Министерство образования Новосибирской области ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

Разработка веб-приложения для игры в шахматы  
с элементами обучения

Пояснительная записка

ПМ.02 Осуществление интеграции программных модулей

ПМ.04 Сопровождение и обслуживание программного обеспечения компьютерных систем

НАТКиГ.721400.010.000

Разработал:

студент группы ПР-21.101

Некрасов Я.А.

**Содержание**

[Введение 3](#_Toc198209320)

[1 Исследование предметной области 4](#_Toc198209321)

[1.1 Описание предметной области 4](#_Toc198209322)

[1.2 Описание групп пользователей и функциональных задач 7](#_Toc198209323)

[1.3 Анализ программных ресурсов необходимых в работе 9](#_Toc198209324)

[1.4 Анализ аналогов и прототипов 11](#_Toc198209325)

[2 Технологический раздел 14](#_Toc198209326)

[2.1 Структура базы данных 14](#_Toc198209327)

[2.2 Разработка структуры (схемы) сайта 14](#_Toc198209328)

[2.3 Разработка дизайн макета сайта 15](#_Toc198209329)

[2.4 Разработка алгоритмов обработки информации 17](#_Toc198209330)

[2.5 Разработка текстового содержания сайта 28](#_Toc198209331)

[2.6 Разработка мультимедийного контента 31](#_Toc198209332)

# Введение

Шахматы являются популярной темой во всем мире. С конца 20 века компьютеры значительно повлияли на шахматное сообщество. На шахматном сервисе «Lichess.org» происходит около 90 миллионов партий между пользователями сервиса ежемесячно. Создание шахматного приложения может повысить доступность этой игры для развития у пользователя логического мышления, навыков планирования.

Дипломный проект по созданию справочного приложения с элементами обучения позволит пользователю осуществлять игру в шахматы с искусственным противником и получать знания о названиях шахматных позиций для дальнейшего самостоятельного изучения.

Целью разработки является создание шахматного приложения с элементами обучения.

Задачами курсового проекта в связи с указанной целью являются:

* изучение предметной области;
* рассмотрение приложения с точки зрения пользователя для выявления функций приложения;
* разработка и реализация дизайна приложения;
* написание кода приложения;
* тестирование полученного продукта.

# Исследование предметной области

## Описание предметной области

Многим людям, имеющим свободное время приходила идея об отдыхе, и для исполнения данного желания имеется множество видов досуга, в число которых входит времяпровождение за настольными играми. Некоторые из настольных игр не только снимают стресс, но также развивают мышление.

Д. И. Бронштейн и Г. Л. Смолян выделяют четыре фактора, обуславливающие творческий характер шахмат:

* радость, которую получает шахматист, создавая художественные ценности самостоятельно;
* зритель, к которому обращены данные ценности. Без публики шахматисты не испытывают творческое напряжение, и их «остывшая душа оказывается в плену бесчисленных вариантов»;
* красота, для одних шахматистов и зрителей проявляющаяся в строгости, законченности и силе логических построений, для других — в простоте, а для третьих красота рождается в борьбе против очевидного. В целом же, можно выделить «эстетические инварианты», которыми восхищаются и которые ожидают увидеть все неравнодушные к шахматам. К таким инвариантам авторы относят: комбинации, манёвры, ловушки, сложную логику плана и геометрическую гармонию взаимодействия;
* гибкая среда, которая позволяет игроку выйти за пределы чистой мысли и проверить силу своих идей на доске;
* в своём определении шахмат как искусства Хамбл во многом ссылается на Д. И. Бронштейна и Г. Л. Смоляна. Так, он формулирует следующие условия, которым должна удовлетворять шахматная партия как произведение искусства:
* шахматная партия должна доставлять эстетическое удовольствие;
* шахматная партия должна обладать «эстетическими качествами» (например, целесообразность ходов, точность комбинаций, наличие скрытых угроз и т. д.);
* шахматная партия должна быть уникальной (оригинальность замысла).

Хамбл не рассматривает шахматные партии в качестве великих произведений искусства, так как они «по своей природе не способны раскрыть глубокие проблемы человечества, поднимаемые великим искусством». Поэтому Хамбл определяет шахматы как искусство малых форм, в отличие от живописи, музыки и т. д.

Данное явление не обошло стороной популярную настольную игру «шахматы», соревновательная игра между двумя игроками, действие которой происходят на игровом поле, с разметкой из клеток восемь на восемь, целью которой является с помощью специального набора фигур таких как: пешка, слон, конь, ладья, ферзь, король. Каждая фигура ходит определенным правилами образом. Целью игры является атака фигуры короля оппонента, от которой он не сможет уйти. Ничья может заключаться в случаях:

* договора между двумя игроками;
* отсутствия возможности сделать ход, не противоречащий  
  правилам игры;
* невозможности поставить мат фигурами, оставшимися на доске;
* первые тридцать ходов прошли без взятия фигур;
* семьдесят ходов не происходило ходов пешками и взятия фигур.

Проведение партий шахмат подразделяют по контролю времени, и самыми популярными и общепринятыми, являются классика, рапид и блиц.

При классическом контроле времени на обдумывание каждому игроку дается не мене часа на партию. Наиболее популярный контроль девяносто минут на сорок ходов. Далее возможны варианты: тридцать ходов до конца, двадцать минут с добавлением секунд за каждый сделанный ход и некоторые другие.

Рапид или быстрые шахматы – такой формат игры, в котором каждому из игроков дается время на обдумывания более десяти, но менее шестидесяти минут. Контроль может быть как с добавлением секунд на каждый сделанный ход, так и без добавления сегодня самый популярный контроль времени рапиде – пятнадцать минут и десять секунд за на каждый ход.

В блице контроль времени: меньше или равно десять минут каждому до окончания партии. Партия в шахматах делится на начало, середину и конец, которые в соответствии с шахматными терминами они называются дебют, миттельшпиль и эндшпиль соответственно. Возможные ходы приводят к новым позициям на доске, для которой можно выделить новое множество ходов. В соответствии со сказанным шахматную партию можно представить в виде дерева, пример которого представлен на рисунке 1.1.

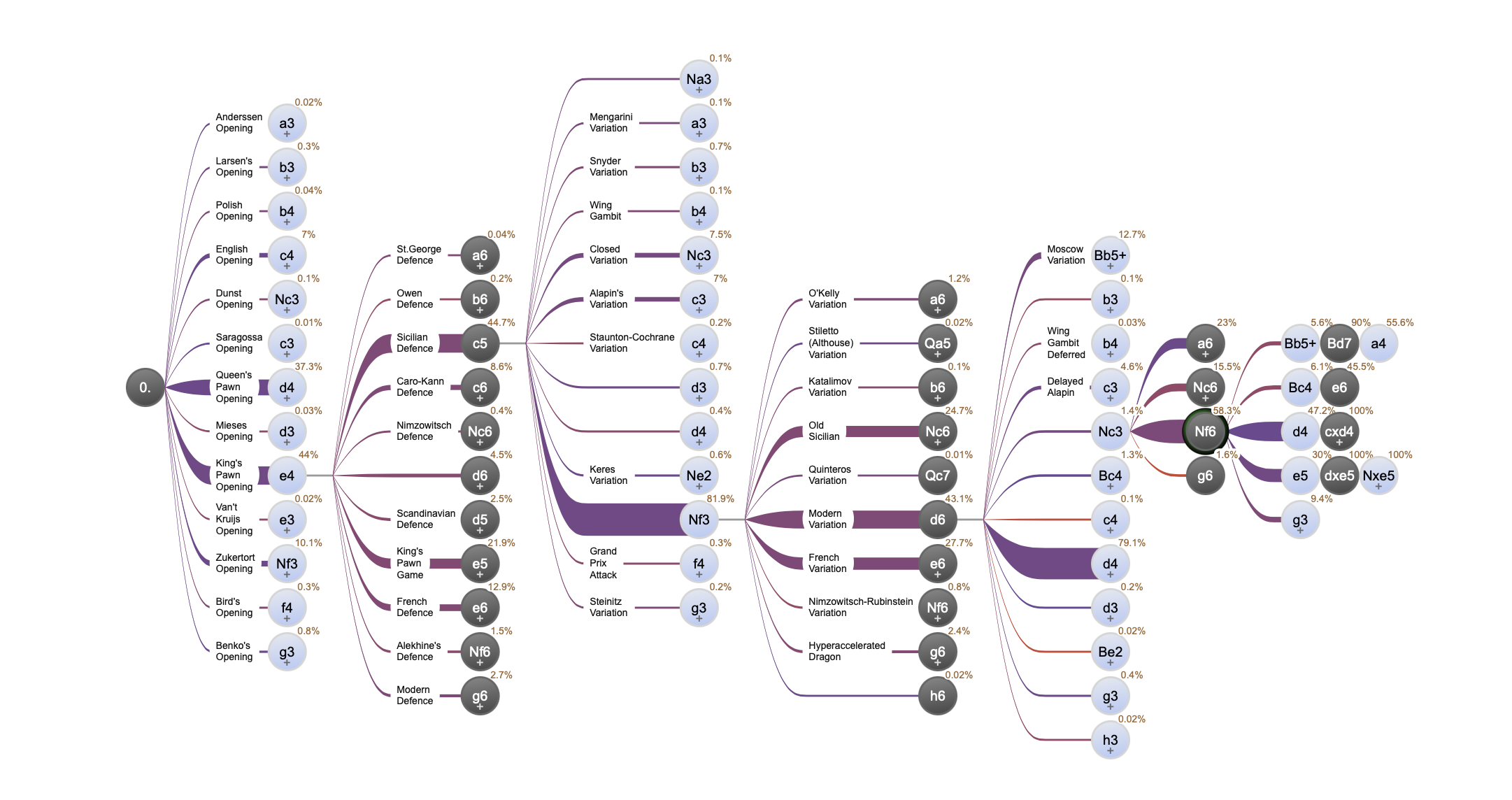


Рисунок 1.1 – Упущенный пример дерева возможных ходов в   
шахматной партии

С каждым ходом дерево растёт. Всего существует двадцать два варианта первого хода, что соответствует двадцати двум вершинам дерева. К пятому ходу число возможных вершин по оценкам составляет 69352859712417 единиц. Таким образом не представляется возможем документирование всех возможных лучших ходов, однако описание начала партии является более простой задачей. Наиболее интересным для шахматного сообщества шахматным началам присуждаются названия в соответствии с турниром, на котором была задокументирована позиция, именами игроков, которые привели к этой позиции, метафорой к внешнему виду расположения фигур. Такие сведения ценны для сообщества игры так как они хранят в себе стратегии, применяемые в игре.

## Описание групп пользователей и функциональных задач

В современном мире шахмат онлайн-платформы играют ключевую роль в обучении, анализе и совершенствовании игровых навыков. Однако у существующих сервисов есть ряд особенностей, которые могут ограничивать пользователей. Например, практически все популярные шахматные платформы сохраняют каждую сыгранную партию в открытом доступе, делая её историю частью публичного профиля игрока. Это может создавать психологический дискомфорт для тех, кто только начинает свой путь в шахматах или экспериментирует с новыми стратегиями. Любая ошибка, неверный ход или неудачная партия остаются зафиксированными в аккаунте, что может отпугивать игроков от свободного изучения новых подходов к игре.

Кроме того, в русскоязычном сегменте интернета наблюдается явный дефицит специализированных платформ, где игроки могли бы не только анализировать свои партии, но и создавать, публиковать и изучать пользовательские руководства по шахматам. Существующие ресурсы чаще всего предлагают либо стандартные учебные материалы, либо форумы с разрозненной информацией, что затрудняет систематизированное обучение.

Веб-приложение направлено на решение этих проблем. Проект стремится создать удобный и функциональный сервис, который предоставит пользователям следующие возможности:

* доступ к качественным руководствам – платформа станет местом, где русскоязычные игроки смогут находить подробные, структурированные и проверенные материалы по различным аспектам шахмат: от основ игры до продвинутых стратегий.
* создание собственных учебных материалов – проект предлагает инструменты для тех, кто хочет делиться своим опытом, будь то разбор дебютов, миттельшпиля, эндшпиля или психологических аспектов игры.
* приватность игрового процесса – в отличие от многих популярных сервисов, программный продукт позволит игрокам сохранять конфиденциальность, выбирая, какие партии сделать публичными, а какие оставить только для личного анализа.

Целевая аудитория проекта:

* русскоязычные игроки, ищущие обучающие материалы – начинающие и любители, которым не хватает структурированных руководств на родном языке.
* авторы контента – шахматисты, тренеры и энтузиасты, желающие создавать учебные статьи, видеоразборы и другие форматы обучающих материалов.
* игроки, ценящие приватность – те, кто хочет избежать публичности каждой сыгранной партии и иметь контроль над своей игровой историей.

Сервис объединит в себе удобство обучения, возможность обмена знаниями и гибкость в настройках приватности, что сделает его востребованным среди русскоязычных шахматистов разного уровня подготовки. Функционал пользователей описан на рисунке 1.2. Текст

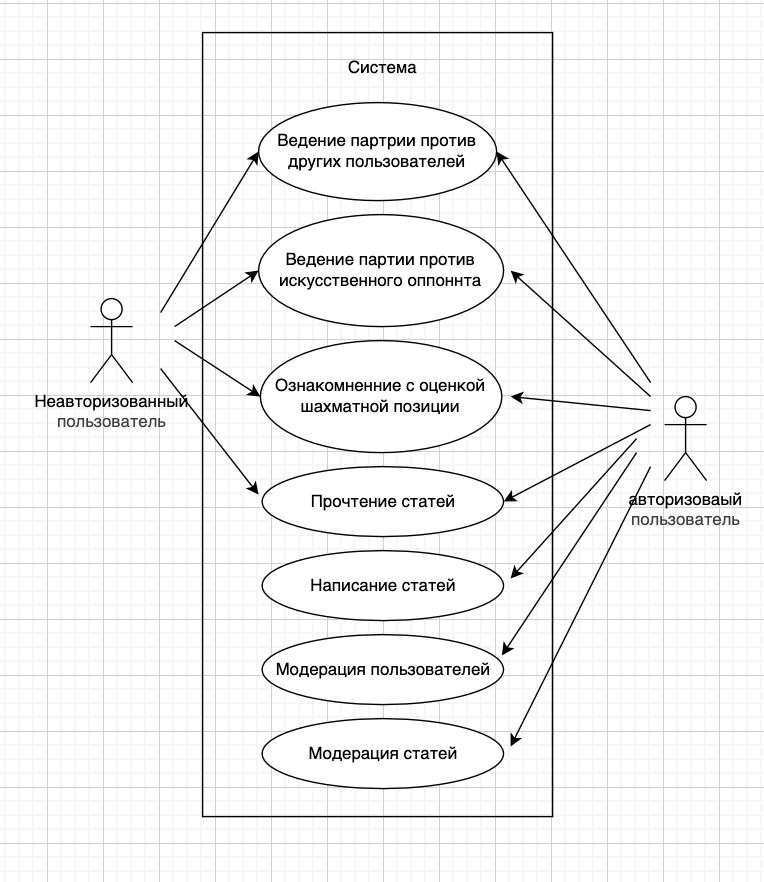


Рисунок 1.2 – диаграмма прецедентов

В проекте используется система ABAC. Пользователи могут иметь атрибуты, которые отвечают за доступ пользователей к функционалу. Как пример: если пользователь имеет атрибут «admin\_users», то на соответствующей странице он сможет модерировать пользователей.

## Анализ программных ресурсов необходимых в работе

**1.3.1 Требования для пользователей**

Поскольку проект представляет собой веб-приложение, пользователям не требуется установка дополнительного программного обеспечения. Достаточно иметь:

* любой современный веб-браузер (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge и др.), который предустановлен в большинстве операционных систем (Windows, macOS, Linux, Android, iOS);
* стабильное интернет-подключение для загрузки и работы с приложением.

Возможные проблемы и их решение:

1. если у пользователя устаревший браузер, система может выдать предупреждение о необходимости обновления.
2. В случае несовместимости или низкой производительности рекомендуется:
   1. обновить браузер до последней версии;
   2. использовать альтернативный браузер (В процессе разработки приоритетным браузером являлся Google chrome)
   3. обратиться к документации проекта или в техподдержку для получения инструкций

1.3.2 Требования для разработки и отладки

Для разработки, тестирования и развертывания проекта потребуются следующие инструменты:

1. Основная среда разработки
   1. WebStorm (платный, мощный инструмент от JetBrains с глубокой интеграцией React и Node.js);
   2. Visual Studio Code (VS Code) (бесплатный, популярный редактор с поддержкой расширений для React, ESLint, Prettier и др.);
   3. Sublime Text (легковесный редактор с плагинами для веб-разработки);
   4. VIM / Neovim (для опытных разработчиков, требует настройки);
   5. Notepad++ (минималистичный вариант для быстрого редактирования кода).
2. Среда выполнения JavaScript (Backend & Frontend)
   1. Node.js (LTS-версия) – необходима для:
      1. Запуска серверной части (если используется Express/Nest.js).
      2. Управления зависимостями через npm (Node Package Manager) или yarn.
      3. Сборки фронтенда (Webpack, Vite).
3. Дополнительные инструменты
   1. Git (система контроля версий, интеграция с GitHub/GitLab/Bitbucket);
   2. Postman (тестирование API);
   3. ESLint + Prettier (проверка стиля кода и автоматическое форматирование);
   4. React Developer Tools (расширение для отладки React-приложений в браузере).
4. База данных:
   1. SQLite для базы шахматных дебютов. Позволяет легко хранить и передавать данные в едином файле;
   2. Prisma для базы данных пользователей. Позволяет разрабатывать и управлять базой данных без необходимости установки стороннего программного обеспечения.

## Анализ аналогов и прототипов

В таблице 1.1 приведено сравнение двух самых популярных шахматных ресурсов и справочных сайтов.

Таблица 1.1 – Сравнение аналогов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Chess.com | Lichess.org | Wikipedia | Разрабатываемый сервис «Shahmatki» |
| Доступность | Сервис заблокирован на территории Российской Федерации | Сервис поддерживается на благотворительной основе и не предоставляет круглосуточную работу сайта | Имеет нарушения законодательства Российской Федерации |  |
| Предоставление ресурсов для разработчиков | Нет | Ежемесячно выставляет в открытый доступ архив партий игроков за месяц | Нет | Планируется предоставление API для доступа к статистике, партиям и обучающим материалам, а также возможность экспорта данных. |
| Функционал | Предоставляет множество форматов с альтернативными правилами игры | Классические шахматы | Написание статей | Сочетание классических шахмат с уникальными обучающими механики. |
| Контент сообщества | Форум сообщества | Написание пользовательских руководств (не популярно на русском языке) | Все статьи написаны сообществом | Полноценная платформа для публикации и обсуждения шахматных статей, гайдов и разборов с модерацией и рейтинговой системой |
| Искусственные оппоненты | Большая часть доступна по платной подписке | Несколько уровней сложности | Функционал шахмат отсутствует | Бесплатные ИИ-соперники с адаптивным обучением под уровень игрока |
| Доступность на русском языке | Переведен не полностью | Переведён не полностью | Возможность перевести партию на русский язык | Полная поддержка русского языка, включая интерфейс, обучающие материалы и техническую документацию. |

Разрабатываемый сервис «Shahmatki» стремится предоставлять уровень пользовательского интерфейса уровнем не ниже Chess.com, предоставление ресурсов для разработчиков и возможности Wikipedia в написании статей, но с функционалом для написания статей на шахматную тематику. Таким образом, «Shahmatki» займет нишу русскоязычного шахматного сервиса с упором на обучение, анализ и комфортное сообщество, предлагая уникальные возможности как для игроков, так и для разработчиков.

# Технологический раздел

## Структура базы данных

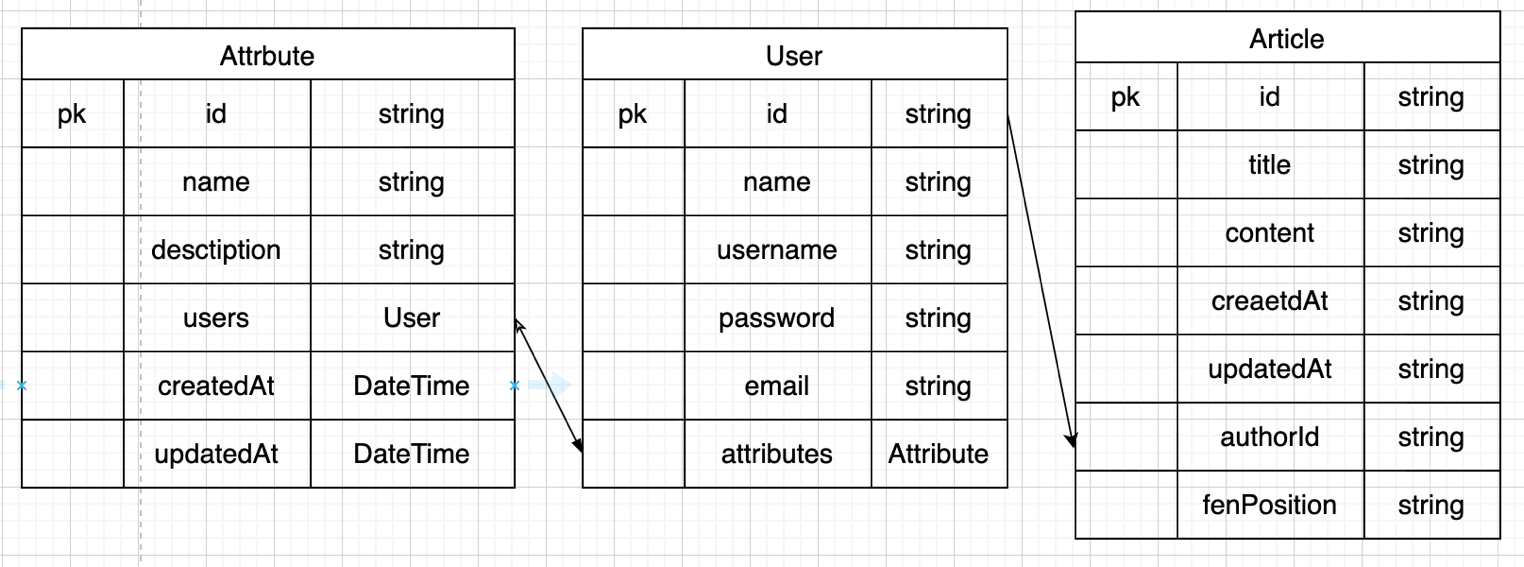
В проекте присутствует 2 базы данных для изоляции логики и предоставления одной из них в открытом доступе. ERD баз данных представлены на рисунках 2.1, 2.2.

Рисунок 2.1 – База данных пользователей

Описание

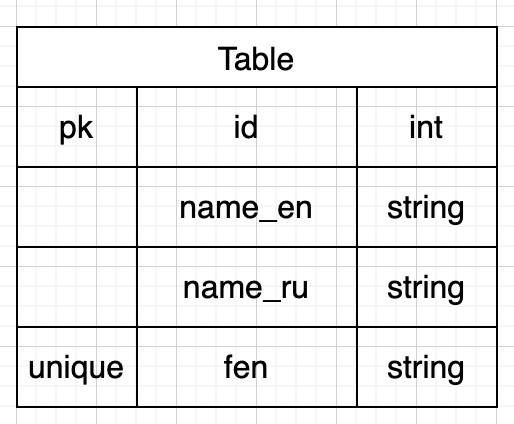


Рисунок 2.2 – База данных шахматных дебютов

## Разработка структуры сайта

2.2.1 структура страниц

Схема сайта изображена на рисунке 2.3.



Рисунок 2.3 – Схема сайта

Почему такие страницы

2.2.2 структура кода

## Разработка дизайн макета сайта

Макет приложения был подготовлен в веб-ресурсе Figma. С соблюдением дизайн-документа Versel. Страницы макета изображены на рисунках 2.4-2.10

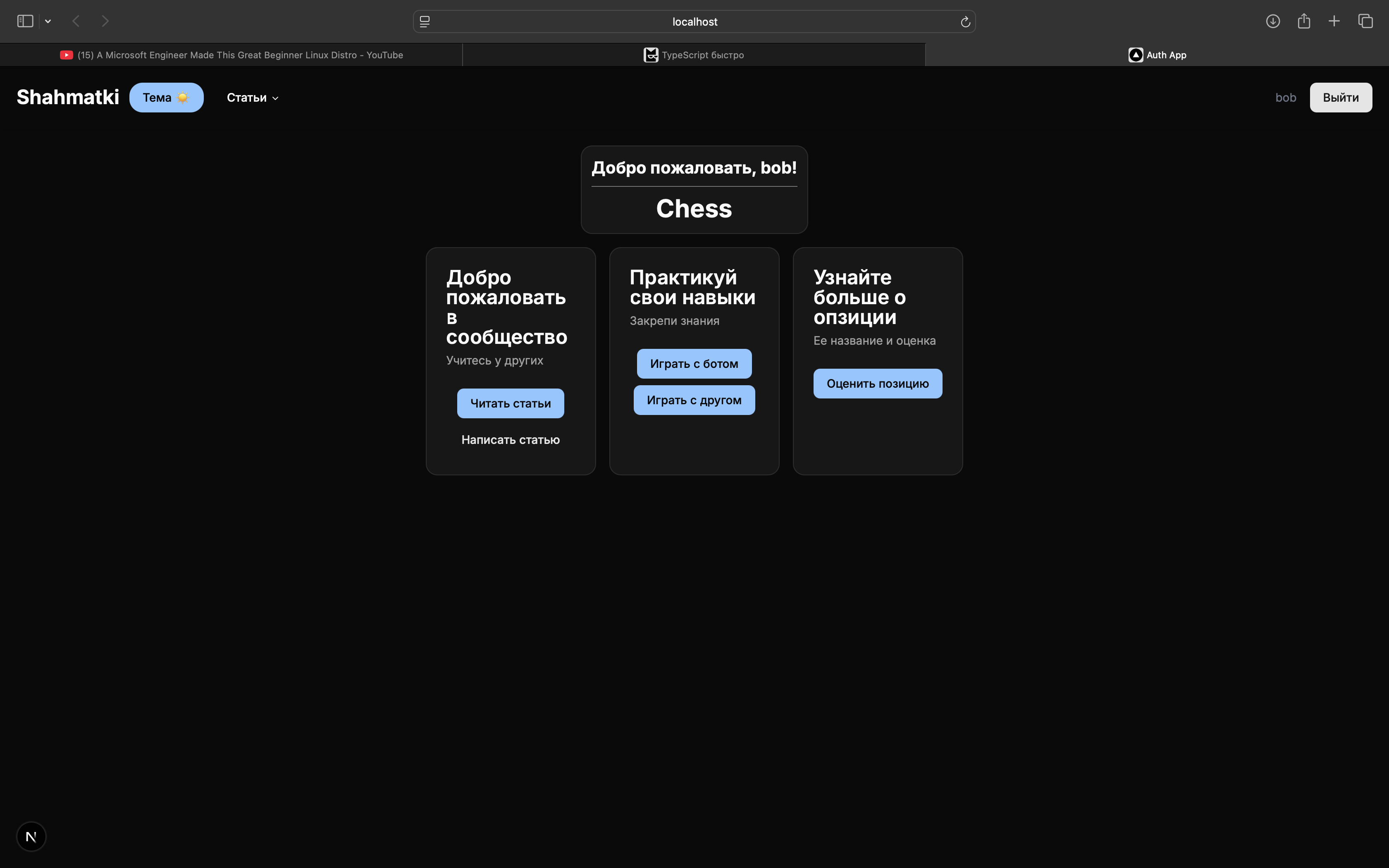


Рисунок 2.4 – Дизайн главной страницы



Рисунок 2.5 – Дизайн страницы анализа позиции

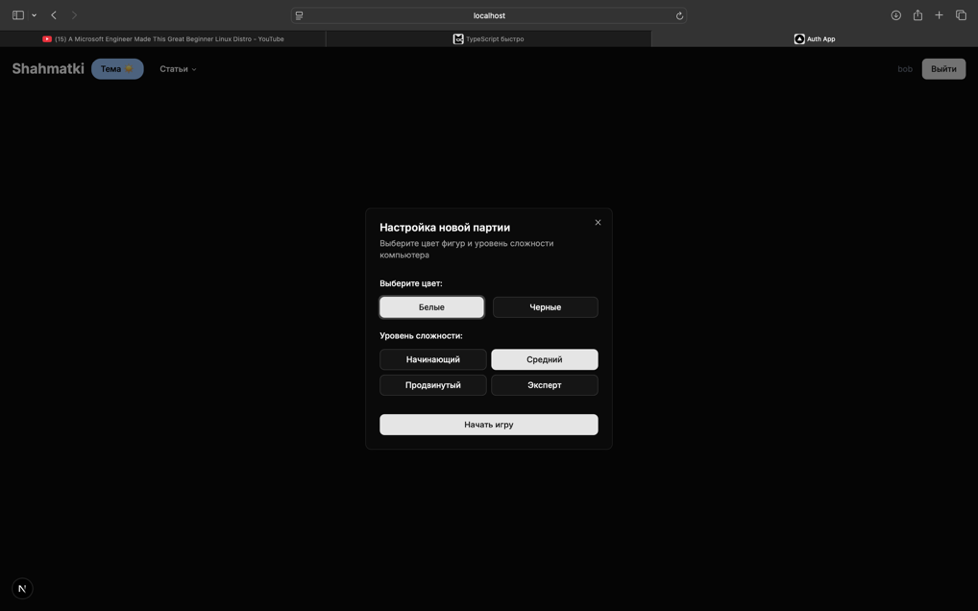


Рисунок 2.6 – Дизайн страницы настнойки искусственного оппонента



Рисунок 2.7 – Дизайн страницы шахмат с ботом

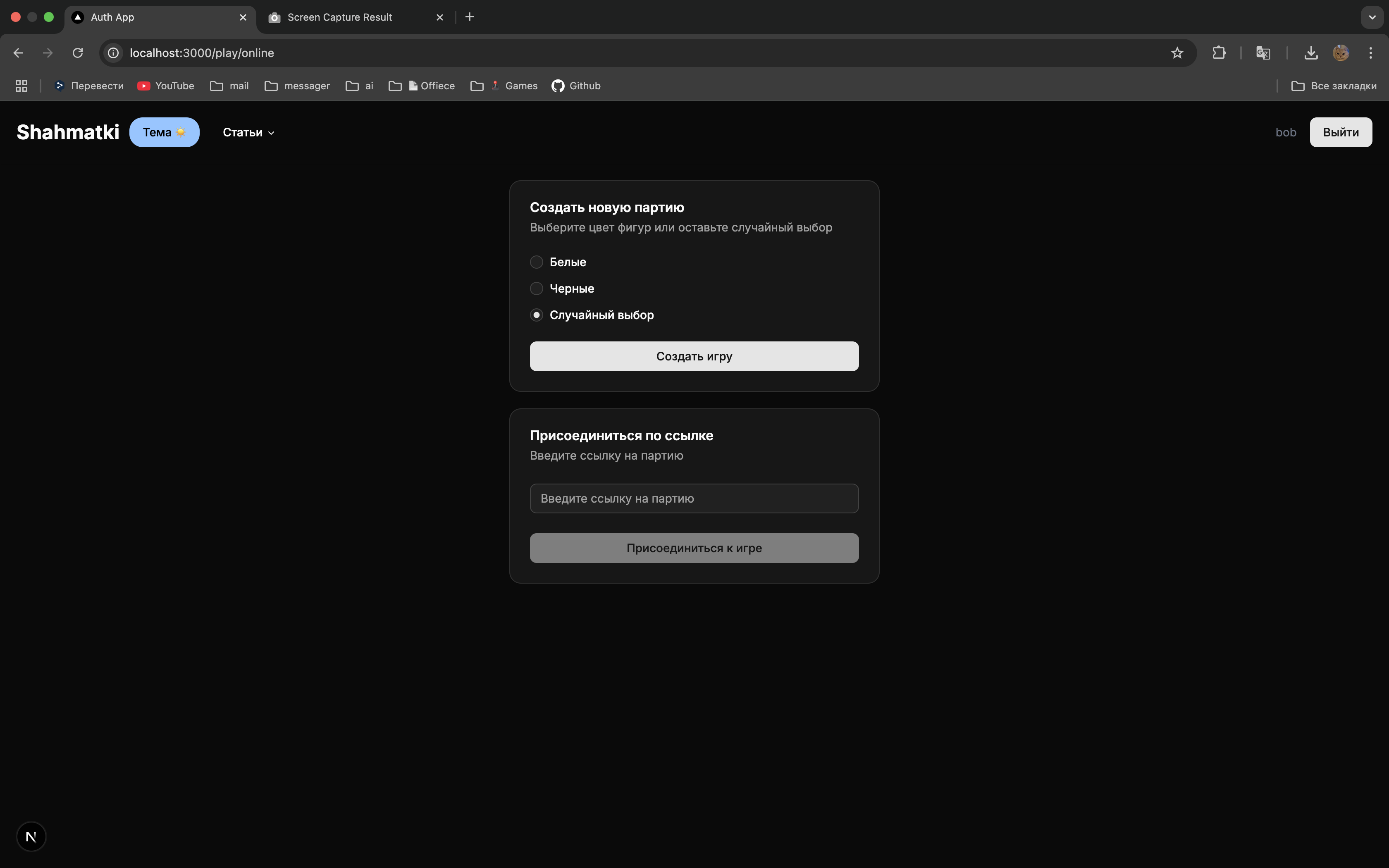


Рисунок 2.8 – Дизайн страницы создания партии онлайн

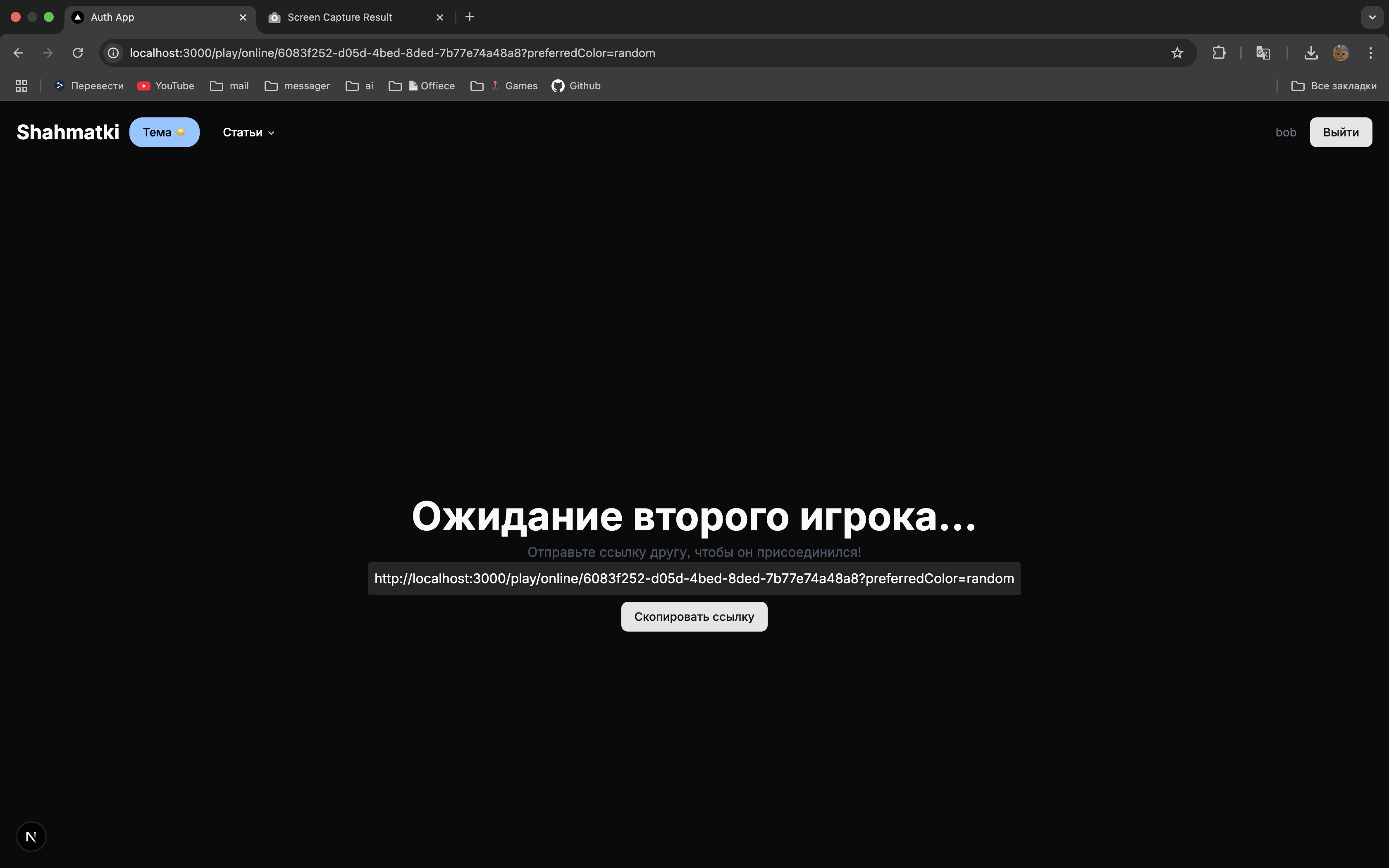


Рисунок 2.9 – Дизайн страницы ожидания второго игрока



Рисунок 2.10 – Дизайн страницы шахмат против другого игрока

При отображении шахматной доски используются контрастные цвета, что помогает пользователям находить доску на экране.

## Разработка алгоритмов обработки информации

2.4.1

Методы обработки пользовательского ввода зависят от конкретных элементов, с которыми производятся взаимодействия, их можно разделить на 4 категории:

* ссылки. Используются для переходов между страницами сайта и на внешние ссылки. В проекте реализованы через компонент «Link» из пакета «next». Является аналогом HTML тэга «a», используемого для переходов по ссылкам, однако при навигации по страницам веб-приложения не обновляет глобальные ресурсы сайта, такие как: глобальные css атрибуты, метаданные;
* кнопки. Используются для вызова действий. Реализован через «button». Чтобы реализовать вызов функции по нажатию необходимо использовать атрибут onClick в формате «onClick={()=> [название функции]}»;
* формы. Используются для ввода текста и выполнения запросов к API.  При нажатии «submit button» данные из input попадают в соответствующую переменную и передаются в функцию автоматически.
* шахматная доска. Реализуется через компонент «Chessboard». При вызове принимает в себя функции через «props» и при срабатывании события вызывает соответствующие функции.

Ниже приведен упрощенный код обработки пользовательского ввода через форму:

<form onSubmit={handleSubmit} className="space-y-6">

<div>

<Label htmlFor="title">Заголовок</Label>

<Input

id="title"

value={title}

onChange={(e) => setTitle(e.target.value)}

required

/>

</div>

<div>

<MarkdownEditor

value={content}

onChange={setContent}

label="Содержание"

/>

</div>

<div className="flex items-center space-x-2">

<Checkbox

id="hasChessBoard"

checked={hasChessBoard}

onCheckedChange={(checked) => setHasChessBoard(checked as boolean)}

/>

<Label htmlFor="hasChessBoard">Добавить шахматную позицию</Label>

</div>

{hasChessBoard && (

{/\* Шахматная доска \*/}

)}

<Button type="submit" disabled={isSubmitting}>

{isSubmitting ? 'Сохранение...' : 'Сохранить'}

</Button>

</form>

В форму передается «onSubmit={handleSubmit}», что означает вызов функции handleSubmit при подтверждении пользовательского ввода. Для подтверждения реализована кнопка с «type="submit"», при нажатии вызывается функция, передаваемая в форму.

2.4.2 Разметка пользовательских статей

Для отображения пользовательских статей используется разметка markdown, это позволяет пользователям оформлять текст руководства. В базе данных хранится запись с контентом статьи. При переходе пользователем на динамический роут отправляется запрос к базе данных. Полученный текст передается в компонент «ReactMarkdown» пакета «react-markdown», который преобразует текст с учетом разметки и оформления. Также пользователь может прикрепить шахматную позицию к руководству, для хранения используется нотация Форсайта – Эдвардса:

* положение фигур со стороны белых. Позиция описывается цифрами и буквами по горизонталям сверху вниз начиная с восьмой горизонтали и заканчивая первой. Расположение фигур на горизонтали записывается слева направо, данные каждой горизонтали разделяются косой чертой /. Белые фигуры обозначаются заглавными буквами. K, Q, R, B, N, P — соответственно белые король, ферзь, ладья, слон, конь, пешка. k, q, r, b, n, p — соответственно чёрные король, ферзь, ладья, слон, конь, пешка. Обозначения фигур взяты из англоязычного варианта алгебраической нотации. Цифра задаёт количество пустых полей на горизонтали, счёт ведётся либо от левого края доски, либо после фигуры (8 означает пустую горизонталь);
* активная сторона: w — следующий ход принадлежит белым, b — следующий ход чёрных;
* возможность рокировки. k — в сторону королевского фланга (короткая), q — в сторону ферзевого фланга (длинная). Заглавными указываются белые. Невозможность рокировки обозначается «-»;
* возможность взятия пешки на проходе. Указывается проходимое поле, иначе «-»;
* счётчик полуходов. Число полуходов, прошедших с последнего хода пешки или взятия фигуры. Используется для определения применения правила 50 ходов;
* номер хода. Любой позиции может быть присвоено любое неотрицательное значение (по умолчанию 1), счётчик увеличивается на 1 после каждого хода чёрных.

Например, позиция «rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1» изображена на рисунке 2.11

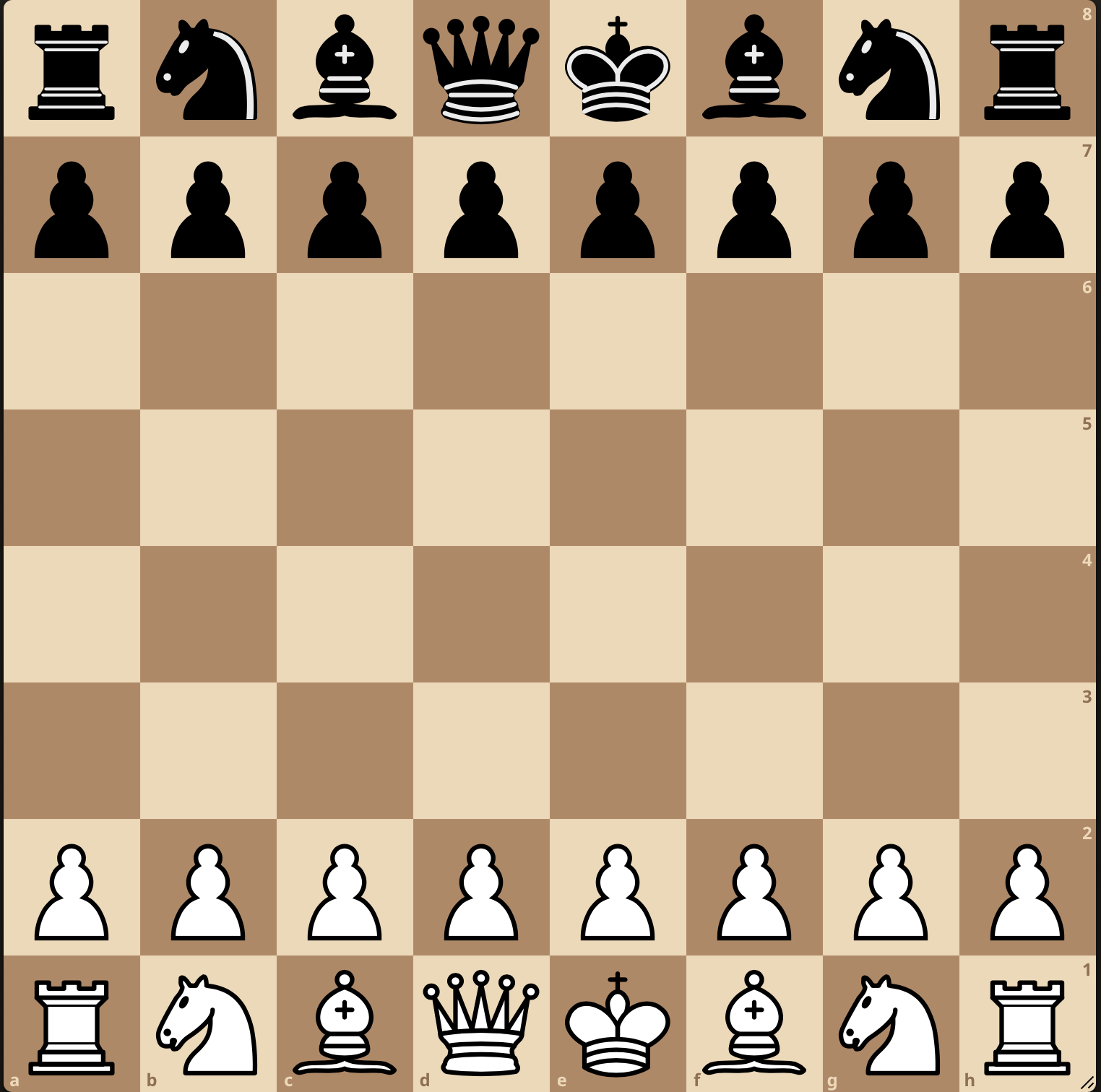


Рисунок 2.11 –  Начальная позиция

2.4.3 Взаимодействие между пользователями в реальном времени

В проекте необходимо реаизовать взаимодействие между пользователямси в реальном времени для ведения шахматных партий онлайн. Для этого необходимо реализовать:

* соединение двух пользователей с сервером;
* минимальные задержки между пользователем и сервером;
* минимизировать нагрузки на сеервер.

Ссылка на таблицу

Таблица …

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | WebSocket | HTTP polling | HTTP long polling | Server-sent events |
| Тип связи | Двухсторонняя (Full-duplex) | Односторонняя | Односторонняя | Односторонняя |
| Задержка | Минимальная | Высокая | Средняя | Низкая |
| Эффективность | Высокая благдаря единому соединению | Низкая из-забольшого количества заголовков | Низкая из-за времени ожидания и количества запросов | Высокая благодаря единому соедиеннию |
| Серверная нагррузка | Низкая (единое соединение) | Высокая (множество запросов) | Высокая (время ожидания и колиечство запросов) | Низкая (единое одностороннее соединение) |
| Поддержка браузеров | Все | Все | Все | Все кроме  Internet Explorer |

Из сравнения видно преимущество webSocket при необходимости реализации двухстороннего соединения.

WebSocket — это протокол двусторонней связи основанный на TCP, который позволяет устанавливать постоянное соединение между клиентом и сервером. Это делает его эффективным решением для приложений, требующих обмена данными в реальном времени (чаты, онлайн-игры, трейдинг, коллаборативные инструменты).

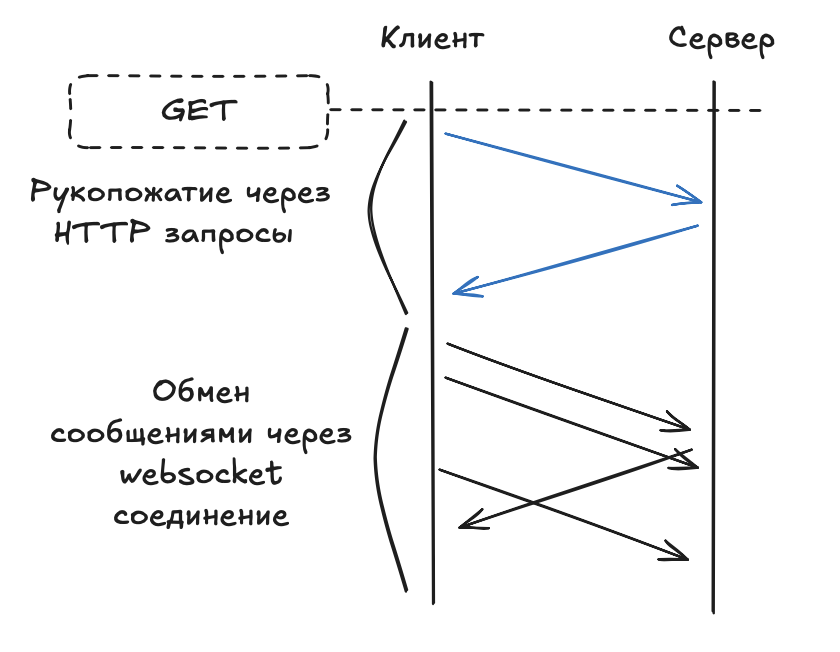


Рисунок 2.12 –  Пример  работы WebSocket cоединения

При создании соедиинения передается:

* url. Соединение создается в отдельном от опсновного приложения потоке для большей устойчивости сервиса;
* roomId. Уникальный идентификатор партии;
* playerId. Уникальный идентификатор пользователя;
* preferredColor. Цвет фигур, выбранный пользователем (белые, черные, случайные).

wss.on("connection", (ws, req) => {

try {

const url = new URL(req.url, `http://${req.headers.host}`);

const roomId = url.searchParams.get("roomId");

const playerId = url.searchParams.get("playerId");

const preferredColor = url.searchParams.get("preferredColor");

if (!roomId || !playerId) {

console.warn("❌ Отклонено подключение: отсутствуют roomId или playerId.");

ws.send(JSON.stringify({ error: "Неверные параметры подключения" }));

ws.close();

return;

}

if (!rooms[roomId]) {

rooms[roomId] = {

game: new Chess(),

players: {},

moveHistory: [],

clients: new Set(),

gameStarted: false,

creatorPreference: preferredColor || 'random',

rematchRequests: new Set(),

};

}

const room = rooms[roomId];

room.clients.add(ws);

ws.playerId = playerId;

Если roomId нет на сервере, то в список партий добавляется объект комнаты, в противном случае к сществующей комате добавляется переданный пользователь.

Пользователь-инициатор комнаты перенаправляется на страницу ожидания, на которой ему предлагается скопировать ссылку-приглошение. Как только второй пользователь перейдет по этой ссылке – создастся запрос на подключение к webSocket и игра между двумя пользователями начнется. Примеры сообщиний между польщователем и webSocket   соединением приведены на рисунках 2.13, 2.14

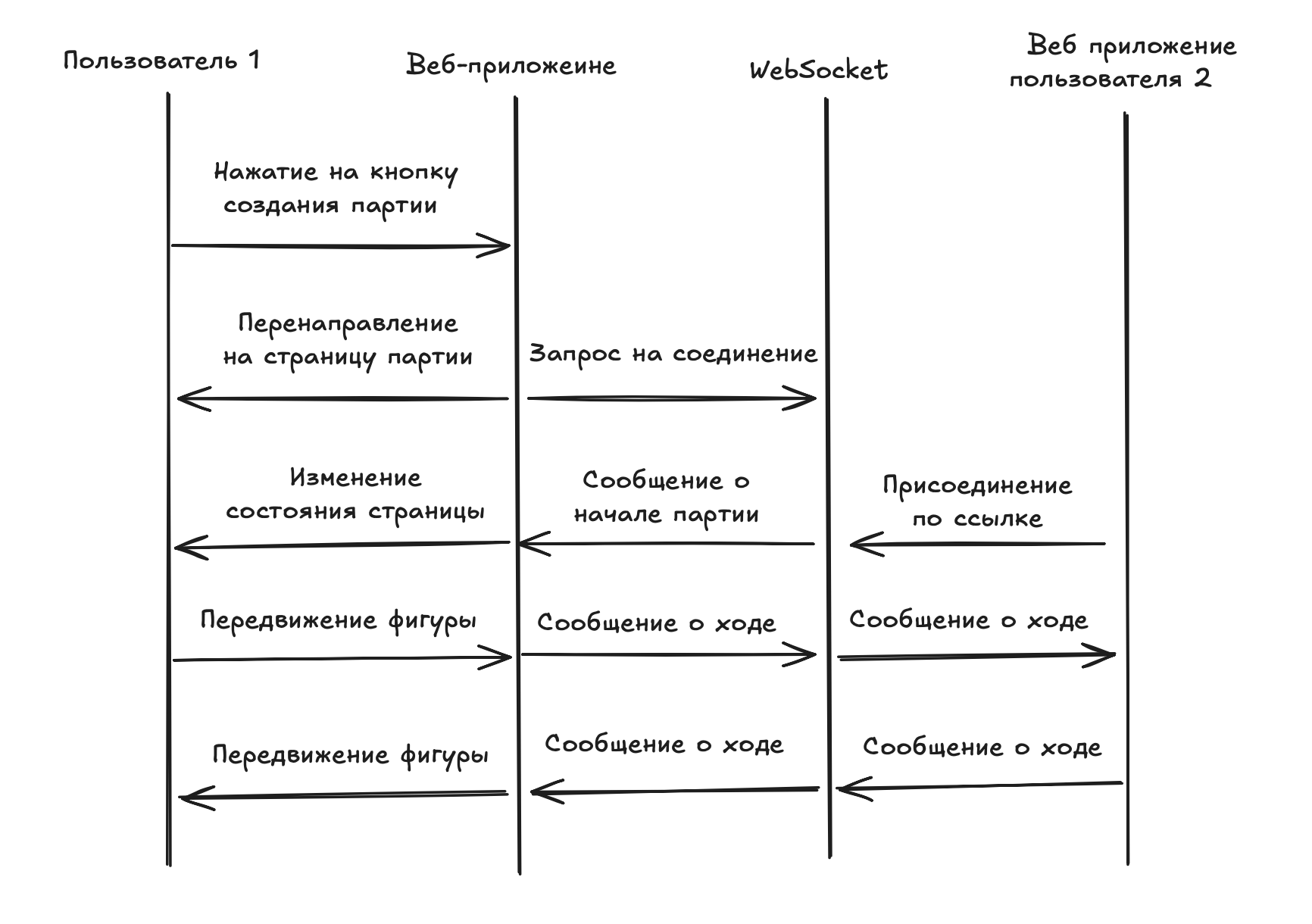


Рисунок 2.13 –  Пример обмена сообщениями при создании комнаты

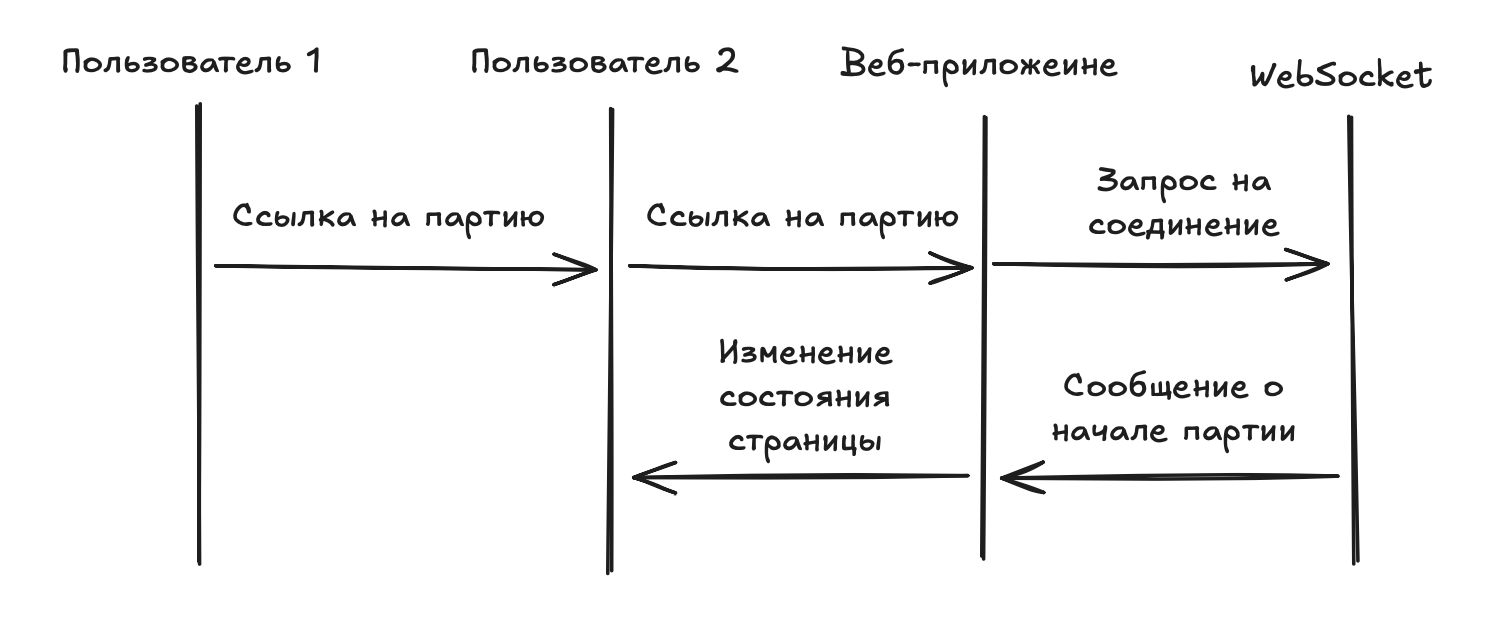


Рисунок 2.14 –  Пример обмена сообщениями присоединении второго пользователя

2.4.4 Верификация пользователя

Для безопасной передачи чувствительных данных пользователю необходимо строго соблюдать процедуры аутентификации и верификации, чтобы предотвратить несанкционированный доступ третьих лиц к учетным записям пользователей.

В Next.js-приложениях для верификации пользовательских запросов используется комбинация технологий:

* JWT (JSON Web Token) - стандартный метод для безопасной передачи данных между сторонами в виде JSON-объекта;
* Auth.js (ранее NextAuth.js) - популярная библиотека для аутентификации в Next.js-приложениях.

 Процесс аутентификации пользователя:

1. Регистрация и вход пользователя. При успешной регистрации или входе система генерирует два типа токенов:
   1. Refresh Token - долгоживущий токен для продления сессии;
   2. Access Token - краткосрочный токен для доступа к защищенным ресурсам;
2. Хранение токенов:
   1. Refresh Token хранится в HTTP-only, Secure, SameSite=Strict cookie с максимальным сроком жизни 30 дней;
   2. Access Token передается в ответе аутентификации и хранится в памяти клиентского приложения (не в localStorage);
3. Использование токенов:
   1. При каждом запросе к защищенным API отправляется Access Token в заголовке Authorization;
   2. Если Access Token истек (через 15 минут), клиент использует Refresh Token для получения нового Access Token без необходимости повторного входа.

Код верификации пользователя приведен ниже:

export const { auth, handlers, signOut, signIn } = NextAuth({

adapter: PrismaAdapter(prisma),

secret: process.env.AUTH\_SECRET,

providers: [

Credentials({

credentials: {

email: { label: "Email", type: "text", placeholder: "email@example.com" },

password: { label: "Password", type: "password" }

},

async authorize(credentials) {

if (!credentials?.email || !credentials?.password) {

throw new Error('Введите email и пароль');

}

try {

const validatedCredentials = loginSchema.parse(credentials);

const user = await prisma.user.findFirst({

where: {

email: validatedCredentials.email,

password: validatedCredentials.password

},

include: {

attributes: true

}

});

if (!user) {

throw new Error('Неверный email или пароль');

}

return user as User;

} catch (error) {

console.error('Auth error:', error);

return null;

}

}

})

],

pages: {

signIn: '/auth',

},

session: {

strategy: "jwt"

},

callbacks: {

async jwt({ token, user, account, trigger }) {

// Always fetch fresh user data with attributes

const dbUser = await prisma.user.findUnique({

where: {

id: user?.id || token.id

},

include: {

attributes: true

}

});

if (dbUser) {

token.id = dbUser.id;

token.username = dbUser.username || '';

token.attributes = dbUser.attributes.map(attr => attr.name);

console.log('JWT Debug: Fresh attributes loaded', {

userId: dbUser.id,

attributes: token.attributes

});

}

return token;

},

async session({ session, token }) {

if (session.user) {

session.user.id = token.id as string;

session.user.name = token.username as string;

session.user.attributes = Array.isArray(token.attributes) ? token.attributes : [];

}

return session;

}

}

});

Ключевые моменты:

1. интеграция с базой даных:
   1. используется PrismaAdapter для связи NextAuth с базой данных;
   2. все операции с пользователями выполняются через Prisma;
   3. пользовательские данные включают связанную сущность attributes для реализации ABAC;
2. поток работы:
   1. пользователь вводит email и пароль на странице /auth;
   2. данные валидируются и ищутся в базе;
   3. при успехе создаётся JWT токен с данными пользователя;
   4. при последующих запросах токен используется для:
      1. проверки аутентификации;
      2. доступа к данным пользователя (ID, имя, атрибуты)
      3. обновления данных пользователя из базы при каждом запросе;
3. собенности реализации:
   1. постоянная синхронизация с базой: JWT колбэк всегда запрашивает свежие данные пользователя, что гарантирует актуальность информации;
   2. расширенная модель пользователя: В отличие от стандартной конфигурации, здесь пользователь имеет дополнительные атрибуты, которые включаются в токен и сессию;
   3. отладка: Добавлено логирование процесса загрузки атрибутов для отслеживания проблем;
   4. безопасность: Используется секрет из переменных окружения, пароли сравниваются напряму.

## Разработка текстового содержания сайта

Разработка текстового содержания выделятся при разработке такового отдельно от оформления стилей и функционала, как оно велось бы при разработке сайта с использованием HTML напрямую, однако в большинстве современных решений этого не происходит.

При разработке веб-приложений на React функционал и текстовое содержание разрабатываются вместе. В проекте используется Tailwind. Это означает, что стили также интегрированы в текстовое содержание сайт напрямую это изображено на рисунках 2.15, 2.16.

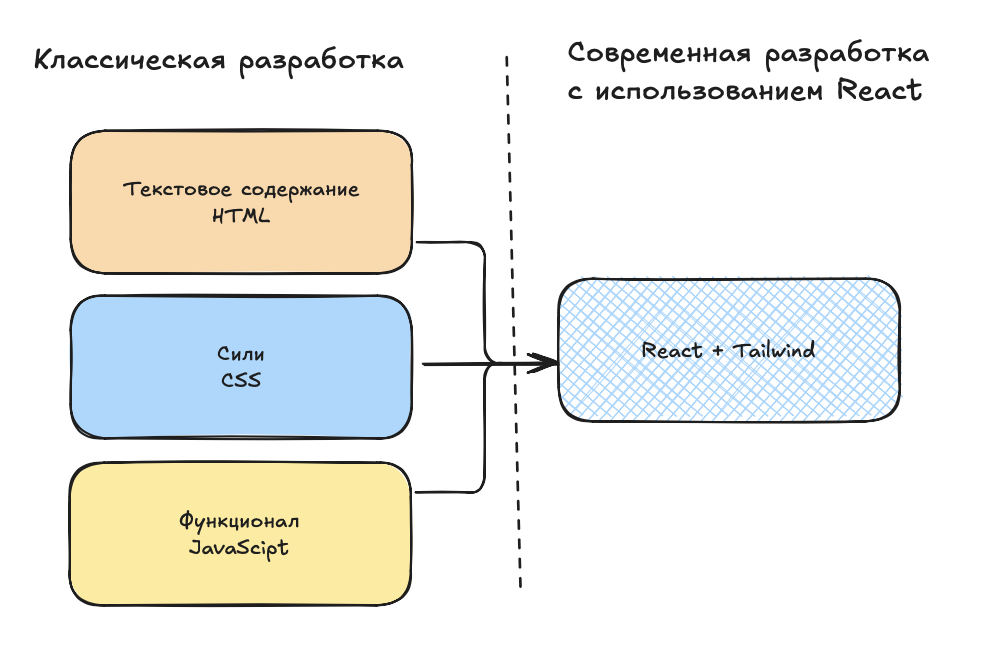


Рисунок 2.15 – Сравнение классической разработки сайтов и React

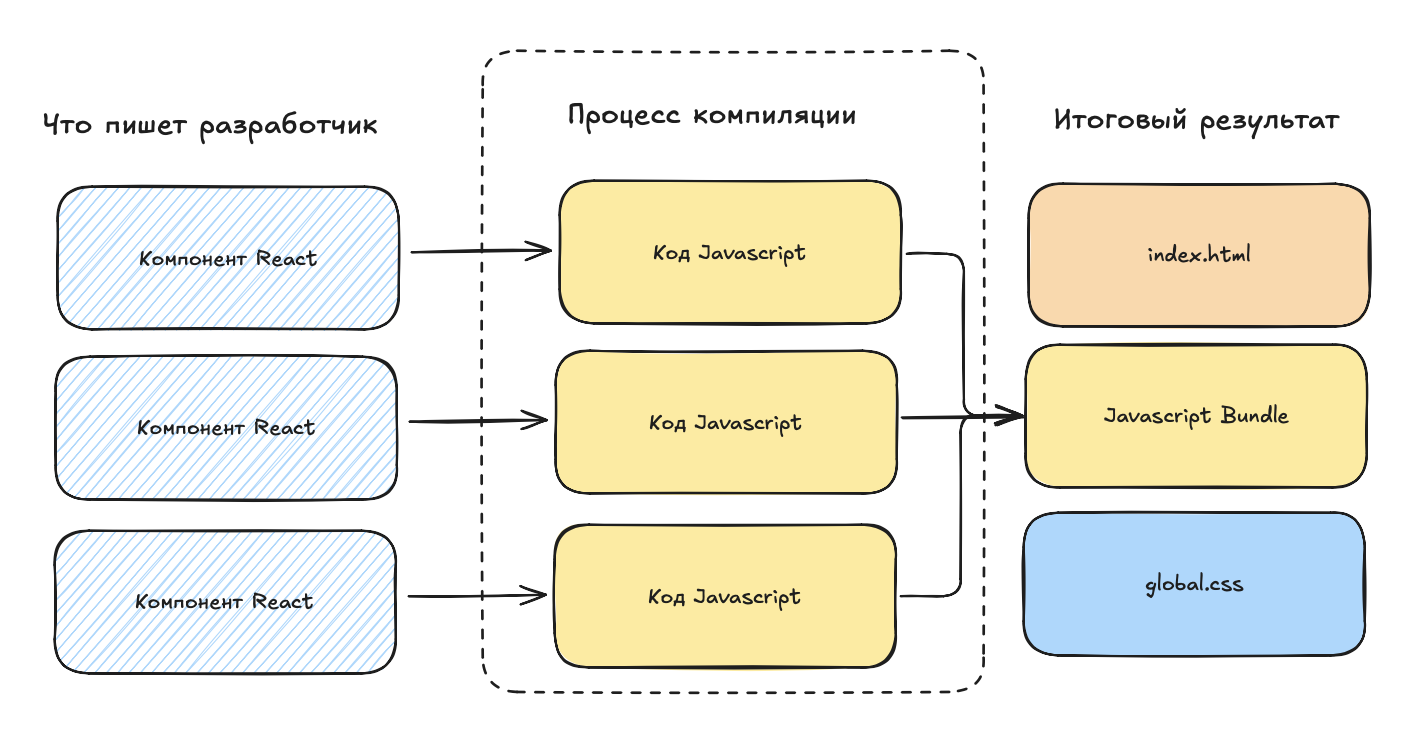


Рисунок 2.16 – Процесс компиляции React

Финальный пользователь получет пакет Javascript и разметку сайта HTML, CSS. Далее браузер клиента исоплняет JavaScript, который генерирует все текстовое содержание и стили сайта. Это самый простой пример, так как современные технологии комплекснее. Процесс компиляции можт включать в себя использоание сторонних сервисов и компиляторов, упрощенная система отображена на рисунке 2.17

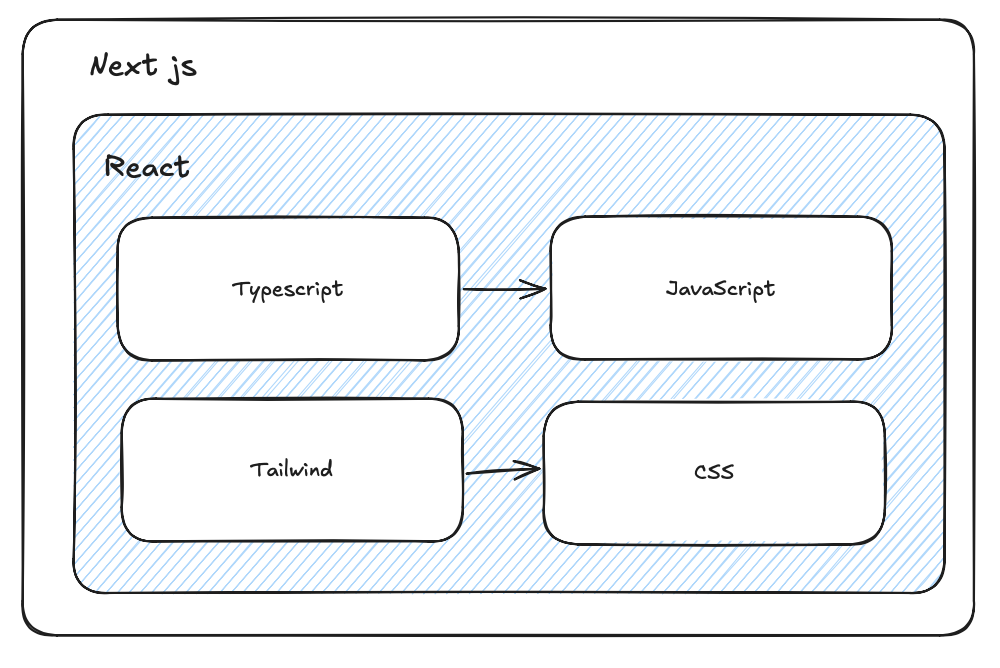


Рисунок 2.17 – Комплескность комписяции современных фреймворков

В проекте используется метафреймвок Next js, основанный на фреймворе React.  Одним из значительных изменений является добавление SSR (рендеринг на стороне сервера), разница между ними отображена на рисунке 2.18.

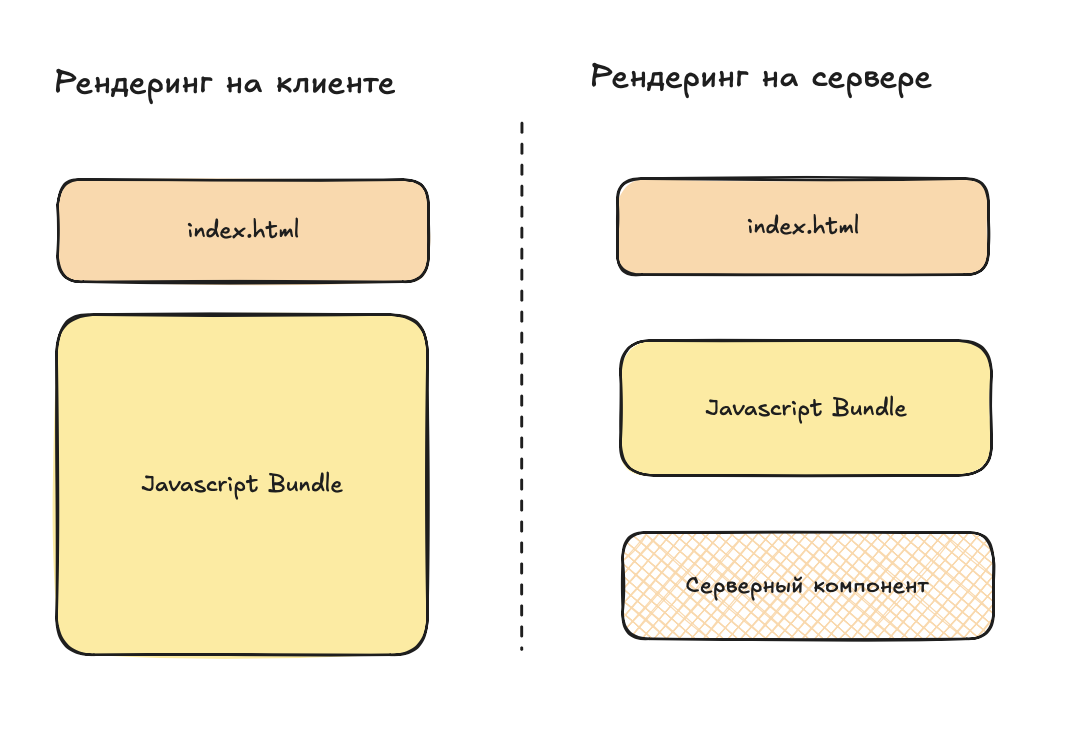


Рисунок 2.18 – Пример разницы между серверным и клиентским рендерингом

Использование серверного рендеринга позволяет уменьшить размер пакета JavaScript так как расчет компонента произошел на сервере. Это позволят снизить нагрузку на устройство и интернет-соединение клиента.

Часть контента генерируется от API. Как пример – пользовательские статьи. В базе данных хранится запись с контентом статьи. При переходе пользователем на динамический роут отправляется запрос к базе данных. Полученный текст передается в компонент «ReactMarkdown» пакета «react-markdown», который преобразует текст с учетом разметки и оформления.

## Разработка мультимедийного контента

Картинки Анимации