

Proyecto Académico: Kodely

Diseño del sistema

Universidad del Valle de Guatemala (UVG)

Campus Altiplano

Curso: Programación Web 1

Ciclo: semestre 7

Fecha: 13 Junio 2025

Encargado del área:

Allan Daniel Ralón Gamboa – 221352

Descripción breve:

Proyecto desarrollado para la asignatura Programación Web 1, cuyo objetivo es crear una plataforma web llamada Kodely, que permita la publicación y visualización de tutoriales técnicos, con autenticación mediante Google y funcionalidades de interacción social

## Estructura general y oficial del orden y distribución del sistema de Kodely:

### Diseño y Funcionamiento del Sistema – Kodely

#### Descripción General

**Kodely** es una plataforma web inspirada en el modelo de sitios como *Dev.to*, centrada en la **publicación, lectura e interacción con tutoriales técnicos**. Su objetivo principal es ofrecer una experiencia simple y directa para que desarrolladores puedan compartir conocimientos sin complicaciones visuales ni sobrecarga de funcionalidades. Está construida utilizando **Django** como framework backend en una **arquitectura monolítica**, con el apoyo de **Firestore Authentication** para la gestión de usuarios mediante inicio de sesión con Google.

#### Estructura del Sistema

El sistema está organizado en **tres grandes módulos funcionales**:

1. **Módulo de Autenticación y Usuarios accounts/**
2. **Módulo de Contenido y Funcionalidad Principal core/**
3. **Módulo de Configuración y Gestión Global kodely/**

A continuación, se detalla cada uno:

#### 1. Módulo de Autenticación y Usuarios accounts/

Este módulo encapsula toda la lógica relacionada con la gestión de cuentas de usuario. Se integró **Firestore Authentication**, lo que permite el **inicio de sesión mediante Google** sin necesidad de almacenar contraseñas o datos sensibles directamente en la plataforma.

#### Funciones destacadas:

- Inicio y cierre de sesión usando Google.
- Restricción de funcionalidades de escritura de contenido a usuarios autenticados.
- Modelo de usuario asociado a la creación de publicaciones y reacciones.

#### Archivos clave:

- `firebase_config.py`: gestiona las credenciales de conexión con Firestore.

- `models.py`: define estructuras de datos como publicaciones (Post) y reacciones (Reaction).
- `views.py`: controla la lógica de acceso y comportamiento relacionado al login.

## 2. Módulo de Contenido y Funcionalidad Principal core/

Aquí se concentran todas las funcionalidades visibles para el usuario y la lógica de la aplicación web.

### Funciones principales:

- Visualización de tutoriales en orden cronológico descendente.
- Creación y edición de publicaciones mediante un **editor WYSIWYG limitado**
- Reacciones con emojis a las publicaciones tipo "me gusta", "asombro", etc..
- Perfil de usuario con sus publicaciones listadas.

### Estructura destacada:

- **templates/**: HTML estructurado para todas las vistas públicas y privadas.
- **static/**: Archivos CSS, JS e imágenes. Destaca el uso de scripts para la autenticación `login.js`, `firebase-init.js`.
- **templatetags/**: Utilidades para personalizar el comportamiento de las plantillas Django.
- **views.py**: define las vistas de creación, edición y visualización de posts.

## 3. Módulo de Configuración Global (kodely/)

Este componente define el comportamiento general del proyecto Django.

### Funciones clave:

- Gestión del enrutamiento (`urls.py`).
- Configuración de apps, base de datos y archivos estáticos (`settings.py`).
- Archivos `asgi.py` y `wsgi.py` para compatibilidad con servidores de producción.

### Flujo de Usuario

1. **Inicio de Sesión**: el usuario ingresa mediante su cuenta de Google usando Firebase.

2. **Acceso al Panel:** al iniciar sesión, puede crear publicaciones, editar su perfil o visualizar su contenido.
3. **Explorar Tutoriales:** todos los usuarios pueden navegar publicaciones desde la pantalla principal.
4. **Interacciones:** pueden reaccionar con emojis y leer el contenido completo al hacer clic en un título.
5. **Cerrar Sesión:** opción clara para cerrar sesión desde el menú superior.

Kodely/

```
└─ accounts/
|   └─ migrations/
|   |   └─ 0001_initial.py
|   |   └─ 0002_post.py
|   |   └─ 0003_reaction.py
|   └─ __init__.py
|   └─ admin.py
|   └─ apps.py
|   └─ firebase_config.py
|   └─ models.py
|   └─ tests.py
|   └─ urls.py
|   └─ views.py
|
└─ core/
|   └─ migrations/
|   |   └─ __init__.py
|   └─ static/
```

- | | └─ css/
- | | | └─ create\_post.css
- | | | └─ dashboard.css
- | | | └─ editar\_post.css
- | | | └─ login.css
- | | | └─ perfil.css
- | | └─ img/
- | | | └─ LogoKodely.svg
- | | | └─ logo\_kodely.svg
- | | └─ js/
- | | | └─ auth.js
- | | | └─ firebase-init.js
- | | | └─ login.js
- | | | └─ logout.js
- | | | └─ main.js
- | └─ templates/
- | | └─ create\_post.html
- | | └─ editar\_post.html
- | | └─ home.html
- | | └─ login.html
- | | └─ perfil.html
- | | └─ ver\_post.html
- | └─ templatetags/
- | | └─ \_\_init\_\_.py
- | | └─ utils.py

- |   └─ \_\_init\_\_.py
- |   └─ admin.py
- |   └─ apps.py
- |   └─ models.py
- |   └─ tests.py
- |   └─ urls.py
- |   └─ views.py
- |
- └─ kodely/
  - |   └─ \_\_init\_\_.py
  - |   └─ asgi.py
  - |   └─ settings.py
  - |   └─ urls.py
  - |   └─ wsgi.py
  - |
  - └─ db.sqlite3
  - └─ firebase\_config.py
  - └─ manage.py
  - └─ requirements.txt
  - └─ .gitignore
  - └─ LICENSE

**Tabla de Rutas del Proyecto Kodely**

Ruta (URL)	Vista asociada	Descripción funcional	Ubicación del código
/	home_view	Página principal con lista de tutoriales publicados (más recientes primero).	core/views.py
/login/	login_view	Vista de login con Google mediante Firebase.	accounts/views.py
/logout/	logout_view	Cierra la sesión del usuario autenticado.	accounts/views.py
/crear-post/	create_post_view	Formulario para crear nuevos tutoriales (requiere autenticación).	core/views.py
/editar-post/<id>/	edit_post_view	Formulario para editar un tutorial publicado por el usuario.	core/views.py
/ver-post/<id>/	view_post_detail	Muestra el contenido completo del tutorial seleccionado.	core/views.py
/perfil/	user_profile_view	Muestra el perfil del usuario con sus tutoriales publicados.	core/views.py
/reaccionar/<id>/	add_reaction_view	Agrega una reacción con emoji a un tutorial específico (vía POST).	core/views.py
/admin/	Django Admin	Panel de administración de Django para superusuarios.	Interno de Django

**Comentarios adicionales:**

- Las rutas dinámicas como /ver-post/<id>/ y /editar-post/<id>/ usan un parámetro <id> que hace referencia al identificador del tutorial en la base de datos.
- El login con Firebase se gestiona desde el **frontend con JavaScript** static/js/login.js, firebase-init.js, mientras que el backend solo sirve como soporte.
- Las funciones están divididas principalmente entre los módulos accounts (autenticación) y core (contenido y vistas principales)

**Tabla de Vistas del Proyecto Kodely**

Nombre de la vista	Descripción funcional	Requiere login	Plantilla usada	Ubicación del código
<b>home_view</b>	Muestra la página principal con todos los tutoriales ordenados por fecha.	No	home.html	core/views.py
<b>login_view</b>	Renderiza la vista de login (Google/Firebase).	No	login.html	accounts/views.py
<b>logout_view</b>	Cierra la sesión del usuario.	Sí	X	accounts/views.py
<b>create_post_view</b>	Formulario para crear un nuevo tutorial.	Sí	create_post.html	core/views.py
<b>edit_post_view</b>	Formulario para editar un tutorial propio.	Sí	editar_post.html	core/views.py
<b>view_post_detail</b>	Muestra el contenido completo de un tutorial específico.	No	ver_post.html	core/views.py
<b>user_profile_view</b>	Muestra el perfil del usuario con su lista de tutoriales.	Sí	perfil.html	core/views.py
<b>add_reaction_view</b>	Agrega una reacción tipo emoji a un tutorial (llamada con POST por JavaScript).	Sí	X	core/views.py



### Vistas adicionales del sistema

Plantilla	Propósito principal	Estilos/CSS vinculados	JavaScript relacionado
<b>home.html</b>	Muestra lista de tutoriales disponibles.	dashboard.css	main.js
<b>login.html</b>	Interfaz de inicio de sesión con Google/Firebase.	login.css	login.js, firebase-init.js
<b>create_post.html</b>	Editor WYSIWYG para nuevos tutoriales.	create_post.css	X
<b>editar_post.html</b>	Permite modificar tutoriales creados.	editar_post.css	X
<b>ver_post.html</b>	Vista detallada del contenido de un tutorial.	ver_post.css	X
<b>perfil.html</b>	Muestra los tutoriales del usuario autenticado.	perfil.css	X

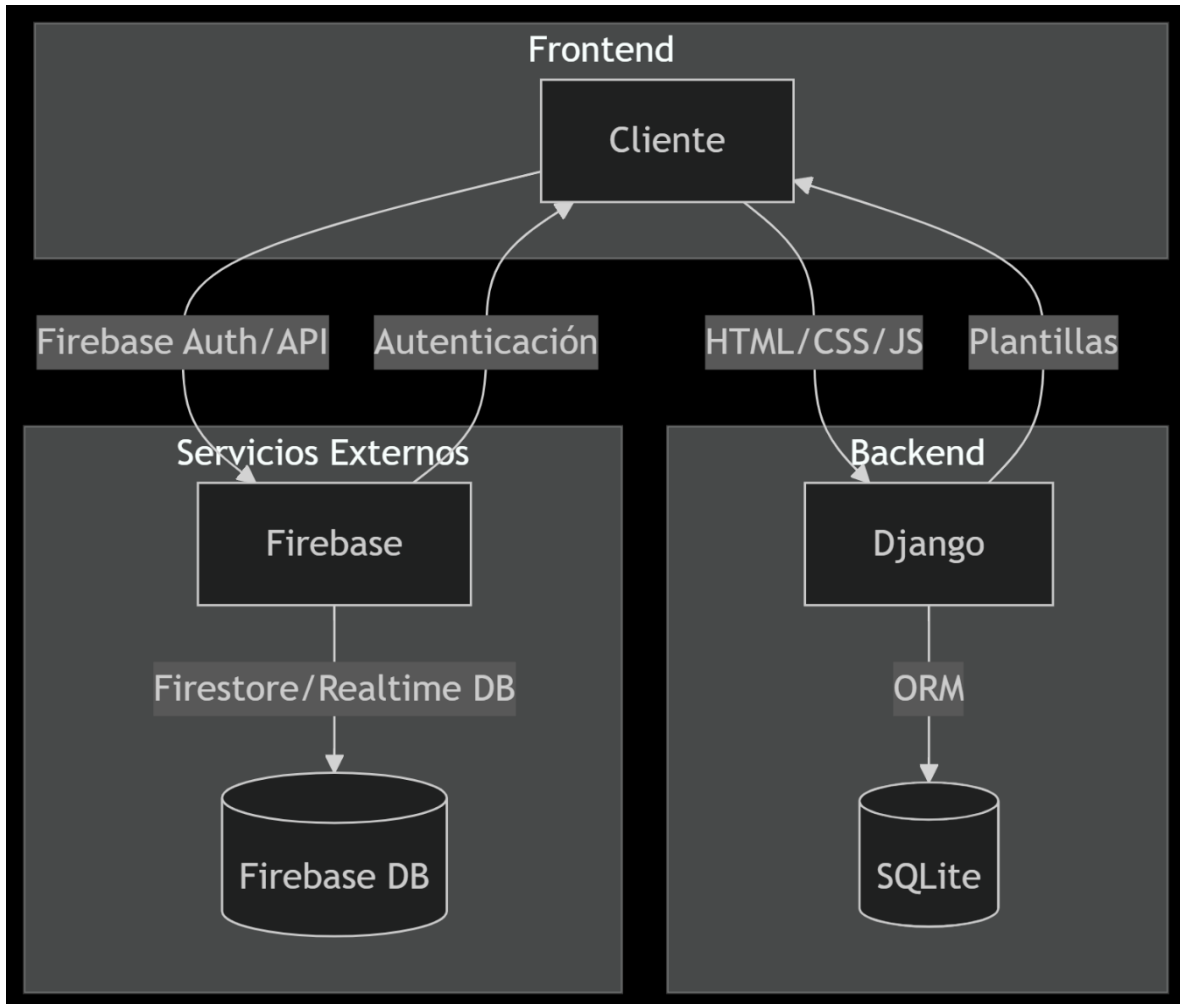
Estas tablas presentadas detallan las vistas principales y adicionales implementadas en el backend de Django para la aplicación Kodely. Cada vista está asociada a una URL específica, una plantilla HTML para la presentación, y especifica si requiere autenticación de usuario para su acceso. Esta organización permite un manejo claro de las rutas y funcionalidades dentro del sistema, facilitando el mantenimiento y la escalabilidad del proyecto.

## Diagramas de sistema y diseño relacionados

### 1. Diagrama de Arquitectura del Sistema

#### Descripción:

Este diagrama muestra la estructura **cliente-servidor** de Kodely, integrando Django (backend) y Firebase (servicios en la nube).



#### Componentes:

- **Cliente:**
  - Interfaz basada en plantillas Django (\*.html) con estilos CSS y lógica JS personalizada.
  - Se comunica con Firebase para autenticación y almacenamiento en tiempo real.
- **Backend (Django):**

- Gestiona rutas (urls.py), lógica de vistas (views.py), y modelos de datos (models.py).
- Usa SQLite como base de datos local para datos relacionales.
- **Firestore:**
  - Proporciona autenticación de usuarios (firebase-config.js) y base de datos NoSQL (Firestore/Realtime DB).

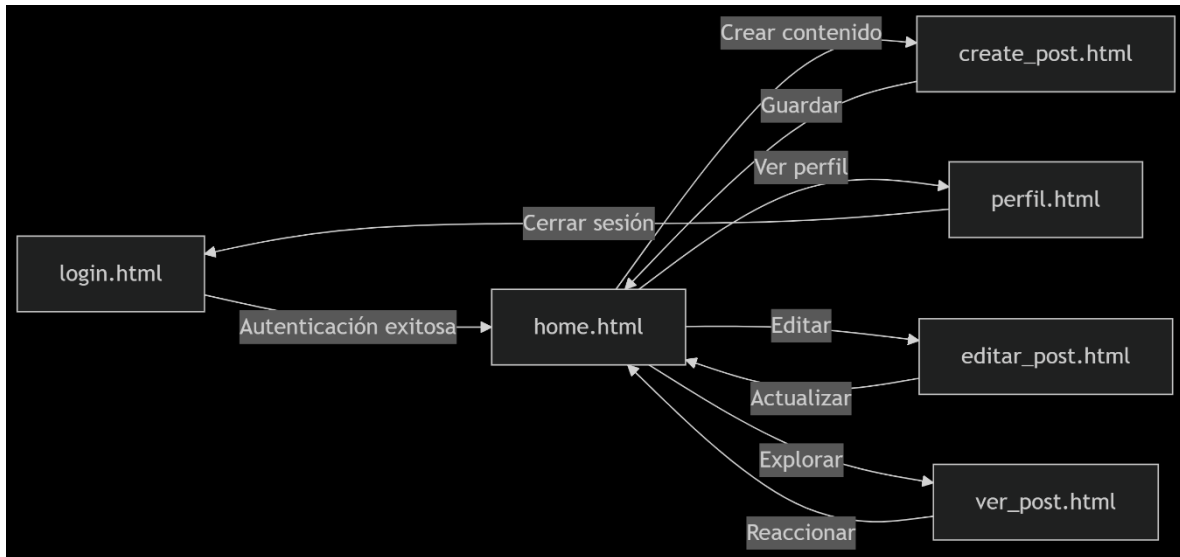
**Flujo de Datos:**

1. El cliente carga plantillas desde Django.
2. Las interacciones del usuario (ej: login) se validan con Firestore.
3. Django sincroniza datos locales (SQLite) con Firestore cuando es necesario.

## 2. Diagrama de Flujo de Navegación

### Descripción:

Visualiza la **ruta de usuario** entre las vistas principales del sistema.



### Puntos Clave:

- **Autenticación:**
  - Todo comienza en login.html, que redirige a home.html tras validar credenciales con Firebase.
- **Gestión de Contenido:**
  - Desde home.html, el usuario puede crear (create\_post.html), editar (editar\_post.html), o explorar posts (ver\_post.html).
- **Perfil y Cierre de Sesión:**
  - La opción de logout en perfil.html devuelve al usuario a login.html.

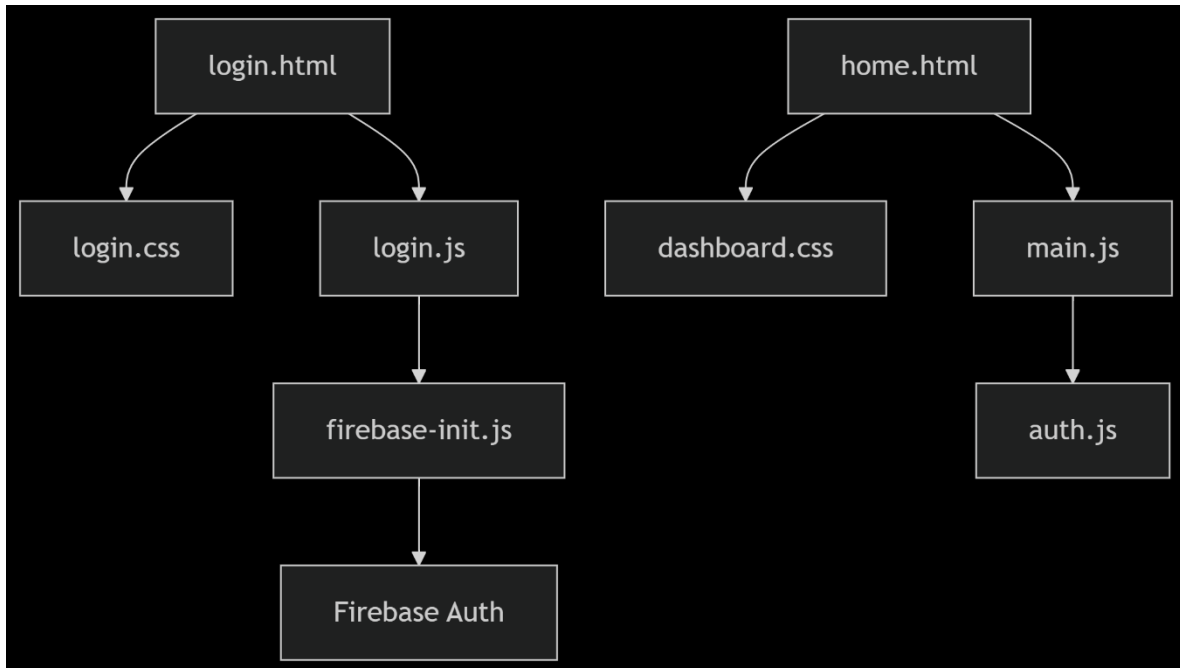
### Interacciones:

- Las flechas representan acciones del usuario (ej: clic en botones).
- Todas las vistas retornan a home.html tras completar una acción.

### 3. Diagrama de Componentes Frontend

#### Descripción:

Mapea la relación entre **templates**, **estilos** y **scripts** para cada vista.



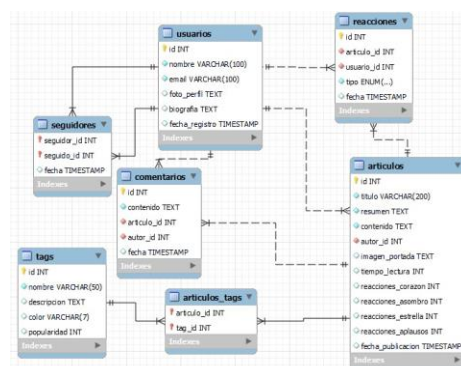
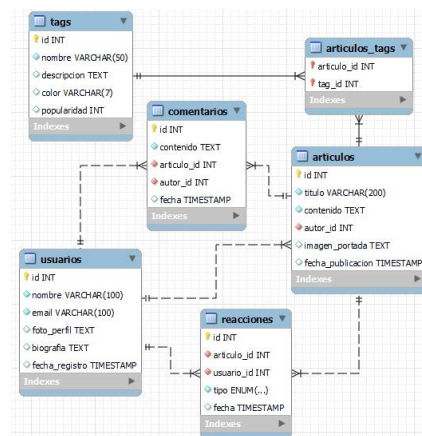
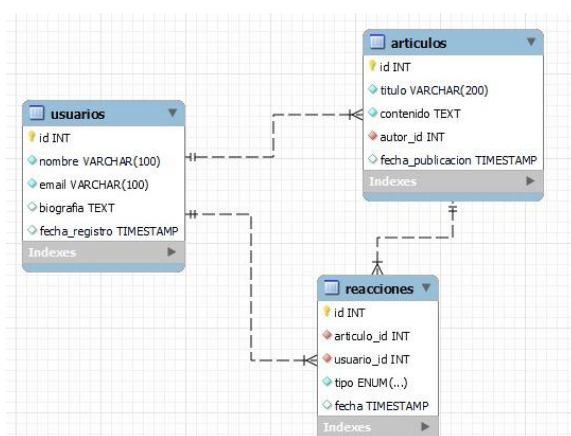
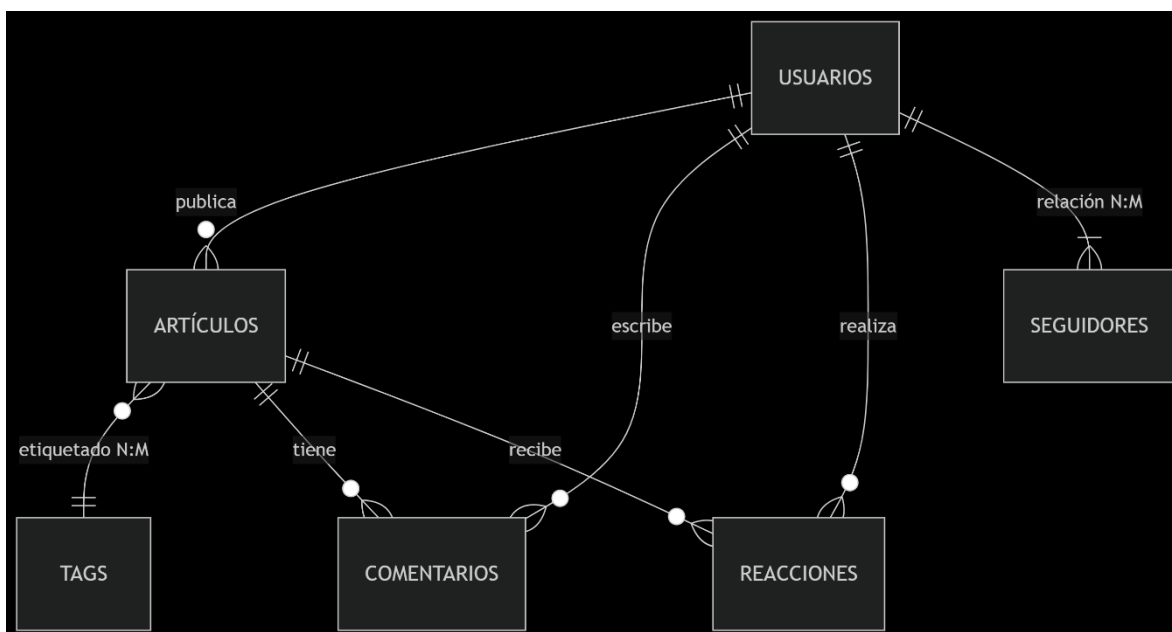
#### Relaciones:

- **Cada vista** login.html, home.html, etc. tiene:
  - Un archivo CSS dedicado login.css para estilos específicos.
  - Un archivo JS login.js para lógica de formularios.
- **Scripts compartidos:**
  - firebase-init.js y auth.js son reutilizados en múltiples vistas.

#### Dependencias:

- Los scripts de autenticación login.js, logout.js dependen de firebase-init.js para configurar la conexión.

## Diagrama de Base de Datos para Kodely



Este diagrama representa **la estructura central de almacenamiento de Kodely**, mostrando cómo se relacionan las tablas entre sí y qué datos críticos contienen. Aquí está el desglose:

## Entidades Principales

### 1. USUARIOS

- Almacena perfiles de desarrolladores.
- **Atributos clave:** id, o sea el identificador único, email para login con Firebase, foto\_perfil como avatar.
- **Relaciones:** Publica artículos, comenta, reacciona y sigue a otros usuarios.

### 2. ARTÍCULOS

- Contiene los tutoriales técnicos.
- **Atributos clave:** titulo, contenido es texto del post, autor\_id es vinculación al usuario.
- **Relaciones:** Recibe comentarios, reacciones y se etiqueta con tags.

### 3. TAGS

- Categoriza artículos por temas como ej: "Python", "WebDev".
- **Atributos clave:** nombre, color (para UI), popularidad (frecuencia de uso).

### 4. REACCIONES

- Registra interacciones con emojis.
- **Atributos clave:** tipo (emoji), articulo\_id (post relacionado).

### 5. COMENTARIOS

- Permite discusiones técnicas.
- **Atributos clave:** contenido, autor\_id (usuario que comenta).

### 6. SEGUIDORES

- Gestiona relaciones entre usuarios (quién sigue a quién).

## Relaciones Clave

### 1. Usuarios y Artículos

- **Relación:** Un usuario puede crear múltiples artículos, pero cada artículo pertenece a un único autor.
- **Ejemplo:** Si Luis es un usuario, puede publicar varios tutoriales, pero cada uno de esos tutoriales solo tendrá a Luis como autor registrado.

### 2. Artículos y Etiquetas

- **Relación:** Un artículo puede tener varias etiquetas, y una etiqueta puede aplicarse a múltiples artículos.
- **Ejemplo:** Un artículo sobre "Introducción a Django" podría tener las etiquetas "Python" y "Web", y esas mismas etiquetas pueden usarse en otros artículos.

### 3. Usuarios y Reacciones

- **Relación:** Un usuario puede dejar múltiples reacciones en diferentes artículos, pero cada reacción es registrada bajo un único usuario.
- **Ejemplo:** Ana puede reaccionar con un "like" a varios artículos, pero cada una de esas reacciones queda vinculada únicamente a su cuenta.

### 4. Artículos y Reacciones

- **Relación:** Un artículo puede recibir múltiples reacciones, pero cada reacción está asociada a un único artículo.
- **Ejemplo:** Un artículo popular podría acumular muchas reacciones, pero cada una de ellas está ligada exclusivamente a ese artículo.

### 5. Usuarios y Comentarios

- **Relación:** Un usuario puede escribir múltiples comentarios en diferentes artículos, pero cada comentario tiene un único autor.
- **Ejemplo:** Carlos puede comentar en varios tutoriales, pero cada comentario mostrará su nombre como autor.

### 6. Artículos y Comentarios

- **Relación:** Un artículo puede tener múltiples comentarios, pero cada comentario pertenece a un único artículo.
- **Ejemplo:** Un artículo sobre Firebase podría tener decenas de comentarios, pero todos ellos están asociados únicamente a ese artículo.



## 7. Usuarios y Seguidores

- **Relación:** Un usuario puede seguir a múltiples usuarios y, a su vez, ser seguido por múltiples usuarios.
- **Ejemplo:**
  - Luis sigue a Ana y a Carlos.
  - Ana sigue a Luis.
  - Carlos sigue a Luis y a Ana.

### Importancia de Estas Relaciones

- **Publicación de contenido:** Garantiza que cada artículo tenga un autor definido.
- **Organización:** Las etiquetas permiten categorizar y filtrar artículos por temas.
- **Interacción:** Las reacciones y comentarios facilitan la participación de la comunidad.
- **Red social:** El sistema de seguidores permite crear conexiones entre usuarios.

### Ejemplo de Funcionamiento

Cuando un usuario, como Ana, publica un artículo:

1. El sistema registra el artículo bajo su nombre relación usuario-artículo.
2. Ana asigna etiquetas como "Programación" y "Python" relación artículo-etiqueta.
3. Otros usuarios pueden comentar relación usuario-comentario-artículo o reaccionar relación usuario-reacción-artículo.
4. Si a Luis le gusta el artículo, puede seguir a Ana para ver más de sus publicaciones relación seguidores.

### Cómo se Usa en Kodely

- **Autenticación:** El id de USUARIOS se vincula con Firebase Auth.
- **Interfaz:** Los tags.color definen estilos visuales en la UI.
- **Analítica:** Conteo de reacciones.tipo muestra tendencias de contenido.

## Tabla de Modelos

Modelos del sistema

Modelo	Campos Principales	Relaciones	Ubicación
<b>Post</b>	title, content, author, created_at, updated_at	FK a User como autor del tutorial	core/models.py
<b>Reaction</b>	post, user, emoji	FK a Post, FK a User	core/models.py

**Notas:**

- Un User puede tener múltiples Post.
- Un Post puede tener múltiples Reaction.
- Cada Reaction pertenece a un único usuario y un único post.

## Tabla de Flujo Resumido de Usuario

Flujo de navegación

Página / Vista	Acceso desde	Acción principal	Redirige a
<b>login.html</b>	Entrada principal	Iniciar sesión con Google	home.html
<b>home.html</b>	Login o navegación	Ver publicaciones	ver_post.html
<b>ver_post.html</b>	Lista de publicaciones	Leer un post completo	X
<b>perfil.html</b>	Barra de navegación	Ver publicaciones propias	editar_post.html
<b>create_post.html</b>	Botón "Publicar"	Crear nueva publicación	home.html
<b>editar_post.html</b>	Desde perfil.html	Editar publicación existente	home.html