УДК 004.9:504:519.6
А.М. Луцків канд.техн.наук, доц., М.Я. Чайковський
Тернопільський національний технічний університет ім. Івана Пулюя, Україна

АРХІТЕКТУРИ ВИСОКОПРОДУКТИВНИХ ВЕБ-СЕРВІСІВ ПРОГНОЗУВАННЯ ПОГОДИ

A.M. Lutskiv Ph.D., Assoc. Prof., M.Y. Chaikovskyy

HIGH-PERFORMANCE WEB-BASED WEATHER FORECASTING ARCHITECTURE

При створенні системи прогнозування погоди, після аналізу вимог, первинним завданням є розроблення загальної архітектури системи та архітектур окремих її складових. Наступною задачею є вибір компонентів (програмних систем, бібліотек, фреймворків, СКБД, технологій програмування) системи з точки зору їх функціональних можливостей. Завершальним етапом є розроблення конкретних компонент системи.

Ключовими вимогами до розроблення високопродуктивного веб-сервісу прогнозування погоди є:

- Точне прогнозування погоди на основі моделі, яка є апробованою й забезпечує належний рівень достовірності прогнозування. Очевидно, що доцільним з точу зору спрощення процесу розробки є використати певну бібліотеку, програмну систему, або сторонній сервіс.
- Надійність роботи системи.
- Можливість масштабуватись та розширюватись у майбутньому.
- Мінімізація витрат на розроблення та підтримку такої системи, шляхом вибору доступних за ціною й добре документованих компонентів, а також, економії людино-годин під час розроблення системи шляхом використання відповідних фреймворків, бібліотек та технологій програмування.

Розглянемо загальну архітектуру проектованої системи (рис.1)

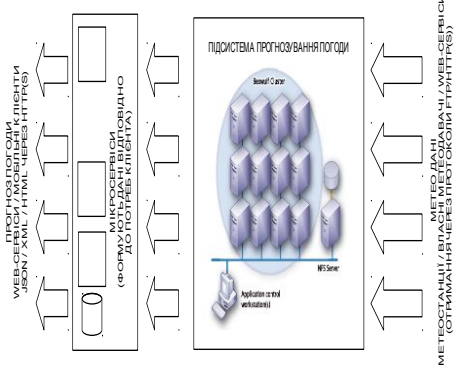

Рисунок 1. Загальна архітектура системи прогнозування погоди

Ключовою підсистемою є підсистема прогнозування погоди, яка базується на достовірній моделі передбачення прогнозу погоди. Такою моделюю обрано WRF [1], яка розроблена за підтримки Національного центру атмосферних досліджень США (рис.2). Модель MM5, на даний час, уже не підтримується відповідним центром досліджень. WRF модель є реалізована на мові Fortran з використанням бібліотеки MPI, що дає змогу розпаралелити її виконання на обчислювальному кластері. Модель передбачає надходження вхідних метеоданих із метеостанцій й повертає вихідні дані у вигляді прогнозу погоди на вказаний період. Конфігурування системи прогнозування погоди забезпечується системними адміністраторами або фахівцями, які суміщають

посади розробника та адміністратора (DevOps).Для отримання даних та надання результатів прогнозування погоди замовникам, доцільно реалізувати додаткові програмні компоненти системи — сервіси. З поміж архітектурних шаблонів доцільно виокремити кілька архітектур, які можуть лежати в основі створюваного сервісу[2]: монолітну, мікросервісну та сервіс-орієнтовану (Service-Oriented Architecture (SOA)).

Оптимальним з точки зору надійності, здатності масштабуватись, розподілу задачі створення сервісів поміж учасниками команди розробників, подальшої підтримки та вдосконалення системи є використання мікросервісної архітектури [2].

Процес створення мікросервісів передбачає, що на основі аналізу вимог до системи, група розробників здійснює наступні задачі:

- здійснює декомпозицію системи на компоненти-сервіси;
- виокремлює ролі в системі;
- виокремлює сутності, які є в системі й визначає взаємозв'язки та їх типи (фактично визначає логічну модель даних);
- розробляє механізми доступу до компонентів системи різним ролям;
- наступним етапом є кодування компонентів системи та системи автоматизованих тестів (модульних та інтеграційних);
- здійснює тестування системи автоматизованими тестами, а також реалізує навантажувальне тестування;
- упродовж усього етапу кодування використовуються засоби неперервної інтеграції.

При розробці даного сервісу, з метою економії коштів та скорочення часу розробки, доцільно використовувати наступні засоби:

- компоненти технології Java (забезпечують кросплатформовість, є добре апробованими й забезпечують велику кількість безкоштовних компонентів розробки);
- створення мікросервісів здійснювати з використанням фреймворку SpringBoot Framework;
- системою керування баз даних буде використовуватись PostgreSQL;
- засобом об'єктно-реляційного відображення — Hibernate, який використовується за замовчуванням у Spring-data;
- для швидкого доступу до REST-функцій сервісу буде використано бібліотеку netty, яка забезпечуватиме асинхронність роботи мікросервісів;
- систему неперервної інтеграції Jenkins;
- систему автоматизованого збирання проекту gradle.

Представлення (front-end) визначатиметься клієнтом (настільна робоча станція чи мобільний термінал), зокрема може бути створений з використанням одного з фреймворків ReactJS, AngularJS, VueJS або інших засобів.

Література

1. A descpription of the advanced research WRF version 2. NCAR Technical Note NCAR/TN-468+STR. / W.C. Skamarock, J.B. Klemp, J. Dudhia et al. // National Center for Atmospheric Research. - 2005. - 88 p.

2. Fowler M. Microservices a definition of this new architectural term / James Lewis, Martin Fowler // [Електронний документ] Режим доступу: URL: https://martinfowler.com/articles/microservices.html