|  |
| --- |
|  |

|  |
| --- |
| **HobbyVerse** |
| Un espacio para explorar, conectar y compartir intereses. |
|  |
| Red social basada en hobbies: Los usuarios se conectan según intereses compartidos, como cocina, lectura o deportes. |
|  |
| **PYE-CAMPANA04** |
| **[Seleccione la fecha]** |
|  |

**Características principales**

**1. Perfil personal centrado en hobbies**

* Los usuarios seleccionan sus hobbies/intereses al registrarse (fotografía, cocina, deportes, lectura, videojuegos, etc.).
* Información destacada: habilidades relacionadas, experiencia y logros.

**2. Grupos y comunidades**

* Comunidades específicas para cada hobby.
* Subgrupos por nivel (principiante, intermedio, avanzado) o especialidad (ej.: "Cocina vegana" dentro de "Cocina").

**3. Feed personalizado**

* Contenido generado por otros usuarios basado en los hobbies seleccionados.
* Publicaciones destacadas, como tutoriales, logros o preguntas.

**4. Eventos y actividades**

* Sección para buscar o crear eventos relacionados con hobbies (clases, talleres, reuniones locales o virtuales).

**5. Colaboraciones y retos**

* Herramienta para encontrar personas con intereses similares y colaborar en proyectos (ej.: escribir un libro, desarrollar un videojuego).
* Retos mensuales o semanales con premios simbólicos o reconocimiento (ej.: "Crea tu mejor plato en 30 minutos").

**6. Marketplace**

* Compra/venta/intercambio de artículos relacionados con hobbies (ej.: cámaras para fotógrafos, libros para lectores).

**7. Sistema de gamificación**

* Gana puntos o insignias por participación activa: crear contenido, asistir a eventos o ayudar a otros usuarios.
* Rankings por hobby para fomentar el compromiso.

**8. Contenido multimedia**

* Tutoriales, consejos y recursos gratuitos creados por expertos.
* Espacios para compartir imágenes, videos o streams en vivo relacionados con actividades.

**9. Modo de descubrimiento**

* Explora nuevos hobbies basados en tus intereses actuales.
* Test interactivo para sugerir hobbies.

**10. Funcionalidades avanzadas (premium)**

* Análisis de progreso en habilidades específicas.
* Acceso exclusivo a cursos, herramientas y expertos.

**Modelo de negocio**

1. **Freemium:**
   * Gratis con características básicas.
   * Suscripción premium para acceso avanzado a cursos, contenido exclusivo y herramientas de análisis.
2. **Publicidad:**
   * Anuncios específicos según los hobbies del usuario (ej.: equipos de senderismo para excursionistas).
3. **Marketplace:**
   * Comisión sobre transacciones realizadas dentro de la plataforma.
4. **Eventos patrocinados:**
   * Empresas pueden promover talleres o actividades para usuarios.

**Plan de desarrollo**

1. **MVP (3-6 meses):**
   * Registro de usuarios.
   * Perfil y selección de hobbies.
   * Feed básico y comunidades.
2. **Versión 1.0 (6-12 meses):**
   * Eventos, colaboraciones y sistema de gamificación.
   * Marketplace básico.
3. **Escalamiento (12+ meses):**
   * Integración de IA para sugerencias.
   * Monetización y funcionalidades premium.

plan técnico

### ****Fase 1: Planificación y Definición del Producto****

#### ****1.1. Requisitos funcionales****

* Registro y autenticación de usuarios.
* Selección y edición de hobbies.
* Creación de publicaciones con texto, imágenes y videos.
* Feed personalizado basado en hobbies.
* Grupos y comunidades.
* Sistema básico de notificaciones.

#### ****1.2. Requisitos no funcionales****

* Escalabilidad: Soportar millones de usuarios concurrentes.
* Seguridad: Autenticación segura, cifrado de datos sensibles.
* Disponibilidad: Tiempo de actividad de al menos el 99.9%.
* Rendimiento: Tiempo de carga inferior a 2 segundos para usuarios promedio.

### ****Fase 2: Arquitectura Técnica****

#### ****2.1. Estructura general****

**Frontend**:

* Framework: **React con Next.js** para renderizado del lado del servidor (SSR) y SEO.
* Estado: React Query o Zustand para manejo de datos asincrónicos.

**Backend**:

* Framework: **Node.js con NestJS** para escalabilidad y modularidad.
* API: **GraphQL** para flexibilidad en las consultas de datos.

**Base de datos**:

* **PostgreSQL** para datos estructurados (usuarios, grupos, publicaciones).
* **Redis** para caché y almacenamiento de sesiones.

**Almacenamiento de archivos**:

* **Amazon S3** para imágenes y videos.
* CDN: **Cloudflare** para distribuir contenido multimedia rápidamente.

**Autenticación y autorización**:

* **OAuth 2.0** con **Auth0** o integración con redes sociales (Google, Facebook).
* Middleware para verificar roles y permisos.

**Real-time**:

* **WebSockets** o **Pusher** para notificaciones y actualizaciones en tiempo real.

**Infraestructura**:

* **Docker** para contenerización.
* **Kubernetes** para orquestación de contenedores.
* **AWS (o GCP)** para servicios en la nube.

Aquí tienes un **plan técnico detallado** para la red social basada en hobbies:

### ****Fase 1: Planificación y Definición del Producto****

#### ****1.1. Requisitos funcionales****

* Registro y autenticación de usuarios.
* Selección y edición de hobbies.
* Creación de publicaciones con texto, imágenes y videos.
* Feed personalizado basado en hobbies.
* Grupos y comunidades.
* Sistema básico de notificaciones.

#### ****1.2. Requisitos no funcionales****

* Escalabilidad: Soportar millones de usuarios concurrentes.
* Seguridad: Autenticación segura, cifrado de datos sensibles.
* Disponibilidad: Tiempo de actividad de al menos el 99.9%.
* Rendimiento: Tiempo de carga inferior a 2 segundos para usuarios promedio.

### ****Fase 2: Arquitectura Técnica****

#### ****2.1. Estructura general****

**Frontend**:

* Framework: **React con Next.js** para renderizado del lado del servidor (SSR) y SEO.
* Estado: React Query o Zustand para manejo de datos asincrónicos.

**Backend**:

* Framework: **Node.js con NestJS** para escalabilidad y modularidad.
* API: **GraphQL** para flexibilidad en las consultas de datos.

**Base de datos**:

* **PostgreSQL** para datos estructurados (usuarios, grupos, publicaciones).
* **Redis** para caché y almacenamiento de sesiones.

**Almacenamiento de archivos**:

* **Amazon S3** para imágenes y videos.
* CDN: **Cloudflare** para distribuir contenido multimedia rápidamente.

**Autenticación y autorización**:

* **OAuth 2.0** con **Auth0** o integración con redes sociales (Google, Facebook).
* Middleware para verificar roles y permisos.

**Real-time**:

* **WebSockets** o **Pusher** para notificaciones y actualizaciones en tiempo real.

**Infraestructura**:

* **Docker** para contenerización.
* **Kubernetes** para orquestación de contenedores.
* **AWS (o GCP)** para servicios en la nube.

### ****Fase 3: Desarrollo del MVP****

#### ****3.1. Backend****

1. **Estructura inicial del proyecto**:
   * Configurar NestJS con módulos clave: AuthModule, UserModule, HobbyModule, PostModule.
   * Integrar Prisma para modelar la base de datos.
2. **Definir esquemas de base de datos (PostgreSQL):**

sql

Copiar código

-- Usuarios

CREATE TABLE users (

id SERIAL PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Hobbies

CREATE TABLE hobbies (

id SERIAL PRIMARY KEY,

name VARCHAR(100) UNIQUE NOT NULL

);

-- Usuarios-Hobbies relación

CREATE TABLE user\_hobbies (

user\_id INT REFERENCES users(id),

hobby\_id INT REFERENCES hobbies(id),

PRIMARY KEY (user\_id, hobby\_id)

);

-- Posts

CREATE TABLE posts (

id SERIAL PRIMARY KEY,

user\_id INT REFERENCES users(id),

content TEXT NOT NULL,

hobby\_id INT REFERENCES hobbies(id),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

1. **Endpoints GraphQL básicos:**
   * registerUser(email, password)
   * login(email, password)
   * getFeed(userId)
   * createPost(content, hobbyId)
   * getHobbies()

#### ****3.2. Frontend****

1. **Configuración inicial:**
   * Instalar Next.js con TypeScript.
   * Configurar Tailwind CSS para diseño.
2. **Páginas clave:**
   * **/register**: Registro y selección de hobbies.
   * **/login**: Inicio de sesión.
   * **/feed**: Feed personalizado.
   * **/hobbies**: Exploración de hobbies.
3. **Componentes reutilizables:**
   * **Card**: Para mostrar publicaciones.
   * **Navbar**: Con links a feed, grupos, etc.
   * **Modal**: Para crear publicaciones.
4. **Consumo de API (React Query):**
   * Fetch de hobbies: useQuery('hobbies', fetchHobbies)
   * Fetch del feed: useQuery('feed', fetchFeed)

### ****Fase 4: Infraestructura y Despliegue****

#### ****4.1. Configuración del entorno****

* Crear un entorno Docker:
  + Contenedor para el backend.
  + Contenedor para PostgreSQL.
  + Contenedor para Redis.

#### ****4.2. Despliegue inicial****

1. Configurar CI/CD con GitHub Actions:
   * Pruebas automáticas al hacer push.
   * Construcción y despliegue automático.
2. Desplegar en AWS:
   * EC2 para backend y frontend.
   * RDS para PostgreSQL.
   * S3 para almacenamiento de multimedia.

#### ****4.3. Escalabilidad****

* Implementar un balanceador de carga (AWS ELB o Nginx).
* Configurar autoescalado para contenedores (Kubernetes).

### ****Fase 5: Pruebas y Optimización****

1. **Pruebas funcionales:** Validar cada flujo de usuario (registro, creación de posts, feed).
2. **Pruebas de carga:** Simular 10k+ usuarios concurrentes con herramientas como **Apache JMeter**.
3. **Optimización del rendimiento:**
   * Caché de consultas comunes con Redis.
   * Preprocesamiento de imágenes y videos para reducir tamaños.

### ****Fase 6: Lanzamiento y Escalamiento****

1. **Beta privada:** Invitar a usuarios para pruebas iniciales y feedback.
2. **Marketing inicial:**
   * Colaboraciones con creadores de contenido relacionados con hobbies.
   * Publicidad en redes sociales.
3. **Escalamiento:**
   * Agregar nuevos módulos como eventos, marketplace y gamificación.