|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| ĐỒ ÁN TỐT NGHIỆP  **XÂY DỰNG VÀ PHÁT TRIỂN ỨNG DỤNG PHÂN TÍCH FILE LOG CỦA DJANGO** |
| Ngành: An toàn thông tin  Mã số: 7.48.02.02  *Sinh viên thực hiện*:  **Đinh Tấn Thiện**  Lớp: AT11E  *Người hướng dẫn*:  **ThS. Nguyễn Quốc Sử**  Khoa khoa học máy tính – Đại học Văn Lang |
| Hồ Chí Minh, 2019 |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| ĐỒ ÁN TỐT NGHIỆP  **XÂY DỰNG VÀ PHÁT TRIỂN ỨNG DỤNG PHÂN TÍCH FILE LOG CỦA DJANGO** |
| Ngành: An toàn thông tin  Mã số: 7.48.02.02  *Sinh viên thực hiện*:  **Đinh Tấn Thiện**  Lớp: AT11E  *Người hướng dẫn*:  **ThS. Nguyễn Quốc Sử**  Khoa khoa học máy tính – Đại học Văn Lang |
| Hồ Chí Minh, 2019 |

# **MỤC LỤC**

[**MỤC LỤC** 2](#_Toc11146358)

[**DANH MỤC KÝ HIỆU VÀ VIẾT TẮT** 4](#_Toc11146359)

[**DANH MỤC BẢNG BIỂU** 4](#_Toc11146360)

[**DANH MỤC HÌNH VẼ** 4](#_Toc11146361)

[**LỜI CẢM ƠN** 4](#_Toc11146362)

[**LỜI NÓI ĐẦU** 4](#_Toc11146363)

[**PHẦN I: ĐẶT VẤN ĐỀ** 5](#_Toc11146364)

[**1.** **Lý do chọn đề tài** 5](#_Toc11146365)

[**2.** **Mục tiêu nhiệm vụ đề tài** 6](#_Toc11146366)

[**3.** **Đối tượng và phạm vi nghiên cứu** 6](#_Toc11146367)

[**4.** **Phương pháp nghiên cứu** 7](#_Toc11146368)

[**5.** **Kết luận phần I** 7](#_Toc11146369)

[**PHẦN II: NỘI DUNG NGHIÊN CỨU** 7](#_Toc11146370)

[**CHƯƠNG 1: CƠ SỞ KHOA HỌC CỦA VẤN ĐỀ NGHIÊN CỨU** 7](#_Toc11146371)

[**1.1.** **Tổng quan về log** 7](#_Toc11146372)

[*1.1.1.* *Giới thiệu log* 7](#_Toc11146373)

[*1.1.2.* *Một số kiểu log* 8](#_Toc11146374)

[*1.1.3.* *Cơ chế ghi log* 9](#_Toc11146375)

[*1.1.4.* *Thu thập log* 10](#_Toc11146376)

[*1.1.5.* *Phân tích log* 13](#_Toc11146377)

[**1.2.** **Tổng quan về log DJANGO** 15](#_Toc11146378)

[*1.2.1.* *Giới thiệu* 15](#_Toc11146379)

[*1.2.2.* *Đặc điểm* 16](#_Toc11146380)

[*1.2.3.* *Cơ chế thu thập log* 16](#_Toc11146381)

[**1.3.** **Kỹ thuật và công nghệ sử dụng trong ứng dụng** 18](#_Toc11146382)

[*1.3.1.* *Thuật toán Prefixspan (thuật toán tính tần suất mẫu)* 18](#_Toc11146383)

[*1.1.1.* *Regular Expression* 20](#_Toc11146384)

[*1.1.2.* *Module logging* 23](#_Toc11146385)

[*1.1.3.* *Matplotlib* 23](#_Toc11146386)

[**1.2.** **Kết luận chương 1** 25](#_Toc11146387)

[**CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ ỨNG DỤNG** 25](#_Toc11146388)

[**2.1.** **Mô hình ứng dụng** 25](#_Toc11146389)

[**2.2.** **Thiết kế dữ liệu** 27](#_Toc11146390)

[**2.3.** **Thiết kế chức năng** 27](#_Toc11146391)

[**2.4.** **Thiết kế giao diện** 30](#_Toc11146392)

[**2.5.** **Kết luận chương 2** 32](#_Toc11146393)

[**CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG** 32](#_Toc11146394)

[**3.1.** **Yêu cầu đối với ứng dụng** 32](#_Toc11146395)

[**3.2.** **Xây dựng các chức năng** 33](#_Toc11146396)

[*3.2.1. Ghi log (logging)* 33](#_Toc11146397)

[*3.2.2. Giám sát log thời gian thực (real-time log monitoring)* 33](#_Toc11146398)

[*3.2.3. Thống kê log (statistic log)* 33](#_Toc11146399)

[*3.2.4. Xác địn hành vi người dùng (determine user behavior)* 33](#_Toc11146400)

[**3.3.** **Xây dựng giao diện** 33](#_Toc11146401)

[**3.4.** **Kết luận chương 3** 39](#_Toc11146402)

[**PHẦN III: TRIỂN KHAI ỨNG DỤNG** 39](#_Toc11146403)

[**1.** **Môi trường thử nghiệm** 39](#_Toc11146404)

[*1.1.* *Cài đặt và cấu hình* 39](#_Toc11146405)

[*1.2.* *Chạy thử nghiệm* 41](#_Toc11146406)

[**2.** **Kết quả thử nghiệm** 42](#_Toc11146407)

[**3.** **Đánh giá kết quả thử nghiệm** 43](#_Toc11146408)

[**4.** **Phương hướng phát triển** 43](#_Toc11146409)

[**5.** **Kết luận phần III** 43](#_Toc11146410)

[**KẾT LUẬN** 44](#_Toc11146411)

[**TÀI LIỆU THAM KHẢO** 44](#_Toc11146412)

[**PHỤ LỤC 1** 44](#_Toc11146413)

[**PHỤ LỤC 2** 48](#_Toc11146414)

[**PHỤ LỤC 3** 53](#_Toc11146415)

# **DANH MỤC KÝ HIỆU VÀ VIẾT TẮT**

|  |  |
| --- | --- |
| || | Phép nối hai chuỗi bít (hoặc chuỗi octet) |
| ||X|| | Độ dài của chuỗi bít (hoặc chuỗi octet) X |
| ⊕ | Phép cộng từng bít theo mô-đun 2 |
|  | Số nguyên nhỏ nhất không nhỏ hơn |
|  | Số nguyên lớn nhất không lớn hơn |
| DB | Data Block |
| EM | Encoded Message |
| EME | Encoding Method for Encryption |
| I2OSP | Integer To Octet String Primitive |
| MGF | Mask Generation Function |
| OAEP | Optimal Asymmetric Encryption Padding |
| PKCS | Public-Key Cryptography Standard |
| PS | Padding String |
| RFC | Request for Comments |
| RSAES | RSA Encryption Scheme |

# **DANH MỤC BẢNG BIỂU**

[Bảng 1.1. Mô tả các level trong log 14](#_Toc11485499)

[Bảng 1.2. Các định dạng được cung cấp trong django 15](#_Toc11485500)

[Bảng 1. 3. Prefix, postfix và các mẫu tuần tự tương ứng 22](#_Toc11485501)

[Bảng 1.4. Minh họa các vòng lặp của thuật toán prefixspan 23](#_Toc11485502)

[Bảng 1.5. Mô tả đầy đủ các ký tự đặt biệt trong RE 25](#_Toc11485503)

[Bảng 1.6. Những phương thức được sử dụng trong RE 26](#_Toc11485504)

[Bảng 1.7. Mô tả phương thức trong biểu đồ 2D 27](#_Toc11485505)

[Bảng 2.1. Minh hoa giao diện liệt kê log trên web 35](#_Toc11485507)

# **DANH MỤC HÌNH VẼ**

[Hình 1.1. File log của hệ điều hành linux. 13](#_Toc11453511)

[Hình 1.2. File log của window 14](#_Toc11453512)

[Hình 1.3. Cơ chế xử lý log trong Django 18](#_Toc11453513)

[Hình 1.4. Minh họa sử dụng biểu thức chính quy 22](#_Toc11453514)

[Hình 1.5. Đồ thị được vẽ bởi matplotlib 25](#_Toc11453515)

[Hình 2.1. Mô hình triển khai ứng dụng 27](#_Toc11453581)

[Hình 2.2. Mô hình hoạt động ứng dụng 28](file:///C:\Users\HT\Downloads\Đinh%20Tan%20Thien-DA.docx#_Toc11453582)

[Hình 2.3. Mô hình lớp dữ liệu của ứng dụng 29](file:///C:\Users\HT\Downloads\Đinh%20Tan%20Thien-DA.docx#_Toc11453583)

[Hình 2.4. Dữ liệu khi trích xuất từ file log 30](#_Toc11453584)

[Hình 2.5. Mô hình mô tả các chức năng của ứng dụng 31](#_Toc11453585)

[Hình 2.6. Mô hình trình tự xử lý log trong ứng dụng 31](#_Toc11453586)

[Hình 2.7. Realtime log model 32](#_Toc11453587)

[Hình 2.8. Giao diện Graylog 33](#_Toc11453588)

[Hình 2.9. Giao diện thống kê log của Graylog 33](#_Toc11453589)

[Hình 2.10. Một số dạng biểu đồ hay dùng biểu diễn log 34](#_Toc11453590)

[Hình 3.1. Mô tả quá trình ghi và lưu log 36](#_Toc11453603)

[Hình 3.2. Log khi được lưu dưới database 38](#_Toc11453604)

[Hình 3.3. Sơ đồ minh họa quá trình đọc real-time log 40](#_Toc11453605)

[Hình 3.4. Các chức năng phân tích được từ log 42](#_Toc11453606)

[Hình 3.5. Kết quả thu thập mẫu hành vi đăng nhập 49](#_Toc11453607)

[Hình 3.6. Sơ đồ hoạt động django 51](#_Toc11453608)

[Hình 3.7. Bảng hiển thị danh sách log 51](#_Toc11453609)

[Hình 4.1. Các file có trong ứng dụng 55](#_Toc11453629)

[Hình 4.2. Các file cơ bản của một website dùng django 56](#_Toc11453630)

[Hình 4.3. Cấu hình log trong setting.py(1) 56](#_Toc11453631)

[Hình 4.4. Cấu hình log trong setting.py(2) 57](#_Toc11453632)

[Hình 4.5. Cấu hình log trong setting.py(3) 58](#_Toc11453633)

[Hình 4 6. Thông báo cấu hình thành công 59](#_Toc11453634)

[Hình 4.7. Giao diện đăng nhập ứng dụng 59](#_Toc11453635)

[Hình 4.8. Giao diện khi đăng nhập thành công 60](#_Toc11453636)

[Hình 4.9. Tổng hợp các log được lưu 60](#_Toc11453637)

[Hình 4.10. Thông kê log traffic graph 61](#_Toc11453638)

[Hình 4.11. Thông kê log traffic information 61](#_Toc11453639)

[Hình 4. 12. Thống kê log lastest requests 62](#_Toc11453640)

[Hình 4.13. Thống kê log top paths 62](#_Toc11453641)

[Hình 4.14. Thống kê log top error paths 63](#_Toc11453642)

[Hình 4.15. Thông kê log top browsers 63](#_Toc11453643)

[Hình 4.16. Realtime log 64](#_Toc11453644)

[Hình 4.17. Thư mục chứa kết quả hành vi người dùng 64](#_Toc11453645)

[Hình 4.18. Kết quả thu thập mẫu đăng nhập 64](#_Toc11453646)

[Hình 4. 19. Kết quả thu thập hành vi người dùng 65](#_Toc11453647)

# **LỜI CẢM ƠN**

Trong quá trình thực hiện đồ án tốt nghiệp này, tôi đã nhận được sự giúp đỡ tận tình của cán bộ hướng dẫn là TS. Nguyễn Quốc Sử – Giảng viên Khoa Đại học Văn Lang, sự quan tâm sâu sát của cán bộ Hệ quản lý học viện kỹ thuật Mật Mã cơ sở phía nam, sự động viên của người thân và bạn bè.

Xin cảm ơn tất cả mọi người đã tạo những điều kiện tốt nhất để tôi hoàn thành đồ án tốt nghiệp này!

|  |  |
| --- | --- |
|  | **SINH VIÊN THỰC HIỆN ĐỒ ÁN**  Đinh Tấn Thiện |

# **LỜI NÓI ĐẦU**

Ngày nay, chúng ta có thể tìm thấy rất nhiều công cụ quản lý và phân tích nhật ký hay gọi là log trên thị trường. Mặt khác, phân tích nhật ký là việc cơ bản trong bảo mật. Trong thực tế, phân tích nhật ký đã cho phép phát triển các công cụ để quản lý sự kiện và thông tin bảo mật SIEM.

Với sự phổ biến ngày càng nhiều các framework phát triển website, bên cạnh đó là việc xuất hiện đi cùng của các ứng dụng, công cụ giám sát và phân tích log. Sau các năm học tại trường với ngành an toàn thông tin, em nhận ra rằng việc phát triển một ứng dụng phân tích file log phù hợp với mục đích cá nhân của từng người sử dụng là rất cần thiết

Đồ án gồm 3 phần lớn :

**PHẦN I. ĐẶT VẤN ĐỀ :** phần này giới thiệu về lý do chọn đề tài cũng như các mục tiêu và nhiệm vụ mà đồ án hướng tới

**PHẦN II. NỘI DUNG NGHIÊN CỨU:** Mô tả các khái niệm thuật toán, công nghệ được sử dụng để thiết kế và xây dựng ứng dụng. Phần này được chia làm 3 chương:

Chương 1: Cơ sở khoa học của vấn đề nghiên cứu

Chương 2: Phân tích thiết kế ứng dụng

Chương 3: Xây dựng ứng dụng

**PHẦN III. TRIỂN KHAI ỨNG DỤNG:**

Trong quá trình tìm hiểu, do hiểu biết còn hạn chế, kiến thức thực tế chưa có nhiều, nên em không tránh khỏi những thiếu sót sai lầm. Rất mong được sự góp ý của thầy cô và các bạn. Em xin cám ơn thầy **ThS. Nguyễn Quốc Sử** đã nhiệt tình hướng dẫn chúng em hoàn thành bài báo cáo này.

Em xin chân thành cảm ơn!

# **PHẦN I: ĐẶT VẤN ĐỀ**

1. **Lý do chọn đề tài**

Ngày nay, lập trình web có rất nhiều ngôn ngữ được sử dụng, trong đó có python đang được sử dụng ngày càng phổ biến. Framework Django là một framework phổ biến nhất của python để xây dựng hệ thống website.

Khi công nghệ lập trình web bằng Django phát triển càng nhanh, càng thuận tiện thì đồng thời cũng nảy sinh những mối nguy hại về bảo mật ngày càng lớn. Với những hệ thống lớn như ngân hàng, thương mại điện tử, quản lý người dùng thì sự an toàn của ứng dụng web là lựa chọn hàng đầu.

Một trong các phương áp thường được sử dụng để giám sát, quản lý và bảo vệ các ứng dụng web là sử dụng các bản ghi log được tạo ra bởi chính hệ thống.

Vần đề được đặt ra là với các file log chứa quá nhiều thông tin làm cho việc đọc và phân tích các dữ liệu được ghi log trở nên khó khăn. Để có thể phát hiện cũng như sớm đưa ra các biện pháp đảm bảo an toàn, chúng ta phải tiến hành kiếm tra file log của hệ thống để đánh giá hệ thống.

Trước khi lựa chọn đề tài, em đã tìm hiểu và có cơ hội được sử dụng các ứng dụng phân tích log được đánh giá cao là SPLUNK và Graylog. Các ứng dụng này đều đòi hỏi tài nguyên cung cấp để cài đặt và hoạt động tương đối cao, bên cạnh đó là chúng có quá nhiều các chức năng gây khó khăn cho người mới tiếp cận và những người cần sự nhạnh gọn, dễ dàng cài đặt. Vì thế em đã tìm tòi các ứng dụng phân tích log gọn nhẹ, dễ dàng cấu hình lại theo mục đích cá nhân và tập trung vào một nền tảng web như django nhưng các ứng dụng phân tích log lại không đáp ứng được. Từ đó, em quyết định lựa chọn đề tài này.

Vì vậy, đồ án “**xây dựng và phát triển ứng dụng phân tích file log của Django**” nhằm cung cấp cho người quản trị cũng như nhà phát triển có thể kiểm tra, đánh giá và sớm đưa ra cảnh báo cho hệ thống khi gặp sự cố.

1. **Mục tiêu nhiệm vụ đề tài**

Log file chỉ ghi thông tin và các sự kiện trên cùng một hệ thống cụ thể, vì vậy chúng rất hữu ích để xác định nguyên nhân gây ra sự cố trong hệ thống nhưng lại không có tiêu chuẩn nào về vị trí, cách sử dụng, định dạng, và kích thước của tệp nhật ký phải được đáp ứng cho các hệ thống và ứng dụng khác nhau, điều này làm cho việc phân tích phức tạp hơn và hạn chế trích xuất các thông tin có ích tối đa từ file log.

Hiện nay, các ứng dụng phân tích log mọi người sử dụng đều được tách riêng biệt với ứng dụng web dẫn tới cần một lượng tài nguyên máy tính lớn để cài đặt và sử dụng, đồng thời không dễ để cẩu hình lại theo ý muốn của người dùng để thuận tiện cho việc phân tích.

Bên cạnh đó, việc ngày càng nhiều hệ thống website sử dụng framework django của python cũng tạo nên nhu cầu về các ứng dụng giúp đọc và phân tích các file log được ghi lại bởi django.

Khi tìm hiểu vấn đề này, một câu hỏi được đặt ra là tại sao ta không sử dụng một ứng dụng mà có thể tích hợp vào ứng dụng web đang dùng, đồng thời có thể tùy chỉnh theo mong muốn và kinh nghiệm phân tích log của bản thân. Vì vậy đồ án này hướng tới các mục tiêu như sau:

* Tìm hiểu, nghiên cứu các công nghệ sử dụng để phát triển ứng dụng web django, phương pháp để tích hợp một ứng dụng dựa vào django.
* Tìm hiểu, nghiên cứu về log, cách ghi log, đặt biệt là log trong django.
* Dựa vào kiến thức tìm hiểu để tiến hành thiết kế dữ liệu, các chức năng và giao diện cho ứng dụng.
* Tiến hành xây dựng ứng dụng đảm bảo hoạt động như đã thiết kế.
* Dựa vào kinh nghiệm trong quá trình làm đồ án và kinh nghiệm bản thân để hoàn thiện hơn ứng dụng.
* Nghiên cứu ứng dụng học máy để nhận diện các cuộc tấn công vào ứng dụng dựa trên các log được ghi.

1. **Đối tượng và phạm vi nghiên cứu**

* Đối tượng nghiên cứu: Là các thông tin, dữ liệu trong file log được tạo ra bởi Django
* Phạm vi nghiên cứu: nghiên cứu xây dựng được ứng dụng trích lọc được các thông tin cơ bản từ file log như lượt truy cập, liệt kê địa chỉ truy cập, thời gian truy cập. Hướng tới nghiên cứu ứng dụng học máy vào để phân tích hành vi người dùng

1. **Phương pháp nghiên cứu**

* Nghiên cứu lý thuyết: tìm hiểu các khái niệm, các thuật toán giúp cho việc thu thập và phân tích log
* Nghiên cứu thực nghiệm: đồ án thực hiện thiết kế và xây dựng ứng dụng phân tích log của django, chạy thử nghiệm và đánh giá kết quả.

1. **Kết luận phần I**

Log là một nguồn thông tin hữu ích, vì nó chứa các bản ghi về mọi hành động được thực hiện trên hệ thống. Trong thực tế, khi được khai thác đúng cách, các log có thể cung cấp những thông tin chi tiết về hiệu suất, cách sử dụng và quản lý hệ thống. Ngoài ra, phân tích này chắc chắn sẽ giúp đưa ra quyết định đúng đắn trong các lĩnh vực quan trọng, ví dụ như bảo mật.

Để tránh những vấn đề này, một phần mềm phân tích event log chỉ tập chung cụ thể vào một hệ thống cụ thể là vô cùng cần thiết, ở đây là các hệ thống website được xây dựng dựa trên framework django. Những công cụ chuyên dụng này thu thập thông tin từ các nguồn và phân tích chúng để đưa ra thông tin chi tiết, có ý nghĩa và dữ liệu hành động. Thêm vào đó, nó giúp cho các quản trị viên CNTT hoạt động hiệu quả hơn, có thể tập trung vào dữ liệu đầu ra thay vì phải chọn lọc giữa rất nhiều dữ liệu log thô và không thể đọc được.

# **PHẦN II: NỘI DUNG NGHIÊN CỨU**

# **CHƯƠNG 1: CƠ SỞ KHOA HỌC CỦA VẤN ĐỀ NGHIÊN CỨU**

* 1. **Tổng quan về log**
     1. *Giới thiệu log*

**Event**

Event là các sự kiện xảy trong hệ thống hay ứng dụng khi chúng đang hoạt động như đăng nhập, đăng kí tài khoản, truy cập vào chức năng nào đó,…

**Log File**

File log là một tập tin văn bản hoặc tệp XML được tạo ra bởi máy chủ hay ứng dụng web có chứa tất cả thông tin về hoạt động trên máy chủ đó, như thông tin người truy cập, thời gian khách hàng viếng thăm, địa chỉ IP, dữ liệu truy vấn,… . File log có rất nhiều tác dụng đối với giám sát hoặc quản trị như phân tích xem người truy cập vào những phần nào trên trang web nhiều nhất,…

Log bao gồm bản ghi hệ thống, các bản ghi ứng dụng, và bản ghi bảo mật. Hệ thống vận hành và bảo trì, và các nhà phát triển có thể đăng nhập cho phần cứng máy chủ và phần mềm thông tin, những lý do cho việc kiểm tra lỗi cấu hình và các lỗi xảy ra. Phân tích các log thường có thể hiểu được tại máy chủ, hiệu suất, bảo mật để có biên pháp khắc phục kịp thời .

Ví dụ:

* Log file của các thiết bị firewall, rouer, moderm,…
* Log file của hệ điều hành windown, linux,…
* Log file của ứng dụng web, mail, database,..
* Log file của các phần mềm,..

Như đã nói ở trên, không có tiêu chuẩn nào cho log. Tuy nhiên, nói chung một file log của bất kì hệ thống, ứng dụng nào phải bao gồm:

* Time-stamp: thông tin về thời gian sự kiện đã xảy ra; ngày, giờ, phút và giây.
* Category: bất kỳ tệp nhật ký nào cũng bao gồm một số loại phân loại sự kiện, cho biết tầm quan trọng hoặc tác động đối với hệ thống.
* Desciption: Ở đây chúng tôi có thể tìm thấy thông tin về sự kiện hoặc điều kiện cụ thể.

**Logging**

Logging là hành động lưu giữ một tập tin. Trong trường hợp đơn giản, thông điệp được ghi vào một tập tin duy nhất và trình tự theo thời gian. Những hành động làm sai lệch tập tin nhật kí phải được ngăn chặn ở mức tối đa.

Logging là một phương tiện theo dõi các sự kiện xảy ra khi hệ thống hay phần mềm hoạt động. Một sự kiện có thể được mô tả một cách tùy ý trong log ( nghĩa là dữ liệu có khả năng khác nhau cho mỗi lần xuất hiện của sự kiện). Các sự kiện xảy ra trong hệ thống đóng một vai trò quan trọng với các mức độ khác nhau.

* + 1. *Một số kiểu log*

Log hệ thống (System Log -Syslog): chứa các bản ghi các sự kiện của hệ điều hành (OS) cho biết cách các hệ thống xử lý và sử dụng tài nguyên. Syslog hiển thị các thông tin, lỗi và cảnh báo liên quan đến hệ điều hành máy tính. Bằng cách xem xét dữ liệu trong log, người quản trị hay người dùng có thể xác định được nguyên nhân xảy ra sự cố hệ thống hoặc các tiến trình nào vận hành thành công.

Log ứng dụng (application): là một file các sự kiện được ghi lại bởi một ứng dụng phần mềm. Nó chứa lỗi, sự kiện, thông tin và cảnh báo. Định dạng và nội dung của log application được xác định bởi nhà phát triển phần mềm hoặc người dùng thay vì bởi hệ điều hành.

Log lỗi (Error): trong khoa học máy tính, nhật ký lỗi là bản ghi các lỗi nghiêm trọng mà ứng dụng, hệ điều hành hoặc máy chủ gặp phải khi hoạt động. Khác với các lỗi được ghi lại trong log system hay application, log error ghi lại tường tận lỗi được phát sinh ra như khi lỗi của một ứng dụng hay xung đột giữ các ứng dụng dẫn tới kéo theo lỗi hệ điều hành. Log error đóng vai trò quan trọng là công cự hữu ích để khắc phụ sự cố và quản trị hệ thống, máy chủ và thậm chí cả mạng.

Log trên web server: nhật ký truy cập có thể hữu ích để xác định số lượng khách truy cập, tên miền mà họ đang truy cập, số lượng yêu cầu cho mỗi trang, mẫu sử dụng theo ngày trong tuần hoặc thậm chí là giờ trong ngày.

* + 1. *Cơ chế ghi log*

**Cơ chế độc lập**

* Các ứng dụng tự ghi nhất ký vào các thư mục riêng rẽ.
* Khó theo dõi nhật ký.
* Nhật ký nhân hệ điều hành không phải ứng dụng.
* Các ứng dụng khó sử dụng nhật ký của nhau.
* Khó phát hiện ứng dụng “có vấn đề”

**Cơ chế tập trung**

* Tác dụng của log là vô cùng to lớn vậy làm thế nào để quản lý log tốt hơn?

Để quản lý log một cách tốt hơn, xu thế hiện nay sẽ sử dụng **log tập trun**g. Vậy log tập trung là gì? Tác dụng của nó thế nào?

Hiểu một cách đơn giản : Log tâp trung là quá trình tập trung, thu thập, phân tích... các log cần thiết từ nhiều nguồn khác nhau về một nơi an toàn để thuận lợi cho việc phân tích, theo dõi hệ thống.

* Tại sao lại phải sử dụng log tập trung?
* Do có nhiều nguồn sinh log
  + Có nhiều nguồn sinh ra log, log nằm trên nhiều máy chủ khác nhau nên khó quản lý.
  + Nội dung log không đồng nhất (Giả sử log từ nguồn 1 có có ghi thông tin về ip mà không ghi thông tin về user name đăng nhập mà log từ nguồn 2 lại có) -> khó khăn trong việc kết hợp các log với nhau để xử lý vấn đề gặp phải.
  + Định dạng log cũng không đồng nhất -> khó khăn trong việc chuẩn hóa
* Đảm bảo tính toàn vẹn, bí mật, sẵn sàng của log.
  + Do có nhiều các rootkit được thiết kế để xóa bỏ logs.
  + Do log mới được ghi đè lên log cũ -> Log phải được lưu trữ ở một nơi an toàn và phải có kênh truyền đủ đảm bảo tính an toàn và sẵn sàng sử dụng để phân tích hệ thống.
* Do đó lợi ích của log tập trung đem lại là
* Giúp quản trị viên có cái nhìn chi tiết về hệ thống -> có định hướng tốt hơn về hướng giải quyết
* Mọi hoạt động của hệ thống được ghi lại và lưu trữ ở một nơi an toàn (log server) -> đảm bảo tính toàn vẹn phục vụ cho quá trình phân tích điều tra các cuộc tấn công vào hệ thống
* Log tập trung kết hợp với các ứng dụng thu thập và phân tích log khác nữa giúp cho việc phân tích log trở nên thuận lợi hơn -> giảm thiểu nguồn nhân lực.
  + 1. *Thu thập log*

Logging cung cấp một tập hợp các chức năng giúp phân loại cụ thể các mức độ nghiêm trọng của sự kiện. Một số mức độ (level) thường gặp là: debug, info, warning, error và critical.

Bảng 1.1. Mô tả các level trong log

|  |  |
| --- | --- |
| **Level** | **Used** |
| DEBUG | Thông tin chi tiết, thường là thông tin để tìm lỗi |
| INFO | Xác nhận sự kiện xảy ra như mong đợi |
| WARNING | Dấu hiệu cho thấy sự kiện gì đó bất ngờ xảy ra hoặc có thể xảy ra về sau, ví dụ như “disk space low”. Ở level này, hệ thống vẫn hoạt động như mong đợi |
| ERROR | Vấn đề nghiêm trọng hơn WARNING, lúc này hệ thống không thể thực hiện một số chức năng |
| CRITICAL | Lỗi nghiêm trọng, thông báo chương trình có thể không thể tiếp tục chạy |

Dựa vào bảng mô tả trên, có thể dễ dàng phân biệt cũng như lựa chọn khi nào sử dụng các level trong việc ghi log. Level DEBUG thường được sử dụng nhất để xem xét các sự cố khi xảy ra dẫn tới ERROR hay CRITITAL.

Thông tin từ các log file rất đa dạng và phong phú, tuy nhiên thật không may mắn là rất nhiều những thông tin là rất phức tạp để phân tích. Sau khi phân loại được mức độ log, tiếp theo cần phải xác định được định dạng log.

**Định dạng log**

Định dạng log là việc chuyển đổi một bản ghi log thành một chuỗi có thể được giải thích bởi con người hoặc hệ thống bên ngoài, từ đó dễ dang trong việc phân tích.

Một số định dạng được cung cấp trong Django:

*Bảng 1.2. Các định dạng được cung cấp trong django*

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Định dạng** | **Mô tả** |
| Asctime | %(asctime)s | Thời gian bản ghi được tạo |
| Filename | %(filename)s | Phần tên tệp của đường dẫn |
| funcname | %(funcname)s | Tên hàm mà hàm ghi log gọi |
| Levelname | %(levelname)s | Mức độ nghiêm trọng của log ('DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL'). |
| Lineno | %(lineno)d | Vị trí của dòng lỗi |
| Message | %(message)s | Thông điệp log |
| Module | %(module)s | Module (phần tên của một tệp) |
| Name | %(name)s | Tên của hàm ghi log được gọi |
| Pathname | %(pathname)s | Đường dẫn của tệp bị lỗi |
| Proccess | %(process)d | Proccess ID |
| Proccessname | %(proccessname)s | Tên proccess |
| Thread | %(thread)d | Thread ID |
| Threadname | %(threadname)s | Tên thread |

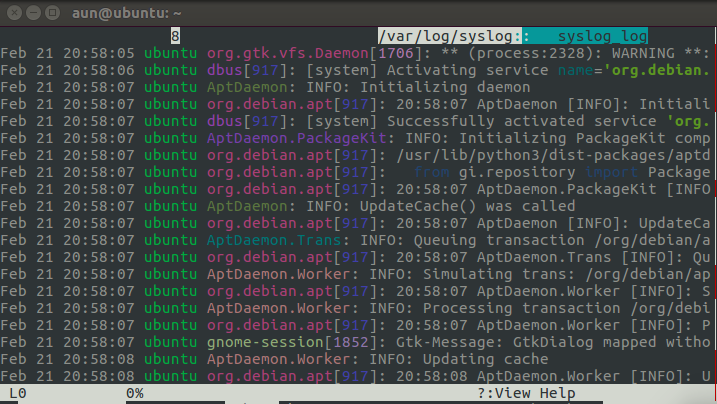
Một số kiểu log:

Unix log:

Các hệ thông unix và linux tạo ra một loạt các thông báo (giống như các log hệ thống), thường tồn tại dưới dạng các plain text, được định dạng như sau:

Ví dụ như:

<date/time><host><message source><message>



Hình 1.1. File log của hệ điều hành linux.

Ví dụ này rất quen thuộc cho ai quản trị hệ thống linux. Định dạng này bao gồm các trường như sau:

Timestamp: Giờ hệ thống của thiết bị ghi nhận log (trường hợp log một đăng nhập từ xa) hoặc của thiết bị tạo log (trong trường hợp tự tạo log).

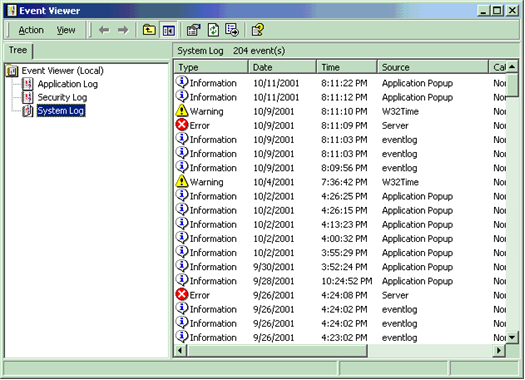
Hostname hoặc IP Address: host name có thể là một tên domain name ví dụ như ns1.example.com hoặc chỉ là tên máy giống như ubuntu trong ví dụ trên.

Message source: nguồn có thể là một phần mềm hệ thống ( **AptDeamon** như trong ví dụ dụ trên) hoặc là một bộ phận (ví dụ như **dbus**) hệ thống mà sản sinh ra log.

Log message: thông báo log có thể có nhiều dạng thông tin khác nhau, thông thường bao gồm các vấn đề xảy ra cụ thể tại vị trí ghi log hay các hành động được thực hiện trong hệ thống.

Windows Log:

Windows (từ NT/2000/XP trở lên) cũng cung cấp logging hệ thống. Tuy nhiên, nó sử dụng định dạng nhị phân (\*evt) để lưu trữ 3 dạng logfile: hệ thống, ứng dụng và an ninh (system, application và security).



Hình 1.2. File log của window

Log hệ thống bao gồm rất nhiều các bản ghi có liên quan tới các vận hành thông thường hoặc bất thường của máy tính. Để đọc được các log Windows, cần sử dụng chương trình hoặc thiết bị có thể đọc được file \*.evt.

Remote Covert Logging

Trong một vài trường hợp (giống như cho honeypots và cho những kịch bản khác), dùng để giấu đi sự có mặt của một logging tập trung từ xa khỏi những người không mong muốn như hacker. Thông thường, file cấu hình syslog cho biết sự hiện diện của logging từ xa và chỉ ra vị trí logging server. Điều này cho phép các hacker có thể tấn công, dò xét các log server, thay đổi hoặc xóa đi vật chứng. Mặt khác, stealthy logging lại rất khó để cho một kẻ tấn công có thể phát hiện ra.

Ví dụ: honey nets

* + 1. *Phân tích log*

Phân tích các log file là một nghệ thuật của việc trích dẫn đầy đủ ý nghĩa thông tin và đưa ra kết luận về một trạng thái an toàn từ các bản ghi thống kế những sự việc được sản sinh bởi máy tính. Phân tích log file không phải là ngành khoa học, nhưng ngày nay, việc tin tưởng vào kỹ năng phân tích độc lập và trực quan cũng như tính chất may mắn trong việc phân tích log chất lượng cũng là một khái niệm khoa học. Định nghĩa việc phân tích log có thể nghe rất khô khan, nhưng quan trọng là rút ra một “kết luận có ý nghĩa”. Nhìn một cách đơn giản vào các file log không phải là phân tích, bởi vì hiếm có những cái gì ngoài những sự nhàm chán và dường như chẳng liên quan gì đến nhau. Trong trường hợp một thiết bị một người sử dụng với rất ít các hoạt động, tất cả những bản ghi log mà chưa được nhìn trước là rất ít nghi ngờ, nhưng trong thực tế lại không dễ dàng như vậy. Hãy thử xem một phân tích log cho những telnet chung. Đầu tiên, hãy nhìn qua toàn bộ log cần phải phân tích(giống như file log của một thiết bị xâm nhập đối với một thông báo tấn công thành công) và tạo quan hệ với những nguồn thông tin khác. Việc tạo quan hệ có nghĩa là thực hiện những thao tác bằng tay hoặc tự động để thiết lập nên mối quan hệ giữa các sự kiện tưởng chừng không liên quan xảy ra trên mạng. Các sự kiện xảy ra trên các thiết bị khác nhau trong các thời điểm khác nhau có thể tạo nên những quan hệ tức thời (xuất hiện trong thời gian ngắn). Đây có phải là một lỗ hổng cho kẻ tấn công có thể phát hiện được? Có phải các quy tắc của các hệ thống phát hiện xâm nhập đưa ra một dự báo sai. Có phải là một ai đó trong số các nhân viên của bạn đang thử quét các lỗ hổng trong mạng của bạn? Trả lời cho những câu hỏi tương tự như vậy là rất cần thiết trước khi lập kế hoạch phản ứng cho các thông báo của IDS. Các cố gắng kết nối, nắm bắt các dịch vụ và những sai lầm đa dạng của hệ thống thường yêu cầu thực thi rất nhiều những việc tạo mối quan hệ với những nguồn thông tin khác nhau theo nhiều mức để đạt được thông tin có ý nghĩa đầy đủ nhất

Việc phân tích log rất đa dạng và tuy vào ngữ cảnh cũng như ý định phân tích, dưới đây là một số kịch bản phân tích thường thấy:

* Theo dõi việc sử dụng Account:

Xác định tài khoản nào đang thử thực hiện đăng nhập và theo dõi tài khoản bị thỏa hiệp.

Logon event không thể ghi nhận khi backdoor, exploited services hoặc các hành động độc hại tương tự gán quyền truy cập đến hệ thống. Bởi vì, những hành động này sử dụng backchannel và sử dụng các APIs để đạt được việc truy cập.

* Phân tích truy cập tập tin và thư mực:

Nhận diện users thử truy cập các tập tin, thư mục, registry key, … được bảo vệ.

* Tìm kiếm tài khoản lạ:

Khi bị tấn công hay phát hiện xâm nhập vào các tập tin và thư mục mật việc tiếp theo thường là kiểm tra các tài khoản truy cập.

* Sự cài đặt của ứng dụng

Một trong những con đường dẫn tới sự cố cũng như con đường để xâm nhập là cài đặt các ứng dụng.

Quan sát logs để nhận ra các phần mềm không được phép hoặc độc hại được cài đặt, theo dõi quá trình phần mềm độc hại được gỡ bỏ trước đó và nhận diện việc cố thử cài đặt phần mềm.

* Các dịch vụ độc hại

Phân tích logs cho việc phát hiện các dịch vụ độc hại chạy lúc khởi động và quan sát các dịch vụ started hoặc stopped xung quanh thời điểm bị thỏa hiệp.

* Xóa bỏ event log

Xác định xem log có bị thay đổi hay xóa bỏ bởi hacker.

* Các thiết bị không được xác thực
  1. **Tổng quan về log DJANGO**
     1. *Giới thiệu*

Từ khi được Guido van Rossum ra mắt từ đầu những năm 1990, sau nhiều năm không ngừng hoàn thiện, Python đang phát triển mạnh mẽ và ngày càng trở nên phổ biến. Đến năm 2016, Python là ngôn ngữ phổ biến thứ 4 thế giới sau Java, C và C++. Python là ngôn ngữ lập trình đa chức năng có thể ứng dụng ở nhiều lĩnh vực khác nhau.

Python là lựa chọn phổ biến khi ta cần viết scripts cho mục đích test và monitor. Python cũng đã từng được dùng nhiều trong lập trình game, khả năng kết hợp với các ngôn ngữ khác của Python là điểm sáng giá “ăn tiền”. Python nổi tiếng đến mức nó từng được [Industrial Light and Magic của George Lucas sử dụng](https://www.python.org/about/success/ilm/) (đảm nhiện hiệu ứng đặc biệt trong Star Wars trilogy đầu tiên) để quản lý quá trình sản xuất phức tạp của bộ phim.

Không như PHP, Python không được xây dựng chuyên cho môi trường web và không có dủ tính năng web cốt lõi cần có. Bởi vậy, chúng ta phải sử dụng web framework để phát triển ứng dụng web trên Python. Kể từ sự xuất hiện của các framework tuyệt vời như Django, lập trình viên web đã bắt đầu sử dụng Python nhiều hơn.

Django là một web framework khá nổi tiếng được viết hoàn toàn bằng ngôn ngữ python, là một framework với dầy đủ các thư viện, module hỗ trợ các web-developer. Django sử dụng mô hình Model-View-Control (MVC)

Một số website phổ biến được xây dựng từ Django là Pinterest, Instagram, Mozilla, Bitbcuket,…

**Log trong Django**

Trong một chương trình lớn , một vấn đề khá đau đầu là thông tin xảy ra rất nhiều, không được phân loại, mất thời gian xác định ảnh hưởng tới hiệu xuất. Vì Django được xây dựng dựa trên ngôn ngữ Python nên module logging là một phần không thể thiếu (được pack kèm bộ cài mặc định) do cộng đồng python phát triển dễ dàng sử dụng và cực kì linh hoạt. Module logging cung cấp các tiện ích như sau:

* Phân chia level cho các thông báo lỗi, cho phép tùy chọn mức độ nghiệm trọng của các thông báo, cho phép hoặc không cho phép hiển thị một thông báo đã được phân loại.
* Cho phép cấu hình output của thông báo lỗi là trên console hoặc file hoặc nguồn khác.
  + 1. *Đặc điểm*

Mục tiêu chính của Django là đơn giản hóa việc tạo các website phức tạp có sử dụng cơ sở dữ liệu. Django tập trung vào tính năng “có thể tái sử dụng” và “có thể tự chạy” của các component, tính năng phát triển nhanh,không làm lại những gì đã làm.

Tương tự python, Django cũng rất dễ học, các SQL table hay form được tự động tạọ và liên kết tới các model nhằm tối ưu khả năng đơn giản. Django còn cũng cấp tính năng admin interface và shell giúp việc quản lý và debug trở nên dễ dàng.

Log trong Django cũng là một đặc điểm nỗi bật, ngoài việc cung cấp các thông tin ghi lại sự việc diễn ra trong hệ thống, nó còn có các dãn cảnh báo gắp vào để dễ dàng nhận diện các thông tin nào là quan trọng cần thiết giúp người quản trị dễ dàng phát hiện các sự cố cũng như đưa ra các giải pháp bảo mật.

* + 1. *Cơ chế thu thập log*

DJANGO sử dụng cơ chế thu thập log tập trung, tức là các ứng dụng web đang chạy sẽ được tập trung lại và tiến hành ghi log.

**Cơ chế xử lý log trong Django**

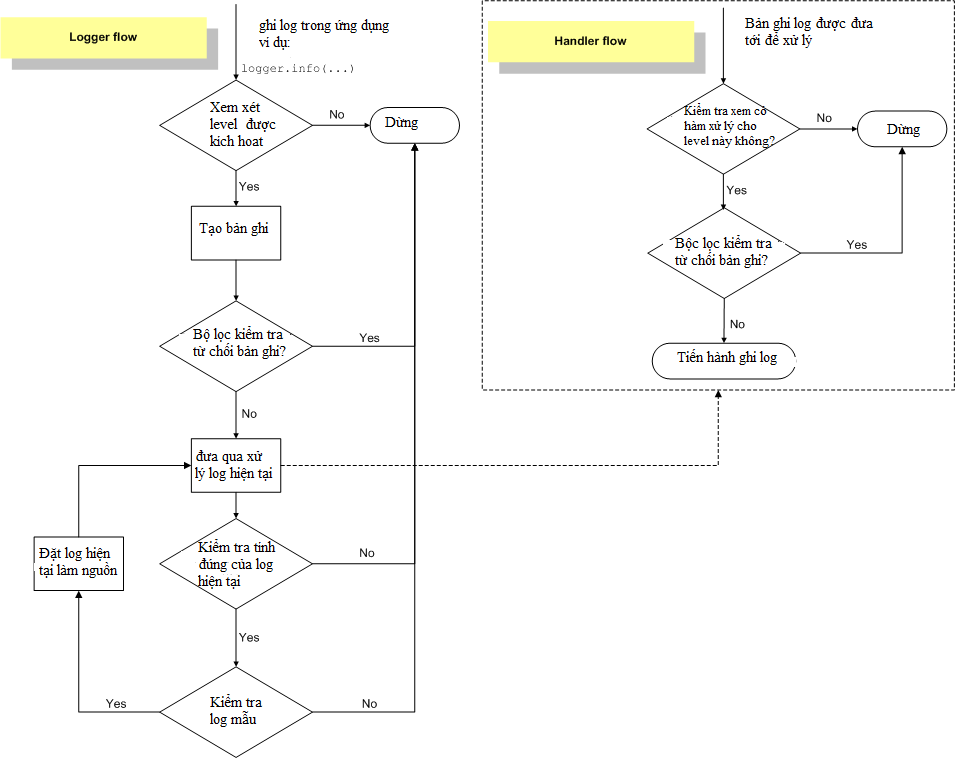
Hoạt động xử lý log trong Django được chia làm 2 luồng chính là: Looger và Handler.

* Logger:

Logger chia làm 3 nhiệm vụ chính. Đầu tiên, đưa ra một số phương thức cho mã ứng dụng để ứng dụng có thể ghi nhật kí khi chạy. Thứ hai, các đội tượng cần ghi log xác định thông điệp log nào sẽ được ghi lại dựa trên mức đông nghiêm trọng (level) . Thứ ba, các đối tượng ghi log sẽ chuyển thông điệp log đến tiến trình xử lý log.

* Handler:

Handler có trách nhiệm gửi các thông điệp log thích hợp (dựa trên level của thông điệp log) đến đích xử lý.



Hình 1.3. Cơ chế xử lý log trong Django

Ví dụ đơn giản nhất khi sử dụng logging:

1. # source: https://docs.python.org/3.6/howto/logging.html#a-simple-example
2. **import** logging
3. logging.warning('Watch out!')  # sẽ in ra warning trên console
4. logging.info('I told you so')  # không in gì cả

Mặc định, mức thông báo sẽ được đặt ở mức WARNING. Các thông báo có mức dưới nó (DEBUG, INFO) vì vậy sẽ không được in ra. Để thay đổi mức này, ta đơn giản chỉ việc đặt lại thông số logging level.

Ngoài ra cũng có thể log ra file:

1. **import** logging
2. logging.basicConfig(filename='example.log',level=logging.DEBUG)
3. logging.debug('This message should go to the log file')
4. logging.info('So should this')
5. logging.warning('And this, too')

Cần chú ý là nội dung log mới sẽ ghi nối tiếp vào cùng một file với nội dung log cũ (lần chạy trước đó).

Logging cho phép thêm các thông tin chi tiết hơn về chương trình khi gọi log như: vị trí gọi log, tên module gọi log, file đang chạy,….. Các thông tin cụ thể đã được liệt kê ở “Bảng 1.2. Các định dạng được cung cấp trong django”.

Một config đầy đủ có thể như sau:

1. **import** logging
2. logging.basicConfig(level=logging.DEBUG, filename='runtime.log', filemode='w', format = (
3. '%(levelname)s:\t'
4. '%(filename)s:'
5. '%(funcName)s():'
6. '%(lineno)d\t'
7. '%(message)s'
8. )
9. )

Việc phân loại chi tiết như trên rất hữu ích khi phải phân tích các file log dài, khi đó ta có thể sử dụng các ứng dụng để đọc và phân tích các thông tin tương ứng một cách nhanh chóng.

* 1. **Kỹ thuật và công nghệ sử dụng trong ứng dụng**
     1. *Thuật toán Prefixspan (thuật toán tính tần suất mẫu)*
* Một số định nghĩa
* Định nghĩa 1 (prefix, projection, postfix)

Giả sử có tất cả các mục của một thành phần được sắp xếp theo thứ tự a, b,c. Với một chuỗi α=(e1e2e3…en) và một chuỗi β=(e'1e'2e'3…e'm) (m ≤ n) là tiền tố (prefix) của α khi và chỉ khi:

* e'i = ei for (i ≤ m-1)
* e'm ⸦ em
* Tất cả các mục trọng (em- e'm) được sắp xếp sau e'm

Với một chuỗi con α và β như thế, β là chuỗi con của α. Một chuỗi con α' của chuỗi α gọi là hình chiếu (projection) của α tương ứng với tiền tố của β khi và chỉ khi:

* α' có tiền tố β
* Không tồn tại chuỗi α'' nào là tiền tố của β mà lớn hơn α'

Với α'=(e1e2e3…en) là hình chiếu của α tương ứng với tiền tố β=(e1e2e3…em-1e'm).

Chuỗi con γ = (e''mem+1...en) gọi là hậu tố của α tương ứng với tiền tố β khi γ=α/β với e''m =(em-e'm)2.

Ví dụ: cho một dãy α=<a(abc)(ac)d(cf)>

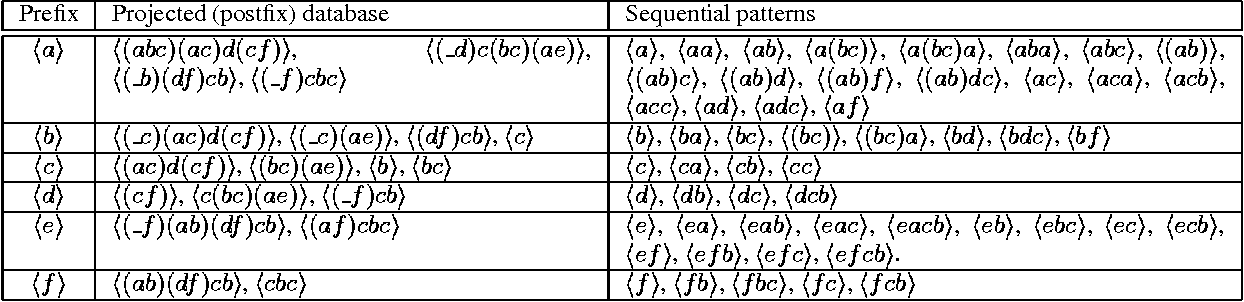
* <a>, <aa>,<a(ab)> và <a(abc)> là tiền tố α
* <(abc)(ac)d(cf)> là hậu tố chuỗi α tương ứng tiền tố <a>
* <(\_bc)(ac)d(cf)> là hậu tố tương ứng tiền tố <aa>
* <(\_c)(ac)d(cf)> là hậu tố tương ứng tiền tố <ab>
* Định nghĩa 2

Cho α là một chuỗi trong dữ liệu S. α-projected database kí hiệu là S|α , là tất cả các tiền tố của các chuỗi trong S tương ứng là tiền tố của α

* Định nghĩa 3

Cho chuỗi α trong chuỗi dữ liệu S và β là một chuỗi có tiền tố là α. Độ hỗ trợ (support count) của β trong α-projected database S|α(β), là số lượng chuỗi γ trong S|α, ký hiệu.

Bảng 1. 3. Prefix, postfix và các mẫu tuần tự tương ứng



* Thuật toán PrefixSpan

Thuật toán PrefixSpan tính tần suất của tất cả các chuỗi con trong tập dữ liệu của câu. Đầu tiên, thuật toán bắt đầu với một tập hợp tần suất của các chuỗi con gồm các từ đơn (single items). Sau đó, thuật toán được mở rộng, với một chuỗi con có kích thước k gắn thêm một từ mới để tính được tần suất của dãy con có kích thước k+1. Thuật toán tính được tất cả tần suất của chuỗi con thông qua lặp đệ quy.

Tuy nhiên, việc mở rộng một chuỗi con bằng cách thêm một nút mới vào bất kì vị trí của lá có thể dẫn đến tình trạng trùng lặp các cây con mới được sinh ra. Để tránh điều này, thuật toán hạn chế vị trí đính kèm một nút mới vào cuối dãy con mới theo thứ tự từ trái sáng phải.

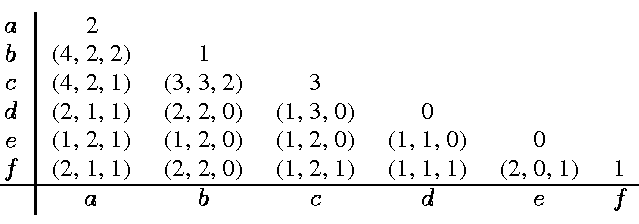
Thuật toán prefixspan được mô tả chi tiết như sau:

1. Đầu vào: một chuỗi dữ liệu S và độ ngưỡng hỗ trọ min-sup.
2. Đầu ra: Tập các mẫu liên tiếp sinh ra từ chuỗi dữ liệu ban đầu.
3. Hàm: PrefixSpan(α,l, S|α)
4. Tham số: α: là mẫu liên tục; l: là đồ dài của α; S|α: α-projected database, nếu α # (); và ngược lại, chuỗi dữ liệu S
5. Phương thức:
6. Quét S|α một lần, tìm tập các tần suất từ β như sau:

* β có thể được thêm vào phần tử cuối cùng của α để thành một mẫu tuần tự hoặc
* <β> có thể thêm vào α mẫu tuần tự

1. Lặp lại với mỗi item thường xuyên β, thêm nó vào chuỗi α để tạo thành chuỗi mới α', in ra α'.
2. Với mỗi α, sinh ra S|α', và gọi lại hàm PrefixSpan(α',l+1,S|α')

Bảng 1.4. Minh họa các vòng lặp của thuật toán prefixspan

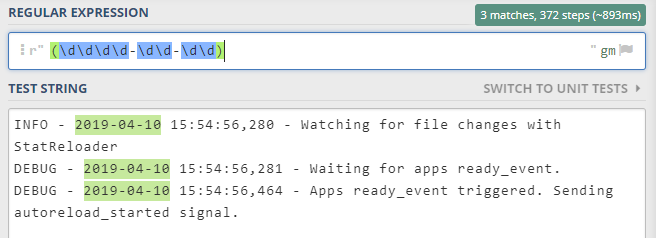


* Đánh giá
* PrefixSpan chỉ tăng số lượng các mẫu tuần tự dài hơn từ các mẫu ngắn hơn của nó. Thuật toán không tạo ra và cũng không kiểm tra bất kì các chuỗi ứng viên nào không tồn tại trong cơ sở dữ liệu dự kiến.
* Chi phí chính của PrefixSpan là xây dựng cơ sở dữ liệu dự kiến
  + 1. *Regular Expression*

Biểu thức chính quy (regular expression) là các mẫu dùng để tìm kiếm các bộ kí tự được kết hợp với nhau trong các chuỗi kí tự. Trong python thì các mẫu biểu thức chính quy cũng đồng thời là các đối tượng, tức là khi ta tạo ra một biểu thức chính quy là ta có một đối tượng tương ứng. Các mẫu này được sử dụng khá nhiều trong python như math, replace, search, split và findall.

Trong một file log chứa rất nhiều thông tin vì vậy điều quan trọng nhất là phải trích xuất được đúng dữ liệu cần, regular expression giúp tạo ra các mẫu để tìm kiếm các dữ liệu dựa trên các mẫu đã đưa ra.

Ví dụ: trong một file log ta cần trính xuất các thời gian xảy ra sự kiện trong file log, ta chỉ cần tạo ra một mẫu với format là ngày tháng năm để có thể lấy dữ liệu. Dưới đây là một ví dụ minh họa:



Hình 1.4. Minh họa sử dụng biểu thức chính quy

Để có thể hiểu được hết các tính năng của biểu thức chính quy mất rất nhiều thời gian và công sức nghiên cứu, trong phạm vi đề tài này sẽ đề cập tới các mẫu được sử dụng để trích lọc các thông tin log. Dưới đây là bảng thông kê chi tiết các mẫu được dùng:

Bảng 1.5. Mô tả đầy đủ các ký tự đặt biệt trong RE

|  |  |
| --- | --- |
| Kí tự (kí hiệu, cờ) | Ý nghĩa |
| \ | Một dấu gạch chéo ngược sẽ biến một kí tự thường liền kế phía sau thành một kí tự đặc biệt, tức là nó không được sử dụng để tìm kiếm thông thường nữa. Ví dụ,  trường hợp kí tự 'b' không có dấu gạch chéo ngược này sẽ được khớp với các kí tự 'b' in thường, nhưng khi nó có thêm dấu gạch chéo ngược, '\b' thì nó sẽ không khớp với bất kì kí tự nào nữa, lúc này nó trở thành kí tự đặc biệt. |
| + | Cho phép kí tự trước nó lặp lại 1 lần hoặc nhiều lần. Tương đương với cách viết {1,}.Ví dụ, /a+/ khớp với 'a' trong chuỗi "candy" và khớp với tất cả kí tự a liền nhau trong chuỗi "caaaaaaandy". |
| . | Dấu . khớp với bất kì kí tự đơn nào ngoại trừ kí tự xuống dòng.Ví dụ, /.n/ khớp với 'an' và 'on' trong chuỗi "no, an apple is on the tree", nhưng không khớp với 'no'. |
| (x) | Khớp 'x' và nhớ kết quả so khớp này, như ví dụ ở dưới. Các dấu ngoặc tròn được gọi là các dấu ngoặc có nhớ. |
| x(?=y) | Chỉ khớp 'x' nếu 'x' theo sau bởi 'y'. |
| x|y | Khớp 'x' hoặc 'y' |
| {n} | Kí tự đứng trước phải xuất hiện n lần. n phải là một số nguyên dương.Ví dụ, /a{2}/ không khớp với 'a' trong "candy", nhưng nó khớp với tất cả kí tự 'a' trong "caandy", và khớp với 2 kí tự 'a' đầu tiên trong "caaandy" |
| [xyz] | Lớp kí tự. Loại mẫu này dùng để so khớp với một kí tự bất kì trong dấu ngoặc vuông. |
| \d | Khớp với một kí tự số. Tương đương với mẫu [0-9].Ví dụ: /\d/ hoặc /[0-9]/ khớp với '2' trong chuỗi "B2 is the suite number." |
| \D | Khớp với một kí tự không phải là kí tự số. Tương đương với mẫu [^0-9]. |
| \s | Khớp với một kí tự khoảng trắng, bao gồm trống - space, tab, phân trang - form feed, phân dòng - line feed. Ví dụ: /\s\w\*/ khớp với ' bar' trong "foo bar." |
| \S | Khớp với một kí tự không phải khoảng trắng. |

Bảng 1.6. Những phương thức được sử dụng trong RE

|  |  |
| --- | --- |
| Phương thức | Mô tả |
| match | Một phương thức của chuỗi dùng để tìm kiếm chuỗi phù hợp với mẫu so khớp. Nó trả về một mảng chứa kết quả tìm kiếm hoặc null nếu không tìm thấy. |
| Search | Một phương thức của chuỗi dùng để tìm kiếm chuỗi phù hợp với mẫu so khớp và trả về vị trí của chuỗi đó hoặc -1 nếu không tìm thấy. |
| replace | Một phương thức của chuỗi dùng để tìm kiếm một chuỗi theo mẫu so khớp và thay thế chuỗi con được khớp với một chuỗi thay thế. |
| split | Một phương thức của chuỗi dùng một biểu mẫu chính quy hoặc một chuỗi bất biến để ngắt chuỗi đó thành một mảng các chuỗi con. |

* + 1. *Module logging*

Module này đã được giới thiệu ở phần tổng quan về log, mục này sẽ tiếp tục giới thiệu các hàm có trong module để cấu hình.

* Formatters: logging.formatters(fmt,datefmt)

Hàm tạo có hai đối số tùy chọn - chuỗi định dạng thông báo (fmt) và chuỗi định dạng ngày (datefmt).

Nếu không truyền chuỗi định dàng vào, mặc định dùng chuỗi định dạng của hệ thống. Ví dụ như định dạng ngày tháng, định dạng ngày mặc định là:

%Y-%m-%d %H:%M:%S

Chuỗi định dạng dưới đây sẽ ghi lại log theo định dạng thời gian và nội dung đọc được

'*%(asctime)s* - *%(levelname)s* - *%(message)s*'

Các định dạng đã được đề cập ở

* Handlers

Handlers có trách nhiệm gửi các thông điệp log đến đích xử lý dựa trên mức độ nghiệm trọng của thông điệp.

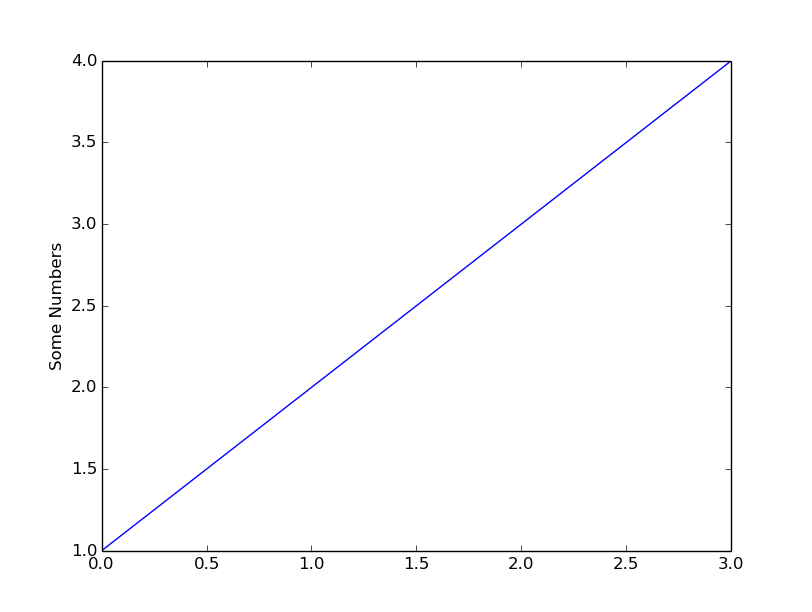
Một số hàm hay sử dụng trong handlers như setlevel(), addFilter()

* + 1. *Matplotlib*

Matplotlib là một tập hợp các hàm mà matplotlib có thể làm việc như MATLAB. Mỗi hàm pyplot tạo lên một sự thay đổi trong đồ thị, ví dụ như tạo một khung đồ thị, vẽ dồ thị trên khung đã tạo, hoặc vẽ nhiều đường biểu diễn trên cùng 1 đồ thị, dãn nhãn cho các trục,... Các trạng thái khác nhau trong matplotlib.pyplot lưu giữ các hàm đã gọi, qua đó lưu mọi thứ của đồ thị hiện tại và vùng đồ thị được vẽ, và hàm của đồ thị được vẽ trực tiếp trên trục tọa độ (axes ở đây mang nghĩa về đồ thị của hình được vẽ chứ không phải dịnh nghĩa nhiều trục trong toán học)

Ví dụ:

1. **import** matplotlib.pyplot as plt
2. plt.plot([1,2,3,4])
3. plt.ylabel('some numbers')
4. plt.show()



Hình 1.5. Đồ thị được vẽ bởi matplotlib

Trên đồ thị cho thấy trục x có khoảng từ 0-3 nhưng trục y lại có khoảng từ 0-4. Vì nếu sử dụng một danh sách đơn hoặc dãy cho hàm plot(), matplotlib sẽ hiểu đấy là chuỗi cho chiều cao của trục y và sẽ tự động đồng bộ nó với trục x. Do phạm vi của python bắt đầu bằng 0, nên vector mặc định của x sẽ có cùng chiều dài với y và bắt đầu từ 0. Vì thế trục x sẽ là [0, 1, 2, 3]

Dưới đây là một số phương thức có thể được xuất hiện trong biểu đồ 2 chiều:

Bảng 1.7. Mô tả phương thức trong biểu đồ 2D

|  |  |
| --- | --- |
| Phương thức | Mô tả |
| color | Xác định màu sắc biểu diễn dữ liệu |
| label | Nhãn của biểu đồ |
| xlabel | Nhãn của trục hoành |
| ylabel | Nhãn của trục tung |
| data | Dữ liệu đầu vào của đồ thị, có dạng (xdata,ydata) |
| xdata | np.array chứa list tọa độ trục hoành |
| ydata | np.array chứa list tọa độ trục tung |

Trong đồ thị phần quan trọng nhất sẽ là dữ liệu đầu vào cho 2 trục tung và hoành. Trong đồ án này sẽ sử dụng matplotlib để vẽ đồ thị realtime log, thống kê lượt truy cập,…

* 1. **Kết luận chương 1**

Qua chương 1, ta có cái nhìn khái quát về log, file log được ghi bởi django, các công nghệ và kỹ thuật được áp dụng để phát triển ứng dụng. Ngoài ra, mục tiêu trong nghiên cứu là có thể áp dụng học máy vào ứng dụng để có thể đưa ra các phân tích một cách nhanh nhất, giảm tải được sự phức tạp trong phân tích.

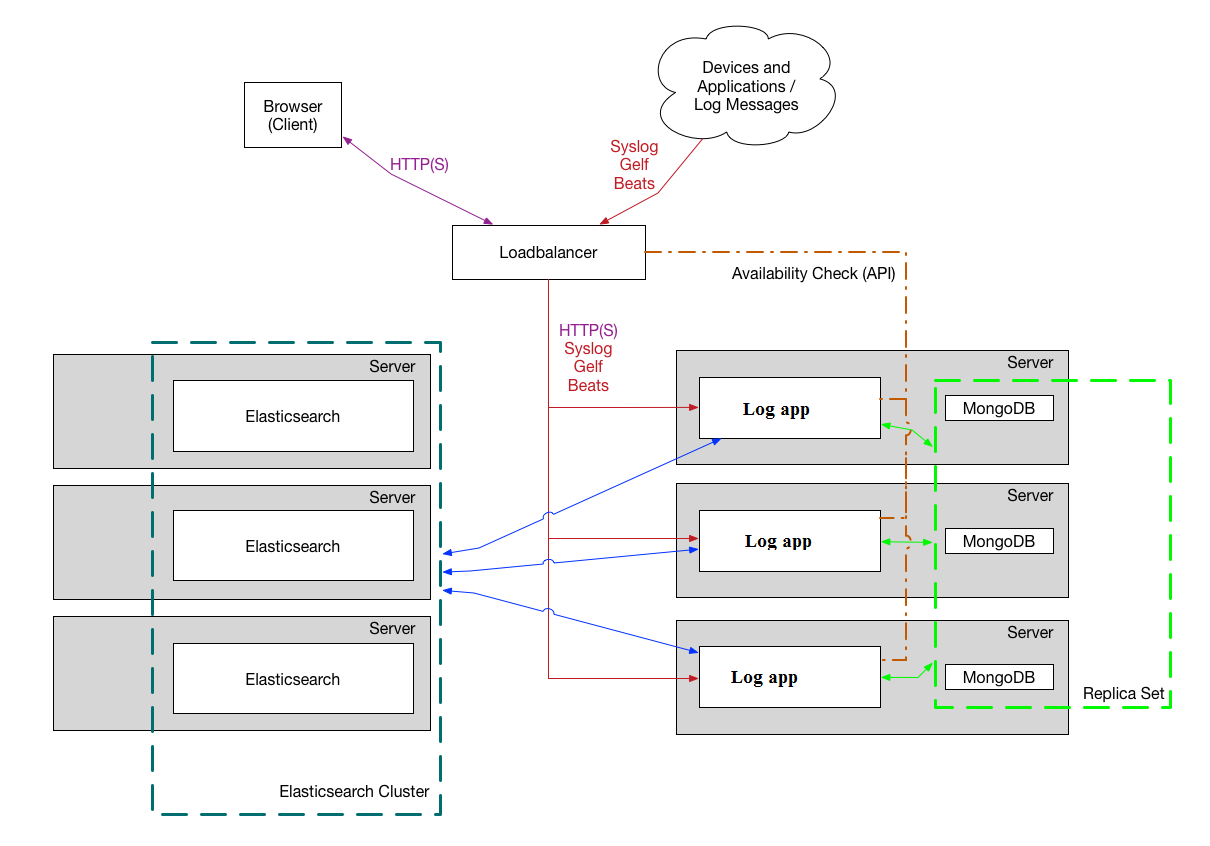
Với các khái niệm và kiến thức đã nghiên cứu, tiếp theo ta sẽ thiết kế ứng dụng đảm bảo các mục tiêu đã đề ra ở phần I.

# **CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ ỨNG DỤNG**

Khi bắt đầu đưa ra ý tưởng về ứng dụng, điều đầu tiên của mục đích đưa ra là phải tạo sự dễ dàng, thuận tiện cho ứng dụng. Vì vậy, ứng dụng sẽ được thiết kế dựa trên việc tích hợp vào cùng và hoạt động song song với hệ thống website Django đang chạy.

* 1. **Mô hình ứng dụng**

Trong đề tài nghiên cứu này hướng tới ứng dụng dựa trên nền django để tích hợp từ đó thuận tiện cho việc cấu hình, cài đặt, dễ dàng cho việc sử dụng.



Hình 2.1. Mô hình triển khai ứng dụng

Database để lưu log sẽ được xem như một bảng (table) trong cơ sở dữ liệu của webserver. Điều này mang tới sự thuận tiện cho việc cài đặt ứng dụng vì chúng ta không cần phải quan tâm đến nơi đặt dữ liệu cũng như sql server. Ở đây chúng ta nên sử dụng một NoSQL để làm database.

Khi có browser (client) truy cập vào server, server log app tiến hành phân tài nguyên, ghi log và lưu log xuống cơ sở dữ liệu.

Hình 2.2. Mô hình hoạt động ứng dụng

Discovered knowledge

Event flow (off-line procesing)

Event Logs

Event flow (real-time)

Event Correlator analyst

output

Event correlation rules

Ứng dụng sẽ được chia làm 2 luồng xử lý: một theo thời gian thực, một theo thống kê trích xuất dữ liệu dựa trên các mẫu có sẵn để huấn luyện.

* 1. **Thiết kế dữ liệu**

Thông tin từ các log file là rất đa dạng và phong phú. Việc xác định rõ những thông tin nào là cần thiết để quản trị và khắc phụ sự cố.

Vì django là một web framework, được sử dụng để xây dựng các hệ thống website mà điều quan trọng trong quản trị website là thời gian ghi log, địa chỉ ip truy cập, path được truy cập, user login và message. Vì vậy ở đây ta cần phải thiết kế dữ liệu để lưu lại được các thông tin ý nghĩa.

Như đã nói ở trên sở dĩ chúng ta nên dùng một NoSQL như mongodb hay postgresql,… chính là vì cấu chúc không cần khai báo cố định các cột vì vậy ta có thể dễ dàng thoải mái trongg việc thay đổi các thông tin ghi log xuống mà không lo dẫn tới việc xung đột dưới database.

**Database Website**

**Log Database**

* Time : datetime
* Methods: GET/POST/PUT/DELETE
* Path: path website
* Referer: url
* User agent: user web brower
* Ip Adrress:
* User

Hình 2.3. Mô hình lớp dữ liệu của ứng dụng

Tùy vào mục đích và nhu cầu mà người dùng có thể thay đổi các thông tin cần lưu lại.

* 1. **Thiết kế chức năng**

Khi bắt đầu thiết kế ứng dụng giám sát nhật ký, việc có một mục tiêu rõ ràng cho phép xác định chính xác các chức năng cần xây dựng.

* Hiển thị: chỉ quan tâm đến các thông tin được trích xuất và thông kê ra nên giao diện chủ yếu sẽ sử dụng các thư viện được tích hợp sẵn hoặc terminal.
* Khả năng tiếp cận: sau khi ghi và lưu lại các file log, cần xây dựng các hàm có thể trích xuất các truy vấn về hành vi của người dùng, thông kê được lưu lượng người dùng truy cập (khách không đăng kí tài khoản và đã có tài khoản),..
* Tích hợp: ứng dụng phải tích hợp được vào django, để có thể sử dụng các tài nguyên được cấu hình sẵn, giúp đơn giản hóa các vấn đề về cài đặt
* Cảnh báo: Đây là quản lý cảnh báo quan trọng khi phát hiện một số sự kiện hoặc mẫu nhất định?

Khả năng mở rộng: opensource để mọi người cùng đóng góp và mở rộng các chức năng.

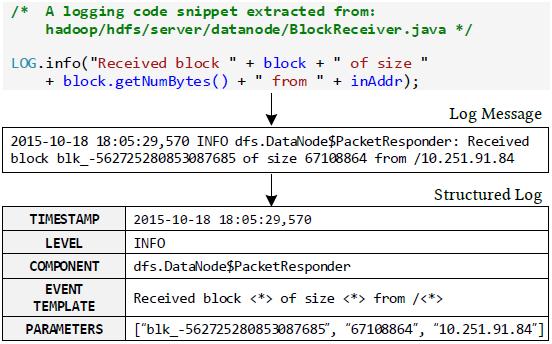
Chức năng đầu tiên cũng như là tiền đề cho các chứng năng khác, đó là trích xuất và lưu lại log.

Dựa vào kinh nghiệm bản thân, để dễ dàng cho việc trích xuất thông tin trong log thì giữa các thông tin cần phân cách bởi một kí tự đặt biệt, ở đây em sử dụng kí tự “-“ để phân cách các thông tin.

Khi sử dụng kí tự “-“ để phân cách thì log sẽ có dạng:

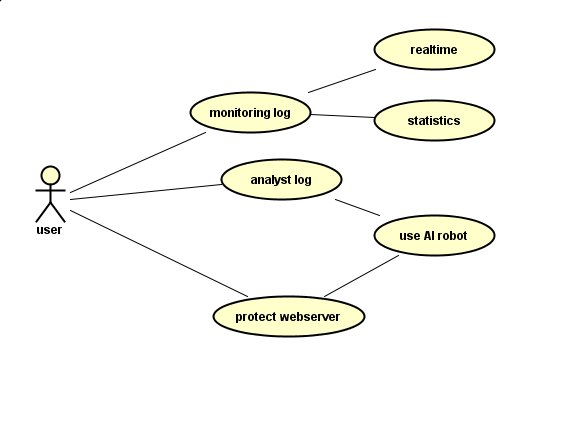
'**{levelname}**-**{asctime}**-**{module}**-**{process:d}**-**{thread:d}**-**{message}**'

Khi trích xuất ta chỉ cần đề ý tới dấu “-“ để nhận dạng sự thay đổi thông tin.



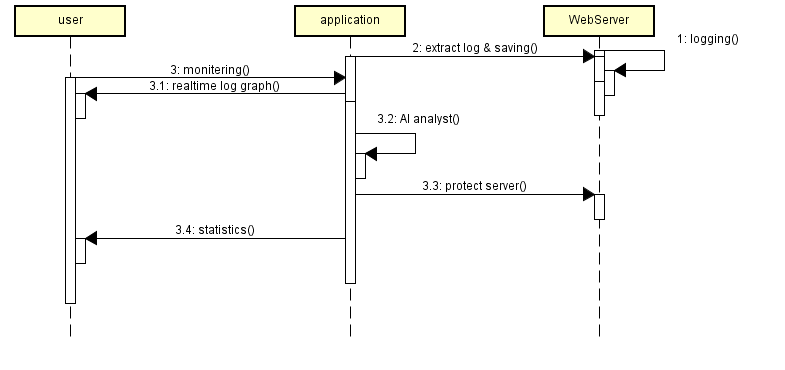
Hình 2.4. Dữ liệu khi trích xuất từ file log

Sau khi trích xuất và lưu lại log xong, tiến hành cung cấp các chức năng giám sát, phân tích cho người dùng.

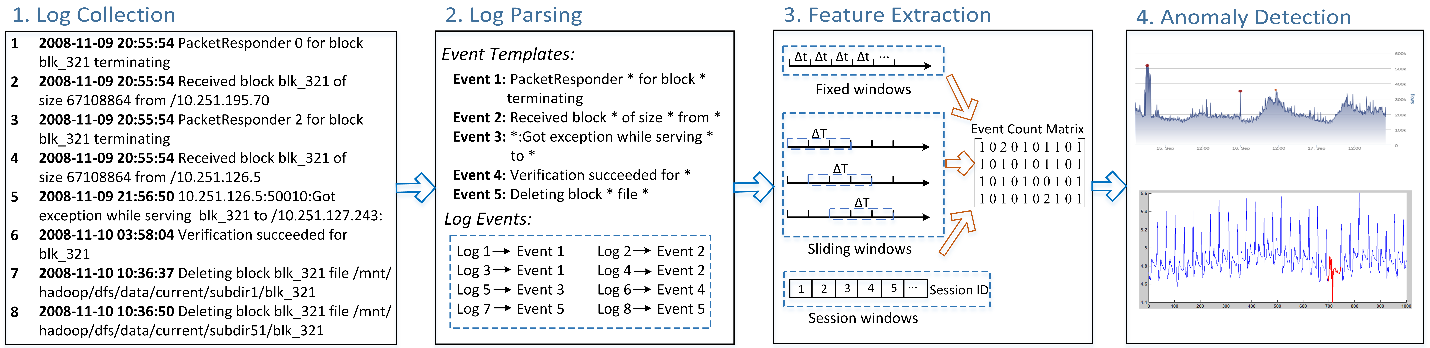


Hình 2.5. Mô hình mô tả các chức năng của ứng dụng

Trong một phiên làm việc, ứng dụng cung cấp khả năng tự động phân tích đưa ra các giải pháp nếu xảy ra sự cố, sau đó thông kê lại cho người dùng



Hình 2.6. Mô hình trình tự xử lý log trong ứng dụng



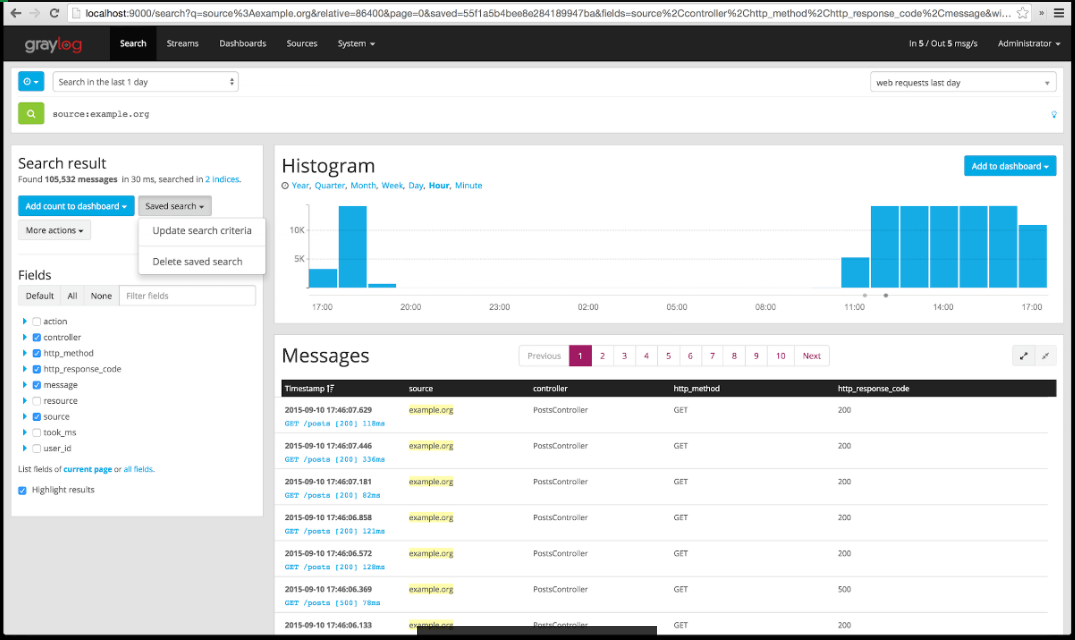
Hình 2.7. Realtime log model

* Realtime log cung cấp số liệu thống kê và dữ liệu theo ngữ cảnh như hiệu suất của hệ thống theo thời gian thực, giống như dữ liệu log được cập nhật theo thời gian thực nhưng vẫn thường trong một khung thời gian rất ngắn từ khi cập nhật dữ liệu. Realtime log sẽ hướng tới sử dụng để phân tích:
* Thống kê chung về web như băng thông và lưu lượng truy cập trang web của khách truy cập theo thời gian thực
* Số liệu về tốc độ yêu cầu
* Các tài nguyên được yêu cầu
* Các trang được xem nhiều nhất
* Các cụm từ dược gửi lên server nhiều nhất
* Statistic log cung cấp số liệu thống kê về tất cả dữ liệu log đã được ghi lại trong file log, hạn chế của chức năng này là không thể cho thấy được các diễn biến đang diễn ra trong hệ thống nhưng nó mang lại được cái nhìn tổng quát về tất cả những gì đã diễn ra từ trước tới nay được ghi lại trong file log như:
* Tổng số lượng log đã được ghi, chi tiết các log đã được trích xuất theo các trường cụ thể
* Thống kê về lượng khách truy cập từ trước tới nay theo thời gian, thể hiện rõ lượng khách đã đăng nhập và chưa đăng nhập
* Thống kê các các trang được xem nhiều nhất trong khoảng thời gian được ghi log
* Thông kê lại các request gần đây nhất, user gửi request, trạng thái request, thời gian gửi request.
* AI analyst sử dụng các mẫu log đã được phân tích trước đó để làm giấu hiệu nhận biết từ đó phân tích log một cách tự động. Với hướng sử dụng học máy vào ứng dụng, ta có thể giảm tại được khối lượng phân tích cho người dùng, mặt khác sử dụng học máy nhằm hướng tới khả năng tự ra quyết định và bảo vệ hệ thống khi nhận biết được mối đe dọa như khả năng nhận biết tấn công truy cập vào hệ thống
  1. **Thiết kế giao diện**

Giao diện chương trình sẽ được tích hợp cùng với website trên django-admin, giúp dễ dàng và trực quan.

Dựa vào các công cụ phân tích log như splunk hay graylog để tạo giao diện cho ứng dụng từ đó tạo cảm giác thân thuộc đỡ mất công làm quen.

Các giao diện tham khảo để thiết kế:



Hình 2.8. Giao diện Graylog

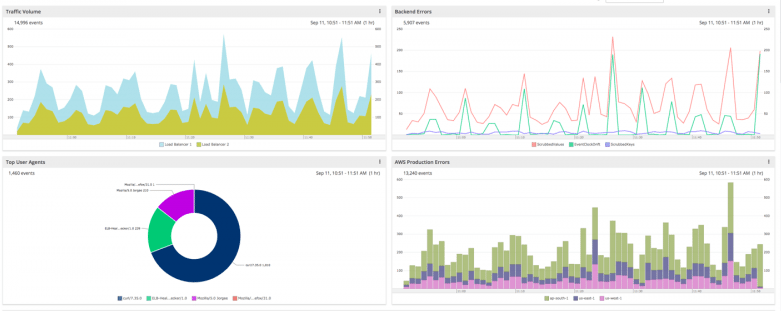
Hình 2.9. Giao diện thống kê log của Graylog

* Với giao diện của các log analysis tools đã có minh họa ở trên ta sẽ thiết kế giao diện cho các log dưới dạng bảng với định dạng như:

Bảng 2.1. Minh hoa giao diện liệt kê log trên web

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Time | Path | Response | Methods | From |
| log1 | Log11 | Log1 | Log1 | Log1 |
| Log2 | Log2 | Log2 | Log2 | Log2 |

* Khác với xem liệt kê các log trong file log, nếu sử dụng bảng để xem thông kê lưu lượng truy cập hay dữ liệu đặc trưng như các trang được truy cập nhiều nhất rất khó để có cái nhìn tổng quan, vì vậy cần sử dụng biểu đồ để biểu diễn các phân tích này



Hình 2.10. Một số dạng biểu đồ hay dùng biểu diễn log

* Để thuận tiện cho việc đọc và phân tích nhanh, ứng dụng cũng sẽ hướng tới cung cấp việc ghi dữ liệu ra các file txt đã được phân loại cụ thể như login.txt, info.txt

Ngoài ra, một số chức năng sẽ sử dụng giao diện terminal để tăng hiệu xuất làm việc như AI phân tích log

* 1. **Kết luận chương 2**

Phân tích thiết kế là một trong các bước quan trọng trong quy trình phát triển xây dựng phần mềm ứng dụng. Dựa vào các thiết kế, ta dễ dàng hình dung và định hướng được phương hướng xây dựng ứng dụng.

Tiếp theo của đồ án sẽ là xây dựng ứng dụng dựa trên những gì đã lên ý tưởng ở chương 2, bên cạnh đó là những ý tưởng được tham khảo trong quá trình sử dụng các công cụ phân tích ứng dụng khác.

# **CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG**

Ứng dụng được xây dựng tích hợp với django nên dựa vào các ưu thế hỗ trợ mạnh mẽ của Django.

## **Yêu cầu đối với ứng dụng**

Sau chương 2, ta dã định được các phương hướng để xây dựng ứng dụng. Nhiệm vụ được đặt ra là phải đảm bảo xây dựng và hoạt động thành công được các chức năng chính. Trong đó quan trọng nhất là việc phải ghi lại được log, thông kê được số lượng người truy cập, địa chỉ truy cập, phần mềm duyệt web, mô tả được hành vi người dùng.

## **Xây dựng các chức năng**

Dựa vào các kỹ thuật và công nghệ đã được giới thiệu ở mục 1.3 của chương 1, ta tiến hành xây dựng các chức năng cho ứng dụng như sau:

* + - Ghi log (logging)
    - Giám sát log thời gian thực (real-time log monitoring)
    - Thống kê log (statitic log)
    - Xác định hành vi người dùng (determine user behavior)

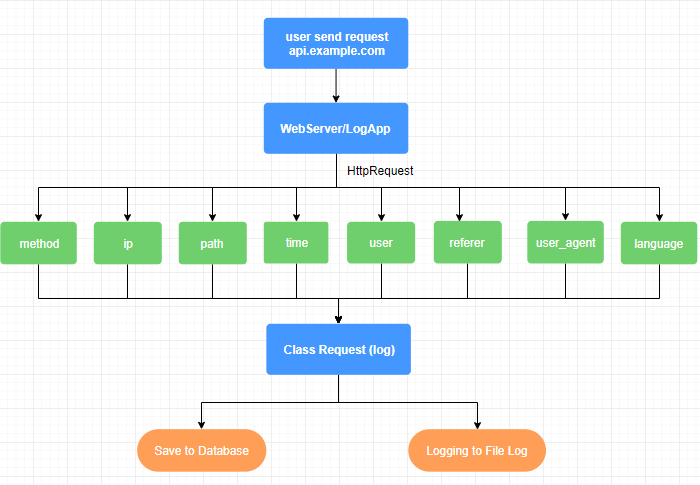
### *3.2.1. Ghi log (logging)*

Đầu tiên, bản chất của ứng dụng là dựa vào các ưu điểm của framework django để tích hợp và sử dụng tài nguyên đã được cấu hình sẵn khi người xây dựng hệ thống đã cài đặt cho website.

Trong django, mọi request gửi tới server được khai báo trong class HttpRequest. Lợi dụng class này, ta có thể bắt request tới server và lấy các thông tin trong đó và lưu lại làm log xuống cơ sở dữ liệu mà website đang sử dụng.

Lý do phải làm điều này vì trong format log của python đã giới thiệu ở mục 1.1.4. Thu thập log đề cập tới với format log trong module logging không thể lấy được các thông tin như ip, user truy cập mà phải trích xuất từ HttpRequest của django.

Theo hình 2.3 ở mục 2.2. Thiết kế dữ liệu, ta tiến hành xây dựng model để lưu log vào cơ sở dữ liệu:



Hình 3.1. Mô tả quá trình ghi và lưu log

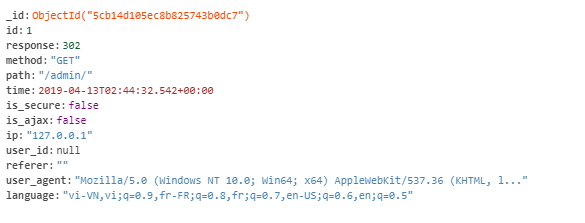
Dưới đây là đoạn code mô tả thành phần trong class Request:

1. response = models.SmallIntegerField(\_('response'), choices=HTTP\_STATUS\_CODES, default=200)
3. # Request information.
4. method = models.CharField(\_('method'), default='GET', max\_length=7)
5. path = models.CharField(\_('path'), max\_length=255)
6. time = models.DateTimeField(\_('time'), default=timezone.now, db\_index=True)
7. is\_secure = models.BooleanField(\_('is secure'), default=False)
8. is\_ajax = models.BooleanField(\_('is ajax'), default=False)
10. # User information.
11. ip = models.GenericIPAddressField(\_('ip address'))
12. user = models.ForeignKey(AUTH\_USER\_MODEL,blank=True,null=True,verbose\_name=\_('user'),on\_delete=models.SET\_NULL)
13. referer = models.URLField(\_('referer'), max\_length=255, blank=True, null=True)
14. user\_agent = models.CharField(\_('user agent'), max\_length=255, blank=True, null=True)
15. language = models.CharField(\_('language'), max\_length=255, blank=True, null=True)

Để lấy dược dữ liệu tương ứng ta xây dựng hàm trích dữ liệu từ HttpRequest được mô ta bởi đoạn code dưới đây:

1. **def** from\_http\_request(self, request, response=None, commit=True):
2. # Request information.
3. self.method = request.method
4. self.path = request.path[:255]
6. self.is\_secure = request.is\_secure()
7. self.is\_ajax = request.is\_ajax()
9. # User information.
10. self.ip = request.META.get('REMOTE\_ADDR', '')
11. self.referer = request.META.get('HTTP\_REFERER', '')[:255]
12. self.user\_agent = request.META.get('HTTP\_USER\_AGENT', '')[:255]
13. self.language = request.META.get('HTTP\_ACCEPT\_LANGUAGE', '')[:255]
15. **if** hasattr(request, 'user') **and** hasattr(request.user, 'is\_authenticated'):
16. is\_authenticated = request.user.is\_authenticated
17. **if** is\_authenticated:
18. self.user = request.user
20. **if** response:
21. self.response = response.status\_code
23. **if** (response.status\_code == 301) **or** (response.status\_code == 302):
24. self.redirect = response['Location']
26. **if** commit:
27. self.save()

Code đầy đủ minh họa cho việc lưu log xuống cơ sở dữ liệu được giới thiệu ở phụ lục 1. Khi lưu thành công, trong cơ sở dữ liệu các log đã được lưu vào với các trường được khai báo rõ ràng, riêng biệt.



Hình 3.2. Log khi được lưu dưới database

Sau khi xây dựng xong model để lưu log tiến hành cấu hình ghi log. Trong django, các cấu hình của hệ thống sẽ được lưu ở file setting.py. Theo mô tả trong document của django cung cấp cho người dùng, để cài đặt logging trong setting file cần khai báo như sau:

1. LOGGING = {
2. 'version': 1,
3. 'disable\_existing\_loggers': False,
4. 'formatters': {
5. 'verbose': {
6. 'format': '{levelname} - {asctime} - {message}',
7. 'style': '{',
8. },
9. },
10. 'handlers': {
11. 'file': {
12. 'class': 'logging.FileHandler',
13. 'filename': 'debug.log',
14. 'formatter': 'verbose'
15. },
16. },
17. 'loggers': {
18. 'django': {
19. 'handlers': ['file'],
20. 'level': 'DEBUG',
21. 'propagate': True,
22. },
23. },
24. }

Chú ý ở phần 'format': '{levelname} - {asctime} - {message}' là phần có thể tùy chỉnh theo nhu cầu ghi log của từng người (hướng tới khả năng đa biến cho mỗi người dùng), 'filename': 'debug.log' là phần khai báo thư mục sẽ ghi log vào đó. Kết quả ta sẽ được một file log có chứa các log dưới dạng như sau:

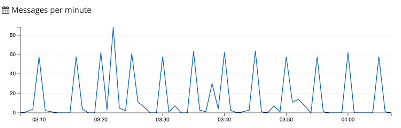
1. DEBUG - 2019-04-10 15:54:5 - Apps ready\_event triggered
2. DEBUG - 2019-04-10 15:54:56 - Watching dir D:\DOAN\DataBadminton\locale

Chức năng đầu tiên cũng là chức năng quan trọng nhất, sau khi có thể lưu và ghi log vào file log, ta có thể sử dụng để tiến hành xây dựng các chức năng tiếp theo.

### *3.2.2. Giám sát log thời gian thực (real-time log monitoring)*

Giám sát thời gian thực là việc đọc log liên tục sau khi vừa ghi log vào file log. Việc giám sát thời gian thực sẽ có một độ trễ nhất định vì phải mất một thời gian ngắn để đọc mà trích xuất các thông tin từ log rồi hiển thị.

Quá trình đó được minh họa như sau:



INFO - 2019-04-10 16:01:28,459 - "POST /Badminton/accounts/login/ HTTP/1.1" 200 10501

Read log

Level

INFO

Timestamp

2019-04-10 16:01:28,459

Message

"POST /Badminton/accounts/login/ HTTP/1.1" 200 10501

Hình 3.3. Sơ đồ minh họa quá trình đọc real-time log

Dưới đây là thuật toán trích xuất log, đếm lượt truy cập và tiến hành vẽ đồ thị biểu diễn theo thời gian thực:

1. **while** True:
2. # read next log
3. line = f.readline()
4. **if** line == '':
5. y\_vec[-1] = 0
6. # draw 0 when no log input
7. line1 = live\_plotter(x\_vec,y\_vec,line1)
8. y\_vec = np.append(y\_vec[1:],0.0)
9. **else**:
10. # split log to format
11. line = line.split(' - ')
12. #get timestamp
13. acstime= re.findall(r"\d\d\d\d-\d\d-\d\d\ \d\d:\d\d:\d\d",line[1])
14. #get message
15. message = re.findall(r"(+.)",line[2])
16. **if** acstime == asctimne\_pre:
17. #count message by timestamp
18. count +=1
19. **else**:
20. asctimne\_pre = acstime
21. y\_vec[-1] = count
22. #draw realtime log by count and timestamp
23. line1 = live\_plotter(x\_vec,y\_vec,line1)
24. y\_vec = np.append(y\_vec[1:],0.0)
25. count = 0

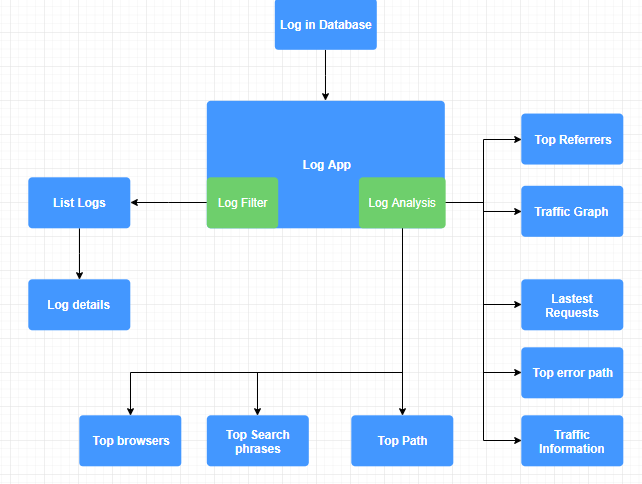
Ngoài ra, phụ lục 2 minh họa chi tiết thuật toán hàm lấy log thời gian thực.

Đối với việc vẽ đồ thị biểu diễn lưu lượng truy cập thì sử dụng thư viện đồ thì matplotlib được giới thiệu ở mục 1.1.3 được minh họa đầy đủ ở phụ lục 3.

### *3.2.3. Thống kê log (statistic log)*

Thống kê log là tổng hợp lại tất cả log cung cấp số liệu về tất cả dữ liệu log đã được ghi lại trong file log, hạn chế của chức năng này là không thể cho thấy được các diễn biến đang diễn ra trong hệ thống nhưng nó mang lại được cái nhìn tổng quát về tất cả những gì đã diễn ra từ trước tới nay được ghi lại trong file log.

Ở phần 3.2.1. Ghi log, ta đã xây dựng được việc lưu log xuống cơ sở dữ liệu, điều này vô cùng quan trọng vì với một số lượng rất lớn log có thể được coi như là dữ liệu lớn (big data) mà ngày nay với sự phát triển của các NoSQL đã cũng cung cấp các thuật toán để tối ưu việc truy vấn lấy dữ liệu lên. Nhờ đó tối ưu được khả năng xử lý với lượng log lớn, dễ dàng xây dựng các chức năng.



Hình 3.4. Các chức năng phân tích được từ log

Hình 3.4. mô tả tổng quan các thông tin thống kê được từ file log, tiếp theo tiến hành xây dựng các hàm thực hiện các chức năng trên.

Các chức năng ở phần thống kê sẽ được xây dựng tích hợp kèm vào django admin để tiện cho người dùng thay vì sử dụng các thư viện riêng biệt để hiển thị như các chức năng giám sát log theo thời gian thực và xác định hành vi người dùng.

* List logs

Ở hình 3.2 ta thấy khi lưu xuống cơ sở dữ liệu các dữ liệu đã được lưu vào các biến tương ứng (có thể xem phụ lục 1 để rõ hơn) rồi mới lưu vào cơ sở dữ liệu. Vì vậy chỉ cần một câu query là có thể lấy được các dữ liệu đó và gắn lại vào các biến tương ứng, trong django cũng cung cấp sẵn các thư viện cho việc lấy dữ liệu lên, ở đây ta dùng hàm:

get\_queryset(table\_database)

với table\_database là tên bảng chứa các log

Dưới đây là code minh họa việc lấy dữ liệu rồi gán lại vào các biến để hiển thị cho người dùng:

1. list\_display = ('time', 'path', 'response', 'method', 'request\_from')
2. fieldsets = (
3. (\_('Request'), {
4. 'fields': ('method', 'path', 'time', 'is\_secure', 'is\_ajax')
5. }),
6. (\_('Response'), {
7. 'fields': ('response',)
8. }),
9. (\_('User info'), {
10. 'fields': ('referer', 'user\_agent', 'ip', 'user', 'language')
11. })
12. )
13. raw\_id\_fields = ('user',)
14. readonly\_fields = ('time',)
16. **def** get\_queryset(self, request):
17. **return** super(RequestAdmin, self).get\_queryset(request).select\_related('user')
19. **def** request\_from(self, obj):
20. **if** obj.user\_id:
21. **return** format\_html(
22. '<a href="?user={0}" title="{1}">{2}</a>',
23. obj.user\_id,
24. \_('Show only requests from this user.'),
25. obj.user,
26. )
27. **return** format\_html(
28. '<a href="?ip={0}" title="{1}">{0}</a>',
29. obj.ip,
30. \_('Show only requests from this IP address.'),
31. )
32. request\_from.short\_description = 'From'
33. request\_from.allow\_tags = True

Biến list\_display chứa danh sách tất cả các log mà chúng ta cần để hiển thị

* Log details

Khác với list log (cung cấp khái quát tất cả các log được ghi lại trước giờ) thì log detail sẽ cung cấp chi tiết về các thành phần có trong một log.

1. fieldsets = (
2. (\_('Request'), {
3. 'fields': ('method', 'path', 'time', 'is\_secure', 'is\_ajax')
4. }),
5. (\_('Response'), {
6. 'fields': ('response',)
7. }),
8. (\_('User info'), {
9. 'fields': ('referer', 'user\_agent', 'ip', 'user', 'language')
10. })
11. )

Biến fieldsets trong hàm mình họa ở list log sẽ chứa đây đủ chi tiết các thông tin log. Khi người dùng cần xem log chi tiết thì chỉ cần trả về fieldsets.

* Traffic graph

Traffic graph có nhiệm vụ tổng hợp lại các log ghi nhận truy cập vào hệ thống, thống kê lại dựa trên các nhãn thời gian. Tương tự như giám sát log theo thời gian thực nhưng khác ở chỗ, tất cả các log được đếm theo một khoảng cách thời gian lớn hơn, cụ thể ở đây ta chọn theo tháng, còn giám sát thời gian thực lại đếm theo mốc thời gian tính bằng giây. Sự khác biệt này là bởi vì lượng log quá lớn cần phải lựa chọn thời gian phù hợp với hai tiêu chí một là đảm bảo cung cấp được cái nhìn khái quát đủ rộng cần thiết, hai là đảm bảo không bị rộng quá dễ dẫn tới làm lượng thông tin không rõ ràng.

Để có thể biểu diễn dưới dạng độ thị, trước hết các tách các log theo khoảng cách các tháng sau đó đếm và chuyển chúng thành dạng đồ thị 2 chiều (x,y) với x là mốc thời gian, y là lưu lượng.

Đầu tiên là hàm tính các mốc thời gian đồng thời xác định các log nằm trong khoảng thời gian đó:

1. **if** days\_count < 10:
2. days\_step = 1
3. **elif** days\_count < 60:
4. days\_step = 2
5. **else**:
6. days\_step = 30
8. days = [date.today() - timedelta(day) **for** day **in** range(0, days\_count + 1, days\_step)]
9. days\_qs = [(day, Request.objects.day(date=day)) **for** day **in** days]

Request.objects.day(date=day) sẽ truy vấn lấy các log dưới cơ sở dữ liệu theo các mốc days sau đó tạo thành list vào biến days\_qs.

Ở đây ta không dùng thư viện matplotlib để vẽ đồ thị biểu diễn lưu lượng truy cập nữa, mà sử dụng framework vuejs để phát triển giao diện hiển thị trên web sẽ được giới thiệu ở mục 3.3.

1. **def** graph(self, days):
2. **return** tuple([{
3. 'data': [(mktime(day.timetuple()) \* 1000, module.count(qs))  **for** day, qs **in** days],
4. 'label': str(ugettext(module.verbose\_name\_plural)),
5. } **for** module **in** self.modules])

Dữ liệu sẽ về các thành phần tọa độ sẽ được lưu lại ở tuple và trả lại phía font-end.

* Traffic information

Ngoài việc thể hiện lưu lượng truy cập, ta cũng sẽ xây dựng hàm thông kê lại số lượng người truy cập để phân loại người dùng hệ thống, người dùng vãng lai trong số lưu lượng đó.

Để làm được điều này ta chỉ cần sử dụng truy vấn vào cột user, is\_ajax và is\_secure (xem lại hình 3.1 để rõ hơn) để lấy dữ liệu.

Ví dụ: get\_queryset(Request).filter(is\_secure=True).count()

Chi tiết về các hàm được minh họa ở phụ lục 4.

* Log information

Tương tự như chức năng traffic information, log information là tập hợp các thông kê về phương tiện được sử dụng để truy cập vào trang web (browsers), request gần đây nhất, nội dung trang web được truy cập (path) ,…

Các thống kê này cũng chỉ cần sử dụng các truy vấn đơn giản để có thể lấy được thông tin cần thiết rất hiệu quả nhờ các thư viện được xây dựng sẵn trong django:

1. **class** LatestRequests(Plugin):
2. **def** template\_context(self):
3. **return** {'requests': Request.objects.all()[:5]}

6. **class** TrafficInformation(Plugin):
7. **def** template\_context(self):
8. INFO\_TABLE = ('today', 'this\_week', 'this\_month', 'this\_year', 'all')
9. INFO\_TABLE\_QUERIES = [getattr(Request.objects, query, None)() **for** query **in** INFO\_TABLE]
11. **return** {
12. 'traffic': modules.table(INFO\_TABLE\_QUERIES)
13. }

16. **class** TopPaths(Plugin):
17. **def** queryset(self):
18. **return** self.qs.filter(response\_\_lt=400)
20. **def** template\_context(self):
21. **return** {
22. 'paths': self.queryset().values('path').annotate(Count('path')).order\_by('-path\_\_count')[:10]
23. }

26. **class** TopErrorPaths(TopPaths):
27. template = 'request/plugins/toppaths.html'
29. **def** queryset(self):
30. **return** self.qs.filter(response\_\_gte=400)

33. **class** TopReferrers(Plugin):
34. **def** queryset(self):
35. **return** self.qs.unique\_visits().exclude(referer='')
37. **def** template\_context(self):
38. **return** {
39. 'referrers': self.queryset().values('referer').annotate(Count('referer')).order\_by('-referer\_\_count')[:10]
40. }

43. **class** TopSearchPhrases(Plugin):
44. **def** template\_context(self):
45. **return** {
46. 'phrases': set\_count(self.qs.search().only('referer').attr\_list('keywords'))[:10]
47. }

50. **class** TopBrowsers(Plugin):
51. **def** template\_context(self):
52. **return** {
53. 'browsers': set\_count(self.qs.only('user\_agent').attr\_list('browser'))[:5]
54. }

Sau khi truy vẫn thành công sử dụng thuật toán prefixspan (phần II chương 1 mục 1.3.1) để xác định tần số:

1. **def** set\_count(items):
2. '''''
3. This is similar to "set", but this just creates a list with values.
4. The list will be ordered from most frequent down.
5. Example:
6. >>> inventory = ['apple', 'lemon', 'apple', 'orange', 'lemon', 'lemon']
7. >>> set\_count(inventory)
8. [('lemon', 3), ('apple', 2), ('orange', 1)]
9. '''
10. item\_count = {}
11. **for** item **in** items:
12. **if** **not** item:
13. **continue**
14. **if** item **not** **in** item\_count:
15. item\_count[item] = 0
16. item\_count[item] += 1
18. items = [(v, k) **for** k, v **in** item\_count.items()]
19. items.sort()
20. items.reverse()
22. **return** [(k, v) **for** v, k **in** items]

Thông kê log là tiền đề để có thể xác định hành vi người dùng, bên cạnh đó là giúp người dùng có thể phân tích log, tất nhiên mức độ thông tin khai thác được còn tuy thuộc vào khả năng của người dùng, chi tiết về các hàm ở phần này được minh họa đầy đủ ở phụ lục 5.

### *3.2.4. Xác định hành vi người dùng (determine user behavior)*

Một file log của một hệ thống website còn được coi là danh sách tham chiếu tới các trang dữ liệu bởi vì mỗi một log được ghi lại còn tương đương với một lần nhấp chuột vào một phần vào đó của website. Điều này mang ý nghĩa rằng, thông qua những dòng log có thể hiểu rõ hơn về các tập tính hành vi của người dùng khi truy cập vào giúp hiểu rõ hơn về người dùng, phần trang web được quan tâm từ đó thu thập được mẫu dữ liệu hành vi tạo tiền đề để phát triển học máy phát hiện hành vi người dùng có mục đích xấu.

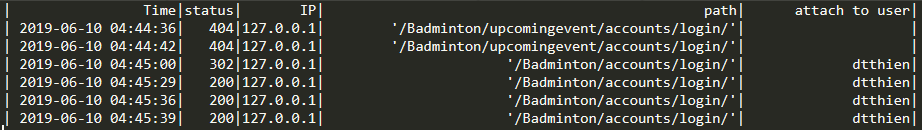
Phát hiện hành vi của người dùng và thực thể lệch lạc như đăng nhập vào một giờ bất thường, lỗi đăng nhập quá mức và xóa tệp khỏi máy chủ thường không được sử dụng bởi một người dùng cụ thể. Tuy nhiên, khi những hành động này được liên kết với nhau, chúng có thể kể một câu chuyện gắn kết và cung cấp bối cảnh bảo mật có ý nghĩa.

Ứng dụng ánh xạ phân biệt các tài khoản người dùng và các định danh liên quan để xây dựng đường cơ sở toàn diện cho hành vi của người dùng, có thể xác định hành vi bất thường của người dùng dựa trên thời gian, số lượng và các mẫu bất thường.

Trước khi có thể khai thác những điều này ta cần khai thác được các log để có thể tạo ra mẫu hành vi của người dùng cụ thể nào đó.

1. **if** 'INSERT INTO "request\_request"' **in** line **and** 'login' **in** line:
2. query = re.findall(r"args=\[(.+)\]",line)
3. **if** re.match(r"(datetime.+\)), F",query[0]):
4. time = re.findall(r"(datetime.+\)), F",query[0])[0]
5. times = re.findall(r"\((.+)\)",time)[0]
6. times = times.replace(' ',',').split(',,')
7. times = datetime(int(times[0]),int(times[1]),int(times[2]),int(times[3]),int(times[4]),int(times[5]))
8. query=query[0].replace(time,'time').split(',')
9. **else**:
10. times = re.findall(r"(\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d)", query[0])[0]
11. query=query[0].replace(times,'time').split(',')
12. status = query[0]
13. method = re.findall(r"(POST|GET|PUT|DELETE)",query[1])[0]
14. ip = re.findall(r"(\d+.\d+.\d+.\d+)",query[6])[0]
15. path = query[2]
16. user = query[9]
17. result = "|%20s|%10s|%20s|%100s|%50s|%20s|\n"%(times,status,str(ip),str(user),str(path),list\_usersname

Trên đây là thuật toán nhằm trích xuất các thông tin về các cuộc đăng nhập của người dùng vào trang web từ file log, các thông số cụ thể của thuật toán được cung cấp ở phụ lục 6 của đồ án. Ứng dụng sẽ thu thập các cuộc đăng nhập này để tạo ra các mẫu hành vi người dùng.



Hình 3.5. Kết quả thu thập mẫu hành vi đăng nhập

Nhờ vào việc thu thập các mẫu, ta có thể tiến hành đánh giá:

* Thời gian không thường xuyên: Một người dùng thường đăng nhập vào khoảng 9 giờ sáng đến 10 giờ sáng đột nhiên đăng nhập vào lúc 5 giờ sáng. Ứng dụng nhanh chóng xác định sự kiện này và đánh dấu nó là hành vi lệch lạc.
* Các mẫu bất thường: xác định các mẫu hành vi bình thường của người dùng. Cập nhật các mẫu này khi chúng thay đổi theo thời gian và thông báo cho quản trị viên khi người dùng đi chệch khỏi các mẫu này. Ví dụ: nếu người dùng thường đăng các bình luận lên trang web trong một ngày khoảng 3 hoặc 4 nhưng đột nhiên đăng lên 17, hành vi này sẽ được xác định là bất thường.
* Số lượng không thường xuyên: có thể chỉ định giá trị ngưỡng cho các sự kiện cụ thể, chẳng hạn như thay đổi mật khẩu người dùng. Ví dụ: nếu các log cho thấy rằng người dùng đã thực hiện hơn 10 lần đổi mật khẩu trong một khoản thời gian ngắn.

Hiện tại trong mức đồ nghiên cứu đồ án còn hạn chế nên chưa thể nghiên cứu ứng dụng học máy vào việc nhận dạng bất thường và đưa ra gợi ý hướng khắc phục.

## **Xây dựng giao diện**

Giao diện của ứng dụng được xây dựng dựa trên việc kết hợp thư viện django admin và framework vuejs để xây dựng giao diện hiển thị.

* VueJS

VueJS là một **progressive framework** được xây dựng dựa vào nền tảng của ngôn ngữ lập trình phía client Javascript, nhiệm vụ chính của VueJS là dùng để xây dựng giao diện phía người dùng (UI). Không giống với các Framework khác, VueJS chỉ tập trung vào phần giao diện người dùng (view) nên bạn hoàn toàn có thể tích hợp vào project một cách đơn giản.

Thư viện vue-graph là một thư viện được xây dựng trên nền vuejs chuyên dụng cho việc biểu diễn đồ thị. Thư viện cung cấp đầy đủ các hàm, các mẫu đồ thị để tích hợp, chỉ cần truyền dữ liệu vào, vue-graph hiển thị đầy đủ kết quả.

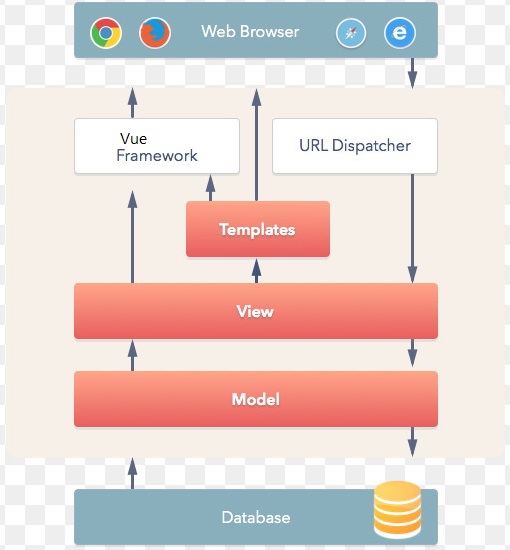
Một số dạng biểu đồ được sử dụng trong ứng dụng là:

* + - Line Chart
    - Circle Chart

Để sử dụng các biểu đồ ta truyền vào 2 tham số là labels ứng với khai báo dữ liệu trục hoành và values ứng với khai báo dữ liệu tung độ.

* Xây dựng giao diện ứng dụng

Dựa vào việc tham khảo các ứng dụng phân tích log khác như splunk, graylog. Tôi đã xây dựng giao diện tương tự như vậy.



Hình 3.6. Sơ đồ hoạt động django

Mỗi khi có request gửi tới, các hàm ở mục 3.2 chương 2 sẽ được gọi sau đó trả về dữ liệu sau đó được hiển thị lại cho người dùng

Trang đầu tiên, sẽ là trang hiển thị danh sách tất cả các log được lưu dưới cơ sở dữ liệu, vì các log được gắn vào các biến nên phù hợp sẽ là hiển thị dưới dạng bảng:



Hình 3.7. Bảng hiển thị danh sách log

Hàm list logs trả lại list\_display tương ứng với các dữ liệu được hiển thị.

Với trang overview hiển thị tất cả các thống kê thì phải khai báo thêm đoạn script để import các thư viện hỗ trợ vẽ đồ thị

1. <script type="text/javascript" src="{% static "request/js/jquery-1.4.2.min.js" %}"></script>
2. <script type="text/javascript" src="{% static "request/js/jquery.flot.js" %}"></script>
3. <script type="text/javascript" src="{% static "request/js/jquery.timeago.js" %}"></script>
4. <script type="text/javascript" src="{% static "request/js/request.js" %}"></script>
5. {% **if** request **and** request.user %}
6. <script type="text/javascript">
7. function loadTrafficGraph(days) {
8. **if** (days == undefined) {
9. path = "{% url "admin:request\_request\_traffic" %}";
10. } **else** {
11. path = "{% url "admin:request\_request\_traffic" %}?days=" + days;
12. }
13. $.getJSON(path, function(data) {
14. trafficGraph($("#trafficgraph"), data);
15. });
16. }
17. $(document).ready(function(){
18. loadTrafficGraph();
19. });
20. </script>
21. {% endif %}
22. {% endblock %}
24. {% block breadcrumbs %}
25. {% **if** request **and** request.user %}
26. <div **class**="breadcrumbs">
27. <a href="{% url "admin:index" %}">{% trans "Home" %}</a> ›
28. <a href="{% url "admin:app\_list" "request" %}">{% trans "Request" %}</a> ›
29. <a href="{% url "admin:request\_request\_changelist" %}">{% trans "Requests" %}</a> ›
30. {% trans "Overview" %}
31. </div>
32. {% endif %}
33. {% endblock %}
35. {% block content %}
36. <div style="width:920px;">
37. <div **class**="module" style="clear: both;">
38. <table style="width: 100%;">
39. <caption>{% trans "Traffic graph" %} ( <a href="javascript:loadTrafficGraph(7);">{% trans "Week" %}</a> | <a href="javascript:loadTrafficGraph(30);">{% trans "Month" %}</a> | <a href="javascript:loadTrafficGraph(365);">{% trans "Year" %}</a> )</caption>
40. </table>
42. <div style="padding: 15px;">
43. <div id="trafficgraph" style="width: 900px; height: 250px;"></div>
44. </div>
45. </div>
47. {% **for** plugin **in** plugins %}
48. <div **class**="module" style="float: left; width: 450px; {% cycle 'clear: both;' 'margin-left: 16px;' %}">
49. {{ plugin.render }}
50. </div>
51. {% endfor %}
52. </div>

Ngoài giao diện web thì ứng dụng còn cung cấp các chức năng sử dụng thư viện matplotlib để chạy độc lập với nền web, tiện lời để phân tích các log django ngoài. Để xem các hàm hiển thị như thế nào có thể xem các phụ lục

## **Kết luận chương 3**

Ứng dụng dựa trên các thiết kế ở chương 2 đã được xây dựng thành công, đáp ứng yêu cầu đặt ra là mô tả chi tiết các bước của các mô hình. Ngoài ra, vì thời gian nghiên cứu còn hạn chế và nhiều công nghệ cũng chưa được tìm hiểu nên ứng dụng còn nhiều chức năng để xây dựng trong tương lại.

# **PHẦN III: TRIỂN KHAI ỨNG DỤNG**

1. **Môi trường thử nghiệm**
   1. *Cài đặt và cấu hình*

Ứng dụng được xây dựng với khả năng tích hợp vào một hệ thống website sử dụng django để xây dựng, nên để có thể chạy được ứng dụng cần có một website django.

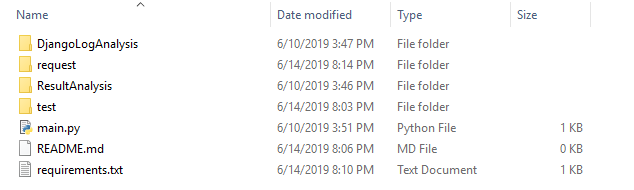
Để tải ứng dụng, đầu tiên truy cập github để tải ứng dụng được upload:

<https://github.com/KunkkaVN/App-Log-Django/>

hoặc dùng lệnh.

1. git clone https://github.com/KunkkaVN/App-Log-Django/

Vì ở đây là phần triển khai thử nghiệm, nên tôi đã có sẵn ứng dụng trên máy, khi tải xog ứng dụng ta sẽ có được các folder sau:



Hình 4.1. Các file có trong ứng dụng

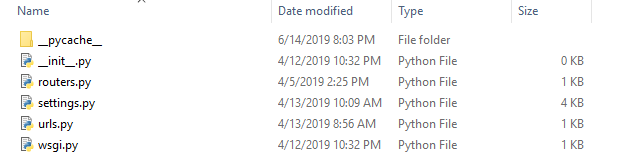
Môi trường thử nghiệm chương trình sẽ là trên nên windows 10, python 3.6. Trong thư mục test có chưa folder DemoDoAn là một website xây dựng trên django 2.2 dùng để chạy thử nghiệm ứng dụng.

Để cài đặt ứng dụng, tới thư mực gốc có chưa file requirements gõ lệnh ở cmd:

1. pip install -r requirements.txt

Sau khi cài đặt xong các thư viện cần thiết để chạy ứng dụng. Tiếp theo, tiến hành cấu hình cho ứng dụng để phù hợp với hệ thống đang chạy.

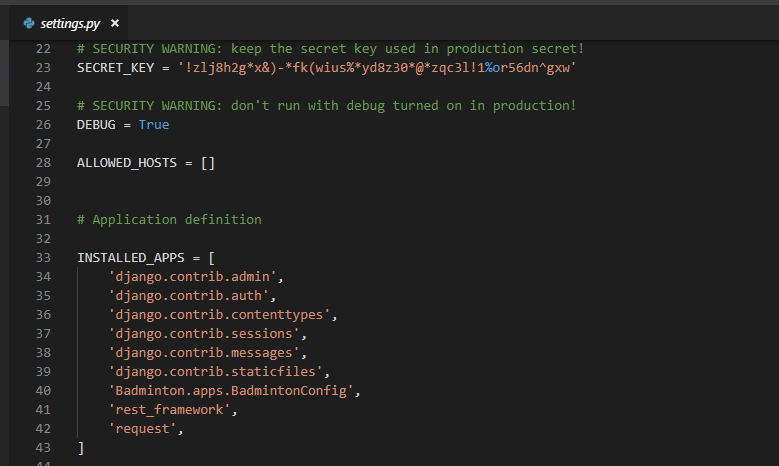
Nếu bạn có hệ thống website django thì có thể vào file setting.py ở phía server backend website để cấu hình bước tiếp theo sau đây, nếu không có website thì tiến hành vào folder test khi tải ứng dụng tiếp tục vào folder DemoDoAn => DemoDoAn, bạn sẽ được các thư mực sau:



Hình 4.2. Các file cơ bản của một website dùng django

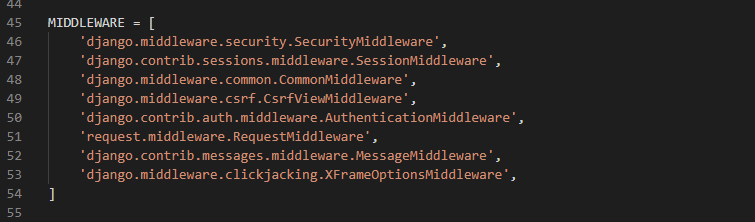
Tiến hành cấu hình log ở file setting.py:

* Thêm request vào trong INSTALLED\_APPS của file setting



Hình 4.3. Cấu hình log trong setting.py(1)

* Thêm request.middleware.RequestMiddleware vào phần MIDDLEWARE .Chú ý: nếu sử dụng django.contrib.auth.middleware.AuthenticationMiddleware thì đặt RequestMiddleware sau nó còn nếu sử dụng django.contrib.flatpages.middleware.FlatpageFallbackMiddleware thì đặt request.middleware.RequestMiddleware phía trước



Hình 4.4. Cấu hình log trong setting.py(2)

* Đảm bảo trang admin của django hoạt động bình thường
* Cấu hình xuất log:

LOGGING = {

    'version': 1,

    'disable\_existing\_loggers': False,

    'formatters': {

        'verbose': {

            'format': '{levelname} - {asctime} - {message}',

            'style': '{',

        },

    },

    'handlers': {

        'file': {

            'class': 'logging.FileHandler',

            'filename': 'debug.log',

            'formatter': 'verbose'

        },

    },

    'loggers': {

        'django': {

            'handlers': ['file'],

            'level': 'DEBUG',

            'propagate': True,

        },

    },

}

* Chú ý là ở phần format có thể thay đổi tùy theo nhu cầu của bản thân, khi cấu hình xong file setting.py sẽ như sau



Hình 4.5. Cấu hình log trong setting.py(3)

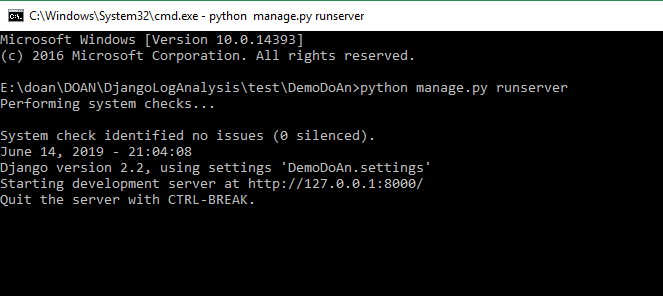
* Sau khi cấu hình xong, chạy lệnh python manage.py migrate để hoàn tất cấu hình.
  1. *Chạy thử nghiệm*

Đầu tiên cần tạo 1 superuser admin để có thể truy cập vào trang admin của django:

1. python manager.py createsuperuser

Tiến hành điền các thông tin về tài khoản. Tiếp theo tiến hành chạy server:

1. python manager.py runserver

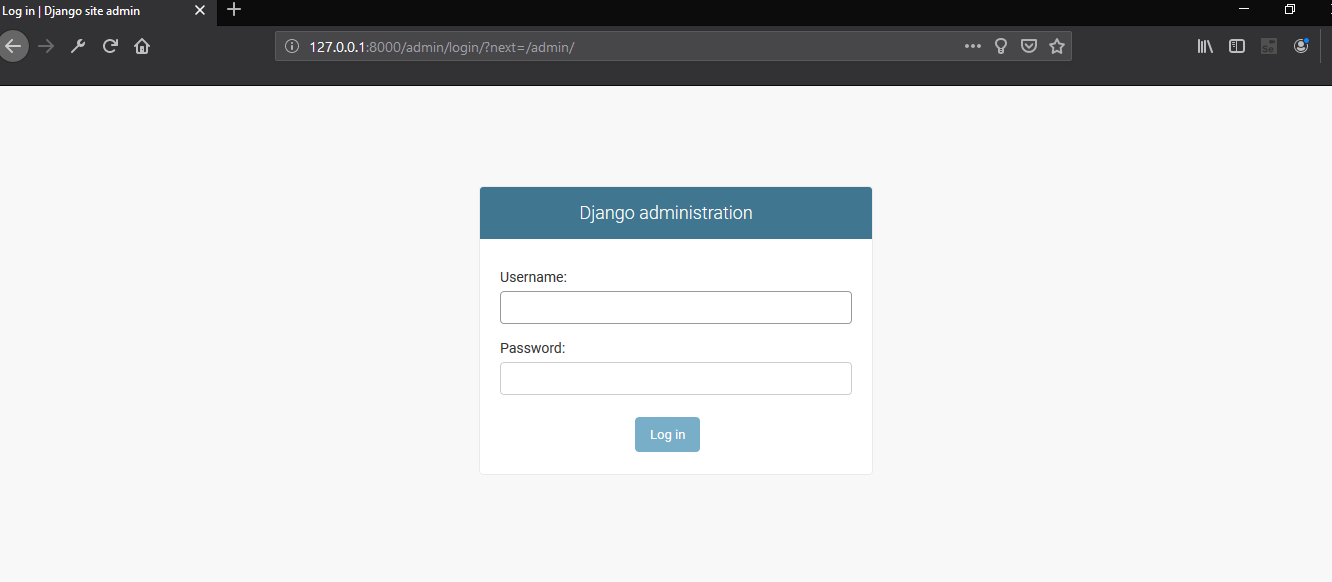
****

Hình 4 6. Thông báo cấu hình thành công

Khi cài đặt và cấu hình thành công, chạy server sẽ có hình như trên.

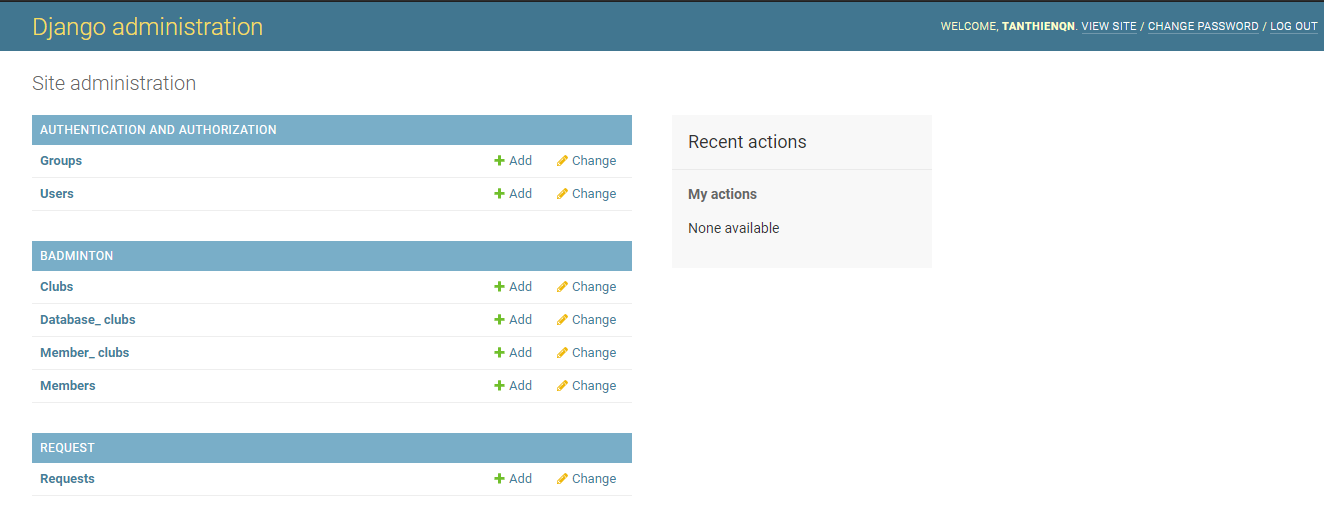
1. **Kết quả thử nghiệm**

Sau khi cài đặt và chạy server thành công, tiến hành đăng nhập vào trang admin của django-admin được mặc định ở url http://127.0.0.0.1:8000/admin :



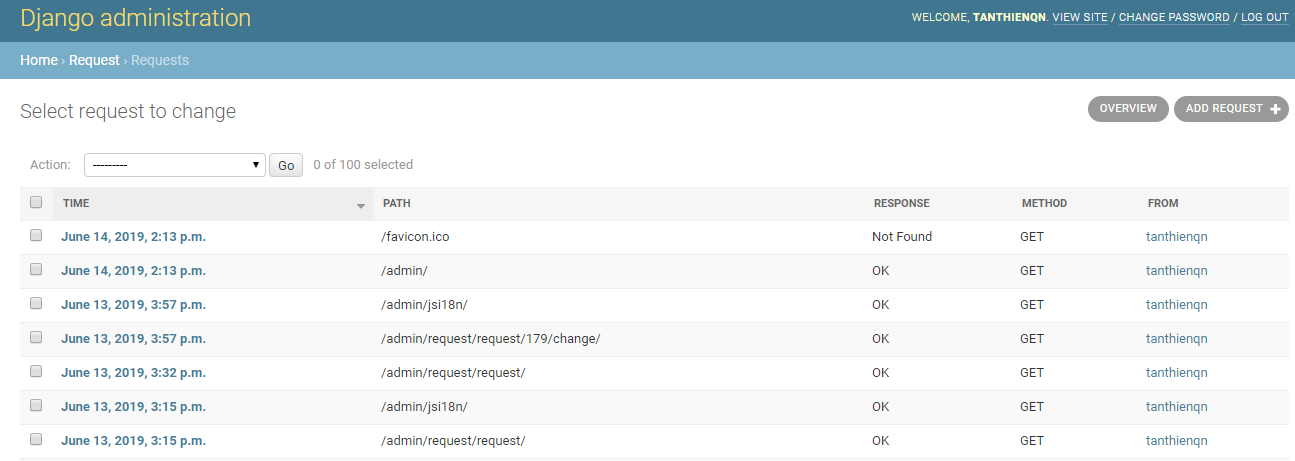
Hình 4.7. Giao diện đăng nhập ứng dụng

Vì ứng dụng được tích hợp sử dụng chung tài nguyên với hệ thống website nên khi đăng nhập vào bạn sẽ thấy các database đang có ở website:



Hình 4.8. Giao diện khi đăng nhập thành công

Như đã giới thiệu ở mục 3.2.1 chương 3 phần II, các log ghi được sẽ được lưu xuống database ở table Requests, click vào bảng Requests ta sẽ xem được tất cả các log được lưu:

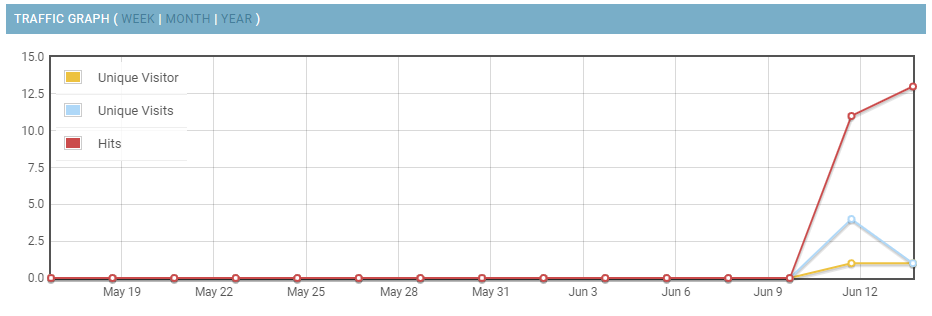


Hình 4.9. Tổng hợp các log được lưu

Tiếp theo sẽ đến kiểm tra chức năng thông kê log đã được xây dựng trước đó. Để xem các thông kê chọn vào nút OVERVIEW.

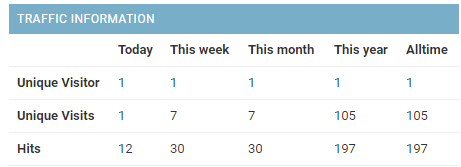
Dưới đây là các kết quả mà ứng dụng đã thông kê được thành công:

* Traffic graph:



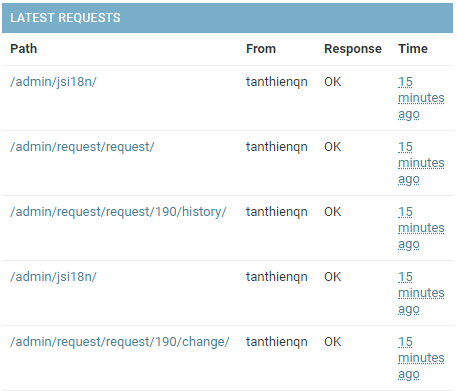
Hình 4.10. Thông kê log traffic graph

* Traffic information:



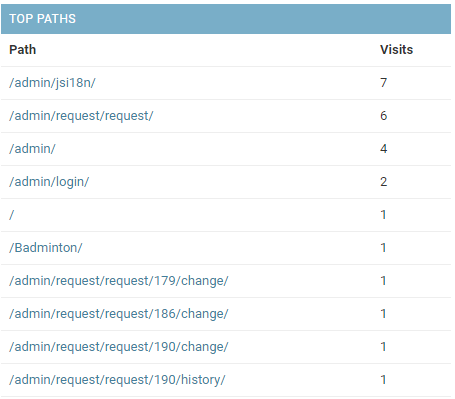
Hình 4.11. Thông kê log traffic information

* Lastest requests:



Hình 4. 12. Thống kê log lastest requests

* Top paths:



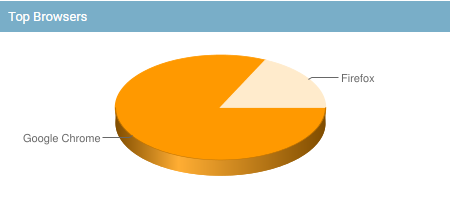
Hình 4.13. Thống kê log top paths

* Top error paths:



Hình 4.14. Thống kê log top error paths

* Top browsers:



Hình 4.15. Thông kê log top browsers

Khác với chức năng thống kê log, chức năng giám sát log theo thời gian thực và thống kê log được xây dựng để chạy riêng biệt chứ không tích hợp vào website để có thể đọc file log nhanh.

Như ở hình 4.1. ta thấy có file main.py, đây là hàm main gọi các hàm con, tại cmd ở đây gõ lệnh:

python main.py --m <mode> --p <path>

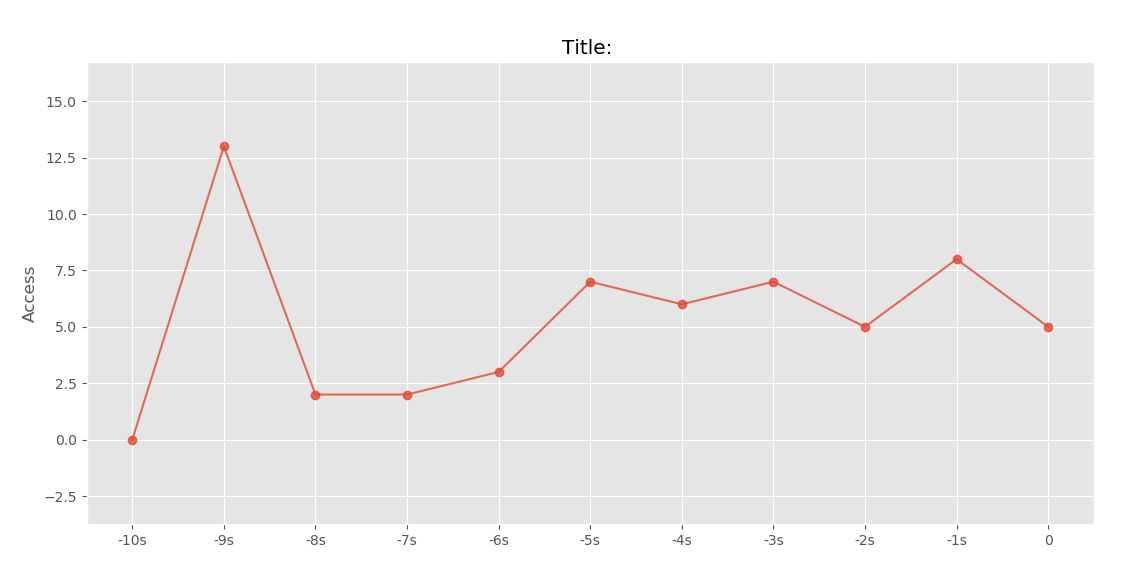
với : mode là chế độ sử dụng realtime hoặc behavior

path là đường dẫn tới file chứa file log

* Realtime log:

python main.py –m realtime --p ..\test\DemoDoAn\debug.log

Với cú pháp này sẽ sử dụng chế độ giám sát thời gian thực

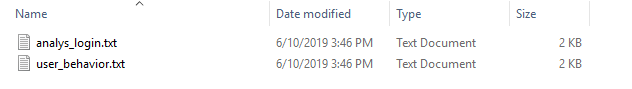


Hình 4.16. Realtime log

* User behavior:

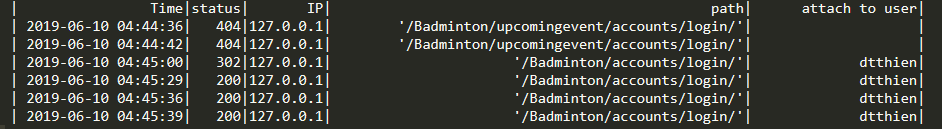
python main.py –m behavior --p ..\test\DemoDoAn\debug.log

Với cú pháp này sẽ sử dụng chế độ thông kê lại các hành vi người dùng, kết quả sẽ được tạo ra các file kết quả trong thư mực ResultAnalysis

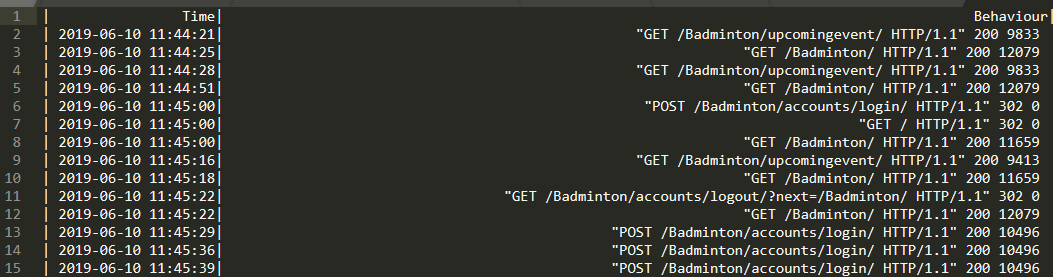


Hình 4.17. Thư mục chứa kết quả hành vi người dùng

Ở đây ta thấy có 2 file kết quả là analys\_login.txt và user\_behavior.txt. Trong đó file analys\_login.txt là nơi chứa tất cả các thông tin về các cuộc đăng nhập còn user\_behavior là tập hợp tất cả các hành động mà user thực hiện.



Hình 4.18. Kết quả thu thập mẫu đăng nhập



Hình 4. 19. Kết quả thu thập hành vi người dùng

Trên đây là tất cả các chức năng đã được thực nghiệm sau khi xây dựng ở chương 3 phần II.

1. **Đánh giá kết quả thử nghiệm**

Qua kết quả chạy thử nghiệm ta thấy, các chức năng đã chạy thành công mà không xảy ra vấn đề gì. Tuy nhiên vì sử dùng sẵn bản demo là website được xây dựng nên dữ liệu truy cập không cao và phức tạp như trong thực tế.

Từ kết quả cũng có thể rút ra một số ưu và nhược điểm của ứng dụng

* 1. *Ưu điểm:*

Một số ưu điểm được rút ra từ ứng dụng:

* + - Tích hợp dễ dàng vào website, không gây khó khăn khi cấu hình.
    - Sử dụng các tài nguyên có sẵn.
    - Sử dụng NoSQL để lưu trữ log, phù hợp với việc ghi dữ liệu cực lớn.
    - Giao điện đơn giản, dễ nhìn.
    - Các chức năng thống kê được các thông tin trích xuất được từ log.
    - Thu thập được hành vi người dùng, đặc biệt là việc đăng nhập hệ thống.
  1. *Nhược điểm*

Bên cạnh những ưu điểm, ứng dụng cũng còn tồn tại một số khuyết điểm như:

* + - Sử dụng tài nguyên của website nên dẫn tới nếu không được phân bố tài nguyên cụ thể sẽ dẫn tới thiếu hụt tài nguyên.
    - Mới chỉ áp dụng được cho các hệ thống website django.
    - Cần phải phát triển AI sao cho phát hiện chính xác hành vi bất thường.

1. **Phương hướng phát triển**

Một hệ thống website lớn phải đối mặt với rất nhiều vấn đề như lượng truy cập lớn, các nội dung hay thảo luận đăng lên hang ngày rất nhiều dẫn tới lượng thông tin ghi log phức tạp và vô cùng nhiều.

Điều này dẫn tới ta không thể đơn giản mà dựa vào bản thân để phân tích hết được chúng, một trong những giải pháp hang đầu hiện này là ứng dụng học máy (machine learning) vào việc tự động phân tích dựa trên việc học các mẫu dữ liệu log trước đó.

Như đã đề cập ở mục 3.2.4 chương 3 phần II, vì thời gian nghiên cứu đồ án có hạn nên hiện tại chưa thể tích hợp AI (trí tuệ nhân tạo) vào cho chức năng xác định hành vi người dùng mà chỉ mới thu thập được các hành vi để tạo mẫu hành vi.

Hiện tại, nhờ sự phát triển mạnh mẽ của công nghệ mà các thư viện machine learning rất phong phú và genism là một trong số đó. Gensim là một bộ thư viện machine learning được google phát triển và hậu thuẫn. Ứng dụng sẽ tiến tới sử dụng bộ thư viện này để phát triển các tinh năng phân tích nhờ AI mới mà trước hết là việc phân tích hành vi người dùng.

Ngoài ra, phương hướng phát triển tiếp theo cho ứng dụng là giới thiệu lên một số diễn đàng, github để mọi người sử dụng và cùng xây dựng cũng như sửa các tính năng chưa hợp lý.

1. **Kết luận phần III**

Sau phần III cũng là phần cuối cùng của đồ án, tôi đã triển khai thành công ứng dụng, đạt được một số kết quả đúng như mong đợi, mục tiêu ban đầu nghiên cứu.

Đồ án sử dụng vẫn còn được sử dụng ở website quy mô nhỏ, trong nội bộ chưa được thực nghiệm ở thực tế. Vì vậy trong thời gian sắp tới, ứng dụng sẽ tiến tới sử dụng cho các website thực tế.

# **KẾT LUẬN**

Các ứng dụng phân tích log ngày càng có tầm quan trọng trong việc quản lý an toàn thông tin, mọi người vẫn thường sử dụng chúng tách riêng biệt với ứng dụng web dẫn tới cần một lượng tài nguyên máy tính lớn để cài đặt và sử dụng, đồng thời không dễ để cẩu hình lại theo ý muốn của người dùng để thuận tiện cho việc phân tích.

Bên cạnh đó, việc ngày càng nhiều hệ thống website sử dụng framework django của python cũng tạo nên nhu cầu về các ứng dụng giúp đọc và phân tích các file log được ghi lại bởi django.Vì vậy đồ án đã hướng tới nhằm giải quyết được các vấn đề đã được đặt ra ở đầu nghiên cứu:

* Tìm hiểu, nghiên cứu các công nghệ sử dụng để phát triển ứng dụng web django, phương pháp để tích hợp một ứng dụng dựa vào django.
* Tìm hiểu, nghiên cứu về log, cách ghi log, đặt biệt là log trong django.
* Dựa vào kiến thức tìm hiểu để tiến hành thiết kế dữ liệu, các chức năng và giao diện cho ứng dụng.
* Tiến hành xây dựng ứng dụng đảm bảo hoạt động như đã thiết kế.
* Dựa vào kinh nghiệm trong quá trình làm đồ án và kinh nghiệm bản thân để hoàn thiện hơn ứng dụng.
* Nghiên cứu ứng dụng học máy để nhận diện các cuộc tấn công vào ứng dụng dựa trên các log được ghi.

# **TÀI LIỆU THAM KHẢO**

[1]. Abhinav Ajitsaria (Sep 12,2018) - “Logging in Python”

[2]. Logging HOWTO – Python 3.6.8 documentation

<https://docs.python.org/3.6/howto/logging.html>

[3]. Regular expression operations – Python 3.6.8 documentation

<https://docs.python.org/3.6/library/re.html>

[4].  DeLaRosa, Alexander (February 8, 2018) - [*"Log Monitoring: not the ugly sister"*](https://web.archive.org/web/20180214153657/https:/blog.pandorafms.org/log-monitoring/)

<https://web.archive.org/web/20180214153657/https://blog.pandorafms.org/log-monitoring/>

[5].  Pyplot tutorial – mathlotlib documentation

<https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html>

[6]. Padmaja & Sheshasaayee, Ananthi. (2014). WEB SERVER LOGS TO ANALYZING USER BEHAVIOR USING LOG ANALYZER TOOL. IJARSE. 3. 514-525.

# **Phụ lục 1: Function: logging**

Mã chương trình phần xây dựng dữ liệu

Model.py

1. # -\*- coding: utf-8 -\*-
2. **from** socket **import** gethostbyaddr
4. **from** django.conf **import** settings
5. **from** django.contrib.auth **import** get\_user\_model
6. **from** django.db **import** models
7. **from** django.utils **import** timezone
8. **from** django.utils.translation **import** ugettext\_lazy as \_
9. **from** six **import** python\_2\_unicode\_compatible
11. **from** . **import** settings as request\_settings
12. **from** .managers **import** RequestManager
13. **from** .utils **import** HTTP\_STATUS\_CODES, browsers, engines
15. AUTH\_USER\_MODEL = getattr(settings, 'AUTH\_USER\_MODEL', 'auth.User')

18. @python\_2\_unicode\_compatible
19. **class** Request(models.Model):
20. # Response information.
21. response = models.SmallIntegerField(\_('response'), choices=HTTP\_STATUS\_CODES, default=200)
23. # Request information.
24. method = models.CharField(\_('method'), default='GET', max\_length=7)
25. path = models.CharField(\_('path'), max\_length=255)
26. time = models.DateTimeField(\_('time'), default=timezone.now, db\_index=True)
28. is\_secure = models.BooleanField(\_('is secure'), default=False)
29. is\_ajax = models.BooleanField(
30. \_('is ajax'),
31. default=False,
32. help\_text=\_('Wheather this request was used via javascript.'),
33. )
35. # User information.
36. ip = models.GenericIPAddressField(\_('ip address'))
37. user = models.ForeignKey(AUTH\_USER\_MODEL, blank=True, null=True, verbose\_name=\_('user'), on\_delete=models.SET\_NULL)
38. referer = models.URLField(\_('referer'), max\_length=255, blank=True, null=True)
39. user\_agent = models.CharField(\_('user agent'), max\_length=255, blank=True, null=True)
40. language = models.CharField(\_('language'), max\_length=255, blank=True, null=True)
42. objects = RequestManager()
44. **class** Meta:
45. app\_label = 'request'
46. verbose\_name = \_('request')
47. verbose\_name\_plural = \_('requests')
48. ordering = ('-time',)
50. **def** \_\_str\_\_(self):
51. **return** '[{0}] {1} {2} {3}'.format(self.time, self.method, self.path, self.response)
53. **def** get\_user(self):
54. **return** get\_user\_model().objects.get(pk=self.user\_id)
56. **def** from\_http\_request(self, request, response=None, commit=True):
57. # Request information.
58. self.method = request.method
59. self.path = request.path[:255]
61. self.is\_secure = request.is\_secure()
62. self.is\_ajax = request.is\_ajax()
64. # User information.
65. self.ip = request.META.get('REMOTE\_ADDR', '')
66. self.referer = request.META.get('HTTP\_REFERER', '')[:255]
67. self.user\_agent = request.META.get('HTTP\_USER\_AGENT', '')[:255]
68. self.language = request.META.get('HTTP\_ACCEPT\_LANGUAGE', '')[:255]
70. **if** hasattr(request, 'user') **and** hasattr(request.user, 'is\_authenticated'):
71. is\_authenticated = request.user.is\_authenticated
72. **if** is\_authenticated:
73. self.user = request.user
75. **if** response:
76. self.response = response.status\_code
78. **if** (response.status\_code == 301) **or** (response.status\_code == 302):
79. self.redirect = response['Location']
81. **if** commit:
82. self.save()
84. @property
85. **def** browser(self):
86. **if** **not** self.user\_agent:
87. **return**
89. **if** **not** hasattr(self, '\_browser'):
90. self.\_browser = browsers.resolve(self.user\_agent)
91. **return** self.\_browser[0]
93. @property
94. **def** keywords(self):
95. **if** **not** self.referer:
96. **return**
98. **if** **not** hasattr(self, '\_keywords'):
99. self.\_keywords = engines.resolve(self.referer)
100. **if** self.\_keywords:
101. **return** ' '.join(self.\_keywords[1]['keywords'].split('+'))
103. @property
104. **def** hostname(self):
105. **try**:
106. **return** gethostbyaddr(self.ip)[0]
107. **except** Exception:  # socket.gaierror, socket.herror, etc
108. **return** self.ip
110. **def** save(self, \*args, \*\*kwargs):
111. **if** **not** request\_settings.LOG\_IP:
112. self.ip = request\_settings.IP\_DUMMY
113. **elif** request\_settings.ANONYMOUS\_IP:
114. parts = self.ip.split('.')[0:-1]
115. parts.append('1')
116. self.ip = '.'.join(parts)
117. **if** **not** request\_settings.LOG\_USER:
118. self.user = None
120. super(Request, self).save(\*args, \*\*kwargs)

# **Phụ lục 2: Function: realtime log**

Mã chương trình mô tả đầy đủ chức năng realtimelog

Realtime\_log.py

1. **import** time
2. **import** re
3. **from** DjangoLogAnalysis.livegraph **import** live\_plotter
4. **import** numpy as np
5. **def** main(path):
6. f = open(path, 'r')
7. count = 0
8. asctimne\_pre=''
9. line1=[]
10. x\_vec = ["-10s","-9s","-8s","-7s","-6s","-5s","-4s","-3s","-2s","-1s","0"]
11. y\_vec = [0,0,0,0,0,0,0,0,0,0,0]
12. **while** True:
13. line = f.readline()
14. **if** line =='':
15. y\_vec[-1] = 0
16. line1 = live\_plotter(x\_vec,y\_vec,line1)
17. y\_vec = np.append(y\_vec[1:],0.0)
18. **else**:
19. **try**:
20. line = line.split(' - ')
21. level= line[0]
22. acstime = line[1]
23. acstime= re.findall(r"\d\d\d\d-\d\d-\d\d\ \d\d:\d\d:\d\d",acstime)
24. message = line[2]
25. **if** acstime == asctimne\_pre:
26. count +=1
27. **else**:
28. asctimne\_pre = acstime
29. y\_vec[-1] = count
30. line1 = live\_plotter(x\_vec,y\_vec,line1)
31. y\_vec = np.append(y\_vec[1:],0.0)
32. count = 0
33. **except**:
34. **pass**

# **Phụ lục 3: Function: draw graph**

Mã chương trình mô tả đầy đủ chức năng vẽ đồ thị

livegraph.py

1. **import** matplotlib.pyplot as plt
2. **import** numpy as np
4. # use ggplot style for more sophisticated visuals
5. plt.style.use('ggplot')
7. **def** live\_plotter(x\_vec,y1\_data,line1,identifier='',pause\_time=0.1):
8. **if** line1==[]:
9. # this is the call to matplotlib that allows dynamic plotting
10. plt.ion()
11. fig = plt.figure(figsize=(13,6))
12. ax = fig.add\_subplot(111)
13. # create a variable for the line so we can later update it
14. line1, = ax.plot(x\_vec,y1\_data,'-o',alpha=0.8)
15. #update plot label/title
16. plt.ylabel('Access')
17. plt.title('Title: '.format(identifier))
18. plt.show()
20. # after the figure, axis, and line are created, we only need to update the y-data
21. line1.set\_ydata(y1\_data)
22. # adjust limits if new data goes beyond bounds
23. **if** np.min(y1\_data)<=line1.axes.get\_ylim()[0] **or** np.max(y1\_data)>=line1.axes.get\_ylim()[1]:
24. plt.ylim([np.min(y1\_data)-np.std(y1\_data),np.max(y1\_data)+np.std(y1\_data)])
25. # this pauses the data so the figure/axis can catch up - the amount of pause can be altered above
26. plt.pause(pause\_time)
28. # return line so we can update it again in the next iteration
29. **return** line1
31. **def** static\_plotter(x\_vec,y1\_data):
32. plt.plot(x\_vec,y1\_data)
33. plt.gcf().autofmt\_xdate()
34. plt.show()

# **Phụ lục 4: Function: traffic**

Mã chương trình mô tả đầy đủ chức năng về ham xác định lưu lượng

Traffic.py

1. # -\*- coding: utf-8 -\*-
2. **from** time **import** mktime
3. **import** django
4. **from** django.core.exceptions **import** ImproperlyConfigured
5. **from** django.db.models **import** Count
6. **from** django.utils.translation **import** ugettext
7. **from** django.utils.translation **import** ugettext\_lazy as \_
8. **from** . **import** settings
9. **from** .utils **import** get\_verbose\_name
11. **class** Modules(object):
12. '''''
13. Set of :class:`.Module`.
14. '''
15. **def** load(self):
16. '''''
17. Import and instanciate modules defined in
18. ``settings.TRAFFIC\_MODULES``.
19. '''
20. **from** importlib **import** import\_module
22. self.\_modules = ()
23. **for** module\_path **in** settings.TRAFFIC\_MODULES:
24. **try**:
25. dot = module\_path.rindex('.')
26. **except** ValueError:
27. **raise** ImproperlyConfigured('{0} isn\'t a traffic module'.format(module\_path))
28. traffic\_module = module\_path[:dot]
29. traffic\_classname = module\_path[dot + 1:]
31. **try**:
32. mod = import\_module(traffic\_module)
33. **except** ImportError as err:
34. **raise** ImproperlyConfigured('Error importing module {0}: "{1}"'.format(traffic\_module, err))
36. **try**:
37. traffic\_class = getattr(mod, traffic\_classname)
38. **except** AttributeError:
39. **raise** ImproperlyConfigured('Traffic module "{0}" does not define a "{1}" class'.format(
40. traffic\_module,
41. traffic\_classname,
42. ))
44. self.\_modules += (traffic\_class(),)
46. @property
47. **def** modules(self):
48. '''''
49. Get loaded modules, load them if isn't already made.
50. '''
51. **if** **not** hasattr(self, '\_modules'):
52. self.load()
53. **return** self.\_modules
55. **def** table(self, queries):
56. '''''
57. Get a list of modules' counters.
58. '''
59. **return** tuple([(
60. module.verbose\_name\_plural, [module.count(qs) **for** qs **in** queries]
61. ) **for** module **in** self.modules])
63. **def** graph(self, days):
64. '''''
65. Get a list of modules' counters for all the given days.
66. '''
67. **return** tuple([{
68. 'data': [(mktime(day.timetuple()) \* 1000, module.count(qs)) **for** day, qs **in** days],
69. 'label': str(ugettext(module.verbose\_name\_plural)),
70. } **for** module **in** self.modules])
72. modules = Modules()
74. **class** Module(object):
75. '''''
76. Base module class.
77. '''
78. **def** \_\_init\_\_(self):
79. self.module\_name = self.\_\_class\_\_.\_\_name\_\_
81. **if** **not** hasattr(self, 'verbose\_name'):
82. self.verbose\_name = get\_verbose\_name(self.module\_name)
83. **if** **not** hasattr(self, 'verbose\_name\_plural'):
84. **if** django.VERSION < (1, 11):
85. **from** django.utils.translation **import** string\_concat
86. self.verbose\_name\_plural = string\_concat(self.verbose\_name, 's')
87. **else**:
88. **from** django.utils.text **import** format\_lazy
89. self.verbose\_name\_plural = format\_lazy('{}{}', self.verbose\_name, 's')
91. **def** count(self, qs):
92. **raise** NotImplementedError('"count" isn\'t defined.')
94. **class** Ajax(Module):
95. verbose\_name\_plural = \_('Ajax')
97. **def** count(self, qs):
98. **return** qs.filter(is\_ajax=True).count()
100. **class** NotAjax(Module):
101. verbose\_name = \_('Not Ajax')
102. verbose\_name\_plural = \_('Not Ajax')
104. **def** count(self, qs):
105. **return** qs.filter(is\_ajax=False).count()
107. **class** Error(Module):
108. verbose\_name = \_('Error')
109. verbose\_name\_plural = \_('Errors')
111. **def** count(self, qs):
112. **return** qs.filter(response\_\_gte=400).count()
114. **class** Error404(Module):
115. verbose\_name = \_('Error 404')
116. verbose\_name\_plural = \_('Errors 404')
118. **def** count(self, qs):
119. **return** qs.filter(response=404).count()
121. **class** Hit(Module):
122. verbose\_name = \_('Hit')
123. verbose\_name\_plural = \_('Hits')
125. **def** count(self, qs):
126. **return** qs.count()
128. **class** Search(Module):
129. verbose\_name = \_('Search')
130. verbose\_name\_plural = \_('Searches')
132. **def** count(self, qs):
133. **return** qs.search().count()
135. **class** Secure(Module):
136. verbose\_name = \_('Secure')
137. verbose\_name\_plural = \_('Secure')
139. **def** count(self, qs):
140. **return** qs.filter(is\_secure=True).count()
142. **class** Unsecure(Module):
143. verbose\_name = \_('Unsecure')
144. verbose\_name\_plural = \_('Unsecure')
146. **def** count(self, qs):
147. **return** qs.filter(is\_secure=False).count()
149. **class** UniqueVisit(Module):
150. verbose\_name = \_('Unique Visit')
151. verbose\_name\_plural = \_('Unique Visits')
153. **def** count(self, qs):
154. **return** qs.exclude(
155. referer\_\_startswith=settings.BASE\_URL,
156. ).count()
158. **class** UniqueVisitor(Module):
159. verbose\_name = \_('Unique Visitor')
160. verbose\_name\_plural = \_('Unique Visitor')
162. **def** count(self, qs):
163. **return** qs.aggregate(Count('ip', distinct=True))['ip\_\_count']
165. **class** User(Module):
166. verbose\_name = \_('User')
167. verbose\_name\_plural = \_('User')
169. **def** count(self, qs):
170. **return** qs.exclude(user\_\_isnull=False).count()
172. **class** UniqueUser(Module):
173. verbose\_name = \_('Unique User')
174. verbose\_name\_plural = \_('Unique User')
176. **def** count(self, qs):
177. **return** qs.aggregate(Count('user', distinct=True))['user\_\_count']

# **Phụ lục 5: Function: log information**

Mã chương trình minh họa hoạt động trích xuất các thông tin log để thống kế

Plugins.py

1. **from** django.db.models **import** Count
2. **from** django.template.loader **import** render\_to\_string
4. **from** . **import** settings
5. **from** .models **import** Request
6. **from** .traffic **import** modules
7. **from** .utils **import** get\_verbose\_name

10. **def** set\_count(items):
11. '''''
12. This is similar to "set", but this just creates a list with values.
13. The list will be ordered from most frequent down.
14. Example:
15. >>> inventory = ['apple', 'lemon', 'apple', 'orange', 'lemon', 'lemon']
16. >>> set\_count(inventory)
17. [('lemon', 3), ('apple', 2), ('orange', 1)]
18. '''
19. item\_count = {}
20. **for** item **in** items:
21. **if** **not** item:
22. **continue**
23. **if** item **not** **in** item\_count:
24. item\_count[item] = 0
25. item\_count[item] += 1
27. items = [(v, k) **for** k, v **in** item\_count.items()]
28. items.sort()
29. items.reverse()
31. **return** [(k, v) **for** v, k **in** items]

34. **class** Plugins(object):
35. **def** load(self):
36. **from** importlib **import** import\_module
37. **from** django.core **import** exceptions
39. self.\_plugins = []
40. **for** module\_path **in** settings.PLUGINS:
41. **try**:
42. dot = module\_path.rindex('.')
43. **except** ValueError:
44. **raise** exceptions.ImproperlyConfigured('{0} isn\'t a plugin'.format(module\_path))
45. plugin, plugin\_classname = module\_path[:dot], module\_path[dot + 1:]
47. **try**:
48. mod = import\_module(plugin)
49. **except** ImportError as e:
50. **raise** exceptions.ImproperlyConfigured('Error importing plugin {0}: "{1}"'.format(plugin, e))
52. **try**:
53. plugin\_class = getattr(mod, plugin\_classname)
54. **except** AttributeError:
55. **raise** exceptions.ImproperlyConfigured('Plugin "{0}" does not define a "{1}" class'.format(
56. plugin,
57. plugin\_classname,
58. ))
60. self.\_plugins.append(plugin\_class())
62. **def** plugins(self):
63. **if** **not** hasattr(self, '\_plugins'):
64. self.load()
65. **return** self.\_plugins
66. plugins = property(plugins)

69. plugins = Plugins()

72. **class** Plugin(object):
73. **def** \_\_init\_\_(self):
74. self.module\_name = self.\_\_class\_\_.\_\_name\_\_
76. **if** **not** hasattr(self, 'verbose\_name'):
77. self.verbose\_name = get\_verbose\_name(self.module\_name)
79. **def** template\_context(self):
80. **return** {}
82. **def** render(self):
83. templates = [
84. 'request/plugins/{0}.html'.format(self.\_\_class\_\_.\_\_name\_\_.lower()),
85. 'request/plugins/base.html',
86. ]
88. **if** hasattr(self, 'template'):
89. templates.insert(0, self.template)
91. kwargs = self.template\_context()
92. kwargs['verbose\_name'] = self.verbose\_name
93. kwargs['plugin'] = self
94. **return** render\_to\_string(templates, kwargs)

97. **class** LatestRequests(Plugin):
98. **def** template\_context(self):
99. **return** {'requests': Request.objects.all()[:5]}

102. **class** TrafficInformation(Plugin):
103. **def** template\_context(self):
104. INFO\_TABLE = ('today', 'this\_week', 'this\_month', 'this\_year', 'all')
105. INFO\_TABLE\_QUERIES = [getattr(Request.objects, query, None)() **for** query **in** INFO\_TABLE]
107. **return** {
108. 'traffic': modules.table(INFO\_TABLE\_QUERIES)
109. }

112. **class** TopPaths(Plugin):
113. **def** queryset(self):
114. **return** self.qs.filter(response\_\_lt=400)
116. **def** template\_context(self):
117. **return** {
118. 'paths': self.queryset().values('path').annotate(Count('path')).order\_by('-path\_\_count')[:10]
119. }

122. **class** TopErrorPaths(TopPaths):
123. template = 'request/plugins/toppaths.html'
125. **def** queryset(self):
126. **return** self.qs.filter(response\_\_gte=400)

129. **class** TopReferrers(Plugin):
130. **def** queryset(self):
131. **return** self.qs.unique\_visits().exclude(referer='')
133. **def** template\_context(self):
134. **return** {
135. 'referrers': self.queryset().values('referer').annotate(Count('referer')).order\_by('-referer\_\_count')[:10]
136. }

139. **class** TopSearchPhrases(Plugin):
140. **def** template\_context(self):
141. **return** {
142. 'phrases': set\_count(self.qs.search().only('referer').attr\_list('keywords'))[:10]
143. }

146. **class** TopBrowsers(Plugin):
147. **def** template\_context(self):
148. **return** {
149. 'browsers': set\_count(self.qs.only('user\_agent').attr\_list('browser'))[:5]
150. }

153. **class** ActiveUsers(Plugin):
154. **def** template\_context(self):
155. **return** {}

# **Phụ lục 6: Function: user behavior**

Mã chương trình minh họa thu thập hành vi người dùng

Login-analysis.py

1. **import** re,time
2. **from** datetime **import** datetime
3. **import** os, sys
4. **from** DjangoLogAnalysis **import** user\_behavior
5. **def** log\_analyst(line,list\_usersname):
6. result=''
7. **if** 'INSERT INTO "request\_request"' **in** line **and** 'login' **in** line:
8. query = re.findall(r"args=\[(.+)\]",line)
9. **if** re.match(r"(datetime.+\)), F",query[0]):
10. time = re.findall(r"(datetime.+\)), F",query[0])[0]
11. times = re.findall(r"\((.+)\)",time)[0]
12. times = times.replace(' ',',').split(',,')
13. times = datetime(int(times[0]),int(times[1]),int(times[2]),int(times[3]),int(times[4]),int(times[5]))
14. query=query[0].replace(time,'time').split(',')
15. **else**:
16. times = re.findall(r"(\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d)", query[0])[0]
17. query=query[0].replace(times,'time').split(',')
18. status = query[0]
19. method = re.findall(r"(POST|GET|PUT|DELETE)",query[1])[0]
20. ip = re.findall(r"(\d+.\d+.\d+.\d+)",query[6])[0]
21. path = query[2]
22. user = query[9]
23. result = "|%20s|%10s|%20s|%100s|%50s|%20s|\n"%(times,status,str(ip),str(user),str(path),list\_usersname)
24. list\_usersname=''
25. **return** result, list\_usersname
27. **def** main(path):
28. f = open(path, 'r')
29. line = 'begin'
30. lines =''
31. list\_usersname = ''
32. login\_detail="|%20s|%10s|%20s|%100s|%50s|%20s| \n"%('Time','status', 'IP','user agent','path','attach to user')
33. **while** line !='':
34. line = f.readline()
35. **if** **not** re.match(r"(INFO|DEBUG|WARNING|ERROR|CRITICAL)",line):
36. lines += line
37. **else**:
38. **if** lines !='':
39. **if** "username" **in** lines **and** "auth\_user" **in** lines:
40. username = re.findall(r"('\w+')",lines)
41. **if** username:
42. username = username[0].split("'")
43. username = username[1]
44. list\_usersname = username
45. result,list\_usersname = log\_analyst(lines,list\_usersname)
46. **if** result !='':
47. login\_detail +=result
48. lines = line
49. f.close()
50. link = os.path.abspath(os.path.dirname(sys.argv[0]))
51. f = open(link+'\ResultAnalysis\\analys\_login.txt','w')
52. f.write(login\_detail)
53. f.close()
54. user\_behavior.main(path)