 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №3**

з дисципліни «Веб-технології та веб-дизайн-2.»

Варіант №13

**Виконав:**

студент гр. БС-83

Розмариця О.А.

**Перевірив:**

ас. Давидько О. Б.,

ас. Матвійчук О. В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2020

**Завдання**:

1. Розробити архітектурну постановку для веб-додатку. Тематику веб-додатку обрати самостійно.
2. Звіт з роботи має містити такі елементи:
   1. Тематика веб-додатку
   2. Базові вимоги до додатку: функціональні вимоги, запланована кількість користувачів, можливі downtime, інтеграції з іншими сервісами, доступність для людей з обмеженими можливостями, безпека, вимоги до тестування.
   3. Опис високорівневої архітектури. Має бути обґрунтовано вибір моноліту, SOA або мікросервісної архітектури. Має бути представлена діаграма компонентів та їх взаємодії.
   4. Постановка має містити опис низькорівневої архітектури для одного із сервісів (у випадку мікросервісів) або хоча б двох модулів у моноліті. Необхідно розробити UML діаграми класів. У випадку моноліту - діаграма взаємодії модулів.
3. У наступних роботах передбачається реалізація створеної постановки.
4. Тематикою буде онлайн бот для телеграму - онлайн калькулятор для геометричних фігур (розрахунок площі, сторін, об'єму і тп. по заданими даними).
5. Користувач буде вибирати фігуру серед запропонованих, вводити доступні йому параметри і вибирати, який параметр йому потрібно знайти (наприклад, площу трикутника по трьом сторонам). У відповідь бот має присилати рішення. Буде можливість написати адміністратору.

Можливості downtime - так як це бот, він має працювати цілодобово, але вимикати його можна краще всього вночі через мінімальну кількість можливих користувачів. інтеграції з іншими сервісами - Це бот для додатку "Telegram" на комп'ютери та смартфони, тому інтеграція з додатком "Telegram". Можливо доведеться користуватися сторонніми сервісами для побудови малюнків математичних рішень. Доступність для людей з обмеженими можливостями – відсутня. Безпека - боту не будуть надаватись будь-які персональні дані, тому безпека буде на "плечах" додатку "Telegram". Вимоги до тестування - тестування введених даних

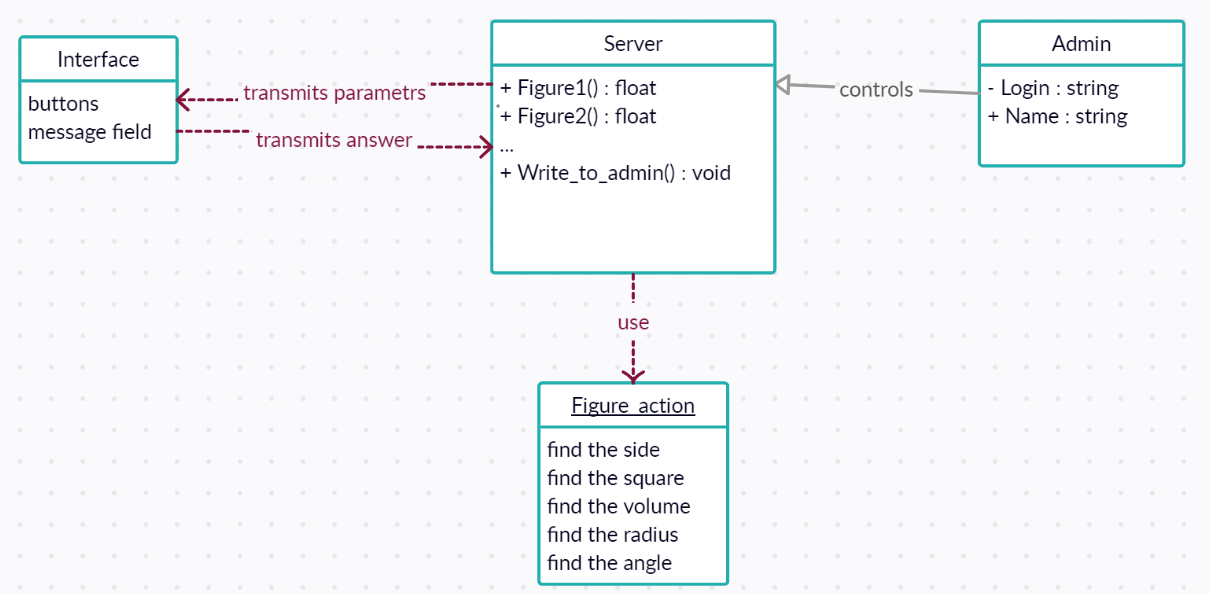
1. Скоріш за все це буде моноліт. Цей вибір обумовлений тим, що додаток не буде масштабним. Він не має великої кількості інтеграцій. І через реалізацією однією людиною та доступних їй ресурсів він не буде масштабуватись

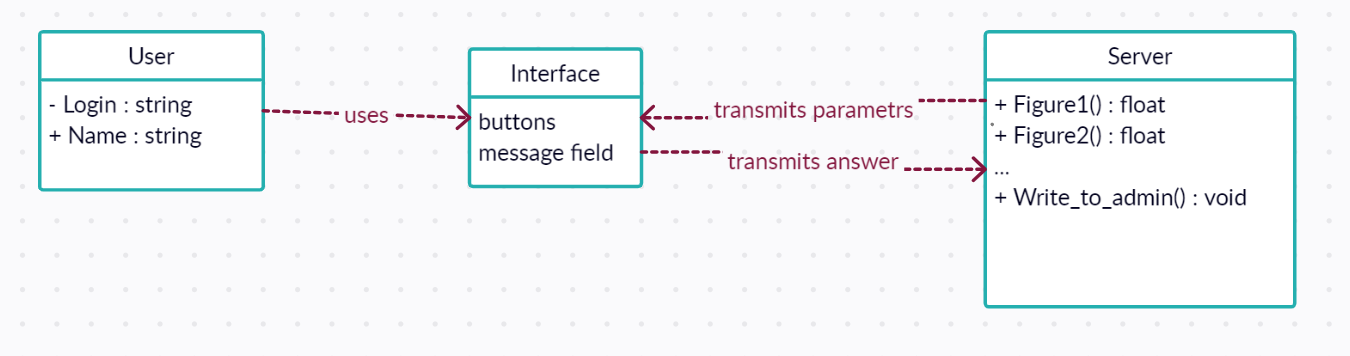
Діаграма компонентів:



1. Модуль "Server" має функціонал виконання задач по заданим даним та надсилання їх відповіді клієнту. Також в разі звернення до адміністратора сервер сповістить його про повідомлення.

Модуль "interface" допомагає користувачеві вибрати необхідну для нього фігуру, вибрати відомі параметри та ввести їх значення. Також через інтерфейс можна вийти на зв'язок з адіністратором. Відповідь на задачу також передається користувачеві через інтерфейс





**Контрольні питання**:

1. Що таке архітектура додатку?

Архітектура додатку - це набір методів і шаблонів, які допомагають розробникам створювати структуровані програми.

Архітектура включає:

* Вибір структурних елементів і їх інтерфейсів, за допомогою яких складена система, а також їх поведінки в рамках співпраці структурних елементів;
* З'єднання обраних елементів структури і поведінки у все більш крупні системи;
* Архітектурний стиль, який направляє всю організацію - все елементи, їх інтерфейси, їх співпраця і їх з'єднання.

1. Різниця між монолітом та мікросервісами

Моноліт використовує централізований цикл обробки запиту користувача, а мікросервіси - децентралізований.

В цілому можна виділити такі плюси і мінуси для цих реалізацій:

|  |  |  |
| --- | --- | --- |
|  | МЛ | МС |
| Декомпозиція і суворі кордони | + | + |
| Можливість наявності власної БД | + | + |
| Простота рефакторінга і перенесення меж | + | - |
| Відсутність накладних витрат на networking | + | - |
| Відсутність накладних витрат на інфраструктуру | + | - |
| Типізація і перевірка компілятором переданих даних | + | - |
| Перевірка компонентних залежностей на етапі компіляції | + | - |
| Підтримка схеми взаємодії компонентів в явному вигляді | + | - |
| Можливість виконання транзакцій між компонентами | + | - |
| Використання різних мов для різних компонентів | - | + |
| Незалежне розгортання / роздільні процеси (~ відмовостійкість) | - | + |
| Можливість незалежного масштабування компонентів | - | + |
| Різний життєвий цикл для різних компонентів | - | + |

1. Що таке шина даних?

Шина даних — шина, призначена для передачі інформації. У комп'ютерній техніці прийнято розрізняти виводи пристроїв за призначенням: одні слугують для передачі інформації (наприклад у вигляді сигналів низького або високого рівнів), інші для поєднання всіх пристроїв (шина адреси) — кому ці конкретні дані призначені.

Основною характеристикою шини даних є її ширина у бітах. Ширина шини даних визначає кількість інформації, яку можна передати за один такт через неї. Паттерн “черга повідомлень”.

1. Чим відрізняються сервіси у SOA та мікросервісах?

* Деталізація сервісів. Компоненти сервісів в архітектурі мікросервісів, як правило, представляють собою спеціалізовані сервіси, які дійсно добре виконують одну задачу. Завдяки SOA розмір сервісних компонентів може варіюватися від невеликих додатків до дуже великих корпоративних сервісів. Насправді, в SOA зазвичай присутній сервісний компонент, Представлений великим продуктом або навіть підсистемою.
* Спільне використання компонентів. Спільне використання компонентів є одним з основних принципів SOA. Насправді, спільне використання компонентів - ось що таке корпоративні сервіси. SOA покращує спільне використання компонентів, тоді як MSA намагається мінімізувати спільне використання через «обмежений контекст». Обмежений контекст відноситься до з'єднання компонента і його даних як єдиного блоку з мінімальними залежностями. Оскільки SOA використовує кілька сервісів для виконання бізнес-запитів, системи, побудовані на SOA, ймовірно, будуть працювати повільніше, ніж MSA.
* Проміжне ПО проти рівня API. У паттерне архітектури мікросервісов зазвичай є так званий рівень API, тоді як в SOA є компонент проміжного програмного забезпечення для обміну повідомленнями. Проміжне програмне забезпечення для обміну повідомленнями в SOA пропонує безліч додаткових можливостей, відсутніх в MSA, включаючи посередництво і маршрутизацію, розширення повідомлень, перетворення повідомлень і протоколів. MSA має рівень API між сервісами і споживачами сервісів.
* Дистанційні сервіси. Архітектури SOA засновані на обміні повідомленнями (AMQP, MSMQ) і SOAP в якості основних протоколів віддаленого доступу. Більшість MSA покладаються на два протоколи - REST і простий обмін повідомленнями (JMS, MSMQ), і протокол, Знайдений в MSA, зазвичай однорідний.
* Гетерогенна сумісність: SOA сприяє поширенню декількох гетерогенних протоколів через компонент проміжного програмного забезпечення для обміну повідомленнями. MSA намагається спростити шаблон архітектури за рахунок скорочення кількості варіантів інтеграції. Якщо ви хочете інтегрувати кілька систем, що використовують різні протоколи, в гетерогенную середу, вам необхідно розглянути SOA. Якщо всі ваші служби можуть бути відкриті і доступні через один і той же протокол віддаленого доступу, тоді MSA - кращий варіант.

1. Паттерни проектування - загальне визначення.

Паттерн проектування - повторювана архітектурна конструкція, що представляє собою рішення проблеми проектування в рамках деякого часто виникає контексту.

1. Що таке DDD?

Предметно-орієнтоване проектування (рідше проблемно-орієнтоване, англ. Domain-driven design, DDD) - це набір принципів і схем, спрямованих на створення оптимальних систем об'єктів. Зводиться до створення програмних абстракцій, які називаються моделями предметних областей. У ці моделі входить бізнес-логіка, що встановлює зв'язок між реальними умовами області застосування продукту і кодом.

1. Що таке CQRS?

Command-query separation (CQS) або англ. command-query responsibility segregation (CQRS) — це принцип імперативного програмування.

Принцип вказує, що метод повинен бути або командою, яка виконує якусь дію, або запитом, що повертає дані, але не одночасно і тим, і іншим. Інакше кажучи, задавання питання не повинно змінювати відповідь. Більш формально, повертати значення можна тільки чистими методами, тобто такими, які не мають побічних ефектів. Слід зазначити, що неухильне дотримання цього принципу унеможливлює відстеження кількості викликів запитів.