### **Міністерство освіти і науки України**

### **Національний технічний університет України «Київський**

### **політехнічний інститут імені Ігоря Сікорського"**

### **Факультет інформатики та обчислювальної техніки**

### **Кафедра інформатики та програмної інженерії**

### **Звіт**

### з лабораторної роботи № 3

### з дисципліни

### «Компоненти програмної інженерії частина 3. Архітектура програмного

### забезпечення»

### **«Мікросервісна архітектура»**

### **Виконав** ІП-21 Скрипець О., Мунтяну А.

### **Перевірив** Зубик Л. В.

### **Київ 2024**

**Лабораторна робота №3**

**Тема:** Застосувати мікросервісну архітектуру для проектування складних систем.

**Мета:** Навчитись формувати та документувати базові технічні рішення з архітектури додатку, зокрема, основні компоненти (сервіси), зв’язки між компонентами, дизайн *API*.

**Завдання:**

1. Між-сервісна взаємодія:
   1. Визначити які сервіси між собою спілкуються синхронно.
   2. Визначити які сервіси між собою спілкуються асинхронно, за допомогою яких систем (kafka/rabbitmq/…)
2. Робота з даними.
   1. Визначити, який патерн використовується для роботи з базою даних (db per service or shared db)
   2. Визначити, як накачуються міграційні скрипти для бази даних.
   3. Визначити, як саме відбувається синхронізація даних між сервісами.
3. Оновити діаграми, зобразити всі зв’язки між сервісами.
4. Вказати, які саме патерни для мікросервісної архітектури використовуються.

**1. Між-сервісна взаємодія.**

Наша веб-платформа для менторства буде розділена на декілька мікросервісів, кожен з яких відповідає за певну функціональність:

* **Сервіс аутентифікації та авторизації** - відповідає за перевірку ідентичності користувачів та надання їм відповідних прав доступу.
* **Сервіс управління користувачами** - забезпечує управління профілями користувачів, включаючи реєстрацію, оновлення та видалення профілів.
* **Сервіс кабінету ментора** - дозволяє менторам керувати своїми профілями, включаючи налаштування графіку, спеціалізації та ін.
* **Сервіс пошуку ментора** - допомагає користувачам знайти відповідного ментора на основі різних критеріїв.
* **Сервіс бронювань** - дозволяє користувачам бронювати час у менторів та відслідковувати свої бронювання.
* **Сервіс консультацій** - забезпечує платформу для проведення онлайн-консультацій між менторами та користувачами.
* **Сервіс оцінювання та відгуків** - дозволяє користувачам залишати відгуки та оцінки менторів, що сприяє прозорості та надійності платформи.

Ці мікросервіси взаємодіють між собою, щоб забезпечити цілісну та ефективну роботу платформи. Кожен сервіс може розвиватися незалежно, що сприяє швидкому внесенню змін та адаптації до вимог користувачів.

Додатково до внутрішніх мікросервісів, веб-платформа для менторства також інтегрується з наступними зовнішніми системами:

* **Зовнішня система оплати** - ця система відповідає за обробку всіх транзакцій оплати на платформі, включаючи оплату за послуги менторства. Це забезпечує безпечність фінансових операцій та зручність для користувачів.
* **Зовнішня система email-сповіщень** - ця система відповідає за відправку автоматичних email-сповіщень користувачам. Це включає сповіщення про успішне проведення оплати, інформація про створене бронювання, нагадування про майбутні консультації тощо.

Ці зовнішні системи дозволяють платформі фокусуватися на своїй основній функціональності - менторстві, в той час як спеціалізовані задачі, такі як оплата та сповіщення, обробляються надійними зовнішніми службами.

#### **а. Синхронна взаємодія:**

Наша веб-платформа для менторства використовує синхронну взаємодію між деякими мікросервісами для забезпечення плавної роботи системи. Ось декілька прикладів:

* **Сервіс аутентифікації та авторизації** та **Сервіс управління користувачами**: Ці два сервіси повинні діяти синхронно при реєстрації нових користувачів та оновленні їхніх профілів. Наприклад, під час реєстрації нового користувача сервіс аутентифікації може створити новий акаунт у сервісі управління користувачами та надати йому доступ до системи.
* **Сервіс бронювань** та **Сервіс кабінету ментора**: Коли користувач бронює час у ментора, ці два сервіси взаємодіють синхронно, щоб оновити графік ментора та записати деталі бронювання.
* **Сервіс оцінювання та відгуків** та **Сервіс управління користувачами**: Після завершення консультації, користувач може залишити відгук та оцінку ментору. Ці два сервіси взаємодіють синхронно, щоб оновити профіль ментора з новим відгуком та оцінкою.
* **Сервіс управління користувачами** та **Сервіс бронювань**: Коли користувач робить бронювання, Сервіс бронювань може потребувати інформацію про користувача з Сервісу управління користувачами для перевірки його статусу або інших деталей. Ця взаємодія відбувається синхронно через REST API.

Ці приклади демонструють, як синхронна взаємодія між мікросервісами може забезпечити ефективну роботу вашої платформи для менторства. Кожен мікросервіс виконує свою специфічну роль, але разом вони працюють як єдина система.

**b. Асинхронна взаємодія:**

Наша веб-платформа для менторства використовуватиме асинхронну взаємодію між деякими мікросервісами для забезпечення високої продуктивності та масштабованості. Ось декілька прикладів:

* **Сервіс бронювань** та **Сервіс консультацій**: Після створення бронювання, Сервіс бронювань може відправити асинхронне повідомлення до Сервісу консультацій через Apache Kafka, щоб підготувати необхідні ресурси для майбутньої консультації.
* **Сервіс оцінювання та відгуків** та **Сервіс кабінету ментора**: Після отримання нового відгуку, Сервіс оцінювання та відгуків може відправити асинхронне повідомлення до Сервісу кабінету ментора через Apache Kafka, щоб оновити рейтинг ментора.
* **Сервіс бронювань** та **Зовнішня система оплати**: Після створення бронювання, Сервіс бронювань може відправити асинхронне повідомлення до Зовнішньої системи оплати через Apache Kafka, щоб ініціювати процес оплати.
* **Сервіс консультацій** та **Сервіс оцінювання та відгуків**: Після завершення консультації, Сервіс консультацій може відправити асинхронне повідомлення до Сервісу оцінювання та відгуків через Apache Kafka, щоб запитати відгук від користувача.
* **Сервіс управління користувачами** та **Зовнішня система email-сповіщень**: Після реєстрації нового користувача, Сервіс управління користувачами може відправити асинхронне повідомлення до Зовнішньої системи email-сповіщень через Apache Kafka, щоб надіслати листа з підтвердженням на email нового користувача.
* **Сервіс кабінету ментора** та **Сервіс пошуку ментора**: Коли ментор оновлює свій профіль або графік, Сервіс кабінету ментора може відправити асинхронне повідомлення до Сервісу пошуку ментора через Apache Kafka, щоб оновити доступність ментора в системі пошуку.
* **Сервіс бронювань** та **зовнішня система email-сповіщень**: Після того, як користувач здійснив бронювання консультації, система має відправити підтвердження бронювання. Це буде реалізовано асинхронно за допомогою системи повідомлень, такої як Apache Kafka, через Async Messaging. Обмін повідомленнями дозволить відокремити процеси та забезпечить більшу стійкість до помилок та більшу масштабованість.

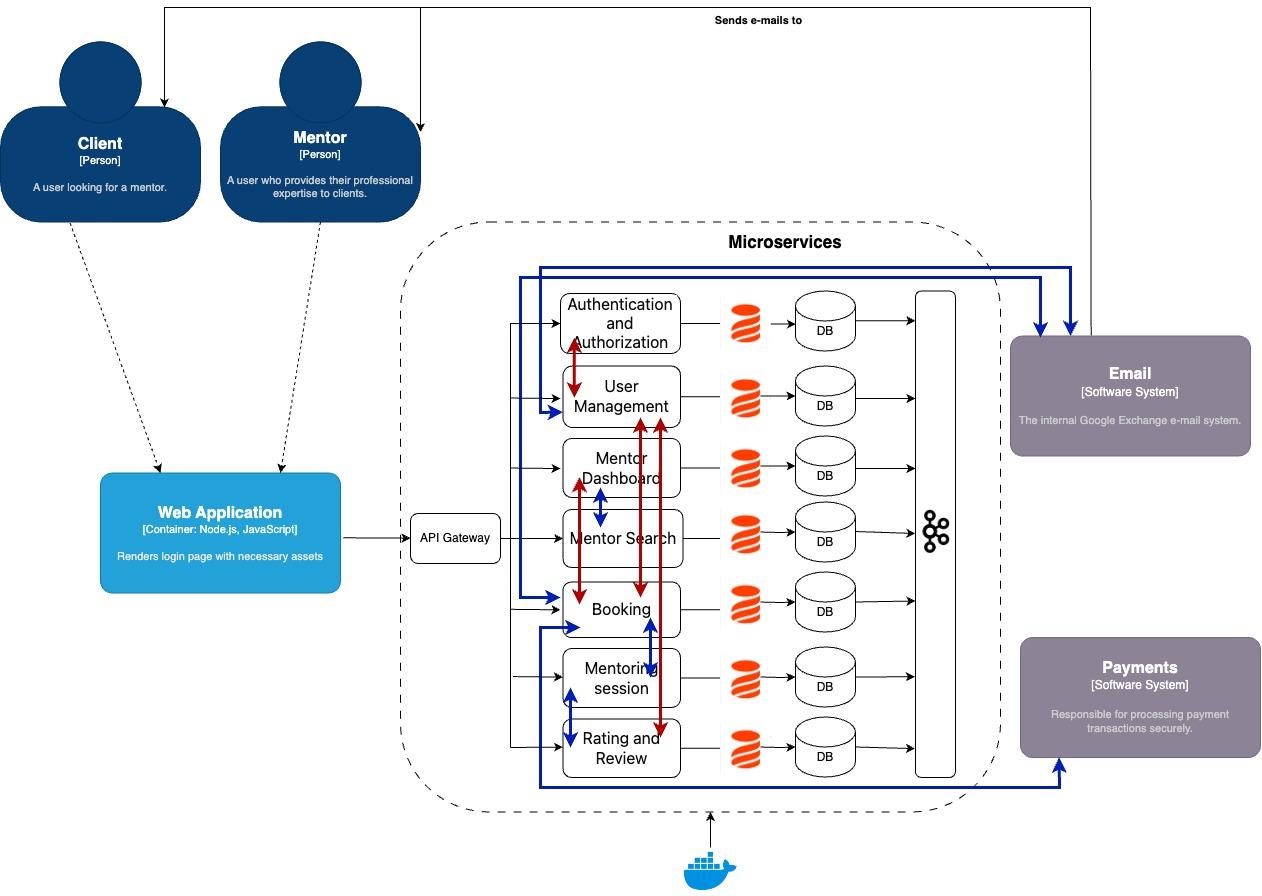
Асинхронні взаємодії відбуваються за допомогою системи обміну повідомленнями, як-от Apache Kafka. Apache Kafka пропонує можливість взаємодії з клієнтами через HTTPS, використовуючи REST Proxy. REST Proxy служить як HTTPS-орієнтований проксі для кластера Kafka. Він надає RESTful інтерфейс до кластера Apache Kafka, що полегшує створення та споживання повідомлень, перегляд стану кластера та виконання адміністративних операцій без потреби в нативному протоколі або клієнтах Kafka.

**2. Робота з даними.**

1. **Патерн роботи з базою даних**: Для роботи з базою даних використовується патерн бази даних на сервіс. Кожен сервіс має свою власну базу даних для забезпечення ізольованості та гнучкості. Також перевагами такого патерну є масштабованість, незалежний розвиток та підтримка, безпека та швидкодія.
2. **Міграційні скрипти:** Кожен сервіс має свої власні міграційні скрипти, які використовуються для оновлення бази даних. Скрипти накачуються автоматично при розгортанні сервісу за допомогою інструменту Liquibase. Використання Liquibase дозволяє ефективно керувати міграціями баз даних у проекті, забезпечуючи контроль, версіонування та автоматизацію процесу змін.
3. **Синхронізація даних:** використання Change Data Capture & Apache Kafka.

* Застосовується Apache Kafka як центральна шина для обміну даними між сервісами та Kafka Connect для збору змін з баз даних і надсилання їх до Kafka. Обрано Apache Kafka через переваги цієї платформи - синхронізацію в реальному часі, масштабованість та надійність.
* Після налаштування Kafka Connect для кожної бази даних, Kafka Connect захоплює зміни в даних зміну та перетворює їх на стандартні Kafka-події, які надсилаються до відповідних тем Kafka, де їх можна обробити іншими сервісами. Сервіси, які потребують доступу до даних, підключаються до відповідних тем Kafka, щоб споживати змінені дані. При отриманні нової події вони можуть оновити свої власні дані або виконати необхідні дії на основі отриманих змін.

**3. Оновити діаграму, зобразити всі зв’язки між сервісами.**

****

**4. Вказати, які саме патерни для мікросервісної архітектури використовуються.**

Наша веб-платформа для менторства використовуватиме декілька патернів мікросервісної архітектури:

1. Database per Service:

Кожен мікросервіс має свою власну базу даних, що дозволяє кожному сервісу незалежно від інших моделювати свою доменну область. Наприклад, Сервіс управління користувачами може мати свою власну базу даних для зберігання даних профілів користувачів.

1. API Gateway:

Цей патерн використовується для обробки клієнтських запитів і перенаправлення їх до відповідних мікросервісів. Наприклад, запити від користувачів до Сервісу пошуку ментора можуть спочатку проходити через API Gateway.

1. Client-side Load Balancing:

Цей патерн допомагає розподіляти навантаження між різними екземплярами мікросервісів. Наприклад, якщо Сервіс бронювань має кілька екземплярів, цей патерн допоможе рівномірно розподілити навантаження між ними.

1. Circuit Breaker:

Цей патерн допомагає запобігти впливу відмови одного сервісу на інші сервіси. Наприклад, якщо Сервіс оцінювання та відгуків тимчасово недоступний, Circuit Breaker може перервати взаємодію з цим сервісом, щоб запобігти впливу цієї відмови на інші сервіси.

1. Event-Driven Data Management:

Цей патерн використовується для синхронізації даних між сервісами. Наприклад, коли користувач робить бронювання через Сервіс бронювань, ця інформація може бути передана до Сервісу кабінету ментора через події.

1. Externalized Configuration:

Цей патерн допомагає управляти конфігурацією мікросервісів за межами коду. Наприклад, конфігурація з’єднання з зовнішньою системою оплати може бути зовнішньою.

1. Service Discovery:

Цей патерн допомагає мікросервісам знаходити і взаємодіяти один з одним. Наприклад, Сервіс бронювань може використовувати Service Discovery для знаходження і взаємодії з Сервісом кабінету ментора.

Ці патерни допоможуть платформі для менторства ефективно використовувати мікросервісну архітектуру, забезпечуючи гнучкість, масштабованість та надійність. Кожен мікросервіс може розвиватися незалежно, що сприяє швидкому внесенню змін та адаптації до вимог користувачів. Зовнішні системи, такі як система оплати та система email-сповіщень, дозволяють платформі фокусуватися на своїй основній функціональності - менторстві, в той час як спеціалізовані задачі, такі як оплата та сповіщення, обробляються надійними зовнішніми службами.