

Національний технічний університет України
«Київський політехнічний інститут ім І.Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Розрахунково-графічна робота
з курсу «Інтеграція програмних систем»

Виконали:
Студенти 4-го курсу
ФІОТ, з груп ІО-34 і ІО-32
Команда(slack) omi-team:
Виноградська Анастасія
Власов Максим
Кривоносов Олексій
Мозговий Іван

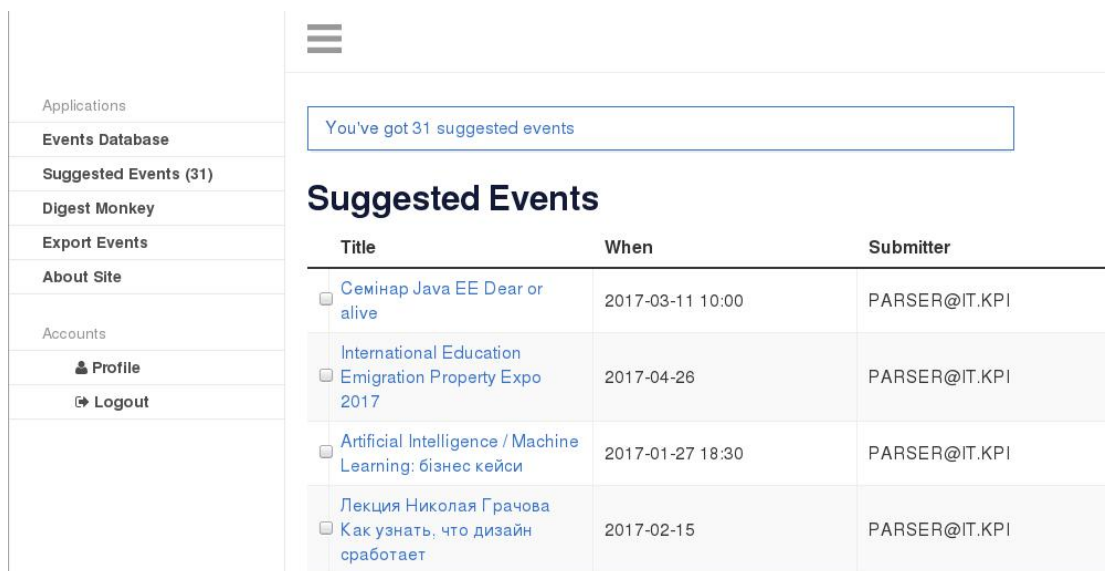
Київ — 2016

1. Опис проекту

В процесі розробки ми долучились до покращення існуючого опенсорц проекту. Даний проект являє собою комплексну систему по агрегації інформації про ІТ-івенти освітньої спрямованості у межах Києва.

В основі роботи лежить event-parser - серверна утиліта, яка дістає данні з зовнішнього світу, проводить базове відсіювання і трансформацію тіла події по заданим патернам.

Оброблені події надсилаються по API у вьюшку, де модератор проводить остаточні модерацію з усіма подіями, що надійшли: Рис. 1.1. і Рис. 1.2.



Suggested Events		
Title	When	Submitter
<input type="checkbox"/> Семінар Java EE Dear or alive	2017-03-11 10:00	PARSER@IT.KPI
<input type="checkbox"/> International Education Emigration Property Expo 2017	2017-04-26	PARSER@IT.KPI
<input type="checkbox"/> Artificial Intelligence / Machine Learning: бізнес кейси	2017-01-27 18:30	PARSER@IT.KPI
<input type="checkbox"/> Лекция Николая Грачова Как узнать, что дизайн работает	2017-02-15	PARSER@IT.KPI

Рис. 1.1. — Вікно перегляду всіх запропонованих подій

Suggested event details

General	Details
<p>TITLE Лекция Николая Грачова Как узнать, что дизайн работает</p> <p>AGENDA Nick Grachov — консультант по лидогенерации, бывший head of product management в Allbiz global. Получил опыт и набил шишки в дизайне и UX в big)mir, korrespondent.net, rabota.ua и многих других проектах на аутсорсе от простых веб-сайтов до сложных систем проведения тендеров и POS-систем.</p> <p>«Можно создавать макеты ради портфолио и красивой картинки. Но для продуктовой компании такой дизайн может не «проканать», потому что он должен быть полезен для продукта и компании. И в такие моменты рождается дизайн-прагматизм. Такой дизайн я успешно «продавал» своим клиентам и такой подход использую до сих пор будучи уже не дизайнером».</p> <p>О чём пойдёт речь:</p> <ul style="list-style-type: none">• что значит дизайн «работает» и как это определить;• 5 секундный тест, «жмак»-тест, опросы, A/B тест, юзабилити тестирование и прочие инструменты;• как замучать своих друзей, родных и близких :) <p>15.02 (среда), начало в 19:30 по адресу ул. Воздвиженская 34а Стоимость — 150грн</p> <p>Регистрация — http://prjctr.com.ua/events/grachov-582ca9d72d062.html</p> <p>До встречи!</p> <p><small>SOCIAL [TECHNICAL INFO. NOT DISPLAY ANYWHERE]</small> <i>From: fb_PROJECTOR ORIGINAL POST </i> Лекция Николая Грачова «Как узнать, что дизайн работает» Nick Grachov — консультант по лидогенерации, бывший head of product management в Allbiz global. Получил опыт и набил шишки в дизайне и UX в big)mir, korrespondent.net, rabota.ua и многих других проектах на аутсорсе от простых веб-сайтов до сложных систем проведения тендеров и POS-систем. «Можно создавать макеты ради портфолио и красивой картинки. Но для продуктовой компании такой дизайн может не «проканать», потому что он должен быть полезен для продукта и компании. И в такие моменты рождается дизайн-прагматизм. Такой дизайн я успешно «продавал» своим клиентам и такой подход использую до сих пор будучи уже не дизайнером». О чём пойдёт речь: — что значит дизайн «работает» и как это определить; — 5 секундный тест, «жмак»-тест, опросы, A/B тест, юзабилити тестирование и прочие инструменты; — как замучать своих друзей, родных и близких :) 15.02 (среда), начало в 19:30 по адресу ул. Воздвиженская 34а Стоимость — 150грн Регистрация — http://prjctr.com.ua/events/grachov-582ca9d72d062.html До встречи!</p>	<p>PLACE ул. Воздвиженская, 34а</p> <p>STARTS AT ENDS AT 2017-02-15 2017-02-15</p> <p>SPECIAL <input type="checkbox"/></p> <p>LEVEL NONE</p> <p>REGISTRATION URL https://fb.com/337393516626087</p> <p>IMAGE URL <input type="checkbox"/></p> <p>SUBMITTED BY PARSER@IT.KPI</p>

Accept

Decline

Рис. 1.2. — Вікно перегляду запропонованої події

Персер розроблений за допомогою NodeJS, view — Django. Данні з парсера передаються у форматі JSON і зберігаються у базу даних.

2. Система автоматичної збірки. Vagga

Vagga — інструмент для створення середовища розробки.

Vagga здатна:

- Побудувати контейнер і запустити програму однією командою, відразу після `git pull`
- Автоматичне перебілдження контейнера, якщо залежності проекту зазнали змін
- Запускати декілька процесів (наприклад, програми та бази даних) однією командою
- Виконувати тестування толерантності мережі

Створений Павлом Коломійцем у 2015 році для потреб Prom.ua (нині EVO Company).

Не рекомендується для продакшену, проте у випадку роботи парсеру, можливостей Vagga і cron-tab цілком достатньо.

3. Сервер безперервної інтеграції. Travis CI

Travis CI - розподілений веб-сервіс для побудови та тестування програмного забезпечення, що використовує GitHub як хостинг вихідного коду.

Веб-сервіс підтримує збірку проектів на багатьох мовах, включно з C, C++, D, JavaScript, Java, PHP, Python і Ruby.

На Travis виконуються наступні задачі:

- Білдінг контейнера Vagga і запуск тестів на ньому
- Білдінг контейнера Dosluk і запуск тестів на ньому
- При успішному проходженні тестів, відбувається автоматичне розгортання на продакшн

Існуює 4 види тестів, які виконуються на сервері CI:

- Статичний аналіз коду
- Звичайні (перевірка кусків коду без залежностей)
- З використанням моків (Заміна результатів тестів на відомий результат)
- Інтеграційні тести (тестування частини бо всього застосунку й відбувається тестування наближене до реальних умов роботи)

4. Експоненційна витримка

Усі сервіси раняться на одному сервері і розносити їх на різні інстанси немає жодного сенсу, враховуючи співвідношення кількості подій, що може обробити парсер і реальної їх кількості.

Виходячи з цього, тестування експоненційної витримки API не є необхідною, тому варто звернутись до тієї частини, що може її потребувати: GET запити з парсера у першоджерела.

В разі `Status != 200 OK`, джерело ігнорується до наступного запуску, а інформація про недоступність джерела логується.

Наразі парсер запускається кожні 30хв, і насправді це рази в 3 частіше, ніж того потребує швидкість додавання нових подій. Так як інтервали в $>5\text{хв}$ не потребують експоненційної витримки, то і не варто її реалізовувати. Це вказано на Рис. 4.1.

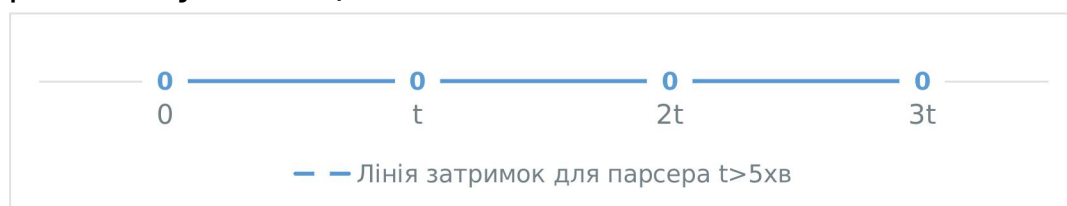


Рис. 4.1. Лінія затримок для парсера $t > 5\text{хв}$

В разі, якби ж таки довелось реалізовувати потокове отримання даних, то для цього було б використано наступний код і графік би змінився на той, що зображено на Рис. 4.2.

```
let timeout = 0
while (true) {
  setTimeout(function() {
    /* our GET-request realization */
    if (err && timeout < 300000) timeout = 1000 + timeout * 2
    else timeout = 0
  }, timeout)
}
```

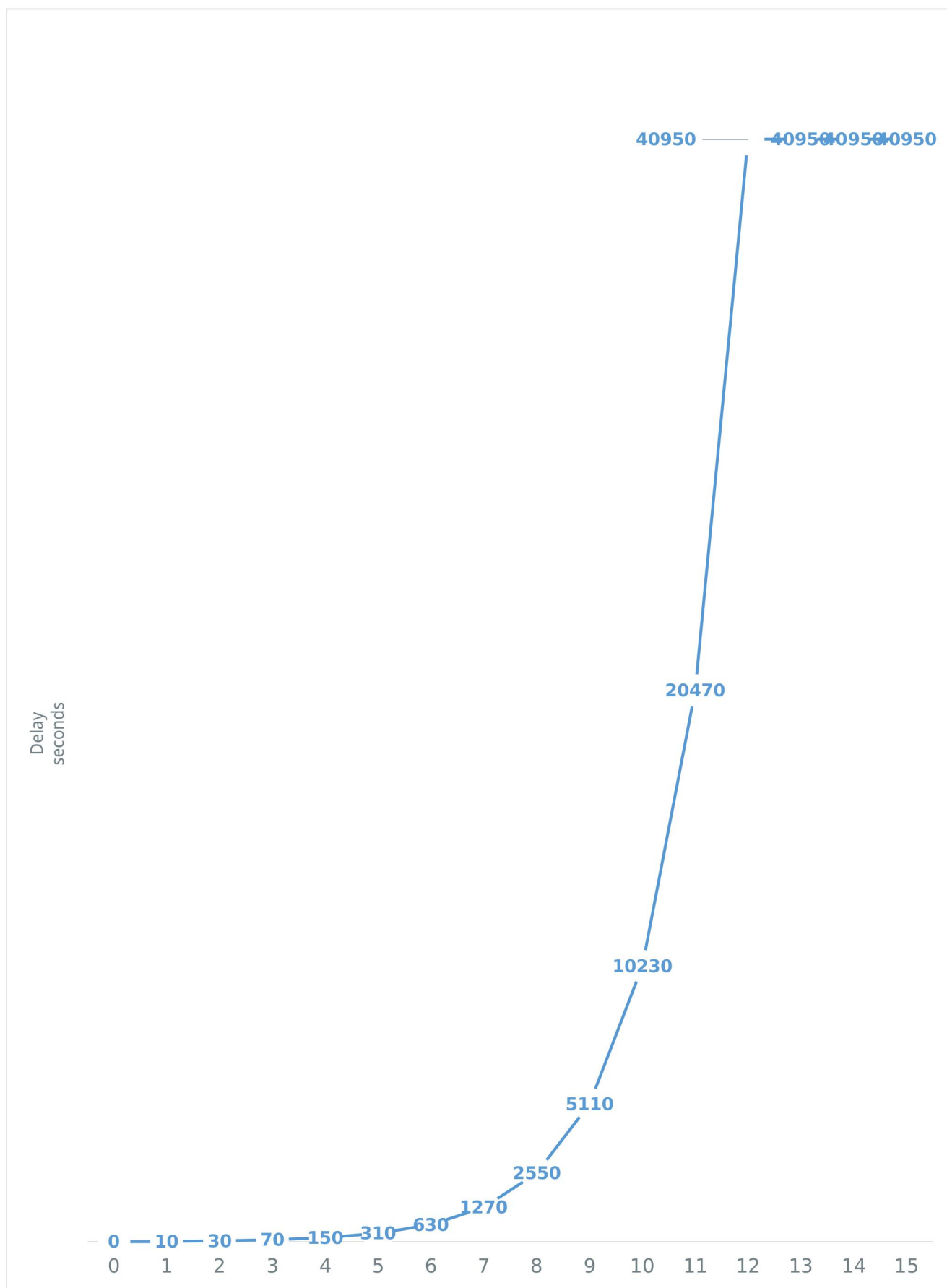


Рис. 4.2. Лінія затримок для парсера $t \leq 5 \text{хв}$