# **LỜI NÓI ĐẦU**

# PHẦN I: ĐẶT VẤN ĐỀ

# PHẦN II: NỘI DUNG NGHIÊN CỨU

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

* 1. **Giới thiệu đề tài**
  2. **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 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,..

**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.

* + 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.

|  |  |
| --- | --- |
| Level | Used |
| DEBUG | Thông tin chi tiết, thường chỉ quan tâm khi cần biết rõ vấn đề của sự kiện xảy ra |
| 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 |

Bảng 1.1 Mô tả các level trong log

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:

|  |  |  |
| --- | --- | --- |
| 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 |

Bảng 1.2. Các định dạng được cung cấp trong django

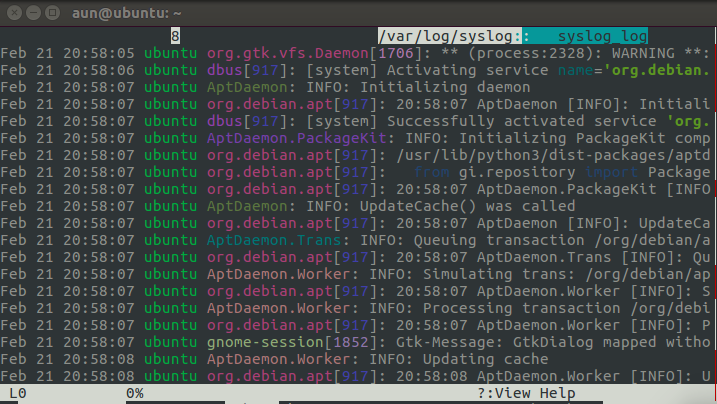
Một số kiểu log:

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>



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).

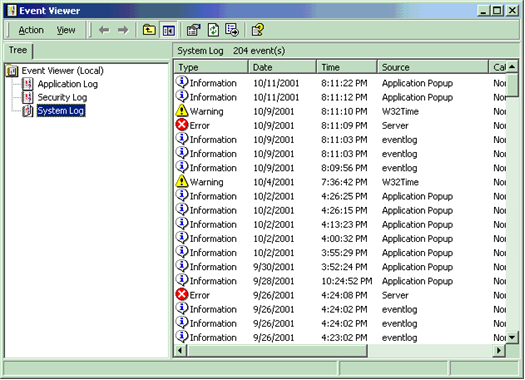
Hostname hoặc IP Address: host name có thể là một tên domain name ví dụ như ns1.example.com hoặc chỉ là tên máy giống như ubuntu trong ví dụ trên.

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).



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**

Django sử dụng module log được xây dựng sẵn trong python để thực hiện việc ghi log cho hệ thống.

* + 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**

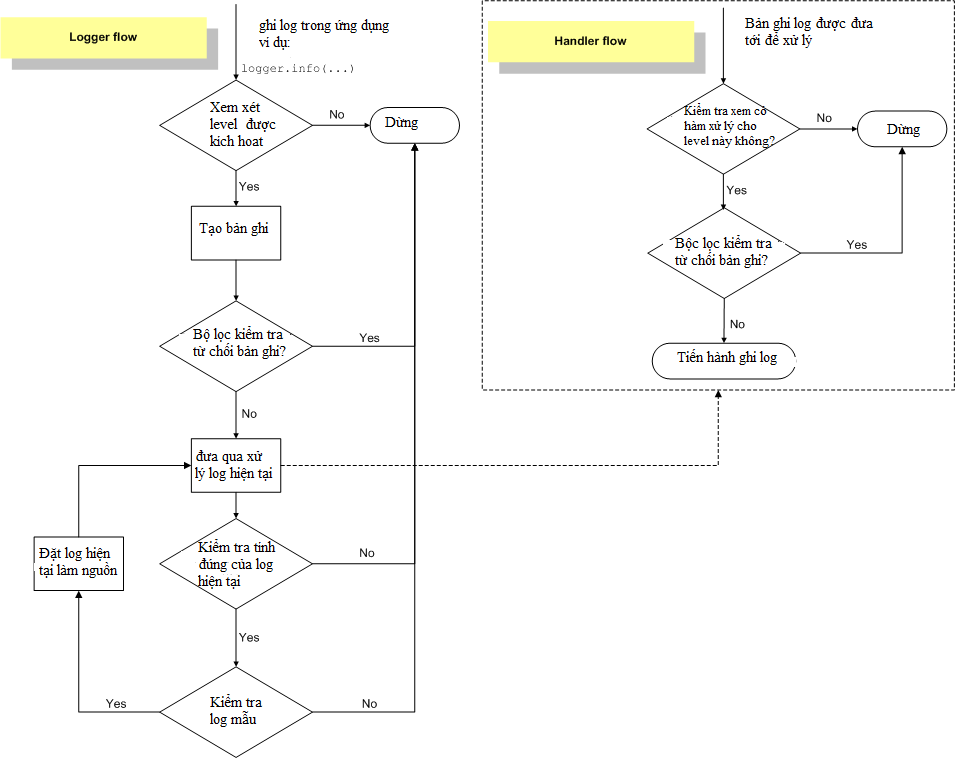
Hoạt động xử lý log trong Django được chia làm 2 luồng chính là: Looger và Handler.

* 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.

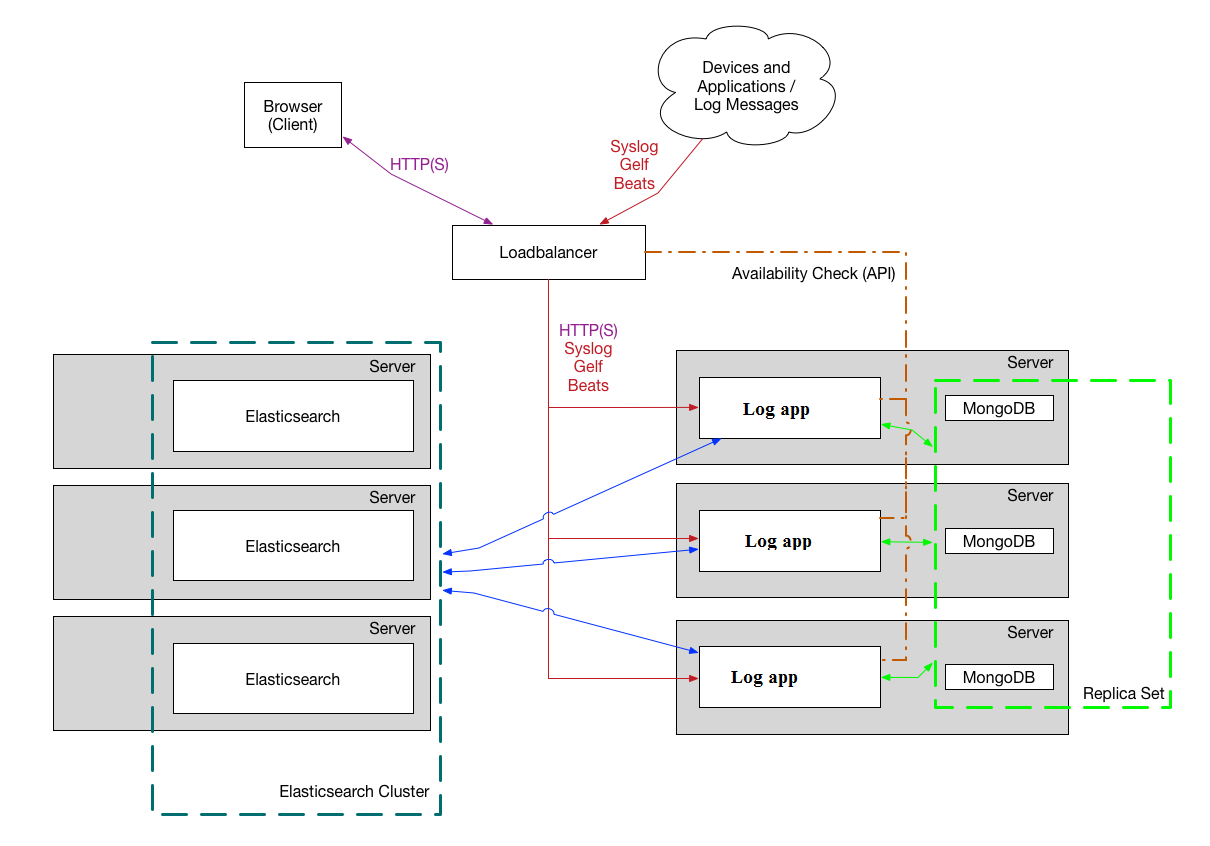


Hình 1.1. Cơ chế xử lý log trong Django

# CHƯƠNG II: PHÂN TÍCH THIẾT KẾ ỨNG DỤNG

2.1. 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

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

2.2. 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.

Hg

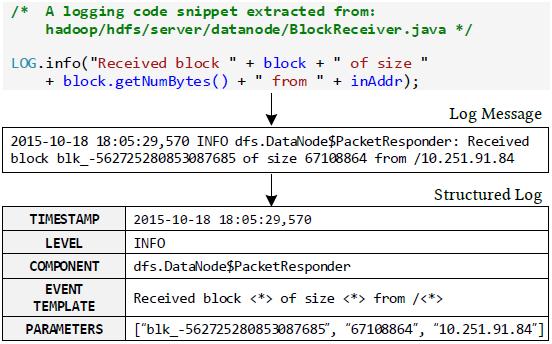
Hình

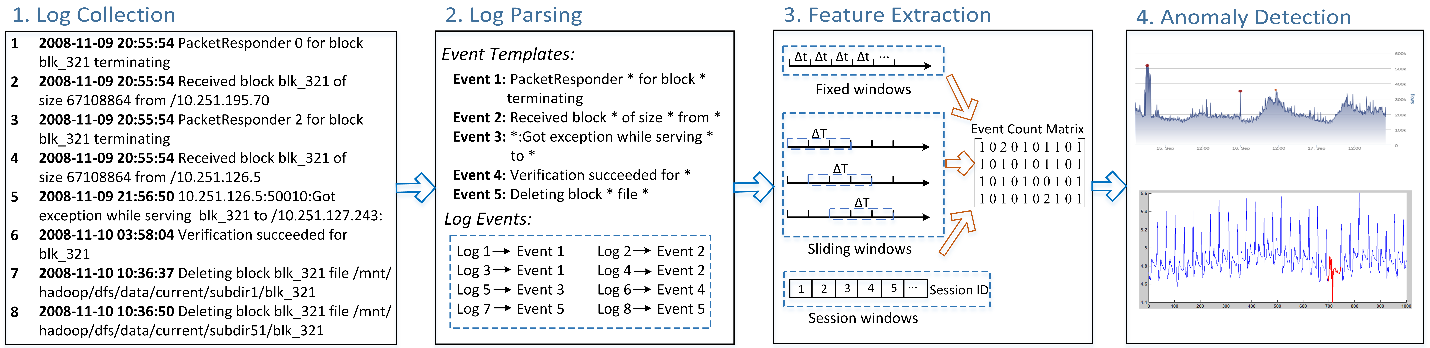
**Database Website**

**Log Database**

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

2.3. Thiết kế chức năng





2.4. Thiết kế giao diện

# CHƯƠNG III: 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.

3.1. Xây dụng dữ liệu

1. **class** Request(models.Model):
2. # Response information.
3. response = models.SmallIntegerField(\_('response'), choices=HTTP\_STATUS\_CODES, default=200)
5. # Request information.
6. method = models.CharField(\_('method'), default='GET', max\_length=7)
7. path = models.CharField(\_('path'), max\_length=255)
8. time = models.DateTimeField(\_('time'), default=timezone.now, db\_index=True)
10. is\_secure = models.BooleanField(\_('is secure'), default=False)
11. is\_ajax = models.BooleanField(
12. \_('is ajax'),
13. default=False,
14. help\_text=\_('Wheather this request was used via javascript.'),
15. )
17. # User information.
18. ip = models.GenericIPAddressField(\_('ip address'))
19. user = models.ForeignKey(AUTH\_USER\_MODEL, blank=True, null=True, verbose\_name=\_('user'), on\_delete=models.SET\_NULL)
20. referer = models.URLField(\_('referer'), max\_length=255, blank=True, null=True)
21. user\_agent = models.CharField(\_('user agent'), max\_length=255, blank=True, null=True)
22. language = models.CharField(\_('language'), max\_length=255, blank=True, null=True)
24. objects = RequestManager()
26. **class** Meta:
27. app\_label = 'request'
28. verbose\_name = \_('request')
29. verbose\_name\_plural = \_('requests')
30. ordering = ('-time',)
32. **def** \_\_str\_\_(self):
33. **return** '[{0}] {1} {2} {3}'.format(self.time, self.method, self.path, self.response)
35. **def** get\_user(self):
36. **return** get\_user\_model().objects.get(pk=self.user\_id)
38. **def** from\_http\_request(self, request, response=None, commit=True):
39. # Request information.
40. self.method = request.method
41. self.path = request.path[:255]
43. self.is\_secure = request.is\_secure()
44. self.is\_ajax = request.is\_ajax()
46. # User information.
47. self.ip = request.META.get('REMOTE\_ADDR', '')
48. self.referer = request.META.get('HTTP\_REFERER', '')[:255]
49. self.user\_agent = request.META.get('HTTP\_USER\_AGENT', '')[:255]
50. self.language = request.META.get('HTTP\_ACCEPT\_LANGUAGE', '')[:255]
52. **if** hasattr(request, 'user') **and** hasattr(request.user, 'is\_authenticated'):
53. is\_authenticated = request.user.is\_authenticated
54. **if** is\_authenticated:
55. self.user = request.user
57. **if** response:
58. self.response = response.status\_code
60. **if** (response.status\_code == 301) **or** (response.status\_code == 302):
61. self.redirect = response['Location']
63. **if** commit:
64. self.save()
66. @property
67. **def** browser(self):
68. **if** **not** self.user\_agent:
69. **return**
71. **if** **not** hasattr(self, '\_browser'):
72. self.\_browser = browsers.resolve(self.user\_agent)
73. **return** self.\_browser[0]
75. @property
76. **def** keywords(self):
77. **if** **not** self.referer:
78. **return**
80. **if** **not** hasattr(self, '\_keywords'):
81. self.\_keywords = engines.resolve(self.referer)
82. **if** self.\_keywords:
83. **return** ' '.join(self.\_keywords[1]['keywords'].split('+'))
85. @property
86. **def** hostname(self):
87. **try**:
88. **return** gethostbyaddr(self.ip)[0]
89. **except** Exception:  # socket.gaierror, socket.herror, etc
90. **return** self.ip
92. **def** save(self, \*args, \*\*kwargs):
93. **if** **not** request\_settings.LOG\_IP:
94. self.ip = request\_settings.IP\_DUMMY
95. **elif** request\_settings.ANONYMOUS\_IP:
96. parts = self.ip.split('.')[0:-1]
97. parts.append('1')
98. self.ip = '.'.join(parts)
99. **if** **not** request\_settings.LOG\_USER:
100. self.user = None
102. super(Request, self).save(\*args, \*\*kwargs)

3.2. Xây dựng các chức năng:

DetialLog.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** {}
156. **from** pymongo **import** MongoClient
157. **import** pymongo
158. **import** datetime
159. **from** config **import** config\_mongodb as mg
160. **def** connect\_db():
161. connection = MongoClient(mg["MONGODB\_HOST"], mg["MONGODB\_PORT"])
162. collection = connection[mg["DB\_NAME"]][mg["COLLECTION\_NAME"]]
163. **return** collection
165. collection = connect\_db()
167. **def** distane\_time(time):
168. time = time + datetime.timedelta(0,900)
169. **return** time
170. **def** GMT7(time):
171. time = time + datetime.timedelta(0,25200)
172. **return** time
173. **def** get\_data(collection):
174. x=[]
175. y=[]
176. number= 0
177. logs = collection.find()
178. begin= logs[0]["time"]
179. **for** log **in** logs:
180. time = log["time"]
181. **print** (time)
182. **if** begin<= time **and** time < distane\_time(begin):
183. number +=1
184. **else**:
185. x.append(GMT7(begin).strftime('%H:%M:%S'))
186. y.append(number)
187. begin = distane\_time(begin)
188. number =0
189. x.append(GMT7(begin).strftime('%H:%M:%S'))
190. y.append(number)
191. **return** x,y
192. x,y = get\_data(collection)
194. **from** livegraph **import** static\_plotter
195. static\_plotter(x,y)
196. f = open("E:\đồ án\DOAN\DemoDoAn\debug.log", 'r')
197. **import** time
198. **import** re
199. **from** livegraph **import** live\_plotter
200. **import** numpy as np
201. **def** main():
202. count = 0
203. asctimne\_pre=''
204. line1=[]
205. x\_vec = ["-10s","-9s","-8s","-7s","-6s","-5s","-4s","-3s","-2s","-1s","0"]
206. y\_vec = [0,0,0,0,0,0,0,0,0,0,0]
207. **while** True:
208. line = f.readline()
209. **if** line =='':
210. y\_vec[-1] = 0
211. line1 = live\_plotter(x\_vec,y\_vec,line1)
212. y\_vec = np.append(y\_vec[1:],0.0)
213. **else**:
214. line = line.split(' - ')
215. level= line[0]
216. acstime = line[1]
217. acstime= re.findall(r"\d\d\d\d-\d\d-\d\d\ \d\d:\d\d:\d\d",acstime)
218. #message = line[2]
219. **if** acstime == asctimne\_pre:
220. count +=1
221. **else**:
222. asctimne\_pre = acstime
223. y\_vec[-1] = count
224. line1 = live\_plotter(x\_vec,y\_vec,line1)
225. y\_vec = np.append(y\_vec[1:],0.0)
226. count = 0
227. **import** matplotlib.pyplot as plt
228. **import** numpy as np
230. # use ggplot style for more sophisticated visuals
231. plt.style.use('ggplot')
233. **def** live\_plotter(x\_vec,y1\_data,line1,identifier='',pause\_time=0.1):
234. **if** line1==[]:
235. # this is the call to matplotlib that allows dynamic plotting
236. plt.ion()
237. fig = plt.figure(figsize=(13,6))
238. ax = fig.add\_subplot(111)
239. # create a variable for the line so we can later update it
240. line1, = ax.plot(x\_vec,y1\_data,'-o',alpha=0.8)
241. #update plot label/title
242. plt.ylabel('Access')
243. plt.title('Title: '.format(identifier))
244. plt.show()
246. # after the figure, axis, and line are created, we only need to update the y-data
247. line1.set\_ydata(y1\_data)
248. # adjust limits if new data goes beyond bounds
249. **if** np.min(y1\_data)<=line1.axes.get\_ylim()[0] **or** np.max(y1\_data)>=line1.axes.get\_ylim()[1]:
250. plt.ylim([np.min(y1\_data)-np.std(y1\_data),np.max(y1\_data)+np.std(y1\_data)])
251. # this pauses the data so the figure/axis can catch up - the amount of pause can be altered above
252. plt.pause(pause\_time)
254. # return line so we can update it again in the next iteration
255. **return** line1
257. **def** static\_plotter(x\_vec,y1\_data):
258. plt.plot(x\_vec,y1\_data)
259. plt.gcf().autofmt\_xdate()
260. plt.show()

Traffic.py

1. **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 \_
9. **from** . **import** settings
10. **from** .utils **import** get\_verbose\_name

13. **class** Modules(object):
14. '''''
15. Set of :class:`.Module`.
16. '''
17. **def** load(self):
18. '''''
19. Import and instanciate modules defined in
20. ``settings.TRAFFIC\_MODULES``.
21. '''
22. **from** importlib **import** import\_module
24. self.\_modules = ()
25. **for** module\_path **in** settings.TRAFFIC\_MODULES:
26. **try**:
27. dot = module\_path.rindex('.')
28. **except** ValueError:
29. **raise** ImproperlyConfigured('{0} isn\'t a traffic module'.format(module\_path))
30. traffic\_module = module\_path[:dot]
31. traffic\_classname = module\_path[dot + 1:]
33. **try**:
34. mod = import\_module(traffic\_module)
35. **except** ImportError as err:
36. **raise** ImproperlyConfigured('Error importing module {0}: "{1}"'.format(traffic\_module, err))
38. **try**:
39. traffic\_class = getattr(mod, traffic\_classname)
40. **except** AttributeError:
41. **raise** ImproperlyConfigured('Traffic module "{0}" does not define a "{1}" class'.format(
42. traffic\_module,
43. traffic\_classname,
44. ))
46. self.\_modules += (traffic\_class(),)
48. @property
49. **def** modules(self):
50. '''''
51. Get loaded modules, load them if isn't already made.
52. '''
53. **if** **not** hasattr(self, '\_modules'):
54. self.load()
55. **return** self.\_modules
57. **def** table(self, queries):
58. '''''
59. Get a list of modules' counters.
60. '''
61. **return** tuple([(
62. module.verbose\_name\_plural, [module.count(qs) **for** qs **in** queries]
63. ) **for** module **in** self.modules])
65. **def** graph(self, days):
66. '''''
67. Get a list of modules' counters for all the given days.
68. '''
69. **return** tuple([{
70. 'data': [(mktime(day.timetuple()) \* 1000, module.count(qs)) **for** day, qs **in** days],
71. 'label': str(ugettext(module.verbose\_name\_plural)),
72. } **for** module **in** self.modules])

75. modules = Modules()

78. **class** Module(object):
79. '''''
80. Base module class.
81. '''
82. **def** \_\_init\_\_(self):
83. self.module\_name = self.\_\_class\_\_.\_\_name\_\_
85. **if** **not** hasattr(self, 'verbose\_name'):
86. self.verbose\_name = get\_verbose\_name(self.module\_name)
87. **if** **not** hasattr(self, 'verbose\_name\_plural'):
88. **if** django.VERSION < (1, 11):
89. **from** django.utils.translation **import** string\_concat
90. self.verbose\_name\_plural = string\_concat(self.verbose\_name, 's')
91. **else**:
92. **from** django.utils.text **import** format\_lazy
93. self.verbose\_name\_plural = format\_lazy('{}{}', self.verbose\_name, 's')
95. **def** count(self, qs):
96. **raise** NotImplementedError('"count" isn\'t defined.')

99. **class** Ajax(Module):
100. verbose\_name\_plural = \_('Ajax')
102. **def** count(self, qs):
103. **return** qs.filter(is\_ajax=True).count()

106. **class** NotAjax(Module):
107. verbose\_name = \_('Not Ajax')
108. verbose\_name\_plural = \_('Not Ajax')
110. **def** count(self, qs):
111. **return** qs.filter(is\_ajax=False).count()

114. **class** Error(Module):
115. verbose\_name = \_('Error')
116. verbose\_name\_plural = \_('Errors')
118. **def** count(self, qs):
119. **return** qs.filter(response\_\_gte=400).count()

122. **class** Error404(Module):
123. verbose\_name = \_('Error 404')
124. verbose\_name\_plural = \_('Errors 404')
126. **def** count(self, qs):
127. **return** qs.filter(response=404).count()

130. **class** Hit(Module):
131. verbose\_name = \_('Hit')
132. verbose\_name\_plural = \_('Hits')
134. **def** count(self, qs):
135. **return** qs.count()

138. **class** Search(Module):
139. verbose\_name = \_('Search')
140. verbose\_name\_plural = \_('Searches')
142. **def** count(self, qs):
143. **return** qs.search().count()

146. **class** Secure(Module):
147. verbose\_name = \_('Secure')
148. verbose\_name\_plural = \_('Secure')
150. **def** count(self, qs):
151. **return** qs.filter(is\_secure=True).count()

154. **class** Unsecure(Module):
155. verbose\_name = \_('Unsecure')
156. verbose\_name\_plural = \_('Unsecure')
158. **def** count(self, qs):
159. **return** qs.filter(is\_secure=False).count()

162. **class** UniqueVisit(Module):
163. verbose\_name = \_('Unique Visit')
164. verbose\_name\_plural = \_('Unique Visits')
166. **def** count(self, qs):
167. **return** qs.exclude(
168. referer\_\_startswith=settings.BASE\_URL,
169. ).count()

172. **class** UniqueVisitor(Module):
173. verbose\_name = \_('Unique Visitor')
174. verbose\_name\_plural = \_('Unique Visitor')
176. **def** count(self, qs):
177. **return** qs.aggregate(Count('ip', distinct=True))['ip\_\_count']

180. **class** User(Module):
181. verbose\_name = \_('User')
182. verbose\_name\_plural = \_('User')
184. **def** count(self, qs):
185. **return** qs.exclude(user\_\_isnull=False).count()

188. **class** UniqueUser(Module):
189. verbose\_name = \_('Unique User')
190. verbose\_name\_plural = \_('Unique User')
192. **def** count(self, qs):
193. **return** qs.aggregate(Count('user', distinct=True))['user\_\_count']

pie\_chart

1. # -\*- coding: utf-8 -\*-
2. **from** django **import** template
3. **from** django.utils.http **import** urlquote
5. register = template.Library()

8. @register.simple\_tag
9. **def** pie\_chart(items, width=440, height=190):
10. **return** '//chart.googleapis.com/chart?cht=p3&chd=t:{0}&chs={1}x{2}&chl={3}'.format(
11. urlquote(','.join([str(item[1]) **for** item **in** items])),
12. width,
13. height,
14. urlquote('|'.join([str(item[0]) **for** item **in** items])),
15. )

3.3. Xây dựng giao diện

Kết hợp thư viện tích hợp sẵn trong thư viện matplotlib cùng django admin để xây dụng giao diện cho ứng dụng.

Index.html

1. {% extends "admin/app\_index.html" %}
2. {% load i18n %}
4. {% block content %}
5. <div id="content-main">
7. {% **if** app\_list %}
8. {% **for** app **in** app\_list %}
9. <div **class**="module">
10. <table summary="{% blocktrans with app.name as name %}Models available in the {{ name }} application.{% endblocktrans %}">
11. <caption><a href="{{ app.app\_url }}" **class**="section">{% blocktrans with app.name as name %}{{ name }}{% endblocktrans %}</a></caption>
12. {% **for** model **in** app.models %}
13. <tr>
14. {% **if** model.perms.change %}
15. <th scope="row"><a href="{{ model.admin\_url }}">{{ model.name }}</a></th>
16. {% **else** %}
17. <th scope="row">{{ model.name }}</th>
18. {% endif %}
20. {% **if** model.perms.add %}
21. <td><a href="{{ model.admin\_url }}add/" **class**="addlink">{% trans 'Add' %}</a></td>
22. {% **else** %}
23. <td> </td>
24. {% endif %}
26. {% **if** model.perms.change %}
27. <td><a href="{{ model.admin\_url }}" **class**="changelink">{% trans 'Change' %}</a></td>
28. {% **else** %}
29. <td> </td>
30. {% endif %}
31. </tr>
33. {% **if** model.perms.change %}
34. <tr>
35. <th scope="row"><a href="{{ model.admin\_url }}overview/">{% trans 'Overview' %}</a></th>
36. <td> </td>
37. <td> </td>
38. </tr>
39. {% endif %}
40. {% endfor %}
41. </table>
42. </div>
43. {% endfor %}
44. {% **else** %}
45. <p>{% trans "You don't have permission to edit anything." %}</p>
46. {% endif %}
47. </div>
48. {% endblock %}

Overview.html

1. {% extends "admin/base\_site.html" %}{% load i18n request\_admin static %}
2. {% block extrahead %}
3. {{ block.super }}
4. <!--[**if** IE]><script language="javascript" type="text/javascript" src="{% static "request/js/excanvas.min.js" %}"></script><![endif]-->
5. <script type="text/javascript" src="{% static "request/js/jquery-1.4.2.min.js" %}"></script>
6. <script type="text/javascript" src="{% static "request/js/jquery.flot.js" %}"></script>
7. <script type="text/javascript" src="{% static "request/js/jquery.timeago.js" %}"></script>
8. <script type="text/javascript" src="{% static "request/js/request.js" %}"></script>
9. {% **if** request **and** request.user %}
10. <script type="text/javascript">
11. function loadTrafficGraph(days) {
12. **if** (days == undefined) {
13. path = "{% url "admin:request\_request\_traffic" %}";
14. } **else** {
15. path = "{% url "admin:request\_request\_traffic" %}?days=" + days;
16. }
17. $.getJSON(path, function(data) {
18. trafficGraph($("#trafficgraph"), data);
19. });
20. }
21. $(document).ready(function(){
22. loadTrafficGraph();
23. });
24. </script>
25. {% endif %}
26. {% endblock %}
28. {% block breadcrumbs %}
29. {% **if** request **and** request.user %}
30. <div **class**="breadcrumbs">
31. <a href="{% url "admin:index" %}">{% trans "Home" %}</a> ›
32. <a href="{% url "admin:app\_list" "request" %}">{% trans "Request" %}</a> ›
33. <a href="{% url "admin:request\_request\_changelist" %}">{% trans "Requests" %}</a> ›
34. {% trans "Overview" %}
35. </div>
36. {% endif %}
37. {% endblock %}
39. {% block content %}
40. <div style="width:920px;">
41. <div **class**="module" style="clear: both;">
42. <table style="width: 100%;">
43. <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>
44. </table>
46. <div style="padding: 15px;">
47. <div id="trafficgraph" style="width: 900px; height: 250px;"></div>
48. </div>
49. </div>
51. {% **for** plugin **in** plugins %}
52. <div **class**="module" style="float: left; width: 450px; {% cycle 'clear: both;' 'margin-left: 16px;' %}">
53. {{ plugin.render }}
54. </div>
55. {% endfor %}
56. </div>
57. {% endblock %}

Plugins

## **trafficinformation.html**

1. {% extends "request/plugins/table.html" %}
2. {% load i18n %}
3. {% block table %}
4. <tr>
5. <td> </td>
6. <th>{% trans "Today" %}</th>
7. <th>{% trans "This week" %}</th>
8. <th>{% trans "This month" %}</th>
9. <th>{% trans "This year" %}</th>
10. <th>{% trans "Alltime" %}</th>
11. </tr>
12. {% **for** key,value **in** traffic %}
13. <tr>
14. <th>{{ key }}</th>
15. {% **for** item **in** value %}
16. <td>{{ item }}</td>
17. {% endfor %}
18. </tr>
19. {% endfor %}
20. {% endblock %}

## **latestrequests.html**

1. {% extends "request/plugins/table.html" %}
2. {% load i18n %}
3. {% block table %}
4. <tr>
5. <th>{% trans "Path" %}</th>
6. <th>{% trans "From" %}</th>
7. <th>{% trans "Response" %}</th>
8. <th>{% trans "Time" %}</th>
9. </tr>
10. {% **for** request **in** requests %}
11. <tr>
12. <td><a href="{% url "admin:request\_request\_changelist" %}?path={{ request.path }}" title="{{ request.path }}">{{ request.path|truncatechars:35 }}</a></td>
13. <td>{% firstof request.get\_user request.ip %}</td>
14. <td>{{ request.get\_response\_display }}</td>
15. <td><a href="{% url "admin:request\_request\_change" request.id %}">
16. <abbr **class**="timeago" title="{{ request.time|date:"c" }}">{{ request.time|date:"D M d H:i:s O Y" }}</abbr>
17. </a></td>
18. </tr>
19. {% endfor %}
20. {% endblock %}

## **topsearchphrases.html**

1. {% extends "request/plugins/table.html" %}
2. {% load i18n %}
3. {% block table %}
4. <tr>
5. <th>{% trans "Phrase" %}</th>
6. <th>{% trans "Visits" %}</th>
7. </tr>
8. {% **for** phrase, visits **in** phrases %}
9. <tr>
10. <td>{{ phrase|truncatechars:60 }}</td>
11. <td>{{ visits }}</td>
12. </tr>
13. {% endfor %}
14. {% endblock %}

# PHẦN III: TRIỂN KHAI ỨNG DỤNG

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

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

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

hoặc

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

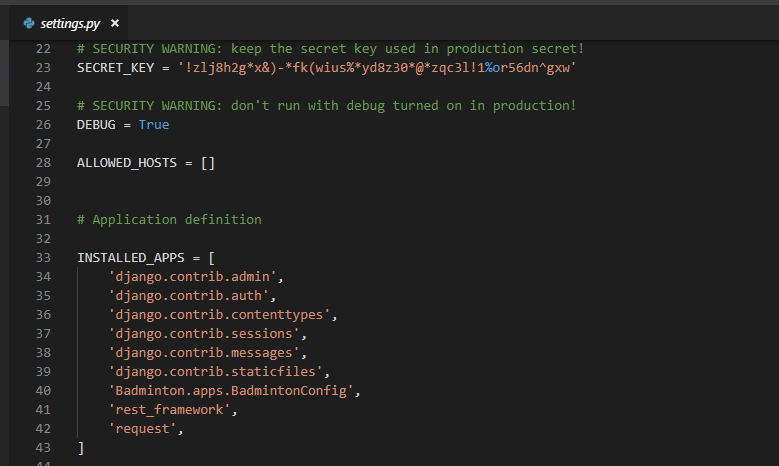
Sau khi tải ứng dụng, tiến hành cài đặt các thư viện cần thiết:

1. pip install -r requirement.txt

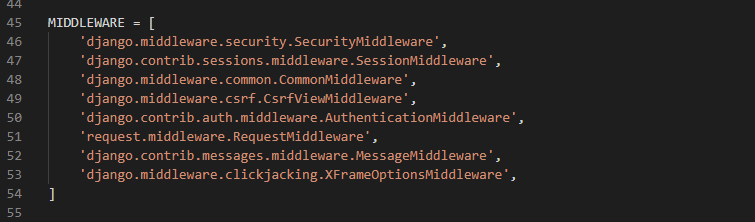
Tiếp theo, tiến hành cấu hình cho ứng dụng để phù hợp với hệ thống đang chạy:

Setting.py

* Thêm request vào trong INSTALLED\_APPS của file setting



* 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



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