Grigorios Dimopoulos
M1548
Dimitris Sartzetakis
M1561

LogDb

23th November 2019

OVERVIEW

LogDb is a web application that given some log files of specific form will parse them, add them in specific database tables and give you a web application interface to query/add/edit/delete the logs, and execute more advanced queries as well. It also includes user account implementation in the way of registration , login and permissions. Also the critical user events are logged in the database.

Web app overview and functionalities.

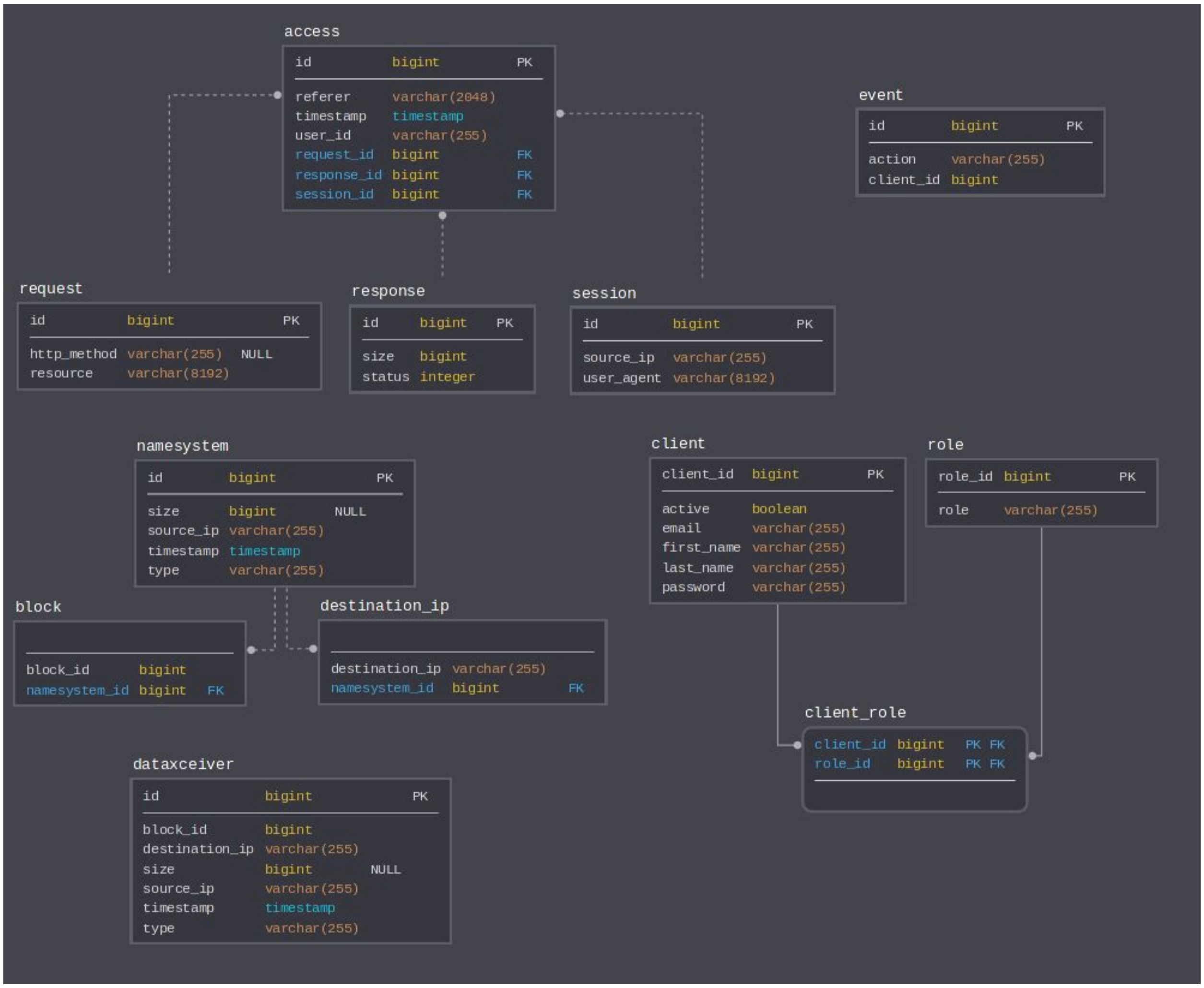
Using the LogDb someone is able:

- Without an account:
    1. Create an account.
    2. Parse specific log files.
    3. Search through logs by ip.
    4. List the logs results in pages using pagination.
- With an account:
    1. Login.
    2. Edit logs.
    3. Insert new logs.
    4. Delete logs.
    5. Eventlog his actions.
    6. Call advanced parameterized stored procedures and see results.

Technologies used for the project:

We implemented the project using, PostgreSQL, Gradle, Java 8 with SpringBoot, Spring Security, Spring JPA, Thymeleaf, Bootstrap, CSS .
Our IDEA of preference was IntelliJ.

Database schema:



access

id	bigint	PK
referer	varchar(2048)	
timestamp	timestamp	
user_id	varchar(255)	
request_id	bigint	FK
response_id	bigint	FK
session_id	bigint	FK

event

id	bigint	PK
action	varchar(255)	
client_id	bigint	

request

id	bigint	PK
http_method	varchar(255)	NULL
resource	varchar(8192)	

response

id	bigint	PK
size	bigint	
status	integer	

session

id	bigint	PK
source_ip	varchar(255)	
user_agent	varchar(8192)	

namesystem

id	bigint	PK
size	bigint	NULL
source_ip	varchar(255)	
timestamp	timestamp	
type	varchar(255)	

client

client_id	bigint	PK
active	boolean	
email	varchar(255)	
first_name	varchar(255)	
last_name	varchar(255)	
password	varchar(255)	

role

role_id	bigint	PK
role	varchar(255)	

block

block_id	bigint	
namesystem_id	bigint	FK

destination_ip

destination_ip	varchar(255)	
namesystem_id	bigint	FK

client_role

client_id	bigint	PK FK
role_id	bigint	PK FK

dataxceiver

id	bigint	PK
block_id	bigint	
destination_ip	varchar(255)	
size	bigint	NULL
source_ip	varchar(255)	
timestamp	timestamp	
type	varchar(255)	

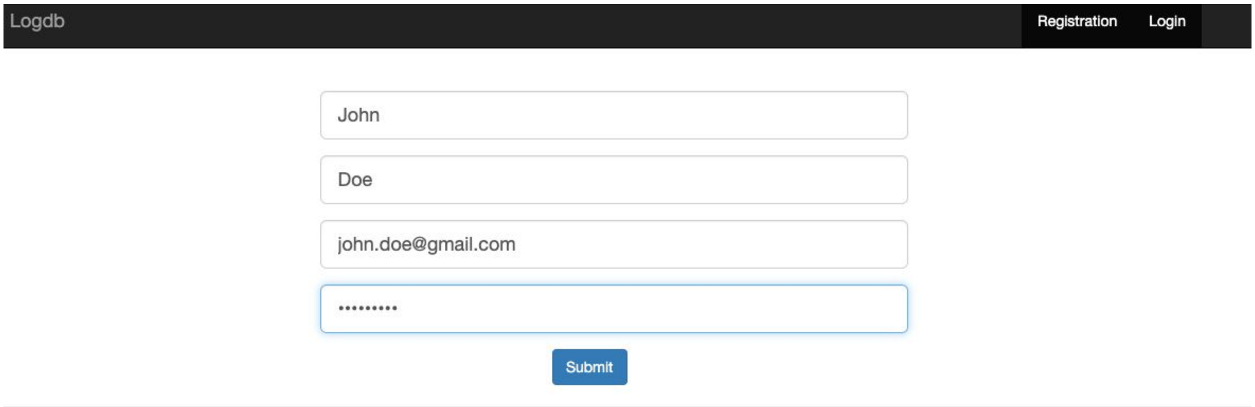
Conventions:

1. Search bar will remember the search term even after a refresh of the page you should clear the field to list all logs.

2. In each page of the log list we have one access log, one Namesystem log and one Dataxreceiver log. This has been done to list all the logs together and not in separate pages.

LogDb snapshots:

- Registration page:



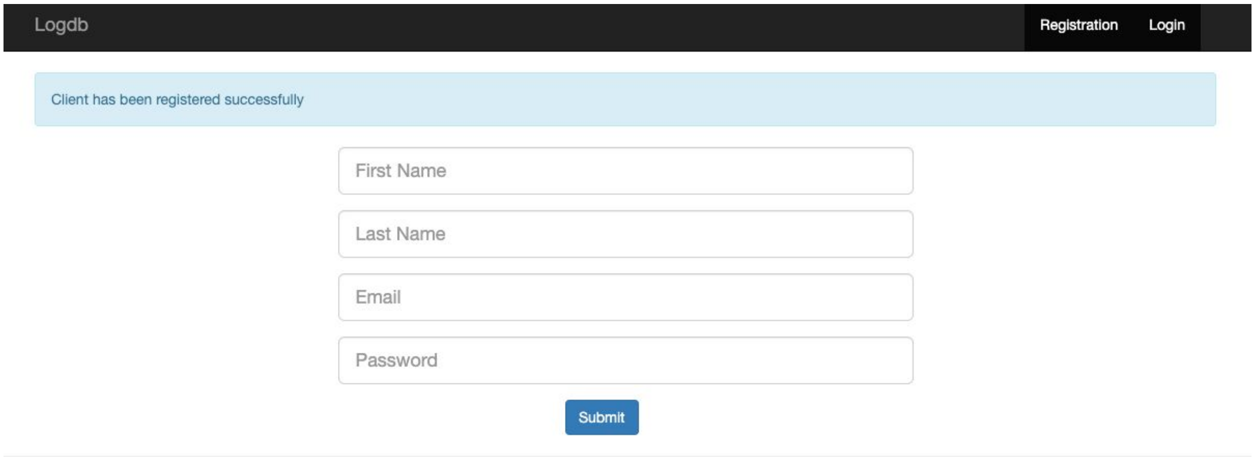
John

Doe

john.doe@gmail.com

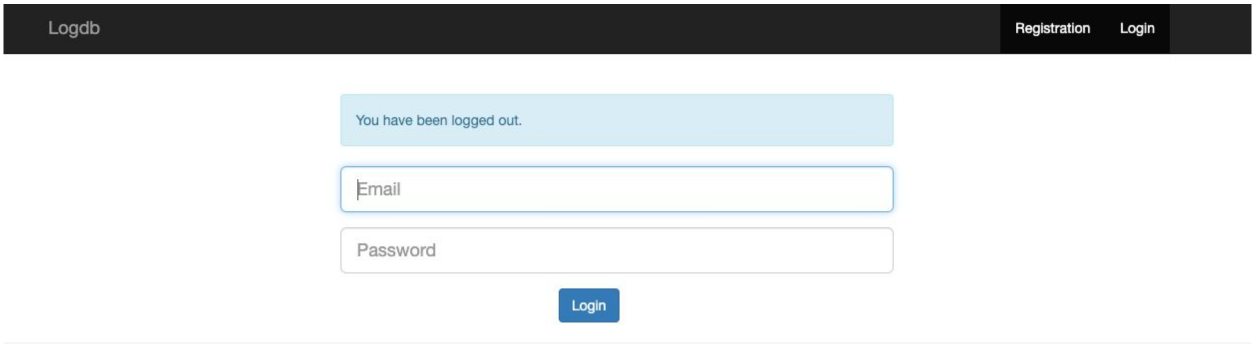
•••••••••

- Successful registration:



Client has been registered successfully

- Logout page:

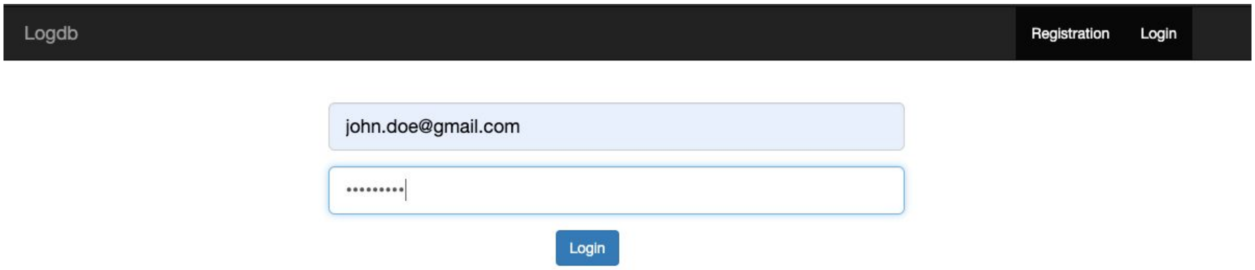


You have been logged out.

Email

Password

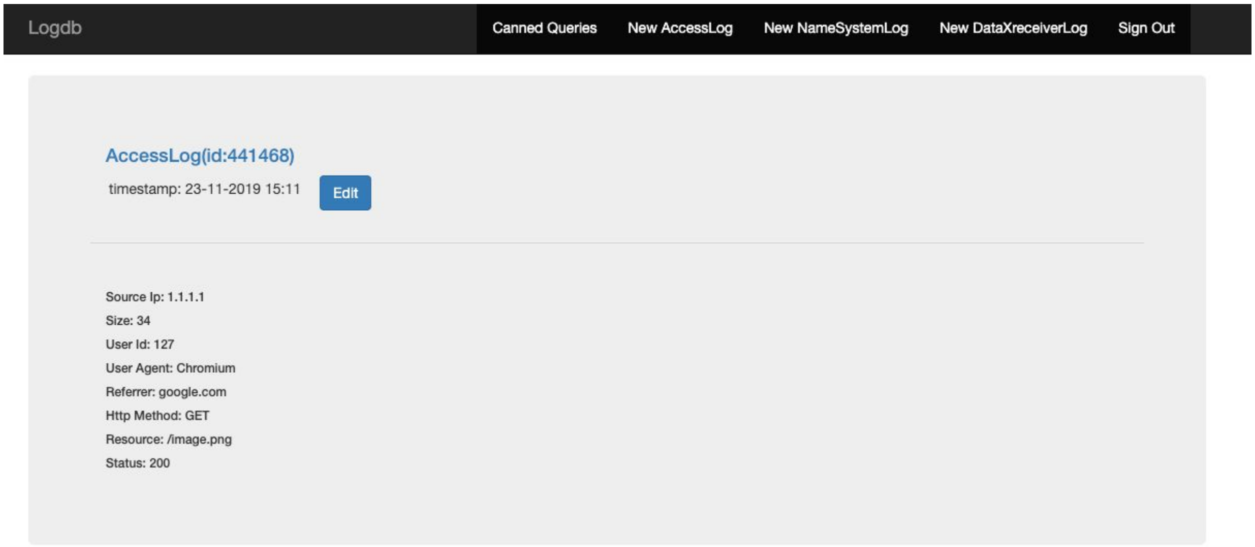
- Login page:



john.doe@gmail.com

•••••••••

- Log details page:

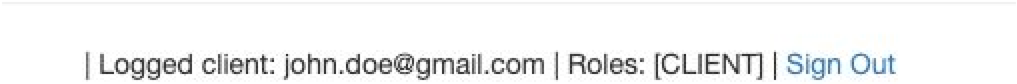


AccessLog(id:441468)

timestamp: 23-11-2019 15:11

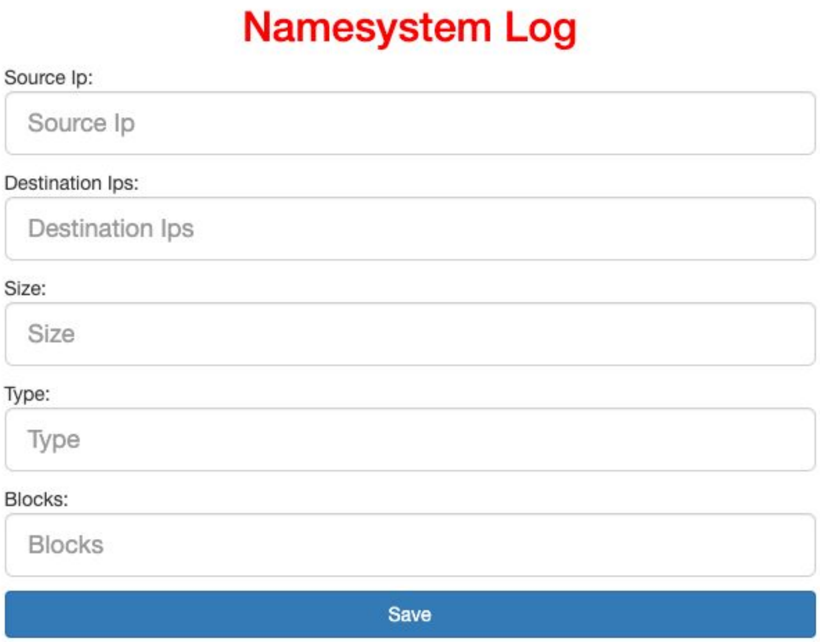
Source Ip: 1.1.1.1
Size: 34
User Id: 127
User Agent: Chromium
Referrer: google.com
Http Method: GET
Resource: /image.png
Status: 200

- Logged client details with role:



| Logged client: john.doe@gmail.com | Roles: [CLIENT] | Sign Out

- Creation of log:



Namesystem Log

Source Ip:

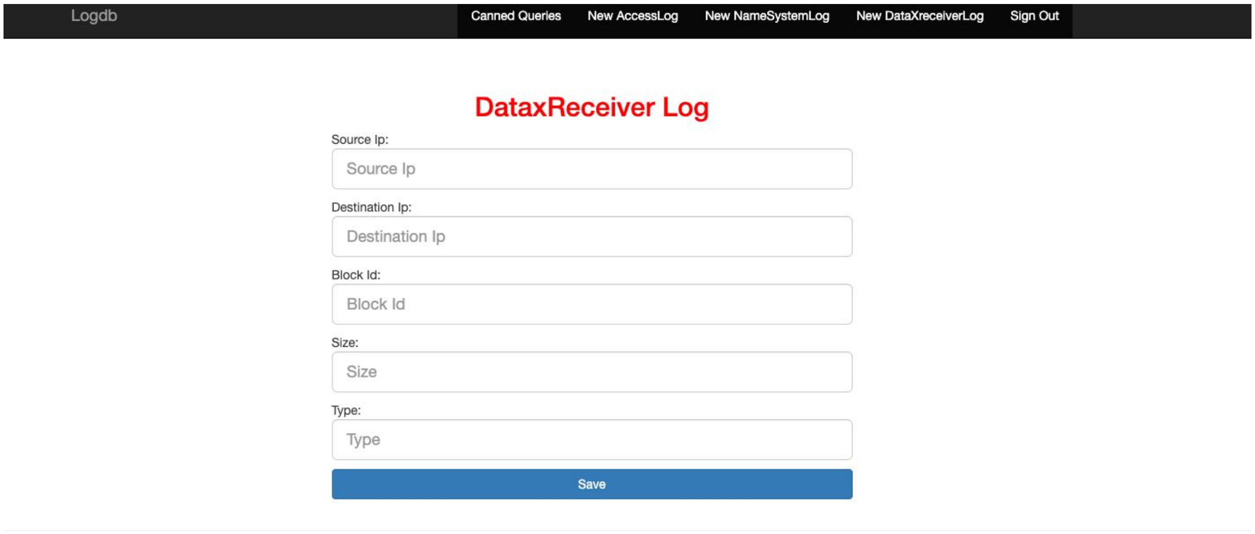
Destination Ips:

Size:

Type:

Blocks:

- Creation of log:



DataxReceiver Log

Source Ip:

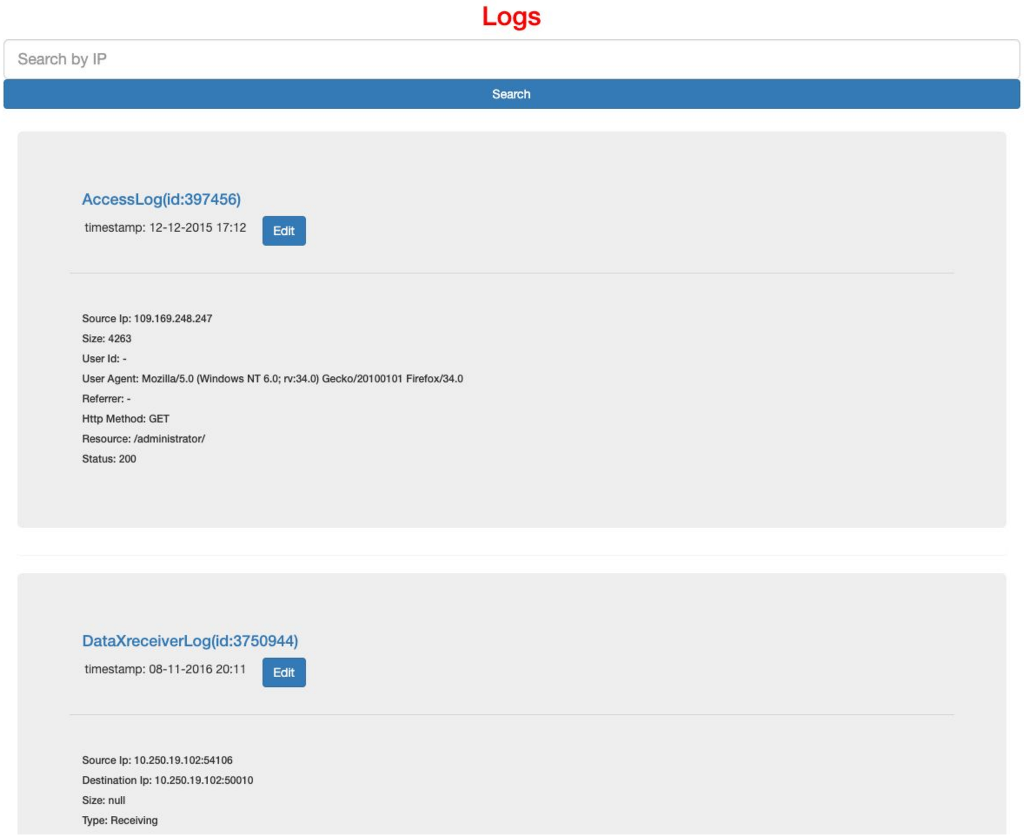
Destination Ip:

Block Id:

Size:

Type:

- Log list view:



Logs

AccessLog(id:397456)

timestamp: 12-12-2015 17:12

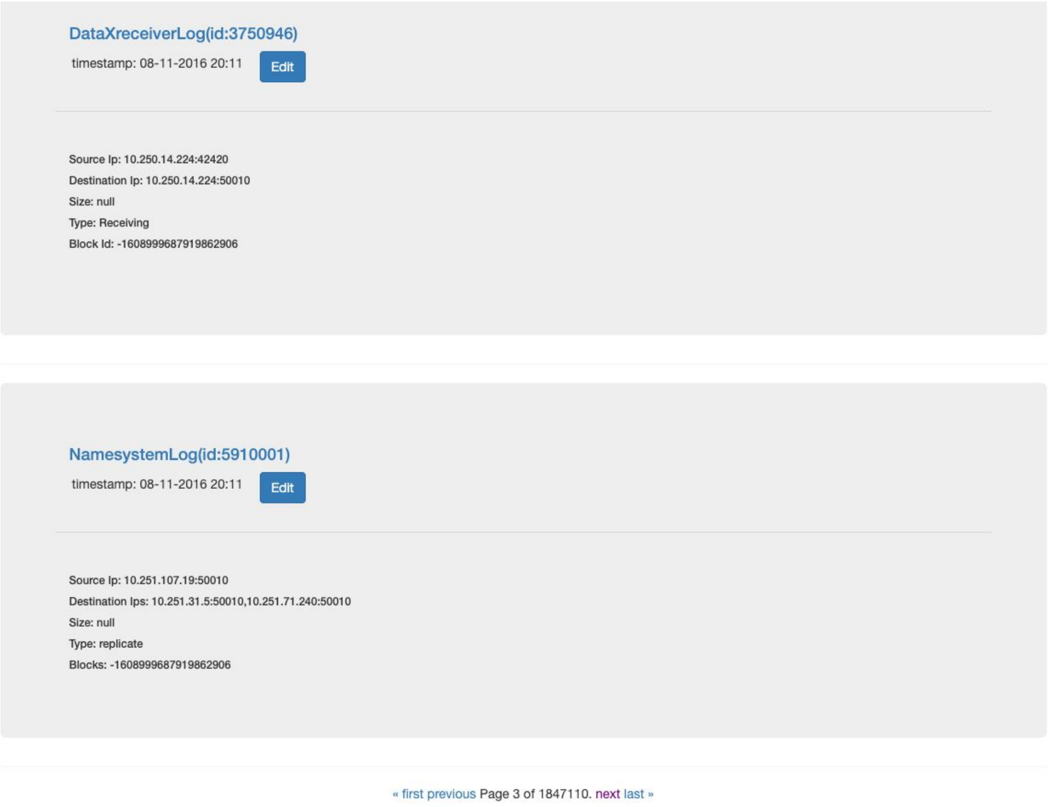
Source Ip: 109.169.248.247
Size: 4263
User Id: -
User Agent: Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0
Referrer: -
Http Method: GET
Resource: /administrator/
Status: 200

DataXreceiverLog(id:3750944)

timestamp: 08-11-2016 20:11

Source Ip: 10.250.19.102:54106
Destination Ip: 10.250.19.102:50010
Size: null
Type: Receiving

- Pagination:



DataXreceiverLog(id:3750946)

timestamp: 08-11-2016 20:11

Source Ip: 10.250.14.224:42420
Destination Ip: 10.250.14.224:50010
Size: null
Type: Receiving
Block Id: -1608999687919862906

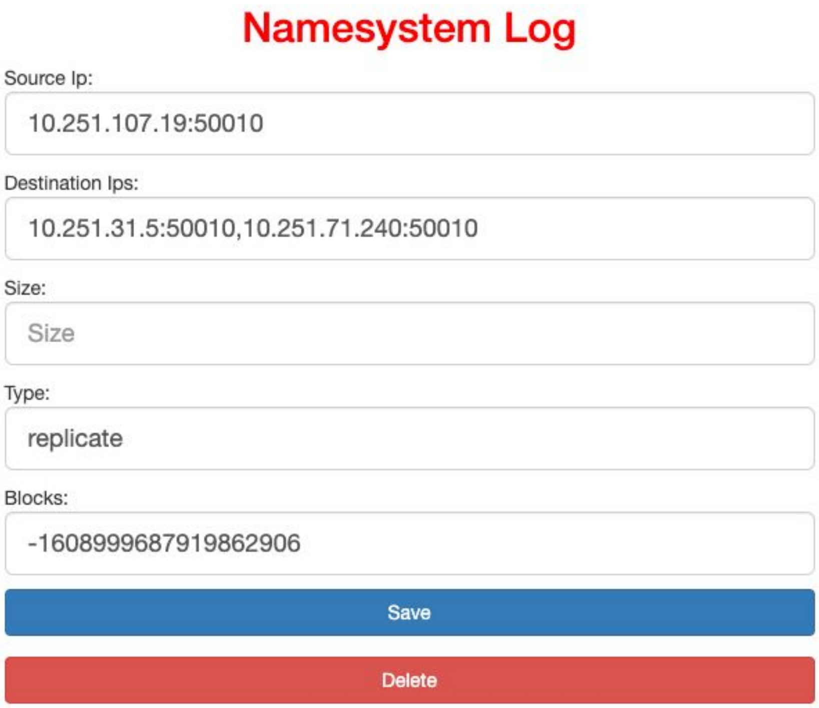
NamesystemLog(id:5910001)

timestamp: 08-11-2016 20:11

Source Ip: 10.251.107.19:50010
Destination Ips: 10.251.31.5:50010,10.251.71.240:50010
Size: null
Type: replicate
Blocks: -1608999687919862906

« first previous Page 3 of 1847110. next last »

- Edit Log:



Namesystem Log

Source Ip: 10.251.107.19:50010

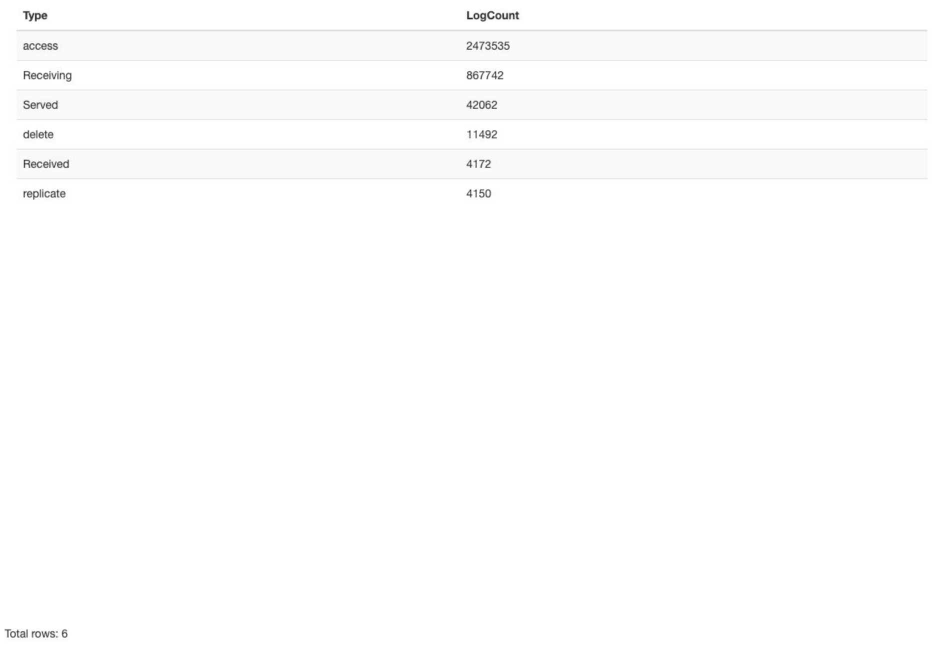
Destination Ips: 10.251.31.5:50010,10.251.71.240:50010

Size: Size

Type: replicate

Blocks: -1608999687919862906

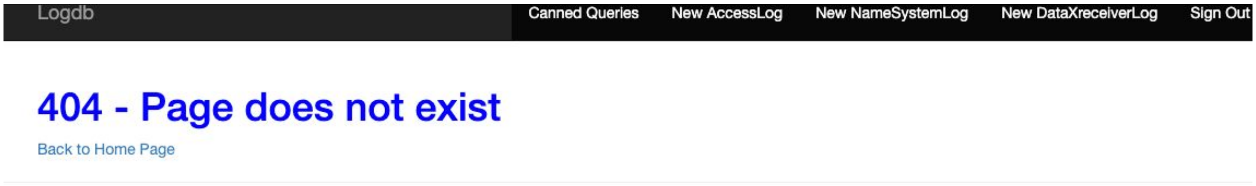
- Canned queries results page:



Type	LogCount
access	2473535
Receiving	867742
Served	42062
delete	11492
Received	4172
replicate	4150

Total rows: 6

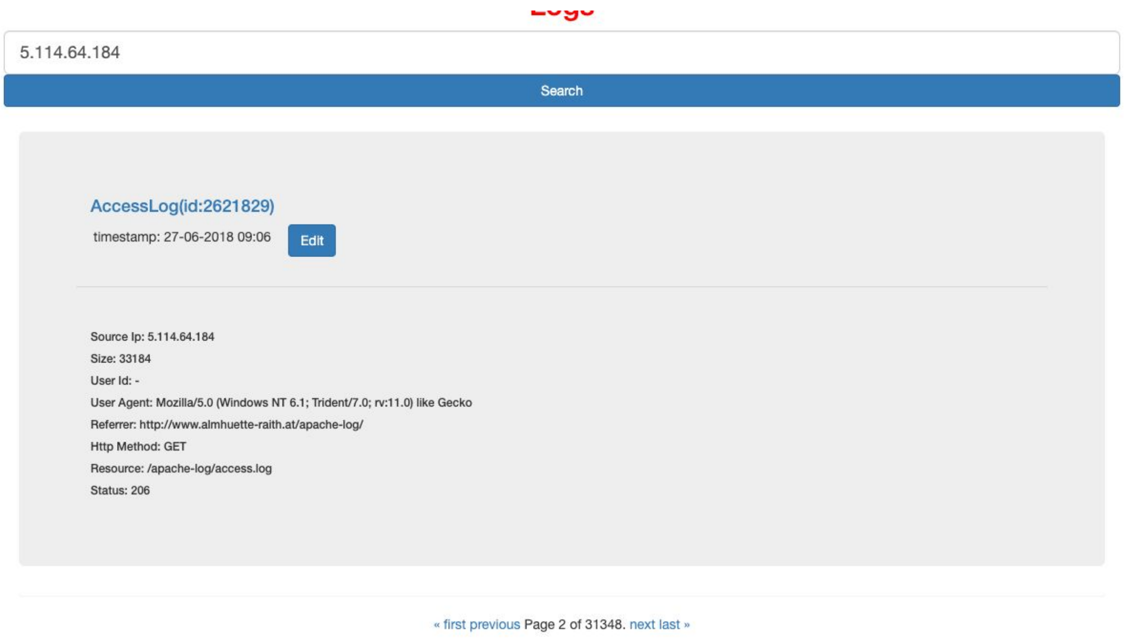
- Page does not exist:



404 - Page does not exist

Back to Home Page

- Search bar in action:



AccessLog(id:2621829)

timestamp: 27-06-2018 09:06

Source Ip: 5.114.64.184

Size: 33184

User Id: -

User Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko

Referrer: http://www.almhuette-raith.at/apache-log/

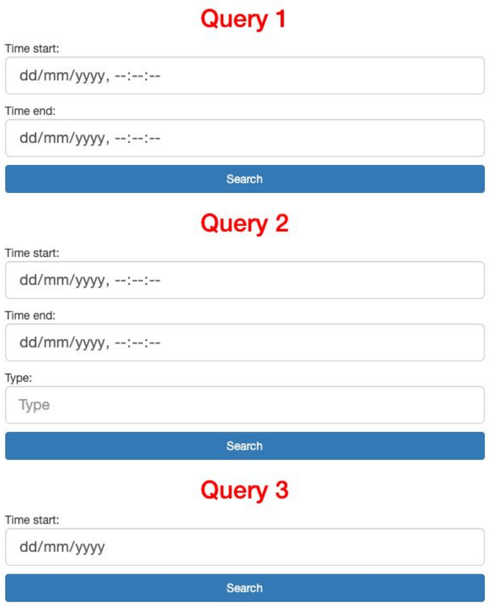
Http Method: GET

Resource: /apache-log/access.log

Status: 206

« first previous Page 2 of 31348. next last »

- Canned queries page:



Query 1

Time start:

dd/mm/yyyy, --:--:--

Time end:

dd/mm/yyyy, --:--:--

Query 2

Time start:

dd/mm/yyyy, --:--:--

Time end:

dd/mm/yyyy, --:--:--

Type:

Type

Query 3

Time start:

dd/mm/yyyy

Build and Run:

To run the application you can simply run:

```
docker run --rm   --name pg-docker -e POSTGRES_PASSWORD=root -d -p 5432:5432 -v
$HOME/docker/volumes/postgres:/var/lib/postgresql/data  postgres
```

With the command above you will start the database that the application will communicate with.

Or edit the application.properties file to match you postgres configuration.
(Also in application.properties you should have
`spring.jpa.hibernate.ddl-auto=create` *the first time you run to insert the data in the database.*
After the first run you can change it to none and comment out the parsing and inserting code from LogDbApplication.java.)

Then you can start the application with gradle using:

./gradlew bootRun