МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧОРНОМОРСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ПЕТРА МОГИЛИ

Морозов Костянтин Юрійович

УДК 004.89

МЕТОДИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ WEB-ПРОЕКТУВАННЯ НА ОСНОВІ БЕЗСЕРВЕРНОЇ АРХІТЕКТУРИ

122 – Комп’ютерні науки

Автореферат
магістерської наукової роботи на здобуття освітньої кваліфікації
«Магістр комп’ютерних наук»

Миколаїв – 2020

Магістерська наукова робота є рукопис.

Робота виконана в Чорноморському національному університеті імені Петра Могили Міністерства освіти і науки України на кафедрі інтелектуальних інформаційних систем

Науковий керівник: к.т.н., доцент, доцент кафедри інтелектуальних інформаційних систем Сіденко Євген Вікторович

Рецензент: д.т.н., доцент, доцент кафедри інженерії програмного забезпечення Давиденко Євген Олександрович

Захист відбудеться «___» лютого 2020 р. о $9^{\underline{00}}$ год. на засіданні екзаменаційної комісії (ауд. 2-403) у Чорноморському національному університеті імені Петра Могили за адресою: 54003, м. Миколаїв, вул. 68-ми Десантників, 10.

З магістерською науковою роботою можна ознайомитися в бібліотеці Чорноморського національного університету імені Петра Могили за адресою: 54003, м. Миколаїв, вул. 68-ми Десантників, 10.

Автореферат представлений «___» лютого 2020 р.

Секретар
екзаменаційної комісії,

к. пед. н., доцент Н. М. Болюбаш

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність роботи визначається високим рівнем розвитку та поширенням різноманітних веб-технологій, веб систем, а також методів та підходів до розробки даних систем.

Метою магістерської наукової роботи є аналіз ефективності сучасних та популярних підходів до веб-проектування та створення нового на їх основі.

Предметом дослідження є методології та практики розробки та проектування веб-застосунків та систем, а також архітектурні та функціональні рішення, що використовуються в enterprise-розробці.

Об'єкт дослідження – способи підвищення ефективності веб-проектування та розробки за допомогою backendless-обчислень.

Магістерська наукова робота складається зі вступу, 4 розділів, висновків, додатків. Загальний обсяг роботи складає __ сторінки, __ рисунків, __ таблиць та _ посилань на літературні джерела.

АПРОБАЦІЯ

1. Морозов К. Ю., Сіденко Є. В. Методи підвищення ефективності web-проектування на основі безсерверної архітектури. Могилянські читання – 2019: тези доповідей: Комп’ютерні науки. Технічні науки, Миколаїв: ЧНУ, 11-16 листопада, 2019. – С. 40-42.
2. Морозов К. Ю., Сіденко Є. В. Дослідження та аналіз підходів до проектування Web-систем. Інтелектуальні інформаційні системи – 2020: тези доповідей, Миколаїв: ЧНУ, 28-31 січня, 2020. – С. 92-94.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У **вступі** магістерської наукової роботи обґрунтовано актуальність обраної теми, сформульовано мету і задачі дослідження, визначено предмет та об’єкт, а також окреслено короткий синопсис предметної області та проводиться короткий огляд поставленої задачі.

В **першому розділі** наведено аналіз предметної області, дано визначення таким поняттям як веб-проектування та архітектура системи. Окреслено види архітектур і архітектурних моделей, розглянуто дві популярні моделі:

1. Клієнт-сервер – архітектура, яка є на даний момент найпоширенішою в розробці та проектуванні веб-систем. Її суть полягає в тому, що система складається з двох головних частин: клієнт і сервер. Клієнтом є підсистема або окремий комп’ютер, який виконує запити до сервера з метою отримання певних даних. Сервер – виділений комп’ютер або програмне забезпечення, яке виконує пошук та обробку даних, а також відправлення цих даних до клієнту (а також багато іншого).
2. Peer-2-peer – архітектура або мережа, в якій всі учасники/компоненти є рівноцінними та рівноправними, тобто кожен вузол може приймати на себе функції як клієнта, так і сервера. Проте відмова одного з компонентів мережі може негативно вплинути на всю систему, навіть повністю вивести її з ладу.

В **другому розділі** було проведено огляд вже існуючих на момент написання роботи архітектурних рішень:

1. *Монолітна архітектура* – суть даної архітектури полягає в чіткому розділенні системи на клієнт і сервер, при цьому сервер є єдиним і неподільним блоком, який всю свою функціональність тримає в одному модулі (компоненті). Сильними сторонами цього підходу є простота в реалізації і, як наслідок, прискорення розробки. Також систему, побудовану на основі даної архітектури, просто розгортати і

масштабувати. Мінусами є дуже погана гнучкість системи, через що будь-яке внесення змін в дану систему потребуватиме велику кількість часових та людських ресурсів, перекриваючи її сильну сторону – швидкість розробки.

2. *Багатошарова архітектура* – дане архітектурне рішення є продовженням ідеї монолітної архітектури, але замість єдиного блоку з серверною частиною, який є так званим «монолітом», з'являється більш гнучка підсистема, яка буде більш охоча до змін і буде поділена в свою чергу на три або більше підмодулі. Традиційно такі підмодулі виділяють як:
    a. *Data layer* (шар роботи із даними) – підмодуль, який тримає в собі функціональність по роботі з джерелом даних (наприклад, базою даних або хмарним сховищем).
    b. *Business layer* (шар бізнес-логіки) – підмодуль, який займається реалізацією так званої бізнес-логіки, або безпосередніх потреб замовника.
    c. *Presentation layer* (шар представлення) – підмодуль, який займається виведенням користувацького інтерфейсу для роботи з системою. Це може бути як окремий застосунок, так і набір контролерів (MVC, Django).
3. Мікросервісна архітектура – в основі даної архітектури лежить так званий сервіс-орієнтований підхід, який заснований на ідеї спілкування між собою віддалених підсистем (наприклад, за допомогою синхронних та асинхронних HTTP-запитів). В даному підході «мікросервісами» є невеликі підмодулі, які зберігають в собі мінімально необхідну функціональність як з точки зори логіки роботи, так і з точки зору бізнес-вимог. Тобто, частина функціональності може бути виокремлена від одного мікросервісу в інший як за семантичними ознаками, так і за вимогами замовника. В простому випадку такі мікросервіси спілкуються між собою за допомогою HTTP-запитів та зберігають дані в одній базі

даних в різних таблицях/колекціях. Проте *Microsoft* рекомендує застосовувати для подібних систем асинхронний підхід, заснований на *event-based* спілкуванні між сервісами. Це означає, що один сервіс буде відправляти іншому не HTTP-запит, а подію (Kafka, EventGrid, RabbitMQ), а відповідний мікросервіс буде певним чином цю подію обробляти і зберігати дані в свою власну базу даних (а не в одну спільну). Перевагами такої архітектури і такого підходу полягають в тому, що кожен мікросервіс працює окремо від всіх інших, і при відмові одного чи більше, система в цілому не буде давати збій, також така архітектура є доволі гнучкою, проте порівняно дорогою та складною в обслуговуванні та розгортанні.

4. Serverless-архітектура – це архітектура, яка продовжує ідею атомаризації підсистем і доводить її до межі – замість окремого сервіса, який спілкується з іншими, даний підхід представляє окремі функції, які викликаються так само, як і інші кінцеві точки в мережі. Такий підхід позбавляє розробників від необхідності підтримувати і активно займатися розгортанням системи, так як це візьмуть на себе постачальники послуг (Microsoft, Amazon). Проте мінусами даного підходу є ціна, яка росте паралельно росту системи, а також низька швидкодія, якщо порівнювати її з попередніми підходами;
5. Backend as a service (BaaS) – дана технологія не є архітектурним підходом в повній мірі, а представляє собою рішення, направлене на позбавлення розробників необхідності писати власну серверну частину, замість якої буде надано серверна частина від постачальника послуг. Явними перевагами такого рішення є, як було сказано, відсутність в необхідності розробляти і проектувати власну серверну частину, цим самим прискорюючи розробку. Проте необхідність в використанні BaaS доволі обмежена, так як більшість систем проектуються і створюються власноруч, замість того, щоб довіряти таку роботи третім сторонам. BaaS не може гарантувати повне покриття бізнес-вимог по швидкодії та

відмовостійкості, а також відсутній доступ до бази даних (що в умовах enterprise-розробки суттєвий недолік), а також майже неможливе внесення змін (пізні BaaS-системи підтримують впровадження користувацького коду, проте це суперечить самій ідеї BaaS). Окрім цього, подібні послуги мають доволі високу вартість, яка залежить насамперед від вимог розробників.

Також в розділі було проведено порівняння даних архітектурних підходів з метою виявлення найбільш оптимальних рішень для різних умов початку роботи над системою.

В **третьому** розділі дипломної роботи було описано вибір засобів реалізації програмної частини дипломної роботи. Для цієї задачі було обрані дві наступні технології:

1. Мова програмування TypeScript. Мова програмування, яка використовується як для клієнтської розробки, так і для серверної розробки з використанням NodeJS. В порівнянні з традиційним JavaScript, TypeScript має строгу типізацію та систему класів-модулів. Було додано сотні вбудованих типів для підтримки комфортної розробки та підтримки веб-систем.
2. СКБД MongoDB. MongoDB є NoSQL базою даних, яка заснована на форматі JSON, який є нотацією для JavaScript-об’єктів. MongoDB оперує колекціями та документами, замість традиційних таблиць та записів. Приводом для обрання даної СКБД була хороша сумісність з мовами TypeScript/JavaScript.

В **четвертому** розділі дипломної роботи було описано створений підхід до проектування веб-систем, який отримав назву **backendless** (бек-енд – частина системи, що зазвичай) відповідає за серверну частину. Тобто, виходячи з назви, backendless – архітектурний підхід, що передбачає позбавлення від серверної частини і винесення частини логіки на клієнтську частину, а частини – на базу даних (збережені процедури в SQL, скрипти для MongoDB). Це зовсім позбавить

розробників необхідності писати серверну частину, тобто всі ресурси (як фінансові, так і людські) можуть бути або секономлені, або відправлені на іншу роботу.

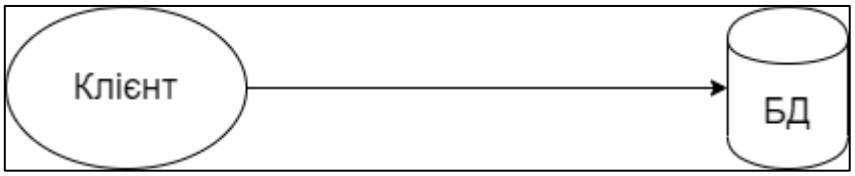

Рис. 1 – схема backendless-архітектури

Досягається подібний ефект від застосування так званого підходу **DADO (Direct Approach to Data Obtaining)**, або «прямий підхід до доступу до даних». Мається на увазі, що дані для певної системи можуть бути отримані напряму, без застосування окремих виділених серверів. Це в рази прискорить саму розробку та спростить підтримку та розгортання системи, яка побудована на основі backendless-обчислень.

В рамках роботи для прикладу було розглянуто DADO-доступ до бази даних MongoDB. Ця база є NoSQL рішенням, що має JSON-подібний синтаксис. Для застосування DADO-підходу для бази даних, необхідно створити окрему клієнтську підсистему, яка за допомогою розробленого в рамках роботи JavaScript-фреймворку буде здійснювати запити до бази даних за допомогою спеціальних структур даних (в майбутньому це буде реалізовано за допомогою окремого протоколу передачі даних).

Запити формуються на клієнтській стороні та представляють собою конструкції форми JavaScript, які під час запиту в runtime будуть представлені у вигляді mongo-запитів, наприклад:

```
db.users.find({ age: { $gte: 18 } });
```

Даний запит дістає із колекції *users* всіх користувачів, вік яких більший, або дорівнює вісімнадцяти.

На мові JavaScript з backendless такий запиту виглядатиме наступним чином:

```
users.filter(user => user.age >= 18);
```

Суть даного запису полягає в тому, що він зовсім ніяк не відрізняється від стандартних конструкцій мови JavaScript. Це ілюструє наступну перевагу backendless – простий інтерфейс для розробників. Тобто фронт-енд розробники, які працюватимуть над backendless-системою, можуть зовсім нічого не знати про роботу з базою MongoDB, проте завдяки простому інтерфейсу робота з базою даних буде простою і не буде відрізнятися від іншої повсякденної роботи фронт-енд прозробника.

Повний запит для отримання користувачів, чий вік більший, або дорівнює вісімнадцяти:

```
const backendless = new BackendlessService();
const request = backendless.withDb('127.0.0.1', 'test');
const users = request.withCollection<User>('users')
                     .getValues();

users = users.filter(user => user.age >= 18);
this.users = await request.execute<User[]>(users);
```

В даному уривку коду відбувається наступне:

1. Створюється екземпляр классу *BackendlessService*, який зберігає в собі всю головну функціональність;
2. Відкривається запит до базу даних. На цьому етапі ініціалізується backendless-запит, який далі буде відправлено до бази;
3. Отримується значення всієї колекції users, яка зберігає дані про користувачів;
4. Фільтрується колекція по критерію повноліття користувача (вік більший або рівний 18);
5. Виконання запиту.

Варто підмітити, що отримання колекції в пункті 3 не являється операцією самою по собі, тобто база не буде окремо завантажувати у пам'ять всіх користувачів, такий запис в JavaScript лише позначає, з якою колекцією розробник надалі буде проводити певні маніпуляції.

На даному етапі розвитку технології доступні такі маніпуляції з даними:

1. getValue – отримання одного значення з колекції.
2. getValues – отримання всіх значень з колекції.
3. addValue – додати одне значення в колекцію.
4. addValues – додати масив значень в колекцію.
5. replaceValue – замінити одне значення в колекції.
6. replaceValues – замінити кілька значень в колекції.
7. replaceValuesWithConditions – замінити кілька значень в колекції, кожне значення – по окремій умові.
8. deleteValue – видалити перше входження елементу в колекції.
9. deleteValues – видалити всі входження елементу в колекції.

Для кожного із цих методів доступна більш детальна документація, з більш детальним описом функціональністю цих функцій, в тексті дипломної наукової роботи.

Також однією із переваг backendless є ціна. На відміну від мікросервісів, serverless та BaaS, дане архітектурне рішення є цілком безкоштовним. Від розробників не буде потрібно витрачати ресурси на допоміжні засоби, так як все досягається засобами backendless, необхідно витрачатися лише на базу даних, а JavaScript-фреймворк поширюється за ліцензією MIT.

Виходячи із попереднього пункту, варто також зазначити, що backendless не лише не потребуватиме затрат коштів, а і взагалі буде економією для будь-якого проекту, особливо для стартапів та невеликих проектів (про це більш детально в дипломній роботі).

Проте в backendless є і недоліки. По-перше, це недостатньо високе оснащення функціональністю програмної частини архітектури. В рамках

магістерської роботи розглянуто лише перспективу backendless-роботи з базою даних, проте для більшості enterprise-продуктів необхідні також додаткові рішення (кешування, збір статистики, агрегація даних). Проте даний недолік є недоліком лише наполовину, так як в перспективі можливий перехід на backendless для великої частини затребуваних технологій. Другий недолік – необхідність винесення бізнес-логіки з серверу на клієнт/базу даних. Для певної категорії продуктів даний недолік може виявитися критичним, так як збільшується кодова база клієнтської підсистеми, а це означає, що збільшиться необхідний розмір файлів JavaScript, який буде завантажений браузером користувача. Окрім того, порушується принцип розділення функціональності, тобто клієнтська підсистема повинна відповідати лише за логіку відображення, а за бізнес-логіку відповідатиме сервер. Проте цим можна знехтувати, якщо винести частину логіки в движок бази даних. Останній недолік є лише обмеженням: для роботи з backendless необхідне відокремлення підсистеми клієнта від підсистеми сервера не лише в логічному сенсі, а і в апаратному. Це означає, що «клієнт» і «сервер» мають бути двома різними програмними сутностями. Наприклад, сервер написаний на ASP .NET Core, Django, Spring, а клієнт – на ReactJs, Angular або VueJS.

У **спеціальній частині** магістерської наукової роботи з «Охорони праці та безпеки в надзвичайних ситуаціях» розглянуто основні засади безпеки роботи в типових умовах для представників IT-галузі – офісні приміщення, промислові комплекси, а також безпека роботи за комп’ютером. Крім того було проаналізовано та освітлено санітарні умови на підприємстві «Global Logic Україна». В результаті аналізу було зроблено висновок про комфортні умови праці, проте наявні проблеми з матеріальним оснащенням (стільці та крісла), які не відповідають ортопедичним нормам, а також часта зайнятість конференц-залів через високу насиченість офісу компанії в м. Миколаєві. Також було наведено детальний інструктаж щодо дій в надзвичайних ситуаціях.

У **методичній частині** розроблено плани практичних робіт на теми «Проектування web-систем».

В **висновках** було проведено короткі висновки по кожним технологіям та аспектам, які були розглянуті в межах роботи, було окреслено основні положення технології backendless, її переваги та недоліки, а також викладено перспективи розвитку.

ЗАГАЛЬНІ ВИСНОВКИ

В даній магістерській науковій роботі було розглянуто та проаналізовано основні на момент написання засоби, технології та архітектури для проектування та розробки веб-систем.

В першому розділі було наведено основну інформацію по предметній області, розглянуто поняття архітектури та розглянуто популярні підходи до проектування.

В другому розділі було розглянуто та проаналізовано такі засоби як: монолітна архітектура, багатошарова архітектура, мікросервісна архітектура, serverless-архітектура, а також BaaS, було проаналізовано їх сильні та слабкі сторони, проаналізовані способи та випадки використання.

В третьому розділі було описано головні положення розробленої технології backendless, розглянуто її переваги і недоліки, дано рекомендації щодо її використання, окреслено перспективи розвитку.

В спеціальній частині здійснено аналіз умов праці в миколаївському офісі компанії «Global Logic» та розроблено план дій у разі надзвичайних ситуацій. В методичній частині розроблено практичні роботи із суміжної до магістерської роботи теми: «Проектування web-систем».

У висновках було підбито підсумки досліджень, проведено короткий опис розглянутих раніше технологій, окреслено перспективи розвитку для backendless.

АНОТАЦІЯ

до магістерської наукової роботи

«Методи підвищення ефективності web-проектування на основі безсерверної архітектури»

Студент: Морозов К. Ю.

Керівник: канд. техн. наук, доц. Сіденко Є. В.

Дана магістерська наукова робота присвячена аналізу технологій та архітектурних підходів до проектування веб-систем.

Об’єкт роботи – способи підвищення ефективності веб-проектування та розробки за допомогою backendless-обчислень.

Предмет роботи – методології та практики розробки та проектування веб-застосунків та систем, а також архітектурні та функціональні рішення, що використовуються в enterprise-розробці.

Мета роботи - аналіз ефективності сучасних та популярних підходів до веб-проектування та створення нового на їх основі.

Практичне значення даної теми полягає в підвищенні ефективності проектування та розробки веб-систем різної степені складності та завантаженості.

Дипломна наукова робота складається з фахового розділу, спеціальної частини з охорони праці та безпеки у надзвичайних ситуаціях та методичної частини.

Пояснювальна записка магістерської роботи складається із вступу, чотирьох розділів, висновків та додатку.

У вступі визначається актуальність теми, визначаються тема та мета роботи, проводиться огляд поставленої задачі.

У першому розділі проводиться огляд предметної області, визначаються основні поняття та розглядається основні моделі архітектур систем.

У другому розділі проводиться детальний огляд існуючих підходів та архітектур, які використовуються в проектуванні та розробці веб-систем на момент написання роботи.

У третьому розділі наводиться список використаних для магістерської наукової роботи технологій.

У четвертому розділі визначаються основні положення створеної в рамках роботи архітектури backendless, визначаються її переваги та недоліки в порівнянні з розглянутими раніше підходами, окреслюються перспективи розвитку.

У висновках проводиться короткий огляд раніше розглянутих підходів, а також проводиться аналіз виконаної роботи.

В спеціальній частині з охорони праці окреслюються умови праці та дії в надзвичайних умовах в компанії «Global Logic Україна».

В цілому, магістерська наукова робота без додатків містить __ сторінок, __ рисунків, __ таблиці.

ABSTRACT

to the master's scientific work

« Methods of increasing of web-designing effectiveness based on backendless architecture»

Student: Morozov K. Y.

Research Manager: Ph. D., Associate Professor Sidenko E. V.

This Master's work is devoted to the analysis of technologies and architectural approaches to web systems design.

Object of Work - ways to improve web design and development performance through backendless computing.

The subject of work is the methodologies and practices of developing and designing web applications and systems, as well as architectural and functional solutions used in enterprise-development.

The purpose of the work is to analyze the effectiveness of modern and popular approaches to web designing and create new ones based on them.

The practical significance of this topic is to increase the efficiency of designing and developing web systems of varying degrees of complexity and workload.

The diploma scientific work consists of a specialty section, a special part on occupational safety and security in emergency situations and a methodical part.

The master's explanatory note consists of an introduction, four chapters, conclusion, and an appendix.

The introduction determines the relevance of the topic, defines the theme and purpose of the work, provides an overview of the task.

The first section examines the subject area, defines the basic concepts, and discusses basic models of system architectures.

The second section provides a detailed overview of the existing approaches and architectures used in the design and development of web systems at the time of writing.

The third section describes the list of technologies used for the master science work.

The fourth section defines the basic provisions created in the framework of backendless architecture, defines its advantages and disadvantages compared to the approaches previously considered, outlines the prospects for development.

The conclusions provide a brief overview of the approaches previously considered, as well as an analysis of the work done.

The special section on labor protection outlines working conditions and emergency response at Global Logic Ukraine.

In general, a master's thesis without appendices contains __ pages, __ drawings, __ tables.