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»

на тему: « Система прогнозування тенісних матчів»

Виконали:
студенти 4-го курсу
ФІОТ групи ІО-33
Денік Юра
Марченко Олексій
Миронюк Дмитро
Ханевич Андрій

Київ 2017

1. Опис проекту

Розроблений додаток дозволяє вибрати 2 гравців і умови в яких вони будуть грати(зараз тільки тип корту) і отримати ймовірність перемоги першого гравця над другим. Ми використовуємо натреновану на реальних даних модель на основі регресії для прогнозування. Веб додаток для отримання результатів прогнозування запускає пітон скрипт. Скріпт десеріалізує натреновану модель передає їх вхідні дані і отримує результат і пише його у стандартний вивід. Далі веб додаток читає з процесу в якому запущений пітон скрипт результаті віддає його клієнту. Також арі дозволяє отримувати гравців що були збережені у базу даних для виводу на клієнті.

2. Система автоматичної збірки. Maven

Maven - це засіб автоматизації роботи з програмними проектами, який спочатку використовувався для Java проектів. Використовується для управління (management) та складання (build) програм. Створений Джейсоном ван Зилом у 2002 році. XML-файл описує проект, його зв'язки з зовнішніми модулями і компонентами, порядок будування (build), папки та необхідні плагіни. Сервер із додатковими модулями та додатковими бібліотеками розміщується на серверах.

Для опису програмного проекту який потрібно побудувати (build), Maven використовує конструкцію відому як Project Object Model (POM), залежності від зовнішніх модулів, компонентів та порядку побудови. Виконання певних, чітко визначених задач - таких, як компіляція коду та пакетування відбувається шляхом досягнення заздалегідь визначених цілей (targets).

Двигун ядра може динамічно завантажувати плагіни з репозиторію, того самого репозиторію, що забезпечує доступ до багатьох версій різних Java-проектів з відкритим кодом. Maven забезпечує підтримку побудови не просто перебираючи файли з цього репозиторію, але й завантажуючи назад артефакти у кінці побудови. Локальний кеш звантажених артефактів діє як первісний засіб синхронізації виходу проектів на локальній системі.

Ключовою особливістю Maven є його мережева готовність (network-ready).

3. Сервер безперервної інтеграції. Travis-ci

Термін «continuous integration» досить старий. Він був введений Мартіном Фаулером у 2000-му році і викладений у статті «Continuous Integration». Українською цей термін можна перекласти як «безперервна інтеграція». Це частина процесу розробки, в якій проект автоматично і безперервно

збирається/тестується в різних середовищах виконання. Задумувалася дана методика для найбільш швидкого виявлення помилок (протиріч) інтеграції проекту.

Принцип досить простий: на окремій машині працює якась служба, в обов'язки якої входить отримання вихідного коду проекту, його збірка, тестування, логування, а також можливість надати для аналізу дані виконання перерахованих операцій.

Виділення окремого серверу і підтримування його в робочому стані, забезпечення наявності необхідних програмних комплексів, налаштування середовища виконання, створення резервних копій даних і т.д - Все це вимагає чимало часу і ресурсів. І цілком логічним здається можливість делегувати цю відповідальність на сторонні сервіси. От якраз таким і є travis-ci - «хостинг безперервної інтеграції для open source співтовариства».

Travis-ci підтримує безліч мов програмування. Для користування сервісом потрібно настроїти спеціальний файл конфігурації .travis.yml.

Задачі, які вирішуються на сервері безперервної інтеграції:

1. Запуск тестів:

    a) Модульні тести

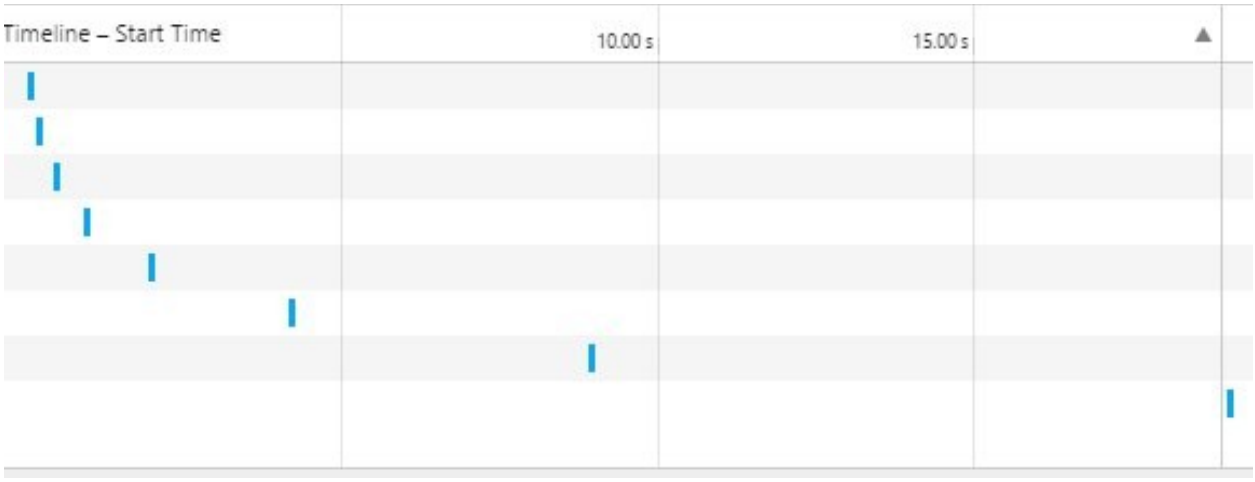
    b) Інтеграційні тести

4. Експоненціальна витримка

Експоненціальна витримка реалізована і використовується у ситуації, коли веб клієнт не може підключитись до серверу. Реалізація на javascript

```
$.ajax({
    url: options.url,
    type: 'POST',
    data: options.data,
    error: function (xhr, status, err) {
        console.log(err.message);
        setTimeout(function () {
            var maxDelay = 15 * 60 * 1000, factor = 2, jitter = 0.1;
            delay = Math.min(delay * factor, maxDelay);
            delay = delay + Math.random() * delay * jitter;
            makePostReq(delay, options);
        }, delay);
    },
    success: function (result, status, xhr) {
        callback(result);
    }
})
```

метод викликається приблизно через 100 мс після невдалої спроби. «Експотенціальність» досягається за рахукнок збільшення інтервалу між запитами по експотенційному закону.



Спостерігається експотенціальне збільшення інтервалу спроби.