DOI: https://doi.org/10.33216/1998-7927-2025-288-2-5-13

УДК 004.89:659

ОСОБЛИВОСТІ СТРУКТУРНИХ ПРОГРАМНИХ РІШЕНЬ З ТЕХНОЛОГІЄЮ RSS

Лаговський В.В., Лаговський О.В., Ніжегородцев В.О., Філоненко М.М., Скасків Л.В.

STRUCTURAL FEATURES OF SOFTWARE SOLUTIONS WITH RSS TECHNOLOGY

Lagovskyi V.V., Lahovskyi O.V., Nizhehorodtsev V.O., Filonenko M.M., Skaskiv L.V.

Стаття присвячена аналізу структурних особливостей та архітектурних підходів до розробки програмних рішень, що інтегрують технологію RSS (Really Simple Syndication). Підкреслюється актуальність RSS для автоматизованого збору, обробки й поширення інформації з веб-ресурсів. Цінність технології полягає у стандартизованому XML-форматі, що спрощує агрегацію та дозволяє ефективно керувати інформаційними потоками.

Розглянуто ключові переваги RSS: оптимізація отримання даних, контроль користувача над контентом без алгоритмів, анонімність, уніфікований формат без реклами, надійність та офлайн-доступ. Проаналізовано недоліки: змінна повнота контенту, відсутність інтерактивності, потреба у спеціалізованих агрегаторах, ризик інформаційного перевантаження та обмежена підтримка мультимедіа.

Обговорюються технічні аспекти побудови RSS-систем. Обґрунтовано критичну важливість автономного збору та збереження даних у власній БД агрегатора. Це запобігає навантаженню на джерела, підвищує швидкість доступу та уможливлює реалізацію пошуку, фільтрації, аналізу. Порівнюється доцільність використання документно-орієнтованих БД (гнучкість) та реляційних (структурованість, цілісність, усунення дублікатів).

Приділено увагу проблемі дублювання новин; розглядаються методи дедуплікації на основі порівняння заголовків/описів, аналізу текстової та семантичної схожості. Висвітлено виклики інтернаціоналізації: необхідність автовизначення мови та уніфікації часу публікації до UTC.

Запропоновано архітектурний підхід з окремим програмним модулем обробки RSS. Модуль включає підсистеми збору/парсингу, обробки (дедуплікація, уніфікація, визначення мови/часу, класифікація), збереження даних у БД та сортування/пошуку (з індексацією). Наголошено на потребі автономної роботи модуля, обробки помилок та взаємодії через API.

У висновках зазначено, що RSS залишається ефективним інструментом, але вимагає продуманих структурних рішень. Правильна архітектура забезпечує ефективну агрегацію, знижує навантаження на джерела, дає масштабованість та інтеграцію функцій. Перспективним напрямком є використання ШІ та машинного навчання для глибшого аналізу й персоналізації RSS-контенту.

***Ключові слова**: RSS, програмний модуль, база даних, структура програмного забезпечення, XML, парсинг, дедуплікація.*

Вступ. У сучасному світі, де інформація стала одним із найважливіших ресурсів, технології, що спрощують її збір, обробку та поширення, набувають особливого значення. Однією з таких технологій є RSS – (Really Simple Syndication), яка дозволяє автоматизувати процес отримання оновлень з різних джерел у зручному форматі. Використання RSS у структурних програмних рішеннях відкриває нові можливості для інтеграції даних, створення динамічних систем та підвищення ефективності роботи з інформацією.

У статті розглядаються особливості впровадження технології RSS у програмні рішення, її переваги, а також ключові аспекти, які необхідно враховувати при розробці таких систем. Дослідження цієї теми є актуальним, оскільки, незважаючи на зростання популярності соціальних мереж та інших методів розповсюдження контенту, впровадження RSS у сучасні програмні рішення залишається актуальним [12]. RSS продовжує залишатися важливим інструментом у сферах, пов'язаних із збором, аналізом та розповсюдженням даних.

RSS використовується для автоматичного отримання оновлень із вебсайтів, зокрема новинних порталів, блогів, подкастів, форумів та інших ресурсів, що регулярно публікують новий контент. Завдяки RSS користувачі можуть підписуватися на стрічки новин і отримувати нові публікації в одному місці - у спеціальних RSS-рідерах або агрегаторах, не заходячи щоразу на кожен сайт окремо. Це зручно для тих, хто стежить за великою кількістю джерел інформації, адже RSS дозволяє економити час і не пропускати важливі оновлення.

Цей успіх намагається перехопити технологія Atom, яка є його логічним продовженням і спробою створити більш сучасний, структурований та універсальний формат для розповсюдження вебконтенту. Однак в спробах удосконалити попередній формат було втрачено його особливості, які з часом виявилися потрібнішими ніж очікувалося.

Аналіз останніх досліджень і публікацій. Опис об'єкту дослідження та розглянутий стандарт RSS 2.0. було наведено в офіційній документації RSS [1]; інформація про розвитком IT сектору у світі та попит на технології швидкої та уніфікованої передачі даних розкрита у дослідженні [2]; тенденції використання технології RSS підтверджені у праці [3]; актуальність RSS та подібних технологій для обробки великими з мовними моделями та подібними нейронними мережами для аналізу інформації представлена в змісті [4]; демонстрація перспектив технологій RSS наведено в джерелі [5]; інші можливості, тренди та тенденції, а також можливість використовувати публічних агрегаторів для популяризації свого продукту чи платформи описані в працях [6-12].

Метою даної роботи є узагальнення необхідних особливостей архітектури додатків або програмних модулів для роботи з технологію RSS і їй подібних. Найбільший акцент буде стояти на аспектах зберігання даних і їх подальшої обробки.

Постановка задачі. Для досягнення цієї мети були розглянуті сучасні методи та інструменти програмування та аналізу даних. Результатом роботи став детальний опис вимог до програмного коду для правильної роботи з обраним типом даних. У дослідженні було описано методи збору інформації з RSS каналів для подальшої обробки та надано рекомендації щодо структури бази даних, які можуть бути використанні в тандемі з технологією RSS. Для програмної реалізації обрано сучасні засоби розробки, апаратна платформа базується на поширеній компонентній базі.

Викладання основного матеріалу. У сучасному інформаційному суспільстві однією з ключових технологій обміну та доставки контенту є RSS (Really Simple Syndication). Цей формат став фундаментальним для організації автоматизованого доступу до оновлень вебресурсів. В контексті програмної інженерії, інтеграція RSS вимагає розробки специфічних структурних рішень, які забезпечують ефективну обробку, зберігання та передачу даних. Програми, що працюють з RSS, зазвичай реалізуються за принципами клієнт-серверної архітектури. Сервери надають стрічки новин у вигляді XML-файлів, а клієнти (RSS-агрегатори) періодично перевіряють наявність оновлень. У таких системах важливе значення має модульність, що дозволяє розмежувати обробку XML-даних, збереження інформації в базах даних і візуалізацію для користувача.

RSS це технологія, яка дозволяє автоматично отримувати оновлення з вебсайтів, блогів, новинних ресурсів або інших джерел контенту у стандартизованому форматі, вона являє собою формат веб-синдикації, що використовується для надання користувачам та програмам доступу до оновлень вебсайтів у машиночитаному форматі [6].

Основна мета RSS полягає в тому, щоб спростити процес поширення та отримання інформації. Замість того, щоб постійно особисто відвідувати різні сайти виявлення та дослідження оновленого контенту, користувачі можуть підписатися на оновлення через RSS-канали, що дозволяє їм залишатися в курсі новин без зайвих зусиль.

RSS використовує XML-формат для структурування даних, що робить його зручним для обробки різними програмами та сервісами. Вебсайти, які підтримують RSS, публікують

свої оновлення у вигляді спеціальних стрічок, які називаються RSS-каналами. Ці стрічки містять заголовки, короткі описи, посилання на повні версії статей або інший контент, а також інші метадані. Коли на сайті з'являється нова публікація, RSS-канал автоматично оновлюється, і користувачі, які підписалися на цей канал, отримують сповіщення через спеціальні програми, відомі як RSS-агрегатори або RSS-рідери. Технічно, RSS-канал вебсайту можна знайти за постійною URL-адресою, і він надає користувачам найновішу версію машиночитаного XML-файлу.

До переваг RSS найчастіше відносять:

– оптимізація процесу отримання актуальної інформації. RSS усуває необхідність у періодичному ручному моніторингу численних вебсайтів, оскільки технологія централізує надходження нового контенту в єдиному інтерфейсі RSS-агрегатора, адже це призводить до значного підвищення ефективності споживання інформації та скорочення часових витрат користувача;

– надання користувачеві контролю над інформаційним потоком, оскільки на відміну від багатьох сучасних платформ, що застосовують алгоритмічну фільтрацію контенту, RSS забезпечує доставку всіх матеріалів із обраних користувачем джерел, а контент зазвичай подається у хронологічній послідовності або відповідно до налаштувань користувача, що унеможливлює пропуск релевантних публікацій через зовнішні алгоритмічні впливи;

– процес підписки на RSS-канали, як правило, є анонімним і не вимагає надання персональних ідентифікаційних даних (наприклад, адреси електронної пошти), що мінімізує ризики несанкціонованого збору даних про інформаційні преференції користувача та знижує ймовірність отримання небажаних комерційних повідомлень;

– використання RSS-агрегаторів часто сприяє концентрованому сприйняттю контенту, а багато таких програмних рішень представляють інформацію (заголовки, анотації, повні тексти) в уніфікованому, мінімалістичному форматі, позбавленому сторонніх візуальних елементів інтерфейсу вихідних вебсайтів, включно з рекламними блоками та іншими компонентами, що можуть відволікати увагу від основного змісту;

– забезпечує надійність та повноту інформаційного моніторингу обраних ресурсів, оскільки відсутній механізм зовнішньої алгоритмічної фільтрації та всі опубліковані матеріали з підписаних каналів доставляються в агрегатор користувача;

– деякі програмні RSS-агрегатори підтримують функцію кешування контенту, що уможливлює його відкладений перегляд в умовах відсутності доступу до мережі Інтернет (офлайн-доступ).

Однак RSS має і свої недоліки:

– неуніфікованість повноти контенту, оскільки відсутній єдиний стандарт щодо обсягу інформації, яка передається через RSS-канал; деякі джерела надають лише заголовки та короткі анотації, що вимагає переходу на оригінальний сайт для ознайомлення з повним текстом статті, інші надають повний контент; форматування та структура даних можуть суттєво відрізнятися між різними фідами, що впливає на консистентність представлення інформації в агрегаторі;

– відсутність інтерактивності, оскільки RSS за своєю архітектурою є переважно односпрямованою технологією передачі даних; стандартний протокол не передбачає вбудованих механізмів для взаємодії користувача з контентом, таких як коментування; оцінювання (лайки) чи поширення матеріалів безпосередньо з інтерфейсу більшості RSS-агрегаторів, а для здійснення таких дій необхідний перехід на першоджерело;

– необхідність використання спеціалізованого програмного забезпечення, оскільки для роботи з RSS-каналами користувачеві потрібен окремий програмний інструмент: RSS - агрегатор (десктопна програма, мобільний додаток або вебсервіс), адже це може створювати певний поріг входження для осіб - не знайомих з цією технологією та вимагає додаткових дій порівняно з прямим відвідуванням вебсайтів або використанням інтегрованих платформ;

– ризик інформаційного перевантаження, хоча й RSS покликаний оптимізувати споживання інформації: підписка на значну кількість джерел та може призвести до протилежного ефекту - інформаційного перевантаження; великий обсяг непрочитаних повідомлень в агрегаторі може ускладнити виокремлення дійсно важливої інформації;

– проблеми з оновленням статичного контенту; інформація, передана через RSS-канал, може бути статичним знімком на момент публікації; якщо оригінальну статтю на сайті було оновлено або виправлено після синдикації,

ці зміни не завжди автоматично відображаються в RSS-фіді та, відповідно, в агрегаторі користувача;

– обмежена підтримка мультимедіа; технічно RSS може передавати посилання на мультимедійні файли, їх вбудоване відображення та інтерактивність в середовищі RSS-агрегаторів часто є менш функціональними та уніфікованими порівняно з досвідом перегляду на оригінальних вебсайтах.

Відповідно до цих особливостей сформувалася чітка ніша для цієї технології. RSS широко використовується на новинних сайтах для швидкого поширення новин, у блогах для інформування підписників про нові статті, а також у подкастах для розповсюдження аудіо- або відеоконтенту. Електронні бібліотеки також часто використовують RSS для повідомлення про нові книги або статті.

Таким чином, RSS залишається важливим інструментом для тих, хто прагне ефективно управляти інформаційними потоками та отримувати актуальні оновлення з обраних джерел. Його простота та універсальність роблять його зручним рішенням для поширення та отримання інформації в Інтернеті. Не зважаючи на чітку структуру RSS-файлів, зустрічаються випадки коли деякі теги ігноруються. Таким чином вони просто залишаються пустими або ж не присутні взагалі. Це дуже важлива особливість яка повинна враховуватися при обробці відповідних файлів.

Збереження RSS-даних є важливим не лише для зручності користувачів, а й для етичного та ефективного використання ресурсів джерел. Якщо для кожного окремого користувача виконувати збір даних безпосередньо з сайтів у режимі реального часу, це може призвести до надмірного навантаження на сервери цих ресурсів. Особливо це актуально для популярних сайтів із великою кількістю читачів. Часті запити від численних користувачів фактично створюють ефект розподіленої атаки на відмову в обслуговуванні (DDoS), навіть якщо це не є навмисною дією. Таке навантаження не лише шкодить власникам сайтів, а й може призвести до обмежень або блокування доступу для сервісів, що збирають RSS-канали.

Натомість оптимальним рішенням є періодичне збереження RSS-даних на стороні агрегатора. Це дозволяє виконувати збір інформації з джерел із оптимальною періодичністю — наприклад, кожні кілька хвилин або годин, залежно від потреб. Збережені дані зберігаються в базі даних і стають доступними для користувачів без потреби повторного звернення до джерела. Такий підхід не лише знижує навантаження на сторонні сайти, а й підвищує швидкість доступу до даних для користувачів, оскільки інформація надається безпосередньо з локального сховища. Крім того, це дозволяє реалізувати додаткові функції, як-от фільтрація, пошук за ключовими словами, аналіз дублювань та персоналізовані рекомендації, не порушуючи етичних принципів використання веб-ресурсів.

Таким чином, збереження RSS-даних у власній базі даних є не лише технічно доцільним, а й необхідним для підтримання стабільної та взаємовигідної взаємодії між сервісом агрегування та джерелами інформації. Вибір між документно-орієнтованою та реляційною базою даних для зберігання даних із RSS-стрічок залежить від природи додатку і способу його використання. Документні бази даних, такі як MongoDB, особливо добре підходять для роботи з RSS-даними завдяки своїй гнучкості. Оскільки RSS-потоки можуть мати різні структури особливості — з різною кількістю вкладених елементів та додатковими метаданими — можливість зберігати дані без жорсткої схеми спрощує збір даних. У таких базах кожен запис зберігається у вигляді JSON-подібного документа, де можна легко зберегти всі деталі окремої новини: заголовок, посилання, опис, дату публікації, категорії та автора. До того ж, документно-орієнтовані бази часто демонструють високу продуктивність при зчитуванні даних і легко масштабуються горизонтально, що важливо при роботі з великою кількістю потоків.

З іншого боку, реляційні бази даних, такі як PostgreSQL або MySQL, підходять для ситуацій, коли структура даних є чітко визначеною та стабільною що ускладнює процес інтеграції нових смуг, які будуть мати яку структурну особливість, на кшталт ігнорування тегів мови чи джерела, однак обробка спрощується, так як в кінцевому результаті ми зводимо всі дані до одного формату. Таким чином, вибір залежить від конкретних потреб: якщо додаток зберігає дані без ніякої обробки то документно-орієнтовані бази даних краще справляються з цією задачею. Однак, якщо перед застосунком стоїть задача перекладу, фільтрації, цензурування, тощо, то реляційні бази даних краще виконують цю роботу.

Одна з ключових проблем при роботі з RSS-стрічками — це повторення ідентичних або

дуже схожих новин, які надходять із різних джерел. Оскільки багато інформаційних сайтів публікують новини на схожі теми, часто трапляється ситуація, коли одна й та сама подія висвітлюється декількома ресурсами майже одночасно. Це призводить до дублювання контенту в базі даних і ускладнює аналіз інформації, адже користувачам доводиться переглядати безліч повторів, щоб знайти унікальні деталі. Проблема загострюється, якщо RSS-потоки налаштовані для автоматичного збору новин, адже система не завжди здатна визначити, чи є новина справді новою, чи лише варіацією вже наявної.

Для розв'язання цієї проблеми часто застосовують алгоритми дедуплікації — методи порівняння заголовків, описів та основного тексту новини. Також можна використовувати аналіз схожості на основі ключових слів або семантичне порівняння змісту, що допомагає виділяти унікальний контент і фільтрувати дублікати. Однак навіть ці підходи не гарантують стовідсоткової точності, особливо коли різні джерела описують ту саму подію різними словами. Тому проблема повторів залишається актуальною задачею для розробників систем агрегування новин.

Також окремо слід зазначити, що для уникнення такої проблеми можна використати внутрішні механізми реляційних баз даних. Так, якщо заголовок, зміст, та посилання на новину співпадають, то можна однозначно визначити що це дублікат. MySQL має ідеальний механізм для вирішення цієї проблеми, а саме властивість унікальності полів. Якщо правильно описати це правило, то дублікатів можна уникнути.

Натомість, документно-орієнтовані бази даних мають багато переваг для зберігання даних із RSS-стрічок, вони не здатні автоматично розв'язувати проблему дублювання ідентичних новин. Оскільки кожна новина зберігається як окремий документ, база даних не виконує перевірку на унікальність вмісту за замовчуванням. Навіть якщо використовувати індекси для пошуку схожих заголовків або посилань, це не гарантує виявлення дубліката, якщо новини дещо відрізняються формулюванням чи структурою. Документно-орієнтовані бази даних добре підходять для швидкого доступу до даних, але вони не пропонують вбудованих механізмів для аналізу текстової подібності чи семантичного порівняння. Тому для боротьби з повтореннями потрібно додатково реалізовувати логіку дедуплікації на рівні застосунку або використовувати зовнішні інструменти, які порівнюють новини за допомогою алгоритмів обробки природної мови чи пошуку схожості. Таким чином, вибір документно-орієнтованої бази даних не усуває проблему дублювання, а лише відкладає її розв'язання на етап обробки даних.

Якщо додаток здійснює збір даних із зовнішніх сайтів, він має працювати автономно та незалежно від дій користувачів. Такий підхід забезпечує стабільність роботи сервісу та знижує навантаження на джерела даних. Це дозволяє користувачам миттєво отримувати потрібні дані без очікування завантаження та мінімізує кількість зовнішніх запитів.

Інтернаціональність є одним із ключових викликів при роботі з RSS-стрічками. Оскільки різні джерела публікують контент різними мовами та в різних часових поясах, їх уніфікація стає складним завданням. Додаткову складність створює різниця в часових поясах, яка впливає на коректне відображення часу публікації новин. Хоча стандарт RSS передбачає вказівку часової зони у форматі ISO 8601, не всі джерела дотримуються цієї вимоги.

На основі проведеного аналізу маємо - обробку RSS-даних, яка є складним процесом та який доцільно виокремити в окремий програмний модуль. Такий модуль має забезпечувати збір, збереження, обробку та сортування інформації, гарантуючи стабільну та ефективну роботу системи.

Розглянемо детально основні складові цього модуля та їхні функції.

1. База даних як центральний елемент модуля. Центральним елементом модуля є база даних, яка виконує роль сховища для зібраної інформації. Вибір між документно-орієнтованою та реляційною базами даних залежить від потреб проєкту. Документно-орієнтовані бази даних, як-от MongoDB чи Firebase Firestore, забезпечують гнучке збереження даних, що важливо для RSS-стрічок із різною структурою. Водночас реляційні бази даних, такі як PostgreSQL або MySQL, зручні для організації зв'язків між даними, а також для пошуку дублікатів та виконання складних запитів. Ідеальним підходом є поєднання обох типів баз даних або використання гібридних рішень. Наприклад, документно-орієнтована база даних може зберігати сирі дані RSS-стрічок, тоді як реляційна - оброблені та структуровані дані для подальшого аналізу. Збереження даних у власній базі дозволяє уникнути повторних запитів до зовнішніх

сайтів, що не тільки прискорює роботу системи, а й запобігає перевантаженню джерел інформації.

2. Парсинг даних (підсистема збору інформації). Перша підсистема модуля відповідає за збір та парсинг даних. Її основне завдання - завантажувати RSS-стрічки з різних джерел і витягувати ключову інформацію, як-от заголовок, опис, дату публікації, посилання на оригінальну статтю та назву джерела. Ця підсистема повинна працювати автономно та періодично, не залежачи від дій користувачів. Для цього можна використовувати планувальники завдань на кшталт cron або вбудовані механізми, такі як Celery у Python. Парсинг має враховувати специфіку кожного джерела, оскільки структура RSS-файлів може відрізнятися. Для обробки стандартних форматів, як-от RSS 2.0 чи Atom, можна використовувати бібліотеки, наприклад, feedparser для Python. Якщо джерело не дотримується стандартів, необхідно налаштувати індивідуальний підхід для парсингу. Після вилучення даних інформація зберігається в базі даних разом із метаданими, такими як час завантаження та унікальний ідентифікатор джерела.

3. Обробка даних (підсистема аналізу та уніфікації). Друга підсистема модуля займається обробкою отриманих даних. Її завдання — привести інформацію до єдиного формату, видалити дублікати, визначити мову та часовий пояс, а також класифікувати новини за категоріями. Для пошуку однакових новин можна порівнювати заголовки та описи. Якщо тексти відрізняються, можна застосувати алгоритми обчислення схожості, наприклад, Levenshtein distance або TF-IDF для аналізу ключових слів. Це допоможе виявити дублікати, навіть якщо формулювання відрізняються. Оскільки RSS-джерела можуть бути різними мовами, необхідно автоматично визначати мову кожної новини. Для цього можна використати бібліотеки, такі як langdetect або langid, які аналізують текст і визначають мову з високою точністю. Якщо мову вказано в метаданих RSS-стрічки, її можна використовувати без додаткового аналізу. Час публікації має бути наведений до єдиного стандарту - зазвичай це UTC. Якщо джерело не вказує часовий пояс, його можна визначити на основі домену сайту або метаданих. У випадку відсутності чіткої інформації використовується часова зона за замовчуванням із можливістю ручного налаштування для конкретних джерел. Для групування новин за темами можна використовувати ключові слова або алгоритми машинного навчання. Наприклад, моделі на основі TF-IDF або Naive Bayes дозволяють автоматично відносити статті до певних категорій: політика, спорт, наука, технології тощо. Якщо потрібна складніша класифікація, можна застосувати нейронні мережі або моделі, такі як BERT, які розуміють контекст тексту. Після обробки результати зберігаються в базі даних у структурованому вигляді, готовому до подальшого використання.

4. Сортування даних (підсистема впорядкування та пошуку). Окремим компонентом модуля є підсистема сортування, яка впорядковує дані відповідно до потреб користувачів. Сортування може здійснюватися за різними критеріями, зокрема:

– за часом публікації - найсвіжіші новини відображаються першими, що є стандартним підходом для агрегаторів;

– за релевантністю - якщо користувач виконує пошук за ключовими словами, результати можна відсортувати за ступенем відповідності пошуковому запиту. Для цього застосовуються алгоритми пошуку за схожістю, наприклад, BM25;

– за популярністю джерела - деякі користувачі можуть надавати перевагу новинам із перевірених або авторитетних сайтів, тому система повинна дозволяти пріоритезувати такі джерела;

– за пріоритетністю тем - якщо користувач підписаний на кілька категорій, можна надавати перевагу новинам із найбільш релевантних для нього тем;

– сортування може комбінувати кілька факторів одночасно, оскільки можна спочатку показувати найсвіжіші новини, але серед них вищі позиції займатимуть статті з авторитетних джерел або ті, що відповідають інтересам користувача;

– для підвищення продуктивності слід реалізувати індексацію даних у базі, використовуючи індекси для часто використовуваних полів, таких як час публікації, мова та категорія, що дозволить швидко виконувати запити навіть за великої кількості даних.

5. Взаємодія з іншими компонентами системи:

– оскільки модуль обробки RSS-даних працює автономно, він повинен взаємодіяти з іншими компонентами системи через

інтерфейси API: інтерфейс для отримання оброблених даних у вигляді JSON або XML, який використовується клієнтським додатком для відображення новин;

– API для пошуку та фільтрації даних за ключовими словами, мовою чи категорією;

– адміністративний інтерфейс для налаштування параметрів збору даних, періодичності оновлення та керування списком джерел;

– автономність і стабільність роботи.

Модуль повинен працювати незалежно від дій користувачів. Збір даних виконується за розкладом або в режимі безперервного моніторингу, залежно від потреб проекту. Для підвищення надійності варто реалізувати механізми обробки помилок: наприклад, якщо джерело тимчасово недоступне, система має повторити спробу через певний інтервал часу. Також слід передбачити обмеження на кількість запитів до одного джерела, щоб уникнути блокування або перевантаження сайту.

Технологія RSS залишається ефективним інструментом для автоматизованого отримання оновлень із різних джерел завдяки своїй стандартизованій структурі на основі XML. Її ключова перевага - це економія часу та централізований доступ до інформації без необхідності відвідувати кожен сайт окремо. Проте для побудови ефективної системи збору та обробки RSS-даних необхідно враховувати низку технічних та організаційних аспектів:

– по-перше, оскільки формат RSS має чітку ієрархічну структуру з елементами <rss>, <channel> та <item>, важливо правильно інтерпретувати ці теги під час парсингу, однак на практиці не всі RSS-канали дотримуються стандарту: деякі теги можуть бути порожніми або відсутніми, що ускладнює обробку даних і вимагає гнучкості від системи;

– по-друге, автономний збір даних є не лише технічно доцільним, а й етично необхідним; якщо кожен користувач робитиме запити до джерел у реальному часі, це створить надмірне навантаження на сервери сайтів, подібне до DDoS-атак, тому рекомендовано періодично завантажувати та зберігати RSS-дані у власній базі даних, що не лише прискорює доступ користувачів до інформації, а й мінімізує зовнішні запити;

– по-третє, вибір типу бази даних залежить від характеру роботи системи; документно-орієнтовані бази даних, такі як MongoDB, добре підходять для зберігання різноманітних структур RSS-даних без необхідності жорсткої схеми; реляційні бази даних, як-от MySQL або PostgreSQL - ефективні для обробки даних, пошуку дублікатів та виконання складних запитів.

Окремою проблемою залишається повторення ідентичних або схожих новин з різних джерел, оскільки більшість сайтів висвітлює однакові події, у базі даних накопичується дубльований контент, що ускладнює аналіз інформації. Для виявлення дублікатів потрібно порівнювати заголовки та описи новин або застосовувати алгоритми на основі ключових слів, такі як Levenshtein distance та TF-IDF. У реляційних БД проблема дублювання вирішується завдяки унікальності полів, наприклад, збігу заголовка, змісту та посилання.

Ще один виклик - це інтернаціональність даних. Оскільки RSS-стрічки можуть надходити різними мовами та в різних часових поясах, необхідно забезпечити їхню уніфікацію. Для автоматичного визначення мови можна використовувати бібліотеки на кшталт langdetect чи langid, тоді як час публікації слід зводити до стандарту UTC. Якщо часовий пояс не вказаний явно, його можна визначати на основі домену сайту або конфігурації для кожного джерела.

Висновок. Загалом обробка RSS-даних потребує створення окремого модуля з чітким розподілом функцій: автономний збір даних за розкладом, парсинг із урахуванням нестандартних структур, видалення дублікатів, визначення мови та часового поясу, а також сортування за часом публікації, релевантністю та іншими критеріями. Правильна організація цього процесу дозволяє не лише знизити навантаження на зовнішні джерела, а й забезпечити користувачам швидкий доступ до актуальної та унікальної інформації.

Структуровані рішення на основі RSS можна легко масштабувати: додавання нових джерел новин або фільтраційних модулів не вимагає повної переробки системи. Завдяки чіткій структурі, RSS-застосунки можуть реалізовувати фільтрацію шкідливих даних, перевірку підписів і сертифікатів, що актуально при роботі з великими інформаційними потоками. Таким чином, технологія RSS задає чіткі вимоги до структури програмних рішень, стимулює використання модульного підходу, спрощує масштабування і забезпечує стабільну передачу інформації. Вона залишається актуальною в епоху цифрових технологій,

особливо у сфері медіа, освіти та IT. Перспективним напрямком дослідження може стати використання ШІ для аналізу, класифікації та персоналізації RSS-контенту. Завдяки машинному навчанню, системи можуть адаптуватися до інтересів користувача, фільтрувати неактуальну або недостовірну інформацію. Технологія RSS ще має багато можливостей для розвитку — особливо в контексті автоматизації, персоналізації та цифрової безпеки.

Література

1. Офіційний веб-сайт посібник *rssboard* [Електронний ресурс]: [Веб-сайт]. – URL: https://www.rssboard.org/rss-specification).
2. Alen Šimec, Mia Čarapina, Sanja Duk RSS as medium for information and communication technology Хорватія 2011.
3. Статистика використання RSS [Електронний ресурс]: [Веб-сайт]. – URL: https://trends.builtwith.com/feeds/RSS.
4. Ramkrishna Patel; Vikas Choudhary; Deepika Saxena; Ashutosh Kumar Singh Review of Stock Prediction Using Machine Learning Techniques 2021 Tirunelveli, India
5. «Is RSS still relevant» веб-стаття [Електронний ресурс]: [Веб-сайт]. – URL: https://modulards.com/en/still-being-the-rss-relevant-in-2024-an-glimpse-at-its-possibilities/.
6. Robert Stelzle & Elias Koch «Using RSS to keep track of the latest Journal Articles» Elephant in the room 19 july 2023.
7. Jim Doree RSS: A Brief Introduction Journal of Manual & Manipulative Therapy 2007.
8. Atul Sajjanhar, Ying Zhao Web Service to Deliver Filtered RSS Items to a Mobile Application ChinaGrid Annual Conference (ChinaGrid), 2012 Seventh.
9. Somnath Paulchoudhury Reading RSS Feed in Google Sheets Web Broadcast July 2020.
10. Basavaraj Kumbar, Mulla K. R A survey study on awareness about rss feeds and social book markings of working library professionals in engineering colleges in karnataka: a study International Journal of Library Science and Research December 2021.
11. RSS Feeds in 2024: Are You Missing Out on This Quiet Powerhouse for Your Website? - Ross Gerring [Електронний ресурс]: [Веб-сайт]. – URL: https://www.itomic.com.au/rss-feeds-in-2024-are-you-missing-out-on-this-quiet-powerhouse-for-your-website/.
12. The Enduring Value of RSS Feeds: Connecting Content and Community - 99 Park Row, доступ отримано січня 21, 2025. URL: https://www.99parkrow.com/2024/09/the-enduring-value-of-rss-feeds-connecting-content-and-community/.

References

1. Ofitsiinyi veb-sait posibnyk rssboard [Elektronnyi resurs]: [Veb-sait]. – URL: https://www.rssboard.org/rss-specification).
2. Alen Šimec, Mia Čarapina, Sanja Duk RSS as medium for information and communication technology Khorvatiia 2011.
3. Statystyka vykorystannia RSS [Elektronnyi resurs]: [Veb-sait]. – URL: https://trends.builtwith.com/feeds/RSS.
4. Ramkrishna Patel; Vikas Choudhary; Deepika Saxena; Ashutosh Kumar Singh Review of Stock Prediction Using Machine Learning Techniques 2021 Tirunelveli, India
5. «Is RSS still relevant» veb-stattia [Elektronnyi resurs]: [Veb-sait]. – URL: https://modulards.com/en/still-being-the-rss-relevant-in-2024-an-glimpse-at-its-possibilities/.
6. Robert Stelzle & Elias Koch «Using RSS to keep track of the latest Journal Articles» Elephant in the room 19 july 2023.
7. Jim Doree RSS: A Brief Introduction Journal of Manual & Manipulative Therapy 2007.
8. Atul Sajjanhar, Ying Zhao Web Service to Deliver Filtered RSS Items to a Mobile Application ChinaGrid Annual Conference (ChinaGrid), 2012 Seventh.
9. Somnath Paulchoudhury Reading RSS Feed in Google Sheets Web Broadcast July 2020.
10. Basavaraj Kumbar, Mulla K. R A survey study on awareness about rss feeds and social book markings of working library professionals in engineering colleges in karnataka: a study International Journal of Library Science and Research December 2021.
11. RSS Feeds in 2024: Are You Missing Out on This Quiet Powerhouse for Your Website? - Ross Gerring [Elektronnyi resurs]: [Veb-sait]. – URL: https://www.itomic.com.au/rss-feeds-in-2024-are-you-missing-out-on-this-quiet-powerhouse-for-your-website/.
12. The Enduring Value of RSS Feeds: Connecting Content and Community - 99 Park Row, dostup otrymano sichnia 21, 2025. URL: https://www.99parkrow.com/2024/09/the-enduring-value-of-rss-feeds-connecting-content-and-community/.

Lagovskyi V. V., Lahovskyi O. V., Nizhehorodtsev V.O., Filonenko M. M., Skaskiv L.V. Structural features of software solutions with RSS technology

The article analyzes structural features and architectural approaches for developing software solutions integrating RSS (Really Simple Syndication) technology.

The relevance of RSS for automated collection, processing, and distribution of information from web resources is emphasized. The technology's value lies in its standardized XML format, simplifying aggregation and enabling effective management of information flows.

Key advantages of RSS are reviewed: optimized data retrieval, user control over content without algorithms, anonymity, a unified ad-free format, reliability, and offline access. Drawbacks analyzed include variable content completeness, lack of interactivity, the need for specialized aggregators, risk of information overload, and limited multimedia support.

Technical aspects of building RSS systems are discussed. The critical importance of autonomous data collection and storage in the aggregator's own database is substantiated. This prevents load on sources, increases access speed, and enables the implementation of search, filtering, and analysis. The feasibility of using document-oriented DBs (flexibility) versus relational DBs (structuredness, integrity, duplicate elimination) is compared.

Attention is paid to the problem of news duplication; deduplication methods based on title/description comparison and textual/semantic similarity analysis are considered. Internationalization challenges are highlighted: the need for language auto-detection and unification of publication time to UTC.

An architectural approach with a separate software module for RSS processing is proposed. The module includes subsystems for collection/parsing, processing (deduplication, unification, language/time detection, classification), DB data storage, and sorting/searching (with indexing). The need for autonomous module operation, error handling, and interaction via API is stressed.

The conclusion states that RSS remains an effective tool but requires well-thought-out structural solutions. Proper architecture ensures efficient aggregation, reduces source load, provides scalability, and allows feature integration. Using AI and Machine Learning for deeper analysis and personalization of RSS content is a promising direction.

Keywords: *RSS, software module, database, software structure, xml, parsing, deduplication.*

Лаговський Володимир Вікторович – к.е.н., доцент кафедри комп'ютерних та інформаційних технологій і систем Державного податкового університету (Ірпінь), vvlagovsky2@gmail.com.

Лаговський Олесь Володимирович – магістр Факультету фінансів та цифрових технологій, освітньо-професійної програми «Інформаційні управляючі системи і технології в економіці» Державного податкового університету (Ірпінь), oles_l@i.ua.

Ніжегородцев Владислав Олександрович – к.пед.н., доцент кафедри комп'ютерних та інформаційних технологій і систем Державного податкового університету (Ірпінь), nizhegorodcev@ukr.net.

Філоненко Михайло Миколайович – к. фіз.-мат. н., завідувач кафедри комп'ютерних та інформаційних технологій і систем Державного податкового університету (Ірпінь), filonenko.mykhaylo@gmail.com.

Скасків Лілія Василівна – к.фіз.-мат.н., доцент кафедри вищої та прикладної математики Державного податкового університету (Ірпінь), liliaskaskiv@gmail.com.

Стаття подана 09.03.2025.