УДК 004.89

ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНИХ СИСТЕМ, В ТОМУ ЧИСЛІ, З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Зінченко Д.О.
Науковий керівник – професор кафедри системотехніки, кандидат технічних наук, доцент, Міщеряков Ю.В.
Харківський національний університет радіоелектроніки, каф. СТ
м. Харків, Україна
тел.: +38(066) 708-54-91, e-mail: dmytro.zinchenko@nure.ua

This work is devoted to the problems of supporting various applications and scalability on the Internet. The project architecture for creating complex projects was considered, which would allow to obtain a flexible system, with the possibility of dividing the program into components. A short list of features of the solution implementation is given and the advantages that can be gained in this way are described.

За весь час існування інтернету відбулась не мала кількість змін, а разом з цим й різні підходи до вирішення проблем. Зараз набираються оберти різні види діяльності, такі як промисловість, сільське господарство, торгівля, фінанси, які можуть ефективно використовувати ресурси інтернету для свого зростання, охоплюючи тим самим велику аудиторію.

У зв’язку з цим велика кількість користувачів потребує значної кількості обчислювальних ресурсів та можливість їх використовувати. Рішенням такого питання є використання клієнт-серверної архітектури. Це концепція інформаційної мережі, в якій основна частина її ресурсів зосереджена на кількох серверах, обслуговуючих своїх клієнтів.

Існують проєкти невеликої складності та ті, що потребують великої кількості функцій та непростої логіки. Розглянемо більш складну архітектуру, де потрібна чітка організація коду та забезпечення масштабованості. Такий підхід характерний для розробки масштабних робіт, соціальних мереж чи комерційних проектів.

Для вирішення таких задач підходить трирівнева архітектура веб-проєктування. Для такої архітектури характерно три логічні рівні: рівень доступу до даних, рівень бізнес-логіки(backend), рівень представлення даних(frontend). Така структура зручна для розробки та підтримки.

Верхній рівень представлення, з яким взаємодіє безпосередньо користувач, включає компоненти UI інтерфейсу, елементи введення даних. Для реалізації рівня можна використати ASP.NET MVC, де створюються компоненти представлення даних, контролери для обробки API запитів.

Прикладний рівень, бізнес-логіки, визначає функціональність програми. Взаємодіє з іншими рівнями системи, з рівнем представлення та

рівнем доступу до даних. Отримані дані з бази даних обробляються та передаються на рівень представлення і навпаки, це дозволяє розділяти обов'язки та знижує залежність між рівнями. Реалізується бібліотеками класів.

Рівень доступу до даних, описує як зберігаються дані та доступні для використання на рівнях вище, забезпечує цілісність та безпеку. Тут містяться класи моделей, клас контексту бази даних. Також тут знаходиться репозиторій, через який рівень бізнес-логіки взаємодіє з базою даних. Зазвичай на цьому рівні використовуються реляційні бази даних, найбільш поширений тип.

Незалежно від цього виділяють ще фізичні рівні, це сервер бази даних, веб-застосунок та веб-браузер користувача. Якщо у якості UI використовується мобільний застосунок, то це ще один фізичний рівень. Фізичні рівні не співпадають з логічними. Це також показує те, що клієнт-серверна архітектура дозволяє створювати окремі застосунки, які можуть виконувати свої функції та зручні в користуванні.

Таким чином основна ідея архітектури клієнт-сервер полягає в поділі мережевого застосунку на кілька компонентів, кожен з яких реалізує специфічний набір сервісів. Компоненти такого застосунку можуть виконуватися на різних комп'ютерах, виконуючи серверні або клієнтські функції. Це дозволяє підвищити надійність, безпеку і продуктивність мережевих застосунків і мережі в цілому. Використовуючи трирівневу архітектуру проєктування ми отримуємо гарне рішення для розробки складних проєктів. Можливості підходу дозволяють зосередитись на окремому шарі всієї структури, для виявлення та виправлення помилок. Розробник з легкістю може використати нову реалізацію та заміняти вже наявні частини коду. Високий рівень безпеки, доступ до даних через проміжний рівень логіки.

Список використаних джерел:

1. Клієнт-серверна архітектура. https://training.qatestlab.com/blog/technical-articles/client-server-architecture/.

2. Архітектура клієнт-сервер. http://inter.ptngu.com/kompyuterni-merezhi/arhitektura-kliyent-server.

3. Клієнт-серверна архітектура та ролі серверів. https://medium.com/@IvanZmerzlyi/