**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой РК6

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П. Карпенко

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Студент группы РК6-43Б

Рогов Матвей Валерьевич

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

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

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к \_\_\_ нед., 50% к \_\_\_ нед., 75% к \_\_ нед., 100% к \_\_\_ нед.

***Техническое задание \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

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

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

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

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

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**Руководитель НИР**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Рогов М.В.**

(Подпись, дата) (И.О.Фамилия)

**Оглавление**

1. Реферат
2. Понятие и история Web-приложения
3. Структура Web-приложения
   1. Общая структура серверной части (Backend)
   2. Структура клиентской части (Frontend)
4. Преимущества и недостатки онлайн программирования
5. Особенности web-приложения для онлайн программирования
6. Технологии, необходимые для реализации данного web-приложения
   1. Технологии для реализации backend
   2. Технологии для реализации frontend
7. Итоговый выбор технологий для выполнения практического задания
8. Использованная литература

**Реферат**

Расчётно-пояснительная записка: 18 страниц, 0 таблиц, 3 рисунка, 9 источников

Цель работы: Выбор наиболее подходящих технологий для выполнения практического задания по теме “Разработка web-приложения для онлайн программирования”

Данная работа посвящена поиску оптимального набора web-технологий для практического задания. Для выполнения цели работы было приведено определение web-приложения и были разобраны самые часто используемые web-технологии. Так же были отобраны наиболее подходящие технологии для выполнения практического задания по теме “Разработка web-приложения для онлайн программирования”. Данные технологии были использованы при выполнении практического задания.

**Введение**

В данной работе рассмотрено понятие web-приложения и разобраны типы таких приложений. Проблема онлайн программирования актуальна и по сей день, поскольку для того, чтобы писать код большой командой и чтобы структура всех используемых файлов не менялась, программистам необходимо держать постоянную связь друг с другом и просматривать все файлы для контроля всех действий. На сегодняшний день существует система контроля версий Git, которая позволяет контролировать структуру всех файлов и изменять некоторые из них без нарушения общей структуры проекта. Но она не позволяет делать это в онлайн режиме, т.е. нет возможности следить за изменением кода в режиме реального времени. Web-приложение, описанное в данной работе, позволит следить за изменениями в различных файлах и изменять код в этих файлах в режиме реального времени. Данная система позволит сократить время разработки любого ПО, поскольку сократится время, отведённое на проверку структур различных файлов данного ПО.

**Понятие и история web-приложения**

**Web-приложение** - клиент-серверное приложение, в котором клиент взаимодействует с web-сервисом при помощи браузера. Логика web-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, а обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому web-приложения являются межплатформенными службами.

В 1989 году Тим Бернерс-Ли изобрёл web 1.0. Эта технология использовалась в основном для создания сайтов в виде статичных страничек. В основном, такие сайты использовались как “визитки”, которые просто демонстрировали какую-то информацию для пользователей.

В 1999 года появилась технология web 2.0. На сегодняшний день эта технология определяется как сет сеть интерактивных web-сайтов и платформ, где контент производят пользователи, а не владелец ресурса, такие как Facebook, Twitter и т.д.

Далее была создана технология web 3.0, которая представляет собой web-инфраструктуру из нескольких основных технологий: блокчейна, машинного обучения и ИИ, семантической паутины и интернета вещей. Каждая из этих технологий - неотъемлемая часть будущей сети со своей ролью в экосистеме.

Последним шагом на сегодняшний день в этой сфере стало создание технологии web 4.0. Сегодня под Web 4.0 идет речь о мобильности и голосовом взаимодействии между пользователем и роботами. Раньше для взаимодействия с Интернетом пользователю приходилось находиться за рабочим столом перед компьютером, но мобильные устройства позволяют пользователю использовать и распространять информацию независимо от местонахождения.

**Структура web-приложения**

Web-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию “клиент-сервер”. В данном случае, технология “клиент-сервер” – это вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. И клиент и сервер – это программное обеспечение, в большинстве случаев расположенное на разных вычислительных машинах и взаимодействующее между собой через вычислительную сеть посредством сетевых протоколов.

**1. Общая структура серверной части (Backend)**

Серверная часть отвечает за обработку запросов, которые приходят со стороны клиентской части, формирование web-страницы после обработки запроса и отправку этой страницы обратно на клиентскую часть. Запросом будет являться практически всё, что пользователь будет делать в данном web-приложении, например открыть какую-то вкладку на этой странице, обновить данную страницу и т.д. Взаимодействие между клиентской и серверной частью происходит при помощи гипертекстового протокола передачи (HTTP). Запрос включает в себя затронутый ресурс (URL), т.е. где происходит данной действие, и метод, определяющий действие запроса (например, удалить что-то или изменить). Сайты с серверной стороны бывают 2 типов: статические и динамические.

Статические сайты – это те сайты, которые при каждом запросе клиента возвращают жёстко закодированное содержимое, т.е. что бы клиент не запросил, сервер сгенерирует всю страничку полностью заново и отправит её пользователю.

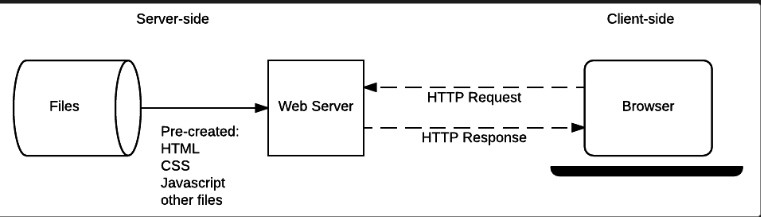


Рис. 1. Структура статического сайта

Динамические сайты – это те сайты, которые могут генерировать часть содержимого ответа только при необходимости.

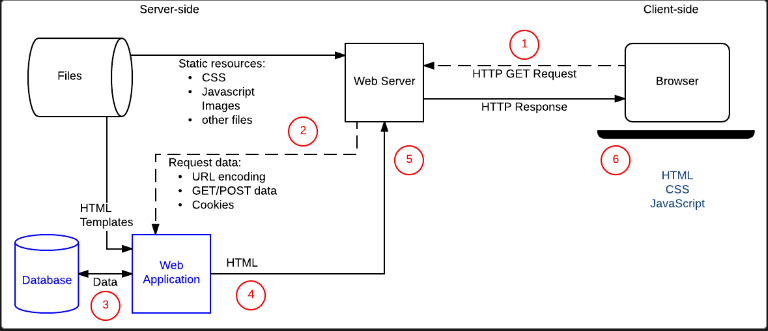


Рис. 2. Структура динамического сайта

Динамические сайты несомненно лучше статических в наше время, поскольку если на сайте огромное количество информации, необходимой для загрузки страницы, то при постоянном обновлении всех данных во время обработки запросов ни у одного сервера не хватит мощности для сохранения высокой скорость работы при большой нагрузке. Для реализации серверной части сайта могут используются языки программирования:

* Python (Framework Django)
* Java (Framework Spring)
* Go (Framework Gin)
* PHP (Framework Laravel)
* Ruby (Framework Ruby-on-rails)

И многие другие.

**2. Структура клиентской части (Frontend)**

Клиентская часть отвечает за отображение результатов запроса. Любой клиент, который находится на каком-либо сайте, взаимодействует с интерфейсом сайта, а именно нажимает на разные кнопки, ссылки, открывает новые вкладки и т.д. Это всё и есть клиентская часть сайта. В отличие от серверной части, клиентская не требует знания каких-то особо сложных алгоритмов, структур данных или специфичных реализация. В ней необходимы другие навыки и знания, такие как творческий подход к делу или умение встать на сторону другого, в данном случае на сторону пользователя, для понимания того, как сделать интерфейс более удобным и простым для понимания. Для реализации этой части сайта используются такие языки, как:

* HTML – Язык гипертекстовой разметки
* CSS – Язык описание внешнего вида документа
* JavaScript – язык программирования для создания интерактивных страниц

**Преимущества и недостатки онлайн программирования**

Сама по себе идея онлайн программирования заключается в том, что на сайте в режиме реального времени смогут писать код несколько человек в одном и том же окне, видя при этом изменения других пользователей и имея возможность исправить код, написанный другим пользователем. У данной системы есть как преимущества, так и недостатки.

Преимущества:

* Видны все изменения в коде в режиме реального времени, т.е. отпадает необходимость каким-либо образом сообщать своим собеседникам о твоих изменениях в том или ином месте
* Можно самому вносить правки в код собеседника, если в этом есть необходимость
* Есть возможность в любой момент времени сохранить этот код в облаке и скачать его оттуда сколь угодно раз в любой момент времени
* Не нужно скачивать специальные утилиты, в которых нужно писать и редактировать код
* При компиляции проекта вся нагрузка идёт на сервер, а не на вашу вычислительную машину

Недостатки:

* Небольшая вычислительная мощность в виду того, что обработка и компиляция кода происходит на сервере. В следствие этого не получится писать крупные проекты
* Возможность использования данной системы зависит от качества подключения к интернету

Проанализировав все преимущества и недостатки онлайн программирования можно прийти к выводу, что данная система подходит для обучения, поскольку её особенности, заключающиеся в том, что можно полностью контролировать действия своего собеседника, могут помочь преподавателю в обучении студентов программированию. Для разработки крупных проектов данная система не подходит, поскольку все процессы будут происходить на сервере, и при нынешних возможностях сервера просто не будут выдерживать столь больших нагрузок. Возможно, в будущем при улучшении качества оборудования получится создать сервер, который сможет производить любые вычисления без каких-либо потерь в производительности.

**Особенности web-приложения для онлайн программирования**

Как и у любого web-приложения, у данной системы будут какие-то свои особенности при её реализации. Первой отличительное особенностью данного приложения станет то, что в нём можно будет работать большому количеству людей в режиме реального времени. Это означает, что если реализовать данное приложение как любой простой сайт, т.е. сделать обновления данных на экране пользователя только при определённом запросе, то тогда, чтобы увидеть изменения в коде, которые внёс другой пользователь, вам будет необходимо постоянно обновлять страницу, что будет занимать немало времени. Чтобы решить данную проблему необходимо реализовать приложение так, чтобы обновление информации на экране пользователя происходило постоянно, без каких-либо на то команд от самого пользователя, чтобы сервер сам с определённой частотой обновлял информацию. Второй особенностью данной системы будет являться то, что в него будет встроено несколько компиляторов, чтобы написанную программу можно было протестировать прямо на сайте. Ведь если не встраивать компилятор внутрь данного приложения, то в нём, можно сказать, пропадает всякий смысл, так как без него он будет являться простым редактором текста. Также, на этом сайте будет реализована система, которая будет отслеживать отступы и табуляции при написании разных операторов, так как вручную выставлять большое количество отступов крайне неудобно.

**Технологии, необходимые для реализации данного web-приложения**

**1. Технологии для реализации backend**

Первой технологией в данном разделе станет *WebSoket*. WebSocket – это протокол двусторонней связи между клиентом и сервером, который позволяет установить постоянное соединение между браузером и сервером. Он используется для передачи данных в режиме реального времени, таких как чаты, игры, потоковое видео и другие приложения, где важна низкая задержка и быстрый обмен данными. WebSocket использует протокол HTTP для установки соединения и обмена сообщениями между клиентом и сервером, но в отличие от HTTP, WebSocket поддерживает двунаправленную связь, что позволяет серверу отправлять данные на клиентскую сторону без запроса от клиента. Первый запрос отправляется на сервер по протоколу HTTP – это запрос “рукопожатия”, направлен на установку соединения. Этот запрос вернёт положительный ответ с том случае, если пользователь ранее прошел проверку на подлинность или был внесён в белый список. После того, как произошло обновление соединения, протокол переключается с HTTP на WebSocket. Все данные при передаче могут быть разбиты на фрагменты, поэтому через этот формат можно отправить сообщение очень большого размера — например, изображение. В этом случае web-сокеты разбивают его на фреймы, каждый из которых содержит небольшой заголовок, в котором указаны длина и тип полезной нагрузки, а также информация о том, является ли этот кадр последним. С помощью WebSocket сервер может открывать соединения с несколькими клиентами — даже несколько соединений с одним и тем же клиентом. Затем он может отправить сообщение одному, нескольким или всем этим клиентам. Наконец, когда соединение можно закрыть, либо клиент, либо сервер могут отправить сообщение «закрыть».

Следующая технология, необходимая для реализации данного приложения – это *Django*. Django — это высокоуровневый framework для создания web-приложений на языке python. Он представляет множество инструментов и функций для упрощения и ускорения разработки web-приложений, включая механизмы аутентификации, маршрутизации URL, работу с базами данных и многое другое. Django также поддерживает шаблонизацию, что позволяет разработчикам создавать динамические web-страницы, используя HTML-шаблоны и язык шаблонов Django. ОН широко используется в различных проектах, от небольших сайтов до крупных web-приложений, таких как Instagram или Pinterest. Django работает на основе архитектуры Model-View-Controller, которая отделяет логику приложения от представления данных. В Django это реализовано через шаблоны, представления и модели:

* Шаблоны – это файлы HTML, которые определяют внешний вид web-страницы. Они могут содержать динамические данные, которые передаются из представлений.
* Представления – это функции Python, которые обрабатывают запросы от пользователей и возвращают данные для отображения на web-страницах. Они могут использовать модели для доступа к базе данных и получения необходимых данных.
* Модели – это классы Python, которые определяют структуру базы данных и её связи. Они позволяют разработчикам создавать, изменять и удалять записи в базе данных.

Django также предоставляет механизмы аутентификации и авторизации, которые позволяют ограничить доступ к определённым страницам и функциям приложения. Он также поддерживает маршрутизацию URL, что позволяет разработчикам определять, какие представления будут вызываться при обращении к определённым URL-адресам.

Следующей технологией стал *Django Channels*. Channels – это расширение Django, которое добавляет возможность обработки асинхронных запросов и взаимодействия вы режиме реального времени. Оно позволяет использовать протокол WebSocket и HTTP/2 для обмена сообщениями между клиентом и сервером, а также поддерживает механизмы работы с очередями и событиями. Django Channels позволяет создавать приложения, которые могут обрабатывать не только HTTP-запросы, но и другие протоколы, такие как WebSocket, MQTT, IRC и другие. Она также предоставляет API для работы с различными бэкэндами очередей сообщений, такими как Redis, RabbitMQ и другие. Django Channels может использоваться для создания чата, онлайн игр, видеоконференций и других приложений, которые требуют обмена сообщениями в режиме реального времени. Принцип работы Django Channels заключается в следующем:

1. Клиент отправляет запрос на сервер через WebSocket
2. Сервер получает запрос и передаёт его в соответствующий consumer (обработчик), который может быть написан на Python или любом другом языке программирования.
3. Consumer обрабатывает запрос и отправляет его обратно клиенту через WebSocket
4. При необходимости, consumer может отправлять сообщения другим клиентам или подписчикам на определённый канал

Также Django Channels поддерживает использование других протоколов, таких как HTTP, HTTP/2, WebSoket Secure и Server-Sent Events. В целом, Django Channels позволяет создавать масштабируемые и быстрые web-приложения, которые могут обрабатывать большое количество одновременных соединений и обеспечивать надёжную доставку сообщений.

**2. Технологии для реализации frontend**

**Первой технологией в данном разделе станет *Bootstrap*. Bootsrtap – это бесплатный фреймворк для разработки web-сайтов и приложений, который содержит набор инструментов для создания адаптивных и мобильных сайтов. Он основан на HTML, CSS и JS и представляет разработчикам готовые компоненты, такие как кнопки, формы, таблицы, модели окна т.д., которые можно легко настроить и использовать в своих проектах. Bootstrap также предоставляет сетку для построения адаптивных меток, а также множество CSS классов для стилизации элементов.**

Классы в Bootstrap делятся на 3 большие группы:

* Для создания сетки – адаптивного макета страницы
* Для стилизации контента — текста, изображений, кода, таблиц и прочей информации
* Служебные — для решения популярных вспомогательных задач, таких как отображение и скрытие элементов, выравнивание текста на странице, настройка цвета фона, отступов и пр.

Кроме классов, в Bootstrap есть компоненты. Это хлебные крошки (путь от начального элемента до уровня иерархии, который сейчас просматривает пользователь), кнопки, выпадающие списки и подсказки и т.д.

Особенности данной технологии:

* **Адаптивный дизайн**

Bootstrap предоставляет мощную сетку для создания адаптивных макетов, которые могут легко адаптироваться к различным устройствам и экранам

* **Готовые компоненты**

Bootstrap содержит большой набор готовых компонентов, такие как кнопки, модели окна, формы и многое другое, которые можно легко настраивать по необходимости и использовать в своих проектах

* **Мощные JS плагины**

Bootstrap предоставляет мощные JS плагины, которые добавляют дополнительную функциональность к вашим сайтам и приложениям.

* **Лёгкая настройка**

Bootstrap позволяет легко настраивать и изменять стили компонентов с помощью CSS классов и переменных.

* **Поддержка браузеров**

Bootstrap поддерживает все современные браузеры и обеспечивает совместимость с более старыми версиями браузеров

Следующей технологией будет *React*. React – это JavaScript-библиотека, которая используется для создания пользовательских интерфейсов. Она была разработана FaceBook и используется в таких приложениях, как Instagram, WhatsApp. ReactJS позволяет создавать компоненты, которые могут быть повторно использованы, что упрощает процесс разработки и поддержки кода. Она также использует виртуальный DOM, что позволяет ей быстро обновлять только те элементы, которые изменились, вместо перерисовки всей страницы.

Особенности ReactJS:

* **Компонентный подход**

ReactJS позволяет создавать компоненты, которые могут быть повторно использованы в разных частях приложения. Это упрощает процесс разработки и поддержки кода

* **Виртуальное DOM-дерево**

ReactJS использует виртуальный DOM, что позволяет ей быстро обновлять только те элементы интерфейса, которые были изменены, вместе перерисовки всей страницы. Это делает ReactJS очень быстрым и эффективным для создания сложных пользовательских интерфейсов.

* **Нисходящий поток данных**

Компоненты могут передавать свойства и данные друг другу, но только в одном направлении — от родительских к дочерним. Это помогает реализовать четкую иерархию, облегчает отладку. Однонаправленный поток данных означает, что программист всегда может понять, откуда именно к элементу поступили данные.

Еще одна особенность потока данных — отсутствие возможности изменять свойства напрямую. Для этого существуют специальные callback-функции. Так называются части исполняемого кода, которые передаются от одного компонента другому в виде параметра функции. Это позволяет поддерживать стабильность: свойства оказываются неизменяемыми после создания, а callback-функция как бы пересоздает их. В результате использования коллбэков поток данных выглядит так: свойства и информация передаются сверху вниз, а события происходят снизу вверх.

* **Синтаксис JSX**

JSX расшифровывается как JavaScript XML. Это расширение языка JavaScript, которое помогает описывать HTML-подобные элементы с помощью кода на React. С помощью синтаксиса на React создают компоненты страницы и гибко управляют ими.

Несмотря на то что элементы похожи на HTML, это по-прежнему язык JavaScript с возможностью быстро и легко изменять DOM с помощью кода. И все же JSX воспроизводится как HTML: по сути разработчик описывает нужный компонент на языке разметки, а тот остается JavaScript-объектом с широкой функциональностью. Это удобно, упрощает программирование, но может запутать начинающих.

* **Модульность**

ReactJS позволяет разбить приложение на модули, что делает его более организованным и удобным для разработки и поддержки

**Итоговый выбор технологий для выполнения практического задания**

Основные параметры, которые будут реализованы в практике:

1. Backend часть
   1. Двусторонняя связь сервера и клиента
   2. Возможность написания кода в определённом окне
2. Frontend часть
   1. Гипертекстовая разметка страницы
   2. Стиль оформления внешнего вида документа

Проанализировав параметры, которые будут реализованы в практическом задании, можно составить конечный список технологий, которые будут использоваться при выполнении самого практического задания:

1. Django framework – framework для написания Backend части
2. Channels (WebSoket) – библиотека, позволяющая реализовать дуплексную связь сервера и клиента
3. Bootstrap – библиотека, позволяющая быстро создать адаптивный дизайн сайта

Данный стек технологий позволит полностью реализовать поставленное практическое задание

**Заключение**

В ходе работы были реализованы все поставленные задачи. Были исследованы самые часто используемые web-технологии. Были отобран необходимый стек технологий, который позволит реализовать web-приложения для онлайн программирования

1. Django framework
2. Channels (WebSoket)
3. Bootstrap

Список опущенных в данном НИР параметров:

1. React.js

Сочетание всех выбранных технологий позволит в полной мере создать web-приложения для онлайн программирования

**Используемая литература**

* <https://ru.hexlet.io/blog/posts/chto-takoe-websocket-i-kak-oni-voobsche-rabotayut>
* <https://channels.readthedocs.io/en/stable/index.html>
* <https://github.com/vasanthk/how-web-works#parsing-basics>
* <https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5>
* <https://forklog.com/sp/web3-0/theory/>
* <https://vc.ru/u/739868-andrey-nikolaev/238951-evolyuciya-do-web-4-0-kratkaya-istoriya-razvitiya-internet-tehnologiy-web-1-0-web-2-0-web-3-0-preimushchestva-web-4-0>
* <https://tutorial.djangogirls.org/ru/django/>
* <https://blog.skillfactory.ru/glossary/bootstrap/>
* <https://blog.skillfactory.ru/glossary/react/>