

ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ИРКУТСКОЙ ОБЛАСТИ
«АНГАРСКИЙ ТЕХНИКУМ СТРОИТЕЛЬНЫХ ТЕХНОЛОГИЙ»

Курсовая работа

Тема: Разработка браузерной игры "Морской бой"

Разработал _____ Биктобиров.С.П.

Руководитель _____ Денисюк А.В.

Ангарск, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
I. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ	6
1.1. Анализ предметной области	6
1.2. Игровая часть.....	17
1.3. Выбор программного обеспечения	18
1.3.1. Система управления базами данных	18
1.3.2. Клиентское приложение	19
1.3.3. Серверная часть приложения.....	21
1.4. Техническое задание.....	22
1.4.1. Общее описание функционала продукта	22
1.4.2. Требования к программной среде	22
1.4.3. Требования к аппаратной среде.....	22
1.4.4. Системные требования для пользователя.....	22
1.4.5. План разработки	23
1.5. Макет таблиц	23
1.6. Руководство по использованию приложения.....	23
1.6.1. Начало работы	23
1.6.2. Внесение новых пользователей в базу данных	24
II. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА	26
2.1. разработка веб-приложение	26
ЗАКЛЮЧЕНИЕ	41
СПИСОК ЛИТЕРАТУРЫ.....	42

						КР-09.02.07-Б-109-25ПЗ					
Изм.	Кол.уч	Лист	№ док.	Подпись	Дата						
Разраб.		Биктобиров.С.П				<u>Лит.</u>			Лист		
Пров.		Денисюк А.В.						3	42		
						ГАПОУ ИО АТСТ					
Н.контр											
Утв.											

ВВЕДЕНИЕ

Актуальность игры Тетрис

Игра Тетрис, созданная в 1984 году, остается актуальной и популярной даже спустя почти четыре десятилетия. Ниже представлены ключевые аспекты, которые подчеркивают важность и актуальность Тетриса в современном мире:

1. Вечная классика

Тетрис является одной из самых известных и признанных игр в истории, и его механика простая, но глубокая. Простота правил делает игру доступной для широкой аудитории, что привлекает как новых игроков, так и тех, кто играл в неё в детстве. Это создает эффект ностальгии и поддерживает интерес к игре на протяжении многих лет.

2. Постоянные инновации

Несмотря на свою давнюю историю, Тетрис продолжает эволюционировать. Существуют различные версии игры, включая мобильные приложения и многопользовательские режимы. Это позволяет адаптировать игру к современным технологиям и интересам игроков, привнося новые элементы, такие как 3D-графика, VR и AR (дополненная реальность).

3. Психологическая польза

Исследования показывают, что игра в Тетрис может иметь положительное влияние на психическое здоровье. Она способствует развитию когнитивных функций, таких как пространственное восприятие, реакция и принятие решений. Также игра может быть использована для уменьшения стресса и тревожности.

4. Образовательный потенциал

Тетрис может использоваться в образовательных целях для развития логического и стратегического мышления. Применение игры в учебных заведениях помогает научить студентов решать проблемы и развивать аналитические навыки.

5. Поддержка киберспорта

Тетрис стал частью киберспортивной культуры, с проведением турниров и соревнований. Это привлекает новых игроков и создает сообщество, которое активно участвует в обсуждениях и соревнованиях, что добавляет новые измерения к традиционному игровому процессу.

6. Доступность на разных платформах

Благодаря своей простой механике и не требовательным системным требованиям, Тетрис доступен на множестве платформ, включая мобильные устройства, ПК и консоли. Это делает игру доступной для самой широкой аудитории, включая людей всех возрастов и социальных групп.

7. Кросс-культурная интеграция

						КР-09.02.07-Б-109-25ПЗ	Лист
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		5

Тетрис является доброй "языковой игрой", которая легко воспринимается людьми разных культур и стран. Эта универсальность делает игру важным элементом в контексте глобализации и интеграции через развлечение.

Заключение

Актуальность игры Тетрис обусловлена её вечным статусом классики, способностью адаптироваться к современным технологиям и интересам, а также положительным воздействием на мозг и развитие навыков. С учётом всех этих факторов, Тетрис остаётся не только игрой, но и важным культурным феноменом, продолжая вдохновлять миллионы людей по всему миру.

Целью работы является разработка игры тетрис с возможностью игры как, одному так и игре 2 на одном ПК.

Для реализации этой цели были поставлены следующие задачи:

1. Изучить предметную область – как создаётся игра тетрис и что является её ключевыми отличиями от других игр;
2. Выбрать подходящий инструментарий и набор программного обеспечения для игры;
3. Разработать базу данных и создать игру с подключением к этой базе данных;

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

I. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1. Анализ предметной области

Как появился тетрис.

В 1984 году Алексей Пажитнов работал инженером-программистом Вычислительного центра при Академии наук СССР.

Будущий инженер родился в семье гуманитариев. Он происходил из интеллигентной творческой семьи, мать – кинокритик, отец – философ и литературовед. Родители приобщили Алексея к искусству, привили страсть к кино.

По словам Алексея, компьютерные игры позволяли преодолеть разрыв между логикой и эмоциями. Другие технологические достижения того времени были уж слишком алгоритмизированы.

На создание игры «Тетрис» его вдохновила головоломка «Пентамино» – популярный в советское время пазл, в которой фигуры из пяти квадратов нужно было собирать в большие формы.

На ее основе он создал «виртуальную» версию на языке Pascal. Количество квадратов сократилось до четырех, а готовые ряды исчезали. Название игра получила от слова «тетрамино» (тетра с греческого – четыре) плюс окончание от слова «теннис». Так получилось абстрактное и звучное название, которое сейчас известно во всем мире.

Потом Пажитнов с помощниками сделал вариант игры на РС, добавил цвета, таблицу результатов. Игра стала популярной в СССР и распространялась на 5,25-дюймовых дискетах, «из рук в руки», естественно, бесплатно. Постепенно она стала известна и в странах соцлагеря.

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

В Венгерском институте проблем кибернетики в 1986 году сотрудники портировали тетрис на компьютеры Commodore 64 и Apple 2. Там игру увидел англичанин Роберт Штайн и, оценив потенциал, предложил за нее Академии наук СССР 10 000 долларов сразу и 75% прибыли.

Тетрис начали готовить к запуску в США. Первые копии игры «из-за железного занавеса» должны были выйти в красной коробке с изображением собора Василия Блаженного, коммунистическими символами, под звук песни «Калинка-малинка».

Но бюрократическая машина в СССР была очень неповоротливой: каждая телеграмма требовала немало согласований, и переговоры шли долго. В какой-то момент в Москве спохватились, поняли скрытую мощь новой игры – и отказались передать права на условиях Штайна.

В 1987 году игра уже вышла в США фактически как пиратская версия. Она моментально стала популярной, получила множество наград. Новый всплеск произошел в 1989 году, когда "Тетрис" вышел на мобильном игровом устройстве - Game Boy.

Тогда же появилась шуточная легенда, что русский тетрис – разработка КГБ для развала американской экономики. В Америке все, от офисных клерков до высших чиновников, играли в тетрис вместо работы.

Тогда развернулся настоящий экшен – «битва за тетрис», в котором участвовали международные корпорации. За правами на игру охотились производители приставок и игровых автоматов. Тогда права принадлежали организации «Элорг» – одной из структур Академии наук СССР. Между компаниями Nintendo и Atari Games шли суды, и дело было окружено интригами и разбирательствами. Дошло даже до того, что медиамагнат

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

Роберт Максвелл, который был вхож в высшие политические круги СССР, звонил Горбачеву с требованием передать права его компании.

Позже тетрис стал одной из самых популярных игр всех времен. Только компания Nintendo заработала на нем по разным оценкам от 2 до 3 млрд долларов. К фантастическим тиражам тетриса пока не удалось никому приблизиться.

Правила Тетриса

В 2001 году The Tetris Company выпустила документ The Guideline (Руководство), в котором описаны правила игры, соблюдение которых обязательно для официальных тетрисов.

Мгновенное падение. В классическом тетрисе можно ускорить падение фигуры, зажав кнопку вниз. В гайдлайновом это называется софт-дропом (soft drop), потому что есть еще хард-дроп (hard drop), в котором фигура падает мгновенно. Это ускоряет игру, превращая её в зрелище.

Предпросмотр фигур. Для построения стратегии важно знать, какие фигуры идут дальше (next). Современные тетрисы показывают 5 фигур или меньше, в отличие от старых, которые показывали одну.

Закрепление фигур. В классическом тетрисе фигуры закреплялись сразу, как касались других фигур. В гайдлайновом это не так. Они замирают мгновенно только при хард-дропе.

Названия фигур. Они названы в честь букв, на которые похожи: I, J, L, O, S, T, Z.

Запас. Если нажать на кнопку, то текущая фигура отправится в запас (hold), а та фигура, что была там до неё, будет выдана вместо неё. Так можно отложить фигуру, которая сейчас мешается, до лучших времён.

Повороты. Для поворотов фигур используется набор правил, который называется супер-системой поворотов (Super Rotation System, SRS).

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

Крутить фигуры можно в обе стороны. Она позволяет различные неочевидные трюки, вроде ти-спинов.

Начисление очков. За очистку 4 линий за раз дают больше очков, чем за очистку 3. За ти-спин дают много очков. И всё такое.

Генератор фигур. В классическом тетрисе фигуры были в полностью случайном порядке. Решал не только навык игры, но и удача. В гайдлайновом алгоритме такой: все 7 фигур собираются в сумку, которая перемешивается и в этом порядке выдаётся игроку. Когда кончается сумка, генерируется новая. Такой генератор гарантирует, что нужная фигура точно придёт.

Мультиплеер. Главная фишка гайдлайнового тетриса. Когда срубаешь линии, оппоненту прилетает мусор, который он должен либо успеть отразить, либо этот мусор появится у него на поле. Вот пример дуэли (серое — мусор):

Побеждает игрок, который думает быстрее всего, умеет наносить большой урон, всё такое. Для мультиплеерного тетриса разработано много тактик. Моя любимая — center 4-wide well. Внимание на счётчик комбо (когда чистишь линии подряд):

Body-parser — это промежуточное ПО для Node.js, которое помогает обрабатывать тела входящих запросов в приложениях Express.

Некоторые особенности body-parser:

Поддержка разных типов данных. Body-parser работает с JSON, URL-кодированными данными, сырыми текстовыми данными и другими типами.

Упрощение доступа к данным. После разбора тело запроса привязывается к свойству req.body, что облегчает работу с данными.

Настройка ограничений по размеру. Можно установить лимиты на размер тела запроса, чтобы тяжёлые payload не перегружали сервер.

Автоматическое определение типа контента. Body-parser автоматически идентифицирует и обрабатывает разные типы контента в зависимости от заголовка Content-Type.

Оптимизация производительности. Эффективное управление операциями разбора снижает нагрузку на производительность и гарантирует отзывчивость программы даже при высокой нагрузке.

Body-parser часто используется в веб-приложениях, построенных с помощью Express.js.

Cookie-parser — пакет из npm, который упрощает анализ заголовков cookie и извлечение их значений. Его можно использовать как в Node.js, так и в браузере.

Некоторые особенности cookie-parser:

Поддержка подписанных cookie. Пакет обеспечивает дополнительный уровень безопасности, проверяя целостность cookie с помощью секретного ключа. [1](#)

Дополнительные функции. Cookie-parser предоставляет такие возможности, как истечение срока действия cookie, ограничение домена и безопасные флаги.

Возможность использования в качестве промежуточного ПО. Это позволяет лучше организовать логику обработки cookie и отделить её от других задач.

Некоторые параметры конфигурации cookie-parser, которые позволяют настроить его поведение:

secret — секретный ключ, используемый для подписания cookie.

signed — булево значение, которое указывает, нужно ли подписывать cookie. По умолчанию равно false.

maxAge — максимальный срок жизни cookie в миллисекундах. По умолчанию равен 0.

path — путь cookie. По умолчанию равен /.

						КР-09.02.07-Б-109-25ПЗ	Лист
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		5

domain — домен cookie. По умолчанию равен текущему домену.

secure — булево значение, которое указывает, нужно ли отправлять cookie только по протоколу HTTPS. По умолчанию равно false.

httpOnly — булево значение, которое указывает, нужно ли делать cookie доступным только через HTTP. По умолчанию равно false.

CORS (Cross-Origin Resource Sharing) — это механизм безопасности в веб-разработке, который позволяет или запрещает веб-браузерам делать запросы на серверы, находящиеся на другом домене.

Сторонним считается любой интернет-ресурс, который отличается от запрашиваемого протоколом, доменом или портом. Доступ предоставляется по специализированным запросам.

Зачем нужен CORS: в целях безопасности браузеры запрещают запросы к другим доменам, используя технику «одного источника» (same-origin policy). Это означает, что веб-страница, загруженная с одного домена, не может получить данные с другого домена без явного разрешения сервера. CORS позволяет серверам указать, с каких доменов разрешено получать данные, обеспечивая безопасность и контроль доступа.

Debug — программа-отладчик, которую используют для проверки и отладки выполняемых файлов. Также иногда так называют процесс отладки программы. [2](#)

Некоторые возможности Debug:

позволяет пошагово выполнять программу и следить за тем, что при этом происходит;

даёт средство обнаружения ошибок при работе с программой, транслированной в машинный язык;

позволяет изменять содержимое регистров и ячеек памяти;

даёт возможность просматривать команды программы с помощью дизассемблирования;

позволяет искать в памяти двоичные или ASCII-значения;

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

даёт возможность перемещать блок памяти из одного места в другое;
позволяет загружать и записывать файлы и сектора дисков.

Debug работала в операционных системах DOS, MS-DOS, OS/2, Microsoft Windows. В более поздние версии операционных систем программа работает через эмулятор DOS и имеет ограниченные возможности.

EJS (Embedded JavaScript Templating) — движок шаблонов для веб-разработки, используемый в Node.js. Он позволяет генерировать динамический контент путём встраивания JavaScript в HTML-контент.

Некоторые особенности EJS:

Минимальный синтаксис для написания логики шаблонов. Он похож на синтаксис Microsoft ASP (Active Server Pages).

Использование JavaScript в качестве языка шаблонов, поэтому разработчикам не нужно изучать новый язык для начала.

Кеширование промежуточных функций и более быстрое выполнение шаблонов на основе V8, базового движка JavaScript Node.js.

Возможность отладки с встроенными исключениями JavaScript.

Реализация функций, ориентированных на веб-разработку, таких как общие шаблоны, поддержка на стороне клиента и т. д..

Возможность хранения шаблонов в автономных файлах .ejs.

Работа с популярными веб-фреймворками, такими как Express и Fastify, без необходимости дополнительной настройки и продвинутой конфигурации.

Наличие полноценного CLI, который помогает разработчикам писать сценарии автоматизации для генерации HTML-документов с использованием шаблонов и файлов данных.

Express — веб-фреймворк и библиотека JavaScript для Node.js. Это свободное и открытое программное обеспечение, которое предназначено для создания веб-приложений и API.

Некоторые особенности Express:

Минималистичный дизайн. Простота позволяет быстро настраивать сервер, определять маршруты и эффективно обрабатывать HTTP-запросы. Гибкость и расширяемость. Можно легко интегрировать различные модули и плагины, чтобы расширить функциональность приложения. Высокая производительность. Фреймворк оптимизирован для работы с большим объёмом запросов и обеспечивает быструю обработку данных. Удобные инструменты для тестирования и отладки приложений. Это позволяет разработчикам быстро выявлять и устранять ошибки, что делает процесс разработки более продуктивным.

Express предоставляет такие механизмы, как написание обработчиков для запросов с различными HTTP-методами в разных URL-адресах, интеграция с механизмами рендеринга для генерации ответов и другие.

HTTP-ошибки — это трёхзначные числовые коды, которые возвращаются веб-сервером в ответ на запросы, отправленные клиентскими программами, такими как веб-браузеры. Коды ответов предоставляют информацию о том, как сервер обработал запрос и какой результат был получен.

Коды HTTP-ответов разделяются на пять основных классов:

Информационные (1xx). Эти коды информируют клиента о том, что сервер продолжает обработку запроса.

Успешные (2xx). Эти коды указывают на успешное выполнение запроса клиента.

Перенаправления (3xx). Эти коды указывают на то, что клиент должен выполнить дополнительные действия для завершения запроса.

Ошибки клиента (4xx). Эти коды указывают на ошибки, связанные с запросом, который сделал клиент.

Ошибки сервера (5xx). Эти коды указывают на ошибки, которые произошли на стороне сервера при обработке запроса.

Некоторые коды ошибок HTTP и их описание:

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

400 Bad Request. Запрос неправильный. Ошибка возникает в случае, если браузер клиента отправляет некорректный запрос серверу.

401 Unauthorized. Для получения запрашиваемого ответа нужна аутентификация.

402 Payment Required. Запрос не может быть выполнен, пока пользователь не произведёт оплату.

403 Forbidden. Запрет доступа к запрашиваемой странице. Он связан с тем, что у пользователя нет прав.

404 Not Found. Сервер даёт ответ, что страница не найдена по данному URL. Например, страница перенесена на другой адрес.

405 Method Not Allowed. Сервер сообщает, что используемый метод не может применяться на данном ресурсе.

406 Not Acceptable. Этот код ошибки указывает, что запрашиваемый контент не может быть распознан из-за кодировки, метода сжатия и других причин.

407 Proxy Authentication Required. Доступ будет открыт, если пройти авторизацию через сервер-посредник (прокси-сервер).

408 Request Timeout. Сервер хочет отключить соединение, так как обработка запроса пользователя вышла за рамки установленного времени.

409 Conflict. Запрос пользователя вызывает конфликт с текущим состоянием сервера или несовместим с другим запросом.

Morgan — промежуточное программное обеспечение для регистрации HTTP-запросов в Node.js. Оно упрощает процесс регистрации запросов за счёт автоматической генерации журналов для входящих запросов.

Некоторые особенности Morgan в коде:

Использование функции

`morgan(format, options)`

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

. Создаёт новую функцию регистратора с заданным форматом и параметрами. Аргумент формата может быть строкой с предопределённым именем, строкой с форматом или функцией, которая создаст запись в журнале. [1](#)

Использование функции

`morgan.compile(format)`

. Она компилирует строку формата в функцию формата для использования в Morgan. Строка формата представляет собой одну строку журнала и может использовать синтаксис токенов.

Работа с токенами. Токен — это базовый placeholder, который можно использовать в строке формата промежуточного программного обеспечения. Некоторые токены, которые предоставляет Morgan: пользовательский агент клиента, запрошенный URL и время ответа.

Morgan предлагает различные предопределённые форматы ведения журнала, такие как комбинированный, общий, dev, короткий и крошечный. Эти форматы определяют структуру и содержание генерируемых журналов. Разработчики также могут создавать собственные форматы ведения журнала, адаптированные к их конкретным потребностям.

MySQL2 — это быстрый, эффективный и многофункциональный клиент MySQL для Node.js. Он построен на основе драйвера MySQL для Node.js и предоставляет высокопроизводительный, неблокирующий и масштабируемый интерфейс для работы с базами данных MySQL в приложениях на Node.js.

С помощью MySQL2 можно выполнять такие задачи, как запросы к базам данных, вставка записей, обновление данных и обработка транзакций.

Некоторые преимущества использования MySQL2:

Улучшенная производительность. Благодаря использованию неблокирующих асинхронных операций приложение на Node.js может

продолжать выполнять другие задачи, ожидая завершения запросов к базе данных, что приводит к более быстрому ответу.

Повышенная безопасность. MySQL2 предоставляет такие функции, как подготовленные заявления и параметризованные запросы, чтобы помочь предотвратить атаки SQL-инъекций, обеспечивая безопасность данных.

Поддержка Promises. MySQL2 поддерживает Promises, что позволяет работать с более элегантным и читаемым асинхронным кодом, делая приложение более поддерживаемым.

Поддержка потоков. MySQL2 позволяет обрабатывать большие наборы данных напрямую из базы данных, уменьшая потребление памяти и улучшая общую производительность при работе с большими объёмами данных.

Управление несколькими соединениями. С помощью MySQL2 можно легко управлять несколькими соединениями с базами данных, что делает его подходящим для приложений, требующих одновременного доступа к базам данных.

MySQL2 можно устанавливать на Linux, Mac OS или Windows без каких-либо проблем.

JavaScript (js) — это язык программирования, который в первую очередь применяют в вебе. С его помощью сайты делают интерактивными: добавляют всплывающие окна, анимацию, кнопки лайков и формы для отправки информации. Его ещё называют главным языком фронтенда — «лицевой» стороны сайта, с которой взаимодействуют пользователи.

HTML (HyperText Markup Language) — это язык гипертекстовой разметки, который используется для создания и структурирования веб-страниц.

Он помогает определять, как содержимое страницы должно отображаться в браузерах. Иными словами, HTML — это своеобразный

каркас, на котором строится страница, включая текст, изображения, ссылки и другие элементы.

Основная цель HTML — структурировать и оформлять контент на сайте. HTML создаёт иерархическую структуру веб-страницы, используя заголовки, абзацы, списки и таблицы. Такая структура помогает пользователю легче ориентироваться на сайте.

С помощью HTML можно:

- делать текстовую разметку — форматировать текст, выделять фрагменты, создавать списки, добавлять сноски;
- встраивать медиа — размещать на сайте изображения, аудио, видео, карты;
- создавать ссылки и навигацию — гиперссылки и списки меню помогают быстрее найти информацию и сориентироваться на странице;
- создавать таблицы — нередко информацию удобно представить в табличном виде;
- создавать формы — формы нужны для регистрации посетителей сайта по телефону и электронной почте, оформления заказов, опросов и сбора обратной связи — отзывов, комментариев, предложений.

1.2. Игровая часть

Фигуры:

В Тетрис используются семь различных фигур, известных как тетромино.

Управление

Игрок управляет падающими тетромино с помощью клавиш для перемещения влево, вправо и или поворота фигуры. Основная задача — установить фигуры так, чтобы они заполнили горизонтальные линии.

3. Очки и урони

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

Каждое раз очищение линии с помощью полной строки дает игроку определенное количество очков. Игра ускоряется с увеличением уровня, что делает её более сложной.

Психология игры

Тетрис также интересен с точки зрения психологии. Исследования показывают, что игра может оказывать позитивное влияние на умственные процессы, такие как пространственное восприятие и способность к планированию. Игроки часто сообщают о состоянии "потока", когда они полностью поглощены игрой и теряют чувство времени.

1.3. Выбор программного обеспечения

1.3.1. Система управления базами данных

Для сохранения данных пользователя на нужно выбрать программное обеспечение для создания базы данных.

В качестве системы управления базы данных было выбрано открытое программное обеспечение MariaDB. Это реляционная СУБД, которая является ответвлением от похожего продукта MySQL с некоторыми модификациями, в основном внутренними, для улучшения эффективности хранения данных и оптимизации скорости взаимодействия с ними. Помимо этого, одно из основных отличий MariaDB от MySQL – открытая лицензия, благодаря которой сообщество смогло сделать множество дополнений для СУБД, например, множество «бэкендов» – подсистем хранения данных, которые отвечают за механизмы хранения данных и организацию конкурентного доступа к ним.

Использование различных подсистем может пригодиться в некоторых специфических случаях, когда данные хранятся необычным для СУБД способом. Например, в MariaDB существует подсистема «ColumnStore», которая позволяет хранить записи в таблице не построчно, как обычно, а в формате колонок, что позволяет хранить огромное количество записей, которые, как правило, не будут изменяться, при этом доступ к ним и поиск

по ним будет осуществляться намного быстрее, что полезно, к примеру, для хранения аналитических данных.

В нашем случае, скорее всего, никакие специфические функции MariaDB не пригодятся, но открытая лицензия позволит использовать эту СУБД бесплатно как в личных, так и в коммерческих целях.

1.3.2. Клиентское приложение

При разработке клиентского приложения для игры Тетрис было решено использовать Node.js как платформу для серверной части. Выбор Node.js обусловлен его асинхронной природой и возможностью обработки большого количества соединений одновременно, что идеально подходит для игр в реальном времени.

Архитектура приложения

В отличие от традиционных архитектур, работа с Node.js позволяет создать приложение, следуя принципам микросервисов и асинхронного взаимодействия. Это дает возможность более гибко управлять логикой игры и клиентскими запросами.

Серверная часть на Node.js

Сервер будет реализован на базе фреймворка Express.js. Пользователи смогут отправлять запросы на сервер для получения данных о текущем состоянии игры, а также для совершения действий, таких как вращение фигур и перемещение.

При открытии веб-приложения браузер отправляет запросы к серверу Node.js, который будет взаимодействовать с базой данных для хранения результатов игр и пользовательских профилей.

Игра будет реализована в режиме реального времени, что подразумевает минимальные задержки при передаче данных между клиентом и сервером. Для этого потребуется создать манифест веб-приложения, который будет содержать:

- Название игры (например, "Tetris")

- Стартовый URL для загрузки игры;
- Языки программирования и технологии
- Серверная часть будут разработаны на следующих технологиях:
- Серверная часть: Node.js, Express.js, (для реального взаимодействия).
- Хранилище данных: База данных (например, MySQL) для хранения профилей пользователей и результатов игр.

Заключение

Использование Node.js в разработке клиентского приложения для игры Тетрис позволит создать эффективное, производительное и удобное для пользователя игровое приложение.

Node.js — это платформа с открытым исходным кодом для работы с языком JavaScript, построенная на движке Chrome V8. Она позволяет писать серверный код для веб-приложений и динамических веб-страниц, а также программ командной строки.

1. Некоторые особенности Node.js:

- Единый язык программирования. Платформа позволяет использовать JavaScript как на клиенте, так и на сервере, что упрощает разработку и синхронизацию кода.
- Скорость и производительность. Благодаря движку V8 и событийному циклу платформа способна обрабатывать большое количество запросов с высокой скоростью и низким потреблением ресурсов.
- Богатая экосистема. Node.js имеет огромное количество модулей, фреймворков и инструментов, которые доступны через пакетный менеджер npm.
- Кроссплатформенность. Платформа работает на таких операционных системах, как Windows, Linux, macOS и даже на микроконтроллерах.

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

- Актуальность. Node.js поддерживает современные стандарты и возможности JavaScript, такие как ECMAScript modules, async/await, promises.

2. Некоторые области применения Node.js:

- Разработка приложений, которые работают в режиме реального времени и поддерживают постоянное взаимодействие между клиентом и сервером. Примеры таких приложений — чаты, игры, видео-стриминг, инструменты для коллаборации.
- Создание микросервисов. Архитектура среды позволяет разрабатывать небольшие изолированные сервисы, которые легко масштабировать и изменять.
- Разработка API. Node.js позволяет создавать RESTful API — интерфейсы для обмена данными между приложениями в формате JSON.
- Разработка приложений для интернета вещей. Node.js позволяет обрабатывать большие объёмы данных в реальном времени и взаимодействовать с разными протоколами и API.
- Разработка образовательных платформ и инструментов. Для помощи студентам и преподавателям в изучении и преподавании программирования.

1.3.3. Серверная часть приложения

При выборе программного обеспечения для создания сервера API был сделан упор на простоту прототипирования и разработки, был выбран Ejs так как он предоставляет разработчикам возможность быстро создавать прототипы благодаря множеству доступных библиотек и фреймворков, таких как Express.js, которые предлагают удобные инструменты для построения API. Разработчики могут сосредоточиться на логике приложения, а не на настройке инфраструктуры, что значительно ускоряет процесс разработки.

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

1.4. Техническое задание

1.4.1. Общее описание функционала продукта

Программный продукт будет выполнен в виде веб приложения с помощью Node.js игра имеет возможность играть как одному, так и вдвоём на одном ПК, данные пользователя сохраняются в базу данных возможность сдать и поставить игру на паузу, также есть отображение следующей фигуры.

1.4.2. Требования к программной среде

Готовый программный продукт должен работать на программной платформе, работающей под управлением операционной системы на базе ядра Windows, например Windows 10 и Windows 11, причем желательно выбирать именно выпуски с долгосрочной поддержкой (от англ. LTS – «long term support»), также не менее важно, что практически все пользователи используют Windows что позволит задеть максимальную аудиторию. Для корректного функционирования серверной части приложения потребуется установленная программная платформа Node.js версии 16.

1.4.3. Требования к аппаратной среде

Поскольку основой программной среды проекта является операционная система Windows, то есть требование как архитектура AMD x86-64

1.4.4. Системные требования для пользователя

Все взаимодействие пользователей и сотрудников с программой будет осуществляться через веб-приложение, поэтому единственное требование к пользователю веб-приложения – установленный веб-браузер, поддерживающий установку прогрессивных веб-приложений либо на основе движка Chromium 39 версии и выше (выпуск от ноября 2014 года) такие как Google Chrome и Microsoft Edge Chromium, либо браузер Safari. Стоит заметить, что браузеры на основе

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

1.4.5. План разработки

Разработка программного продукта планируется в срок до 6 месяца.

Процесс разработки будет включать в себя:

- Развертывание тестового стенда с СУБД MariaDB для проектирования базы данных;
- Разработку дизайна интерфейса с помощью CSS;
- Разработку клиентского веб-приложения;
- Тестирование программного продукта на локальном компьютере;
- Настройку облачной серверной машины под управлением операционной системы Windows;
- Развертывание проекта на сервере.

1.5. Макет таблиц

Теперь, когда было определено, на основе какого программного обеспечения будет разработана база данных и известно, какие существуют ограничения, можно определиться с физической структурой данных. Ниже приведена структура таблиц базы данных в том виде, в котором она будет объявлена в системе управления базами данных.

Таблица 1. Структура таблицы «Сотрудники»

Название поля	Тип данных	Пример данных
ID	Счетчик	-
Имя (first_name)	Строка	«Владимир»; «Иван»; «Николай»
Пароль	Счетчик	1234
Счёт игрока 1	Счетчик	199
Счёт игрока 2	Счетчик	200

1.6. Руководство по использованию приложения

1.6.1. Начало работы

						КР-09.02.07-Б-109-25ПЗ	Лист
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		5

Для начала работы с веб-приложением для администрирования необходимо вести в веб-браузере на базе Chromium (напр. Google Chrome, Microsoft Edge) название сервера после зарегистрироваться и выбрать желаемый режим игры после чего наслаждаться процессом.

1.6.2. Внесение новых пользователей в базу данных

Для того, чтобы добавить нового пользователя в базу данных, необходимо на главном экране приложения ввести свой данные после чего выбрать режим игры.

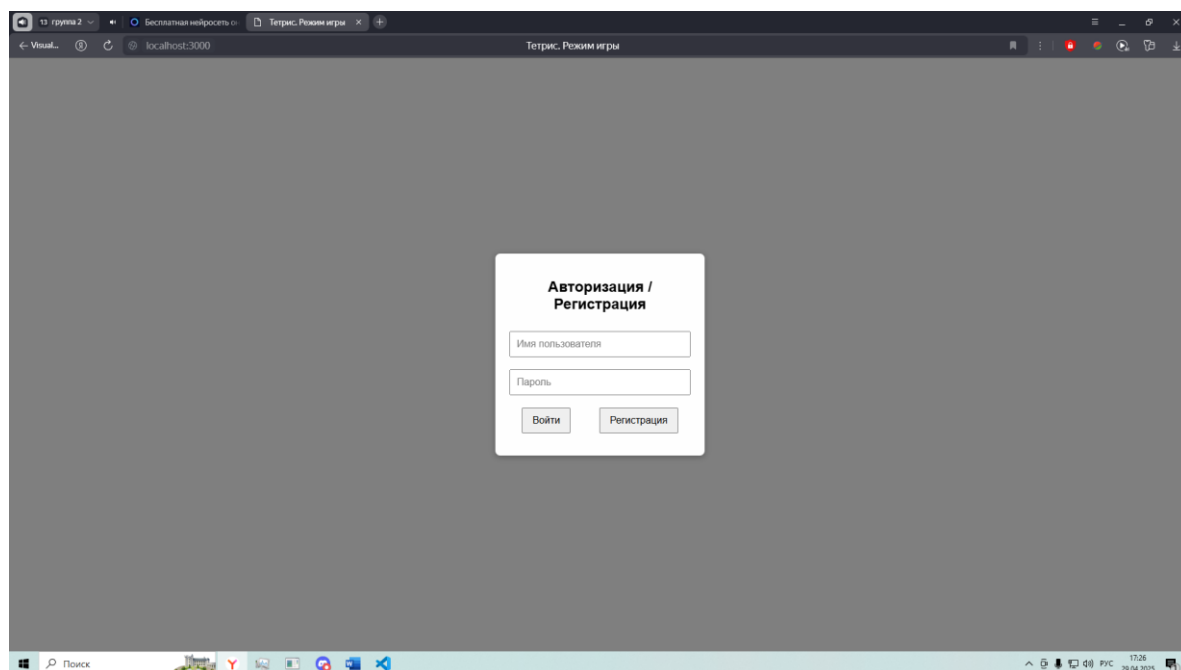


Рис. 1. Авторизация

Затем, выбираем нужный на режим игры:

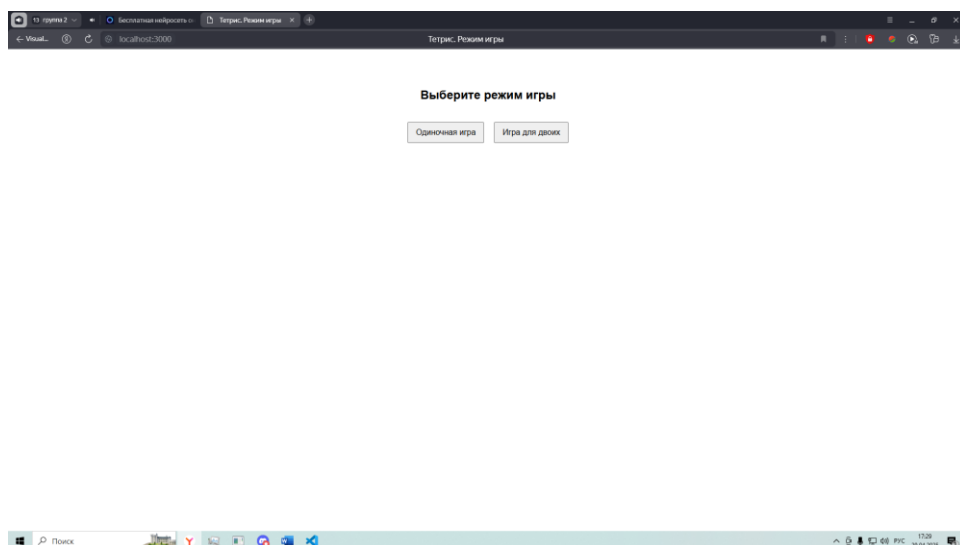
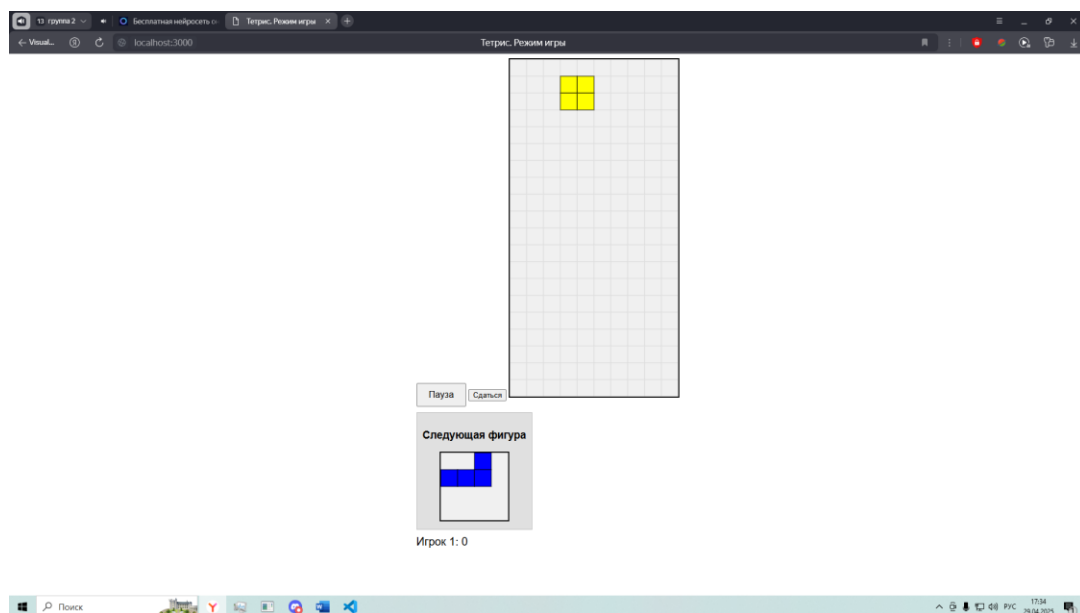
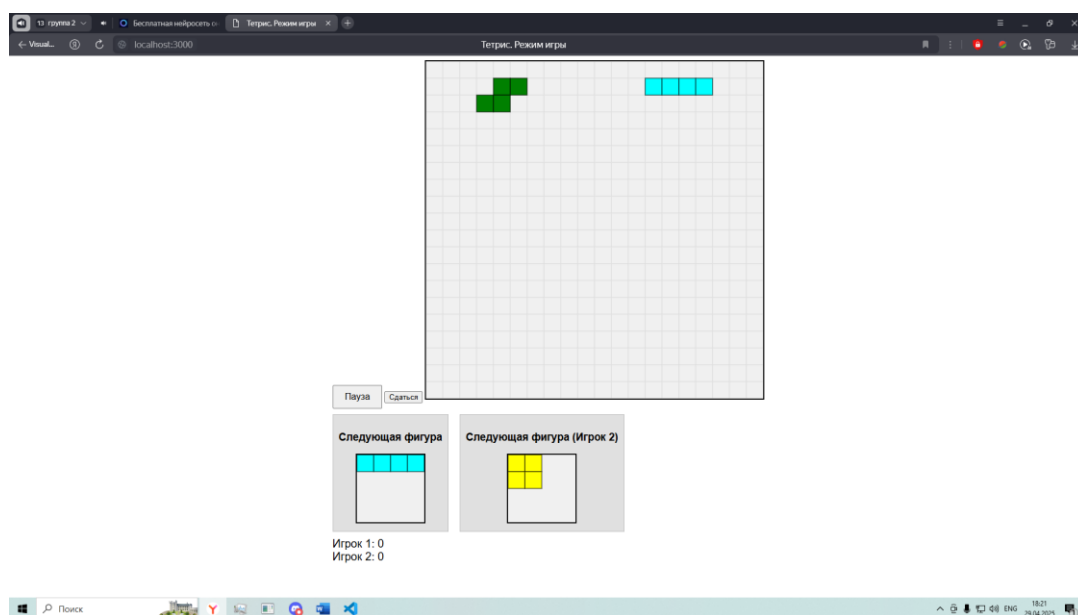


Рис. 2. Выбор режима игры

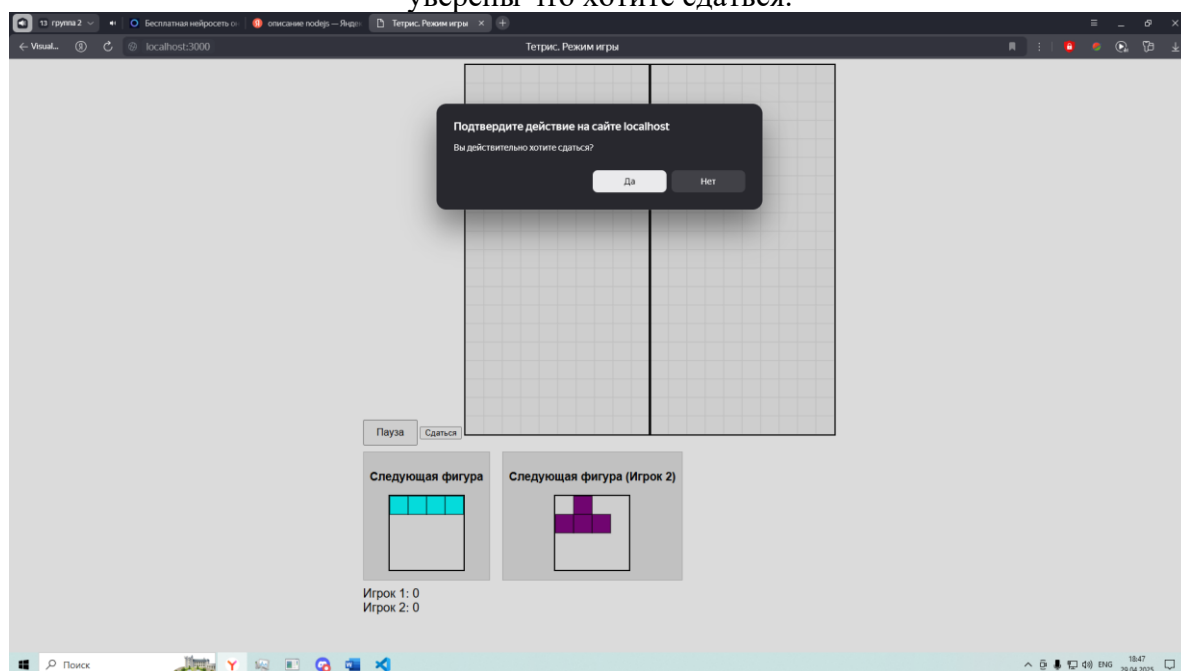
После выбора режима игры на одного



либо на двоих создастся поле игры, где вы уже начнёте играть



Также присутствует кнопка сдаться после нажатия на которую у вас спрашивают вы уверены что хотите сдаться.



II. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

Прежде чем приступать к разработке, стоит отметить, что необходимо иметь установленную программную платформу Node.js версии 16 или выше, а также какую-либо среду разработки, например, Visual Studio Code (используется в рисунках ниже).

2.1. разработка веб-приложение

Чтобы приступить к работе с кодом нужно сгенерировать шаблон с помощью команды `npm`. Для этого нужно открыть командную строку (терминал) в любой доступной папке и использовать следующую команду

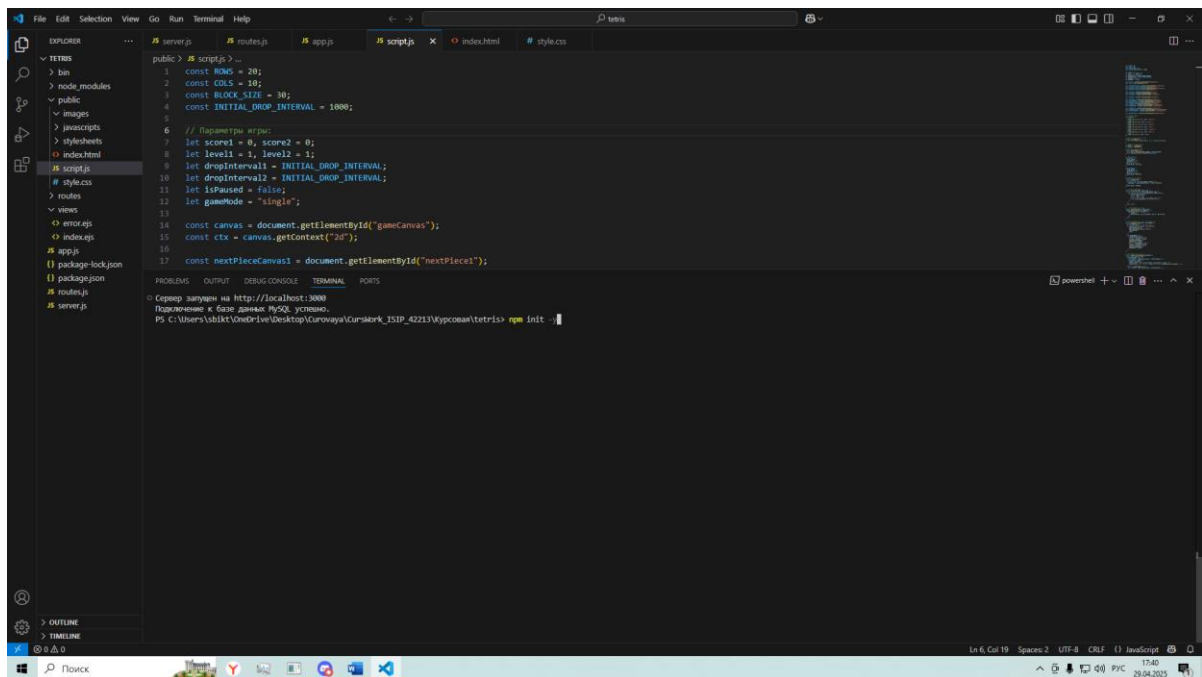


Рис. 13. Генерация шаблона

При нажатии на Enter он начнет автоматически генерировать шаблон проекта, не требуя от пользователя вводить все необходимые данные.

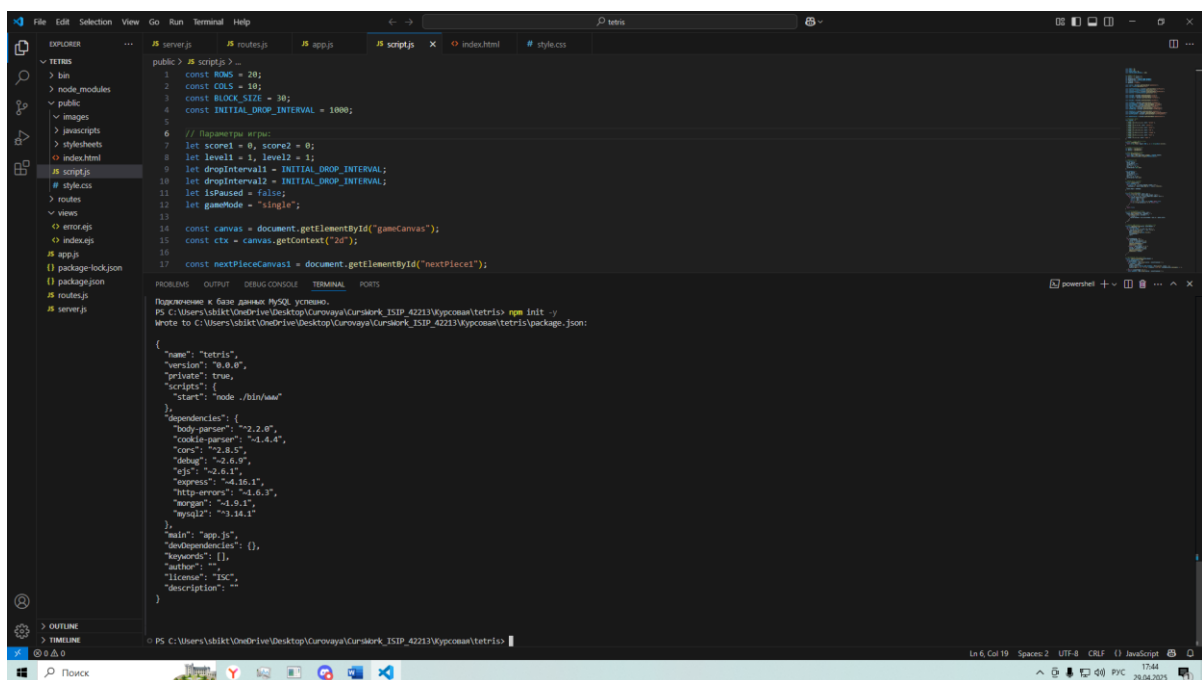


Рис. 14.

Затем нужно установить необходимые для нашего кода зависимости такие как express, core или body-parser:
express:

Это веб-фреймворк для Node.js, который упрощает процесс создания серверных приложений. С его помощью можно легко обрабатывать HTTP-запросы и организовывать маршрутизацию. В контексте вашей игры Тетрис, Express может быть использован для настройки сервера, который будет обрабатывать запросы от клиента, такие как запросы на обновление состояния игры или получение данных о результатах.

cors:

CORS (Cross-Origin Resource Sharing) - это механизм, позволяющий ограничить доступ к ресурсам веб-приложений для веб-сайтов из других доменов. Когда ваш сервер Express обрабатывает запросы из браузера (например, когда фронтенд игры взаимодействует с бэкендом), может возникнуть необходимость в разрешении этих запросов.

Пакет cors помогает настроить необходимые заголовки для этого, позволяя вашему клиенту (например, игровому клиенту) обращаться к серверу, даже если они находятся на разных доменах.

body-parser:

Этот пакет используется для разбора тела входящих запросов в форматах, таких как JSON или URL-кодированный. При разработке игры вам может понадобиться принимать данные от клиента, например, ход игрока или результаты игры. body-parser позволяет вашему серверу корректно обрабатывать эти данные и использовать их в логике приложения.

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

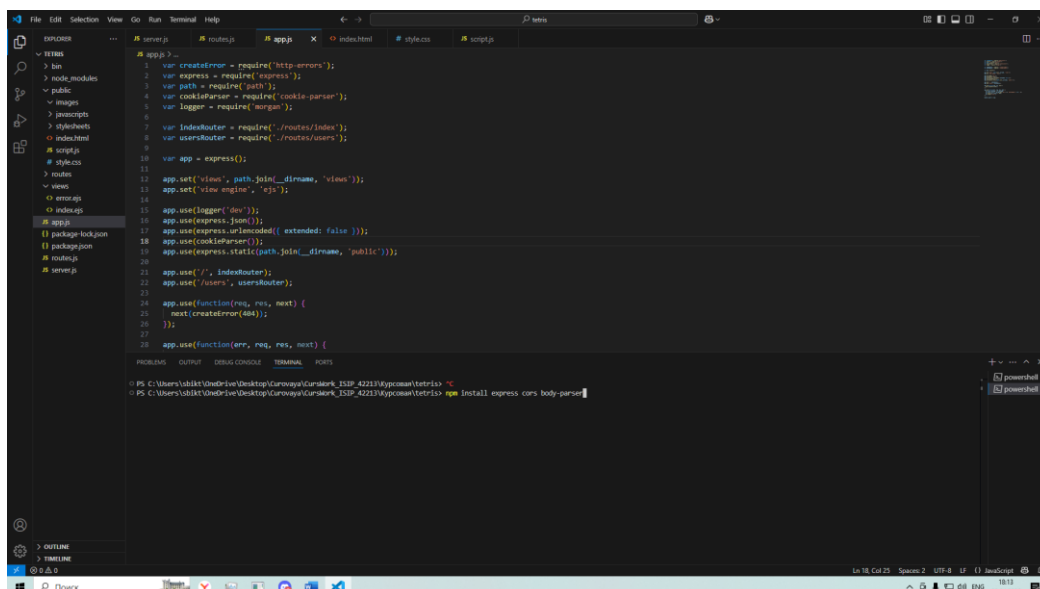


Рис. 15. Установка библиотек

Создаём html страницу и подключаем библиотеки и CSS также в body подключаем файл script.js

В секции <head> определены метатеги, которые обеспечивают корректное отображение на разных устройствах и устанавливают кодировку UTF-8 для поддержки мультиязычных символов. Также подключается внешний файл стилей style.css, который отвечает за внешний вид приложения. Название страницы указывается в теге <title>.

Далее идёт секция которая представляет собой окно авторизации/регистрации. Внутри этого окна находится заголовок, два поля ввода — одно для имени пользователя и одно для пароля, а также две кнопки для выполнения действий: входа в систему и регистрации.

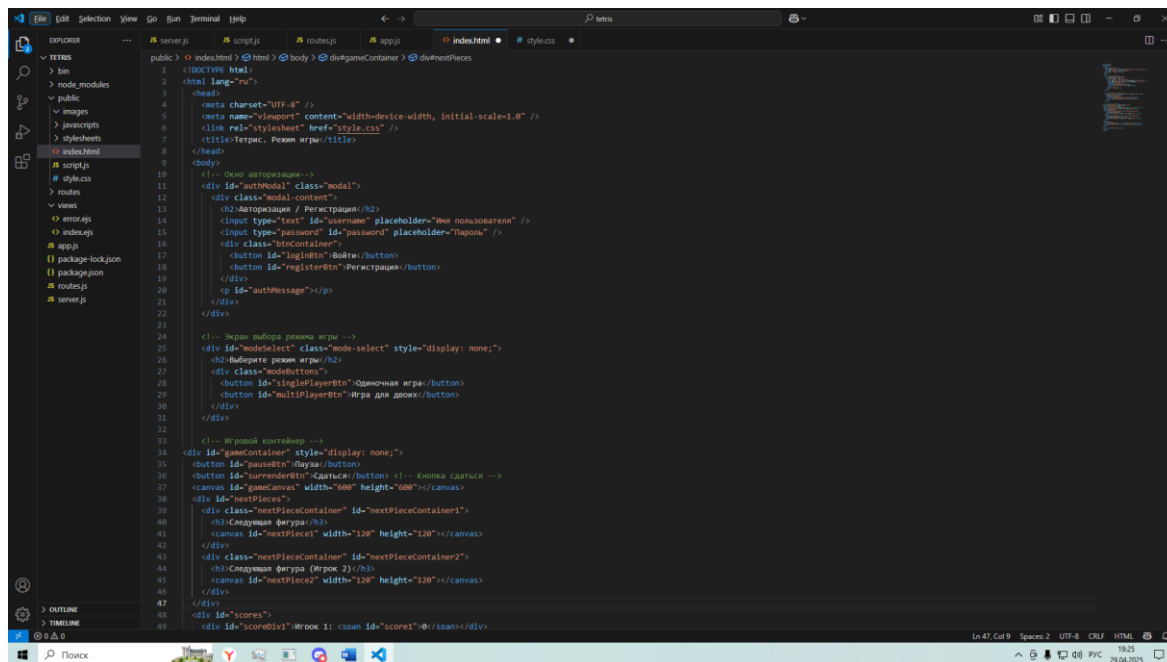
Потом идёт раздел который скрыт по умолчанию и будет отображаться после авторизации. Он позволяет игроку выбрать режим игры: одиночный или многопользовательский. Каждый режим представлен соответствующей кнопкой.

Затем идёт что раздел отвечает за сам игровой процесс. Он также скрыт изначально и отображается после выбора режима игры. Внутри контейнера для игры имеется кнопка "Пауза", основное игровое поле,

						КР-09.02.07-Б-109-25ПЗ	Лист
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		5

представленное с помощью элемента `<canvas>`, а также два контейнера для отображения следующей фигуры для каждого игрока. Также предусмотрены разделы для отображения текущего счета каждого из игроков.

В самом низу документа происходит подключение внешнего JavaScript-файла `script.js`, который будет отвечать за логику игры и взаимодействие с элементами интерфейса.



```
public > index.html > @html > @body > @div#gameContainer > @div#nextPieces
1 <!DOCTYPE html>
2 <html lang="ru">
3 <head>
4 <meta charset="UTF-8" />
5 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6 <link rel="stylesheet" href="style.css" />
7 <title>Тетрис. Проект №1</title>
8 </head>
9 <body>
10 <!-- Оверлей авторизации -->
11 <div id="authModal" class="modal">
12 <div class="modal-content">
13 <h2>Авторизация / Регистрация</h2>
14 <input type="text" id="username" placeholder="Ваш логин/имя" />
15 <input type="password" id="password" placeholder="Пароль" />
16 <div class="btnContainer">
17 <button id="loginBtn">Войти</button>
18 <button id="registerBtn">Регистрация</button>
19 </div>
20 <p id="authMessage"></p>
21 </div>
22 </div>
23
24 <!-- Выбор режима игры -->
25 <div id="modeSelect" class="mode-select" style="display: none;">
26 <h2>Выбор режима игры</h2>
27 <div class="modeButtons">
28 <button id="singlePlayerBtn">Одиночная игра</button>
29 <button id="multiPlayerBtn">Игра для двоих</button>
30 </div>
31 </div>
32
33 <!-- Игровой контейнер -->
34 <div id="gameContainer" style="display: none;">
35 <button id="pauseBtn">Пауза</button>
36 <button id="surrenderBtn">Сдаться</button> <!-- Кнопка сдачи -->
37 <canvas id="gameCanvas" width="600" height="600"></canvas>
38 <div id="nextPieces">
39 <div class="nextPieceContainer" id="nextPieceContainer1">
40 <div class="nextPiece" width="120" height="120"></div>
41 </div>
42 <div class="nextPieceContainer" id="nextPieceContainer2">
43 <div class="nextPiece" width="120" height="120"></div>
44 </div>
45 <div class="nextPiece" width="120" height="120"></div>
46 </div>
47 </div>
48 <div id="scores">
49 <div id="scoreDiv1">Игрок 1: <span id="score1">0</span></div>
```

В CSS мы настраиваем дизайн HTML страницы, кнопок, контейнеров и оставшегося интерфейса

```

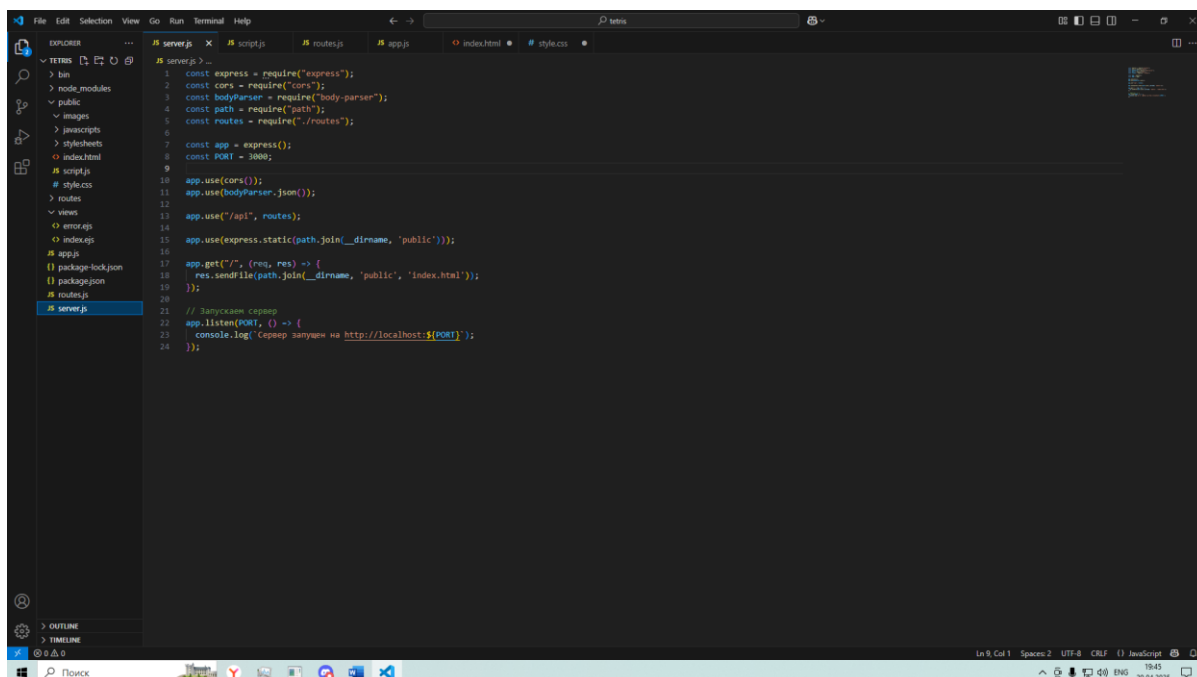
1 body {
2   display: flex;
3   flex-direction: column;
4   align-items: center;
5   font-family: sans-serif;
6 }
7
8 canvas {
9   border: 2px solid #000;
10  background-color: #f0f0f0;
11 }
12
13 #nextPlaces {
14   display: flex;
15   gap: 20px;
16   margin-top: 10px;
17 }
18
19 .nextPlaceContainer {
20   text-align: center;
21   background: #f0f0f0;
22   padding: 10px;
23   border: 1px solid #000;
24 }
25
26 #scores {
27   margin-top: 10px;
28   font-size: 20px;
29 }
30
31 #pauseBtn {
32   margin-top: 10px;
33   padding: 10px 20px;
34   font-size: 16px;
35 }
36
37 /* Скрытие модального окна авторизации */
38 .modal {
39   display: flex;
40   position: fixed;
41   z-index: 100;
42   left: 0;
43   top: 0;
44   width: 100%;
45   height: 100%;
46   overflow: auto;
47   background-color: rgba(0, 0, 0, 0.5);
48   align-items: center;
49   justify-content: center;

```

Файл server.js отвечает за создание сервера и выводит та нашу страницу html

Импорт необходимых модулей:

- Express: Импортируется фреймворк Express, который облегчает создание веб-приложений на Node.js.
- CORS: Импортируется модуль CORS, который позволяет управлять кросс-домашними запросами (Cross-Origin Resource Sharing).
- body-parser: Импортируется модуль body-parser, используемый для разбора тела запросов, чтобы работать с данными, отправляемыми в формате JSON.
- path: Импортируется встроенный модуль path, который содержит утилиты для работы с файловыми путями.
- routes: Импортируется локальный модуль routes, содержащий определения маршрутов API для приложения.



Наш файл routes.js отвечает за сохранение данных пользователей в базе MySQL и проверка зарегистрированных пользователей

```
const express = require("express");
```

```
const router = express.Router();
```

Здесь импортируется фреймворк Express, который облегчает создание веб-приложений на Node.js. Создается экземпляр маршрутизатора (router), который будет использоваться для определения маршрутов (endpoint'ов) приложения.

```
let users = [];
```

Объявляется пустой массив users, который будет использоваться для хранения объектов пользователей. Каждый объект пользователя будет содержать имя пользователя, пароль и два счёта (score1 и score2).

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

```

1 const express = require("express");
2 const router = express.Router();
3 const mysql = require("mysql");
4
5 // Подключение к базе данных
6 const db = mysql.createConnection({
7   host: "localhost",
8   user: "root",
9   password: "elena",
10  database: "tetris_game"
11 });
12
13 // Проверка подключения
14 db.connect(err => {
15   if (err) throw err;
16   console.log("Подключение к базе данных MySQL успешно.");
17 });
18
19 router.post("/register", (req, res) => {
20   const { username, password } = req.body;
21   if (!username || !password) {
22     return res.status(400).json({ success: false, message: "Введите логины и пароли." });
23   }
24
25   // Проверка существования пользователя
26   db.query("SELECT * FROM users WHERE username = ?", [username], (err, results) => {
27     if (err) throw err;
28
29     if (results.length > 0) {
30       return res.status(400).json({ success: false, message: "Пользователь уже существует." });
31     }
32
33     // Сохранение нового пользователя
34     db.query("INSERT INTO users (username, password) VALUES (?, ?)", [username, password], err => {
35       if (err) throw err;
36       res.json({ success: true, message: "Регистрация успешна." });
37     });
38   });
39 });
40
41 router.post("/login", (req, res) => {
42   const { username, password } = req.body;
43
44   // Поиск пользователя по имени и паролю
45   db.query("SELECT * FROM users WHERE username = ? AND password = ?", [username, password], (err, results) => {
46     if (err) throw err;
47
48     if (results.length === 0) {
49       return res.status(400).json({ success: false, message: "Неверный логин или пароль." });
50     }
51   });
52 });

```

В файле script.js заложена вся логика нашей игры

Сначала создаются правила игры

ROWS = 20: Это количество строк в игровом поле. В Тетрисе игра идет на прямоугольной сетке, где каждая строка соответствует ряду, который игрок должен заполнить.

COLS = 10: Это количество столбцов в игровом поле. Сочетание ROWS и COLS определяет размер игрового поля.

BLOCK_SIZE = 30: Это размер блока (или 'тета'), из которого состоят тетрамино. Устанавливает физический размер блоков, что влияет на визуальное представление игры.

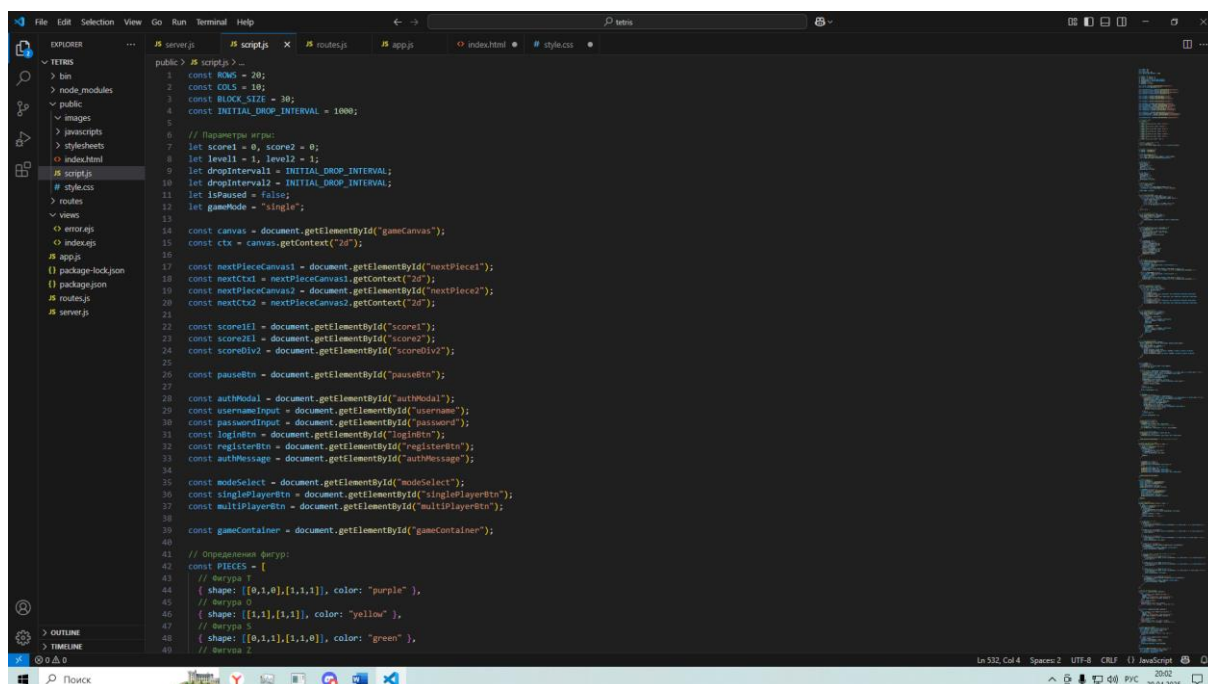
INITIAL_DROP_INTERVAL = 1000: Это начальный интервал времени (в миллисекундах), через который выпадают тетрамино. Чем меньше этот интервал, тем быстрее падают блоки, что увеличивает сложность игры.

score1 и score2:

Эти переменные хранят текущий счет для двух игроков. В зависимости от того, сколько рядов игрок смог очистить, их счет будет увеличиваться.

gameMode:

Строка, которая определяет режим игры. Значение "single" указывает на одиночный режим, но может быть и "multi" для многопользовательского режима.



1. Определение фигур

Фигуры в Тетрисе называются «тетромино» и представлены следующими стандартными формами:

I (прямая линия)

O (квадрат)

T (буква T)

S (зигзаг)

Z (обратный зигзаг)

J (обратная L-образная фигура)

L (L-образная фигура)

Каждая фигура состоит из 4 квадратных единиц, которые могут располагаться в различных позициях.

2. Определение цветовой схемы

Для каждого тетромино нужно выбрать уникальный цвет, чтобы игрок мог легко различать фигуры. Например:

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

I — голубой

O — желтый

T — фиолетовый

S — зелёный

Z — красный

J — синий

L — оранжевый

3. Создание графических представлений

Фигуры могут быть представлены в виде двумерного массива, где каждый элемент массива соответствует квадрату. Например, форма «Т» может выглядеть так:

```
[  
  [0, 1, 0],  
  [1, 1, 1],  
  [0, 0, 0]  
]
```

Здесь:

1 — это заполненная часть тетромино (цвет),

0 — пустое пространство.

4. Ротация тетромино

Важно добавить возможность ротации фигур. Это можно сделать путем преобразования двумерного массива. Например, для изменения ориентации «Т» можно реализовать функцию, которая будет поворачивать массив на 90 градусов по часовой стрелке.

5. Создание игровых правил

Реализуйте правила для размещения фигур на игровом поле. Необходимо учитывать:

Пересечение: Тетромино не должны перекрывать уже занятые клетки поля.

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

Границы поля: Фигуры не должны выходить за пределы игрового поля.

7. Анимация и движение

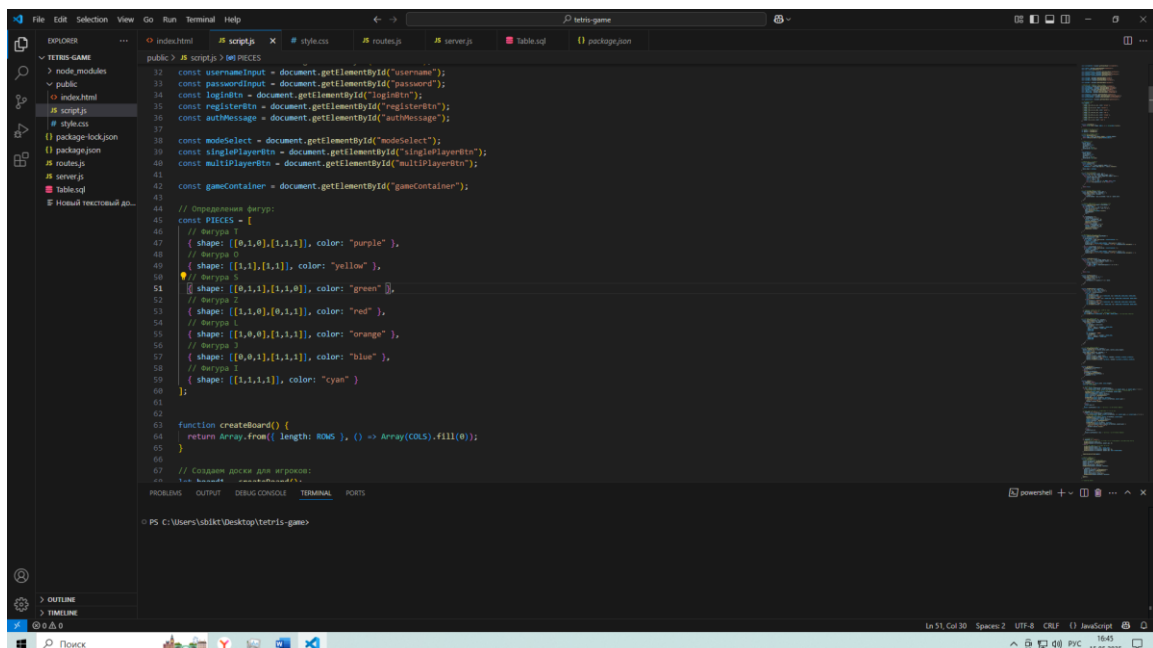
Реализация механики падения фигур, включая:

Скорость падения.

Управление пользователем (вправо, влево, вниз, поворот).

8. Обработка столкновений и линии

Когда тетромينو достигает нижнего предела или соприкасается с другой фигурой, необходимо проверить завершённые линии и удалить их, что даст игроку дополнительные очки.



Тут идёт функция пристыковки фигуры к доске и также подсчёт очков и удаление линий если она полностью заполнена также если закрываете 2 и более линий очки множатся что будет мотивирует игроков к закрыванию как можно больших линий

Создание игровой доски:

Функция createBoard создает двумерный массив (доску) заданного размера (ROWS x COLS), инициализируя каждую клетку значением 0. Это представляет пустую игровую доску, где 0 означает, что клетка свободна. Функция getRandomPiese генерирует случайную фигуру (т.е. тетромينو) из массива PIECES, используя случайный индекс. Клонирование с

							КР-09.02.07-Б-109-25ПЗ	Лист
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата			5

помощью `JSON.parse(JSON.stringify(...))` обеспечивает создание независимого экземпляра фигуры, чтобы изменения не повлияли на оригинал.

Тут создаются объекты `player1` и `player2`, хранящие информацию о каждом игроке. Каждый объект содержит:

```
const player1 = {  
  board: board1,  
  currentPiece: null,  
  nextPiece: null,  
  pos: { x: 3, y: 0 },  
  lastDropTime: Date.now()  
};  
  
const player2 = {  
  board: board2,  
  currentPiece: null,  
  nextPiece: null,  
  pos: { x: 3, y: 0 },  
  lastDropTime: Date.now()  
};
```

`board`: ссылка на игровую доску игрока.

`currentPiece`: текущая играемая фигура (в начале может быть `null`).

`nextPiece`: следующая фигура, которая будет использоваться (также может быть `null`).

`pos`: положение фигуры на доске (изначально фигура располагается в верхней части игры).

`lastDropTime`: время последнего падения фигуры, что может использоваться для управления скоростью игры.

Функция `rotate` поворачивает текущую фигуру (тетромино) на 90 градусов по часовой стрелке, меняя её форму в массиве.

						КР-09.02.07-Б-109-25ПЗ	Лист
							5
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		

Функция collision проверяет, есть ли столкновение текущей фигуры с границами доски или другими фигурами, находящимися на доске. Если фигура выходит за пределы или пересекает уже занятые клетки, функция возвращает true.

Функция placePiece фиксирует фигуру на доске, изменяя значения в доске так, чтобы они соответствовали цвету фигуры. Это делает занятые клетки видимыми.

```
function placePiece(board, piece, pos) {  
  piece.shape.forEach((row, rowIndex) => {  
    row.forEach((value, colIndex) => {  
      if (value) {  
        board[rowIndex + pos.y][colIndex + pos.x] = piece.color;  
      }  
    })  
  })  
}
```

Функция removeFullLines проверяет строки на заполненность (все клетки не равны 0). Если строка заполнена, она удаляется, и в начале добавляется новая пустая строка. После удаления строк обновляется счет игрока, вызывается функция для увеличения уровня и скорости игры, а также сохраняется новый счет.

```
function removeFullLines(board, playerNumber) {  
  let linesRemoved = 0;  
  for (let row = ROWS - 1; row >= 0; row--) {  
    if (board[row].every(cell => cell !== 0)) {  
      board.splice(row, 1);  
      board.unshift(Array(COLS).fill(0));  
      linesRemoved++;  
      row++;  
    }  
  }  
}
```

Функция updateLevelAndSpeed обновляет уровень игрока в зависимости от набранных очков. Уровень увеличивается каждые 100 очков, и скорость игры также увеличивается — время падения фигур сокращается.

```
function updateLevelAndSpeed(playerNumber) {
```

```
const levelThreshold = 100;
```

```
if (playerNumber === 1) {
```

Функция `canMoveDown` проверяет, может ли текущая фигура двигаться вниз. Если вниз есть граница доски или же другой игрок, функция возвращает `false`.

```
function canMoveDown(piece) {
```

```
  for (let y = 0; y < piece.shape.length; y++) {
```

```
    for (let x = 0; x < piece.shape[y].length; x++) {
```

```
      if (piece.shape[y][x] !== 0) {
```

```
        let newY = piece.y + y + 1;
```

```
        if (newY >= ROWS || board[newY][piece.x + x] !== 0) {
```

```
          return false;
```

Функция `fixPiece` помещает текущую фигуру на игровой доске. Если клетка угловой части фигуры не равна 0, она фиксирует её значением клетки (цветом).

```
function fixPiece(piece) {
```

```
  piece.shape.forEach((row, y) => {
```

```
    row.forEach((value, x) => {
```

```
      if (value !== 0) {
```

```
        board[piece.y + y][piece.x + x] = value;
```

Функция `drawBoard` отвечает за визуализацию игровой доски на экране.

Она перебирает каждую клетку доски и заполняет её цветом. Если клетка занята, она закрашивается цветом фигуры, если свободна — светло-серым цветом. Также добавляется разделительная линия между досками двух игроков, если включен многопользовательский режим.

```
function drawBoard(board, offsetX) {
```

```
  for (let row = 0; row < ROWS; row++) {
```

```
    for (let col = 0; col < COLS; col++) {
```

```
      const cell = board[row][col];
```

```

if (cell !== 0) {

    ctx.fillStyle = cell;

    ctx.fillRect(offsetX + col * BLOCK_SIZE, row * BLOCK_SIZE,
BLOCK_SIZE, BLOCK_SIZE);

    ctx.strokeStyle = "#000";

    ctx.strokeRect(offsetX + col * BLOCK_SIZE, row * BLOCK_SIZE,
BLOCK_SIZE, BLOCK_SIZE);

```

Тут выполняются функции для первого игрока, а также если играете вдвоём, то и для второго

```

234 function update() {
235     ctx.clearRect(0, canvas.width, canvas.height);
236     const now = Date.now();
237     // const now = Date.now();
238     if (now - player1.lastDropTime > dropInterval1) {
239         if (collision(player1.board, player1.currentPiece, { x: player1.pos.x, y: player1.pos.y + 1 })) {
240             placePiece(player1.board, player1.currentPiece, player1.pos);
241             removeFullLines(player1.board, 1);
242             player1.currentPiece = player1.nextPiece;
243             player1.nextPiece = getRandomPiece();
244             player1.pos = { x: 1, y: 0 };
245             drawNextPiece(player1.nextPiece, nextCtx1);
246             if (collision(player1.board, player1.currentPiece, player1.pos)) {
247                 alert("Враг оказался!");
248                 document.location.reload();
249             }
250         } else {
251             player1.pos.y++;
252             player1.lastDropTime = now;
253         }
254     }
255     // Враг 2
256     if (gameMode === "multi") {
257         if (now - player2.lastDropTime > dropInterval2) {
258             if (collision(player2.board, player2.currentPiece, { x: player2.pos.x, y: player2.pos.y + 1 })) {
259                 placePiece(player2.board, player2.currentPiece, player2.pos);
260                 removeFullLines(player2.board, 2);
261                 player2.currentPiece = player2.nextPiece;
262                 player2.nextPiece = getRandomPiece();
263                 player2.pos = { x: 3, y: 0 };
264                 drawNextPiece(player2.nextPiece, nextCtx2);
265                 if (collision(player2.board, player2.currentPiece, player2.pos)) {
266                     alert("Враг врага 2 оказался!");
267                     document.location.reload();
268                 }
269             } else {
270                 player2.pos.y++;
271                 player2.lastDropTime = now;
272             }
273         }
274     }
275     // Draw game board & pieces
276     drawBoard(player1.board, 0);
277     if (gameMode === "multi") {
278         drawBoard(player2.board, COLS * BLOCK_SIZE);
279     }
280 }

```

ЗАКЛЮЧЕНИЕ

В ходе разработки веб-приложения Тетрис, мы достигли ряда значительных результатов, которые подтверждают успешность реализации проекта. Основные этапы работы включали проектирование пользовательского интерфейса, реализацию логики игры, а также оптимизацию работы приложения для различных платформ и устройств.

Основные достижения:

1. Интуитивно понятный интерфейс. Были разработаны компоненты интерфейса, которые обеспечивают удобное взаимодействие пользователя с приложением, что значительно улучшает пользовательский опыт.

2. Эффективная игровая механика. Реализация алгоритмов для управления движением фигур и механикой игры позволила создать увлекательный и динамичный игровой процесс, который соответствует классическому Тетрису с добавленными улучшениями.

3. Возможности для дальнейшего развития. Несмотря на уже достигнутые результаты, в приложении заложены возможности для дальнейших улучшений, таких как добавление новых режимов игры, многопользовательского функционала и достижения системы.

Перспективы

В будущем, проект может быть расширен новыми функциями и уровнями сложности, что привлечет еще больше пользователей. Также, возможно, интеграция с социальными сетями для повышения вовлеченности игроков будет способствовать развитию сообщества.

В заключение, разработка веб-приложения Тетрис стала ценным опытом в изучении веб-технологий, работы с графикой и взаимодействия с пользователем. Я уверен, что результат моей работы удовлетворит пользователей и будет способствовать популяризации классической игры в цифровом формате.

						КР-09.02.07-Б-109-25ПЗ	Лист
Изм.	Кол.уч	Лист	№ докум.	Подпись	Дата		5

СПИСОК ЛИТЕРАТУРЫ

1. Это официальный сайт Node.js, где содержится полная документация по API и функциональности. Node.js. (n.d.). Node.js. Retrieved from <https://nodejs.org/en/docs/>
2. Документация Express содержит подробные объяснения, инструкции и примеры по работе с фреймворком. Express.js. (n.d.). Express - Node.js web application framework. Retrieved from <https://expressjs.com/>
3. Официальная документация MySQL, которая предоставляет полные сведения о работе с MySQL, SQL-запросами и управлении базами данных. Документация по MySQL:MySQL. (n.d.). MySQL Documentation. Retrieved from <https://dev.mysql.com/doc/>
4. Видеокурс, который охватывает основные аспекты работы с Node.js, Express, MongoDB, и созданием RESTful API. Онлайн-курс: "The Complete Node.js Developer Course":Andrew Mead. (2021). The Complete Node.js Developer Course (3rd Edition). Udemy. Retrieved from <https://www.udemy.com/course/the-complete-nodejs-developer-course-2/>
5. Онлайн-курс: "Building Web Applications in Node.js and Express":
6. John Dyer. (n.d.). Building Web Applications in Node.js and Express. Coursera. Retrieved from <https://www.coursera.org/learn/building-web-apps-nodejs-express>
7. Курс по созданию веб-приложений с использованием Node.js и Express. Подходит как для новичков, так и для опытных разработчиков.
8. Введение в Express.js для начинающих, с простыми примерами и объяснениями ключевых понятий. Blog Post: "A Beginner's Guide to Express.js":

9. Hossain, M. (2020). A Beginner's Guide to Express.js. Flaviocopes.com.
Retrieved from <https://flaviocopes.com/express/>
10. Платформа, где можно задавать вопросы и находить ответы по темам JavaScript, Node.js, Express и других технологий. Сообщество и обсуждения: Stack Overflow. (n.d.). Stack Overflow. Retrieved from <https://stackoverflow.com/>
11. Полуэктова, Н. Р. Разработка веб-приложений: учебное пособие для среднего профессионального образования / Н. Р. Полуэктова. — Москва: Издательство Юрайт, 2021. — 204 с. — (Профессиональное образование). — ISBN 978-5-534-14744-5. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/479863>;
12. Стружкин, Н. П. Базы данных: проектирование: учебник для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва: Издательство Юрайт, 2023. — 477 с. — (Профессиональное образование). — ISBN 978-5-534-11635-9. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/518499>;
13. Тузовский, А. Ф. Проектирование и разработка web-приложений: учебное пособие для среднего профессионального образования / А. Ф. Тузовский. — Москва: Издательство Юрайт, 2021. — 218 с. — (Профессиональное образование). — ISBN 978-5-534-10017-4. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/475437>;