**Муниципальное бюджетное общеобразовательное учреждение**

**Сергиево - Посадская гимназия им. И.Б. Ольбинского**

Конкурс

самостоятельных творческих работ

им. П.А.Флоренского

*Попов Константин,*

*учащийся 10 «А» класса*

**Сайт для учителя**

Проект

**Руководитель**:

*Свиридкин Игорь Васильевич*

*учитель информатики*

**Сергиев Посад**

**2023**

**Оглавление**

[Введение 3](#_Toc133865649)

[Теория 4](#_Toc133865650)

[Основная часть 5](#_Toc133865651)

[HTML 5](#_Toc133865652)

[CSS 9](#_Toc133865653)

[JavaScript 12](#_Toc133865654)

[NodeJS 14](#_Toc133865655)

[MongoDB 16](#_Toc133865656)

[Покупка домена 18](#_Toc133865657)

[Деплой сайта 19](#_Toc133865658)

[Источники 20](#_Toc133865659)

# **Введение**

В рамках подготовки к уроку или объяснения материала, учителю часто приходится прикреплять различные файлы к Школьному Порталу. Но в виду нестабильности работы данного сервиса часто это бывает невозможно. В рамках своего проекта я хочу создать сайт для учителя информатики для упрощения процесса обучения.

На данном сайте будет возможность прикреплять различные домашние задания и материалы к ним, тестировать учеников и размещать информацию касательно экзаменов и олимпиад. Таким образом, каждый гимназист будет иметь доступ к важной информации по предмету, а учителю станет легче проводить занятия. Правом на размещение информации будут обладать учителя информатики гимназии, а также некоторые ученики. В дальнейшем на сайте могут появиться разделы по другим предметам.

Цели:

* Создание сайта для учителя

Задачи:

* Разработать дизайн и макет сайта
* Сверстать сайт
* Настроить работу сервера
* Добавить возможность редактировать сайт через окно администратора
* Заполнить сайт необходимой информацией
* Предоставить доступ учителю

# **Теория**

Для того, чтобы создать сайт для учителя, нужно понять, как он работает и из чего состоит. Сфера создания сайтов (или веб-разработка) мне интересна довольно давно. Но для создания этого сайта пришлось узнать много нового.

Во-первых, сайты бывают статические и динамические. В первом случае на ваш запрос сервер всегда отправляет одинаковый ответ. То есть информация на данном сайте статична. Таким сайтом может быть, например, сайт-визитка. А во втором случае сервер формирует ответ исходя из вашего запроса и информации из базы данных. Так, например, каждый ученик видит в своем кабинете Школьного Портала только свои домашние задания, а не все имеющиеся на сервере.

Во-вторых, вся веб-разработка делится на 2 части: Frontend и Backend. Первый занимается дизайном и наполнением сайта, а второй отвечает за работу сервера, его взаимодействие с базой данных. Статические сайты не имеют Backend составляющей. Даже в небольших компаниях за эти две области отвечают разные специалисты.

Теперь, когда мы разобрались с работой сайта, самое время приступать к его созданию. Для этого надо определиться с инструментами, которые мы будем использовать. В качестве Frontend составляющих я использовал язык гипертекстовой разметки HTML, язык стилей CSS и язык программирования JavaScript. За работу сервера и базы данных отвечали NodeJS, MongoDB и шаблонизатор Handlebars.

Далее необходимо создать макет сайта используя такие программы, как Figma или Photoshop, чтобы иметь представление о дизайне сайта. Но я в своей работе данный пункт опустил для экономии времени.

# **Основная часть**

## **HTML**

Рассмотрим процесс создания Frontend части сайта на примере страницы с вопросами учащихся. Начать необходимо с HTML. Для начала создается скелет сайта, который содержит в себе указание языка сайта, то есть в нашем случае русский, а также теги <head> и <body>. В первом содержится информация для браузера: стандарт кодировки, название страницы и ее логотип, а также ссылки на подключение файлов с CSS, JavaScript и используемым шрифтом. Итоговый код выглядит так:

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Домашняя страница</title>

    <link href="css/index.css" rel="stylesheet">

    <link rel="icon" href="assets/logo.svg">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65" crossorigin="anonymous"/>

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link href="https://fonts.googleapis.com/css2?family=Montserrat&family=Nunito:ital,wght@1,300&display=swap"

        rel="stylesheet">

    <script src="js/questions.js" defer></script>

</head>

В теге <body> создаем отдельные блоки под шапку сайта (тег <header>), основную часть (тег <main>) и подошву сайта (тег <footer). Так как в шапке сайта находится меню, то создадим его в виде списка с ссылками на другие страницы сайта. Рядом расположим блок с кнопкой возврата на стартовую страницу. Для кнопки возврата воспользуемся картинкой в SVG формате, так как данный формат изображения не теряет качество при масштабировании. В итоге получается следующий код:

<header>

        <ul class="menu">

            <li class="itemMenu">

                <a class="linkOnMenu" href="/dz">Задания</a>

            </li>

            <li class="vertMenu itemMenu">

                <div class="vertical-line"></div>

            </li>

            <li class="itemMenu">

                <a class="linkOnMenu" href="/materials">Материалы</a>

            </li>

            <li class="vertMenu itemMenu">

                <div class="vertical-line"></div>

            </li>

            <li class="itemMenu" id="itemMenu">

                <a class="linkOnMenu" href="/questions">Вопросы</a>

            </li>

            <li class="vertMenu itemMenu">

                <div class="vertical-line"></div>

            </li>

            <li class="itemMenu">

                <a class="linkOnMenu" href="/about">О нас</a>

            </li>

        </ul>

        <div class="user">

            <h5 class="userName">{{name}}</h5>

            <a class="exit" href="/">

                <img height="30px" width="30px" src="assets/exit.svg">

            </a>

        </div>

    </header>

Далее следует сделать поле для поиска нужного ученику вопроса среди уже заданных. Вводимые данные будут отправляться на сервер с помощью GET-запроса. Для создания такого поля создадим тег <form>, содержащий тег <input> для ввода текста и кнопку <button> для отправки данных на сервер:

<form class="mb-3" action="/questions">

<div class="input-group">

<input type="text" value="{{search}}" name="search" class="form-control inputSearch" placeholder="Поищите ответ здесь..." aria-describedby="button-addon2">

<button class="btn btn-outline-dark btnClear" type="button">Очистить</button>

<button class="btn btn-outline-primary btnSearch" type="submit">

Искать вопрос</button>

</div>

</form>

Каждый тег в HTML содержит все в себе так же один или несколько атрибутов. У каждого тега атрибуты разные. На примере тега <input> объясню небольшую часть из них. Атрибут type отвечает за тип отправляемого контента на сервер. Name дает название тем данным, которые отправляются на сервер. Placeholder отображает надпись внутри поля ввода. Атрибут class используется для применения CSS и JavaScript содержимого.

Далее на сайте должна находиться таблица с заданными вопросами. Таблица в HTML создается с помощью тега <table>. В таблице будут присутствовать только 2 столбика: вопрос и уведомление о наличие ответа на него. Пропишем это в теге <thead>. В теге <tbody> позже наладим вывод информации о вопросах, полученной с сервера.

<table class="table table-sm table-bordered">

<thead class="tableHead">

<tr>

<th scope="col">Вопрос</th>

<th scope="col" width="25%">Ответ?</th>

</tr>

</thead>

<tbody>

</tbody>

</table>

После этого стоит сделать форму для добавления новых вопросов. Кроме вышеупомянутых тегов <form> и <input> необходимо воспользоваться тегом <textarea>, так как он дает возможность отправлять на сервер многострочный текст.

<form action="/question\_update" method="POST" class="questionForm">

                <div class="modal-header mb-2">

                    <h5 class="modal-title" id="exampleModalLabel">

Задайте ваш вопрос:</h5>

                </div>

                <div class="mb-3">

                    <input id="studentName" class="form-control" type="text" name="student" placeholder="Ваше прекрасное имя...">

                </div>

                <div class="mb-3 form-check">

                    <input type="checkbox" class="form-check-input" id="anon">

                    <label class="form-check-label" for="anon">

Задать вопрос анонимно</label>

                </div>

                <div class="input-group mb-3">

                    <textarea class="form-control" type="text" name="text" rows="3"

                        placeholder="Ваш вопросик..."></textarea>

                </div>

                <button class="btn btn-primary mb-3">

                    Отправить

                </button>

</form>

На этом верстка основной части страницы завершена. Следует сделать так называемый «подвал» страницы. Там будут располагаться контакты автора. Сделаем их также в виде списка при помощи тегов <ul> и <li>:

<ul>

<li class="aboutUs">

<img src="/assets/vk.svg" width="40px">

<a class="I" href="https://vk.com/little\_man2306">VK</a>

</li>

<li class="aboutUs">

<img src="/assets/tg.svg" width="40px">

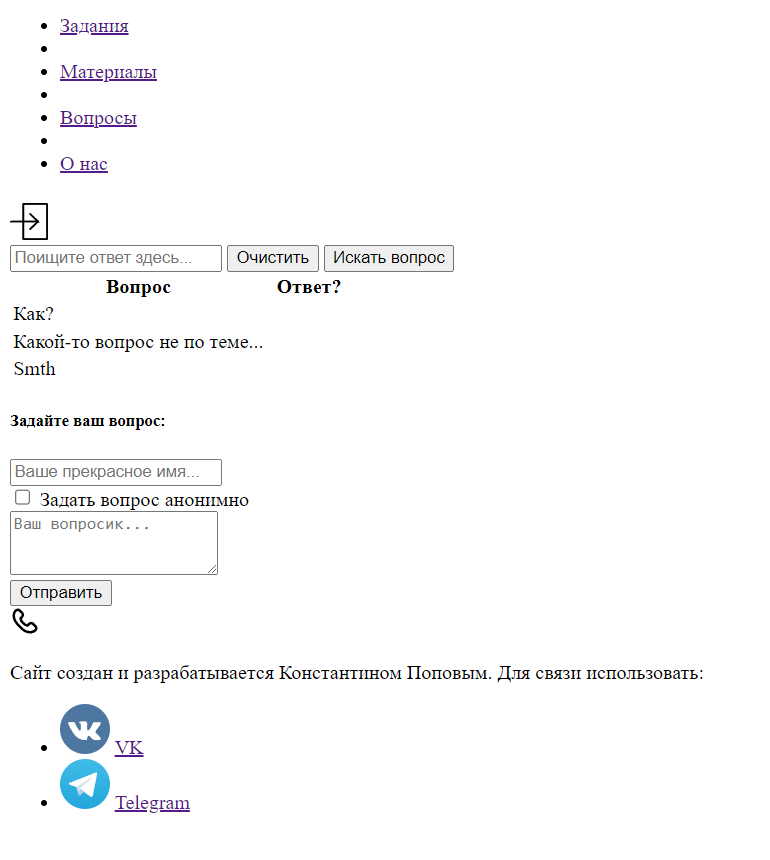
<a class="I" href="https://t.me/Elona\_muska">Telegram</a>

</li>

</ul>

## **CSS**

После всех действий, совершенных выше, мы получаем примерно такой результат:



Для того, чтобы наш сайт выглядел красиво, необходимо использовать CSS. С помощью данного инструмента можно задать определенные свойства у каждого объекта. Данных свойство очень много, но вот некоторые их них: цвет, фон, размеры, внутренние и внешние отступы, шрифт текста и его положение и так далее. На весь проект понадобилось около 600 строк свойств. Для того, чтобы применить свойства к определенному блоку, нужно обратиться к нему при помощи его атрибута class или id. Рассмотрим на примере блока, который содержит таблицу сс вопросами и форму их поиска. Данный блок имеет в HTML файле следующий вид:

<section class="questionList">

…

</section>

CSS файл же содержит следующее:

.questionList {

    box-sizing: content-box;

    padding: 15px;

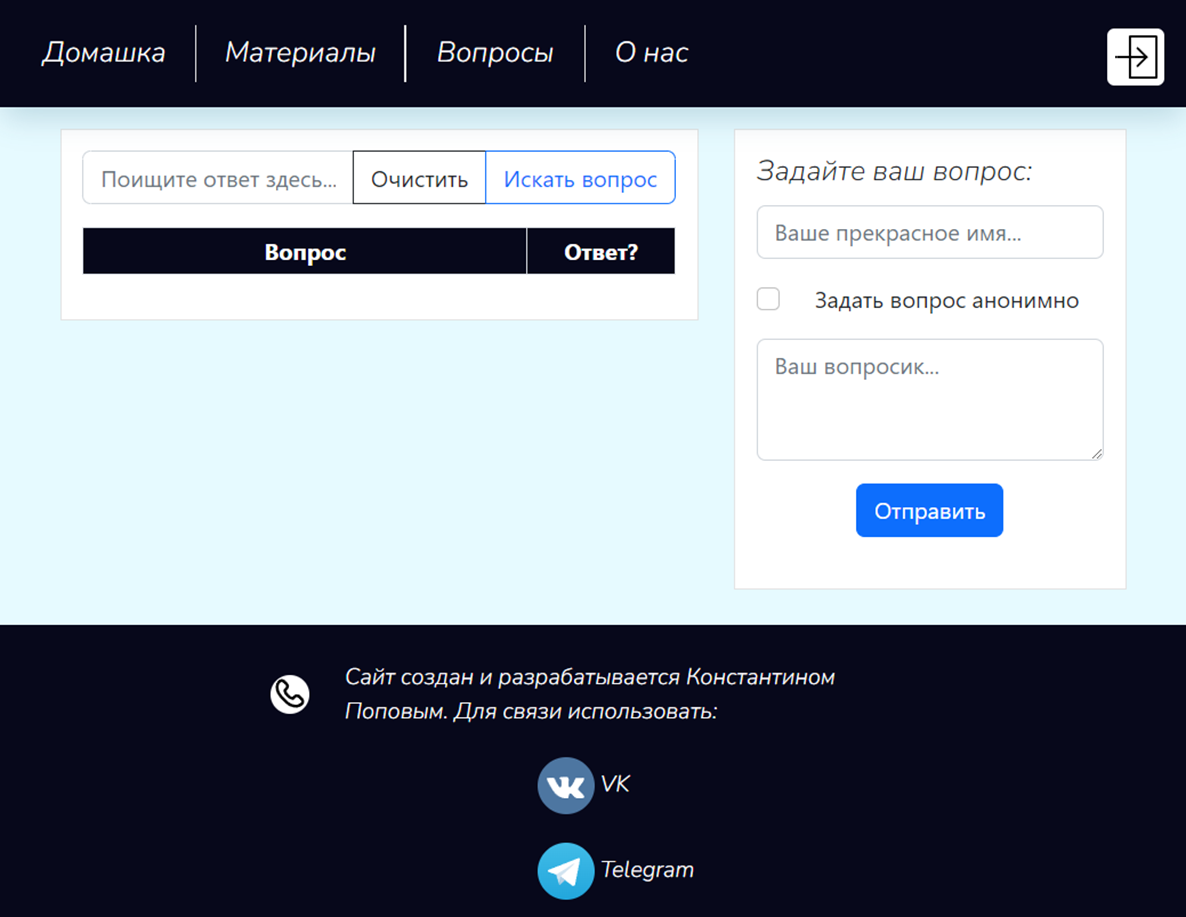
    border: 1px solid #e3e3e3;

    background-color: white;

    max-width: 50%;

}

При обращении к классу объекта необходимо перед названием поставить точку, а при использовании id – ‘#’. Рассмотрим используемые свойства. Первая строчка делает размер блока зависимым только от размера содержимого. Padding – это внутренние отступы, то есть все содержимое будет располагаться на расстоянии 15 пикселей от границ блока. Свойство border с данными характеристика создает у блока сплошную границу светло-серого цвета толщиной в 1 пиксель. В предпоследней строчке мы делаем фоновым цветом белый. Свойство max-width ограничивает ширину нашего блока, давая ему возможность занимать только ее половину. После данных манипуляций у нас получится примерно такой результат:



## **JavaScript**

Наш сайт будет содержать малое количество JavaScript-кода, так как рендеринг страниц будет происходить только на сервере. Данный подход подразумевает большую нагрузку на сервер, но это не является проблемой, зато написание кода упрощается в разы. Именно поэтому в нашем случае JS нужен только для создания интерактивных пользовательских элементов. На странице вопросов будет кнопка очищения поля поиска вопросов и чекбокс для того, чтобы задать вопрос анонимно. Рассмотрим код для последнего. Для начала следует объявить переменные:

let anonBtn = document.querySelector(`#anon`);

let inputName = document.querySelector(`#studentName`);

После чего необходимо поместить на кнопке обработчик нажатия по ней, то есть при клике на нее будет выполняться определенный алгоритм действий:

anonBtn.addEventListener(`input`, function() {

    if (anonBtn.checked) {

        inputName.disabled = true;

        inputName.value = ``;

        inputName.placeholder = `Теперь вы анонимный котеночек (`;

    }

    else {

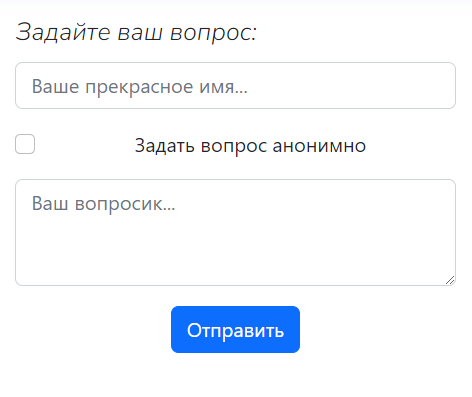
        inputName.disabled = false;

        inputName.placeholder = `Ваше прекрасное имя...`;

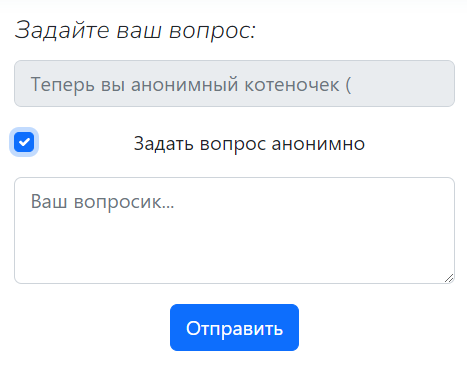
    }

})

Проделав вышесказанные действия получаем следующий результат:



При нажатии происходят изменения:



На этом написание Frontend-части проекта завершено, приступим к Backend-разработке.

## **NodeJS**

Для написания серверного кода воспользуемся таким пакетом, как NodeJS. Он позволяет писать код сервера используя язык программирования JavaScript. Это ускорит разработку проекта, так как не придется изучать новый язык программирования.

Так как серверная часть моего проекта отвечает за большинство предполагаемых функций, код для нее получился весьма большим – больше 400 строк. Рассмотрим, какую работу выполняет сервер на странице с вопросами.

Для начала создадим роут ‘/questions’ и создадим для нее асинхронную функцию, то есть на определенных участках кода дальнейшее выполнение будет остановлено, пока не будет выполнен этот участок кода. После примем запрос от пользователя из поля поиска вопросов, который мы создали ранее. А также добавим конструкцию try. Таким образом, при ошибке в выполнении кода пользователь увидит страницу с ошибкой. Получим следующее:

app.get(`/questions`, async function(req, res) {

    let search = req.query.search;

    try {

…

    } catch(error) {

        res.render(`error`);

    }

});

Рассмотрим работу сервера в том случае, если поискового запроса нет, то есть пропишем условие, где проявляем наличие, собственно, запроса. А далее обращаемся к базе данных и находим все заданные вопросы, сортируя их по дате добавления от новых к старым. После чего рендерим страницу и отдаем ее пользователю:

if (!search) {

let questions = await Question.find().sort({

createdAt: -1});

res.render(`questions`, {

questions: questions

});

}

При наличии поискового запроса схема аналогичная, только теперь после получения данных проверяем каждый вопрос на соответствие запросу пользователя:

else {

            let searchL = search.toLowerCase();

            let questions = await Question.find().sort({

                createdAt: -1});

            let questionSearch = [];

            for (let i = 0; i < questions.length; i++) {

                let item = questions[i];

                let itemText = item.text.toLowerCase();

                if (itemText.includes(searchL)) {

                    questionSearch.push(item);

                }

            }

            res.render(`questions`, {

                questions: questionSearch,

                search: search

            });

        }

При работе с базой данных очень важно использовать именно асинхронную функцию, то есть код не будет выполняться до тех пор, пока ответ от базы не будет получен.

Код для сервера готов, осталось подключить базу данных и настроить ее.

## **MongoDB**

Для подключения к базе данных необходимо вновь вернуться к написанию серверного кода. Чтобы реализовать возможность взаимодействия базы данных и сервера надо подключить пакет Mongoose и подключиться к самой базе:

let mongoose = require(`mongoose`);

mongoose.connect(`mongodb://127.0.0.1:27017/spg`)

Наша база данных называется ‘spg’ и будет храниться на этом же сервере. Каждая база данных содержит в себе одну или несколько коллекций. А в каждой коллекции, в свою очередь, хранится документ или информация. Соответственно, информация о вопросах будет храниться в коллекции ‘questions’. Для корректной работы сервера с базой надо прописать схему и создать модель – так сервер поймет, что именно хранится в сервере и как к этой информации обращаться:

let QUschema = new mongoose.Schema({

    text: String,

    student: String,

    time: String,

    teacher: String,

    answer: String

}, {

    timestamps: true

});

let Question = mongoose.model(`question`, QUschema);

После данных манипуляций при запуске сервера база данных создастся автоматически. Но необходимо еще реализоваться функцию добавления вопросов в базу, приступим к этому.

Добавление вопроса будет сделано уже при помощи post-запроса, так информация будет защищена и невидна пользователям. Остальная часть аналогична коду выше. В конце нужно вернуть пользователя на предыдущую страницу.

app.post(`/question\_update`, async function(req, res) {

    let teacher = req.body.teacher;

    let text = req.body.text;

    let date = new Date();

    let time = String(date.getHours() + `:` + (("0" + date.getMinutes()).slice(-2)) + ` ` + (("0" + date.getDate()).slice(-2)) + `.`+ String(("0" + (date.getMonth()+1)).slice(-2)) + `.`+ date.getFullYear());

    let student = req.body.student;

    if (teacher!=`Выберите учителя:` && text) {

        if (!student) {

            student = `Анонимный котеночек`;

        }

        let ques = new Question( {

            text: text,

            time: time,

            student: student,

            teacher: teacher

        })

        await ques.save();

    }

    res.redirect('/questions');

})

## **Покупка домена**

Так как каждый сайт в Интернете находится под своим доменным именем, то для работы нашего проекта необходимо так же купить домен. В моем случае я воспользовался сайтом reg.ru для покупки. За 129 рублей мною на год был приобретен домен konst.space. После покупки я за 200 рублей купил электронный сертификат о регистрации имени:



Важно также отметить, что мой сайт будет доступен только через протокол http, который не является безопасным. Для использования протокола https необходимо платить от 3 500 рублей в год.

## **Деплой сайта**

Настало время загрузить наш сайт в Интернет. Для этого воспользуемся сервисом reg.ru. Так как наш сайт использует в качестве серверного языка программирования NodeJS, он должен работать не при помощи хостинга, а на удаленном сервере. То есть передо мной стояла задача арендовать выделенный сервер, на котором будут запущены сайт и база данных. И с этой задачей я справился. Цена за аренду, которую я заплатил, составила 370 рублей в месяц.

Таким образом я получил в распоряжение удаленный сервер с операционной системой Ubuntu. Для настройки сервера понадобится две программы: Putty и Filezilla. Первая дает возможность подключаться к серверу и управлять им через командную строку. А второе нужно для переброски файлов проекта на сервер.

После все манипуляций наш сайт наконец-то работает и готов для использования пользователями.

Из-за отсутствия денежных средств у создателя на момент 1 Мая 2023 года сайт не работает.

# **Источники**

1. Resources for Developers, by Developers: [Электронный ресурс]. URL: <https://developer.mozilla.org/ru.> (Дата обращения: 01.05.2023).
2. База знаний REG.ru: [Электронный ресурс]. URL: [https://help.reg.ru/support](https://help.reg.ru/support/). (Дата обращения: 01.05.2023).
3. Установка MongoDB на Ubuntu 18.04: [Электронный ресурс] // Eternalhost. URL: <https://eternalhost.net/base/vps-vds/ustanovka-mongodb-ubuntu>. (Дата обращения: 01.05.2023).