

# Proyecto Tumblr

**Proyecto DAW 16/17**

**Autor: Cristopher Quispe**

**INS RIBERA BAIXA 2**

**01 de Junio del 2016 El Prat de Llobregat**

# Índice

1. Proyecto .....	3
1.1 Descripción.....	3
1.2 Motivación .....	3
2. Meteor.....	3
2.1 ¿Qué es Meteor? .....	4
2.2 El pasado .....	4
2.3 El Futuro.....	4
2.4 Como funciona .....	5
2.5 Mongo.....	6
2.6 Componentes.....	8
3. Mockups.....	9
4. Código.....	13
5. Conclusión .....	13
6. Bibliografía .....	13
6.1 Libros.....	13
6.2 Guías y tutoriales .....	14
6.3 Paquetes .....	14
6.4 Documentación oficial .....	15
6.5 Enlaces .....	15

# **1. Proyecto**

## **1.1 Descripción**

Mi proyecto se basa en un la web de Tumblr, una red social en la que compartir imágenes o simplemente notas para los usuarios que te siguen.

En mi aplicación podrás crear nuevos posts, subir imágenes o links de Youtube, comentar, votar comentarios o posts, seguir a los diferentes usuarios, comparar el número de seguidores y posts, editar tu perfil, entrar con Twitter o Facebook, traducciones dependiendo del idioma del navegador (ES, EN) y también cuenta con un sistema de notificaciones dependiendo de la acción (comentario, like y follow).

## **1.2 Motivación**

La motivación principal del proyecto era aprender un nuevo framework con el cual poder crear aplicaciones web de manera rápida y sencilla. En este caso elegí Meteor porque nos permite crear aplicaciones web solo con usando Javascript como lenguaje base, tanto en el cliente como en el servidor.

# **2. Meteor**



## **2.1 ¿Qué es Meteor?**

Meteor es una plataforma para poder crear aplicaciones web en un corto periodo de tiempo. El lenguaje que utiliza es Javascript tanto en el servidor como en el cliente. Se caracteriza por comunicar tanto el back-end como el front-end usando el mismo lenguaje, lo que nos permite entender muchísimo mejor como funciona nuestra aplicación.

Otra de las características de Meteor es que es reactivo, cada vez que se haga un cambio en la base de datos Meteor se encargará de mostrarnos esos cambios en nuestra aplicación en tiempo real, siempre estará en comunicación constante con nuestra base de datos.

## **2.2 El pasado**

Normalmente las webs servían HTML con los datos ya incrustados, siempre se tenía que preguntar al servidor y este devolver una respuesta. La web se tenía que recargar para poder ver los datos que nos enviaba el servidor. Teníamos la aplicación del servidor normalmente con PHP y la de cliente con algún framework de JS (Angular, jQuery, etc). Normalmente intentabas camuflar esto con AJAX, llamadas al servidor a por los datos.

## **2.3 El Futuro**

En las webs modernas lo importante es la información que se muestra al usuario. Si hay un cambio, tanto si es una notificación o han creado un nuevo post, el usuario debe recibir esos datos sin tener que recargar la página. Y no solo a ese usuario, sino todos los usuarios conectados a nuestra aplicación y a los que le afecten estos cambios también tendrán que recibir estos datos.

Todo esto debe ser en tiempo real, ya que lo importante para el usuario es ver todos esos cambios reflejados en su navegador o móvil, sin tener que hacer nada en absoluto.

Aparte nos permite actualizar nuestra aplicación de manera fácil, si hemos cambiado el código de servidor o arreglado algún fallo, simplemente tendremos que subir nuevamente el código (Hot Push Code) y todos los usuarios tendrán la nueva versión del código sin tener que actualizar la aplicación. Si son cambios visuales o que afecten al HTML la web se recargara automáticamente mostrando los nuevos cambios del HTML.

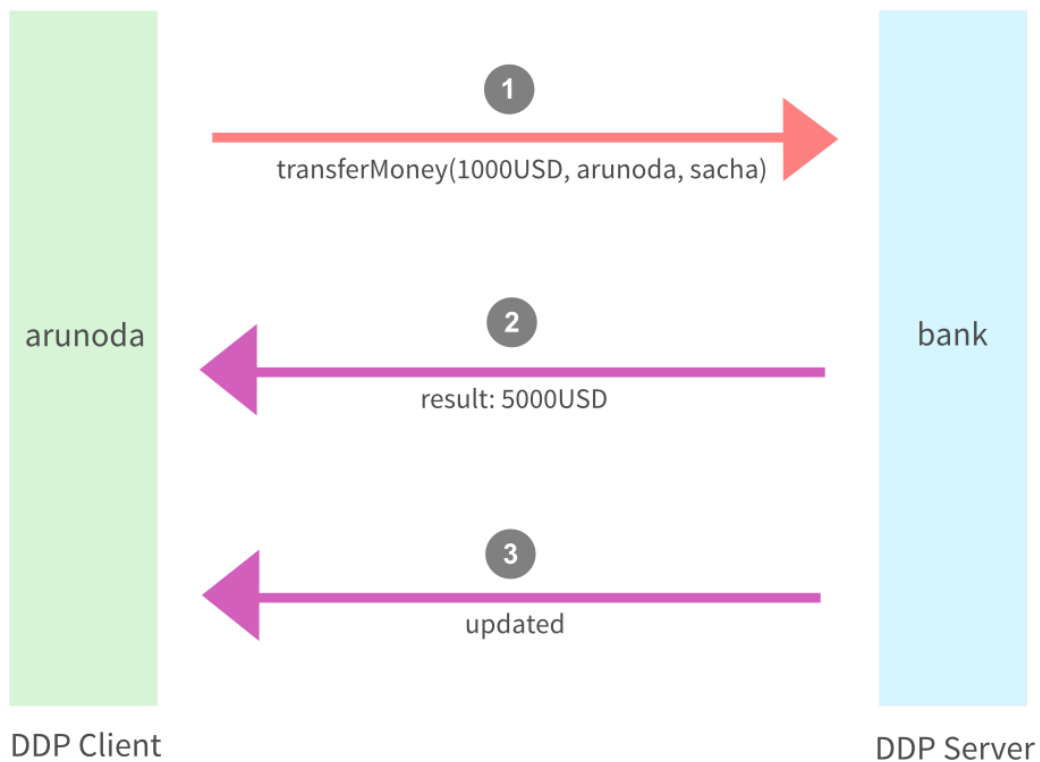
## **2.4 Como funciona**

Meteor se comunica mediante un protocolo DDP (Meteor's Distributed Data Protocol) con los métodos para mantener los cambios de la web en tiempo real.

Los métodos son llamadas a procedimientos remotos (RPC), que se utiliza para guardar los eventos de entrada del usuario y datos que provienen del cliente. Si está familiarizado con las API REST o HTTP, se puede pensar en ellos como las peticiones POST al servidor, pero con muchas características más interesantes y optimizadas para la construcción de una aplicación web moderna.

El protocolo DDP hace principalmente dos cosas:

- Maneja llamadas a procedimiento remoto (RPC).
- Gestiona los datos.



1. El cliente DDP (Arunoda) invoca el método `transferMoney` con tres parámetros: 1000USD, Arunoda y Sacha.
2. Luego, después que la transferencia ha sido aceptada, el servidor DDP (banco) envía un mensaje con el saldo actualizado de la cuenta de Arunoda. Si hubo un error, habrá un error en lugar de un resultado.
3. Después, el servidor envía otro mensaje DDP con el mensaje actualizado del método, notificando la transferencia ha sido enviada a Sacha con éxito y lo ha aceptado.

## 2.5 Mongo

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En lugar de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en

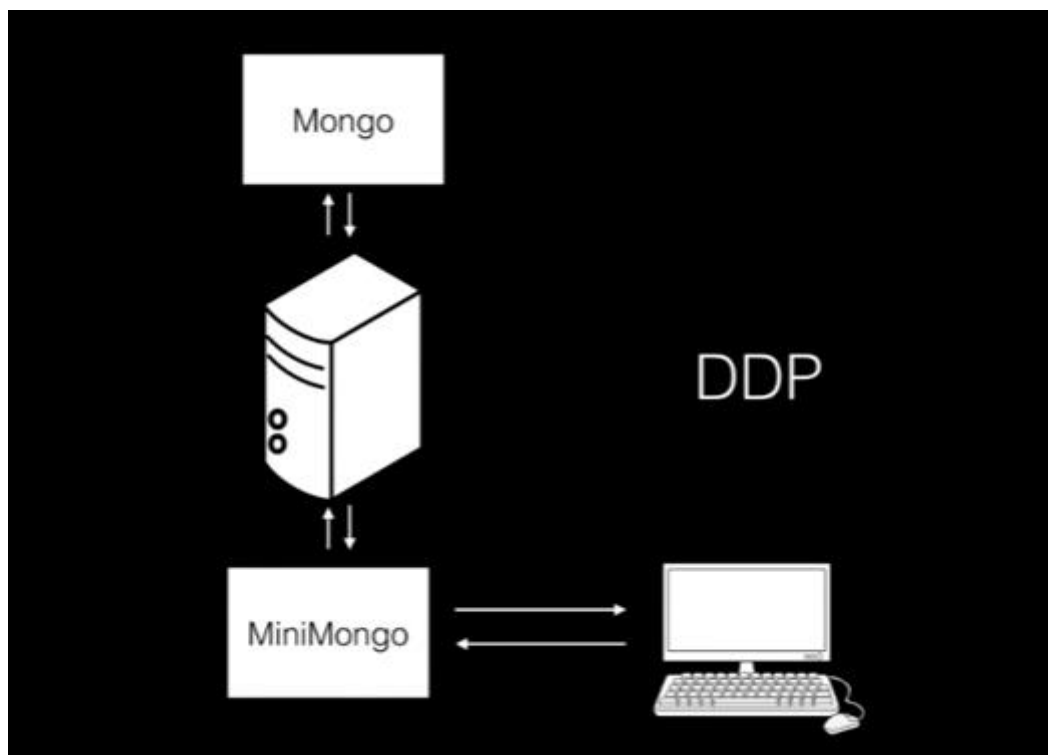
documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Meteor lo gestiona de manera diferente, en la parte del cliente, tiene una copia llamada Mini Mongo en la cual podremos hacer consultas de Mongo pero de manera limitada ya que solo tendremos los datos que nos devuelve el servidor.

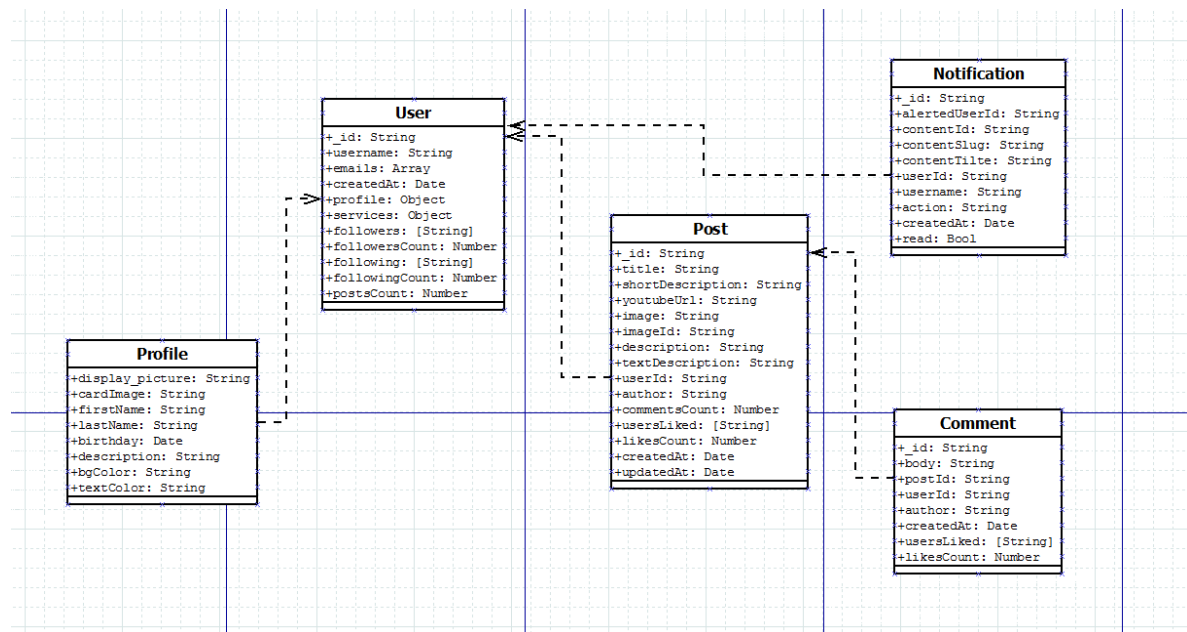
Luego en la parte de servidor tendremos acceso a todas las colecciones de la base de datos, tendremos una sintaxis muy similar a Mongo pero con algunos cambios.

Podemos agregar nuevos documentos, actualizarlos o eliminarlos, pero siempre verificando los datos en los métodos y comparándolos con el esquema, así evitamos que ingresen datos que no estén validados antes.

Para que el usuario reciba los datos al cliente de Minimongo tendremos que publicarlos antes en el servidor y hacer que el usuario se suscriba a esos datos para que puedan manipularlos en cliente, pero siempre con ciertas limitaciones.



La base de datos de mi aplicación esta esquematizada, mediante un paquete llamado SimpleSchema. Así los datos ingresados a la base de datos deben cumplir esos requerimientos o no se añadirán. Mi base datos tendría el siguiente esquema:



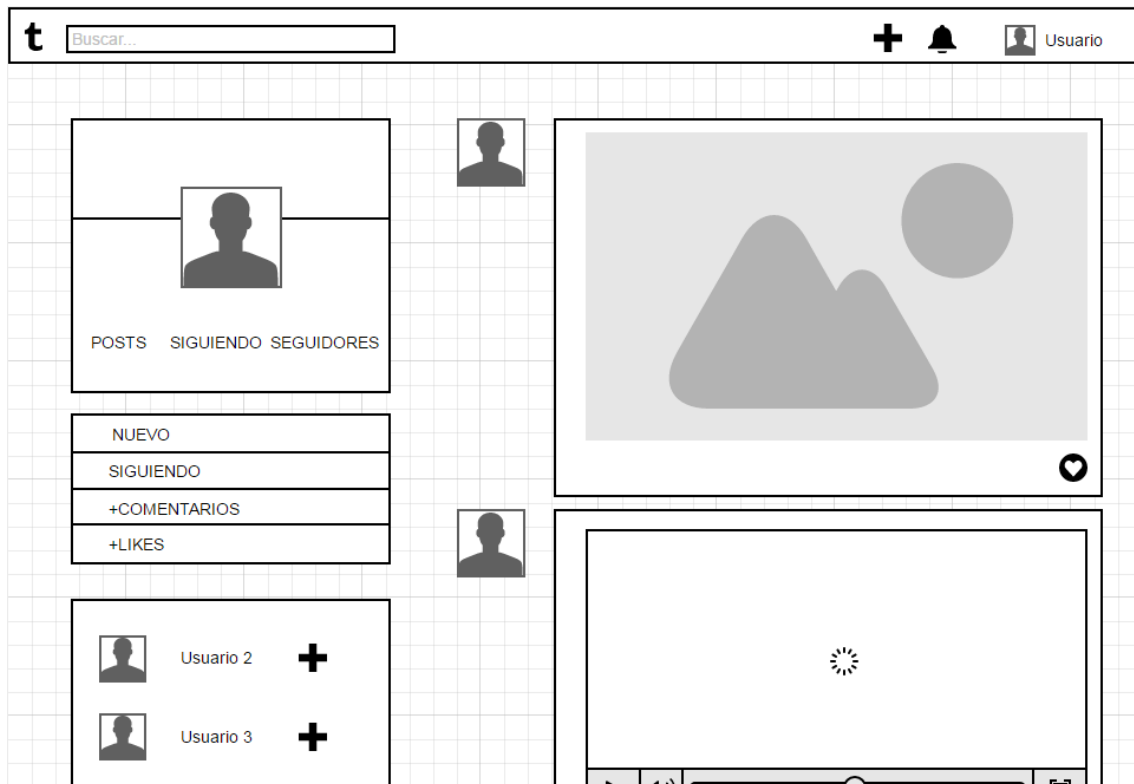
## 2.6 Componentes

- **LiveQuery**: Consultas en a la base de datos en tiempo real.
- **DDP**: Protocolo para comunicar cliente/servidor.
- **Minimongo**: Consultas de mongo en cliente.
- **Blaze**: Sistema de platillas de Meteor.
- **Tracker**: Modifica la web cuando hay nuevos cambios.

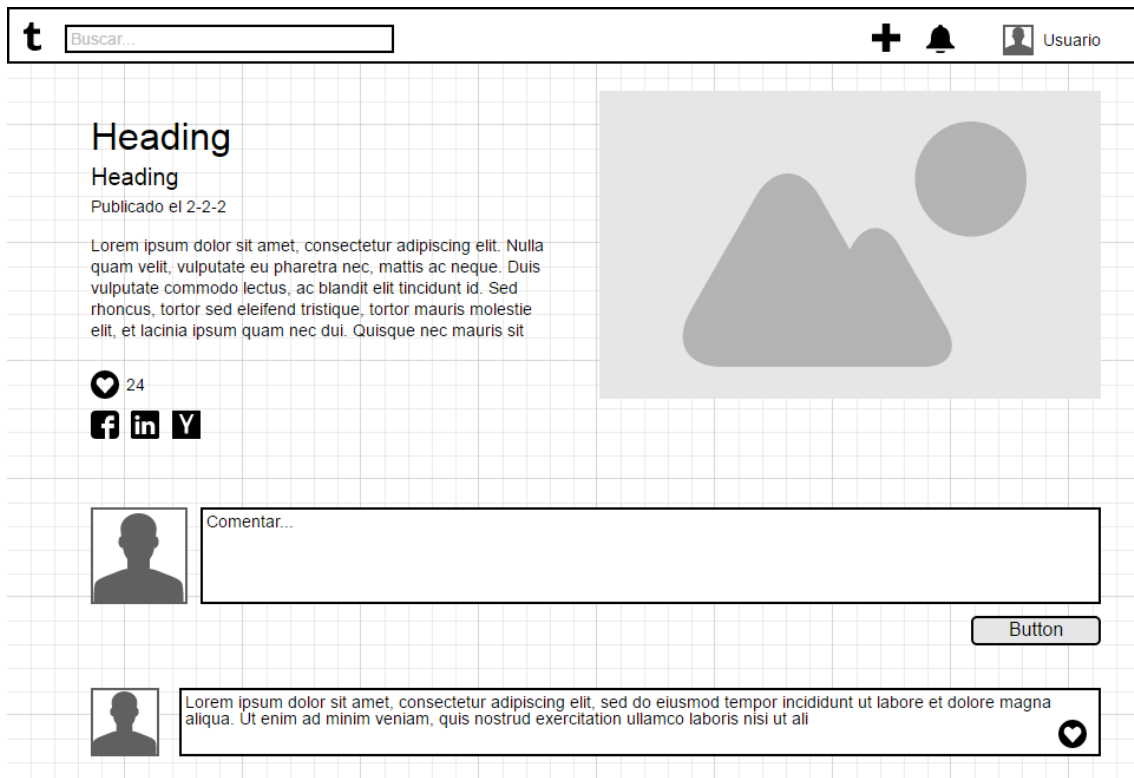


### 3. Mockups

#### Inicio / Homepage



#### Página del Post / Post Page



## Perfil de usuario / User Profile

**t**

+

Usuario

Posts

123

Siguiendo

32

Seguidores

23

Heading

Heading

45

Usuario1 

+

Usuario2 

+

Usuario3 

+

## Editar Perfil / Edit Profile

**t**

+

Usuario

Posts

123

Siguiendo

32

Seguidores

23

Button

Usuario1 

+

Usuario2 

+

Usuario3 

+

Seguidores – Siguiendo / Followers - Following

t

Buscar...

+

Usuario

Posts Siguiendo Seguidores

123 32 23

Usuario1 +

Usuario2 +

Usuario3 +

Búsqueda / Search

t

Buscar...

Usuario1

Usuario2

Usuario3

Usuario4

Usuario5

Usuario5

Usuario6

Post 1 Titulo

Post 2 Titulo


Post 3 Titulo




Post 4 Titulo

Post 5 Titulo


Post 6 Titulo


Iniciar Sesión / Sign In





Usuario

 Entrar con Twitter

 Entrar con Facebook


PROYECTO




Usuario

Contraseña


Entrar


Registro / Sign Up





Usuario

 Entrar con Twitter

 Entrar con Facebook

PROYECTO

Usuario

Email

Contraseña

Contraseña

Entrar

## 4. Código

El código está subido a Github, podréis seguir todo los commits realizados durante todo el transcurso del desarrollo del proyecto y como ha ido evolucionando la aplicación.

También podréis ver los paquetes recomendados que me han solucionado muchos problemas y ahorrado trabajo.

<https://github.com/cristopher29/ProyectoMeteor>

## 5. Conclusión

Con las cosas que me quedo de este proyecto es que he aprendido muchas cosas, no solo de Meteor, sino cómo funcionan otras aplicaciones basadas en NodeJS. Como se comunica el cliente y el servidor, como manejan las rutas, el rendimiento una vez subida la aplicación a Heroku.

Otra parte fundamental es los libros que he leído sobre Meteor para poder aprender a utilizarlo, ya que cuando empecé no sabía absolutamente nada, también aprendí de viendo código de otras personas para ver cómo trabajan y comparar su estructura de proyecto con la mía.

Así que si queréis empezar con Meteor os recomiendo los siguientes enlaces:

## 6. Bibliografía

### 6.1 Libros

**Libros gratuitos** (Aunque ponga que son de pago, tenéis todo la información en la web)

Para aprender lo básico de Meteor os recomiendo este libro:

**Your First Meteor Application**

<http://meteortips.com/>

Una vez tengamos los conocimientos básicos podemos pasar al segundo:

#### **Your Second Meteor Application**

<http://meteortips.com/second-meteor-tutorial/>

Luego pasaremos a un libro un poco más “avanzado”, aunque nada complicado, quizás algunas partes estén desactualizadas pero va muy bien para aprender. Y además en castellano!

#### **Discover Meteor**

<http://es.discovermeteor.com/>

#### **Libros de pago**

Si queréis aprender cómo funciona Meteor por dentro os recomiendo este libro.

<https://www.manning.com/books/meteor-in-action>

### **6.2 Guías y tutoriales**

Guía oficial de Meteor (TODO List)

<https://www.meteor.com/tutorials/blaze/creating-an-app>

Videos de Meteor (LevelUpTuts)

<https://www.youtube.com/channel/UCyU5wkjgQYGRB0hIHMwm2Sg>

### **6.3 Paquetes**

Paquetes que no pueden faltar en tu aplicación

- accounts-password o accounts-ui (Para empezar)
- iron:router
- aldeed:collection2
- aldeed:autoform
- msavin:mongol
- reywood:publish-composite
- momentjs:momento

- check
- reactive-var
- service-configuration
- meteorhacks:subs-manager
- anti:i18n (opcional)
- manuelschoebel:ms-seo (opcional)
- meteorhacks:fast-render (opcional)
- meteorhacks:kadira (opcional)

## 6.4 Documentación oficial

No podía faltar la documentación oficial

<https://docs.meteor.com/>

## 6.5 Enlaces

### Webs de ayuda

<http://justmeteor.com/blog/>

<https://themetorchef.com/>

<https://github.com/oortcloud/unofficial-meteor-faq>

### Enlaces de ayuda

<http://blog.benmcmahen.com/post/41741539120/building-a-customized-accounts-ui-for-meteor>

<https://medium.com/all-about-meteorjs/extending-meteor-users-300a6cb8e17f#.shp8ypgq2>

<http://www.stefanhayden.com/blog/2015/05/25/user-profile-edit-with-autoform-and-simpleschema-in-meteor-js/>

<http://www.nodebeginner.org/index-es.html>

<http://yauh.de/2015/05/06/the-illustrated-guide-to-mobile-apps-with-meteor/>

<http://www.webtempest.com/meteorjs-fromscratch-1>