Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Северо-Восточный федеральный университет им. М.К. Аммосова»

Институт математики и информатики

Кафедра «Информационные технологии»

**РЕАЛИЗАЦИЯ САЙТА НА ОСНОВЕ ФРЕЙМВОРКА NEXT.JS**

КУРСОВАЯ РАБОТА

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Выполнил: студент III курса  
группы Б-ИВТ-19-2 ИМИ СВФУ  
Васильев Игорь Георгиевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Научный руководитель: старший преподаватель кафедры «Информационные технологии» ИМИ СВФУ  
Владимир Семенович Леверьев  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Якутск 2022

Содержание

[Введение 3](#_Toc106157707)

[Глава 1 4](#_Toc106157708)

[1.1 Требования к сайту 4](#_Toc106157709)

[1.2 Выбор фреймворка для реализации веб-приложения 6](#_Toc106157710)

[Angular.js 7](#_Toc106157711)

[React.js 8](#_Toc106157712)

[Vue.js 10](#_Toc106157713)

[Next.js 11](#_Toc106157714)

[1.3 Сранение Gatsby.js и Next.js 13](#_Toc106157715)

[Svelte 15](#_Toc106157716)

[Глава 2 16](#_Toc106157717)

[2.1 Структура проекта 16](#_Toc106157718)

[2.2 Интерфейс сайта/дизайн 22](#_Toc106157719)

[Заключение 24](#_Toc106157720)

[Библиографический список 25](#_Toc106157721)

[Глоссарий 26](#_Toc106157722)

# Введение

В данной курсовой работе был проведён анализ популярных фреймворков для веб-разработки и был выбран нами подходящий для нашей цели.

**Актуальность темы.**

Сайт с результатами олимпиад даст возможность участникам дополнить свое портфолио из даты базы сайта без каких-либо проблем и поможет верификации результатов намного быстрее без необходимости скачивать файлы и поиска результатов.

Сайт с результатами олимпиад поможет сделать олимпиаду более престижной и поможет быть воспринятым более серьезно сохранив результаты прошлых лет, чтобы они не пропадали без вести.

Также сайт с результатами поможет олимпиаде с публичностью, давая всем желающим возможность просмотреть результаты проведенных олимпиад.

**Цель.  
Реализация сайта олимпиад по программированию Республики Саха**

**Задачи:**

1. Собрать требования к сайту
2. Выбор компонентов технологического стека
3. Написание программного кода
4. Клиентское тестирование прототипа
5. Разработка дизайна веб-страниц
6. Финальное тестирование

# Глава 1

* 1. Требования к сайту

Задача «разработать веб-приложение» очень сильно изменилась за последнее десятилетие.

Изначально под разработкой сайта понималась следующая совокупность действий: его написание с помощью использования HTML- и CSS-технологий, размещение и ручная публикация на хостинге.

Сегодня подходы к разработке, требования заказчиков и желания конечных пользователей увеличиваются и усложняются, а их внимание обращено на такие параметры как:

* скорость разработки;
* время загрузки страницы;
* кроссплатформенность приложения;
* автоматизация разработки;
* стабильность приложения.

Обеспечить высокую скорость разработки позволяют различные open source фреймворки и библиотеки, которые упрощают написание кода и позволяют решать типовые задачи с минимальными затратами со стороны разработчика. Кроме указанных ранее преимуществ, в них уже реализована стандартная логика по взаимодействию с HTTP-сервером и клиентским API для построения интерфейсов, что также минимизирует возможность допустить ошибку из-за человеческого фактора.

Время получения контента пользователем – производительность веб-приложения – становится ключевым показателем и его преимуществом или недостатком в сравнении с конкурентами. При этом важно понимать, что проблема низкой скорости интернет-соединения до сих пор актуальна

Один из современных подходов в разработке информационных систем предполагает использование системы контейнеризации приложений для создания изолированного окружения. Преимуществом данного подхода является независимость окружения каждого приложения от установленных на исполняющей контейнеры ЭВМ программ, что обеспечивает прозрачность и предсказуемость работы приложения и защищает от пересечения зависимостей разных версий. Также система управления контейнерами предоставляет механизмы управления жизненным циклом контейнеров и обеспечивает целостность декларативно заданной среды.

Автоматизация разработки позволяет упростить процессы разработки и тестирования, благодаря этому требования заказчиков и пожелания пользователей становится возможным выполнить в кратчайшие сроки.

Методологии непрерывной интеграции, непрерывной развёртки и непрерывной доставки встроены в большинство систем контроля версий, что упрощает их поиск и настройку.

Гарантировать стабильность приложения позволяет настройка различных мониторингов, которые в реальном времени могут сообщать о каких-либо ошибках в работе приложения. Ошибка работы приложения в ключевом месте может стоить очень дорого для бизнеса, поэтому очень важно уметь быстро реагировать на их возникновение, чтобы разработчики имели возможность устранить их в кратчайшие сроки

* 1. Выбор фреймворка для реализации веб-приложения

Для веб-разработки создано очень много фреймворков и библиотек, и порой очень сложно сделать правильный выбор, чтобы получить оптимальное соотношение между скоростью и качеством разработки.

В настоящее время для создания веб-приложения недостаточно использовать только HTML/JS/CSS, возможностей которых было достаточно 10 лет назад как пользователям, так и разработчикам. Сейчас существует много инструментов, которые не только упрощают разработку, но и позволяют применять эффективные подходы к разработке, а также повышают отзывчивость интерфейса даже на тех устройствах, которые уже «не поддерживаются».

Далее будет проведен анализ некоторых популярных фреймворков (библиотек) для веб-разработки. В конечном счете, будет выбран один, который идеально подходит для реализации вышеописанной инфраструктуры и будет соответствовать ряду критериев, которые будут определены ниже.

В качестве рассматриваемых фреймворков (библиотек) для веб-приложения было решено взять следующие:

* Angular.js
* Vue.js
* React.js
* Gatsby.js vs Next.js
* Svelte

# Angular.js

Angular – это JavaScript Фреймворк, реализующий паттерн MVVM (Model-View-ViewModel), основанный в 2009 году, отлично подходит для создания интерактивных веб-приложений.

Преимущества Angular:

* наличие новых функций, таких как Generation Angular, использующих библиотеки NPM из CLI, Generation;
* наличие подробной документации, позволяющей разработчику получить всю необходимую информацию, не прибегая к помощи его коллег. Однако это и своеобразный недостаток – требует больше времени для обучения;
* односторонняя привязка данных обеспечивает исключительное поведение приложения, что сводит к минимуму риск возможных ошибок;
* MVVM позволяет разработчикам работать отдельно над одним и тем же разделом приложения, используя один и тот же набор данных;
* внедрение зависимостей от компонентов, связанных с модулями и модульностью в целом;
* структура и архитектура, специально созданные для большой масштабируемости проекта.

Недостатки Angular:

* разнообразие различных структур (Injectables, Components, Pipes, Modules и т. д.) усложняет процесс изучения по сравнению с ним же в React и Vue.js, у которых есть только «Component»;
* относительно медленная производительность, учитывая различные показатели.

Компании, использующие Angular: Microsoft, Autodesk, McDonald’s, UPS, Cisco Solution Partner Program, AT&T, Apple, Adobe, GoPro, Clarity Design System, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase

# React.js

React.js – это JavaScript библиотека, разработанная Facebook в 2013 году, отлично подходит для создания современных одностраничных приложений любого размера и масштаба.

Преимущества React:

* легко изучить благодаря простому дизайну, использованию JSX или TSX (HTML-подобный синтаксис) для шаблонов и очень подробной документации. Разработчики тратят больше времени на написание современного JavaScript и меньше беспокоятся о коде, специфичном для фреймворка;
* очень быстрый благодаря реализации React Virtual DOM и различным оптимизациям рендеринга;
* отличная поддержка рендеринга на стороне сервера, что делает его мощной платформой для контент-ориентированных приложений;
* первоклассная поддержка Progressive Web App (PWA) благодаря генератору приложений «Create-React-App» (CRA)
* привязка данных является односторонней, что приводит к уменьшению количества нежелательных побочных эффектов;
* Redux – самая популярная платформа для управления состоянием приложений в React, её легко учить и использовать;
* React реализует концепции функционального программирования (далее FP), создавая простой в тестировании и многократно используемый код;
* приложения могут быть созданы с помощью TypeScript или Facebook’s Flow, имеющими встроенную поддержку JSX;
* переход между версиями, как правило, очень прост: Facebook предоставляет «кодовые модули» для автоматизации большей части процесса.

Недостатки React:

* React не однозначен и оставляет разработчикам возможность выбирать лучший способ развития. Это может быть решено сильным лидерством проекта и хорошими процессами;
* сообщество делится по способам написания CSS в React, которые разделяются на традиционные таблицы стилей (CSS Modules) и CSS-in-JS (то есть Emotion и Styled Components);
* React отходит от компонентов на основе классов, что может стать препятствием для разработчиков, которым более комфортно работать с объектно-ориентированным программированием (ООП);
* смешивание шаблонов с логикой (JSX/TSX) может сбить с толку некоторых разработчиков при первых знакомствах с React.

Компании, использующие React: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft, Slack, Storybook и многие другие

# Vue.js

Vue.js – это JavaScript фреймворк, основанный в 2013 году, который идеально подходит для создания высокоадаптируемых пользовательских интерфейсов и сложных одностраничных приложений.

Преимущества Vue.js:

* усиленный HTML. Это означает, что Vue.js имеет много характеристик схожих с Angular, а это, благодаря использованию различных компонентов, помогает оптимизации HTML-блоков;
* Vue.js имеет очень подробную документацию, которая может ускорить процесс обучения для разработчиков и сэкономить много времени на разработку приложения, используя только базовые знания HTML и JavaScript;
* дизайн и архитектура Vue.js дизайн и архитектуру Angular и React;
* Vue.js можно использовать как для создания одностраничных приложений, так и для более сложных веб-интерфейсов приложений. Важно, что небольшие интерактивные элементы можно легко интегрировать в существующую инфраструктуру без негативных последствий;
* Vue.js может помочь в разработке довольно больших шаблонов многократного использования, которые могут быть сделаны почти за тоже время, что и более простые;
* Vue.js весит около 20 КБ, сохраняя при этом свою скорость и гибкость, что позволяет достичь гораздо лучшей производительности по сравнению с другими фреймворками.

Недостатки Vue.js:

* Vue.js занимает довольно небольшую долю рынка по сравнению с React или Angular, поэтому сложно эффективно обмениваться знаниями и опытом.

Компании, которые используют Vue.js: Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab и Laracasts, Adobe, Behance, Codeship, Reuters

# Next.js

Проведение сравнительного анализа преимуществ и недостатков, существующих на сегодняшний день популярных фреймворков и библиотек позволяет сделать выбор в пользу разработки с использованием React.js.

Как было сказано ранее, React.js – библиотека, позволяющая только разрабатывать пользовательские интерфейсы. Для того, чтобы использовать технологии интернационализации, SSR и SEO, необходим отдельный фреймворк, реализующий эти функции.

«Поверх» React.js работает ограниченное число фреймворков: CRA и Next.js. Проведём сравнительный анализ.

CRA содержит настроенную среду для разработки SPA-приложений, но слабо поддерживает работу с SSR и SEO.

Отличительными особенностями фреймворка Next.js является не только возможность эффективной работы с SSR и SEO, но и ряд других преимуществ, рассмотрим их подробнее:

* эффективная работа с DOM-деревом, так как в основе его лежит React.js;
* «ленивая» загрузка картинок, которая позволяет не «нагружать» устройство пользователя, когда в этом нет необходимости, например, когда картинка находится за пределами экрана пользователя;
* поддержка интернационализации, которая эффективно реализуется с помощью клиентского роутинга;
* настроенная среда, необходимая для сборки приложения;
* поддержка аналитики веб-приложения;
* поддержка SSR;
* поддержка гибкой настройки SEO.

Таким образом, возможности Next.js позволяют построить требуемую инфраструктуру, описанную ранее в Главе 1, удовлетворяющую перечисленным ранее критериям.

**Gatsby.js**

Gatsby — это современный веб-фреймворк, построенный на основе React и GraphQL. Основное внимание фреймворк уделяет встроенной производительности: по умолчанию он создает молниеносно быстрые сайты. Он создает статичную сборку, чтобы сделать сайт быстрее. Это одна из причин, почему Gatsby часто упускают из виду как еще один генератор статичных сайтов.

Несмотря на то, что Gatsby построен на основе React, у него есть собственная экосистема, включающая плагины, темы и стартеры. Gatsby расширяемый по умолчанию. Он создается как статичный сайт во время сборки и размещается как простые HTML-страницы.

Преимущества

* Может использоваться для создания статических сайтов и PWA, которые соответствуют актуальным веб-стандартам.
* Gatsby.js использует новейшие технологии, включая React.js, Webpack, GraphQL, современные ES6 + JavaScript и CSS.
* Возможность использовать любой из пакетов, которые уже использовали с NPM, и большое количество плагинов.
* Статический контент, сгенерированный Gatsby.js, легко читается поисковыми системами.
* Богатая и обширная коллекция плагинов.

Недостатки

* Необходимы знания React и GraphQl.
* Не является хорошим решением для создания больших сайтов корпоративного масштаба.

1. Сранение Gatsby.js и Next.js

Gatsby.js - это генератор статических сайтов. Генератор статического сайта генерирует статический HTML во время сборки. Он не использует сервер.

Next.js - это в основном инструмент для рендеринга страниц на стороне сервера. Он динамически генерирует HTML каждый раз, когда поступает новый запрос с использованием сервера.

Gatsby.js просто генерирует чистый HTML/CSS/JS в build time, тогда как Next.js создает HTML/CSS/JS в run time. Таким образом, каждый раз, когда приходит новый запрос, он создает новую HTML-страницу с сервера.

Несмотря на то, что они решают разные проблемы, Gatsby.js и Next.js имеют много общего.

Сложность обучения для фреймворков не очень высокая, если вы уже знаете, как создавать сайты на основе React, ведь опыт разработчиков стоит на первом месте для этих фреймворков. Мы можем настроить и запустить все очень быстро, а добавление новых функций к простым приложениям происходит легко, потому что оба фреймворка предлагают исчерпывающую документацию. Оба фреймворка поддерживают горячую перезагрузку (Hot reloading) для более быстрой разработки.

Кэширование и производительность встроены. Нам не нужно беспокоиться о разделении и оптимизации кода: это работает, и по умолчанию они кодируют разделение (Code splitting) на основе каждого маршрута страницы. Оба фреймворка имеют встроенную маршрутизацию для создания новых страниц.

Они выполняют интеллектуальную загрузку страниц путем асинхронной предварительной загрузки ссылок для следующих страниц при прокрутке страницы. Оценка Lighthouse для хорошо построенных сайтов Gatsby.js и Next.js будет выдающейся.

Вместо теоретических плюсов и минусов, которые вы можете легко найти в Интернете, мы используем другой подход и определим, какой фреймворк будет лучшим выбором, исходя из рассматриваемой проблемы.

**Выводы**

Каждый из фреймворков особенный, поэтому мы выберем лучший вариант между Gatsby и Next.js для этого случаев использования:

Благодаря учебному руководству и краткости документации по Next.js возникает такое ощущение, что этот фреймворк можно изучить быстрее, чем Gatsby.js.

В основе системы загрузки данных, используемой в Next.js, лежат асинхронные функции и API Fetch. В результате при освоении Next.js у разработчика не возникает ощущения того, что ему необходимо изучить GraphQL для того чтобы пользоваться возможностями фреймворка на полную мощность.

В Next.js есть встроенная поддержка TypeScript, в то время как в Gatsby то же самое реализуется с помощью особого плагина, который даже не выполняет проверку типов (для этого нужен отдельный плагин).

При переводе проекта на Next.js это привело к появлению некоторых проблем, выражавшихся в виде ошибок компиляции, так как я даже не знал о том, что в проекте не всё благополучно с типами.

Благодаря тому, что в Next.js улучшена поддержка SSG, этот фреймворк является мощным инструментом, позволяющим, на уровне каждой отдельной страницы, выбирать метод работы с ней. Это может быть SSR, SSG или CSR.

Если у нас много контента или мы ожидаем что наш контент будет расти со временем, статические веб-страницы — не лучшее решение для вас.

Так что, если у вас есть сайт с контентом, который, как мы ожидаем, будет расти и расти со временем, тогда Next.JS — лучший выбор для нас.

Кроме того, если вам нужна большая свобода доступа к вашим данным, стоит также выбрать Next.JS.

# Svelte

Svelte — это принципиально новый подход к созданию пользовательских интерфейсов. Когда традиционные фреймворки, такие как React и Vue, выполняют основную часть своей работы в браузере, Svelte переносит эту работу на этап компиляции, который происходит при сборке приложения.

Вместо использования таких тяжелых приёмов, как сравнение состояний виртуального DOM, Svelte пишет код, который точечно обновляет DOM при изменении состояния вашего приложения.

Svelte может быть идеальным фреймворком для создания веб-компонентов, но в настоящее время есть некоторые вещи, которые сдерживают его

Пока другие фреймворки занимаются поддержкой интерфейса в актуальном состоянии и выполняют другие действия прямо в браузере. Svelte же делает своё дело во время сборки проекта. В этом и заключается основная разница между другими фреймворками и Svelte

Вот что об этом сказано в документации по Svelte:  
«Svelte конвертирует ваше приложение в идеальный JavaScript-код во время сборки проекта, а не интерпретирует код приложения во время его работы».

Svelte не использует техник наподобие сравнения реальной и виртуальной DOM.

Компилятор Svelte берёт декларативный компонент и превращает его в эффективный императивный низкоуровневый код, который напрямую взаимодействует с DOM.

Так как Svelte использует компилятор, а не виртуальную DOM, это позволяет снизить нагрузку на браузер, повысить скорость работы страниц, облегчить JS-движку браузера сборку мусора. Браузеру не приходится решать некоторые задачи во время выполнения кода страницы, что улучшает производительность проектов.

Svelte — пока еще слишком молодая технология, но она уже показывает огромный потенциал, но не стоит забывать, что, когда React только появился, он тоже выглядел инструментом без каких-либо недостатков, но по мере развития и его внедрения появились различные непростые ситуации для которых приходится искать обходные пути и компромиссы, но, как минимум, стоит внимательно наблюдать за разработкой библиотеки Svelte.

# Глава 2

1. Структура проекта

**Страница** — это компонент React, который экспортируется из файла с расширением .js, .jsx, .ts или .tsx, находящегося в директории pages.

Каждая страница ассоциируется с маршрутом (роутом) по названию. Например, страница pages/Students.js будет доступна по адресу /Students. Обратите внимание, что страница должна экспортироваться по умолчанию (export default).

Маршрут для страницы pages/posts/[id].js будет динамическим, т.е. такая страница будет доступна по адресам posts/1, posts/2 и т.д.

По умолчанию все страницы рендерятся предварительно (pre-rendering). Это приводит к лучшей производительности и SEO. Каждая страница ассоциируется с минимальным количеством JS. При загрузке страницы запускается JS-код, который делает ее интерактивной (данный процесс называется гидратацией — hydration).

Существует 2 формы предварительного рендеринга: генерация статической разметки (static generation, SSG, рекомендуемый подход) и рендеринг на стороне сервера (server-side rendering, SSR). Первая форма предусматривает генерацию HTML во время сборки и его повторное использование при каждом запросе. Вторая — генерацию разметки при каждом запросе. Генерация статической разметки является рекомендуемым подходом по причинам производительности.

Кроме этого можно использовать рендеринг на стороне клиента (client-side rendering), когда определенные части страницы рендерятся клиентским JS.

**Получение данных (Data Fetching)**

Существует 3 функции для получения данных, необходимых для предварительного рендеринга:

* getStaticProps (SSG): получение данных во время сборки
* getStaticPaths (SSG): определение динамических роутов для предварительного рендеринга страниц на основе данных
* getServerSideProps (SSR): получение данных при каждом запросе

Добавление стилей на уровне компонента

Next.js из коробки поддерживает CSS-модули. CSS-модули должны иметь название [name].module.css. Они создают локальную область видимости для соответствующих стилей, что позволяет использовать одинаковые названия классов без риска возникновения коллизий. CSS-модуль импортируется как объект (обычно, именуемый styles), ключами которого являются названия соответствующих классов.

При сборке CSS-модули конкатенируются и разделяются на отдельные минифицированные CSS-файлы, что позволяет загружать только необходимые стили.

**Компонент Image и оптимизация изображений**

Компонент Image, импортируемый из next/image, является расширением HTML-тега img, предназначенным для современного веба. Он включает несколько встроенных оптимизаций, позволяющих добиться хороших показателей Core Web Vitals. Эти оптимизации включают в себя следующее:

* улучшение производительности
* обеспечение визуальной стабильности
* ускорение загрузки страницы
* обеспечение гибкости (масштабируемости) изображений

**Обновление в режиме реального времени**

Next.js поддерживает обновление компонентов в режиме реального времени с сохранением локального состояния в большинстве случаев (это относится только к функциональным компонентам и хукам). Состояние компонента также сохраняется при возникновении ошибок (не связанных с рендерингом).

**Оптимизация шрифтов**

Next.js автоматически встраивает шрифты в CSS во время сборки:

Для добавления на страницу шрифта используется компонент Head, импортируемый из next/head

Для добавления шрифта в приложение следует создать кастомный документ.

**Обслуживание статических файлов**

Статические ресурсы должны размещаться в директории public, находящейся в корневой директории проекта. Файлы, находящиеся в директории public, доступны по базовой ссылке /:

Данная директория также отлично подходит для хранения таких файлов, как robots.txt, favicon.png, файлов, необходимых для верификации сайта Гуглом и другой статики (включая .html).

**TypeScript**

Next.js поддерживает TypeScript из коробки:

**Маршрутизация**

Маршрутизация в Next.js основана на концепции страниц.

Файл, помещаемый в директорию pages, автоматически становится роутом.

Файлы index.js привязываются к корневой директории:

* pages/index.js -> /
* pages/blog/index.js -> /blog

Роутер поддерживает вложенные файлы:

* pages/blog/first-post.js -> /blog/first-post
* pages/dashboard/settings/username.js -> /dashboard/settings/username

Динамические сегменты маршрутов определяются с помощью квадратных скобок:

* pages/blog/[slug].js -> /blog/:slug (blog/first-post)
* pages/[username]/settings.js -> /:username/settings (/overfrenzy/settings)
* pages/post/[...all].js -> /post/\* (/post/2021/id/title)

**Связь между страницами**

Для маршрутизации на стороне клиента используется компонент Link:

**import** Link **from** 'next/link'

**export** **default** **function** **Home**() {

**return** (

<**ul**>

<**li**>

<**Link** href="/">

Главная

</**Link**>

</**li**>

<**li**>

<**Link** href="/about">

О нас

</**Link**>

</**li**>

<**li**>

<**Link** href="/blog/first-post">

Пост номер раз

</**Link**>

</**li**>

</**ul**>

)

}

* / → pages/index.js
* /about → pages/about.js
* /blog/first-post → pages/blog/[pid].js

Для динамических сегментов можно использовать интерполяцию:

**import** Link **from** 'next/link'

**export** **default** **function** **Post**({ posts }) {

**return** (

<**ul**>

{posts.map((post) => (

<**li** key={post.id}>

<**Link** href={`/blog/${encodeURIComponent(post.pid)}`}>

{post.title}

</**Link**>

</**li**>

))}

</**ul**>

)

}

Перехват всех путей

Динамические роуты могут быть расширены для перехвата всех путей посредством добавления многоточия (...) в квадратные скобки. Например pages/post/[...pid].js будет совпадать с /post/a, /post/a/b, /post/a/b/c и т.д.

Обратите внимание: вместо slug можно использовать любое другое название, например, [...param].

Совпавшие параметры передаются странице как параметры строки запроса (pid в данном случае) со значением в виде массива. Например, query для /post/a будет иметь такую форму:

{ "pid": ["a"] }

А для /post/a/b такую:

{ "pid": ["a", "b"] }

Роуты для перехвата всех путей могут быть опциональными — для этого параметр необходимо обернуть еще в одни квадратные скобки ([[...pid]]).

**Кеширование**

Кеширование уменьшает время ответа и количество запросов к внешним сервисам. Next.js автоматически добавляет заголовки кеширования к иммутабельным ресурсам, доставляемым из \_next/static, включая JS, CSS, изображения и другие медиа.

Для ревалидации кеша страницы, которая была предварительно отрендерена в статическую разметку, используется настройка revalidate в функции getStaticProps.

**MUI**

material-ui — reactJS's фрэймворк, предоставляющий готовые google решения для быстрой и довольно простой web разработки.

Material-UI достаточно крупная библиотека, где ключевой частью react компонентов и стилизации является @mui/material

|- pages

|- <page\_name>

|- \_app.tsx # Next.js использует компонент App для инициализации страниц.

|- index.tsx # Это корневая страница, которая отображается первой

|- students-2005.tsx # Страница с данными участников олимпиады 2005 года

|- students-2016.tsx # Страница с данными участников олимпиады 2005 года

|- ... # Страница с данными участников олимпиады 20xx годов

|- students-2021.tsx # Страница с данными участников олимпиады 2021 года

|- students.tsx # Страница для выбора участников по годам

|- tuy-2005.tsx # Страница с данными олимпиады 2005 года

|- tuy-2016.tsx # Страница с данными олимпиады 2016 года

|- ... # Страница с данными олимпиады 20xx годов

|- tuy-2021.tsx # Страница с данными олимпиады 2021 года

|- components

|- AppBar1.js # Верхняя панель в синем цвете

|- AppBar2.js # Верхняя панель в фиолетовом цвете

|- public

|- favicon16.png # Иконка приложения

|- shared

|- data2005.json # Данные о участников 2005 года

|- data2016.json # Данные о участников 2016 года

|- ... # Данные о участников 20xx годов

|- data2021.json # Данные о участников 2021 года

|- styles

|- index.module.css # CSS для главного компонента

|- tables.module.css # CSS для таблиц

Students-2005.tsx берет данные из data2005.json:

**export** async **function** getStaticProps() {  
 const filePath = path.join(process.cwd(), "./shared/data2005.json");  
 const jsonData = await fsPromises.readFile(filePath);  
 const objectData = JSON.parse(jsonData.toString());

Appbar подключается:

**import** AppBar **from** '../components/AppBar1';  
<AppBar />

favicon16.png подключается в \_app.tsx

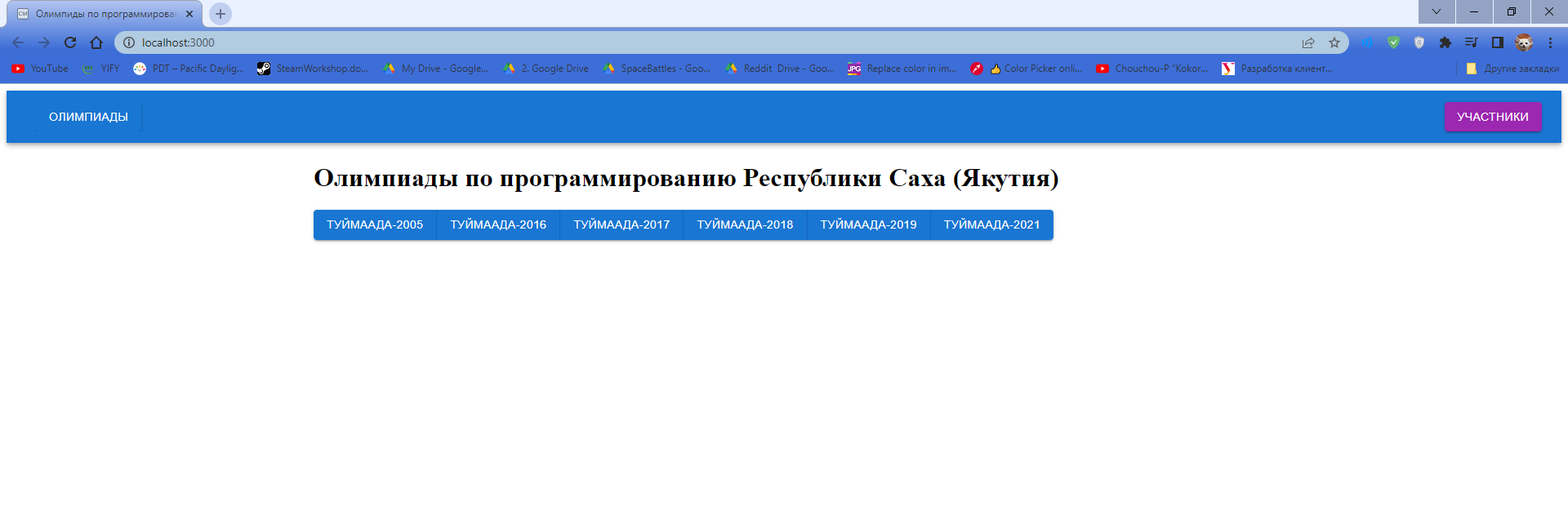
<link rel="icon" href="/favicon16.png" />

CSS стили подключаются:

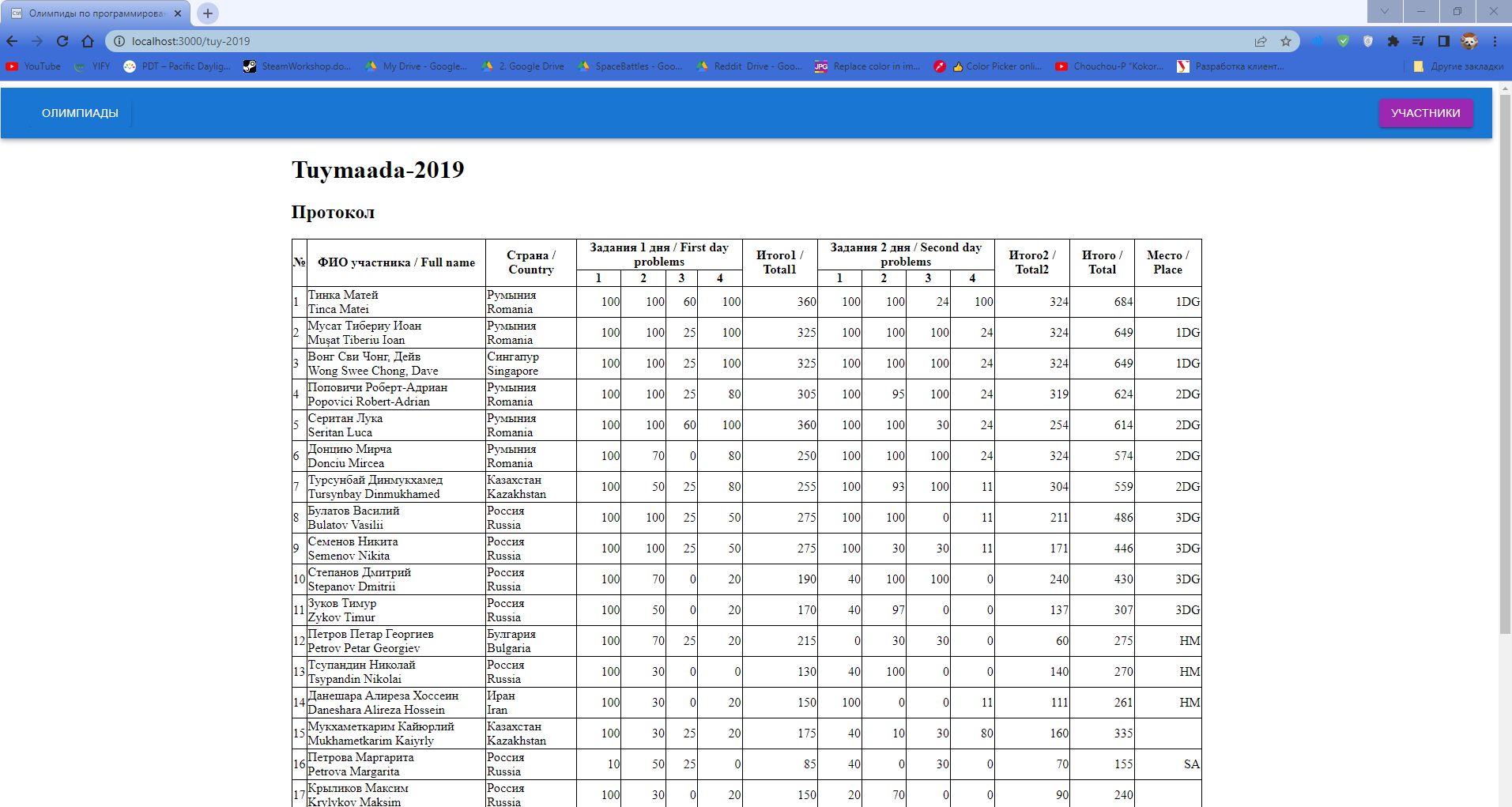
**import** styles **from** '../styles/index.module.css';

* 1. Интерфейс сайта/дизайн

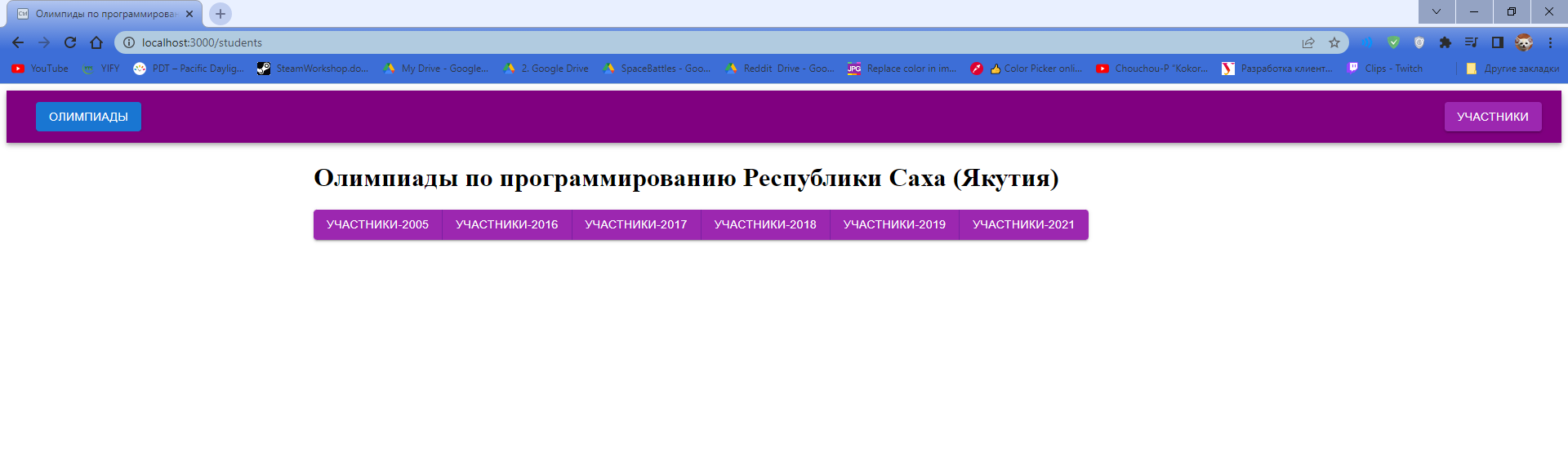
Для дизайна этого сайта была использована библиотека Material-UI



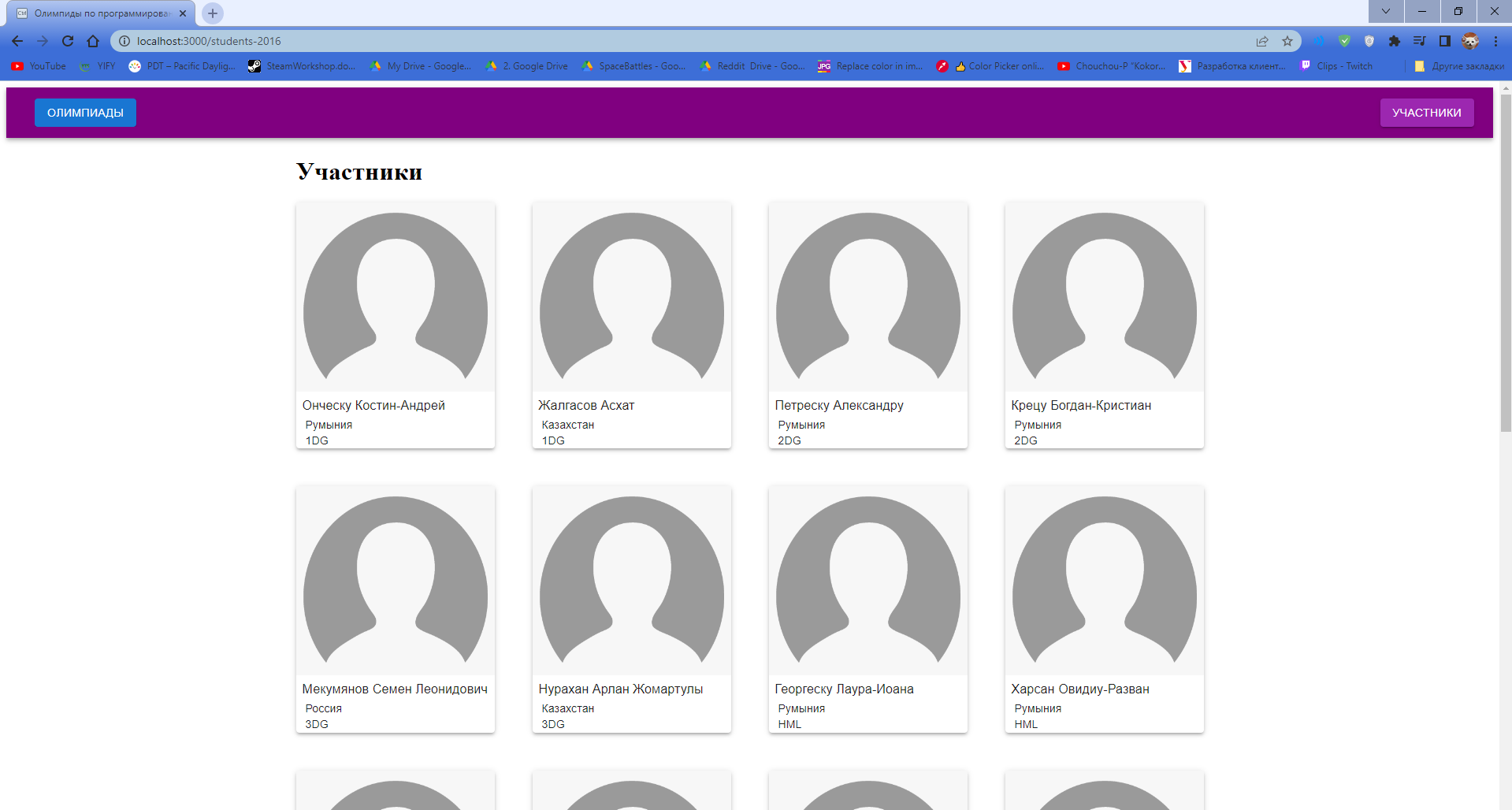
Главная страница



Таблицы с данными об олимпиаде



Страница участников



Страница с данными об участниках

# Заключение

Целью нашей работы была разработка веб-сайта олимпиад по программированию Республики Саха

Результатом работы стала демонстрация практики создания веб-приложения, собранная на основе современных тенденций в мире веб-разработки, а также были рассмотрены достоинства и недостатки существующих популярных инструментов и подходов.

В ходе данной курсовой работы был представлен сравнительный анализ фреймворков для разработки веб-приложений с разбором преимуществ и недостатков, исходящих из целей поставленной нами задачи.

Исходя из них же, нами был аргументирован и выбран фреймворк Next.js

Помимо этого, были реализованы и другие задачи проекта, а именно:

* Проектирование инфраструктуры сайта олимпиад;
* Разработка веб-приложения согласно требованиям

Таким образом, можно сделать вывод, что цель работы, а именно – реализация сайта олимпиад по программированию Республики Саха была достигнута в полном объёме.

# Библиографический список

1. Документация по TypeScript c официального сайта:  
   URL: <https://www.typescriptlang.org>  
   (дата обращения: 14.06.2022)
2. Документация по Angular c официального сайта:  
   URL: <https://angular.io>  
   (дата обращения: 14.06.2022)
3. Документация по Vue.js  
   URL: <https://vuejs.org>  
   (дата обращения: 14.06.2022)
4. Документация по React c официального сайта:  
   URL: <https://ru.reactjs.org>  
   (дата обращения: 14.06.2022)
5. Документация по Next.js c официального сайта:  
   URL: <https://nextjs.org>  
   (дата обращения: 14.06.2022)
6. Документация по Gatsby.js c официального сайта:  
   URL: <https://gatsbyjs.com>  
   (дата обращения: 14.06.2022)
7. Официальный сайт Material UI  
   URL: <https://mui.com>  
   (дата обращения: 14.06.2022)

# Глоссарий

**Фреймворк** – (от англ. framework – каркас, структура) – это набор библиотек, который облегчает разработку любых продуктов: веб-сайтов и веб-сервисов, мобильных или десктопных приложений.

**JS** – (сокращение от JavaScript [1]) – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

**DOM** – (от англ. Document Object Model – объектная модель документа) – это независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-документов, а также изменять содержимое, структуру и оформление таких документов.

**SSR** – (от англ. Server-Side Rendering – серверный рендеринг) – рендеринг на сервере клиентской части или универсального приложения в HTML.

**SEO** – (от англ. search engine optimization – поисковая оптимизация) – комплекс мероприятий по внутренней и внешней оптимизации для поднятия позиций сайта в результатах выдачи поисковых систем по определённым запросам пользователей, с целью увеличения сетевого трафика, потенциальных клиентов и последующей монетизации этого трафика. SEO может быть ориентировано на различные виды поиска, включая поиск информации, товаров, услуг, изображений, видеороликов, новостей и специфические отраслевые поисковые системы

**TypeScript** – язык программирования, позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере или использовать совместно с серверной платформой Node.js. TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов, а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода.

**NPM** – (от англ. Node Package Manager). Представляет собой менеджер пакетов, входящий в состав Node.js.

**CLI** – (от англ. Command Line Interface – интерфейс командной строки) – разновидность текстового интерфейса между человеком и компьютером, в котором инструкции компьютеру даются в основном путём ввода с клавиатуры текстовых строк, а в UNIX-системах возможно применение мыши.

**Объектно-ориентированное программирование** (ООП) – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Google **Lighthouse** — это инструмент аудита с открытым исходным кодом, который помогает разработчикам повысить производительность и доступность своих веб-проектов.