Государственное учреждение образования

«Средняя школа № 9 г. Орши»

исследовательская работа

**«**Исследование возможностей стека MERN

для создания видеохостинга**»**

#### Автор:

**Хренков Даниил Денисович**,

*учащийся X класса*

*ГУО ”Средняя школа № 9 г. Орши“*

**Руководитель:**

**Пурышкина Елена Викторовна,**

*учитель информатики*

*ГУО ”Средняя школа № 9 г. Орши“*

г. Орша

2022

Содержание

[Введение 3](#_Toc99310430)

[Основная часть 4](#_Toc99310431)

[Заключение 10](#_Toc99310432)

[Список использованных источников 11](#_Toc99310433)

[Приложение 12](#_Toc99310434)

# Введение

Каждый человек хотя бы раз в жизни пользовался видеохостингом. Видеохостинг – это онлайн-платформа, на которой можно размещать свои видеоролики. Это такие сервисы как [YouTube](https://www.youtube.com/), [Vimeo](https://vimeo.com/), [Dailymotion](https://www.dailymotion.com.video/) и другие. У каждого из этих сервисов есть свои преимущества и недостатки, но, на мой взгляд, главный недостаток всех этих ресурсов – наличие большого количества рекламы, которая отвлекает и тратит наше драгоценное время. Особую актуальность тема исследования приобрела в связи с эпидемиологической обстановкой, так как резко встал вопрос о дистанционном сопровождении образовательного процесса, поэтому я решил изучить данную тему и попробовать создать школьный видеохостинг, который позволил бы педагогам нашей школы реализовать интерактивное взаимодействие учеников и учителей. Взять для изучения я решил стек MERN, как наиболее популярный.

**Цель исследования**: изучить возможности стек технологий для создания full-stack приложения.

**Гипотеза**: стек MERN обладает необходимыми возможностями для созданиявеб-приложений, которые состоят из интерфейсных, серверных компонентов и компонентов базы данных.

**Задачи:**

1. Изучить технологии для создания веб-приложений.
2. Создать на базе стека MERN платформу для школьного видеохостинга.

# Основная часть

Стек MERN, который я решил использовать, - это JavaScript-стек. Он включает в себя четыре компонента с открытым исходным кодом: MongoDB, Express, React и Node.js[1]. Эти компоненты обеспечивают комплексную среду для работы разработчиков.

Перед началом разработки любого продукта, необходимо понимать, какой функционал нам нужно реализовать, делаем ли мы конечный продукт для полноценного использования. Исходя из этого, необходимо сосредоточиться на качестве и скорости работы или MVP (*Minimal Viable Product* – минимально жизнеспособный продукт) для исследования рынка и демонстрации видения, тогда фокусируемся на скорости и простоте разработки, а также на масштабируемости, чтобы систему в будущем было легко разбить на микросервисы и продолжать развивать в соответствии с реальными потребностями. В нашем случае - второе. Будем использовать технологии, которые ускорят создание проекта, но при этом позволят сделать его масштабируемым.

Далее переходим к проектированию API – программного интерфейса приложения. Опираясь на планируемый функционал, понимаем, что нам нужно два основных сервиса: пользователи и видео. В первом будем пока хранить всю информацию о пользователе и логику работы с JWT токенами.Данные о действиях пользователя желательно хранить, как отдельные сущности, но мы в данный момент ориентируемся на скорость и простоту, поэтому будем хранить их прямо в модели пользователя. Клиентскую часть реализуем в виде SPA, которое будет использовать API.

**Создание API**

Начинаю с выбора технологий: буду использовать Node.js (Express) и облачную базу данных MongoDB Atlas (технологии выбраны из соображений о высокой скорости разработки)[2]. Само приложение сделаю монолитным из тех же соображений.

Файловая структура получилась следующего вида:

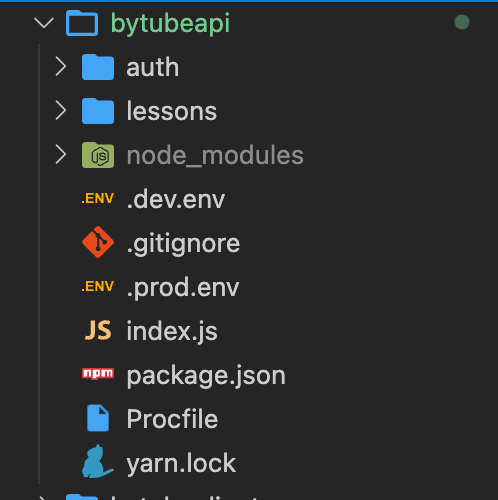


Рисунок 1. Скриншот файловой структуры

* файл index.js - точка входа в приложение;
* файлы .env содержат переменные окружения для удобного запуска приложения в разных режимах;
* файл package.json содержит описание приложения и зависимости;
* рrocfile необходим для деплоя приложения;
* файл yarn.lock создается пакетным менеджером yarn для решения проблем с зависимостями;
* директории auth и lessons содержат в себе одноимённые сервисы;
* файл gitignore нужен для игнорирования некоторых файлов системой контроля версий Git.

**Сервис Auth (аутентификация) имеет вид:**

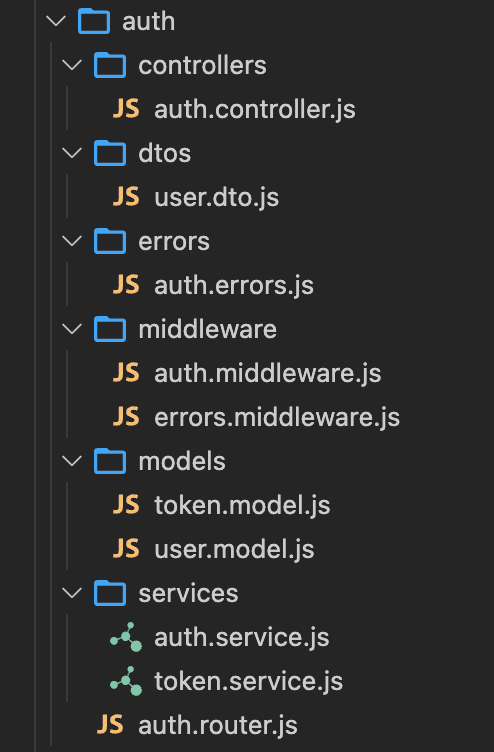


Рисунок 2. Скриншот сервиса **Auth**

* Директория controllers содержит в себе контроллеры ответа на запросы связанные с сервисом.
* Директория dtos содержит сущности используемые для удобной передачи информации о модели в базе данных.
* Директория errors содержит простую логику обработки наиболее популярных ошибок.
* Директория middleware содержит прослойку для проверки запросов (например проверка валидности токена)
* Директория models содержит модели для СУБД.
* Директория services содержит сервисы реализующие внутреннюю логику приложения (отделены от controllers из соображений об изоляции логики от обработки запросов).

Файл **router** - содержит ручки для использования сервиса.



Рисунок 3. Скриншот файла **router**

Сервис **Lessons** реализован похожим образом:

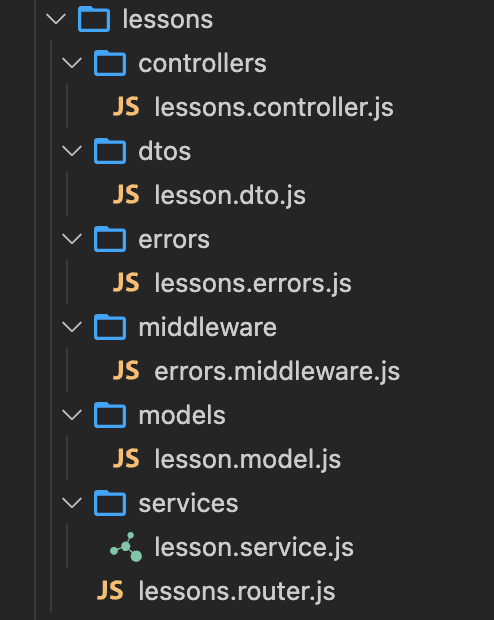


Рисунок 4. Скриншот сервиса **Lessons**

Стоит отменить, что приложение сейчас способно хранить только ссылки на видео, которые хранятся на других платформах в открытом доступе, связано это с тем, что для полноценной системы хранения видео необходимо большое количество памяти, желательно целый FTP сервер, создание которого потребует значительных затрат, как по времени, так и денежных, и сложной системы модерации, которая решала бы проблемы с авторским правом.

Так как проект нацелен на демонстрацию, нас устраивает такое поведение, но в будущем изменить его будет очень легко, благодаря масштабируемой архитектуре (достаточно высокой изолированности сервисов друг от друга, а также разделения слоев их логики)

**Создание клиентской части**

Попробую использовать React[3], SCSS[4] для создания интерфейса и Redux[5] - для стейт-менеджмента.

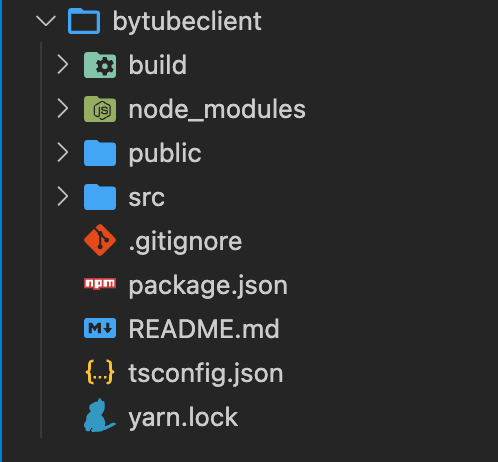


Рисунок 5. Структура файлов клиентской части

* Директория build содержит оптимизированную сборку проекта.
* Директория public содержит только index.html (главный и единственный HTML файл приложения).
* Директория src содержит всю логику работы SPA приложения.

При создании клиентского приложения упор был также сделан на скорость, но при этом оно остаётся хорошо масштабируемым.

Например, в директории **router** находятся файлы, отвечающие за маршрутизацию, они выделены в отдельный слой логики для простоты внесения изменений.

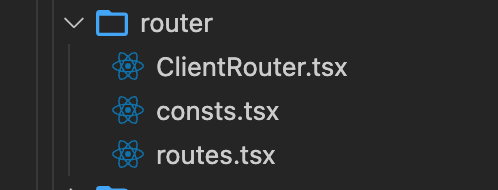


Рисунок 6. Скриншот директории **router**

В директории **store** также присутствует разделение логики на отдельные компоненты:

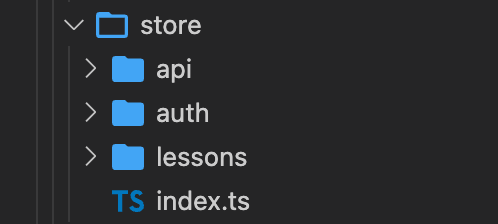


Рисунок 7. Скриншот директории **store**

Для завершения MVP:

* внедрение Docker для простоты использования на разных платформах.
* миграция на локальную базу данных, чтобы не было лишнего времени ожидания у запросов к СУБД.

Для подготовки к релизу:

* переход на микросервисную архитектуру и полное разделение логики
* внедрение брокера сообщений для удобного взаимодействия между сервисами (Kafka или RabbitMQ)
* создание админки и ролей пользователя для повышения удобства работы с приложением и более сложного разделения прав доступа
* создание FTP сервера для хранения всех файлов пользователей
* добавление нового функционала, полноценной системы рекомендаций и поиска (возможно с применением машинного обучения), монетизации контента.

# Заключение

Используя приобретенные в ходе исследования знания, мы создали и частично наполнили контентом приложение, которое поможет реализовать интерактивное взаимодействие учеников и учителей нашей школы. Особенно это актуально для работы с детьми–инвалидами или часто болеющими.

Дистанционный формат можно эффективно использовать и для проведения профориентационной работы с учащимися различных возрастных групп: провести виртуальные экскурсии по наиболее популярным в регионе образовательным учреждениям высшего и среднего профессионального образования, провести интервью с родителями – представителям разных профессий.

Актуальность работы в том, что приложение может заменить для нашей школы иностранные ресурсы, такие как YouTube, Dailymotion, на которых присутствуют ссылки на нежелательный контент. Приложение имеет интуитивно понятный интерфейс, что делает его доступным для использования неподготовленными пользователями.

Цель и задачи исследования достигнуты.

Видео загрузки контента на школьный видеохостинг доступно по ссылке <https://youtu.be/FPbajhiRCOc>

# Список использованных источников

1. <https://www.machinelearningmastery.ru/everything-you-need-to-know-about-the-mern-stack-43d27ddd480a/>
2. <https://jinv.ru/MongoDB-and-Mongoose/sozdanie-bazy-dannyh-v-oblachnom-servise-mongodb-atlas/>
3. <https://habr.com/ru/company/ruvds/blog/428077/>
4. <https://webcademy.ru/blog/346/>
5. <https://tproger.ru/translations/redux-for-beginners/>

# Приложение

Созданное приложение имеет простой и интуитивно понятный интерфейс.

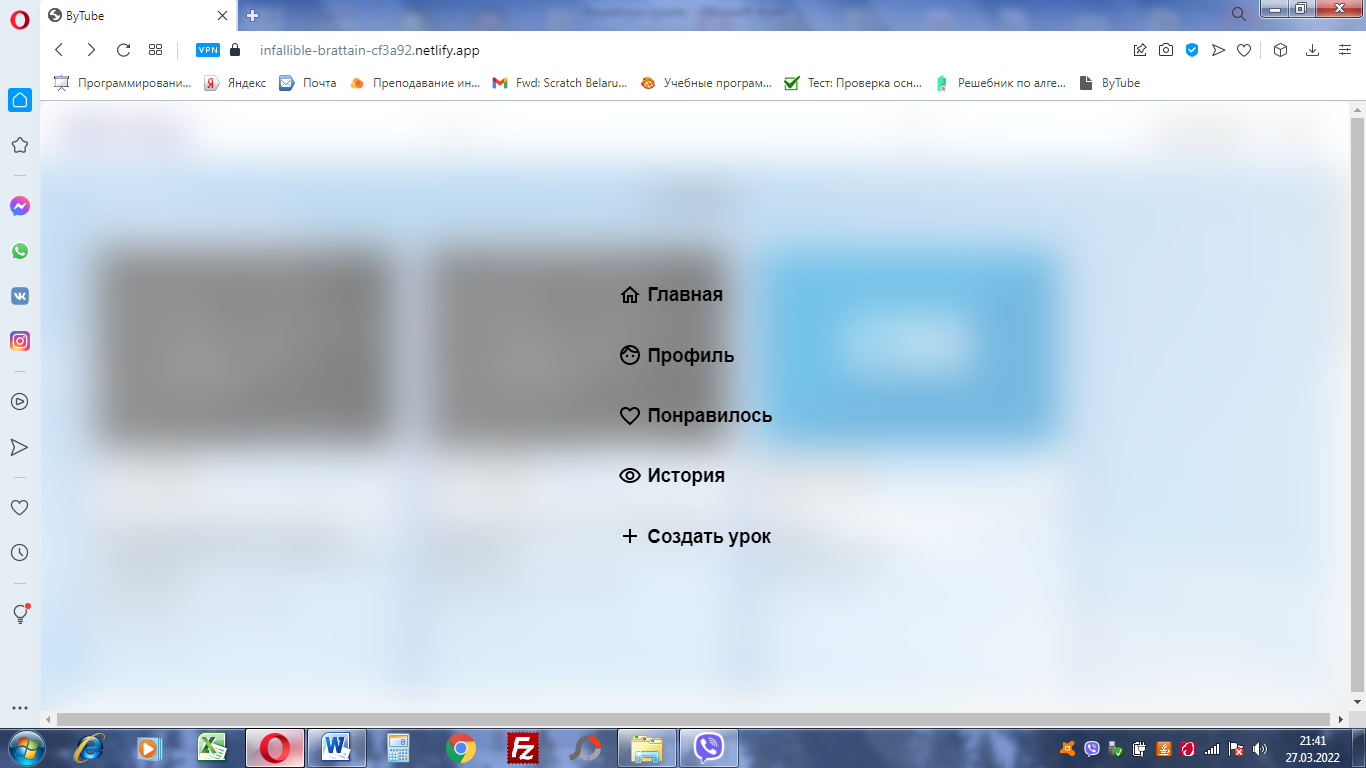


Рисунок 1. Главное меню приложения

После несложной процедуры регистрации пользователь может добавлять и удалять ссылки на свои видеоролики, просматривать историю, отмечать понравившийся контент.

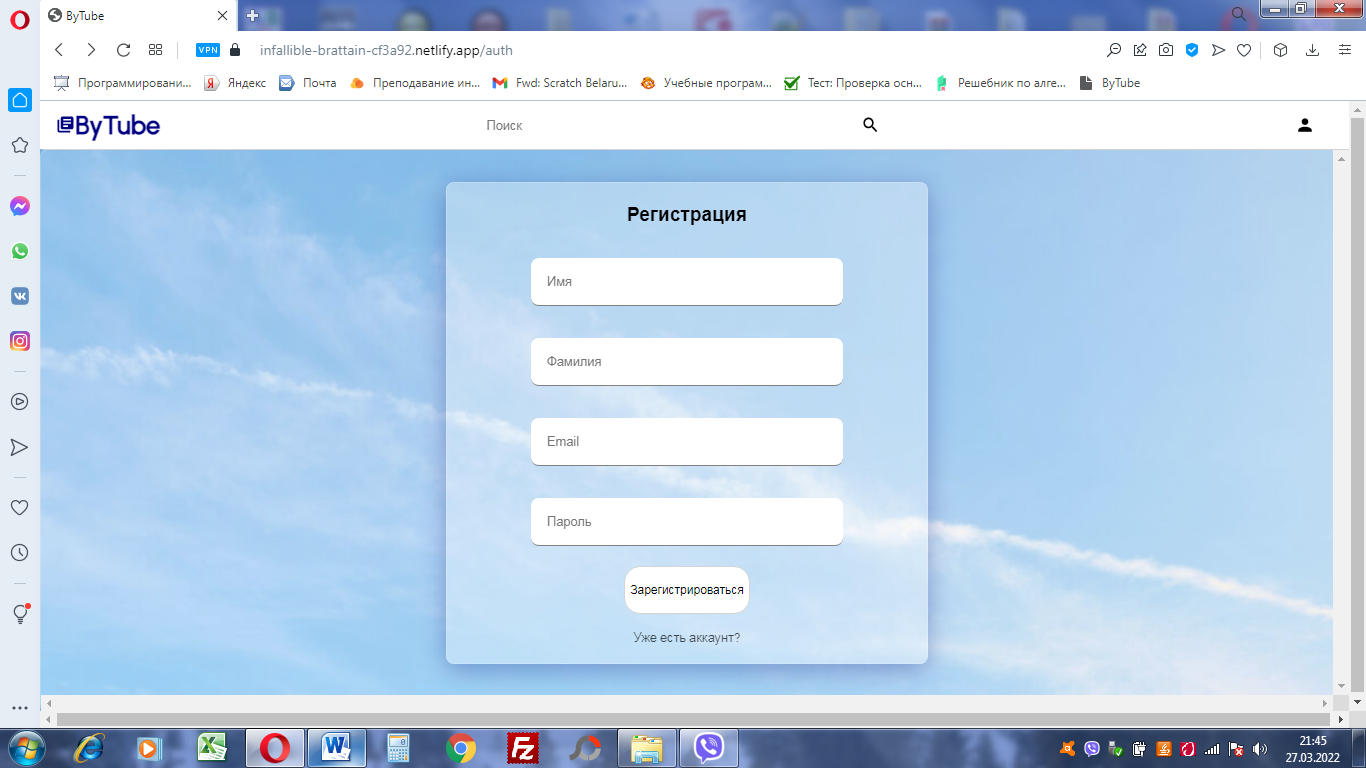


Рисунок 2. Окно регистрации

Зарегистрированные пользователи вводят свой адрес электронной почты и пароль:

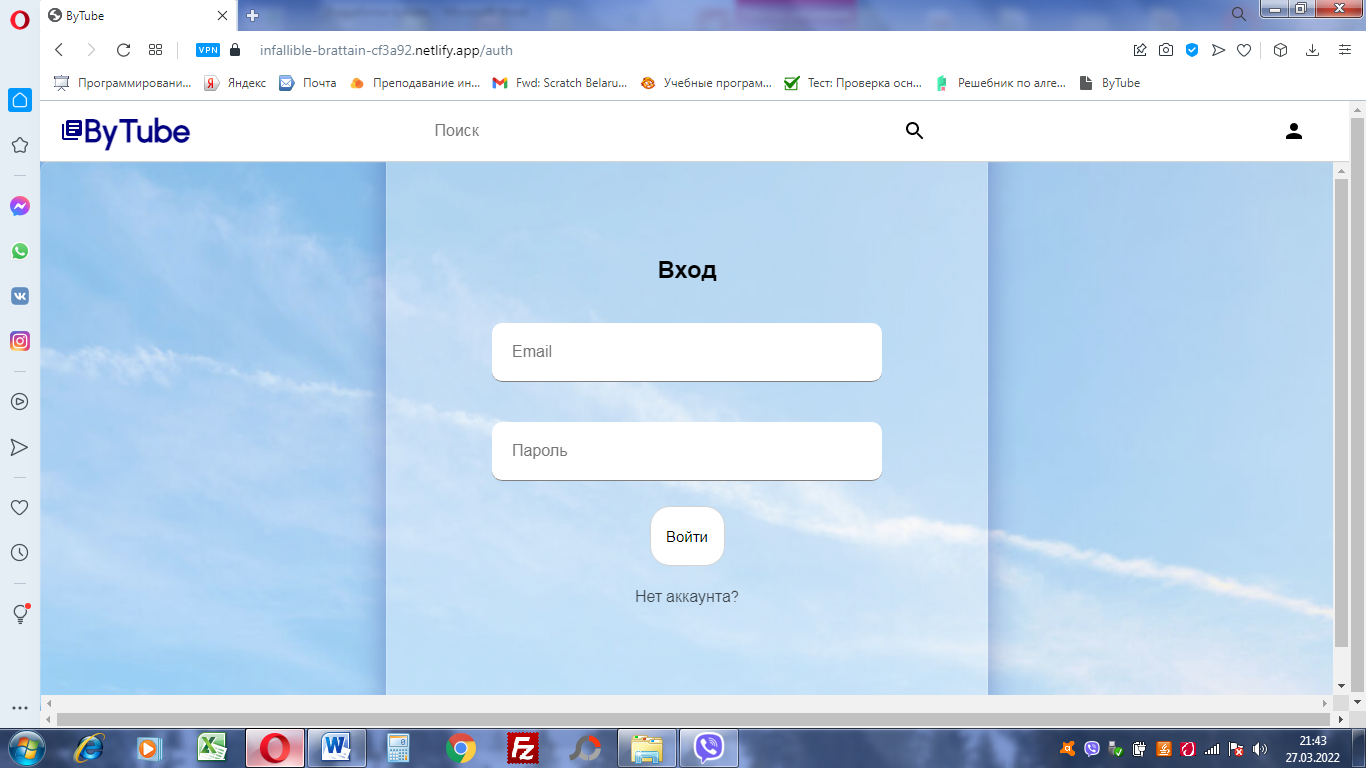


Рисунок 3. Окно авторизации

В приложении реализована возможность поиска через поисковую строку.

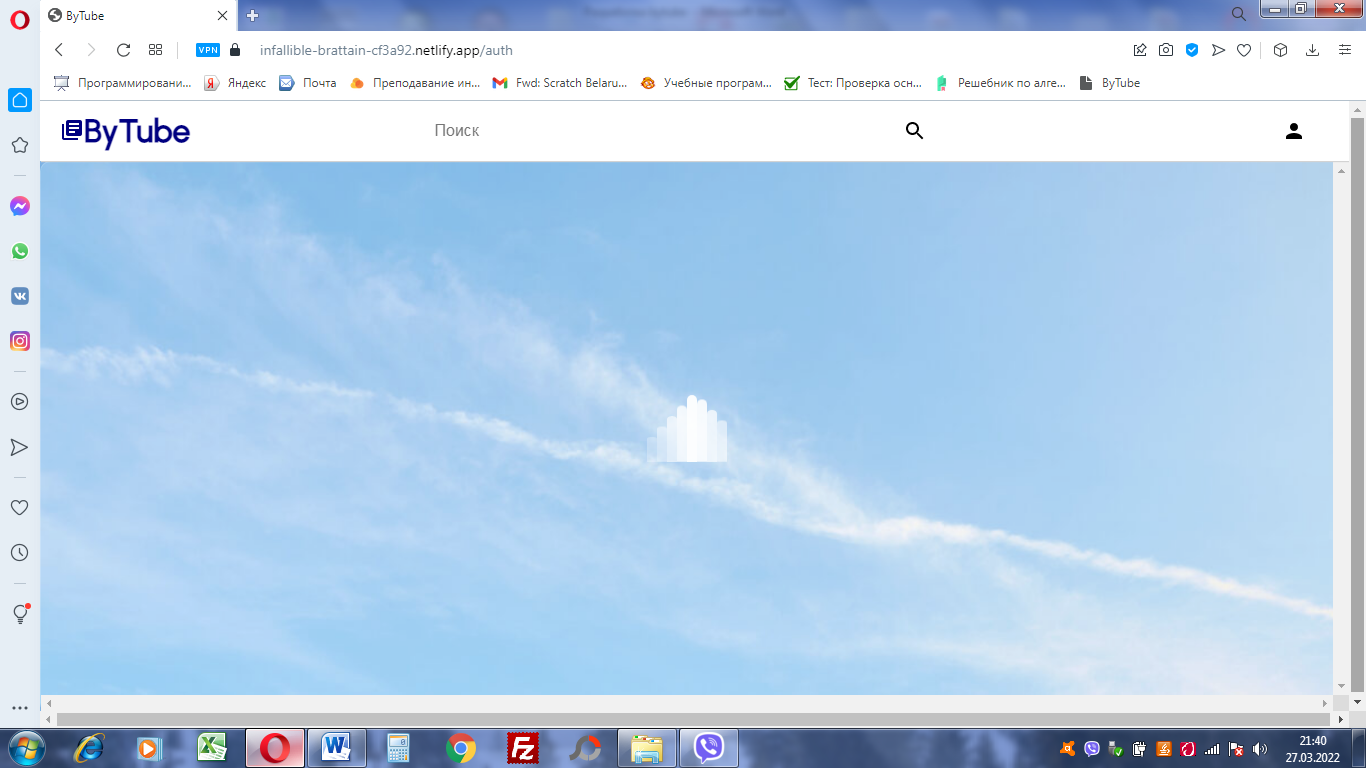


Рисунок 4. Поисковая строка

Вставлять можно ссылки на видео, хранящееся на облачных сервисах, что позволяет экономить дисковое пространство на хостинге.

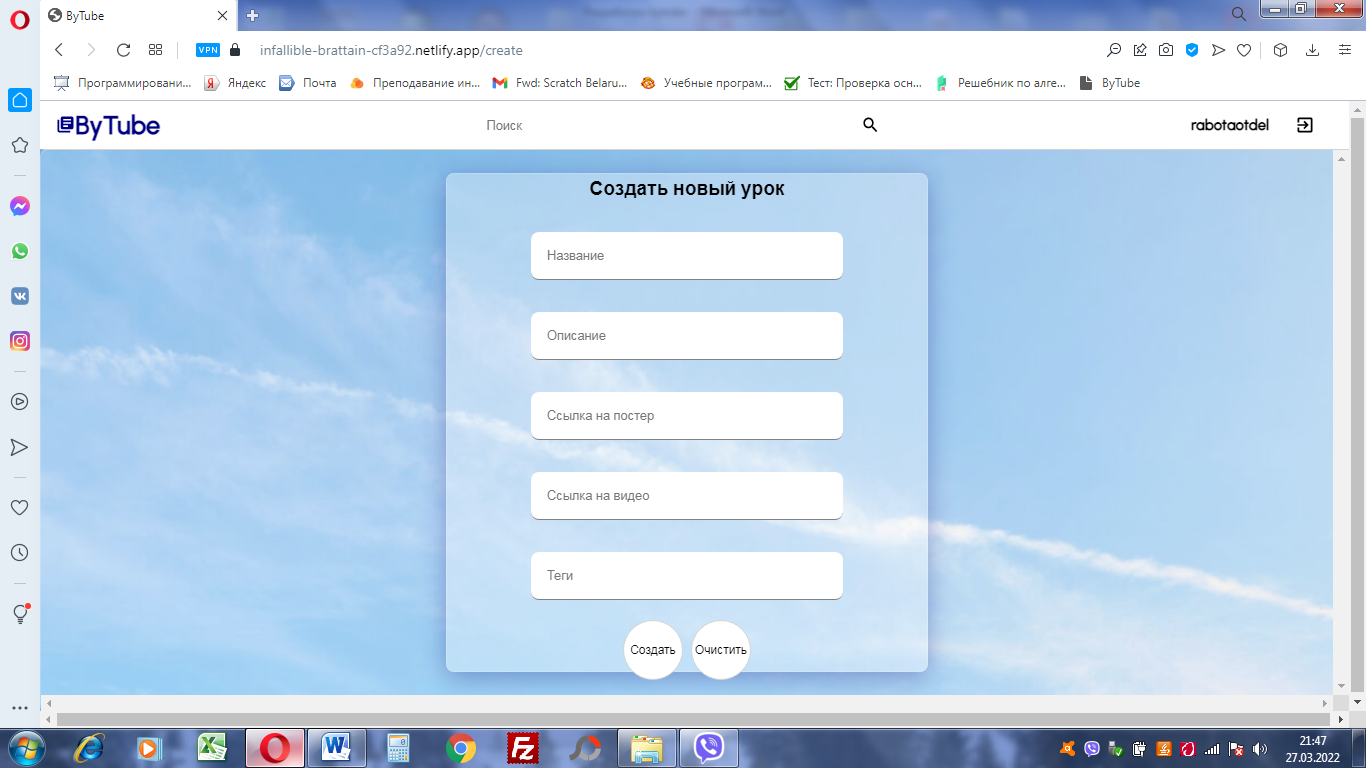


Рисунок 5. Окно добавления ссылки на новый контент