### **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 № 2

### з дисципліни

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

### забезпечення»

### «**Архітектура веб-застосунку для пошуку ментора**»

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

### **Перевірив** Зубик Л. В.

### **Київ 2024**

**Практична робота №2**

**Тема:** Формування та документація технічних рішень додатку. (*System/Software design*)

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

**Завдання:**

***Продовжуєте тему із першої лабораторної***

1. Визначити нефункціональні вимоги
2. Визначити архітектуру системи та архітектурні рішення.
3. Визначити правила опису *API* (*API design guidelines*). Вибрати та розібрати існуючий *guideline* або розробити власний.
4. Описати *API* компонент на базі Open API, обов’язково використовуючи обраний *guideline*. Для реалізації цього пункту можна вибрати один з двох підходів
   1. Згенерувати Open API документацію із коду (але в цьому випадку реалізація демонстраційного сервісу мінімалістична).
   2. Написати Open API документацію вручну.
   3. Для презентації використати swagger або Redoc

**Виконання**

1. **Нефункціональні вимоги.**

**Продуктивність:**

* *Сценарій:* Платформа повинна відповідати на дії користувача протягом 500 мілісекунд з моменту надсилання запиту та завантажувати список менторів, профіль ментора чи його актуальний календар протягом 3 секунд.
* *Критерій:* 95% запитів повинні мати час відповіді менше 500 мс при одночасному навантаженні 1000 користувачів, а списки менторів та їх актуальні дані мають завантажуватись протягом 3 секунд або менше при тестуванні на різних пристроях та швидкостях Інтернету.
* *Сценарій:* Система повинна надсилати сповіщення на електронну пошту користувача про деталі зарезервованої сесії та про успішну оплату протягом 5 хвилин з моменту виникнення події.
* *Критерій:* 95% сповіщень про зарезервовану сесію та успішну оплату мають бути надіслані користувачам протягом 5 хвилин з моменту виникнення події при одночасному навантаженні 1000 запитів на надсилання сповіщень.

**Масштабованість:**

* Сценарій: Платформа повинна забезпечувати масштабованість для відповіді на збільшення кількості користувачів протягом піку активності.
* *Критерій*: Система повинна бути здатна автоматично масштабуватись горизонтально, додавати нові сервери для обробки навантаження без втрати продуктивності або доступності. Таким чином система має витримувати пікове навантаження 10,000 одночасних користувачів без значного падіння продуктивності.

**Надійність**:

* *Сценарій*: Система повинна мати можливість автоматично відновлюватися протягом 5 хвилин у випадку відмови сервера.
* *Критерій*: Середній час відновлення після відмови сервера повинен бути менше 5 хвилин, а доступність системи повинна бути не менше 99.9% часу.

**Безпека**:

* Сценарій: Усі особисті дані користувачів, включаючи фінансові дані, повинні бути зашифровані у стані спокою та під час передачі.
* Критерій: Всі з'єднання з веб-сайтом повинні використовувати протокол HTTPS. Доступ до чутливої інформації користувачів та адміністративних функцій повинен бути обмежений на основі контролю доступу на основі ролей (Role-Based Access Control, RBAC), щоб запобігти несанкціонованим діям.

**Сумісність:**

* *Сценарій*: Платформа повинна забезпечувати сумісність з основними сучасними веб-браузерами, такими як Google Chrome, Mozilla Firefox, Safari та Microsoft Edge, з метою забезпечення єдиної користувацької взаємодії на різних пристроях та веб-браузерах.
* *Критерій*: Підтримка платформи в різних веб-браузерах має забезпечуватися коректно й безперервно, за винятком запланованих оновлень.

**Підтримка користувачів:**

* *Сценарій*: Система підтримки користувачів повинна бути доступна для допомоги користувачам у вирішенні будь-яких технічних проблем або запитань, які вони можуть мати.
* *Критерій*: Канали підтримки (електронна пошта) повинна бути доступна протягом визначених годин, щоб забезпечити користувачам своєчасну допомогу та підтримку.

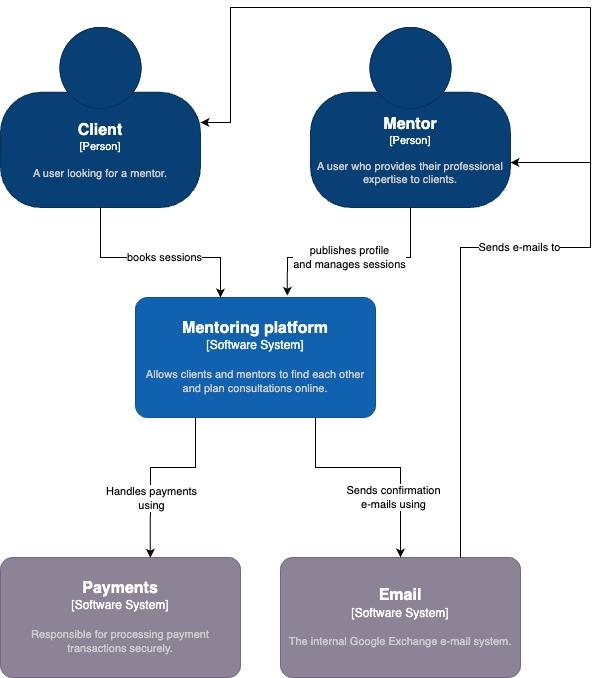
**Юзабіліті:**

* *Сценарій*: Користувачі повинні легко розуміти і використовувати інтерфейс платформи без особливих навичок або інструкцій.
* *Критерій*: Користувацький інтерфейс повинен бути протестованим та продуманим, щоб бути інтуїтивно зрозумілим і дружнім до користувача, забезпечуючи легку навігацію та доступ до основних функцій з мінімальною кількістю натискань.

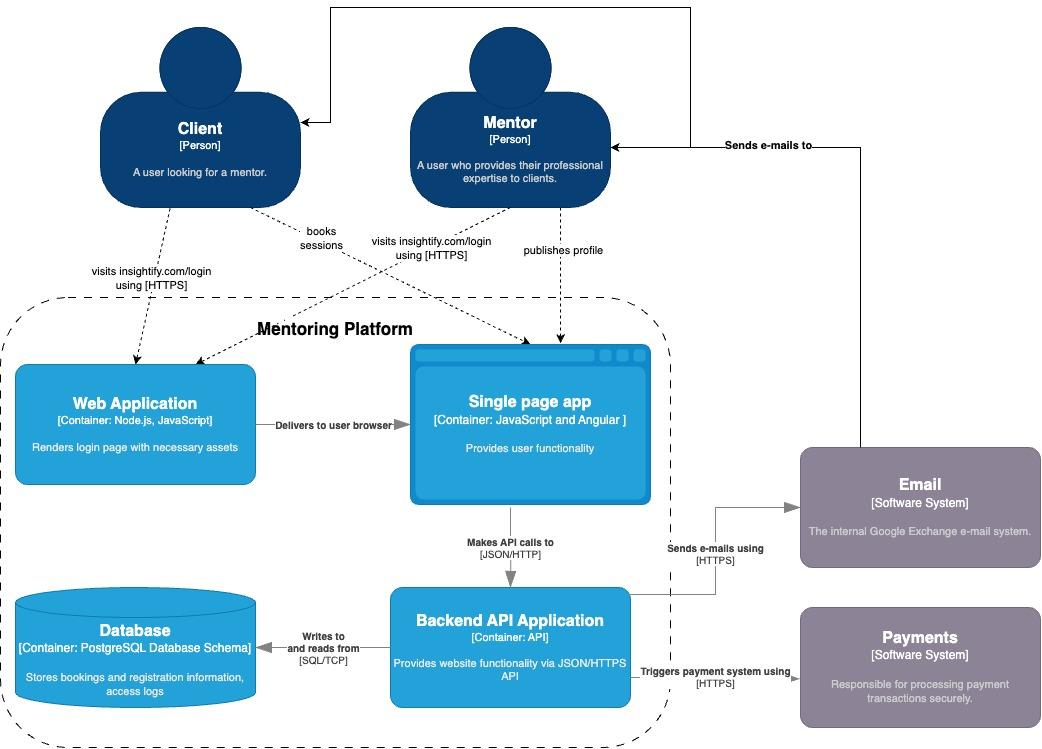
**Обслуговуваність:**

* *Сценарій*: Адміністраторам системи повинно бути легко виявляти, діагностувати та виправляти проблеми з функціонуванням системи, а також впроваджувати нові функції.
* *Критерій*: Система повинна мати інструменти моніторингу, документації та бути реалізованою за допомогою модульної архітектури та найкращих практик написання коду.

**2. Архітектура системи та архітектурні рішення.**

****

Діаграма С4, рівень 1 (Context)



Діаграма С4, рівень 2 (Containers)

**Особливості платформи:**

* Веб-додаток, призначений для передачі та отримання навичок і досвіду за допомогою менторства.
* Ментори можуть зареєструватися на платформі та опублікувати свій профіль. При цьому вони завантажують своє фото, описують кваліфікації та навички, вказуть освітні досягнення, роки досвіду, вільні години та ставку.
* Клієнти можуть реєструватися на платформі та здійснювати пошук ментора за навичками чи різноманітними фільтрами.
* Клієнти мають змогу ознайомитись з профілем будь-якого ментора, після чого можна обрати відповідний час консультації серед його вільних годин.
* Бронювання консультації можливе після оплати, яка відбувається через інтегровані платіжні шлюзи. При цьому підтвердження оплати та дані бронювання надсилаються на пошту клієнта.
* Обрані дата і час повинні бути забезпечені для ментора та клієнта як зарезервовані для конкретної консультації. Вони можуть бути перенесені або відмінені клієнтом не пізніше, ніж 24 години до консультації.
* Клієнти мають змогу оцінювати менторів та залишати відгуки про сесії, а також бачити відгуки інших у профілі ментора.

**3. API design guidelines.**

1. Протокол роботи: HTTPS

2. Формат даних: JSON

3. Організація API навколо ресурсів: REST API реалізовано для ресурсів, що відображають обʼєкти, над якими виконуються операції, та можуть бути сутністю або службою. Вони забезпечують спосіб ідентифікації ресурсу за його URI.

4. Правила іменування ресурсів та полів

* Обирати чіткі, лаконічні англійські терміни для назв.
* Використовувати іменники у множині, які точно описують призначення або зміст ресурсу чи поля. Уникати невизначених або надто загальних термінів, що можуть призвести до плутанини.
* Застосовувати lowerCamelCase для назв, що складаються з декількох слів.
* Вибирати назви, які однозначно описують entity або дію. Уникати термінів, які можуть бути трактовані по-різному (в т.ч. types, elements, resources, items, values, entries, instances, objects тощо).
* Для унікальних ідентифікаторів ресурсів використовувати конкретні назви замість загальних. Наприклад, замість id краще використовувати userId або orderId.
* Параметри запитів також мають слідувати правилу lowerCamelCase. Це стосується всіх параметрів, включно з тими, що використовуються для фільтрації, сортування та пагінації.

5. Правила версіонування

* Всі версії API мають бути чітко позначені та легко ідентифіковані. Необхідно використовувати прості числові позначення, такі як v1, v2, тощо, де v позначає "версія".
* Необхідно включати версію API безпосередньо у базовий URL запиту, наприклад: https://api.example.com/v1/resource.
* Рекомендується випуск альфа- і бета-версій перед випуском стабільної версії для обмеженого кола тестувальників. Альфа-версії можуть бути нестабільними і містити значні зміни, в той час як бета-версії повинні бути ближчі до фінального продукту і зосереджені на виявленні помилок та дрібних вдосконаленнях.
  + Альфа-версії повинні позначатися як vXalpha, наприклад, v2alpha, для вказівки на ранню стадію розробки.
  + Бета-версії повинні позначатися як vXbeta, наприклад, v2beta, що вказує на більш стабільну, але тестову версію.
* При внесенні змін у API, які не сумісні з попередніми версіями, необхідно випускати нову версію.
* Необхідно забезпечити чітку та доступну документацію для кожної версії API, включаючи альфа- і бета-версії. Варто включати інформацію про нові функції, виправлення помилок, видалені або застарілі елементи.

6. HTTP методи, що використовуються: стандартні - GET (Отримати об’єкт/список), POST (створити), PUT (редагувати), DELETE (видалити).

7. Формати запитів та відповідей (request/response payload) для кожного типу запиту.

а. формат (query params/body) для всіх типів запитів

GET /mentors/{mentorId}: Отримати інформацію про ментора за його ідентифікатором.

POST /mentors: Створити новий профіль ментора. Дані про ментора передаються в тілі запиту.

GET /mentors: Отримати список всіх менторів.

POST /sessions: Створити нову сесію менторства. Дані про сесію передаються в тілі запиту.

GET /sessions/{sessionId}: Отримати інформацію про сесію менторства за її ідентифікатором.

POST /feedbacks: Залишити відгук про сесію менторства. Дані про відгук передаються в тілі запиту.

GET /feedbacks/{feedbackId}: Отримати відгук за його ідентифікатором.

PUT /feedbacks/{feedbackId}: Оновити відгук за його ідентифікатором. Дані для оновлення передаються в тілі запиту.

DELETE /feedbacks/{feedbackId}: Видалити відгук за його ідентифікатором.

b. пагінація

Пагінація може бути реалізована за допомогою параметрів запиту page та limit. Наприклад, GET /mentors?page=3&limit=20 поверне третю сторінку списку менторів з 20 менторами на сторінку.

c. фільтри/сортування

Фільтри та сортування можуть бути реалізовані за допомогою параметрів запиту. Наприклад, GET /mentors?skill=python&sort=-experience поверне менторів з навичками Python, відсортованих за досвідом у спадаючому порядку.

d. формат помилок

Помилки повинні повертатися у консистентному форматі, що містить код помилки, повідомлення та, за необхідності, детальний опис або посилання на документацію для отримання додаткової інформації.

8. Можливі статус коди, їх пояснення

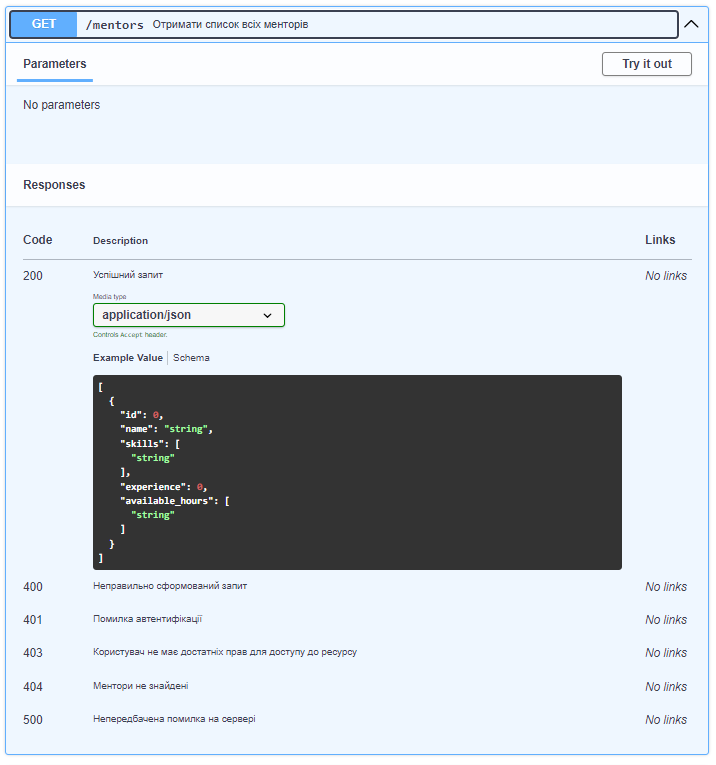
* 200 OK: Успішний запит. Використовується для повернення успішних відповідей на запити. Наприклад, коли користувач отримує список менторів або деталі профілю конкретного ментора.
* 201 Created: Ресурс був успішно створений. Використовується для повертання успішного результату після створення нового ресурсу, наприклад, після реєстрації нового ментора.
* 400 Bad Request: Запит був неправильно сформований. Це може статися, наприклад, якщо користувач надсилає неправильно сформовані дані при спробі реєстрації нового ментора.
* 401 Unauthorized: Помилка автентифікації. Використовується, коли користувач намагається отримати доступ до ресурсу, для якого він не авторизований, наприклад, до профілю іншого користувача.
* 403 Forbidden: Користувач не має достатніх прав для доступу до ресурсу. Це може статися, наприклад, якщо користувач намагається видалити профіль ментора, який належить іншому користувачу.
* 404 Not Found: Ресурс не знайдено. Це може статися, наприклад, якщо користувач намагається отримати доступ до профілю ментора, який був видалений.
* 500 Internal Server Error: Непередбачена помилка на сервері. Це може статися, наприклад, якщо виникає проблема з базою даних під час обробки запиту.

**4. Дизайн API та Open API.**

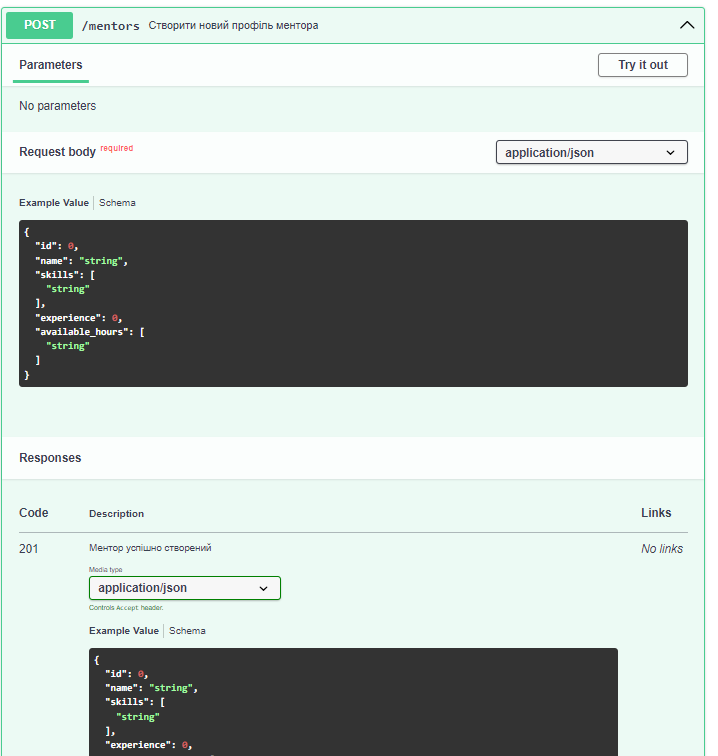
Файл з кодом прикріплено.

****

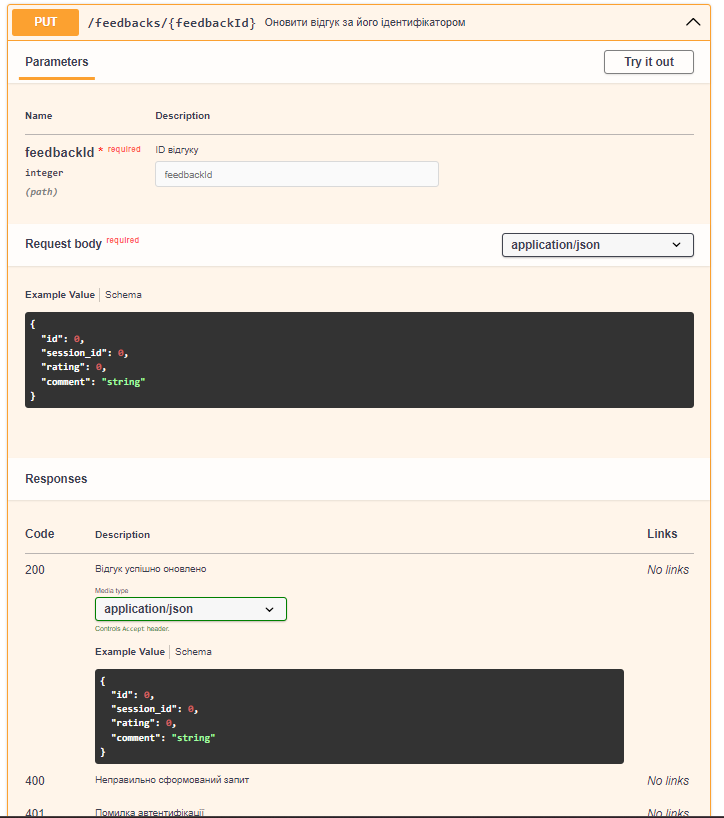
Get/mentors



Post/mentors



Put/feedbacks{feedbackId}



Delete/feedbacks{feedbackId}

