ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ

КАФЕДРА СИСТЕМНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**КУРСОВА РОБОТА**

з дисципліни «Архітектура та проектування програмного забезпечення»

на тему: Відкритий науковий портал

Студента курсу IV групи АС-131

напряму підготовки 6.050103

О. С. Карлова

Керівник проф. В.В. Любченко

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Одеса – 2016

**АНОТАЦІЯ**

В даній курсовій роботі створюються та порівнюються два архітектурних проекти для системи відкритого наукового порталу. Для цього, по-перше, виконується специфікація вимог до програмної системи, виявляються атрибути якості та визначаються найвпливовіші з них.

По-друге, спираючись на сценарії варіантів використання, формується перший архітектурний проект. Другий архітектурний проект формується з використанням архітектурних тактик та шаблонів.

Після створення двох архітектурних проектів відбувається їх аналіз за допомогою методів ATAM (метод аналізу компромісних архітектурних рішень) та SAAM (метод аналізу архітектури на основі сценаріїв).

Далі відбувається опис розробленої архітектури.

На завершальному етапі формуються висновки по результатам виконаної роботи.

**ЗМІСТ**

[ВСТУП 4](#_Toc467827023)

[1 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ 5](#_Toc467827024)

[1.1 Функціональні вимоги 5](#_Toc467827025)

[1.2 Нефункціональні вимоги 8](#_Toc467827026)

[2 БАЗОВИЙ АРХІТЕКТУРНИЙ ПРОЕКТ 10](#_Toc467827027)

[2.1 Архітектурний проект на основі сценаріїв варіантів використання 10](#_Toc467827028)

[2.2 Архітектурний проект з використанням архітектурних стилів, шаблонів та тактик 13](#_Toc467827029)

[3 АНАЛІЗ БАЗОВОЇ АРХІТЕКТУРИ 18](#_Toc467827030)

[3.1 Аналіз архітектури за методом SAAM 18](#_Toc467827031)

[3.2 Аналіз архітектури за методом ATAM 19](#_Toc467827032)

[4 ОПИС АРХІТЕКТУРНОГО ПРОЕКТУ 21](#_Toc467827033)

[4.1 Документування подань 21](#_Toc467827034)

[4.2 Документування інтерфейсів 22](#_Toc467827035)

[ВИСНОВОК 24](#_Toc467827036)

[СПИСОК ЛІТЕРАТУРИ 25](#_Toc467827037)

[ДОДАТОК А. РЕЗУЛЬТАТИ АНАЛІЗУ АРХІТЕКТУРИ ЗА МЕТОДОМ SAAM 26](#_Toc467827038)

[ДОДАТОК Б. РЕЗУЛЬТАТИ АНАЛІЗУ АРХІТЕКТУРИ ЗА МЕТОДОМ ATAM 28](#_Toc467827039)

# ВСТУП

У даній курсовій роботі архітектурні рішення створюються для системи, відкритого наукового порталу. Головними особливостями цієї системи є взаємодія з користувачем і надання необхідної інформації.

Головною метою архітектурного проектування є створення архітектури, яка б найкращим чином враховувала функціональні та нефункціональні вимоги до системи.

Через це до задач архітектурного проектування можна віднести визначення ймовірних варіантів архітектурних рішень для системи, порівняння цих варіантів, аналіз запропонованих рішень на підтримку усіх вимог та вдосконалення архітектури у разі необхідності. Також, важливою частиною архітектурного проектування є документування отриманих результатів та обґрунтування прийнятих під час розробки рішень.

# СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

## Функціональні вимоги

Функціональні вимоги до системи формуються у вигляді діаграми прецедентів, яку зображено на рис. 1.1.

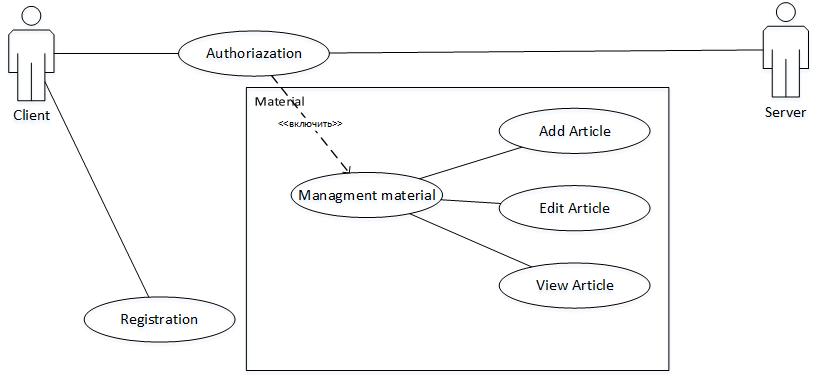


Рисунок 1.1 — Діаграма варіантів використання

Далі для кожного варіанту використання надається сценарій використання.

Ім'я: "Реєстрація"

Передумова: Вхід в програму користувача.

основний сценарій

Гарантія успіху: Реєстрація

основний сценарій

1. Користувач заходить в програму

2. Користувач вводить дані для реєстрації і натискає кнопку реєстрації

3. Система підтверджує реєстрацію та видає дані для доступу.

Альтернативний сценарій.

3.1 Система не дозволяє реєстрацію. Перехід до п 2

Назва: "Авторизація"

Передумова: Вхід в програму користувача.

основний сценарій

Гарантія успіху: Вхід в панель управління

Основний сценарій

1 Користувач заходить в програму

2 Користувач вводить дані для авторизації

3. Система підтверджує авторизацію і видає права доступу, і користувач входить в панель управління.

Альтернативний сценарій.

3.1 Система не підтверджує дані. Перехід до п 2.

Назва: "Додавання матеріалу"

Передумова. Зареєстрований користувач авторизувався.

Гарантія успіху: Новий матеріал - буде додано.

основний сценарій

1. Користувач вибирає категорію (групу) матеріалу який хоче додати

2. Система перевіряє авторизацію і наявність прав у даного користувача.

а) Система забороняє додавання матеріалу, через брак прав. Вихід

3. Користувач додає потрібний матеріал.

7. Система перевіряє авторизацію користувача підтверджує додавання матеріалу.

7.а Система надала відмову в доступі. Висновок помилки. Вихід.

Назва: "Перегляд матеріалів (View)"

Передумова. Користувач вибрав необхідну публікацію

Гарантія успіху: користувач отримав доступ до обраної інформації.

1. Користувач зайшов в розділ виведення статей

2. Система перевіряє авторизацію і права користувача

3. Система виводить список доступних матеріалів

4. Користувач вибрав потрібний результат пошуку.

5. Система відкриває обраний об'єкт

a. Система видала помилку, повернення до п1.

6. Користувач після перегляду виходить.

Назва: "Редактор матеріалів (Edit) "

Передумова. Користувач вибрав необхідну публікацію, додану поточним користувачем.

Гарантія успіху: користувач внесе зміни до обраної інформації.

1. Користувач вибрав нудну статтю.

2. Система перевіряє авторизацію користувача

3. Користувач вносить зміни

4. Користувач підтверджує зміни і відправляє їх на сервер.

a. Система видала помилку, повернення до п1.

5. Користувач виходить з модуля.

## 

## Нефункціональні вимоги

Нефункціональні вимоги є вимогами до якості системи. Вони визначаються за атрибутами якості. Сценарієм атрибута якості це вимога шляхом виконання якої цей атрибут реалізується [1].

Атрибути якості та визначені для них сценарії надано у таблиці 1.1.

Таблиця 1.1 — Сценарії якості

|  |  |  |
| --- | --- | --- |
| Атрибут | Властивість ПЗ | Сценарій |
| Функціо-  нальність | Захищеність | При спробі авторизації з невірним логіном / паролем система повинна відхилити вхід з імовірністю 0,99. |
| Ефектив-  ність | Часові характеристики | Час виконання запиту до сервера не повинно перевищувати 5 секунд. |
| Обробка даних на сервері не повинна перевищувати 10 мс |
| Використання ресурсів | У ході формування зображення система використовує не більш ніж 512 Мб оперативної пам'яті. |
| Під час порівняння зображень системою використовується не більше ніж 512 Мб оперативної пам'яті. |
| Для роботи системи необхідно не більше 100 мб фізичної пам'яті на диску. |
| Надійність | Стійкість до відмов | При відправці некоректних запитів, сервер повинен відновити роботу з ймовірність не менше 0,95. |
| При отриманні некоректного відповіді, клієнт повинен відновити роботу з ймовірність не менше 0,5 |

Продовження таблиці 1.1

|  |  |  |
| --- | --- | --- |
| Мобіль-ність | Адаптованість | Програма повинна коректно працювати на всіх машинах з ос Windows 8 і вище з ймовірністю 0,8 |
| Програма повинна коректно працювати на всіх версіях Java починаючи з 8 і вище з ймовірністю 0,8 |

# БАЗОВИЙ АРХІТЕКТУРНИЙ ПРОЕКТ

## Архітектурний проект на основі сценаріїв варіантів використання

Побудова архітектурного проекту на основі сценарії виконується здебільшого спираючись на власний досвід розробника архітектури.

Для цього відбувається виділення компонентів системи, визначення інтерфейсів цих компонентів, з'єднання або роз'єднання компонентів за необхідності збільшити або зменшити коло їх обов'язків.

У результаті, отриманий набір компонентів повинен забезпечувати реалізацію всієї функціональності, не повинен викликати помітних проблем з продуктивністю та має складатися з компонентів, що мають чітко окреслене коло обов'язків.

Таким чином, було виявлено компоненти, на котрих побудовано архітектуру системи:

* Client — компонент, котрий організує взаємодію клієнту з користувачем та сервером;
* Server — компонент, котрий несе відповідальність за роботу зі кліентом та компонентом системі;
* ArticleList, UserList
* — компонент, котрий займаеться обробкою данних між сервером та БД.

Взаємодію цих компонентів зображено за допомогою процесного подання архітектури на рис. 2.1 — 2.8.

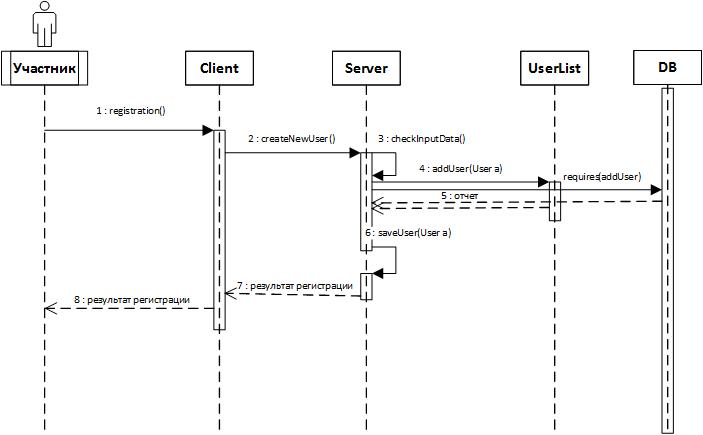


Рисунок 2.1 — Діаграма взаємодії для сценарію "Зареєструватися"

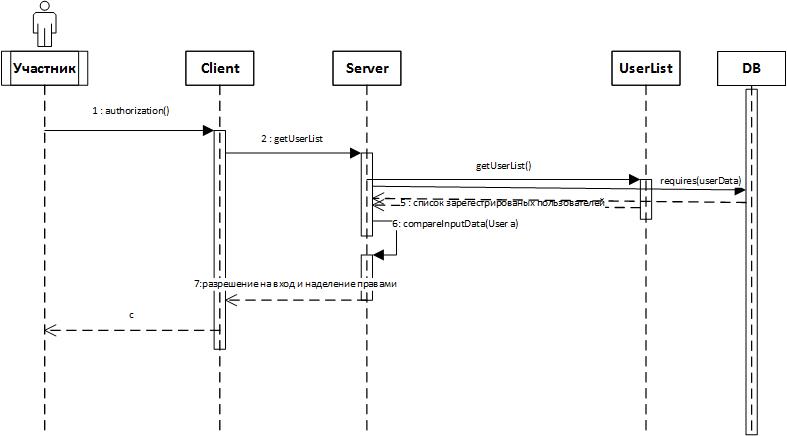


Рисунок 2.2 — Діаграма взаємодії для сценарію " Авторизуватися "

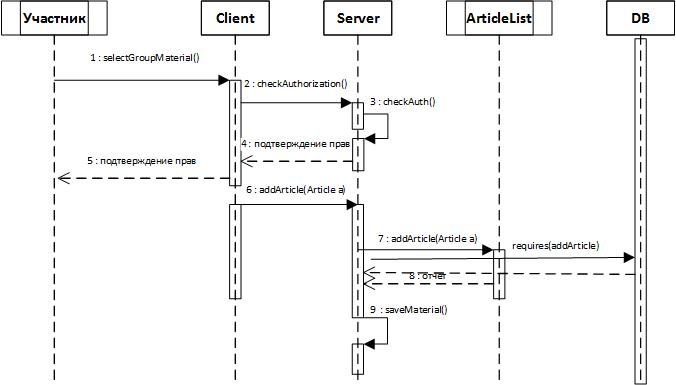


Рисунок 2.3 — Діаграма взаємодії для сценарію "Додати статтью"

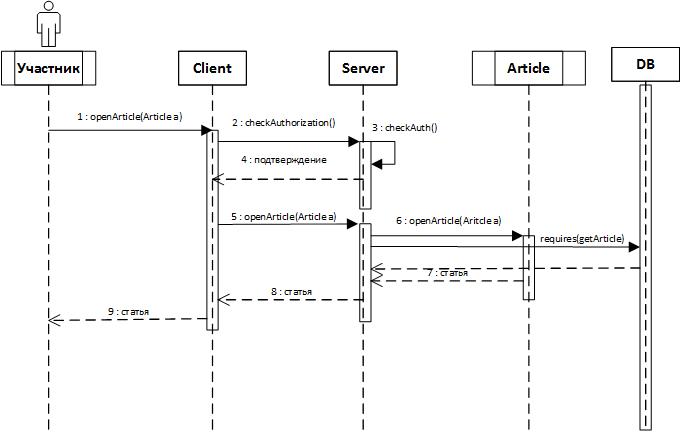


Рисунок 2.4 — Діаграма взаємодії для сценарію "Переглянути статтю"

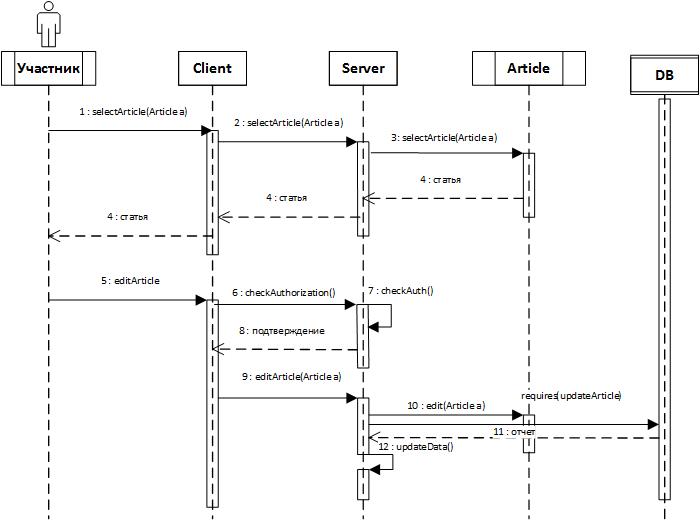


Рисунок 2.5 — Діаграма взаємодії для сценарію "Редагувати статтю"

На рисунку 2.6 надано діаграму, що описує логічне подання архітектури програмної системи.

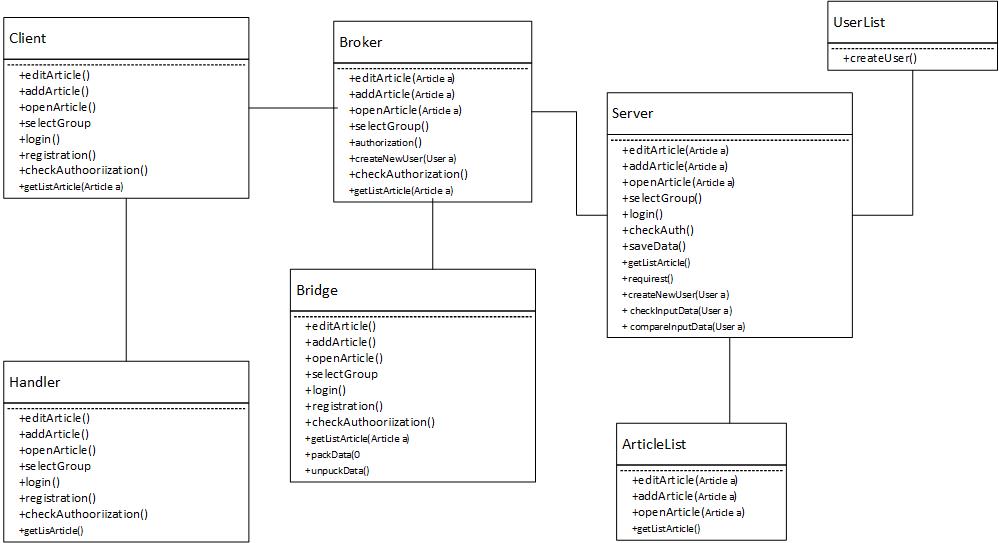


Рисунок 2.9 — Логічне подання архітектури програмної системи на основі сценаріїв

## Архітектурний проект з використанням архітектурних стилів, шаблонів та тактик

Архітектурні стилі це певні шаблони, що визначають структуру компонентів та спосіб їх взаємодії для родини програмних систем з загальним контекстом. Вони дозволяють визначати доречну архітектуру для системи в залежності від її важливіших характеристик таких як, наприклад, розподіленість, інтерактивність, тощо. Стилі полегшують процес проектування архітектури та дозволяють повторно використовувати накопичений досвід.

Крім стилів, для задоволення нефункціональних вимог до системи, використовуються архітектурні тактики. Вони дозволяють забезпечити підтримку визначених у вимогах сценаріїв якості безпосередньо в архітектурному проекті.

Для визначених у табл. 1.1 сценаріїв було обрано тактики, які представлені у таблиці 2.1.

Таблиця 2.1 — Сценарії якості та визначені для них тактики

|  |  |  |
| --- | --- | --- |
| Атрибут | Сценарій | Тактика |
| Функціо-нальність | Користувач буде сприйнятий системою як зловмисник після невдалих спроб отримати доступ до системи з невірним логіном та паролем | Тактика "Авторизація".  Забезпечує підтвердження наявності в користувача доступу до даних та функціоналу системи. |
| Ефектив-ність | Час виконання запиту до сервера не повинно перевищувати 5 секунд.(переглянути статтю) | Тактика "Підвищення обчислювальної ефективності".  Покращення алгоритмів обробки даних. |
| Додавання статті, не повина займати на сервері біль 10 мс. | Тактика "Підвищення обчислювальної ефективності".  Покращення алгоритмів обробки даних. |
| У ході додавання, або перегляду статті система використовує не більш ніж 512 Мб оперативної пам'яті. | Тактика "Підвищення обчислювальної ефективності".  Покращення алгоритмів обробки даних. |
| Під час загрузки статті системою використовується не більше ніж 512 Мб оперативної пам'яті. | Тактика "Підвищення обчислювальної ефективності".  Покращення алгоритмів обробки даних. |

Продовження таблиці 2.1

|  |  |  |
| --- | --- | --- |
| Надій-ність | Користувач отримує відповідь на свій запит з вірогідністю не менше ніж 0,92. | Тактика "Виключення".  У разі неможливості виконати запит користувача компонент генерує виключення, яке повідомляє користувача про помилку, після чого оброблюється належним чином. |
| Система повідомить користувача про некоректність вхідних даних не менше ніж у 85% випадків  Тактика "Виключення".  У разі виявлення некоректних вхідних даних буде згенероване виключення, користувачу буде надано повідомлення про помилку. | |

Через те, що система є інтерактивною та клієнт-сервірна, для проектування її архітектури був використаний архітектурний стиль та Брокер MVC (Model-View-Controller) через компонент міст .

За цим шаблоном система була декомпозована на такі компоненти:

* BrokerA:Broker — представляє компонент шаблону Брокер ;
* BridgeA: Bridge - представляє компонент Broker шаблону Брокер ;
* BridgeB: Bridge - представляє компонент шаблону Брокер ;
* BrokerB:Broker - представляє компонент шаблону Брокер ;
* View – представляє компонент View шаблону MVC ;
* UserList, ArticleList – реалізує Controller шаблону MVC.

Процесне подання архітектури з використанням архітектурних стилів представлено на рис. 2.10 — 2.17.

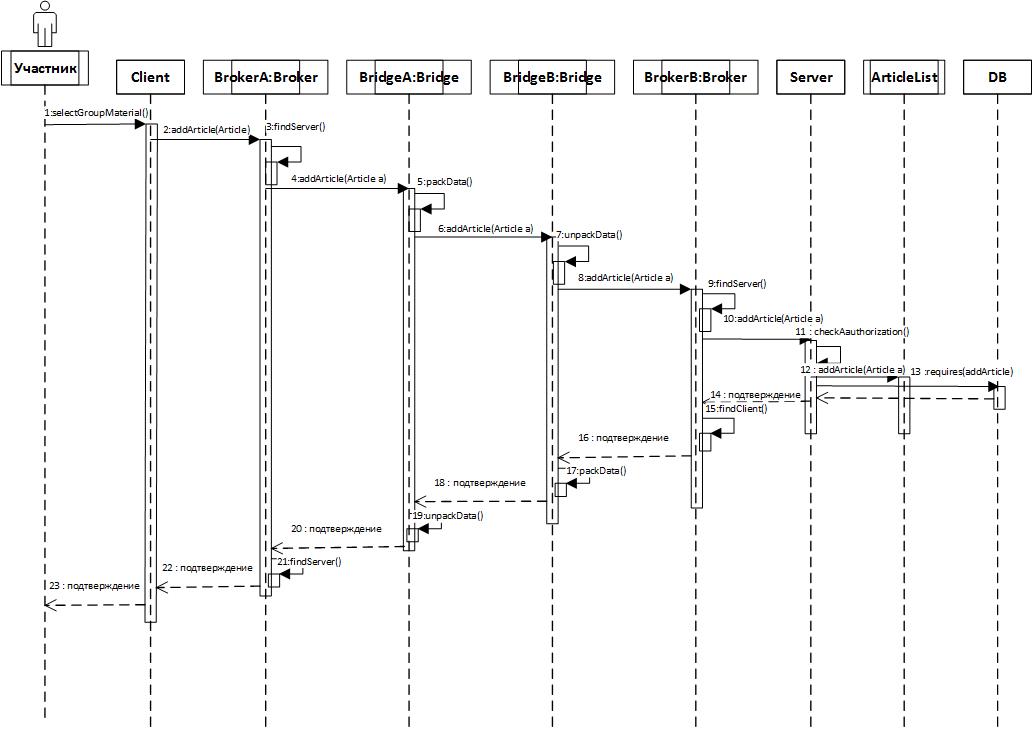


Рисунок 2.10 — Діаграма взаємодії для сценарію "Додати статтю"

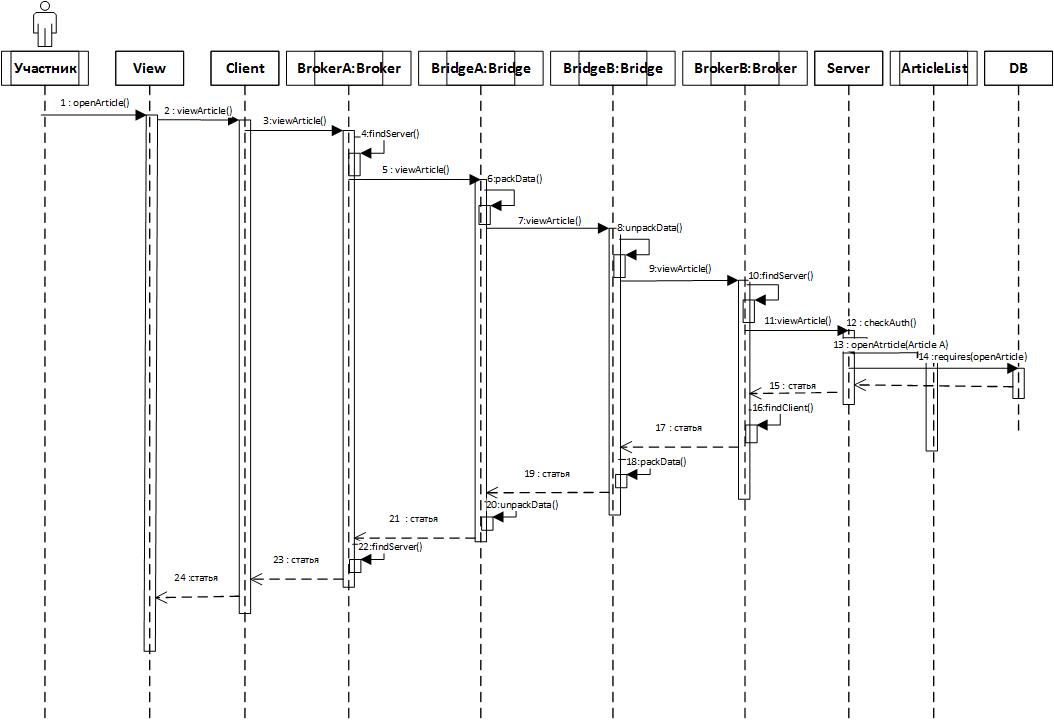


Рисунок 2.11 — Діаграма взаємодії для сценарію " Переглянути статтю "

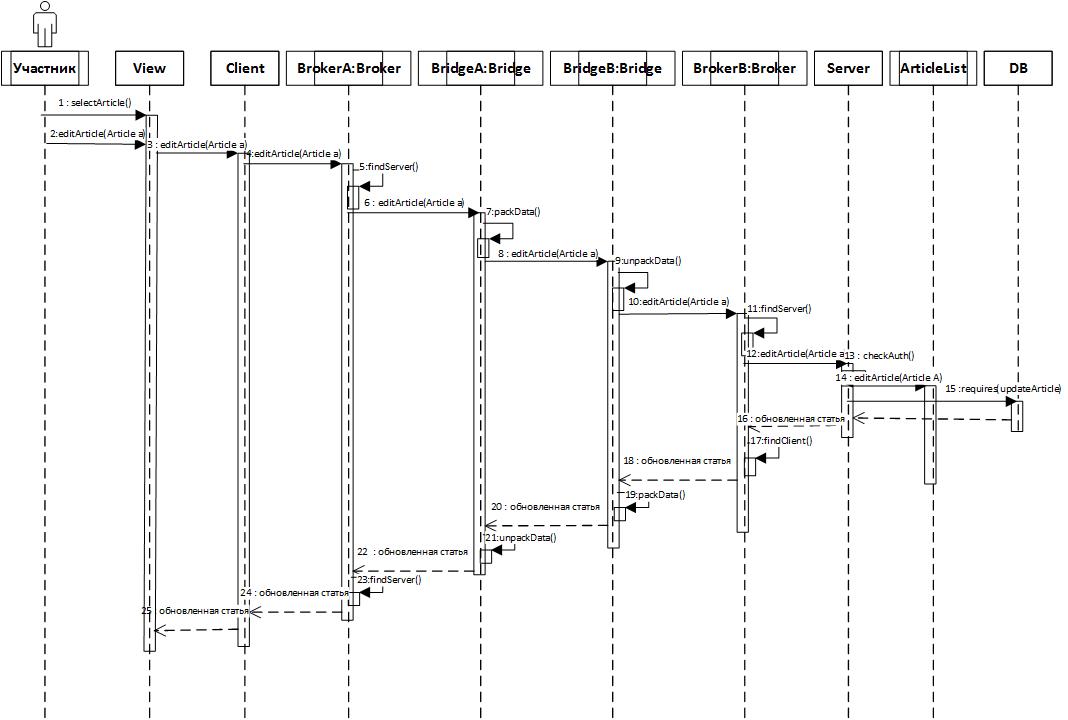


Рисунок 2.12 — Діаграма взаємодії для сценарію "Редагувати статтю"

На рис. 2.13 надано діаграму, що описує логічне подання архітектури програмної системи на основі стилів.

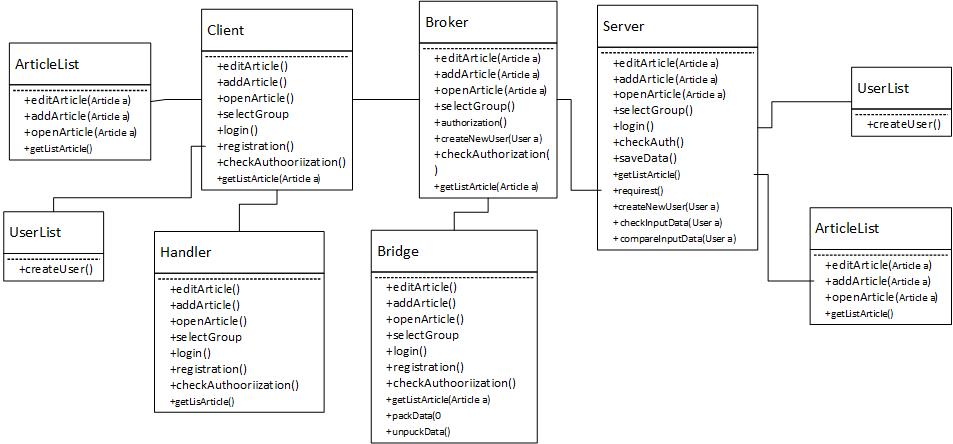


Рисунок 2.18 — Логічне подання архітектури програмної системи на основі стилів

# АНАЛІЗ БАЗОВОЇ АРХІТЕКТУРИ

## Аналіз архітектури за методом SAAM

У методі аналізу архітектури на основі сценаріїв (SAAM) відбувається оцінка складності внесення змін в архітектурний проект. Для цього визначаються додаткові функції системи, після чого оцінюється можливість реалізації цієї функції на розробленій архітектурі, а також кількість різних за рівнем складності змін, які необхідно внести.

Складність змін визначається за допомогою встановлення коефіцієнтів складності для різних типів змін: складність внесення нового компоненту або зв'язку між компонентами оцінюється коефіцієнтом 2, а внесення нового інтерфейсного методу — коефіцієнтом 1.

На першому етапі виконання аналізу за методом SAAM було визначено сценарій, який може бути новим для проектованої системи — "Змінити пароль та почту". Цей сценарій може знадобитися у випадку, коли користувач має бажання сформувати зображення не зі всього набору вхідних даних, а тільки з тих, що мають певні властивості. Нижче надано опис сценарію.

Назва: Зміна пароля

1) Клієнт заходить в вікно редагування профілю

2) Клієнт уводить старий пароль, і також новий пароль

3) Клієнт натискає змінити пароль. Система перевіряє новий пароль і підтверджує його зміна.

a. Система надає відмову. Повідомлення про помилку, вихід.

4) Клієнт виходить з модуля зміни пароля

Для внесення сценарію "Змінити пароль" в архітектурний проект на основі сценаріїв (рис. 2.9) необхідно додати ще два методи. Нові методи отримують назву changePass(),UpdateUserData().

Після визначення змін, що мають бути реалізовані, можна підрахувати трудомісткість внесення сценарію "Змінити пароль" в архітектурний проект на основі сценаріїв. Кількість нових нових методів — 2. Враховуючи коефіцієнти складності трудомісткість буде дорівнювати: 1+1 = 2.

Для внесення сценарію "Змінити пароль " в архітектурний проект з використанням стилів (рис. 2.18) необхідно додати лише 2 методи: у компоненти Bridge I Broker.

Після визначення змін, що мають бути реалізовані, можна підрахувати трудомісткість внесення сценарію "Змінити пароль" в архітектурний проект з використанням стилів. Кількість нових компонентів — 0, нових зв'язків — 0 та нових методів — 2. Враховуючи коефіцієнти складності трудомісткість буде дорівнювати: 1+1 = 2.

Спираючись на отримані оцінки можна зробити висновок, що архітектура, виконана з використанням архітектурних стилів, підтримує внесення змін, додавання нових сценаріїв тощо.

Доопрацювання кращої, згідно з результатами аналізу, архітектури надано в додатку A.

## Аналіз архітектури за методом ATAM

У методі аналізу компромісних архітектурних рішень відбувається оцінка розробленої архітектури. Визначається, чи підтримує вона все необхідні вимоги до системи, чи існують ризики для використаних архітектурних підходів, а також точки чутливості та компромісу.

На першому етапі виконання аналізу будується дерево корисності для всіх визначених у специфікації вимог сценаріїв якості. Побудова дерева також включає пріоритизацію сценаріїв. Кожен сценарій оцінюється з використанням відносних показників таких як високий / середній / низький (H/M/L) за такими властивостями як важливість для замовника і складність реалізації.

Таким чином було виявлено чотири сценарії з показниками пріоритетів вище за L. Результати аналізу цих сценаріїв надано в додатку Б.

# ОПИС АРХІТЕКТУРНОГО ПРОЕКТУ

Для опису архітектурного проекту використовуються логічне та процесне подання. Логічне подання — це представлення статичної організації архітектури, яке демонструє компоненти системи, те, як вони пов'язані між собою та які мають інтерфейси. Процесне подання — це представлення динамічної організації архітектури, яке демонструє взаємодію компонентів між собою у часі.

Процесне та логічне подання взаємопов'язані. Взаємодія компонентів у процесному поданні відбувається шляхом звертання компонентів один до одного через їх інтерфейси, позначені у логічному поданні.

Архітектурний проект був розроблений для системи, яка надає можливість додавати, переглядати, або ж редагувати наукові матеріали, які вибрані користувачем а також представлені на екрані.

Система являється клієнт-сервірною, тому при проектуванні архітектури головним архітектурнім стилем було обрано Брокер, а також MVC, тому що система пов’язана с користувачем.

## Документування подань

Первину презентацію елементів та зв'язків між ними у процесному та логічному поданнях було зображено на рис. 2.10 — 2.18.

За шаблоном Broker архітектура була поділена на наступні компоненти:

* Client
* Server
* Broker
* Bridge
* ArticleList
* UserList

Для цієї системи був обрано саме стиль Брокер, тому що – він найкраще підходить до систем яка взаємодіє з різними компонентами.

Саме «Брокер», призначений для розподілених систем, з скількох незалежних, але пов'язаних між собою компонентів (наприклад, клієнт-сервер). Шаблон дозволяє реалізувати з'єднувати ці компоненти. Наша система, є клині-серверним додатком, а значить що постійно потрібно застосування цього шаблону в кожному сценарії.

Сам шаблон являє собою 6 компонентів: Сервер, клієнт, Брокер, Проксі (сторона клієнта), проксі (сторона сервера), і сам міст.

## Документування інтерфейсів

У даному розділі надається опис інтерфейсів компонентів розробленої архітектури.

До інтерфейсних методів компоненту Client належать:

* addArticle() - інтерфейс, який забезпечую додавання нового матеріалу, за допомоги сповіщення компоненту Server;
* editArticle() - інтерфейс, редагування матеріалу, за допомоги сповіщення компоненту Server.
* openArticle() – відкриття статті, за допомоги сповіщення компоненту Server
* selectGroup(), вибір тематики нашого матеріалу, за допомоги сповіщення компоненту Server
* login() – інтерфейс авторизаціі, за допомоги сповіщення компоненту Server
* registration() інтерфейс регистрації, за допомоги сповіщення компоненту Server
* updateUserData() – оновлення даних користуваяа, за допомоги сповіщення компоненту Server

До інтерфейсних методів компоненту Server належать:

* addArticle() - інтерфейс, який забезпечую додавання нового матеріалу
* editArticle() - інтерфейс, редагування матеріалу
* openArticle() – відкриття статті
* selectGroup(), вибір тематики нашого матеріалу
* login() – інтерфейс авторизаціі
* registration() інтерфейс регистрації
* updateUserData() – оновлення даних користуваяа.

До інтерфейсних методів компоненту ArticleList належать:

* editArticle() - інтерфейс, редагування статті;
* AddArticle() - інтерфейс, додавання статті;
* openArticle() - інтерфейс, відкриття статті;

До інтерфейсних методів компоненту UserList належать:

* changePas() - інтерфейс, зміну пароля у системі;
* createUser() - інтерфейс, котрий створює нового користувача у системі;

До інтерфейсних методів компоненту Bridge I Broker належать усі методи описані вище.

## 

## ВИСНОВОК

В даній курсовій роботі було створено та порівняно два архітектурних проекти для системи формування та порівняння зображень.

Була виконана специфікація вимог до програмної системи, виявлені атрибути якості та визначені найвпливовіші з них.

Був створений архітектурний проект спираючись на сценарії варіантів використання.

Також був сформований другий архітектурний проект з використанням архітектурних стилів, тактик та шаблонів.

За результатами аналізу архітектурних проектів за методом SAAM (метод аналізу архітектури на основі сценаріїв) було визначено, що архітектура, виконана з використанням архітектурних стилів, краще підтримує внесення змін, додавання нових сценаріїв тощо.

За результатами архітектурних проектів за методом ATAM (метод аналізу компромісних архітектурних рішень) були визначені точки чутливості та не-ризики архітектурного рішення.

Також було сформовано опис архітектурного проекту.

# СПИСОК ЛІТЕРАТУРИ

1. Басс Л., Клементс П., Кацман Р. Архитектура программного обеспечения на практике. – СПб.: Питер, 2006. – 576 с..

# ДОДАТОК А. РЕЗУЛЬТАТИ АНАЛІЗУ АРХІТЕКТУРИ ЗА МЕТОДОМ SAAM

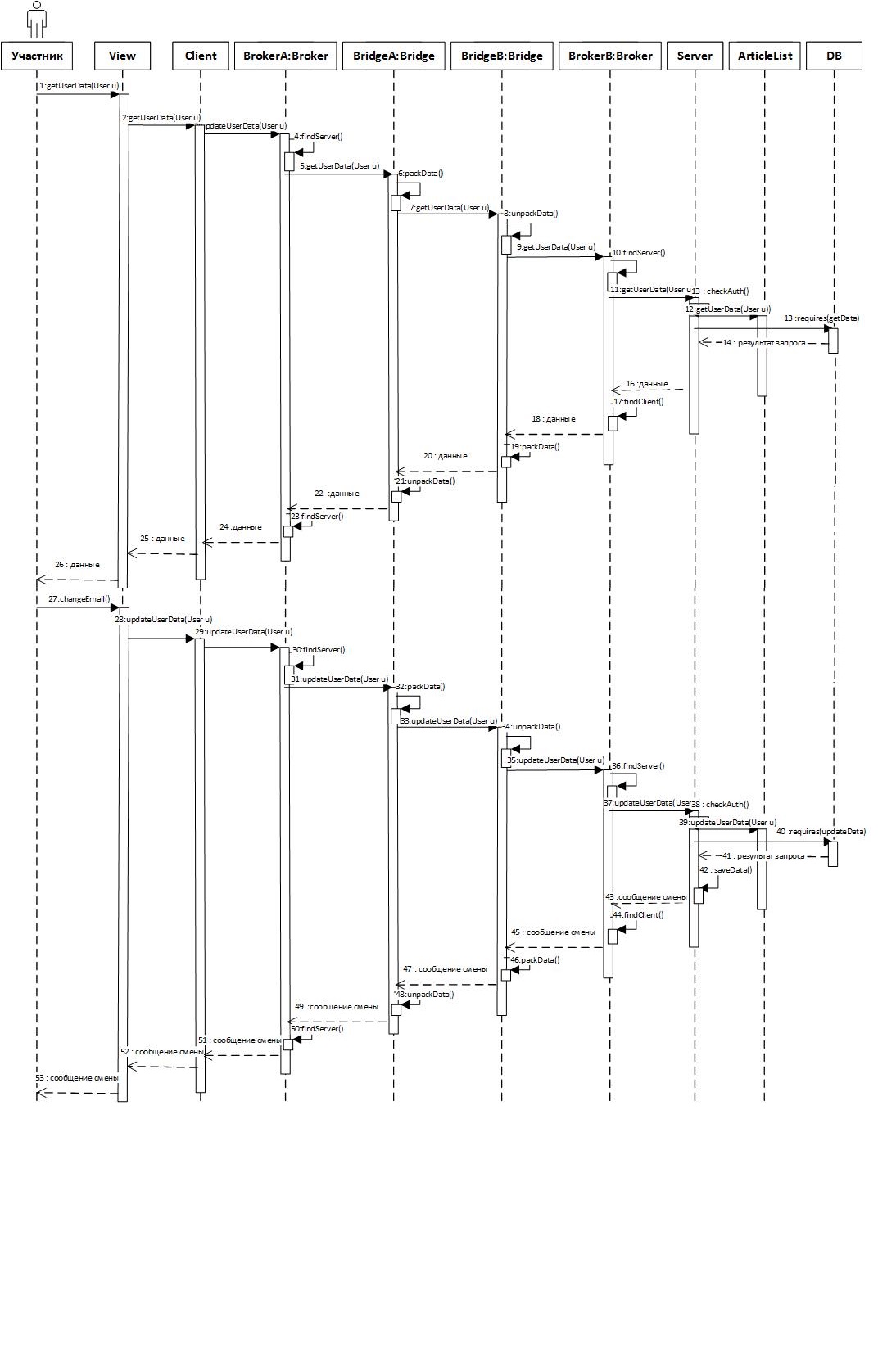


Рисунок A.1 — Діаграма взаємодії для сценарію " Змінити пароль"

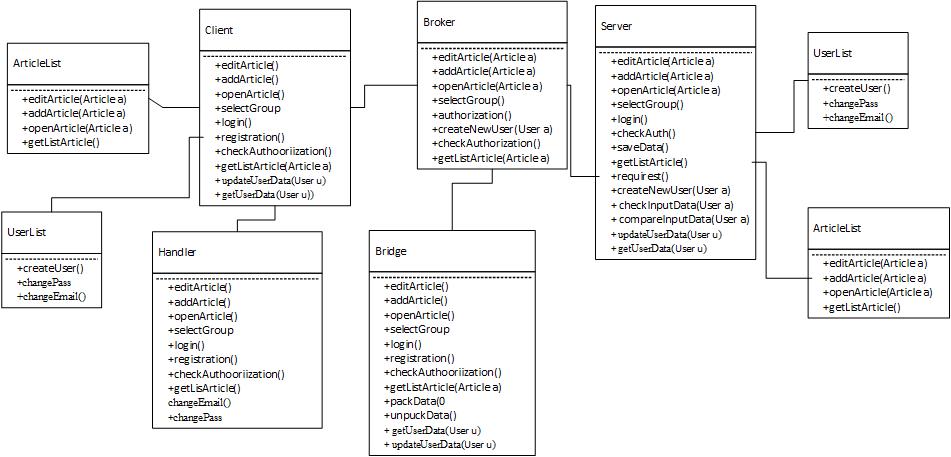


Рисунок A.2 — Доопрацьоване логічне подання архітектури

# ДОДАТОК Б. РЕЗУЛЬТАТИ АНАЛІЗУ АРХІТЕКТУРИ ЗА МЕТОДОМ ATAM

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | | | |
| Сценарий # 1 | Время выполнения запроса к серверу не должно превышать 5 секунд | | | | |
| Атрибут(и) | Производительность | | | | |
| Среда | Нормальный режим | | | | |
| Влияние | Пользователь выполняет запрос | | | | |
| Реакция | Система выполняет его менее чем за 5 секунд | | | | |
| Архитектурные решения | Чувствительность | | Компромисс | Риск | Не-риск |
| *Введение параллелизма* | + | |  | + |  |
| *Сокращение количества событий, которые обрабатываются* |  | |  |  | + |
| Аргументация | | Введение параллелизма сложный и трудоемкий процесс. В связи с тем, что расчетов в системе практически нет (лишь получение и передача данных, так же их запись) – параллелизм особо выигрыша не принесет, потому логичнее ограничить количество поступающих запросов. | | | |
| Архитектурная диаграмма | |  | | | |

Рисунок Б.1 — Аналіз архітектурних рішень для сценарію А1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сценарий # 2 | Обработка данных на сервере не должна превышать 10 мс. | | | | |
| Атрибут(и) | Производительность | | | | |
| Среда | Нормальный режим | | | | |
| Влияние | Пользователь выполняет запрос | | | | |
| Реакция | Сервер выполняет его менее чем за 10 миллисекунд | | | | |
| Архитектурные решения | Чувствительность | | Компромисс | Риск | Не-риск |
| *Введение параллелизма* |  | |  | + |  |
| *Сокращение количества событий, которые обрабатываются* |  | |  |  | + |
| Аргументация | | Сценарий аналогичен предыдущему, но касается только сервера, где и логичнее всего будет создать очередь. | | | |
| Архитектурная диаграмма | |  | | | |

Рисунок Б.2 — Аналіз архітектурних рішень для сценарію А2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Атрибут(и) | Использование ресурсов | | | | |
| Среда | Нормальный режим | | | | |
| Влияние | Занято ОП>512 | | | | |
| Реакция | Очистить память | | | | |
| Архитектурные решения | Чувствительность | | Компромисс | Риск | Не-риск |
| *Увеличение ресурсов* | + | | + |  | + |
| Аргументация | | Если в системе возникнут проблемы с перегрузкой ОП, она очистит все не нужные на данный момент данные. | | | |
| Архитектурная диаграмма | | - | | | |
| Сценарий #4 | | При отправке некорректных запросов, сервер должен восстановить работу с вероятность не менее 0,95. | | | |
| Атрибут(и) | | Устойчивость к отказам | | | |
| Среда | | Нормальный режим | | | |
| Влияние | | Отправка с клиента на сервер запроса, содержащего ошибку | | | |
| Реакция | | Сервер сообщает об ошибке клиенту и продолжает работу. | | | |
| Архитектурные решения | Чувствительность | | Компромисс | Риск | Не-риск |
| *Транзакции* |  | |  |  | + |
| *Исключение* |  | |  |  | + |
| Аргументация | | Используем обе тактики. Если запрос определен как некорректный, сервер генерирует ошибку и отправляет ее на клиент. При какой-либо ошибке ничего выполнено не будет, т.к. выполняется либо все, либо ничего для сохранения целостности данных. | | | |
| Архитектурная диаграмма | |  | | | |

Рисунок Б.3 — Аналіз архітектурних рішень для сценарію А3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сценарий # 4 | При получении некорректного ответа, клиент должен восстановить работу с вероятность не менее 0,5 | | | | |
| Атрибут(и) | Устойчивость к отказам | | | | |
| Среда | Нормальный режим | | | | |
| Влияние | Клиент выполняет запрос на сервер и получает ответ с ошибкой | | | | |
| Реакция | Клиент выводит сообщение об ошибке пользователю и продолжает работу | | | | |
| Архитектурные решения | Чувствительность | | Компромисс | Риск | Не-риск |
| *Исключение* |  | |  |  | + |
| *Сердцебиение* | + | |  |  | + |
| Аргументация | | Выбираем исключение. Сервер не должен знать о неисправностях клиентов, т.к. обработкой ошибок он не занимается. Если возникает ошибка – клиент сообщает о ней и выполняет повторный запрос данных у сервера. | | | |
| Архитектурная диаграмма | |  | | | |

Рисунок Б.4 — Аналіз архітектурних рішень для сценарію А4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сценарий # 5 | При потере соединения с сервером вероятность восстановления повторного соединения составляет не менее 0,5. | | | | |
| Атрибут(и) | Возможность к восстановлению | | | | |
| Среда | Нормальный режим | | | | |
| Влияние | Разрыв соединения между сервером и клиентом | | | | |
| Реакция | Клиент пытается переподключится к серверу | | | | |
| Архитектурные решения | Чувствительность | | Компромисс | Риск | Не-риск |
| *Исключение* |  | |  |  | + |
| Аргументация | | При возникновении неисправности генерируем ошибку, в обработке ее пытаемся переподключиться к серверу | | | |
| Архитектурная диаграмма | |  | | | |

Рисунок Б.5 — Аналіз архітектурних рішень для сценарію А4