>monad-posting
Manual de técnico

Iván Molina Rebolledo

Base de Datos
Primavera 2021
Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla

30 de abril de 2021

Índice

1. Sitio

https://monadposting.ivmoreau.com

2. Planteamiento del sistema

Red social de microblogging. Un sistema minimalista que permita la interacción entre personas de la forma más simple posible. Adherido al principio KISS.

Pretendo diseñar algo acorde a las necesidades modernas de la web, pero que se mantenga fácil y simple.

Esta idea nace como una contrarrespuesta a la web moderna que sobreabusa de herramientas y frameworks que sólo dañan la experiencia del usuario. Establezca de ejemplo a la red social Facebook, cuya interfaz *overengineered* está llena de problemas (en especial la versión web móvil del sitio).

Internet aún puede ser simple.

3. Requisitos funcionales y no funcionales

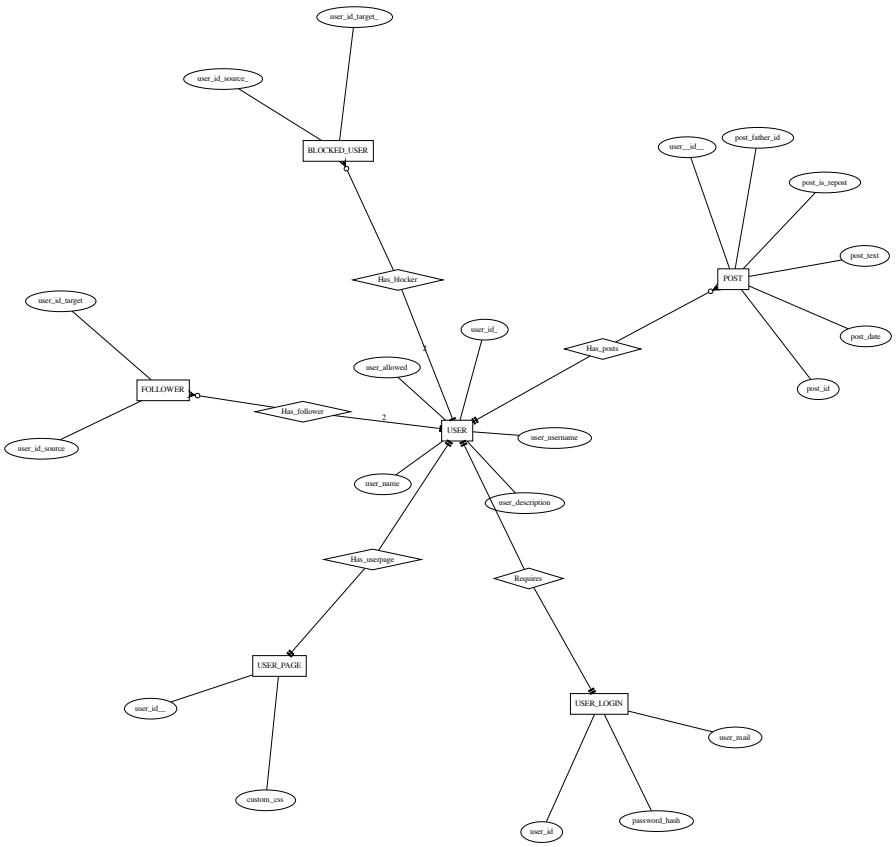
3.1. Funcionales

1. Perfil de usuario
2. El usuario debe poder personalizar nombre, descripción, contraseña.
3. Funcionalidades sociales: repost, publicar, seguidores.
4. Pagina principal que muestre lo más reciente.
5. Personalización avanzada de pagina de perfil con CSS de usuario.
6. Máximo de 400 caracteres por post.
7. LaTeX.

3.2. No funcionales

1. Simplicidad.
2. Responsive design UX.
3. Rapido y ligero.
4. KISS.
5. Altamente personalizable.
6. Cuentas seguras.

4. Diseño conceptual



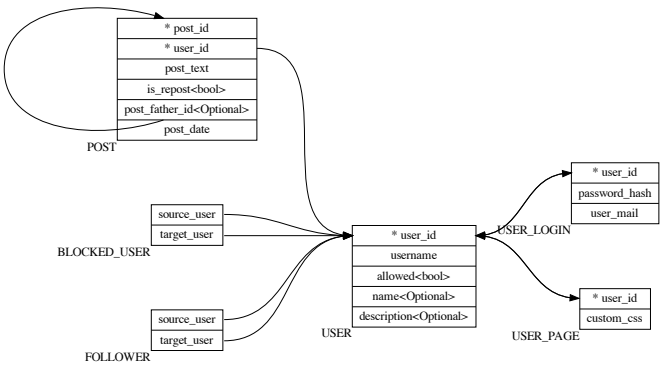
5. Diccionario de datos

Cuadro 1: Diccionario de datos del proyecto

Tabla	Identificador	¿Es index?	¿Opcional?	Tipo	¿Es relacionado a otra tabla?	Ejemplo	Notas
General	user_id	Sí	No	Numerico, i32	Sí, todas.	123423	
Post	post_text	No	Sólo si es repost	String	No	"They are working hard to make up [...]"	
Post	post_is_repost	No	No	boolean	No	true	
Post	post_father_id	No	Sí	Numerico, i32	La misma tabla.	123423	
Post	post_date	No	No	Date, i32	No.	2019-04-22...	CURTIME()
User	user_username	No	No	String	No	RealDonaldTrump	
User	user_name	No	Sí	String	No	Donald Trump	En caso de no existir, se toma el nombre de usuario
User	user_description	No	Sí	String	No	45th President of the United States [...]	
User	user_allowed	No	No	boolean	No	false	Indica si el usuario está "baneado".
UserLogin	password_hash	No	No	String	No	a9cff14a06c17da821e3c7e25d5880fic	Guardar las contraseñas "de forma segura".
UserLogin	user_mail	No	No	String	No	something@something.thing	
UserPage	custom_css	No	Sí	String/Blob de texto	No	#home background-color: #ff3f4;	
BlockedUser, Followers	user_source_user, user_target_user	No	No	Numerico, i32	Al usuario.	23352	

I

6. Diseño lógico de la Base de Datos



[1]user_allowed Indica si tiene permitido acceder a la plataforma. El uso de este valor es raro, así que su empleo (el banear) requiere acceso directo a la base de datos. Por diseño; ya que la plataforma no valora niveles de acceso.

7. Normalización

Forma Normal 1. Está en esta forma.

1. Primary key (no duplicate tuples): Sí.
2. No repeating groups: Sí.
3. Atomic columns (cells have single value): Sí.

Forma Normal 2. Está en esta forma.

1. 1NF: Sí.
2. Single Column Primary Key: Sí.

Forma Normal 3. No está en esta forma.

1. 2NF: Sí.
2. Has no transitive functional dependencies: Sí.

El modelo cumple con BCNF.

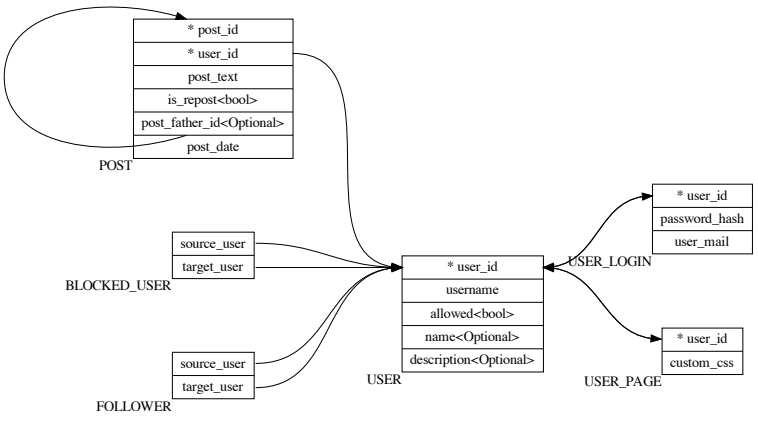
Forma Normal 4. Está en esta forma.

1. 3NF: Sí.
2. Every non-trivial multivalued dependency begins with a superkey: Sí.

Forma Normal 5. Está en esta forma.

1. 4NF: Sí.
2. Cannot be decomposed into any number of smaller tables without loss of data.: Sí.

Modelo final:



Nota: está normalizado.

8. Implementación

Un usuario tiene distribuida su información en las tres tablas propuestas con la finalidad de tener un sistema de consulta un poco más eficiente. Datos como los de acceso sólo requieren ser consultados durante el login del usuario, o el contenido css que sólo se requiere al visitar el perfil. Aparte permite tener una mejor estructura para el funcionamiento del usuario y para las posibles mejoras que puedan llegar a ser impelemntadas (pensar a futuro).

Pasa algo similar con la tabla de bloqueados y seguidores.

Esta base de datos fue implementado con MariaDB (fork compatible con MySQL) usando el engine INNODB. El base de datos de conecta con el sistema a través de *unix sockets*; en un intento de no habilitar TCP para la conexión entre el ordenador donde se desarrolló y el servidor, fue usado SSH y no hubo ningún problema de compatibilidad en usar el socket de MariaDB Ubuntu en MacOS. El sistema en producción se ejecuta directamente sobre el servidor.

Para el framework fue utilizado Flask con SQLAlchemy, que permitió hacer la implementación lógica del sitio usando el lenguaje de programación Python. En un principio se pensaba utilizar Actix (Rust) por las ventajas que suponía en cuanto a velocidad y async; esto no fue posible debido a la incompatibilidad entre ciertas librerías fundamentales para el proyecto.

El código del programa está dividido en tres módulos (aparte del archivo principal) específicos. Todo el proceso de autentificación se realiza en el modulo auth.py, el cual re-

suelve las rutas de registro y login. El módulo db.py tiene la misión de ser el modulo que gestiona la conexión con la base de datos; solamente se encarga de generar la instancia para acceder.

El modulo interface.py es el modulo más relevante para la funcionalidad. En el está programado todo lo referente a la interacción del usuario. Aquí es importante recalcar que el index está dividido en dos rutas para poder ir mostrando post de manera secuencial de treinta en treinta. Esto es necesario suponiendo una cantidad de usuarios considerable (a no ser que queramos ver miles de posts en la pagina principal; cosa que probablemente no es posible).

SQLAlchemy permite ejecutar comandos en la base de datos usando una manera más idiomatica con respecto a Python. Sin embargo, también permite hacer las queries de manera directa como si de cualquier comando sql se tratase. En este proyecto se optó por la segunda opción específicamente por efectos de demostración de lo aprendido en el curso; ya que en el uso practico podría resultar mejor y más legible la API que el ORM nos ofrece.

En cuanto al diseño, se ha decidido el actual por su minimalismo. Quiero aclara que esta no es una excusa y que realmente es parte del planeamiento de simplicidad. El tema no es específico de este proyecto, sino que es el tema que deriva de mi sitio web actual (ivmoreau.com) que fue personalmente construido con LaTeX.css para simplicidad. Añadido a eso, el usuario tiene la opción de personalizar su propio perfil tanto como la CSS specification le permita; esto puede resultar en muy buenos usos por los más expertos.

Se intentó prescindir de Javascript lo más posible (pero se usa), porque vivimos en una época en la que quieren forzar Javascript en todo lo que sea Turing-Completo. No obstante, nos es muy útil al momento de ser utilizado junto a KaTeX, que nos permite un bello rendering de matemáticas con una sintaxis LaTeX-like. Esta es una de las funciones que más diferencian a la red social de otras (salvo por Stack Exchange, el cual usa Mathjax).

El rendering es hecho con las plantillas de Jinja2, de las cuales fue adaptadas mínimas partes desde el sitio superior (plantillas Handlebars). Aquí hay un poco de manejo de lógica (a lo PHP) que se encarga de determinar lo que le muestra al usuario. También hay llamadas a la base de datos para cosas especificas de la información de los usuarios.

En cuanto al equipo de desarrollo, el programa requirió de lo siguiente:

1. MacOS Big Sur (para development).

2. Ubuntu 20.04 (para producción; aunque la base de datos siempre se usó desde el servidor Ubuntu)

3. MariaDB 15.1 (sólo con unix sockets activados)
4. SSH (para hacer tunnelling del socket desde Ubuntu a OSX)
5. Python 3.9.4 (para development)
6. Python 3.8.5 (producción)
7. Visual Studio Code con el linter de Microsoft para Python.
8. Nginx como proxy (para producción).

En cuanto al licenciamiento, basta con lo que dicta la Licencia MIT para todos los archivos excluyendo los archivos del tema (que pienso hacer open source en un futuro; por ahora es obra privada). Cabe destacar que las claves privadas fueron retiradas del proyecto empaquetado por motivos de seguridad.

El nombre de la red social es una referencia al símbolo $>$ usado para citar, la palabra mónada (simplemente por referencia a la estructura matemática) y la palabra posting por la finalidad del programa.

9. Extras

El código fuente viene adjunto en el .zip. Este incluye la base de datos modelada en un .sql file.

Se requiere tener SQLAlchemy, Flask y mysqlclient instalados en un virtualenv.

Para el deployment basta con configurar nginx de reverse proxy para que haga routing hacia el servidor waitress-serve.

La ejecución en producción se hace con lo siguiente:

```
nohup waitress-serve --port=8005 --call
  'monad_posting:create_app' > log.txt 2>&1 &
```