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 №2

Технології розроблення програмного забезпечення

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ

CI-сервер

Виконав:
Студент групи ІА-22
Вдовиченко А.Х.

Перевірив:
Мягкий М. Ю

Київ 2024

Тема: Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи
Мета: Проаналізувати тему, намалювати схему прецеденту, діаграму класів, розробити основні класи і структуру бази

Теоретичні відомості

Уніфікована мова моделювання (UML)

UML — це універсальна методологія візуального проектування, що використовується для створення моделей, які описують ключові аспекти програмного забезпечення. Вона забезпечує розробників засобами для аналізу, документування та візуалізації структури та поведінки системи.

Діаграма сценаріїв використання

Ця діаграма демонструє, як різні користувачі або зовнішні системи взаємодіють із програмною системою. Вона дозволяє визначити межі системи, встановити функціональні завдання та зрозуміти очікувані результати її роботи.

Основні компоненти:

- **Користувачі (актори):** будь-які суб'єкти, що взаємодіють із системою (люди, пристрої або інші системи).
- **Сценарії (випадки використання):** описують, які функції виконує система для різних акторів.
- **Зв’язки:** показують, як саме актори взаємодіють із певними функціями системи.

Діаграма використовується для формування загального уявлення про систему та її функціональні можливості.

Текстові описи сценаріїв використання

Для уточнення інформації, яка відображена на діаграмі, розробляються текстові сценарії, що деталізують взаємодію між користувачами та системою. Вони зазвичай містять:

- **Умови початку роботи:** обставини або стани, які повинні бути виконані перед початком сценарію.
- **Результат:** те, чого досягає система після завершення сценарію.
- **Основний алгоритм дій:** послідовність кроків, які виконуються в рамках процесу.
- **Альтернативні шляхи:** описують виняткові ситуації або відхилення від стандартного сценарію.

Діаграма класів

Даний тип діаграми відображає логічну структуру системи, включаючи класи, їхні характеристики та способи взаємодії.

Складові діаграми:

- **Класи:** базові елементи, що описують структуру об’єктів системи.
- **Характеристики:** змінні, які зберігають інформацію про стан об’єктів.
- **Функції:** дії, які можуть виконувати об’єкти.
- **Зв’язки між класами:** включають різні типи взаємодії, такі як асоціація, агрегація, композиція та спадкування.

Ця діаграма є корисною під час проектування баз даних або об’єктно-орієнтованих систем.

Проектування баз даних

База даних забезпечує ефективну організацію, зберігання та доступ до інформації. Процес проектування включає створення логічної та фізичної моделей даних:

- **Логічна модель:** описує структуру даних на концептуальному рівні, включаючи таблиці, зв’язки та ключі.
- **Фізична модель:** деталізує способи фізичного зберігання даних, їхнє розташування на носіях і методи доступу.

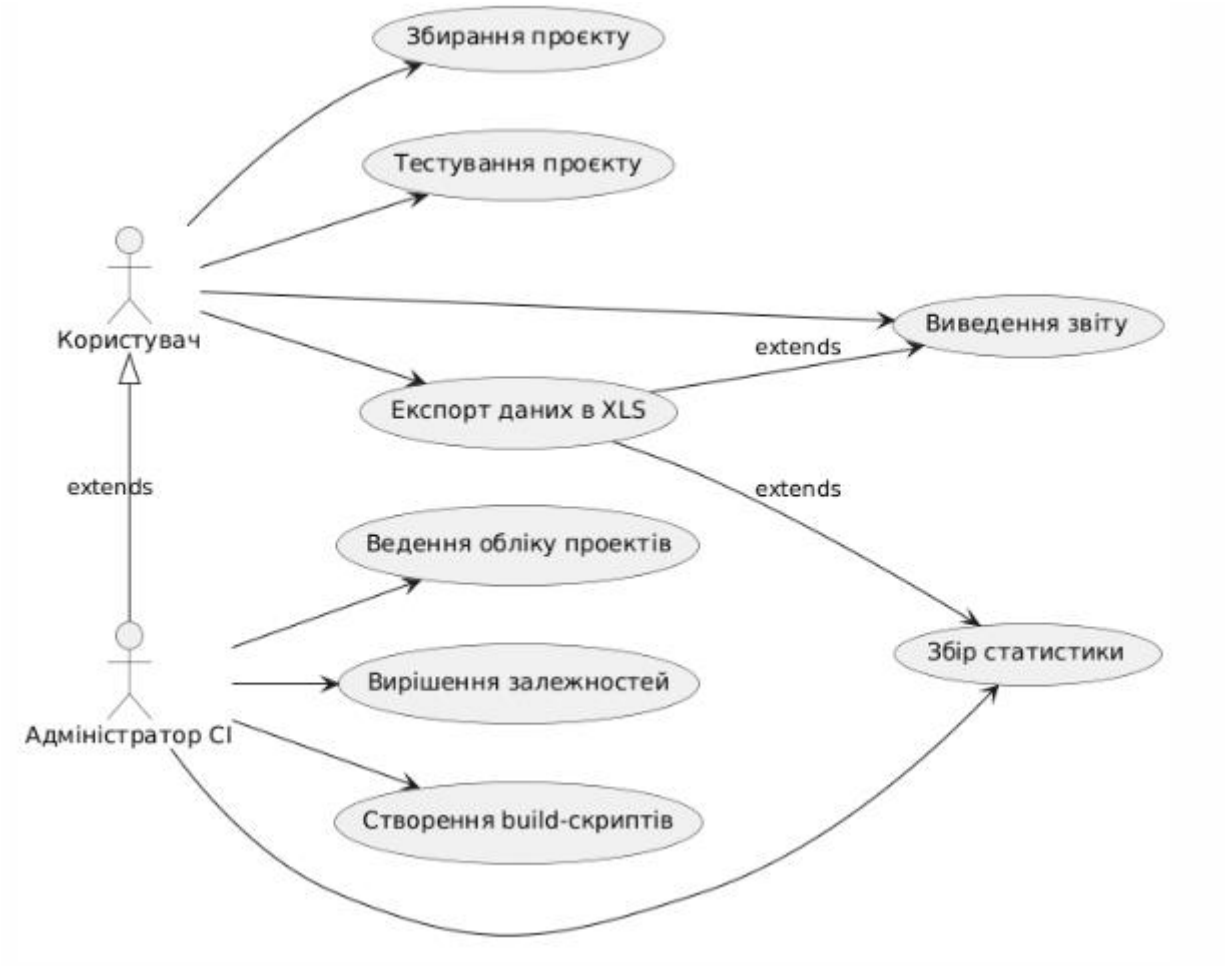
Оптимізація структури бази даних
Для уникнення дублювання даних і забезпечення їхньої узгодженості використовується процес нормалізації. Основні етапи нормалізації:

1. **Перша нормальна форма:** кожен стовпець містить лише одне значення.
2. **Друга нормальна форма:** всі стовпці, які не є ключовими, залежать від первинного ключа.
3. **Третя нормальна форма:** усуває залежність між неключовими стовпцями.

Хід роботи

Схема прецедентів

Схема прецедентів, що відповідає темі «CI-сервер».



На основі діаграми опишемо 3 сценарії використання:

Сценарій 1: Збирання проєкту

Передумови: Проект знаходиться у системі контролю версій, і до нього були внесені зміни.
Постумови: Проект успішно зібраний, або створено повідомлення про помилку.
Взаємодіючі сторони: Розробник, CI-сервер.
Короткий опис: Даний сценарій описує процес автоматичного запуску збірки проєкту після внесення змін до репозиторію.

Основний потік подій:

1. Розробник вносить зміни до репозиторію та робить push змін.
2. CI-сервер автоматично виявляє зміни в репозиторії.
3. CI-сервер ініціює процес збирання проєкту.

4. CI-сервер завантажує залежності та починає компіляцію проєкту.
5. Після завершення збірки CI-сервер відправляє повідомлення про успішну або невдалу збірку.

Виключення: Невдала збірка: Якщо в процесі компіляції виникає помилка, CI-сервер зупиняє збірку і надсилає повідомлення про помилку розробнику.

Примітки: Цей сценарій може бути ініційований вручну або автоматично, залежно від налаштувань CI-сервера.

Сценарій 2: Створення нового проекту на CI-сервері

Передумови: Адміністратор CI має доступ до CI-сервера, і репозиторій з вихідним кодом проєкту доступний для інтеграції.
Постумови: Проект успішно створений на CI-сервері, налаштовані параметри збірки, і сервер готовий автоматично збирати проект.
Взаємодіючі сторони: Адміністратор CI, CI-сервер.
Кроткий опис: Сценарій описує процес налаштування нового проекту на CI-сервері для його автоматичної збірки, тестування і моніторингу.

Основний потік подій:

1. Адміністратор CI заходить у веб-інтерфейс або консоль CI-сервера.
2. Адміністратор натискає на опцію створення нового проекту.
3. CI-сервер запитує основні параметри проекту, такі як назва проекту, репозиторій з вихідним кодом (URL Git, SVN тощо), гілка для відстеження.
4. Адміністратор вводить дані проекту, включаючи URL репозиторію та гілку, яку потрібно збирати.
5. CI-сервер запитує вибір технології збірки або build-скрипта (наприклад, Maven, Gradle для Java, MSBuild для C#, або інший інструмент залежно від мови).
6. Адміністратор вказує відповідний build-скрипт або конфігураційний файл для збирання (наприклад, pom.xml для Maven або build.gradle для Gradle).
7. Адміністратор налаштовує тригери для автоматичного запуску збірки (наприклад, на кожен push у репозиторій або за розкладом).
8. CI-сервер створює проєкт і перевіряє доступ до репозиторію.
9. Адміністратор зберігає конфігурацію проєкту, і CI-сервер готовий до запуску збірки при внесенні змін у код.

Виключення:

- **Невірний URL репозиторію**: Якщо вказаний репозиторій недоступний або URL невірний, CI-сервер повідомляє про помилку і пропонує перевірити дані.
- **Невідповідний build-скрипт**: Якщо CI-сервер не може знайти або розпізнати вказаний build-скрипт (наприклад, помилка у конфігурації файлу), сервер зупиняє налаштування і відправляє повідомлення про помилку адміністратору.

Примітки: Після створення проєкту адміністратор може також налаштувати додаткові параметри, такі як інтеграція з тестовими фреймворками або налаштування середовищ для деплою.

Сценарій 3: Ведення статистики збірок

Передумови: У системі вже виконувались кілька збірок проектів.
Постумови: Статистика збірок збережена і доступна для перегляду.
Взаємодіючі сторони: Адміністратор CI, CI-сервер.
Короткий опис: Сценарій описує процес збирання та збереження статистики успішних і невдалих збірок для аналізу продуктивності.

Основний потік подій:

1. CI-сервер автоматично зберігає дані про кожну виконану збірку.
2. Адміністратор CI може переглядати звіти з успішних та невдалих збірок у вигляді статистичних даних.
3. CI-сервер відображає детальну інформацію про кожну збірку, включаючи дату, час, статус (успішна або невдала) та причини можливих помилок.
4. Адміністратор CI використовує ці дані для покращення процесу збірки.

Виключення. Відсутність даних: Якщо сервер з якоїсь причини не зміг зберегти дані збірки, адміністратору буде надіслано повідомлення про цю помилку.

Примітки: немає.

Діаграма класів

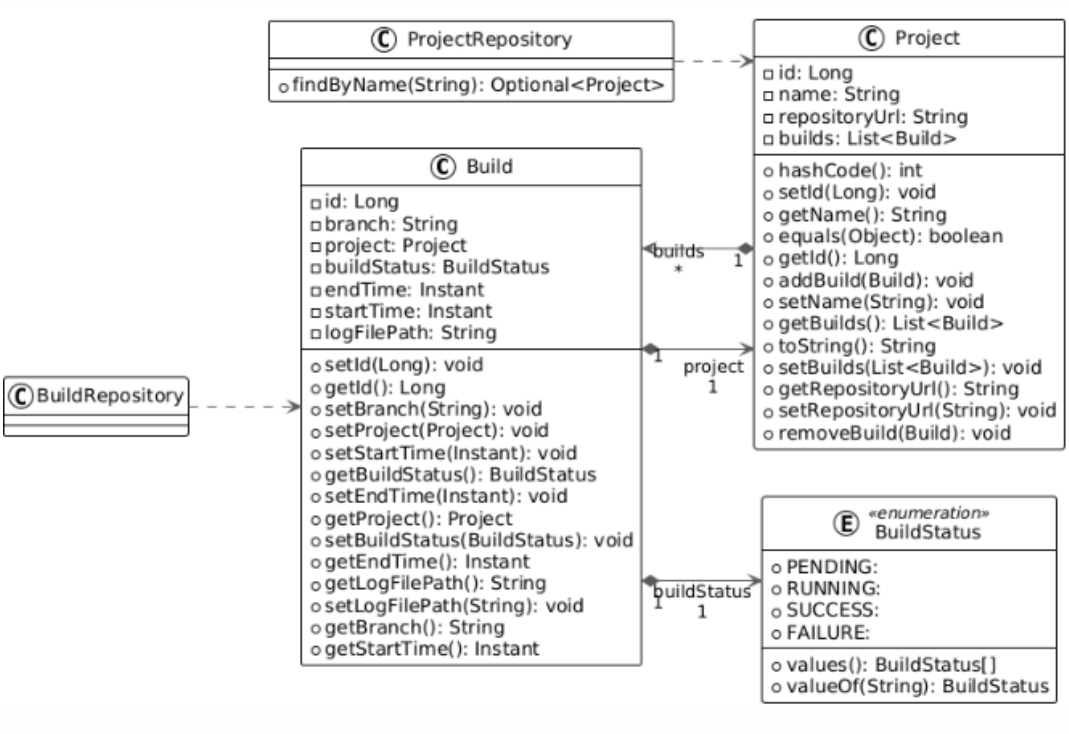
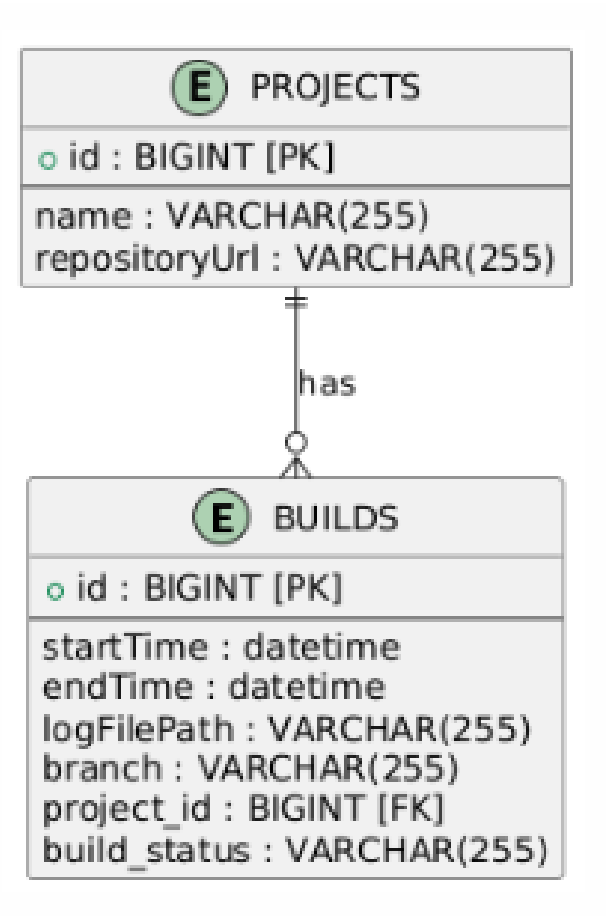


Схема бази даних



Висновки

На цій лабораторній роботі я проаналізував свою тему, створив діаграму прецедентів, діаграму класів та спроектував базу даних.