

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №1

Выполнил:

Рыбалко Олег

Группа K33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

В данной лабораторной работе было необходимо сверстать сайт блога, где можно было бы публиковать статьи на различные темы.

Ход работы

1. Создание проекта

Для начала необходимо выбрать стек, при помощи которого будут выполняться дальнейшие лабораторные работы. В данном варианте будет использован фреймворк React, сервер Vite, Typescript в качестве языка разработки, Redux для хранения глобального стейта веб-сайта (например для хранения данных авторизации пользователя) и SCSS для описания стилей компонентов.

Для создания проекта с данным стеком выполним команду *yarn create vite*

```
> yarn create vite
yarn create v1.22.19
[1/4] 🔍 Resolving packages...
[2/4] 🚚 Fetching packages...
[3/4] 🔗 Linking dependencies...
[4/4] 🔗 Building fresh packages...

success Installed "create-vite@4.4.1" with binaries:
  - create-vite
  - cva
✓ Project name: ... lab1
✓ Select a framework: > React
✓ Select a variant: > TypeScript

Scaffolding project in /Users/rybalkooleg/lab1...

Done. Now run:

  cd lab1
  yarn
  yarn dev

✨ Done in 4.49s.
```

После выполнения данной команды мы получим директорию со структурой, показанной ниже

```
> tree .
.
├── README.md
├── index.html
├── package.json
├── public
│   └── vite.svg
├── src
│   ├── App.css
│   ├── App.tsx
│   ├── assets
│   │   └── react.svg
│   ├── index.css
│   ├── main.tsx
│   └── vite-env.d.ts
├── tsconfig.json
├── tsconfig.node.json
└── vite.config.ts

3 directories, 13 files
```

Для удобства создадим несколько папок, в которых будем хранить различные файлы:

- src/components – модули для компонентов
- src/layouts – модули для компонентов страниц
- src/locales – директория для хранения файлов интернализации
- src/store – директория для хранения модулей стейта Redux
- src/styles – директория для хранения общих стилей scss

2. Создание страниц регистрации и входа пользователя

При регистрации мы будем запрашивать у пользователя следующие данные:

1. Адрес электронной почты
2. Его уникальное имя для платформы
3. Пароль
4. Проверка пароля

Для выравнивания формы по центру страницы используем flex-box. При помощи bootstrap можно легко задать display: flex при помощи имени класса, добавив имя класса *d-flex*. Для выравнивания по центру используем имена классов *justify-content-center*, *align-items-center*.

Для компонентов ввода текста и кнопки используем стили Bootstrap, добавив лишь свойства закругления углов для полей в центре.

```
.formEmail {  
  margin-bottom: -1px;  
  border-bottom-right-radius: 0;  
  border-bottom-left-radius: 0;  
}  
  
.formPassword, .formUsername {  
  margin-bottom: -1px;  
  border-radius: 0;  
}  
  
.formRetypePassword {  
  margin-bottom: 10px;  
  border-top-left-radius: 0;  
  border-top-right-radius: 0;  
}
```

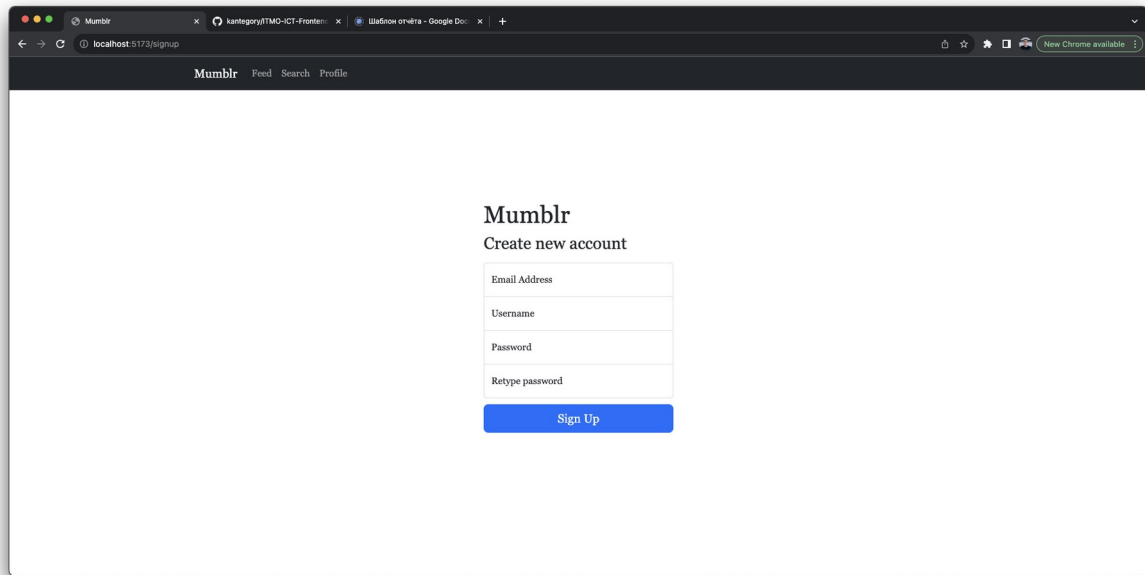
Для получения текста в каждом компоненте используется хук под названием *useTranslation*, который возвращает объект с переводами для текущего выбранного языка.

```
export function SignInLayout() {  
  const { t } = useTranslation('signin')  
  const { t: tGlobal } = useTranslation('global')
```

Более того, для перехода между страницами используется функция модуля *react-router-dom* *navigate*, которую мы получаем из хука *useNavigate*. Данный хук используется во множестве компонентов для осуществления переходов между страницами. А для того, чтобы изменять функции, которые зависят от реактивных переменных, мы используем хук *useCallback*, передавая зависимые переменные в качестве аргумента к этой функции

```
const navigate = useNavigate()  
const [userData, setUserData] = useState<UserData>({  
  username: '',  
  password: '',  
})  
  
const signIn = useCallback(() => {  
  store.dispatch(loginAction(userData))  
  navigate(`/profile/${userData.username}`)  
}, [userData, navigate])
```

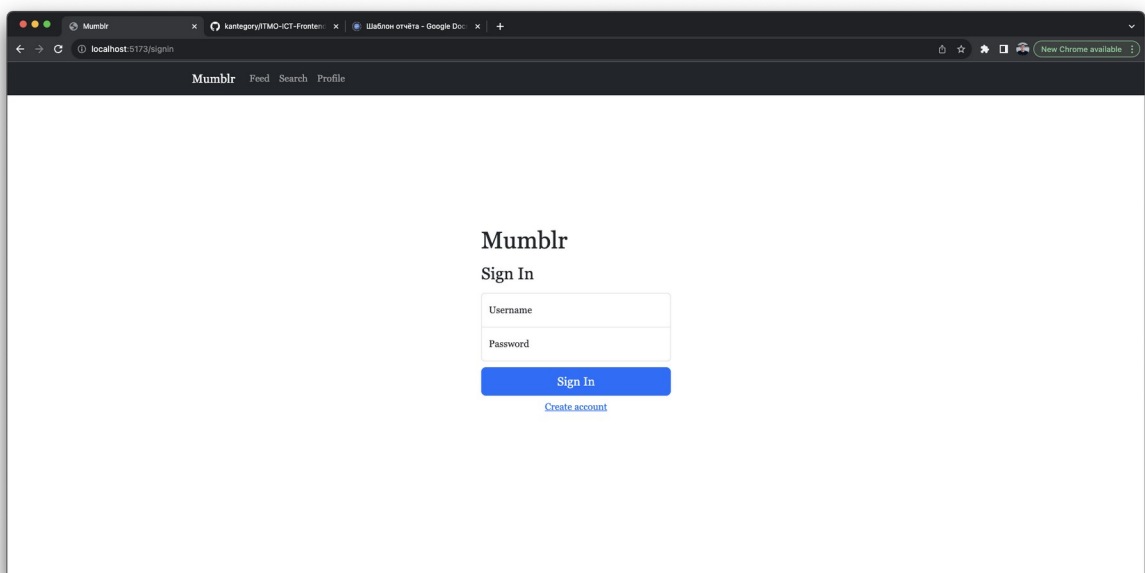
В итоге получаем страницу, как показано ниже



The screenshot shows a web browser window with the URL `localhost:5173/signup`. The page has a dark header with the 'Mumblr' logo and links for 'Feed', 'Search', and 'Profile'. The main content area is white and features the 'Mumblr' logo and the text 'Create new account'. Below this is a form with four input fields: 'Email Address', 'Username', 'Password', and 'Retype password'. A blue 'Sign Up' button is positioned at the bottom of the form.

После нажатия на кнопку *Sign Up* мы записываем данные в `localStorage` и обновляем на `redux-state`, для дальнейшей проверки данных пользователя. `Redux` нам необходим для достижения реактивности, так как `localStorage` не поддерживает ее из коробки.

Страница входа пользователя существенно не отличается от страницы регистрации и выглядит, как показано на скриншоте ниже.



The screenshot shows a web browser window with the URL `localhost:5173/signin`. The page has a dark header with the 'Mumblr' logo and links for 'Feed', 'Search', and 'Profile'. The main content area is white and features the 'Mumblr' logo and the text 'Sign In'. Below this is a form with two input fields: 'Username' and 'Password'. A blue 'Sign In' button is positioned at the bottom of the form, with a link 'Create account' below it.

3. Создание страницы пользователя

На странице пользователя будет отображаться следующая информация:

1. Уникальное имя пользователя
2. Аватар пользователя, полученный при помощи <https://robohash.org>
3. Описание профиля
4. Список публикаций пользователя
 1. Текст
 2. Количество лайков
 3. Кнопка для удаления публикации (если мы находимся в своем профиле)
5. Кнопка для создания новой публикации

Каждая публикация – это отдельный компонент, в котором показано название, текст и количество лайков. Список данных компонентов рендерится в Bootstrap-контейнере и каждый пост – это новая строка (row).

MARS

Mars is one of the most explored bodies in our solar system, and it's the only planet where we've sent rovers to roam the alien landscape. NASA missions have found lots of evidence that Mars was much wetter and warmer, with a thicker atmosphere, billions of years ago. Mars was named by the Romans for their god of war because its reddish color was reminiscent of blood. The Egyptians called it "Her Desher," meaning "the red one." Even today, it is frequently called the "R...




MERCURY

Mercury—the smallest planet in our solar system and nearest to the Sun—is only slightly larger than Earth's Moon. Its surface is covered in tens of thousands of impact craters. From the surface of Mercury, the Sun would appear more than three times as large as it does when viewed from Earth, and the sunlight would be as much as 11 times brighter. Despite its proximity to the Sun, Mercury is not the hottest planet in our solar system— that title belongs to nearby Venus, ...

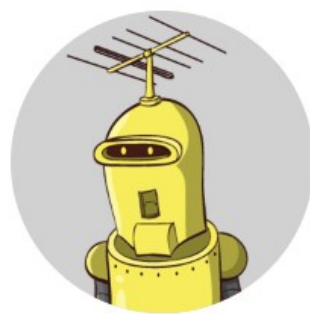


Для создания новой публикации используется кнопка в виде ссылки (при помощи btn-link), которая открывает модальное окно с формой для заполнения.



The image shows a modal window titled "New Post" with a close button (X) in the top right corner. Inside the modal, there are two input fields: "Title" and "Body". The "Body" field is a larger text area with a small icon in the bottom right corner. At the bottom right of the modal, there is a blue button labeled "Post". The modal is overlaid on a background that shows a dark header and a light gray body with some text visible at the bottom: "MA" and "Mars is one of the most explored bodies in our solar system, and it's the only".

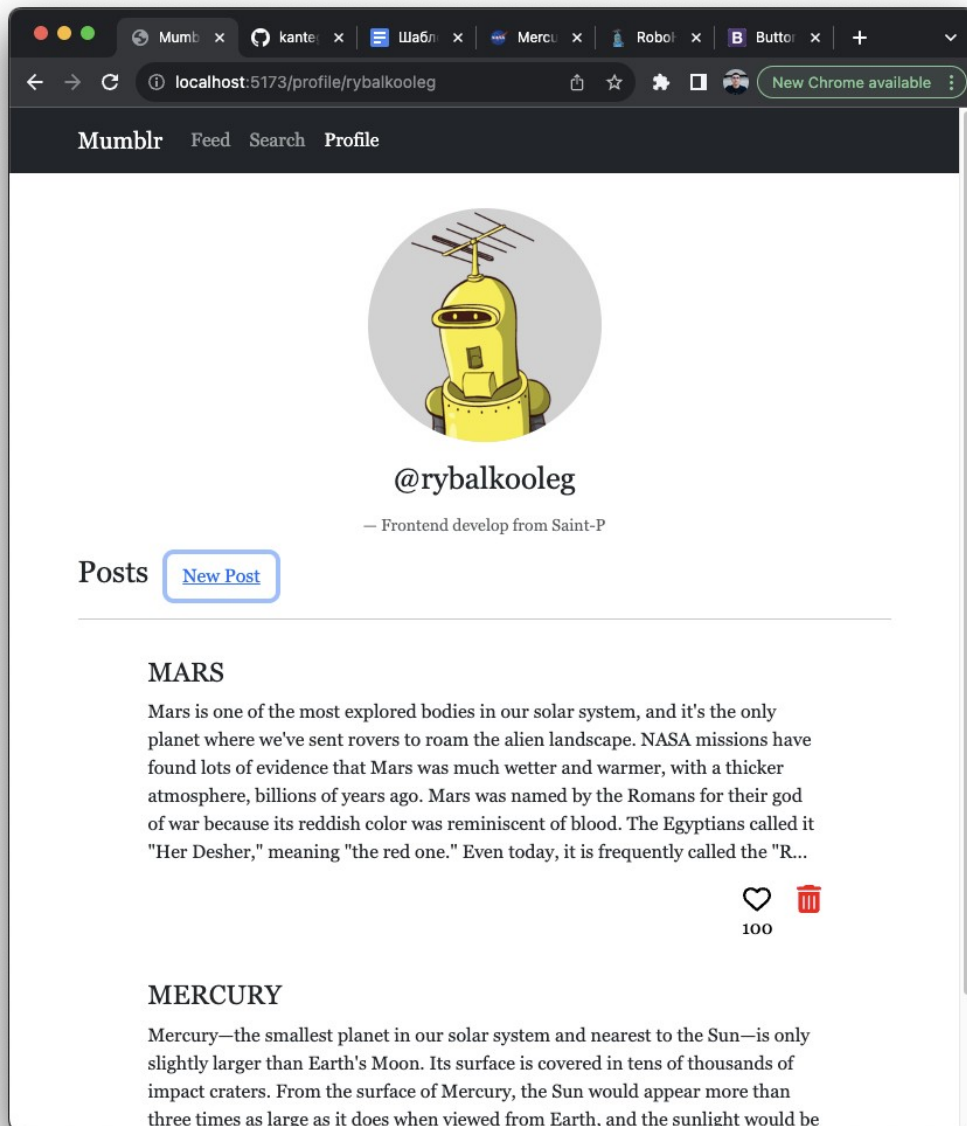
На странице профиля пользователя в самом верху отображается информация о нём и фотография, для которой прописан @media стиль, который уменьшает фотографию для устройств с маленькой шириной экрана.



@rybalkooleg

— Frontend develop from Saint-P

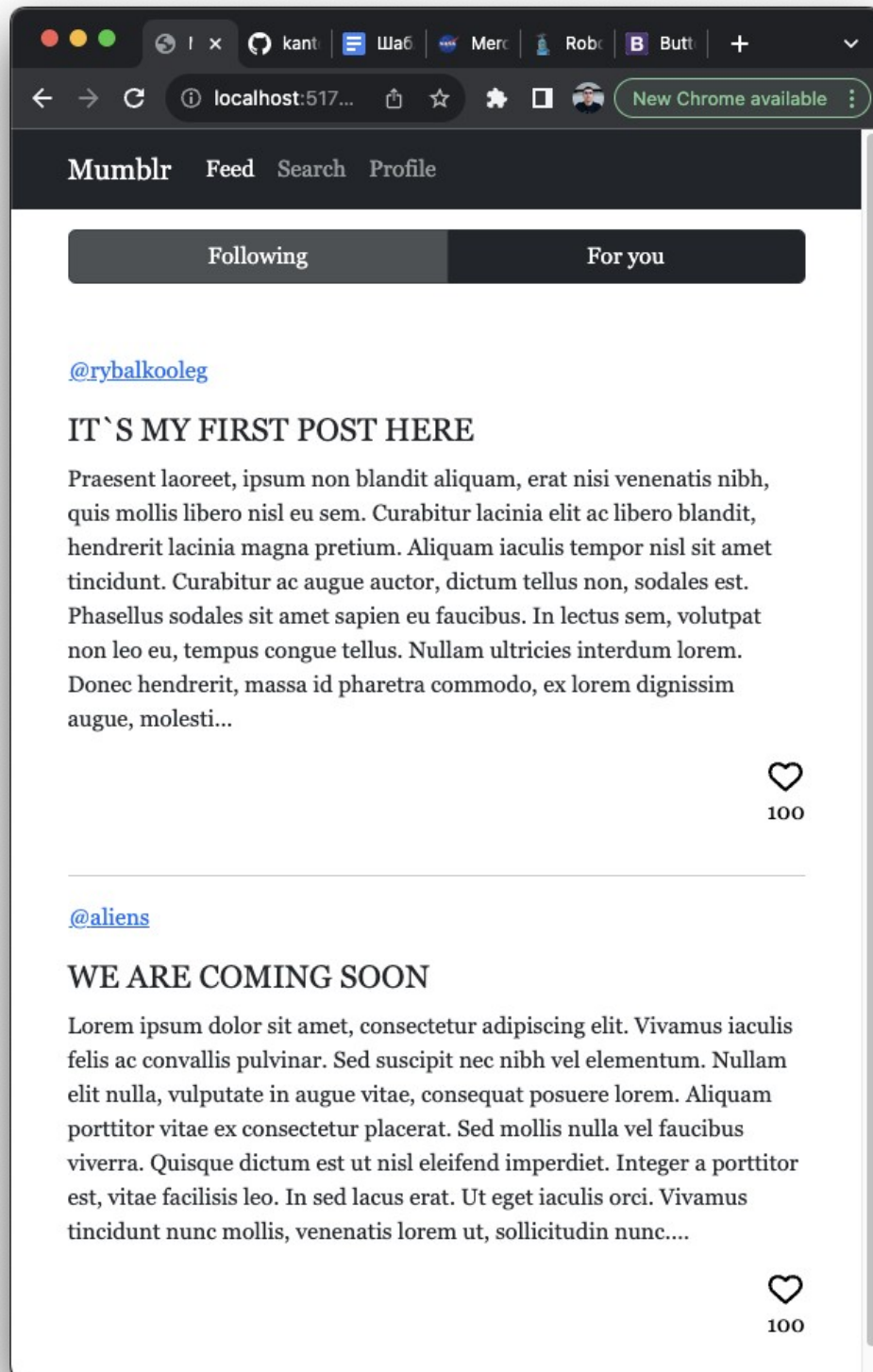
Вся страница пользователя выглядит так, как показана на скриншоте ниже

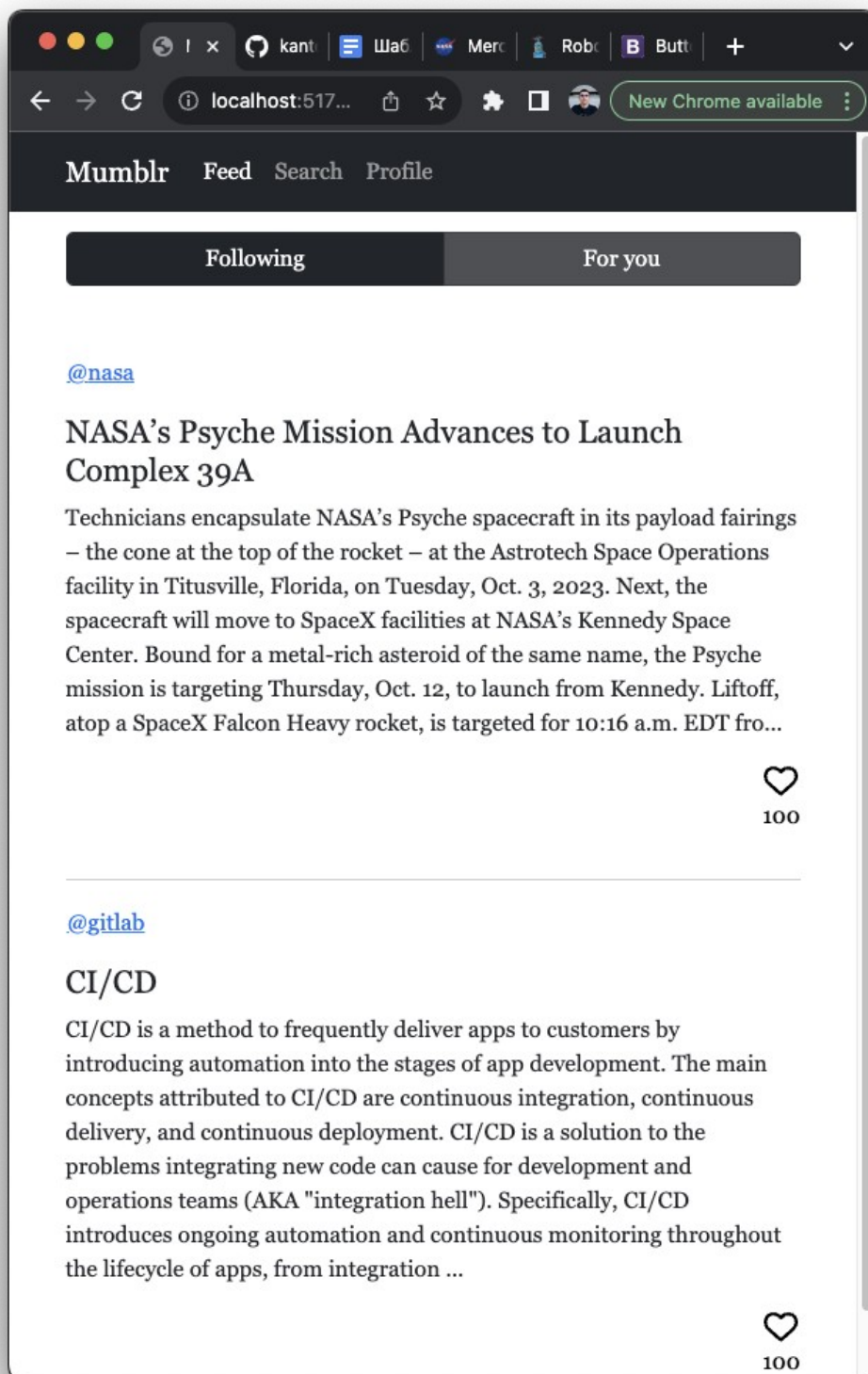


4. Создание страницы для ленты публикаций

Для создания страницы с лентой публикаций нам необходимо создать новый компонент для поста, так как на предыдущем компоненте нет информации о создателе публикации.

Страница с лентой будет иметь два состояния – показ ленты, состоящей из публикаций пользователей, на которых мы подписаны и также лента с рекомендациями. Два этих состояния показаны ниже.



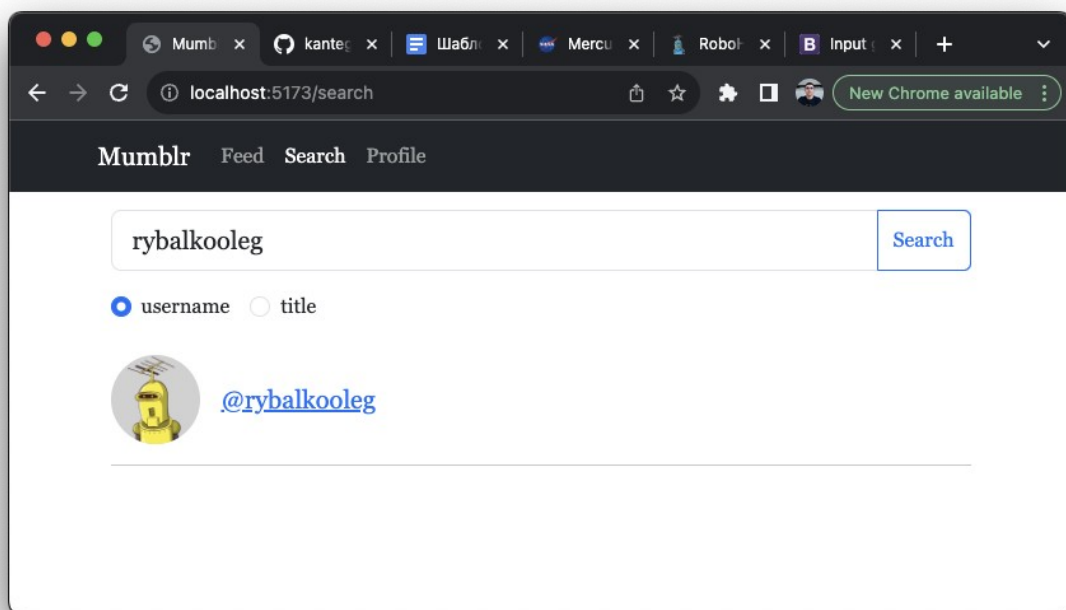


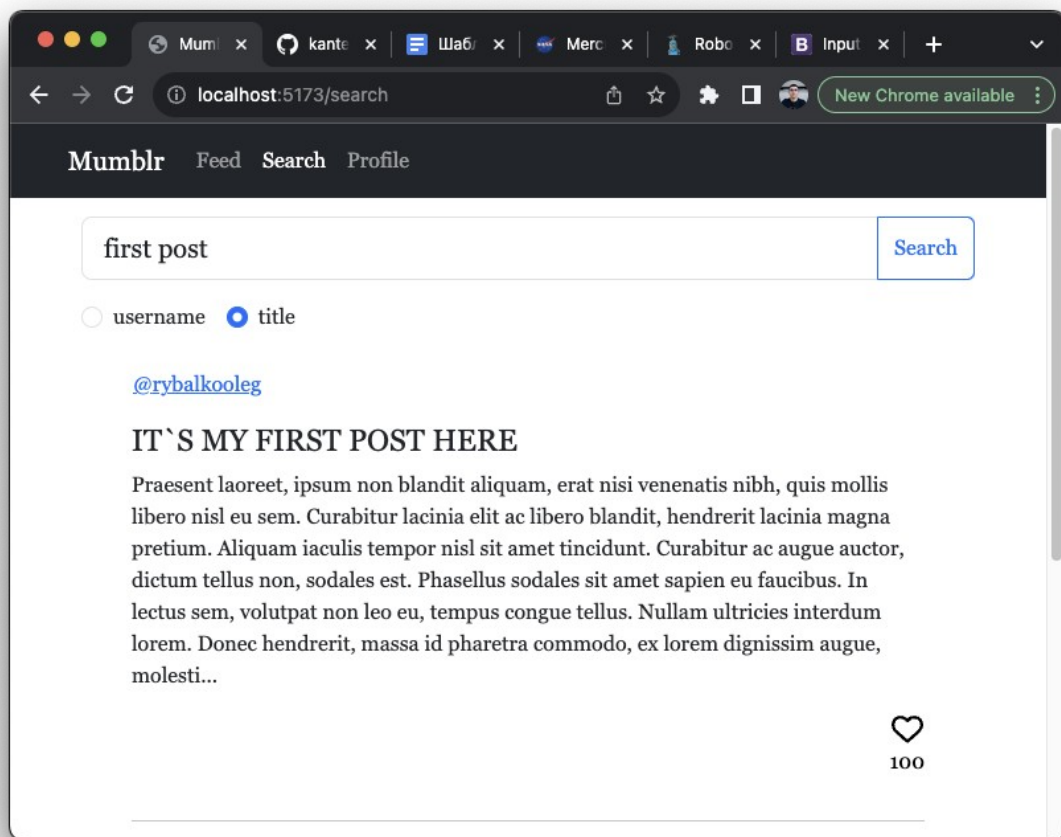
5. Создание страницы с поиском

На странице поиска у нас будут доступны два фильтра поиска:

1. Поиск по имени пользователя
2. Поиск по названию статьи

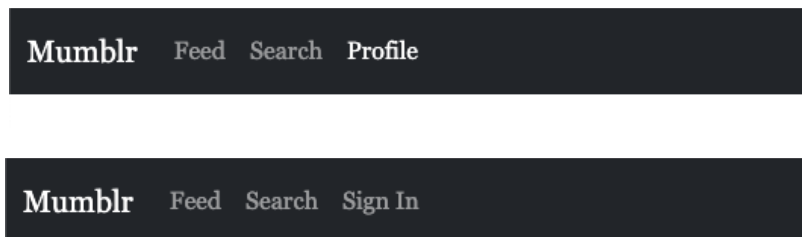
Для реализации строки поиска будем использовать *div* с классом *input-group*, так как нам необходимо объединить строку ввода и кнопку для поиска. Ниже строки поиска мы расположим две radio-кнопки, которые будут отвечать за фильтр поиска.





6. Создание navbar

Для создания *navbar* я воспользовался уже готовыми Bootstrap-стилями и реализовал динамическое отображение кнопок в зависимости от состояния пользователя — если пользователь вошел в аккаунт, то показывается кнопка для перехода в профиль, иначе — кнопка для входа в аккаунт.



Вывод

В данной лабораторной работе я смог на практике применить знания, полученные о фреймворке Bootstrap и создать макет веб-сайта

блога, который в дальнейших лабораторных работах я буду подключать к API. На данном этапе у нас получился сверстаный сайт, на котором все данные явно прописаны в коде и нет получения, обновления или удаления данных на сервере.