технологічного стеку, автоматизованих інструментів для тестування та візуалізації, а також систематизація процесів розробки сприяє підвищенню масштабованості, швидкості розробки та зменшенню впливу людських помилок. Це створює стабільну основу для розвитку Web 3.0 та впровадження нових децентралізованих сервісів.

Список використаних джерел:

1. Ethereum Development Documentation. Official Ethereum Protocol Documentation. URL: https://ethereum.org/en/developers/docs/ (дата звернення: 16.03.2025).
2. Hardhat Documentation. Ethereum development environment. URL: https://hardhat.org/ (дата звернення: 16.03.2025).
3. Foundry Book. Smart Contract Development Suite. URL: https://book.getfoundry.sh/ (дата звернення: 16.03.2025).
4. OpenZeppelin Contracts. Standard Smart Contract Library. URL: https://docs.openzeppelin.com/contracts/ (дата звернення: 16.03.2025).
5. Security Considerations for Smart Contracts / S. DeFi Security. Ethereum Foundation. URL: https://ethereum.org/en/developers/docs/smart-contracts/security/ (дата звернення: 16.03.2025).

УДК 004.42

УПРАВЛІННЯ З'ЄДНАННЯМИ З БАЗОЮ ДАНИХ У FLASK-ЗАСТОСУНКУ ДЛЯ ЗАПОБІГАННЯ БЛОКУВАННЮ SQLite

Валовий А.Є
angegter@gmail.com
Черкаський державний фаховий бізнес-коледж
Фальченко Н.Г.
м.Черкаси, Україна

Під час розробки веб-застосунку для автоматизованої перевірки алгоритмічних задач на Python виникла необхідність зберігати результати перевірок у базі даних. Як сховище було обрано SQLite через його простоту та

відсутність необхідності у налаштуванні окремого сервера. Проте в процесі тестування системи виявилась критична проблема, пов'язана з одночасним доступом кількох запитів до бази даних.

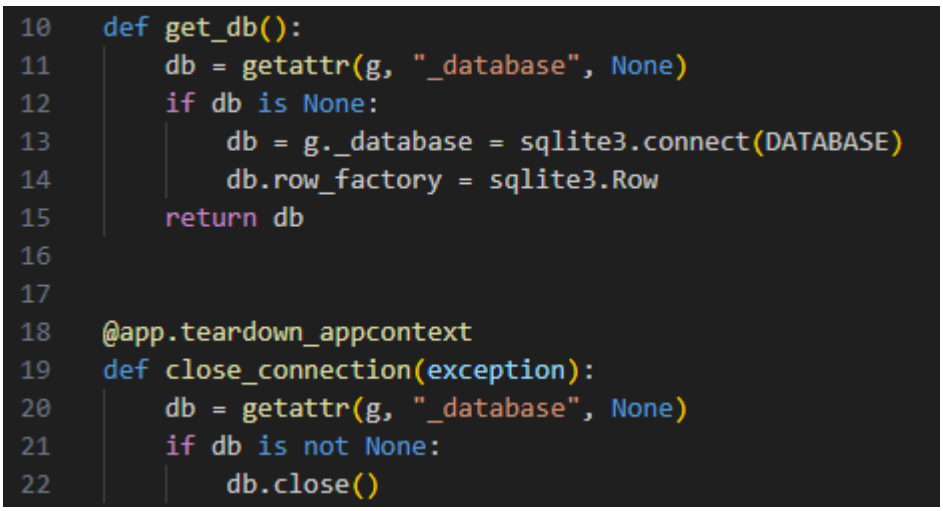
Flask, починаючи з версії 1.0, за замовчуванням обробляє HTTP-запити у багатопотоковому режимі (threaded=True). Це означає, що кілька запитів можуть оброблятися одночасно у різних потоках. SQLite за замовчуванням забороняє використання одного з'єднання у різних потоках через параметр check_same_thread=True.

При одночасному надходженні двох запитів на перевірку коду Flask намагався використати те саме з'єднання з БД у різних потоках. Це призводило до виключення sqlite3.OperationalError: database is locked, через яке другий запит аварійно завершувався, а результат перевірки не зберігався. Відтворити помилку вдалось лише при одночасному відкритті кількох вкладок браузера з однаковою задачею.

Перший підхід полягав у передачі параметра check_same_thread=False при створенні з'єднання. Це знімає обмеження SQLite на використання з'єднання між потоками, проте не вирішує проблему конкурентного запису – з'єднання залишається спільним, і при одночасних транзакціях блокування все одно виникає.

Коректним рішенням виявилось створення окремого з'єднання для кожного HTTP-запиту з використанням об'єкта g з модуля flask. Об'єкт g є локальним для поточного контексту запиту: з'єднання відкривається при першому зверненні до бази у функції get_db() і автоматично закривається після завершення запиту через декоратор teardown_appcontext. Таким чином кожен потік отримує власне ізольоване з'єднання.

Додатково було увімкнено режим WAL (Write-Ahead Logging) командою PRAGMA journal_mode=WAL. У цьому режимі операції читання не блокують запис і навпаки, що суттєво підвищує пропускну здатність при конкурентних запитах.



```
10  def get_db():
11      db = getattr(g, "_database", None)
12      if db is None:
13          db = g._database = sqlite3.connect(DATABASE)
14          db.row_factory = sqlite3.Row
15      return db
16
17
18  @app.teardown_appcontext
19  def close_connection(exception):
20      db = getattr(g, "_database", None)
21      if db is not None:
22          db.close()
```

Рисунок 1 – Реалізація управління з'єднаннями з базою даних у Flask

Після впровадження підходу з об'єктом g помилка database is locked зникла повністю. Система коректно опрацьовує одночасні запити на перевірку коду від різних користувачів. Кожне з'єднання існує рівно стільки, скільки триває HTTP-запит, що унеможливлює його захоплення іншим потоком. Увімкнення WAL-режиму додатково скоротило час відповіді при паралельному навантаженні.

Використання глобального з'єднання з SQLite у багатопотоковому Flask-застосунку призводить до блокувань і втрати даних. Правильним рішенням є створення з'єднання на рівні запиту через контекстний об'єкт g з автоматичним закриттям після відповіді. Такий підхід разом із WAL-режимом забезпечує стабільну роботу застосунку при одночасних зверненнях користувачів.

Список використаних джерел

1. Flask Documentation. The Application Context. URL: https://flask.palletsprojects.com/en/3.0.x/appcontext/ (дата звернення: 18.03.2026).
2. SQLite Documentation. WAL-mode. URL: https://sqlite.org/wal.html (дата звернення: 18.03.2026).
3. Grinberg M. Flask Web Development. 2nd ed. O'Reilly Media, 2018. 316 p.
4. Hettinger R. Python Cookbook: Recipes for Mastering Python 3. 3rd ed. O'Reilly Media, 2013. 706 p.

5. SQLite Documentation. SQLite Frequently Asked Questions. URL: https://sqlite.org/faq.html (дата звернення: 18.03.2026).

УДК 004.65

ІНТЕРАКТИВНА МАПА ЗМІН КЛІМАТУ НА ОСНОВІ ВЕЛИКИХ ДАНИХ

Сас Д. А.
dominikasas9110@gmail.com
Черкаський державний фаховий бізнес-коледж
Люта М. В.
м. Черкаси, Україна

Зміна клімату є однією з найбільших проблем сьогодення [4]. Вона проявляється у підвищенні середньої температури повітря, зміні кількості опадів та у погодних явищах, таких як посухи, повені та урагани. Ці зміни впливають не лише на природу, але й на життя людей, економіку та сільське господарство.

Для дослідження кліматичних змін сьогодні активно використовуються інформаційні технології. Одним із найсучасніших підходів є використання великих даних (Big Data) [2]. Великі дані – це дуже великі обсяги інформації, які неможливо обробити звичайними способами.

Інтерактивна мапа змін клімату – це застосунок, який дозволяє збирати, аналізувати та показувати екологічні дані у зручному вигляді. Інтерактивна мапа може містити різні шари інформації. Наприклад, окремо можна переглянути рівень забруднення повітря, викиди вуглекислого газу, стан льодовиків або рівень моря. Користувач може переглядати інформацію про різні регіони світу, зміну температури та інші показники, може сам обирати, які саме дані його цікавлять, що робить додаток більш зручним і гнучким.

Основою такої системи є відкриті дані, які надаються міжнародними організаціями та науковими центрами [3]. Вони включають дані з метео-станцій, супутників, океанічних досліджень та інших джерел. Ці дані постійно оновлюються, тому мапа показує актуальну ситуацію.

Однією з переваг такого додатку є зручність використання. Навіть люди без спеціальних знань можуть легко зрозуміти, як змінюється клімат.