

ПРАКТИЧНЕ ЗАНЯТТЯ № 1 (JSON)

КРИТЕРІЇ ОЦІНКИ ПРОЄКТУ

Вимога	Задовільно (60%)	Добре (80%)	Відмінно (100%)
Організація проєкту			
Проєкт містить усі артефакти, необхідні для його запуску та перевірки працездатності.	+	+	+
Проєкт відповідає обраній предметній області.	+	+	+
Схема та валідація JSON			
Створена та описана JSON Schema, що є коректною та валідною, відповідає обраній предметній області та описує не лише загальні характеристики.	+	+	+
Створений та описаний JSON-документ, що є коректним та валідним щодо розробленої схеми.	+	+	+
Опис властивостей (полів)			
Описані всі поля JSON-документа із зазначенням частини обмежень (minimum, maximum, minLength, maxLength, pattern тощо).	+	+	+
При описі властивостей складних типів використано вкладені об'єкти (object) або масиви (array).	+	+	+
Використання додаткових інструкцій			
Має бути описане поле (property), що містить визначений список (enum) можливих значень.		+	+
Описане обов'язкові поля (required), їх тип та обмеження на значення.		+	+

Комплексне застосування			
Описані всі поля, із зазначенням всіх типів обмежень (з переліку: minLength, maxLength, minimum, maximum, exclusiveMinimum, exclusiveMaximum), є описані регулярні вирази (pattern).			+
При описі компонентів використані як прості (string, number, boolean), так і складні (object, array) типи.			+
Проект містить коректне використання items для валідації елементів масивів.			+

Примітка: Якщо не виконані пункти виділені жирним шрифтом, то проєкт взагалі не оцінюється.

ПРАКТИЧНЕ ЗАНЯТТЯ № 1
КРИТЕРІЇ ОЦІНКИ ПРОЄКТУ

Вимога	Задов (60)	Добре	Відмін
Проект містить усі артефакти потрібні для його запуску і перевірки працездатності.	+	+	+
Проект відповідає обраній предметній області	+	+	+
Створена і описана XSD-схема XML-документа, що є коректною (well-formed) і валідною (valid), відповідає обраній предметній області, та описує не лише загальні характеристики.	+	+	+
Створений і описаний XML-документ, що є коректним (well-formed) і валідним (valid) відносно розробленої схеми.	+	+	+
Визначено простір імен та його префікси.	+	+	+
Елементи XML-документа описані із зазначенням обмежень (зі списку: minInclusive, minExclusive, maxInclusive, maxExclusive, length, maxLength, minLength, fractionDigits, totalDigits), використанням регулярних виразів (<xsd: pattern>), а також з використанням і без використання <xsd: restriction>;	+	+	+
Описані всі елементи XML-документа, із зазначенням всіх типів обмежень (список обмежень: minOccurs, maxOccurs, minInclusive, minExclusive, maxInclusive, maxExclusive, length, maxLength, minLength, fractionDigits, totalDigits), є описані регулярні вирази (<xsd: pattern>), є елементи XML-документа, що описані з використанням і без використання <xsd: restriction>.		±	+
При описі компонентів складних типів використані всі 3-ри інструкції: <xsd: all>, <xsd: choice>, <xsd: sequence>;		±	+

Вимога	Задов (60)	Добре	Відмін
При описі компонентів складних типів використана інструкція (зі списку: <xsd: all>, <xsd: choice>, <xsd: sequence>);	+	+	+
Має бути описаний компонент (<xsd: element>), що містить визначений список (<xsd: enumeration>) можливих прийнятих значень;	+	+	+
Описаний обов'язковий атрибут, його тип і обмеження на значення;		+	+
Описаний необов'язковий атрибут, його тип і обмеження на значення;		±	+
Створена таблиця стилів XSLT, яка дозволяє відобразити XML документ в браузері. При цьому використовуються різні формати даних (шрифт: розмір шрифту, накреслення; таблиці, параграфи, списки; реалізована фільтрація даних).		±	+

Примітка: Якщо не виконані пункти виділені жирним шрифтом, то проект взагалі не оцінюється.

ПРАКТИЧНЕ ЗАНЯТТЯ № 2
КРИТЕРІЇ ОЦІНКИ ПРОЄКТУ

Вимога	Задов (60+%)	Добре (75+%)	Відмін (90+%)
Організація та запуск			
Проект містить усі артефакти, потрібні для його запуску і перевірки працездатності.	+	+	+
Проект відповідає обраній предметній області та вимогам роботи (наявність відповідних моделей даних).	+	+	+
Розроблено REST Web-сервіс, який коректно запускається.	+	+	+
Внутрішня Архітектура			
Застосунок має внутрішній поділ на три шари: Controller (обробка зовнішніх викликів), Service (бізнес-логіка), Data/Repository .	+	+	+
Міжшарові DTO: У кожному сервісі використовуються DTO для передачі даних між шарами.		+	+
Міжсервісні DTO: При міжсервісній комунікації (OrderService → PaymentService) використовуються спеціалізовані DTO (наприклад, PaymentRequest, PaymentResponse), а не прямі Entity-об'єкти.		+	+
Контракт та документація			
Проект сервісу запускається та надає контракт (WADL, OpenAPI/Swagger).	+	+	+
Розроблено консольний клієнт REST Web-сервісу, який відповідає предметній області.	+	+	+
Проект клієнта запускається та демонструє виклик усіх бізнес-методів Web-сервісу.	+	+	+
Проект клієнта демонструє виклик усіх бізнес-методів Web-сервісу, також з демонстрацією помилок (для перевірки коректності статус кодів відповідей).		±	+
Реалізація REST-методів			
Web-сервіс містить методи, що викликаються по HTTP GET та POST (мінімальний CRUD).	+	+	+
Web-сервіс містить методи, що викликаються по HTTP GET, POST, PUT або PATCH, DELETE (повний CRUD).		±	+

Вимога	Задов (60+%)	Добре (75+%)	Відмін (90+%)
Реалізація ресурсів			
Web-сервіс має методи доступу до ресурсів за ідентифікатором, використовуючи Path Variables (/ {id}) та можливість фільтрації через Query Parameters (?status=...).	±	+	+
Web-сервіс коректно реалізує ієрархію ресурсів (Sub-resource) та/або логіку вкладених ресурсів (наприклад, /orders/ {id}/items).		+	+
Web-сервіс демонструє використання шаблону Resource Locator або еквівалентного механізму для навігації між ресурсами.			+
Інженерні практики			
Реалізовано зберігання даних (persistence) між викликами бізнес-методів (даних предметної області).	− ¹	± ²	+
Усі методи можуть повертати дані у форматах JSON та XML (забезпечено Content Negotiation).	± ³	± ⁴	+
Реалізовано кастомний механізм обробки винятків (Exception Handler) для встановлення коректних статус-кодів відповідей при помилках (наприклад, 404, 400).		+	+
Реалізовано просунуту кастомізацію обробки даних (наприклад, custom message converters) або впровадження контекстних об'єктів (Context Injection).			+

Примітка: Якщо не виконані пункти затінені сірою заливкою, то проєкт взагалі не оцінюється.

¹ Реалізовано як заглушки, що повертають якісь коректні дані, а не null.

² Якщо реалізовано як заглушки, то вони мають повертати якісь коректні дані, а не null.

³ Усі методи можуть повертати дані в одному з форматів JSON **чи** XML.

⁴ Усі методи можуть повертати дані в одному з форматів JSON **чи** XML.

ПРАКТИЧНЕ ЗАНЯТТЯ № 3

КРИТЕРІЇ ОЦІНКИ ПРОЄКТУ

Вимога	Задов (60%)	Добре (75%)	Відмін (90+%)
Архітектура та Декомпозиція			
Проект містить усі артефакти , потрібні для його запуску і перевірки працездатності.	+	+	+
Проект відповідає обраній предметній області та вимогам роботи (наявність відповідних моделей даних).	+	+	+
Проект розділений на мінімум три незалежні процеси (мікросервіси).	+	+	+
Кожен мікросервіс запускається як окреми процес .	+	+	+
Кожен мікросервіс має чітко визначену межу відповідальності (один бізнес-домен).	+	+	+
Кожен мікросервіс звертається тільки до своєї БД.		+	+
Внутрішня Архітектура			
Кожен мікросервіс має внутрішній поділ на три шари: Controller (обробка зовнішніх викликів), Service (бізнес-логіка), Data/Repository .	+	+	+
Міжшарові DTO : У кожному сервісі використовуються DTO для передачі даних між шарами.		+	+
Міжсервісні DTO : При міжсервісній комунікації використовуються спеціалізовані DTO , а не прямі Entity-об'єкти.		+	+
Конфігурація та Запуск			
Для кожного мікросервісу налаштування мережі (порти, адреси) і даних (якщо необхідно) винесено в конфігураційні файли.		+	+
Система запускається комплексно. Консольний клієнт викликає загальнодоступні сервіси, а ті, за необхідності, викликають приховані.	+	+	+
Міжсервісна Комунікація (IPC)			
Реалізовано синхронний виклик між сервісами (REST over HTTP) для забезпечення виконання бізнес-процесів.	+	+	+
Клієнтська логіка виклику мікросервісів інкапсульована та перевикористовується через шар сервісів.		+	+
Використовуються JSON-схеми для визначення та валідації внутрішніх контрактів взаємодії між сервісами.			+

Надійність та Консистентність			
Забезпечено коректне повернення статус-кодів (HTTP) від прихованих сервісів до загальнодоступних.	+	+	+
Реалізовано механізм, що забезпечує зв'язок ідентифікаторів сутностей між сервісами.		+	+
Реалізовано механізм ігнорування/обробки збоїв (fail-fast або fail-silent) при виклику сусідніх сервісів, щоб уникнути каскадного збою системи.			+
Обґрунтовано (усно або письмово), як вирішується проблема розподілених транзакцій при невдалому виклику.			+

Примітка: Якщо не виконані пункти **затінені сірою заливкою**, то проєкт взагалі не оцінюється.

ПРАКТИЧНЕ ЗАНЯТТЯ № 4

КРИТЕРІЇ ОЦІНКИ ПРОЄКТУ

Вимога	Задов (60%)	Добре (80%)	Відмін (100%)
Створення та Реєстрація Сервісів			
Створено та успішно запущено Discovery Service (наприклад, Eureka Server).	+	+	+
Створено та успішно запущено API Gateway як окремий інфраструктурний компонент.	+	+	+
Усі мікросервіси успішно реєструються на Discovery Service під унікальними логічними іменами.	+	+	+
Автоматизація конфігурації: Кожен мікросервіс, Gateway та Discovery Server мають коректну конфігурацію, що дозволяє їм автоматично знаходити один одного (наприклад, адресу самого Discovery Service) без ручного вводу IP-адрес.		+	+
Функціональність API Gateway			
Єдина точка входу: Консольний клієнт (або веб-клієнт) направляє всі зовнішні запити виключно на одну адресу Gateway .	+	+	+
Динамічна маршрутизація: Gateway налаштований на маршрутизацію запитів до загальнодоступних сервісів і використовує Discovery Service для динамічного виявлення їхніх фізичних адрес.	+	+	+
Ізоляція прихованих сервісів: Gateway не надає прямих маршрутів до прихованих сервісів.			+
Оновлення Міжсервісної Комунікації			
Рефакторинг виклику: Внутрішня (міжсервісна) комунікація змінена з жорстко закодованого URL на логічне ім'я сервісу .	+	+	+
Міжсервісне балансування навантаження: Використовується механізм балансування навантаження на стороні сервіса клієнта для динамічного виявлення та розподілу викликів внутрішніх сервісів .		+	+
Навантажувальне тестування: Продемонстровано балансування запитів між двома (чи більше) екземплярами одного мікросервісу.			+
Надійність та Конфігурація			
Fail-over (за наявності Discovery): Система коректно обробляє ситуацію, коли один екземпляр сервісу виходить з ладу (і видаляється з Discovery), і запити автоматично перенаправляються на інший живий екземпляр .			+

Примітка: Якщо не виконані пункти **затінені сірою заливкою**, то проєкт взагалі не оцінюється.

ПРАКТИЧНЕ ЗАНЯТТЯ № 5

КРИТЕРІЇ ОЦІНКИ ПРОЄКТУ

Вимога	Задов (60%)	Добре (80%)	Відмін (100%)
Інтеграція та Контракт			
Клієнт успішно запускається та спрямовує всі запити на API Gateway .	+	+	+
Клієнт демонструє успішний обмін даними (GET/POST) з мінімум двома публічними сервісами.	+	+	+
Коректна обробка вхідних даних: Клієнт відображає дані, отримані з мікросервісів, у зрозумілому для користувача вигляді.	+	+	+
Функціональна повнота (CRUD)			
Реалізовано функціональність "Перегляд (Read): Відображення повного списку замовлень та списку ресторанів.	+	+	+
Реалізовано функціональність "Створення (Create)": Наявність форми для створення нової сутності, яка успішно відправляє POST-запит.		+	+
Реалізовано функціональність "Редагування/Видалення (Update/Delete)" для однієї ключової сутності.		+	+
Бізнес-логіка та Зв'язки			
Складна агрегація: Клієнт демонструє агрегацію даних з кількох сервісів.			+
Користувацька взаємодія: Реалізовано елементи динамічної взаємодії (наприклад, перегляд деталей замовлення).		+	+
Обробка каскадних операцій: Клієнт демонструє виклик операції, яка запускає каскад внутрішніх міжсервісних викликів на бекенді.			+
Надійність та Дизайн			
Обробка помилок (UX): Реалізовано механізм дружнього сповіщення користувача при отриманні помилки від Gateway (HTTP 4xx/5xx).	+	+	+
Валідація вводу (Frontend): Реалізовано базову перевірку вхідних даних на стороні клієнта перед відправкою запиту.		+	+
Захист від дублювання (Ідемпотентність): Реалізовано механізм запобігання дублюванню для неідемпотентних запитів.			+
Дизайн та Юзабіліті: Інтерфейс є інтуїтивно зрозумілим , має логічну навігацію та адекватний візуальний дизайн.			+

Примітка: Якщо не виконані пункти затінені сірою заливкою, то проєкт взагалі не оцінюється.