САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа 2: взаимодействие с внешним API

Выполнил:
Беломытцев А. И.

Группа:
K3439

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Варианты остаются прежними. Теперь Вам нужно привязать то, что Вы делали в ЛР1 к внешнему API средствами fetch/axios/xhr. Реализуйте моковое API средствами JSON-сервера и подключите к нему авторизацию, как в примерах, которые мы рассматривали в рамках тем "Иммитация работы с API".

Например, для приложения для просмотра прогнозов погоды задание выглядит следующим образом:

Реализовать получение погоды (прогноз на ближайшие 7 дней) из открытого API OpenWeatherMap, взависимости от геолокации пользователя. Реализовать вывод полученного прогноза в виде 7 карточек в три ряда (первый ряд - крупная карточка, второй ряд - три карточки в меньшем размере, третий ряд - четыре карточки в маленьком размере).

Ход работы

Так же как и для предыдущей лабораторной в качестве варианта выбрал сайт для моего проекта браслета для повышения продуктивности. Для страницы с возможностью фильтрации проект списка полезных ютуб каналов. Используется Bootstrap 5.3, что позволяет использовать тёмную тему. Для бэкенда использовался FastAPI.

1 Лендинг

Бэкенд для лэндинга очень прост, что логично (листинг 1). Таким же образом предоставляются и другие страницы. А дополнительное взаимодействие с API не требуется.

Листинг 1 – Бэкенд лэндинга

```
@router.get('/')
async def home(request: Request):
  return templates.TemplateResponse(
    request=request, name='home.html'
  )
```

2 Вход

На листинге 2 показан JavaScript код для работы формы для входа на сайт с возможностью перейти к регистрации.

Листинг 2 – JavaScript код для входа

```
const formElement = document.querySelector("form");
formElement.addEventListener("submit", async (e) => {
  e.preventDefault();

  const formData = new FormData(formElement);
  const urlData = new URLSearchParams();
  urlData.append("username", formData.get("username"));
  urlData.append("password", formData.get("password"));

  const response = await fetch("/auth/token", {
    method: "POST",
    headers: {
      "Content-Type": "application/x-www-form-urlencoded",
    },
    body: urlData,
  });

  const data = await response.json();
  if (response.ok) {
    localStorage.setItem("access_token", data.access_token);
    window.location.href = "/profile";
  } else {
    alert("Sign in failed");
  }
});
```

3 Регистрация

На листенге 3 показан JavaScript код для работы формы для регистрации на сайте с возможностью перейти ко входу.

Листинг 3 – JavaScript код для регистрации

```
const formElement = document.querySelector("form");
document.querySelector("form").addEventListener("submit",
async (e) => {
  e.preventDefault();

  const formData = new FormData(formElement);

  console.log(Object.fromEntries(formData));

  const response = await fetch("/auth/", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(Object.fromEntries(formData)),
  });

  if (response.ok) {
    window.location.href = "/login";
  } else {
    alert("Sign up failed");
  }
});
```

4 Профиль

На листенге 4 показан JavaScript код для страницы профиля. На ней есть имя пользователя, email, ID, дата регистрации, о пользователе, информация о браслете, список добавленных пользователем ютуб каналов, кнопки для выхода и удаления аккаунта.

Листинг 4 – JavaScript код для профиля

```
// Check authentication first
const token = localStorage.getItem("access_token");
if (!token) {
  window.location.href = "/login";
}

function logout() {
  localStorage.removeItem("access_token");
  location.reload();
}

function toggleApiKey() {
  const apikeyInput = document.getElementById("apikey");
  const eyeIcon = document.getElementById("eye-icon");
  if (apikeyInput.type === "password") {
    apikeyInput.type = "text";
    eyeIcon.className = "bi bi-eye-slash";
  } else {
    apikeyInput.type = "password";
    eyeIcon.className = "bi bi-eye";
  }
}

function copyApiKey() {
  const apikeyInput = document.getElementById("apikey");
  navigator.clipboard.writeText(apikeyInput.value);
  const btn = event.target.closest("button");
  const originalHTML = btn.innerHTML;
  btn.innerHTML = '<i class="bi bi-check"></i>';
  setTimeout(() => {
    btn.innerHTML = originalHTML;
  }, 1500);
}

function toggleChannels() {
  const container = document.getElementById("channels-container");
  const btn = document.getElementById("toggle-channels-btn");
  const icon = btn.querySelector("i");
  if (container.style.display === "none") {
    container.style.display = "block";
    btn.innerHTML = '<i class="bi bi-chevron-up me-1"></i>Hide Channels';
  } else {
    container.style.display = "none";
    btn.innerHTML = '<i class="bi bi-chevron-down me-1"></i>Show Channels';
  }
}

async function getData() {
  const response = await fetch("/profile-data", {
    method: "GET",
    headers: {
      Authorization: "Bearer " + token,
    },
  });

  // Check if response is unauthorized
  if (response.status === 401 || response.status === 403) {
    localStorage.removeItem("access_token");
    window.location.href = "/login";
    return;
  }
```

```
  const data = await response.json();
  console.log(data);

  const responseYT = await fetch("/youtube/channels.json", {
    method: "GET",
  });
  const dataYT = await responseYT.json();
  console.log(dataYT);
  const myChannels = [];
  for (const channel of dataYT) {
    if (channel.user_id == data.id) {
      myChannels.push(channel);
    }
  }

  // Update counter
  const channelCount = myChannels.length;
  document.getElementById(
    "channels-count"
  ).innerHTML = `<i class="bi bi-collection-play me-2"></i>You added <b>${channelCount}</b>
YouTube channel${
    channelCount !== 1 ? "s" : ""
  }.`;

  // Show toggle button if there are channels
  if (channelCount > 0) {
    document.getElementById("toggle-channels-btn").style.display =
      "inline-block";
  }

  displayChannels(myChannels);

  // Update header
  document.getElementById("header-username").textContent =
    data.username || "Profile";
  document.getElementById("header-email").textContent = data.email || "";

  // Update profile fields
  document.getElementById("username").textContent = data.username || "-";
  document.getElementById("email").textContent = data.email || "-";
  document.getElementById("user-id").textContent = data.id || "-";
  document.getElementById("created-at").textContent = data.created_at
    ? new Date(data.created_at).toLocaleString()
    : "-";
  document.getElementById("about").textContent = data.about || "-";

  document.getElementById("apikey").value = data.apikey || "";
  document.getElementById("pins").textContent =
    JSON.stringify(data.pins) || "-";
  document.getElementById("oldpins").textContent =
    JSON.stringify(data.oldpins) || "-";
  document.getElementById("schedule").textContent =
    JSON.stringify(data.schedule) || "-";
  document.getElementById("lasttime").textContent = data.lasttime || "-";
  document.getElementById("lastpin").textContent = data.lastpin || "-";
  document.getElementById("battery").textContent = data.battery || "-";
  document.getElementById("extension-settings").textContent =
    data.extension_settings || "-";
}
getData();
```

5 Список каналов

Нажав на кнопку “Add new channel” можно добавить новый канал в список. На листенге 5 показано взаимодействие с API ютуба для добаления нового канала.

Листинг 5 – Взаимодействие с YouTube API

```
@router.post('/add')
async def channel_add(user: user_dependency, db: db_session, channel=Form(), lang=Form(),
tags=Form('')):
  if user is None:
    raise HTTPException(status_code=401)
  try:
    if '/channel/' in channel:
      ytid = channel[-24:]
                                                                          info          =
requests.get(f'https://youtube.googleapis.com/youtube/v3/channels?part=statistics,snippet&i
d={ytid}&key={YT_API_KEY}').json()
    else:
      return RedirectResponse('/youtube', status_code=302)
    stat = info['items'][0]['statistics']
    icon = info['items'][0]['snippet']['thumbnails']
    ytid = info['items'][0]['id']

    channel = models.Channel(
      id=ytid,
      url='https://www.youtube.com/channel/' + ytid,
      title=info['items'][0]['snippet']['title'],
      views=int(stat["viewCount"]),
      subs=int(stat["subscriberCount"]),
      video_count=int(stat["videoCount"]),
      lang=lang,
      icon_default=icon['default']['url'],
      icon_medium=icon['medium']['url'],
      icon_high=icon['high']['url'],
      tags=tags.split(',') if tags else [],
      user_id=user['id'],
    )
    db.add(channel)
    db.commit()
    return {'added_channel': info['items'][0]['snippet']['title']}
  except Exception as e:
    print(channel, e)
  return RedirectResponse('/youtube', status_code=302)
```

На листинге 6 показан JavaScript код общий как для списка каналов, так и для каналов в профиле. Для импорта которого используется следующий код.

```
<script src="{{ url_for('static', path='js/youtube.js') }}"></script>
```

Листинг 6 – Общий JavaScript код для списка каналов

```
const lang = {
  ru: "🇷🇺",
  en: "🇺🇸",
  es: "🇪🇸",
  de: "🇩🇪",
  fr: "🇫🇷",
  pt: "🇧🇷",
};
```

```
const colors = {
  "Google Docs": "#4285F4",
  Skillbox: "#6A4CFF",
  IT: "#3B82F6",
  Skyeng: "#00AEEF",
  Illustrator: "#FF9A00",
  Chemistry: "#1E9E63",
  Electronics: "#007ACC",
  "C++": "#00599C",
  ОГЭ: "#D32F2F",
  ESP8266: "#5C6BC0",
  Adobe: "#FF0000",
  "DaVinci Resolve": "#0DADEA",
  Dart: "#0175C2",
  Photoshop: "#31A8FF",
  Flutter: "#02569B",
  Java: "#ED8B00",
  PHP: "#777BB4",
  SQL: "#00618A",
  C: "#283593",
  Physics: "#5E35B1",
  Сотка: "#FFCC00",
  ЕГЭ: "#1565C0",
  "Popular Science": "#C62828",
  CSS: "#1572B6",
  Умскул: "#FF6F00",
  Economics: "#388E3C",
  FastAPI: "#009688",
  Web: "#2196F3",
  Biology: "#4CAF50",
  Перевод: "#455A64",
  Medicine: "#00ACC1",
  HTML: "#E44D26",
  "Русский язык": "#8D6E63",
  "After Effects": "#9393FF",
  Python: "#306998",
  JavaScript: "#F7DF1E",
  Figma: "#F24E1E",
  Linux: "#FCC624",
  Университет: "#3949AB",
  Geography: "#00897B",
  Blender: "#F5792A",
  Вебиум: "#4A47FF",
  "C#": "#68217A",
  Programming: "#1976D2",
  Education: "#3F51B5",
  "Cinema 4D": "#1A76D1",
  English: "#D32F2F",
  ESP32: "#424242",
  Arduino: "#00979D",
  Premiere: "#9933FF",
  Аудио: "#7E57C2",
  "Data Science": "#673AB7",
  History: "#8D6E63",
  Math: "#1A237E",
  Space: "#0D47A1",
  Social: "#1976D2",
};

function formatNumber(num) {
  if (num >= 1000000000) {
    return (num / 1000000000).toFixed(1) + "B";
  } else if (num >= 1000000) {
    return (num / 1000000).toFixed(1) + "M";
  } else if (num >= 1000) {
    return (num / 1000).toFixed(1) + "K";
  }
  return num.toString();
}

function displayChannels(channels) {
  if (channels.length === 0) {
    document.getElementById("channels-container").innerHTML =
      '<p class="text-center">No channels match your filters.</p>';
    return;
  }
```

```
  // Calculate tag frequency across all channels
  const tagFrequency = {};
  channels.forEach((channel) => {
    channel.tags.forEach((tag) => {
      tagFrequency[tag] = (tagFrequency[tag] || 0) + 1;
    });
  });

  let html = '<div class="row">';

  channels.forEach((channel) => {
    // Sort tags by popularity
    const sortedTags = [...channel.tags].sort(
      (a, b) => (tagFrequency[b] || 0) - (tagFrequency[a] || 0)
    );

    const tags = sortedTags
      .map(
        (tag) =>
          `<span class="badge me-1" style="background-color: ${
            colors[tag] || "grey"
          };">${tag}</span>`
      )
      .join("");

    html += `
        <div class="col-md-6 col-lg-4 mb-3">
          <a href="https://www.youtube.com/channel/${
            channel.id
          }" target="_blank" rel="noopener noreferrer" class="text-decoration-none">
              <div class="card h-100" style="transition: transform 0.2s, box-shadow 0.2s;"
onmouseover="this.style.transform='scale(1.025)';    this.style.boxShadow='0    4px    8px
rgba(0,0,0,0.2)';" onmouseout="this.style.transform='scale(1)'; this.style.boxShadow='';">
              <div class="card-body">
                <div class="d-flex align-items-center">
                  <img src="${
                    channel.icon_medium
                  }" width="60" height="60" style="border-radius: 50%;" class="me-3">
                  <div class="flex-grow-1">
                    <h5 class="card-title mb-1">
                      ${channel.title} ${lang[channel.lang] || channel.lang}
                    </h5>
                    <div>
                      <small class="text-muted">
                        <strong>${formatNumber(channel.subs)}</strong> subs •
                        <strong>${formatNumber(channel.views)}</strong> views •
                        <strong>${formatNumber(
                          channel.video_count
                        )}</strong> videos
                      </small>
                    </div>
                  </div>
                </div>
                <div class="mt-2">
                  ${tags}
                </div>
              </div>
            </div>
          </a>
        </div>
      `;
  });

  html += "</div>";
  document.getElementById("channels-container").innerHTML = html;
}
```

На листенге 7 показан JavaScript код для списка полезных YouTube каналов с возможностью фильтрации по языкам и тегам, а также с сортировкой. При наведении на карточку канала, она увеличивается.

Листинг 7 – JavaScript код для страницы со списком каналов

```
let allChannels = [];
let allTags = new Set();
let selectedTag = "";
let selectedAddTags = new Set();

function toggleAddChannelForm() {
  const card = document.getElementById("addChannelCard");
  const btn = document.getElementById("toggleAddForm");
  if (card.style.display === "none") {
    card.style.display = "block";
    btn.textContent = "❌ Cancel";
    btn.className = "btn btn-secondary";
  } else {
    card.style.display = "none";
    btn.textContent = "➕ Add New Channel";
    btn.className = "btn btn-success";
  }
}

async function submitChannel(event) {
  event.preventDefault();

  const formData = new FormData();
  formData.append("channel", document.getElementById("channel").value);
  formData.append("lang", document.getElementById("lang").value);
  formData.append("tags", document.getElementById("tags").value);

  const token = localStorage.getItem("access_token");

  try {
    const response = await fetch("/youtube/add", {
      method: "POST",
      headers: {
        Authorization: `Bearer ${token}`,
      },
      body: formData,
    });

    if (response.ok) {
      alert("Channel added successfully!");
      document.getElementById("addChannelForm").reset();
      selectedAddTags.clear();
      updateAddBadgeStyles();
      toggleAddChannelForm();
      loadChannels();
    } else if (response.status === 401) {
      alert("You must be logged in to add channels.");
    } else {
      alert("Error adding channel. Please try again.");
    }
  } catch (error) {
    console.error("Error:", error);
    alert("Error adding channel. Please try again.");
  }
}

function updateStats(channels) {
  const total = channels.length;
  const byLang = {};

  channels.forEach((ch) => {
    byLang[ch.lang] = (byLang[ch.lang] || 0) + 1;
  });

  const ruCount = byLang["ru"] || 0;
  const enCount = byLang["en"] || 0;
  const otherCount = total - ruCount - enCount;

  document.getElementById(
    "statsText"
  ).innerHTML = `<strong>${total}</strong> channels total.
    <strong>${ruCount}</strong> Russian,
    <strong>${enCount}</strong> English,
    <strong>${otherCount}</strong> other.`;
}

function filterAndSort() {
```

```
  const langFilter = document.getElementById("filterLang").value;
  const tagFilter = selectedTag;
  const sortBy = document.getElementById("sortBy").value;

  let filtered = allChannels.filter((channel) => {
    const langMatch = !langFilter || channel.lang === langFilter;
    const tagMatch = !tagFilter || channel.tags.includes(tagFilter);
    return langMatch && tagMatch;
  });

  // Sort
  filtered.sort((a, b) => {
    switch (sortBy) {
      case "subs":
        return b.subs - a.subs;
      case "subs_asc":
        return a.subs - b.subs;
      case "views":
        return b.views - a.views;
      case "views_asc":
        return a.views - b.views;
      case "videos":
        return b.video_count - a.video_count;
      case "videos_asc":
        return a.video_count - b.video_count;
      case "title":
        return a.title.localeCompare(b.title);
      default:
        return 0;
    }
  });

  updateStats(filtered);
  displayChannels(filtered);
}

function updateAddTags() {
  const tagsInput = document.getElementById("tags");
  const tagsArray = Array.from(selectedAddTags);
  tagsInput.value = tagsArray.join(",");
}

function updateBadgeStyles() {
  document.querySelectorAll("#tagBadges .badge").forEach((badge) => {
    if (badge.dataset.tag === selectedTag) {
      badge.style.backgroundColor = badge.dataset.color;
      badge.style.border = `1px solid ${badge.dataset.color}`;
      badge.style.color = "#ffffff";
      badge.style.fontWeight = "600";
      badge.style.transform = "scale(1.05)";
    } else {
      badge.style.backgroundColor = "transparent";
      badge.style.border = `1px solid ${badge.dataset.color}`;
      badge.style.color = badge.dataset.color;
      badge.style.fontWeight = "500";
      badge.style.transform = "scale(1)";
    }
  });
}

function updateAddBadgeStyles() {
  document.querySelectorAll("#addTagBadges .badge").forEach((badge) => {
    const tag = badge.dataset.tag;
    if (selectedAddTags.has(tag)) {
      badge.style.backgroundColor = badge.dataset.color;
      badge.style.border = `1px solid ${badge.dataset.color}`;
      badge.style.color = "#ffffff";
      badge.style.fontWeight = "600";
    } else {
      badge.style.backgroundColor = "transparent";
      badge.style.border = `1px solid ${badge.dataset.color}`;
      badge.style.color = badge.dataset.color;
      badge.style.fontWeight = "500";
    }
  });
}
```

```
async function loadChannels() {
  try {
    const response = await fetch("/youtube/channels.json");
    allChannels = await response.json();

    // Extract all unique tags with frequency count
    const tagFrequency = {};
    allChannels.forEach((channel) => {
      channel.tags.forEach((tag) => {
        allTags.add(tag);
        tagFrequency[tag] = (tagFrequency[tag] || 0) + 1;
      });
    });

    // Get colors from youtube.js
    const tagColors = typeof colors !== "undefined" ? colors : {};

    // Populate tag badges for adding new channel
    const addTagBadgesContainer = document.getElementById("addTagBadges");
    Array.from(allTags)
      .sort((a, b) => tagFrequency[b] - tagFrequency[a])
      .forEach((tag) => {
        const badge = document.createElement("span");
        badge.className = "badge";
        badge.style.cursor = "pointer";
        badge.style.backgroundColor = "transparent";
        badge.style.border = `1px solid ${tagColors[tag] || "#6c757d"}`;
        badge.style.color = tagColors[tag] || "#6c757d";
        badge.style.transition = "all 0.2s ease";
        badge.style.fontWeight = "500";
        badge.textContent = tag;
        badge.onclick = () => {
          if (selectedAddTags.has(tag)) {
            selectedAddTags.delete(tag);
          } else {
            selectedAddTags.add(tag);
          }
          updateAddBadgeStyles();
          updateAddTags();
        };
        badge.dataset.tag = tag;
        badge.dataset.color = tagColors[tag] || "#6c757d";
        addTagBadgesContainer.appendChild(badge);
      });

    // Populate tag badges, sorted by popularity (frequency)
    const tagBadgesContainer = document.getElementById("tagBadges");
    Array.from(allTags)
      .sort((a, b) => tagFrequency[b] - tagFrequency[a])
      .forEach((tag) => {
        const badge = document.createElement("span");
        badge.className = "badge";
        badge.style.cursor = "pointer";
        badge.style.backgroundColor = "transparent";
        badge.style.border = `1px solid ${tagColors[tag] || "#6c757d"}`;
        badge.style.color = tagColors[tag] || "#6c757d";
        badge.style.transition = "all 0.2s ease";
        badge.style.fontWeight = "500";
        badge.textContent = `${tag} (${tagFrequency[tag]})`;
        badge.onclick = () => {
          selectedTag = selectedTag === tag ? "" : tag;
          updateBadgeStyles();
          filterAndSort();
        };
        badge.dataset.tag = tag;
        badge.dataset.color = tagColors[tag] || "#6c757d";
        tagBadgesContainer.appendChild(badge);
      });

    // Allow manual tag input to update badge selection
    const tagsInput = document.getElementById("tags");
    tagsInput.addEventListener("input", () => {
      const tags = tagsInput.value
        .split(",")
        .map((t) => t.trim())
        .filter((t) => t);
      selectedAddTags = new Set(tags);
```

```
      updateAddBadgeStyles();
    });

    filterAndSort();
  } catch (error) {
    console.error("Error loading channels:", error);
    document.getElementById("channels-container").innerHTML =
      '<p class="text-danger">Error loading channels.</p>';
  }
}


loadChannels();
```

Вывод

В результате работы реализовано взаимодействие с API для лендинга, входа, регистрации, профиля и списка каналов с использованием JavaScript и FastAPI.