МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп’ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувачка кафедри

Савченко А.С.

“___” _____ 2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ “МАГІСТРА”

ЗА СПЕЦІАЛІЗАЦІЄЮ “ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)”

Тема: “Веб-застосунок для аналізу кредитних пропозицій фінансового ринку”

Виконавець: Скоцик Іван Сергійович

Керівник: Колісник Олена Василівна

Нормоконтролер: Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

Галузь знань, спеціальність, спеціалізація: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології (за галузями)" .

ЗАТВЕРДЖУЮ

Завідувачка кафедри

_____ Савченко А.С.

"___" _____ 2020 р.

З А В Д А Н Н Я

на виконання дипломної роботи студента

Скоцика Івана Сергійовича

(прізвище, ім'я, по батькові)

1. Тема роботи: "Веб-застосунок для аналізу кредитних пропозицій фінансового ринку" затверджена наказом ректора №1891/ст. від 02.10.2020р.

2. Термін виконання роботи: з 05.10.2020р. по 31.12.2020р.

3. Вихідні дані до роботи: оптимізація продуктивності веб-застосунку.

4. Зміст пояснювальної записки: проблематика неперервного розвитку та оптимізації додатку. Огляд сучасних інструментів для відстеження продуктивності веб-застосунку. Огляд інструментів для моніторингу вразливостей застосунку та його залежностей. Мінімізація ризиків втрати даних компанії та клієнтів в наслідок кібератаки.

5. Перелік обов'язкового графічного матеріалу: рисунки та презентація Power Point.

6. Календарний план-графік

№ з/п	***Завдання***	***Термін виконання***	***Підпис керівника***
1	Провести аналіз існуючого веб-застосунку для аналізу кредитних пропозицій фінансового ринку	05.10.2020р. – 15.10.2020р.	
2	Дослідити проблематику неперервного розвитку та оптимізації додатку	16.10.2020р. – 22.10.2020р.	
3	Огляд сучасних інструментів для відстеження продуктивності веб-застосунку	23.10.2020р. – 01.11.2020р.	
4	Огляд інструментів для моніторингу вразливостей застосунку та його залежностей	02.11.2020р. – 12.11.2020р.	
5	Оптимізувати та мінімізувати ризик втрати даних компанії та клієнтів в наслідок кібератаки	13.11.2020р. – 22.11.2020р.	
6	Оформити пояснювальну записку дипломної роботи	23.11.2020р. – 04.12.2020р.	
7	Оформлити графічну частину дипломної роботи	05.12.2020р. – 10.12.2020р.	
8	Підготовити доповідь та презентацію до захисту дипломної роботи	11.12.2020р. – 14.12.2020р.	
9	Підготовитися до захисту дипломної роботи	15.12.2020р. – 21.12.2020р.	

7. Консультація з окремого(мих) розділу(ів) роботи:

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв

8. Дата видачі завдання: 05.10.2020р.

Керівник дипломної роботи _____ Колісник О.В.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Скоцик І.С.

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Веб-застосунок для аналізу кредитних пропозицій фінансового ринку» містить 83 сторінки, 47 рисунків, 9 наукових джерел.

Ключові слова: ПРОДУКТИВНІСТЬ, МОНІТОРИНГ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ШВИДКІСТЬ ВИКОНАННЯ, РЕФАКТОРИНГ, БЕЗПЕКА ДАНИХ, ІНСТРУМЕНТИ, ПРОГРАМНИЙ КОД, ВРАЗЛИВІСТЬ, БАЗА ДАНИХ.

Мета дипломного проекту: вибір інструментів для моніторингу продуктивності додатку та відстеження вразливостей безпеки застосунку та його залежностей. Інтеграція та використання вибраних інструментів для зменшення ризиків безпеки та покращення продуктивності додатку.

Об'єкт дослідження: Веб-застосунок для аналізу кредитних пропозицій фінансового ринку.

Предмет дослідження: Моніторинг продуктивності та відстеження вразливостей безпеки веб-застосунку.

Методи та задачі дослідження: Аналіз застосунку та його залежностей на рахунок можливих вразливостей безпеки, мінімізація ризиків втрати даних користувачів та компанії, аналіз швидкості виконання програмного коду додатку, рефакторинг та зменшення часу виконання підпрограм інформаційної системи.

Результат роботи: Інтегрований інструмент NewRelic для моніторингу продуктивності додатку, з його використанням зменшений час виконання фрагментів програмного коду. Інтегрований інструмент відстеження вразливостей безпеки застосунку та його залежностей, з його використанням знижений ризик втрати даних користувачів та компанії.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ - програмне забезпечення

БД - база даних

OSS - open-source software, відкрите програмне забезпечення

SQL - structured query language, мова структурованих запитів

NVD - Національна база вразливостей, National vulnerability database

NIST - Національний інститут стандартів і технологій США, National Institute of Standards and Technology

ВСТУП

У наш час людство все більше віддає перевагу інформаційним технологіям, це зумовлено потребою людини встигати за змінами, обробляти величезну кількість інформації, яка надходить звідусіль. Тому інформаційні технології мають сьогодні пріоритетне значення в багатьох сферах діяльності й визначають майбутній розвиток суспільства.

Одним з найбільш швидкорозвиваючихся секторів інформаційних технологій є комп'ютерна інформаційна мережа (Інтернет). Вона з'явилася в 1969 році, першим хто запропонував ідею Всесвітньої комп'ютерної мережі був американський науковець, математик Джозеф Ліклайдер у 1962 році. З початку ідея такої мережі призначалася для військових, вона надал б можливість швидко обмінюватися інформацією між частинами у різних куточках світу, але так сталося що ця ідея змінила світ цивільного населення.

Не можливо представити життя сучасної людини без інтернету, більшість інформації, яку вона отримує щодня поступає саме з всесвітньої мережі. Отримання прогнозу погоди, новин, переписка електронною поштою, купівлю в онлайн-магазинах, інтернет-кассах, прослуховування музики, підтримка зв’язку з рідними на будь-якій відстані, навіть оплата комунальних послуг не виходячи з дому, все це стало неодмінною частиною нашого життя.

Інтернет з кожним днем все більше поглиблюється у різні сфери і значною мірою змінює їх. Цього року у зв'язку з карантином інтернет допоміг перевести велику частину економіки в онлайн і не дав зупинити розвиток людства.

З технічної точки зору інтернет – це всесвітня комп'ютерна мережа, яка представляє собою інформаційне середовище, з мільйонів локальних та глобальних мереж. Інтернет надає широкий спектр інформаційних ресурсів та послуг, таких як взаємопов'язані гіпертекстові документи та програми Всесвітньої павутини (WWW), електронна пошта, телефонія та обмін файлами. До середини 2015 року кількість користувачів досягла 3,3 млрд осіб. Багато в чому це було обумовлено значним

поширенням стільникових мереж з доступом в Інтернет стандартів 3G і 4G, розвитком соціальних мереж і здешевленням вартості інтернет-трафіку [1]. Він відіграє важливу роль, в житті суспільства, слугує фізичною основою доступу до веб-сайтів і багатьох систем передачі даних.

Не обійшов стороною він і бізнес. Мережа допомагає знаходити нових клієнтів і партнерів, продавати свої продукти і послуги, замовляти сировину, контролювати свої витрати і надходження, просувати компанію, рекламувати продукцію і багато іншого. Бізнес стає все більш глобальним, за допомогою інтернету компанії виходять на нові ринки, співпрацюють з клієнтами та партнерами з усіх куточків світу.

Значною мірою змінилася і сфера фінансів, люди звикли проводити валютні операції в інтернеті, переводити кошти між рахунками, оформлювати кредити, депозити, змінювати ліміти на операції та робити інші грошові маніпуляції не виходячи з дому. Навіть з'являються так звані нео-банки, це фінансові установи, які не мають відділень, що обслуговують клієнтів. Такі установи обслуговують клієнтів повністю віддалено через всесвітню мережу за допомогою веб-сайтів, мобільних додатків та застосунків для персональних комп'ютерів.

Така різка зміна такого консервативного сектору економіки як фінансові установи, в першу чергу пов'язана з бажанням людей зберегти свій час і провести час за улюбленою справою замість того щоб цілий день заповнювати бланки у відділенні банку і стояти у черзі.

Це спричинило появу конкуренції не тільки на основі кращою пропозиції та стабільності банку, а й зручності, доступності і якості застосунків, онлайн-підтримки та веб-сервісів.

Почали з'являтися сервіси, які допомагають користувачам одночасно порівняти кредитні пропозиції різних банків та інших фінансових установ, прочитати відгуки існуючих клієнтів, вибрати зручний для себе термін та ставку.

Такі сервіси стали дуже популярні, бо максимально скорочують час отримання кредиту та допомагають знайти раніше невідому установу з кращими умовами.

Метою даної роботи є покращення веб-застосунку для аналізу кредитних пропозицій фінансового ринку.

РОЗДІЛ 1

ОГЛЯД ІСНУЮЧОГО ВЕБ-ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ КРЕДИТНИХ ПРОПОЗИЦІЙ ФІНАНСОВОГО РИНКУ ТА ІНСТРУМЕНТІВ МОНІТОРИНГУ ПРОДУКТИВНОСТІ

1.1. Опис бізнес складової компанії та огляд конкурентів

Бізнес компанії базується на веб-застосунку, що дозволяє користувачу порівняти пропозиції різних фінансових установ для взяття кредиту на навчання або іпотеку. Компанія веде свою діяльність виключно у Сполучених Штатах Америки, але її послугами можуть скористатися громадяни будь-якої країни світу, що легально знаходяться на території США. Для різних типів фінансових продуктів компанія має від восьми до шістнадцяти партнерів-кредиторів від яких користувач може отримати детальну пропозицію, яка буде включати в себе інформацію про термін та розмір виплат, процентну ставку, її тип (фіксована або плаваюча). Приклад сторінки зі списком пропозицій зображено на рис 1.1.

На ринку існує декілька сервісів, що надають користувачам схожі можливості.

Компанія має декілька важливих переваг, яких немає у конкурентів:

- аналіз профілю користувача виконується на сервісах компанії, тобто веб-застосунок не виконує запити на API кредиторів, тому ніхто з них не знає про бажання клієнта взяти кредит і немає його контактів;
- завжди надається точна процентна ставка, в той час коли у деяких конкурентів вона надається у вигляді можливого проміжку;

За даними веб-порталу Statista [2] ринок іпотек у Сполучених Штатах Америки склав 16 трильйонів доларів у 2018 році. А ринок студентських кредитів складає півтора трильйона доларів за даними видання Forbes [3].

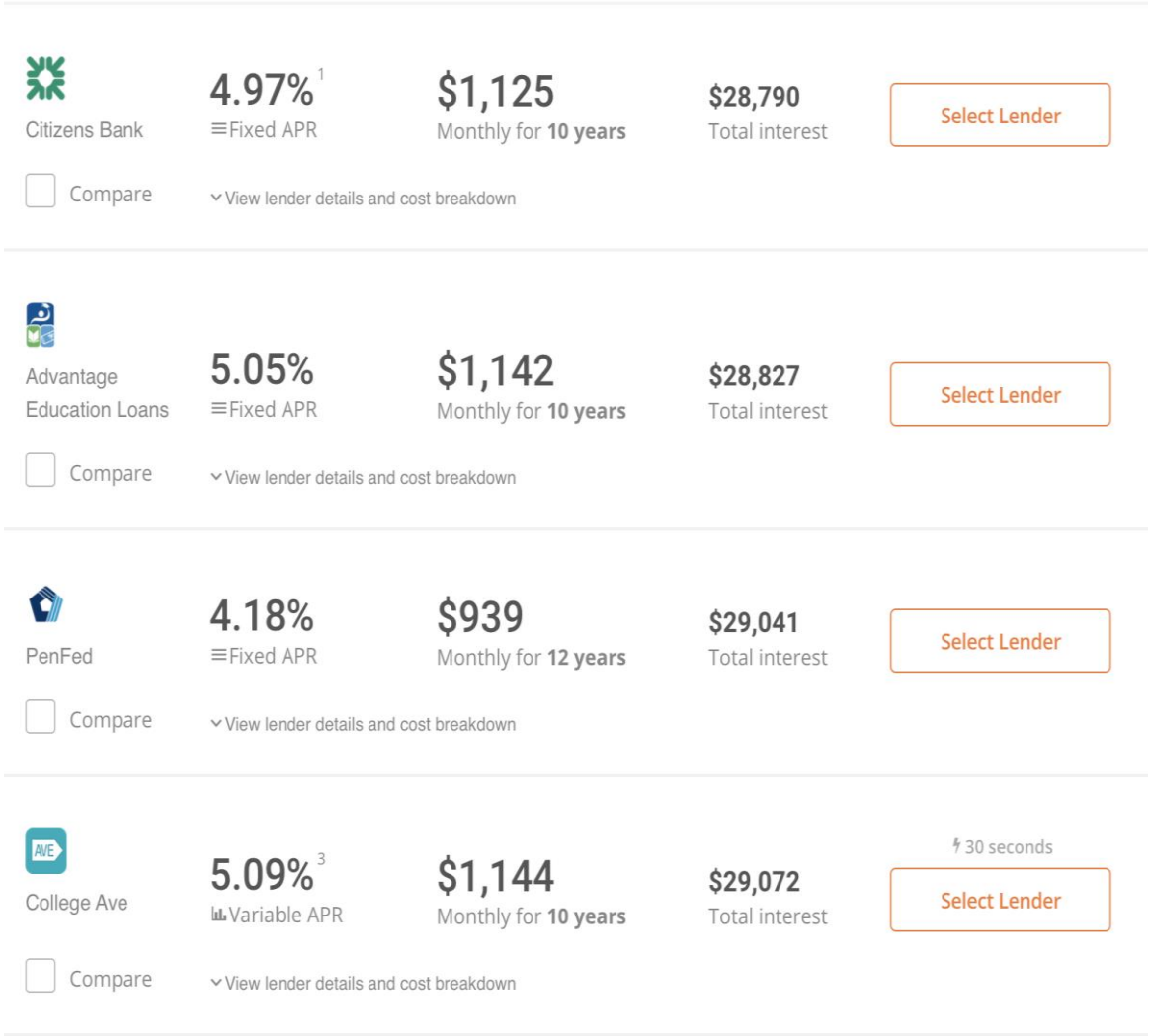


Рис. 1.1. Сторінка зі списком пропозицій

Такий об'єм ринку неодмінно привертає увагу багатьох великих компаній, що створює величезну конкуренцію. Тому компанії привертають велику увагу постійному вдосконаленню своїх інформаційних систем і застосунків. Одним з основних аспектів якості таких додатків є швидкість обробки даних користувача і отримання пропозицій, іншими словами швидкість виконання програмного коду. Аналіз профілю включає в себе величезну кількість обрахунків для кожного кредитора, що може зайняти деякий час. Одна з головних задач - це постійна оптимізація цих розрахунків.

Для обробки кредитного профайлу користувач вводить велику кількість персональної інформації, наприклад ssn (Social Security number) - номер соціального страхування. Втрата такої вразливої інформації може призвести до шахрайства. Тому фінансові IT-компанії звертають велику увагу на інформаційну безпеку, щоб захистити своїх користувачів, а разом і з тим свою репутацію.

Для оптимізації своїх інформаційних систем та веб-застосунків IT-компанії з фінансового сектору регулярно проводять аналіз їх технічного стану для пошуку фрагментів коду, які потребують покращення. А також щоденно перевіряють свої додатки щодо можливої загрози безпеки. В наш час все це можливо частково автоматизувати за допомогою спеціальних інструментів та сервісів.

1.2. Огляд інструментів для аналізу технічного стану існуючого веб-застосунку

Інструменти для аналізу технічного стану інформаційних систем також узагальнено називають APM від англійського Advanced Performance Management. Засоби APM - це компоненти, які містять повне рішення для управління продуктивністю додатків. Кожен інструмент працює в тандемі, щоб надати різноманітну інформацію, необхідну бізнесу, щоб забезпечити працездатність та доступність свого програмного забезпечення. Рішення APM відрізняються унікальними наборами інструментів моніторингу, але найкращі включають, зокрема, три: аналіз впливу на бізнес, моніторинг кінцевих користувачів та штучний інтелект.

Моніторинг користувачів системи включає відстеження фактичного досвіду ваших клієнтів. Це можна зробити двома основними способами: моделюючи та тестуючи синтетичні взаємодії користувачів або пасивно контролюючи взаємодію з кінцевими користувачами, коли користувачі взаємодіють із вашими програмами в режимі реального часу. Завдяки цим можливостям команди підтримки додатків можуть оптимізувати кожну точку дотику на шляху користувача, фіксуючи та виправляючи помилки, збої, деталі завантаження сторінки та інші критичні показники, перш ніж кінцевий користувач помітить проблему.

Інструменти моніторингу продуктивності додатків (APM) стали важливими для підприємств, оскільки їхні технології працюють із дедалі складнішими програмами.

При виборі такого інструменту потрібно звернути увагу на такі критерії:

- Підтримка стеку технологій, який використовувався при створенні вашої інформаційної системи. Такі APM зазвичай використовують специфіку тієї чи іншої технології або мови програмування для точнішого аналізу, тому вони найчастіше працюють з обмеженою кількість мов.
- Тарифний план. Усі інструменти моніторингу продуктивності додатків відрізняються ціною за використання та функціоналом включеним у той чи інший план. Тому потрібно зрозуміти, який функціонал потрібен саме нашій організації, щоб не переплачувати за непотрібні функції. Ціна таких APM завжди прямо пропорційна навантаженню додатку та кількості операцій які потрібно аналізувати. Також великі компанії можуть розраховувати на знижку через великі нагрузки.
- Простота використання. Вивчення нового інструменту та його інтеграція у вже існуючу систему може зайняти багато часу у команди інженерів. Цей час також потрібно враховувати, оскільки він оплачується з коштів компанії. Тому потрібно вибирати APM з зрозумілою, всеохоплюючою документацією та прикладами інтеграції.
- Робота сервісу аналізу з хмари. Деякі інструменти моніторингу продуктивності додатків працюють з хмари, а деякі потрібно самостійно встановлювати на власний сервер та підтримувати його стан. Краще вибирати сервіс, роботу якого не потрібно буде підтримувати самостійно.
- Наявність потрібних метрик для аналізу технічного стану. Потрібно щоб APM надавав саме ті метрики які важливі для вашою інформаційної системи. Наприклад, вам не важливі метрики стосовно інтеграції системи зі сторонніми сервісами, якщо ваш додаток немає таких інтеграцій.
- Технічна підтримка. Важливо, щоб компанія, яка поставляє інструменти моніторингу продуктивності додатків, мала змогу консультувати та

відповідати на питання від команди інженерів, що будуть виникати під час інтеграції APM у існуючу інформаційну систему.

- Безпека даних. Інструменти для аналізу технічного стану інформаційних систем будуть мати доступ до всіх фрагментів додатку а також даних користувачів, тому потрібно вибирати APM що притримується високих стандартів безпеки. Свідчити про це можуть наявні сертифікати безпеки від всесвітньо відомих організацій у сфері кібербезпеки або звіти про проведені зовнішні аудити незалежними компаніями.

Спираючись на ці критерії потрібно обирати інструмент моніторингу продуктивності додатків, який підходить саме нашій організації під її потреби та фінансові можливості. Розглянемо існуючі на сьогоднішній день APM та оберемо інструмент, що надалі будемо використовувати для аналізу технічного стану інформаційної системи та для подальшого покращення показників продуктивності додатку.

1.2.1. SolarWinds

Якщо ваш IT-відділ підтримує велику організацію, вам знадобиться APM, який інтегрується з іншими модулями моніторингу інфраструктури. SolarWinds надає набір інструментів підтримки інфраструктури, які написані на одній платформі. Всі ці інструменти обмінюються даними, тож чим більше їх ви впровадите, тим кращим буде розуміння продуктивності ваших програм та всіх служб, які їх підтримують.

Продуктивність програм дуже тісно пов’язана з продуктивністю хостів, на яких працює програмне забезпечення, тому дуже гарною ідеєю є використання інструменту, який поєднує моніторинг додатків та серверів, і саме це забезпечує дане програмне забезпечення. Сервери, які ви включаєте у своє середовище управління, можна розподілити на декількох сайтах, а також засіб буде контролювати хмарні сервіси, що працюють в хмарних сервісах від Microsoft та Amazon, відповідно Azure та AWS.

Монітори та дисплеї інформаційної панелі не покладаються на стратегію "одного розміру". Програмне забезпечення адаптується для відображення критичної інформації, що стосується кожного додатка, тому ви побачите різні показники для програми управління базами даних, аніж веб-серверу. Ця адаптованість зумовлена низкою шаблонів. Монітор серверів та додатків включає понад 1200 шаблонів, що охоплюють усі основні програми, які сьогодні доступні на ринку. Приклад панелі моніторингу інформаційної системи зображено на рис. 1.2.

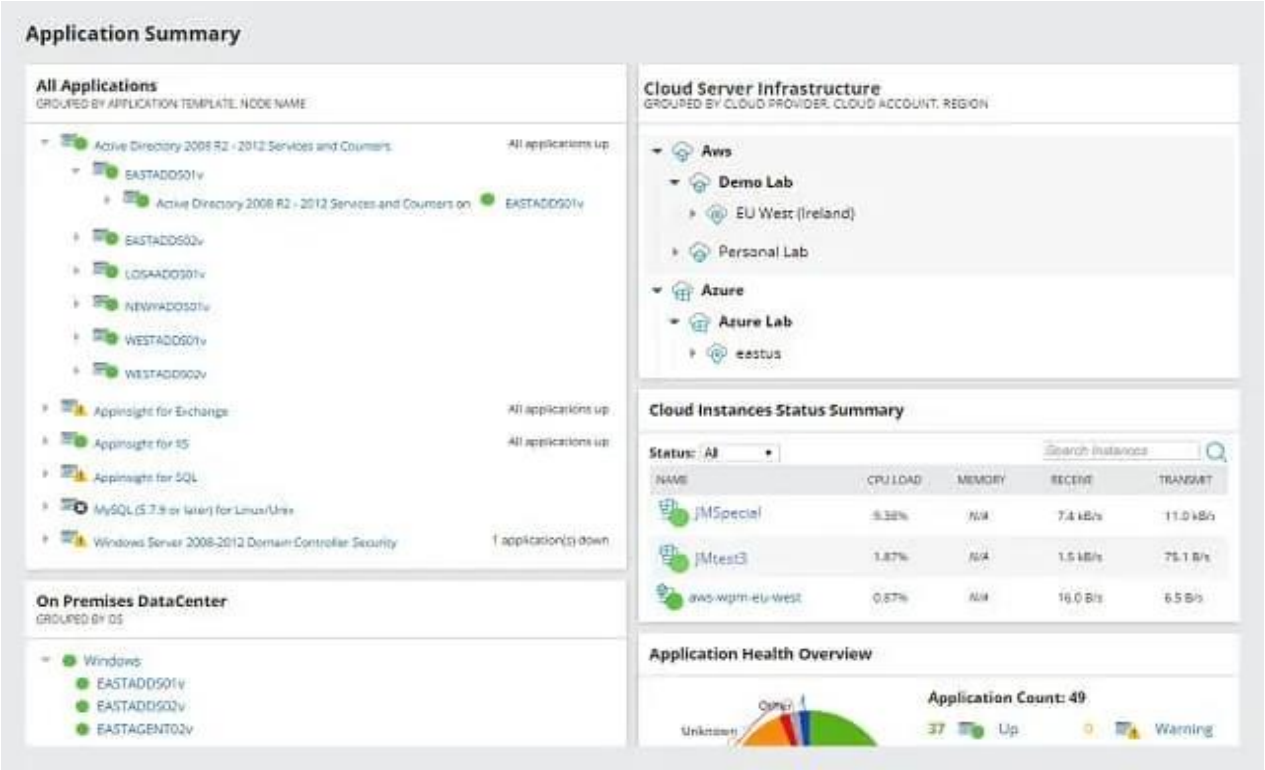


Рис. 1.2. Панель моніторингу SolarWinds

Елемент моніторингу сервера цього програмного забезпечення поширюється на статуси сервера зберігання, який може бути локальним або в хмарі. Ви можете розширити глибину інформації, зібраної цим модулем, додавши на монітор ресурсів зберігання SolarWinds.

Чудовою особливістю цього інструменту є модуль AppStack. Це дає вам візуальне представлення програм, які працюють у вашій системі, з усіма рівнями підтримки служб та обладнання. Отже, ви можете бачити стан програми, метрики для сервера додатків, на якому вона працює, а також, якщо ви працюєте з

віртуальним середовищем, усі сервери та мережі, що лежать в основі цієї віртуальної машини. Це особливо корисно, якщо ви працюєте з програмами у хмарному середовищі або на віддалених серверах, оскільки це допомагає вам миттєво побачити, де основні служби тягнуть продуктивність вашого додатка вниз.

SolarWinds продає безстрокову ліцензію на це програмне забезпечення, тому як тільки ви заплатите за нього, воно назавжди залишиться вашим. Однак ціна включає лише перший рік підтримки та оновлення інструменту, тому вам доведеться платити за підтримку в наступні роки. Це програмне забезпечення доступне лише для завантаження в середовищі Windows Server.

APM може аналізує виконання та продуктивність коду, написаного на семи найбільш часто використовуваних мовах для веб-додатків: Java, Node.js, PHP, .NET, Python, Ruby та Go.

Моніторинг встановлюється за лічені хвилини, і він автоматично виявляє всі ваші програми та статуси сервера. Якщо ви також придбаєте SolarWinds Network Performance Monitor і SolarWinds NetFlow Traffic Analyzer, ви одразу ж контролюватиме всі фактори, які можуть спричинити збій або погану роботу програм вашого бізнесу.

SolarWinds Server & Application Monitor ідеально підходить для моніторингу додатків та їх допоміжної інфраструктури. Він автоматично виявляє програми у вашій мережі разом із їх екологічними залежностями. Потужний інструмент для встановлення першопричин під час усунення несправностей.

Вимогою до вашої інформаційної системи є лише використання Windows Server.

Ключові особливості:

- Підтримує багато популярних фреймворків та мов (Java, .NET, Python, PHP, Ruby тощо).
- 150+ інтеграцій та плагінів (AWS, Apache, MongoDB, NGINX, MySQL тощо).
- Інформаційні панелі в реальному часі та налаштування попередження.
- Перегляд поведінки окремих запитів у режимі реального часу.

- Збирати та співвідносити власні показники (команди cURL, агенти з відкритим кодом).
- Впровадження методів відстеження під час виробництва.
- Виявлення та виправлення вузьких місць у програмах.

1.2.2. Datadog

Datadog пропонує хмарний моніторинг інфраструктури, програм та журналів подій. Ви можете зареєструватися лише для монітора продуктивності програми або використовувати всі три системи в поєднанні для покращення видимості служби. Усі ці послуги стягуються за передплатою, і ціна залежить від кількості хостів, якими ви керуєте. Система може контролювати локальні, хмарні та гібридні системи.

Datadog відстежує кожен запит на додаток і відстежує його доставку, що включає вивчення подій у стеках додатків. Будь-які аномалії, які з’являються під час доставки програми, викликають тригер тривоги. Ці сповіщення накопичуватимуться, тож ви миттєво зможете побачити на інформаційній панелі, яка програма має проблеми, а яка служба, здається, є причиною проблеми.

Інструмент аналізує події, перевіряючи продуктивність коду, написаного на семи найбільш часто використовуваних мовах для онлайн-служб: Java, Node.js, PHP, .NET, Python, Ruby та Go.

Інформаційна панель дуже приваблива і включає графічне представлення даних для полегшення розпізнавання стану, її можна побачити на рис 1.3. Ви зможете переключати подання, щоб зосередити увагу на кінцевих користувачах, які мають найгіршу продуктивність, і збільшити масштаб окремих користувачів, щоб вивчити проблеми, з якими стикається доставка програм у цьому сеансі.

Рис. 1.3. Панель моніторингу Datadog

Ви можете налаштувати інформаційну панель і навіть створити різні подання для призначення окремим членам команди. Система включає API, тому ви навіть можете інтегрувати елементи інформаційної панелі Datadog у власні сторінки моніторингу корпоративної системи. Взаємодія між програмним забезпеченням Help Desk та Datadog зменшує потребу в службі підтримки перемикатися між програмами, щоб пов’язувати повідомлення про помилки людини з автоматизованими попередженнями.

Ви отримаєте додаткову допомогу від Datadog Real-Time APM, якщо інтегруєте його з інфраструктурним пакетом компанії. Два пакети призначені для спільного використання даних, тому відстеження продуктивності додатків до статусів інфраструктури є безперебійним.

Існує безкоштовна версія монітора інфраструктури, який буде керувати даними з п'яти хостів. Ви не можете отримати APM безкоштовно постійно, але ви можете отримати 14-денну безкоштовну пробну версію.

Ключові особливості:

- Підтримує популярні веб-фреймворки (Django, Ruby on Rails, Gin та Spring).
- Використовує машинне навчання для виявлення помилок.
- Інтеграція сповіщень (Slack, HipChat та Campfire).
- 250+ інтеграцій (AWS, Apache, Azure, Docker, GitHub, Java, Jira, Kubernetes, Microsoft Team тощо).
- Моніторинг контейнерів, хмарних екземплярів, локальних та гібридних архітектур.
- Потоки даних карти.
- Налаштовувані панелі інструментів.

1.2.3. NewRelic

NewRelic випускає цілий ряд інструментів моніторингу, і APM є одним із таких. Інші доступні модулі - NewRelic Infrastructure, яка спостерігає за станом базового обладнання та послуг, NewRelic Browser, який допомагає онлайн-компаніям відстежувати досвід користувачів, та NewRelic Insights, який є модулем аналізу. APM NewRelic працюватиме без будь-якого з цих інших модулів. Однак поєднання APM з іншими модулями дасть вам глибше знання про причини проблем та допоможе вам вдосконалити ваш Інтернет-сервіс.

Система NewRelic - це онлайн-сервіс, і якщо вашу інфраструктуру також надають хмарні служби, програмне забезпечення для моніторингу буде інтегровано безпосередньо з ними. Монітор можна інтегрувати в сервери AWS, Google, Microsoft Azure та Rackspace. Він також може інтегруватися з вашими власними локальними серверами.

Служба аналізує події, перевіряючи продуктивність коду, написаного на семи найбільш часто використовуваних мовах для онлайн-служб: Java, Node.js, PHP, .NET, Python, Ruby та Go. Це означає, що він також зможе відстежувати діяльність вашого користувацького програмного забезпечення, а не лише добре відомих попередньо написаних програм.

APM контролює весь ваш інтернет-трафік, включаючи той, що надходить від мобільних додатків. Ця послуга призначена для онлайн-бізнесу, такого як інформаційні веб-сайти та веб-магазини. Інструмент може підтримувати міграцію програм з локальних серверів на хмарні служби, а також корисний під час управління випуском нових продуктів або оновлення існуючих веб-служб. Як передача даних на хмарний сервер NewRelic, так і зберігання даних охоплюються шифруванням та аутентифікацією користувачів.

Інформаційна панель служби містить багато графічних елементів, які допоможуть швидко визначити проблеми та успіхи в роботі. Приклад панелі моніторингу інформаційної системи зображено на рис. 1.2.

Рис. 1.4. Панель моніторингу NewRelic

Основна інформація, яка потрібна, знаходиться на дисплеї моніторингу програм. Ви побачите час відгуку, показники пропускної здатності та коефіцієнти помилок як у вигляді цифр, так і у вигляді графіків з вибіркою часу. Ви також можете відстежувати подорожі окремих відвідувачів через ваш веб-сайт, а також

отримувати огляд роботи програми. Інтерфейс включає моніторинг баз даних, і ви можете здійснювати пошук зібраних даних для аналізу проблем продуктивності.

Інформаційна панель включає функції спільної роботи команди, такі як обмін інформацією та створення нотаток. Ви можете налаштувати інформаційну панель і створювати групи користувачів, надаючи різним членам команди доступ до різних подань даних та елементів керування. APM NewRelic можна інтегрувати із системами управління довідковою службою.

Для APM NewRelic доступні три плани. Версія Pro APM є вершиною рядка, і вона доступна на 14-денній безкоштовній пробній версії. Скорочена версія, яка називається APM Essentials, не має аналітичних функцій служби. Важливим елементом хмарної служби є термін зберігання даних, що дозволяє запитувати продуктивність з часом та отримувати аналітичну інформацію про доставку ваших додатків. Якщо ви не настільки зацікавлені в цьому збереженні даних, ви можете отримати APM Lite, який є безкоштовною версією основних елементів APM, але дані зберігаються лише 24 години.

Ключові особливості:

- Відстежуйте оператори SQL, відповідальні за низьку продуктивність.
- Миттєво пропонуємо тенденції продуктивності.
- Дозволяє аналіз запитів SQL.
- Надайте розширені сповіщення.
- Діагностика на рівні коду.
- Відстеження перехресних додатків.

1.2.4. AppDynamics

AppDynamics розпочала свою діяльність як незалежна компанія, але зараз вона належить компанії Cisco Systems. Залучення цього мережевого гіганта має призвести до того, що ця система моніторингу перетвориться на лідера галузі, тож слід спостерігати за цим.

Інформаційна панель служби дуже зайнята, і вона зосереджена на ваших програмах та службах, які їх підтримують. Інший погляд на дані про ефективність

слід за випадками користувачів, щоб показати швидкість доставки та попит на різні послуги та програми вашого підприємства. Оскільки в цьому інструменті майже немає метрик інфраструктури, ви, мабуть, обрали б AppDynamics для онлайн-бізнесу, який повністю надається хмарними службами.

Карта стеку програм, показана на рис 1.5, пояснює тенденції продуктивності всіх програм, які активні у вашій системі на дисплеї моніторингу в режимі реального часу. Служби, що підтримують ці програми, також включені, і проблеми, висвітлені в цих підтримуючих системах, можуть висвітлювати проблеми стану. Ці проблеми, пов'язані з базовою інфраструктурою, потребують вивчення за допомогою окремого пакету моніторингу.

Програмне забезпечення відстежує ваше прикладне програмне забезпечення в режимі реального часу та створює історичні дані, що дозволяє створити базовий рівень стандартної продуктивності. Одне лише це може підказати вам внести зміни у забезпечення пропускної спроможності, якщо воно відображає низьку продуктивність. Якщо на базовому рівні все виглядає добре, можна перейти до діагностичного інструменту, щоб перевірити, чи працюють програми цілодобово. Можно визначити, які програми можуть зіткнутися з проблемами під час високого попиту. Модуль взаємодії з користувачем дозволяє побачити, звідки походить попит на ваші послуги та які товари та послуги є найбільш популярними. Це інструмент для бізнес-аналізу, а також моніторингу продуктивності додатків.

Рис. 1.5. Карта стеку программ AppDynamics

AppDynamics доступний у безкоштовній версії, яка називається AppDynamics Lite, а також є платна версія, яка доступна на 15-денній безкоштовній пробній версії. Стандартний платний пакет називається APM Pro, але є два вищих пакети: APM Advanced, який додає метрики сервера та мережі, та APM Peak, який також включає функції маркетингового аналізу. Послуга надається в хмарі за замовчуванням, але замість цього ви можете вимагати встановлення програмного забезпечення. Якщо ви використовуєте хмарну версію, вам все одно доведеться встановити деяке програмне забезпечення. Це агент сервера додатків, і його можна встановити лише в 64-розрядному середовищі Linux та 64-розрядному Windows.

Ключові особливості:

- Перегляд конкретних даних щодо відвідувань користувачів.
- Видимість БД / сервера / програми.
- Діагностика на рівні коду.
- Знімки транзакцій.
- Підтримка технологій, таких як Java, .NET, PHP, Node.js та інші.
- Показники рівня машини та інфраструктури.

- 150 інтеграцій (AWS Cloud, PagerDuty, Docker тощо).

1.2.5. Splunk

Splunk застосовує штучний інтелект (ШІ) до свого програмного забезпечення для моніторингу. Компанія виробляє ряд інструментів моніторингу, включаючи управління інфраструктурою. Однак продукт, необхідний для моніторингу ваших програм, називається Splunk IT Service Intelligence. Незважаючи на те, що це окремий продукт для пакету моніторингу інфраструктури, він включає функції, що контролюють події та стан обладнання, щоб ви могли зрозуміти причини низької продуктивності додатків.

Елемент AI інструменту створює прогнозні звіти, які показують, де можуть виникнути проблеми з ємністю. Інструмент управління додатками також вказує на піки та мінімальні попити та передбачає, коли відбудеться наступний сплеск попиту. Це дозволить вам планувати партійні роботи в тихі часи та зменшити тиск на вашу інфраструктуру або придбати додаткові потужності. Якщо у вас вже є інструмент моніторингу додатків, ви можете просто вибрати додатковий модуль ITSI для продуктивності програми. Це проаналізує файли журналів вашого існуючого APM, щоб зосередитись на можливостях прогнозування. Інструмент також збирає інформацію з ваших журналів подій для виявлення проблем із наданням послуг.

Splunk пропонує 15-безкоштовну пробну версію пісочниці IT Service Intelligence. Цей пакет є доповненням до одного з основних пакетів моніторингу IT Splunk, який буде керувати вашою інфраструктурою. Основний пакет, який пропонує компанія, називається Splunk Enterprise. Якщо у компанії немає локальної інфраструктури, але вона повністю підтримується хмарними службами, слід вибрати Splunk Cloud.

Існує також спеціалізований пакет, який контролюватиме послуги AWS. Приклад панелі моніторингу інформаційної системи зображено на рис. 1.6.

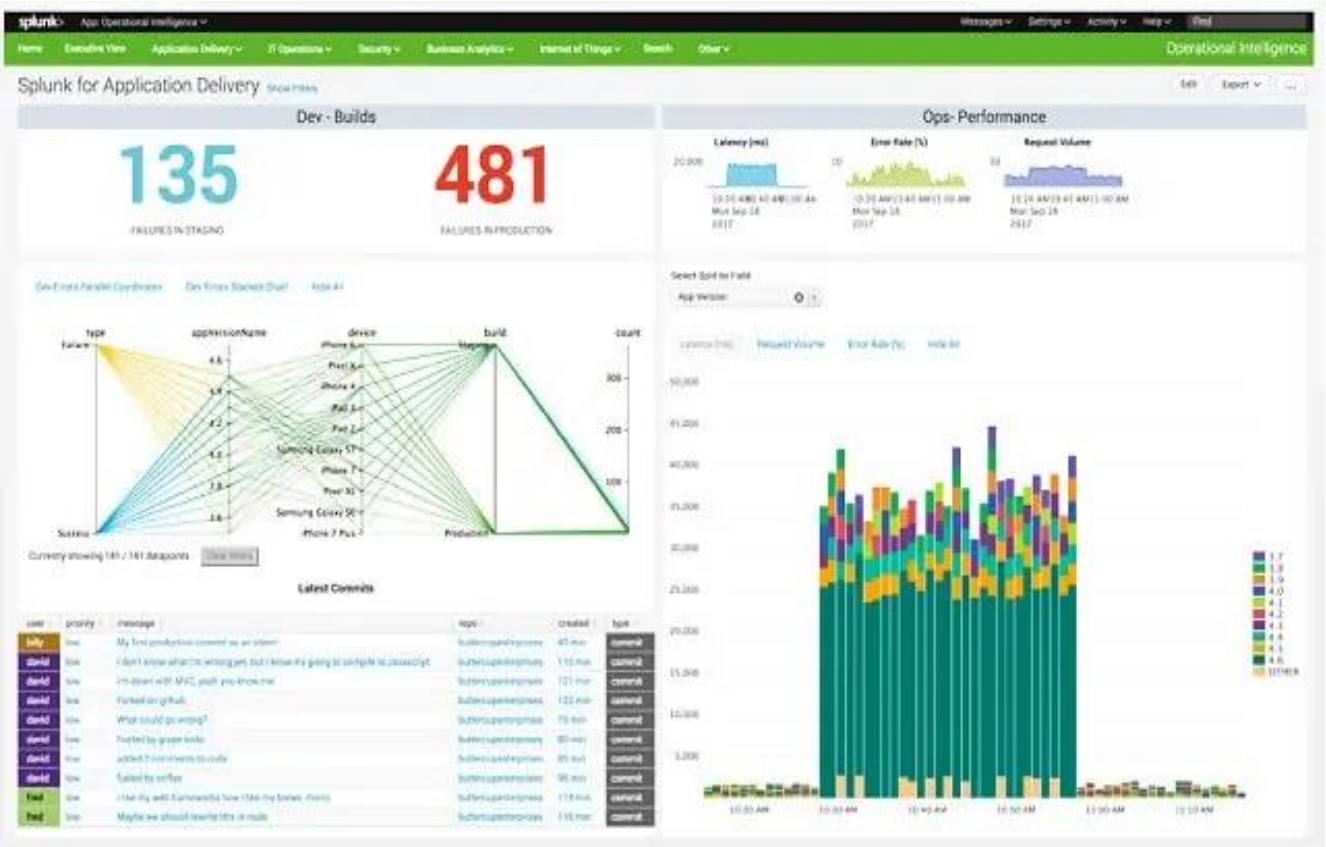


Рис. 1.6. Панель моніторингу Splunk

Splunk Enterprise доступний на 60-денній безкоштовній пробній версії, і ви можете отримати 15-денну безкоштовну пробну версію Splunk Cloud. Більш дешева версія Splunk під назвою Splunk Light має обмеження в 200 ГБ даних на день, і ви можете отримати її на 30-денній безкоштовній пробній версії. Існує також безкоштовна версія, яка має обмеження в 500 Мб даних на день. Однак ви не можете додати IT-сервісний аналіз до безкоштовного Splunk.

Splunk Enterprise, Splunk Light і Splunk Free реалізовані як локальне програмне забезпечення, а до Splunk Cloud доступ через Інтернет здійснюється через браузер. Програмне забезпечення Splunk можна встановити на Windows, Linux та Mac OS.

Ключові особливості:

- централізоване, уніфіковане уявлення про IT-послуги за допомогою динамічних інформаційних панелей;
- статистика для інфраструктури (моніторинг інфраструктури та усунення несправностей);

- моніторинг хмар AWS;
- безпека підприємства;
- автоматизація завдань та робочі процеси за допомогою Phantom;
- зберігання та передача даних через Universal Forwarder.

1.2.6. Вибір інструменту для моніторингу додатку

Розглянувши п’ять найпопулярніших на сьогоднішній день інструментів для аналізу технічного стану існуючого веб-застосунку можно прийти до висновку, що для потреб організації найкраще підходить NewRelic. Він відповідає всім критеріям попередньо визначеним у розділі 1.2. Також даний APM підтримує весь стек технологій існуючої інформаційної системи, має цілодобову технічну підтримку. Дуже важливо, що підтримка включає в себе інженерів, що можуть допомогти з інтеграцією інструменту у додаток. APM має декілька сертифікатів з кібербезпеки, працює з хмари, що дозволить нашій команді зберегти час на конфігурації та підтримці власного серверу та дозволить зосередитись на покращенні показників веб-застосунку.

1.3. Інтеграція інструменту NewRelic у веб-застосунок

Перед безпосередньою інтеграцією інструменту детальніше розглянемо як він працює. Спрощену схему роботи NewRelic можна побачити на рисунку 1.7. Користувачі використовують додатки компанії в яких присутній інструмент.

Треба зауважити, він є агностиком, тобто працює однаково незалежно від платформи. Це можуть бути веб-застосунки, мобільні або настільні додатки. Не залежить він і від операційної системи, Android, IOS, Ubuntu, Debian, Fedora, MacOS, Windows - усі вони підтримуються. Єдиною умовою є підтримка мови програмування, яка використовувалася при розробці програмного забезпечення. Це не є проблемою оскільки на даний момент інструмент підтримує більше двадцять поширених мов програмування, серед яких Java, C#, Javascript, Python, PHP, Ruby, Typescript, Golang, Swift, Kotlin та інші. На сьогоднішній день можна констатувати,

що NewRelic є найбільш універсальним інструментом для аналізу технічного стану програмного забезпечення.

Інструмент має таку гнучкість завдяки методу інтеграції, він вбудовується безпосередньою у програмний код інформаційної системи. Під час виконання програми він збирає та обраховує потрібні метрики, зберігає детальну інформацію про помилки у коді і місце їх виникнення.

Далі бібліотека-агент відправляє зібрані дані до сервісів NewRelic, що зберігають та обробляють отриману інформацію, потім порівнюють з минулими результатами. Це робиться для того, щоб на графічному інтерфейсі створити інтерактивні таблиці та панелі, приклад яких можна побачити на рисунку 1.8.

Надалі команда розробників працює з графічним інтерфейсом інструменту, аналізуючи потрібні дані. На їх основі можна робити висновки про потребу рефакторингу коду.

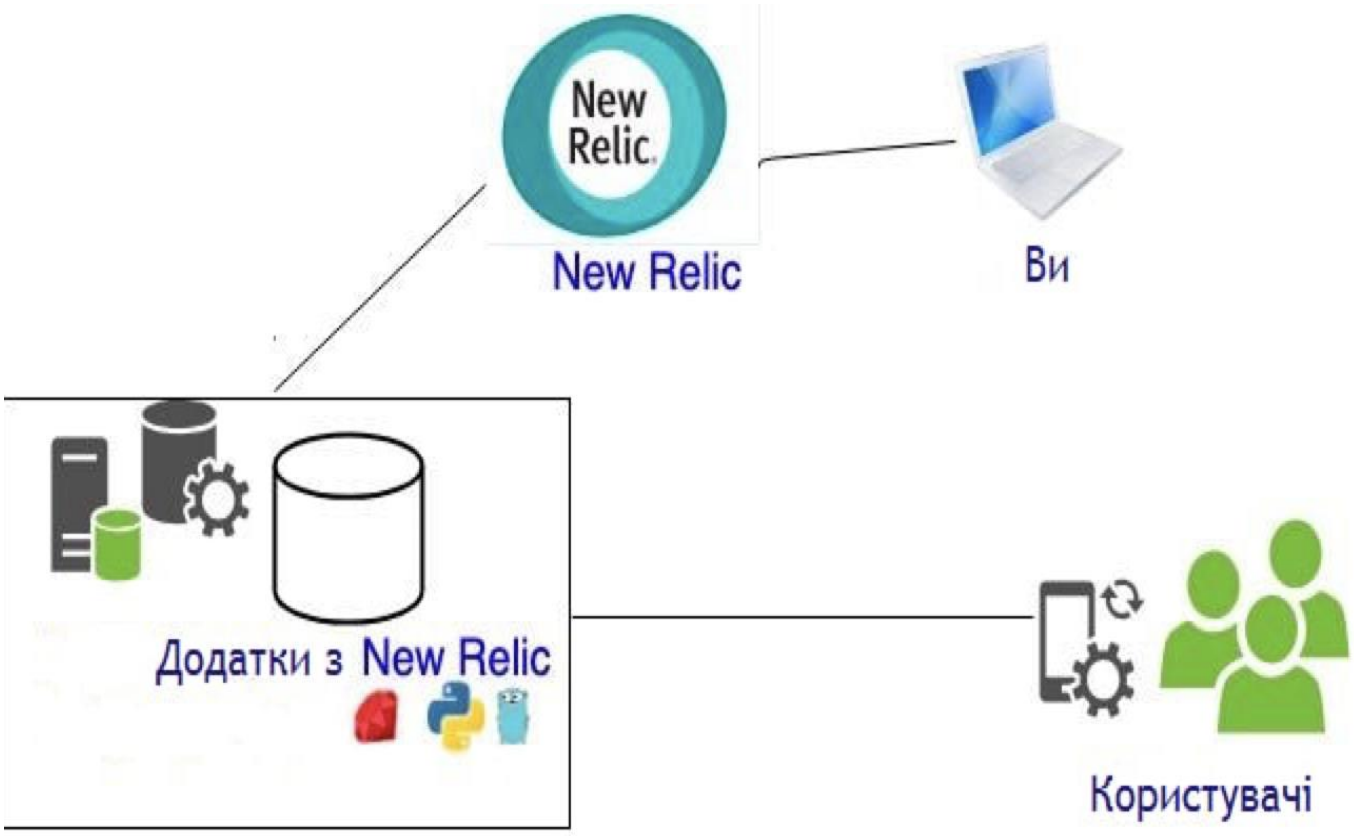


Рис. 1.7. Схема роботи NewRelic

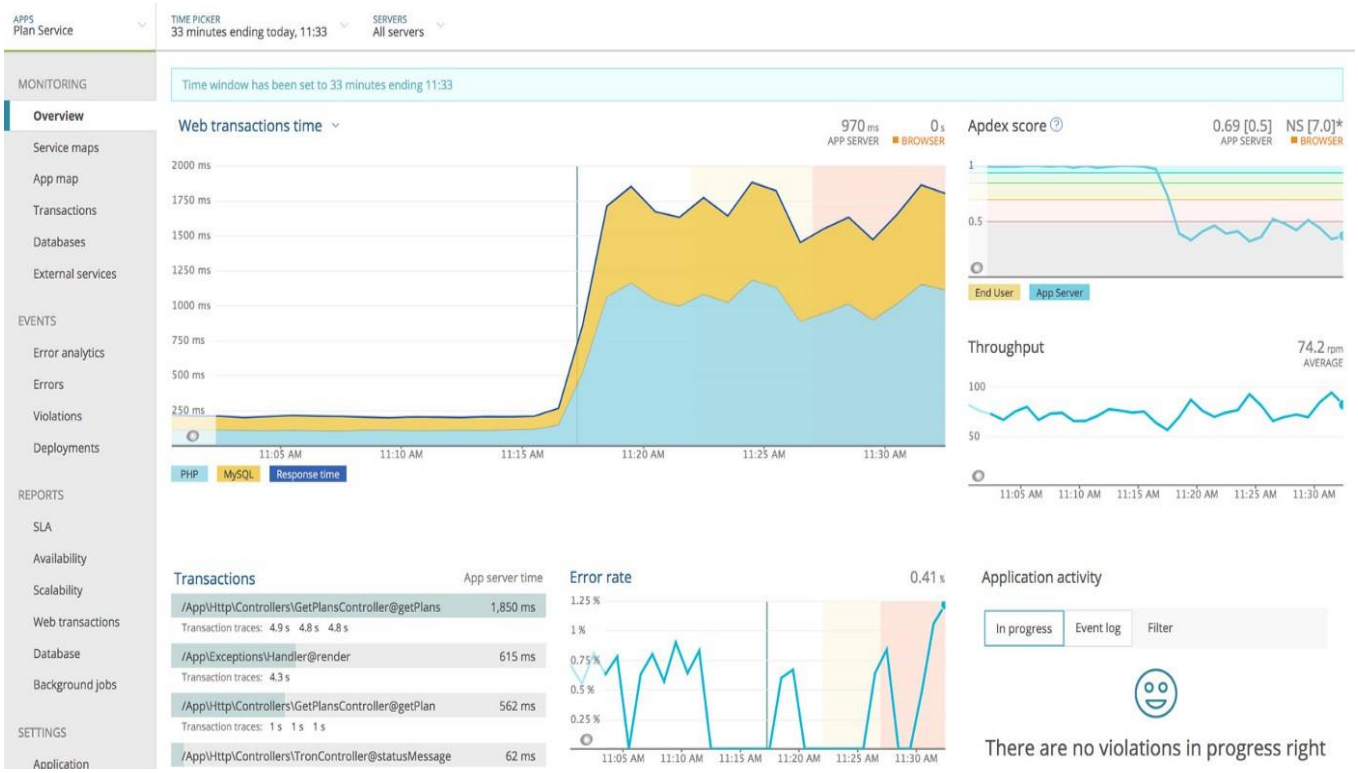


Рис. 1.8. Інтерактивна панель NewRelic

Інтегрувати APM NewRelic у існуючий додаток будемо згідно офіційної документації, що надає компанія NewRelic на своєму веб-сайті. Даний інструмент інтегрується за допомогою бібліотеки-агента, що буде відстежувати роботу інформаційної системи, записувати системні повідомлення та рахувати метрики. Веб-застосунок написаний на мові програмування Ruby, тому будемо використовувати бібліотеку та інструкцію саме для цієї мови. Для інтеграції потрібно:

- додати агент до Gemfile (файл, що зберігає усі залежності додатку) за допомогою команди gem 'newrelic_rpm';
- виконати команду bundle install у корнівій теці проекту у терміналі [4]. Результат виконання команди зображений на рис 1.9.

```
Successfully installed newrelic_rpm-6.13.1
1 gem installed
```

Рис. 1.9. Результат виконання команди bundle install

Надалі відбувається конфігурація бібліотеки та встановлення даних для коректної роботи. Для цього потрібно створити файл newrelic.yml (у папці config додатку), також відомий як файл конфігурації, і назвати програму. Щоб це зробити необхідно виконати наступне:

- Увійти в NewRelic.
- У меню облікового запису вибрати “Налаштування облікового запису”.
- На сторінці налаштувань облікового запису знайти розділ “Завантажити чистий файл конфігурації” та натиснути на файл newrelic.yml.
- Скопіювати файл newrelic.yml у підкаталог конфігурації програми.
- Змініть ім’я програми за замовчуванням на значуще ім’я.

Крім того, існує можливість створити файл newrelic.yml вручну за допомогою виконання такої команди у терміналі:

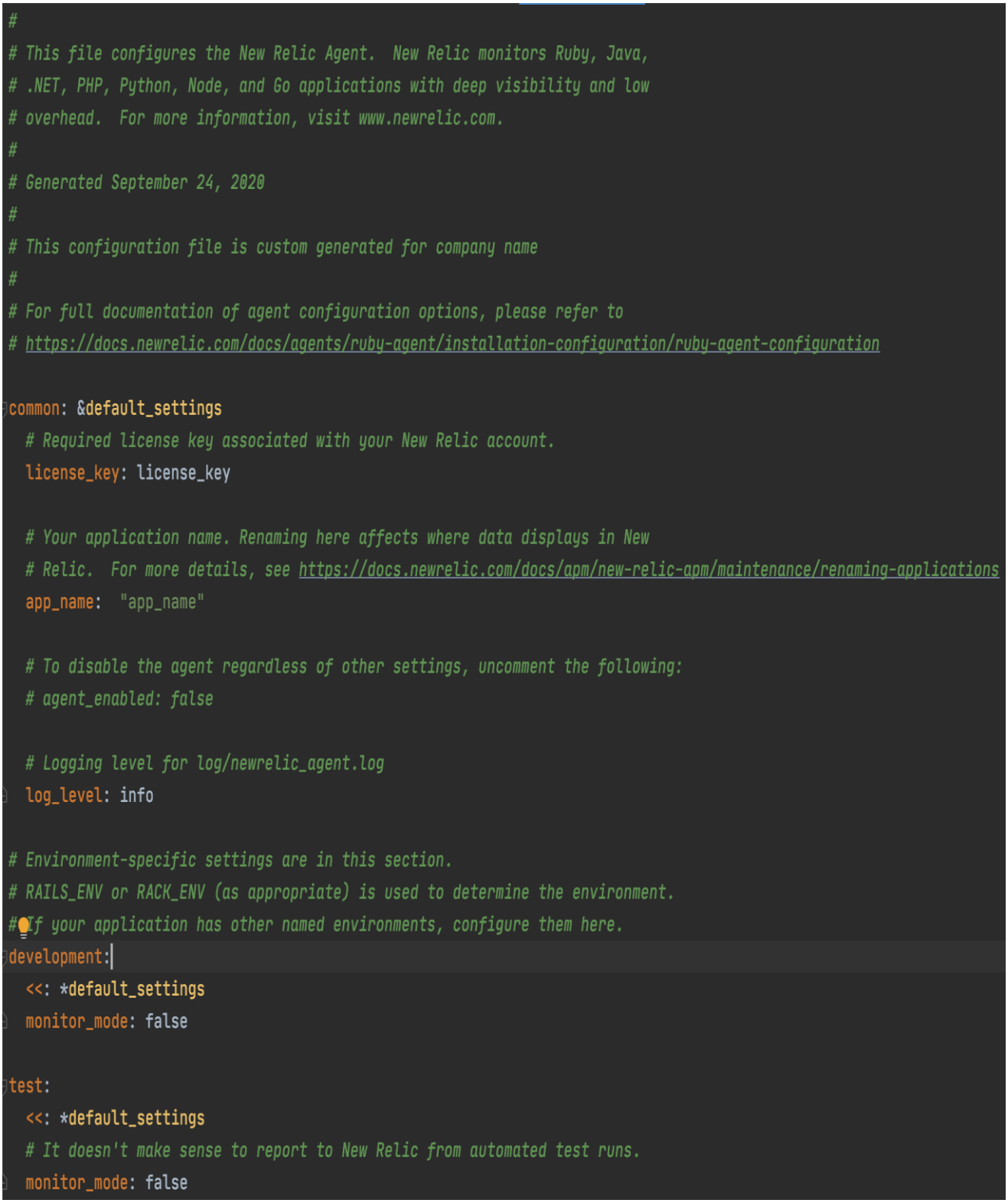
newrelic install --license_key="KEY" "APPLICATION_NAME"

В даній команді KEY - це ключ ліцензії, який користувач чи організація отримує після оплати тарифного плану, а APPLICATION_NAME - це назва додатку який буде аналізуватися, саме за цим ім'ям можливо ідентифікувати сервіс у веб-інтерфейсі NewRelic.

Агент Ruby newrelic.yml - це стандартний файл конфігурації YAML. Зазвичай він включає розділ за замовчуванням угорі, а також розділи нижче для кожного середовища програми; наприклад, розробка, випробування та виробництво [5].

Агент Ruby визначає, з якого розділу конфігураційного файлу newrelic.yml читати, переглядаючи певні змінні середовища для отримання середовища програми. Це корисно, наприклад, коли використовується інформація для налаштування конфігурації log_level у виробничому середовищі, і необхідно отримати детальні налаштування конфігурації log_level (наприклад, налагодження у середовищі розробки).

Створимо файл за допомогою виконання команди у терміналі та додаємо деякі додаткові параметри. Результат можна побачити на рисунках 1.10 та 1.11.



```
#
# This file configures the New Relic Agent.  New Relic monitors Ruby, Java,
# .NET, PHP, Python, Node, and Go applications with deep visibility and low
# overhead.  For more information, visit www.newrelic.com.
#
# Generated September 24, 2020
#
# This configuration file is custom generated for company name
#
# For full documentation of agent configuration options, please refer to
# https://docs.newrelic.com/docs/agents/ruby-agent/installation-configuration/ruby-agent-configuration

common: &default_settings
  # Required license key associated with your New Relic account.
  license_key: license_key

  # Your application name. Renaming here affects where data displays in New
  # Relic.  For more details, see https://docs.newrelic.com/docs/apm/new-relic-apm/maintenance/renaming-applications
  app_name:  "app_name"

  # To disable the agent regardless of other settings, uncomment the following:
  # agent_enabled: false

  # Logging level for log/newrelic_agent.log
  log_level: info

# Environment-specific settings are in this section.
# RAILS_ENV or RACK_ENV (as appropriate) is used to determine the environment.
# If your application has other named environments, configure them here.
development:
  <<: *default_settings
  monitor_mode: false

test:
  <<: *default_settings
  # It doesn't make sense to report to New Relic from automated test runs.
  monitor_mode: false
```

Рис. 1.10. Файл newrelic.yml

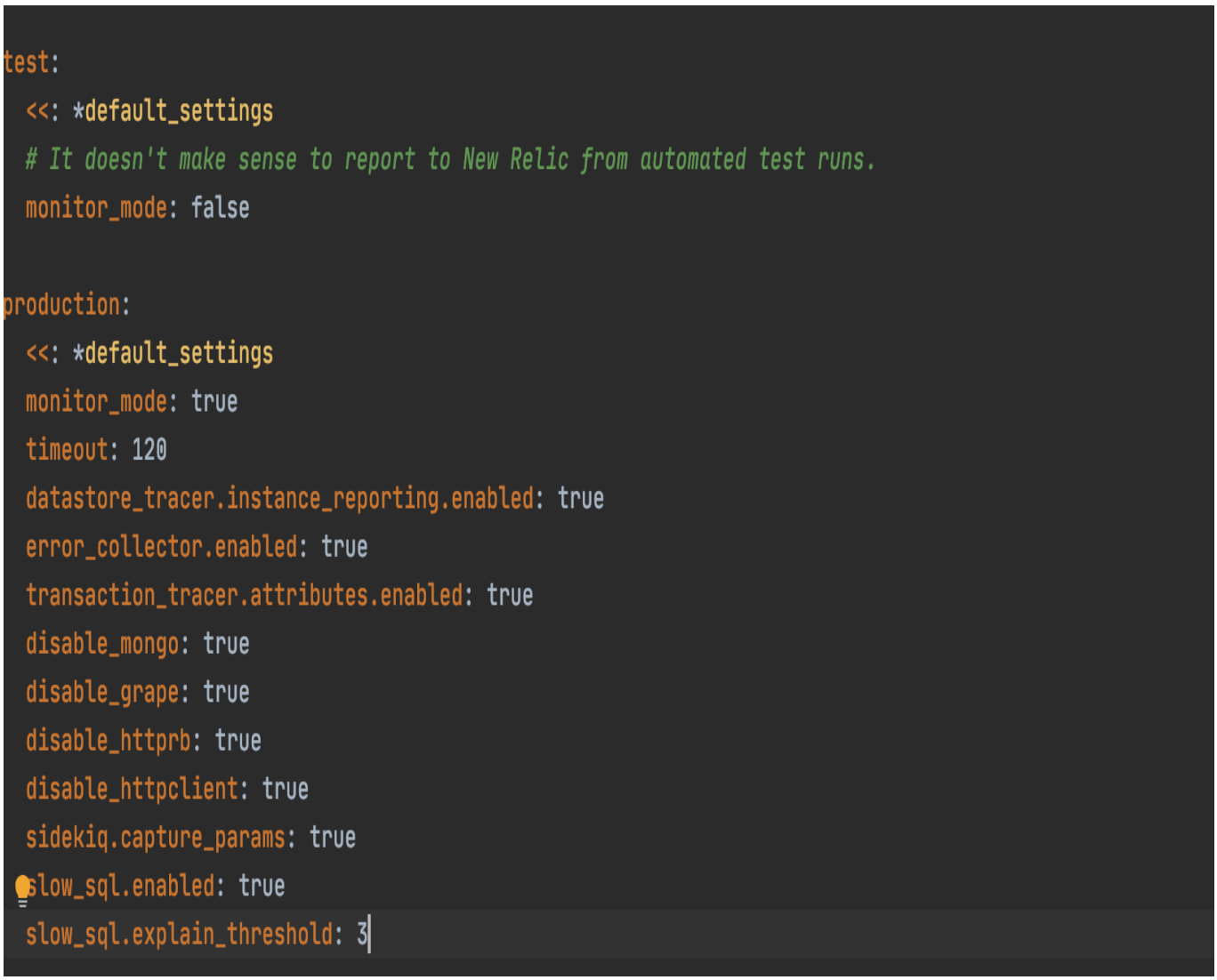


Рис. 1.11. Файл newrelic.yml

Пояснення до значень та функцій деяких параметрів файлу конфігурації:

- Параметр monitor_mode має значення false для стану development та test, це означає, що ми не будемо аналізувати роботу додатку при локальній розробці та виконанні тестів. Для production стану цей параметр має значення true.
- Параметр timeout визначає максимальну кількість секунд, яку агент повинен витратити, намагаючись підключитися до додатку. Значення цього параметру 120 секунд, тобто 2 хвилини. Якщо агент не зміг підключитися до інформаційної системи, то це означає що вона перестала відповідати, та вірогідно працювати. Іншими словами, додаток більше не доступний для користувачів.

- Параметр datastore_tracer.instance_reporting.enabled має значення true, якщо значення false, агент не повідомлятиме показники экземпляра сховища даних.
- Параметр error_collector.enabled має значення true. Якщо значення true, агент фіксує простежені помилки та показники підрахунку помилок.
- Параметр transaction_tracer.attributes.enabled має значення true. Якщо значення true, агент фіксує атрибути зі слідів транзакцій.
- Параметр disable_mongo має значення true. Якщо значення true, агент не встановлюватиме додаткові інструменти для бібліотек що працюють з не реляційною базою даних Mongo.
- Параметр disable_grape має значення true. Якщо значення true, агент не встановлюватиме додаткові інструменти для бібліотеки grape.
- Параметр sidekiq.capture_params має значення true. Якщо значення true, агент дозволяє збирати аргументи завдання для слідів транзакцій та простежених помилок у бібліотеці Sidekiq.
- Параметр slow_sql.enabled має значення true. Якщо значення true, агент збирає повільні запити SQL.
- Параметр slow_sql.explain_threshold має значення 3. Це вказаний поріг швидкості виконання запитів у секундах. Агент збирає повільні запити SQL, які перевищують цей поріг та пояснює методи по їх пришвидшенню.

ВИСНОВОК ДО РОЗДІЛУ 1

Ринок надання кредитних та іпотечних пропозицій є високо конкурентним, що змушує компанії, які на ньому працюють постійно розвиватися та шукати нові можливості залучення клієнтів. Світ невпинно діджиталізується і користувачі все більше віддають перевагу послугам, що надаються онлайн, це швидко, зручно, без черг, великої кількості паперів та потреби кудись їхати. Ці два фактори напряму вплинули на щорічне зростання долі кредитів оформлених віддалено. Це зумовило швидкий ріст трафіку на інформаційних системах компаній, що працюють у сфері фінансів. На сьогоднішній день для того, щоб продовжувати свою діяльність такі компанії вимушені вкладати великі кошти у розвиток своїх систем та застосунків, постійно покращувати продуктивність додаків, щоб вони успішно обробляли невпинно зростаючу кількість користувачів, а як результат і запитів на веб-сервіси компанії. Для цього потрібно моніторити роботу існуючої інформаційної системи та виявляти слабкі місця, де програмний код або запити виконуються повільно. Надалі виправляти знайдені фрагмети і знову моніторити, це має бути циклічний процес. Якщо компанія хоче вижити у сучасному високотехнологічному світу, такі процеси мають бути звичкою, такою ж сталою як сон для людини.

Процес аналізу технічного стану застосунків вручну дуже важкий і вимагає багато часу, тому вірним помічником у ньому має стати спеціальний інструмент для моніторингу. До його вибору потрібно підходити максимально відповідально, адже неправильно вибраний APM може не спростити задачу аналізу, а лише змарнувати час і кошти.

Саме тому провівши аналіз декількох популярних на сьогоднішній день інструментів для аналізу технічного стану інформаційних систем було зроблено висновок, що для потреб нашої організації найкраще підходить NewRelic. Він підтримує весь стек технологій існуючого веб-додатку, має цілодобову технічну підтримку.

Цей інструмент моніторингу має декілька сертифікатів з кібербезпеки, а сама компанія NewRelic раз у півроку проходить незалежний аудит безпеки, його

висновки компанія публікує на своєму сайті. Це дуже важливо, оскільки використовуючи інструмент для аналізу продуктивності додатку, ви довіряєте сторонній компанії програмний код вашої системи і дані ваших клієнтів.

Даний АРМ працює з хмари, що дозволить команді зберегти час на конфігурації та підтримці власного серверу та дозволить зосередитись на покращенні показників веб-застосунку.

Також було встановлено бібліотеку-агент у існуючий веб-додаток та виконану його конфігурацію. Це дозволить збирати метрики інформаційної системи та проводити заходи щодо покращення її продуктивності.

РОЗДІЛ 2

МОНІТОРИНГ ПРОДУКТИВНОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ЇЇ РЕФАКТОРИНГ

2.1. Поняття рефакторингу коду

Постійне покращення та пришвидшення фрагментів коду інформаційної системи прийнято виконувати шляхом так званого рефакторингу.

У комп'ютерному програмуванні та розробці програмного забезпечення рефакторинг - це процес реструктуризації існуючого комп'ютерного коду - зміни факторингу, без зміни його зовнішнього поведінки. Рефакторинг призначений для поліпшення дизайну, структури або реалізації програмного забезпечення (його не функціональних атрибутів) при збереженні його функціональності. Потенційні переваги рефакторінга можуть включати поліпшену читаність коду і зниження складності; вони можуть поліпшити ремонтопридатність вихідного коду і створити більш просту, зрозумілу або більш виразну внутрішню архітектуру або об'єктну модель для поліпшення розширюваності. Інша потенційна мета рефакторінга - підвищення продуктивності. Розробники програмного забезпечення постійно стикаються з проблемою створення програм, які працюють швидше або використовують менше пам'яті.

Зазвичай при рефакторингу застосовується серія стандартизованих базових мікро-рефакторингів, кожен з яких зазвичай являє собою крихітне зміна вихідного коду комп'ютерної програми, яка або зберігає поведінку програмного забезпечення, або, принаймні, не змінює його відповідність функціональним вимогам.

Багато середовищ розробки надають автоматизовану підтримку для виконання технічних аспектів цих базових рефакторингів. Якщо все зроблено правильно, рефакторинг коду може допомогти розробникам програмного забезпечення виявляти і виправляти приховані або неактивні помилки або уразливості в системі за рахунок спрощення базової логіки і усунення непотрібних рівнів складності. Якщо все зроблено неправильно, це може призвести до невиконання вимоги про те, що зовнішні функції не повинні бути змінені, появи нових помилок або й того, й іншого.

Хоча рефакторинг коду неофіційно проводиться протягом десятиліть, дисертація доктора філософії Вільяма Грісволда 1991 року є однією з перших великих наукових робіт по рефакторингу функціональних і процедурних програм, за нею йде дисертація Вільяма Опдайка 1992 року про рефакторинг об'єктно-орієнтованих програм [8]. Ці дисертації є першими науковими роботами в яких описаний процесу рефакторингу. Вони надають каталог загальних методів, є опис того, як застосовувати метод, і індикатори того, коли він повинен застосовуватися.

Рефакторинг зазвичай мотивується відчуттям складності коду. Наприклад, розглянутий метод може бути дуже довгим або майже дублювати інший найближчий метод. Після виявлення такі проблеми можна вирішити шляхом рефакторінга вихідного коду або перетворення його в нову форму, яка поводиться так само, як і раніше, але виглядає простіше.Для довгої підпрограми можна виділити одну або кілька дрібніших підпрограм; або для повторюваних процедур дублювання може бути видалено і замінено однією спільною функцією. Невиконання рефакторінга може призвести до накопичення технічної заборгованості; з іншого боку, рефакторинг - один з основних засобів погашення технічного боргу.

Є три основні переваги рефакторингу:

- Ремонтопридатність. Помилки виправляти легше, тому що вихідний код легко читати, а наміри його автора легко зрозуміти. Цього можна досягти, скоротивши великі монолітні підпрограми до набору індивідуально коротких, добре названих одноцільових методів. Цього можна домогтися, перемістивши метод в більш відповідний клас або додавши коментарі.

- Можливість розширення. Легше розширити можливості програми, якщо воно використовує впізнавані шаблони проектування і забезпечує деяку гнучкість там, де їх раніше не було.
- Пришвидшення фрагментів коду. Під час рефакторингу відбувається переосмислення коду, що призводить до його спрощення та використання більш ефективних методів.

Інжиніринг продуктивності може усунути неефективність програм, відому як роздування програмного забезпечення, що виникає через традиційні стратегії розробки програмного забезпечення, які спрямовані на мінімізацію часу розробки програми, а не часу, необхідного для виконання коду. Проектування продуктивності також може адаптувати програмне забезпечення до обладнання, на якому воно працює, наприклад, для використання переваг паралельних процесорів і векторних модулів.

Перед рефакторингом слід налаштувати автоматичні модульні тести, щоб гарантувати, що процедури, як і раніше працюють належним чином. Модульні тести можуть забезпечити стабільність навіть великим рефакторингам, якщо вони покривають увесь функціонал наявної інформаційної системи [9].

Під час модульного тестування, рефакторинг є ітеративним процесом, в якому виконується невелике перетворення програми, перевіряється її правильність і виконується ще одне невелике перетворення. Якщо в якийсь момент тест не проходить, остання невелика зміна скасовується і повторюється іншим способом. Пройшовши безліч маленьких кроків, програма значно змінюється, що призводить до досягнення поставлених цілей. Щоб цей ітеративний процес був практичним, тести повинні виконуватися дуже швидко, інакше програмісту доведеться витрачати більшу частину свого часу на очікування завершення тестів. Прихильники екстремального програмування та іншої гнучкої розробки програмного забезпечення описують цю діяльність як невід'ємну частину циклу розробки програмного забезпечення.

2.2. Аналіз продуктивності коду інформаційної системи за допомогою інструменту NewRelic

Інструмент для моніторингу технічного стану додатку включає в себе аналітику багатьох метрик, що впливають на його продуктивність. Це можуть бути аналіз швидкості виконання SQL запитів, асинхронних робіт, запитів до сторонніх ресурсів та сервісів, підпрограм, швидкості відповіді сервісу і так далі.

Розпочнемо з секції огляду швидкості виконання запитів до бази даних. На рисунку 2.1 можна побачити SQL запити, відсортовані за часом виконання. Відкривши конкретний запит можна знайти детальну інформацію про середній час виконання, кількість запитів за проміжок часу, класс в коді з якого від був зроблений. Однією з унікальних функцій NewRelic є аналіз SQL запитів та надання рекомендацій по оптимізації.

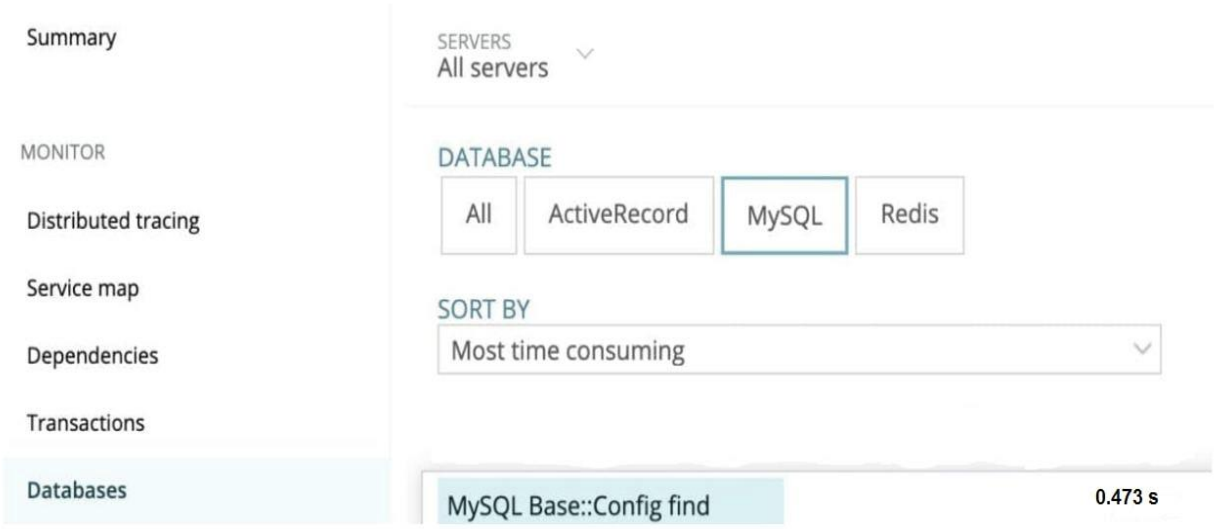


Рис. 2.1. Секція баз даних

Розглянемо безпосередній приклад, найповільніший запит - MySQL Base::Config find. Відкривши результати аналізу даного запиту, можемо побачити інформацію про таблиці, що приймали участь у ньому, короткі результати огляду та рекомендації по оптимізації. Цю інформацію можна побачити на рисунку 2.2.

Table	Hint
loan_lender_vertical_configs	—This table was retrieved with a full table scan, which is often quite bad for performance, unless you only retrieve a few rows. —The table was retrieved with this index: —No index was used in this part of the query. —You can speed up this query by querying only fields that are within the index. Or you can create an index that includes every field in your query, including the primary key. —Approximately 66 rows of this table were scanned.
loan_lenders	—The table was retrieved with this index: `PRIMARY` —You can speed up this query by querying only fields that are within the index. Or you can create an index that includes every field in your query, including the primary key. —Approximately 1 row of this table was scanned.

Рис. 2.2. Результати аналізу запиту

Наступною розглянемо секцію огляду асинхронних робіт. На рисунку 2.3 зображена дана секція з списком робіт відсортованих за сумарним часом виконання. Для порівняння під кожною работою, наведено середній час виконання за останні двадцять чотири години, попередні двадцять чотири години та тиждень.

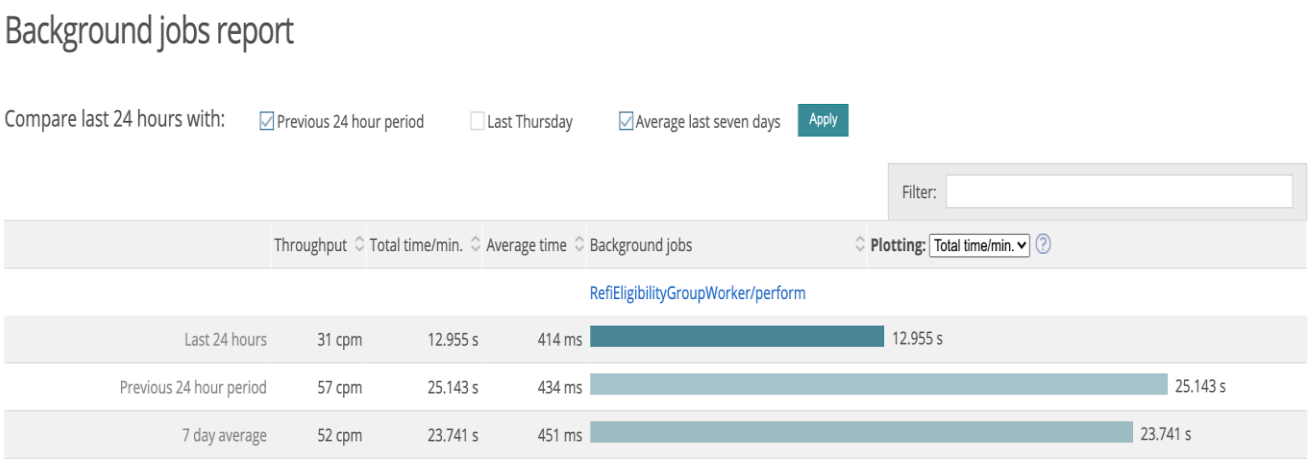


Рис. 2.3. Секція аналізу асинхронних робіт

Така аналітика дає нам змогу побачити залежність швидкості виконання програмного коду з навантаженням на інформаційну систему. Також можна відслідковувати як ті чи інші зміни у коді впливають на продуктивність системи.

2.3. Рефакторинг фрагментів коду, що потребують пришвидшення

Згідно інформації NewRelic по найповільнішого SQL запиту, зображеній на рисунку 2.2, таблиця що використовувалося при виконанні запиту не була проіндексована. Рекомендацією інструменту є індексація даної таблиці.

Індекс - об'єкт бази даних, який створюється з метою підвищення продуктивності пошуку даних. Таблиці в базі даних можуть мати велику кількість рядків, які зберігаються в довільному порядку, і їх пошук по заданому критерію шляхом послідовного перегляду таблиці рядок за рядком може займати багато часу. Індекс формується із значень одного або декількох стовпців таблиці і покажчиків на відповідні рядки таблиці і, таким чином, дозволяє шукати рядки, що задовольняють критерію пошуку. Прискорення роботи з використанням індексів досягається в першу чергу за рахунок того, що індекс має структуру, оптимізовану під пошук - наприклад, збалансованого дерева.

Існує два типи індексів: кластерні і некластерні. При наявності кластерного індексу рядка таблиці впорядковані за значенням ключа цього індексу. Якщо в таблиці немає кластерного індексу, таблиця називається купою. Некластерний індекс, створений для такої таблиці, містить тільки покажчики на записи таблиці. Кластерний індекс може бути тільки одним для кожної таблиці, але кожна таблиця може мати декілька різних некластерних індексів, кожен з яких визначає свій власний порядок проходження записів.

Пошук у базі даних з індексами та без них, схожий на пошук слова у книзі. Без індексів - це як шукати слово в усій книзі перебираючи, слово за словом. З індексами - це як шукати слово у змісті з вказівкою на шлях до слова.

Для пришвидшення найповільнішого SQL запиту використаємо рекомендацію NewRelic та додамо некластерну індексацію, за допомогою міграції зображеної на рисунку 2.4.

```ruby
class AddIndexToLoanLenderVerticalConfig < ActiveRecord::Migration[5.2]
  def change
    add_index :loan_lender_vertical_configs, :loan_lender_id
    add_index :loan_lender_vertical_configs, :vertical
  end
end
```

Рис. 2.4. Міграція бази даних

Перевіривши секцію огляду швидкості виконання запитів до бази даних через деякий час після додавання індексації, можемо побачити, що час виконання запиту Base::Config find значно зменшився і він вже не є найповільнішим. Оновлена секція аналізу SQL-запитів зображена на рисунку 2.5.

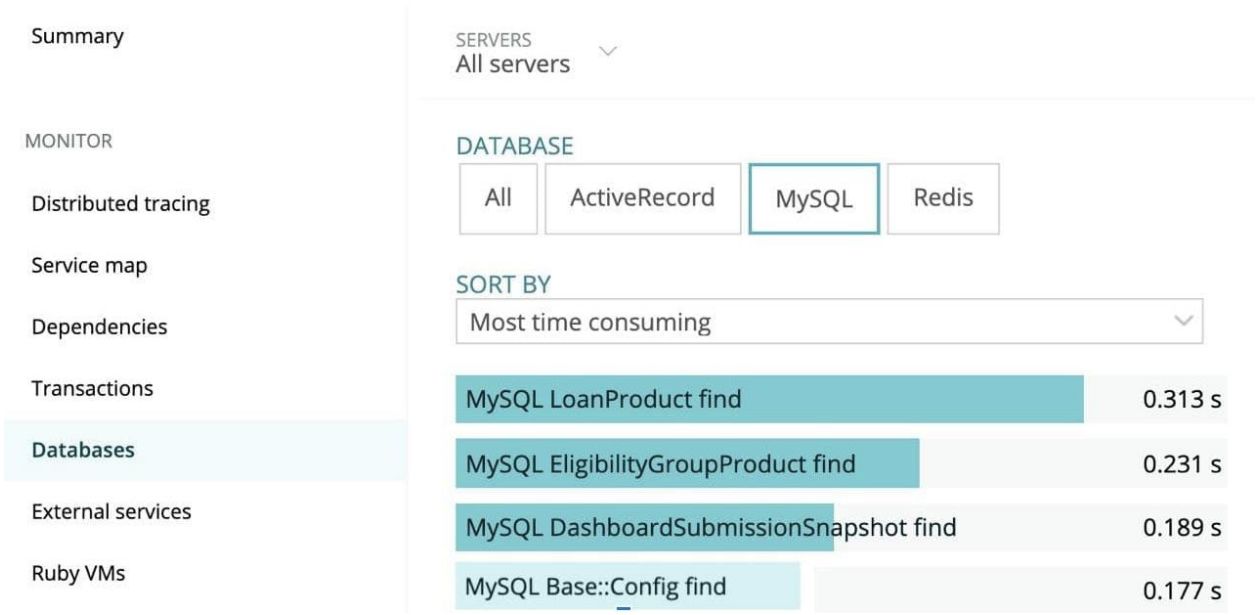


Рис. 2.5. Секція баз даних

Проаналізуємо та проведемо рефакторинг найповільнішої асинхронної робити RefiEligibilityGroupWorker. У ньому зустрічається поширені в програмній інженерії помилки, з отримання непотрібних даних з бази даних та перебору всіх рядків таблиці при потребі у одному результаті. Розглянемо фрагмент коду, що включає обидві вищезгадані помилки на рисунку 2.6.

Рис. 2.6. Фрагмент коду

Метод participant_phone повертає телефон користувача за його електронною адресою. Першою помилкою є перебір всіх рядків таблиці з декількома мільйонами записів задля отримання одного запису. Метод where повертає масив об'єктів, що задовольняють умову, в свою чергу метод first повертає перший елемент масиву. Другою помилкою є отримання непотрібних даних з бази, які не будуть використовуватися у подальшому. У змінну participant записується об'єкт з усіма його атрибутами, далі з цього об'єкта береться лише один атрибут - телефон. Щоб виправити ці помилки перепишемо метод, як вказано на рисунку 2.7

Рис. 2.7. Змінений фрагмент коду

Метод find_by буде шукати перший елемент, що задовольняє умову, це означає, що якщо знайдений об'єкт будет п’ятим пошук не буде проводитися по декількох мільйонах записах, а закінчиться після п'ятого запису. Також замість отримання усіх полів запису буде взяте лише поле phone.

Наступним знайденим фрагментом, що потребує покращення є так звана помилка n+1. Вона виникає коли йде ітерація по масиву об'єктів і при кожній ітерації робиться запит до бази даних для отримання інформації з пов'язаного

об'єкта. В такому випадку сумарна кількість запитів до бази даних буде складати кількість об'єктів у масиві плюс один запит. Для детальнішого пояснення розглянемо фрагменти коду зображені на рисунках 2.8 - 2.10.

```
3  class Participant < ApplicationRecord
4    has_many :loans
5
```

Рис. 2.8. Модель Participant

```
3  class Participant::Loan < ActiveRecord::Base
4    include ColumnEncryptor
5    include ClassyEnum::ActiveRecord
6
7    belongs_to :participant
```

Рис. 2.9. Модель Participant::Loan

```
59
60  def get_ref_types(lender_name)
61    loans = Participant::Loan.where(lender_name: lender_name)
62    ref_types = []
63    loans.each do |loan|
64      ref_types << loan.participant.reference_type
65    end
66    ref_types
67  end
68
```

Рис. 2.10. Метод get_ref_types

Модель Participant формує об'єкти користувачів, вона має залежність один до багатьох з моделлю Participant::Loan, тобто один користувач може мати багато кредитів, але конкретний кредит має лише одного користувача. Метод get_ref_types повертає масив типів посилань/рекомендацій, для заданого кредитора. Спочатку у змінну loans зберігається масив кредитів, що були видані даним кредитором - це один запит у базу даних. Далі всередині циклу на кожній ітерації для кожного

кредиту отримується тип посилання, через об'єкт participant і записується у змінну ref_types. Для цього потрібно стільки ж запитів до бази даних, скільки є об'єктів в масиві loans. Це і є помилка n+1. Якщо масив loans буде складатися з трьох об'єкти, сумарна кількість запитів буде дорівнювати чотирьом. Але якщо розмір масиву збільшиться до семи тисяч, то кількість запитів відповідно зросте до семи тисяч і одного. Неповний список запитів до БД, потрібних для виконання даного методу зображений на рисунку 2.11.

```
2020-12-09T12:21:09.706+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant::Loan Load (255.6ms)  SELECT  `participant_loans`.* FROM `participant_loans`
WHERE `participant_loans`.`lender_name` = 'GLELSI/ANTIGO CO-OP CU' LIMIT 20
2020-12-09T12:21:11.021+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant Load (253.7ms)  SELECT  `participants`.* FROM `participants`
 WHERE `participants`.`id` = 20532 LIMIT 1
2020-12-09T12:21:11.307+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant Load (252.0ms)  SELECT  `participants`.* FROM `participants`
 WHERE `participants`.`id` = 20565 LIMIT 1
2020-12-09T12:21:11.561+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant Load (252.8ms)  SELECT  `participants`.* FROM `participants`
 WHERE `participants`.`id` = 20566 LIMIT 1
2020-12-09T12:21:11.816+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant Load (253.3ms)  SELECT  `participants`.* FROM `participants`
 WHERE `participants`.`id` = 20636 LIMIT 1
2020-12-09T12:21:12.068+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant Load (250.9ms)  SELECT  `participants`.* FROM `participants`
WHERE `participants`.`id` = 20643 LIMIT 1
2020-12-09T12:21:12.320+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant Load (251.4ms)  SELECT  `participants`.* FROM `participants`
WHERE `participants`.`id` = 20925 LIMIT 1
2020-12-09T12:21:12.576+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant Load (254.9ms)  SELECT  `participants`.* FROM `participants`
 WHERE `participants`.`id` = 20949 LIMIT 1
2020-12-09T12:21:12.828+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant Load (251.3ms)  SELECT  `participants`.* FROM `participants`
 WHERE `participants`.`id` = 20980 LIMIT 1
2020-12-09T12:21:13.083+0200 [70235021582380][0.1.0][][][ActiveRecord::Base:] DEBUG :   Participant Load (254.4ms)  SELECT  `participants`.* FROM `participants`
 WHERE `participants`.`id` = 20986 LIMIT 1
```

Рис. 2.11. Список запитів до бази даних

Для підвищення ефективності цього прикладу нам потрібно зменшити кількість незалежних запитів до бази даних. У Ruby on Rails це робиться шляхом попереднього завантаження пов’язаних відносин або, іншими словами, збору пов’язаних даних лише за допомогою одного запиту до бази даних. Це реалізується за допомогою методу includes, що приймає відношення, яке потрібно завантажити. Оновлений метод get_ref_types можна побачити на рисунку 2.12.

```ruby
def get_ref_types(lender_name)
  loans = Participant::Loan.where(lender_name: lender_name)
  ref_types = []
  loans.includes(:participant).each do |loan|
    ref_types << loan.participant.reference_type
  end
  ref_types
end
```

Рис. 2.12. Оновлений метод get_ref_types

При такій реалізації кількість запитів до бази даних завжди буде сталою незалежно від розміру масиву loans і буде дорівнювати двом. Тобто якщо масив loans буде складатися з трьох об'єкти, сумарна кількість запитів буде дорівнювати двом. І якщо розмір масиву збільшиться до семи тисяч, то кількість запитів залишиться рівною двом. Список запитів до бази даних, потрібних для виконання даного методу можна побачити на рисунку 2.13.

```
2020-12-09T12:21:19.646+0200 [7023502158238][0.1.0][][][ActiveRecord::Base:] DEBUG :  Participant::Loan Load (255.4ms)  SELECT `participant_loans`.*
FROM `participant_loans` WHERE `participant_loans`.`lender_name` = 'GLELSI/ANTIGO CO-OP CU' LIMIT 20
2020-12-09T12:21:19.910+0200 [7023502158238][0.1.0][][][ActiveRecord::Base:] DEBUG :  Participant Load (251.9ms)  SELECT `participants`.*
FROM `participants` WHERE `participants`.`id`
IN (20532, 20565, 20566, 20636, 20643, 20925, 20949, 20980, 20986, 21030, 21047, 21049, 21050, 21052, 21055, 21082, 21083, 21170, 21173, 21256)
```

Рис. 2.13. Список запитів до бази даних

Першим кроком до усунення запитів N + 1 є переконання, що ваша інженерна команда розуміє проблему та знає про нещодавно введені запити, які можна оптимізувати.

Перегляд запитів на об'єднання коду з головною гілкою та обмін знаннями - це хороший спосіб розпочати, але навіть при великій вазі до деталей такі помилки все одно можуть потрапити у програмний код продакшн версії. Для боротьби з цим, ми завжди прагнемо автоматизувати процеси аналізу і налаштовуємо інструменти для повідомлення про потенційні проблеми.

Бібліотека Bullet - чудовий інструмент для цієї мети. Він відстежує та повідомляє про неефективні запити. Щоб налаштувати цю бібліотеку на проекті, що використовує Ruby on Rails, спочатку потрібно додати його у Gemfile за допомогою команди: gem "bullet". Далі потрібно додати конфігурацію до файлу app/environments/test.rb, як показано на рисунку 2.14.

```
170
171     config.after_initialize do
172       Bullet.enable = true
173       Bullet.bullet_logger = true
174       Bullet.raise = true # raise an error if an n+1 query occurs
175     end
176
```

Рис. 2.14. Конфігурацію бібліотеки Bullet

Наступним кроком буде додавання конфігурації бібліотеки Bullet до бібліотеки rspec, яка відповідає за юніт-тестування додатку. Для цього потрібно додати інформацію до файлу spec/spec_helper.rb, як показано на рисунку 2.15.

```
18
19  RSpec.configure do |config|
20    if Bullet.enable?
21      config.before( scope :each) { Bullet.start_request }
22      config.after( scope :each)  { Bullet.end_request }
23    end
24  end
25
```

Рис. 2.15. Конфігурацію бібліотеки Bullet у файлі spec_helper.rb

Наступною знайденою помилкою є пошук по заздалегідь відомому двовимірний масиву, де перший елемент кожного масиву - це значення, за яким робиться пошук, а другий елемент масиву - це значення, яке шукають. Наприклад, на рисунку 2.16, першим елементом масиву mba_schools є універсальний національний код навчального закладу, а другим елементом масиву є його код минулої версії.

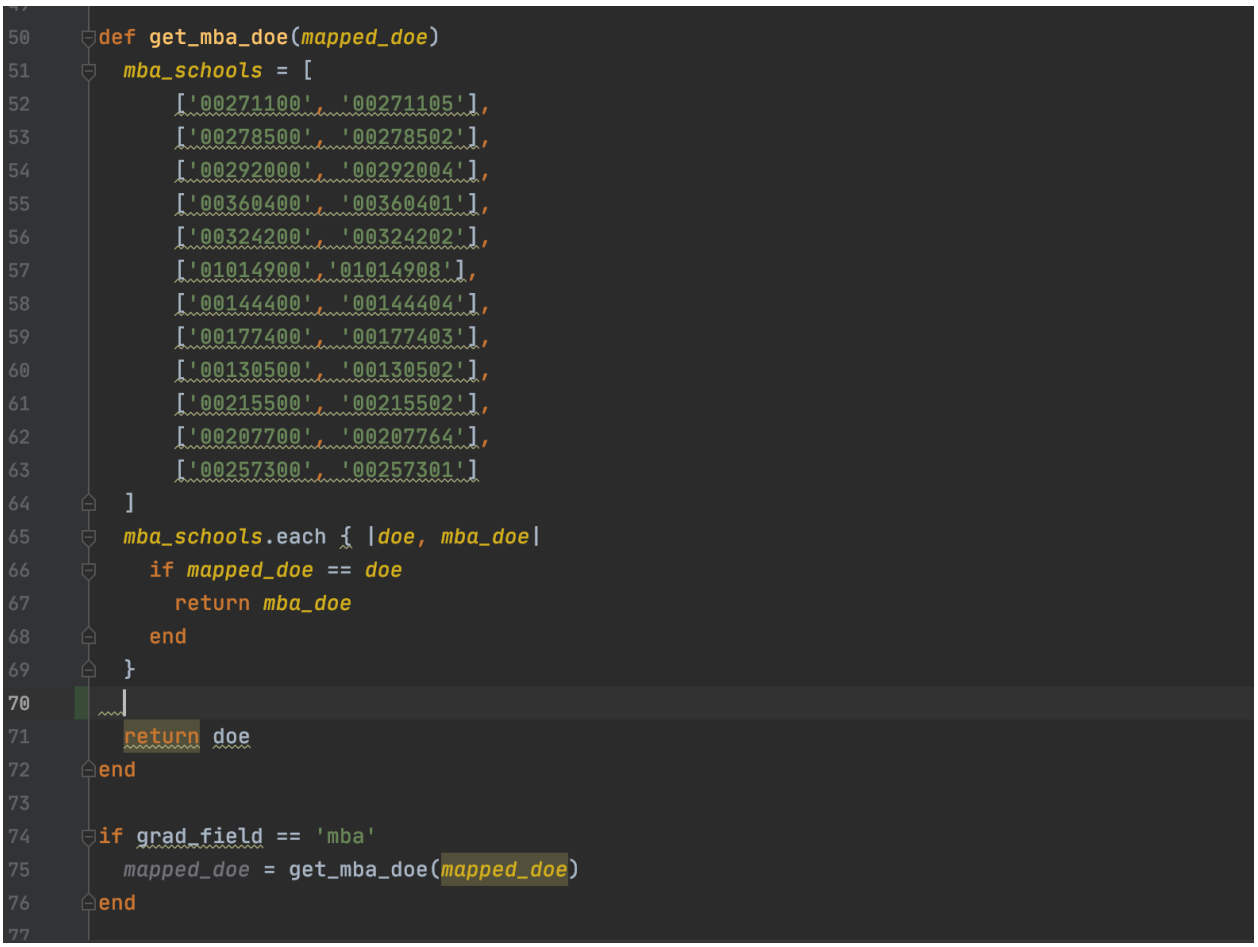


Рис. 2.16. Двовимірний масив шкіл

Всередині методу get_mba_doe проводиться пошук по цьому масиву за допомогою циклу для переведення коду з нової версію на минулу. Для такого випадку краще використовувати хеш замість двовимірного масиву. Перший елемент масиву буде ключем хешу, а другий його значенням. Пошук по хешу значно швидший ніж по масиву, оскільки він не включає циклів та повного перебору.

Ключ хешу є унікальним значенням і за метод функціонування схожий на індексацію у базі даних. Порівняння процесів пошуку у хеші та двовимірному масиві, схоже на пошук слова у книзі. Використовуючи двовимірний масив - це як шукати слово в усій книзі перебираючи, слово за словом. Використовуючи хеш - це як шукати слово у змісті з вказівкою на шлях до слова.

На рисунку 2.17 можна побачити оновлений метод get_mba_doe, що працює з хешом та без циклів. На даний момент різниця в швидкості виконання, збільшилася

лише трохи. Але якщо розмір хешу буде значно більшим, то різниця в часі виконання обох методів буде величезною.

```
50      def get_mba_doe(mapped_doe)
51        mba_doe_map = {
52              "00271100" => "00271105",
53              "00278500" => "00278502",
54              "00292000" => "00292004",
55              "00360400" => "00360401",
56              "00324200" => "00324202",
57              "01014900" => "01014908",
58              "00144400" => "00144404",
59              "00177400" => "00177403",
60              "00130500" => "00130502",
61              "00215500" => "00215502",
62              "00207700" => "00207764",
63              "00257300" => "00257301"
64        }
65
66        mba_doe_map[mapped_doe]
67      end
68
69      if grad_field == 'mba'
70        mapped_doe = get_mba_doe(mapped_doe)
71      end
```

Рис. 2.17. Оновлений метод get_mba_doe

Через деякий час після виправлення усіх вищезгаданих видів помилок та фрагментів коду, пришвидшення запитів до бази даних та проведення рефакторингу асинхронної робити RefiEligibilityGroupWorker, були отримані оновлені дані з інструменту для моніторингу інформаційної системи NewRelic. Їх можна побачити на рисунку 2.18. Швидкість виконання роботи збільшилася майже в два рази. Треба зауважити, що у період отримання оновлених даних трафік та навантаження було значно меншим, що також вплинуло на швидкість виконання роботи. Під час першого аналізу було від тридцяти двох по п'ятдесяти двох викликів класу за хвилину. Під час повторного аналізу після рефакторингу було від дванадцяти до двадцяти двох викликів класу за хвилину.

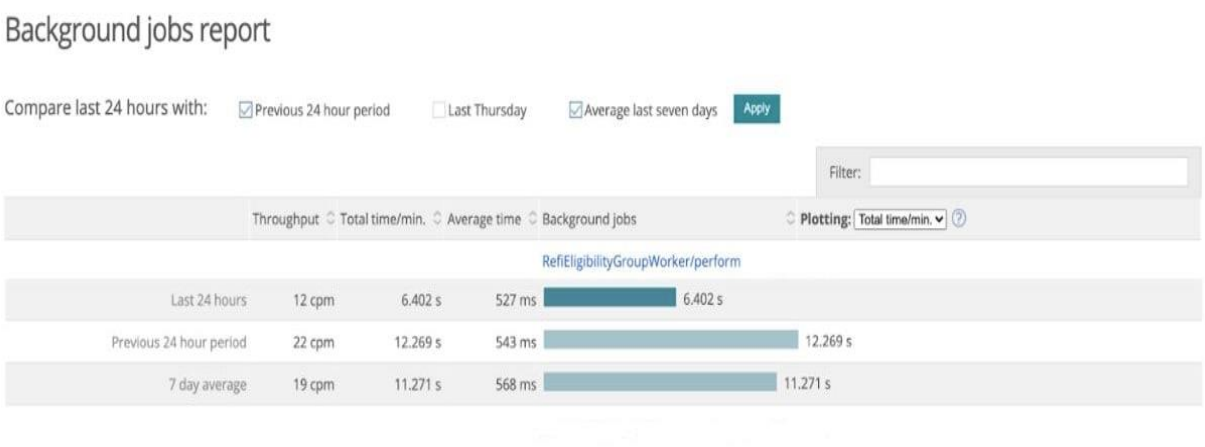


Рис. 2.18. Секція аналізу асинхронних робіт

ВИСНОВОК ДО РОЗДІЛУ 2

У сучасній економіці мобільний додаток або веб-сайт - це не просто частина бізнесу, а невід'ємна його частина. Іноді, це навіть джерело основного доходу компанії. Ось чому проблеми з продуктивністю інформаційних систем можуть серйозно перешкоджати росту і прибутковості бізнесу. Сьогодні люди хочуть бездоганного цифрового досвіду, і будь-які проблеми, з якими вони стикаються, мають вирішуватися в режимі реального часу.

Розвиток бізнесу та невпинне збільшення кількості користувачів призводить до потреби у постійному збільшенні пропускної здатності інформаційної системи. Досягти цього можна шляхом покращення та збільшення інфраструктури додатку, мається на увазі кількість серверів, ефективність їх роботи та потужність. Також цього можна досягти пришвидшенням виконання програмного коду застосунку. Для цього потрібно моніторити додатки та виявляти фрагменти програм, що виконуються повільно. Допомогти в цьому можуть спеціальні інструменти, що аналізують швидкість виконання коду та обраховують різні метрики пов'язані з продуктивністю інформаційної системи.

Використовуючи рішення для моніторингу продуктивності додатків (APM), компанії можуть контролювати, чи відповідає їх інформаційна система стандартам продуктивності, виявляти помилки та потенційні проблеми та забезпечувати бездоганний досвід користувачів завдяки пильному моніторингу своєї інфраструктури. Найкращі рішення APM надають командам розробників інформацію, необхідну для пришвидшення роботи додатку та збільшення його пропускної здатності, а також виявляють та виправляють проблеми з продуктивністю до того, як вони вплинуть на кінцевого користувача.

Нами були проаналізовані найпопулярніші на сьогоднішній день інструменти моніторингу інформаційної системи та було обрано NewRelic як рішення для подальшого використання. З його допомогою були оптимізовані повільні запити до бази даних, проведено рефакторинг фрагментів програмного коду, що в свою чергу сприяло зменшенню часу виконання підпрограм.

Для забезпечення швидкості та стійкості інформаційної системи моніторинг та рефакторинг мають бути частиною повсякденної роботи інженерної команди. Авжеж це займає час, який команда могла б витрачати на розробку нового функціоналу, але в далекій перспективі це принесе більше вигоди для бізнесу.

РОЗДІЛ 3

ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Проблематика безпеки інформаційних систем

На сьогоднішній день світ дуже швидко змінюється і все нові і нові сфери нашого життя діджиталізуються. Усе від оплати комунальних послуг до купівлі продуктів чи оформлення державних документів можна зробити віддалено, не виходячи з дому. Все більше і більше чутливих даних ми відправляємо на сервери різних компаній та державних організацій, як от наприклад номер паспорта або кредитної картки. Витік цих даних може призвести до тяжких наслідків, таких як викрадення коштів або навіть втрата особистості. Саме тому безпека даних користувачів та інформаційної системи вцілому - є найголовнішою ціллю будь-якої компанії. Втрата даних неодмінно призведе до знищення іміджу бізнеса та відсутності довіри до нього з боку потенційних клієнтів.

Існує багато шляхів, які можуть використовувати зловмисники для атак на сервіси та викрадення інформації. Згідно з даними постачальника системи безпеки Cenzic, найбільш поширеними методами у березні 2018 року були:

- міжсайтовий скриптинг;
- SQL-ін'єкція;
- DoS-атака;
- міжсайтова підробка запиту.

Усіх їх об'єднує походження - помилка при розробці програмного забезпечення. При побудові будь-якого рішення, завжди потрібно пам'ятати про потребу мінімізації загрози для безпеки системи.

У наш час розробники постійно використовують так зване відкрите програмне забезпечення у своїй роботі. ПЗ з відкритим програмним кодом (OSS) від англійського Open-source software - це тип комп’ютерного програмного забезпечення, в якому вихідний код випускається за ліцензією, в якій власник авторських прав надає користувачам права використовувати, вивчати, змінювати та розповсюджувати програмне забезпечення кому завгодно і з будь-якою метою. Програми з відкритим кодом можуть зазвичай розробляються спільно групою незалежних інженерів для громадськості. OSS є яскравим прикладом відкритої співпраці. Більшість сучасних мов програмування, операційних систем, бібліотек та модулів є відкритим програмним забезпеченням. Їх використання значно пришвидшило розвиток програмної інженерії, оскільки вони дозволяють додавати або змінювати вже створений функіонал замість того щоб розробляти його з нуля.
Але повсемісне використання відкритого програмного забезпечення може нести загрозу безпеці інформаційної системи, оскільки воно може містити фрагменти коду, що можуть призвести до витоку інформації. Використовуючи OSS розробники мають покладатися на те що творці відкритого проекту мінімізували ризики для безпеки додатку. Тому до додавання будь-якої бібліотеки у програмний код потрібно відноситися відповідально.

Гарною практикою для забезпечення безпеки застосунку вважається використання інструментів для сканування додатку та його залежностей на рахунок можливих вразливостей. Таке рішення дозволяє перевірити наявність можливої загрози у коді відкритого програмного забезпечення та підказує потребу оновити це ПЗ до останньої версії. Оскільки такі інструменти мають доступ до застосунку та його залежностей, рекомендується вибирати загальновідомі та перевірені часом рішення, які використовують великі компанії. Це гарантія безпеки даних компанії та коду інформаційної системи, які є комерційною тайною.

3.2. Огляд інструментів для сканування додатку щодо можливих вразливостей

Головним критерієм вибору інструменту для сканування застосунку та його залежностей є підтримка стеку технологій, які використовуються у даному застосунку. Не менш важливою є підтримка градації загроз за степенем можливості виникнення та наслідками. На сьогоднішній день, у кібербезпеці найпоширенішою є градація за Національною базою вразливостей інформаційних систем від Національного інституту стандартів і технологій США, англійською National vulnerability database (NVD) of National Institute of Standards and Technology (NIST). Національний інститут стандартів і технологій - це лабораторія фізичних наук і нерегулююче агентство Міністерства торгівлі США. Його місія - сприяти інноваціям та конкурентоспроможності промисловості. Діяльність NIST організована за лабораторними програмами, які включають науку і техніку в наномасштабі, інженерію, інформаційні технології, дослідження нейтронів, вимірювання матеріалів та фізичні вимірювання. У NVD усі вразливості мають унікальний номер, степень загрози, перелік уражених версій програмного забезпечення та його конфігурацій, опис методів можливого використання зловмисниками та наслідки. На рисунку 3.1 зображено сторінка вразливості з ідентифікаційним номером CVE-2020-15169. У секції Quick Info (Швидкі дані) можна побачити дату додавання до бази, оновлення та ресурс з якого отримано інформацію про вразливість.

У секції Current Description (Поточний опис) можна знайти короткий опис вразливості та номер версій ПЗ, що мають її виправлення, авжеж якщо лише коли така версія існує. На рисунку 3.2 можна побачити секції Severity (Серйозність) та References to Advisories, Solutions and Tools (Посилання на поради, рішення та інструменти). Перша включає в себе відносні оцінки ступеню загрози за градацією компанії Github та Національної бази вразливостей інформаційних систем. Github - найбільший веб-сервіс для зберігання коду програмного забезпечення та його спільної розробки, це дуже впливова компанія у сучасному світу інформаційних технологій. Друга секція включає в себе посилання на сторонні ресурси з

інформацією про опис методів можливого використання зловмисниками та наслідки цієї вразливості. На рисунку 3.3 можна побачити посилання на сторінку з додатковою інформацією про проблему від організації, що повідомила про вразливість та секцію Known Affected Software Configuration (Відома вражена версія програмного забезпечення). Як зрозуміло з назви вона включає в себе перелік уражених версій програмного забезпечення та його конфігурацій.

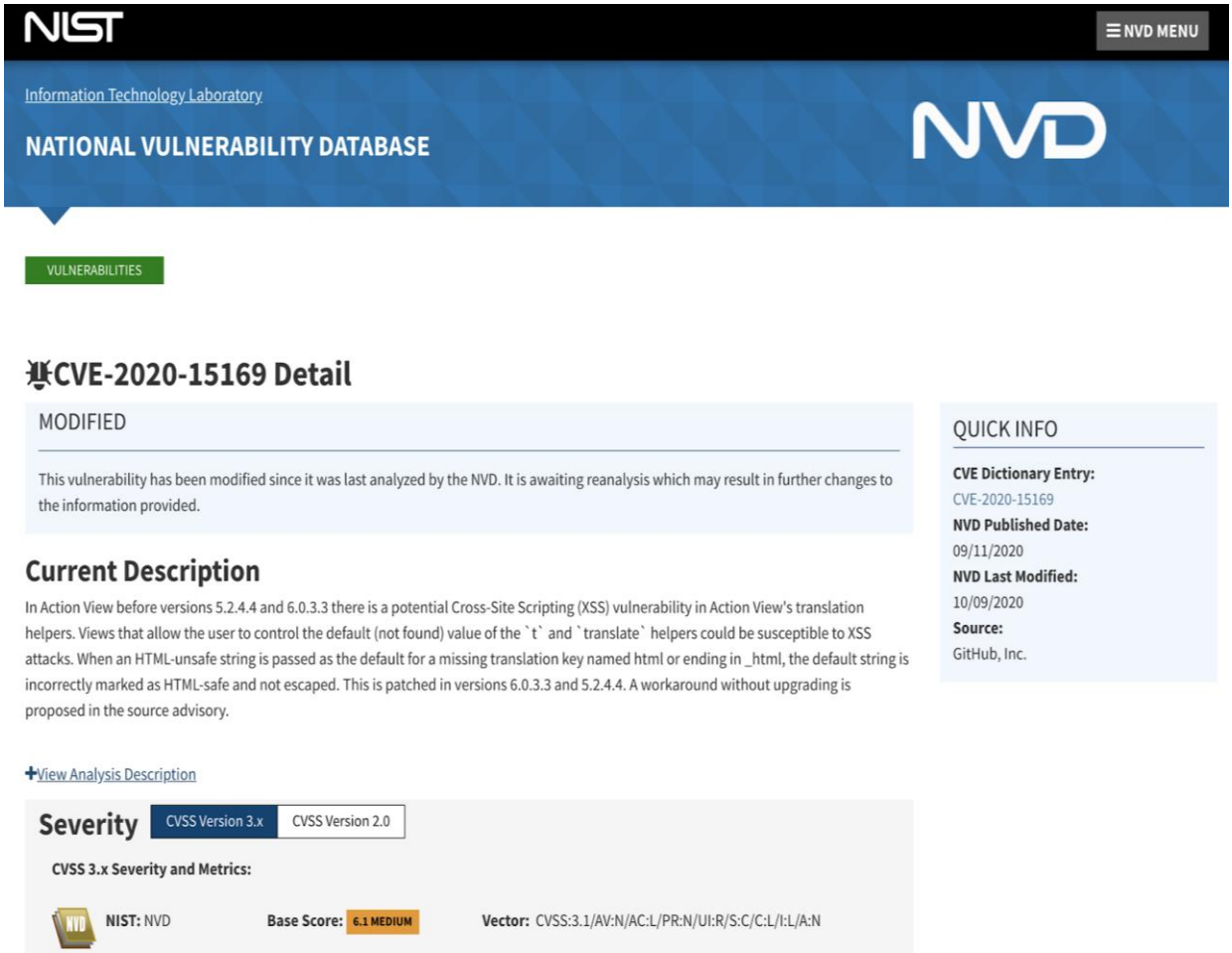


Рис. 3.1. Інформація про вразливість

+View Analysis Description

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 6.1 MEDIUM **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

CNA: GitHub, Inc. **Base Score:** 5.4 MEDIUM **Vector:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:L/A:N

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

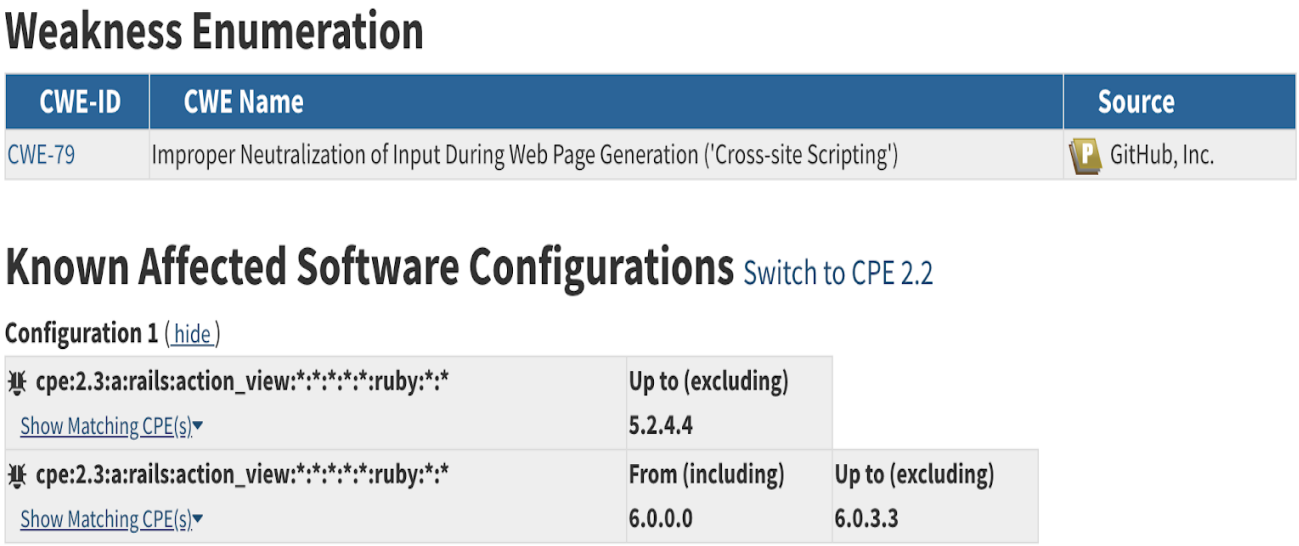
Note: It is possible that the NVD CVSS may not match that of the CNA. The most common reason for this is that publicly available information does not provide sufficient detail or that information simply was not available at the time the CVSS vector string was assigned.

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

Hyperlink	Resource
https://github.com/rails/rails/security/advisories/GHSA-cfjv-5498-mph5	Patch Third Party Advisory
https://lists.debian.org/debian-lts-announce/2020/10/msg00015.html	
https://lists.fedoraproject.org/archives/list/package-announce@lists.fedoraproject.org/message/XJ7NUWXAEVRQCROIIBV4C6WXO6IR3KSB/	
https://www.debian.org/security/2020/dsa-4766	

Рис. 3.2. Інформація про вразливість



Weakness Enumeration

CWE-ID	CWE Name	Source
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	GitHub, Inc.

Known Affected Software Configurations Switch to CPE 2.2

Configuration 1 (hide)

cpe:2.3:a:rails:action_view:*:*:*:*:*:ruby:*:* Show Matching CPE(s)	Up to (excluding) 5.2.4.4	
cpe:2.3:a:rails:action_view:*:*:*:*:*:ruby:*:* Show Matching CPE(s)	From (including) 6.0.0.0	Up to (excluding) 6.0.3.3

Рис. 3.3. Інформація про вразливість

Інструменти для сканування додатку та його залежностей, що спираються на описану вище градацію є більш надійними. Використовуючи їх можна бути впевненим у релевантності загроз застосунку та отримати додаткову інформацію про них з Національною бази вразливостей інформаційних систем США.

3.2.1. Інструмент Gemnasium

Gemnasium - комерційний інструмент, який пропонує безкоштовні стартові плани. Gemnasium має власну базу даних, яка базується на кількох джерелах. Однак, хоча вразливості щодня перевіряються вручну, рекомендації не публікуються автоматично.

Gemnasium надає унікальну функцію автоматичного оновлення, яка використовує спеціальний алгоритм для тестування розумних комбінацій наборів залежностей замість тестування всіх комбінацій, що економить купу часу. Gemnasium підтримує Ruby, NPM (JavaScript), PHP, Python і Bower (JavaScript). Ще однією унікальною пропозицією від Gemnasium є інтеграція Slack - користувачі отримують повідомлення через Slack у режимі реального часу, як тільки виявляється консультація. Приклад результатів сканування інформаційної системи за допомогою інструменту Gemnasium можна побачити на рис 3.4.

Філіп Лафук'єр з Gemnasium зазначив, що в планах на майбутнє буде корпоративна версія Gemnasium, яка працює в приміщеннях клієнтів із підтримкою більшої кількості мов, починаючи з Java.

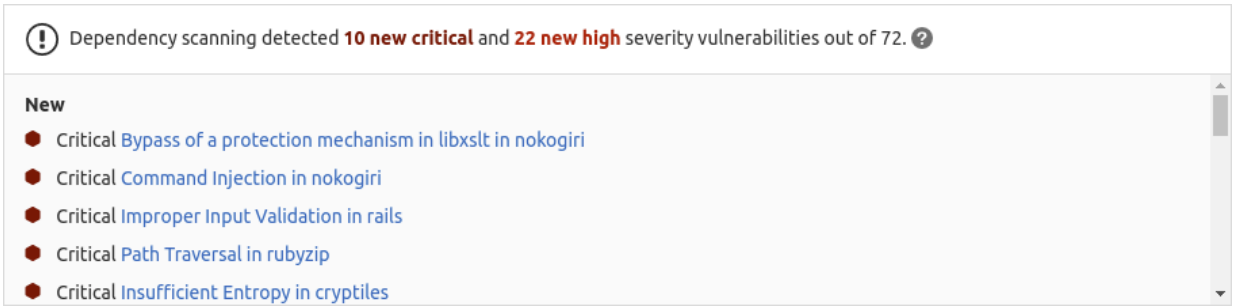


Рис. 3.4. Результати сканування за допомогою інструменту Gemnasium

3.2.2. Інструмент RetireJS

RetireJS - це програма для перевірки залежностей із відкритим кодом, специфічна для JavaScript. Проект в першу чергу орієнтований на простоту використання. Ось чому він має кілька компонентів, включаючи сканер командного рядка та плагіни для Grunt, Gulp, Chrome, Firefox, ZAP та Burp. RetireJS також зробив службу перевірки сайту доступною для розробників JS, які хочуть з'ясувати, чи використовують вони бібліотеку JavaScript з відомими вразливими місцями.

RetireJS отримує інформацію про вразливість з Національної бази вразливостей інформаційних систем від Національного інституту стандартів і технологій США (NIST NVD), а також безліч інших джерел, включаючи списки розсилки, системи відстеження помилок та блоги для популярних проектів JavaScript. Ерленд Офтедал творець RetireJS вважає, що безпека - це проблема кожного, і потрібна подальша співпраця: "Я хотів би, щоб автори популярних фреймворків з відкритим кодом самі починали повідомляти про виправлення безпеки командам таких інструментів, як Retire.js, щоб забезпечити безпеку для користувачів свого програмного забезпечення". Приклад роботи з консольною утилітою RetireJS можна побачити на рис 3.5.

Його можна використовувати чотирма способами:

- Сканер командного рядка для сканування програми Node.js.

- Плагін Grunt (Grunt-пенсія), який використовується для сканування додатків із підтримкою Grunt.
- Розширення браузера (Chrome та Firefox). Вони перевіряють відвідувані сайти на наявність посилань на небезпечні бібліотеки та містять попередження на консолі розробника.
- Плагін Burp та OWASP Zap, що використовується для тестування на проникнення.

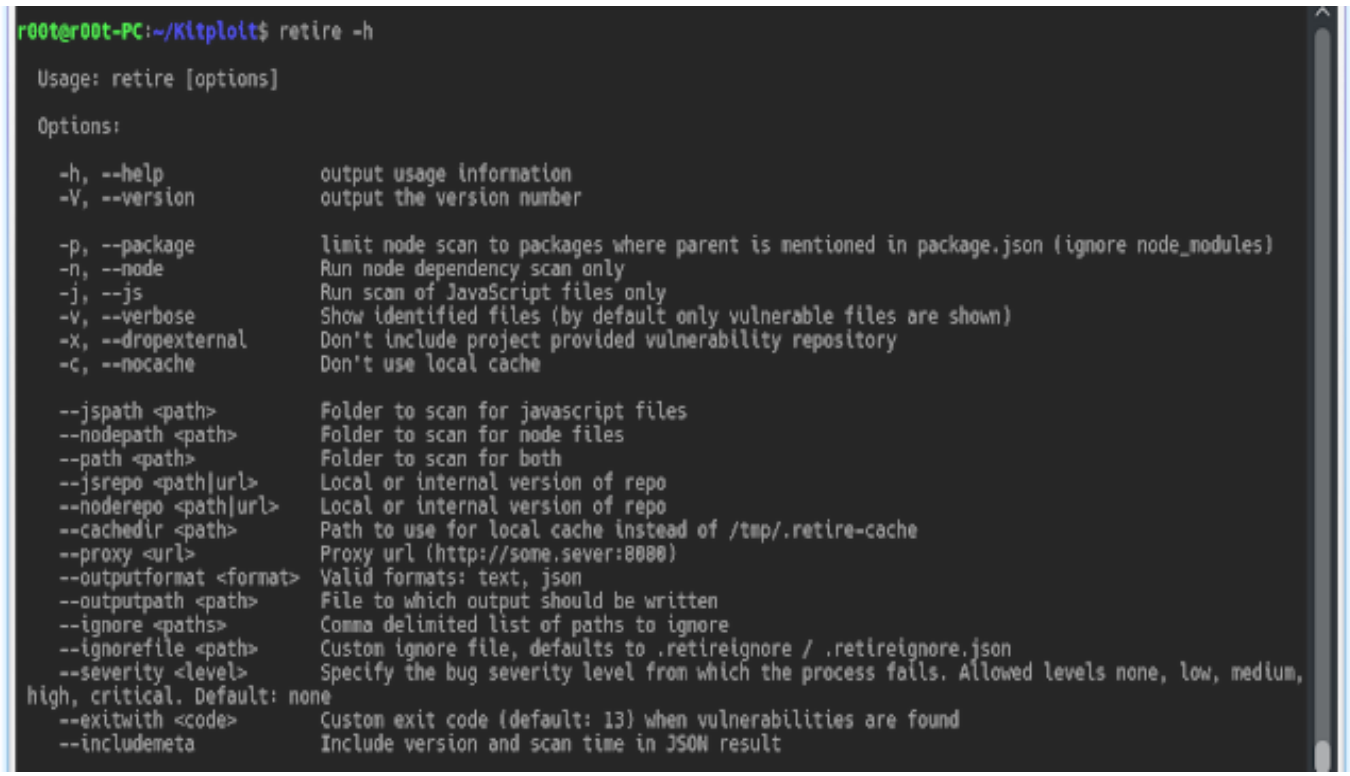


Рис. 3.5. Робота з консольною утилітою RetireJS

3.2.3. Інструмент Snyk

Snyk - це відкрита платформа безпеки для виявлення вразливостей у вихідному коді програмного забезпечення. Він ефективно працює і в контейнерних програмах. Подібно до того, як антивірус сканує ваш пристрій і виявляє загрози, так само, як він сканує ваш вихідний код і він надає ступінь серйозності вразливості та класифікує їх як основні незначні або критичні, що може допомогти вам скласти уявлення про терміновість проблеми та підхід до прийняття відповідних заходів. Зробивши це на крок далі, він також надає опис вразливість, позиція в коді, де

вразливість присутня, та надання виправлення цієї вразливості. Його можна інтегрувати як у ваші сховища Git, так і в конвеєр CI / CD. CI / CD - це комбінація безперервної інтеграції (CI, від англійської continuous integration) і безперервної доставки або безперервного розгортання (CD, від англійської continuous delivery; continuous deployment.)

З удосконаленням машинного навчання та глибокого навчання такі інструменти є досить перспективними для створення інтелектуальних ботів безпеки, які можуть ефективно створювати безпечні середовища для майбутніх розгортань.

Ключовою особливість Snyk є пітримка багатьох мов програмування та фреймворків. Ось деякі з них: Ruby, Python, Java, C#, ASP.NET, Javascript, React, Django, Angular, Vue.js, Ruby on Rails, Spring. Користувацький інтерфейс даного інструменту можна побачити на рис 3.6, 3.7.

Snyk має власну базу даних про вразливості, яка отримує свої дані Національної бази вразливостей інформаційних систем від Національного інституту стандартів і технологій США (NIST NVD). Основна увага Snyk зосереджена на масштабуванні обробки відомих вразливостей у всій організації та її командах, за допомогою кращих інструментів співпраці та більш глибоких інтеграцій з сервісом збереження вихідного коду - GitHub. Генеральний директор Snyk, Гай Поджерні, зазначив, що майбутні плани Snyk включають створення інструментів виконання, які дадуть розробникам кращу видимість та контроль під час запуску сторонії пакетів з відкритим кодом у своїх інформаційних системах.

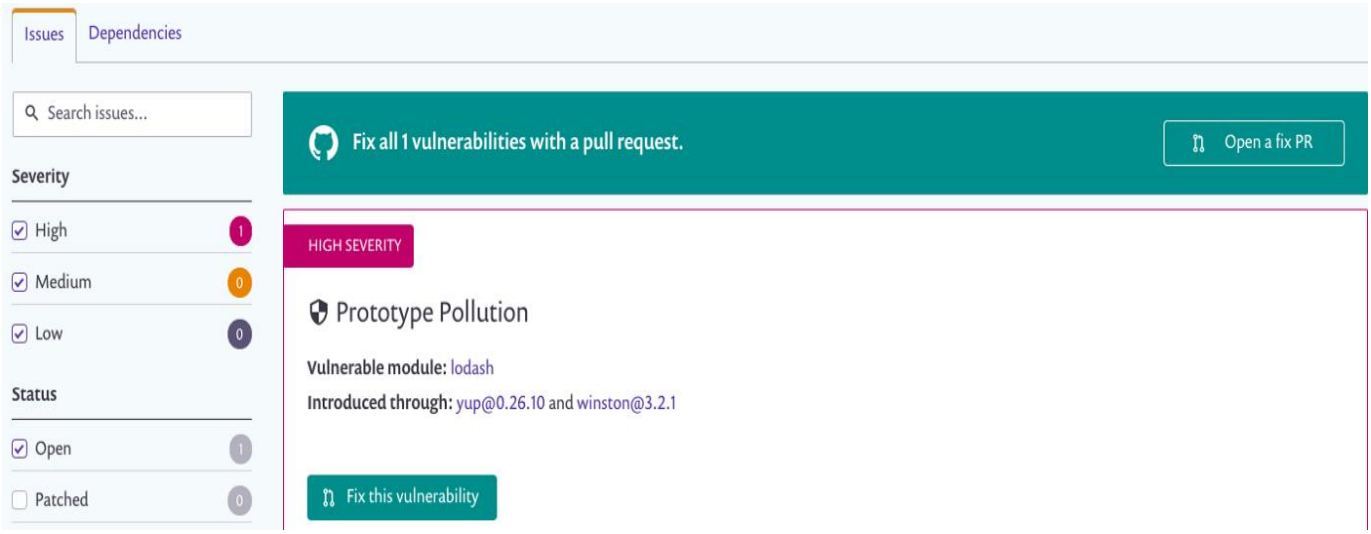


Рис. 3.6. Користувацький інтерфейс Snyk

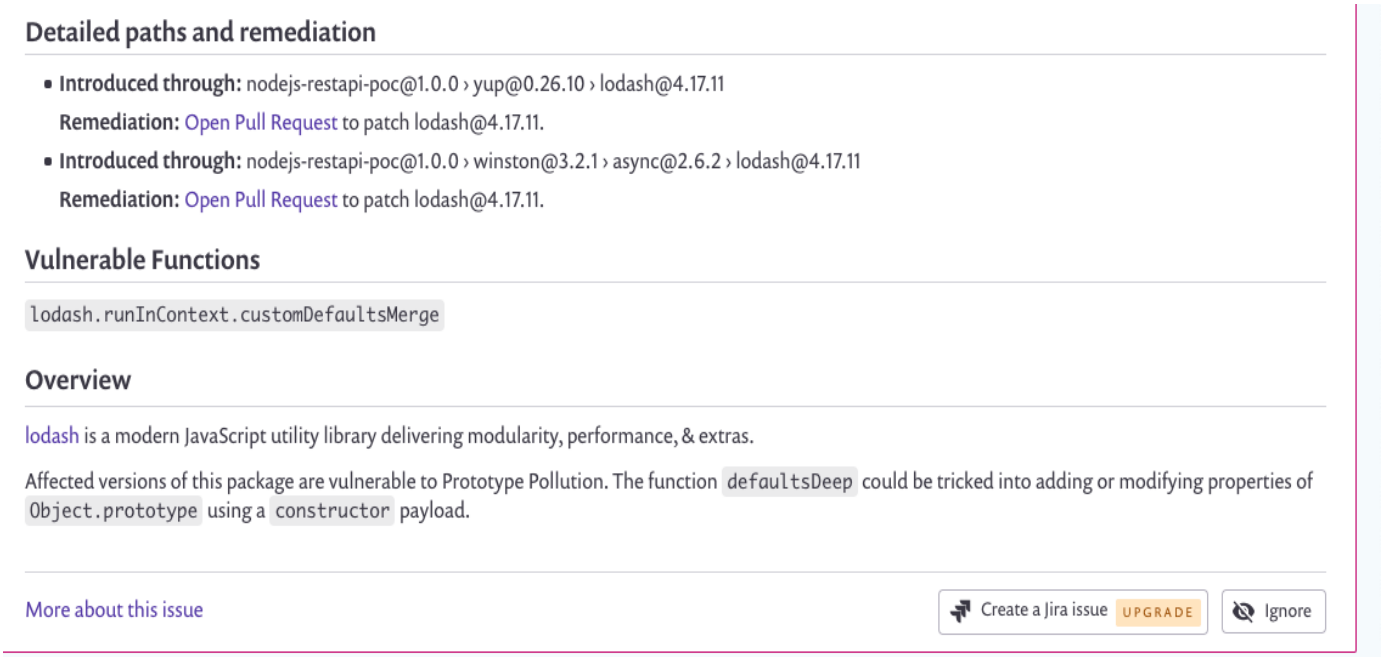


Рис. 3.7. Користувацький інтерфейс Snyk

3.2.4. Інструмент Hakiri

Hakiri - комерційний інструмент, який пропонує перевірку залежності для проектів, що розташовані на сервісі збереження вихідного коду - GitHub та створені на основі фреймворку Ruby та Rails, за допомогою статичного аналізу коду. Він пропонує безкоштовні плани для державних проектів з відкритим кодом та платні

плани для приватних проектів. Як джерело даних він використовує Національну базу вразливостей інформаційних систем від Національного інституту стандартів і технологій США (NIST NVD) та базу даних Ruby Advisory. Коли виявляється нова вразливість, ви отримуєте електронною поштою сповіщення з її деталями та способами зменшення ризиків для безпеки. Почати роботу з Hakiri дуже просто: просто підключіть свій обліковий запис GitHub, виберіть будь-який репозиторій та відділення Ruby та зачекайте кілька секунд, перш ніж Hakiri проведе всі свої тести безпеки. Користувацький інтерфейс даного інструменту можна побачити на рис 3.8, 3.9.

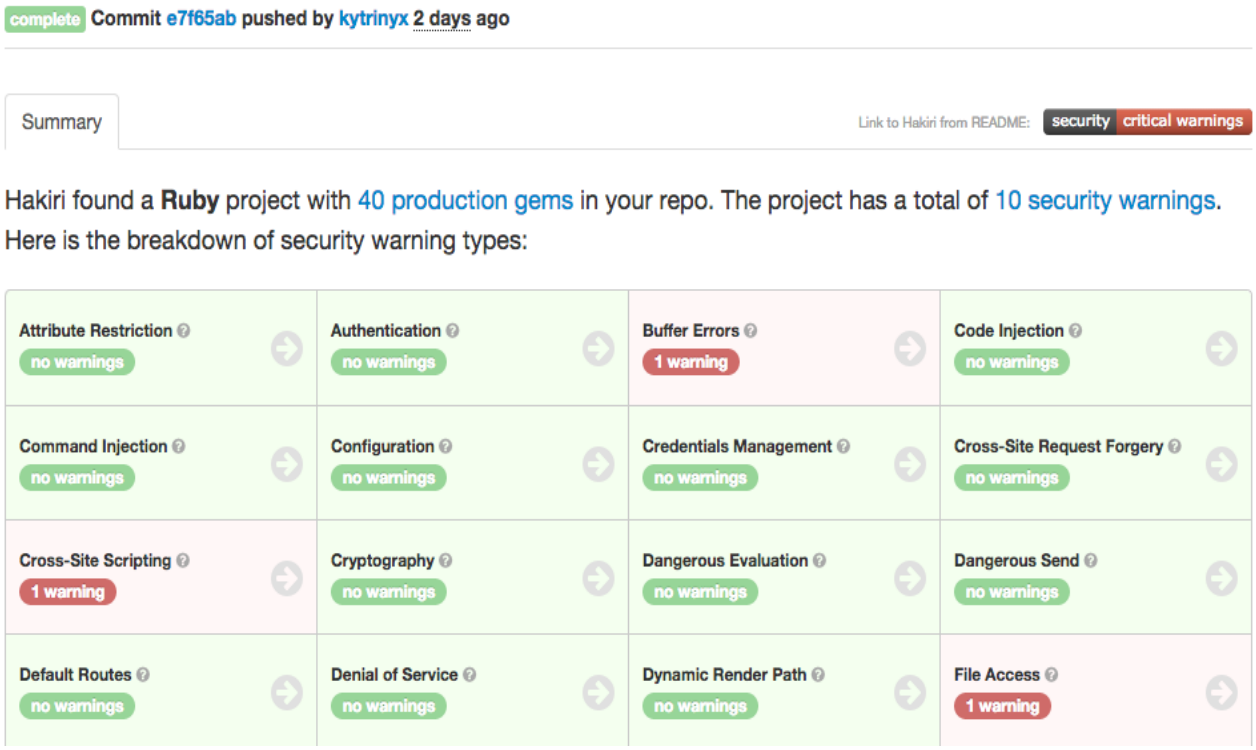


Рис. 3.8. Користувацький інтерфейс Hakiri

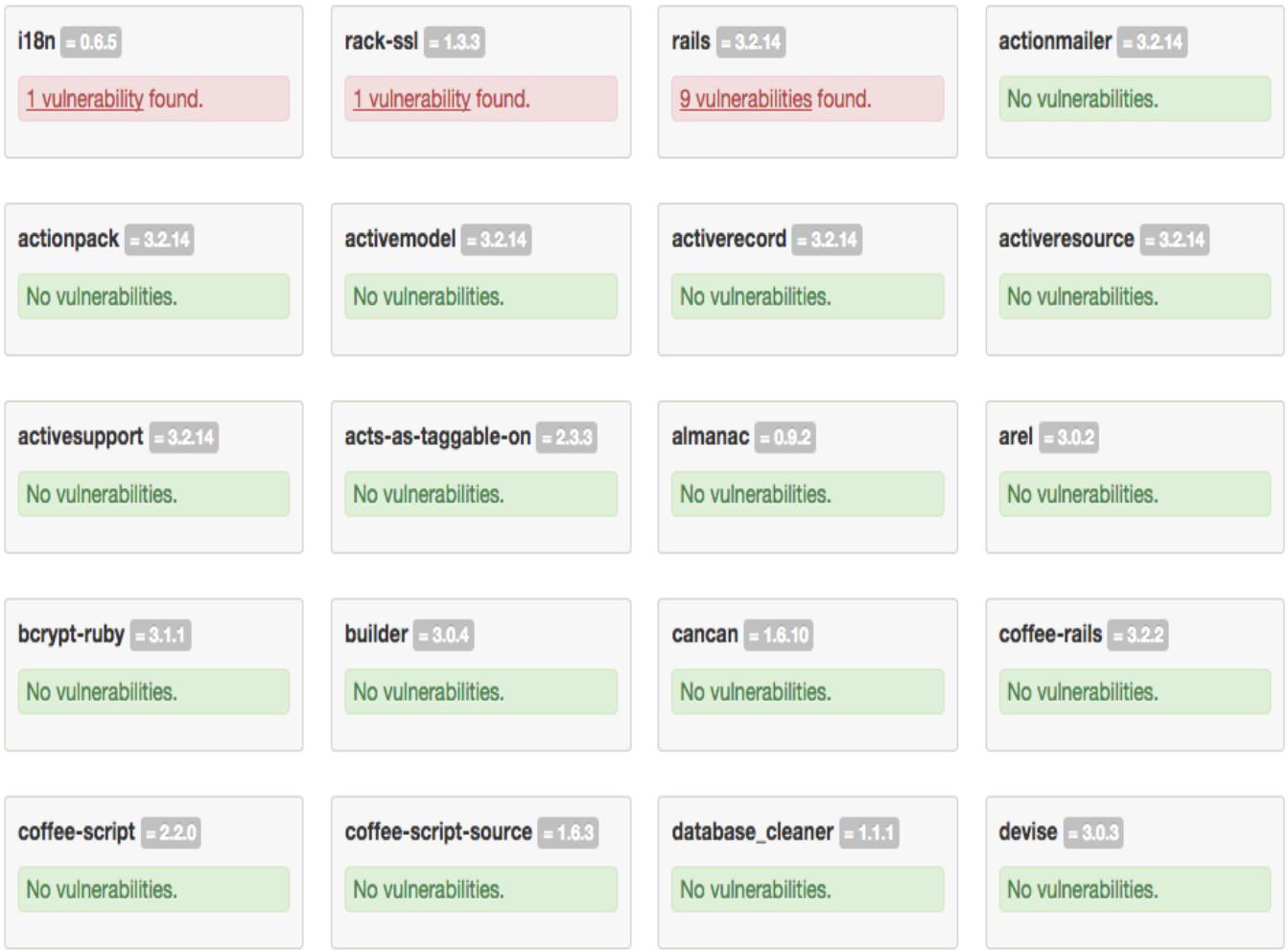


Рис. 3.9. Користувацький інтерфейс Hakiri

3.2.5. Інструмент OSSIndex

OSSIndex - це безкоштовний сервіс, який компанія Sonatype надає розробникам, щоб перевірити, чи є в будь-якій бібліотеці відомі та досліджені вразливості безпеки. OSSIndex надає просту у використанні функцію пошуку для швидкого пошуку вразливостей у будь-якій бібліотеці. OSSIndex підтримує кілька технологій. Він витягує інформацію про залежності з таких пакетних менеджерів як NPM, Nuget, Maven Central Repository, Bower, Chocolatey та MSI (що означає, що він охоплює екосистеми JavaScript, .NET / C # та Java). OSSIndex також безкоштовно надає API за допомогою якого можна отримати детальну інформацію про вразливості.

Індекс OSS містить сукупну інформацію з багатьох джерел інформації про вразливість, зокрема:

- БД загальні вразливості та ризики (CVE).
- Зростаючий перелік публічних джерел вразливостей.
- Внески громади.
- Проблеми, пов’язані з безпекою, виявлені спеціалізованими дослідниками OSS Index.

OSSIndex не надає підготовлених деталей безпеки та рекомендацій щодо відновлення. Важливо пам’ятати, що якщо OSSIndex не виявив вразливостей для бібліотеки, яку ви шукаєте, це не означає, що в ній їх немає. Це означає лише те, що OSS Index не знайшов жодного з джерел, яке б заявляло про їх наявність у даній бібліотеці.

Також OSSIndex отримує інформацію про вразливість з Національної бази вразливостей інформаційних систем від Національного інституту стандартів і технологій США (NIST NVD). Кен Дак з OSSIndex планує найближчим часом включити автоматичне імпортування вразливостей з деяких ключових списків розсилки, баз даних та систем відстеження помилок. Користувацький інтерфейс даного інструменту можна побачити на рис 3.10, 3.11.

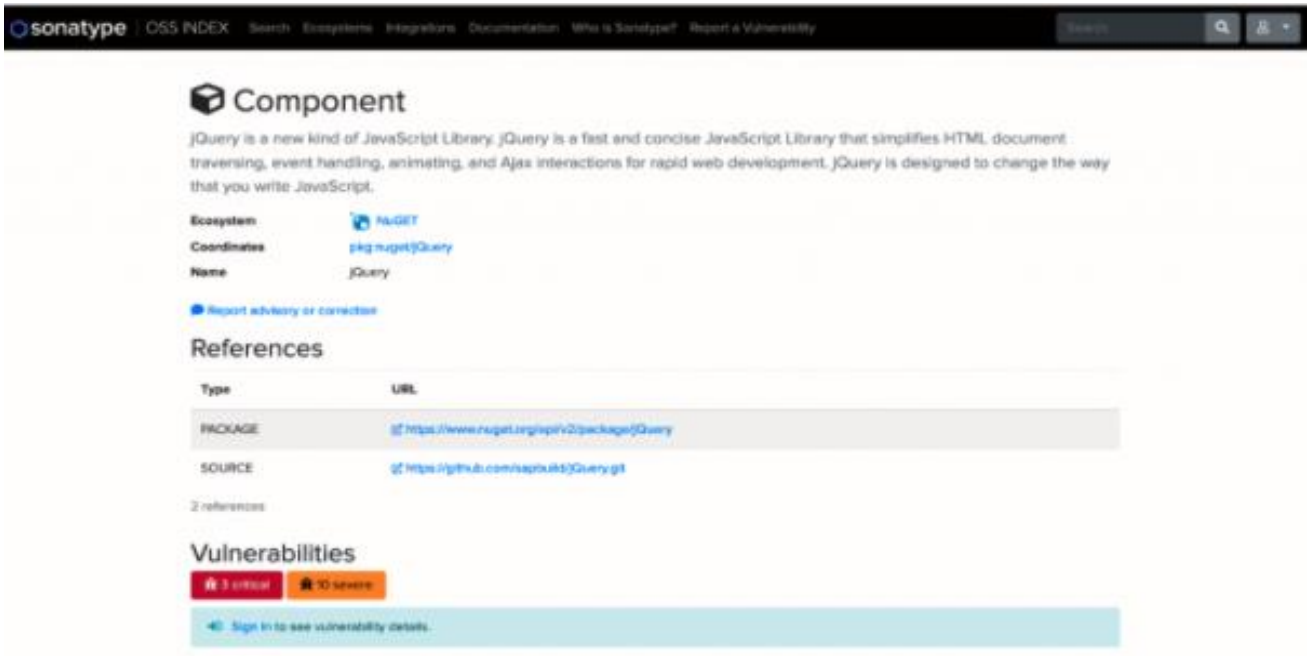


Рис. 3.10. Користувацький інтерфейс OSSIndex

Vulnerabilities

	Title	Severity	CVSS Score
	[CVE-2016-10707] jQuery 3.0.0-rc.1 is vulnerable to Denial of Service (DoS) due to removing a log...	Critical	7.5
	CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Critical	7.2
	CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Critical	7.2
	CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Severe	6.1
	CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Severe	6.1
	[CVE-2019-11358] Improper Neutralization of Input During Web Page Generation ("Cross-site Scripting")	Severe	6.1
	[CVE-2014-6071] Improper Neutralization of Input During Web Page Generation ("Cross-site Scripting")	Severe	6.1
	[CVE-2012-6708] Improper Neutralization of Input During Web Page Generation ("Cross-site Scripting")	Severe	6.1
	[CVE-2015-9251] Improper Neutralization of Input During Web Page Generation ("Cross-site Scripting")	Severe	6.1
	CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Severe	6.1
	CWE-121: Stack-based Buffer Overflow	Severe	5.1
	[CVE-2007-2379] The jQuery framework exchanges data using JavaScript Object Notation (JSON) with...	Severe	5
	[CVE-2011-4969] Improper Neutralization of Input During Web Page Generation ("Cross-site Scripting")	Severe	4.3

13 vulnerabilities

Рис. 3.11. Користувацький інтерфейс OSSIndex

3.2.6. Інструмент SRC:CLR

SRC:CLR скорочено від Source Clear - це комерційний інструмент із кількома цікавими атрибутами. Він має власну базу даних, яка використовує Національну бази вразливостей інформаційних систем від Національного інституту стандартів і технологій США (NIST NVD), але також отримує інформацію про вразливість зі списків розсилки та кількох інших джерел.

Він пропонує масу плагінів для декількох середовищ розробки, систем розгортання та сховищ джерел, а також інтерфейс командного рядка. Нарешті, Source Clear використовує "ідентифікацію вразливих методів", що дозволяє зрозуміти, чи використовує програма вразливість, виявлену в залежності. Це функція, яка різко зменшує помилкові спрацьовування та надає розробникам детальні цільові звіти щодо важливих вразливостей.

Основна увага SourceClear - це розробка програмного забезпечення з відкритим кодом. Оскільки розробники все частіше споживають і поширюють безкоштовні компоненти та бібліотеки з відкритим кодом та сторонніми розробниками, їх продукти можуть стати вразливими до злому. Інструменти SourceClear допомагають розробнику, повідомляючи їм, яким відкритим кодом вони користуються, хто його створив, що він робить (або що може робити) у своїх додатках та які компоненти мають уразливості. Вони стають частиною робочого процесу розробників та вивчають ризики безпеки відкритого коду в режимі реального часу. Їх засоби аналітики та машинного навчання аналізують компоненти з відкритим кодом та звітують про їх походження, створення та вплив на програми. Вони розповідають розробникам, які вразливості можуть використовувати хакери та як їх запобігти. Послуга також дозволяє користувачам сканувати свої сховища GitHub і працювати в їх системах безперервної інтеграції.

Наразі SourceClear підтримує Java, JavaScript, Ruby on Rails, Node.js та Python. Команда розробників інструменту оголосила про плани підтримувати Scala та C.

3.2.7. Вибір інструменту для сканування веб-застосунку щодо можливої загрози безпеки

Порівнявши шість популярний на сьогоднішній день інструменти для сканування застосунку та його залежностей можна зробити висновок, що для наших цілей найкраще підійде snyk. Він підтримує стек технологій компанії та має зручний користувацький інтерфейс. На відміну від багатьох інших інструментів, snyk не зосереджений лише на декількох мовах програмування або фреймворках, а має підтримку усіх основних сучасних технологій. Це означає, що якщо в майбутньому команда розробників забажає використовувати нову мову або фреймворк, не потрібно буде шукати, вивчати та інтегрувати новий інструмент для сканування інформаційної системи, а буде можливо використовувати вже напрацьовані практики.

Не менш важливо, що у snyk є підтримка градації загроз за ступенем можливості виникнення та наслідками. Вона спирається на градацію від Національної бази вразливостей інформаційних систем від Національного інституту стандартів і технологій США, тому можна бути впевненим у релевантності даних. Також інструмент надає додаткову інформацію про перелік уражених версій програмного забезпечення та його конфігурацій, опис методів можливого використання зловмисниками та наслідки.

Snyk має можливості інтегрування, як з основними сервісами для зберігання програмного коду - Github, Gitlab, Bitbucket так і сервісами неперервної інтеграції та розгортання - Circle CI, Travis CI, Github Actions.

Цей інструмент є комерційним, тому за нього потрібно платити. Але крім недоліку у вигляді додаткових витрат це має і переваги - snyk має команду підтримки, яка може допомогти з інтеграцією та відповісти на питання розробників. Також це означає, що інструмент буде мати постійне оновлення, виправлення помилок, отримувати нові функції та підтримку останніх технологій.

Усі ці переваги роблять snyk - найкращим інструментом для сканування інформаційних систем. Він може стати надійним помічником для забезпечення безпеки застосунків, а як наслідок захистити дані користувачів та імідж компанії.

3.3. Використання інструменту Snyk для усунення можливої загрози безпеки

Snyk має можливість працювати у двох режимах: CLI та UI. CLI від англійського Common Language Infrastructure - це різновид текстового інтерфейсу користувача й комп'ютера, в якому інструкції комп'ютеру можна дати тільки введенням із клавіатури текстових рядків (команд). Також відомий під назвою консоль. UI від англійського User Interface - це графічний інтерфейс користувача.

У графічному інтерфейсі користувача, що зображений на рисунку 3.12, можна побачити список можливих інтеграцій. Вибравши потрібний сервіс для зберігання програмного коду та ввівши дані для входу в нього, отримуємо список доступних репозитаріїв для сканування, рисунок 3.13.

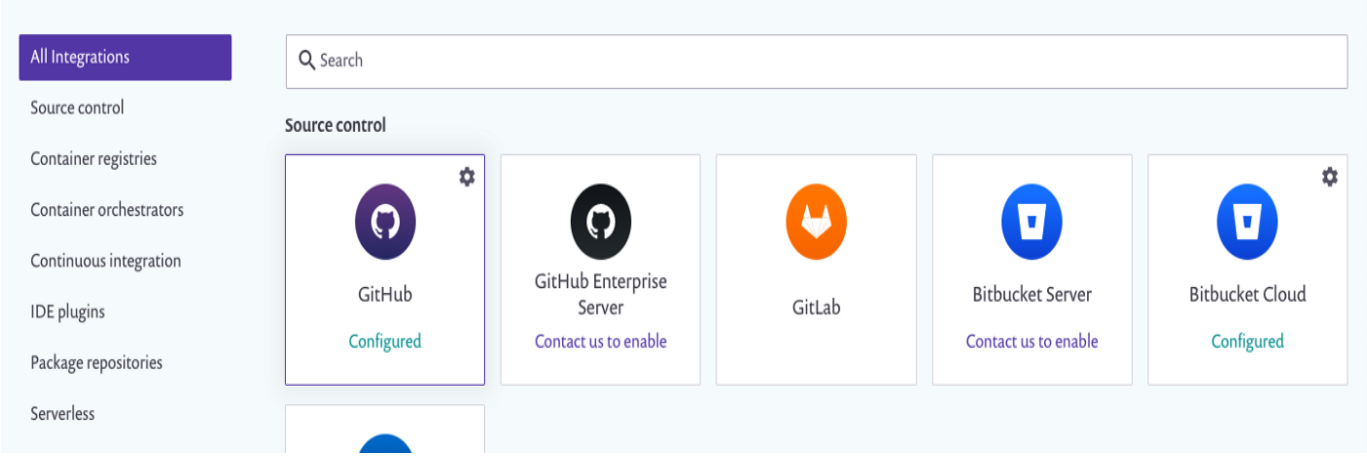


Рис. 3.12. Графічному інтерфейсі користувача Snyk

Наступним кроком, потрібно вибрати проект, що буде додано у список відстеження. Надалі він буде автоматично скануватися інструментом, після кожної доданої зміни, немає потреби кожного разу його додавати. Після закінчення сканування коду репозиторію, інструмент видає список знайдених вразливостей.

Відкривши будь-яку вразлисть можна побачити детальну інформацію, яка включає унікальний номер, степень загрози, перелік уражених версій програмного забезпечення та його конфігурацій, опис методів можливого використання зловмисниками та наслідки, рис 3.14, 3.15.

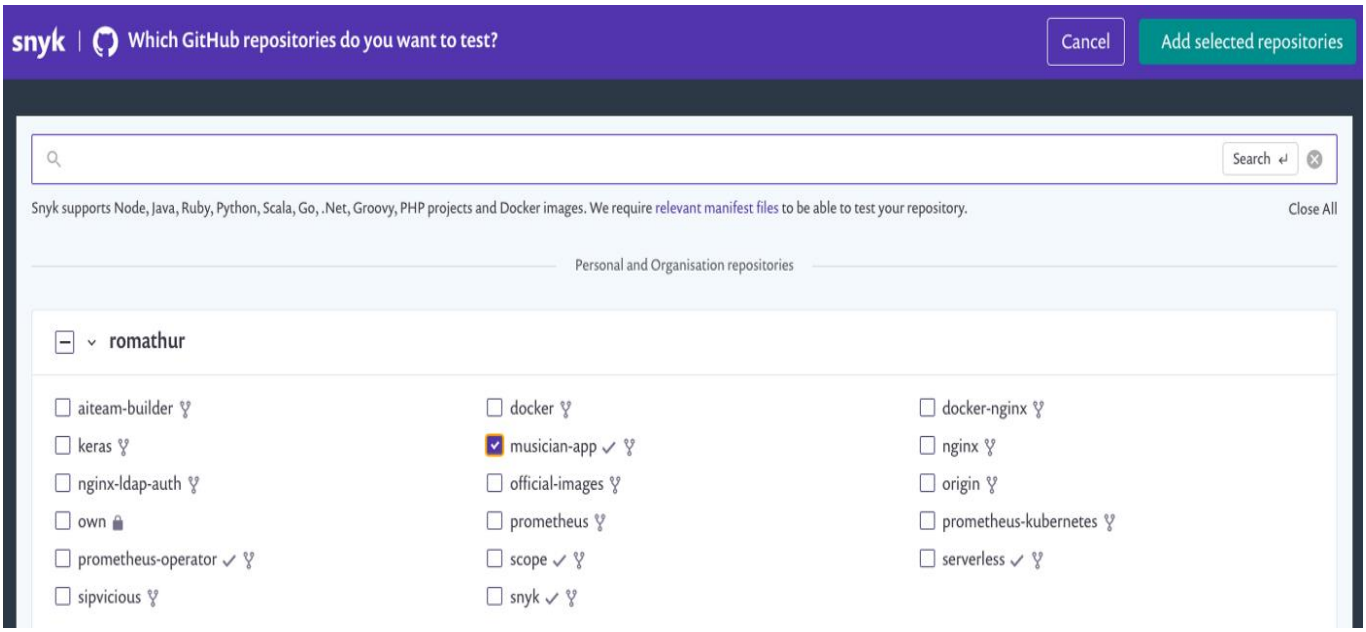


Рис. 3.13. Список доступних репозитаріїв

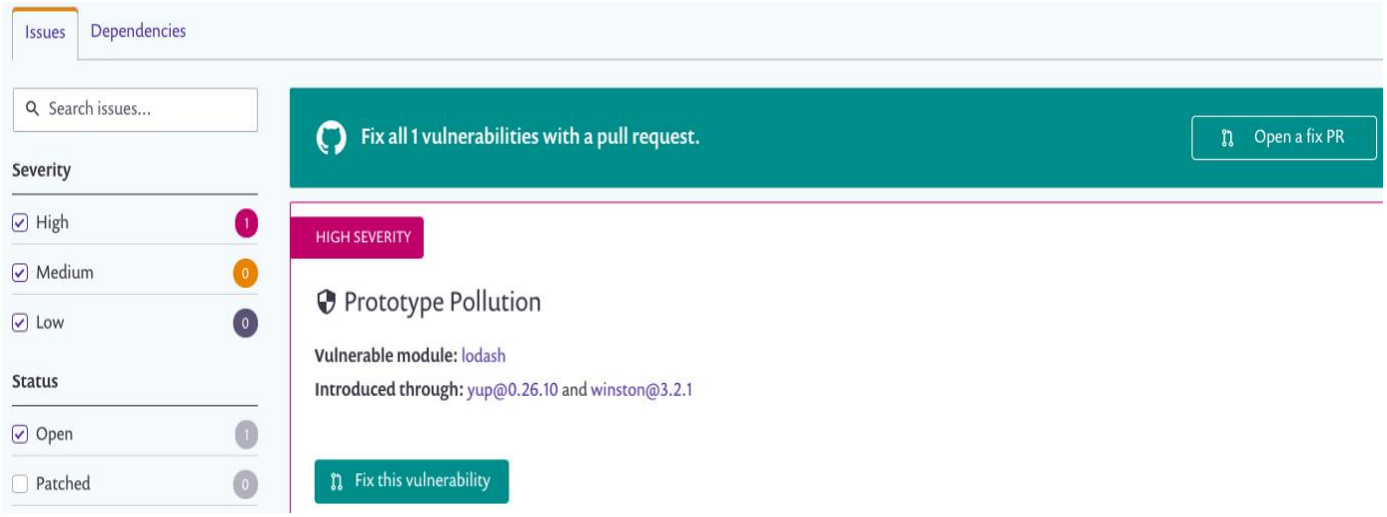


Рис. 3.14. Детальна інформація про вразливість

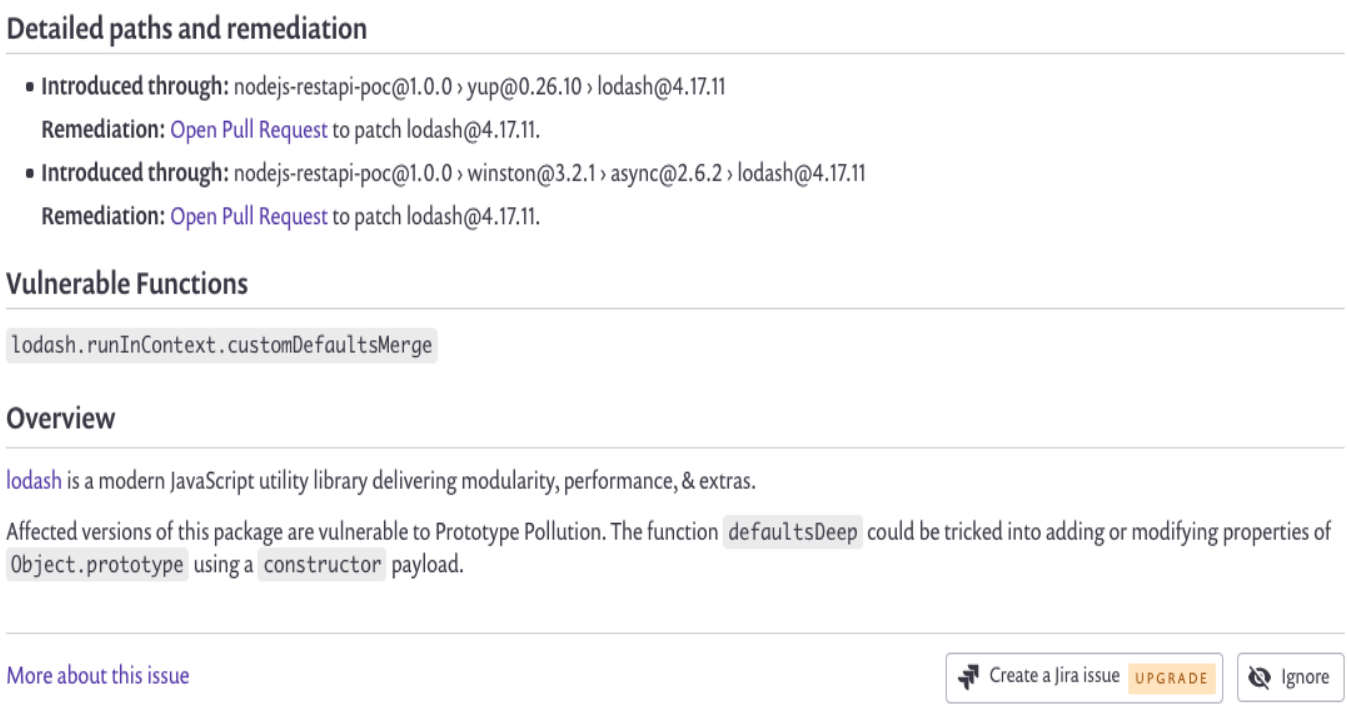


Рис. 3.15. Детальна інформація про вразливість

На сторінці з детальною інформацією можна побачити кнопку для виправлення вразлистові. Вона є опціональною, оскільки не завжди існує новіша версія програмного забезпечення, що включає в себе виправлення або інші шляхи зменшення ризиків для безпеки додатку. Після натискання цієї кнопки інструмент автоматично створить нову гілку у системі для зберігання програмного коду та відправить запит на злиття з основною гілкою, як показано на рисунку 3.16.

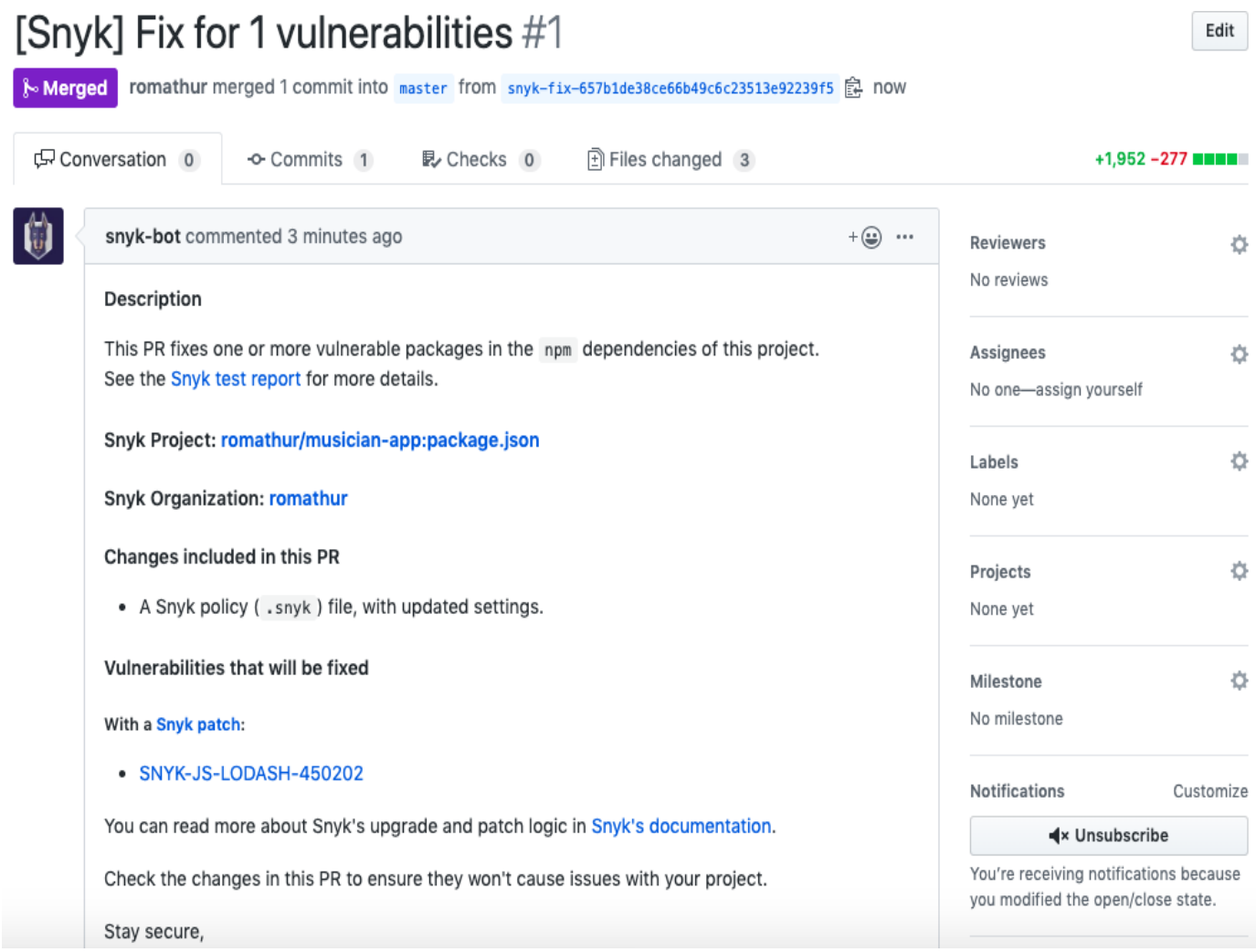


Рис. 3.16. Запит на злиття з основною гілкою

Іншим можливим шляхом взаємодії з даним інструментом є використання консольної утиліти. Для того, щоб її встановити потрібно відкрити термінал у кореневій папці проекту на локальній машині і виконати команду npm install snyk, як показано на рисунку 3.17. Умовою для її виконання є наявність на локальній машині інструменту npm.

NPM від англійського Node Package Manager - це менеджер пакетів для мов програмування JavaScript та Typescript. Для середовища виконання Node.js це менеджер пакетів за замовчуванням. Він включає в себе клієнт командного рядка, який також називається npm, а також онлайн-базу даних публічних та приватних пакунків, яка називається реєстром npm. Реєстр доступний через клієнт, а доступні пакунки можна переглядати та шукати через веб-сайт npm. Інструкцію по його

встановленню можна знайти на офіційному сайті інструменту - https://www.npmjs.com.

Рис. 3.17. Виконання команди npm install snyk

Наступним кроком є виконання команди snyk wizard у кореневій папці проекту, як показано на рисунку 3.18. Ця команда запускає аналіз додатку та його залежностей на наявність загроз для безпеки. Як і у графічній версії snyk, даний застосунок буде автоматично сканuватися інструментом, після кожної доданої зміни, немає потреби кожного разу виконувати команду. Після закінчення сканування коду репозиторію, інструмент видає список знайдених вразливостей з коротким описом кожної та посиланнями на веб-сайт. Пройшовши за посиланням будь-якої вразлисті можна побачити детальну інформацію про неї, яка включає унікальний номер, степень загрози, перелік уражених версій програмного забезпечення та його конфігурацій, опис методів можливого використання зловмисниками та наслідки. Виконавши команду snyk help, можна отримати список іншим команд і їх опис. Дана консольна утиліта повністю дублює функіонал графічного інтерфейсу користувача.

Обидва режими працюють разом, це означає що якщо спочатку виконати аналіз безпеки за допомогою команд у терміналі, то надалі відкривши веб-версію можна отримати результати сканування додатку та його залежностей.

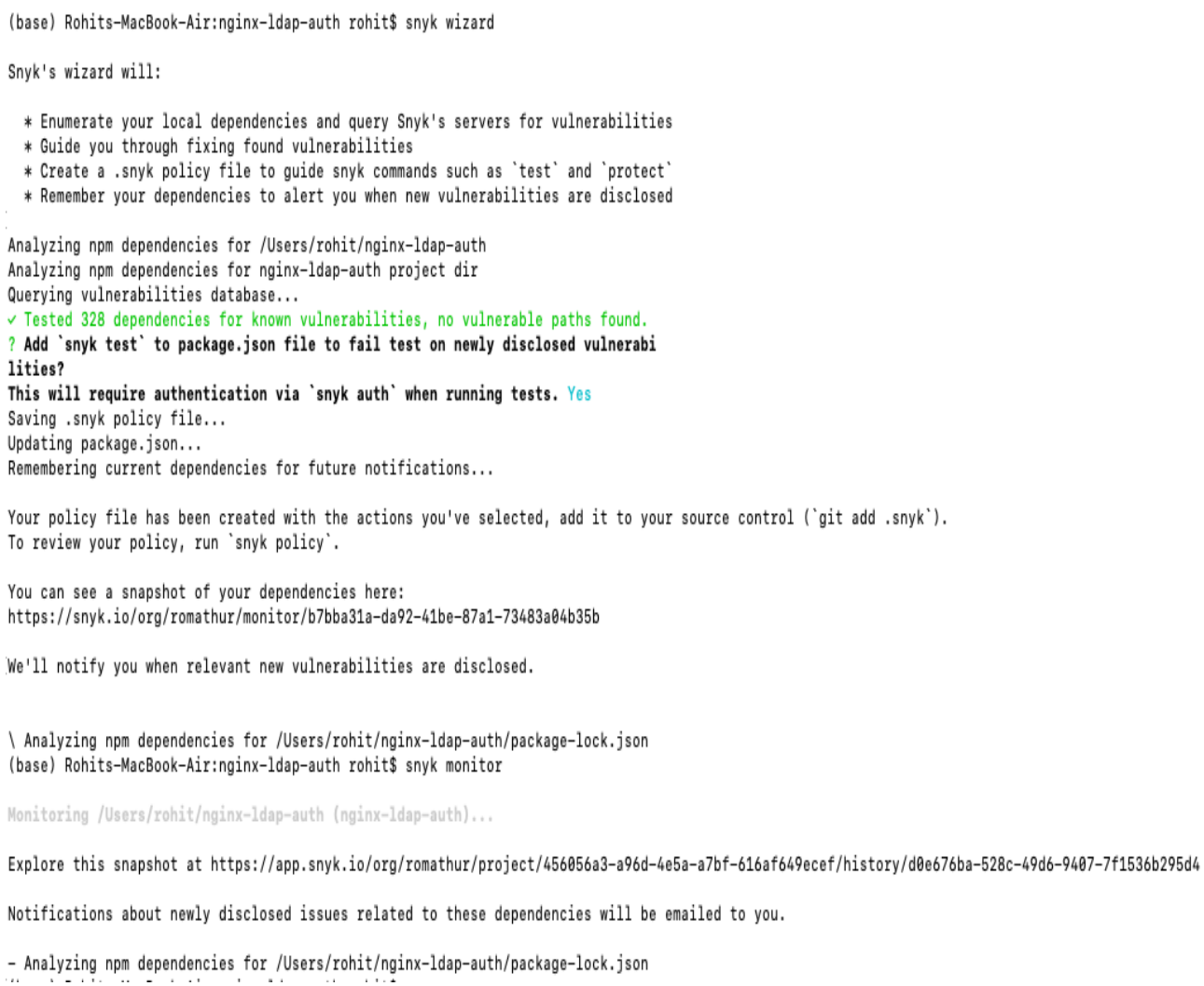


Рис. 3.18. Виконання команди snyk wizard

Після проведеного аналізу у коді можуть бути знайдені проблеми, які можуть мати шлях вирішення, наприклад оновлення програмного забезпечення. . Якщо , на вашу думку, вразливість не є важливою і ви не хочете вносити зміни, у вас є можливість змінити політику snyk та ігнорувати дану вразливість, надавши дуже дату закінчення терміну дії ігнорування. Також може бути випадок, коли в даний час ще немає версії з виправленням вразливості, тоді можна відредагувати файл політики snyk для ігнорування загрозу протягом деякого часу, через вказаний період snyk нагадає про потребу виправлення даної вразливості, як будильник. Ви також можете встановити частоту сканування в графічному інтерфейсі як щоденний щомісячний або щотижневий.

3.3.1. Аналіз результатів використання Snyk

Використання інструменту для сканування додатку щодо можливої загрози безпеки має стати постійною звичкою для команди розробки, щоб досягти бажаного результату. Забезпечення безпеки даних користувачів та компанії має бути безперервним процесом, нові вразливості або методи їх використання з'являються щоденно. Саме тому одноразовий аналіз додатку та його залежностей, не призведе до досягнення цілі, а є лише втратою часу і коштів компанії.

Як видно з рисунку 3.19, регулярне використання інструменту snyk протягом трьох місяців дозволило поступово зменшити кількість можливих загроз безпеки з більш ніж півтори тисячі до менше ніж п'ятсот у всій інформаційній системі. А кількість сервісів, що мають вразливості з більш ніж сто пятидесяти до менше ніж ста. Як видно з графіку тенденція на зменшення загроз безпеки є постійною.

Треба також зауважити, що неможливо оцінювати кібербезпеку застосунку спираючись лише на ці цифри. Тому що не всі ці можливі загрози безпеки насправді є небезпечними та потребують вирішення. Наприклад, вразливість SQL-ін'єкції у будь-якій формі в адміністративній панелі не несе ніякого сенсу, оскільки її можна використати лише маючи доступ у цю панель. Якщо зловмисники отримали такий доступ, то це вже катастрофа, оскільки з неї вони можуть отримати більшість даних користувачів у відкритому вигляді. Тому в даному випадку запобігання даній вразливості немає сенсу, краще використати ресурси на забезпечення безпеки доступу до адміністративної панелі.

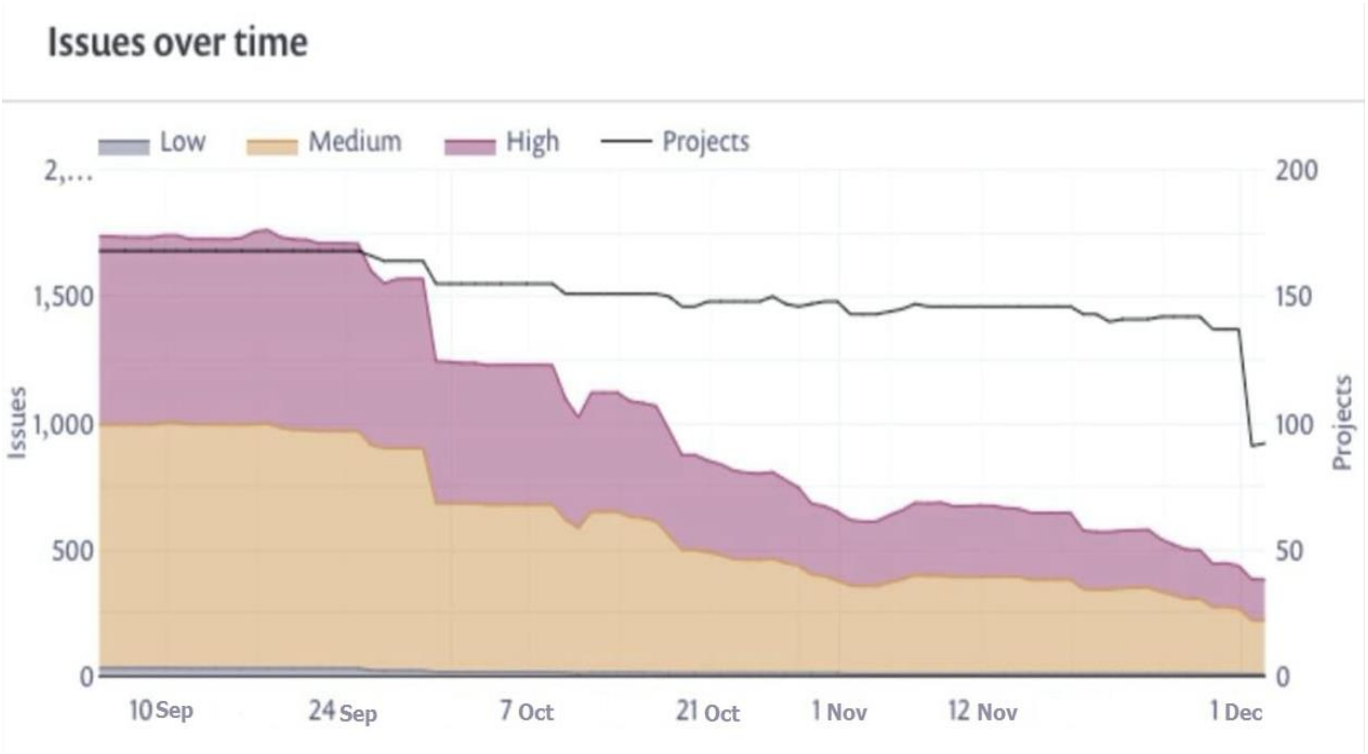


Рис. 3.19. Графік зміни кількості вразливостей

ВИСНОВОК ДО РОЗДІЛУ 3

Побудувати сучасний додаток практично неможливо, не покладаючись на сторонні бібліотеки. Програмне забезпечення з відкритим кодом стало надзвичайним бумом для розробки програмного забезпечення. Це допомогло розробникам створювати дедалі складніші програми, які роблять Інтернет таким, яким він є сьогодні.

Однак використання сторонніх бібліотек з відкритим кодом також представляє собою загрозу з точки зору безпеки. Наприклад, ви можете використовувати усі найкращі практики проектування програмного забезпечення, проводити регулярні аудити для мінімізації ризиків для безпеки даних компанії та користувачів. Але лише одна вразливість у будь-якій бібліотеці може зробити вас вразливими, незважаючи на всі ваші зусилля.

Опитування, яке було проведено виданням zdnet у 2015 році, повідомляє, що 78% підприємств у всьому світі користуються програмним забезпеченням з відкритим програмним кодом (OSS), а 19% використовують OSS опосередковано. Лише 3% підприємств зовсім не використовують таке ПЗ [6]. Більшість розгорнутих веб-систем значною мірою покладаються на компоненти з відкритим кодом і, таким чином, піддаються різним вразливостями.

Веб-розробники іноді винні в тому, що сильно покладаються на бібліотеки з відкритим програмним кодом для всього, від автентифікації до інтерфейсу користувача. Є програми, які повністю побудовані на платформах з відкритим кодом, таких як Magento, Drupal, Wordpress, WooCommerce, Joomla тощо.

Вручну відслідковувати можливі вразливості інформаційної системи майже неможливо. Для цього потрібно перевіряти кожну залежність в надійному джерелі, наприклад у Національній базі вразливостей інформаційних систем Національного інституту стандартів і технологій США. Але якщо на час перевірки залежність немає вразливостей, це не дає гарантій, що вони не будуть знайдені через невеликий проміжок часу. Використана бібліотека також може мати свої залежності, а вони в свою чергу використовувати декілька сторонніх бібліотек з відкритим кодом і так

далі. Багато модулів з відкритим програмним кодом використовують інші OSS для своєї роботи. Наявність однієї вразливості у бібліотеці, робить вразливими усі сервіси та модулі, що використовують її. Це абсолютно унеможливлює гарантію відсутності можливих ризиків безпеки у додатку.

Саме тому протягом багатьох років з’являлися різні інструменти - як безкоштовні, так і комерційні - які допомагають вам керувати вразливостями з відкритим кодом у додатку та його залежностях. Інструмент командного рядка з відкритим кодом від OWASP, який називається OSSIndex. Він може бути використаний для моніторингу коду PHP, Java, .Net та Ruby. Крім того, OSSIndex має кілька модулів перевірки та подає звіти про вразливість із безлічі джерел. Крім цього, існують комерційні організації, такі як SourceClear, Hakiri та Snyk, які полегшують аналіз безпеки програмного забезпечення, мають інтеграції з різними сервісами збереження коду, використовуючи власну базу даних вразливостей, яку наповнюють з кількох джерел.

Авжеж неможливо на сто відсотків гарантувати, що якщо залежності додатку не мають вразливостей, то інформаційна система є неприступною. Але постійний аналіз, оновлення використаних бібліотек, та виправлення проблем безпеки зменшують ризики втрати даних компанії та користувачів і наближають систему до абсолютної безпеки. Але потрібно ще раз зауважити, що такої безпеки неможливо досягти.

Саме тому було проаналізовано декілька існуючих інструментів для сканування додатку та його залежностей щодо можливої загрози безпеки. В результаті було обрано snyk та інтегровано його в існуючу інформаційну систему. Його регулярне використання протягом трьох місяців дозволило поступово зменшити кількість можливих загроз безпеки з більш ніж півтори тисячі до менше ніж п'ятсот у всіх сервісах та додатках.

ВИСНОВКИ

Щороку, все більше людей віддають перевагу купівлі та отриманні послуг онлайн. Ця тенденція змушує компанії швидко змінюватися і переорієнтовуватися. Щоб зрозуміти важливість наявності веб-сайтів та мобільних додатків для сучасного бізнесу, потрібно розглянути основних гравців різних економічних сфер України.

Почнемо з рітейлу, цей сектор значно змінився через популярність онлайн покупок. Онлайн-магазин Розетка став п'ятим найвідуванішим веб-сайтом країни, ним користується майже половина українців за даними агенції Kantar [7]. Компанії, що працюють у сфері торгівлі товарами широкого вжитку представили цього року веб-сайти та мобільні додатки. Серед них такі гіганти як Сільпо, Фора, Varus, АТБ, Ашан, Метро, Fozzy. За допомогою цих застосунків можна замовити доставку продуктів додому, переглянути спеціальні пропозиції магазинів, надати касиру карту лояльності, знайти найближчий супермаркет або навіть оплатити покупки без черги.

Мережі магазинів Eva, Watsons, Prostor, що спеціалізується на косметиці та побутовій хімії, випустили мобільні та веб-застосунки для отримання спеціальних пропозицій, оформлення заказів додому або на пошту, перевірки наявності товару у конкретному магазині. Мережі спортивних магазинів Adidas, New Balance, Спортмастер, Decathlon та найбільша в країні мережа магазинів взуття Intertop представили свої інформаційні системи, що надають змогу зробити замовлення, приміряти, отримати або повернути товар без відвідування фізичного магазину.

Значно змінилася і сфера банківських послуг України monobank - банк без відділень, взаємодія з яким реалізується лише за допомогою мобільного додатку став третім серед усіх банків країні та першим серед комерційних за кількістю випущених карток, попереду лише державні ПриватБанк та Ощадбанк. Це означає, що користувачам набридло ходити до відділень, стояти в чергах і витрачати найцінніший ресурс сьогодення - час. Середній вік користувачів monobank двадцять чотири роки, це дозволяє зробити припущення, що в майбутньому поколінні

отримання банківських послуг віддалено стане стандартом індустрії. Доводить дане припущення й кількість скачувань банківських мобільних застосунків. Додаток Приватбанку має більше десяти мільйонів скачувань на операційній системі Android та більше п'яти мільйонів на операційній системі IOS. Це неймовірна кількість для країни з населенням у трохи менше ніж сорок мільйонів.

COVID-19, карантин та потреба залишатися вдома лише пришвидшили діджиталізацію. Бізнес, який раніше не інвестував у свої інформаційні системи зараз отримує лише втрати і майже немає шансів на подальше існування. З іншої сторони компанії, які перші почали вкладати у цей напрямок, вже встигли відшліфувати свої продукти та набрати базу користувачів. Зараз, це дає змогу їх швидко розвиватися та отримувати нових клієнтів, оскільки навіть люди, які не люблять онлайн покупки змушені їх робити.

Всі ці зміни і тенденції створюють нову проблему для бізнесу - безпека даних користувачів. Сучасні додатки зберігають багато персональної інформації та платіжних даних. Їх втрата може призвести до втрати довіру до продукту з боку користувачів. Зменшити ризики для безпеки можливо постійно аналізуючи інформаційну систему та її залежності за допомогою спеціальних інструментів. Нами було оглянуто найпоширеніші рішення, і обрано snyk. Інтеграція цього інструменту у застосунок дозволила виявити загрози для безпеки даних компанії та користувачів та мінімізувати можливість їх виникнення.

У залученні великою кількістю користувачів є і інша сторона - велике навантаження на сервіси, і не кожна система має змогу його витримати. Це створює потребу у постійному моніторингу продуктивності додатків та покращенню їх слабких місць. Стійкість до навантаження сучасних застосунків залежить не лише від інфраструктури системи, мається на увазі кількості серверів та їх конфігурації, а й від якості програмного коду. Його ефективність та правильність написання впливає на швидкість виконання, що в свою чергу впливає на пропускну спроможність додатку.

Код потребує постійного аналізу та рефакторингу, допомогти в цьому можуть спеціальні інструменти моніторингу інформаційних систем. Нами було оглянуто найпопулярніші рішення і вибрано інструмент NewRelic. Його інтеграція дозволила проаналізувати фрагменти коду, що виконувалися повільно і пришвидшити їх.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Каррыев Б. В. Интернет, краткая история и влияние на общество. — LAP Lambert Academic Publishing, 2015. — 368 с.;
2. Mortgage industry of the united states [Електронний ресурс] // statista – Режим доступу до ресурсу: https://www.statista.com/topics/1685/mortgage-industry-of-the-united-states
3. Student loan debt still impacting millennial homebuyers // Forbes. – 2019. – С. 28 – 31
4. Install new relic ruby agent [Електронний ресурс] // newrelic – Режим доступу до ресурсу: https://docs.newrelic.com/docs/agents/ruby-agent/installation/install-new-relic-ruby-agent.
5. Ruby agent configuration [Електронний ресурс] // newrelic – Режим доступу до ресурсу: https://docs.newrelic.com/docs/agents/ruby-agent/configuration/ruby-agent-configuration.
6. Its an open source world [Електронний ресурс] // zdnet – Режим доступу до ресурсу: https://www.zdnet.com/article/its-an-open-source-world-78-percent-of-companies-run-open-source-software/.
7. Рейтинг популярних сайтів за січень 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://tns-ua.com/news/reyting-populyarnih-saytiv-za-sichen-2020.
8. Скотт В. Эмблер, Прамодкумар Дж. Садаладж. Рефакторинг баз данных: эволюционное проектирование. — М.: «Вильямс», 2007. — 368 с.
9. Джошуа Кериевски. Рефакторинг с использованием шаблонов. — М.: «Вильямс», 2008. — 400 с.