УДК 004.9

ВЕБ-ДОДАТОК УПРАВЛІННЯ ТОВАРНИМИ ЗАПАСАМИ НА ОСНОВІ ТРИРІВНЕВОЇ АРХІТЕКТУРИ

Никуляк Артем, Месюра Володимир

Вінницький національний технічний університет

Анотація

Здійснено аналіз трирівневої архітектури програмного забезпечення у проектуванні веб-додатків. Описано структуру, функціональні можливості рівнів та переваги архітектури для проектування і розробки веб-додатку управління товарними запасам.

Abstract

Was performed the analysis of three-tier software architecture in the design of web-applications. The structure, functional capabilities of levels and advantages of architecture for designing and developing inventory management web application was described.

Вступ

При проектуванні програмного рішення основним кроком є вибір архітектури. Вибір архітектури системи здійснюється з врахуванням цілей системи, її вхідних і вихідних даних. Вдало підібрана архітектура може забезпечити зручність масштабування і підтримки спроектованого програмного забезпечення та зменшити час, необхідний для його розробки. Такі можливості для веб-додатку управління товарними запасами надає трирівнева архітектура.

Результати дослідження

Веб-додаток управління товарними запасами підприємства розробляється для підприємства продажу комп'ютерної техніки. Програмний модуль надасть зручні інструменти для здійснення управління товарами підприємства, відстежування гарантійних термінів товарів. Створення та відстежування заявок на гарантійне обслуговування, зміна їх статусів після виконання. Ці процеси можуть бути усішно впроваджені у веб додатку на основі трирівневої архітектури.

Трирівнева архітектура (англ. three-tier або Multitier architecture) – це тип архітектури програмного забезпечення, що складається з трьох "ярусів" або "шарів" логічного обчислення. Вони часто використовуються в програмах як специфічний тип клієнт-серверної системи. Трирівнева архітектура забезпечує багато переваг для середовища та процесу розробки шляхом розбиття на модулі користувальницького інтерфейсу, бізнес-логіки та зберігання даних. Це дає більше можливостей для груп розробників, дозволяючи їм оновлювати певну частину програми незалежно від інших частин. Ця додаткова гнучкість може поліпшити загальний час виходу на ринок і скоротити час розробки, надаючи командам розробників можливість заміни або модернізації незалежних рівнів, не змінюючи інших частин системи.

Ця архітектурна систем часто використовується для вбудовування та інтеграції стороннього програмного забезпечення в існуючу програму. Ця інтеграційна гнучкість також робить систему зручною для вбудовування програмного забезпечення для аналітики, і з цієї причини часто використовуються постачальниками аналітичних засобів. Трирівневі архітектури часто використовуються у хмарних або локальних додатках, а також у програмних сервісах (SaaS) [1].

Веб-додаток управління товарними запасами на основі трирівневої архітектури складаються з наступних рівнів:

– Presentation Tier. Рівень презентації – це шар інтерфейсу в трирівневій системі, що визначає інтерфейс користувача. Цей користувальницький інтерфейс є графічним, доступним через веб-браузер або веб-додаток і відображає вміст і інформацію для користувачів системи. Цей рівень часто побудований на веб-технологіях HTML5, JavaScript, CSS, і виконує доступ до інших шарів через виклики API.

– Application tier**.** Рівень застосування містить функціональну бізнес-логіку, яка виконує основні операції керування товарами у веб-додатку. Цей рівень розробляється за допомогою мови програмування C # .NET.

– Data tier. Рівень даних складається з системи збереження та доступу до даних і бази даних підприємства. Доступ до даних здійснюється на рівні додатків через виклики API.

Структуру веб-додаку з трирівнерою архітектурою зображено на рисунку 1.

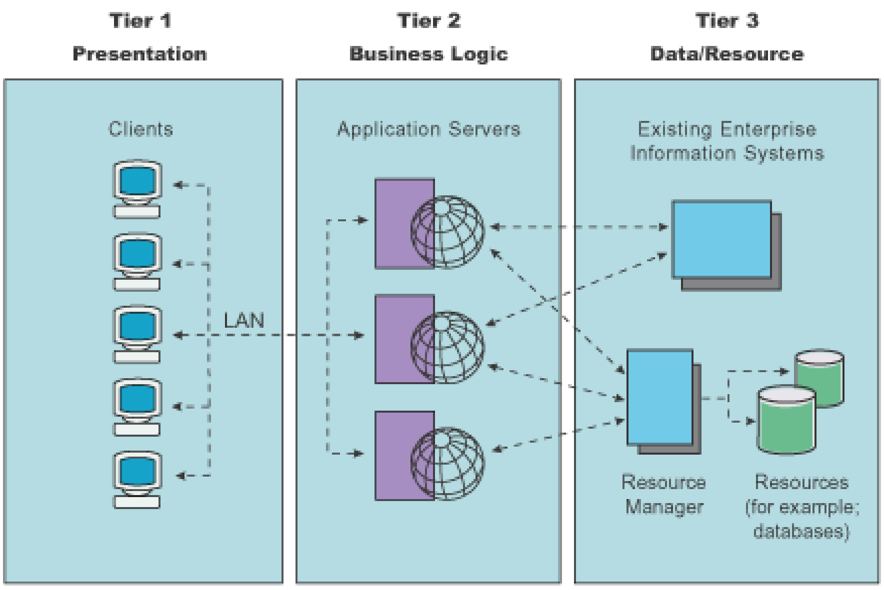

Рисунок 1 – Структура веб-додатку управління товарними запасами на основі трирівневої архітектури

Використання трирівневої архітектури дає багато переваг, включаючи швидкість розробки, масштабованість, продуктивність і доступність. Розбиття на модулі різних рівнів програми дає командам розробників можливість розробляти та підвищувати продуктивність з більшою швидкістю, ніж розробка унікальної кодової бази, оскільки певний шар може бути оновлений з мінімальним впливом на інші шари. Це також може допомогти покращити ефективність розвитку, дозволяючи командам зосередити увагу на їхній основній компетенції[2].

Список використаних джерел:

1. Paul Clements; Felix Bachmann; Len Bass. Documenting Software Architectures: Views and Beyond. – Second Edition. – Addison-Wesley Professional, 2010. – 79с. – ISBN 978-0-13-248861-7.

2. Len Bass, Paul Clements, Rick Kazman. Software Architecture in Practice, Third Edition. Addison Wesley, 2012. – 127с. – ISBN 978-0321815736