УДК 528.9

ПРИКЛАД АРХІТЕКТУРИ WEB-ДОДАТКУ З ВИКОРИСТАННЯМ ВІЛЬНОГО ПРОГАМНОГО ЗАБЕЗПЕЧЕННЯ ТА СПЕЦИФІЧНІ АСПЕКТИ РЕАЛІЗАЦІЇ ЙОГО ГЕОІНФОРМАЦІЙНОЇ КОМПОНЕТИ

С.Я. Майстренко, Т.О. Загреба, К.В. Хурцилава
Інститут проблем математичних машин і систем НАН України
e-mail: taras@irpin.com

Типова структура веб-додатків

Створення веб-додатків з кожним роком набуває все більшої популярності, адже спрямоване на полегшення роботи користувача та надає постійний доступ до потрібних даних. Веб-додаток представляє собою клієнт-серверний додаток, в якому клієнтом виступає браузер, а сервером - веб-сервер. Робота веб-додатку розподіляється між сервером і клієнтом, дані зберігаються, переважно, на сервері, а обмін інформацією між клієнтом і сервером відбувається через мережу. Як правило, веб-додаток складається з трьох основних складових: клієнтської і серверної частин та бази даних.

Клієнтська частина веб-додатку представляє собою графічний інтерфейс користувача, який відображається в браузері і дозволяє користувачеві взаємодіяти з серверною частиною веб-додатку.

Основною мовою для розробки клієнтської частини веб-додатку є HTML з використанням стилів CSS. Крім того, при створенні графічного інтерфейсу часто використовуються скрипти на JavaScript, а також додаткові, вбудовані в веб-сторінку компоненти, створені, наприклад, на Java.

Серверна частина веб-додатку - це програма на сервері, що відповідає за обробку запитів від браузерів користувачів. Сервер опрацьовує відповідні запити, викликаючи скрипти, що відповідають за формування веб-сторінки, описаної мовою HTML, і результати відсилає клієнтам через мережу.

Для програмування серверної частини веб-додатків можуть використовуватись такі сучасні мови програмування, як Java, C / C ++, Perl, PHP, Ruby, мови VB.NET (C # та інші, що підтримують .NET). Слід зазначити, що найбільш поширеною мовою є PHP [1].

Конфігурація сервера, що базується на операційній системі Linux, мова програмування PHP, веб-сервер Apache і СУБД MySQL вважається стандартом і навіть має акронім LAMP (від скорочення Linux, Apache, MySQL, PHP) [2].

База даних використовується для зберіганням даних.

Браузер через Інтернет відправляє HTTP-запити до веб-сервера. Веб-сервер в свою чергу викликає скрипти, які, в разі потреби, звертаються до бази даних та повертають клієнтам відповідні оновлені веб-сторінки.

Приклад реалізації Web-додатку

Розглянемо можливу практичну реалізацію Web-додатку з використанням вільного програмного забезпечення на прикладі прототипної версії системи «Повітря», розробленої авторами доповіді для розрахунку зон уражень внаслідок викидів небезпечних речовин в атмосферу [3].

Для розробки клієнтської части інтерфейсу користувача використано HTML, стилі CSS та мову JavaScript .

Наразі для відтворення роботи веб-застосунків у реальному часі широку популярність отримали в основному дві технології: без використання протоколу WebSocket на основі звичайного AJAX-підходу [4] та з використанням WebSocket [5].

Websocket - це протокол повнодуплексного двобічного зв'язку поверх TCP-з'єднання, призначений для обміну даними між браузером і веб-сервером у режимі real-time (реального часу). Це стало можливим за рахунок того, що «клієнт» і «сервер» стали

рівноправними учасниками обміну даними. Крім того, WebSocket-специфікація представляє WebSocket JavaScript інтерфейс, що визначає повнодуплексне односокетне з'єднання для зменшення зайвого мережевого трафіку і затримок.

Важливим моментом створення web-застосунків real-time з використанням технології WebSocket є забезпечення швидкої передачі даних. Дана технологія використовується в додатках Магазину Windows, продажу білетів, миттєвих повідомленнях соціальних мереж та ін.

Головні проблеми використання технології Websocket: підтримка не у всіх браузерах, так як це нова специфікація HTML5; відсутність обмеження терміну життя запиту; у випадку використання прозорих проксі-серверів можлива підміна кеша даних, що передаються від сервера, даними від зловмисника. Проблема виявилася досить серйозною для того, щоб розробники Firefox та Opera оголосили, що в майбутніх версіях їхніх браузерів підтримка веб-сокетів буде за замовчуванням відключена аж до усунення проблеми небезпеки цього протоколу.

Для розробки системи «Повітря» використано AJAX-підхід до побудови веб-застосунків, при якому веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані. Переваги: простота реалізації, поширеність, підтримка у всіх сучасних браузерах.

Використання AJAX дозволяє значно скоротити трафік при роботі з веб-додатком завдяки тому, що замість завантаження всієї сторінки достатньо завантажити тільки частину, що змінилася, або взагалі тільки отримати/передати набір даних в форматі JSON або XML, а потім змінити вміст сторінки за допомогою JavaScript.

Застосування AJAX дозволяє знизити навантаження на сервер в кілька разів. Зокрема, всі сторінки сайту найчастіше генеруються за одним шаблоном, включаючи незмінні елементи («шапка», «навігаційна панель» і т.д.) Для їх генерації потрібні звернення до різних файлів, час на обробку скриптів (а іноді і запити до БД) – все це можна опустити, якщо замінити повне завантаження сторінки генерацією та передачею лише змістовної частини. Дизайн сторінки також зазвичай містить безліч файлів, пов'язаних з оформленням (картинки, стилі), на повторну обробку яких не треба витрачати час, використовуючи AJAX (економія на кількості HTTP-з'єднань значно вигідніша, ніж на скороченні трафіку кожного з них). Оскільки завантаження частини, що змінилася, значно швидше, даний підхід призводить прискорення реакції інтерфейсу. Тобто, до того, що користувач бачить результат своїх дій швидше і без мерехтіння сторінки, яке виникає при повному її перезавантаженні.

Як СУБД в системі вибрано PostgreSQL з додатком PostGIS для зберігання і управління просторовими даними та забезпеченням виконання, запитів до них.

Для візуалізації результатів розрахунків та для отримання додаткової інформації в системі створено геоінформаційну компоненту на основі OpenStreetMap з використовувалася широко відомої JavaScript-бібліотеки Leafleat.js ([6]). Використання OpenStreetMap дозволяє мати актуальні карти в якості картографічної основи. Однак для виконання додаткового аналізу, такого, наприклад, як розрахунок часу підходу хмари хімічної небезпечної речовини, площі забруднення та втрат населення населеного пункту, що потрапив в прогнозовану зону хімічного забруднення, та інших показників, для визначення яких необхідна просторова інформація, даних OpenStreetMap недостатньо. Для проведення таких розрахунків необхідні полігональні шари лісових масивів, областей, районів, населених пунктів з інформацією про кількість населення. В той час, як в Україні нараховується більше 30000 населених пунктів, відповідний полігональний шар OpenStreetMap містить інформацію лише для близько 13000 адміністративно-територіальних одиниць і тільки для деяких з них вказано кількість населення. Тому для вирішення такої задачі в системі «Повітря» використовуються (наявні в розробників

системи) відповідні тематичні шари карти України масштабу 1:200000 в географічній системі координат GCS_Pulkovo_1942. Оскільки в OpenStreetMap використовується Світова геодезична система 1984 (WGS-84), для сумісного використання додаткові тематичні шари було переведено до системи координат WGS-84. Слід відзначити, що для запобігання суттєвим спотворенням трансформація виконувалась за 7-ми параметричним перетворенням.

Для використання в системі додаткові тематичні шари за допомогою додатку PostGIS розміщено в картографічну БД PostgreSQL. PostGIS дозволяє досить ефективно виконувати просторові запити до БД навіть для значних об'ємів просторових даних.

Наприклад, для визначення населених пунктів, що потрапили в зону аварії, досить виконати запит:

"/sql/postgis_geojson.php?geotable=o4city_r&geomfield=geom","sql=SELECT ST_AsGeoJSON(geom) As geojson, codekoatuu, title, populat FROM o4city_r WHERE ST_Overlaps(geom, ST_SetSRID(ST_GeomFromGeoJSON('"+current_zone+"'), 4326)) Or ST_Within(geom, ST_SetSRID(ST_GeomFromGeoJSON('"+current_zone+"'), 4326)) Or ST_Contains(geom, ST_SetSRID(ST_GeomFromGeoJSON('"+current_zone+"'), 4326)) ORDER BY title",

де codekoatuu – код адміністративно-територіального устрою України, title – назва населеного пункту, populat – кількість населення, current_zone – геометричний об'єкт на карті, що визначає конкретну зону аварії.

Однак, в деяких випадках оператор ST_AsGeoJSON() може перетворювати коректну геометрію в некоректну. Наприклад, в результаті виконання запиту:

SELECT ST_IsValidDetail(taste.geom) As before_ST_AsGeoJSON, ST_IsValidDetail(ST_SetSRID(ST_GeomFromGeoJSON(ST_AsGeoJSON(taste.geom)), 4326)) As after_ST_AsGeoJSON FROM (SELECT 'MULTIPOLYGON(((30.4744203524661, 50.3737220277689, 30.4739792303552, 50.3734744492378, 30.4728661693293, 50.3728579942544, 30.4717342535146, 50.3722392022996, 30.4705837621944,
...
((30.495936401641, 50.3886176502811, 30.4959364016411, 50.3886176502812, 30.4955038164484 50.3882003007841, 30.495936401641, 50.3886176502811)))'::geometry As geom) As taste;

В результаті консультацій з розробниками PostGIS (після звернення на баг-трекер PostGIS) було з'ясовано, що перетворення з числового формату в текстовий (GeoJSON) – може бути джерелом для помилок. Особливо, якщо вихідна геометрія не надто "хороша", хоча опреатор ST_IsValidDetail () визначав таку геометрію, як вірну.

Для вирішення проблеми було прийняте рішення:

- спочатку виправляти геометрію за допомогою ST_MakeValid();
- так як, ST_MakeValid () може перетворити полігон в колекцію геометрій, то в разі полігону, використовувати опреатор ST_BuildArea(), а потім - ST_SnapToGrid().

Подібні проблеми, ймовірно, виникли через те, що геодані зберігаються в проекції CRS 4386, де одиницею виміру є градуси. Що призводить до великої кількості знаків після коми, а опреатор ST_AsGeoJSON () максимально повертає не більше 15 знаків.

В системі частина просторових операцій виконується на клієнті в браузері з використанням JavaScript-бібліотеки TurfJS [7] з достатньо широкими можливостями для роботи з просторовими даними. Як перевагу TurfJS серед подібних бібліотек, можна зазначити надання можливості виконання перетворення геометрій в форматі GeoJSON.

Слід також зауважити, що PostGIS і бібліотека Turf дають різні площі при виконанні округлення даних до одного/двох знаків після коми для одних і тих же

полігонів, навіть при використанні однакових методів обчислень. Це необхідно враховувати при виконанні додаткових розрахунків з використанням геометричних об'єктів для надання необхідної інформації користувачу під час діалогу та при формуванні звітів.

Крім того, при використанні бібліотеки Turf для визначенням центроїда за допомогою turf.PointonSurface(), в деяких випадках можливо отримати точку за межами полігона. Тому, необхідна додаткова перевірка і альтернативна функція для визначення точки в середині полігону.

Для невеликих службових JSON таблиць на клієнті зручним є використання JavaScript-бібліотеки AlaSQL, що підтримує підмножину SQL-99. З цієї причини застосування бібліотеки AlaSQL в багатьох випадках дозволяє без значних перетворень тексту програми передавати обробку даних з сервера на клієнт, і навпаки.

Закінчення

Використання сучасних web-технологій забезпечує такі важливі властивості сучасних інформаційних систем, як розподіленість, мультиплатформеність та доступ через мережу.

Розробка системи з використанням вільного програмного забезпечення дозволяє відмовитись від затрат на закупівлю, як привило, дорогого програмного забезпечення.

Візуалізація результатів розрахунків за допомогою сумісного використання OpenStreetMap та додаткових тематичних шарів в PostGIS забезпечує можливість отримувати додаткові дані для проведення аналізу, такі, наприклад, як проценти потрапляння конкретного населеного пункту у відповідну зону ураження та можливі втрати населення.

Використання JavaScript-бібліотеки TurfJS та PostGIS забезпечує розподіл виконання просторових операцій між браузерами клієнтів та сервером з метою забезпечення прийнятного відклику системи.

Практична розробка системи для розрахунку зон уражень внаслідок викидів небезпечних речовин в атмосферу у архітектурі веб-сервер - веб-клієнт показала доцільність використання аналогічного підходу до розробки систем, що мають у своєму складі геоінформаційну компоненту для забезпечення візуалізації та виконання геопросторового аналізу.

Література

1. Клієнт-серверна архітектура [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура.

2. LAMP [Електронний ресурс]. – Режим доступу: //https://ru.wikipedia.org/wiki/LAMP.

3. Майстренко С.Я., Ковалець І.В., Беспалов В.П., Загреба Т.О., Хурцилава К.В. Прототипна версія системи «повітря» для розрахунку зон уражень внаслідок викидів небезпечних речовин в атмосферу на основі використання скринінгових моделей та web-технологій // Математичні машини і системи. - 2016. - №4. - С. 33-41.

4. Дари К., Бринзаре Б.,Черчез Тоза Ф., Бусика М. AJAX и PHP: разработка динамических веб-приложений. – СПб.: Символ Плюс, 2007. – 336 с., ил. ISBN 5 93286 077 4.

5. WebSocket [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/WebSocket.

6. Leaflet [Електронний ресурс]. – Режим доступу: http://leafletjs.com//.

7. Advanced geospatial analysis for browsers and node [Електронний ресурс]. – Режим доступу: http://turfjs.org/.