аспект;
- доступность на мобильных устройствах.

Согласно данным собранных сервисом Udonis, после пандемии индустрия игр, особенно казуальных, претерпевает большие изменения [3]. Несмотря на колебания потребительских расходов, доходы от мобильной рекламы продолжали расти, достигнув только в США отметки в \$6.28 миллиардов в 2023 году, возврат от рекламных расходов остается ключевым показателем эффективности рекламных стратегий, что изображено на рисунке 1.1 [4]. В тренд выходят гибридно-казуальные игры (Hybrid Casual) – это игры, которые смешивают элементы бывших наиболее популярными гипер-казуальных (Hyper-Casual) игр с элементами жанров средней категории, изображено на рисунке 1.2. Таким играм удастся привлечь наибольшее внимание аудитории. Казуальные игры со соревновательным аспектом при этом становятся новым трендом развития. Ключом к успеху авторы статьи считают правильный выбор жанров для объединения в одну игру.

По данным, полученным из статей Capermint и Room8 Studio, при разработке игры в казуальном жанре, необходимо учитывать несколько ключевых факторов, чтобы обеспечить ее успех [5, 6]:

- возможность начала игры без предварительной подготовки;
- простота механики;
- основа мотивации игрока;
- сбалансированный игровой процесс;
- привлекательное визуальное и звуковое оформление.

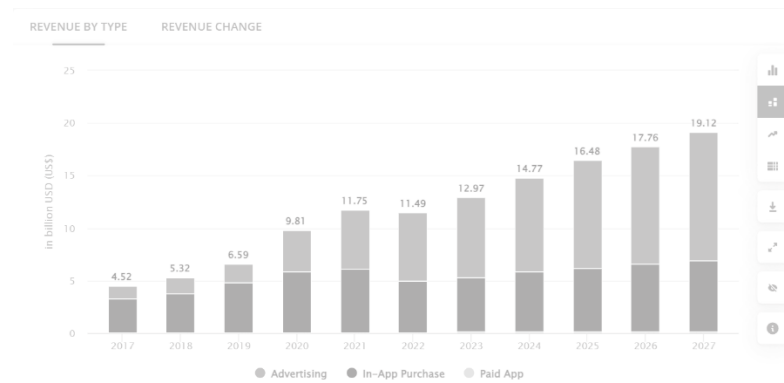


Рисунок 1.1 – Прогноз возврата от рекламных стратегий



Рисунок 1.2 – Доля казуальных игр по скачиваниям в мире

Игра должна позволять пользователям включаться в игровой процесс в течение нескольких секунд после запуска. Процесс вхождения в игру должен быть минимизирован, а сама игра – интуитивно понятной. Визуальные элементы должны быть привлекательными и захватывающими, чтобы привлечь внимание игроков и удержать их интерес. Игровые механики должны быть простыми, но иметь динамическую сложность для удержания интереса игрока, при этом, механика должна позволять адаптировать сложность под нужды игрока. Игра должна предоставлять четкие и ясные цели для выполнения, создавать ощущения вызова, достижения и награды – это побуждает игроков вкладывать больше времени и усилий. Существуют различные механики управления, которые используются в казуальных играх [7]:

- перетаскивание;

- рисование;
- нажатие в срок;
- джойстик;
- падение или подъем;
- прицеливание;
- письмо;
- выбор;
- объединение.

По статистике, наиболее популярными остаются механики выбора, перетаскивания и объединения, рисования и поворота.

Также не менее важен способ показа прогресса игроку, эта механика ведет игрока вперед и погружают его в процесс приключения и достижения целей, все это необходимо для эффективного вовлечения, удержания игрока, и монетизации игры [8]. В статье Udonis о системах прогресса в мобильных играх рекомендуется вводить новый функционал постепенно, создать ощущение прогресса и достижения, чтобы удержать внимание игрока, не перегрузить его новой информацией, создать у него желание вернуться к игре. В ней же предлагаются основные типы систем прогресса в играх:

- сбор ресурсов;
- увеличение уровня героев;
- квесты и задания;
- строительство и декор;
- повествование;
- система достижений и дерево навыков;
- таблицы лидеров;
- прогресс в команде;
- прогресс, ограниченный в реальном времени.

Тут же подмечается, что прогресс должен иметь вариативность, возможность выбора, ключевые точки, прогресс должен быть ощутим как через визуальные компоненты, так и через внутриигровые механики, должен быть сопряжен с прогрессом в игре, а награда должна быть сопоставима со сложностью задачи, за которую она предоставляется. Вывод таков, что система прогресса является ключевым аспектом успеха игры, а универсального подхода к ее разработке не существует.

Изучив ресурсы в сети Интернет, можно сделать вывод, что большинство существующих проектов по этой теме достаточно примитивны и ограничиваются классической реализацией без каких-либо улучшений, расположены на сайтах-агрегаторах, и предназначены скорее для ознакомления с концептом игры, нежели для полноценного игрового опыта. Однако существуют аналоги, которые успешно адаптировали игровой процесс и предоставляют услуги. Основную проблему предоставляет тот факт, что существующий концепт игры «Тетрис» защищен авторскими правами [9]. Решение, представленное в проекте, экспериментальное и нацелено на исследование подходов к проектированию и разработке игровых механик.

1.1.1 Игра «Тетрис» версии NES

Игра «Тетрис» для NES (Nintendo Entertainment System) является одной из самых известных и влиятельных версий этой классической головоломки, представлен на рисунке 1.1 [10].

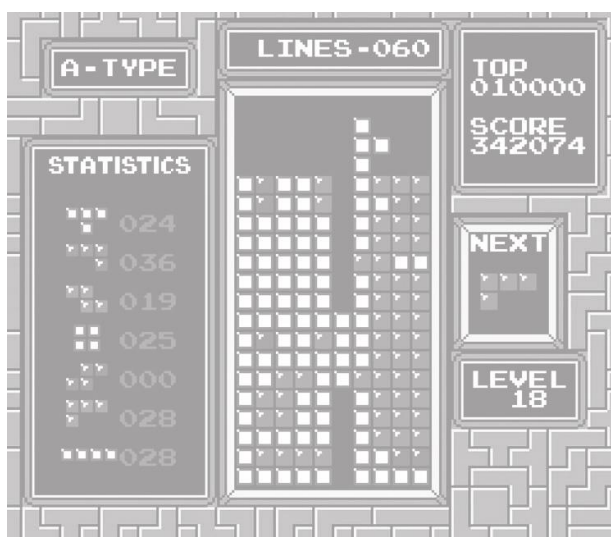


Рисунок 1.1 – «Тетрис» версии NES

Игра была разработана и выпущена компанией Nintendo в 1989 году. Механика игры по версии NES считается классической для игры «Тетрис», и заключается в том, что игроку предлагается управлять фигурами «тетрамино», состоящими из четырех квадратных блоков. Фигуры могут иметь различные формы, включая кубики, палки, Т-образные и другие. Игрок может вращать и перемещать фигуры по горизонтали, чтобы они падали по игровому полю. Игровое поле представляет собой вертикальную прямоугольную область, состоящую из сетки квадратных ячеек. Задача игрока – заполнять горизонтальные линии поля, используя фигуры. Когда заполняется полностью горизонтальная линия, она исчезает, и игрок получает очки. Чем больше линий уничтожается за один раз, тем больше очков получает игрок. Очки могут использоваться для отслеживания успехов игрока и сравнения результатов с другими игроками. Со временем скорость падения фигур увеличивается, что делает игру сложнее. Игровой процесс разделен на уровни, и каждый уровень представляет собой определенный набор скоростей. Игра продолжается до тех пор, пока фигуры могут помещаться на игровом поле. Если фигуры достигают верхней части поля, игра заканчивается.

При разработке игры на основе механики «Тетрис» можно обратить внимание на следующие полезные аспекты:

- простота и интуитивность управления;
- постепенное увеличение сложности;
- очки и система рейтинга;
- графика и звуковые эффекты;

- ясное отображение прогресса;
- возможность сохранения и загрузки;
- множество режимов игры.

1.1.2 Сервис Chess.com

Один из успешных аналог сервиса, Chess.com, представлен на рисунке 1.2 [11]. Несмотря на то, что он посвящен игре шахматы, интересен он тем, что переносит классическую игру в сеть, что является задачей проекта. Сейчас Chess.com предоставляет широкий спектр возможностей для шахматистов разного уровня, включая возможность игры с реальными соперниками, обучение, участие в турнирах, и взаимодействие в сообществе. Сервис также предоставляет статистику, аналитику и другие инструменты для обучения и улучшения навыков игроков.

Положительные аспекты рассматриваемого сервиса:

- многопользовательский режим;
- рейтинговая система;
- учебные ресурсы;
- анализ игры;
- ежедневные и еженедельные задания;
- коммуникация и сообщество, интеграция с социальными сетями;
- мобильное приложение.

Сервис Chess.com позволяет играть в шахматы с другими людьми со всего мира в реальном времени. Возможность отслеживать прогресс игроков и предоставлять им обратную связь помогает поддерживать мотивацию к игре.



Рисунок 1.2 – Сервис Chess.com

Сервис позволяет игрокам отслеживать их результаты, обсуждать их с другими пользователями и делиться ими в социальных сетях. Так же Chess.com имеет мобильное приложение, которое позволяет игрокам играть в любое время и в любом месте. Существенных недостатков во время анализа сервиса обнаружено не было.

Исходя из анализа и выявленных положительных аспектов сервиса Chess.com, при разработке сервиса-форума «Тетрис» стоит обратить внимание и реализовать не только возможность игры, но и интегрированные образовательные ресурсы и социальные функции, такие как форумы и чаты.

Для формирования активного сообщества вокруг игры необходим функционал, способствующий формированию соревновательного духа и сплоченности пользователей путем обсуждения стратегий и организации соревнований. Обязательно требуется возможность отслеживания личного прогресса, анализа собственных игр и изучение эффективных стратегий. Основной трудностью же при таком подходе является отсутствие командного элемента у игры Тетрис в своей основе.

1.1.3 Игра Tricky Towers

Другим интересным аналогом может являться Tricky Towers, представлена на рисунке 1.3 – по смыслу игра несколько напоминает «Тетрис», однако она довольно сильно отличается от оригинала [12]. Интересна она тем, что в корне меняет основные механики игры, интегрируя совершенно новые, что позволяет добавить как соревновательный элемент в игру, так и механику прогресса, что привлекает и затягивает игрока.

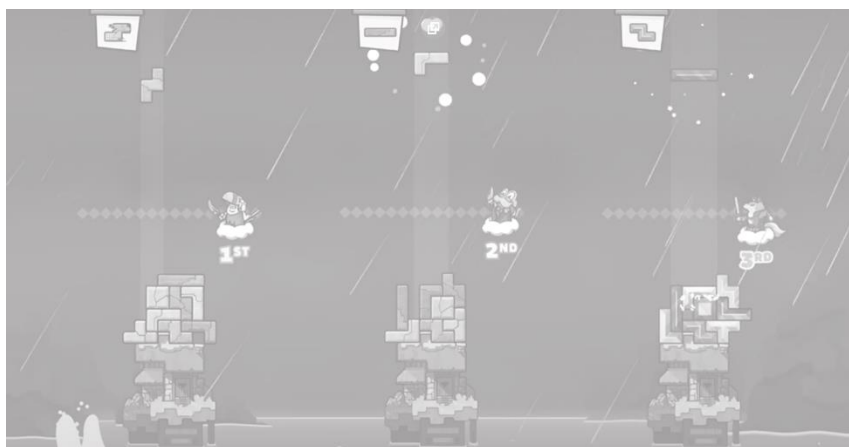


Рисунок 1.3 – Игра Tricky towers

Главной особенностью Tricky Towers является использование физики в игровом процессе. Фигуры здесь такие же, как и в оригинальной игре, но подчиняются законам физики. Игрока вынуждают балансировать на узком участке, и, если фигура не может закрепиться, или центр тяжести кластера блоков оказывается за пределами безопасной зоны – этот кластер обваливается, разрушая башню и отдаляя игрока от цели. Игроки могут использовать магические способности и специальные блоки, чтобы помешать своим соперникам или усилить собственную башню. Это добавляет элементы стратегии и взаимодействия между игроками, а также соревнование, заключающееся в наиско-

рейшем достижении верхней границы карты, которая движется в зависимости от общего прогресса каждого из игроков.

Tricky Towers предлагает разнообразные режимы, такие как гонка на высоту, выживание, пазл и другие. Она обладает яркой и красочной графикой, которая создает приятную визуальную атмосферу. Звуковое сопровождение также играет важную роль, поддерживая атмосферу. В игре также присутствует система рейтинга, система кланов и система покупок доступных игроку специальных эффектов.

Анализируя положительные аспекты игры Tricky Towers, можно выделить важность графической составляющей, взаимодействия между игроками, многопользовательской игры, соревновательной составляющей, использования вспомогательных механик. Отрицательными аспектами приложения могут стать следующие: приложение распространяется только по платной подписке, требует установки на устройство, локализация отсутствует. Этот проект полностью изменил механику для внедрения соревновательной составляющей.

1.1.4 Игра 1010!

Игра 1010! – это вариант известного пазла с фигурами из тетриса, игра-головоломка, доступна на мобильных устройствах и в онлайн-версии, представлена на рисунке 1.4 [13]. Она была разработана компанией Gram Games и впервые выпущена в 2014 году.

Цель игры 1010! – заполнить горизонтальные и вертикальные линии на игровом поле 10x10 квадратов. В игре предоставляются три фигуры одновременно, которые необходимо поместить на поле. Когда полная горизонтальная или вертикальная линия заполняется, она исчезает, освобождая место для следующих фигур. Игра продолжается до тех пор, пока есть свободное место на поле для размещения фигур.

Одна из особенностей 1010! – отсутствие временного ограничения или счетчика ходов. В игре можно принимать решения спокойно и внимательно планировать ходы. Однако, если разместить все предоставленные фигуры на поле невозможно, игра завершится.

Основными особенностями игры являются ее простота в доступе и освоении. Игра так же выполнена в минималистичном стиле, цвета сочные и яркие. Создатели игры ведут социальные сети, поддерживая обратную связь со своими пользователями.

1.1.5 Вывод из анализа аналогов

Исходя из анализа аналогов, можно сделать вывод о том, что для успеха проекта, требуется наличие сильного соревновательного и командного элемента, а также приветствуется наличие элемента развития и прогресса. Все это способствует удержанию внимания и более сильному вовлечению в игровой процесс. Механика классической игры «Тетрис» не предполагает соревновательного эффекта, явного и четкого отображения прогресса и многопользова-

тельского режима, а это значит, что необходимо разработать совершенно новую механику, и конечный продукт будет серьезно отличаться от существующего концепта игры «Тетрис».

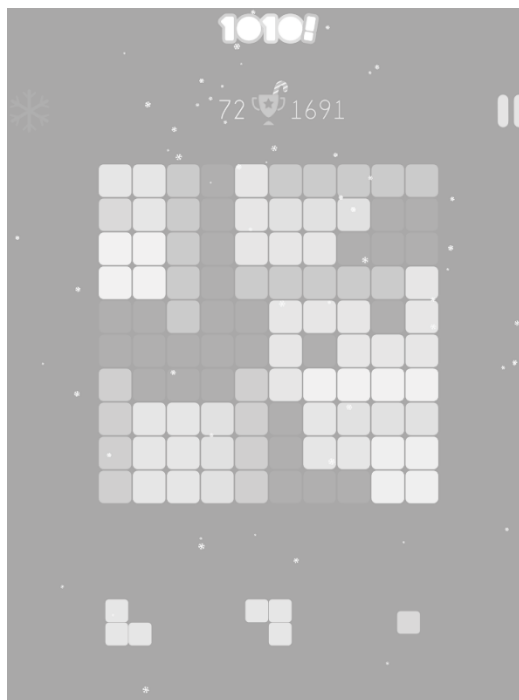


Рисунок 1.4 – Игра 1010!

Для обеспечения простого доступа к игре приложение будет разрабатываться в первую очередь для доступа с браузера, без обязательных требований к установке и регистрации. Механика игры должна остаться простой, на освоение основ не должно уходить более 5 минут.

1.2 Разработка возможных концепций игры

Всю концепцию игры «Тетрис» можно разделить на несколько основных механик:

- составные части фигур – блоки;
- поступление фигур – приток блоков;
- игровое поле, стакан – ограничитель;
- уничтожение линий блоков – отток и награждение;
- переполнение стакана – условие окончания игры, штраф;
- ускорение игры – увеличение игровой сложности.

Для каждой из компонент выделяются их параметры, которые позволят выявить возможности для модификации концепта игры.

1.2.1 Составные части фигур

Составные части фигур – блоки – участвуют в заполнении игрового поля и плотно связаны со всеми механиками игры. У блоков можно выделить параметры: размер, правила передвижения по полю, взаимодействия с другими блоками, внешний вид. В игре «Тетрис» размер блока всегда одинаков, он инертен и статичен, или движется по правилам управления фигурой, внешний вид не влияет на процесс игры. Потенциал для внесения новизны тут огромен, рассмотрим варианты:

- переменное количество очков за сбор блока;
- специальный тип очков за сбор блока;
- специальное поведение при взаимодействии с другими блоками.

Например, блок может иметь различную стоимость за сбор, что будет менять количество очков за линию, собранную с этим блоком. Он может предоставлять очки другого вида, например, для повышения уровня или получения других бонусов для активации других механик. Блок может быть подверженным динамике по типу «песок» – осыпаясь вниз, занимая пустоты; «жидкость» – распространяясь в любые пустоты рядом с собой; «разрушитель» – разрушая другие блоки, встречающиеся на пути по различным правилам; «призрак» – проходя сквозь другие блоки; «заполнитель» – создавая другие блоки вокруг себя; и др.

1.2.2 Приток блоков

Приток блоков в игре «Тетрис» осуществляется в виде заранее заготовленных фигур, каждая из которых состоит из четырех блоков. Фигуры поступают последовательно, каждый раз выбирается случайная по принципу «мешок» [14]. Параметрами тут можно считать принцип генерации фигуры, тип и состав фигуры. Для разнообразия игры можно ввести фигуры с другим количеством блоков, а также процедурно созданные фигуры. Состав фигуры можно выбирать в зависимости от других стимулирующих параметров, например уровня игрока. Приток может быть замедлен, остановлен или ускорен, выдавая новую фигуру не дожидаясь игрока.

1.2.3 Отток блоков

Главной механикой игры «Тетрис» остается правило оттока блоков – сбор полных линий и начисление очков за одновременный сбор большего их числа. Однако отток блоков можно производить по другим причинам, например, с помощью типа блоков «разрушитель» или с помощью игровых событий. Начисляемые очки предлагается определять суммой стоимостей отдельных блоков, количеством линий, и действующим модификатором счета. За сбор однотипных линий начисляются дополнительные очки.

1.2.4 Игровое поле

Игровое поле в классической версии «Тетрис» ограничено стаканом фиксированного размера, за границы которого нельзя выйти. Параметрами для

изменения здесь может быть ширина и высота игрового поля, а также проходимость стен, и поведение блоков за границей игрового поля. Одним из видов поведения может быть предоставление возможности игроку пользоваться пространством вне игрового поля, отнимая очки за постановку блоков в такой ситуации, для соблюдения игрового баланса. Это нововведение необходимо, если поле сможет расширяться или сужаться в процессе игры. При расширении игрового поля предлагается либо оставить блоки неизменными, либо заполнять освободившееся пространство путем процесса «осыпания» – наиболее неустойчивые блоки рядом со стенами заполняют образовавшиеся пустоты.

1.2.5 Штрафы и окончание игры

В оригинальной версии игрок не теряет счет, игра штрафует за неэффективное заполнение поля, уменьшая количество доступного игроку времени на постановку каждой новой фигуры. Окончание игры происходит при выходе за границы игрового поля. В перспективной версии условие окончания игры можно поменять, управляя лимитом поражения, меняя высоту или грузоподъемность платформы. Игрока можно защитить от штрафа или поражения, создав специальный бонус. Штрафы можно внедрить для балансировки других внедренных механик, например, неограниченное игровое поле.

1.2.6 Увеличение игровой сложности

Игровая сложность растет со временем и определяет скорость падения фигуры. В перспективной версии, игровую сложность можно определять не только скоростью падения, а еще типом фигур, размером поля, силой соперника, и др.

1.2.7 Другие игровые механики

Для создания командного режима предлагается объединять накопление счета игроков в команде. Командный счет помогает продвигаться вперед, в процессе получая дополнительные бонусы для команды. Командный счет необходимо визуализировать для всех игроков, а также добавить механики взаимодействия другим командам для влияния на него.

На основе выявленных параметров можно добавить игровые бонусы и события, влияющие на игровой процесс, в том числе рейтинг и дерево развития для игроков. Предполагаемый список игровых бонусов и событий представлен ниже:

- «инфляция» – стоимость всех расположенных блоков на платформе снижается;
- «золотая лихорадка» – все блоки на поле становятся максимальной стоимости, доступной игроку;
- «буря» – игра случайно создает блоки некоторого типа, которые перемещаются вниз по полю, мешая игроку;
- сокращение или расширение платформы;

- «выходной» – полные линии не собираются;
- «ударник» – собираются даже неполные линии;
- «шторм» – управление фигурой усложняется постоянным ее смещением;
- «хаос» – генератор начинает создавать процедурные фигуры;
- объединение и разделение – случайные части поля осыпаются, рассыпаются или схлопываются.
- «заморозка» – появление новых фигур и управление заблокировано;
- «гравитон» – сила гравитации уменьшается или увеличивается;
- «атака» – командный счет уменьшается;
- «защита» – защищает игрока от следующего негативного события.

За игру пользователь получает рейтинг и игровые очки, в зависимости от своего вклада. Рейтинг определяет статус игрока и помогает при поиске равного противника, открывает новые возможности в дереве навыков и т. д.

Игровые очки можно расходовать на приобретение и улучшение навыков, и улучшение генератора. Параметрами генератора являются тип и стоимость создаваемых блоков, скорость, разнообразие фигур. Навыками являются возможность вызова эффектов и их сила, глобальный модификатор получения счета, командные бонусы. Для улучшения мотивации игрока внедряются игровые задания на сбор определенного типа или количества блоков, линий, совершения определенного типа действий, побед и др.

1.3 Требования к сервису

На основе проведенного анализа и с учетом требований, указанных в задании на дипломное проектирование, были сформулированы задачи и требования к проектируемому сервису.

1.3.1 Назначение разработки

Основная задача – создать сервис архитектуры клиент-сервер на основе игры «Тетрис». Предоставить пользователю доступ к сервису для проведения досуга и развития реакции, внимания и стратегического мышления, обмена информацией с другими пользователями, участия в соревнованиях. Предоставить разработчику доступ к документации и структуре сетевого протокола для реализации возможности взаимодействия с системой «Тетрис».

1.3.2 Состав выполняемых функций

Список выполняемых сервисом функций определен на основе анализа концепций аналогов и разработки собственной концепции:

1) возможности классической игры «Тетрис»:

- а) дискретное поле с возможностью размещения блоков;
- б) случайная генерация семи классических фигур по принципу «мешок»;
- в) начисление очков за заполнение линий и их уничтожение;
- г) поворот и перемещение фигур;

- 2) расширенные возможности игры «Тетрис»:
 - а) сохранение фигуры в буфере для дальнейшего ее использования;
 - б) предпросмотр следующей фигуры для планирования действий;
- 3) система развития игрока:
 - а) механизм рейтинга игрока на основе его производительности;
 - б) система достижений, предоставляющая цели и награды за определенные действия или результаты;
 - в) дерево навыков, позволяющее игрокам развивать специальные умения и получать преимущества;
 - д) развитие генератора, который со временем предлагает более ценные фигуры;
 - е) система набора игровых очков;
- 4) игровые события:
 - а) всеобщие события;
 - б) активируемые события;
 - в) появление специальных блоков с уникальными свойствами;
 - г) временные бонусы, которые повышают производительность игрока;
 - д) события всех перечисленных типов;
- 5) командный визуальный индикатор:
 - а) визуальный индикатор, наполняющийся по мере игры;
 - б) бонусы, получаемые командой-первооткрывателем при достижении определенных целей;
- 6) система сбора рейтинга и игровых очков:
 - а) заработок очков на основе производительности игрока;
 - б) бонусы за достижения и выполнение заданий;
 - в) рейтинговая система для сравнения и подбора равных игроков;
- 7) система игровых заданий:
 - а) задания на выполнение определенных целей в игре;
 - б) бонусы и награды за успешное выполнение заданий;
 - в) постепенное увеличение сложности заданий;
- 8) однопользовательский режим для тренировки навыков и выполнения заданий:
 - а) режим игры для индивидуальной тренировки и улучшения навыков;
 - б) выполнение заданий без участия других игроков;
- 9) многопользовательский режим, одиночный и в команде:
 - а) онлайн-соревнования с другими игроками;
 - б) многопользовательский режим с командными соревнованиями;
 - в) быстрый поиск матча и игры против случайных соперников;

- г) создание частной игровой комнаты для игры с друзьями;
- д) создание публичной игровой комнаты;
- 10) визуальные эффекты:
 - а) визуальные эффекты при уничтожении линий;
 - б) особые эффекты при использовании бонусов и достижении целей, продвижении в дереве навыков;
- 11) музыкальное сопровождение и музыкальные эффекты:
 - а) фоновая музыка во время игры;
 - б) звуковые эффекты при действиях игрока;
 - в) настройка звуковых параметров;
 - г) звуковые эффекты при игровых событиях, победе и поражении;
- 12) соревновательные функции игры:
 - а) создание и учет аккаунтов игроков;
 - б) привязка игрового счета к аккаунту;
 - в) глобальная и локальная таблицы лидеров;
- 13) сетевые составляющие игры:
 - а) подключение к игре через сеть;
 - б) аутентификация и авторизация игроков;
 - в) игра без авторизации;
 - г) учет и отображение игрового состояния других игроков;
- 14) общение игроков между собой посредством текстовых сообщений и сообщений с графическими вложениями;
- 15) личный кабинет:
 - а) просмотр истории игр;
 - б) просмотр рейтинга;
 - в) приобретение навыков и бонусов;
 - г) информация должна храниться в базе данных;
- 16) функции блоков:
 - а) динамики всех перечисленных типов;
 - б) система ценности блоков;
 - в) комбинированные динамики блоков;
- 17) игровой баланс.

1.3.3 Требования к составу и параметрам технических и программных средств

Для выполнения задачи потребуются одна или две серверные машины. Требования формировались из расчета пикового использования на 1000 активных пользователей и игровых сессий [15]:

- а) процессор: Intel Xenon, 16 ядер и более, частота более 3 МГц;
- б) оперативная память: минимум 64 ГБ;
- в) хранение данных: жесткий диск объемом не менее 1 ТБ с высокой скоростью чтения/записи;

г) быстрый и стабильный интернет-канал для обеспечения своевременного обмена данными с клиентами;

д) система резервного копирования данных и мониторинга состояния сервера.

Требования к сетевому оборудованию:

а) брандмауэр и средства защиты от DDoS-атак для обеспечения безопасности сервера;

б) высокоскоростной сетевой коммутатор с поддержкой гигабитных соединений.

Требования к программным средствам:

а) используемые пакеты не должны иметь уязвимостей уровня высокий и выше;

б) версии всех пакетов должны быть новейшими на момент создания сервиса;

в) использование устаревших пакетов, влияющих на безопасность сервиса, недопустимо;

г) используемая СУБД должна быть новейшей версии на момент создания сервиса, и постоянно обновляться после.

1.3.4 Требования к информационной и программной совместимости

Требования к информационной и программной совместимости необходимы для обеспечения успешного взаимодействия и интеграции между модулями системы, их список представлен ниже:

а) поддержка стандартов передачи данных, таких как JSON, Socket.IO версии 4.0 и более, для обмена данными с клиентами;

б) совместимость с основными браузерами (Chrome, Firefox, Edge);

в) машина, на которой используется сервис, должна поддерживать WebGL2.

1.3.5 Обоснование выбора языка и сред разработки

Для достижения поставленных целей, а именно написания клиента и сервера, был выбран язык программирования высокого уровня JavaScript. Выбор обусловлен гибкостью языка, в частности JavaScript подходит для написания как браузерных, так и самостоятельных приложений, подходит для написания клиент-серверной архитектуры, для него существует множество библиотек и фреймворков, существенно упрощающих задачу программиста.

Для реализации серверной части приложения была выбрана программная платформа Node.js, предназначенная для написания серверных приложений на JavaScript. Выбор обусловлен ее кроссплатформенностью и простотой построения на ней серверных приложений, наличием качественной и полной документации. Для него так же был взят быстрый минималистичный фреймворк Express, работающий внутри среды исполнения Node.js, который имеет множество служебных методов для работы с HTTP и промежуточными обработчиками. Коммуникация между клиентом и сервером обычно производится

по протоколу HTTP, который широко распространен, однако не подходит для реализации игры по сети в реальном времени, так как является протоколом «запрос-ответ», и сервер не может оповещать клиента о изменениях состояния в реальном времени. Веб сервер приложения будет реализовывать архитектурный подход Rest API, для аутентификации и авторизации клиента, функционал чата и игрового процесса, который должен быть осуществлен в реальном времени, будет предоставляться по протоколу Socket.IO, что позволит открыть постоянное двунаправленное сетевое соединение между браузером и сервером. Для использования этого протокола была выбрана библиотека Socket.IO, написанная на JavaScript.

Для реализации клиентской части была выбрана JavaScript библиотека React. Ее выбор обусловлен наличием задачи создать интерактивный пользовательский интерфейс, наличием большого количества готовых решений и удобством написания, использования и переиспользования компонентов пользовательского интерфейса.

Для реализации графической части была взята программная библиотека для языка JavaScript React Three Fiber, адаптированная для использования библиотека Three.js для React. Она поддерживает возможности визуализации двумерной и трехмерной графики, динамического освещения и шейдинга.

Разработка сервиса не зависит от выбора операционной системы, однако для развертывания сервера предпочтительна операционная система типа Unix. Сервер, клиент и база данных будут распространяться в виде контейнера Docker для простоты переноса сервиса на любую машину, например, на арендованной машине у подходящего облачного сервиса.

Для хранения информации о пользователях, о их связях, о сообщениях и о игровом процессе будет использована СУБД MySQL – бесплатная для использования СУБД, поддерживающая широкий набор команд языка SQL, требующийся для решения задачи. Для простоты использования на JavaScript будет использоваться Sequelize – ORM-библиотека для приложений на Node.js. Для фиксации прогресса написания программного средства используется система контроля версий Git.

2 АНАЛИЗ ТРЕБОВАНИЙ К СЕРВИСУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Описание функциональности сервиса

Описание функциональности производится на основе технического задания, разработанного в первом разделе. Так как выбранная архитектура клиент-серверная, то для хранения данных необходимо разработать модель базы данных. Целевой пользователь системы один – игрок, для которого нужно разработать модель вариантов использования сервиса.

2.1.1 Варианты использования сервиса

Варианты использования сервиса сконцентрированы вокруг ее главного потребителя – игрока, которому должен быть доступен весь базовый функционал системы. Однако игрок может рассматриваться системой по-разному в зависимости от того, авторизирован он или нет, есть ли у него определенный рейтинг, приобрел ли он платную подписку. Схема вариантов использования выполнена в редакторе Enterprise Architect и представлена в приложении Б. В простейшем случае, в системе существуют два актера – игрок и авторизированный игрок. Функционал при этом можно разделить на общий, и частный – доступный только авторизированному игроку.

Игрок – пользователь, который не зарегистрирован в системе. Ему доступен общий функционал, такой как быстрый поиск соперника, который помещает игрока в случайную комнату с размером, зависящим от нагрузки сервера, переговоры с другими игроками в глобальном чате под анонимным псевдонимом, а также авторизация и регистрация, эксклюзивно для него. При регистрации позволяется выбрать игровой псевдоним, и в дальнейшем участвовать в играх под ним, накапливать игровые очки, рейтинг и участвовать в глобальном соревновании.

Авторизированному же игроку доступен весь основной функционал системы, просмотр персональной информации, которая включает историю сыгранных партий, рейтинг и тренд, сохранение статистики, отображение в таблицы лидеров, частная переписка с другими игроками, выбор комнаты перед началом игры. Предполагается, что пользователю, который приобрел платную подписку, не будет необходимо смотреть рекламу в конце игровой сессии. В личном кабинете пользователь сможет просмотреть свой баланс, и приобрести улучшения навыков и генератора за накопленные игровые очки. Ему будет доступен просмотр дерева игровых навыков, которые позволят выбрать свой путь развития и свою уникальную стратегию. В личном кабинете пользователя также доступно меню частных переписок, где пользователь может найти другого игрока по его псевдониму, и начать с ним переписку. В мессенджере можно отправлять текстовые сообщения и сообщения с вложениями-картинками, просматривать такие сообщения, в том числе в реальном времени, выделять сообщения реакциями и отвечать на них.

Всем игрокам доступен базовый функционал управления комнатой, а именно выбор команды, переписка с участниками комнаты, сигнализация о готовности к игре, и просмотр сигналов других игроков. В случае готовности всех игроков, или по истечению заданного времени сервером, игра начнется автоматически. Всем игрокам доступно базовое управление игрой, передвижение фигуры, сбор линий и накопление игрового счета. Однако авторизированному игроку доступно использование дополнительных навыков, который тот приобрел за счет игрового рейтинга. Игрок может покинуть игру досрочно, в таком случае, команда, лишившаяся игрока, получает бонус, зависящий от рейтинга ушедшего игрока, а тот в свою очередь получает штраф к рейтингу. По окончании игры проигрывается рекламный ролик. У пользователя есть возможность продолжить игру, просмотрев такой ролик.

2.1.2 Разработка инфологической модели базы данных

Для разработки инфологической модели базы данных следует определить основные сущности, их атрибуты и связи. Опираясь на требования к сервис-форуму игры «Тетрис», определенных в подразделе 1.3.2, основными сущностями базы данных можно считать игроков, сообщения в личный чатах, достижения и навыки игроков, а также игровые комнаты и рекорды – записи о статистике игры и вклада каждого отдельного игрока в командный зачет. Остальные сущности вытекают из необходимости взаимодействия и связей основных. В таблице 2.1 находятся описание сущностей инфологической модели базы данных.

Таблица 2.1 – Список сущностей инфологической модели базы данных

Отношение	Описание и назначение	Основные атрибуты	Краткое описание связей с другими отношениями	Особенности
Игрок	Содержит основную информацию о игроке	ID Статус Имя Фамилия Почта Псевдоним Рейтинг Когда создан Когда онлайн Количество связей Очки	Рекорд Рейтинг Статус (М:М) Роль Комната Комната (владелец) Сообщение Реакция Игрок (М:М) Достижение (М:М)	Рейтинг, тренд и количество связей - кешированные поля

Продолжение таблицы 2.1

Отноше- ние	Описание и назначение	Основные атрибуты	Краткое описание связей с другими отношениями	Особенности
Статус игрока	Определяет состояние игрока	ID Имя Время начала и конца	Игрок (М:М)	Перечисли- мый расширяемый тип, с каждым значением ассоциирова- но зарезервиро- ванное имя
Сообще- ние	Сообщение игрока другому игроку	ID Отправитель Получатель Время создания Время изменения Время удаления Счетчик реакций Счетчик ответов	Вложение Игрок Сообщение (М:М)	Кэширован- ный счетчик реакций и ответов Время изменения и удаления пусто по умолчанию Сообщение нельзя удалить
Вложе- ние	Прикрепленн ый файл к сообщению	ID Сообщение Имя файла	Сообщение	Файл должен быть изображением
Реакция	Реакция пользователя на сообщение	Игрок Сообщение Тип реакции	Сообщение Игрок	Не обязательно
Рекорд	Содержит информацию о игре	ID Игрок Счет Время игры	Игрок (М:М)	Может содержать и другие данные
Страны	Информация о стране	ID Короткий код Флаг малый	Игрок	Выяснить топ страну по счету, игрокам

Продолжение таблицы 2.1

Отноше- ние	Описание и назначение	Основные атрибуты	Краткое описание связей с другими отношениями	Особенности
Комната	Содержит информацию о игровой комнате, нужна для организации соревновани й, совместной онлайн игры)	ID Владелец Имя Пароль Ограничение числа участников Настройки Статус Количество команд Ограничения по рейтингу	Игрок	Пустой пароль означает что комната публичная Если ограничение равно нулю, то комната принимает любое количество
Достиже- ние	Содержит информацию о достижении и условия для получения	ID Имя Описание Требования Награда Сколько выполнило	Игрок (М:М)	Прогресс достижения, дата и срок определяются в связи
Навык	Информация о навыке	ID Имя Описание Требования Правила использования	Игрок (М:М)	Уровень навыка определяется в связи Обновление генератора пассивный это навык

В основе разработки базы данных и работы с SQL языками лежит реляционная модель данных, где каждая таблица представляет отдельную сущность, а столбцы этой таблицы представляют атрибуты [16]. Эти сущности и атрибуты отражают ключевые аспекты предметной области, учитывая требования заказчика к функциональности сервиса. Для упрощения восприятия разработки и дальнейшего расширения и редактирования, был выбран векторный редактор, в котором была создана графическая схема инфологической модели базы данных, представленная на рисунке 2.1. Выбор редактора, такого как

Enterprise Architect, обусловлен необходимостью создания графической схемы, которая не только наглядно представляет структуру базы данных, но и обеспечивает удобство последующего редактирования и расширения. Этот инструмент предоставляет возможность создавать профессиональные диаграммы, включая сущности, связи и атрибуты. Так же он позволяет создавать код для конкретной СУБД на основе созданной модели, и синхронизироваться с ней.

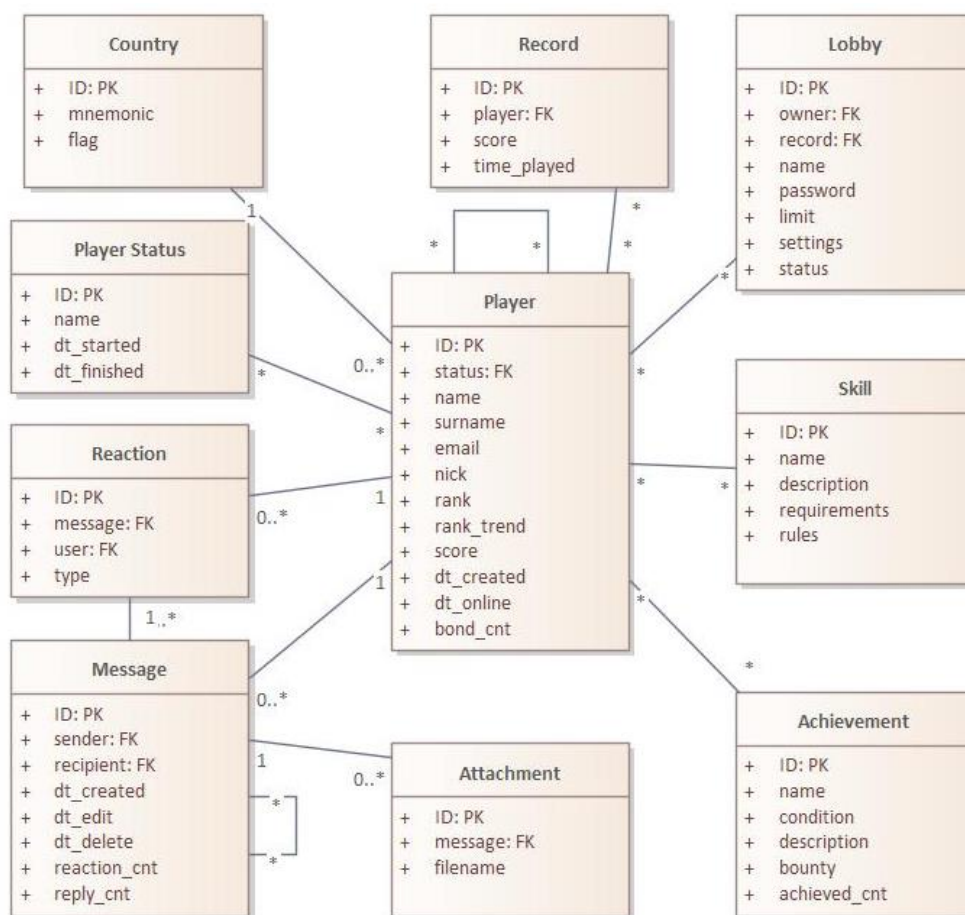


Рисунок 2.1 – Схема инфологической модели базы данных

Схема демонстрирует взаимосвязи между основными сущностями, их атрибутами и ключами. Полученная графическая схема служит основой для дальнейшей работы над физической моделью базы данных.

2.2 Спецификация функциональных требований

Сервис-форум игры «Тетрис» представляет из себя онлайн-сервис для пользователей, на котором они могут играть как в одиночную игру, так и с другими игроками, в том числе на других устройствах, вести как частные, так и публичные обсуждения, делиться своими рекордами. Сервис должен вести

общую таблицу лидеров. Список основных аспектов, по отношению к которым будут составлены требования, представлен ниже:

- основной игровой процесс;
- игровой баланс;
- сетевая и командная игра;
- система игровых аккаунтов;
- мессенджер;
- визуальное и музыкальное сопровождение;
- управление.

Группа требований «И» – требования к игровому процессу, охватывают только непосредственное времяпрепровождения в игре, и нужны, чтобы обеспечить качество игрового процесса. Группа требований «Б» – требования к игровому балансу, необходимы, чтобы игрок не терял интерес по причине чрезмерной сложности или простоты игры. Группа требований «К» – требования к сетевой и командной игре, необходимы, чтобы этот режим ощущался более динамично и оставался предпочтительным. Группа требований «А» – требования к системе игровых аккаунтов, необходимы, чтобы игрок мог получить быстрый доступ ко всему основному функционалу игры и основной информации о игре. Группа требований «Ч» – требования к мессенджеру, необходимы, чтобы игрок имел полноценные возможности для обмена информацией, реакциями и изображениями. Группа требований «В» – требования к визуальному и музыкальному сопровождению, необходимы для создания положительного впечатления, настроения или эмоционального отклика у аудитории. Одновременно с этим, критически важна информативность интерфейса и способ представления информации. Группа требований «У» – требования к управлению, необходимы для повышения комфорта и удобства использования. Требования по управлению определяют, какие элементы управления должны быть доступны, как они должны быть размещены, и каким образом они должны работать. Список всех функциональных требований представлен в таблице 2.2.

Таблица 2.2 – Список функциональных требований к сервису

№	Описание
Группа «И» – требования к основному игровому процессу	
И1	Игровой процесс должен быть доступен как оффлайн или без регистрации в системе, так и онлайн или с регистрацией и авторизацией пользователя
И2	Сервис должен обеспечивать игровой процесс в соответствии с классическими и расширенными правилами игры «Тетрис», определенных в техническом задании
И3	Сервис должен обеспечивать динамику игрового процесса путем увеличения сложности игры
И4	Пользователь может приостановить игровой процесс с сохранением результатов на время всей игровой сессии

Продолжение таблицы 2.2

№	Описание
Группа «И» – требования к основному игровому процессу	
И5	Игровая сессия считается завершенной, если закончился игровой таймер, или грузоподъемность платформы была превышена, или пользователь закрыл вкладку с игрой, или соединение с сервером было разорвано иным способом
И6	Буфер может использоваться один раз за смену фигуры на игровом поле (использование буфера не является сменой фигуры). Если буфер заполнен, фигура, находящаяся в нем ранее, становится активной частью игрового поля
И7	Пользователь может досрочно осуществить смену фигуры, поместив текущую фигуру в максимально возможное нижнее положение или ускорить ее перемещение
И8	Игровые навыки могут использоваться при соблюдении всех необходимых для них условий
И9	Сложность игры влияет на скорость падения фигур, частоту и силу событий, уровень генератора, грузоподъемность платформы
И10	Уровень генератора влияет на стоимость и пропорцию типов генерируемых блоков
И11	Пользователь может защититься от поражения, просмотрев рекламное видео
И12	После завершения игровой сессии ее результат сохраняется со статусом завершения (естественно, досрочно или аварийно)
И13	Пользователь должен иметь возможность получать достижения за игровой прогресс
Группа «Б» – требования к игровому балансу	
Б1	Продвижение по дереву навыков и генератора не должно давать чрезмерное преимущество игроку в своей рейтинговой категории
Б2	Подбор игроков для матчей должен происходить в одной рейтинговой категории
Б3	Скорость накопления рейтинга должна быть такая, чтобы игрок не мог продвинуться в категории, сыграв менее 10 матчей
Б4	Скорость накопления очков должна соответствовать категории игрока, и скорости роста стоимости продвижения по дереву навыков
Б5	Скорость роста стоимости не должна быть степенной или экспоненциальной
Б6	Рост доходов с генератора должен быть сбалансирован с ростом стоимости навыков и их силы
Б7	Скорость заполнения объема победной шкалы должен быть близок к постоянному для всех категорий игроков
Б8	У навыков должно быть ограничение на скорость использования

Продолжение таблицы 2.2

№	Описание
Группа «Б» – требования к игровому балансу	
Б9	Внутриигровые покупки не должны нарушать игровой баланс и создавать проблемы другим игрокам сервиса
Группа «К» – требования к сетевой и командной игре	
К1	Если ни один из игроков либо одна из команд не могут получить решающее преимущество, игра должна увеличить сложность
К2	Время игры ограничено. Если по окончании таймера ни одна из команд не получила преимущество, объявляется ничья
К3	Авторизированный пользователь должен иметь возможность просматривать список доступных игровых комнат
К4	Пользователь должен иметь возможность присоединиться к случайной комнате
К5	Авторизированный пользователь должен иметь возможность создавать свою игровую комнату за свой счет
К6	Авторизированный пользователь может присоединиться к одной из комнат, при наличии в ней свободных мест и знания пароля, если комната не публичная
К7	Выход из игровой комнаты происходит автоматически при завершении игровой сессии
К8	В игровой комнате пользователь должен иметь возможность просматривать внутренний чат, внутреннюю таблицу лидеров и игровое состояние других игроков, подключенных к комнате (не одновременно)
К9	Пользователь должен иметь возможность сигнализировать о готовности к соревнованию и просматривать состояния готовности других пользователей
К10	Соревнование должно начинаться автоматически при достижении состояния готовности всех пользователей
К11	Время отклика игры должен не зависеть от скорости интернет-соединения
К12	Информация о состоянии соревнования, а также о подключениях и отключениях пользователей в комнату должна дублироваться во внутренний чат
К13	По окончании игры пользователь должен увидеть свой вклад в игру и статус игры (поражение, победа, ничья)
Группа «А» – требования к системе игровых аккаунтов	
А1	Игровой аккаунт доступен только при регистрации и авторизации пользователя
А2	Пользователь может просматривать свои достижения и прогресс к ним, забирать награды

Продолжение таблицы 2.2

№	Описание
Группа «А» – требования к системе игровых аккаунтов	
A3	Игровой аккаунт содержит следующую информацию: имя, фамилия, почта, страна, псевдоним, место в таблице лидеров, рейтинг, счет, короткая текстовая информация, а также информацию о играх пользователя, список пользователей, с которыми можно вести коммуникацию
A4	Пользователь может наблюдать свою позицию в таблице лидеров не зависимо от своего результата. Таблица лидеров генерируется динамически
A5	Пользователь может взять себе любой незанятый псевдоним, состоящий из латинских символов и символа ‘_’
A6	Пользователь может видеть платные опции и совершать покупки
A7	Пользователь может просматривать свои навыки, требования к ним, и расходовать счет на них
A8	Пользователь не может просматривать аккаунты других игроков
A9	Пользователь может выбрать игровую комнату для совершения игрового процесса по параметрам количество пользователей, количество команд, а также перейти в меню управления своей комнатой
Группа «Ч» – требования к мессенджеру	
Ч1	Пользователь может отправлять текстовые сообщения, которые видны только конкретному пользователю
Ч2	Обмен информацией между пользователями происходит через сообщения
Ч3	Сообщения без целевого пользователя располагаются на его персональной странице
Ч4	Пользователь может прикреплять до девяти графических файлов к сообщению общим объемом не более 100 мегабайт
Ч5	Пользователь может редактировать сообщения (только текст), при этом собеседник должен видеть, что сообщение было изменено
Ч6	Пользователь не может удалять сообщения, а только помечать их как удаленные
Ч7	Пользователь может отвечать на другие сообщения
Ч8	Пользователь может реагировать на другие сообщения
Ч9	У сообщения видно количество ответов на него, если оно более нуля
Ч10	Пользователь производит поиск собеседника по его псевдониму
Ч11	Отправлять сообщения пользователю можно только если тот принял запрос на переписку

Продолжение таблицы 2.2

№	Описание
Группа «В» – требования к визуальному и музыкальному сопровождению	
B1	События и навыки должны объявляться при их возникновении или использовании, все время действия пользователь явно должен видеть и слышать активность
B2	Открытие навыков и достижений должно сопровождаться аудиоэффектами
B3	Сбор линий и получение счета должны сопровождаться аудиовизуальными эффектами
B4	Внешний вид блоков должен быть представлен явно и четко, должны быть однозначно понятны тип и стоимость блока
B5	Интерфейс приложения должен быть минималистичен
B6	Информация о состоянии соревнования, а также о подключениях и отключениях пользователей в комнату должна сопровождаться аудиосигналами
B7	Рейтинг пользователя должен быть представлен визуально
B8	Платформа должна окрашиваться в цвет, который является индикатором заполнения игрового поля и близости игрока к поражению
B9	Должна быть возможность регулировки громкости звуковых эффектов, основной мелодии игры, полного отключения звуковых эффектов, основной мелодии игры
Группа «У» – требования к управлению	
У1	Управление игрой должно осуществляться и мышью, и с клавиатуры, при этом клавиши должны подсказываться рядом с функцией, что они выполняют
У2	Управление фигурой осуществляется стрелками или клавишами WASD
У3	Активация событий должна происходить нажатием одной клавиши, при возможности начинающей мнемонику
У4	Управление чатом должно базироваться на управлении существующих аналогов

На основе этих функциональных требований в дальнейшем будет разработан проект сервиса и произведено его тестирование. Это обеспечит четкое понимание того, что должно быть реализовано в сервисе и как обеспечить качественный игровой опыт для пользователей.

3 ПРОЕКТИРОВАНИЕ СЕРВИСА

3.1 Разработка архитектуры сервиса

На основе списка функциональных требований к сервису, определенных в разделе 2.2, задача разделяется на модули. Так как выбранная архитектура клиент-серверная, необходимо разработать структуру клиента и сервера, базу данных и сетевой протокол взаимодействия между клиентом и сервером. Список модулей, которые необходимо спроектировать, сводится к следующим:

- игровая логика;
- мессенджер;
- взаимодействие клиент-сервер;
- графический модуль;
- аутентификация и авторизация;
- игровые события и развитие;
- компоненты пользовательского интерфейса.

Игровая логика включает в себя базовый и расширенный функционал игры «Тетрис», и не требует серьезных математических решений. Отдельно следует рассмотреть новые динамики блоков и формулы получения игрового счета.

Графическая составляющая в игре «Тетрис» реализована на основе заранее заготовленных примитивов. Дополнительные эффекты создаются с помощью шейдеров.

Музыкальное сопровождение включает классическую мелодию игры «Тетрис», звуки специальных эффектов, звуки нажатий и переходов.

Сетевой протокол формируется с целью организовать связь между клиентом и сервером, между пользователями, и должен иметь возможность обмена информацией в режиме реального времени, должен содержать функции для обмена сообщениями и синхронизации игрового процесса.

Для хранения информации о пользователях, о их связях, о сообщениях и о игровом процессе требуется спроектировать базу данных.

3.2 Разработка логической и физической модели базы данных

База данных подверглась процессу нормализации для устранения избыточности данных и улучшения структуры. Для соблюдения третьей нормальной формы раскрыты связи типа многие ко многим и созданы вспомогательные таблицы для эффективной реализации отношений между сущностями. Некоторые таблицы потребовали включения в них дополнительной информации, список таких сущностей представлен в таблице 3.1. Остальные нормальные формы для модели не являются критическими.

Таблица 3.1 – Список сущностей для раскрытия связей типа многие ко многим

Отноше- ние	Основные атрибуты	Отношения, между которыми раскрывается связь	Особенности
Статус игрока	Время начала Время окончания	Игрок Статус игрока	У игрока может быть множество статусов. Перспективное развитие
Связь между игроками	Статус связи Время создания, изменения	Игрок	Статусы связи: отправлена, принята, отклонена, удалена
Достиже- ние игрока	Время создания, получения награды	Достижение Игрок	Факт приобретения и получения достижения
Навык игрока	Время создания, получения, уровень навыка	Навык Игрок	Уровень навыка игрока
Рекорд игрока	Персональный счет Рейтинг Место Количество линий Навыков использовано	Игрок Рекорд	Рекорд – запись о командной игре. Это параметры персонального вклада
Место игрока в комнате	Готовность Игровой счет Команда игрока	Игрок Комната	Информация о готовности и о команде игрока, может быть пустой. Игрок обязан быть в команде для начала игры

3.3 Разработка алгоритмов модулей сервиса

3.3.1 Проектирование модуля игровой логики

Игровой модуль, схема алгоритма работы которого представлена на рисунке 3.1, представляет из себя детерминированный конечный автомат с единственным входом (код события), состоянием, и выходными сигналами отрисовки и конца игры. Детерминированность обусловлена требованиями к предметной области. Для сохранения разнообразия игрового процесса вводится понятие ключа генератора. При наличии такого ключа вся последовательность генерации может быть полностью воспроизведена. Для сохранения тайны иг-

рового процесса ключ генерации создается на сервере случайным неизвестным для клиента образом.

Базовая логика игры «Тетрис» предельно проста и сводится к рассмотрению двух частей игрового поля – подвижной и неподвижной. В момент обновления меняется позиция фигуры – подвижной части поля, и проверка, является ли такая позиция возможной (т. е. пересекается ли фигура с уже расположенными – неподвижной частью поля). Есть два типа невозможных позиций – обратимая и необратимая. Если ситуация обратимая – коллизия произошла из-за вращения или сдвига вправо-влево, то фигуре возвращается исходная позиция. Если нет – то фигура присоединяется к неподвижной части поля, применяя все правила динамики фигуры. Если в этот момент в неподвижной части поля появляется полностью заполненная горизонтальная линия – ее необходимо удалить, увеличив при этом модификатор счета игрока. Далее на поле помещается новая подвижная фигура и цикл замыкается. Если место для новой фигуры нет – игра считается завершенной.

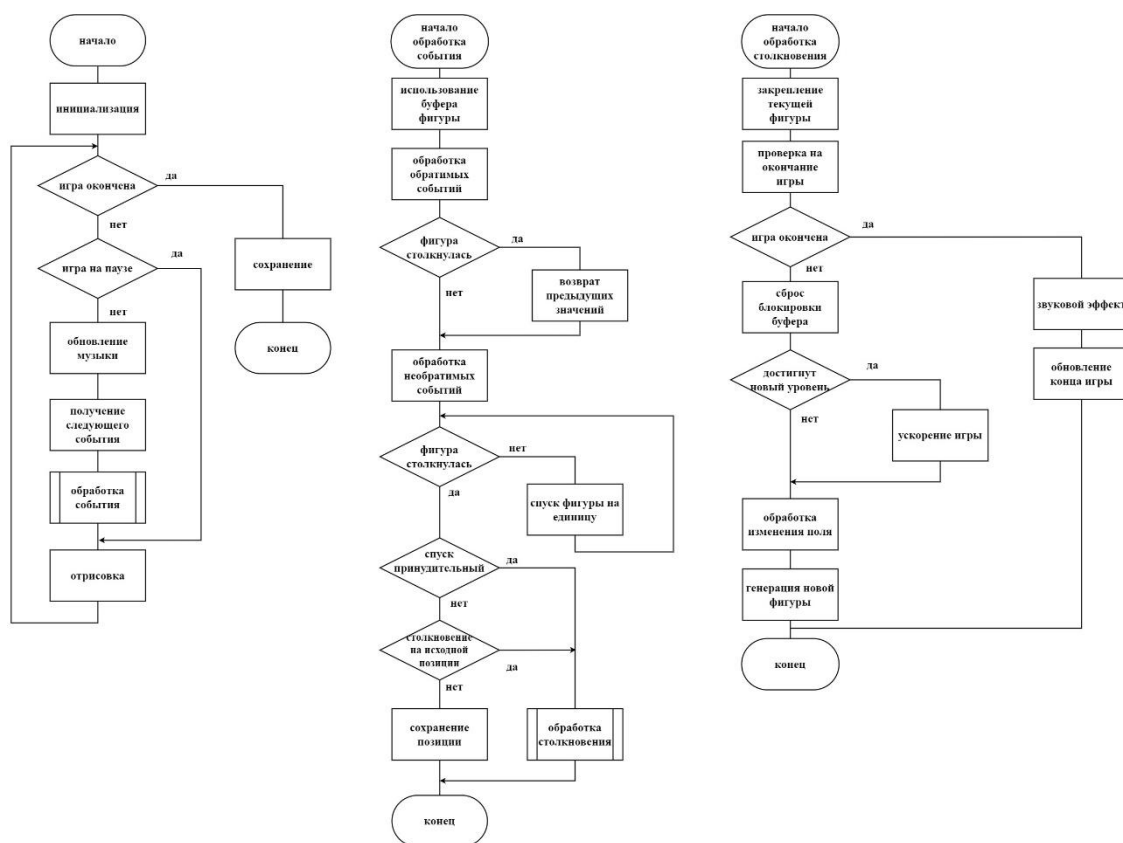


Рисунок 3.1 – Схема алгоритма игрового модуля

Обновление игрового процесса вызывается по таймеру обновлений, где время до следующего срабатывания определяется динамически исходя из скорости обновлений игры. Так как время срабатывания таймера не детерминировано, существует обязательное правило, которое исходит из требования к детерминированности алгоритма восстановления согласованности клиента и

сервера – если наступило время обновления и событие таймера не успело произойти – обновление вызывается принудительно до обработки другого события.

Алгоритм расширенной версии игры выполнен согласно механикам, указанным в подразделе 1.2.1 и функциональным требованиям, указанным в подразделе 1.3.2, пункт 16, схема алгоритма представлена в приложении В. Применение динамики блоков привязано к игровому таймеру событий.

Согласно требованию И10, для обеспечения динамики игрового процесса требуется разработать принцип генерации блоков, определение стоимости и типа каждого следующего блока. Для этого вводится понятие базовой стоимости блока и базового типа блока. Генератор на каждом уровне имеет определенный шанс выпадения того или иного типа блока. С каждым уровнем генератора игрок может поднять вероятность выпадения тех или иных блоков, а общий уровень генератора увеличивает шанс выпадения более ценных блоков. Вероятность выпадения вычисляется через подсчет математического ожидания средней стоимости притока. В таблице 3.2 указана зависимость доли выпадения блока от уровня генератора, а также математическое ожидание стоимости блока, коэффициент повышения стоимости. Коэффициент стоимости, как и скорость притока, растет по принципу ряда Фибоначчи – простой и удобный способ создавать прогрессирующие математические системы [17]. Коэффициент сложности – разница в скорости притока и цены – должен определяться динамически исходя из прогресса игрока.

Таблица 3.2 – Зависимость между уровнем генератора и его параметрами

Уровень	Математическое ожидание	Доля выпадения	Коэффициент стоимости
1	10	0	1
2	15	1	2
3	25	1,5	3
4	40	2,5	5
5	65	4	8
6	105	6,5	13
7	170	10,5	21
8	275	17	34
9	445	27,5	55
10	720	44,5	89

3.3.2 Проектирование мессенджера

Проектирование мессенджера велось в соответствии 2.2.5, спецификация функциональных требований для мессенджера. Пользователю на одной странице доступно сразу все возможные взаимодействия с мессенджером: выбор получателя, просмотр переписки с ним, получение обновлений переписки с ним в реальном времени, возможность отправки, изменения, удаления сооб-

щений и прикрепления к ним до девяти файлов изображений. Согласно функциональному требованию У4, управление мессенджером проектировалось с опорой на существующий популярный мессенджер Telegram. Для ввода сообщения предусматривается многострочное поле для ввода с возможностью указать до девяти файлов вложений, рядом с каждым сообщением есть кнопки для удаления, редактирования, реакции или ответа. Для выбора собеседника предусматривается поисковая строка. Поиск происходит по нажатию кнопки «Поиск». При отсутствии связи, целевому пользователю отправляется запрос в виде сообщения. Переписка при этом недоступна в оба направления. Статус связи и его ранг отображаются рядом с псевдонимом пользователя. Сообщения отправителя располагаются справа, получателя слева. Обмен сообщениями происходит в реальном времени, для этого не требуется обновлять страницу и совершать иные действия.

3.3.3 Проектирование взаимодействия клиент-сервер

Основой как сервера, так и клиента является цикл обработки событий, которые могут возникать при взаимодействии сервера с клиентом, от пользователя и от таймера обновлений. Краткая схема алгоритма взаимодействия изложена в приложении Г.

Основная проблема создания клиент-серверного игрового приложения является задача сохранения авторитета сервера при принятии решений с сохранением комфорта игрового процесса для клиента, требование K11. Так как сервер является главным в принятии решений, то клиент обязуется согласовывать все свои действия с сервером. Это может стать проблемой, так как при взаимодействии клиента и сервера существует некая ощутимая для пользователя задержка, которая вызывает дискомфорт и неудобство пользования приложением. Решением этой проблемы является реализация модуля предсказаний и восстановления согласованности на стороне клиента [18].

Смысл заключается в том, что при подключении к серверу на нем создается игровой процесс, который идентичен игровому процессу на клиенте. После клиент, получая событие пользовательского ввода или таймера обновлений, обрабатывает это событие, и получает новое состояние игры. Это состояние становится текущим состоянием и отображается пользователю, а также заносится в циклический буфер, необходимый для отслеживания изменений состояния на клиенте. Происходит это практически мгновенно, как если бы пользователь играл в одиночную игру. Однако параллельно этому полученное событие отправляется на сервер, где помещается в очередь событий. Сервер с определенной фиксированной скоростью по очереди опустошает очереди событий всех клиентов, последовательно обрабатывая события. Важно, чтобы обработка события при одинаковом исходном игровом состоянии обязательно порождала идентичный результат, то есть игра должна быть детерминированной. Таким образом, игровой процесс образует детерминированный конечный автомат. После обработки событий сервер отправляет итоговое игровое состояние на клиент. Клиент принимает состояние сервера, которое всегда является

устаревшим, и сравнивает его с соответствующим сохраненным состоянием в циклическом буфере. Если зафиксировано несовпадение – значит произошла ошибка или рассинхронизация, и требуется согласование. Согласование происходит следующим образом: игровое состояние откатывается до серверного игрового состояния, затем все события, прошедшие с момента серверного состояния, выполняются вновь, все промежуточные порождаемые состояния также сохраняются в циклический буфер. Все это происходит за один момент в цикле обработки так, что изменения не мешают игровому процессу пользователя. Одновременно с этим состояние на сервере считается единственно верным, и именно оно будет являться определяющим при выставлении счета игроку.

При отсутствии подключения к серверу клиент попросту не отправляет и не ожидает синхронизации от сервера, продолжая игровой процесс локально. Важно, что при возобновлении соединения пользователь не может заставить сервер принять состояние клиента за истинное, то есть сохранение результата этого игрового процесса уже невозможно.

Для определения текущего состояния соединения клиента с сервером были введены следующие состояния:

- @OFFLINE – соединение с сервером отсутствует или было разорвано;
- @ON-LINE – соединение с сервером установлено, клиент ожидает регистрации;
- @REGSTRD – клиент зарегистрирован в сети и готов к работе. В этом режиме все возможности доступны;
- @REJECTED – клиенту отказано в регистрации;
- @CNCLOST – соединение с сервером было прервано, дальнейший игровой процесс не может быть сохранен.

3.3.4 Проектирование графического модуля

Графический модуль проектируется согласно списку функциональных требований к графическому модулю, подраздел 2.2.6. Основные задачи графического модуля: визуализация отдельных компонент игры в единую сцену. Для выполнения данной задачи используется обертка React Three Fiber над графической библиотекой Three.js, предназначенной для кроссплатформенной визуализации трехмерной графики. Выбор обусловлен наличием исчерпывающей документации, обилием примеров использования в общем доступе, возможностью расширения с двумерной до трехмерной графики и наоборот, гибкая интеграция с React.

3.3.5 Проектирование модуля аутентификации и авторизации

Так как доступ к игре должен быть максимально простым и доступным, для этого можно использовать технологию единого входа – метод аутентификации, который позволяет пользователям безопасно аутентифицироваться сразу в нескольких приложениях и сайтах, используя один набор учетных данных [19]. Эта технология позволяет пользователю быстро зайти на сайт, не

требуя от него никаких дополнительных действий и информации. Таким образом, пользователь может быстро входить на сайт без использования пароля, в базе данных нет необходимости хранить чувствительную информацию, а проект модуля сводится к сохранению токена пользователя в базе данных.

3.3.6 Проектирование модуля игровых событий и развития

Модуль игровых событий и развития проектируется согласно списку функциональных требований к графическому модулю, подраздел 2.2.2 и 2.2.3. Основная задача определить параметры и способ представления информации о событиях, интегрировать их в игровой процесс. Так как основной игровой процесс является конечным автоматом, то события легко интегрируются в общую систему, а принцип их возникновения определяется настройками комнаты. Параметрами события являются: вероятность возникновения, сила и длительность эффекта события. Все эти параметры хранятся в отдельной таблице в базе данных, а их обработчик идентичен на сервере и клиенте.

3.3.7 Проектирование компонент пользовательского интерфейса

Пользовательский интерфейс игрового окна состоит из трех главных участков – поле для игры, поля для текста и места для предпросмотра фигур, и одного дополнительного участка – поле для меню.

Для их отображения на экран используется модуль для работы с графикой. Основную часть места занимает игровое поле, а дополнительная информация располагается справа от него. Цвета были выбраны максимально контрастными для удобства восприятия, а клавиши для управления – максимально знакомые пользователю с учетом необходимости привыкания и запоминания и возможности случайного нажатия.

Участок меню на момент разработки предназначен для отображения внутреннего состояния комнаты согласно требованиям K9 и K12.

Интерфейс остального приложения разрабатывался в едином стиле, с сохранением удобства пользования, четкости и ясности происходящего. Игровой интерфейс определен в требованиях к графике в подразделе 3.3.4.

3.3.8 Проектирование игровой комнаты

Игровая комната предназначена для самостоятельной организации пользователями командной сетевой игры. Схема алгоритма игровой комнаты представлена в приложении Д. Игровая комната проектируется в соответствии требованиями к сетевой и командной игре K3-K10, K12. Игрок попадает в комнату двумя способами, через выбор игровой комнаты в игровом аккаунте или быстрый поиск соперника на главной странице сервиса. Игровая комната должна содержать актуальный список игроков, возможность присоединения к любой из команд, а также обмен сообщениями между участниками комнаты для самоорганизации – для этого разрабатывается отдельный механизм обмена сообщениями внутри комнаты.

4 СОЗДАНИЕ СЕРВИСА

4.1 Общие принципы

Создание сервиса заключается в создании модулей клиента (внешнего интерфейса), сервера (игровой логики) и базы данных, а также протокола их взаимодействия. Для осуществления контроля версий и удобства разработки был создан репозиторий GitHub. Для упрощения развертывания сервиса предполагается упаковка каждого модуля в контейнеры Docker. Так как проект требует создания игровой логики, то разработка на чистом JavaScript будет затруднительной по причине отсутствия контроля типизации и высокой сложности отладки логических модулей. Проект для работы будет использовать сторонние пакеты, которые управляются пакетным менеджером npm. Список основных зависимостей первого уровня проекта сервера представлен в таблице 4.1, клиента в таблице 4.2.

Таблица 4.1 – Список основных зависимостей проекта сервера

Наименование	Описание	Причина использования
Express	Быстрый, гибкий минималистичный фреймворк для Node.js. Используется для создания веб-приложений	Предоставляет ряд абстракций для серверной логики, обработка сессий, куков, файлов
Sequelize	Гибкий, простой в использовании фреймворк для взаимодействия с базами данных	Предлагает абстракцию для взаимодействия с базой данных, не требует написания SQL запросов вручную, предлагает асинхронный интерфейс
Socket.IO	Дуплексный протокол взаимодействия между клиентом и сервером в реальном времени, основан на механике событий	Для создания игры в реальном времени подходит идеально, есть встроенная возможность деления подключений на комнаты, гибкая рассылка сообщений
TypeScript	Надстройка над JavaScript, которая позволяет использовать строгую типизацию	Упрощает процесс создания игровой логики за счет контроля типов
nodemon	Автоматически отслеживает изменения в файлах проекта и перезапускает его по мере необходимости	Значительно ускоряет и упрощает отладку

Для развертывания интерфейса веб-приложения используется инструмент Nginx – TCP/UDP прокси сервер. Веб-приложение имеет следующие точки входа: расположение «/socket-io» – для соединения по протоколу Socket.IO, здесь изменяется протокол взаимодействия, а дальнейшие запросы к перенаправляются серверу игровой логики. Точки входа «/api», «/images», «/attachments» также перенаправляются. Удобство такого подхода заключается в том, что клиенту достаточно взаимодействия только с одним сервером.

Таблица 4.2 – Список основных зависимостей проекта клиента

Наименование	Описание	Причина использования
React	Библиотека для создания пользовательский интерфейсов	Необходимость создания пользовательского интерфейса. Наличие подробной документации
Socket.IO client	Дуплексный протокол взаимодействия между клиентом и сервером в реальном времени, основан на механике событий	Необходим для взаимодействия с сервером по протоколу Socket.IO
React Three Fiber, three.js	Библиотека высокого уровня для создания 2D и 3D графики на React	Необходимо создание графического отображения игрового процесса
Crypto	Библиотека для безопасной генерации случайных чисел, хеш-строк и ключей	Необходима для генераций ключей сессии, а также при аутентификации и авторизации клиента

В дальнейших разделах представлена детальная информация о разработке игровой логики и интерфейса.

4.2 Разработка клиента

Клиент, или внешний интерфейс, состоит из следующих модулей: модуль взаимодействия с сервером – обеспечивает взаимосвязь между игровым состоянием на сервере и на клиенте, компоненты игрового пользовательского интерфейса игры, компоненты пользовательского интерфейса аккаунта, маршрутизатор.

4.2.1 Разработка модуля взаимодействия с сервером

Модуль взаимодействия с сервером необходим для создания клиентской части алгоритма восстановления согласованности клиента и сервера, и представляет собой объект-обертку над игровой логикой с использованием обмена событиями по протоколу Socket.IO. Упрощенная схема алгоритма модуля

представлена в приложении Г. Текст программы модуля представлен в приложении Ж. Список полей и методов обертки представлен в таблице 4.3.

Таблица 4.3 – Список полей и методов модуля взаимодействия с сервером

Название поля/ метода	Тип поля/ возвращаемый тип	Параметры	Назначение
game	Tetris	—	Игровое состояние, логика
socket	Экземпляр Socket.IO client	—	Поддержание сетевого взаимодействия с сервером
Объекты времени	Время в долях миллисекунд	—	Расчет точного времени того или иного события, определение порядка событий
Буферы событий и состояний	Кольцевой буфер игровых событий и состояний	—	Запись истории игровых состояний для восстановления согласованности
timer	Handle	—	Таймер вызова обработчика onTimer()
constructor()	—	Tetris, Socket.IO client	Инициализация всех полей, запуск таймера
onTimer()	—	—	Определение необходимости обработки события игрового поля
onServerUpdate()	—	Игровое состояние	Событие от сервера. Проверяется необходимость восстановления согласованности
onServerSynced()	—	Игровое состояние	Событие от сервера. Инициализация всех полей, принудительная перезапись игрового состояния состоянием сервера

Продолжение таблицы 4.3

Название поля/ метода	Тип поля/ возвращаемый тип	Параметры	Назначение
reconcile()	—	—	Выполняет восстановление согласованности, применяя все события повторно поверх состояния, пришедшего с сервера
processEvent()	—	Номер игрового события	Гарантирует стабильность событий обновления игрового поля, обрабатывает событие, пополняя буфер

4.2.2 Разработка компонент пользовательского интерфейса

Компоненты пользовательского интерфейса разрабатывались с использованием библиотеки React. Основная идея React – разработка и повторное использование отдельных компонентов пользовательского интерфейса. Список разработанных компонент и их назначение представлен в таблице 4.4.

Таблица 4.4 – Список компонент пользовательского интерфейса

Название	Включаемые компоненты	Назначение
App	Game, RegisterForm, LoginForm, Account, Game, NotFound	Главная страница, маршрутизатор для основных компонент сервиса
RegisterForm	InputField, Loader	Форма для регистрации, содержит поля для ввода данных регистрации, обработчик нажатий, ссылки на регистрацию и игру
InputField, TextField	—	Обертка React для упрощения взаимодействия и снятия значений с HTML компонент input
LoginForm	InputField, Loader	Форма для авторизации, содержит поля для ввода данных регистрации, обработчик нажатий, ссылки на регистрацию и игру

Продолжение таблицы 4.4

Название	Включаемые компоненты	Назначение
NotFound	—	Отображение информации о том, что запрашиваемый ресурс не был найден. Заглушка, путь по умолчанию
Account	UserCard, Personal, Chat, Rooms, Help, Leaderboard, RoomCreate	Аккаунт пользователя, маршрутизатор для компонент аккаунта
UserCard	—	Содержит основную информацию о пользователе. Строится на объекте системы «Игрок»
Personal	—	Содержит список рекордов игрока. Строится на объекте системы «Рекорд»
Help	—	Содержит список вспомогательной информации о игре
Leaderboard	SortableTable	Содержит таблицу рекордов. Строится на объекте системы «Таблица лидеров»
Rooms	SortableTable	Содержит таблицу комнат. Строится на объекте системы «Комната»
SortableTable	—	Вспомогательный компонент для работы с интерактивными таблицами. Автоматически разбирает массив объектов по предоставленному шаблону, позволяет задавать заголовки и сортируемые поля
RoomCreate	InputField	Форма для создания комнаты, содержит поля для ввода данных комнаты, обработчик нажатий
Chat	ListContainer, MessageContainer, ResizeableInput	Отображает чат между пользователями. Содержит список игроков для общения, список сообщений, и поля для ввода, а также обработчик для связи с сервером
ListContainer	—	Вспомогательный компонент для отображения интерактивного списка по заданному шаблону

Продолжение таблицы 4.4

Название	Включаемые компоненты	Назначение
Message Container	Message	Отображает список сообщений. Главная задача, обеспечить постепенную загрузку сообщений и автоматическую прокрутку вниз
Message	—	Вспомогательный компонент для отображения сообщения. Строится на объектах системы «Сообщение» и «Вложение». Содержит все обработчики изменения сообщения: удаление, изменение, реакция, ответ
Game	GameCanvas, RoomLobby, Loader, GameTitle	Главная страница игры, маршрутизатор для основных компонент игры
Loader	—	Отображение бесконечной загрузки
GameTitle	TitlePrompt, TetrisFigure	Отображение игровой заставки — падающих фигур «Тетрис», и главного игрового меню
TitlePrompt	—	Графический интерактивный примитив 2D текста
RoomLobby	TextField	Отображение состояния игровой комнаты. Обработка связи с сервером. Строится на объектах системы «Комната», «Игрок»
GameCanvas	GameDisplay, ProgressBar, TetrisText, TetrisFigure	Отображение игрового состояния, а также информации о игроке и о комнате, если есть. Строится на объектах системы «Комната», «Игрок», «Игра». Обработка связи с сервером, обработка событий через игровой модуль взаимодействия с сервером
ProgressBar	—	Отображение состояния соревнования
TetrisText	—	Графический примитив 2D текста
GameDisplay	TetrisFigure, TetrisEffects, TetrisField	Отображение игрового состояния
TetrisEffects	—	Отображение специальных эффектов поля игры

Продолжение таблицы 4.4

Название	Включаемые компоненты	Назначение
TetrisField	TetrisBlock	Отображение поля игры как матрицы блоков в соответствии с их типами
TetrisFigure	TetrisBlock	Отображение фигуры игры в соответствии с ее позицией. Строится на объекте системы «Фигура»
TetrisBlock	—	Отображение блока согласно его типу, стоимости и цвету

После компоновки всех модулей получается готовый пользовательский интерфейс.

4.2.3 Выбор стилей

Для упрощения разработки все компоненты интерфейса были выполнены согласно цветовой палитре, все цвета были вынесены в отдельный файл стилей. Цвета выбраны в пастельных тонах. Используемые цвета представлены в таблице 4.5.

Таблица 4.5 – Цветовая палитра пользовательского интерфейса

Название цвета	Код	Назначение	Пример
shadow-color	#f2f2f288	Тень для панелей	
background-color	#f2f2f2	Фон сайта	
card-color-primary	#f7f7f7	Основной цвет панели	
card-color-secondary	#f2f2f2	Цвет второстепенной панели	
card-color-third	#dad9e4	Цвет панели третьего порядка	
text-color-primary	#3d405c	Цвет основного текста	
text-color-secondary	#adabbe	Цвет второстепенного текста	
team-color-blue	#84aaff	Цвет команды 1	
team-color-red	#ff8484	Цвет команды 2	
team-color-green	#9fff84	Цвет команды 3	
team-color-yellow	#fff584	Цвет команды 4	

Существуют разные представления о стиле пользовательского интерфейса, на дату разработки считается современным стиль окон с закруглениями углов панелей, в таком стиле выполнена новая Windows 11. Таким образом, панели приложения выполнены со скруглениями на углах.

4.3 Разработка сервера

Сервер, или игровая логика, состоит из следующих частей: модуль регистрации, аутентификации и авторизации клиента, модуль взаимодействия с клиентом, модуль чата, а также базы данных.

4.3.1 Разработка модуля регистрации, аутентификации и авторизации клиента

Модуль регистрации, аутентификации и авторизации клиента требуется для обеспечения безопасности и управления доступом в системе или приложении. Он также позволяет пользователям создавать учетные записи. Список функций и их назначений модуля представлен в таблице 4.6.

Таблица 4.6 – Методы модуля регистрации и авторизации клиента

Название метода	Параметры	Назначение
generate AccessToken	Идентификатор пользователя и его роль	Создание нового токена для пользователя
authenticate Token	Заголовки запроса HTTP	Промежуточная функция Express для проверки переданного токена на правильность, аутентификация пользователя
is Authenticated	Socket	Как authenticateToken, но для протокола Socket.IO. Подключает информацию из токена внутрь сессии Socket.IO
register	Тело запроса HTTP формы регистрации	Проверка данных для регистрации, создание нового пользователя если его еще нет в системе
login	Тело запроса HTTP формы авторизации	Проверка данных для авторизации, проверка наличия пользователя в системе, установка статуса «в сети», установка куки с токеном
logout	Токен пользователя	Проверка токена пользователя, очистка куки, установка статуса «не в сети»

Разработанные методы позволяют в полном объеме охватить требования к авторизации и аутентификации в системе.

4.3.2 Разработка триггеров, представлений и хранимых процедур

Для контроля целостности данных и для обеспечения дополнительной бизнес-логики были разработаны следующие триггеры [16]:

- триггер при создании рекорда: автоматически обновляет кэшированный рейтинг пользователя;
- триггер при создании или удалении связи: автоматически обновляет кэшированный счетчик связей пользователя;
- триггер при реакции на сообщение: актуализирует счетчик реакций в сообщении;
- триггер при ответе на сообщение: актуализирует счетчик ответов на сообщение;
- триггер проверки времени создания, изменения, начала и конца: актуализирует время создания и изменения, проверяет чтобы время конца было больше времени начала;
- триггер проверки наличия свободного места в комнате: блокирует вход в комнату, если в ней нет свободного места, обновляет кэшированный счетчик свободного места в комнате;
- триггер при создании рекорда: проверяет достижения игрока.

Для упрощения доступа к основной информации о состоянии системы были созданы следующие представления:

- представление «Сообщения для пользователя»: это представление содержит информацию о сообщениях, исходящих и входящих для конкретного пользователя, включая отправителя, получателя, время создания, вложения и количество лайков;
- представление «Пользователи комнаты»: выводит актуальную информацию о комнате со списком игроков в ней;
- представление «Таблица лидеров»: выводит актуальную таблицу лидеров по счету с указанным количеством игроков.

Для упрощения работы с базой данных были созданы следующие хранимые процедуры:

- получение максимального счета конкретного игрока;
- получение максимального счета конкретной страны;
- получение количества игроков из конкретной страны.

Для ускорения работы базы данных были созданы необходимые индексы, в частности поиск по псевдониму, поиск по счету рекордов.

4.3.3 Разработка модуля взаимодействия с клиентом

Модуль взаимодействия с клиентом необходим для создания серверной части алгоритма восстановления согласованности клиента и сервера, также, как и модуль взаимодействия с сервером, представляет собой объект-обертку над игровой логикой с использованием обмена событиями по протоколу Socket.IO. Упрощенная схема алгоритма модуля представлена в приложении Г. Текст программы модуля представлен в приложении И. Список полей и методов обертки представлен в таблице 4.6.

Таблица 4.6 – Поля и методы модуля игровой логики

Название поля/ метода	Тип поля/ возвращаемый тип	Параметры	Назначение
game	Tetris	—	Игровое состояние, логика
socket	Экземпляр Socket.IO соединения	—	Поддержание сетевого взаимодействия с сервером
Объекты времени	Время в долях миллисекунд	—	Расчет точного времени того или иного события, определение порядка событий
inputQueue	Очередь событий от клиента	—	Накопление событий от клиента до следующего цикла обработки
stateBuffer	Циклический массив игровых состояний	—	История игровых состояний клиента
data	Объекты базы данных	—	Связь с базой данных через объекты системы «Игрок», «Комната»
start Competition()	—	Ключ генерации, обработчики	Начать соревнование
onSync()	—	Информация о пользователе	Инициализация игрового процесса, буферов, создание игрового состояния, синхронизация
onInput()	—	Игровое событие	Помещение события в очередь событий
onScoreUpdate()	—	Счет	Оповещение комнаты о изменении счета у участника
onGameOver()	—	Финальный счет	Обновление рекорда, синхронизация рекорда с клиентом

Продолжение таблицы 4.6

Название поля/ метода	Тип поля/ возвращаемый тип	Параметры	Назначение
process()	—	—	Обработка всех ожидающих в очереди событий

4.4 Разработка игровой логики

Алгоритм игровой логики выполняется отдельно от работы остального сервиса, так как его выполнение должно осуществляться на стороне сервера и на стороне клиента, что обусловлено требованием к наличию алгоритма восстановления согласованности клиента и сервера. Принцип разработки игровой логики описан в подразделе 3.3.1. Текст программы игровой логики представлен в приложении К. Модуль игровой логики состоит из трех классов: Random – необходим для детерминированной генерации случайных чисел, Figure – необходим для управления игровой фигурой, Tetris – необходим для контроля игрового состояния. Эти поля не влияют на проверку неравенства игровых состояний. Список главных полей классов и методов модуля представлен в таблице 4.7.

Таблица 4.7 – Поля и методы модуля игровой логики

Название поля/ метода	Тип поля/ возвращаемый тип	Параметры	Назначение
Класс Random			
seed	Целое	—	Ключ генерации
prev	Целое	—	Последнее состояние генератора
nextRandInt()	Целый	Диапазон генерации	Возвращает случайное число в указанном диапазоне
Класс Figure			
id	Перечислимый	—	Тип фигуры «Тетрис»
rotation	Перечислимый	—	Положение вращения
pos (x, y)	Целый	—	Позиция на поле
type	Перечислимый	—	Код типа
color	Перечислимый	—	Код цвета
value	Целый	—	Закодированное значение фигуры (для алгоритма обработки коллизий)

Продолжение таблицы 4.7

Название поля/ метода	Тип поля/ возвращаемый тип	Параметры	Назначение
Класс Figure			
genSequence()	—	Random	Создает новую последовательность из неповторяющихся типов фигур
constructor()	Figure	Тип фигуры, Random, или прототип фигуры	Создает новую фигуру
Класс Tetris			
currentFigure	Figure	—	Текущая фигура
nextFigures	Массив Figure	—	Очередь из фигур для появления
heldFigure	Figure	—	Фигура в буфере
Флаги состояния	Логический	—	Состояние игры (пауза, буфер, конец, и т.д.)
field	Массив типов блоков	—	Игровое поле
Счет	Целый	—	Текущий игровой счет
Скорость игры	Целый	—	Скорость обновлений в миллисекундах
random	Random	—	Генератор случайных чисел
Функции обратного вызова	Функция	—	Оповещение об обновлениях игрового состояния
constructor()	—	Прототип	Создает новое игровое состояние из прототипа или случайно
processEvent()	—	Код события	Обработать событие, обновив состояние
checkOnEnd()	Логический	—	Проверить игру на окончание
removeLine()	—	Номер линии	Сдвигает все блоки поля ниже на место удаленной линии

Продолжение таблицы 4.7

Название поля/ метода	Тип поля/ возвращаемый тип	Параметры	Назначение
Класс Tetris			
placeFigure()	—	Figure	Помещает фигуру на игровое поле
collideFigure()	Логический	Figure	Проверить, касается ли фигура блоков на игровом поле
checkOnLine()	—	—	Проверить наличие полных линий на поле, собрать их и начислить счет
process FieldEffects()	—	—	Обработать события игрового поля, алгоритм из приложения В
check BlockCollide()	Логический	Блок поля, тип блока фигуры	Проверить если два блока столкнулись
moveBlock()	Логический	Блок откуда, блок куда	Передвинуть блок, выполнив при этом все эффекты, вернуть признак если движение произошло
mergeBlock()	—	Блок откуда, блок куда	Произвести объединение характеристик блоков
nullifyBlock()	—	Блок	Уничтожить блок без эффектов
explodeBlock()	—	Блок, признак инициатора	Уничтожает блок, начисляя за него счет
killBlock()	—	Блок	Уничтожает блок, начисляя за него штраф

Реализация алгоритма восстановления согласованности требует проверки равенства игровых состояний. Для этого используются методы `deepCopy()` и `deepCompare()`. Первый создает полную копию игрового состояния, за исключением полей обратных функций и игровых специальных эффектов. Второй выполняется рекурсивно и проверяет равенство всех полей двух объектов.

5 ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

5.1 Общий принцип и инструменты тестирования

Тестирование было проведено путем комплексных проверок кода на различных наборах данных, включая наполнение базы данных. Все основные функции были протестированы.

Во время тестирования путем непрерывной игры и проверки отдельных функций игрового процесса были выявлены и исправлены недочеты. Вспомогательными средствами для отслеживания таких недочетов послужили программные средства Git, WebStorm и Selenium IDE, а также средства отладки браузеров Opera и Microsoft Edge. Для тестирования базы данных использовался инструмент MySQL Workbench.

5.2 Тестирование основного игрового процесса

Для тестирования игрового процесса, который может в себя включать бесконечное множество входных наборов данных (событий) и, как следствие, сценариев игры, необходимо было определиться с игровым сценарием максимально простым для воспроизведения и охватывающим все возможности игры, включая граничные. Таким образом, был определен основной тестовый сценарий [20], который необходим для проведения тестирования критического пути.

Основной набор действий здесь и далее – это такой набор действий, который представляет собой простейший полный сценарий игрового процесса. Он включает в себя:

- запуск игры;
- постановку на паузу;
- перезапуск игры;
- вращение всех фигур во все 4 позиции;
- помещение фигуры в буфер;
- досрочный и ускоренный спуск фигуры;
- сбор 1, 2, 3, 4 линий;
- завершение игры досрочно;
- завершение игры естественным путем (переполнение стакана).

Основной набор действий считается выполненным успешно, если:

- Выполнение действий произошло без существенных задержек;
- Количество согласований минимально (менее 0,5% от всех событий);
- Отсутствуют нарушения целостности неподвижной части поля;
- Соблюдена последовательность начисления счета и выставление максимального счета.

Основной набор действий соответствует требованиям к основному игровому процессу, требованию к сетевой и командной игре K11, а также требо-

ваниями к визуальному и музыкальному сопровождению и требованиями к управлению, У1-У3.

Тесты здесь проводились вручную и автоматически методом черного ящика [20]. Список тестов приведен в таблице 5.1. Критерий успешного выполнения тестирования основного алгоритма программы является выполнение всех тестов из таблицы 5.1.

Таблица 5.1 – Список тестов основного алгоритма программы

Имя теста и соответствующие требования	Выполняемые действия	Ожидаемый результат	Статус
Т1 Работа клиента без связи с сервером (требование И2)	Запуск клиента, запуск игры, выполнение основного набора действий	Изначальный счет 0, игровой процесс выполнен успешно, псевдоним игрока при этом @DEFAULT, статус @OFFLINE	Успех
Т2 Работа клиента в связке с сервером, без задержки (требование К11)	Запуск клиента и сервера на одном хосте (для минимизации задержек), выполнение основного набора действий без регистрации	Изначальный счет 0, игровой процесс выполнен успешно, псевдоним игрока при этом @DEFAULT, статус @ON-LINE. Отображение таблицы лидеров (без @DEFAULT)	Успех
Т3 Работа клиента в связке с сервером с регистрацией (требование А9)	Запуск клиента и сервера на одном хосте (для минимизации задержек), выполнение регистрации и авторизации, выполнение основного набора действий, выполнение выхода, выполнение повторной авторизации, выполнение основного набора действий	Изначальный счет 0 игровой процесс выполнен успешно, псевдоним игрока при этом такой же, что и при регистрации, статус @REGSTRD. При повторном входе псевдоним игрока при этом такой же, что и при регистрации, статус @REGSTRD, исходный максимальный счет равен максимальному счету, достигнутому при предыдущем тесте	Успех

Продолжение таблицы 5.1

Имя теста и соответствующие требования	Выполняемые действия	Ожидаемый результат	Статус
T4 Одновременная работа нескольких клиентов в связке с сервером с регистрацией и без	Запуск сервера и трех клиентов на одном хосте, один клиент в режиме «оффлайн», второй зарегистрирован, третий в режиме «он-лайн». Для каждого клиента алгоритм такой же, как в соответствующем предыдущем тесте	Выполнение всех требований предыдущих тестов, дополнительно при достижении рекорда зарегистрированным пользователем, обновление таблицы лидеров отображается у всех подключенных пользователей	Успех

5.3 Тестирование базы данных

Отдельно от тестирования основных модулей сервиса, которое включает в себя взаимодействие с базой данных, было проведено тестирование базы данных. Суть тестирования заключалась в наполнении каждой таблицы базы данных некоторыми данными, в каждой таблице было создано 20 записей, за исключением таблиц статусов и ролей, их наполнение ограничено требованиями предметной области.

Так как проверить все возможные наборы данных не представляется возможным, данные выбирались по принципу 50% положительных данных (те, вставка которых должна порождать положительный результат) и 50 % отрицательных данных. Таким образом, покрываются 2 вида ошибок. Список тестов приведен в таблице 5.3. Для выполнения каждого теста база данных очищалась, оставляя только записи в следующих таблицах. Таблица ролей содержит записи, представленные в таблице 5.2. Таблица регионов и статусов содержит в себе только одну запись. Таблица пользователей содержит одну запись пользователя для проверки на защиту от дублирования записей. При создании пользователя, в соответствии ему создавалась запись рекорда, данные которой генерировались случайно.

Таблица 5.2 – Записи в таблице ролей

ID роли	Имя роли	Псевдоним роли
1	member	—
20	root	@SUSROOT
50	shared	@DEFAULT
100	server	@SERVERS

Таблица 5.3 – Список тестов базы данных

Имя теста и соответствующие требования	Выполняемые действия	Ожидаемый результат	Статус
T5 Регистрация пользователя	Ввод допустимых имени, фамилии, уникального псевдонима, пароля	Вставка элемента в таблицу пользователей базы данных. Роль и статус выставлен по умолчанию (1 и 1 соответственно)	Успех
T6 Регистрация существующего пользователя	Ввод существующего псевдонима	Вставка элемента отклонена	Успех
T7 Публикация сообщения с вложениями	Ввод допустимых по длине сообщения и идентификаторов пользователей	Вставка элемента в таблицу сообщений базы данных. Вставка элементов вложений в таблицу вложений	Успех
T8 Публикация сообщения с 10 и более вложениями	Ввод допустимых по длине сообщения и идентификаторов пользователей, а также имена 10 (и более) файлов	Вставка всех элементов отклонена	Успех
T9 Создание рекорда с меньшим счетом	Ввод допустимых данных для рекорда с меньшим счетом	Вставка элемента отклонена	Успех
T10 Выполнение хранимой процедуры – создание таблицы рекордов на 15 пользователей	В таблице рекордов более 15 рекордов, принадлежащих 10 пользователям, каждый рекорд имеет уникальный счет.	Таблица, имеющая 2 поля и 10 записей – псевдоним пользователя и максимальный рекорд для этого пользователя (из рекордов, относящихся к этому пользователю), отсортированная по счету в обратном порядке	Успех

5.4 Тестирование игрового аккаунта и мессенджера

Модуль мессенджера и игрового аккаунта неразрывно связан с модулями авторизации и регистрации, с серверным программным интерфейсом, ба-

зой данных и с пользовательским интерфейсом. Основная задача мессенджера – обеспечить доставку и просмотр сообщений для всех зарегистрированных пользователей в сети. Список тестов модуля приведен в таблице 5.4. Для выполнения тестов в ней считается, что в системе авторизованно два пользователя. У обоих пользователей открыта вкладка чат и выбран диалог с другом другом. Пользователь А – отправитель, Б – получатель.

Таблица 5.4 – Список тестов модуля мессенджера и игрового аккаунта

Имя теста и соответствующие требования	Выполняемые действия	Ожидаемый результат	Статус
T11 Отправка сообщения (требования Ч1, Ч2)	Набрать сообщение в поле ввода. Нажать кнопку «Send message»	Сообщение появляется у обоих пользователей. У А оно считается исходящим и отображается слева, а у Б входящим, отображается справа	Успех
T12 Редактирование сообщения (требование Ч5)	Пользователь А: нажать на кнопку с символом карандаша возле сообщения, после повторяет это же действие	У А при первом нажатии карандаш выделяется, а текст сообщения появляется в поле для ввода, текст кнопки меняется на «Edit message». Повторное нажатие приводит поле для ввода и кнопку карандаша в исходное состояние	Успех
T13 Редактирование сообщения (требование Ч5)	Пользователь А: нажать на кнопку с символом карандаша возле сообщения, после изменяет текст и нажимает на «Edit message»	Текст сообщения меняется у обоих пользователей. Снизу сообщения появляется плашка с временем изменения сообщения. Поле для ввода и кнопка карандаша у А приведено в исходное состояние	Успех
T14 Удаление сообщения (требование Ч6)	Пользователь А: нажать на кнопку с символом мусорной корзины возле сообщения	Сообщение пропадает у обоих пользователей	Успех

Продолжение таблицы 5.4

Имя теста и соответствующие требования	Выполняемые действия	Ожидаемый результат	Статус
T15 Отправка сообщения с вложениями (требование Ч4)	Набрать сообщение в поле ввода. Выбрать до девяти изображений. Нажать кнопку «Send message»	Сообщение появляется у обоих пользователей, при этом отображены все вложения, при нечетном количестве верхнее занимает всю ширину, при четном вложения делятся на сетку	Успех
T16 Отправка сообщения с вложениями (требование Ч4)	Пользователь А: набрать сообщение в поле ввода. Выбрать файлы. Нажать кнопку «Send message»	Сообщение не отправляется, текст сообщения остается в поле для ввода	Успех
T17 Реакция на сообщение (требование Ч8)	Пользователь А: выбрать входящее сообщение и нажать кнопку с символом сердца	При наведении мыши кнопка реакции приобретает первичный цвет у А. При нажатии кнопка приобретает первичный цвет у А, и появляется у Б. на сообщении. Повторное нажатие снимает цвет	Успех
T18 Реакция на сообщение (требование Ч8)	Выбрать исходящее сообщение и нажать кнопку с символом сердца	Нажатие ни на что не влияет у обоих пользователей	Успех
T19 Автоматическая прокрутка ленты вниз (требование У4)	Пользователь А: выбрать пользователя Б из списка. Не совершать прокрутки. Совершить прокрутку	У пользователя А лента автоматически спускается по мере загрузки или получения сообщений. После совершения прокрутки автоматический спуск прекращается	Успех
T20 Просмотр игрового аккаунта (требования А1, А3)	Открыть игровой аккаунт пользователя	На дисплее отображается: псевдоним, имя, фамилия, почта, статус, рейтинг, регион, а также список рекордов	Успех

Продолжение таблицы 5.4

Имя теста и соответствующие требования	Выполняемые действия	Ожидаемый результат	Статус
T21 Просмотр таблицы лидеров (требование A4)	Открыть игровой аккаунт пользователя. Перейти во вкладку таблица лидеров	Таблица лидеров содержит записи игроков со следующей информацией: место, псевдоним, регион и счет. Таблицу можно отсортировать	Успех
T22 Авторизация (требование A1)	Открыть форму для авторизации. Ввести корректные данные для А	Снизу отображаются рабочие ссылки на регистрацию и переход к игре без авторизации. Происходит перенаправление в игровой аккаунт	Успех
T23 Авторизация (требование A1)	Открыть форму для авторизации. Ввести некорректные данные для А	Снизу формы появляется панель с предупреждением «user do not exists»	Успех
T24 Регистрация (требования A1, A5)	Открыть форму для авторизации. Ввести корректные данные для нового пользователя	Снизу отображаются рабочие ссылки на авторизацию и переход к игре без авторизации. Происходит перенаправление на авторизацию	Успех
T25 Регистрация (требования A1, A5)	Открыть форму для авторизации. Ввести некорректные данные для нового пользователя	При несоблюдении требований к данным отображается панель с предупреждением «bad input!»	Успех

5.5 Тестирование функционала игровых комнат

Функционал игровых комнат является связующей компонентой между игровыми аккаунтами, мессенджером и игровым процессом, и необходим для обеспечения соревновательной одиночной и командной игры, согласно требованиям группы К, указанным в таблице 2.2, а также требованиям A9, Ч2. Список требований к функционалу представлен в таблице 5.5.

Таблица 5.5 – Список тестов функционала игровых комнат

Имя теста и соответствующие требования	Выполняемые действия	Ожидаемый результат	Статус
T26 Выбор комнаты для быстрой схватки	На главной панели игры нажать кнопку «Quickfight»	Пользователь будет перемещен в публичную и максимально заполненную комнату. Если таковой не оказалось, будет создана пустая комната со случайным именем	Успех
T27 Выбор и смена команды	В комнате нажать кнопку «Join» напротив всех команд по очереди. Одна из команд должна быть заполнена	Псевдоним пользователя изменяет цвет в соответствии с цветом команды, счетчик заполненности обновляется, увеличивается в целевой команде, уменьшается в оставленной	Успех
T28 Отправка сообщения во внутренний чат	Ввести сообщение в поле для ввода. Нажать Enter или кнопку со стрелкой	Сообщение появится последнем в списке сообщений у всех пользователей	Успех
T29 Присоединение другого пользователя к комнате	С другого аккаунта подключиться к комнате	В чате комнаты появится сообщение о присоединении игрока, с его псевдонимом	Успех
T30 Отключение игрока	С другого аккаунта отключиться из комнаты, перезагрузив страницу, или нажав «Leave Room»	В чате комнаты появится сообщение об отключении игрока, с его псевдонимом	Успех
T31 Заполнение комнаты	Создать необходимое количество пользователей и условия для начала игры	В чате комнаты появится сообщение с временем ожидания до начала игры. По его истечении при неизменности составов команд пользователей перенаправит на вкладку с игрой	Успех

Продолжение таблицы 5.5

Имя теста и соответствующие требования	Выполняемые действия	Ожидаемый результат	Статус
T32 Выбор комнаты из списка комнат	Во вкладке список комнат авторизованным пользователем нажать кнопку «Join»	Если есть свободное место в комнате, счетчик в комнате увеличится, кнопка «Join» станет кнопкой «Leave». Попытка подключения к другой комнате не имеет эффекта. Нажатие «Leave» уменьшает счетчик в комнате и возвращает кнопку	Успех
T33 Функционал списка комнат	Во вкладке список комнат нажать поля доступные для сортировки	Поля, доступные к просмотру: номер, имя комнаты, псевдоним владельца, количество команд, распределение участников, а также кнопки присоединения и управления	Успех
T34 Удаление комнаты	Пользователь является владельцем комнаты. Нажать кнопку «maintain»	Комната отключается и пропадает из списка. Всех участников перенаправляет на главную страницу	Успех
T35 Добавление комнаты	Ввод всех корректных данных	При успешном создании появится сообщение, а после произойдет перенаправление в меню комнат. В случае ошибки появится соответствующее сообщение	Успех

5.6 Автоматическое тестирование

Автоматическое тестирование было произведено как для отдельных компонент интерфейса, так и для комплексных интерфейсных модулей, для создания тестов использовался инструмент для автоматизированного тестирования Selenium IDE. Список и результаты тестов представлены на

рисунке 5.1. Если определенный компонент не появился на экране в течение 5000 миллисекунд, тест считался проваленным.

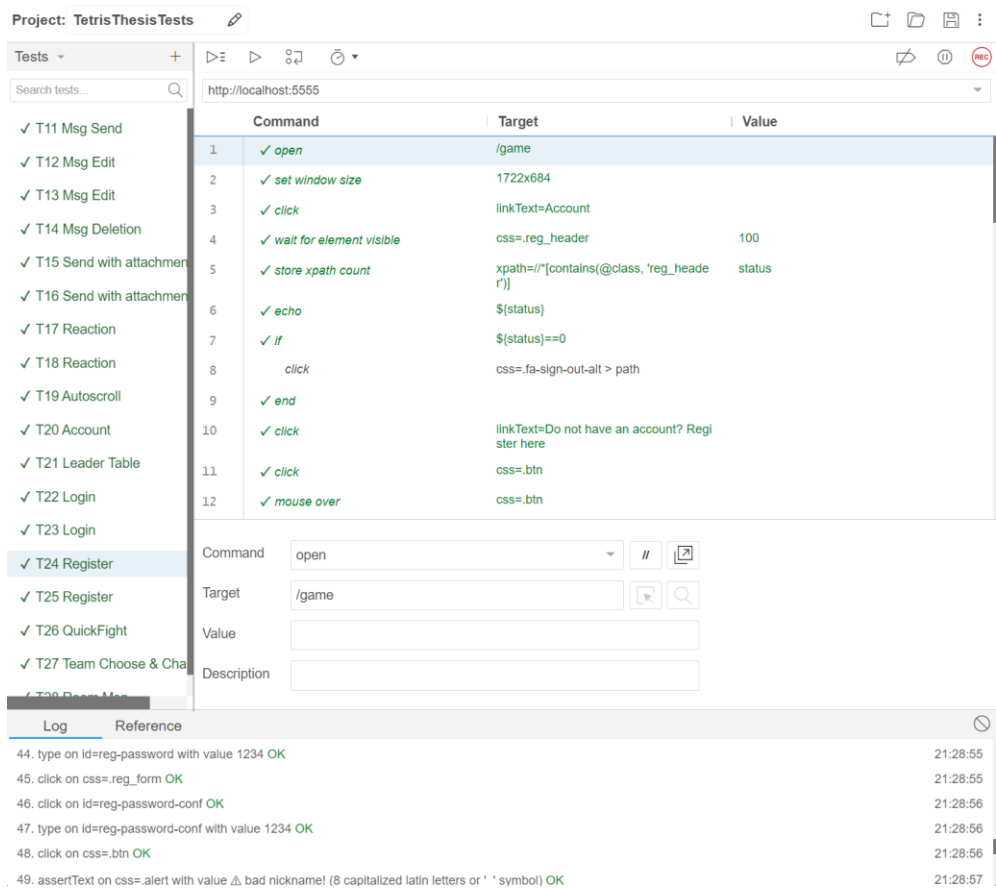


Рисунок 5.1 – Тесты в среде Selenium IDE

Отдельно проверялся основной игровой процесс, где действия игрока и последовательности нажатия клавиш были записаны заранее, а метриками успеха являлись количество синхронизаций и средняя задержка реакции игры. При достижении количества синхронизаций менее 1% от общего количества событий и средней задержки не более 20 миллисекунд тесты считались пройденными.

5.7 Вывод из тестирования сервиса

Для проверки готовности приложения к эксплуатации использовались следующие метрики: соотношение успешно пройденных тестов к их общему количеству, и соотношение количества требований, проверяемые тестами, к общему количеству требований [20]. Приложение считалось готовым к использованию при достижении значения метрик 0,8 и выше. По итогам тестирования приложение признается готовым продуктом, пригодным к эксплуатации согласно заданным функциональным требованиям.

6 РУКОВОДСТВО ПО УСТАНОВКЕ И ЭКСПЛУАТАЦИИ СЕРВИСА

6.1 Руководство по установке сервиса

Сервис представляет собой состоит из трех модулей: клиентской части, серверной части и базы данных. Для удобства использования и развертывания сервиса в облаке, все эти модули были упакованы в контейнер Docker. Это позволяет запускать сервис на любой машине, где установлен Docker и настроены соответствующие порты. Ниже представлен список шагов, которые необходимо выполнить для подготовки к эксплуатации сервиса:

- установите Docker на вашу рабочую машину и настройте его в соответствии с документацией;
- извлеките актуальную версию сервиса, выполнив команду `git clone https://github.com/archql/NodeJS-React-Tetris-Service` с помощью системы контроля версий Git;
- при необходимости, поменяйте порт, на котором будет доступен сервис в файле `docker-compose.yml`, по умолчанию установлен порт 80;
- откройте терминал или командную строку, перейдите в папку с проектом и выполните команду `docker-compose create`;
- переместите контейнеры на рабочую машину;
- запустите контейнеры на рабочей машине с помощью Docker;
- настройте брандмауэр, разрешив входящие TCP соединения на порт 80;
- при необходимости, настройте трансляцию NAT.

По окончании выполнения руководства, сервис должен быть доступен в браузере по заданному IP адресу и порту. Работоспособность сервиса отслеживается через встроенную консоль Docker.

6.2 Руководство по эксплуатации сервиса для администратора

Сервис полностью автономен и не требует вмешательства извне. Если потребуется внести изменения в код или конфигурацию сервиса, необходимо остановить контейнеры с помощью команды `docker-compose down`, внести необходимые изменения и затем повторно запустить контейнеры командой `docker-compose up`.

6.3 Руководство по эксплуатации сервиса для игрока

Одна из основных задач разработки заключалась в том, чтобы сервис был прост и понятен в освоении, и не требовал дополнительных ресурсов. Данное руководство было подготовлено для ознакомления с сервисом без непосредственного взаимодействия с ним. Любой пользователь изначально оказывается на главной странице сервиса, которая представлена на рисунке

6.1. Панель навигации сверху содержит краткую информацию о игроке: его регион, рейтинг и псевдоним, или же псевдоним по умолчанию если игрок не авторизован. Также она содержит основные ссылки: переход в личный кабинет, переход в мессенджер, переход на главную страницу игры, переход в менеджер комнат, в таблицу лидеров, а также на страницу с информацией о игре.

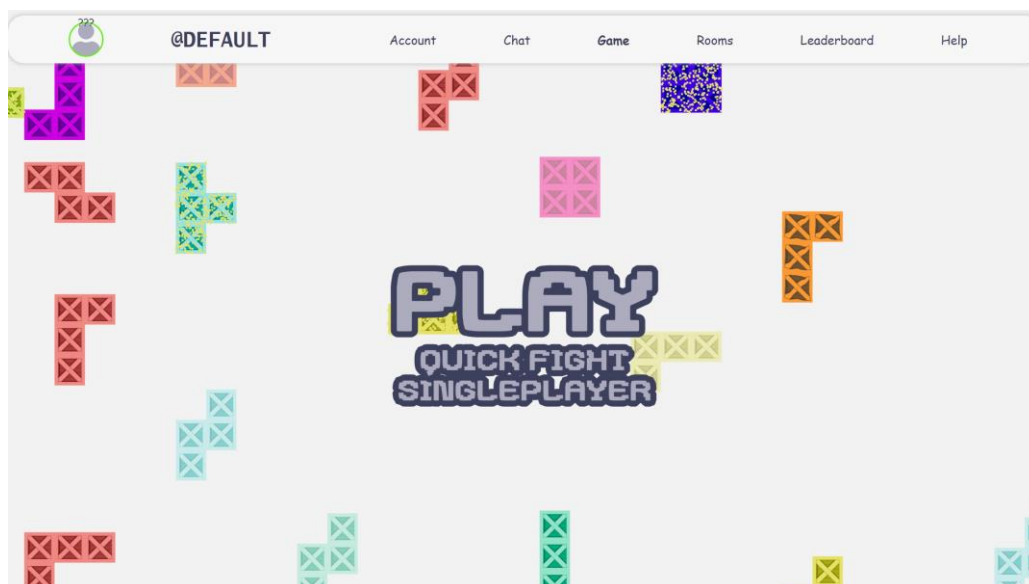


Рисунок 6.1 – Главная страница сервиса

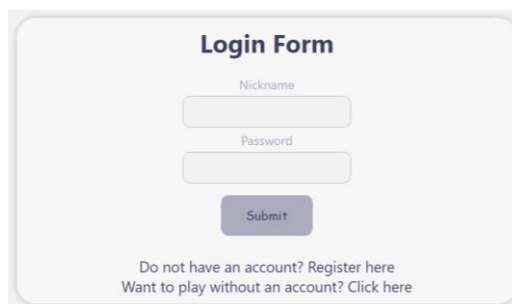
Снизу расположена декоративная игровая заставка с падающими фигурами «Тетрис», а также игровое меню с возможностью перехода к одиночной игре «Singleplayer», и к быстрому поиска соперника – «Quick Fight».

6.3.1 Регистрация и авторизация

Для регистрации на сервисе необходимо перейти по ссылке «Account» с главной страницы. На экране появится форма авторизации, представлена на рисунке 6.2. Для авторизации достаточно ввести свой псевдоним и пароль, а после нажать кнопку «Submit». После этого появится плашка ожидания ответа от сервера. Если данные корректны, браузер перенаправит пользователя на страницу личного кабинета. Если же нет, снизу появится панель с сообщением об ошибке.

Чтобы зарегистрироваться, необходимо нажать на ссылку «Do not have an account? Register here». После перехода появится форма для регистрации, представлена на рисунке 6.3.

Для регистрации достаточно ввести свое имя, фамилию (любые не пустые), псевдоним, почту, регион и пароль, а после нажать кнопку «Submit». Если данные корректны, браузер перенаправит пользователя на страницу авторизации. Если же нет, снизу появится панель с сообщением об ошибке.



Login Form

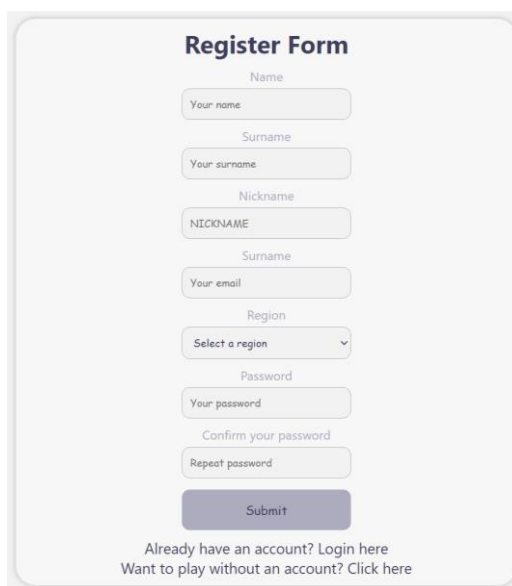
Nickname

Password

Submit

Do not have an account? Register here
Want to play without an account? Click here

Рисунок 6.2 – Форма авторизации



Register Form

Name

Your name

Surname

Your surname

Nickname

NICKNAME

Surname

Your email

Region

Select a region

Password

Your password

Confirm your password

Repeat password

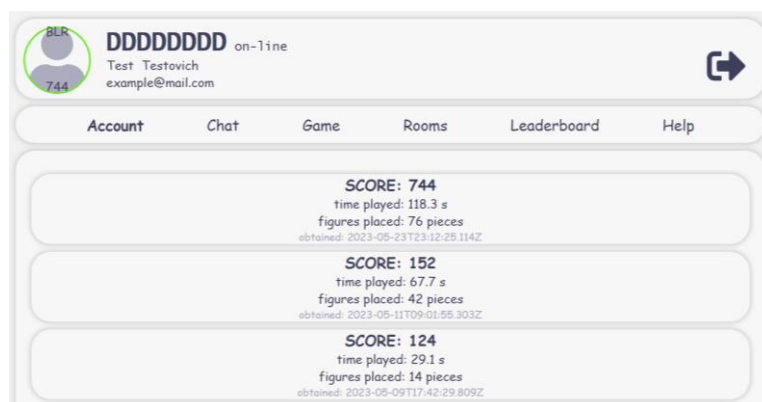
Submit

Already have an account? Login here
Want to play without an account? Click here

Рисунок 6.3 – Форма регистрации

6.3.2 Личный кабинет

Личный кабинет игрока предназначен для просмотра личной информации, а также игровых рекордов, достижений и навыков, представлен на рисунке 6.4.



BLK 744

DDDDDDDD on-line

Test Testovich
example@mail.com

Account Chat Game Rooms Leaderboard Help

SCORE: 744
time played: 118.3 s
figures placed: 76 pieces
obtained: 2023-06-23T23:12:26.114Z

SCORE: 152
time played: 67.7 s
figures placed: 42 pieces
obtained: 2023-06-11T09:01:56.303Z

SCORE: 124
time played: 29.1 s
figures placed: 14 pieces
obtained: 2023-06-09T17:42:29.809Z

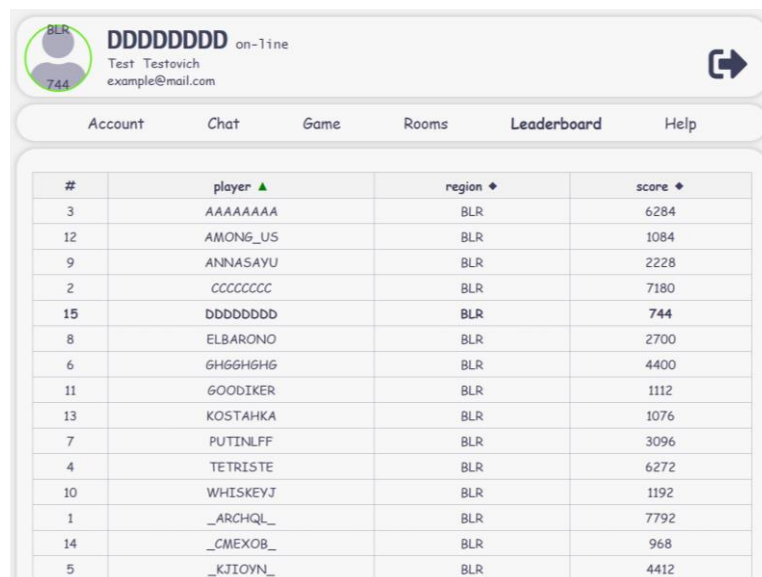
Рисунок 6.4 – Личный кабинет игрока

На верхней панели находятся псевдоним, имя, фамилия, почта игрока, его статус (в сети или нет), рейтинг и регион. Справа расположена кнопка выхода из аккаунта.

Панель ниже предназначена для навигации, а остальное пространство занято таблицей рекордов игрока, в которой располагается достигнутый счет, время игровой сессии, количество использованных фигур, а также дата игры.

6.3.3 Таблица лидеров

Таблица лидеров располагается в пункте навигации «Leaderboard». Представляет собой таблицу из 16 записей лучших игроков, куда независимо от места входит запись текущего игрока. Таблицу лидеров можно отсортировать по полям псевдоним, регион, счет игрока. На рисунке 6.5 представлена таблица лидеров, отсортированная по полю псевдоним в возрастающем порядке. Запись текущего игрока «DDDDDDDD» выделена в таблице.



#	player ▲	region ◆	score ◆
3	AAAAAAA	BLR	6284
12	AMONG_US	BLR	1084
9	ANNASAYU	BLR	2228
2	CCCCCCC	BLR	7180
15	DDDDDDDD	BLR	744
8	ELBARONO	BLR	2700
6	GHGGHGHG	BLR	4400
11	GOODIKER	BLR	1112
13	KOSTAHKA	BLR	1076
7	PUTINLFF	BLR	3096
4	TETRISTE	BLR	6272
10	WHISKEYJ	BLR	1192
1	_ARCHQL_	BLR	7792
14	_CMEXOB_	BLR	968
5	_KJIOVN_	BLR	4412

Рисунок 6.5 – Таблица лидеров отсортированная по псевдониму

Сортируемые поля таблицы отмечены символом ромб. Для сортировки таблицы необходимо нажать на соответствующее поле. Символ зеленый треугольник вверх обозначает сортировку по возрастанию, красный треугольник вниз – сортировку по убыванию. Для переключения сортировок, и последующего отключения, необходимо совершать повторные нажатия по соответствующему полю до достижения нужного эффекта.

6.3.4 Таблица выбора игровой комнаты

Таблица выбора игровой комнаты располагается в пункте навигации «Rooms». Сверху располагается ссылка для создания своей комнаты. Переход по ней перемещает игрока на форму создания комнаты, представлена на рисунке 6.6. Для создания комнаты необходимо ввести имя комнаты, количество

команд (2..4), количество игроков в одной команде (1..4), а также пароль, если требуется создать приватную комнату. Если данные корректны, появится сообщение о успешном создании комнаты, игрок автоматически занимает место в комнате, а после браузер перенаправит пользователя на страницу выбора игровой комнаты. Если же нет, снизу появится панель с сообщением об ошибке. Таблица выбора игровой комнаты, отсортированная по владению, представлена на рисунке 6.7, и содержит следующие поля: идентификатор; имя; псевдоним владельца; количество команд и игроков в каждой из них; количество участников в специальной нотации: изначально идет количество участников без команды, а далее через разделитель количество участников в каждой из команд; признак публичности комнаты; а также кнопки подключения «join» или отключения «leave», и кнопка управления комнатой «maintain», если игрок является ее владельцем.

Рисунок 6.6 – Заполненная форма создания комнаты

#	name	owner	teams	members	private?	
4	My super room	DDDDDDDD	2 of 3	1 / 0 / 0	yes	leave maintain
2	Duel-5185eec7b3a2	@SYSROOT	2 of 1	1 / 0 / 0	no	join
3	XXX	_ARCHQL_	2 of 1	0 / 0 / 0	no	join

Рисунок 6.7 – Таблица игровых комнат, отсортированная по владению

Доступны следующие специальные сортировки: сортировка по полю команд осуществляется по количеству команд; сортировка по полю участников – по количеству свободных мест; сортировка по признаку занимаемого места; сортировка по признаку владения.

Для занятия места в комнате необходимо нажать кнопку «join». Если игрок уже занимает место в какой-либо комнате, или комната уже заполнена, эффекта не последует, в противном случае игрок появится в счетчике участников, а кнопка «join» станет кнопкой «leave», которая необходима для выхода из комнаты. Кнопка «maintain» удаляет комнату.

6.3.5 Мессенджер

Мессенджер располагается в пункте навигации «Chat», и представляет собой простой интерфейс для обмена сообщениями между пользователями системы, изображен на рисунке 6.8.

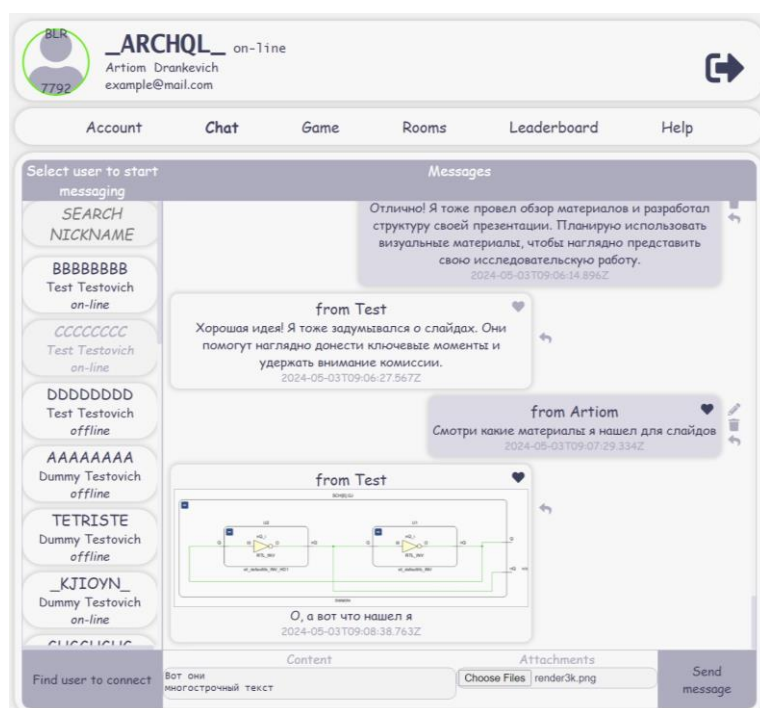


Рисунок 6.8 – Диалог между пользователями

На панели слева располагается список пользователей, с которыми доступно взаимодействие, а также поле для поиска пользователя по псевдониму. Для выбора диалога с пользователем необходимо нажать на соответствующий псевдоним на панели. Повторное нажатие возвращает чат в исходное состояние. На панели справа располагаются сообщения диалога с выбранным пользователем, по умолчанию панель пуста. Снизу располагаются средства управления – кнопка поиска пользователя «Find user to connect», многострочная панель ввода сообщения «Content», панель выбора вложений «Attachments», кнопка отправки сообщения «Send Message». Для того чтобы прикрепить вложения к сообщению, необходимо нажать «Choose Files» на панели «Attachments» и выбрать желаемые файлы. Вложения автоматически компонируются в зависимости от их количества, пример для пяти изображений представлен на рисунке 6.9.

Для того, чтобы оставить реакцию на сообщение, необходимо нажать на символ сердца рядом с интересующим сообщением отправителя. Повторное нажатие снимает реакцию. Для редактирования сообщения необходимо нажать на символ карандаша рядом с сообщением. При этом символ выделяется, текущий текст сообщения помещается в поле «Content», а кнопка «Send Message» заменяется на кнопку «Edit Message». Нажатие на нее меняет текст сообщения у обоих пользователей, а также добавляет плашку «modified» к сообщению, для различимости исходных сообщений от измененных. Редактировать вложения сообщения при этом нельзя. Нажатие на кнопку с символом мусорной корзины удаляет сообщение из диалога у обоих пользователей.

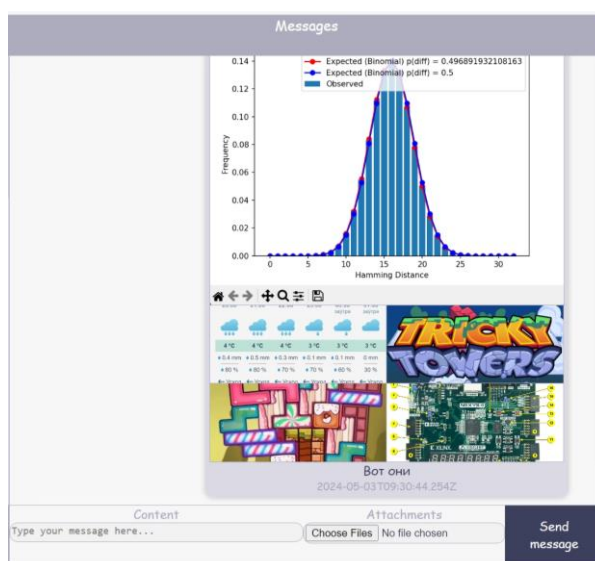


Рисунок 6.9 – Сообщение с пятью вложениями

6.3.6 Игровая комната

Игровая комната недоступна из панели навигации напрямую, для перехода к ней необходимо присоединиться к комнате через таблицу выбора игровой комнаты или выбором пункта «Quick Fight» на главной странице сервиса. Меню игровой комнаты представлено на рисунке 6.10.

На изображении в меню находятся два игрока. Псевдоним игрока, состоящего в команде, отображается в соответствии с цветом команды, если игрок не состоит в команде, его цвет стандартный. На панели слева отображается список доступных команд, их заполнение, а также кнопка присоединения к команде «Join», при нажатии на которую проверяются условия присоединения, и если они соблюдены – игрок добавляется в команду, меняет цвет, и счетчик мест в команде. Ниже располагается кнопка выхода из комнаты «Leave Room», при нажатии на которую игрок покидает команду и комнату.

Слева находится панель внутренних сообщений комнаты, в которой игроки могут просматривать сообщения от других участников комнаты, а также

информационные системные сообщения. Снизу находится многострочное поле для ввода сообщения, а также кнопка со стрелкой для отправки. Отправку сообщения можно также совершить и по нажатию клавиши «Enter».

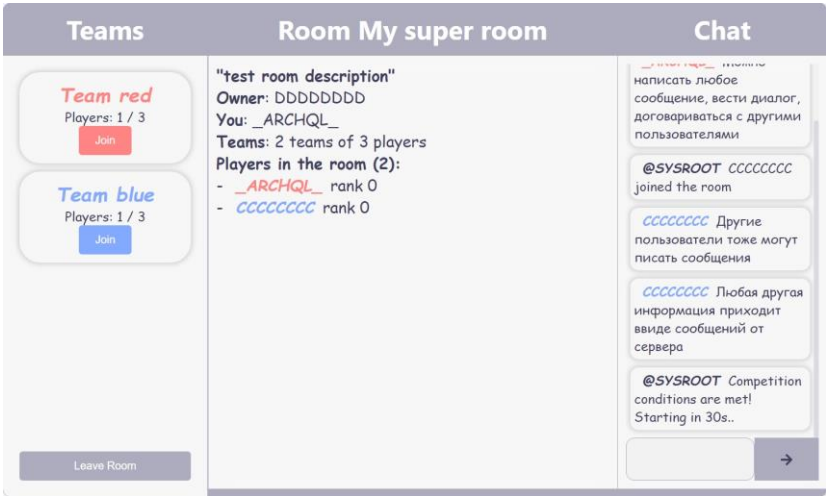


Рисунок 6.10 – Меню игровой комнаты

В центре расположена панель с информацией о текущем состоянии комнаты. В ней указаны: описание комнаты, псевдоним владельца комнаты, псевдоним текущего игрока, количество команд и максимальное количество участников в них, текущее количество игроков в комнате и их список.

Информация о начале соревнования сообщается во внутренний чат от имени сервера. После соблюдения всех условий начала соревнования и истечения таймера всех участников автоматически перенаправит в игровой процесс.

6.3.7 Игровой процесс

Игровой процесс осуществляется на игровом поле, которое представлено на рисунке 6.11. Управление игровым процессом осуществляется с клавиатуры. Список клавиш и их действий представлен в таблице 6.1.

Таблица 6.1 – Список клавиш и их действий

Клавиша	Действие	Примечание
Стрелка вправо	Движение активной фигуры вправо	—
Стрелка влево	Движение активной фигуры влево	—
Стрелка вверх	Вращение активной фигуры по часовой стрелке	—
Стрелка вниз	Вращение активной фигуры против часовой стрелки	—

Продолжение таблицы 6.1

Клавиша	Действие	Примечание
Пробел	Быстрый спуск фигуры вниз	За каждый пройденный блок начисляется счет
Клавиша SHIFT	Ускорение спуска фигуры в два раза	За каждый пройденный блок начисляется счет
Клавиша Н	После использования фигура из буфера становится текущей (если был пуст – сгенерируется новая), а текущая фигура поместится в буфер. Текущая фигура поместится на исходную позицию.	После использования буфер блокируется до генерации следующей фигуры
Клавиша Р	Постановка или снятие игры с паузы	Когда на паузе – управление фигурой заблокировано. Недоступно в соревновании
Клавиша R	Перезапуск игры	Прогресс будет утерян, за исключением счета, если тот превышал максимальный. Недоступно в соревновании
Клавиша ESC	Досрочное завершения игры. При первом нажатии останавливает игру без возможности возобновления. При втором нажатии происходит перенаправление на главную страницу	В соревновании заканчивает его досрочно и засчитывается за поражение независимо от результата команды
Другие клавиши	Использование специальной игровой способности	Их назначение может меняться в дальнейших обновлениях

Игровой процесс содержит четыре основных поля. Слева расположена информация о игровой комнате: название и список игроков по командам с указанием текущего счета каждого участника. Сверху расположена панель с таймером до окончания игры, а также панель прогресса каждой из команд, которая заполняется по мере приобретения участниками команды игрового счета. Посередине расположена информация о текущем статусе или событии игры, а также сам игровой стакан, в котором и происходит все взаимодействие. Справа расположена информация о текущем состоянии игры: псевдоним игрока, его счет, следующая фигура и фигура в буфере (может быть пустой). В игре существует пять типов блоков, представлены на рисунке 6.12.

Цельный блок, на рисунке 6.12 крайний слева, представляет собой обычный блок игры «Тетрис», и не имеет специальных механик. На поле представлен постоянным цветом. Блок типа «разрушитель», на рисунке 6.12 второй

справа после того, как занимает свое место на поле, уничтожает все блоки вокруг, начисляя счет за их уничтожение и отображая специальный эффект. На поле представлен коробкой, которая плавно меняет цвет с белого на красный и наоборот.

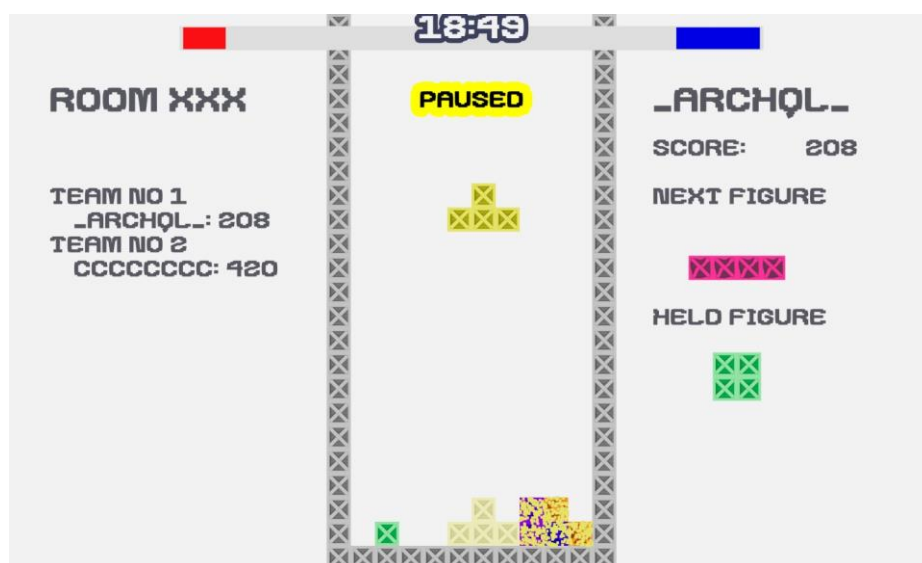


Рисунок 6.11 – Игровой процесс



Рисунок 6.12 – Все типы игровых блоков

Блок типа «призрак», на рисунке 6.12 по центру, проходит сквозь другие блоки на поле, может быть расположен в любом месте поля сочетанием клавиш Shift+Enter. После расстановки становится обычным цельным блоком, смешивая цвет и увеличивая стоимость замененных блоков на свою стоимость. Блок типа «жидкость», на рисунке 6.12 второй справа, после постановки распространяется в любые пустоты рядом с собой, выбирая доступные пути по приоритету вниз, вниз и влево, вниз и вправо. Блок типа «песок», на рисунке 6.12 справа, после постановки распространяется в любые пустоты под собой, меняясь местами с блоком «жидкость» при возможности. Обработка эффектов постобработки происходит во время обновления игрового таймера.

Игровой процесс сопровождается различными эффектами, а именно сбор линий, взрыв и объединение блоков. Все разновидности игровых эффектов представлены на рисунке 6.13. Во время сбора линий на месте собранного блока отображается анимация с числом, цвет которого идентичен цвету блока, а номинал равен его цене. При взрыве блока типа «разрушитель»

число всегда красного цвета, и равно счету, собранному при уничтожении соседних блоков, плюс стоимость самой «разрушителя».

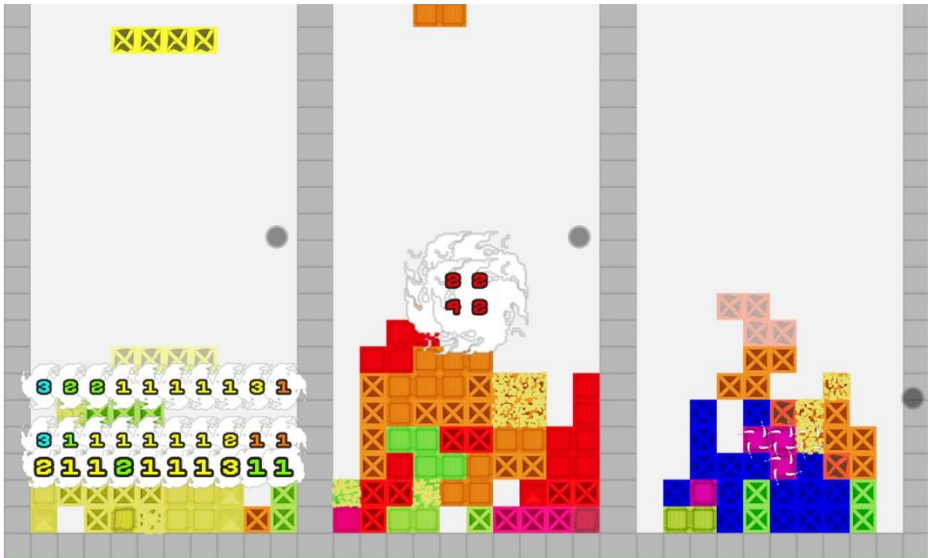


Рисунок 6.13 – Разновидности игровых эффектов

Соревнование считается законченным тогда, когда одна из команд достигает условия победы – заполнения панели прогресса. В этом случае игра для всех участников прерывается, отображая панель конца соревнования, которая представлена на рисунке 6.14.



Рисунок 6.14 – Панель окончания соревнования

На панели указана информация о персональной информации и статусе игрока, а также информация обо всех участниках комнаты, их вкладе, а также полученный бонус или штраф к рейтингу.

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ МНОГОПОЛЬЗОВАТЕЛЬСКОЙ ИГРЫ «ТЕТРИС»

7.1 Характеристика сервиса, разрабатываемого для реализации на рынке

Целью разработки является предоставление игрокам удобной платформы для общения, развлечения, укрепления командного духа, совместного времяпрепровождения, обмена опытом и соревнований, привлечение внимания к другим проектам, получении прибыли.

Сервис разрабатывается индивидуальным предпринимателем, в разработке приложения участвуют 3 человека, сроки разработки составляют 4 месяца. Потенциальные клиенты – любители классических аркадных игр и игр головоломок, сервис прост в освоении и использовании, и подходит для людей всех возрастных групп. Особое внимание будет уделено адаптивности и доступности приложения на различных устройствах, включая компьютеры и мобильные телефоны Android. Это позволит игрокам получать доступ к форуму в любое время и из любого места, создавая активное и динамичное сообщество игроков. Сервис будет обеспечивать возможность игры как со случайными игроками, так и с друзьями и коллегами, при отсутствии необходимости обязательной регистрации для максимального упрощения игроков в процесс.

В Сети существует несколько проектов, которые рассматриваются как примеры или аналоги хорошей реализации подхода к переработке механики игр.

Сервис Chess.com - платформа для игры в шахматы онлайн с более чем 100 миллионами пользователей со всего мира [11]. Ключевые особенности: бесплатная игра в шахматы, активное сообщество, большое количество тактических головоломок, мощный компьютерный противник. Не представляет прямой конкуренции, но является успешным примером реализации цели для другой игры.

Игра Tricky Towers. Предоставляет новый взгляд на игру «Тетрис», многопользовательские режимы, соревновательный аспект, многоплатформенность [12]. Не представляет прямой конкуренции, так как механика предоставления доступа различна, но имеет схожую аудиторию. Минусом проекта может стать обязательный игровой взнос для доступа к проекту, отсутствия возможности доступа из Сети.

Разрабатываемый сервис не стремится заместить своих конкурентов, а представить новый подход к времяпрепровождению, а также свежий взгляд и адаптацию для миллионов поклонников игры «Тетрис» по всему свету. Преимуществами сервиса является наличие встроенной системы взаимодействия и обмена сообщениями между игроками, гибкий API, позволяющий сторонним разработчикам взаимодействовать с игрой, доступ из Сети, не требующий специальных клиентов или инфраструктуры, свежий взгляд на игровой про-

цесс, который позволяет усилить соревновательную составляющую и повысить интерес и вовлечение в игровой процесс.

Предполагаемая модель монетизации сервиса смешанная и состоит из комбинации моделей реклама внутри приложения (In-app Advertising), бесплатный доступ к основному функционалу и платный к расширенному (Freemium), покупки внутри приложения (In-app Purchases) [21]. Список услуг, которые планируется монетизировать: продолжение игровой сессии, получения дополнительного рейтинга, увеличение внутриигрового уровня, возможность организовывать игровые сессии с пользовательскими настройками и правилами.

Доступ к приложению будет обеспечен из браузера, а также размещен на специализированных сайтах-агрегаторах.

7.2 Расчет инвестиций в разработку сервиса для его реализации на рынке

Инвестиции для разработчика сервиса являются затраты на его разработку, которая включает в себя следующие статьи затрат [21]:

1. Основная заработная плата разработчиков.
2. Дополнительная заработная плата разработчиков.
3. Отчисления на социальные нужды.
4. Прочие расходы.
5. Расходы на реализацию.
6. Общая сумма затрат на разработку и реализацию.

7.2.1 Основная заработная плата разработчиков

Расчет затрат на основную заработную плату команды разработчиков осуществляется в таблице 7.1 и производится по следующей формуле:

$$Z_o = K_{\text{пр}} \sum_{i=1}^n Z_{\text{ч},i} t_i, \quad (7.1)$$

где n – количество исполнителей, занятых разработкой конкретного ПО;
 $K_{\text{пр}}$ – коэффициент премий (1,5);
 $Z_{\text{ч},i}$ – часовая заработная плата i -го исполнителя, р.;
 t_i – трудоемкость работ, выполняемых i -м исполнителем, ч.

7.2.2 Дополнительная заработная плата разработчиков

Дополнительная заработная плата включает выплаты, предусмотренные законодательством о труде, и рассчитывается по формуле 7.2:

$$З_д = (З_о \cdot Н_д)/100\%, \quad (7.2)$$

где $З_о$ – затраты на основную заработную плату разработчиков, р.;
 $Н_д$ – норматив дополнительной заработной платы, (12%).

Таблица 7.1 – Расчет затрат на основную заработную плату разработчиков

Категория исполнителя	Месячный оклад, р.	Часовой оклад, р.	Трудоемкость работ, ч	Итого, р.
Руководитель проекта	1848	11	336	3696
Маркетолог	1848	11	168	1848
Инженер-программист	2100	12,5	672	8400
Итого				13944
Премия (50%)				6972
Всего основная заработная плата				20916

Дополнительная заработная плата разработчиков составит:

$$З_д = (20916 \cdot 12)/100 = 2509,92 \text{ р.}$$

7.2.3 Отчисления на социальные нужды

Отчисления на социальные нужды включают в предусмотренные законодательством отчисления в фонд социальной защиты (34%) и фонд обязательного страхования (0,6%) в процентах от основной и дополнительной заработной платы и рассчитываются по формуле 7.3:

$$Р_{соц} = ((З_о + З_д) \cdot Н_{соц})/100\%, \quad (7.3)$$

где $Н_{соц}$ – норматив отчислений на социальные нужды (34,6%).

Отчисления на социальные нужды составят:

$$Р_{соц} = ((20916 + 2509,92) \cdot 34,6)/100 = 8105,37 \text{ р.}$$

7.2.4 Прочие расходы

Прочие расходы включают изучение задания, литературы, анализ решений, проведение экономических расчетов и др.

Прочие расходы рассчитываются по формуле 7.4:

$$P_{\text{пр}} = (Z_o \cdot H_{\text{пр}})/100\%, \quad (7.4)$$

где $H_{\text{пр}}$ – норматив прочих расходов, 10%.

Таким образом, прочие расходы составят:

$$P_{\text{пр}} = (20916 \cdot 10)/100 = 2091,6 \text{ р.}$$

7.2.5 Расходы на реализацию

Расходы на реализацию включают продвижение, поддержку сервиса и др., и рассчитываются по формуле 7.5:

$$P_p = (Z_o \cdot H_p)/100\%, \quad (7.5)$$

где H_p – норматив расходов на реализацию, 30%.

Таким образом, расходы на реализацию составят:

$$P_p = (20916 \cdot 30)/100 = 6274,8 \text{ р.}$$

7.2.6 Общая сумма затрат на разработку и реализацию

Общая сумма затрат рассчитывается как сумма всех статей расходов на разработку сервиса. Расчет общей суммы затрат представлен в таблице 7.2.

Таблица 7.2 – Расчет общей суммы затрат

Наименование статьи затрат	Значение, р.
Основная заработная плата разработчиков	20916
Дополнительная заработная плата разработчиков	2509,92
Отчисления на социальные нужды	8105,37
Прочие расходы	2091,6
Расходы на реализацию	6274,8
Общая сумма затрат на разработку и реализацию	39897,69

7.3 Расчет экономического эффекта от реализации сервиса на рынке

Экономический эффект организации-разработчика сервиса представляет собой прирост чистой прибыли от его предоставления на рынке потребителям, величина которого зависит от таких параметров как количество посетителей сервиса, количество игр, сыгранных пользователем, количество игроков, получивших доступ к платному функционалу. Оценки этих значений могут варьироваться, и составляются на основе воронки продаж. Ее структура

непостоянна и зависит количества постоянных клиентов. Изначально, поток посетителей образуется только за счет показа рекламы в специализированных сервисах, таких как Google Ads (средняя стоимость за переход CPC = 0,2 USD; средняя стоимость за установку CPI = 1.22 USD), Facebook Ads Manager (CPC = 0,33; CPI = 1,04), Yandex Ads (CPC = 0,03), Telegram Ad Platform и др. [22, 23]. Так же поток посетителей образуется за счет размещения сервиса на специальных сайтах-агрегаторах, что является еще более продуктивным подходом для удержания аудитории, например, популярный сайт liftoff согласно отчету за 2023, имеет CPI = 0,59 для казуальных игр, что является наиболее точной оценкой [3]. Рассчитаем примерное количество переходов Π по формуле 7.6:

$$\Pi = \frac{P_p}{K \cdot CPC'} \quad (7.6)$$

где K – курс доллара, на момент написания 3,27 р.;
 CPC' – средняя стоимость за переход, 0,1 USD.

Исходя из расходов на реализацию и курса USD на момент создания сервиса, примерное количество переходов Π составит 19189 человек, примерное количество постоянных пользователей сайта Φ составит 3000 человек. Для расчета дохода с сервиса, были приняты следующие постоянные, доход с показа рекламы составляет 1-2 USD, доход с покупки платного контента составляет от 1 до 10 USD [24]. Согласно статистике, внутриигровые покупки обычно совершают от 1% до 5% игроков [25]. Количество переходов по рекламе возьмем из тех же сервисов, согласно статистике, для жанра казуальных игр количество переходов выше, и обычно составляет 0,5-3%, а доля просмотров по рекламе за вознаграждение значительно выше и составляет 80-90%, доля переходов 30%, удержание пользователей составляет 22% за день, 4% за неделю, 1,2% за месяц, среднее затраченное время на игру составляет 14 минут [26]. Количество игровых сессий N рассчитывалось по формуле 7.7:

$$N = \Pi \cdot \frac{C}{T} \quad (7.7)$$

где Π – количество переходов;
 T – среднее время на игровую сессию, 30 минут;
 C – среднее время удержания внимания, 14 минут.

Общее время игры составит 268645 минут, а количество игровых сессий $N = 8955$. Предполагаемый доход является суммой дохода от рекламы и от внутриигровых продаж. Примерный доход от рекламы D_p рассчитывается по формуле 7.8:

$$D_p = \left(\frac{N \cdot \chi_{\pi} \cdot \chi_k}{10000} + \frac{N \cdot \chi_{\pi}}{100} \right) \cdot T, \quad (7.8)$$

где N – число игровых сессий, 8955;
 χ_{π} – доля просмотров по рекламе за вознаграждение, 85%;
 χ_k – доля переходов по рекламе за вознаграждение, 30%;
 χ_b – доля переходов по рекламе без вознаграждения, 1%;
 T – тарифная ставка, 1,5 USD.
 Таким образом, примерный доход от рекламы составит:

$$D_p = (8955 \cdot (0,85 \cdot 0,3 + 0,01)) \cdot 1,5 = 3559,56 \text{ USD} = 11639,75 \text{ р.}$$

Примерный доход от продаж дополнительных возможностей D_{π} рассчитывается по формуле 7.9:

$$D_{\pi} = T \cdot \frac{N \cdot \chi_{\pi}}{100}, \quad (7.9)$$

где N – число игроков, 19189;
 χ_{π} – доля пользователей, приобретших дополнительный контент, 1,8%;
 T_d – средняя тарифная ставка дополнительного контента, 5 USD.
 Таким образом, примерный доход от продаж дополнительных возможностей составит:

$$D_{\pi} = (19189 \cdot 1,8)/100 \cdot 5 = 1727,01 \text{ USD} = 5647,32 \text{ р.}$$

Суммарный доход D вычисляется по формуле 7.10:

$$D = D_{\pi} + D_p, \quad (7.10)$$

где D_p – доход от рекламы; D_{π} – доход от продаж дополнительных возможностей.
 Суммарный доход составил 17287,07 р.

Расчет прибыли без учета налога на прибыль осуществляется по формуле 7.11:

$$\Pi = D - \text{НДС}, \quad (7.11)$$

где Π – цена реализации ПО заказчику, р.;
 НДС – сумма налога на добавленную стоимость, р.

Сумма НДС рассчитывается по формуле 7.12:

$$\text{НДС} = \frac{Д \cdot Н_{\text{дс}}}{100 + Н_{\text{дс}}}, \quad (7.12)$$

где $Н_{\text{дс}}$ – ставка налога на добавленную стоимость согласно действующему законодательству (20%).

Таким образом, налог составит:

$$\text{НДС} = (17287,07 \cdot 20)/(100 + 20) = 2881,18 \text{ р.}$$

Значение прибыли составит:

$$\Pi = 17287,07 - 2881,18 = 14405,90 \text{ р.}$$

Расчет чистой прибыли осуществляется по формуле 7.13:

$$\Pi_{\text{ч}} = \Pi \cdot (1 - Н_{\text{п}}/100), \quad (7.13)$$

где $Н_{\text{п}}$ – ставка налога на прибыль, (20%).

$$\Pi_{\text{ч}} = 14405,90 \cdot (1 - 20/100) = 11524,72 \text{ р.}$$

7.4 Расчет показателей экономической эффективности разработки и реализации сервиса на рынке

Для оценки экономической эффективности инвестиций рассчитываются следующие показатели экономической эффективности инвестиций [21]:

1. Простой срок окупаемости инвестиций (PP).
2. Средняя норма рентабельности инвестиций (ARR).
3. Чистый дисконтированный доход (NPV);
4. Динамический (дисконтированный) срок окупаемости инвестиций (DPP);
5. Индекс доходности инвестиций (PI).

Простой срок окупаемости инвестиций рассчитывается по формуле 7.14:

$$PP = \frac{\frac{1}{n} Z_t}{\sum_{t=1}^n \Delta \Pi_{\text{ч}t}}, \quad (7.14)$$

где Z_t – затраты в году t , равны 0 во всех годах, кроме 1, р.;

n – расчетный период;

$\Delta \Pi_{\text{ч}t}$ – прирост чистой прибыли в году t в результате реализации, р.

Средняя норма рентабельности инвестиций вычисляется по формуле 7.15:

$$ARR = \frac{1}{PP} \quad (7.15)$$

Чистый дисконтированный доход рассчитывается по формуле 7.16:

$$NPV = \sum_{t=1}^n \Delta\Pi_{qt} \cdot \alpha_t - \sum_{t=1}^n Z_t \cdot \alpha_t, \quad (7.16)$$

где α_t — коэффициент дисконтирования, рассчитанный для года t .

Динамический срок окупаемости инвестиций рассчитывается по формуле 7.17:

$$DPP = n, \text{ при котором } \sum_{t=1}^n \Delta\Pi_{qt} \cdot \alpha_t \geq \sum_{t=1}^n Z_t \cdot \alpha_t \quad (7.17)$$

Индекс доходности инвестиций рассчитывается по формуле 7.18:

$$PI = \frac{\sum_{t=1}^n \Delta\Pi_{qt} \cdot \alpha_t}{\sum_{t=1}^n Z_t \cdot \alpha_t} \quad (7.18)$$

Приведение доходов и затрат к настоящему времени осуществляется посредством домножения на коэффициент дисконтирования, который вычисляется по формуле 7.19:

$$\alpha_t = \frac{1}{(1 + d)^{t-t_p}}, \quad (7.19)$$

где d — требуемая норма дисконта, которая по своему смыслу соответствует устанавливаемому инвестором желаемому уровню рентабельности инвестиций, доли единицы;

t — порядковый номер года, доходы и затраты которого приводятся к расчетному году;

t_p — расчетный год, к которому приводятся доходы и инвестиционные затраты ($t_p = 1$).

Норма дисконта равна ставке рефинансирования Национального банка Республики Беларусь, $d = 0,095$.

Расчет параметров экономической эффективности инвестиций представлен в таблице 7.3.

По результатам расчета $DPP = 4$; $ARR = 0,289$; $PP = 3,46$ лет. Все параметры соответствуют области допустимых значений ($3 \leq DPP \leq 4$; $ARR \geq d$; $3 \leq PP \leq 4$).

Таблица 7.3 – Расчет экономической эффективности инвестиций

Показатель	Значение по годам расчетного периода			
	1-й год	2-й год	3-й год	4-й год
Результат				
Прирост чистой прибыли, р.	11524,72	11524,72	11524,72	11524,72
Дисконтированный результат, р.	11524,72	10524,85	9611,74	8777,84
Затраты				
Инвестиции в реализацию, р.	39897,69	0	0	0
Дисконтированные инвестиции, р.	39897,69	0	0	0
Чистый дисконтированный доход по годам, р.	-28372,97	10524,85	9611,74	8777,84
Чистый дисконтированный доход нарастающим итогом, р.	-28372,97	-17848,12	-8236,38	541,47
Коэффициент дисконтирования, доли единицы	1,000	0,913	0,834	0,762
Индекс доходности, доли единицы	0,289	0,553	0,794	1,014

7.5 Вывод технико-экономического обоснования

Согласно данным банков Республики Беларусь, на март 2024 года средняя ставка по долгосрочным кредитам для юридических лиц составляет 10–14%. Полученные значения основных экономических показателей: общие затраты на разработку 39897,69 руб., ожидаемая прибыль в год – 11524,72 руб., норма рентабельности 28,89%. Исходя из того, что показатель уровня рентабельности превышает ставку по долгосрочным кредитам, можно сделать вывод о целесообразности разработки и реализации программного средства.

Простой срок окупаемости составил 3 года 6 месяцев, динамической срок окупаемости проектируемого сервиса 4 года. Однако реальный срок окупаемости будет отличаться от расчетного, так как был сделан ряд допущений, например, фиксированный курс валют, стабильная ситуация на рынке, ставки по доходу и расходу на рекламу, процент посещений и удержания аудитории.

ЗАКЛЮЧЕНИЕ

Мною была проведена работа по проектированию, разработке и тестированию клиент-серверного приложения многопользовательской игры «Тетрис». Проект потребовал провести серьезную исследовательскую работу по актуализации концепта игры, поиска экономически успешного решения. В рамках работы был выполнен полный жизненный цикл создания программного обеспечения: произведено создание проектной документации, разработка технического задания, проектирование, разработка и тестирование сервиса. Для успешного завершения этого проекта потребовалось ознакомиться с правилами разработки проектной документации, изучить технологии разработки приложений на языке программирования JavaScript и TypeScript, а в частности программной платформы Node.js, предназначенной для написания серверных приложений на JavaScript и библиотеки React для реализации клиентской части. Потребовалось изучить и реализовать принципы клиент-серверного взаимодействия по принципам REST API и Socket.IO. Одним из ключевых этапов разработки было создание базы данных для проекта. Процесс проектирования базы данных был выполнен с учетом специфики предметной области, обеспечивая эффективное хранение и управление данными приложения. Для синхронизации приложения потребовалось изучить алгоритм предсказания и восстановления согласованности клиента и сервера и адаптировать его под специфику предметной области. Для создания графики потребовалось изучить принцип работы с современной реализацией графической библиотеки React Three Fiber. В результате работы был создан готовый продукт для использования и реализации. В дальнейшем планируется расширение и обновление продукта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Vokigames [Электронный ресурс]. – Казуальные игры: что это, какими они бывают и как развивается жанр. – Режим доступа: <https://vokigames.com/ru/kazualnye-igry-cto-eto-kakimi-oni-byvayut-i-kak-razvivaetsya-zhanr/>. – Дата доступа: 04.04.2024 г.
- [2] eGamersWorld [Электронный ресурс]. – Тенденции мобильного гейминга: От казуальных игр до портативных AAA-тайтлов. – Режим доступа: <https://rus.egamersworld.com/blog/mobile-gaming-trends-from-casual-games-to-portable-XQprUQi1tX>. – Дата доступа: 04.04.2024 г.
- [3] Udonis [Электронный ресурс]. – Casual Games Market in 2024: Trends, CPIs, ROAS, and More!. – Режим доступа: <https://www.blog.udonis.co/mobile-marketing/mobile-games/casual-games>. – Дата доступа: 04.04.2024 г.
- [4] Statista [Электронный ресурс]. – Casual Games - Worldwide. – Режим доступа: <https://www.statista.com/outlook/amo/app/games/casual-games/worldwide>. – Дата доступа: 03.05.2024 г.
- [5] Capermint [Электронный ресурс]. – How to Develop Hyper Casual Game & How much does it costs?. – Режим доступа: <https://www.capermint.com/blog/hyper-casual-game-development/>. – Дата доступа: 04.04.2024 г.
- [6] ROOM8 Studio [Электронный ресурс]. – SMART & CASUAL: THE STATE OF TILE PUZZLE GAMES LEVEL DESIGN. – Режим доступа: <https://room8studio.com/news/smart-casual-the-state-of-tile-puzzle-games-level-design-part-1/>. – Дата доступа: 04.04.2024 г.
- [7] CrazyLabs [Электронный ресурс]. – The Ultimate Guide for Hyper-Casual Game Mechanics in 2023. – Режим доступа: <https://www.crazylabs.com/blog/guide-for-hyper-casual-game-mechanics/>. – Дата доступа: 04.04.2024 г.
- [8] Udonis [Электронный ресурс]. – Progression Systems in Mobile Games: the Ultimate Guide. – Режим доступа: <https://www.blog.udonis.co/mobile-marketing/mobile-games/progression-systems>. – Дата доступа: 04.04.2024 г.
- [9] Tetris [Электронный ресурс]. – Corporate Bios. – Режим доступа: <https://tetris.com/bios>. – Дата доступа: 04.04.2024 г.
- [10] RetroGames [Электронный ресурс]. – Tetris - Nintendo NES system. – Режим доступа: https://www.retrogames.cz/play_1030-NES.php. – Дата доступа: 04.04.2024 г.
- [11] Chess.com [Электронный ресурс]. – Шахматы по сети. – Режим доступа: <https://www.chess.com/>. – Дата доступа: 04.04.2024 г.
- [12] Tricky Towers [Электронный ресурс]. – Tricky Towers. About. – Режим доступа: <https://www.trickytowers.com>. – Дата доступа: 04.04.2024 г.
- [13] Zynga [Электронный ресурс]. – An easy-to-learn, addictive, and minimalist puzzle game you won't be able to put down. – Режим доступа: <https://www.zynga.com/games/1010>. – Дата доступа 04.04.2024 г.

[14] Tetris Wiki [Электронный ресурс]. – Random Generator. – Режим доступа: https://tetris.fandom.com/wiki/Random_Generator. – Дата доступа: 04.04.2024 г.

[15] CUBA.platform [Электронный ресурс]. – Servers to support 1000 concurrent users. – Режим доступа: <https://forum.cuba-platform.com/t/servers-to-support-1000-concurrent-users/805/2>. – Дата доступа: 04.04.2024 г.

[16] Куликов, С.С. Реляционные базы данных в примерах: практическое пособие для программистов и тестировщиков. / С.С. Куликов. - ЕРАМ Systems, 2020–2023. – 422 с г.

[17] Adam Russel Design [Электронный ресурс]. – Using Fibonacci for Game Balance. – Режим доступа: <https://adamrusselldesign.wordpress.com/2012/06/25/using-fibonacci-for-game-balance/>. – Дата доступа: 06.05.2024 г.

[18] Gabriel Gambetta [Электронный ресурс]. – Fast-Paced Multiplayer (Part II): Client-Side Prediction and Server Reconciliation, 2022. – Режим доступа: <https://www.gabrielgambetta.com/client-side-prediction-server-reconciliation.html>. – Дата доступа: 24.04.2024 г.

[19] Habr [Электронный ресурс]. – Как работает single sign-on (технология единого входа)?. – Режим доступа: <https://habr.com/ru/companies/nixys/articles/563244/>. – Дата доступа: 04.04.2024 г.

[20] Куликов, С.С. Тестирование программного обеспечения: базовый курс. / С.С. Куликов. – 3-е изд. - ЕРАМ Systems, 2015–2023. – 301 с г.

[21] Экономика проектных решений: методические указания по экономическому обоснованию дипломных проектов : учебно-методическое пособие / Горовой В. Г. [и др.]. – Минск : БГУИР, 2021. – 107 с г.

[22] Business of Apps [Электронный ресурс]. – Cost Per Install (CPI) Rates (2023). – Режим доступа: <https://www.businessofapps.com/ads/cpi/research/cost-per-install/>. – Дата доступа: 30.03.2024 г.

[23] Store Growers [Электронный ресурс]. – YouTube Ads Benchmarks (2024). – Режим доступа: <https://www.storegrowers.com/youtube-ads-benchmarks/>. – Дата доступа: 30.03.2024 г.

[24] Social Shepherd [Электронный ресурс]. – 19 Essential Google Ads & PPC Statistics You Need to Know in 2024. – Режим доступа: <https://thesocialshepherd.com/blog/google-ads-ppc-statistics/>. – Дата доступа: 30.03.2024 г.

[25] ThinkApps [Электронный ресурс]. – App Monetization Statistics: Freemium vs Premium vs Paymium. – Режим доступа: <https://thinkapps.com/blog/post-launch/paid-vs-freemium-app-monetization-statistics/>. – Дата доступа: 30.03.2024 г.

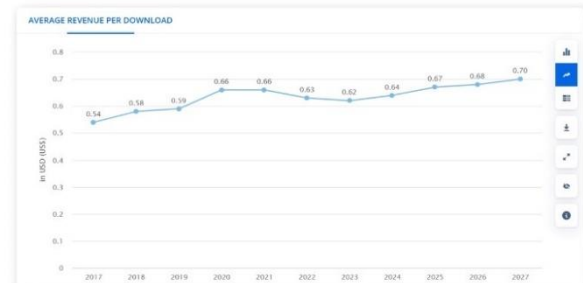
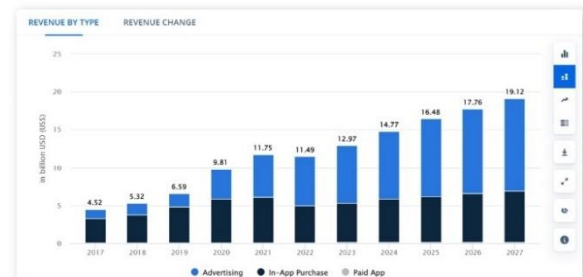
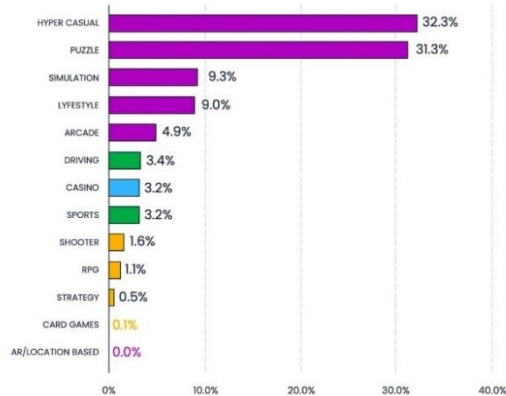
[26] Udonis [Электронный ресурс]. – Puzzle Games Report on Players, Monetization, and Advertising. – Режим доступа: <https://www.blog.udonis.co/mobile-marketing/mobile-games/puzzle-games-report/>. – Дата доступа: 30.03.2024 г.

ПРИЛОЖЕНИЕ А (обязательное) Исследование индустрии казуальных игр

Исследование индустрии казуальных игр

Udonis	TETRIS	TRICKY TOWERS	1010!	Мой проект
ПВП	★	★	★	★
командная игра				★ команда на команду
система рейтинга	★	★	★	★ подбор соперника таблица лидеров
социальный аспект			★	★ встроенная соцсеть
мобильное приложение			★	~ ограниченные возможности
простота	★	~	~	★
доступность	~		★	★ веб-клиент
разнообразие	★	★	★	!!! новые типы блоков
краткие сессии			~	~
яркое оформление	★	★	★	!!!

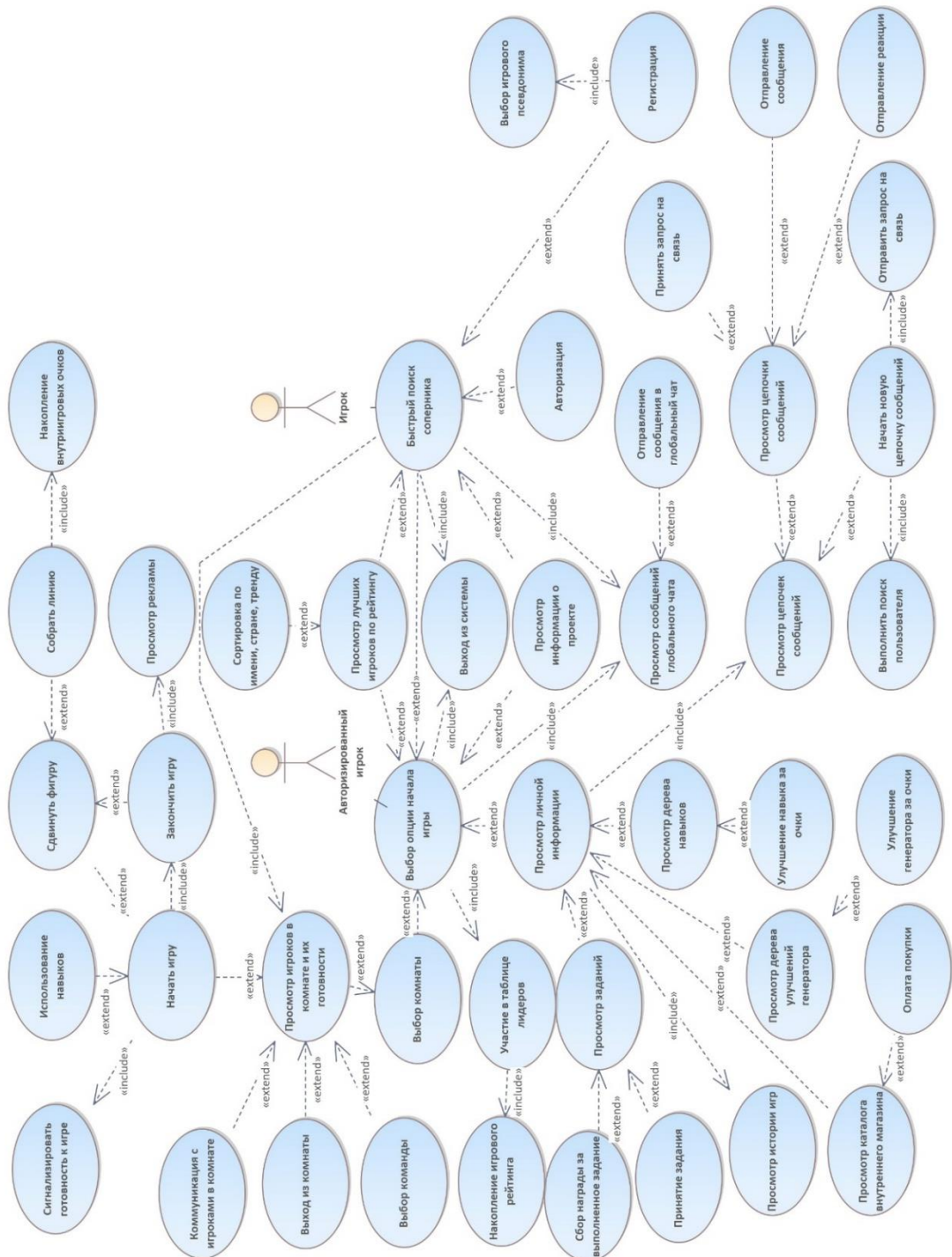
Genre drivers across all installs



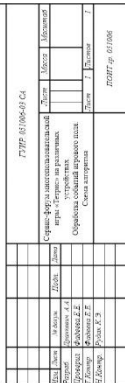
ПРИЛОЖЕНИЕ Б (обязательное)

Диаграмма вариантов использования сервиса

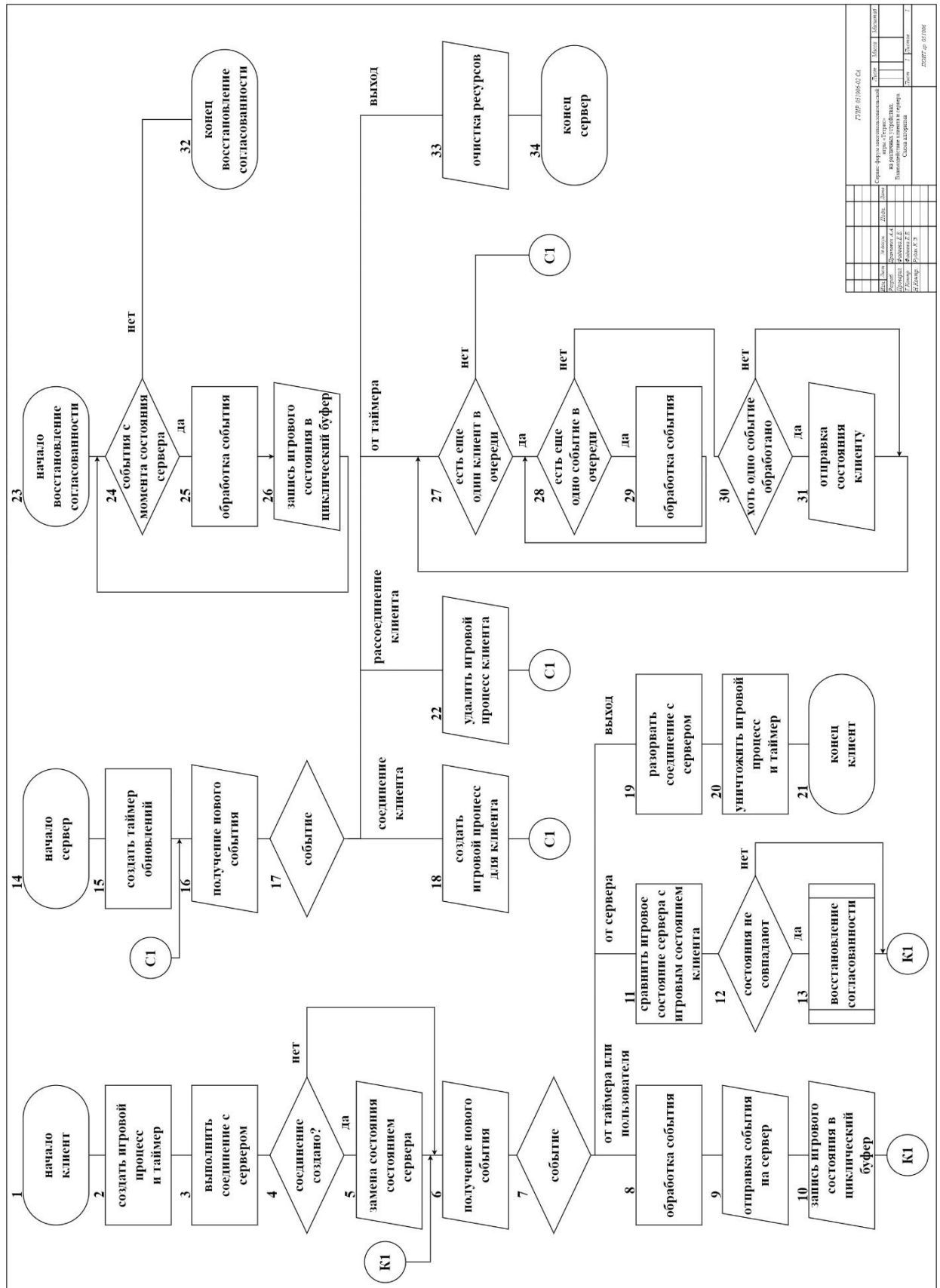
Диаграмма вариантов использования



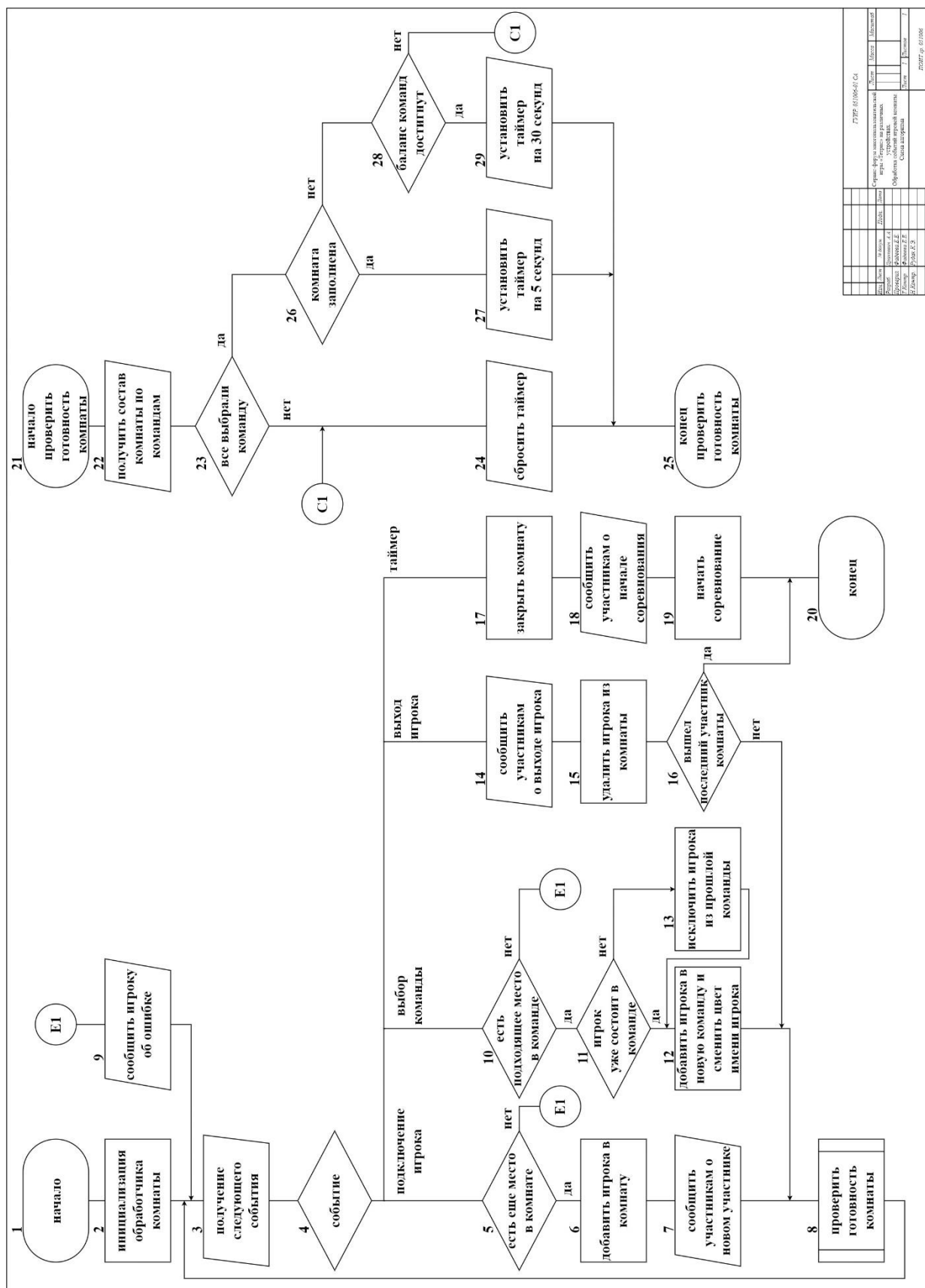
Обработка событий игрового поля



ПРИЛОЖЕНИЕ Г **(обязательное)** **Взаимодействие клиента и сервера**



ПРИЛОЖЕНИЕ Д **(обязательное)** **Обработка событий игровой комнаты**



ПРИЛОЖЕНИЕ Ж

(обязательное)

Текст программы модуля игрового взаимодействия с сервером

```
import {GameInput, GameState} from "../server_client_globals.js";
import {BUFFER_SIZE, TPS} from "../server_client_globals.js";
import {Tetris} from "../tetris.js";
import {sequelize} from "../bin/db.js";
import {QueryTypes} from "sequelize";
import {PlayerData} from "../player_data";

export class ServerGameSessionControl {
  // Database user
  data: PlayerData = null;
  game = null;
  socket;
  io;

  // competition mode does not allow pause or stop
  competition: boolean = false;
  onCompetitionViolation: (data: PlayerData) => void;
  onCompetitionEnd: (data: PlayerData, score: number) => void;

  inputQueue: GameInput[] = [];
  stateBuffer: GameState[] = new Array(BUFFER_SIZE);

  currentTick: number;
  currentEvent: number;
  time;
  timeStarted;
  timeGameStarted;

  gameTick;
  gameTime; // time of last processed evt

  // special boolean to show if game was just resumed/paused
  justResumed: boolean = false;

  constructor(socket, io, data: PlayerData = null) { // happens on connection
    this.io = io;
    this.socket = socket;
    this.data = data;
  }

  static async getLeaderboard() {
    const topRecordsQuery = `
      SELECT u.user_nickname, r.user_max_score, u.user_id, u.user_region
      FROM users u
      INNER JOIN (
        SELECT record_user_id, MAX(record_score) as user_max_score
        FROM records
        GROUP BY record_user_id
      ) r ON u.user_id = r.record_user_id
      ORDER BY r.user_max_score DESC
      LIMIT 15;
    `;

    return await sequelize.query(topRecordsQuery, { type: QueryTypes.SELECT });
  }

  startCompetition(seed: number,
    onCompetitionViolation: (data: PlayerData) => void,
    onCompetitionEnd: (data: PlayerData, score: number) => void)
  {
    this.game.initializeFrom(seed);
  }
}
```

```

    this.game.paused = false;
    this.competition = true;
    this.onCompetitionViolation = onCompetitionViolation;
    this.onCompetitionEnd = onCompetitionEnd;
    this.time = performance.now();
    this.timeStarted = this.time;
    this.timeGameStarted = this.time;
    this.currentTick = 0;
    this.currentEvent = 0;
    this.gameTime = 0;
    this.gameTick = 0;
    // clear input
    this.inputQueue.length = 0;
    // set game state buffer
    const bufferIndex = this.currentEvent % BUFFER_SIZE;
    this.stateBuffer[bufferIndex] = new GameState(this.currentTick, this.currentEvent,
        this.time - this.timeStarted, this.game.deepCopy());
    //
    // force sync
    this.socket.emit('game sync', this.stateBuffer[0]);
}

onSync(data: PlayerData) {
    this.data = data
    //
    this.justResumed = false;
    // init
    this.time = performance.now();
    this.timeStarted = this.time;
    this.timeGameStarted = this.time;
    this.currentTick = 0;
    this.currentEvent = 0;
    this.gameTime = 0;
    this.gameTick = 0;
    // create brand-new game
    this.game = new Tetris(null);
    this.game.gameOverCallback = (score: number, newRecord: boolean) =>
this.onGameOver(score, newRecord);
    this.game.scoreUpdateCallback = (score: number, delta: number) =>
this.onScoreUpdate(score, delta);
    // get user nickname
    this.game.status = data?.getStatus()
    // clear input
    this.inputQueue.length = 0;
    // set game state buffer
    const bufferIndex = this.currentEvent % BUFFER_SIZE;
    this.stateBuffer[bufferIndex] = new GameState(this.currentTick, this.currentEvent,
        this.time - this.timeStarted, this.game.deepCopy());
    // send game state
    this.socket.emit('game sync', this.stateBuffer[bufferIndex]);
}

onInput(input: GameInput) {
    //
    // filter out pause and stop events
    if (this.competition && this.onCompetitionViolation && [27, 80,
82].includes(input.input.id)) {
        // drop competition mode
        this.competition = false;
        this.onCompetitionViolation(this.data);
    }
    //
    this.inputQueue.push(input);
    // TOO MANY PACKETS
    if (this.inputQueue.length > BUFFER_SIZE) {
        this.socket.disconnect("TOO MANY PACKETS");
    }
}

```

```

    // if avg dt between packets is too small -- kick
    // if > 30 packets / tick -- kick
}

onScoreUpdate(score: number, delta: number) {
  if (this.data?.ru) {
    this.data.ru.ru_last_score = score;
    this.io.to(this.data?.getRoomId()).emit('game score', {
      ru_last_score: score,
      ru_user_id: this.data.ru.ru_user_id
    });
  }
}

onGameOver(score: number, newRecord: boolean) {
  this.socket.emit('game over');
  //
  if (this.competition && this.onCompetitionEnd) {
    this.competition = false;
    this.onCompetitionEnd(this.data, score);
  }
  const gameTime = this.time - this.timeGameStarted;
  this.timeGameStarted = this.time;
  // user is in the room
  (async () => {
    //
    await this.data.makeGameRecord({
      record_score: this.game.score,
      record_time_elapsed: gameTime,
      record_figures_placed: this.game.placed,
    });
    this.io.to(this.data.getRoomId()).emit('room game over', this.data?.ru);
  })();
}

process() {
  // game is null - so there is nothing to be processed
  if (!this.game) {
    return;
  }
  // there are no pending messages - ignore
  if (this.inputQueue.length === 0) {
    return;
  }
  // get current time
  this.time = performance.now();
  //
  let bufferIndex: number;
  while (this.inputQueue.length > 0) {
    const input = this.inputQueue.shift();
    bufferIndex = input.event % BUFFER_SIZE;
    // timer event
    let gameDelta = (input.time - this.gameTime);
    //
    if (this.game.playing && !this.game.paused) {
      const gameTicksSkipped = Math.floor(gameDelta / (this.game.softDrop ?
        (this.game.tickSpeed / 4) : this.game.tickSpeed));
      if (gameTicksSkipped > 0) {
        // 1 or more game ticks passed since last update
        let gameTicksToProcess = 0;
        if (input.input.id !== 7) { // means that at least one 7 event is skipped
          gameTicksToProcess++;
        } else {
          this.gameTick++; // apply effect of 7 event id
        }
        if (!this.justResumed) { // means that we need to process all skipped ticks
          gameTicksToProcess += gameTicksSkipped - 1;
        }
      }
    }
  }
}

```

```

        }
        for (let i = 0; i < gameTicksToProcess; ++i) {
            this.gameTick++;
            this.game.processEventSilent(7);
        }
        //
        this.gameTime = input.time;
    }
    // handle pause
    const isPaused: boolean = this.game.paused;
    // process event
    this.game.processEventSilent(input.input.id);
    this.stateBuffer[bufferIndex] = new GameState(input.tick, input.event, input.time,
this.game.deepCopy());
    //
    this.justResumed = ( isPaused || this.game.paused ) && !( isPaused &&
this.game.paused );
    }
    // update client
    this.socket.emit('game update', this.stateBuffer[bufferIndex]);
    }
}

```


ПРИЛОЖЕНИЕ И (обязательное)

Текст программы модуля игрового взаимодействия с клиентом

```
// shared
import {GameInput, GameState} from "./server_client_globals";
import {BUFFER_SIZE, TPS} from "./server_client_globals";
import type {Tetris} from "./tetris";

function arrayEquals(a: any[], b: any[]) {
    return a.length === b.length &&
        a.every((val, index) => isDeepEqual(val, b[index]));
}

function isDeepEqual (object1: any, object2: any) {

    const isObjects = isObject(object1) && isObject(object2);
    const isArray = Array.isArray(object1) && Array.isArray(object2);
    const other = !isArray && !isObjects;
    if (other) {
        return object1 === object2;
    }
    if (isArray) {
        return arrayEquals(object1, object2);
    }

    const objKeys1 = Object.keys(object1);
    const objKeys2 = Object.keys(object2);

    if (objKeys1.length !== objKeys2.length) return false;

    for (let key of objKeys1) {
        const value1 = object1[key];
        const value2 = object2[key];

        if(!isDeepEqual(value1, value2)) {
            return false;
        }
    }
    return true;
}

function isObject (object) {
    return object != null && typeof object === "object";
}

export class ClientGameSessionControl {

    socket = null;
    // were master over game
    game: Tetris = null;
    //shared
    currentTick;
    currentEvent;
    time;
    //
    gameTick;
    gameTime;

    // client specific
    inputBuffer: GameInput[] = new Array(BUFFER_SIZE);
    stateBuffer: GameState[] = new Array(BUFFER_SIZE);
    lastProcessedState: GameState;
    lastServerProcessedState: GameState;
```

```

//
timer: any;
timeStarted;
//
globalTime;

constructor(game, socket) {
  this.game = game;
  this.socket = socket;

  const time = performance.now();
  this.time = time
  this.gameTime = time;
  this.timeStarted = time;
  this.globalTime = time;
  this.currentTick = 0; // TODO
  this.currentEvent = 0;
  this.gameTick = 0;
  this.timer = setInterval(this.onTimer, 0);
}

// on timer
onTimer = () => {
  const now = performance.now();
  const tickDelta = (now - this.time);
  const gameDelta = (now - this.gameTime);

  this.globalTime = now;
  if (tickDelta > 1000 / TPS) {
    this.currentTick++;
    this.time = now;
  }
  if (this.game.playing && !this.game.paused) {
    if (gameDelta > (this.game.softDrop ?
      (this.game.tickSpeed / 4) : this.game.tickSpeed)) {
      this.gameTick++;
      this.gameTime = now;
      this.processEvent(7);
    }
  }
}

destroy() {
  clearInterval(this.timer);
}

// client specific
sync() {
  this.socket.emit('sync');
}

// must be linked to socket
onServerUpdate(serverState: GameState) {
  this.lastServerProcessedState = serverState;
  // check if server & client are synchronized
  const clnt = this.stateBuffer[this.lastServerProcessedState.event % BUFFER_SIZE];
  const serv = this.lastServerProcessedState
  if (!isDeepEqual(clnt, serv))
  {
    this.#reconcile();
  }
}

onServerConnect() {
  this.game.status = "connected";
  this.game.renderCallback && this.game.renderCallback();
}

```

```

onServerDisconnect () {
    if (this.game.playing && this.game.score > 0) {
        this.game.status = "connLost";
    } else {
        this.game.status = "offline";
    }
    this.game.renderCallback && this.game.renderCallback();
}

onServerSynced(serverState: GameState) {
    // setup all variables to match server's ones
    this.lastServerProcessedState = serverState;
    this.currentTick = serverState.tick;
    // +1 means that serverState is processed
    // and added to state buffer
    this.currentEvent = serverState.event + 1;
    this.time = performance.now();
    this.gameTime = performance.now();
    this.timeStarted = performance.now();
    //
    this.#reconcile();
}

#reconcile() {
    //
    this.lastProcessedState = this.lastServerProcessedState;
    const serverBufferIndex = this.lastServerProcessedState.event % BUFFER_SIZE;
    // reset problematic point
    this.stateBuffer[serverBufferIndex] = this.lastServerProcessedState;
    this.game.constructFromPrototype(this.lastServerProcessedState.state);
    // simulate all ticks up to the current
    let eventToProcess = this.lastServerProcessedState.event + 1;
    while (eventToProcess < this.currentEvent) {
        const bufIndex = eventToProcess % BUFFER_SIZE;
        const input = this.inputBuffer[bufIndex];
        // reprocess input
        this.game.processEventSilent(input.input.id);
        this.stateBuffer[bufIndex] = new GameState(input.tick, input.event, input.time,
this.game.deepCopy());
        //
        eventToProcess++;
    }
    // call update with latest state
    this.game.renderCallback && this.game.renderCallback();
}
// Clients update
processEvent(event: number) {
    // determine new tick value
    // const newTime = performance.now(); // milliseconds, floating number
    // const ticksPassed = Math.floor(newTime / TPS) - Math.floor(this.time / TPS);
    // this.currentTick += ticksPassed;
    // this.time = newTime;

    // guarantee 7th event when time from last 7th event is expired
    if (this.game.playing && !this.game.paused && event !== 7
        && ((this.globalTime - this.gameTime) >
            (this.game.softDrop ? this.game.tickSpeed / 4 : this.game.tickSpeed))) {
        // force process 7 event and ignore next one
        this.onTimer();
    }
    // add input to input buffer
    let bufferIndex = this.currentEvent % BUFFER_SIZE;
    // add new Input event
    this.inputBuffer[bufferIndex] = new GameInput(this.currentTick, this.currentEvent,
        this.globalTime - this.timeStarted, {id: event});
    // process event

```

```
    this.game.processEvent(event);
    // add game state to state buffer
    this.stateBuffer[bufferIndex] = new GameState(this.currentTick, this.currentEvent,
        this.globalTime - this.timeStarted, this.game.deepCopy());
    // increase number of processed events
    this.currentEvent++;
    // send input to server
    this.socket.emit('input', this.inputBuffer[bufferIndex]);
  }
}
```

ПРИЛОЖЕНИЕ К (обязательное) Текст программы модуля игровой логики

```
// EXACT THE SAME CODE AS ON THE CLIENT

export const RANDOM_MAX = 0x7FFFFFFF;
export class Random {

  seed: number;
  prev: number;

  constructor(seed: number, prev: number = undefined) {
    this.seed = seed;
    this.prev = prev ? prev : seed;
  }

  nextRandInt(from: number, to: number) : number {
    //; pseudo random generator (A*x + B) mod N
    let x = this.prev;
    x = (29 * x + 47) & RANDOM_MAX;
    this.prev = x;

    const res = x % (to - from + 1) + from;
    return Number(res);
  }
}

export const FIELD_W = 12;
export const FIELD_H = 23;
export const RECT_MODIFIER = 0.95;

const SPEED_MUL = 0.90;

const INC_EVERY_FIGS = 15; // 2^N-1

const TOP_LINE = 0;

const FIG_START_Y = -1;

const START_TICK_SPEED = 800;

export const COLOR_TABLE = [
  [1-0.071, 1-0.071, 1-0.071],
  [1-0.271, 1-0.271, 1-0.271],
  [1-0.1, 1-0.1, 1-0.1],
  [1.0, 1.0, 1.0],
  [1.0, 0.5098, 0.0],
  [0.0, 0.0, 1.0],
  [0.2549, 0.0, 1.0],
  [0.5098, 0.0, 1.0],
  [0.745, 0.0, 1.0],
  [1.0, 0.0, 0.745],
  [1.0, 0.0, 0.5098],
  [1.0, 0.0, 0.2549],
  [1.0, 0.0, 0.0],
  [1.0, 0.2549, 0.0],
  [1.0, 0.745, 0.0],
  [1.0, 1.0, 0.0],
  [0.745, 1.0, 0.0],
  [0.5098, 1.0, 0.0],
  [0.2549, 1.0, 0.0],
  [0.0, 1.0, 0.2549],
  [0.0, 1.0, 0.5098],
]
```

```

    [0.0, 1.0, 0.745    ],
    [0.0, 1.0, 1.0      ],
    [0.0, 0.745, 1.0    ],
    [0.0, 0.5098, 1.0   ],
    [0.0, 0.2549, 1.0   ]
];

export function mergeColors(col1: number, col2: number) {
  if (col1 === 0) return col2;
  if (col2 === 0) return col1;
  if (col1 === 3 || col2 === 3) return 3
  return Math.floor(((col1 - 4) + (col2 - 4)) / 2) + 4
}

export const STATUS_TABLE = Object.freeze({
  offline:      '@OFFLINE',
  connected:    '@ON-LINE',
  registered:    '@REGSTRD',
  rejected:      '@REJECTED',
  keyFail:       '@UUIDERR',
  connLost:      '@CNCLOST',
});

export const FigureType = Object.freeze({
  none: '',
  tnt: 'tnt',
  ghost: 'ghost',
  liquid: 'liquid',
  sand: 'sand',
  border: 'border',
  at(id: number) {
    if (id === null || id === undefined) return id
    return FigureType[Object.keys(FigureType)[id]];
  },
  from(item: string) {
    return Object.keys(FigureType).indexOf(item)
  }
});
export const FigureTypeLength = Object.keys(FigureType).length - 3;
export const FigureGhostId = FigureType.from('ghost')

type BlockType = {
  color: number,
  type: number | undefined
};

type EffectType = {
  type: number,
}

export class Figure {
  // defines fig number
  id: number;
  // defines rotation 0..3
  rotation: number;
  // defines x pos
  x: number;
  // defines y pos;
  y: number;
  // defines color
  color: number;
  value: number;
  //
  type: number;

  constructor(id: number, prototype: any = undefined) {
    if (prototype instanceof Random) {

```

```

        this.id = id;
        this.toDefaultPos();
        this.#generateColor(prototype);
        this.#generateType(prototype);
    } else if (prototype) {
        Object.assign(this, prototype);
    } else {
        this.id = id;
        this.toDefaultPos();
        this.#generateColor(new Random(0));
    }
}

toDefaultPos() {
    this.rotation = 0;
    this.y = FIG_START_Y;
    this.x = FIELD_W/2 - 2; //nextRandInt(1, FIELD_W - 1);
    this.from();
}

from(id: number = undefined, rotation: number = undefined) {
    this.value = Figure.figArr[id || this.id][rotation || this.rotation];
}

#generateType(random: Random) {
    this.type = random.nextRandInt(0, FigureTypeLength - 1);
}

#generateColor(random: Random) {
    this.color = random.nextRandInt(4, COLOR_TABLE.length - 4);
}

rotate(diff) {
    this.rotation = (this.rotation + diff + 4) & 3;
    this.from();
}

static figArr = [
    [0b0110_0110_0000_0000,      0b0110_0110_0000_0000,      0b0110_0110_0000_0000,
0b0110_0110_0000_0000], // O
    [0b0000_1111_0000_0000,      0b0010_0010_0010_0010,      0b0000_0000_1111_0000,
0b0100_0100_0100_0100], // I
    [0b1000_1110_0000_0000,      0b0110_0100_0100_0000,      0b0000_1110_0010_0000,
0b0100_0100_1100_0000], // J
    [0b0010_1110_0000_0000,      0b0100_0100_0110_0000,      0b0000_1110_1000_0000,
0b1100_0100_0100_0000], // L
    [0b0110_1100_0000_0000,      0b0100_0110_0010_0000,      0b0000_0110_1100_0000,
0b1000_1100_0100_0000], // S
    [0b1100_0110_0000_0000,      0b0010_0110_0100_0000,      0b0000_1100_0110_0000,
0b0100_1100_1000_0000], // Z
    [0b0100_1110_0000_0000,      0b0100_0110_0100_0000,      0b0000_1110_0100_0000,
0b0100_1100_0100_0000] // T
]
static figCount = Figure.figArr.length;

static genSequence(random: Random) {
    const figCount = Figure.figCount;
    const sequence = [];
    for (let i = 0; i < figCount; i++) {
        sequence.push(new Figure(i, random));
    }
    // shuffle array
    for (let i = figCount - 1; i > 0; i--) {
        const j = random.nextRandInt(0, i);
        [sequence[i], sequence[j]] = [sequence[j], sequence[i]];
    }
    return sequence;
}

```

```

    }
}

export class Tetris {
  // current figure
  currentFigure: Figure;
  figPreviewY: number;
  heldFigure = null;
  nextFigures: Figure[];
  nextFigureNumber: number;
  //
  paused: boolean = false;
  playing: boolean = false;
  softDrop: boolean = false;
  held: boolean = false;
  score: number = 0;
  highScore: number = 0;
  placed: number = 0;
  tickSpeed: number = START_TICK_SPEED;
  //
  timePlayed: number;

  field: BlockType[] = new Array(FIELD_W * FIELD_H);

  effects: EffectType[] = new Array(FIELD_W * FIELD_H);

  // render callback
  renderCallback: () => void = null;
  gameOverCallback: (score: number, newRecord: boolean) => void = null;
  scoreUpdateCallback: (score: number, delta: number) => void = null;

  status = "offline"

  random: Random = null;

  name: string = "@DEFAULT"

  deepCopy() {
    const clone = JSON.parse(JSON.stringify(this));
    // clear callbacks (they are different)
    clone.renderCallback = null;
    clone.gameOverCallback = null;
    clone.scoreUpdateCallback = null;
    clone.effects = null;
    return clone;
  }

  initializeFrom(seed: number) {
    //this.#endGame(); TODO mk it silent or ???
    this.#initialize(seed);
  }

  constructor(/*renderCallback: (buffer: RenderBuffer) => void,*/ prototype = undefined){
    if (prototype) {
      this.constructFromPrototype(prototype);
    } else {
      this.#initialize();
    }
    // this.callback = renderCallback;
  }

  constructFromPrototype(prototype) {
    // preserve callbacks functions
    const callback = this.renderCallback;
    const gameOverCallback = this.gameOverCallback;
    const scoreUpdateCallback = this.scoreUpdateCallback;
    const effects = this.effects;
    //

```



```

    Object.assign(this, prototype);
    // convert figures to objects too
    this.currentFigure = this.currentFigure && new Figure(undefined, this.currentFigure);
    this.heldFigure = this.heldFigure && new Figure(undefined, this.heldFigure);
    for (let i = 0; i < this.nextFigures.length; ++i) {
        this.nextFigures[i] = this.nextFigures[i] && new Figure(undefined,
this.nextFigures[i]);
    }
    // setup random
    this.random = new Random(this.random.seed, this.random.prev);
    // setup callbacks back
    this.renderCallback = callback;
    this.gameOverCallback = gameOverCallback;
    this.scoreUpdateCallback = scoreUpdateCallback;
    this.effects = effects;
    // safety check for effects
    if (this.effects[0] === undefined) {
        this.#initEffects()
    }
}

//##### KEY EVENT #####
// - processes a key code, stored in key
// - then calls render
// - Special control codes:
//   - Game update - 7 (undefined in VK table)
//   - Ignore downward collision - 0 (only for actual figure, not preview)
//   - Shift key up - 15 (undefined in VK table)
processEvent(key: number) {
    this.processEventSilent(key);
    this.renderCallback && this.renderCallback();
}

//##### KEY EVENT #####
// - processes a key code, stored in key
// - Special control codes:
//   - Game update - 7
//   - Ignore downward collision - 0 (only for actual figure, not preview)
processEventSilent(key: number) {
    if (key === 27) { // ESC
        this.#endGame();
        this.paused = false;
        return;
    }
    if (key === 82) { // 'R'
        this.#endGame();
        this.#initialize();
        return;
    }
    if (this.playing === false)
        return;
    if (key === 80) { // 'P'
        this.paused = !this.paused;
        return;
    }
    if (key === 15) { // shift up
        this.softDrop = false;
        return;
    }
    if (this.paused === true)
        return;
    if (key === 16) { // shift
        this.softDrop = true;
        return;
    }
    if (!this.held && key === 72) // 'H', buffer used
    {

```

```

        this.held = true;
        [this.currentFigure, this.heldFigure] = [this.heldFigure, this.currentFigure];
        if (!this.currentFigure) {
            this.#nextFigure();
        } else {
            this.currentFigure.toDefaultPos();
        }
    }
    // store data b4 changes
    let fig = this.currentFigure;
    const xPos = fig.x;
    const rot = fig.rotation;
    switch (key) {
        case 38: // ^
            fig.rotate(-1);
            break;
        case 40: // v
            fig.rotate(+1);
            break;
        case 37: // <
            fig.x--;
            break;
        case 39: // >
            fig.x++;
            break;
    }
    if (this.#collideFigure(fig)) {
        // restore changes
        fig.rotation = rot;
        fig.rotate(0); // apply rotation
        fig.x = xPos;
    }
    if (key === 7) { // update event
        this.timePlayed += this.tickSpeed;
        fig.y++;
        this.#processFieldEffects();
    }
    // save old Y
    let yPos = fig.y;
    // go down loop
    if (!(this.softDrop && fig.type === FigureGhostId)) {
        while (!this.#collideFigure(fig)) {
            fig.y++;
        }
        fig.y--;
    }
    // collided
    if (key !== 0 && (key === 32 || fig.y < yPos)) // collision hard
    {
        this.#placeFigure(fig);
        if (this.#checkOnEnd()){
            this.#endGame();
        } else {
            this.held = false;
            this.placed++;
            this.#checkOnLine();
            this.#nextFigure();

            if ((this.placed & INC_EVERY_FIGS) === 0) {
                this.tickSpeed *= SPEED_MUL;
            }
        }
    }
    } else {
        [yPos, fig.y] = [fig.y, yPos];
        this.figPreviewY = yPos;
    }
}

```

```

#nextFigure() {
  this.currentFigure = new Figure(undefined, this.nextFigures[this.nextFigureNumber]);
  this.nextFigureNumber++;
  if (this.nextFigureNumber >= Figure.figCount) {
    this.nextFigureNumber = 0;
    this.nextFigures = Figure.genSequence(this.random);
  }
  this.processEventSilent(0);
}

#initField() {
  for (let i = 0; i < FIELD_H - 1; ++i) {
    let posY = i * FIELD_W;
    const lastPosY = posY + FIELD_W - 1;
    this.field[posY] = {color: 1, type: FigureType.from('border')};
    this.field[lastPosY] = {color: 1, type: FigureType.from('border')};
    for (let j = posY + 1; j < lastPosY; ++j) {
      this.field[j] = {color: 0, type: 0};
    }
  }
  for (let j = (FIELD_H - 1) * FIELD_W; j < FIELD_H * FIELD_W; ++j) {
    this.field[j] = {color: 1, type: FigureType.from('border')};
  }
}

#initEffects () {
  for (let i = 0; i < FIELD_H; ++i) {
    let posY = i * FIELD_W;
    for (let j = 0; j < FIELD_W; ++j) {
      this.effects[posY + j] = {type: 0}
    }
  }
}

#checkOnEnd() {
  const yPos = TOP_LINE * FIELD_W;
  for (let j = yPos + 1; j < yPos + FIELD_W - 1; j++)
  {
    if (this.field[j].color !== 0) {
      return true;
    }
  }
  return false;
}

#placeFigure(fig: Figure) {
  const x = fig.x;
  const y = fig.y;
  let figure = fig.value;
  const color = fig.color;
  const type = fig.type;

  for (let i = y; i < 4 + y; i++) { // 4 is fig w and h
    const yPos = i * FIELD_W;
    for (let j = x; j < 4 + x; j++) {
      if ((figure & 0x8000) !== 0) {
        this.#mergeBlock({
          color: color,
          type: type
          // TODO process types
        }, this.field[yPos + j])
      }
      figure <<= 1;
    }
  }
}

```

```

#collideFigure(fig: Figure) {
  const x = fig.x;
  const y = fig.y;
  let figure = fig.value;

  for (let i = y; i < 4 + y; i++) { // 4 is fig w and h
    if (i < 0) {
      continue;
    }
    const yPos = i * FIELD_W;
    for (let j = x; j < 4 + x; j++) {
      if ((figure & 0x8000) !== 0 && this.#checkBlockCollide(this.field[yPos + j],
fig.type))
        return true;
      figure <<= 1;
    }
  }
  return false;
}

#checkOnLine() {
  let scoreBenefit = 0;
  for (let i = 0; i < FIELD_H - 1; i++) {
    let lineChecked = true;
    const yPos = i * FIELD_W;
    let j = yPos + 1;
    while (lineChecked && j < yPos + FIELD_W - 1) {
      lineChecked = this.field[j].color !== 0;
      j++;
    }
    if (lineChecked) {
      this.#removeLine(i);
      scoreBenefit++;
    }
  }
  if (scoreBenefit > 0) {
    scoreBenefit = ((2 << scoreBenefit) - 1) << 2;
    this.score += scoreBenefit;
    this.scoreUpdateCallback && this.scoreUpdateCallback(this.score, scoreBenefit)
  }
}

#removeLine(lineNumber) {
  for (let j = 1; j < FIELD_W - 1; j++) {
    this.effects[lineNumber * FIELD_W + j].type = 1 // line clear
  }
  for (let y = lineNumber; y > 0; y--) {
    const yPos2 = (y - 1) * FIELD_W;
    const yPos1 = y * FIELD_W;
    for (let j = 1; j < FIELD_W - 1; j++) {
      this.field[yPos1 + j] = this.field[yPos2 + j];
    }
  }
}

#endGame() {
  //
  const newRecord = this.highScore < this.score;
  this.gameOverCallback && this.gameOverCallback(this.score, newRecord);
  if (newRecord) {
    this.highScore = this.score;
  }
  //
  this.playing = false;
}

```

```

#initialize(seed: number = null) {
  this.#initField();
  this.#initEffects();
  // set game data
  this.score = 0;

  this.playing = true;
  this.paused = true;
  this.softDrop = false;
  this.placed = 0;
  //
  this.timePlayed = 0;
  // generate new random seed
  this.random = new Random(seed || Math.floor(Math.random() * RANDOM_MAX));

  // set fig data
  this.nextFigures = Figure.genSequence(this.random);
  this.nextFigureNumber = 0;
  this.figPreviewY = FIG_START_Y;
  this.#nextFigure();
  // reset game speed
  this.tickSpeed = START_TICK_SPEED;
  // reset hold feature
  this.held = false;
  this.heldFigure = null;
}

#processFieldEffects() {
  let changes = false
  for (let y = FIELD_H - 1; y >= 0; --y) {
    let posY = y * FIELD_W;
    for (let x = FIELD_W - 1; x >= 0; --x) {
      const block = this.field[posY + x]
      switch (FigureType.at(block.type)) {
        case 'liquid': {
          // check left-right-down neighbours
          if (y >= FIELD_H - 1) {
            this.#killBlock(block)
          } else {
            // first check down
            let tmp = this.field[posY + x + FIELD_W]
            if (!this.#moveBlock(block, tmp)) {
              // check left right
              let rng = this.random.nextRandInt(0, 1);
              let l = (x > 1) ? this.field[posY + x + FIELD_W - 1] : null
              let r = (x < FIELD_W - 1) ? this.field[posY + x + FIELD_W + 1]
              : null

              if (r && l) {
                if (rng) {
                  if (this.#moveBlock(block, r)) {
                    changes = true;
                    break
                  }
                }
              } else {
                if (this.#moveBlock(block, l)) {
                  changes = true;
                  break
                }
              } else {
                l = null
              }
            }
          }
        }
      }
    }
  }
  if (l) {
    changes = this.#moveBlock(block, l)
  } else if (r) {
    changes = this.#moveBlock(block, r)
  }
}

```

```

        } else {
            changes = true
        }
    }

    } break;
    case 'sand': {
        if (y >= FIELD_H - 1) {
            this.#killBlock(block)
        } else {
            changes = this.#moveBlock(block, this.field[posY + x + FIELD_W])
        }
    } break;
    case 'tnt': {
        changes = true;
        // explode
        this.#explodeBlock(this.field[posY + x], true)
        this.effects[posY + x].type = 2
        this.#explodeBlock(this.field[posY + x + 1])
        this.#explodeBlock(this.field[posY + x - 1])
        this.#explodeBlock(this.field[posY + x - FIELD_W])
        this.#explodeBlock(this.field[posY + x + FIELD_W])
    } break;
    }
}

if (!changes) {
    this.#checkOnLine()
}

}

#checkBlockCollide(block: BlockType, type: number) {
    return (block.type !== FigureGhostId && type !== FigureGhostId && block.color !== 0)
|| block.type === FigureType.from('border');
}

#canMoveBlock(from: BlockType, to: BlockType) {
}

#moveBlock(from: BlockType, to: BlockType) {
    if (!to.color) {
        Object.assign(to, from)
        this.#nullifyBlock(from)
        return true;
    }
    else if (to.type === FigureGhostId) {
        to.color = mergeColors(to.color, from.color)
        to.type = from.type !== FigureGhostId ? from.type : 0
        this.#nullifyBlock(from)
        return true
    }
    else if (from.type === FigureGhostId) {
        to.color = mergeColors(to.color, from.color)
        this.#nullifyBlock(from)
        return true
    }
    else if (to.type === FigureType.from('liquid') && from.type !==
FigureType.from('liquid')) {
        // exchange
        const temp = Object.assign({}, to);
        Object.assign(to, from)
        Object.assign(from, temp)
        return true
    }
    return false
}
}

```

```

#mergeBlock(from: BlockType, to: BlockType) {
    // to.color defines if it keeps ghost prop
    from.color = mergeColors(to.color, from.color)
    if (from.type === FigureGhostId /* && to.color */) {
        from.type = to.type !== FigureGhostId ? to.type : 0
    }
    Object.assign(to, from)
}

#nullifyBlock(block: BlockType) {
    if (!block) return
    block.color = 0
    block.type = 0
}

#explodeBlock (block: BlockType, self: boolean = false) {
    if (!block || block.type === FigureType.from('border') || (block.type ===
FigureType.from('tnt') && !self)) return
    this.#nullifyBlock(block)
}

#killBlock(block: BlockType) {
    this.#nullifyBlock(block)
}
}

```

Обозначение					Наименование		Дополнительные сведения		
					<u>Текстовые документы</u>				
БГУИР ДП 1–40 01 01 047 ПЗ					Пояснительная записка		112 с.		
					Отзыв руководителя				
					Рецензия				
					<u>Графические документы</u>				
ГУИР.051006-01 СА					Обработка событий игровой		Формат А1		
					комнаты.				
					Схема алгоритма				
ГУИР.051006-02 СА					Взаимодействие клиента и		Формат А1		
					сервера.				
					Схема алгоритма				
ГУИР.051006-03 СА					Обработка событий игрового поля.		Формат А1		
					Схема алгоритма				
ГУИР.051006-01 ПЛ					Исследование индустрии		Формат А1		
					казуальных игр.				
					Плакат				
ГУИР.051006-02 ПЛ					Диаграмма вариантов		Формат А1		
					использования сервиса.				
					Плакат				
ГУИР.051006-03 ПЛ					Пользовательский интерфейс.		Формат А1		
					Плакат				