MICROSERVICES
ON RAILS

@fbarriosCL

ARCHDAILY

¿Qué es archdaily?

ArchDaily es el **sitio web** de arquitectura más visitados del mundo.

169 million

Monthly Pageviews

13 million

Monthly Visits

3 million

Facebook Fans

230 thousand

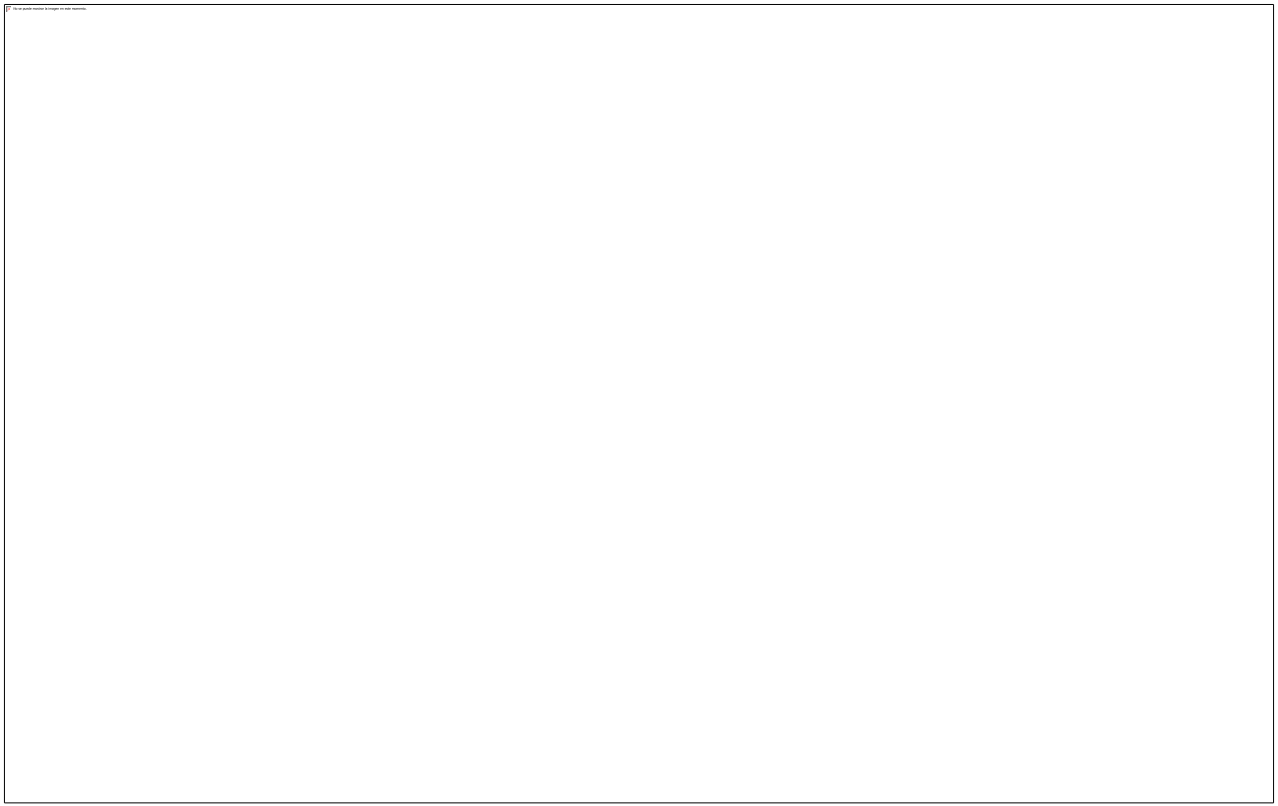
Newsletter Subscribers

Source: Google Analytics and Mailchimp

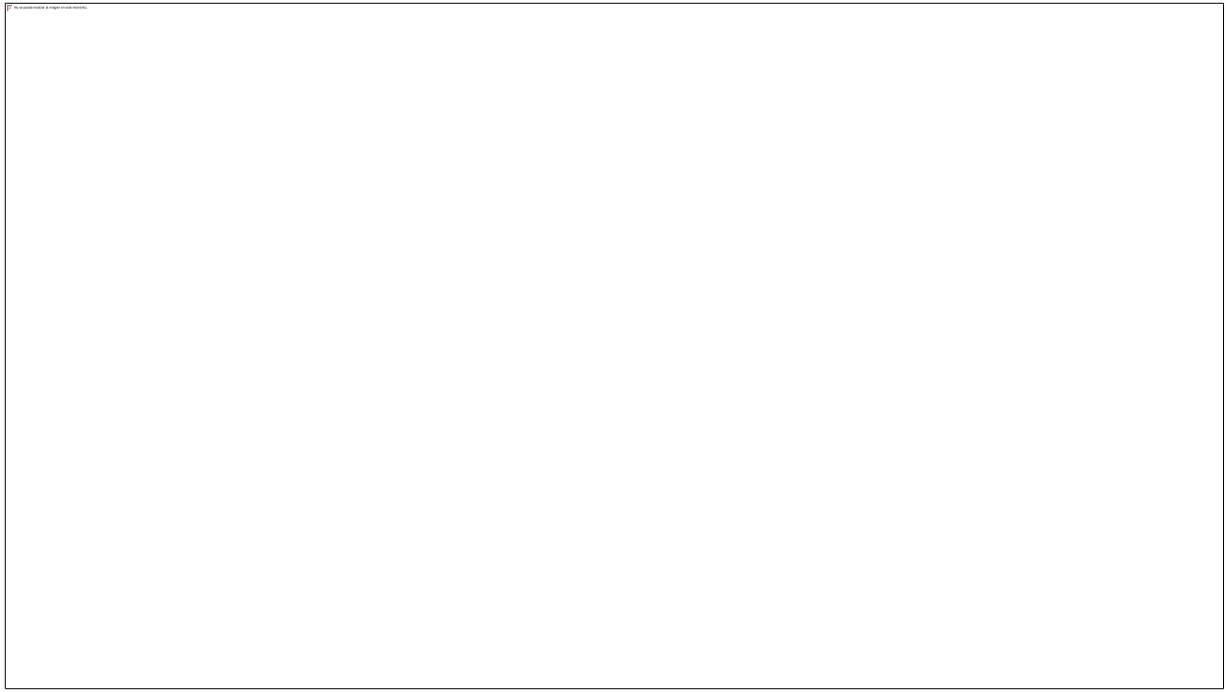
Source: Ranking Alexa

FOUNDERS

*En marzo de **2008** partimos a Nueva York para presentarlo y en **2009** ya éramos el sitio más leído del mundo*



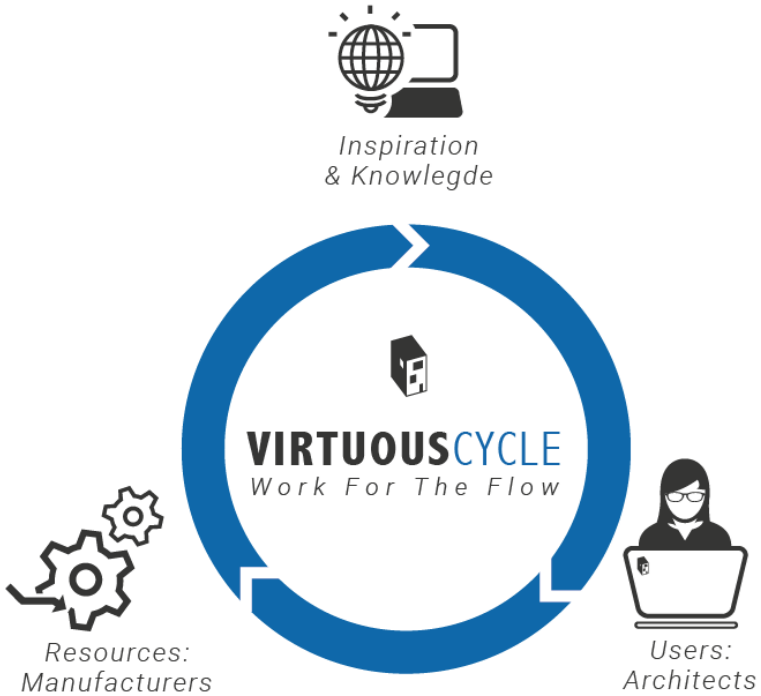
DEVELOPER



TEAM

ARCHDAILY

¿Qué hacemos?



ARCHDAILY

¿Qué es archdaily?

ADVERTISING

WE LOVE WHAT DO

CASO DE USO

Necesitamos conocer las **métricas** de nuestros **clientes**.

TEAM 1

LEAD DEVELOPER

Pablo Acuña.

https://github.com/pacuna/partitionable

https://github.com/pacuna/dckerize

http://pacuna.io/

169 million

Monthly Pageviews

13 million

Monthly Visits

3 million

Facebook Fans

230 thousand

Newsletter Subscribers

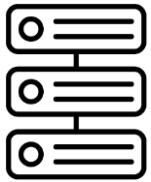
Source: Google Analytics and Mailchimp

Source: Ranking Alexa

FIRST ITERATION

FIRST ITERATION

Obtener métricas de nuestros servicios, arquitectura monolítica.



- Construimos una aplicación **RAILS API** llamada **neufert-api**.
- ***Milestone***: **comunicar**, **obtener** y **procesar** las métricas de los servicios en **tiempo real**.

SERVICES

FIRST PROBLEM ITERATION

Arquitectura monolítica.

- El obtener las métricas para un **cliente** entregaba un **timeout** sobre 2 +minutos.
- Poco **eficiente** y **escalable**.

SECOND ITERATION

MICRO SERVICES

MICRO SERVICES

Arquitectura micro servicios.

Una “***arquitectura de microservicios***” es un enfoque para desarrollar una aplicación software como una **serie de pequeños servicios** que corresponden a un área de **negocio** de la aplicación, cada uno ejecutándose de forma **autónoma** y comunicándose mediante **API**.

ADVANTAGE MICRO SERVICES

Arquitectura micro servicios.

- Es un enfoque al momento de desarrollar una aplicación, como una serie de **pequeños servicios**.
- Comunicación mediante **API**.
- Independiente su codigo debe ser **despegado sin afectar** al **resto** de las aplicaciones.
- Pequeños **codigos** no debe tomar mas de dos semanas en escribir.
- Se pueden escribir en **múltiples lenguajes** de programación

OTHERS ADVANTAGE

Arquitectura micro servicios.

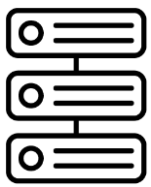
- No existen **reglas** ni **tamaño**.
- Facil de **entender**.
- Facilita la **gestión** de equipos **multifuncionales** e **independientes**.
- Facil de **escalar**.

“Tu arquitectura es reflejo de la organización de tu equipo y visceversa”

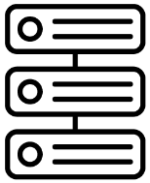
LET'S DO IT

SECOND ITERATION

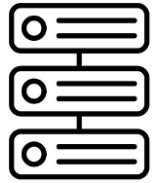
First Step: store data



store-dfp



store-ga



store-mc

Stack:

- Rails 5 API
- Postgresql
- https://github.com/pacuna/partitionable

Objetivos

- Persistencia de datos.

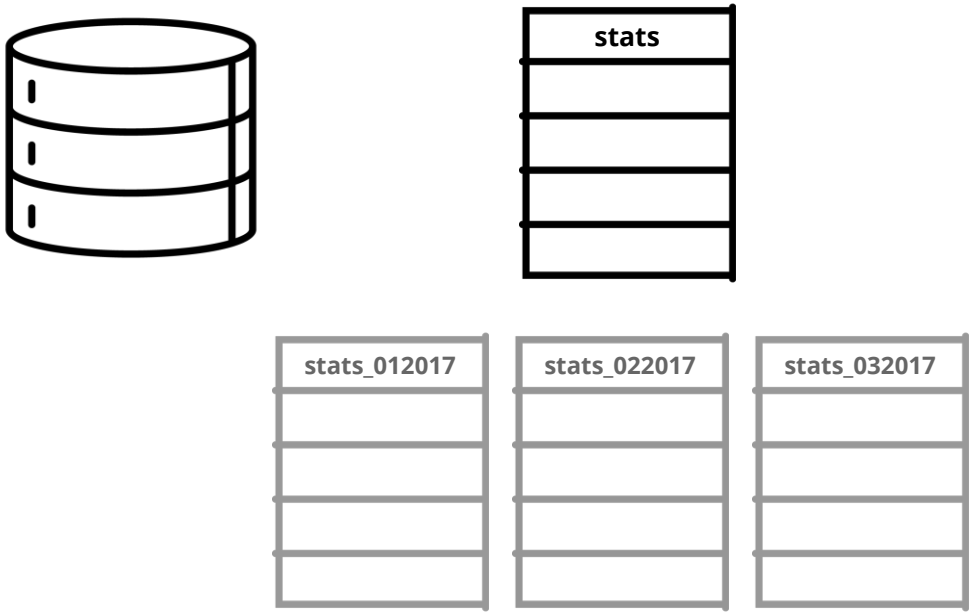
PARTITIONABLE

First Step: store data

- PostgreSQL supports basic table partitioning.
- https://www.postgresql.org/docs/9.1/static/ddl-partitioning.html
- https://github.com/pacuna/partitionable

PARTITIONABLE

First Step: store data

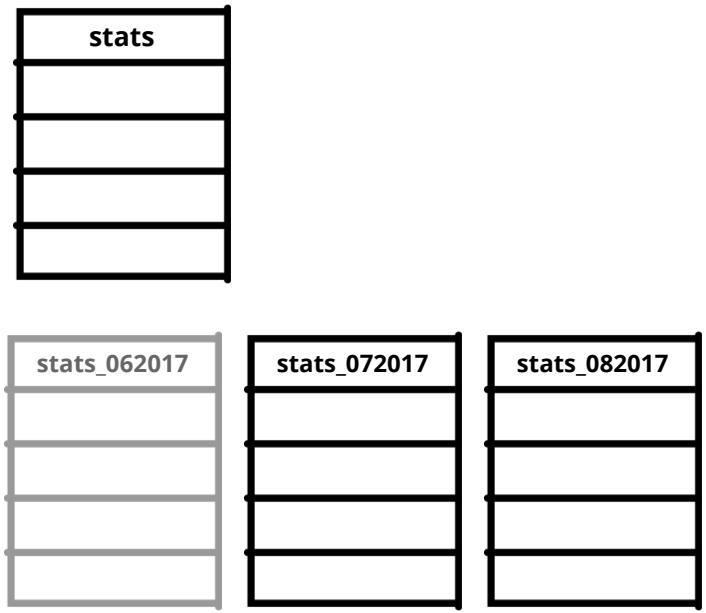


Particionable asume que el modelo que desea particionar, y tiene un atributo de **fecha**.

PARTITIONABLE

Store data

```
Stats.where(date: Date.today..Date.today-1.months)
```



STORE MC

For example…

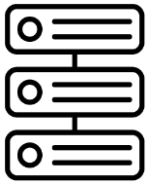
```ruby
class Mailchimp
  def initialize
    @gibbon = Gibbon::Request.new(api_key: Settings.mailchimp.api_key)
  end

  def get_campaigns_between(site, start_date, end_date, type, count=5000)
    @gibbon.campaigns
    .retrieve(params: { offset: 0, count: count, since_send_time: from, before_send_time: to })
      .select{ |campaign| campaign['title'] }
    .map{|s| ... }
  end

  def get_campaign_for_site(site, date, type = 'regular')
    final_date = Date.strptime(date, "%Y-%m-%d") + 1.day
    get_campaigns_between(site, date, final_date.strftime("%Y-%m-%d"), type)
  end

  def self.sites
    %w(br mx co pe cl us)
  end
end
```

STORE MC

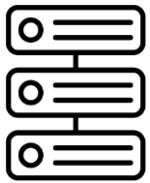
```ruby
class CampaignStat < ApplicationRecord
    def initialize
    @mailchimp = Mailchimp.new
    @yesterday = Date.today-1.day
    @campaigns = {}
  end
  # For each market, get the last logdate and fetch the
  # campaigns data from that date until yesterday and save it to the db
  def fetch_campaigns_and_saved
    Mailchimp.sites.each do |site|
      date = last_logdate_saved(site)
      date.upto(@yesterday) do |day|
        @campaigns[site] = @mailchimp.get_campaign_for_site(site, day, 'regular')
        create_from_mailchimp(@campaigns[site], site) if @campaigns[site].present?
      end
    end
  end

  # this method should receive a site and return the last logdate saved in the db.
  def last_logdate_saved site
    return false unless Mailchimp.sites.include?(site)
    campaignstat = CampaignStat.where(site: site).order(logdate: :asc).last
    return campaignstat.logdate unless campaignstat.nil?
    return Date.today-2.year if campaignstat.nil?
  end
end
```

SECOND ITERATION

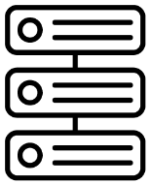
Second Step: process data



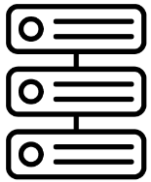
store-dfp



store-ga



stats-middleware



store-mc

Stack

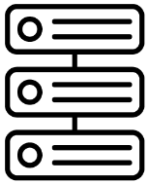
- Rails 5 API
- S/N DB

Objetivos

- Procesar stats(datos) y transformar en información.
- Entrega todas las stats listas para ser utilizadas REST.
- Datos centralizados.

SECOND ITERATION

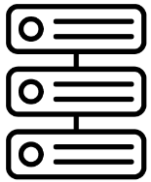
Third step: show data



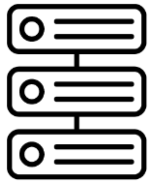
store-dfp

KET - Kam efficient Tools

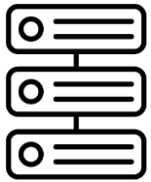
store-ga



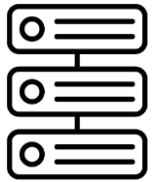
stats-middleware



Dashboard



store-mc



APM - Account Performance Monitoring

STATS MIDDLEWARE

STATS MIDDLEWARE

Resumen de Métricas

Productos

Proyectos Vinculados

Contactos

Resumen de Métricas

Período Reporte 21 Meses

5.503.321
Exposición de Marca

497.940
Detalle de Proyectos Vinculados

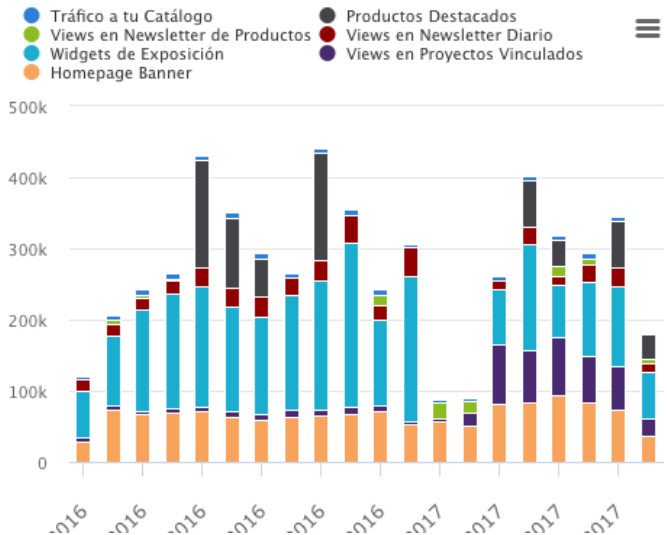
133.010
Tráfico a tu Catálogo

2.331 Hrs.
Tiempo de Interacción

ENVÍANOS TU FEEDBACK

Exposición de Marca

Tráfico a tu Catálogo



Top 6 productos

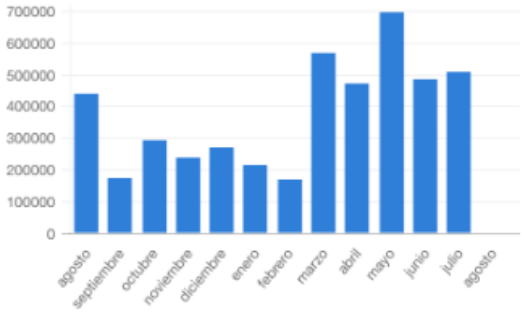
Producto	Tráfico	Tiempo Promedio
1 Tipos de Cristal	16.434	04: 23 min
2 Cortinas de Cristal	13.386	03: 18 min
3 Termopaneles	11.549	03: 12 min
4 Pisos de Cristal	6.056	03: 40 min
5 Ventanas y Puertas de PVC	5.968	02: 16 min
6 Fachadas y Muros Cortina	5.509	03: 24 min

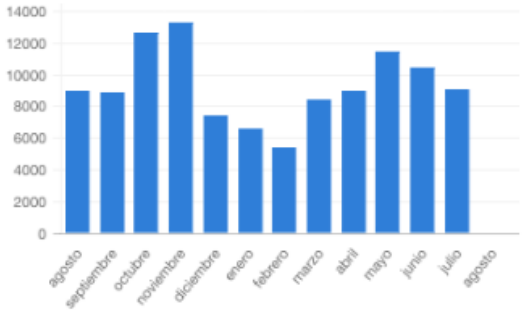
Customer Success Manager

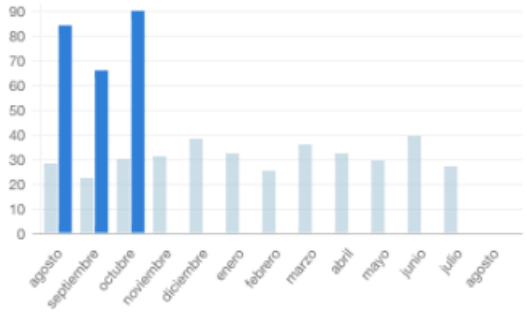
Macarena Assadi

macarenaassadi@archdail...

Resultados generales







Exposición de Marca

Widget explora el catálogo:	**1.024.046**
Widget materiales utilizados en esta obra:	**336.097**
Proveedores en especificaciones de proyectos:	**980.524**
Views de Newsletter Diario:	**173.254**
Views de Newsletter de Productos:	**145.173**
Total de views de banners:	**1.955.488**
Tráfico de tu catálogo:	**111.551**
Views de Facebook:	*8.974*

Tráfico de tu catálogo

Pageviews de Fichas:	**101.125**
Pageviews de Landing:	**10.134**
Tiempo promedio:	**00h 02m 19s**
Tiempo Total:	**2119h 36m 58s**
Interacciones en facebook:	*292*

Interacciones profundas

Cotizaciones

Fichas Activas:	**132**
Clicks en botón Cotizar / Contactar:	**240**
Cotizaciones por Formulario:	**369**

CONCLUSIÓN

- Fácil mantenimiento, códigos pequeños.
- Difícil debugeo, encontrar un error vas de endpoint en endpoint hasta encontrar el punto y coma.

THANK YOU

MICRO SERVICES

HOW CREATE DEVELOPMENT ENVIRONMENT

CREATE MICRO SERVICE

- rails new myapp --database=postgresql

CREATE MICRO SERVICE

- https://github.com/pacuna/dckerize
- gem install dckerize

CREATE MICRO SERVICE

- username: myapp
  password: mysecretpassword
  host: myapp-db

CREATE MICRO SERVICE

- cd myapp
- dckerize up myapp

CREATE MICRO SERVICE

- $ docker-compose build
  $ docker-compose up

CREATE MICRO SERVICE

```yaml
version: '2'
services:
  nginx-proxy:
    image: jwilder/nginx-proxy
    container_name: nginx-proxy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /var/run/docker.sock:/tmp/docker.sock:ro
    networks:
      - default
      - archdaily
networks:
  archdaily:
    driver: bridge
```

CREATE MICRO SERVICE

```
myapp [master] ⚡cat /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost

192.168.99.100      newsroom2.dev.archdaily.com
192.168.99.100      neufert.dev.archdaily.com
192.168.99.100      kenneth.dev.archdaily.com
192.168.99.100      billing.dev.archdaily.com
192.168.99.100      ket.dev.archdaily.com
192.168.99.100      alfred.dev.archdaily.com alfred.dev.plataformaarquitectura.cl
192.168.99.100      my.dev.archdaily.com
192.168.99.100      nimrod.dev.archdaily.com
192.168.99.100      store-mc.dev.archdaily.com
192.168.99.100      store-dfp.dev.archdaily.com
192.168.99.100      store-dfp2.dev.archdaily.com
192.168.99.100      neufert-dashboard.dev.archdaily.com
192.168.99.100      stats-middleware.dev.archdaily.com
192.168.99.100      neufert-public.dev.archdaily.com
192.168.99.100      search.dev.archdaily.com
192.168.99.100      chrome-notifications.dev.archdaily.com
192.168.99.100      neufert-admin.dev.archdaily.com
192.168.99.100      advertising.dev.archdaily.com
192.168.99.100      statham.dev.archdaily.com
192.168.99.100      apm.dev.archdaily.com
192.168.99.100      nessus.dev.archdaily.com
192.168.99.100      quantum.dev.archdaily.com
192.168.99.100      burp-suite.dev.archdaily.com

192.168.99.100      plataform.dev.backtrackacademy.com
```

THANK YOU