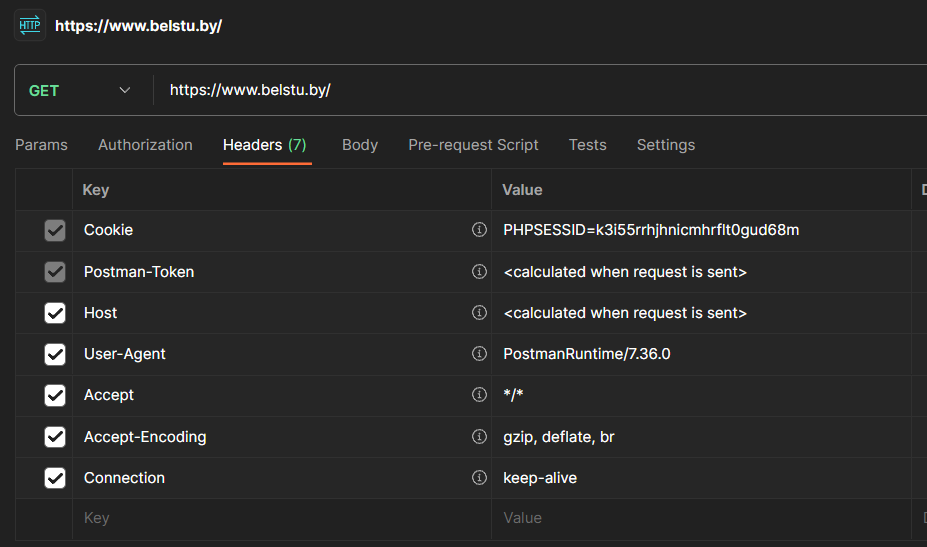
**Задание 1**

3. POSTMAN: выполните GET-запрос к сайту БГТУ: https://www.belstu.by/

База, вроде было легко

4. Определите заголовки запроса. Поясните их назначение.



**Cookie** – содержит информацию, которую сервер отправляет браузеру пользователя. Браузер сохраняет эту информацию и отправляет её обратно на сервер с каждым последующим запросом. Это позволяет серверу узнать, что несколько запросов пришли от одного и того же браузера, что полезно, например, для аутентификации пользователя.

**Postman-Token** – используется инструментом Postman для предотвращения нежелательного кэширования запроса. Значение этого заголовка - это уникальный идентификатор, который генерируется для каждого запроса.

**Host** – содержит имя домена, для которого предназначен запрос, и, опционально, номер порта. Если порт не указан, то используется умолчательный порт протокола/сервиса (например, “80” для HTTP).

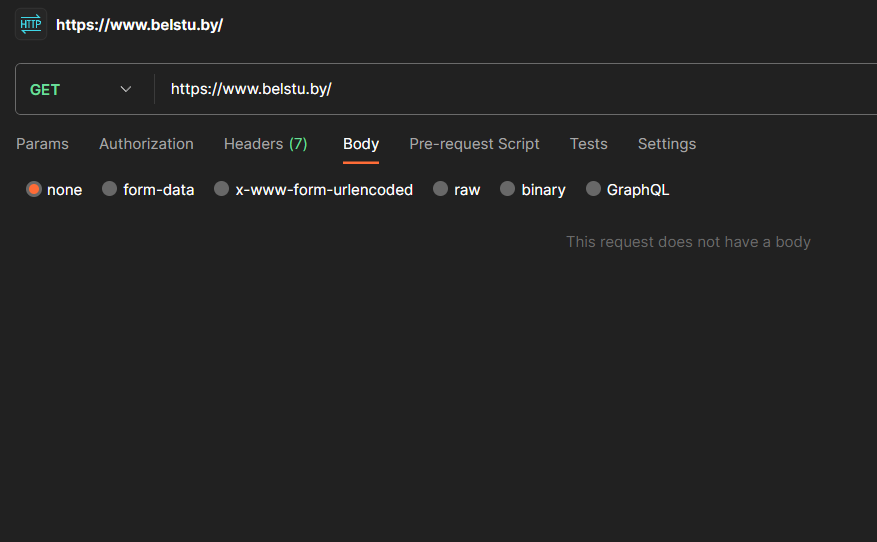
**User-Agent** – содержит информацию о клиенте, который отправляет запрос. Это может включать в себя детали о браузере, операционной системе и другую информацию, которая может быть полезна для сервера при формировании ответа.

**Accept** – какие типы контента (выраженные как MIME-типы), клиент может понять. Сервер затем выбирает одно из предложений и информирует клиента о своём выборе с помощью заголовка ответа Content-Type. (Если \*/\* - то клиент, у нас это браузер, может принять любой тип контента (html, json, png …)

**Accept-Encoding** – какие методы сжатия контента поддерживаются клиентом. Сервер затем может использовать эту информацию, чтобы определить, как сжать ответ для передачи по сети. (Если gzip, deflate, br – то это просто алгоритмы сжатия, инфа по каждому думаю излишне)

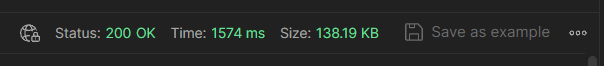
**Connection** – определяет, остаётся ли сетевое соединение активным после завершения текущей транзакции (запроса). Если в запросе отправлено значение keep-alive, то соединение остаётся и не завершается, позволяя выполнять последующие запросы на тот же сервер. (Кроме keep-alive еще может быть close (закрыть соединение), update (переключение на другой протокол))

5. Определите содержимое тела запроса. Поясните.

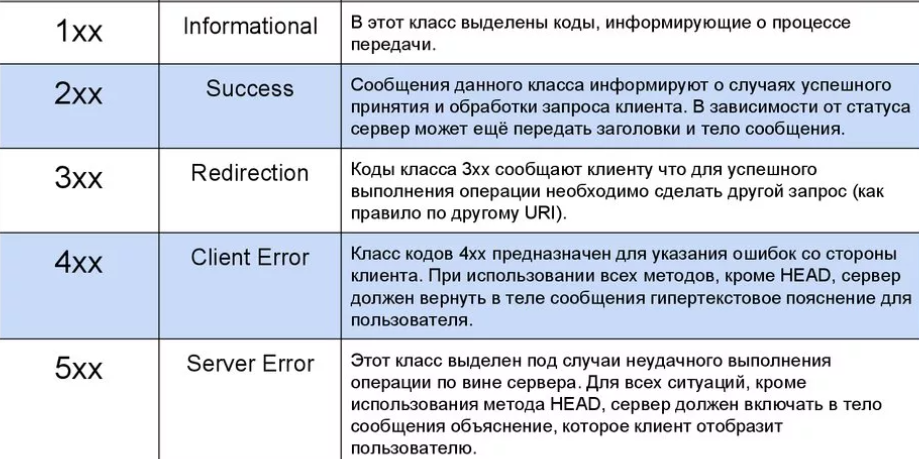


Ну мы же не посылали ничего в запросе, поэтому тут и ничего нет

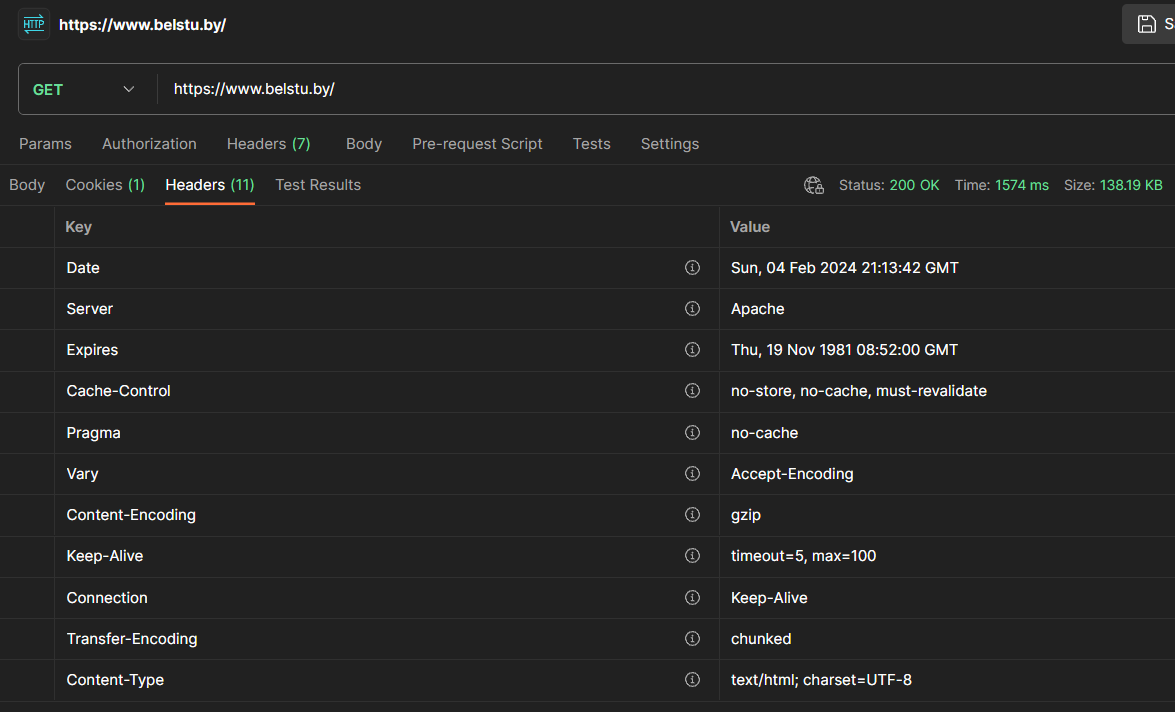
6. Определите статус ответа. Поясните значение.



200 OK – запрос успешно выполнен.



7. Определите заголовки ответа. Поясните их назначение.



**Date** – дата и время, когда было создано сообщение

**Server** – содержит информацию о программном обеспечении, используемом сервером для обработки запросов (В данном случае, у меня на сервере работает веб-сервер Apache, еще может быть nginx, GWS, Microsoft IIS)

**Expires** – содержит дату/время, по истечении которой ответ сервера считается устаревшим. Прошедшая или невалидная дата, например 0, обозначает, что ресурс уже устарел (Если в заголовке Expires указана дата в прошлом, это означает, что ресурс уже устарел. Однако, если в ответе сервера установлен заголовок Cache-Control с директивами max-age, заголовок Expires игнорируется)

**Cache-Control** – определяет поведение браузера при кэшировании. Он может содержать различные директивы, определяющие, должен ли браузер кэшировать данные и насколько свежими должны быть эти данные (Может содержать: no-store (запрещает кеширование), no-cache (необходимость отправить запрос на сервер для валидации ресурса перед использованием закешированных данных), must-revalidate (кешированный ответ должен быть проверен на актуальность перед использованием))

**Pragma** – используется для обратной совместимости с HTTP/1.0, где заголовок Cache-Control ещё не присутствует

**Vary** – определяет, как сопоставить будущие заголовки запроса, чтобы решить, можно ли использовать кешированный ответ, а не запрашивать новый с исходного сервера (Если установлен Accept-Encoding, то сервер может отправлять разные версии ресурса в зависимости от того, какие кодировки сжатия поддерживаются клиентом, может еще быть User-Agent (разные версии в зависимости от, к примеру, браузера), Accept-Language (разные версии в зависимости от языка пользователя), Origin (в зависимости от источника запроса)

**Content-Encoding** – указывает на кодировку, применённую к телу сообщения. Это позволяет клиенту информацию как декодировать тело, чтобы получить медиа-тип ссылающийся на заголовок Content-Type

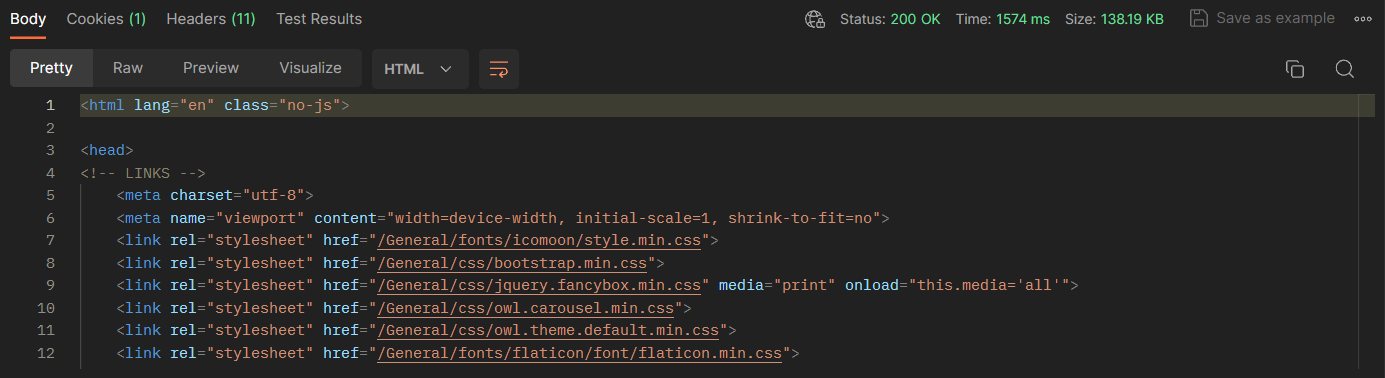
**Keep-Alive** – используется для управления постоянными соединениями. Он может указывать, сколько времени и сколько запросов может оставаться открытым текущее неиспользованное соединение (Timeout – кол-во секунд между двумя последовательными запросами на одном и том же соединении, max – максимальное кол-во запросов, которое можно отправить по одному соединению)

**Connection** – определяет, остаётся ли сетевое соединение активным после завершения текущей транзакции (запроса). Если в запросе отправлено значение keep-alive, то соединение остаётся и не завершается, позволяя выполнять последующие запросы на тот же сервер. (Кроме keep-alive еще может быть close (закрыть соединение), update (переключение на другой протокол))

**Transfer-Encoding** – указывает форму кодирования, используемую для безопасной передачи полезной нагрузки пользователю (У меня chunked – значит передается чанками (по частям), есть еще gzip, compress, deflate)

**Content-Type** – описывает MIME-тип содержимого, передаваемого в теле сообщения

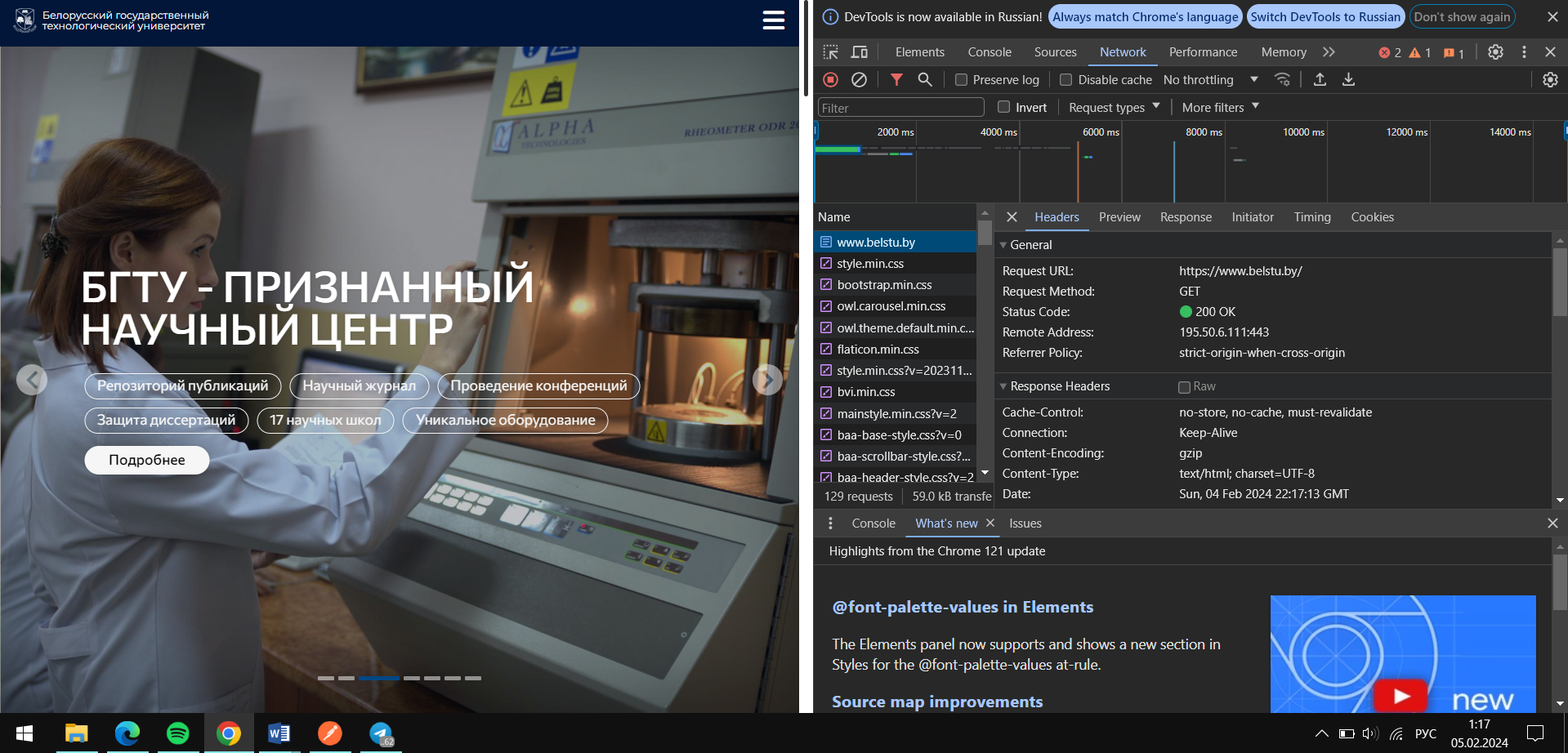
8. Определите содержимое тела ответа. Поясните.



В теле ответа будет HTML-код сайта belstu.by

**Задание 2**

1. **WEB BROWSER**: выполните GET-запрос к сайту БГТУ: <https://www.belstu.by/>
2. Используйте «**Инструмент разработчи**ка» браузера



1. Определите заголовки запроса. Определите содержимое тела запроса.
2. **Host:** Адрес сервера и порт, на который был отправлен запрос
3. **Connection:** Указывает на то, что соединение должно быть сохранено активным после завершения запроса
4. **Sec-Ch-UA**: Политика безопасности браузера
5. **Sec-Ch-UA-Mobile**: Указывает, является ли браузер мобильным (1) или нет (0)
6. **Sec-Ch-UA-Platform**: Указывает ОС, на которой работает браузер
7. **DNT:** Указывает на наличие Do Not Track (DNT) заголовка, который может указывать на предпочтения пользователя относительно отслеживания
8. **Upgrade-Insecure-Requests:** Сигнализирует серверу о том, что браузер поддерживает апгрейд запросов к защищённым версиям, таким как HTTPS
9. **User-Agent**: Содержит информацию о вашем браузере и его версии. Ваш браузер сообщает эти данные в качестве части запроса к серверу, чтобы веб-сайты могли адаптировать содержимое под различные браузеры или устройства.  
     
   Эта строка показывает, что ваш браузер имитирует версии различных браузеров для обеспечения максимальной совместимости с веб-сайтами. Например:

* Mozilla/5.0 указывает на общий формат User-Agent, который начал использоваться в Mozilla.
* Windows NT 10.0 указывает на операционную систему Windows 10.
* Win64; x64 указывает на то, что используется 64-битная версия операционной системы.
* AppleWebKit/537.36 указывает на движок рендеринга, который используется в браузере (в данном случае, это движок WebKit, который также используется в Safari).
* KHTML, like Gecko является дополнительной информацией, указывающей на подобие других движков рендеринга, таких как Gecko (используется в Firefox) и KHTML (использовался в браузере Konqueror).
* Chrome/119.0.0.0 указывает на версию Chrome (в данном случае, 119.0.0.0), однако эта версия может быть схожа с версией Chrome или Chromium, которую вы используете.
* Safari/537.36 указывает на Safari, который также использует движок WebKit.

Этот User-Agent может включать информацию о нескольких браузерах, таких как Chrome, Safari, и других, чтобы максимально поддерживать различные особенности веб-сайтов или обеспечивать совместимость с различными стандартами рендеринга страниц.

1. **Accept**: Перечислены типы контента, которые браузер готов принять, с их предпочтениями
2. **Sec-Fetch-Site**: Указывает на сайт, с которого приходит запрос
3. **Sec-Fetch-Mode**: Определяет режим запроса запрос: например, "navigate" для навигации по URL
4. **Sec-Fetch-User**: Сигнализирует, что пользователь активен и участвует во взаимодействии с браузером. (???)
5. **Sec-Fetch-Dest:** Указывает, что цель запроса - получение документа (HTML страницы)
6. **Accept-Encoding**: Сообщает серверу, какие методы сжатия контента поддерживает браузер
7. **Accept-Language**: Задает предпочитаемые языки для отображения контента на веб-странице.
8. **Cookie**: Передает информацию о cookies, которые хранятся в браузере и относятся к данному сайту. В данном случае, предоставляет информацию о различных сессионных данных и параметрах сохраненных на клиентской стороне.
9. Определите содержимое тела запроса.

Из задания 1

1. Определите статус ответа.

Ну тут понятно

1. Определите заголовки ответа.

Из задания 1, они идентичны

1. Определите содержимое тела ответа.

Ну по сути тоже результат наша веб страница

**Задание 3**

1. Разработайте html-страницу с помощью Блокнота, которая в браузере отображается следующим образом. Страница по ссылке БГТУ должна переходить на сайт БГТУ, при нажатии кнопки высвечивается информация о браузере, а в консоль разработчика должно выводиться Hello World.

Код:

<!DOCTYPE html>

<html>

<head>

<title>Lab1</title>

</head>

<body>

<h1>Лаборанторная работа №1</h1>

<p><a href="https://www.belstu.by/">БГТУ</a></p>

<button onclick="showBrowserInfo()">Информация о браузере</button>

</body>

<script>

function showBrowserInfo() {

var info = document.createTextNode("Информация о браузере: " + navigator.userAgent);

var newLine = document.createElement('br');

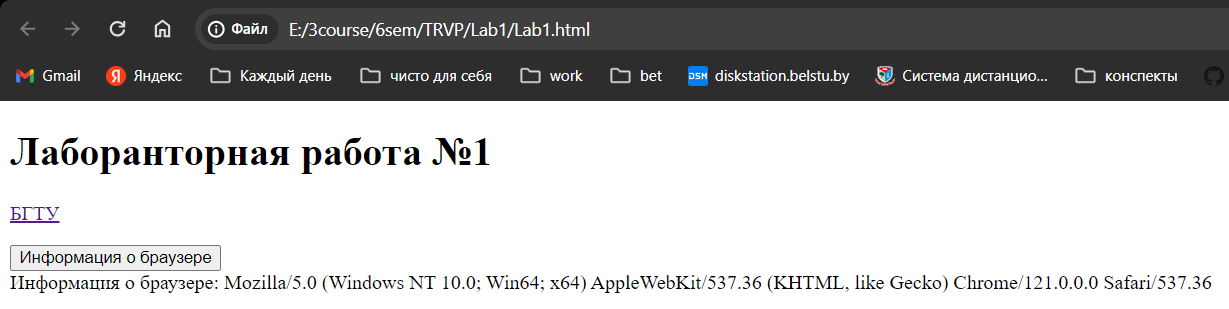
document.body.appendChild(newLine);

document.body.appendChild(info);

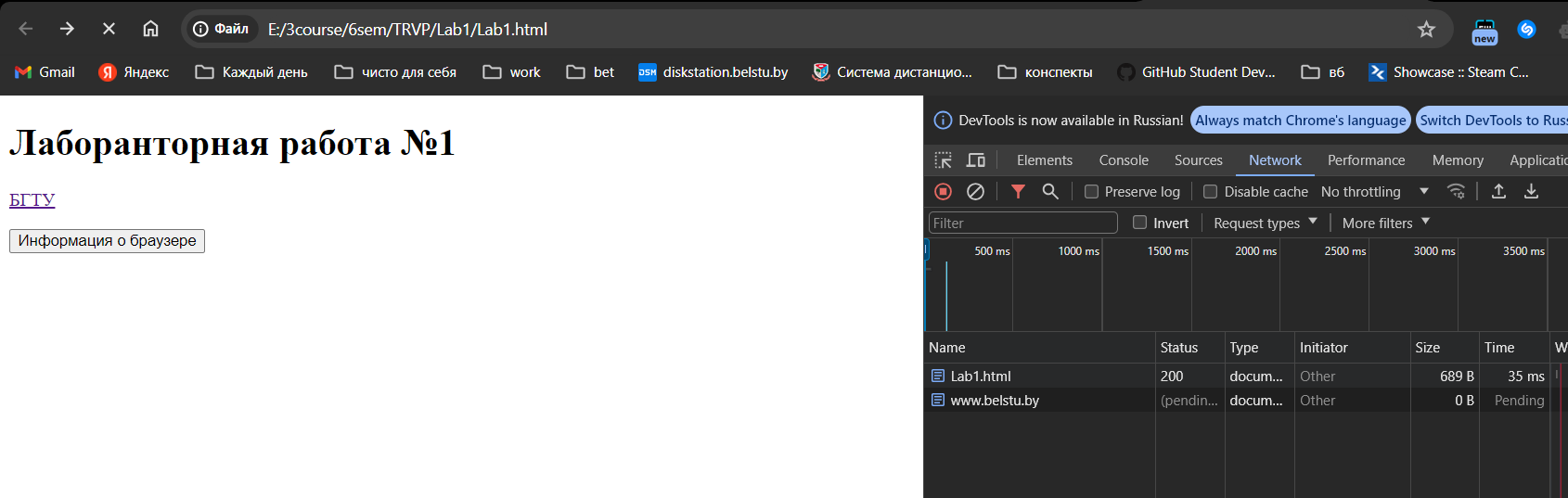
console.log("Hello World");

}

</script>

</html>



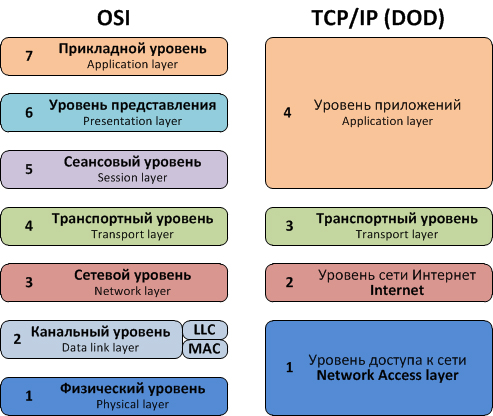


**Задание 4. Ответьте на вопросы**

1. **В каком документе описывает протокол HTTP?**

RFC (Request for Comments)

1. **Какого уровня HTTP-протокол в моделях OSI/ISO, TCP/IP?**

На прикладном уровне модели OSI/ISO и на уровне приложений в модели TCP/IP



1. **Что такое web-приложение?**

Веб-приложение - это программное обеспечение, которое запускается в веб-браузере. Оно позволяет пользователям взаимодействовать с данными и функциями через интернет.

1. **Какие порты использует HTTP-протокол?**

80 – HTTP

443 - HTTPS

1. **Что такое RFC?**

RFC (Request for Comments) - это серия документов, содержащих технические спецификации и стандарты, широко применяемые в Интернете.

RFC: жизненный цикл описан в RFC 2026, этапы: Draft Internet, Proposed Standard, Draft Standard, Internet Standard (RFC/STD), Historic (вышедшие из употребления). RFC может содержать не только стандарты, но и концепции (Experimental – результаты экспериментов, Informational – информационные, Best Current Practice – опыт применения).

1. **Почему HTTP-протокол называют ассиметричным?**

HTTP-протокол называют ассиметричным, потому что инициирование запросов к серверу осуществляется самим получателем, обычно веб-браузером

1. **Что такое MIME?**

MIME: Multipurpose Internet Mail Extensions - многоцелевые расширения Internet-почты. Используется и как стандарт кодирования Internet-сообщений.

