УДК 004.27:004.823]:004.65
DOI: 10.31866/2617-796X.6.2.2023.293618

Ірина Овчарук,
кандидат технічних наук,
доцент кафедри інформаційних технологій,
Державний університет інфраструктури та технологій,
Київ, Україна
ovch05@ukr.net
https://orcid.org/0000-0003-4255-5816

Ілля Тихонков,
магістрант кафедри інформаційних технологій,
Державний університет інфраструктури та технологій,
Київ, Україна
i.tykhonkov@gmail.com
https://orcid.org/0009-0009-1267-2537

АРХІТЕКТУРА ФРЕЙМВОРКУ ДЛЯ ПРОЄКТУВАННЯ ВИСОКОНАВАНТАЖЕНИХ ВЕБЗАСТОСУНКІВ

Мета статті: опис архітектури власного фреймворку Minegraph для проєктування спрощення процесу розробки сучасних високонавантажених вебзастосунків.

Методами дослідження є огляд та аналіз сучасних технологій для розробки архітектури та проєктування фреймворків.

Новизною дослідження є аналіз фреймворків для розробки вебзастосунків, їх особливостей та застосування залежно від завдань, аналіз їх архітектури. У статті акцентовано увагу на проблемі використання готових рішень під час створення високонавантажених та масштабованих вебзастосунків. Представлено авторську розробку фреймворку для масштабованих та високонавантажених додатків будь-якого рівня складності, особливості запропонованих моделей, а також докладно описано архітектуру фреймворку Minegraph, проаналізовано особливості цього фреймворку, виокремлено його сильні та слабкі сторони.

Висновки. У статті проаналізовано наявні фреймворки для розробки вебзастосунків. Докладно представлено авторську розробку архітектури фреймворку Mineraph, а саме: драйвери для взаємодії з базами даних, NoSQL бази даних; особливості моделі для надання можливості додавати або змінювати основну логіку для обробки запитів; моделі, де зберігається логіка для обробки запитів та її особливості, висвітлені питання стосовно гнучкого механізму для налагодження основної логіки застосунків.

Створений фреймворк є готовим структурним каркасом для розробки масштабованих та високонавантажених застосунків будь-якого рівня складності. Важливо зазначити, що розробка на цьому фреймворку не потребує глибокого аналізу архітектури вебсистем, а лише базових знань про середовище розробки та вміння орієнтуватися в документації. У зазначеному фреймворку враховано переваги попередніх аналогів, він містить передові технології, а це підвищує продуктивність та ефективність розробки масштабованих та

високонавантажених вебзастосунків. У роботі описано особливості моделі фреймворку Minegraph, проаналізовано особливості цього фреймворку, виокремлено його сильні та слабкі сторони. Фреймворк Minegraph виходить за межі структурних можливостей, безперешкодно інтегрує та використовує новітні технології для обробки та зберігання даних. Крім цього, він використовує також хмарні рішення, що призводить до значного покращення функціональності системи та одночасно зменшує витрати на управління інфраструктурою.

Фреймворк, представлений у цій роботі, є не просто структурною основою, а й надзвичайно універсальним та адаптивним інструментом, готовим вміщати застосунки найвищої складності та масштабності.

Ключові слова: фреймворк; архітектура фреймворків; вебзастосунки; вебкаркас; подійно-орієнтована модель; бізнес-логіка; бізнес-правила; масштабованість; транзакція; база даних.

Вступ. Нині люди не можуть уявити своє життя без доступу до глобальної мережі. Інтернет щодня пропонує нові технології та фреймворки, і сучасному фахівцю потрібно оперативно реагувати на ці зміни і робити свої адаптації гнучкими та масштабованими (Розломій та Науменко, 2022). Тож IT-спеціаліст повинен стежити за актуальними тенденціями в індустрії, вивчати нововведення.

Одним з основних помилкових підходів у розробці архітектури є використання розробниками звичних технологій для створення корпоративних застосунків, а потім застосування різних технологічних конструкцій, що може призвести до заплутаного коду та розширення архітектури, проблем із масштабованістю. Правильно розроблена архітектура є ключовою для малих і великих проєктів, оскільки відсутність планування архітектури може призвести до серйозних проблем. Ідеально побудована архітектура допомагає зменшити витрати на розробку вебзастосунків і визначає успіх проєкту.

У статті зосереджено увагу на проблемі використання готових рішень під час створення високонавантажених та масштабованих вебзастосунків. Саме недостатність простого, гнучкого та інтуїтивно зрозумілого фреймворку для розробки таких застосунків спонукає створити свій власний вбудований фреймворк для подібних завдань.

Результати дослідження. Однією з ключових особливостей фреймворку Minegraph є наявність граф-орієнтованого API, яке ефективно управляє роботою системи, особливо в нетривіальних випадках. Це забезпечує розділення бізнес-логіки від обробки запитів користувача. Усі зміни, що відбуваються в системі, зберігаються у вигляді об'єктів подій, відомих як «event». Ці події зберігаються в базі даних у вигляді різних шарів, що забезпечує високий рівень надійності і нульові втрати даних.

Одна з основних переваг фреймворку Minegraph – це його легко масштабований граф-орієнтований API для бекенду. Він має кілька моделей, які можуть бути вибрані відповідно до потреб системи.

Архітектура Minegraph має надзвичайно гнучку структуру. Крім драйверів для взаємодії з базами даних, у фреймворку є шар, який дає можливість використовувати NoSQL бази даних, водночас граф-орієнтовані. Для оптимізації пошуку передбачено окремий модуль, що працює з пошуковим двигуном Elasticsearch та має значні переваги перед реляційними базами даних. Фреймворк має вбудовану підтримку транзакцій та можливість згладження даних.

Створений фреймворк є готовим структурним каркасом для розробки масштабованих та високонавантажених застосунків будь-якого рівня складності. Розробка на цьому фреймворку не потребує глибокого аналізу архітектури вебсистем, а лише базових знань про середовище та вміння орієнтуватися в документації. Фреймворк містить передові технології для обробки та зберігання даних, а також використовує хмарні рішення для покращення функціональності системи та зниження затрат на її інфраструктуру.

Зараз є велика кількість фреймворків для розробки вебзастосунків, що дає змогу ефективно зменшити час для створення нового вебдзастосунку.

Один із цих фреймворків – *Yii Framework* (n.d.), що є вебкаркасом, розробленим на PHP і заснованим на парадигмі MVC. Він відзначається високою продуктивністю, надійністю і багатофункціональністю та надає широкий набір можливостей для швидкої і легкої розробки оптимізованих вебзастосунків. Значною перевагою є чітка та всеосяжна документація, а також різноманітні моделі (Є. Щербаков та М. Щербакова, 2020). Використовуючи Yii Framework, можливо створювати проєкти різного масштабу, і фреймворк надає інструменти для допомоги в тестуванні та налагодженні застосунків.

Ще одним популярним фреймворком для розробки вебзастосунків є *Django* (n.d.) Це вільний фреймворк для створення вебзастосунків використовує підхід MVC. Проєкт дає можливість будувати вебсайти з одним або кількома застосунками, які можливо підключати в проєкт, дотримуючись рекомендацій.

Обидва фреймворки – Django та Yii Framework – надають можливість вибору бази даних для роботи, постачають необхідні інтерфейси і дають змогу розробляти контролери для обробки запитів (Філімонова та Селіванова, 2021). Вони дуже ефективні у своїх сферах використання, але не відповідають потребам у вирішенні проблеми підтримки високонавантажених вебзастосунків.

Однією з ключових особливостей фреймворку Minegraph є наявність граф-орієнтованого API, яке ефективно управляє роботою системи, особливо в нетривіальних випадках. Це забезпечує розділення бізнес-логіки від обробки запитів користувача. Усі зміни, що відбуваються в системі, зберігаються у вигляді об'єктів подій, відомих як «event» (Бурлаченко та Доценко, 2023). Ці події зберігаються в базі даних у вигляді різних шарів, що забезпечує високий рівень надійності і нульові втрати даних.

За допомогою архітектури EGF2 можливо створювати масштабовані та об'ємні системи обробки даних з меншими зусиллями. Ця архітектура робить процес розробки та підтримки великих систем більш зручним і ефективним завдяки заздалегідь визначеній моделі подій та графоорієнтованому підходу до управління даними.

Одна з основних переваг фреймворку Minegraph – це його легко масштабований граф-орієнтований API для бекенду. Він містить кілька моделей, які можуть бути вибрані відповідно до потреб системи.

Архітектура Minegraph має надзвичайно гнучку структуру. По-перше, крім драйверів для взаємодії з базами даних, у фреймворку є шар, який дає можливість використовувати NoSQL бази даних, водночас граф-орієнтовані. Для оптимізації пошуку передбачено окремий модуль, який працює з пошуковим двигуном Elasticsearch та має значні переваги перед реляційними базами даних (*Elastic*,

n.d.). Додатково фреймворк має вбудовану підтримку транзакцій та можливість згладження даних. Один з особливих типів моделей відомий як «гібридний» і дає змогу комбінувати надійність реляційних баз даних і швидкість NoSQL рішень. Основною метою фреймворку Minegraph є об'єднання всіх цих механізмів у єдину гармонійну систему, інтуїтивно зрозумілу для розробників.

У фреймворку є два типи контролерів для обробки основних запитів, які включаються в моделі logic і client-api.

Модель client-api надає можливість додавати або змінювати основну логіку для обробки запитів PUT, POST, GET і DELETE для об'єктів і для зв'язків. Основна логіка містить стандартні операції видалення, відновлення, отримання і створення об'єктів – вже вбудована у фреймворк і не вимагає додаткової реалізації. Але її можливо розширити, наприклад, для змін об'єктів або їх зв'язків. Головний аспект полягає в тому, що ці контролери виконуються, перш ніж відповідь надсилається користувачеві.

У моделі logic зберігається логіка для обробки запитів після того, як відповідь вже відправлена користувачеві. Інформація про зміну об'єкта передається в logic-модуль через події. Тут можливо визначити правила обробки для кожного конкретного запиту, а також встановити порядок їх виконання. Цей підхід добре гармонує з подійно-орієнтованою природою Node.js. Кожне правило може ініціювати інші правила під час зміни стану інших об'єктів, що сприяє ланцюговій реакції.

Фреймворк Minegraph надає гнучкий механізм для налаштування основної логіки застосунків. Якщо логіка налаштована вірно, це може зробити застосунки швидкими та надійними.

Архітектура EGF2 представлена на рис. 1.

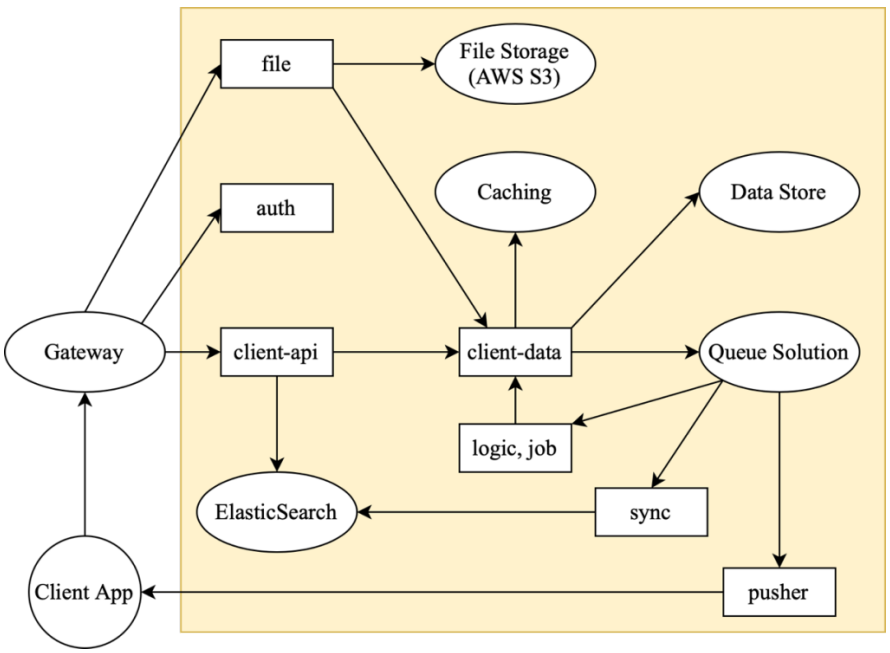

Рис. 1. Архітектура EGF2

Область кольору містить основні моделі EGF2. Стрілки вказують на взаємодію моделей, які виконують запити або передають дані іншим моделям. EGF2 складається з численних модулів, які поділяються на багато мікромодулів.

Фреймворк EGF2 володіє графоорієнтованим API. Така архітектура дає можливість наявності численних залежностей у проєкті та швидкого переходу між ними. Аналогічний підхід використовується у корпорації Facebook. Цей фреймворк призначений для застосунків із великою кількістю об'єктів та їх взаємозв'язками, дає змогу легко додавати або видаляти нові класи об'єктів. EGF2 виокремлюється на ринку серед інших фреймворків завдяки двом головним аспектам: швидкому масштабуванню та відмовостійкості.

Кожен модуль у цьому фреймворку може працювати незалежно. Немає потреби турбуватися про деталі зберігання на рівні бази даних. Об'єкти додаються в систему за допомогою конфігурації, що усуває потребу в міграції даних.

Модуль «Client-API» виконує роль обробника клієнтських запитів у цьому фреймворку. Він виконує наступні завдання:

– приймає клієнтські запити;

– контролює автентифікацію запитів, щоб розуміти, хто саме здійснив запит до «Client-API». Виклики аутентифікації використовують службовий токен аутентифікації, який передається разом із запитом;

– здійснює контроль доступу до запиту. ACL-компонент клієнтського API може перевіряти користувача з певною роллю, чи дозволено виконувати конкретну дію у графі. Це може бути створення об'єкта чи зв'язку, зміну об'єкта, видалення об'єкта або зв'язку. Інформація про права доступу, який може використовуватися знову для всіх служб, збирається у графі конфігурації;

– модуль «Client-API» відповідає за виконання валідації полів об'єкта при запитах PUT і POST. Всі обмеження вказані у графі конфігурації;

– після перевірки запиту підтверджує вхід та дає дозвіл на роботу з модулем «client-data» для подальших маніпуляцій із даними;

– містить основні контролери обробки запитів.

Модуль «Client-API» підлягає масштабуванню, зокрема автоматичному масштабуванню з використанням AWS C3M.

Модуль «Client-data» надає доступ до даних системи. Він пропонує практично той же API, що й модуль «Client-API». «Client-data» недоступний з інтернету і видимий лише для інших модулів системи.

Модуль «Client-data» є єдиним компонентом, за допомогою якого можливо маніпулювати даними в системі. Всі сервіси використовують цей модуль, і перевагою є те, що ніхто не має прямого доступу до БД ззовні. «Client-data» використовує БД та, за необхідності, служби кешування, щоб задовольнити потреби стосовно даних.

«Client-data» повністю адаптований і має можливість легкого масштабування шляхом додавання додаткових запущених екземплярів сервісу. Цей модуль приховує деталі про використання БД і забезпечує єдиний шлях доступу до даних для інших сервісів.

Модуль «Auth» є сервісом, відповідальним за аутентифікацію та реєстрацію користувачів. Він надає набір ендпоінтів, пов'язаних із реєстрацією користувачів та входом до системи. Крім того, модуль «Auth» надає внутрішні ендпоінти для інших сервісів, зокрема для модуля «Client-API».

Модуль «Logic» відповідає за бізнес-логіку. Цей сервіс є повністю визначеним і має властивість легкого масштабування. Він стежить за змінами бази даних і реагує на них відповідно до своїх контролерів. Модуль «Client-API» також може містити деякі правила бізнес-логіки при обробці, а їхня відмінність полягає в тому, що «Logic» виконує свою бізнес-логіку вже після відправки запиту до «Client-API».

З використанням модуля «Logic» виникає можливість легко додати конкретну реалізацію бізнес-правил, що спрощує та організовує бізнес-логіку в системі. Під час побудови системи з використанням EGF2 модуль «Logic» є місцем, де міститься значна частина користувацького коду.

Модуль «Sync» є повністю масштабованим сервісом, відповідальним за синхронізацію кластера «ElasticSearch» зі змінами в даних. Цей модуль сприймає зміни, які відбуваються у базі даних, і відповідно реагує, належним чином оновлюючи дані в ElasticSearch для подальшого швидкого пошуку. Цей пошук доступний для клієнта і для інших модулів.

Модуль «Pusher» є масштабованим сервісом, відповідальним за надсилання повідомлень: через електронну пошту, SMS або WebSockets. Він підтримує різні шаблони повідомлень та також сприймає зміни в базі даних.

Модуль «File» є немасштабованим сервісом, який дає можливість зберігати файли в розподіленій хмарній системі AWS S3 bucket. Він підтримує різні варіанти обробки зображень шляхом зміни їхніх розмірів.

Модуль «Job» використовують для обробки довгих та складних завдань. Наприклад, генерація звітів, експорт, імпорт тощо обробляються за допомогою цього модуля. Це найбільш значущий модуль, який є у архітектурі EGF2.

Однією з ключових переваг запропонованого фреймворку є подійно-орієнтована модель архітектури, яка дає можливість відстежувати оновлення безпосередньо в базі даних та підключати до неї обробники, або, іншими словами, тригери. Цей механізм дає змогу підписуватися на сервіси та перехоплювати оновлення даних, замість того щоб реагувати на запити. Такий підхід дає можливість писати більш винахідливу архітектуру не на рівні запитів, а на рівні даних, уникати дублювання коду та логічних помилок. Кожен розроблений тригер виконується синхронно з іншими, що позбавляє від потреби писати багатопотоковий застосунок та використовувати переваги асинхронного середовища розробки Node.js.

Вся основна логіка розбита на компонентні атомарні логічні частини і виконується синхронно одна з одною, що дуже спрощує відстеження витоків пам'яті і є важливим для високонавантажених проєктів. Наприклад, сервіс Logic дає змогу створювати кілька правил обробки для кожної події, встановлюючи їх послідовність або даючи можливість виконуватися синхронно. Кожне таке правило несе з собою одну логічну дію і є однією невеликою функцією. Такий підхід дає змогу контролювати логіку застосунка, уникати великих кодових блоків.

Це розбиття на основні логічні частини також дає можливість логувати помилки та в подальшому їх усувати. Кожна подія записується у спеціальну таблицю Event, а відповідно забезпечено уникнення можливості появи невідстежуваних операцій із даними та витоків даних.

Графова структура даних дає можливість зберігати залежності у вигляді сутностей, що спрощує дії з ними, а також дає змогу зв'язувати дані будь-яким зручним способом, уникати глибокої вкладеності залежностей та звільнити базу даних від зайвих запитів.

Технології збереження даних, що використовуються – elasticsearch, rethinkdb, cassandra, Amazon S3 bucket, – піддаються дуже легкій масштабованості. Сама можливість масштабування та інструменти її реалізації надаються самим програмним продуктом.

Сервіс sync синхронізує дані з пошуковим двигуном elasticsearch. Дані видаляються та створюються паралельно з тим, як вони створюються та видаляються в основній базі даних, розробникам не потрібно писати додаткового коду, достатньо налагодити конфігурацію, решту відіграє сервіс. Поміж основних обробників можливо написати власні, у випадку необхідності виконати які-небудь маніпуляції з даними перед їхнім відправленням у двигун.

Основним обмеженням elasticsearch є затримка під час транзакцій: при відправці запиту сервер отримує відповідь до того, як відбудеться транзакція в elasticsearch, затримка становить у середньому 0,5 секунди, тому цей сервер не підходить для зберігання аутентифікаційних токенів та для різних реактивних даних. Незважаючи на цей недолік, цей двигун вважається найшвидшим та має найгнучкіше API для запитів.

З огляду на велику кількість модулів, цей фреймворк менше підходить для малих проєктів через складність налаштування, але ідеально підходить для середніх і великих проєктів.

Висновки. У роботі описано особливості моделі фреймворку Minegraph, проаналізовано особливості цього фреймворку, виокремлено його сильні та слабкі сторони. Потрібно зазначити, що він містить передові технології для обробки та зберігання даних, а також використовує хмарні рішення для покращення функціональності системи та зниження затрат на її інфраструктуру.

Всебічна дослідницька робота із фреймворком Minegraph не лише розкрила його основні компоненти, функції, але й довела його глибоке значення в сучасній розробці програмних застосунків. Фреймворк, представлений у цій роботі, є не просто структурною основою, а й надзвичайно універсальним та адаптивним інструментом, готовим вміщати застосунки найвищої складності та масштабності. Важливо підкреслити, що для використання повного потенціалу цього фреймворку не потрібна глибока експертиза в архітектурі вебсистем, а достатньо базового розуміння середовища розробки та вміння користуватися документацією для успішної розробки проєктів.

До того ж фреймворк Minegraph виходить за межі структурних можливостей, безперешкодно інтегруючи та використовуючи передові технології для обробки та зберігання даних. Крім цього, він розумно використовує хмарні рішення, що

призводить до значного покращення функціональності системи та одночасно зменшує витрати на управління інфраструктурою.

У сучасному технологічному ландшафті фреймворк Minegraph є незамінним та потужним інструментом для розробників. Його гнучкість, масштабованість та вбудована здатність до інновацій дають можливість розробникам створювати та підтримувати високорівневі застосунки з неперевершеною ефективністю та результативністю. Цей фреймворк – не просто інструмент; це вікно до інновацій у цифровій сфері.

Під час вивчення фреймворку Minegraph стає очевидним, що його вплив сягає далеко за межі розробки. Він є зміною парадигми у підході до створення програмних застосунків, пропонуючи комплексне та оптимізоване рішення, яке надає розробникам можливість навігувати в складному світі сучасних технологій із впевненістю, відкриваючи нові можливості у цифровій сфері.

У статті проаналізовано наявні фреймворки для розробки вебзастосунків. Докладно представлено авторську розробку архітектури фреймворку Mineraph, а саме: драйвери для взаємодії з базами даних, NoSQL бази даних; особливості моделі з можливістю додавати або змінювати основну логіку для обробки запитів; моделі, де зберігається логіка для обробки запитів та її особливості, висвітлені питання стосовно гнучкого механізму для налагодження основної логіки застосунків.

У створеному фреймворку враховано переваги попередніх аналогів і всі новітні технології, він є готовим структурним каркасом для розробки масштабованих та високонавантажених застосунків будь-якого рівня складності. Важливо зазначити, що розробка на цьому фреймворку не потребує глибокого аналізу архітектури вебсистем, а лише базових знань про середовище розробки та вміння орієнтуватися в документації.

СПИСОК ПОСИЛАНЬ

Бурлаченко, І.С. та Доценко, Д.В., 2023. Продуктивність ORM для серверних фреймворків вебзастосунків. В: *Ольвійський форум – 2023: стратегії країн Причорноморського регіону в геополітичному просторі*, XVII Міжнародна наукова конференція, м. Миколаїв,Україна, 15-18 червня 2023 р. Миколаїв: Чорноморський національний університет імені Петра Могили, с.7.

Розломій, І.О. та Науменко, С.В.. 2022. Фреймворки для розробки серверної частини web-додатків. В: *Інформаційна безпека та комп'ютерні технології,* Матеріали V Міжнародної науково-практичної конференції, Кропивницький, Україна, 19-20 травня 2022 р. Кропивницький: Центральноукраїнський національний технічний університет, с. 42-43.

Філімонова, Т.О. та Селіванова, А.В., 2021. Вибір платформи програмної реалізації веб-додатку. In: *Emerging Trends in Academic Research*, Conference Proceedings of the 1 st International Conference, Dublin, Ireland, February 10-12, 2021. Primedia elaunch, с.39-41.

Щербаков, Є.В. та Щербакова, М.Є., 2020. Особливості масштабування веб-додатків. *Вісник Східноукраїнського національного університету імені Володимира Даля*, [e-journal] 8(264), с.15-19. https://doi.org/10.33216/1998-7927-2020-264-8-15-19

Django, n.d. [online]. Available at: <https://www.djangoproject.com> [Accessed 14 July 2023].
Elastic, n.d. [online]. Available at: <https://www.elastic.co> [Accessed 14 July 2023].
Yii Framework, n.d. [online]. Available at: <https://www.yiiframework.com> [Accessed 14 July 2023].

REFERENCES

Burlachenko, I.S. and Dotsenko, D.V., 2023. Produktyvnist ORM dlia servernykh freimvorkiv vebzastosunkiv [ORM performance for web application server frameworks]. In: *Olviiskyi forum – 2023: stratehii krain Prychornomorskoho rehionu v heopolitychnomu prostori* [Olbia Forum – 2023: strategies of the countries of the Black Sea region in the geopolitical space], XVII International Scientific Conference, Mykolaiv, Ukraine, June 15-18, 2023. Mykolaiv: Petro Mohyla Black Sea National University, p.7.
Django, n.d. [online]. Available at: <https://www.djangoproject.com> [Accessed 14 July 2023].
Elastic, n.d. [online]. Available at: <https://www.elastic.co> [Accessed 14 July 2023].
Filimonova, T.O. and Selivanova, A.V., 2021. Vybir platformy prohramnoi realizatsii veb-dodatku [Choosing a web application software implementation platform]. In: *Emerging Trends in Academic Research*, Conference Proceedings of the 1 st International Conference, Dublin, Ireland, February 10-12, 2021. Primedia elaunch, pp.39-41.
Rozlomii, I.O. and Naumenko, S.V., 2022. Freimvorky dlia rozrobky servernoi chastyny web-dodatkiv [Frameworks for the development of the server part of web applications]. In: *Informatsiina bezpeka ta kompiuterni tekhnolohii* [Information security and computer technologies], Proceedings of the V International Scientific and Practical Conference, Kropyvnytskyi, Ukraine, May 19-20, 2022. Kropyvnytskyi: Central Ukrainian National Technical University, pp.42-43.
Shcherbakov, Ye.V. and Shcherbakova, M. Ye., 2020. Osoblyvosti masshtabuvannia veb-dodatkiv [Scaling features of web applications]. *Visnik of the Volodymyr Dahl East Ukrainian national university*, [e-journal] 8(264), pp.15-19. https://doi.org/10.33216/1998-7927-2020-264-8-15-19
Yii Framework, n.d. [online]. Available at: <https://www.yiiframework.com> [Accessed 14 July 2023].

UDC 004.27:004.823]:004.65

Iryna Ovcharuk,
PhD in Technical Sciences,
Associate Professor at the Department
of Information Technologies,
State University of Infrastructure and Technology,
Kyiv, Ukraine
ovch05@ukr.net
https://orcid.org/0000-0003-4255-5816

Illia Tykhonkov,
Master's student at the Department
of Information Technologies,
State University of Infrastructure and Technology,
Kyiv, Ukraine
i.tykhonkov@gmail.com
https://orcid.org/0009-0009-1267-2537

FRAMEWORK ARCHITECTURE FOR DESIGNING HIGH-LOAD WEB APPLICATIONS

The purpose of the article is to describe the architecture of the proprietary Minegraph framework for designing a simplified process of developing modern high-load web applications.

The research methods are a review and analysis of modern technologies for developing architecture and designing frameworks.

The novelty of the study is the analysis of frameworks for developing web applications, their features and applications depending on the tasks, and the analysis of their architecture. The article focuses on the problem of using off-the-shelf solutions when creating high-load and scalable web applications. The author's own development of a framework for scalable and highly loaded applications of any level of complexity, the features of the proposed models are presented, and the architecture of the Minegraph framework is described in detail, the features of this framework are analysed, and its strengths and weaknesses are highlighted.

Conclusions. The article analyses the existing frameworks for developing web applications. The author's development of the Mineraph framework architecture is presented in detail, namely: drivers for interaction with databases, NoSQL databases; features of the model to allow adding or changing the basic logic for processing requests; models where the logic for processing requests is stored and its features; issues related to a flexible mechanism for debugging the basic logic of applications are highlighted.

The created framework is a ready-made structural framework for the development of scalable and highly loaded applications of any level of complexity. It is important to note that development on this framework does not require in-depth analysis of web system architecture, but only basic knowledge of the development environment and the ability to navigate the documentation. This development framework takes into account the advantages of previous analogues and contains advanced technologies, which increases the productivity and efficiency of developing scalable and highly loaded web applications. The paper describes the features of the Minegraph framework model, analyzes the features of this framework,

and identifies its strengths and weaknesses. The Minegraph framework goes beyond structural capabilities, seamlessly integrates and uses the latest technologies for data processing and storage. In addition, it also uses cloud solutions, which leads to a significant improvement in system functionality while reducing infrastructure management costs.

The framework presented in this paper is not just a structural framework; it is an extremely versatile and adaptive tool, ready to accommodate applications of the highest complexity and scale. This framework has advanced technologies for data processing and storage and also uses cloud solutions that help improve the system's functionality and reduce the cost of its infrastructure.

Keywords: framework; framework architecture; web applications; web framework; event-driven model; business logic; business rules; scalability; transaction; database.

04.08.2023