

Proyecto Final - DAW



MEMORIA DE PROYECTO

TWITTER CLONE

Christian Ramos Ortiz

Índice

- **Página 3** Descripción del proyecto
- **Página 3** Tecnologías usadas
- **Página 4** Backend y Frontend
- **Página 5** Definición de la base de datos
- **Página 6** API (PHP / MYSQL)
- **Página 7** API (Clase Twitter)
- **Página 8** API (API_KEY)
- **Página 9** Sistema de login con API_KEY
- **Página 10** Consultas y relaciones en SQL
- **Página 11** (PUSHER) Feed en tiempo real
- **Página 12** Pretty URL (.htaccess)
- **Página 13** Depuración de errores y testeo
- **Página 14** Interfaz / Estilo de diseño
- **Página 15** Responsive
- **Página 16** Sitemap
- **Página 17** Despliegue (OVH)
- **Página 18** VPS (CentOS, Nginx, Php, Mysql, VestaCP)
- **Página 19** Git (Github)
- **Página 20** Justificación empresarial del proyecto
- **Página 21** Organigrama

Descripción del proyecto

El proyecto ha sido hacer un clon de la interfaz, y las funcionalidades de twitter.

Incluye: Sistema de tweets, perfiles, follows/followings, retweets, favs, comentarios, y notificaciones. El feed se actualiza en tiempo real cuando uno de tus seguidores publica un tweet, esto hace que no tengas que recargar la página manualmente.

Tecnologías usadas

- HTML
- CSS / SASS
- JAVASCRIPT
- PHP
- SESSIONS
- COOKIES
- JSON
- AJAX
- MYSQL
- NGINX
- PUSHER (Real Time API)
- VESTACP
- HOSTING (VPS, DOMINIO, SUBDOMINIOS, DNS)
- GITHUB
- POSTMAN
- VSCODE
- XAMPP
- MYSQL WORKBENCH
- MARKDOWN
- SSH / FTP
- SSL / HTTPS
- API
- BACKEND & FRONTEND

BACKEND

Con lo primero que empecé fue por la parte del backend. Por que al final es la parte más funcional de la aplicación.

El backend se basa en una API hecha con PHP y Mysql. A la que le pasas una serie de parámetros por HTTP y devuelve los datos en formato JSON.

Muchas de las URL de la API para poder ser usadas hay que pasarles como uno de los parámetros una API_KEY, que más adelante explicaré más a fondo.

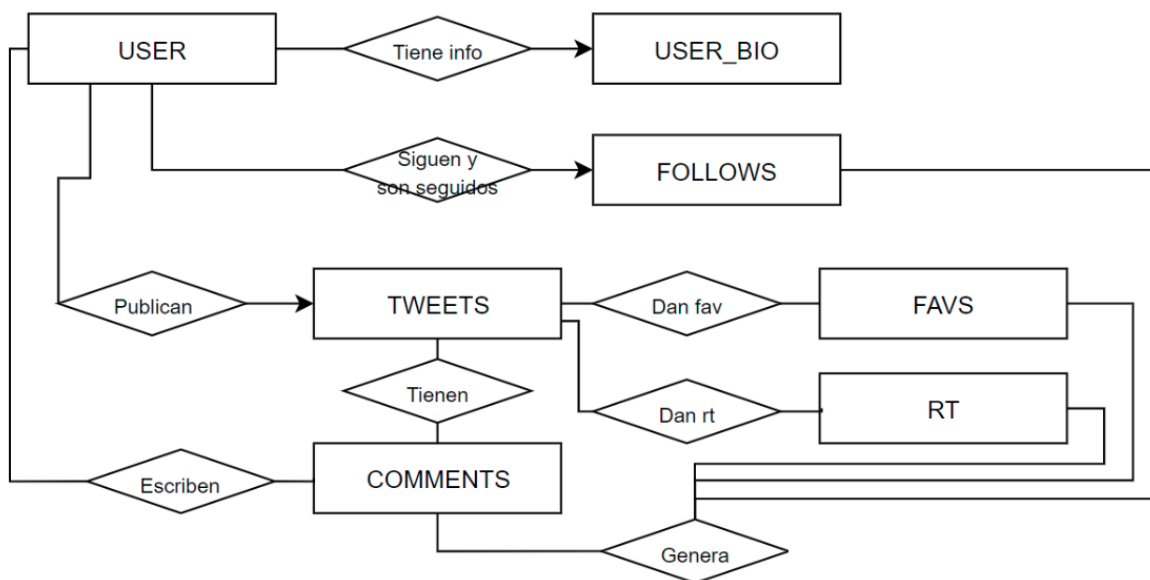
FRONTEND

El frontend está formado solamente por archivos HTML, CSS y Javascript. Todos los datos que aparecen en la interfaz (nombres de usuarios, tweets, favs, rt...) se optionen por peticiones AJAX a la API.

Se ha intentado que la interfaz sea lo más parecida a la real de Twitter. El CSS y HTML se ha hecho desde cero sin uso de ningún framework, esto hace que la carpeta de los archivos CSS sea de menor tamaño y cargue mas rapido, al solo tener los ID y CLASS que la interfaz necesita.

Definición de la base de datos

El diseño del modelo E/R de la base de datos lo hice con una aplicación online que se llama draw.io. Para después pasarlo a SQL y crearlo desde Mysql Workbench.



USER(id, nick, name_twitter, pass_hash, api_key, reg_date)

USER_BIO(user_id, bio, avatar_url, banner_url)

FOLLOWS(following_user_id, followed_user_id)

TWEETS(user_id, tweet_id, date, content, img_url)

FAVS(tweet_id, user_fav_id, date)

RT(tweet_id, user_rt_id, date)

COMMENTS(tweet_id, user_id, content, date)

API (PHP / MYSQL)

- deleteTweet.php
- exploreFeed.php
- fav_remove.php
- fav.php
- feed.php
- follow.php
- followers.php
- followings.php
- login.php
- logout.php
- notificationFeed.php
- postTweet.php
- profile.php
- profileFeed.php
- randomUsers.php
- register.php
- rt_remove.php
- rt.php
- search.php
- settings.php
- settingsLoad.php
- tweet.php
- tweetComments.php
- tweetPage.php
- unfollow.php

La API está formada por varios archivos .php, a los por que por el metodo GET o POST se les pasan una serie de parametros, dependiendo la acción que haga ese archivo.

Se usa una clase llamada "Twitter" que contiene todos los métodos de la API.

Ejemplo del archivo "login.php":

```
login.php •
backend > login.php
1  <?php
2  require_once './Model/twitter.php';
3
4  if(isset($_REQUEST["nick"]) && isset($_REQUEST["pass"])){
5      $instagram = new Twitter();
6      $instagram->login($_REQUEST["nick"],$_REQUEST["pass"]);
7  }
8  ?>
```

API Clase Twitter

La clase "Twitter" es la clase PHP que contiene todos los métodos de la API. Métodos para devolver el perfil de un usuario, sus seguidores, publicar tweets, hacer comentarios, etc...

Ha esta clase el usuario final no accede directamente, si no que accede por la URL de los archivos antes mencionados.

```
twitter.php x
backend > Model > twitter.php

40
41 class Twitter {
42
43     function __construct() {
44     }
45
46
47     // Función para descargar una imagen al servidor desde una URL
48     public function downloadURL($url, $type) {
49         $file_name = basename($url);
50         if(file_put_contents( "img/".$_COOKIE["id"]."_" . $type . ".jpg",file_get_contents($url))) {
51             echo "Imagen subida correctamente";
52         }
53         else {
54             echo "Error en la subida de la imagen";
55         }
56     }
57
58
59     // Función para chequear si existe ya un usuario con ese nick
60     public function checkUser($nick) {
61         $conexion = DB::connectDB();
62
63         $sql = "SELECT nick FROM user WHERE nick='".$nick."'";
64         $result = $conexion->query($sql);
65         $conexion->close();
66
67         if ($result->num_rows>0) {
68             return true;
69         } else {
70             return false;
71         }
72     }
73
74
75     // Función para chequear que la api_key sea válida
76     public function checkApiKey($api_key) {
77         $conexion = DB::connectDB();
78         $sql = "SELECT api_key FROM user WHERE api_key='".$api_key."' and id=".$_COOKIE["id"];
79         $result = $conexion->query($sql);
80         $conexion->close();
81     }
82 }
```

API API_KEY

Una API_KEY es la solución he usado para poder dividir a los usuarios registrados de los no registrados.

La API_KEY se genera como un string aleatorio y único para cada usuario cada vez que hace un login correcto. Se guarda en el servidor en la DB en la tabla de "users", y al usuario en forma de cookie.

Una de sus funciones también es la de poder identificar que petición hace cada usuario. Como cada API_KEY es única se puede saber cual es el usuario que está haciendo la petición.

Ejemplo del archivo "feed.php" que devuelve en formato JSON los datos del feed del usuario que ha hecho la petición. Sin necesidad de tener que enviar como parametro su nick o id.

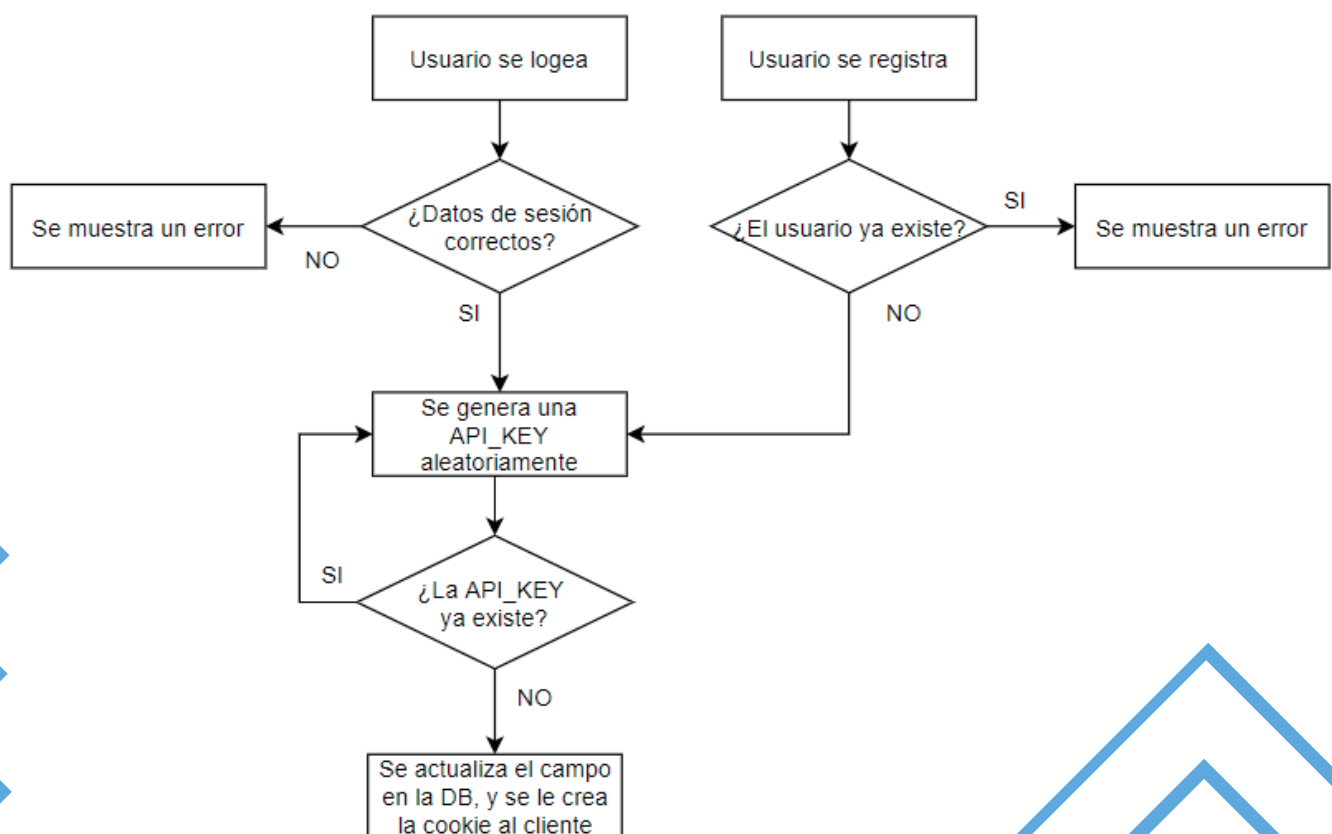
```
feed.php
backend > feed.php
1  <?php
2  require_once './Model/twitter.php';
3
4  if(isset($_REQUEST["api_key"]) && isset($_REQUEST["page"])){
5      $instagram = new Twitter();
6      $instagram->followsFeed($_REQUEST["api_key"], intval($_REQUEST["page"]));
7  }
8  ?>
```


Sistema de login con API_KEY

Al hacer el login correctamente, se actualiza la API_KEY del usuario por una nueva generada aleatoriamente, esto para mayor seguridad, las API_KEY no son fijas para que no se puedan sniffear las cookies y tener acceso indefinidamente a tu perfil.

Para más seguridad las contraseñas no se guardan en texto plano, si no que se guarda solamente el HASH de esta, y al hacer el login se comprueba que los HASH coincidan.

Si un usuario se registra automáticamente hace el login y se le genera la API_KEY.

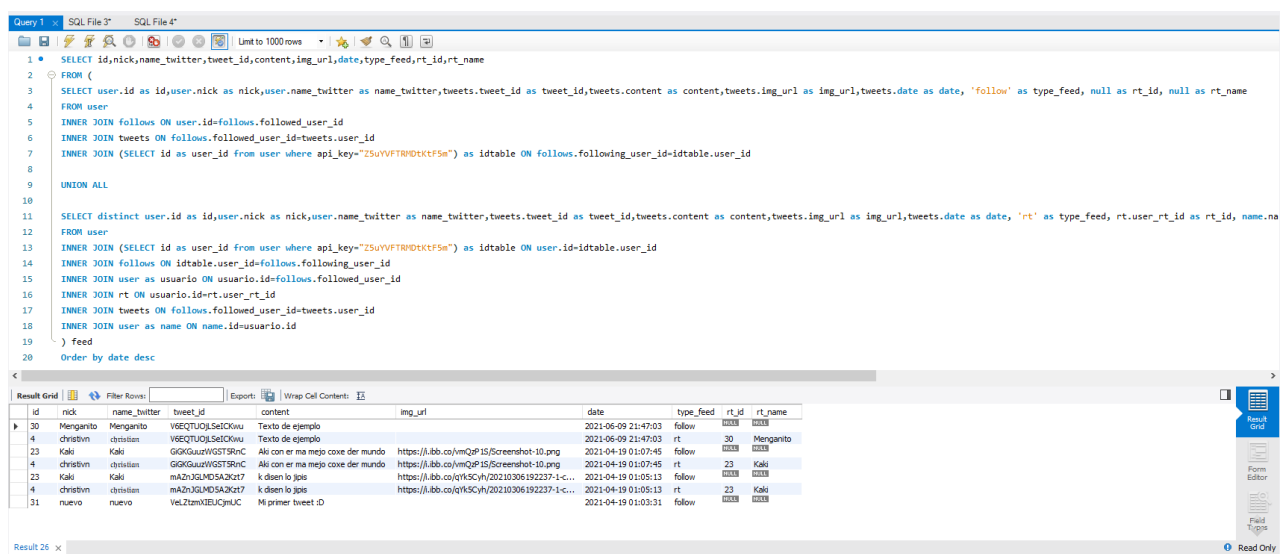


Consultas y relaciones en SQL

Las consultas SQL se han hecho antes probandolas desde Mysql Workbench, para después usarlas con PHP en la API.

Mayormente las consultas son sencillas, SELECT, INSERT, UPDATE, DELETE de una sola tabla. Pero hay otras como las del Feed o Notificaciones que son relaciones de hasta 4 tablas.

Ejemplo de la consulta para obtener los datos del Feed de un usuario:



The screenshot displays the MySQL Workbench interface. The top pane shows a SQL query (Query 1) designed to fetch a user's feed. The query uses multiple JOINs to combine data from the 'user', 'follows', 'tweets', and 'retweets' tables. It includes a subquery to filter tweets by a specific API key. The bottom pane shows the 'Result Grid' with 26 rows of data. The columns include user ID, nickname, name, tweet ID, content, image URL, date, feed type, and related IDs. The data is sorted by date in descending order.

```
1 SELECT id,nick,name_twitter,tweet_id,content,img_url,date,type_feed,rt_id,rt_name
2 FROM (
3 SELECT user.id as id,user.nick as nick,user.name_twitter as name_twitter,tweets.tweet_id as tweet_id,tweets.content as content,tweets.img_url as img_url,tweets.date as date, 'follow' as type_feed, null as rt_id, null as rt_name
4 FROM user
5 INNER JOIN follows ON user.id=follows.followed_user_id
6 INNER JOIN tweets ON follows.followed_user_id=tweets.user_id
7 INNER JOIN (SELECT id as user_id from user where api_key="ZSuVVFRNDktF5m") as idtable ON follows.following_user_id=idtable.user_id
8
9 UNION ALL
10 SELECT distinct user.id as id,user.nick as nick,user.name_twitter as name_twitter,tweets.tweet_id as tweet_id,tweets.content as content,tweets.img_url as img_url,tweets.date as date, 'rt' as type_feed, rt.user_rt_id as rt_id, name.na
11 FROM user
12 INNER JOIN (SELECT id as user_id from user where api_key="ZSuVVFRNDktF5m") as idtable ON user.id=idtable.user_id
13 INNER JOIN follows ON idtable.user_id=follows.following_user_id
14 INNER JOIN user as usuario ON usuario.id=follows.followed_user_id
15 INNER JOIN rt ON usuario.id=rt.user_rt_id
16 INNER JOIN tweets ON follows.followed_user_id=tweets.user_id
17 INNER JOIN user as name ON name.id=usuario.id
18 ) feed
19 Order by date desc
20
```

id	nick	name_twitter	tweet_id	content	img_url	date	type_feed	rt_id	rt_name
30	Menganto	Menganto	V6EQTU0JLSetCKouu	Texto de ejemplo		2021-06-09 21:47:03	follow		
4	christyn	christian	V6EQTU0JLSetCKouu	Texto de ejemplo		2021-06-09 21:47:03	rt	30	Menganto
23	Kali	Kali	GIGXGuazWGS7SRnC	Ali con er ma mejo cone der mundo	https://i.bbb.co/vmQp15/Screenshot-10.png	2021-04-19 01:07:45	follow		
4	christyn	christian	GIGXGuazWGS7SRnC	Ali con er ma mejo cone der mundo	https://i.bbb.co/vmQp15/Screenshot-10.png	2021-04-19 01:07:45	rt	23	Kali
23	Kali	Kali	mA2nJQLMDSA2Kzt7	k disen lo jpis	https://i.bbb.co/qYKSCyh/20210306192237-1-c...	2021-04-19 01:05:13	follow		
4	christyn	christian	mA2nJQLMDSA2Kzt7	k disen lo jpis	https://i.bbb.co/qYKSCyh/20210306192237-1-c...	2021-04-19 01:05:13	rt	23	Kali
31	nuevo	nuevo	VelZtmXIEUCjnuC	M primer tweet :D		2021-04-19 01:03:31	follow		

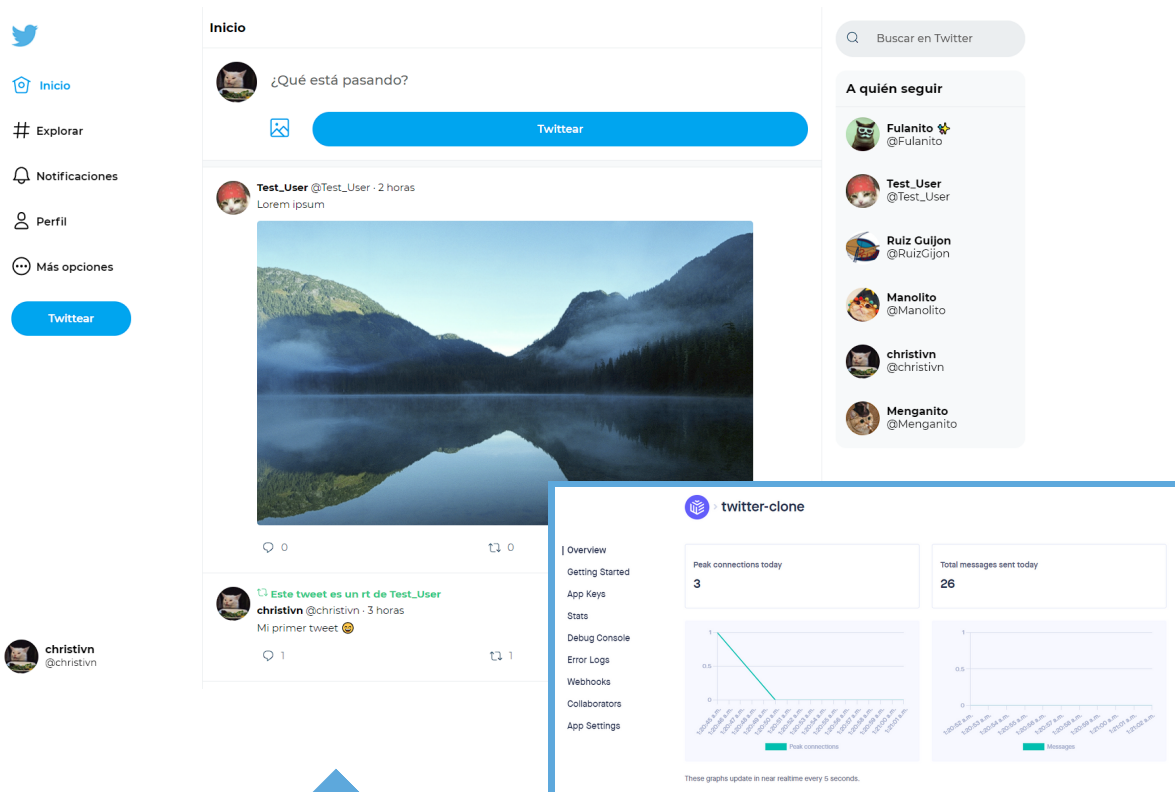


Feed en tiempo real

Para poder hacer que el feed se refresque de manera automática, cuando uno de los usuarios a los que sigues a publicado un tweet, he usado una aplicación llamada Pusher.

Está aplicación mediante un SDK que ellos ofrecen para varios lenguajes entre ellos PHP puedes hacer aplicaciones en tiempo real de una manera más sencilla y hace que el servidor no se sature.

Para su instalación también hubo que instalar Composer.



Pretty URL (.htaccess)

El frontend al estar formado por archivos .html, no quedaba bien estéticamente, y acababan siendo URL más largas con esa extensión. Por lo que decidí quitarlas haciendo uso del .htaccess

← → ↻ ⚠ No es seguro | christivn.es/demo/twitter-clone/tweet?tweet_id=uS0A7OMJLBFTVKC

```
.htaccess
1 RewriteEngine on
2 RewriteCond %{REQUEST_FILENAME} !-d
3 RewriteCond %{REQUEST_FILENAME}\.html -f
4 RewriteRule ^(.*)$ $1.html
5
```

Depuración de errores y testeo

La depuración del backend se hizo usando POSTMAN, para enviar los parámetros necesarios a la API y comprobar los datos del JSON que devolvía.

El frontend a solo ser HTML y JS, lo que más se depuró fue el AJAX y comprobar que fueran bien las peticiones y se imprimieran los datos en el DOM. Para ello se usó la herramienta de google chrome de DevTools. Esta herramienta permite hacer snippets para hacer código de prueba antes de incorporarlo al código del frontend de la web.

Backend



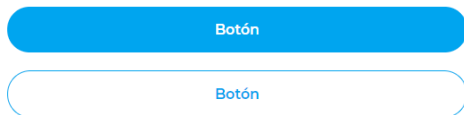
Frontend



Interfaz / Estilo de diseño

Fuente: Montserrat

Botones



Paleta de colores

Azul: #1da1f2

Blanco: #FFFFFF

Titulos

Lorem Ipsum

Lorem Ipsum

Lorem Ipsum

Formularios

(URL) Foto de perfil
/api/img/36_avatar.jpg

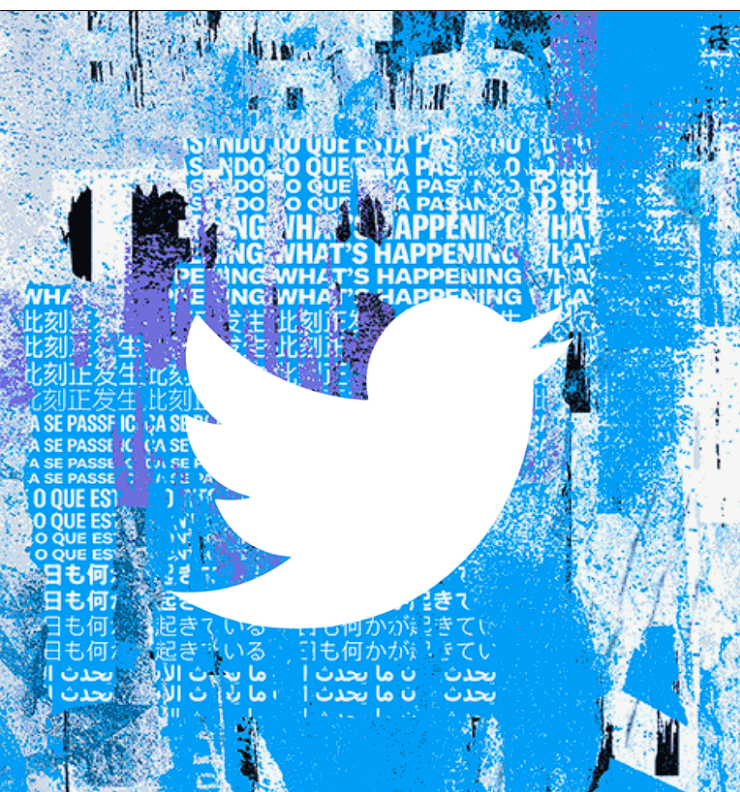
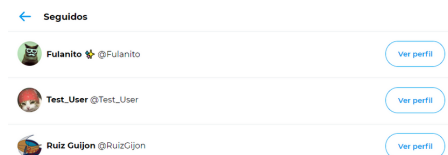
(URL) Foto del banner
img/banner.jpg

Nombre
Test_User

Nick
@Test_User

Biografía

Listas



Lo que está pasando ahora

Únete a Twitter hoy mismo.

Regístrate

Iniciar sesión

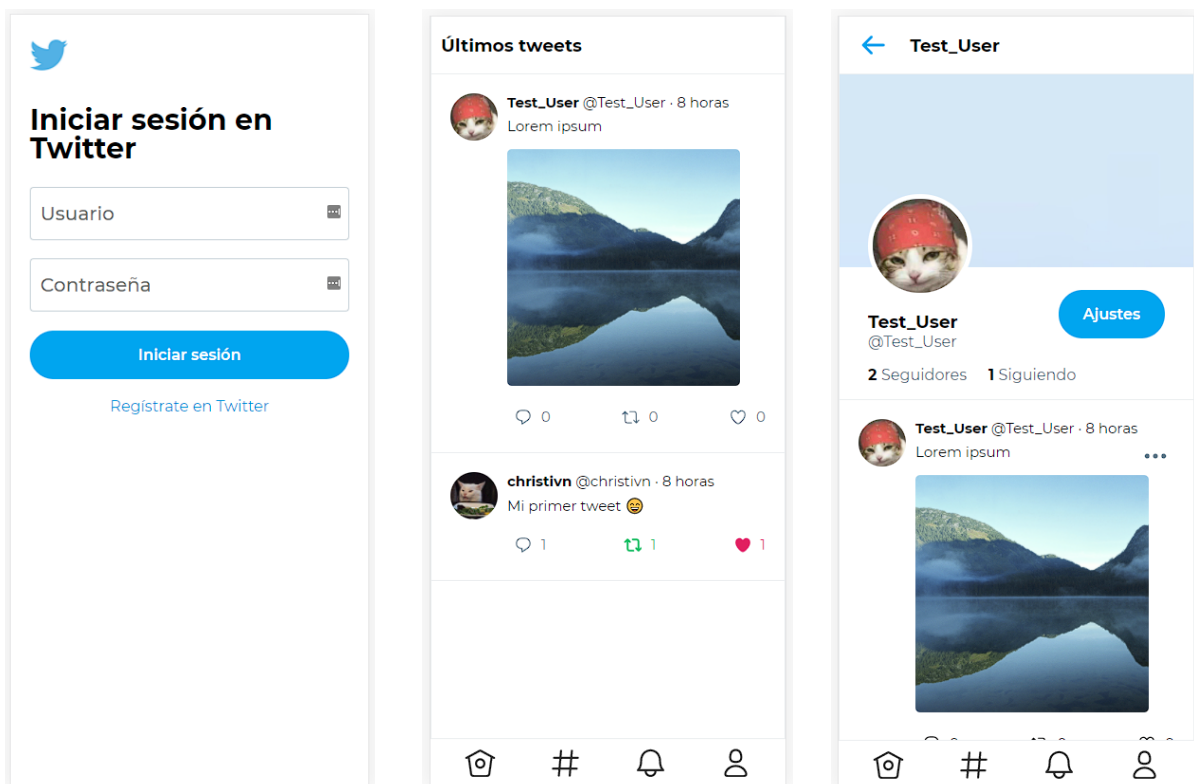
Responsive

Para el responsive de la aplicación se ha usado Flexbox, y media queries.

El menú de navegación cambia dependiendo del dispositivo que el usuario esté usando en ese momento. Si está desde ordenador le aparecerá un menú vertical a la izquierda, y si entra desde tablet o móvil, le aparecerá un menú horizontal en la parte inferior de la pantalla.

Todo el contenido se adapta perfectamente a las dimensiones de la pantalla donde se esté usando.

Ejemplos en un dispositivo móvil:



Sitemap

Usuarios logueados

- Página principal
 - Index
 - Login

Usuarios no logueados

- Feed principal
 - Feed
 - Explorer
 - Notifications
 - Profile
 - Followers
 - Followings
 - Tweet
 - Settings

Despliegue

El hosting que he elegido para el despliegue del proyecto es OVH, es uno de los proveedores de servidores más grandes y confiable de Europa.

Para el despliegue he usado una VPS con CentOS, al que se le han instalado los servicios necesarios para que funcione como servidor web (Nginx, PHP, Mysql).

Al servicio web se podía acceder por IP (**https://37.187.198.52**), pero tenía un dominio (**http://christivn.es/**) donde pienso montar un portfolio de los proyectos que haga y tener hay las demos. Así que el proyecto está hosteado actualmente en:
http://christivn.es/demo/twitter-clone



Dominios / christivn.es / Zona DNS

christivn.es

Expira el 10 jun. 2022 Renovar

Acciones

Información general Zona DNS Servidores DNS Redirección DynHost Glue Tareas recientes Correo electrónico y listas de correo

Aquí puede ver la configuración de los distintos registros de su dominio.

También puede configurar estos registros para asociar su dominio a los distintos servicios haciendo clic en «Añadir un registro».

Todos Buscar un dominio...

Domino	TTL	Tipo	Destino
<input type="checkbox"/> christivn.es	0	NS	dns15.ovh.net.
<input type="checkbox"/> christivn.es	0	NS	ns15.ovh.net.
<input type="checkbox"/> christivn.es	0	MX	10 mx3.mail.ovh.net.
<input type="checkbox"/> christivn.es	0	MX	1 mx4.mail.ovh.net.
<input type="checkbox"/> christivn.es	0	A	37.187.198.52
<input type="checkbox"/> api.christivn.es	0	A	37.187.198.52
<input type="checkbox"/> www.christivn.es	0	A	37.187.198.52
<input type="checkbox"/> christivn.es	0	TXT	"1 www.christivn.es"
<input type="checkbox"/> autoconfig.christivn.es	0	CNAME	mailconfig.ovh.net.
<input type="checkbox"/> autodiscover.christivn.es	0	CNAME	mailconfig.ovh.net.
<input type="checkbox"/> _autodiscover._tcp.christivn.es	0	SRV	0 0 443 mailconfig.ovh.net.
<input type="checkbox"/> ftp.christivn.es	0	CNAME	christivn.es.
<input type="checkbox"/> _imap._tcp.christivn.es	0	SRV	0 0 993 ssl0.ovh.net.

Añadir un registro

Restaurar mi zona DNS

Modificar el TTL por defecto

Editar en modo de texto

Eliminar la zona DNS

Guías

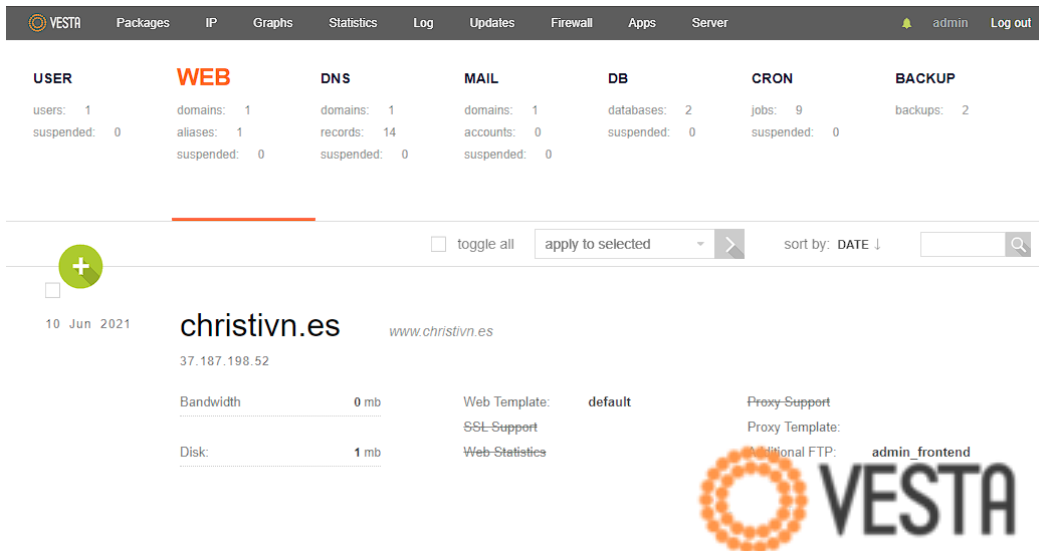
Zona DNS

VPS CentOS

El hardware de la VPS es de 1vCore, 2GB de ram y 20GB SSD. De sistema operativo usa CentOS al que se le ha instalado para desplegar la aplicación: Nginx, Php, Mysql, Git y VestaCP.

Git para hacer el despliegue desde el repositorio donde está subido el proyecto (<https://github.com/christivn/twitter-clone>) con un git clone.

VestaCP es un panel de control parecido a Cpanel pero gratuito, esto para facilitar la configuración de las DNS, y el dominio del sitio.



The screenshot displays the VestaCP web interface. At the top is a navigation bar with tabs: VESTA, Packages, IP, Graphs, Statistics, Log, Updates, Firewall, Apps, and Server. On the right of the bar are a bell icon, the text 'admin', and a 'Log out' link. Below the navigation bar is a dashboard with seven summary cards: USER (1 users, 0 suspended), WEB (1 domains, 1 aliases, 0 suspended), DNS (1 domains, 14 records, 0 suspended), MAIL (1 domains, 0 accounts, 0 suspended), DB (2 databases, 0 suspended), CRON (9 jobs, 0 suspended), and BACKUP (2 backups). Below these cards is a table of domains. The first domain listed is 'christivn.es', added on '10 Jun 2021', with IP '37.187.198.52' and website 'www.christivn.es'. To the right of the domain name are links for 'Bandwidth' (0 mb), 'Disk' (1 mb), 'Web Template: default', 'SSL-Support', 'Web-Statistics', 'Proxy-Support', 'Proxy Template:', and 'Additional FTP: admin_frontend'. The Vesta logo, consisting of an orange circular pattern and the word 'VESTA', is positioned at the bottom right of the interface.

USER	WEB	DNS	MAIL	DB	CRON	BACKUP
users: 1	domains: 1	domains: 1	domains: 1	databases: 2	jobs: 9	backups: 2
suspended: 0	aliases: 1	records: 14	accounts: 0	suspended: 0	suspended: 0	
	suspended: 0	suspended: 0	suspended: 0			

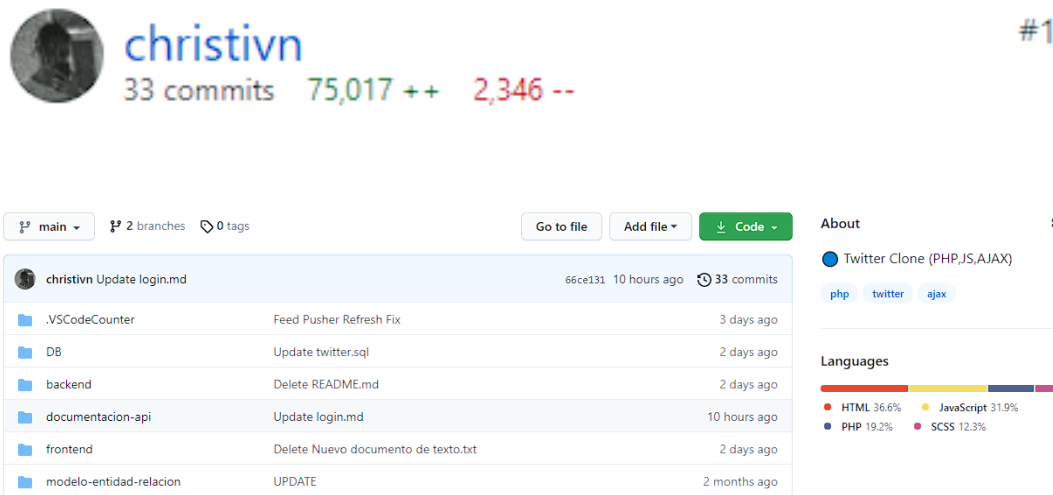
Domain	Date	IP	Website	Bandwidth	Disk	Web Template	SSL-Support	Web-Statistics	Proxy-Support	Proxy Template	Additional FTP
christivn.es	10 Jun 2021	37.187.198.52	www.christivn.es	0 mb	1 mb	default					admin_frontend

Git (Github)

Todo el desarrollo del proyecto y las versiones de este se ha gestionado con Github.

El repositorio es público el link es:

<https://github.com/christivn/twitter-clone>



The screenshot shows the GitHub repository page for 'christivn/twitter-clone'. The repository is public and has 33 commits, 75,017 additions, and 2,346 deletions. The repository is located at <https://github.com/christivn/twitter-clone>. The repository is a clone of the Twitter project, built with PHP, JS, and AJAX. The repository is a clone of the Twitter project, built with PHP, JS, and AJAX. The repository is a clone of the Twitter project, built with PHP, JS, and AJAX.

christivn
33 commits 75,017 ++ 2,346 --

#1

main 2 branches 0 tags

Go to file Add file Code

File	Commit	Time	Commits
christivn Update login.md	66ce131	10 hours ago	33 commits
.VSCodeCounter	Feed Pusher Refresh Fix	3 days ago	
DB	Update twitter.sql	2 days ago	
backend	Delete README.md	2 days ago	
documentacion-api	Update login.md	10 hours ago	
frontend	Delete Nuevo documento de texto.txt	2 days ago	
modelo-entidad-relacion	UPDATE	2 months ago	

About

Twitter Clone (PHP,JS,AJAX)

php twitter ajax

Languages

HTML 36.6% JavaScript 31.9% PHP 19.2% SCSS 12.3%

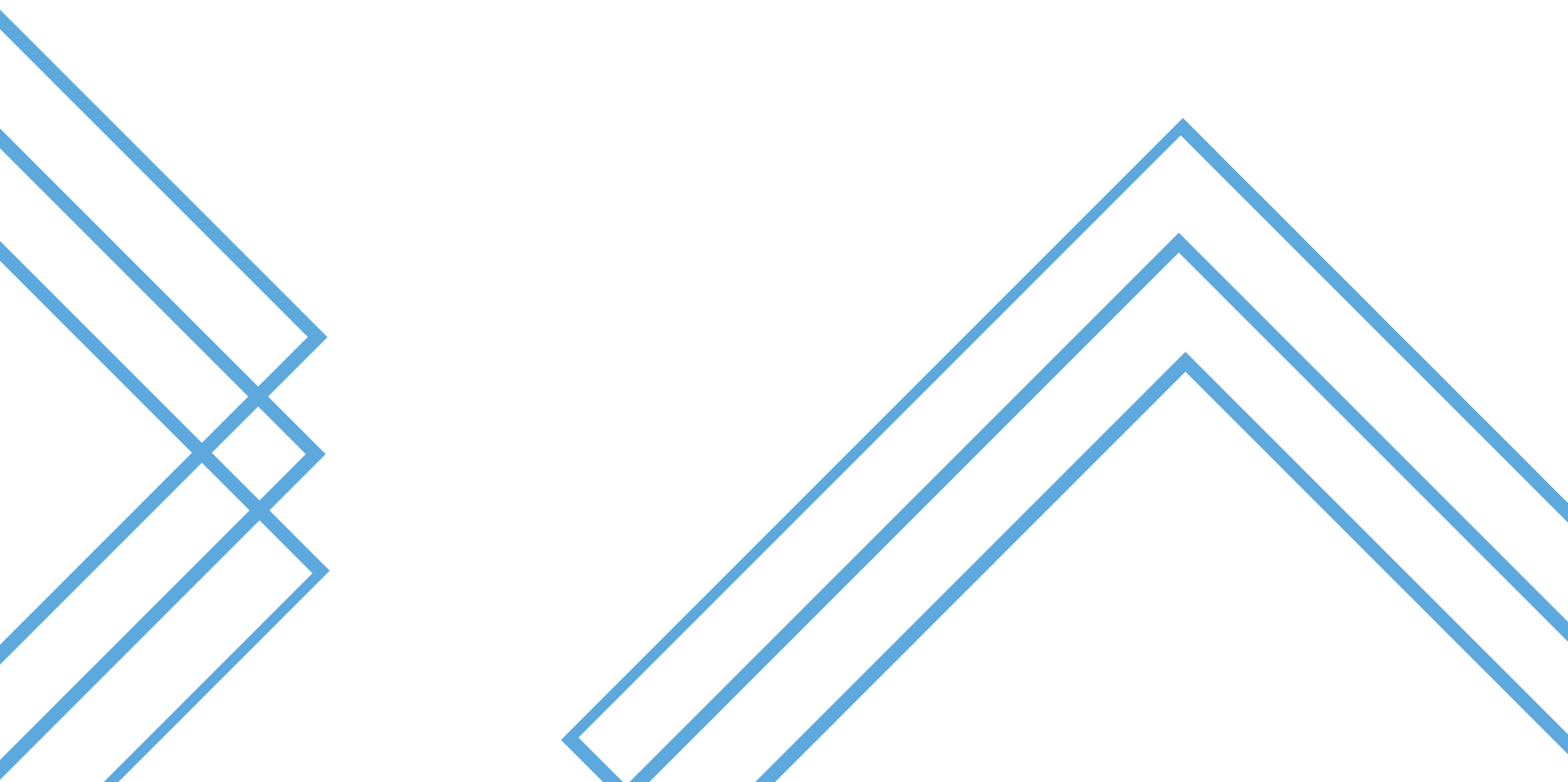
Justificación empresarial del proyecto

La principal idea del proyecto es formar parte de mi portfolio profesional donde ir publicando los proyectos que vaya haciendo.

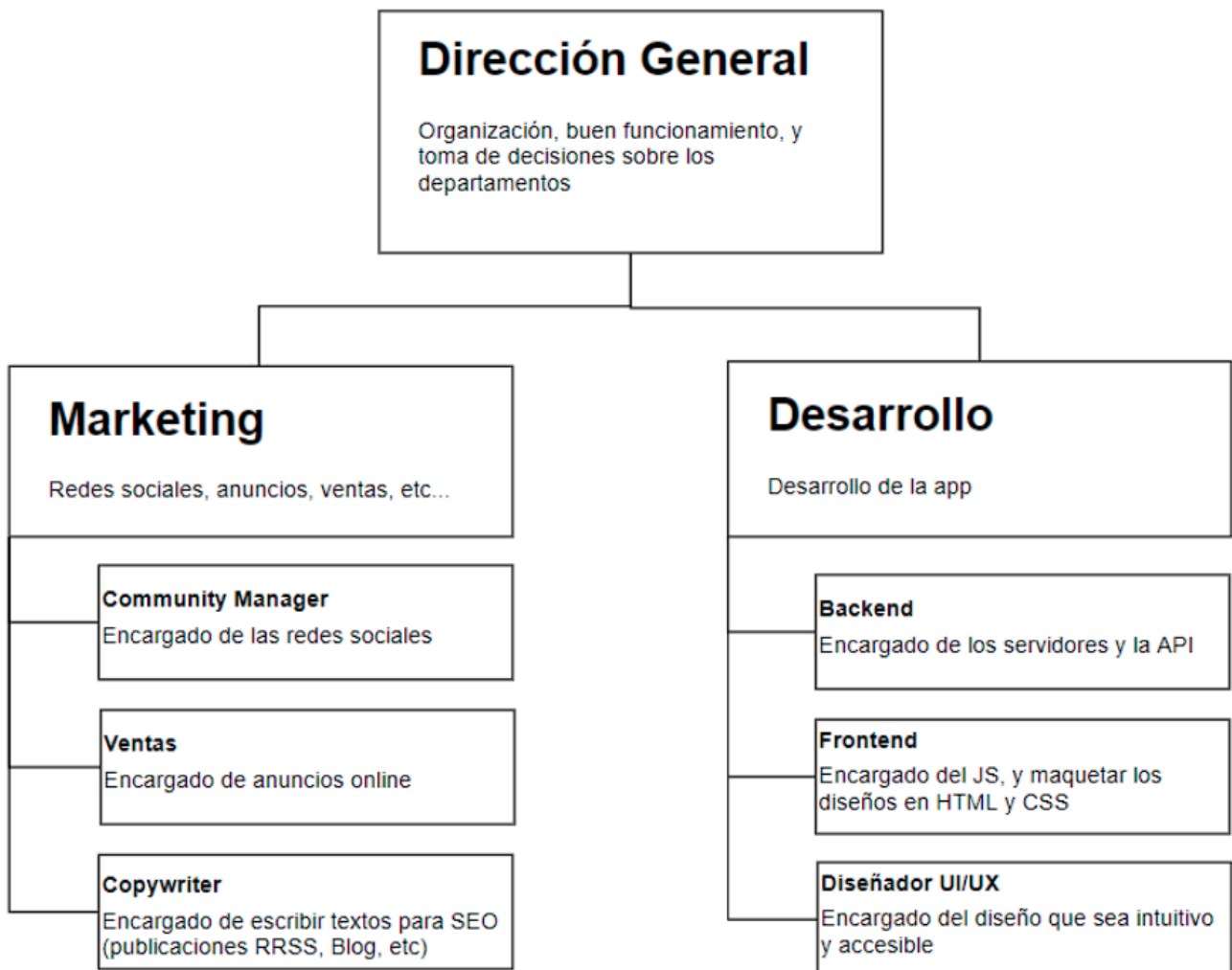
Eso no impide poderse usar para aplicaciones del mundo real. Cambiando los logotipos de Twitter y los colores corporativos, se podría usar como chat de trabajo para equipos, como Slack. O para comunidades online haciendo la función de red social/foro.

Público al que va dirigido

Al estar publicado en Github y ser completamente Open Source cualquier persona puede descargar, colaborar, o modificar el código del proyecto.



Organigrama





FIN

Christian Ramos Ortiz

