УДК 004.4

Найдюк В.І.
Черноволик Г.О.

РОЗРОБКА АРХІТЕКТУРИ ТА АЛГОРИТМІВ СЕРВІСУ СИНХРОНІЗАЦІЇ ОНЛАЙН КАЛЕНДАРІВ

Вінницький національний технічний університет

Анотація

У статті розглянуто процес розробки архітектури та алгоритмів для створення сервісу синхронізації календарів. Було розглянуто серверно-клієнтську архітектуру, розроблено базу даних, та створено алгоритм пошуку дублікатів подій календаря.

Ключові слова: синхронізація, PostgreSQL, бази даних, алгоритм.

Abstract

The article discusses the process of developing the architecture and algorithms for creating a calendar synchronization service. The server-client architecture was considered, a database was developed, and an algorithm for searching for duplicate calendar events was created.

Keywords: synchronization, PostgreSQL, data bases, algorithm.

Вступ

Важливим при впровадженні хмарних додатків у роботу бізнесу чи організації є можливість їх інтеграції. Для уникнення повторного внесення тих самих даних часто використовують автоматизовані синхронізації даних, таких як календарі та їх події, контакти, завдання, і т.д. Це дозволяє уникнути непотрібної роботи та спростити роботу із сервісами, проте важливим є також те, аби забезпечити якість даних, уникнути випадкової втрати, або ж непотрібного створення дублікатів.

Тому актуальною є розробка веб-сервісу для вирішення питання, з яким стикаються користувачі хмарних веб-застосунків, що працюють з календарями. Сервіс дозволить користувачам у реальному часі синхронізувати два або більше календарі у популярних веб-застосунках, таких як iCloud Calendar, Microsoft Outlook Calendar, Google Calendar, численних CRM (customer relationship management) та ERP (enterprise resource planning) системах та ін.

При розробці веб-сервісу було використано клієнт-серверну архітектуру. В основі клієнт-серверної архітектури лежать дві компоненти: клієнт і сервер.

Розробка загальної архітектури продукту та бази даних

При розробці веб-сервісу було використано клієнт-серверну архітектуру. В основі клієнт-серверної архітектури лежать дві компоненти: клієнт і сервер.

Клієнт – комп'ютер на стороні користувача, який відправляє запит до сервера для надання інформації або виконання певних дій. Це може відбуватись через локальний додаток або ж веб-браузер.

Сервер – більш потужний комп'ютер або обладнання, призначене для вирішення певних завдань з виконання програмних кодів, виконання сервісних функцій за запитом клієнтів, надання користувачам доступу до певних ресурсів, зберігання інформації і баз даних.

Принцип роботи полягає в тому, що декілька серверів обробляють запит клієнта. Розподіл операцій знижує навантаження на сервер.

Загальна архітектура веб-сервісу складається із наступних елементів (рис 1):

1. Front end (UI) – користувацький інтерфейс.
2. Back end (API) – серверна частина доступу до даних.
3. Data layer – база даних, а також модулі інтеграції із даними календарів сторонніх сервісів та застосунків.

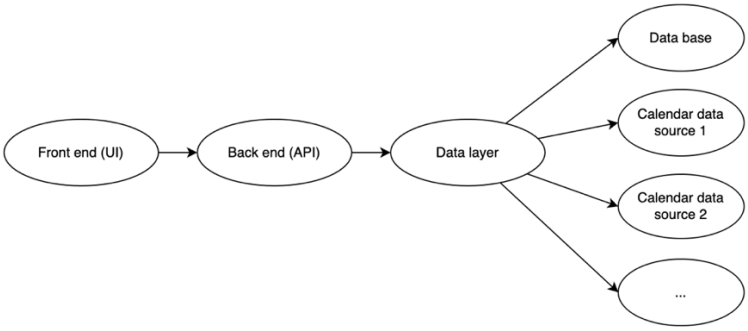

Рисунок 1 – Загальна архітектура веб-сервісу

Для надійного збереження усіх потрібних даних веб-сервісу було вирішено використати реляційну базу даних PostgreSQL.

База даних може бути визначена як структурна сукупність даних, що підтримуються в активному стані та відображає властивості об'єктів зовнішнього (реального) світу. В базі даних містяться не тільки дані, але й описи даних, і тому інформація про форму зберігання вже не схована в сполучення «файл-програма», вона явним чином декларується в базі [1].

Для оптимальної роботи веб-сервісу було прийнято рішення представити дані у нормальній формі використовуючи наступні сутності:

User. Представляє користувача системи, та містить наступні поля:

- Id – ключ та порядковий номер.
- Email – електронна адреса користувача.
- Name – ім'я користувача.
- Password – хеш-значення паролю.
- Salt – salt-значення паролю.

Sync. Представляє одну синхронізацію між вдома календарями користувача, та містить наступні поля:

- UserId – ID користувача.
- Name – назва.
- Deleted – чи видалена ця синхронізації.
- State – поточний стан (активна чи ні).

SyncDataSource. Поєднює синхронізацію та джерело даних календаря, та містить наступні поля:

- SyncId – ID синхронізації.
- DataSourceId – ID джерела даних.
- PartId – номер календаря (0 або 1), якому відповідає це з'єднання.
- Data – дані у JSON форматі, необхідні для доступу до календаря.

DataSource. Представляє джерело даних подій календаря, та містить наступні поля:

- Name – назва.
- LogoUrl – посилання на картинку логотипу сервісу.
- SettingsJsonSchema – формат даних доступу.
- IsOAuthSupported – чи підтримує джерело протокол OAuth.
- CompanyId – ID компанії сервісу.

Company. Представляє компанію, яка є власником чи розробником сервісу онлайн календарів, та містить наступні поля:

- Name – назва.
- Description – опис.
- Website – посилання на веб-сторінку сервісу.

Зв’язок між описаними вище сутностями зображено на рисунку 2.

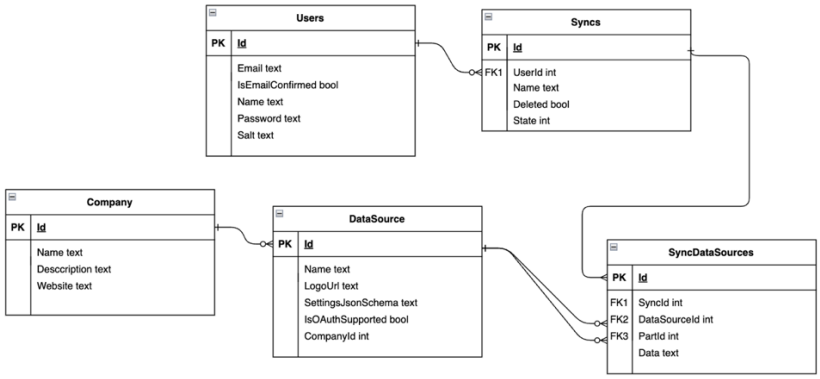

Рисунок 2 – ER-діаграма моделі бази даних

Розробка алгоритмів роботи програми

Для роботи веб-сервісу було розроблено метод синхронізації календарів – процес, згідно з якого відбувається завантаження, обробка та синхронізація подій календарів зовнішнього користувацького чи бізнес застосунку. Правильний вибір цього методу є ключовим аспектом ефективної роботи веб сервісу та максимізації його користі користувачу.

Після аналізу усіх необхідних кроків було сформовано наступний метод пошуку дублікатів контактів:

1. Відповідно до вибраного зовнішнього застосунку відбувається завантаження модуля доступу до даних. Для завантаження подій використовується зовнішнє API відповідного застосунку, операції з яким реалізовані у модулі інтеграції.
2. Відбувається авторизація користувача цього застосунку. Для більшості сервісів використовується сучасний протокол OAuth 2.0 authorization code flow, який призначений для випадків, коли дані завантажуються із зовнішнього сервісу до серверної частини. Деякі сервіси можуть не підтримувати такий метод, тоді буде використано Basic Authentification.
3. У разі успішної авторизації завантажуються поточні дані подій користувача. Для цього використовується GET запит до API вибраного сервісу, який дозволяє зчитати дані усіх подій (ймовірно, розбивши запити на сторінки).
4. Дані подій приводяться у спільний (уніфікований) формат, що дозволяє їх подальшу обробку. Для цього було реалізовано спеціальну процедуру мапінгу даних у спільний формат.
5. Використовуючи алгоритм порівняння подій вони розбиваються на групи, кожна з яких відповідає за ту саму подію. Події обробляються одна за одною та додаються в існуючу групу, або ж створюють нову групу, якщо це нова подія.
6. Події, які були знайдені лише в одному із двох календарів, що синхронізовуються, створюються у відповідному іншому календарі використовуючи модуль інтеграції (INSERT метод).
7. Події, які були оновлені в одному із двох календарів оновлюються у відповідному іншому календарі використовуючи модуль інтеграції (PATCH метод).
8. Видалені події видаляються із календаря використовуючи DELETE запит модуля інтеграцій.

Іншою важливою частиною веб-сервісу є алгоритм, згідно з яким виконується пошук дублікатів, а саме яким чином дві події перевіряються на однаковість. Саме завдяки цьому алгоритму сервіс гарантує, що не буде створено дублікатів подій.

Для якомога більшої кількості пар подій, які справді є дублікатами, алгоритм повинен повертати позитивний результат, і в цей самий час мінімізувати хибні позитиви, де події, які не є дублікатами, були визначені як дублікати.

Після детального аналізу питання було розроблено систему умов, за якими пара подій буде

перевірятись на однаковість. Ця система умов була реалізована у вигляді алгоритму пошуку дублікатів подій, зображеного блок-схемою на рисунку 3.

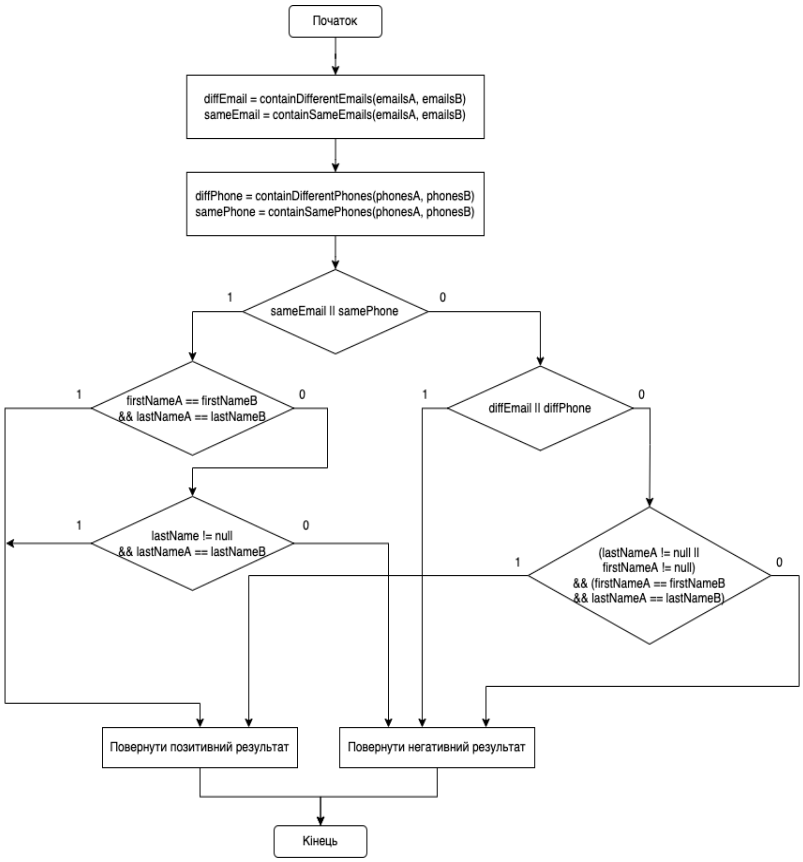

Рисунок 3 – Блок-схема алгоритму пошуку дублікатів подій

Висновки

Отже, було розглянуто та проаналізовано дані, з якими працює веб-сервіс синхронізації календарів, а також визначено процеси передачі даних між сервісом, користувацьким інтерфейсом та зовнішніми інтегрованими застосунками. Було розроблено модель робити веб-системи синхронізації календарів, а також спроєктовано модель бази даних. Також було розроблено алгоритм пошуку дублікатів подій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Пєтух А.М.,Романюк О.Н., Романюк О.В. Бази даних. Мови запитів, управління транзакціями, розподілена обробка даних, Навчальний посібник. ВНТУ, 2016

Найдюк Валерія Іванівна – студентка групи ПІ-22мз, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: leranaydyuk7@gmail.com.

Науковий керівник: ***Черноволик Галина Олександрівна*** – кандидат технічних наук, доцент кафедри програмного забезпечення, Вінницький національний технічний університет**,** Вінниця

Naidiuk Valriia I. – Department of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, email: leranaydyuk7@gmail.com.

Supervisor: ***Chernovolyk Halyna O.*** – PhD, Associate Professor of Software, Vinnytsia National Technical University, Vinnytsia.