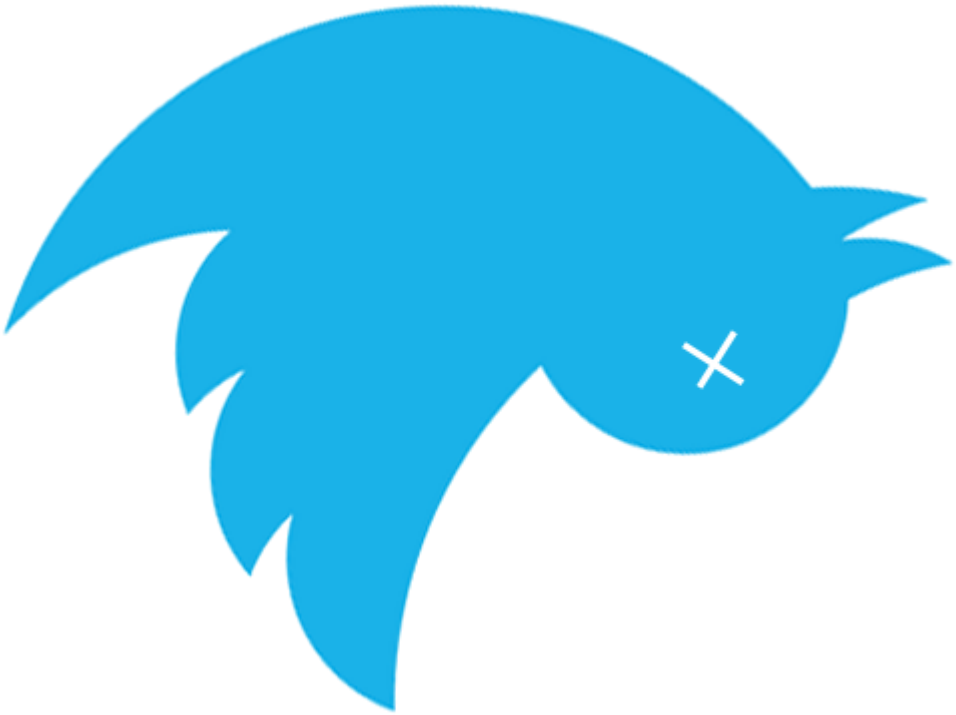
Proyecto integrado
Clon de Twitter



Autor	David Gómez Redondo
Tutor	Francisco Javier Ávila Sánchez
Centro	I.E.S. Francisco Romero Vargas (Jerez de la Frontera)
Estudios	Desarrollo de aplicaciones web
Curso	2022 / 2023

Índice

Introducción

Clon de Twitter, como su propio nombre indica, es una aplicación de funcionalidades y diseño similares a las de la conocida red social. Más concretamente, Twitter es una excusa para desarrollar un proyecto completo que permite a sus usuarios consumir pequeñas píldoras de contenido compartido por otros usuarios de la aplicación.

Los fundamentos de casi cualquier red social, y por extensión de este proyecto, hunden sus raíces en la Web2. La segunda generación de la web puso énfasis en la interacción con los usuarios, que se convertían por primera vez en sujetos que construyen internet. Cualquier persona con acceso a internet podía crear contenido e interactuar con el contenido de otros.

La cesión de la autoría de la mayoría de contenidos a los usuarios vino con múltiples desafíos, vigentes a día de hoy en el mundo del desarrollo web. Los sitios web destacan por su alto nivel de interactividad y su uso masivo del lenguaje de programación JavaScript para conseguirla.

El desarrollo de este clon se realiza con el conjunto de tecnologías T3. Este incluye lo mejor del ecosistema de desarrollo en TypeScript montado sobre los mejores cimientos: Next.js “El framework de React”. Estas tecnologías modernas se encuentran a la altura de casi cualquier desafío en el mundo web y permiten crear Single Page Applications sin renunciar al SEO y el rendimiento.

La infraestructura para compartir este proyecto con el mundo recae sobre Vercel y PlanetScale. Estas dos empresas ofrecen sus servicios de manera gratuita para uso personal y hobby, sus limitaciones son bastante generosas y más que suficientes para iniciar este proyecto sin tener que incurrir en costes adicionales. Para guardar las imágenes que suben los usuarios se recurre a Cloudinary por motivos similares.

Por último, y no menos importante, el desarrollo se realiza utilizando Visual Studio Code junto con Git y GitHub para el control de versiones. En conclusión, se pretende encarar un problema ya habitual con las más nuevas y potentes herramientas que se ponen a nuestra disposición y las mejores prácticas.

Objetivos

El principal objetivo de Clon de Twitter es reproducir la funcionalidad mínima de la aplicación a la que se hace referencia. Para conseguirlo la aplicación debe permitir al usuario:

- Visitar la web tanto desde un dispositivo móvil como desde escritorio, con un diseño adaptativo que evita la pérdida de información o perjudique la experiencia de usuario.
- Autenticarse.
- Tener un perfil visitable y editable con nombre, foto de perfil, biografía y fondo (estando autenticado).
- Publicar un nuevo tweet (estando autenticado), que contenga solo texto, texto e imagen o solo imagen.
- Ver los tweets más recientes publicados por todos los usuarios.
- Ver los tweets más recientes publicados por un usuario concreto desde su perfil.
- Que se sigan cargando tweets anteriores al hacer scroll hacia abajo.
- Acceder a una vista detallada de un tweet.
- Interaccionar con los tweets, por ejemplo, indicando que le gustan (estando autenticado).
- Conocer el autor, fecha de publicación y número de interacciones de un tweet.
- Ser informado de que un usuario ha indicado que le gusta uno de tus tweets.
- Seguir a otro usuario (estando autenticado).
- Ver los tweets más recientes de los usuarios que sigues (en lugar de los de todos los usuarios).
- Consultar qué usuarios sigue y siguen a un usuario concreto.

Tecnologías escogidas y justificación

Tal y como se comenta en la introducción, se ha empleado el conjunto de tecnologías T3 (1). Este incluye principalmente Next.js, TypeScript, Tailwind CSS, tRPC, Prisma y NextAuth.js, cada una de ellas tiene su propio espacio en esta sección. También se abordan en esta sección los servicios de infraestructura de Vercel y PlanetScale, así como Cloudinary para almacenamiento de imágenes.

Create T3 App

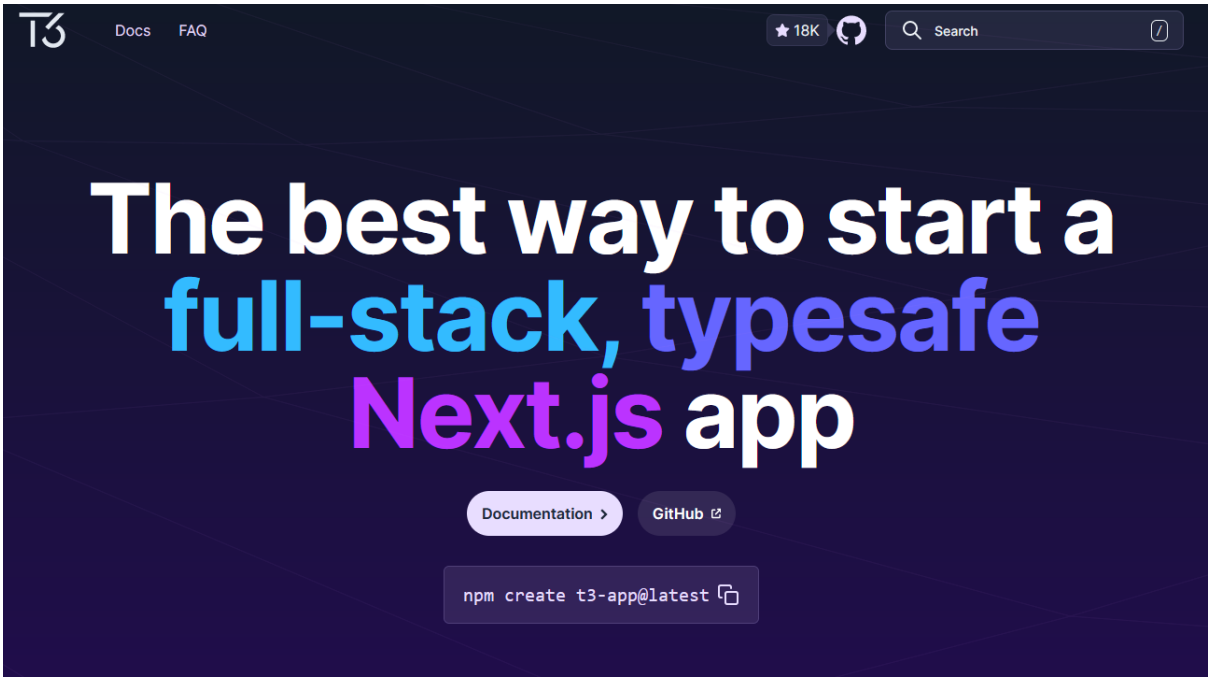


Figura 1. Página de inicio del T3 (10/06/2023).

T3 se centra en ofrecer una experiencia de desarrollo sencilla, modular, full-stack y con tipado estático gracias a TypeScript. La plantilla que genera su herramienta de línea de comandos permite al desarrollador centrarse desde el primer minuto en su proyecto y en la entrega de valor.

Next.js

Next.js (2) es un framework para desarrollo web que permite diseñar la interfaz web utilizando React (3) y agrega una serie de características vitales para crear sitios web profesionales. Ha sido desarrollado por Vercel, haciendo que su alojamiento en este servicio sea algo trivial. El propio sitio web de React recomienda su uso.

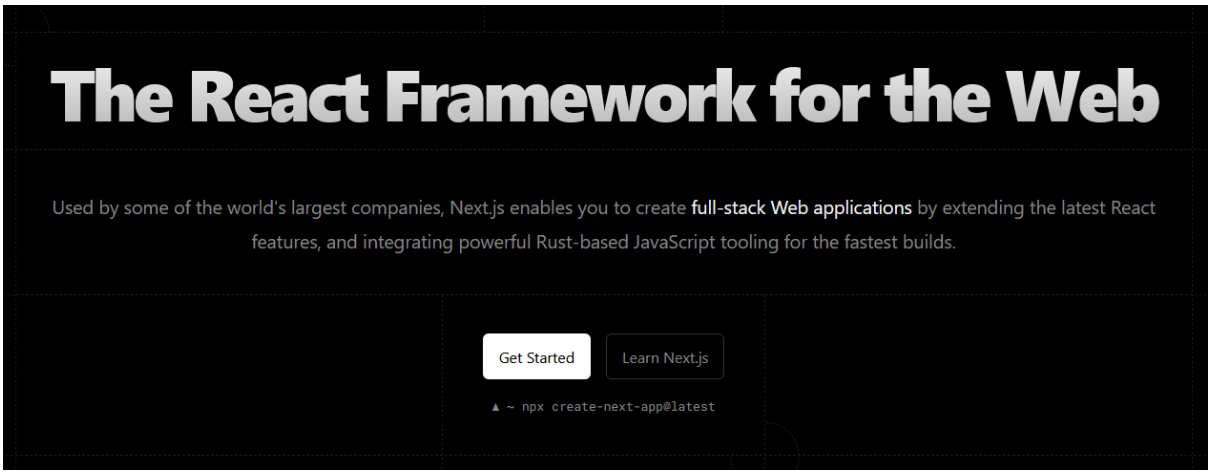


Figura 2. Página de inicio de Next.js (10/06/2023).

La principal característica que aporta Next.js es la posibilidad de renderizar la interfaz en el lado del servidor con un sistema de enrutamiento basado en el sistema de archivos. Gracias a esto se ofrece al desarrollador la posibilidad de emplear distintas estrategias de renderizado y una serie de optimizaciones imposibles de conseguir con un sitio web estático basado en React y otras librerías del lado del cliente.

Al construir una aplicación en Next.js se solucionan algunas importantes carencias de React relativas al SEO y a la entrega y ejecución de JavaScript en el lado del cliente. Además, el desarrollo de ciertos componentes se simplifica notablemente, en general, esta tecnología ofrece una buena experiencia de desarrollo y es compatible con TypeScript.

TypeScript

TypeScript (4) es un lenguaje de programación compilado que es un superset de JavaScript. Ofrece una experiencia de desarrollo con tipado fuerte y un paso de compilación a JavaScript que evita la llegada de lo que antes eran errores en tiempo de ejecución a producción.

Figura 3. Página de inicio de TypeScript (10/06/2023).

Otras de las características más relevantes de este lenguaje es la inclusión de interfaces y la posibilidad de analizar estáticamente el código, algo que incorporan los entornos de desarrollo integrados y de lo que se benefician también los desarrollos en JavaScript.

La elección de TypeScript sobre JavaScript es una tendencia en el sector que se ve reflejada en algunas encuestas (5), esta tecnología ya aparece entre los lenguajes de programación más populares por encima de Java, C# o php. Usar TypeScript evita bugs y facilita el mantenimiento y escalado de un proyecto. Por todo esto, la experiencia de desarrollo con este lenguaje mejora mucho frente a JavaScript y se opta por desarrollar todo el proyecto con él.

Tailwind CSS

Tailwind (6) es un framework de CSS que abandona el uso de ficheros de estilo y ofrece un conjunto de clases de utilidad que se usan directamente en el HTML. La práctica totalidad de las propiedades CSS tienen una clase equivalente cuando se trabaja con este framework.

La elección de Tailwind CSS se justifica en lo ágil que vuelve el desarrollo. Una frase que se suele atribuir a Phil Karlton dice que solo hay dos cosas complicadas en programación: la invalidación de caché y nombrar cosas. Con este framework nunca más hay que pensar en ponerle nombre a las clases CSS o qué convención seguir. Tampoco hay que preocuparse de cómo organizar y estructurar los ficheros de estilo.

Por supuesto, esta tecnología ofrece más que lo mencionado. Destaca su capacidad para aceptar valores arbitrarios en las clases, permite purgar las clases no usadas para optimizar el tamaño del artefacto y se despoja de prácticamente todos los estilos por defecto del navegador.

tRPC

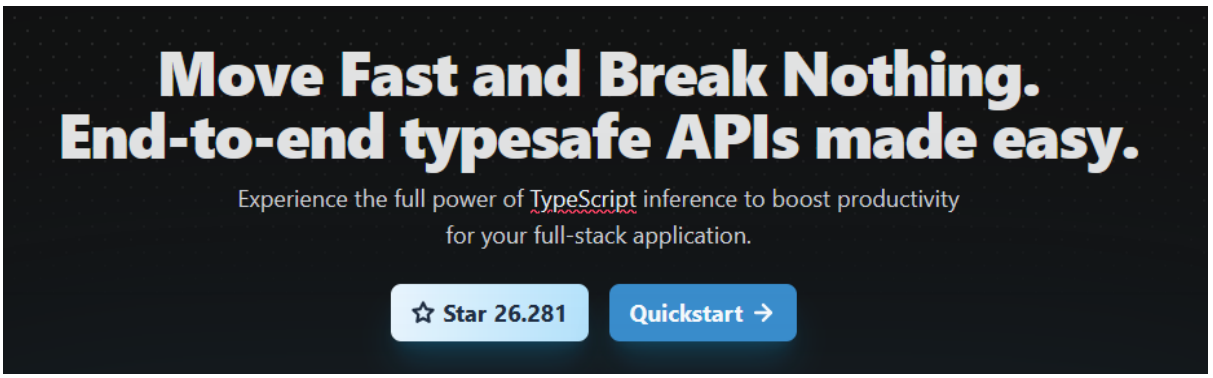


Figura 4. Página de inicio de tRPC (10/06/2023).

tRPC (TypeScript Remote Procedure Call) (7) es una librería de TypeScript que ofrece una alternativa a las APIs REST y GraphQL para proyectos full-stack que comparten repositorio.

Esta tecnología permite definir funciones en el lado del servidor utilizando el patrón de diseño constructor (Builder) (8) que luego se consumen en el cliente. Los procedimientos creados de este modo están perfectamente tipados y permite abstraerse del protocolo HTTP muy cómodamente.

tRPC se integra perfectamente con Next.js utilizando la librería zod (9) para definir los tipos de los procedimientos y React Query (10) en el lado del cliente, dando así la mejor experiencia de desarrollo y permitiendo realizar cambios sin miedo a provocar bugs.

Prisma

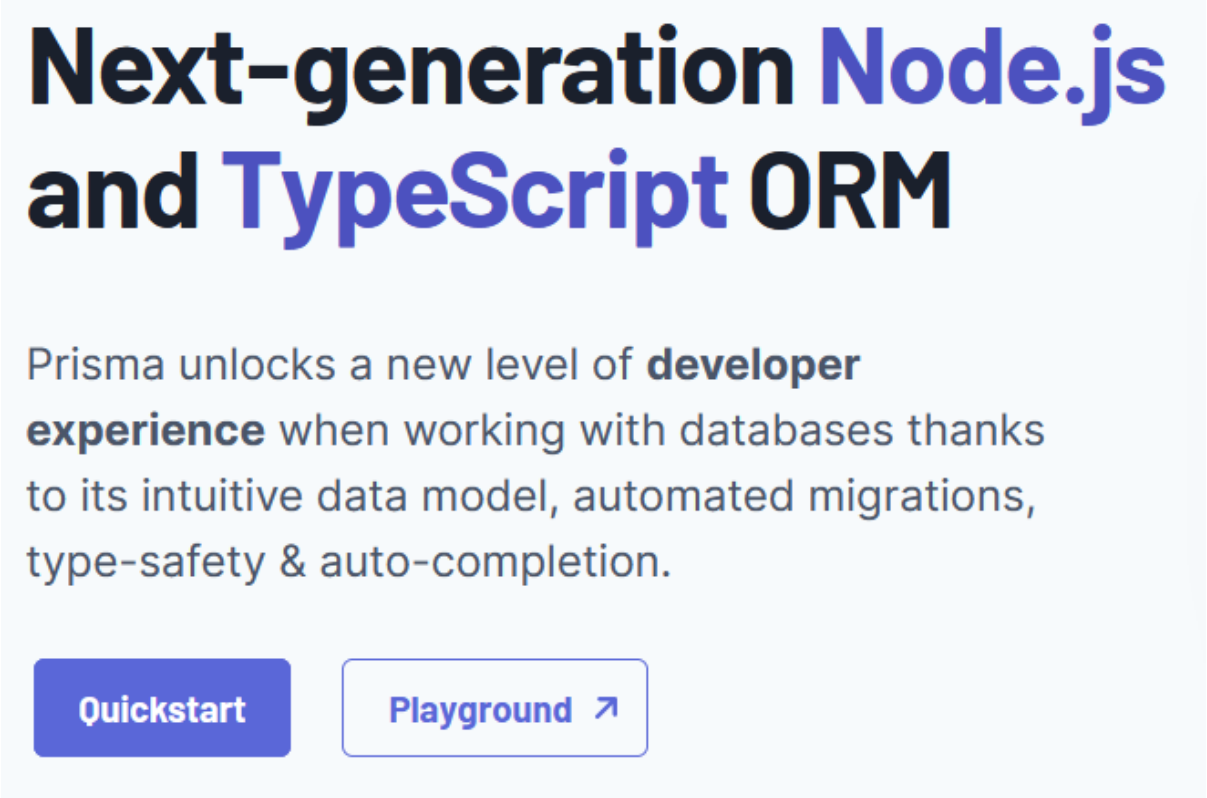


Figura 5. Página principal de Prisma (11/06/2023).

Prisma (11) es el object relational mapper elegido para este proyecto. Este ofrece la posibilidad de trabajar con múltiples bases de datos, tanto SQL como noSQL de manera agnóstica gracias al uso de un lenguaje de definición de entidades propio.

A partir del modelo de datos definido, Prisma permite realizar migraciones automáticas y también generar un cliente completamente tipado. Este cliente puede ser empleado por el desarrollador cómodamente y le permite acceder a modelos autogenerados con todos los métodos deseables para consultar y modificar la información.

A pesar de que TypeScript es un lenguaje de tipado fuerte, sigue siendo más “libre” que Java o C#. Definir los modelos de datos basándose en clases o interfaces de este es problemático porque es fácil introducir problemas al trasladarlo a base de datos. El esquema de Prisma y su generalización automática de código y migraciones solventa este problema, convirtiéndo en una solución ideal.

NextAuth.js

NextAuth.js es un proyecto open source empujado por la comunidad de desarrolladores de Next.js para ofrecer una solución de autenticación fácil, sencilla y segura. Destaca lo fácil que es incluir distintos proveedores de autenticación.

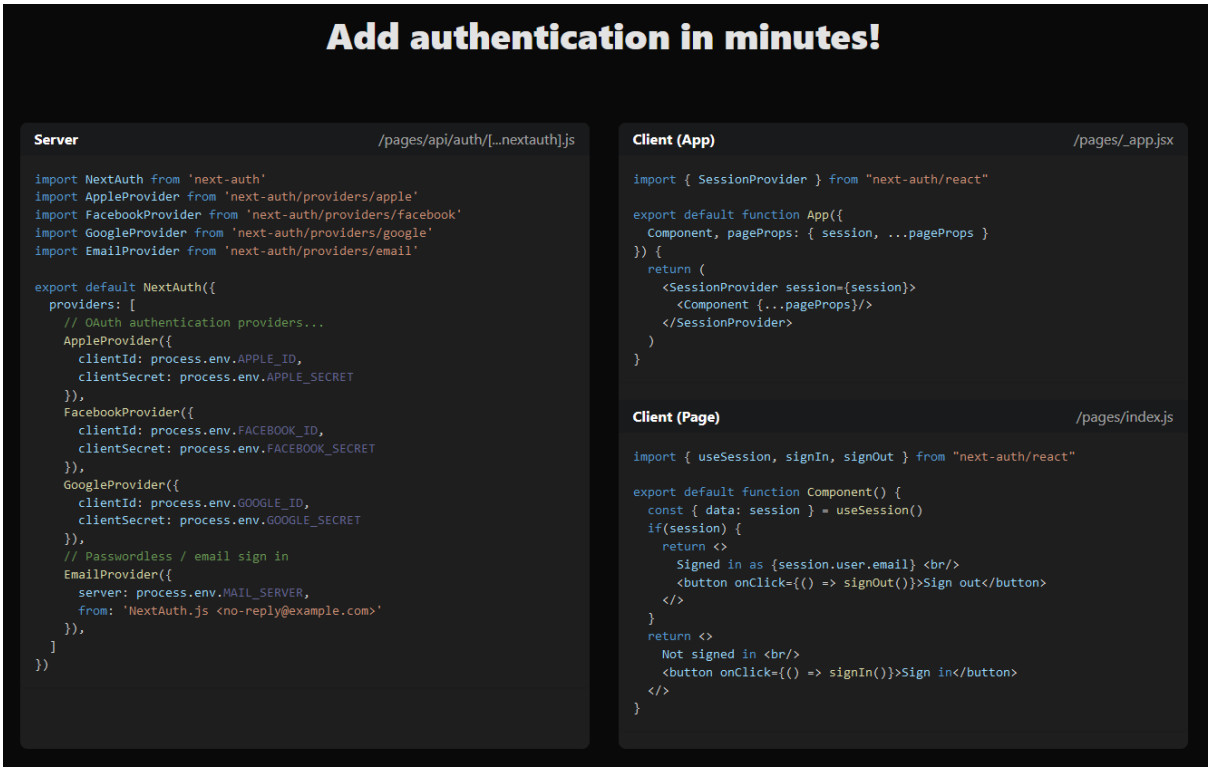


Figura 6. Ejemplo de uso de NextAuth.js en su página principal (11/06/2023).

NextAuth.js es una gran elección para este proyecto porque permite tener un sistema de autenticación muy rápido que permite al desarrollador centrarse en la aplicación. Además, el amplio número de proveedores que soporta y las opciones para integrar soluciones basadas en email, sin contraseña, active directory, … hace que el proyecto pueda ofrecer distintas alternativas en el futuro implementadas casi sin esfuerzo.

Vercel

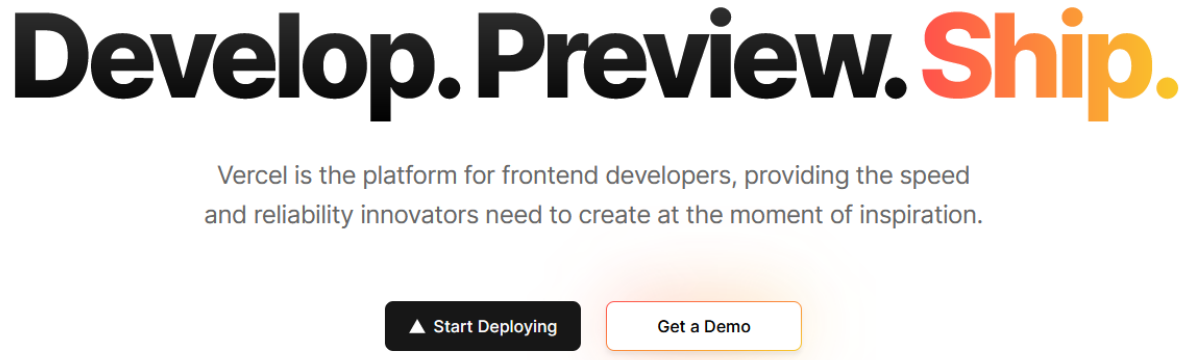


Figura 7. Página principal de Vercel (11/06/2023).

Vercel (12) es una plataforma orientada a desarrolladores que ofrece servicios de infraestructura serverless (13). Esta plataforma ofrece una herramienta de línea de comandos y una integración sencilla e intuitiva con GitHub.

En Vercel los desarrolladores pueden desplegar sus sitios web estáticos y obtener dominios automáticamente generados. Este servicio es inicialmente gratuito para uso personal, con bastante margen hasta tener que incurrir en costes. Gracias a su integración con GitHub es sencillo vincular estos despliegues a un repositorio y generarlos cada vez que se integran cambios en las distintas ramas del proyecto. Permite asimismo desplegar varias ramas del proyecto e indicar una rama de producción y otra de preview.

Además de sitios web estáticos, Vercel permite alojar aplicaciones Next.js, Nuxt y SvelteKit. Frameworks creados por esta misma compañía o que colaboran con sus creadores como es el caso de Rich Harris y SvelteKit. Así, esta plataforma se convierte en la primera opción para cualquier proyecto Next.js, ya que permite sacar todo el potencial del framework y de los distintos servicios de infraestructura de Vercel.

PlanetScale

PlanetScale (14) ofrece un servicio de base de datos MySQL serverless que destaca, entre otras cosas, porque ofrece la posibilidad de trabajar con la base de datos usando ramas para las distintas versiones al estilo de Git.

Con PlanetScale es muy fácil distinguir entre la base de datos de producción y la de desarrollo y luego ir reconciliando los cambios necesarios. Además, la plataforma no solo se encarga del manejo del servidor, sino que ofrece un panel con información sobre el desempeño de la base de datos muy completo. También dispone de una consola web para realizar consultas sobre la base de datos directamente desde el navegador.

Desarrollar desde el principio con latencia y aplicar cambios de manera incremental a la base de datos son dos cosas que se buscaban para este proyecto y PlanetScale las ofrece con una generosa suscripción gratuita.

Cloudinary

Cloudinary (15) es una plataforma para el almacenamiento y gestión de imágenes y videos. Esta ofrece una sencilla API REST para subir imágenes de manera programática y pone a disposición de los desarrolladores opciones para optimizar y transformar las imágenes.

Esto último no se utiliza a fecha de entrega de este proyecto pero es importante destacar que está ahí y que se puede explorar para mejorar la gestión de recursos multimedia.

Diseño de la aplicación

Modelo entidad-relación

Se representa en la figura 8 el diagrama entidad-relación que implemente la aplicación.

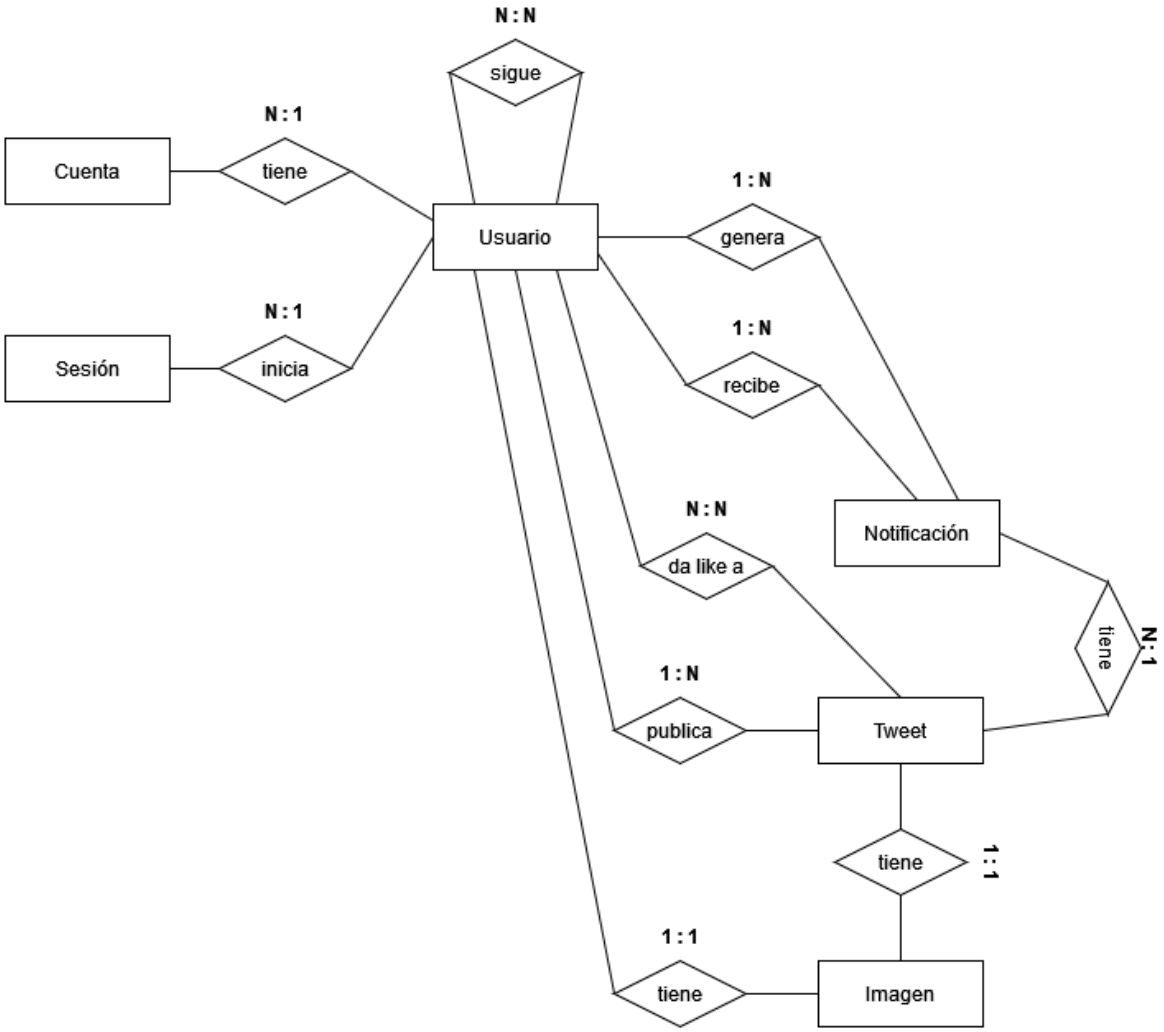


Figura 8. Diagrama entidad-relación con el diseño de la aplicación.

El usuario es el centro de toda la aplicación puesto que es quien inicia todas las acciones con consecuencias en la persistencia de datos y el estado del sitio web. Los usuarios al iniciar sesión con su proveedor de identidad crean una cuenta y una sesión, la sesión se va renovando, puede cerrarse y puede tener lugar en otros dispositivos.

El usuario puede publicar tweets o interactuar con los existentes, esto genera nuevas notificaciones que se entregan al usuario autor del tweet. También el usuario puede disponer de una imagen para el fondo de su perfil y este mismo modelo de imagen se emplea para almacenar aquellas que se usan en tweets.

La utilización de un modelo de imagen en lugar de emplear un simple campo de texto en distintas tablas con la url correspondiente tiene varias motivaciones:

- Esta aproximación permite almacenar metadatos de la imagen como ancho y alto. Esta información es relevante para realizar optimizaciones con el tamaño de las imágenes, puesto que las dimensiones originales son útiles para determinar la relación de aspecto.
- Se puede agregar metainformación sobre la imagen incorporada por el usuario. Se permite al usuario agregar texto alternativo a la imagen, que es elemento crítico para garantizar la accesibilidad.
- La existencia de este modelo facilitará agregar nuevas características en el futuro, por ejemplo, etiquetar una imagen como contenido sensible.

Modelo físico

User

`id`	varchar(191) NOT NULL
`name`	varchar(191)
`email`	varchar(191)
`emailVerified`	datetime(3)
`image`	varchar(191)
`backgroundId`	varchar(191)
`bio`	varchar(191) NOT NULL DEFAULT ''

Clave primaria `id`, `email` es una campo restringido a valores únicos y `backgroundId` es clave foránea a la tabla Image.

Tweet

`id`	varchar(191) NOT NULL
`userId`	varchar(191) NOT NULL
`content`	varchar(191) NOT NULL
`createdAt`	datetime(3) NOT NULL DEFAULT current_timestamp(3)
`imageUrl`	varchar(191)
`imageId`	varchar(191)

Clave primaria `id`, la combinación de los campos `createdAt` e `id` es única, `userId` e `imageId` son claves foráneas a sus respectivas tablas.

Session

`id`	varchar(191) NOT NULL
`sessionToken`	varchar(191) NOT NULL
`userId`	varchar(191) NOT NULL
`expires`	datetime(3) NOT NULL

Clave primaria `id`, `sessionToken` es un campo restringido a valores únicos, `userId` es una clave foránea a la tabla User.

Notification

`id`	varchar(191) NOT NULL
`tweetId`	varchar(191) NOT NULL
`notifiedId`	varchar(191) NOT NULL
`notifierId`	varchar(191) NOT NULL
`type`	varchar(191) NOT NULL
`read`	tinyint(1) NOT NULL DEFAULT '0'
`createdAt`	datetime(3) NOT NULL DEFAULT current_timestamp(3)

Clave primaria `id`, la combinación de los campos `createdAt` e `id` es única, `notifiedId` e `notifierId` son claves foráneas a la tabla User.

Like

`userId`	varchar(191) NOT NULL
`tweetId`	varchar(191) NOT NULL

La clave primaria es la combinación de ambos campos y cada uno es una clave foránea a sus respectiva tabla.

Image

`id`	varchar(191) NOT NULL
`width`	int NOT NULL
`height`	int NOT NULL
`secureUrl`	varchar(191) NOT NULL
`alt`	varchar(191) NOT NULL DEFAULT 'No alternative text provided'

Clave primaria `id`.

Account

`id`	varchar(191) NOT NULL
`userId`	varchar(191) NOT NULL
`type`	varchar(191) NOT NULL
`provider`	varchar(191) NOT NULL
`providerAccountId`	varchar(191) NOT NULL
`refresh_token`	text
`access_token`	text
`expires_at`	int
`token_type`	varchar(191)
`scope`	varchar(191)
`id_token`	text
`session_state`	varchar(191)

Clave primaria `id`, la combinación de los campos `provider` y `providerAccountId` es única y `userId` es una clave foránea a la tabla User.

_Followers

`A` varchar(191) NOT NULL
`B` varchar(191) NOT NULL

Esta tabla permite representar la relación many to many (`A` sigue a `B`) entre usuarios que siguen o son seguidos por otros usuarios. La combinación de los campos es única y el campo `B` es clave foránea de la tabla User.

Arquitectura de la aplicación

Estructura del proyecto

La estructura del proyecto viene mayormente determinada por el uso de Next.js, teniendo como principales directorios los siguientes:

- **prisma**: Contiene la definición del modelo de datos que emplea Prisma.
- **public**: Contiene ficheros estáticos que se sirven en la raíz del proyecto después de compilar la aplicación.
- **src**: Contiene todo el código fuente de la aplicación.
    - **components**: Incluye ficheros tsx donde se definen componentes de React.
    - **interfaces**: Incluye ficheros ts donde se definen interfaces.
    - **pages**: Este directorio es utilizado por Next.js para generar las rutas de la aplicación, contiene ficheros tsx donde se definen componentes de servidor de React. Los subdirectorios y los nombres de los ficheros son significativos pues determinan cómo se van a conformar las rutas de la aplicación. Incluye un directorio api donde se exponen los recursos definidos con tRPC.
    - **server**: Contiene algunos ficheros de configuración y arranque, además del directorio api.
        - **api**: Instancia el tRPC router necesario para poder registrar procedimientos.
            - **routers**: Incluye ficheros ts que definen routers específicos (tweet, profile, notification).
            - **services**: Incluye los procedimientos definidos con ts y tRPC que se registran en los ficheros del directorio routers.
    - **styles**: Este directorio permite añadir hojas de estilos globales.
    - **utils**: Incluye algunas funciones útiles que no están directamente relacionadas con alguna entidad de dominio, configuración o función específica.

Recursos externos

Los iconos de la aplicación provienen de la librería React Icons (16), que incluye una gran colección de iconos svg como componentes de React personalizables y listos para incluir en cualquier proyecto basado en esta tecnología.

Manual de despliegue y usuario

Para desplegar esta aplicación se requiere estar registrado en diferentes servicios, que son: GitHub, PlanetScale, Discord, Cloudinary, Vercel.

Github

Es necesario registrarse en GitHub para mantener nuestro repositorio online ya sea de forma pública o privada. El despliegue con Vercel utiliza como punto de partida un repositorio en GitHub.

PlanetScale

Se accede a PlanetScale utilizando las credenciales de GitHub o unas únicas para el servicio. Una vez autenticado se puede crear una nueva base de datos en la región más conveniente.

Después de que la base de datos está creada se visita la pestaña Branches y se promociona la rama main (que viene por defecto) a rama de producción. A continuación, se crea una nueva rama de nombre dev basada en main en la misma región.

Ahora que la base de datos está creada y se tiene rama de desarrollo y de producción hay que obtener la información necesaria para conectarse a estas. Accediendo a cada rama se puede pulsar el botón connect, lo que muestra una ventana modal donde se puede seleccionar Prisma en Connect with. Al seleccionar Prisma se muestra un texto copiable con DATABASE_URL, que será la variable de entorno para acceder a la base de datos correspondiente.

Una vez añadida la variable de entorno se puede hay que hacer que Prisma aplique la migración a la base de datos. Con el proyecto en local, las dependencias instaladas y DATABASE_URL de la rama dev en el fichero .env se ejecuta el comando npx prisma db push desde la raíz del proyecto. Después de un momento los cambios se habrán aplicado, lo que se puede comprobar en el sitio de PlanetScale.

Por último, ahora que la rama dev tiene las tablas creadas y relacionadas hay que crear una deploy request para aplicar esos mismos cambios a la rama main.

Discord

Para utilizar Discord como proveedor de identidades hay que crear una aplicación desde su portal para desarrolladores (17). De hecho, se crearán dos aplicaciones: una para desarrollo y otra para producción.

Una vez registrados en Discord se accede al mencionado portal y se crea una nueva aplicación con el botón New Application. Se nombra la aplicación y se aceptan los términos y condiciones, es oportuno incluir alguna mención para distinguir entre la aplicación de desarrollo y de producción.

Con la aplicación creada se visita la sección OAuth2, donde se copia el valor de Client Id para la variable de entorno DISCORD_CLIENT_ID y se pulsa el botón Reset secret para obtener el Client Secret que irá a la variable de entorno DISCORD_CLIENT_SECRET. Esto hay que repetirlo con la segunda aplicación que será empleada en otro entorno.

Para que la autenticación funcione correctamente hay que añadir un redireccionamiento. Para desarrollo local la url que hay que introducir es http://localhost:3000/api/auth/callback/discord, en general la url tiene la forma <base app url>/api/auth/callback/discord. Una vez desplegada la aplicación y conocido su dominio habrá que hacer el mismo proceso. Así se consigue que la autenticación funcione en el entorno de producción o cualquier otro entorno con distinto dominio.

Cloudinary

Se accede a Cloudinary utilizando GitHub como en otras ocasiones y se crea una nueva cuenta con el plan gratuito. Para utilizar Cloudinary de manera programática es necesario crear un nuevo preset, que es la utilidad de esta plataforma para configurar un enlace de subida a la nube de nuestro usuario.

Para crear el preset se accede a la configuración de Cloudinary y se selecciona la pestaña Upload, en esta se puede encontrar una sección llamada Upload presets donde se pueden crear y editar estos. Desde aquí se clica en Add Upload Preset y en la nueva vista se le da un nombre al preset, se indica el signing mode Unsigned y una carpeta con el mismo nombre que el preset (donde se subirán todos los recursos), después simplemente de guarda pulsa el botón Save.

Una vez creado, se incluye el entorno de Cloudinary (abajo a la izquierda en el sitio web) y el nombre del Preset en el fichero cloudinaryConfig.ts dentro de src/utils. En esta ocasión no se crean dos entornos distintos porque el plan gratuito solo ofrece uno.

Vercel

Tras registrarse en Vercel y desde su Dashboard se añade un nuevo proyecto, en la siguiente vista se vincula la cuenta de GitHub correspondiente y se selecciona el repositorio adecuado. Al tratarse de un proyecto en Next.js no es necesario preocuparse por más configuración con la excepción de las variables de entorno.

Las siguientes variables de entorno son necesarias para el correcto funcionamiento de la aplicación, una vez incluidas se da el ok y el despliegue debe tener lugar.

- NEXTAUTH_URL
- DISCORD_CLIENT_SECRET
- DISCORD_CLIENT_ID
- DATABASE_URL
- NEXTAUTH_SECRET

En local estas se han definido dentro de un fichero .env en la raíz del proyecto. Sin embargo, recuerda que no deben ser las mismas, ya que se diferencian dos entornos. Aquí se emplean las variables obtenidas para los entornos de producción.

Gestión del proyecto

Problemas encontrados

Anidación de anchor

Un problema reseñable es el de la anidación de enlaces en un elemento de la interfaz. La tarjeta que presenta un tweet incluye enlaces al perfil del autor, al detalle del tweet y a la imagen si la hubiera. Concretamente, se quiere que cualquier cosa que no sea hacer clic para realizar una interacción, ir al perfil del autor o visitar la imagen resulte en acceder al detalle del tweet.

Para conseguir esto lo primero que se piensa es envolver todo con un anchor. Esta aproximación funciona pero va en contra del estándar HTML, para evitar esta violación del estándar que puede incurrir en problemas desconocidos hay que replantear la solución. Con esto en mente se crea el componente LinkOverlay, que aplica una solución a este problema encontrada en StackOverFlow (18). El principio es sencillo, ubicar el anchor fuera y extenderlo con CSS dándole posicionamiento absoluto.

Hooks en componentes de servidor

Otro problema, que es recurrente y habitual al emplear Next.js, es intentar emplear hooks en las páginas que se renderizan en el lado del servidor. No es un problema grave porque se identifica rápidamente. La solución más sencilla es devolver un componente dentro del componente del lado del servidor, este componente separado sí que permite emplear hooks porque se carga del lado del cliente.

Actualizar propiedades de un nodo HTML al cargar

También relacionado con el uso de hooks, es interesante comentar lo que ocurre con el textarea del formulario para crear un tweet. Este textarea se redimensiona con el contenido del texto que se escribe utilizando su propiedad scrollHeight. Para acceder a esta propiedad del elemento react ofrece el hook useRef, y con un efecto se puede conseguir el comportamiento deseado. Sin embargo, cuando la interfaz se renderiza lo hace sin que el textarea actualice su valor, pues aún no existe.

La estrategia aquí no es del todo intuitiva. Se asigna la propiedad ref del textarea a una variable que en lugar de emplear useRef directamente emplea useCallback sin dependencias. Este useCallback se ejecuta una vez cuando el componente carga y memoriza el resultado para no tener que ejecutarse en cada reconciliación con el Virtual DOM. El callback que se le pasa como argumento recibe la referencia al nodo donde está siendo usado y se puede usar esa referencia para actualizar su tamaño y asignarla a una variable que sí emplea el useRef y que será la empleada en el efecto.

Modificaciones sobre el proyecto

El proyecto inicial se descartó por completo. Esto tuvo lugar por mi mala organización y mis problemas para configurar apropiadamente una comunicación en tiempo real basada en WebSockets. Además de esto, comprendí que abordar la autenticación a bajo nivel implicaba una cantidad de trabajo adicional con la que no contaba en modo alguno.

Este nuevo proyecto se comenzó a fecha de 13 de mayo y se presentó la idea de emplearlo en lugar del descrito en el anteproyecto el día 15 de mayo. Tras el visto bueno del tutor se ha continuado trabajando en él.

A partir de entonces se ha ido trabajando conjuntamente servidor y cliente (algo fácil dada la naturaleza de Next.js), funcionalidad a funcionalidad. Los cambios se integraban en la rama de desarrollo y se comprobaba su funcionamiento en su despliegue. Cuando una funcionalidad cumplía lo mínimo en ese escenario se enviaba a la rama principal y se desplegaba al entorno de producción. Desde el mismo día 13 de mayo el sitio web se encuentra en producción recibiendo mejoras sin comprometer la integridad de la base de datos.

Posibles mejoras

Son múltiples las mejoras que puede recibir un proyecto de estas características dado que imita a un proyecto real de una gran complejidad como es Twitter. Algunas de las más sencillas y realizables pueden ser:

- La inclusión de una vista con las personas que han dado me gusto a un tweet.
- Permitir cambiar la imagen de perfil en lugar de usar únicamente la del proveedor de identidades.
- Mejorar las opciones del fondo del perfil, que actualmente no advierte sobre las dimensiones recomendadas y no ofrece ninguna opción para ajustarlo.
- Permitir marcar la imagen de un tweet como contenido sensible.
- Permitir borrar los propios tweets.
- Permitir reportar tweets.
- Incorporar un portal para moderadores que tengan privilegios para borrar tweets y bloquear usuarios.

Conclusión

La realización de este proyecto ha reforzado mi apuesta por TypeScript y su ecosistema. El conjunto de herramientas propuesto por T3 ha resultado ser una acierto que me ha permitido desarrollar de manera segura la aplicación, centrándome en crear nuevas funcionalidades, sin preocuparme por la infraestructura y algunas cuestiones tediosas como la autenticación.

Me siento satisfecho porque a pesar del poco tiempo con el que he trabajado finalmente el resultado es vistoso y tiene el mejor rendimiento que se podía desear. Además, el proyecto queda abierto para agregar mejoras y he podido comprobar cómo la base de datos puede ir ajustándose a las necesidades del negocio en el tiempo, sin necesidad de tenerlo todo atado desde el principio.

También he aprendido nuevas tecnologías con este proyecto, llevaba mucho tiempo queriendo utilizar Prisma y tRPC y unir estas dos a mis conocimientos de React y Next.js me ha permitido por fin explorarlas. Estoy deseando seguir ahondando en ellas y exprimir todo su potencial.

Bibliografía

1. T3 | Introduction (10/06/2023)
2. Docs | Next.js (10/06/2023)
3. React (10/06/2023)
4. TypeScript (10/06/2023)
5. Stack Overflow Developer Survey 2022 (10/06/2023)
6. Tailwind CSS (10/06/2023)
7. tRPC Introduction (10/06/2023)
8. Builder (10/06/2023)
9. Zod (10/06/2023)
10. TanStack Query | React Query, Solid Query, Svelte Query, Vue Query (10/06/2023)
11. What is Prisma? (Overview) (11/06/2023)
12. Vercel (11/06/2023)
13. Serverless computing (11/06/2023)
14. PlanetScale (11/06/2023)
15. Cloudinary (11/06/2023)
16. React Icons (11/06/2023)
17. Discord Developer Portal (11/06/2023)
18. Stack Overflow nested anchors solution (11/06/2023)