TRABAJO DE GRADO
Opción Seminario-Diplomado.

Red social para Deportista utilizando una Arquitectura basada en Microservicios y despliegue con Contenedores

Corporación Universitaria Remington.
Facultad de Ingeniería
Ingeniería en Sistemas

José Fernandy Saavedra Duran
Narciso Yunda Yunda
Tutor: Diego Fernando Marín Lozano

Opción de Trabajo de grado Seminario-Diplomado
2025

Agradecimientos

La Corporación Universitaria Remington ha estado con nosotros a lo largo de nuestra formación profesional, y se lo agradecemos mucho. A nuestras familias, que nos dieron su aliento para concluir este proceso académico, así como su apoyo constante y paciencia. A nuestro tutor, Diego Fernando Marín Lozano, por su dirección, sabiduría y compromiso a lo largo de la realización de este trabajo. Finalmente, a nosotros mismos, por la constancia, el empeño y el fervor con los que afrontamos el desafío de diseñar un proyecto con repercusiones tanto sociales como tecnológicas.

Tabla de Contenido

Resumen

Este documento sugiere crear una red para gente aficionada al deporte. La meta es unir a usuarios que quieran hacer ejercicio. El programa usa microservicios. Se hizo con FastAPI. El lenguaje base es Python. Docker se usa para los servicios separados en cajas.

El programa quiere ayudar a los deportistas a hablar más. busca formar grupos de fans o de gente muy buena. Esto permite compartir ideas. También de animo a todos por medio de un sitio web especial. Usamos Python por ser flexible y crecer bien. Usamos FastAPI para hacer las conexiones rápidas en cada servicio pequeño.

Docker ayuda a manejar los lugares de trabajo de forma sencilla. Esto da facilidad para moverlo. Tambien permite crear cada parte por separado y poner todo en marcha fácil.

Palabras clave

Python, FastAPI, Docker, Red Social, Servicios Pequeños, Puerta de Conexión, Cajas, Programa de Deporte, Estructura Separada

Pregunta orientadora de la búsqueda

¿Cómo puede una red social desarrollada con arquitectura de Microservicios en Python y desplegada con Docker mejorar la interacción y el seguimiento de actividades entre deportistas?

Sustentación teórica de la pregunta

Las plataformas de redes sociales, en la actualidad, cumplen una función fundamental en la interacción interpersonal, la difusión de información y el establecimiento de comunidades virtuales que reúnen a individuos con intereses comunes. No obstante, es importante destacar que muchas de estas plataformas no están orientadas específicamente hacia el ámbito deportivo, lo que limita su capacidad para abordar las necesidades particulares de los atletas, tales como la motivación, la posibilidad de monitorizar su avance en el entrenamiento físico y la obtención de asesoramiento de profesionales en el área de la salud. (Guamán, 2018)

Los atletas de nivel profesional y de alto rendimiento, requieren de un espacio en línea para documentar y exhibir su avance, intercambiar experiencias y comunicarse con otros usuarios que persiguen objetivos común esté tipo de página ayuda a los deportistas a tener un estado más saludable (Lobillo, 2016)**.** Aunque en la actualidad existan aplicaciones que permitan el monitoreo físico o el registro de entrenamientos se encuentran aplicaciones como "Strava" o "Nike Run Club", en las aplicaciones anteriormente mencionadas podemos evidenciar que no cumplen o no ofrecen una experiencia integral de red social centrada en la interacción, con la comunidad y la motivación colaborativa (Tenorio, 2025)

El crecimiento de las redes sociales cambió cómo la gente se relaciona y comparte datos. En el ámbito deportivo, estos medios ahora son súper importantes para inspirar, observar cómo van las cosas y formar grupos de cada deporte que se puede practicar.

Python se considera una herramienta de codificación actual, robusta y de gran adopción para crear programas de internet y servicios de conexión. Ahora bien, FastAPI proporciona una estructura moderna para construir esos mismos servicios de conexión veloces, usando anotaciones de tipo, genera documentación por sí mismo y funciona perfectamente con la programación asíncrona, por ende, es perfecto para diseños basados en pequeños servicios. Diversas vivencias en escuelas y empresas prueban lo efectivo que es FastAPI para construir de prisa sistemas que trabajan en distintas partes.

Docker da un método de cajones virtuales donde puedes meter programas y todo lo que necesitan, asegurando que donde se corra sea igual y funcione bien, perfecto para ponerlos en servidores o en la nube. La idea de Microservicios, juntada con Python, FastAPI y Docker, ayuda a crear programas nuevos rápido, porque deja poner cada pedazo por separado, permite crecer solo lo que se necesita y asegura que el software ande firme y se pueda replicar en distintos lugares. (Alarcón, 2025)

Problema

Los canales y páginas web dirigidos a los aficionados del deporte tienen hoy en día varios inconvenientes que impactan la forma en que los deportistas documentan sus avances, comparten información y se comunican con ellos. Debido a que el contenido suele estar desorganizado, esparcido o bloqueado en programas privados, algunas de estas aplicaciones no facilitan a los competidores el acceso a información importante. Esto hace difícil la revisión y obstaculiza la creación de equipos cohesivos que trabajen juntos. Además, la falta de instrumentos personalizados para monitorear ejercicios reduce las probabilidades de que las personas revisen sus marcas, establezcan metas y reciban comentarios útiles acerca de su actividad física.(García, 2023).

Muchas de las aplicaciones sociales para deportes se basan en diseños unificados, donde todas las operaciones corren dentro de un único componente. Esto complica mucho el aumentar la capacidad del sistema, puesto que al añadir funciones nuevas o modificar las existentes se arriesga a afectar a otros componentes y causar problemas amplios o que el sistema se venga abajo del todo (OMS., 2024). Esta organización centralizada una considerable cantidad de recursos informáticos y repercute en el rendimiento, provocando que las plataformas sean más lentas, menos confiables y tengan una disponibilidad reducida para los atletas (Tapia, 2020).

Metodología

El desarrollo del proyecto “Red social para deportistas” se realizó mediante el cumplimiento de las actividades en función de los objetivos propuestos, con base en ello se trabajó en 3 fases principales:

Fase 1: Exploración y Diseño Arquitectónico

En esta fase se realizó diferentes tareas relacionadas con la Exploración del estado del arte, investigación sobre arquitectura de Microservicios y la definición de la estructura técnica del sistema. Se estudio la metodología ágil Scrumban y el uso de tableros Kanban para la gestión del proyecto. (Salvay, 2017).

La información requerida para la plataforma se obtuvo por medio de entornos virtuales como scholar.google.com, books.google.com los cuales ofrecen documentos de proyecto de grado, artículos, investigaciones sobre Microservicios y arquitectura distribuidas, entre otros.

Fase 2: Desarrollo con Arquitectura de Microservicios

Esta fase involucra todas las actividades del desarrollo de la aplicación software las cuales se dividieron en seis componentes principales organizados como Microservicios independientes:

Microservicios de Autenticación: Este servicio encarga del registro de nuevos usuarios, inicios de sesión, validación de credenciales y gestión de tokens JWT.

Microservicios de Gestión de Datos: Gestiona la información principal de la plataforma incluyendo perfiles de deportistas, publicaciones, eventos deportivos y relaciones entre usuarios.

Microservicios de Notificaciones: Coordina él envió y gestión de notificaciones a los usuarios sobre actividades relevantes en la plataforma.

Microservicios de Analytics: Procesa y analiza datos de actividades deportivas, generando métricas y reportes de rendimiento.

API Gateway: Componente central que enruta todas las solicitudes del frontend hacia los Microservicios correspondientes.

Frontend: Interfaz de usuario desarrollada con Flask que consume los servicios a través del API Gateway

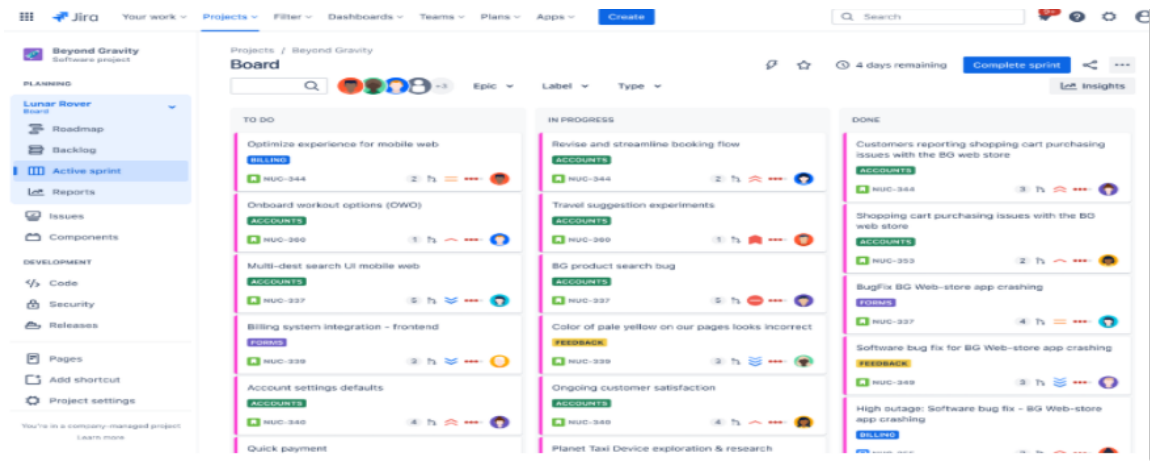
Fase 3: validación e Integración

Esta fase involucró todas las reuniones, revisiones, correcciones y validaciones realizadas durante el progreso del desarrollo, con especial enfoque en la integración entre Microservicios y la validación del flujo completo de comunicación.

Modulo Registro e Inicio de Sesión: Este módulo consiste en la creación de un nuevo usuario, el correcto ingreso a la base de datos, e Inicio de Sesión y su respectiva validación.

Modulo Teoría: Este módulo involucra toda la información encontrada acerca de la Metodología ágil Scrumban, la cual se encuentra distribuida en ocho temas principales: Definición Metodología ágiles, Definición Scrumban, componentes, principios, roles, ventajas cuando implementar, puntos claves.

Modulo Jira: Este módulo ofrece una vista al usuario de todas las actividades pertenecientes a su proyecto, consta de 5 columnas: Backlog, Ready, Doing, Review, Done


Jira
Your work
Projects
Filter
Dashboards
Teams
Plans
Apps
Create
Search
Beyond Gravity
Software project
PLANNING
Lunar Rover
Board
Roadmap
Backlog
Active sprint
Reports
Issues
Components
DEVELOPMENT
Code
Security
Releases
Pages
Add shortcut
Project settings
You're in a company-managed project
Learn more
Projects / Beyond Gravity
Board
4 days remaining
Complete sprint
Insights
Epic
Label
Type
TO DO
Optimize experience for mobile web
BILLING
NUC-344
Onboard workout options (OWO)
ACCOUNTS
NUC-360
Multi-dest search UI mobile web
ACCOUNTS
NUC-337
Billing system integration - frontend
FORMS
NUC-339
Account settings defaults
ACCOUNTS
NUC-340
Quick payment
IN PROGRESS
Revise and streamline booking flow
ACCOUNTS
NUC-344
Travel suggestion experiments
ACCOUNTS
NUC-360
BG product search bug
ACCOUNTS
NUC-337
Color of pale yellow on our pages looks incorrect
FEEDBACK
NUC-339
Ongoing customer satisfaction
ACCOUNTS
NUC-340
Planet Taxi Device exploration & research
DONE
Customers reporting shopping cart purchasing issues with the BG web store
ACCOUNTS
NUC-344
Shopping cart purchasing issues with the BG web store
ACCOUNTS
NUC-353
BugFix BG Web-store app crashing
FORMS
NUC-337
Software bug fix for BG Web-store app crashing
FEEDBACK
NUC-349
High outage: Software bug fix - BG Web-store app crashing
BILLING

Desarrollo

Para solventar la problemática expuesta anterior mente se propone un sistema en Microservicios, tales como autentificación, gestión de perfiles, gestión de contenidos o información digital, entrenamientos y rendimientos esto se desarrollan a través de Apis e interfaz modulares, cada uno actuado de manera autónoma sin afectar entre sí.

Además, estarán aislados mediante contenedores Docker lo cual nos permite que la red social incluya todas las herramientas necesarias para ejecutarse de manera fiable y segura en cualquier equipo. ((Sanchez, 2022),contenedores Docker), esta estructura facilita detectar fallas e intervenir en el Microservicios afectado sin que se vean comprometidos los demás servicios, integrando FastAPI con Python para disminuir duplicidad de código garantizando tiempos de respuesta óptimos (Cruz, S.F).

Finalmente, los servicios se pueden incrementar según sea la necesidad sin afectar la estructura general, dando como resultado una red social modular, ágil, estable y sostenible.

Comunicación y planificación.

Con la problemática claramente definida de acuerdo con análisis realizado sobre las redes sociales para deportistas mencionadas anteriormente, el siguiente paso es definir las iteraciones y los requisitos que se desarrollaran en cada una como se muestra en la siguiente tabla.

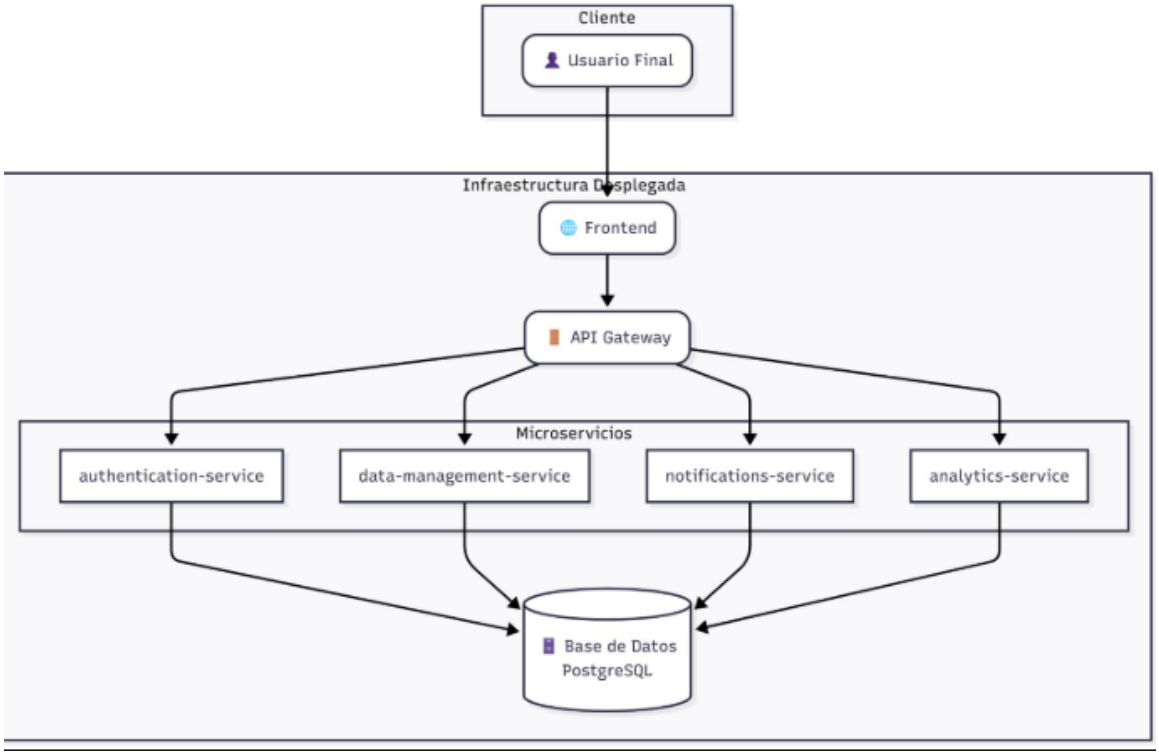
Tabla 1 Tabla de Iteraciones

Incremento	Desarrollo	Entregable principal	Responsable(s)
Incremento 1	Microservicios de autenticación	Código autenticación y pruebas	José Saavedra
Incremento 2	API funcionando	Gateway funcional con rutas y pruebas	Narciso Yunda
Incremento 3	Interfaz de usuario conectada a API Gateway	Frontend operativo integrado y pruebas	José Saavedra
Incremento 4	Orquestación y documentación automática MkDocs	URL local con Documentación de página web	José Saavedra, Narciso Yunda
Incremento 5	Despliegue con Contenedores	Configuración Docker Compose para el despliegue	José Saavedra, Narciso Yunda

Diagrama de arquitectura de proyecto.

En el siguiente diagrama se expone la arquitectura propuesta para la red social orientada a deportistas. Este esquema permite visualizar la forma en que se organizan los componentes del sistema, así como la interacción entre los diferentes microservicios, el API Gateway y las bases de datos que soportan la plataforma (ver lustración 1)

Ilustración 1 Diagrama de arquitectura

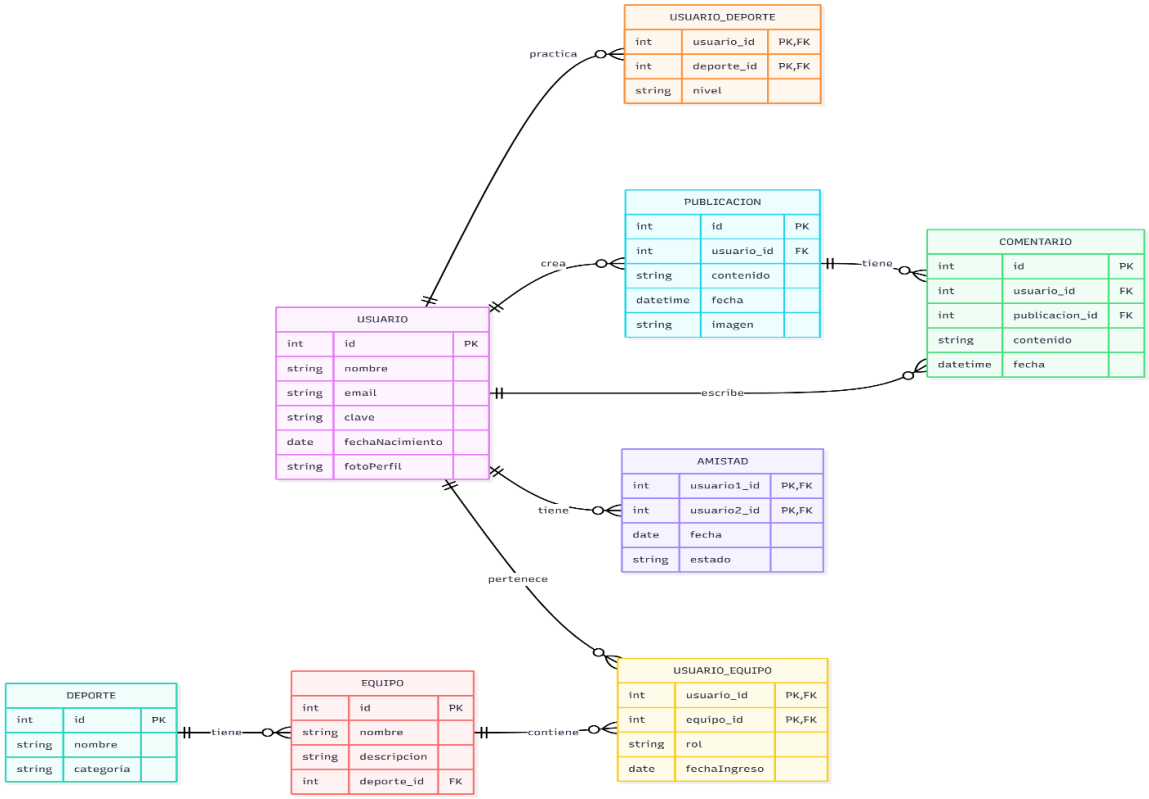


fuente: José Saavedra, Narciso Yunda

Diagrama de bases de datos.

En el siguiente apartado se muestra el diagrama correspondiente al modelo de base de datos diseñado para la plataforma. Este esquema permite comprender la estructura lógica utilizada para almacenar la información de usuarios, actividades, publicaciones y demás elementos fundamentales del sistema, así como la relaciones entre las distintas entidades definidas (ver ilustración 3)

Ilustración 2 Diagrama base de datos



fuente: José Saavedra, Narciso Yunda

Incremento 1: Autenticación

En esta primera etapa se implementa el módulo de autentificación, el cual constituye la base de acceso seguro para los usuarios de la red social. Para ello se integraron tecnologías como FastAPI, MongoDB y JWT, permitiendo gestionar de manera eficiente el registro, identificación y validación de cada usuario dentro del sistema. Este incremento garantiza que únicamente quienes hayan creado una cuenta puedan interactuar con la plataforma, fortaleciendo así la seguridad y la integridad de la información.

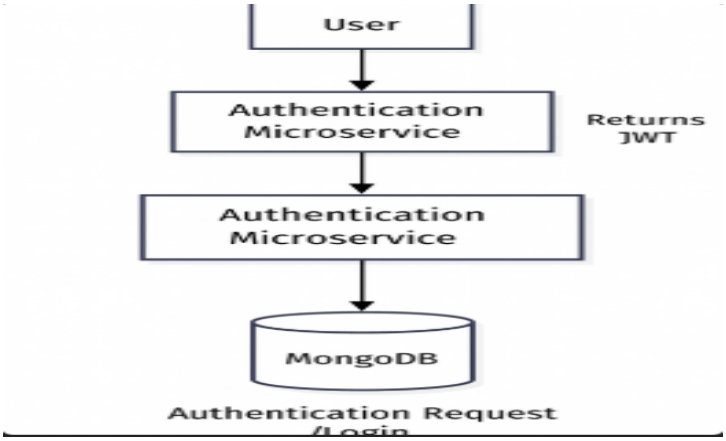
Desarrollo realizado

- ✓ **Registro de usuarios:** creación de nuevos perfiles mediante él envió de datos básicos hacia el microservicio de autenticación.
- ✓ **Inicio de sesión:** verificación de credenciales previamente almacenadas en la base de datos
- ✓ **Generación de JWT:** emisión automática de un token seguro para usuarios autenticados, utilizado en peticiones posteriores.
- ✓ **Validación de credenciales:** comprobación del token enviado por el cliente en cada solicitud que requiera autorización.

Diagrama de autenticación:

En la siguiente ilustración se presenta el flujo general de funcionamiento del proceso de autentificación implementando en este incremento (ver ilustración 3), donde se observa la interacción entre el usuario, el microservicio de autenticación y la base de datos en cargada de almacenar las credenciales.

Ilustración 3 Diagrama de autenticación



fuente: José Saavedra, Narciso Yunda

En el siguiente código muestra la forma de crear autentificación se ve la creación de una función de atención a la petición POST del URL "/api/registro/...", se hace uso del decorador @app. route () para indicar el método GET Y POST.

```
@app.route("/registro", methods=["GET", "POST"])
def registro():
    """Registrar un nuevo usuario."""
    if request.method == "POST":
        user_data = {
            "username": request.form.get("username"),
            "email": request.form.get("email"),
            "password": request.form.get("password"),
        }
```

Incremento 2: Microservicios de Eventos

En este incremento se implementan los microservicios responsables de administrar la información principal que los usuarios publican e intercambian dentro de la red social.

Para ello se empleó FastAPI junto con Python, permitiendo desarrollar servicios independientes, ligeros y de alto rendimiento.

En esta etapa, los deportistas pueden crear y consultar publicaciones, así como gestionar eventos deportivos, lo cual constituye la base de la interacción social dentro de la plataforma

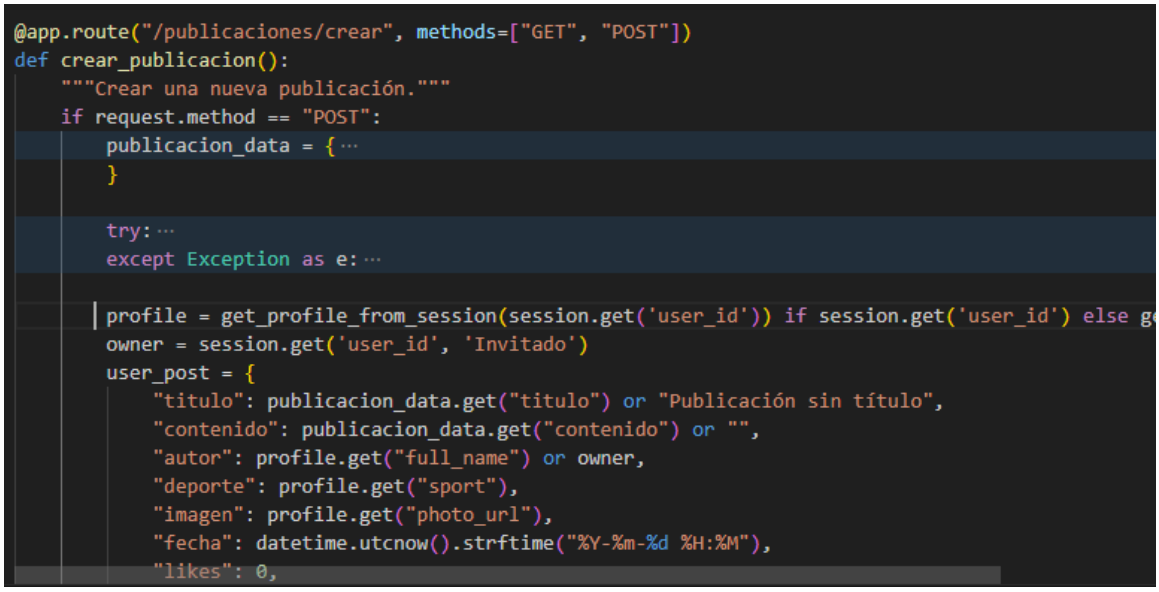
Desarrollos realizados

- ✓ **Microservicio de eventos:** encargado de registrar, actualizar y consultar actividades deportivas, tales como competencias, entrenamientos grupales o encuentros programados por los usuarios.
- ✓ **Microservicio de publicaciones:** permite a los deportistas compartir contenido, registrar experiencias, comentar actividades y visualizar el flujo de información generado por la comunidad.

En el siguiente código se ve la creación de Microservicios de eventos una función de atención a la petición POST del URL "/api/eventos/crear...", se hace uso del decorador @app. route () para indicar el método POST, GET y el URL.

```python
@app.route("/eventos/crear", methods=["GET", "POST"])
def crear_evento():
    """Crear un nuevo evento."""
    if request.method == "POST":
        evento_data = {
            "nombre": request.form.get("nombre"),
            "descripcion": request.form.get("descripcion"),
            "fecha": request.form.get("fecha"),
            "lugar": request.form.get("lugar") or "Por definir",
```

En el siguiente código se ve la creación de Microservicios de publicaciones una función de atención a la petición POST del URL "/api/publicaciones/crear...", se hace uso del decorador @app.post () para indicar el método POST, GET y el URL



```
@app.route("/publicaciones/crear", methods=["GET", "POST"])
def crear_publicacion():
    """Crear una nueva publicación."""
    if request.method == "POST":
        publicacion_data = {···
        }

        try:···
        except Exception as e:···

        profile = get_profile_from_session(session.get('user_id')) if session.get('user_id') else g
        owner = session.get('user_id', 'Invitado')
        user_post = {
            "titulo": publicacion_data.get("titulo") or "Publicación sin título",
            "contenido": publicacion_data.get("contenido") or "",
            "autor": profile.get("full_name") or owner,
            "deporte": profile.get("sport"),
            "imagen": profile.get("photo_url"),
            "fecha": datetime.utcnow().strftime("%Y-%m-%d %H:%M"),
            "likes": 0,
```

Incremento 3: API Gateway

En este incremento se implementa el API Gateway, un componente central que actúa como punto unificado de entrada para todas las solicitudes que llegan desde el frontend hacia los distintos microservicios. Esta capa posibilita que se administre el enrutamiento de solicitudes de manera organizada, se implementen políticas de seguridad y se garantice una comunicación eficaz entre los elementos distribuidos de la arquitectura.

Para asegurar un acceso seguro, se incluye la validación de tokens JWT en el API Gateway. Estos tokens son empleados para constatar quién es el usuario antes de permitirle que interactúe con los servicios internos.

En el siguiente código se ve la creación de API Gateway se hace uso del decorador @router.get () para indicar el método service, path y el URL.

```python
@router.get("/{service_name}/{path:path}")
async def forward_get(service_name: str, path: str, request: Request):
    if service_name not in SERVICES:
        raise HTTPException(status_code=404, detail=f"Service '{service_name}' not found.")

    service_url = f"{SERVICES[service_name]}/{path}"
```

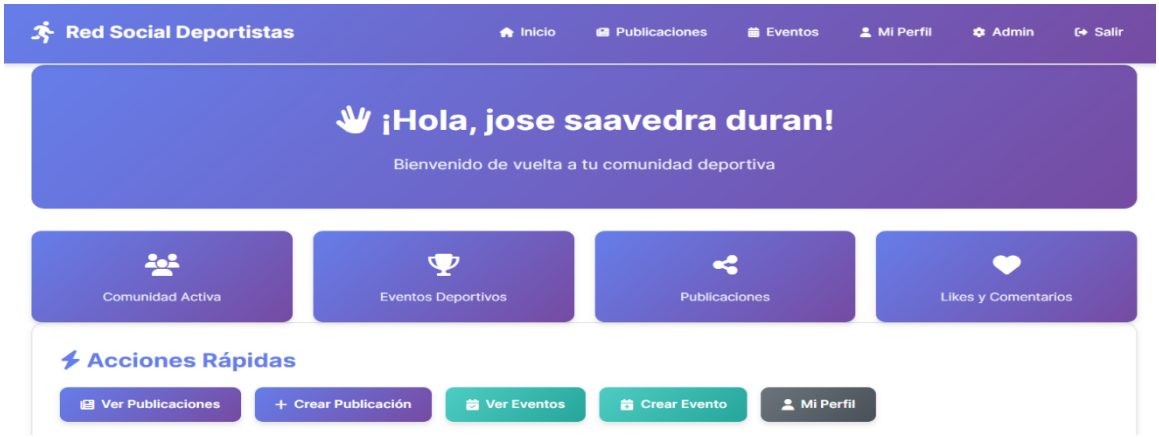
Incremento 4: Frontend

Para esta fase se implementa el Frontend de la plataforma, encargado de proporcionar la interfaz visual mediante la cual los usuarios interactúan con la red social. Se emplea el framework Flask, una herramienta ligera y flexible que facilita la creación de visitas dinámicas y el consumo eficiente de los microservicios expuestos a través dl API Gateway.

El frontend permite a los deportistas realizar acciones esenciales como iniciar sesión, registrarse, visualizar publicaciones, consultar eventos deportivos y navegar entre las distintas funcionalidades de la aplicación

A continuación, se presenta la pantalla de inicio (ver ilustración 4)

Ilustración 4 pantalla de inicio



fuente: José Saavedra, Narciso Yunda

Incremento 5: Despliegue y documentación

En esta etapa se lleva a cabo el proceso de despliegue de la plataforma mediante el uso de contenedores administrados con Docker, herramienta que permite ejecutar cada microservicio de forma aislada y reproducible. El objetivo principal de este incremento es asegurar que la arquitectura distribuida funcione de manera consistente en cualquier entorno, facilitando tanto la ejecución local como su futura implementación en servidores o infraestructura en la nube. La Contenerización se aplica a todos los componentes del sistema incluyendo los microservicios, el API Gateway y el frontend garantizando que cada uno cuente con sus propias dependencias y configuraciones esto permite que la plataforma sea más estable y sencilla de mantener.

Desarrollos realizados

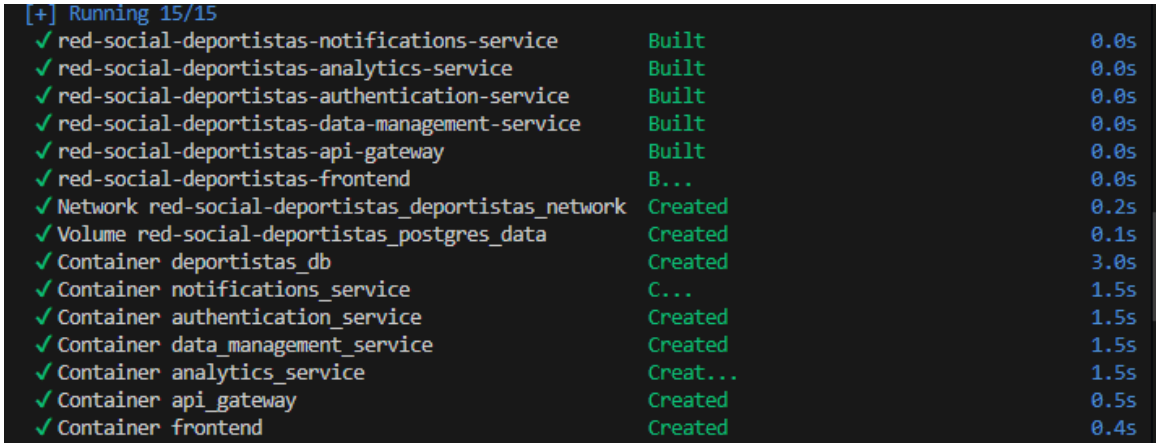
- ✓ **Contenerización con Docker:** creación de imagenes y definición de contenedores para cada microservicio, el API Gateway y el frontend, asegurando un entorno de ejecución consistente y modular
- ✓ **Documentación del despliegue:** elaboración de la guía técnica del sistema, con detalles sobre la configuración, ejecución, y mantenimiento de los contenedores que conforman la plataforma.

Código Fuente de contenedores.

```
authentication-service:
  build:
    context: ./services/authentication
    dockerfile: Dockerfile
  container_name: authentication_service
  ports:
    - "8001:8001"
  environment:
    - DB_HOST=db
    - DB_PORT=5432
    - DB_NAME=deportistas_db
    - DB_USER=deportistas_user
    - DB_PASSWORD=deportistas_pass
  depends_on:
    - db
```

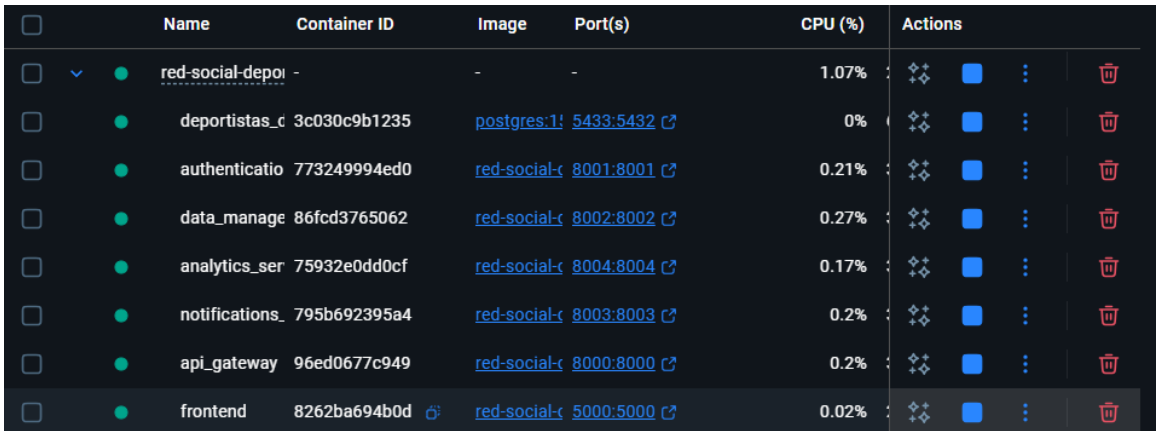
En la siguiente captura de pantalla vemos la salida del comando: Docker Compose up --build, donde se observa la creación de las imágenes de cada uno de los contenedores que componen esta aplicación, y el inicio de cada contenedor de acuerdo con lo definido en el archivo Docker-compose.yml

Ilustración 5 Creación de contenedores



fuente: José Saavedra, Narciso Yunda

Ilustración 6 Lista de Contenedores



fuente: José Saavedra, Narciso Yunda

Documentación.

La documentación para el proyecto se crea utilizando MkDocs, una herramienta que facilita el proceso de redacción de la documentación a través de su estructuración y publicación. Los documentos calves para esta información incluyen el sistema principal, la arquitectura utilizada, los microservicios creados, así como la guía de ejecución y mantenimiento son los documentos fundamentales para este registro. (ver ilustración 7)

Esta sección brindará información detallada sobre la configuración del entorno, cómo utilizar los endpoint de la API, los parámetros requeridos para cada solicitud, y las instrucciones para desplegar el sistema utilizando contenedores Docker. Esta guía será un recurso fundamental para permitir la comprensión, reproducción y revolución del proyecto.

Código de documentación con MkDocs

En el fragmento de código a continuación, se presenta la configuración aplicada para organizar documentación del proyecto mediante MkDocs, en la cual se especifican las secciones primordiales, los tópicos y los documentos relacionados con el contenido técnico.

mkdocs.yml

```yaml
site_name: Documentación - Red Social Deportistas
site_description: 'Documentación técnica del proyecto de red social para deportistas.'
site_author: 'jose saavedra y yudai narciso'
site_url: https://Josesaavedr.github.io/Red-social-Deportistas
repo_url: https://github.com/Josesaavedr/Red-social-Deportistas.git
repo_name: Josesaavedr/Red-social-Deportistas
edit_uri: edit/main/docs/
```

Ilustración 7 Documentación de código

fuente: José Saavedra, Narciso Yunda

Conclusiones

La red social para Deportistas utiliza una solución tecnológica innovadora que fusiona las funciones sociales, de registro deportivo y de organizaciones de eventos en una Plataforma diseñados especialmente para atletas y entrenadores. Se trata de una aplicación que ha sido diseñada con Docker lo cual muestra lo posible de usar la tecnología moderna para la interrelación y motivación deportiva.

A través de su diseño fundamentado en Microservicios, el sistema alcanza un nivel impresionante de escalabilidad, disponibilidad y flexibilidad, Esto permite que cada componente opere de manera independiente y eficiente. Además, La configuración para múltiples bases de datos, como PostgreSQL, MongoDB y Redis garantiza un manejo optimo de los diferentes tipos de datos, desde la información estructurada de usuarios y eventos hasta las actividades deportivas.

Este proyecto en general no solo hace más fácil la interacción entre deportistas, si no que tambien promueven la motivación, el monitoreo de desempeño y la participación en comunidades activas. Por tanto, la Red Social para Deportistas se establecen como una plataforma actual, sólida y flexible, que puede satisfacer las demandas de un ecosistema deportivo en permanente expansión.

Referencias

Alarcón. (2025). Arquitectura de microservicios basada en contenedores para despliegue ágil de aplicaciones IoT en la nube. https://dialnet.unirioja.es/servlet/articulo?codigo=10056918

Guamaán, C. E (2018). ESTRATEGIA DE APRENDIZAJE DEL PROTOTIPO DE RED SOCIAL TSOUL MEDICO–DEPORTISTA: PROTOTYPE LEARNING STRATEGY OF TSOUL SOCIAL NETWORKING PHYSICIAN-ATHLETE. *Opuntia Brava, 9(2), 112–119*. https://opuntiabrava.ult.edu.cu/index.php/opuntiabrava/article/view/153

Lobillo. (2016). El papel de las redes sociales en las estrategias de expansión de mercado en los clubes de fútbol españoles. El uso personalizado de Twitter en lengua árabe del Real Madrid CF y FC Barcelona. https://dialnet.unirioja.es/servlet/articulo?codigo=5827708

Salvay. (2017). *Kanban y Scrumban orientados a Proyectos de Tecnología de la Información* .