Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №9
Технологія розроблення програмного забезпечення

«Клієнт-серверна архітектура. Мікросервіси.»

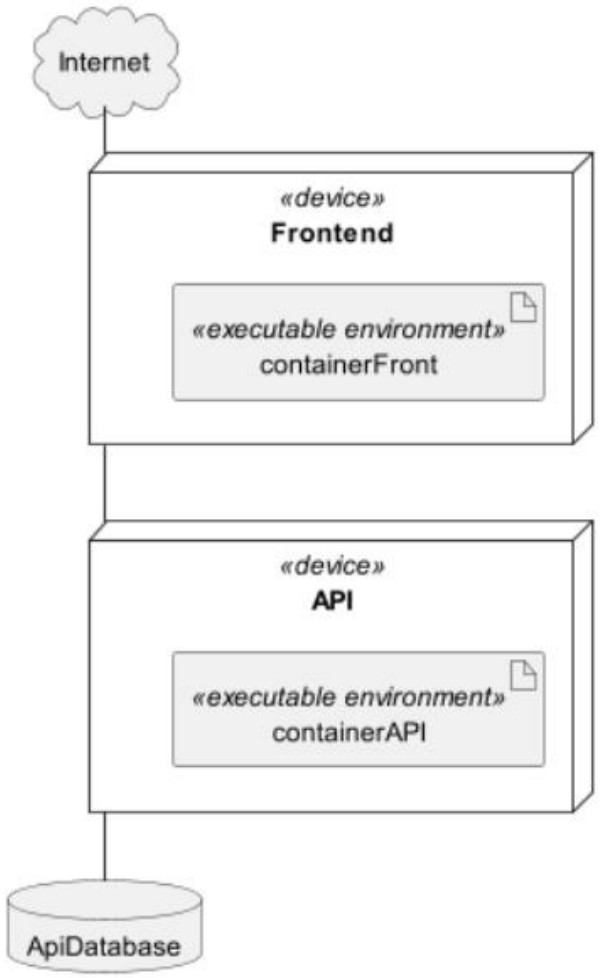
Варіант 19

Виконав
студент групи ІА-13
Павлюк Оскар Ігорович

Перевірив:
Мягкий М.Ю.

Київ 2023р.

Тема : Клієнт - серверна архітектура
Виконання: Діаграма розгортання



Пояснення Архітектури:

WebSocketConfig та AdaptedWebSocketConfigurer: Конфігурація WebSocket для нашого додатку. registerStompEndpoints визначає точку входу WebSocket, а configureMessageBroker налаштовує брокер повідомлень для обміну повідомленнями між сервером та клієнтом.

ChatController: Контролер для обробки повідомлень в чаті. Метод sendMessage відправляє повідомлення на адресу "/app/chat.sendMessage", а addUser визначає, коли користувач приєднується до чату та повідомляє інших користувачів про це.

WebSocketEventListener: Слухач подій WebSocket, який відслідковує, коли користувачі приєднуються та від’єднуються від сервера, та надсилає сповіщення про це іншим користувачам у чаті.

DecoratorMediator, BadWordCheckDecorator, ConsoleMessageDecorator, MessageDecorator: DecoratorMediator взаємодіє з різними декораторами. BadWordCheckDecorator перевіряє повідомлення на наявність неприпустимих слів, а ConsoleMessageDecorator виводить повідомлення в консоль перед тим, як воно буде відправлене у чат.

HTML/CSS/JS код: Це клієнтська частина додатку. Коли користувач вводить ім'я, відбувається з'єднання через WebSocket, і коли користувач відправляє повідомлення, воно обробляється на сервері та відображається на всіх підключених клієнтах.

Щоб краще зрозуміти взаємодію, можна розглядати кожен компонент окремо:

Коли користувач вводить ім'я, відбувається подія connect, створюється WebSocket, відправляється ім'я на сервер, та клієнт приєднується до чату.

Повідомлення, відправлені клієнтом, надсилаються на сервер, де вони обробляються через декоратори, та повертаються на сервер для розсилання іншим клієнтам.

Відсутність зв'язку через WebSocket викликає виклик методу handleWebSocketDisconnectListener, який обробляє подію відключення користувача та повідомляє інших учасників чату.

Цей додаток ілюструє використання WebSocket для створення реального часу обміну повідомленнями та застосування паттерну "Decorator" для обробки повідомлень перед їхнім відправленням.

Висновок: На цій лабораторній роботі я познайомився з різними типами архітектур, а також на практиці спроектував та реалізував паттерн «Client-Server», згідно зі своїм варіантом, у своєму проекті.