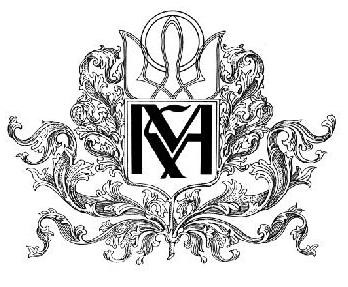
Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики



**Аналіз сучасних підходів до проектування архітектури веб застосунків на прикладі сервісу опитування**

**Текстова частина до курсової роботи**

**за спеціальністю „Комп’ютерні науки” 6.050103**

Керівник курсової роботи

ст.викладач Вовк Н.Є

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(підпис)*

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2020 р.

Виконав студент

Федюченко М.І

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2020 р.

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультімедійних систем,

Доцент., к. ф.-м. н. О.П.Жижерун

(підпис)

„\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

на курсову роботу

студенту Федюченку Михайлу Ігоровичу факультету інформатики 3-го курсу

ТЕМА Аналіз сучасних підходів до проектування архітектури веб застосунків на прикладі компоненту користувача для сервісу опитувань

Зміст ТЧ до курсової роботи:

1. Індивідуальне завдання
2. Календарний план
3. Анотація
4. Вступ
5. Основні типи архітектур сучасних веб додатків
6. SPA архітектура та найкращі практики
7. Огляд створеного додатку та використаних практик
8. Висновки
9. Список використаної джерел

Дата видачі „\_\_\_” \_\_\_\_\_\_\_\_\_ 2017 р. Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Завдання отримав \_\_\_\_\_\_\_\_\_\_

(підпис)

**Календарний план виконання роботи:**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Назва етапу курсової роботи | Термін виконання етапу | Примітка |
| 1. | Отримання теми курсової роботи. | 15.10.2019 |  |
| 2. | Огляд технічної літератури за темою роботи | 30.10.2019 |  |
| 3. | Виконання аналізу архітектурних підходів побудови веб-додатків | 15.02.2020 |  |
| 4. | Написання першого розділу | 20.02.2020 |  |
| 5. | Аналіз найкращих практик при побудові односторінкового додатку | 20.03.2020 |  |
| 6. | Написання другого розділу | 30.03.2020 |  |
| 7. | Створення веб-застосунку для опитування студентів на основі SPA архітектури. | 01.04.2020 |  |
| 8. | Написання останнього розділу | 09.04.2020 |  |
| 9. | Корегування роботи згідно із зауваженнями керівника | 10.05.2020 |  |

СтудентФедюченко М.І.

Керівник Вовк Н.Є.

**“\_\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Зміст

*Анотація*

У роботі розглянуто сучасні підходи до побудови веб-застосунків, детально розглянуто SPA-підхід та його переваги. Розроблено веб застосунок на основі SPA-підходу та з використанням досліджених практик, що забезпечують гнучкість застосунку, масштабованість, ефективність та багаторазовість використання.

## ВСТУП

З розвитком людства , об’єми створеної людиною інформації збільшуються і для отримання її виникає потреба створення нових джерел інформації, чи покращення-модифікації вже існуючих. Веб-браузер – це найпоширеніше у світі програма, потрібність котрої є беззаперечною. Певне кожен користувався веб-ресурсами, щоб знайти потрібний матеріал, чи переглядав відео, фотографії та іншу інформацію представлену в мережі інтернет. У цьому нам допомагають веб застосунки, котрі не потребують встановлення, є доступними для будь-якої платформи через більшість браузерів. А також можливість обробки великих об’ємів інформації, що здійснюється за лічені секунди у хмарних сховищах з відображенням користувачу без використання потужностей використовуваного девайсу. Це викликає попит на створення ефективних веб-застосунків, з можливістю їх багаторазового використання, масштабування при виникненні нових потреб та гнучкості використання. Усе це залежить від архітектурної складової та принципів їх побудови.

Актуальність правильної архітектури тільки зростає з бажанням людини ще більше автоматизувати процеси власної діяльності, тому зараз складається тенденція на використання практик, що призводять до загальної ефективності в розробці додатків, технічному обслуговуванні коду та часу розробки.

Метою курсової роботи є визначення основних сучасних підходів до побудови архітектури веб-додатків та з’ясування найкращих практик при створенні односторінкового застосунку.

Текстова частина курсової роботи складається з трьох розділів.

У першому розділі розглядаються основні сучасні архітектурні підходи при побудові веб застосунків, а також їх порівняння. Наведено огляд архітектурних моделей та технологій для їх створення.

У другому розділі окреслюються шляхи підвищення ефективності при побудові SPA веб-застосунку, обґрунтовуються найкращі з розглянутих практик.

Третій розділ присвячено опису архітектурної побудови SPA веб-додатку на основі сервісу опитування.

Постановка задачі.

1. Виконати аналіз архітектурних підходів при побудові веб-додатків, зокрема

* На MPA архітектурі
* На SPA архітектурі
* На Serverless архітектурі

1. Виконати порівняльний аналіз архітектурних підходів.
2. Виконати детальне дослідження практик та архітектурних моделей при побудові клієнтського компоненту веб-додатку на основі SPA-архітектури.
3. Розробити аргументовану структуру клієнтського компоненту з урахуванням досліджених практик та моделей.
4. Зробити висновки щодо актуальності та доречності використаних архітектурних практик.
5. Основні архітектурні підходи при побудові веб-застосунків

Перш за все важливо зрозуміти, що таке веб-додаток і в чому його особливість. Веб-додатком називають тип програми побудованої на основі клієнт-серверної архітектури і його особливість в тому, що сам веб-додаток розташований і виконується на віддаленому сервері, тоді як клієнт отримує тільки результати власної роботи. Робота такої програми заснована на опрацюванні запитів користувача та надсилання результатів клієнту через мережу інтернет, тому і не залежить від операційної системи певного користувача. Зазвичай за відображення результатів запитів та отримання вхідних даних для подальшого відправлення на сервер відповідає браузер (Google Chrome, Opera, Mazila, тощо).

Практично будь-який веб-додаток складається з двох основних компонентів

* Клієнтська компонента – код, що знаходиться в браузері та реагує на певний ввід користувача, є представленням функціональності веб- застосунку ,користувач має повний доступ до цієї компоненти.
* Серверна компонента – код , що знаходиться на сервері та відповідає на HTTP запити, відповідальний за збереження інформації, складається щонайменше з бази даних та логіки програми, дана компонента недоступна для користувача.

Архітектура веб-додатків описує взаємодію між компонентами що працюють одночасно, базами даних та системами проміжного програмного забезпечення в мережі Інтернеті. Тобто від того як розподілена логіка програми між серверною та клієнтською компонентами залежить тип архітектури веб-додатку.

* 1. Багато сторінковий тип архітектури (MPA)

Багатосторінковий тип архітектури – це традиційний тип архітектури для веб застосунків, при якому виконується завантаження усієї сторінки, та перезавантаження на абсолютно нову, якщо користувач взаємодіє із застосунком.

Багатосторінкові програми - це традиційні веб-програми, які завантажують всю сторінку та відображають нову, коли користувач взаємодіє з веб-додатком.

Кожен раз, коли дані обмінюються вперед та назад, від сервера запитується нова сторінка для відображення у веб-браузері. Цей процес потребує часу для створення сторінок на сервері, надсилання їх клієнтові та відображення у браузері, що може вплинути на досвід роботи користувача.

Однак AJAX дозволила зробити лише певний компонент програми, але це робить процес розробки ще більш складним і складним. Багатосторінкові програми мають більш класичну архітектуру. Кожна сторінка надсилає запит на сервер і повністю оновлює всі дані. Навіть якщо ці дані невеликі. Таким чином, продуктивність витрачається на показ одних і тих же елементів.

* 1. Мікро сервісний тип архітектури

Мікро сервісна архітектура – це форма сервісно-орієнтованої архітектури технологія розробки програмного забезпечення, яка структурує додаток як колекцію незалежних мікро-сервісів, що комунікують між собою використовуючи певні механізми, зазвичай HTTP/REST. Кожен мікро-сервіс є самодостатнім і повинен реалізувати конкретну бізнес-потребу, тому його можливо створити й запустити незалежно від інших , навіть на зовсім іншій мові програмування.

Усі більше постачальників послуг (Twitter, eBay, Netflix та Amazon) надають перевагу мікро сервісній архітектурі. Наприклад до 2012 року Walmart Canada використовували монолітний тип архітектури, що в піках активного використання сервісу призводило до обробки шести мільйонів сторінок на хвилину, що займало багато часу, тому й зменшувало кількість купівель через сервіс. Після оновлення архітектурного підходу на мікро сервіси їм вдалося збільшити дохід на двадцять відсотків за рахунок того, що час простою було скорочено і компанія отримала змогу використовувати дешевші сервери, збільшити кількість покупок через мобільні пристрої на дев’яносто вісім відсотків.

* + 1. Філософія мікро сервісної архітектури

Філософія мікро сервісів дуже близька до філософії сімейства відомих операційних систем Unix : (“Зробіть одну річ і зробіть її добре”  by [Peter (Raymond) H. Salus](https://en.wikipedia.org/wiki/Peter_H._Salus" \o "Peter H. Salus) in A Quarter-Century of Unix (1994):[[1]](https://en.wikipedia.org/wiki/Unix_philosophy#cite_note-taoup-ch1s6-1))

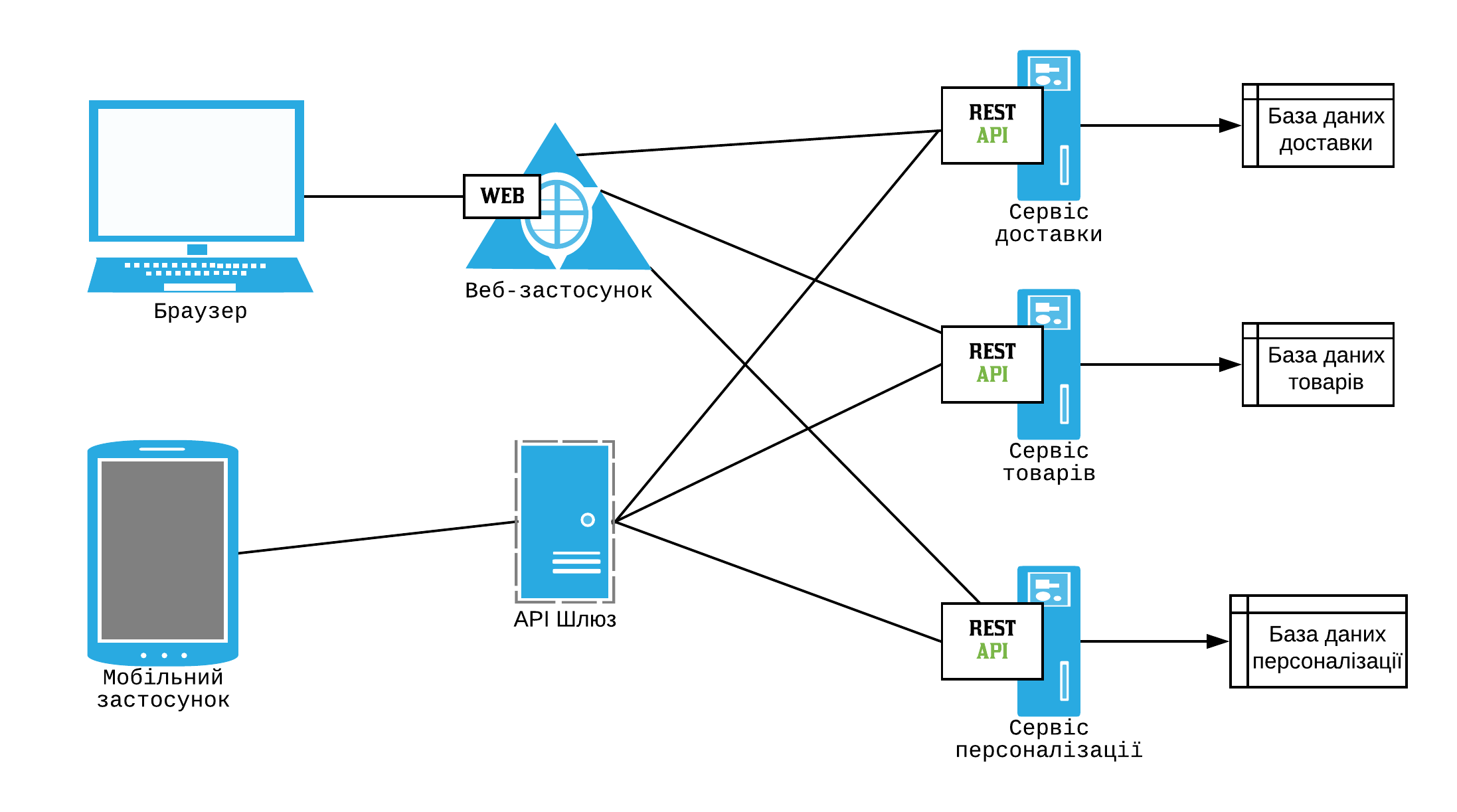
Архітектура мікро сервісів використовує бібліотеки, однак її головний спосіб розбиття застосунку шляхом поділу його на сервіси. Перевага незалежності розвертання кожного сервісу зумовлена тим, що при зміні логіки одного з них не потрібно буде виконувати повторне розвертання усіх сервісів програми, а тільки того, що зазнав змін. Даний архітектурний підхід допомагає не порушувати інкапсуляцію компонентів за рахунок використання віддалених викликів сервісами програми.

Архітектурний підхід починається з різних типів клієнтів, що можуть бути будь-якими пристроями, котрі намагаються виконувати певні можливості керування, наприклад пошук, налаштування тощо.

Оскільки клієнти не звертаються до сервісів напряму, шлюз API є вхідною точкою, що виконує переадресацію запитів клієнтів у відповідні мікро сервіси. Першою перевагою використання API шлюзу є те, що ми можемо виконувати зміни над сервісом без попередження клієнта, а також що сервіси можуть використовувати протоколи спілкування не пристосовані для вебу.

Існують два основні типи повідомлень для комунікації сервісів між собою:

* Синхронні повідомлення – у випадку, якщо клієнт очікує відповіді від сервісу, прийнято використовувати REST, адже це розподілене середовище у котрому кожна функціональність представлена відповідним ресурсом для проведення операцій
* Асинхронні повідомлення – у випадку, коли клієнт не очікує відповіді. Зазвичай використовуються такі протоколи, як AMQP, STOMP, MQTT, чия основа повідомлень є чітко визначеною і ці повідомлення мають бути сумісними між реалізаціями.



Особливість котру ми спостерігаємо на рисунку полягає в тому, що кожній бізнес потребі було виділено окремий мікро сервіс, для обробки власних даних та виконання різних функціональних можливостей. Ці мікро сервіси мають власне середовище розгортання для виконання своїх функціональних потреб та власні бази даних для фіксації даних. Усі мікро сервіси спілкуються між собою через сервер, що не зберігаю стан, який є або шиною повідомлень, або REST клієнтом. Усі внутрішні точки з'єднані з шлюзом API. Отже, кожен, хто підключається до шлюзу API, автоматично підключається до повної системи.

* + 1. Переваги та недоліки мікро сервісної архітектури

Тенденція на мікро сервісну архітектуру набуває популярності, адже в міру збільшення складності програмного забезпечення, можливість розбиття функціональних областей застосунку на колекції незалежних сервісів може принести багато переваг розробникам програмного забезпечення, що передбачають:

1. Масштабованість – мікро сервіси, що складаються з компонентів легко можуть бути інтегровані з іншими програмами чи сервісами через інтерфейси, наприклад REST.
2. Прискорена розробка програмного забезпечення – через можливе використання різних мов програмування стає ширшим вибір кола розробників. А також різні команди можуть працювати паралельно над окремими компонентами, без очікування результатів один-одного. Інші ж зацікавлені сторони можуть в цей час вдосконалювати вже реалізовані компоненти.
3. Можливість повторного використання – оскільки мікро сервіси не є одним проєктом і розташовані навколо певних бізнес потреб, то легко можуть бути повторно використаними в інших проєктах, що також і зменшує час розробки.
4. Краще розуміння кодової бази – оскільки кожен сервіс являє собою єдину функціональну область або бізнес потребу, то це покращує подальшу підтримку та налагодження.
5. Ефективна ізоляцій можливих проблем – немає потреби у постійному переході між структурними шарами застосунку, розробники знають, де саме потрібно шукати проблеми, котра потребує вирішення. Якщо зачіпається один сервіс, його можна легко видалити або вирішити без пошкодження інших частин програм

Мікро сервісний тип архітектури, як і будь який інший тип не є ідеальним, виявлено і певні недоліки:

1. Складність інтеграційного тестування – зумовлене можливістю існування компонентів що належать іншим системам, чи середовищам.

Адже перед початком тестування потрібне підтвердження кожного залежного сервісу.

1. Комунікація – оскільки канали зв’язку мають бути визначені у певних інтерфейсах, можливе виникнення помилок між мікро сервісами при недотриманню визначеного інтерфейсу.
2. Потреба у належному моніторингу – кожний сервіс залежить від власного API та платформи розгортання й для належного моніторингу потрібно слідкувати за багатьма сутностями та мати контроль над усією інфраструктурою.

# Список литературы

Raymond, E. S. (б.д.). *The Art of Unix Programming.* Addison-Wesley Professional. Получено 7 5 2020 г., из http://www.catb.org/~esr/writings/taoup/html/