СЛАЙД 1

Тема моєї курсової роботи “GraphQL, як альтернатива REST API”. Дана тема була обрана у зв’язку зі стрімким розвитком технології GraphQL, а також тому що вона набуває все більшої і більшої популярності, але на ринку ще немає достатньої кількості розробників які нею володіють.

СЛАЙД 2

Метою роботи є демонстрування базового функціоналу технологій GraphQL та REST API, а також їх порівняльна характеристика, огляд переваг та недоліків кожної з технологій.

СЛАЙД 3

Основною задачею, яка ставилась переді мною в ході даної курсової роботи – це було дослідження та освоєння архітектурного стилю REST API та технології GraphQL. Ставилась також задача про покращення навичок роботи з мовою програмування javascript, оскільки саме на ній з використанням платформи node js і були розроблені тестові додатки для демонстрації роботи даних технологій. Також необхідно було здійснити порівняння можливостей та принципів роботи REST API та GraphQL.

СЛАЙД 4

Розглядаючи технологію REST API одразу видно, що вона складається з 5-ти основних понять, які ви можете бачити на даному слайді. Саме з цих 5-ти понять і формується запит. З назви методу запиту можна зрозуміти, що саме REST запит буде робити з даними. Наприклад, якщо це GET запит, то чітко зрозуміло, що ми запрошуємо дані для перегляду, якщо ж це POST запит, то зрозуміло, що ми збираємось виконати якусь певну дію з цими даними (швидше за все запис даних). Шлях запиту його ще називають кінцевою точкою або endpoint вказує адресу за якою ми можемо звернутись до цих даних. Тіло запиту використовується в тих випадках, коли ми запрошуємо дані для запису або їх оновлення і зрозуміло, що передаємо якісь певні параметри для виконання цих дій. Статус відповіді надходить клієнту у будь-якому випадку і саме по цьому коду клієнт може зорієнтуватись чи його запит пройшов успішно(якщо повернувся статус 200 все пройшло успішно), чи наприклад якщо клієнт запрошує не існуючі дані йому повернеться статус 404 не знайдено(Not found). Тіло відповіді найчастіше має представлення даних у форматі JSON або HTML, а відповідь зазвичай містить тіло у тому випадку якщо клієнт надіслав на сервер GET запит. У інших же випадках частіше вертаються статуси відповіді.

СЛАЙД 5

Стосовно першого пункту

Тому що ніхто не може визначитися з тим, що саме уявляють в собі всі методи запиту, коди відповіді і т.п. Наприклад те що в одній компанії мається на увазі

під статусом 200 ОК в іншій може нести зовсім іншу інформацію. Це робить дану технологію непередбачуваною для обговорення, оскільки не існує загальноприйнятих правил.

Стосовно другого пункту

Більшість клієнтських і серверних додатків підтримують не всі статуси відповіді, а браузери підтримують не всі HTTP-методи, як от наприклад PUT I DELETE. Саме в таких випадках метод POST може нести зовсім інше смислове навантаження і наприклад відповідати за видалення ресурсу.

Стосовно третього пункту

Для відлагодження запиту прийдеться перевірити одразу цілу купу інформації:

- Метод HTTP запиту, наприклад, POST
- Адресу запиту, наприклад, /garage/3
- Метод, який ми насправді маємо на увазі (в тілі запиту), наприклад, DELETE
- Тіло запиту, наприклад, данні з форми
- Код відповіді, наприклад, 200 ОК
- Код, який ми маємо на увазі (в тілі відповіді), наприклад 206 Partial Content
- Тіло відповіді

Проблеми REST API:

1. REST API великих додатків настільки складний, що це затримує та ускладнює розробку додатку.
2. Розробники хочуть відокремити Front-End та Back-End для пришвидшення розробки додатку.
3. У компаній є мобільний клієнт, і вони турбуються про швидкість відповіді клієнту на його запит.
4. Чіткість формування запиту, тобто клієнт отримає лише те, що йому потрібно і ніякої надлишкової інформації.

СЛАЙД 6

Давайте тепер розглянемо технологію GraphQL та принцип її роботи.
Для того аби запросити дані за допомогою GraphQL, вони спочатку повинні бути визначені в його схемі. Оскільки зі слайду ви вже побачили, що GraphQL є строго типізованим і що для опису полів сутності використовуються вбудовані типи. GraphQL запит кардинально відрізняється від REST запиту, оскільки з використанням GraphQL ми можемо запросити рівно стільки даних скільки нам потрібно і при цьому ми користуємось лише однією кінцевою точкою, але різними запитами. Після того, як GraphQL прийняв наш запит і передав на сервер, на сервері спрацьовує спеціальна функція розпізнавач, яка читає наш запит і повертає рівно стільки даних скільки нам потрібно.

СЛАЙД 7-8

Давайте тепер розглянемо основні плюси і мінуси даних технологій та проведемо порівняльну характеристику.

Основною перевагою GraphQL є те, що він надає клієнту можливість отримати всі необхідні дані за один запит, на відміну від REST з використанням якого клієнт змушений багаторазово звертатися до сервера за необхідною інформацію. Саме ця перевага GraphQL вирішує і ще одну проблему з недовантаженням даних або їхнім надлишковим завантаженням, оскільки у клієнта є можливість чітко вказати, що саме йому потрібно.

(Ще декілька слів про третій стовпець порівняння в таблиці)

СЛАЙД 9

(Розповідаю про те, що відбувається на скріншотах.)

СЛАЙД 10

(Декілька слів про висновки зі слайду.)