Національний технічний університет України

«Київський політехнічний інститут»

Кафедра обчислювальної техніки

Проект

«Деталізована мапа КПІ»

з дисципліни **«Організація баз даних»**

Виконали:

студенти 2 курсу

ФІОТ гр. ІО-34

Бригада 2

Власов М.Д.

Кривоносов О.О.

Куриленко О.О.

Мозговий І.В.

Шпартько О.В.

Запити зацікавлених осіб

Вступ

1. Мета
2. Короткий огляд продукту
3. Контекст
4. Ділові правила та приписи

4.1. Призначення електронної деталізованої мапи

4.2. Політика взаємодії з користувачем

4.3. Сценарії

4.3.1. Пропозиція модератору, щодо додавання нового об'єкту

1. Функціональність

5.1. Можливості користувача

5.2. Можливості модератора

5.3. Можливості адміністратора/розробника

1. Практичність

6.1. Масштабованість інтерфейсу

6.2. Інтерфейс користувача

1. Надійність

**Вступ**

У цьому документі описуються запити зацікавлених осіб по відношенню до розроблювальної системи "Електронна деталізована мапа КПІ" в якості яких виступають: замовник - будь-яка фізична або юридична особа, що зацікавлені у розробці даної карти, і користувачі системи.

**1. Мета**

Метою документа є визначення основних вимог до функціональності, продуктивності, експлуатаційної придатності, а також визначення бізнес-правил і технологічних обмежень, пред'явлених предмету розробки.

**2. Короткий огляд продукту**

Система "Електронна деталізована мапа КПІ" складається з двох розділів програмного забезпечення:

1. Централізована база даних на головному сервері
2. Локальний програмний продукт на боці користувача, який міститиме в собі базу даних, оновлюватиме її з бази даних на сервері і пропонуватиме модератору додати нові записи у базу у відкладеному режимі, у разі відсутності інтернет-з'єднання на момент готовності запису.

"Електронна деталізована мапа КПІ" є електронною консистенцією детального покажчика і звичайної мапи, що на порядок збільшує зручність і швидкість знаходження потрібного об'єкту в межах території КПІ.

**3. Контекст**

Перелік вимог, перерахованих у цьому документі, є основою технічного завдання на розробку системи "Електронна деталізована мапа КПІ"

**4. Ділові правила та приписи**

**4.1. Призначення електронної деталізованої мапи**

* + Ця система буде цікава всім без вийнятку зрізам суспільства, що хоч якось дотикаються до КПІ: студентам, викладачам, абітурієнтам, батькам абітурієнтів, студентам інших ВНЗ, які оберуть КПІ, як територію свого культурного дозвілля.
  + Ця система дозволить швидко орієнтуватись на просторах КПІ, в залежності від потреб тієї чи іншої людини (чи то дешева та смачна їжа, чи то пошук аудиторії, чи то друк паперів, чи то перукарня, чи пральня)

**4.2. Політика взаємодії з користувачем**

* + Користувач за допомогою Google Store завантажує .apk-файл, погоджується з умовами встановлення і за декілька секунд клієнтська частина додатку буде встановлена
  + Користувач, отримуючи додаток, відразу ж отримує доступ до об'єктів, що вже були позначенні адміністрацією проекту чи користувачами, які раніше встановили цей додаток
  + Для того, щоб додати об'єкт на мапу, користувачеві потрібно натиснути на вибрану для додавання ділянку до появи контекстного меню, вибрати пункт "Додати об'єкт" та заповнити форму. Опісля об'єкт відправляється на модерацію. Якщо модерація пройшла вдало, то цей об'єкт додається до системи і з’являється у інших користувачів
  + У формі "Додати об'єкт" користувач може прикріпити опис, фото та теґи
  + За результатом модерації, користувачу надходить повідомлення про додавання чи не додавання запропонованого об'єкту

**4.3. Сценарії**

**4.3.1. Пропозиція модератору, щодо додавання нового об'єкту**

**Учасники:**

* Користувач
* Модератор

**Передумови:**

* Користувач має на своєму пристрої клієнт програми
* Користувач ідентифікований за допомогою облікового запису Google

**Результат:**

* Новий об'єкт на мапі

**Основний сценарій:**

* 1. Користувач затискає місце на мапі, вибирає пункт меню "Додати об'єкт", заповнює форму. Програма відсилає данні на сервер для обробки
  2. Модератор ознайомлюється з пропозицією і додає об'єкт до бази даних
  3. Користувачу виводиться повідомлення, про успішне додавання об’єкту

**Виключні ситуації:**

* У разі нікчемності чи образливості наданої користувачем інформації, модератор не додає об'єкт до бази даних

**5. Функціональність**

Основні вимоги до функціональності, пред'явлені зацікавленими особами, відносяться до трьох категорій:

* 1. Адміністратор
  2. Модератор
  3. Користувач

**5.1. Можливості користувача**

* Перегляд мапи з вибраними умовними позначками
* Перегляд поповерхової план-схеми корпусів
* Перегляд і додавання локальних об'єктів, що не синхронізуються з сервером (улюблені місця)
* Пошук корпусу по назві факультету, його номеру, адресі будинку
* Перегляд відкритих Wi-Fi мереж і їх радіус дії
* Перегляд відгуків про об'єкт (доступно через інтернет)
  + Пропозиція модератору, щодо надання нового об'єкту на мапу
  + Додавання відгуку про об'єкт

**5.2. Можливості модератора**

* + Додавання нових об'єктів, в тому числі, на основі пропозицій користувачів
  + Редагування і видалення об'єктів

**5.3. Можливості адміністратора**

* + Призначення/видалення модераторів
  + Додавання нових об'єктів, в тому числі, на основі пропозицій користувачів
  + Редагування і видалення об'єктів

**6. Практичність**

**6.1. Масштабованість інтерфейсу**

Інтерфейс має змінювати свої розміри, відносно до роздільної здатності екрану пристрою

**6.2. Інтерфейс користувача**

Інтерфейс користувача має відповідати наступним вимогам:

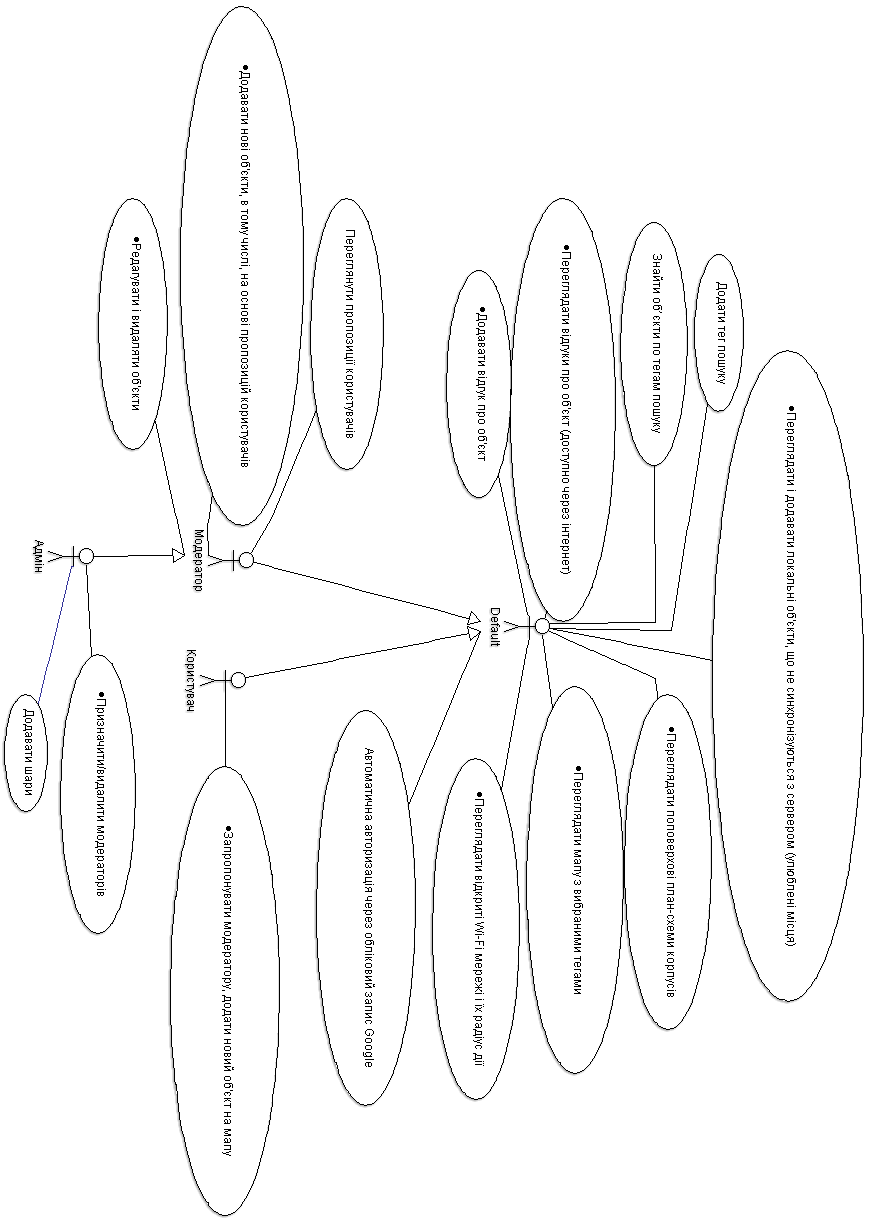
* 1. Бути зрозумілим і не мати надлишкових функцій
  2. Бути виконаним з урахуванням енергомічних вимог
  3. Мати вхідний гайд по основним функціям, для швидкого ознайомлення користувача з додатком

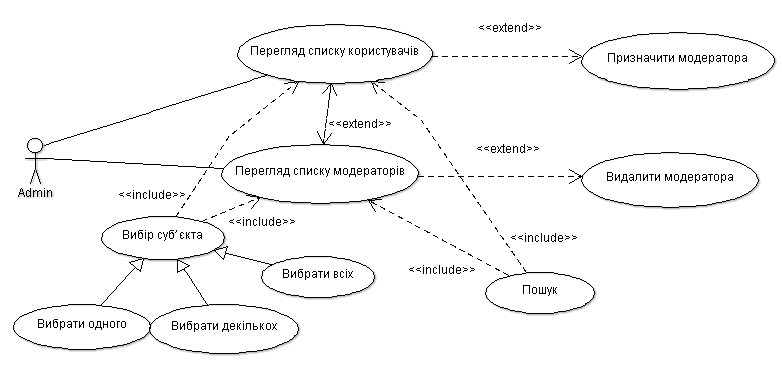
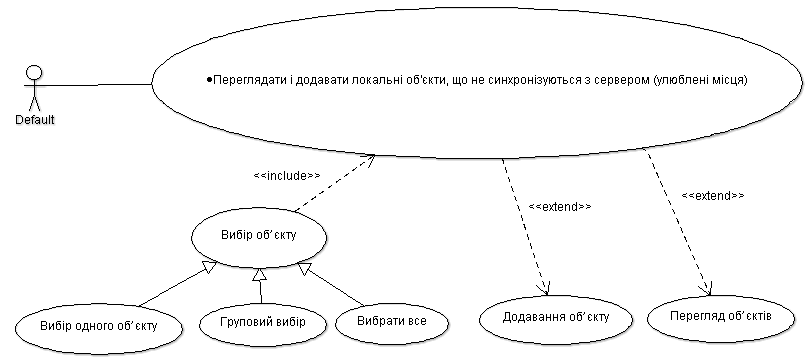
**7. Надійність**

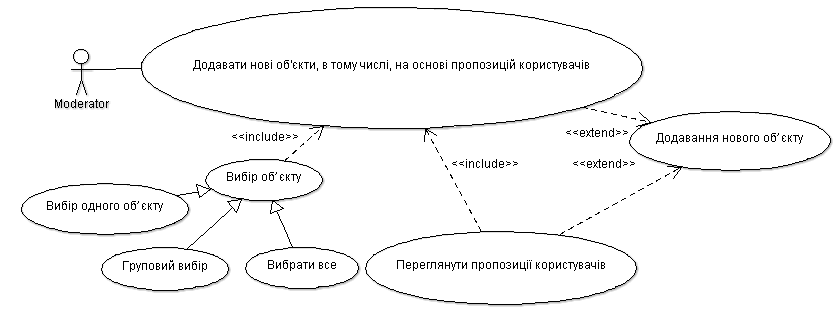
Протягом усього терміну підтримки додатку, повинна бути забезпечена цілісність, незмінність ззовні і достовірність внесених даних

Для забезпечення збереження та цілісності використовуватиметься метод резервного копіювання.

Для збереження незмінності та достовірності використовуватиметься комплекс технологічних і адміністративних процедур, що перешкоджають випадковій або навмисній зміні даних ззовні.







**ID: AD.M01**

**Назва:** Додавання об’єкту

**Учасники:** Модератор, система

**Передумови:** Користувач авторизований, користувач належить до групи «модератори», чи груп, що наслідують групу «модератори»

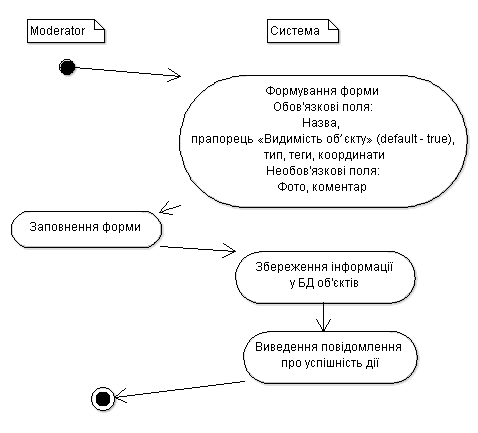
**Результат:** Створення і заповнення строки у БД об’єктів

**Основний сценарій:**

1. Модератор відправляє запит на додавання об’єкту
2. Система підготовлює форму з наступними полями:
   1. Назва об’єкту (обов’язкове)
   2. Прапорець «Видимість об’єкту» (by default = true)
   3. Тип об’єкту (обов’язкове)
   4. Теги (обов’язкове)
   5. Координати (обов’язкове)
   6. Фото
   7. Коментар
3. Модератор заповнює форму
4. Система зберігає данні у БД
5. Система виводить повідомлення про успішне додавання об’єкту

**Виключні ситуації:**

* Система тимчасово не доступна
* Незаповнені обов’язкові поля

****

**ID: AD.M02**

**Назва:** Редагування об’єкту

**Учасники:** Модератор, система

**Передумови:** Користувач авторизований, користувач належить до групи «модератори», чи груп, що наслідують групу «модератори»

**Результат:** Заміна даних у БД об’єктів

**Основний сценарій:**

1. Модератор відправляє запит на редагування об’єкту
2. Система підготовлює форму з наступними полями:
   1. Назва об’єкту (обов’язкове)
   2. Прапорець «Видимість об’єкту»
   3. Тип об’єкту (обов’язкове)
   4. Теги (обов’язкове)
   5. Координати (обов’язкове)
   6. Фото
   7. Коментар
3. Система заповнює форму з БД
4. Модератор редагує данні у формі
5. Система замінює данні у БД
6. Система виводить повідомлення про успішну заміну даних у БД

**Виключні ситуації:**

* Система тимчасово не доступна
* Незаповнені обов’язкові поля

**ID: AD.A01**

**Назва:** Видалення модератора

**Учасники:** Адміністратор, система

**Передумови:** Користувач авторизований, користувач належить до групи «адміністратори», чи груп, що наслідують групу «адміністратори»

**Результат:** Перенесення користувача з групи «модератори» в групу «користувачі»

**Основний сценарій:**

1. Адміністратор вибирає модератора, клацає на кнопку «видалити модератора»
2. Система переносить користувача з групи «модератори» до групи «користувачі»
3. Система виводить повідомлення про успішне перенесення модератора у групу «користувачі»

**Виключні ситуації:**

* Система тимчасово не доступна



**ID: AD.A02**

**Назва:** Призначення модератора

**Учасники:** Адміністратор, система

**Передумови:** Користувач авторизований, користувач належить до групи «адміністратори», чи груп, що наслідують групу «адміністратори»

**Результат:** Перенесення користувача з групи «користувачі» в групу «модератори»

**Основний сценарій:**

1. Адміністратор вибирає модератора, клацає на кнопку «призначити модератором»
2. Система переносить користувача з групи «користувачі» до групи «модератори»
3. Система виводить повідомлення про успішне перенесення користувача у групу «модератори»

**Виключні ситуації:**

* Система тимчасово не доступна



**ID: AD.D01**

**Назва:** Додавання локального об’єкту

**Учасники:** Default, система

**Передумови:** Користувач авторизований

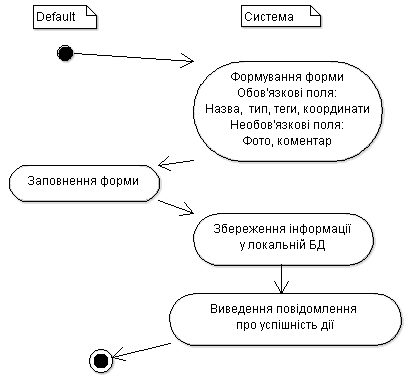
**Результат:** Створення і заповнення строки у локальній БД локальних об’єктів

**Основний сценарій:**

1. Користувач відправляє запит на додавання об’єкту
2. Система підготовлює форму з наступними полями:
   1. Назва об’єкту (обов’язкове)
   2. Тип об’єкту (обов’язкове)
   3. Теги (обов’язкове)
   4. Координати (обов’язкове)
   5. Фото
   6. Коментар
3. Користувач заповнює форму
4. Система зберігає данні у локальній БД
5. Система виводить повідомлення про успішне додавання локального об’єкту

**Виключні ситуації:**

* Незаповнені обов’язкові поля



**ID: AD.D02**

**Назва:** Додавання відгуку про об’єкт

**Учасники:** Default, система

**Передумови:** Користувач авторизований

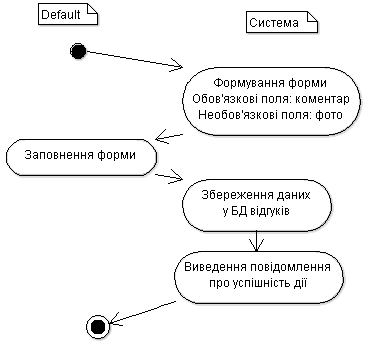
**Результат:** Створення і заповнення строки у БД відгуків до цього об’єкту

**Основний сценарій:**

1. Користувач відправляє запит на додавання відгуку про об’єкт
2. Система підготовлює форму з наступними полями:
   1. Коментар (обов’язкове)
   2. Фото
3. Користувач заповнює форму
4. Система зберігає данні у БД відгуків до цього об’єкту
5. Система виводить повідомлення про успішне додавання об’єкту

**Виключні ситуації:**

* Система тимчасово не доступна
* Незаповнені обов’язкові поля



**ID: AD.K01**

**Назва:** Пропозиція користувача, щодо додавання нового об’єкту

**Учасники:** Користувач, система

**Передумови:** Користувач авторизований, користувач належить до групи «користувачі»,

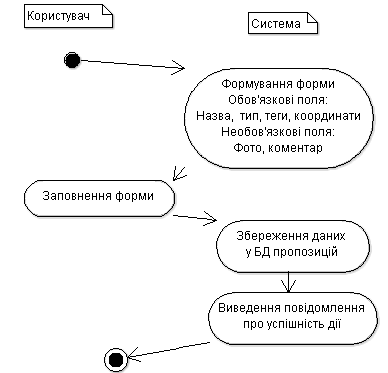
**Результат:** Створення і заповнення строки у БД пропозицій об’єктів

**Основний сценарій:**

1. Користувач відправляє запит на додавання об’єкту
2. Система підготовлює форму з наступними полями:
   1. Назва об’єкту (обов’язкове)
   2. Тип об’єкту (обов’язкове)
   3. Теги (обов’язкове)
   4. Координати (обов’язкове)
   5. Фото
   6. Коментар
3. Користувач заповнює форму
4. Система зберігає данні у БД пропозицій об’єктів
5. Система виводить повідомлення про успішне відправлення об’єкту на премодерацію

**Виключні ситуації:**

* Система тимчасово не доступна
* Незаповнені обов’язкові поля



**ID: AD.D03**

**Назва:** Додати тег пошуку

**Учасники:** Default, система

**Передумови:** Користувач авторизований

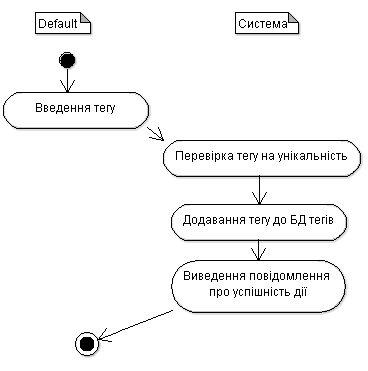
**Результат:** Додавання строки в БД тегів

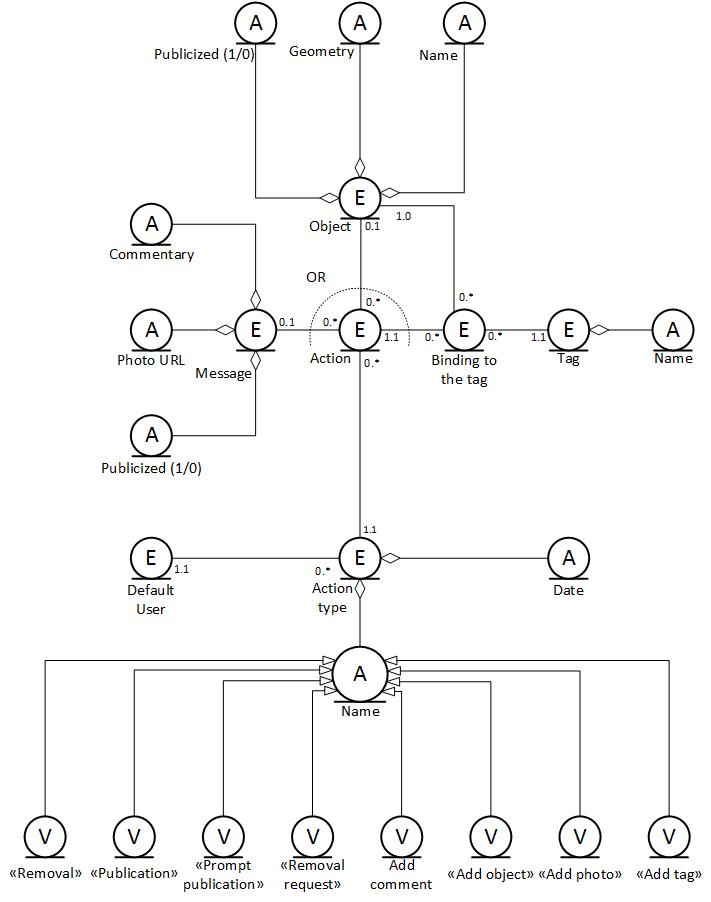
**Основний сценарій:**

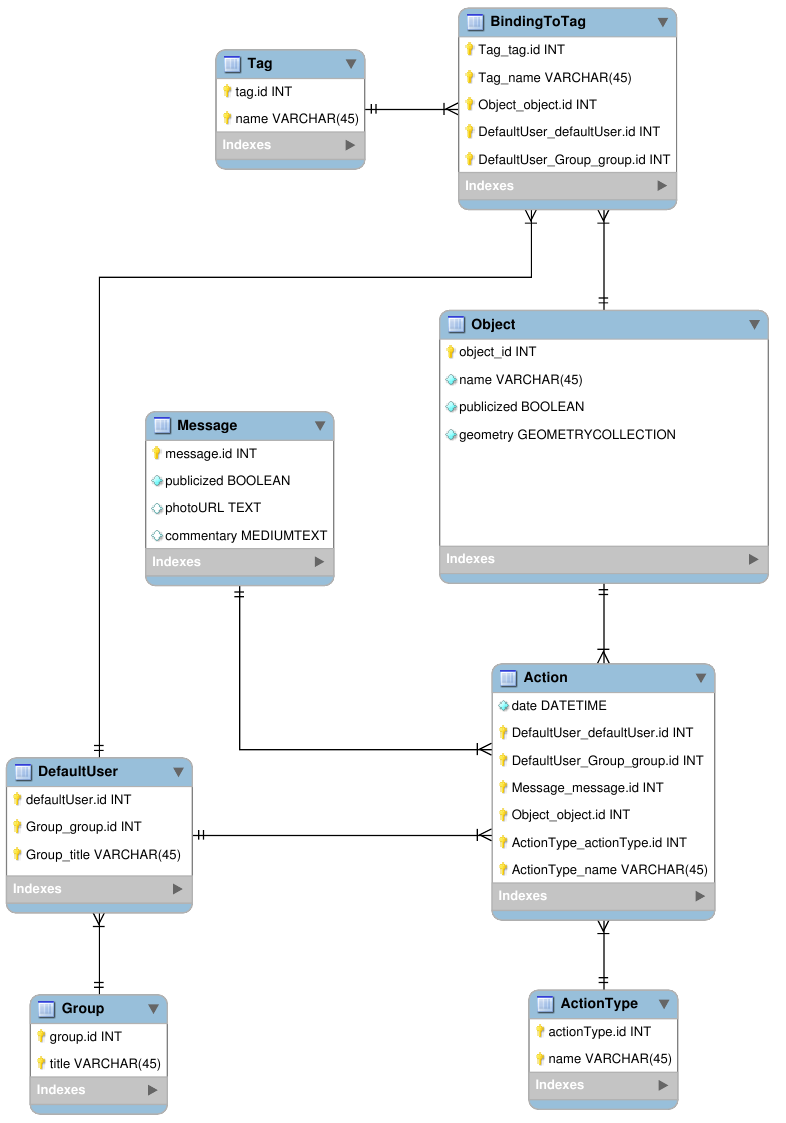
1. Користувач відправляє запит на додавання тегу
2. Система перевіряє тег на унікальність
3. Система виводить повідомлення про успішне створення нового тегу

**Виключні ситуації:**

* Система тимчасово не доступна
* Тег неунікальний







**Java реалізація**

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **interface** IActionType {

**void** setActionId(**int** id);

**void** setActionName(String name);

}

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **interface** IBindingToTag **extends** IObject, IDefaultUser, ITag{

}

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **interface** IDefaultUser **extends** IGroup {

**void** setDefaultUserId(**int** id);

}

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **interface** IGroup {

**void** setGroupId(**int** id);

**void** setGroupTitle(String title);

}

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**import** java.text.DateFormat;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **interface** IAction **extends** IObject, IMessage, IActionType, IDefaultUser {

DateFormat ***date*** = **new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

**public** **default** DateFormat getDate() {

**return** ***date***;

}

}

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **interface** IMessage {

**void** setMessageId(**int** id);

**void** setMessagePublicized(**boolean** publicized);

**void** setMessagePhotoURL(String photoURL);

**void** setMessageCommentary(String commentary);

}

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**import** org.json.\*;

**public** **interface** IObject {

**void** setObjectId(**int** id);

**void** setObjectName(String name);

**void** setObjectPublicized(**boolean** publicized);

**void** setObjectGeometry(Object geometry);

}

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **interface** ITag {

**void** setTagId(**int** id);

**void** setTagName(String name);

}

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **class** Object **extends** ObjectDao **implements** IObject {

**private** **int** id;

**private** String name;

**private** **boolean** publicized;

**private** Object geometry;

/\*\*

\* Constructor for Geometry

\* **@version** 0.1

\* **@param** string

\*/

**public** Object(String string) {

String Object = string;

}

/\*\*

\* Constructor for create Object in class 'Run'

\*/

**public** Object() {

}

@Override

**public** **void** setObjectId(**int** id) {

**this**.id = id;

}

@Override

**public** **void** setObjectName(String name) {

**this**.name = name;

}

@Override

**public** **void** setObjectPublicized(**boolean** publicized) {

**this**.publicized = publicized;

}

@Override

**public** **void** setObjectGeometry(Object geometry) {

**this**.geometry = geometry;

}

**public** **int** getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** **boolean** getPublicized() {

**return** publicized;

}

**public** Object getGeometry() {

**return** geometry;

}

}

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.PreparedStatement;

**import** java.util.ArrayList;

**import** java.util.List;

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **class** ObjectDao {

// Need to connect to the database and CRUD-operations

**public** **static** String *url* = "jdbc:mysql://127.0.0.1:3306/Map\_of\_KPI";

**public** **static** String *name* = "root";

**public** **static** String *password* = "1111";

**public** **static** Connection *con* = *ConnectToDB*(*url*, *name*, *password*); // Are you

// Con,

// Bill?

// (:

/\*\*

\* CRUD: Creating an object in the database

\*

\* **@param** obj

\* **@return** obj

\* **@throws** SQLException

\*/

**public** **static** Object create(Object obj) **throws** SQLException {

String sql = "INSERT INTO Object (object\_id, name, publicized, geometry) VALUES (?,?,?,?);";

**try** (PreparedStatement ps = *con*.prepareStatement(sql)) {

/\*

\* This block is required in order to avoid errors when entering the

\* same values in the "ps" and "obj"

\*/

**int** tempId = 1;

String tempName = "Obj1";

**boolean** tempPublicized = **false**;

Object tempGeometry = **new** Object("125.6, 10.1");

ps.setInt(1, tempId);

ps.setString(2, tempName);

ps.setBoolean(3, tempPublicized);

ps.setObject(4, "some coordinates");

ps.executeUpdate();

obj.setObjectId(tempId);

obj.setObjectName(tempName);

obj.setObjectPublicized(tempPublicized);

obj.setObjectGeometry(tempGeometry);

}

**return** obj;

}

/\*\*

\* CRUD: Reading an object in the database

\*

\* **@param** obj

\* **@param** id

\* **@return** obj

\* **@throws** SQLException

\*/

**public** **static** Object read(Object obj, **int** id) **throws** SQLException {

String sql = "SELECT \* FROM Object WHERE object\_id = ?;";

**try** (PreparedStatement ps = *con*.prepareStatement(sql)) {

ps.setInt(1, id);

ResultSet rs = ps.executeQuery();

rs.next();

obj.setObjectId(rs.getInt("object\_id"));

obj.setObjectName(rs.getString("name"));

obj.setObjectPublicized(rs.getBoolean("publicized"));

obj.setObjectGeometry((Object) rs.getObject("geometry"));

}

System.***out***.println("Object id: " + obj.getId());

System.***out***.println("Name: " + obj.getName());

System.***out***.println("Publicized: " + obj.getPublicized());

System.***out***.println("Geometry: " + obj.getGeometry());

**return** obj;

}

/\*\*

\* CRUD: Updating an object in the database

\*

\* **@param** obj

\* **@param** id

\* **@throws** SQLException

\*/

**public** **static** **void** update(Object obj, **int** id) **throws** SQLException {

String sql = "UPDATE Object SET name = ?, publicized = ?,"

+ " geometry = ? WHERE object\_id = ?;";

**try** (PreparedStatement ps = *con*.prepareStatement(sql)) {

/\*

\* This block is required in order to avoid errors when entering the

\* same values in the "ps" and "obj"

\*/

String tempName = "Name of object";

**boolean** tempPublicized = **true**;

Object tempGeometry = **new** Object("137.1 24.4");

ps.setString(1, tempName);

ps.setBoolean(2, tempPublicized);

ps.setObject(3, tempGeometry);

obj.setObjectName(tempName);

obj.setObjectPublicized(tempPublicized);

obj.setObjectGeometry(tempGeometry);

}

**return**;

}

/\*\*

\* CRUD: Deleting an object in the database

\*

\* **@param** obj

\* **@throws** SQLException

\*/

**public** **static** **void** delete(Object obj) **throws** SQLException {

String sql = "DELETE FROM Object WHERE object\_id = ?;";

**try** (PreparedStatement ps = *con*.prepareStatement(sql)) {

ps.setInt(1, 4);

ps.executeUpdate();

obj.setObjectPublicized(**false**);

}

**return**;

}

/\*\*

\* Displaying all objects from the database

\*

\* **@param** obj

\* **@return** list of objects

\* **@throws** SQLException

\*/

**public** **static** List<Object> readAll(Object obj) **throws** SQLException {

String sql = "SELECT \* FROM Map\_of\_KPI.Object;";

List<Object> list = **new** ArrayList<Object>();

**try** (PreparedStatement ps = *con*.prepareStatement(sql)) {

ResultSet rs = ps.executeQuery();

**while** (rs.next()) {

obj.setObjectId(rs.getInt("object\_id"));

obj.setObjectName(rs.getString("name"));

obj.setObjectPublicized(rs.getBoolean("publicized"));

obj.setObjectGeometry((Object) rs.getObject("geometry"));

list.add(obj);

System.***out***.println("\n");

System.***out***.println("Object id: " + obj.getId());

System.***out***.println("Name: " + obj.getName());

System.***out***.println("Publicized: " + obj.getPublicized());

System.***out***.println("Geometry: " + obj.getGeometry());

}

System.***out***.println("\n");

**return** list;

}

}

/\*\*

\* Connect to the database

\*

\* **@param** url

\* **@param** name

\* **@param** password

\* **@return** null

\*/

**public** **static** Connection ConnectToDB(String url, String name,

String password) {

**try** {

Class.*forName*("com.mysql.jdbc.Driver");

Connection con = DriverManager.*getConnection*(url, name, password);

System.***out***.println("Connected.");

**return** con;

} **catch** (SQLException | ClassNotFoundException e) {

e.printStackTrace();

}

**return** **null**;

}

/\*\*

\* Close connection

\*

\* **@throws** SQLException

\*/

**public** **static** **void** End() **throws** SQLException {

*con*.close();

}

}

/\*\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **class** Run **extends** Object {

**public** **static** **void** main(String[] args) **throws** Exception {

Object objectT = **new** Object();

*create*(objectT);

*readAll*(objectT);

*update*(objectT, 1);

*read*(objectT, 1);

*create*(objectT);

*read*(objectT, 2);

*delete*(objectT);

*readAll*(objectT);

*End*();

}

}