

dfaſjñdañjdsfafa

## Capítulo 5

# Despliegue en la nube

Párrafo de entradilla, aclarando que para desplegar la aplicación web desarrollada se ha elegido una tecnología de computación en la nube, Amazon Web Services.

### 5.1. Computación en la nube

[Esto a una sección propia del capítulo 1]

Otra de las tecnologías usadas en este TFG es la computación en la nube. Esta computación

La computación en la nube es un paradigma que permite ofrecer servicios de computación a través de internet. Cuando los proveedores utilizan la palabra cloud, se refieren a la posibilidad de configurar y redimensionar los recursos que se usan de forma rápida y sencilla, o manualmente vía web o usando APIs REST.

Este tipo de servicios son tan dinámicos que se cobran por horas o minutos dependiendo de la plataforma de computación en la nube. Además los recursos de computación en la nube suelen estar virtualizados, aunque en algunas ocasiones pueden ser máquinas físicas.



Figura 5.1: Proveedores más conocidos de computación en la nube

Los proveedores de la computación en la nube ofrecen diferentes tipos de servicios, tanto de bajo nivel como de alto nivel.

1. Servidores virtuales (Instancias)
2. Gestión del sistema operativo que tendrán los servidores (Imagen)
3. Sistemas de copia de seguridad de los servidores completos
4. Balanceadores de carga entre servidores

5. Bases de datos administradas
6. Servicios de gestión de logs, monitorización y alarmas
7. Plataforma auto-escalable para ejecución de aplicaciones

Los servicios ofrecidos por los proveedores pueden ser de diferentes tipos de abstracción:

separadas por espacio en blanco

1. **Infraestructura como servicio (IaaS):** IaaS proporciona acceso a recursos informáticos situados en un entorno virtualizado, la "nube" (cloud), a través de una conexión pública. En el caso de IaaS, los recursos informáticos ofrecidos consisten en infraestructura de procesamiento. La definición de IaaS abarca aspectos como el espacio en servidores virtuales, conexiones de red, ancho de banda, direcciones IP y balanceadores de carga.
2. **Plataforma como Servicio (PaaS):** El modelo PaaS permite a los usuarios crear aplicaciones de software utilizando herramientas suministradas por el proveedor. Los servicios PaaS pueden consistir en funcionalidades preconfiguradas a las que los clientes puedan suscribirse, eligiendo las funciones que deseen incluir para resolver sus necesidades y descartando aquellas que no necesiten.
3. **Software como Servicio (SaaS):** El modelo SaaS se conoce también a veces como "software bajo demanda", y la forma de utilizarlo se parece más a alquilar el software que a comprarlo. Con las aplicaciones tradicionales, el software se compra al principio como un paquete, y una vez adquirido se instala en el ordenador del usuario. La licencia del software puede también establecer limitaciones en cuanto al número de usuarios y/o dispositivos en los cuales puede instalarse.

## 5.2. Amazon web services

[todo esto a capítulo 3!]



Figura 5.2: AWS

Amazon Web Services es el proveedor más famoso y más completo en infraestructura como servicio (IaaS), ya que ofrece un conjunto de servicios y un modelo de precios muy completo y que se ajusta a las necesidades de cada cliente.

Dispone de varios tipos de instancias según su hardware:

1. **Instancias estándar:** Pequeñas, medianas, grandes y extra-grandes
2. **Instancias con gran cantidad de memoria**

3. Instancias con CPU de alto rendimiento
4. Instancias en cluster: Con redes de alta velocidad entre ellas
5. Instancias de GPU para clústeres

Además Amazon Web Services dispone de diferentes servicios entre los que destacamos:

1. **Amazon Elastic Block Store:** Disco duro de las instancias persistente, donde sus datos permanecen cuando la instancia se apaga.
2. **Varias ubicaciones:** El cliente puede elegir la ubicación para reducir la latencia a los usuarios de los servicios. Además, existen varias zonas de disponibilidad dentro de la misma ubicación para minimizar el impacto de las catástrofes.
3. **Direcciones Elastic IP:** Por defecto las instancias tienen IPs internas en la red de Amazon, asociando una IP pública a una instancia.
4. **Amazon Cloud Watch:** Servicio de monitorización de instancias con sistema de alarmas y gráficas de uso de recursos (memoria, CPU, red...)
5. **Auto Scaling:** Se pueden configurar reglas para que Amazon inicie más instancias cuando la carga de las existentes supere un umbral y volver a bajar cuando la carga disminuya.
6. **Elastic Local Balancer:** Dispositivos que reparten las peticiones web a cada una de las instancias que se han creado con el escalado automático o manual.
7. **Imagenes de Instancias (AMI):** Amazon permite la gestión de imágenes de las instancias (AMI), pudiendo crear y gestionar varias imágenes. Se puede iniciar una instancia con cualquier imagen.

### ~~5.2.1. ¿Por qué AWS?~~

~~La cuestión de porque H~~emos elegido ~~A~~WS como servicio de computación en la nube no es otra que por el gran impacto que está teniendo en el mundo laboral, al ~~margen que~~ sus competidores directos como son Azure y Google Cloud.

Otro factor que juega a favor de AWS con respecto a sus competidores es que a la hora de desplegar una aplicación en la nube la experiencia de usuario ~~me~~ resulta mucho más intuitiva que la de sus competidores. También debo destacar la importancia de ~~tener~~ ofrecer un buen soporte ~~como del que AWS dispone~~, donde en cualquier momento te resuelven las posibles dudas ~~que tengas~~ a la hora de desplegar la aplicación.

Por último y como factor de bastante importancia para aquellos pequeños emprendedores que quiera empezar a desarrollar sus ideas, es el bajo coste que supone tener una aplicación desplegada en la nube de ~~A~~WS. Por el momento, el primer año de tu cuenta en AWS es gratuita, restringida a ciertas limitaciones.

### ~~5.2.2. ¿Quién lo utiliza?~~

Cada vez son más las empresas que deciden utilizar la computación en la nube ya que desde 2006, ~~X~~ Amazon Web Services proporciona servicios de infraestructura de TI para empresas en forma de servicios web.

Uno de los principales beneficios de la computación en la nube es la oportunidad de reemplazar importantes gastos de infraestructura con costes reducidos que se escalan dependiendo de la dimensión de su negocio.

Gracias a la nube, las empresas ya no tienen que planificar ni adquirir servidores y otra infraestructura de TI con semanas o meses de antelación. Pueden disponer en cuestión de minutos de cientos o de miles de servidores y ofrecer resultados más rápidamente.

Hoy en día, Amazon Web Services proporciona una plataforma de infraestructura escalable, de confianza y de bajo costo en la nube que impulsa cientos de miles de negocios de 190 países de todo el mundo. Con centros de datos en Estados Unidos, Europa, Brasil, Singapur, Japón y Australia. **Por ejemplo:**

~~A continuación enumeraremos algunas de las empresas con más éxito en AWS.~~

1. **Amazon.com:** Es el minorista online más grande del mundo. En 2011, Amazon.com pasó de utilizar el backup en cinta a usar Amazon S3 en la cloud para realizar copias de seguridad de la mayoría de las bases de datos de Oracle de las que se encarga. Mediante el uso de AWS, Amazon.com logró eliminar el software de backup y experimentó una mejora de desempeño 12 veces mayor, de forma que pudo reducir el tiempo de restablecimiento de 15 a 2,5 horas aproximadamente en situaciones seleccionadas.
2. **Netflix:** El referente de la televisión en streaming ~~X~~ usa AWS para proporcionar miles de millones de horas de vídeo casa mes a sus ~~más~~ de 60 millones de suscriptores. Así puede hacer uso de miles de servidores y terabytes de almacenamiento en cuestión de minutos para que sus usuarios puedan ver series y películas desde cualquier parte del mundo en sus tabletas o teléfonos móviles.
3. **Dropbox:** El famoso y conocido servicio de alojamiento de archivos multiplataforma en la nube ~~X~~ utiliza hasta el momento AWS como repositorio para almacenar todos los archivos que los usuarios de Dropbox suben a la red.
4. **Bankinter:** Utiliza AWS como ~~en~~ su aplicación de simulación de riesgo crediticio, pasando de 23 horas a 20 minutos. **[redacción regular, mejorala]**
5. **FC Barcelona:** Su sitio web, que aloja más de 6 000 páginas y 12.000 fotos digitales, también usa AWS para su mantenimiento.
6. **Harvard Medical School** Desarrolla nuevos modelos de pruebas de genomas en tiempo récord.
7. **Mapfre:** Ahorró 1,3 millones de euros en infraestructura y redujo el desarrollo de semanas a días

### 5.3. Lanzar una instancia en AWS

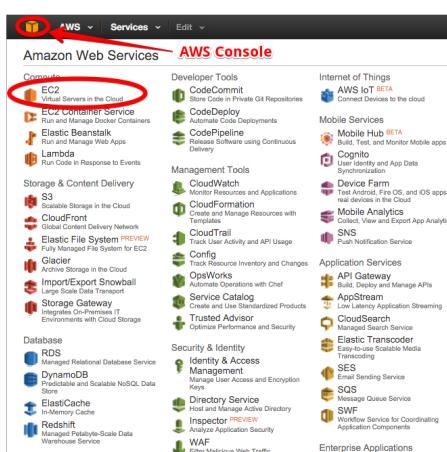
Ahora que ya sabemos en detalle de lo que consiste AWS, ~~antes de poder~~ subir nuestra aplicación a producción debemos de crear una instancia en AWS.

Estos son los pasos a tener en cuenta a la hora de crear una instancia en AWS:

- **Paso 1. Crear Key Pairs** Los Key Pairs se utilizan para iniciar sesión de forma segura en los servicios de AWS. Crearemos un Key Pairs para acceder a nuestra instancia de EC2.

se debe navegar

1. Para crear nuevos pares de claves, ~~navegue~~ hasta AWS Console y luego ~~haga~~ clic en EC2.



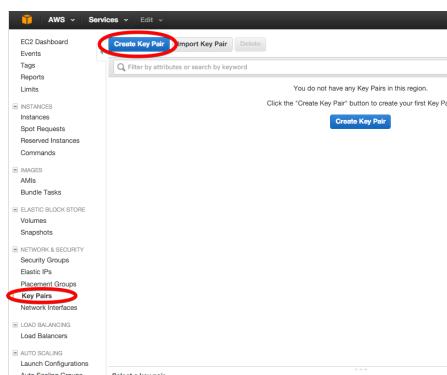
Un poco más grande, tan pequeño no se lee bien.

[cuidado con los corta-pegas de internet, retoca la redacción]

Figura 5.3: Crear Key Pairs Paso 1

se debe hacer

2. En el panel izquierdo, ~~haga clic~~ en Key Pairs, luego ~~haga~~ clic en Crear Key Pairs



un poco más grande

Figura 5.4: Crear Key Pairs Paso 2

Se debe introducir

la

hacer

3. Ingrese un nombre para ~~la~~ clave, luego ~~haga~~ clic en Crear Key Pairs. El Key Pairs se descargará automáticamente. Debe mover ~~esta~~ clave a un directorio diferente.

se

**Se**  
Importante: Deberá cambiar los permisos de esta clave para que sean de solo lectura, consulte el siguiente código:

```
chmod 400 youKeyName.pem
```

- **Paso 2. Lanza una instancia de EC2 con Bitnami** En este paso, lanzaremos una instancia de EC2 desde Amazon Machine Image (AMI).

Con AMI, <sup>se</sup> puede activar una instancia de EC2 que esté lista para el desarrollo sin demasiada configuración. Bitnami proporciona una imagen MEAN preconfigurada, que usaremos para configurarlo rápidamente.

1. Primero, navegue a la consola de AWS, haga clic en AWS Marketplace.

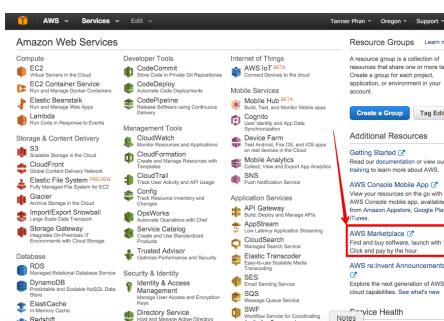


Figura 5.5: Lanza una instancia Paso 1

2. Search for MEAN powered by Bitnami, then select the 64-bit AMI to continue.

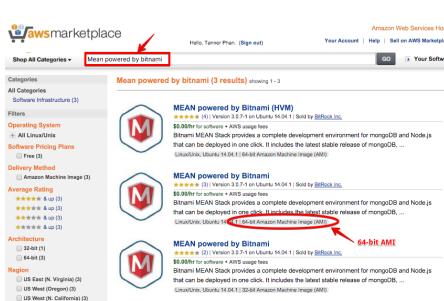


Figura 5.6: Lanza una instancia Paso 2

3. En Pricing Details con el fin de obtener la mejor velocidad de entrega, seleccione la región más cercana y luego haga clic en Continuar.

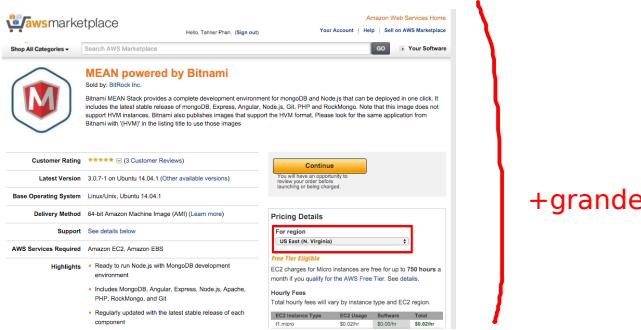


Figura 5.7: Lanza una instancia Paso 3

4. Para el grupo de seguridad, **Asegurarse** Crear nuevo, según la configuración del vendedor.
5. **Asegúrese** de tener los siguientes métodos de conexión:

- 1. SSH , My IP
- 2. HTTP , Anywhere
- 3. HTTPS , Anywhere

**1-Click Launch**  
Review, modify, and launch      **Manual Launch**  
With EC2 Console, APIs or CLI

Click "Launch with 1-Click" to launch this software with the settings below

The default settings are provided by the software seller and AWS Marketplace.

▶ **Version**  
3.0.7-1 on Ubuntu 14.04.1, released 11/20/2015

▶ **Region**  
US West (N. California)

▶ **EC2 Instance Type**  
m1.small

▶ **VPC Settings**  
Will launch into: subnet-ddfb95b8

▼ **Security Group**

**Updated:** Due to a change in other settings, security group settings is updated. A security group acts as a firewall that controls the traffic allowed to reach one or more instances. Learn more about Security Groups.

You can create a new security group based on seller-recommended settings or choose one of your existing groups.

**Description:**  
A new security group will be generated by AWS Marketplace. It is based on recommended settings for MEAN powered by Bitnami version 3.0.7-1 on Ubuntu 14.04.1 provided by BitRock Inc.

Connection Method	Protocol	Port Range	Source (IP or Group)
SSH	tcp	22 - 22	My IP 108.179.160.127/32
HTTP	tcp	80 - 80	Anywhere 0.0.0.0/0
HTTPS	tcp	443 - 443	Anywhere 0.0.0.0/0

**Warning:**  
Rules with source of 0.0.0.0/0 allows all IP addresses to access your instance.  
We recommend limiting access to only known IP addresses.

Figura 5.8: Lanza una instancia Paso 5

6. Seleccione la Key Pair que creaste en el paso 1.



Figura 5.9: Lanza una instancia Paso 6

**hacer**

- Finalmente, haga clic en Iniciar para iniciar la instancia.

- Paso 3. Conéctate a tu EC2** Para conectarte por SSH a su instancia, necesitará la IP pública de su instancia y el Key Pair que creó.

**hacer**

- Dentro de EC2, haga clic en Instancias, seleccione la instancia recién iniciada en el paso anterior y luego haga clic en Conectar.
- Copiar el código que aparece marcado en rojo en la siguiente imagen.

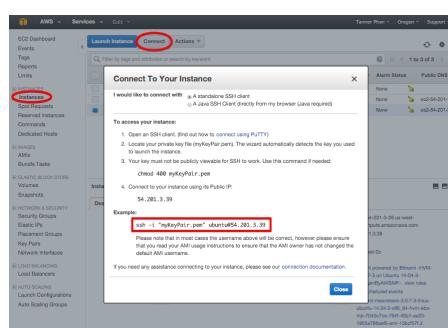


Figura 5.10: Conéctate a tu EC2 Paso 2

**navegar**

- En su terminal, navegue hasta el directorio donde está guardado su par de claves, luego pegue el último paso del código.

```
The authenticity of host '54.201.14.233 (54.201.14.233)' can't be established.
ECDSA key fingerprint is SHA256:IT5axDl1G0HtY8Uhi1OB7NqM8EtH3B07LzR8F/f1T/xE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.201.14.233' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-71-generic x86_64)

*** Bitnami MEAN 3.0.7-3
*** Bitnami Wiki: https://wiki.bitnami.com/
*** Bitnami Forums: https://community.bitnami.com ***
bitnami@ip-172-31-1-86:~$
```

+ grande

Figura 5.11: Conéctate a tu EC2 Paso 3

- **Paso 4. Conseguir un dominio gratuito** Una vez que ya tenemos la instancia creada y hemos sido capaces de conectarnos a ella por SSH, llega el momento de conseguir un dominio para que los usuarios se puedan conectar a la aplicación lo más fácilmente posible. El dominio que he utilizado es un dominio gratuito, proporcionado por el portal freedom.com, la forma de conseguirlo es la siguiente:

1. Vaya a <https://my.freenom.com> e inscríbase.

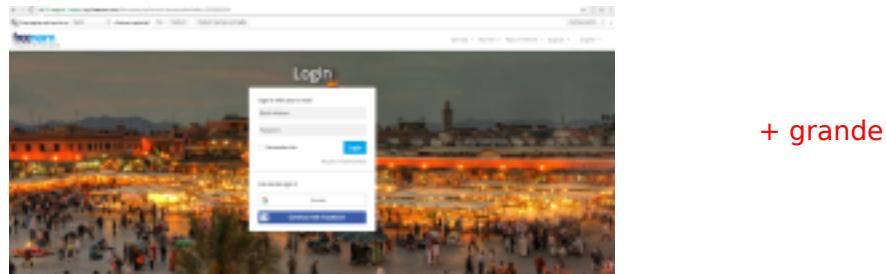


Figura 5.12: Paso 1.1

2. Una vez que nos hemos logueado, tenemos que ir a la sección de My Domains y allí elegir un dominio que esté disponible.



Figura 5.13: Paso 1.2

3. En nuestro caso, el dominio elegido es [www.classcity.tk](http://www.classcity.tk). Una vez creado nuestro dominio, comenzamos la configuración haciendo click en el botón Manage Domain.

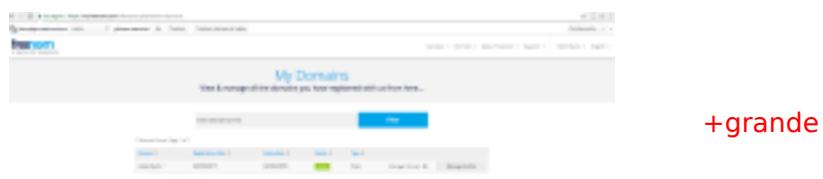


Figura 5.14: Paso 1.3

**de la gestión**

4. Dentro del Management del dominio, nos vamos a la sección Manging Tools, y hacemos click en name server tal y como se puede ver en la siguiente imagen,

\texttt{}

63



Figura 5.15: Paso 1.4

5. Debe cambiar los servidores de nombres y seleccionar usar servidores de nombres personalizados.



Figura 5.16: Paso 1.5

- **Paso 4. Obtener un certificado SSL gratis con AWS Certificate Manager para CloudFront**

Un certificado SSL es un fichero informático generado por una entidad de servicios de certificación que asocia unos datos de identidad a una persona física, organismo o empresa, confirmando de esta manera su identidad digital en Internet. Necesitamos un certificado SSL para que los usuarios que accedan a nuestra página lo hagan de una forma segura por el protocolo HTTPS.

Por otro lado tenemos Amazon CloudFront, es un servicio de red de entrega de contenido (CDN) global que proporciona datos, vídeos, aplicaciones y API de forma segura a sus espectadores con baja latencia y altas velocidades de transferencia.

se

Para conseguir el certificado SSL en AWS necesitas seguir los siguientes pasos:

1. Tienes que ir a AWS manager certificate desde la consola de Amazon web services y hacer click en get started.  
*texttt*
2. A continuación te pedirá el dominio creado previamente y tendrás que hacer click en Review and request.
3. Una vez enviada la solicitud de certificado a la Autoridad certificadora, un email de confirmación será enviado a nuestra cuenta de correo vinculada con el dominio, es decir a admin@classcity.tk. En nuestro caso como disponemos de un dominio gratuito debemos hacer una redirección desde el portal <http://www.tkmalias.tk/es/pageA00E1.html>, y desde allí conseguir redireccionar nuestra cuenta admin@classcity.tk a un correo personal, para recibir la confirmación del certificado.

4. Una vez que recibimos un correo como el que aparece en la siguiente imagen, realizamos la confirmación del certificado haciendo click en el enlace que viene en el correo



\texttt{}

Figura 5.17: Correo de confirmación

Una vez que tenemos el certificado SSL generado es momento de configurar cloudfront proporcionándole de dicho certificado, para ello debemos seguir los siguientes pasos:

1. Buscar cloudfront desde la consola de AWS.
2. Una vez allí debemos crear una distribución
3. En el momento de elegir la configuración del SSL certificate, marcaremos la opción Custom SSL Certificate, tal y como podemos ver en la siguiente imagen. **5.18**



Figura 5.18: SSL

Si ahora vamos a nuestro dominio e introducimos antes el protocolo HTTPS, es decir en nuestro caso <https://www.classcity.tk>, veremos que efectivamente accedemos a nuestra aplicación de una forma segura. (**Figura 5.19**)



Figura 5.19: SSL

## 5.4. Lanzar una aplicación MEAN en producción sobre AWS

Una vez que dispongamos de una instancia, un dominio, un certificado SSL y una distribución cloufront, tenemos todo listo para poder desplegar nuestra aplicación MEAN en producción. Para ello debemos seguir los siguientes pasos:

1. Conectarnos desde nuestra terminal por SSH a nuestra máquina virtual de AWS

```
sudo ssh -i /.ssh/keypair.pem root@ipinstance
```

2. Clonar en la máquina virtual de AWS nuestro repositorio git donde almacenemos la aplicación MEAN.

```
rooti@ip sudo git clone https://github.com/RoboticsURJC-students/2016-tfg-Mario-Fernandez.git
```

3. Instalar dependencias de la aplicación web por parte del Front End lado cliente

```
rooti@ip cd 2016-tfg-Mario-Fernandez  
rooti@ip /2016-tfg-Mario-Fernandez sudo npm install
```

4. Correr Angular2 con angular-ci

```
rooti@ip /2016-tfg-Mario-Fernandez sudo npm run build
```

5. Una nueva carpeta se crea al terminar el proceso ./dist, debemos copiarla entera en nuestra carpeta htdocs de apache.

```
rooti@ip /2016-tfg-Mario-Fernandez sudo cp -r ./dist/* ../httdocs
```

6. Ahora deberíamos poder ir a <https://www.classcity.tk> y poder ver nuestra aplicación Angular2 corriendo en producción.



Figura 5.20: <https://www.classcity.tk>

7. No olvidemos que en el stack MEAN, aún falta que configuremos el servidor en producción. Para ello lo primero es instalar las dependencias necesarias para el backend. lado del servidor.

```
rooti@ip /2016-tfg-Mario-Fernandez cd backend  
rooti@ip /2016-tfg-Mario-Fernandez/backend sudo npm install
```

### servidor

8. Antes de ponernos a ejecutar nada en la parte del ~~backend~~, debemos configurar ciertas rutas en ~~nuestro~~ servidor apache que va ser el encargado de recibir la peticiones de entrada. Para ello debemos hacer lo siguiente:

- Editamos el archivo de configuración de apache llamado httpd.conf para que cuando llegue las solicitudes de tipo / app / \* redirigir a localhost: 8080, que será donde estará escuchando nuestra aplicación ~~de gestión de clases particulares~~

```
ProxyPassMatch ^/app/(.*)$ http://localhost:8080/$1  
ProxyPass /app/(.*)$ http://localhost:8080/  
ProxyPassReverse /app/(.*)$ http://localhost:8080/
```

- Y cuando llega la solicitud de type / socket / \* redirigir a localhost: 8000, que será el puerto donde este escuchando nuestro chat.

```
ProxyPassMatch ^/socket/(.*)$ http://localhost:8000/  
$1  
ProxyPass /socket/(.*)$ http://localhost:8000/  
ProxyPassReverse /socket/(.*)$ http://localhost:8000/
```

### lado servidor

9. A continuación en el ~~backend~~ correremos la base de datos con la propiedad screen, para que cuando se termine la conexión por ssh el proceso siga corriendo y no se corte.

```
rooti@ip /2016-tfg-Mario-Fernandez/backend sudo mkdir data  
rooti@ip /2016-tfg-Mario-Fernandez/backend sudo screen  
mongod --dbpath ./data
```

10. Y por último ejecutamos el fichero ~~server.js~~ también con la propiedad screen, y comprobaremos que funciona correctamente.

```
rooti@ip /2016-tfg-Mario-Fernandez/backend sudo node server  
.js screen
```