

# Мікросервісний підхід до розробки клієнтської частини веб-застосунків

Студент:

ДА-72мп

Петенок Олена

Науковий керівник:

доцент, к.т.н,

Булах Богдан Вікторович

- **Мета роботи:**  
дослідити застосування мікросервісного підходу до розробки клієнтської частини веб-застосунків.
- **Об'єкт дослідження:**  
розробка веб-застосунків.
- **Предмет дослідження:**  
застосування та аналіз мікросервісного підходу до розробки клієнтської частини веб-застосунків.

# Актуальність роботи

- Сучасні веб-застосування часто більше спираються на клієнтську частину, ніж на серверну.
- Вимоги до функціоналу веб-застосовувань постійно збільшуються.
- Існуючі функціонал та контент потребують підтримки та оновлення.

# Підходи до розробки клієнтської частини веб-застосунків

# Монолітний підхід

- Застосування будується як єдине ціле.
- Для внесення зміни в систему доводиться збудувати та розгорнути оновлену версію.

# Переваги монолітного підходу

- Легше мати великий і послідовний UX.
- Єдині кодова база, база даних та стек розробки передбачають швидкий обмін знаннями в команді.
- Простіше тестування (в порівнянні з іншими підходами).
- Виправданий за невеликих застосувань.

# Недоліки монолітного підходу

- Монолітні застосування можуть перетворитися на "велику кулю бруду" (коли розробники не розуміють всієї програми).
- Масштабування може бути складним.
- Єдиний стек розробки може обмежити доступність «правильного» інструментарію для роботи.

# Компонентний підхід

- Заснований на повторному використанні.
- Акцентує увагу на поділі функціональних можливостей.



# Переваги компонентного підходу

- Компоненти є модульними, згуртованими, замінюваними, повторно використовуваними, слабо пов'язаними.
- Це дозволяє легко вносити зміни та добудовувати новий функціонал.

# Недоліки компонентного підходу

- Схильність до надмірної інженерії, проектування кожного аспекту інтерфейсу користувача як компоненту може бути надлишковим.

# Мікросервісний підхід

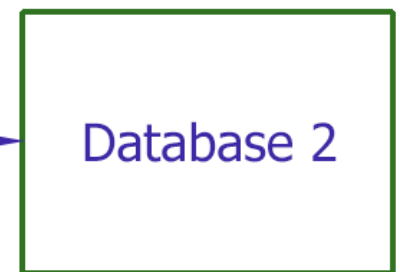
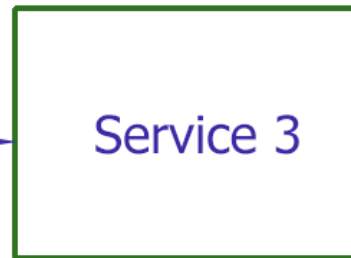
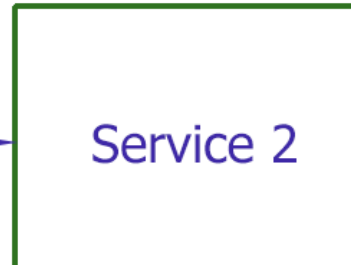
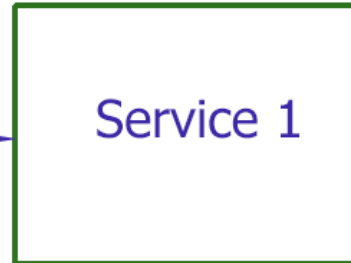
- Суть в тому, щоб думати про веб-застосування як про складову слабо пов'язаних або незалежних функцій, що належать незалежним групам розробників.
- Команди мають визначену сферу розробки чи завдання.
- Команди незалежно розвивають функціонал повною мірою: від бази даних до інтерфейсу користувача.

# Мікросервісний підхід на клієнті

User Interface



Server



# Основні особливості та вимоги мікросервісного підходу

- Незалежність від стеку розробки.
- Команди обирають та оновлюють стек розробки незалежно.
- Ізолювання коду команд.
- Можливість використання різних БД незалежними сервісами.
- Встановлення незмінної конвенції щодо найменувань, де ізоляція неможлива: простір імен CSS, події, локальні сховища, файли cookie.

# Переваги мікросервісного підходу

- Покращує модульність, полегшує розуміння, розробку та тестування, підвищує стійкість до ерозії архітектури.
- Паралелізує розробку, дозволяючи автономним командам самостійно розробляти, розгортати та масштабувати сервіси.
- Незалежність від стеку розробки дозволяє добудовувати новий функціонал у старі проекти без рефакторингу старого функціоналу.

# Комерційні переваги мікросервісного підходу

- Можливість розширення штату компанії без прив'язки до стеку розробки.
- Можливість аутсорсингу без ризиків внесення змін в існуючу версію застосування.
- Відсутня необхідність “міграції” з бібліотеки на бібліотеку.

# Недоліки мікросервісного підходу

- Необхідність чіткого поділу завдань між командами на початкових стадіях розробки.
- Необхідність встановлення незмінної конвенції щодо найменувань, де ізоляція неможлива.
- Схильність до надмірної інженерії: проектування кожного аспекту інтерфейсу користувача як сервісу може бути надлишковим.



# Способи реалізації клієнтської частини веб- застосунків за мікросервісного підходу

# Використання компонентів як інтеграційного шару

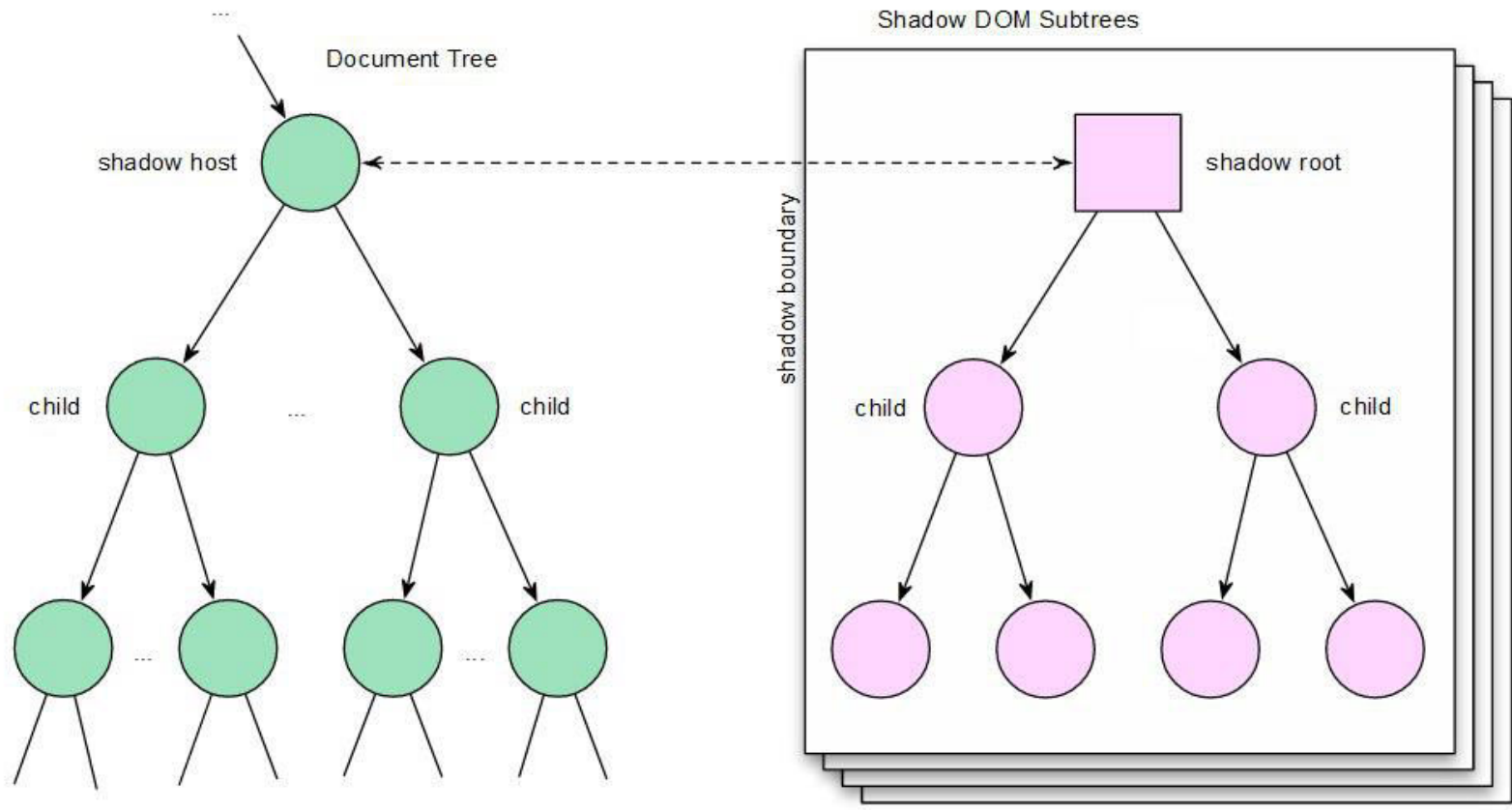
- **Custom Elements**

наприклад:

```
<books-list></books-list>
```

# Використання компонентів як інтеграційного шару

- **Shadow DOM** (приєднання прихованого дерева DOM до реального)



# Використання компонентів як інтеграційного шару

- **Virtual DOM**

(віртуальне дерево DOM зберігається у пам'яті та синхронізується з реальним деревом DOM бібліотекою)

```
<header class="header-background" id="header"></header>
<section id="header-padding"></section>
<section class="main-background gallery" id="gallery-select"></section>

<section>
  <div class="main-background gallery">
    <div class="uk-container">
      <div uk-grid>
        <div ng-class="width" ng-repeat="lists in gallery">
          <ul ng-repeat="item in lists.elements">
            <li class="image-block"><img src='{{item.src}}' alt='{{item.alt}}'></li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</section>

<footer class="footer-background" id="footer"></footer>
```

# Використання спільних подій як інтеграційного шару

- Сервіси обробляють спільні вхідні та вихідні події (спільна шина подій або імплементація шаблону проектування Publish-Subscribe).

# Використання сторонніх бібліотек як інтеграційного шару

- Мета-бібліотека Single-SPA дає можливість:
  - використання кількох бібліотек на сторінці;
  - використання Lazy load code для покращення первинного часу завантаження.
- Project Mosaic:
  - набір сервісів та бібліотек із специфікацією, що визначають, як сервіси взаємодіють між собою.
- **Великі компанії обережно ставляться до використання стороннього коду через питання безпеки.**

# Приклади реалізації клієнтської частини веб- застосунків за мікросервісного підходу

# <https://micro-frontends.org/>

## The Model Store

basket: 0 item(s)

### Related Products



Tractor Porsche-Diesel Master  
419



buy for 66,00 €





# https://allegro.pl


Strefa MarekInspiracjymoda.allegrosklep Allegro

wystaw przedmiotmoje allegrowyloguj

allegro

czego szukasz?

wszystkie działy

koszyk jest pusty

ESI

Elektronika

Moda i uroda

Dom i zdrowie

Dziecko

Kultura i rozrywka

Sport i wypoczynek

Motoryzacja

Kolekcje i sztuka

Firma


Strefa okazji

REKLAMA

# #TheNextGalaxy

Premiera już dziś o godzinie 19:00

ZOBACZ



SAMSUNG

Nowy Galaxy

adidas Originals - świetna cena!


Laptop HP 250

Wyprzedaż przed sezonem

Promocja Tchibo Caffissimo

ESI


wyjątkowe okazje dla Ciebie (zobacz więcej)



-30%

Spawarka inwertorowa mma 250a maska spawalnicza


670,07 zł 475,98 zł



-21%

Zestaw 5x Semilac Lampa Led Diamond


380,00 zł 299,00 zł



-31%

Pufa SAKO XXXL + pufa pod nogi różne kolory


169,00 zł 109,00 zł



-6%

Piekarnik z mikrofalą Samsung NQ50H5537KB


1 999,00 zł 1 879,00 zł 3%



-24%

Gra planszowa Talisman Magia i Miecz

249,00 zł 166,90 zł 3%



Łóżko prakt

999,00 zł

ESI

Клієнтська частина  
власного веб-  
застосування,  
розробленого за  
мікросервісного  
підходу

# Архітектура власного веб-застосування

Складається з сервісів:

- header
- footer
- фільтр (ширини відображення списків зображень в галереї)
- галерея

# Архітектура власного веб-застосування

- Сервіси header, footer та фільтр повністю незалежні.
- Сервіс галерея залежний від сервісу фільтр за результуючим деревом DOM, але незалежний від реалізації фільтру.
- У елементів `<li>` сервісу фільтр існує по два обробники подій натискання на елемент (один обробляється у фільтрі та відповідає за відображення елемента `<ul>`, а інший обробляється у галереї та відповідає за відображення ширини зображень).

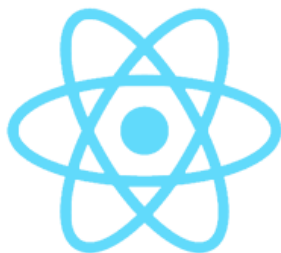
# Обрані засоби реалізації

UIKit



3.0.0

React



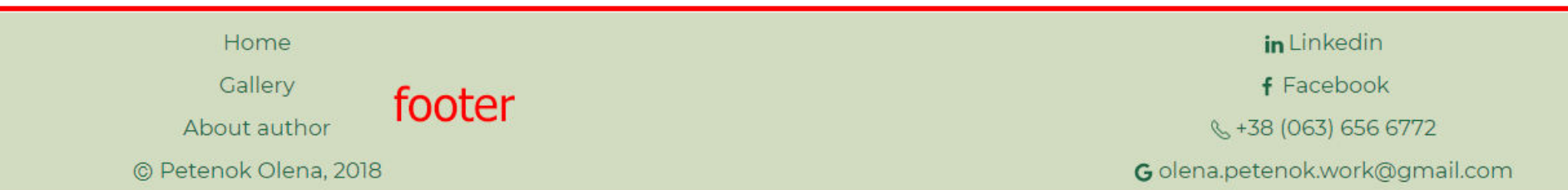
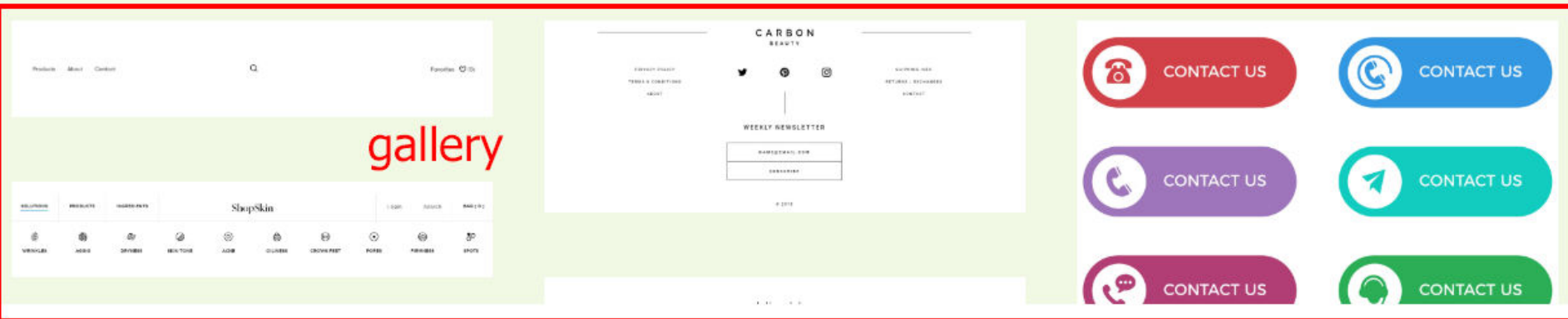
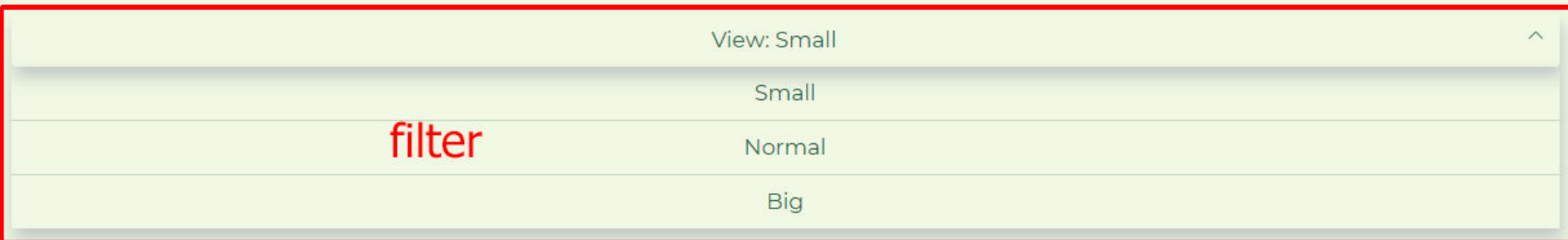
16.5

Angular



1.6.4

# Інтерфейс користувача веб-застосування



# Засоби реалізації різних версій веб-застосування

№	Засіб реалізації сервісу Header	Засіб реалізації сервісу Footer	Засіб реалізації сервісу Фільтр	Засіб реалізації сервісу Галерея
1	HTML	HTML	React	React
2	React	React	React	React
3	HTML	HTML	React	Angular
4	React	React	React	Angular

# Аналіз швидкості завантаження реалізованих версій

All data is presented in seconds (s)	Page Speed Insights:		Pingdom Website Speed Test, desktop			Average
	Mobile	Desktop	Germany, Frankfurt	UK, London	USA, Washington	
HTML, React	3.80	0.80	1.33	1.65	4.85	2,486
React	4.00	0.90	1.25	1.57	4.67	2,478
HTML, React, Angular	4.30	0.90	1.25	1.81	5.20	2,692
React, Angular	4.10	0.90	1.17	2.29	5.12	2,71



# Рекомендації з вибору підходу до розробки за результатами аналізу швидкості:

- використання статичного HTML по-різному впливає на швидкість, його використання опціональне;
- використання однакової бібліотеки для різних сервісів щоразу давало невеликий виграш у швидкості (тобто компонентні застосування працюють швидше за мікросервісні, але переваги швидкодії можуть нівелюватися недоліком меншої «свободи дій» компонентного підходу і для великих проектів використання мікросервісного підходу доцільніше).

# Практична цінність результатів дослідження:

- Проведено аналіз підходів до розробки клієнтської частини веб-застосунків.
- Визначені переваги та недоліки цих підходів до розробки.
- Проведено аналіз швидкості завантаження клієнтської частини веб-застосунків за різної реалізації.
- Означено рекомендації з вибору підходу до розробки в залежності від пріоритетів компанії та величини веб-застосування.

# Висновки

- Основними перевагами мікросервісного підходу є незалежність сервісів від стеку розробки, полегшення внесення змін в існуючі проекти та можливість розробляти нові сервіси з використанням сучасних інструментів без рефакторингу старих сервісів.

Дякую за увагу!