МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени Н.Э.Баумана

ОТЧЕТ

о прохождении производственной практики

Место прохождения практики: «разработка Веб-сайта»

Студент Нгуен Тхань Дат

1 курс, учебная группа ИУ9-22Б

Руководитель практики:

Директор:

Москва 2019

Содержание

Введение, постановка задачи ……………………………3

Описание Веб-сайта………………………………………..4

I:Front-end (UI / UX для веб-сайта)……….…..……..4

II: Back-end и база данных……………………….….13

Заключение ………………………………………………...22

**Введение, постановка задачи**

Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. (Wikipedia)

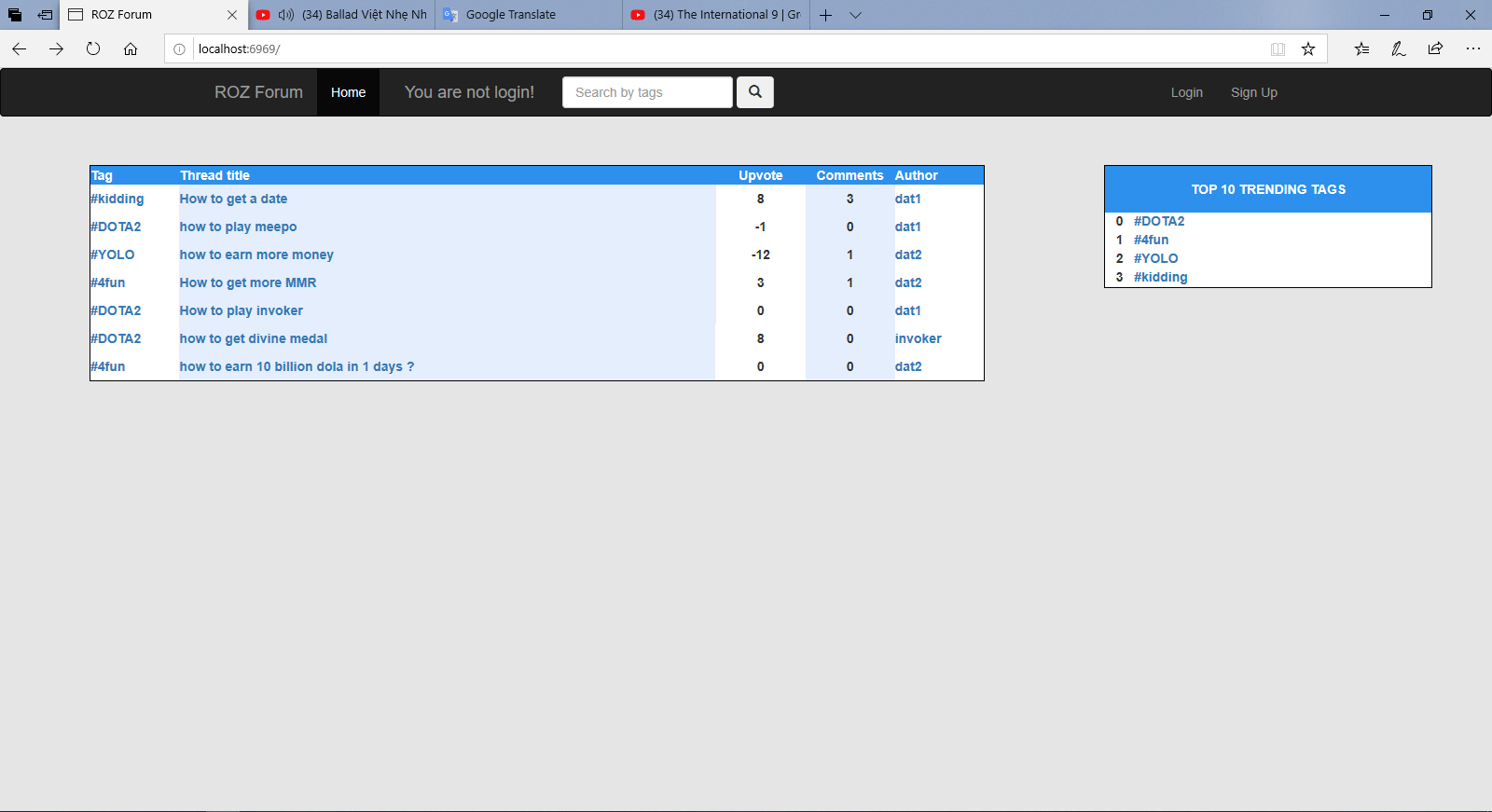
Описание проблемы: Создайте форум (веб-сайт) только с Javascript. Не используя Library или Framework. Форум (сайт) состоит из 2 частей. Первый - это Front-end, который написан на HTML, CSS и использует Javascript для доступа к HTML элементу - DOM. Второй - это Back-end, который использует NodeJS, написанный Javascript. Back-end создал две основные идеи: создание «HTTP Web Server» с помощью модуля “http” в NodeJS и использование файла JSON в качестве базы данных. Для удобства проверки я не использую никакой защиты для базы данных, аутентификации, Salting hasher method...

**Описание Веб-сайта**

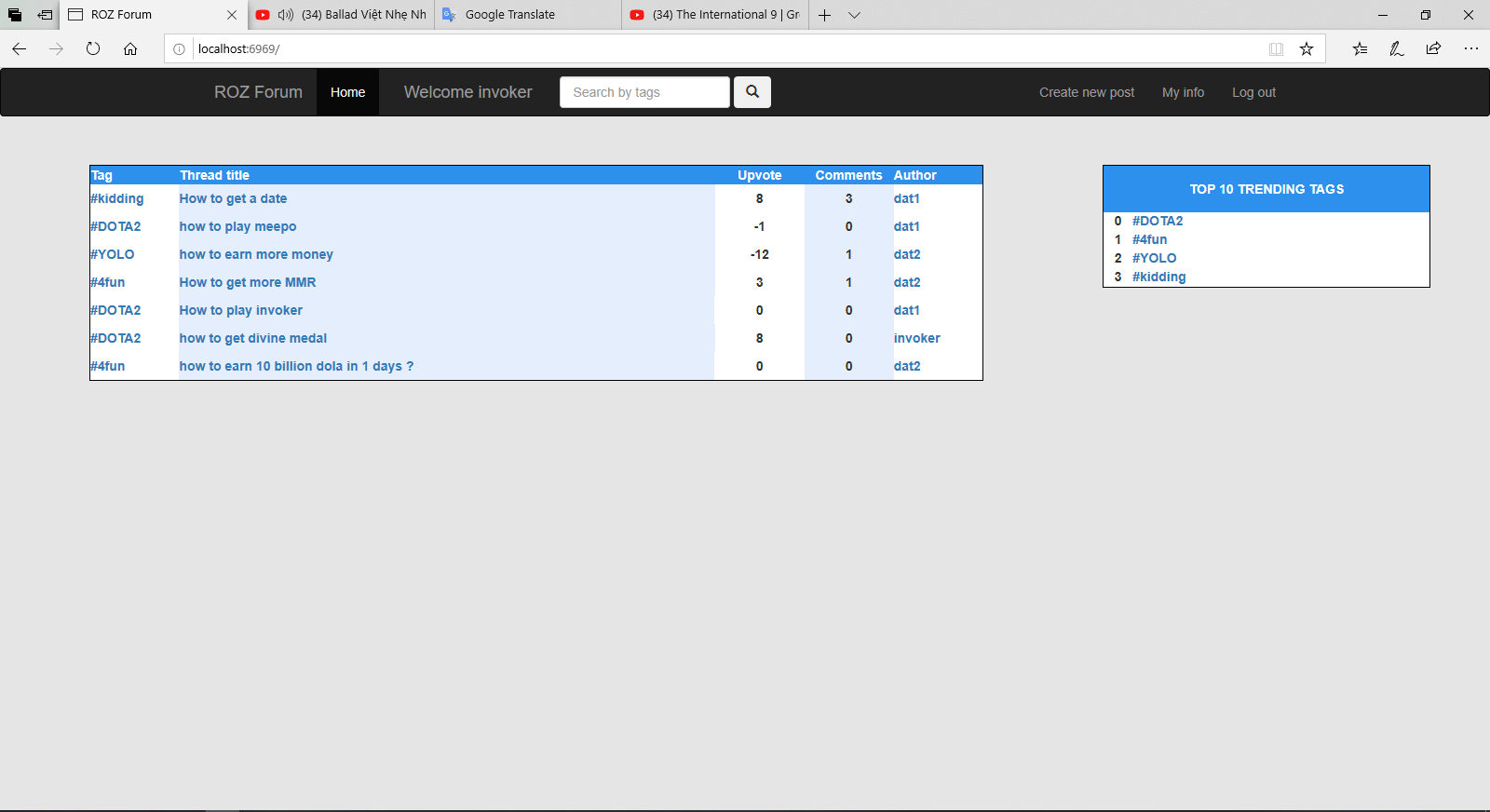
**I: Front-end (UI / UX для веб-сайта)**

**1: Главная страница**

*до входа пользователя*

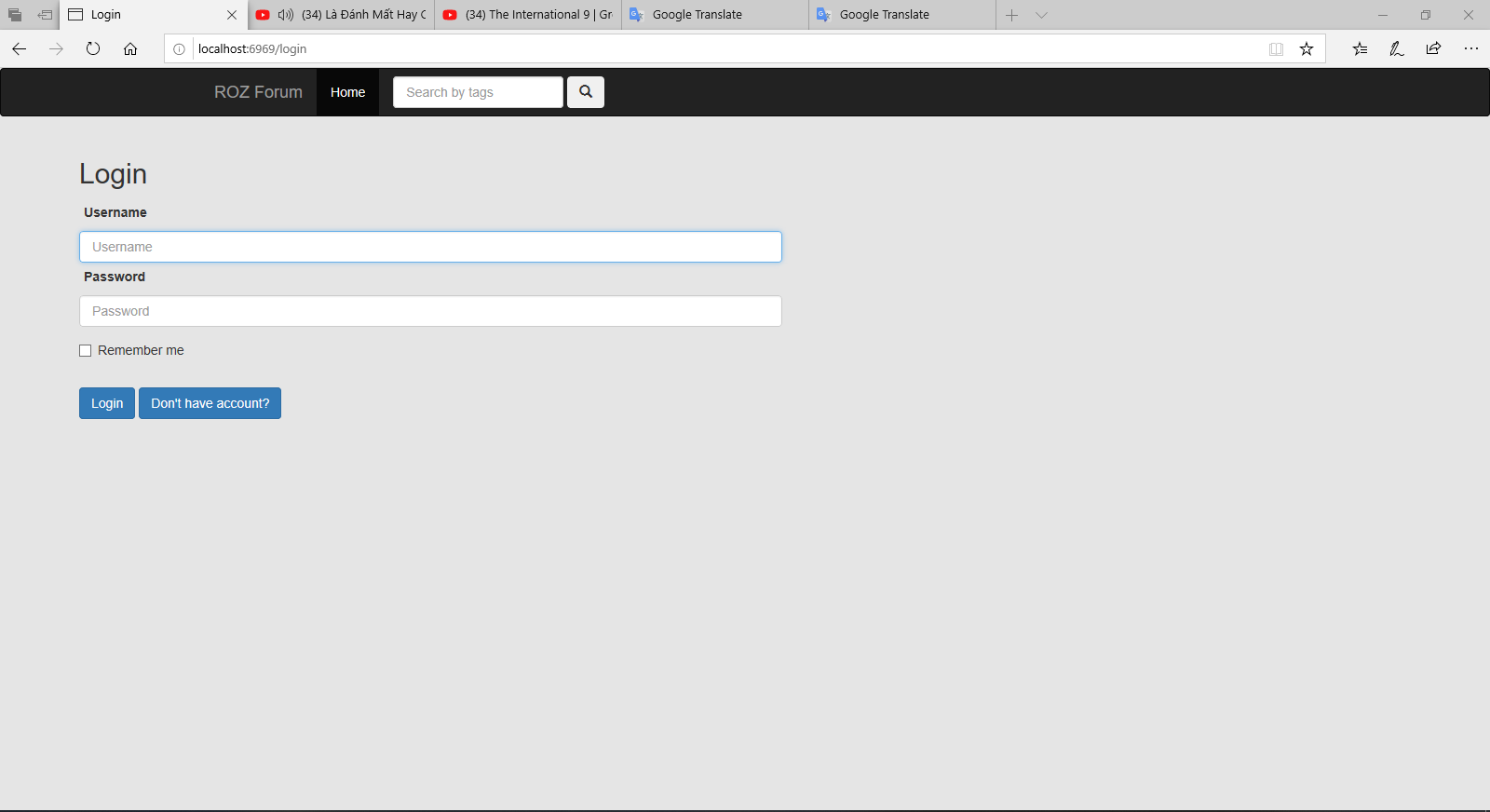


*после входа пользователя*



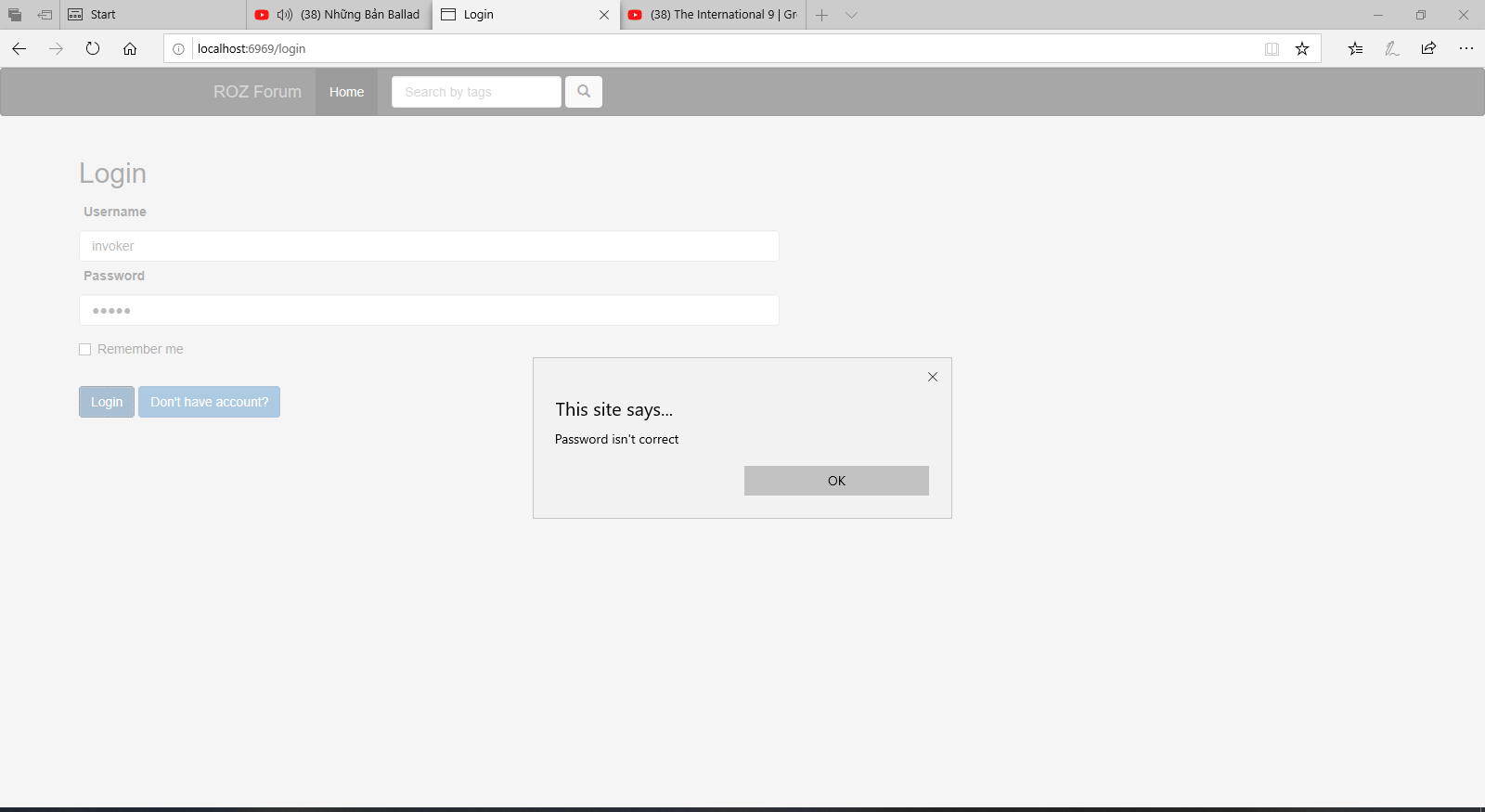
* В верхней части сайта есть кнопка перенаправления на главную страницу. Далее идет уведомление, оно будет меняться при входе пользователя в систему или нет. Далее это место для пользователя, ищущего тему по определенному “tag”. Если пользователь не авторизуется, отобразятся 2 кнопки «Войти» и «Зарегистрироваться». При входе пользователя в систему будут отображаться кнопки «Создать новую тему», «Информация о пользователе», «Выйти из системы».
* На главной странице есть 2 таблицы
  + Первый показывает все темы, написанные пользователем. Каждый ряд - это каждый тем. Первый столбец является “tag” тем. При нажатии будет перенаправлен на страницу, на которой показаны все темы с этим “tag”. Второй столбец - это заголовок темы, щелкающий для отображения всей этой темы (содержимое, комментарии, ...). Следующий столбец - «upvote», если кому-либо нравится тем, «upvote» будет увеличиваться, в противном случае «upvote» будет уменьшаться. Следующий столбец - количество комментариев в этой теме. Последний столбец - автор, который пишет эту тему. Нажав, чтобы показать эту информацию пользователя.
  + Второй показывает топ 10 трендовых “tag”. Нажав на каждый “tag”, чтобы показать все темы с этим “tag”.
* Kак это устроенo
  + Используя HTML и CSS для создания и организации
  + Используя JavaScript для доступа к HTML элементу – DOM
  + Используя xmlHttpRequest для доступа к базе данных.

**2: Страница для входа пользователя**

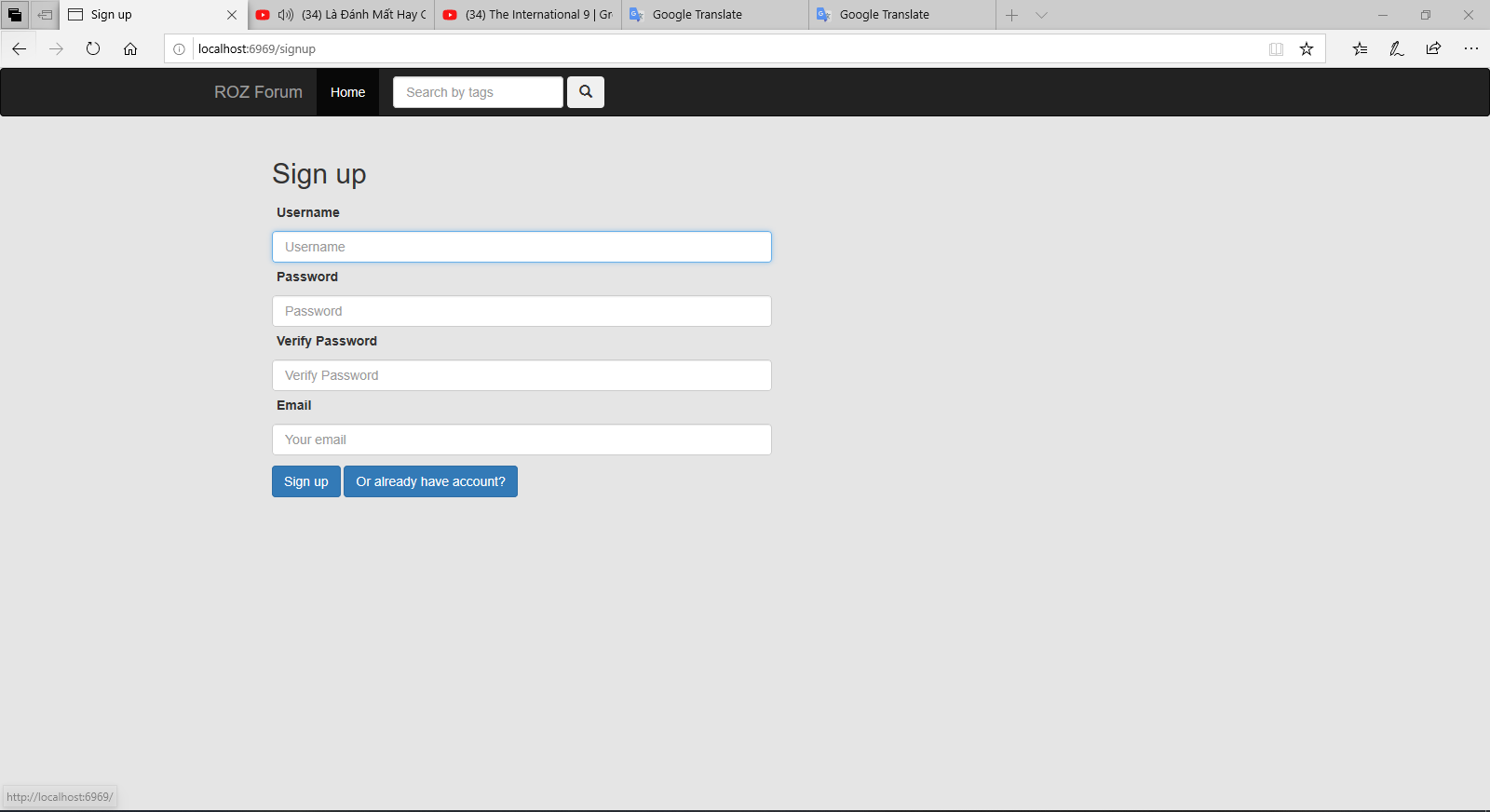


* Это HTML «form». С помощью HTTP-метода «POST» для отправки данных пользователя на сервер.
* Я обновил "onsubmit" на этой "form". Так что этот брaузер заметит, если пользователь напишет неправильно.

Kогда пользователь пишет пароль не правильно :

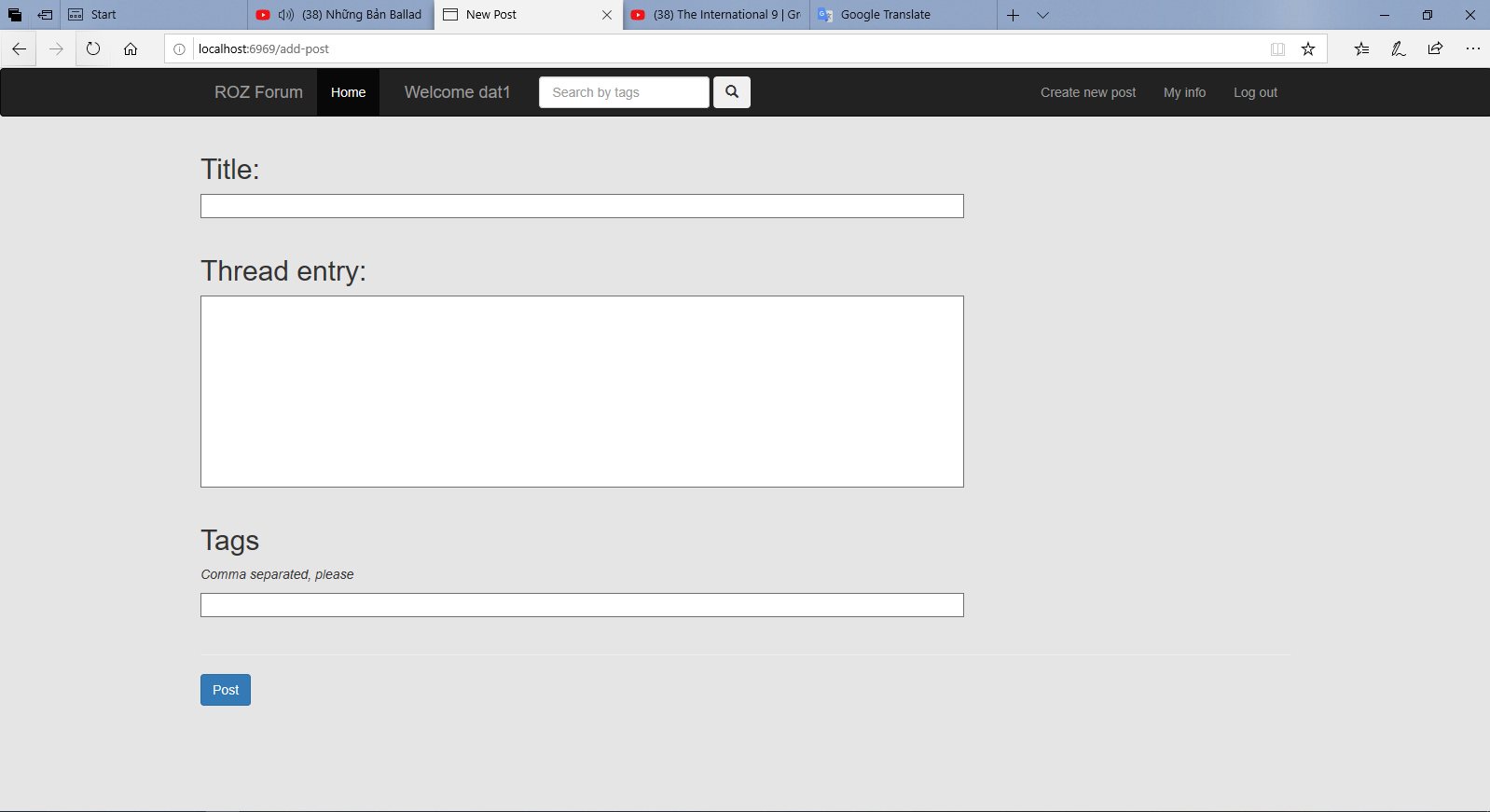


**3: Cтраница для регистрации пользователя**



* Это похоже на страницу для входа пользователя
* Я использовал "Regex” выражение, чтобы применить требование. Какой пользователь должен следовать за ними, чтобы зарегистрироваться

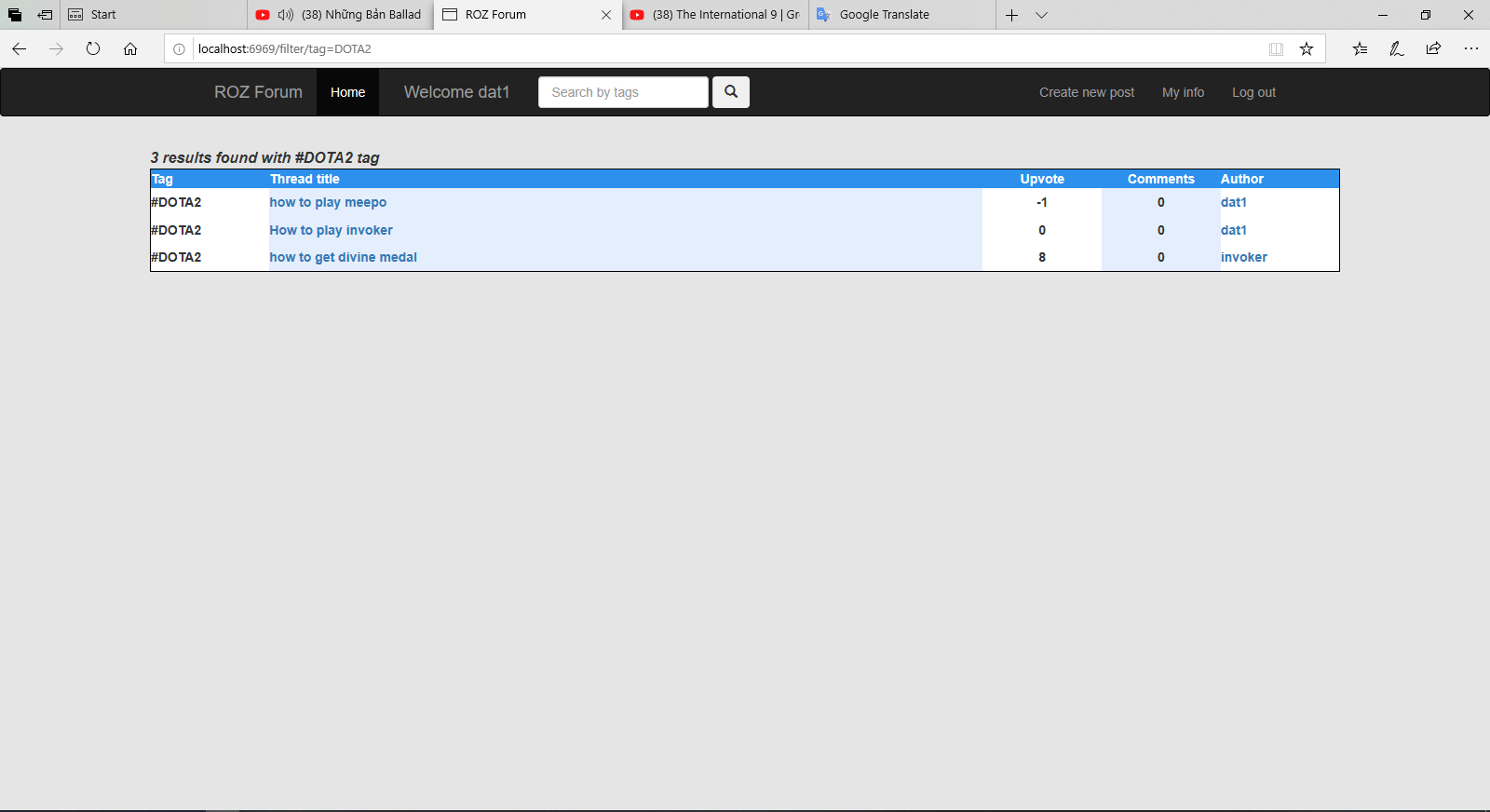
**4: Cтраница для пользователя написать новую тему**



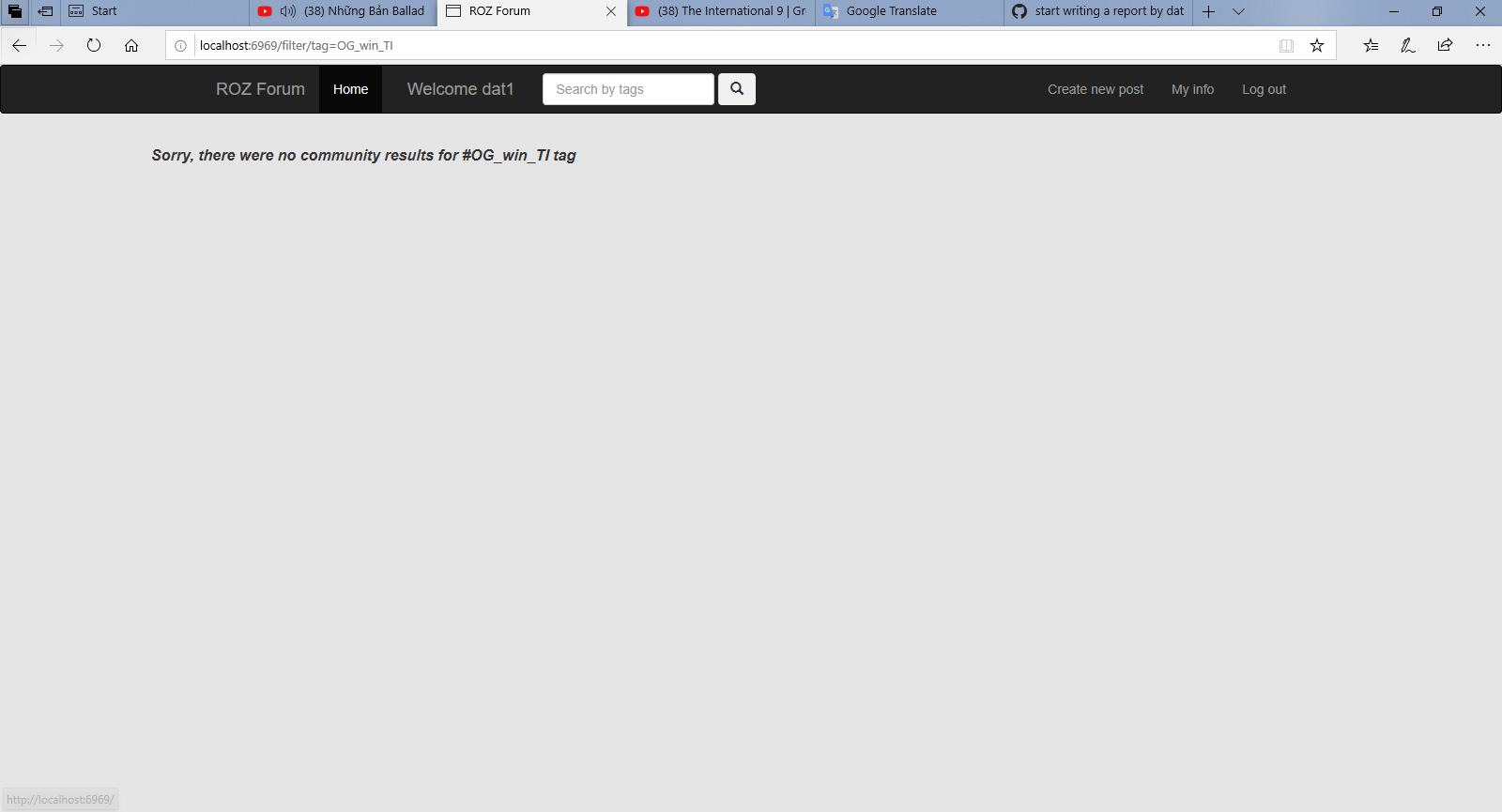
* Это HTML «form». С помощью HTTP-метода «POST» для отправки данных темы на сервер.

**5: Cтраница, показывающая все темы, которые пользователь ищет**

Если сервер найдет какой-либо тем, который совпадает с пользователем хочет найти :

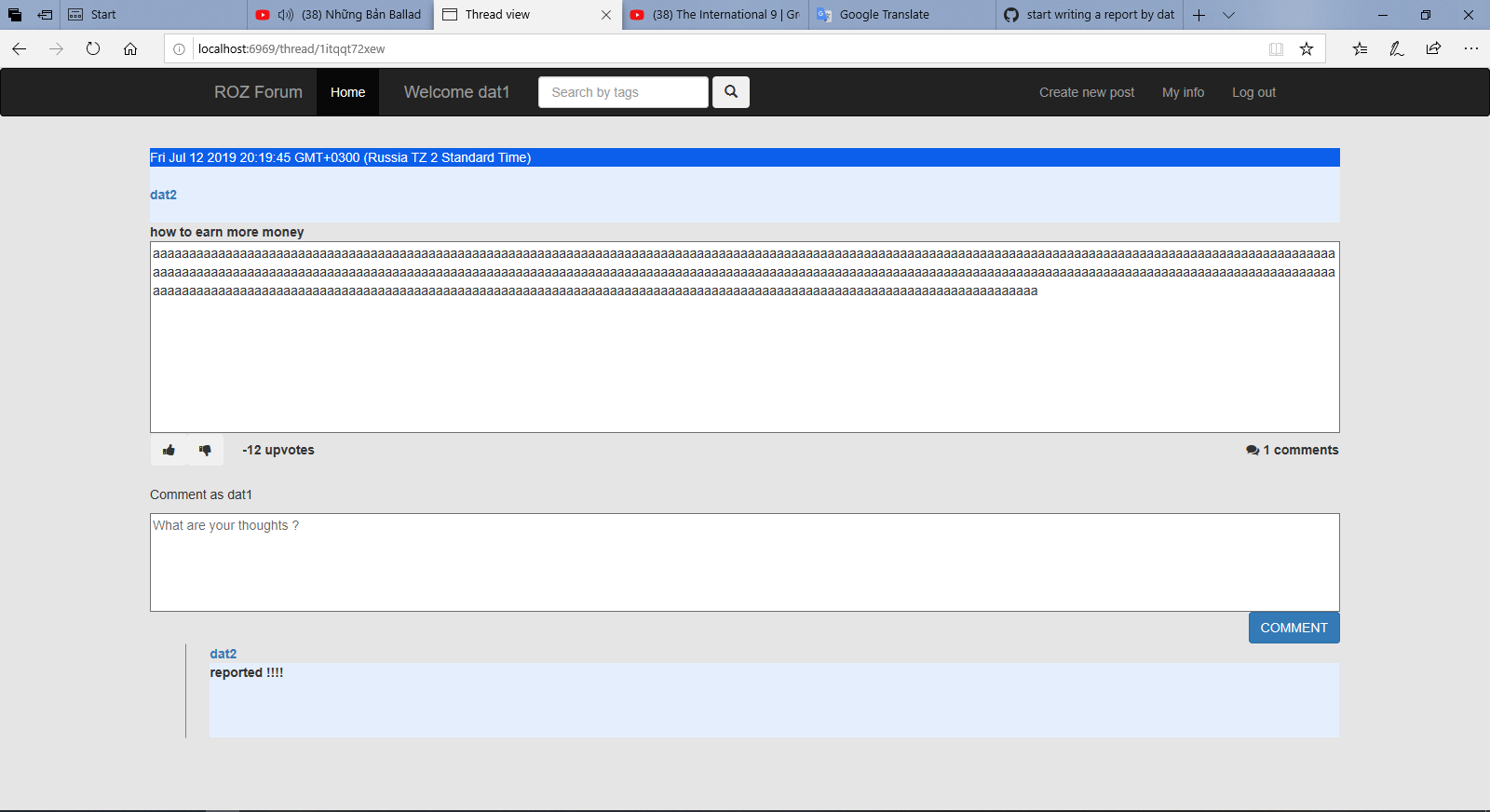


В противном случае:



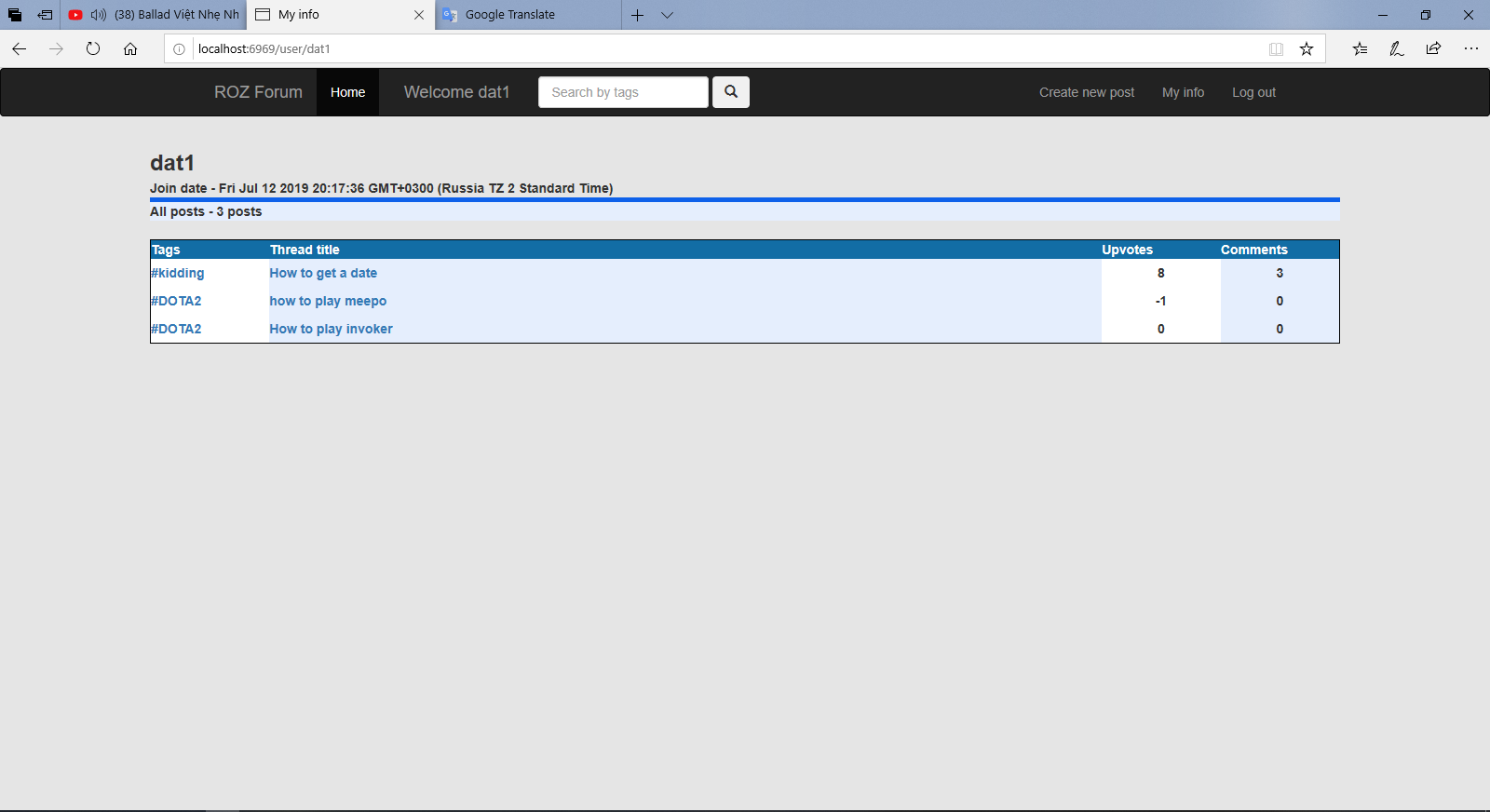
* Когда пользователь пишет “tag”, который он хочет найти в строке поиска (с энак «#» или нет, не имеет значения). Сервер будет искать в базе данных тем и находить все темы с этим “tag”.

**6: Cтраница для показа темы**



* Эта страница показывает
  + время, когда автор пишет эту тему
  + автор, пользователь, который написал эту тему
  + название и содержание этой темы
  + кнопка нравится / не нравится, количество «upvote», количество комментариев
  + место для написания комментария к этой теме (показывать только когда пользователь в сети)
  + все комментарии, которые написал другой пользователь
* Когда пользователь нажимает на темы, который он хочет видеть на главной странице, сервер просматривает базу данных и показывает точно темы.
* Кнопка «нравится / не нравится» - это HTML-метод «POST» для увеличения / уменьшения атрибута «upvote» в данном базe.
* Каждая база данных тем имеет свою базу данных для комментариев. Публикация комментария - это HTML-метод «POST», он отправляет серверу информацию об этом комментарии, затем сервер найдет эту тему в базе данных и добавит комментарий в эту базу данных.
* Пользователь также может изменить свои собственные комментарии, нажав кнопку «Edit» (эта кнопка отображается только для пользователя, который в сети). Эта кнопка имеет атрибут «onclick»,значит после нажатия выбранный HTML-элемент изменится на другой элемент.

**7: Страница для отображения информации о пользователе**

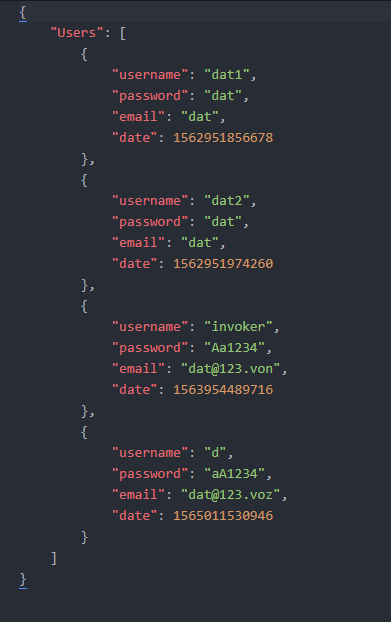


* Когда пользователь нажимает на имя пользователя, которое он хочет видеть, сервер просматривает базу данных пользователя, чтобы найти этого пользователя для показа (имя пользователя на сервере только одно, при регистрации на сервере не разрешается повторять имя пользователя)

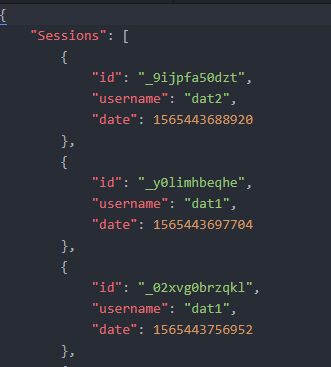
**II: Back-end и база данных**

**1: База данных описания:**

*База данных пользователей База данных тем*



*База данных пользовательских сессий*

База данных пользователей для сохранения всей информации о пользователях, которые были зарегистрированы.

База данных тем предназначена для сохранения всех тем, опубликованных пользователями.

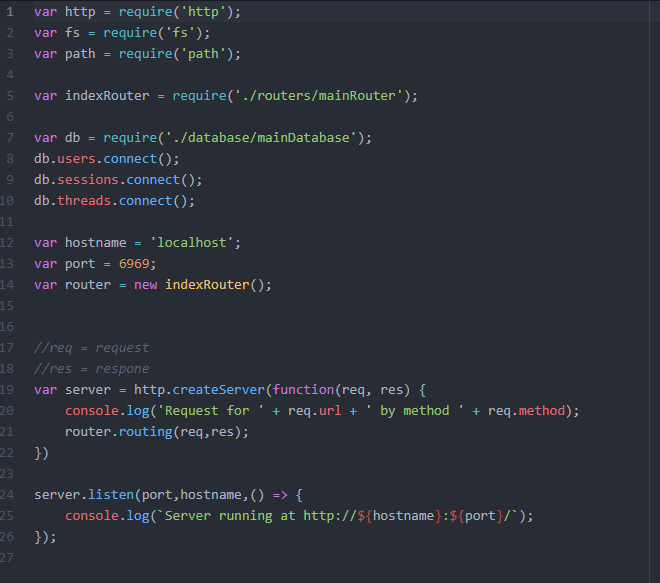
База данных сессий пользователей для сохранения файлов cookie пользователя, который находится в сети. Cookies - это информация, которую браузер может сохранить. Поэтому пользователи, которые входят в систему, могут оставаться на нашем сайте.

**2: Связь между сервером и базой данных**

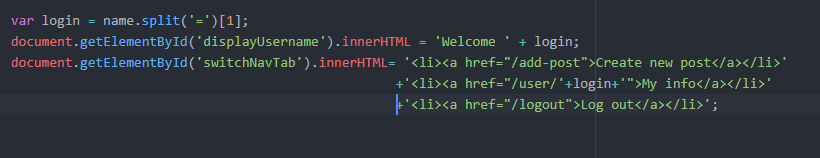


* Использование модуля javascript 'fs' для доступа к файловой системе. Это позволяет читать / записывать файл по системному пути.
* В первый раз файл базы данных JSON точно не существует. В это время мы получим код «ENOENT». Вот почему нам нужно создать новый файл JSON при получении этого кода.

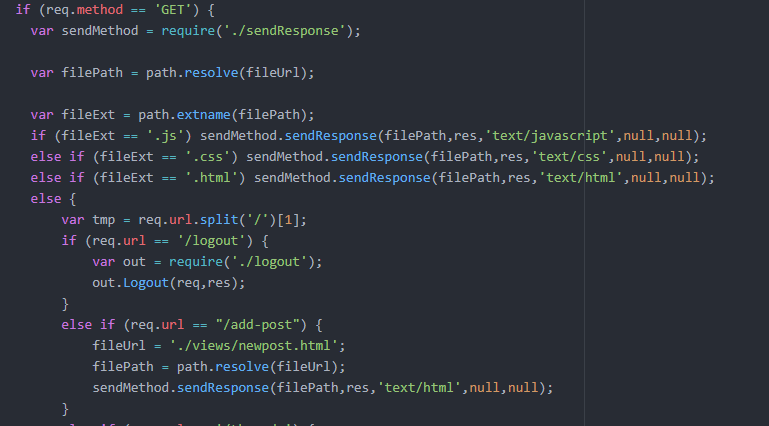
**3: Сервер (Back-end) и как он соединяется с front-end.**



* Использование модуля "http" в nodeJS для создания HTTP-сервера. Соединитесь с 3 базами данных, о которых я говорил ранее.
* «back-end» работают как некоторые функции с 2 основными атрибутами - «request» и «response». «request» как совокупность всей информации от «front-end» отправляется на «back-end». «response» - это набор действий, которые сервер отправляет обратно на «front-end».
* «back-end» распространяет «request» по его методу. Я использовал 2 метода «request» - это «GET» и «POST».
* Метод «GET» означает, что «front-end» только хочет видеть некоторую информацию. Таким образом, «back-end» только нужно искать эту информацию и показывать «front-end» с помощью «response».
* Пример o “GET” mетодe, перенаправить страницу:



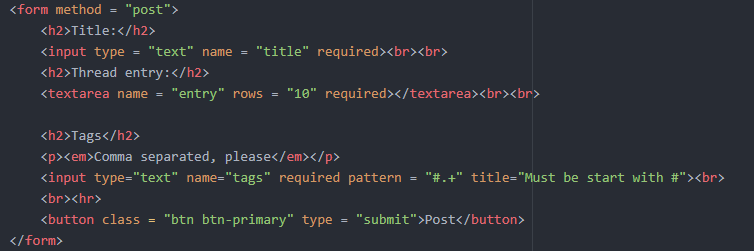
* Это «span» в HTML. Kогда пользователь нажимает на это, «front-end» отправляет «request» с методом «GET», который хочет, чтобы сервер показывал страницу с записью ссылки в «href».



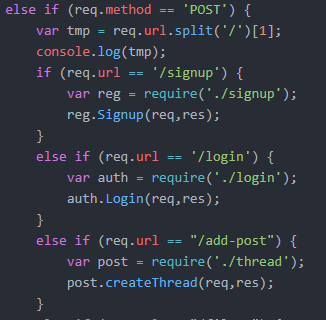
* Вся информация в атрибуте «request». «request.method» - метод запроса, «request.url» - ссылка, которую «front-end» хочет «back-end» найти. После получения запроса «back-end» будет обрабатывать и возвращать «front-end» то, что нужно, используя «response»



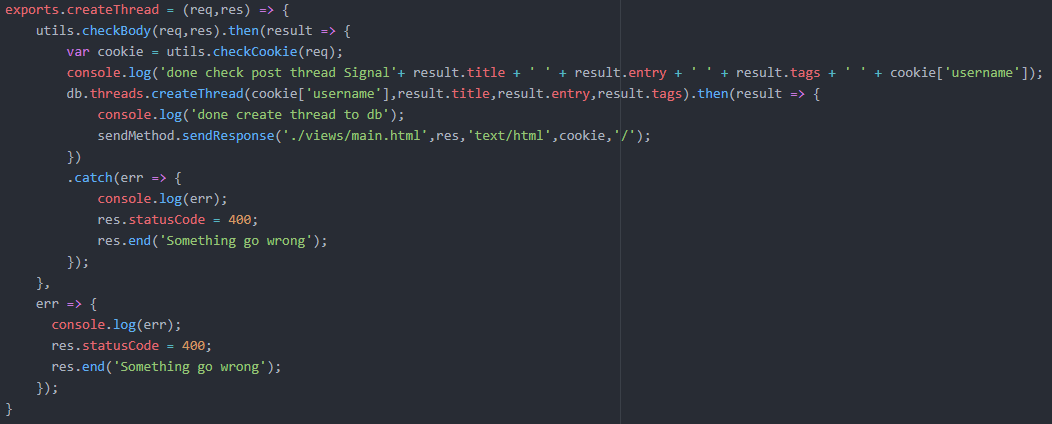
* Что касается метода «POST», то это похоже на метод «GET», но он не только видит, «front-end» также хочет, чтобы «back-end» обрабатывал некоторую информацию.
* Пример o “POST” mетодe:



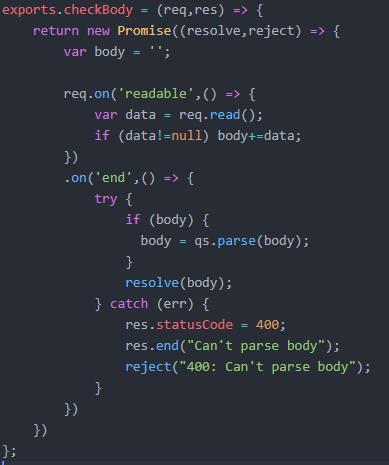
* Это «форма» в HTML с методом «POST». Он отправил некоторую информацию "back-end", которую пользователь пишет в "front-end". И хочет, чтобы "back-end" обрабатывал эту информацию. В этом случае пользователь хочет написать новую тему.

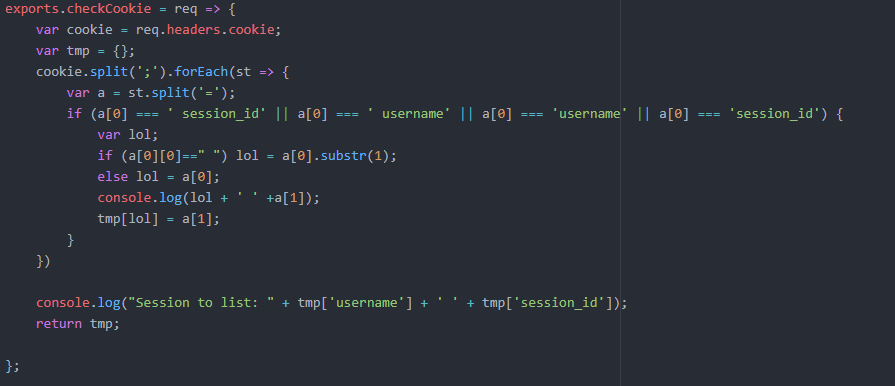


* «back-end» распознает «request» из «front-end» и начинает обрабатывать. После обработки "back-end" отправит обратно в "front-end" с помощью "response"



* Вот как «back-end» получает информацию из «request»



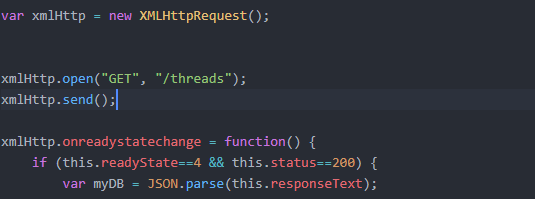


* Во-первых, как получить информацию, которую пользователь записывает в «форме» в «интерфейсе». Второй читает «cookies», чтобы найти, какой пользователь сейчас в сети
* В этом примере пользователь хочет написать новую тему. После «back-end» были получены все сведения, которые пользователь записывает в «front-end» (в данном случае это тема). «Back-end» свяжется с базой данных тем, чтобы добавить новую тему, которую написал пользователь.



* Отображение других страниц на веб-сайте, добавление комментариев в тему, добавление / неприязнь к теме аналогично приведенным мною примерам. Теперь я покажу технику, которую я использовал:

*1: Достигните к базе данных от "front-end"*



* Я использую xmlHTTPRequest для доступа к базе данных из "front-end". Это как метод «GET» для запроса «back-end», который хочет показать базу данных, которую я хочу.

*2: Достигните HTML-элемента - DOM с помощью JavaScript.*

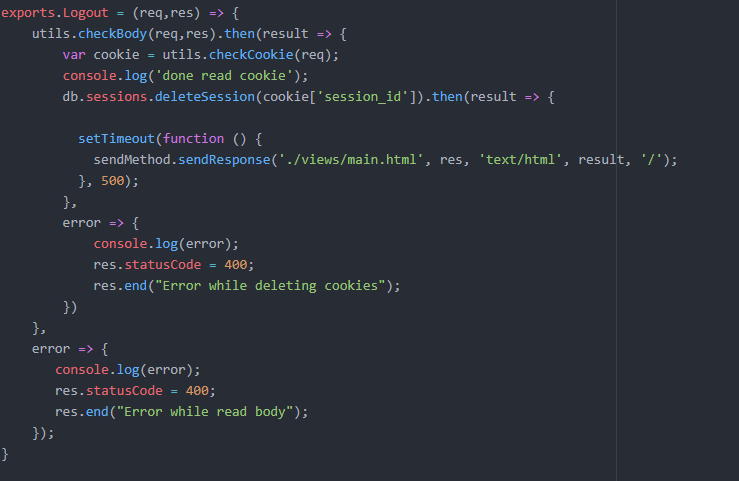
* Я использовал это много в "front-end". Это позволяет "front-end" не статично, это будет изменять интерфейс страницы в каждом случае. Например, верхняя панель страницы будет меняться при входе пользователя в систему, оставлять комментарии к теме и создавать место для темы, показывать только при входе пользователя в систему ....

*3: Держите пользователя оставаться в сети с помощью “cookies”*

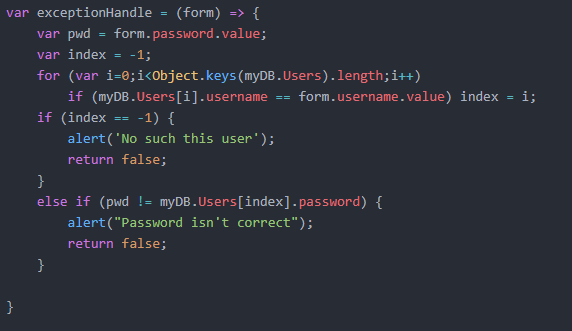
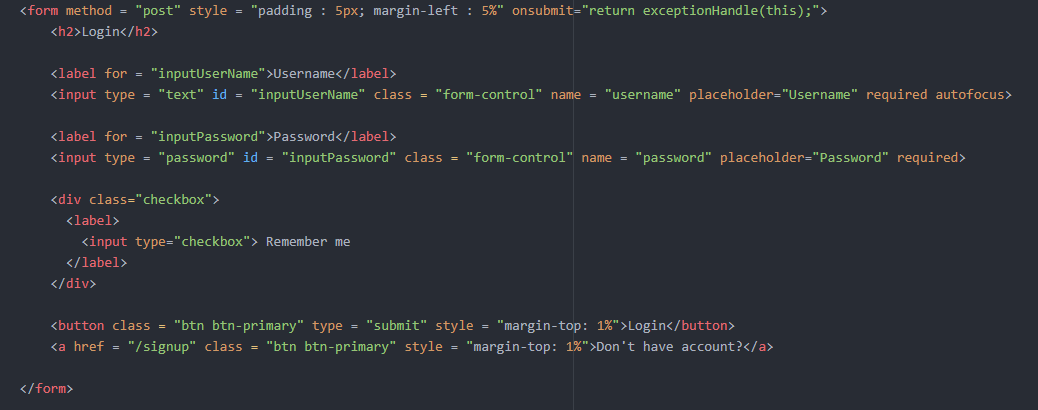
* «cookies» - это фрагменты данных, которые браузер может сохранять. Когда пользователь заходит на мой сайт, я создаю для него «cookie» и добавляю его в браузер, чтобы сохранить его.



* И если пользователь выходит из системы. Я удалю этот «cookie» из базы данных и установлю в браузере время «жизни» этого «cookie» на 0. Это значит, что браузер больше не спасет его.



*4: Добавление EventListener на страницу HTML. Это «onsubmit», «onclick», ... Пример с «onsubmit», после неправильного входа пользователя браузер предупредит пользователя*



Как обычно, необходимо создать некоторую защиту для базы данных, чтобы обеспечить безопасность информации пользователей. Некоторым популярным методом является использование «Salt hasher», предоставление «Token» для каждого пользователя, «двухсторонняя» защита между сервером и базой данных… Но это усложняет процесс проверки, поэтому я не использую их защиту. Без защиты проверка довольно проста, потому что вы можете просто получить доступ к базе данных и просмотреть данные.

**Заключение**

За время прохождения практики я выполнил задания, напрямую связанные с моей специальностью и справился с поставленной передо мною задачей. Созданный мной Web – портал может быть использовано на практике и внедрено на какое либо торговое предприятие.

Во время практики я улучшил свой знания в сфере Web – программирования, а также усвоил новые знания в сфере Web – дизайна.Помимо этого, мною были получены навыки работы с системами контроля версий (git)