Таким чином, архітектура сучасного чат-бота формується як модульна програмна структура, що поєднує інтерфейсні компоненти, механізми обробки природної мови, систему керування діалогом та підсистеми доступу до даних і зовнішніх сервісів. Використання таких архітектурних принципів забезпечує масштабованість, спрощує розвиток функціоналу та дозволяє створювати ефективні діалогові системи для веб-сервісів, мобільних застосунків і корпоративних інформаційних платформ [1,2].

Список використаних джерел

1. Ciesla R. The Book of Chatbots: From ELIZA to ChatGPT. Springer Nature, 2024. 159 c.
2. Skuridin A., Wynn M. Chatbot Design and Implementation: Towards an Operational Model for Chatbots. Mdpi. 17.04.2024. URL:https://www.mdpi.com/2078-2489/15/4/226 (дата звернення: 15.03.2026).
3. Aleedy M., Atwell E., Meshoul S. A Multi-Agent Chatbot Architecture for AI-Driven Language Learning. Mdpi. 01.10.2025. URL: https://www.mdpi.com/2076-3417/15/19/10634 (дата звернення: 15.03.2026).

УДК 004.738.5

РОЗРОБКА ФРОНТЕНДУ ЗА ДОПОМОГОЮ РІЗНИХ ФРЕЙМВОРКІВ: ПЕРЕВАГИ, НЕДОЛІКИ ТА ЗАВДАННЯ

Стеценко Я. І.
yanastetforcomp@gmail.com
Черкаський державний фаховий бізнес-коледж
Подорошко Д. І.
м. Черкаси, Україна

Сучасна розробка фронтенду є фундаментальною складовою створення прикладних рішень, що відповідають високим вимогам користувачів щодо швидкодії, масштабованості та безпеки. Еволюція вебтехнологій призвела до домінування односторінкових додатків (SPA), які забезпечують миттєву реакцію

без перезавантаження сторінок. Ці рішення базуються на JavaScript та TypeScript. Вибір інструменту є стратегічним рішенням, що впливає на життєвий цикл продукту. Індустрія сформувала чітку трійку лідерів: React, Vue.js та Angular, кожен з яких має унікальну архітектуру та специфічні сфери застосування, які потребують детального технічного аналізу.

Технологія React, бібліотека для створення користувацьких інтерфейсів, підтримується Meta. Її ключова інновація – використання Віртуального DOM (Virtual DOM). React обчислює різницю між поточним та новим станом (алгоритм узгодження) і точково оновлює лише необхідні елементи реального DOM, що сприяє підвищенню продуктивності у складних інтерфейсах.

Суворий компонентний підхід дозволяє розбивати інтерфейси на дрібні, незалежні та придатні для повторного використання блоки коду. Це значно спрощує командну розробку, тестування та рефакторинг. Величезна спільнота та широка екосистема роблять React універсальним інструментом.

Серед недоліків: високий поріг входження через відсутність суворої стандартизації архітектури «з коробки» (розробникам доводиться самостійно обирати інструменти для маршрутизації та управління станом, наприклад, Redux або Zustand). Часті оновлення парадигм, зокрема перехід до функціональних компонентів з хуками, вимагають постійного навчання та адаптації існуючого коду. React використовують для створення інтерактивних дашбордів, розгалужених платформ електронної комерції, соціальних мереж та стрімінгових сервісів. Наприклад, під час розробки проєкту MusicHub компонентний підхід React дозволив команді ефективно реалізувати динамічний каталог нотного матеріалу з миттєвим пошуком та створити безперебійний процес оформлення підписки, забезпечивши високу стабільність платформи. [1]

Vue.js – це прогресивний фреймворк, розроблений для об'єднання найкращих архітектурних рішень та створення максимально зрозумілого та гнучкого продукту. Він характеризується надзвичайно адаптивною системою інтеграції, дозволяючи використовувати його як легку бібліотеку для

інтерактивності на окремих сторінках, або як повноцінний фреймворк для складних корпоративних додатків (із Vue Router та Pinia).

Найсильнішою стороною є реактивна система збору залежностей, яка працює практично непомітно для розробника, автоматично відстежуючи зміни в даних та миттєво оновлюючи інтерфейс. Розробники високо оцінюють парадигму однофайлових компонентів (JavaScript, HTML, CSS в єдиному файлі), що підвищує візуальну зрозумілість структури проєкту. Слабкою стороною традиційно виділяють дещо меншу популярність у великому корпоративному секторі порівняно з конкурентами, хоча ситуація швидко змінюється. Vue.js часто застосовується для розробки фронтенд-інтерфейсів сучасних фінансових сервісів та цифрових гаманців. Висока швидкість рендерингу, легкість у підтримці коду та зручність управління складним фінансовим станом дозволяють створювати надійні клієнтські рішення, які витримують значні навантаження. [2]

Angular – це комплексний фреймворк, який розробляється та підтримується корпорацією Google. На відміну від React та Vue, Angular пропонує філософію повної комплектації, містячи вбудовані стандартизовані інструменти для всіх можливих завдань веброзробки: від маршрутизації та роботи з формами до HTTP-запитів та глибокого модульного тестування. Базовою мовою програмування є TypeScript, що забезпечує сувору типізацію коду, використання об'єктно-орієнтованих патернів та виявлення архітектурних помилок ще на етапі компіляції.

Архітектура Angular суворо базується на концепціях модульності, сервісів та впровадження залежностей (Dependency Injection), що робить його ідеальним вибором для великих інженерних команд, де критично важливо дотримуватися єдиного суворого стилю написання коду. Angular характеризується складнішим порогом входження, оскільки для продуктивної роботи розробникам необхідно глибоко опанувати концепції реактивного програмування за допомогою бібліотеки RxJS. Монолітність та великий початковий розмір кінцевого пакета часто роблять його технічно невиправданим для невеликих проєктів. Основна

ніша застосування Angular – розробка масштабних корпоративних систем управління, складних CRM та банківських порталів, що мають високі вимоги до стабільності, прогнозованості та довготривалої підтримки. [3]

Окрім вибору безпосередньо фреймворку, сучасна розробка фронтенду вимагає врахування архітектури безпеки, зокрема, впровадження концепції нульової довіри (Zero Trust) на рівні клієнтського додатка. Фронтенд-розробники повинні забезпечувати надійне зберігання авторизаційних даних, правильне налаштування політик спільного використання ресурсів та безпечну обробку токенів сесій. Хоча сучасні версії фреймворків автоматично екранують дані, мінімізуючи ризики міжсайтового скриптингу (XSS), розвиток технологій штучного інтелекту стимулює появу нових загроз. Тому вибір сучасного та регулярно оновлюваного фреймворку є критичною необхідністю для забезпечення стабільності інформаційних систем та захисту даних користувачів.

Детальний технічний аналіз сучасного інструментарію веброзробки підтверджує, що вибір фронтенд-технології залежить від специфіки конкретного продукту, вимог до його довгострокового масштабування та наявного досвіду команди:

- React залишається найбільш універсальним та гнучким вибором для більшості динамічних платформ, де ключову роль відіграє насичена взаємодія з користувачем та швидкість відмальовування компонентів.
- Vue.js приваблює структурною елегантністю, відносно низьким порогом входження та простотою поступової інтеграції, що робить його затребуваним при створенні цифрових гаманців та легких комерційних сервісів.
- Angular міцно широко застосовується для побудови корпоративних інформаційних систем найвищої складності, де пріоритетами виступають сувора типізація, жорстка прогнозованість архітектури та наявність єдиного стандартизованого набору інструментів.

Практика розробки підтверджує, що грамотний та обґрунтований вибір технологічного стека дозволяє не лише значно оптимізувати процеси

програмування, але й гарантувати створення надійного, стійкого до сучасних кіберзагроз та максимально зручного інтерфейсу, що відповідає найвищим стандартам якості програмного забезпечення.

Список використаних джерел:

1. React – A JavaScript library for building user interfaces. Official Documentation. URL: https://react.dev/.
2. MDN Web Docs: Front-end web developer. Mozilla Corporation. URL: https://developer.mozilla.org/en-US/docs/Learn/Front-end_web_developer
3. OWASP Top 10 Client-Side Security Risks. Open Worldwide Application Security Project. URL: https://owasp.org/www-project-top-10-client-side-security-risks/.
4. Web Security Guidelines. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/Security.
5. Osmani A. Learning JavaScript Design Patterns: A JavaScript and React Developer's Guide. 2nd Edition. O'Reilly Media, 2023. 280 p.

УДК 004.41, 004.75

ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ РОЗПОДІЛЕНИХ СИСТЕМ КОНТРОЛЮ ВЕРСІЙ

Базюк В.Р
makamagol@gmail.com
Черкаський державний фаховий бізнес-коледж
Марченко С. В.
Черкаси, Україна

Розподілені системи контролю версій (Distributed Version Control Systems, DVCS) є критично важливим класом інженерного програмного забезпечення, що поєднує збереження історії змін, асинхронну реплікацію, цілісність даних і підтримку паралельної розробки. Сучасні системи цього класу, зокрема Git, Mercurial, Fossil і Darcs, реалізують подібні базові функції, але істотно