



| | | |
|---|---|--|
| PROCESO | | |
| MÓDULO | | |
| NOMBRE DEL FORMATO | | |
| INFORME DE ACTIVIDADES SEMANALES | | |
| CLASIFICACIÓN DE LA INFORMACIÓN | | |
| Pública <input type="checkbox"/> | Pública Clasificada <input checked="" type="checkbox"/> | Pública Reservada <input type="checkbox"/> |

PROYECTO FONDO EMPRENDER SENA

SEMANA DEL 24 DE FEBRERO AL 2 DE MARZO DE 2026

Presentado por:

NELSON RICARDO GÓMEZ MARTÍNEZ

OFICINA DE SISTEMAS DIRECCIÓN GENERAL

SENA

Bogotá DC

2026



ENTREGABLES:

→ Tarea #86: Crear el stack de desarrollo

Creamos un Stack basado en Docker donde se contemplaron todas las recomendaciones hechas por el Director Nacional. El ecosistema ha sido diseñado bajo el patrón de Monolito Desacoplado con integración de Microservicios, una arquitectura empresarial estructurada para garantizar tolerancia a fallos, escalabilidad horizontal y un aislamiento estricto de responsabilidades.

En lugar de unificar la plataforma en un único bloque de código tradicional propenso a cuellos de botella de rendimiento, la infraestructura se construyó sobre un entorno 100% contenerizado y orquestado. Esta estrategia separa físicamente la capa de presentación, las transacciones core y el procesamiento cognitivo avanzado, comunicándolos a través de una red virtual privada y de alta velocidad.

EVIDENCIAS DEL ENTREGABLE

A continuación, se puede observar las pruebas referentes a los entregables:

[Crear el stack de desarrollo - Task #86 - Fondo Emprender](#)

The screenshot shows a Jira task page for '#86 Crear el stack de desarrollo'. The task is assigned to Nelson Ricardo Gómez Martínez and Diana Carolina Montoya Valencia. It includes four attachments: 'docker-compose.yml' (2.4 KB), 'image.png' (255.6 KB), 'Descripción del Stack.docx' (20.3 KB), and '20200303_FE_QA_TSK_86.docx' (313.9 KB). A comment from Diana Carolina Montoya Valencia dated 03 Mar 2024 11:01 states: 'Se validó el cierre de la tarea y documentación completa de prueba y stack de desarrollo adjunto'. A comment from Nelson Ricardo Gómez Martínez dated 26 Feb 2024 15:00, edited on 26 Feb 2024 16:01, describes the Docker-based stack architecture. Below the comments is a terminal window showing Docker commands and system resource usage.



→ Tarea # 87: Especificar proceso de instalación local del Stack

La tarea consistió en documentar de manera detallada el proceso de instalación y verificación de un entorno de desarrollo basado en Docker, comenzando con la clonación del repositorio mediante git clone, la descripción de la estructura de carpetas y la ubicación del archivo docker-compose.yml. Se registraron los comandos necesarios para levantar el stack completo con docker compose up -d, el tiempo estimado de instalación inicial y la verificación de contenedores en ejecución con docker ps, además de los criterios para confirmar que la instalación fue exitosa. Finalmente, se incluyeron pruebas de funcionamiento para cada componente: frontend en Next.js accesible vía Nginx en <http://localhost:8080/>, backend en Laravel 12 con el endpoint /api/status, agente de IA en FastAPI con sus endpoints y Swagger UI, y validaciones de la capa de datos en PostgreSQL y Redis por consola

EVIDENCIAS DEL ENTREGABLE

A continuación, se puede observar las pruebas referentes a los entregables:

[Especificar proceso de instalación local - Task #87 - Fondo Emprender](#)

The screenshot shows a Jira task page for 'Especificar proceso de instalación local' (Task #87) under the 'Fondo Emprender' project. The task is marked as 'CLOSED' and was created by Dana Pilar Solarte on 24 Feb 2024 at 17:54. It is assigned to Nelson Ricardo Gómez Hernández. The task description includes a list of requirements and a table of attachments.

Descripción: Escribir paso a paso para clonar el repositorio y preparar el entorno de desarrollo con el stack Docker existente.

- Documentar comando git clone [url del repositorio]
- Documentar estructura de carpetas clonadas
- Documentar ubicación del archivo docker-compose.yml
- Documentar comando docker compose up -d para levantar stack completo
- Documentar tiempo estimado de primera instalación
- Documentar verificación de contenedores con comando docker ps
- Documentar cómo saber si la instalación fue exitosa
- Guardar en /docs/Installation.md

Para probar que quedó bien instalado:

- Frontend (Next.js - Interfaz de Usuario)**
 - Via Proxy Nginx (Punto de entrada web) http://localhost:8080/
 - Resultado esperado: Visualización de la pantalla de inicio por defecto de Next.js.
- Backend (Laravel 12 - Lógica de Negocio)**
 - Endpoint de Estado: http://localhost:8080/api/status
 - Resultado esperado: Un objeto JSON devolviendo: {"mensaje": "¡Salud! Backend!", "status": "online", "framework": "L2.x.x"}. Esto confirma que Nginx enrutó hacia PHP-FPM correctamente.
- Agente IA (FastAPI - FEMBOT)**
 - Endpoint Base (JSON): http://localhost:8080/api/ai/
 - Resultado esperado: Un objeto JSON devolviendo: {"status": "online", "agent": "FEMBOT"}.
 - Interfaz Interactiva (Swagger UI): http://localhost:8080/api/ai/docs
 - Resultado esperado: La consola visual de FastAPI donde podrá ejecutarse pruebas directas sobre los endpoints de inteligencia artificial que construyamos.
- Capa de Datos (Acceso por Consola)** Dado que PostgreSQL y Redis operan en la capa de persistencia, no exponen URLs HTTP por seguridad arquitectónica. Su validación se realiza estrictamente por terminal.
 - PostgreSQL (SCHE & AB): docker compose exec db psql -U sem_admin -d s3gite_db -i
 - Redis (Cache y Cola): docker compose exec cache redis-cli ping (debe responder PONG).

11 Adjuntamientos

| Attachment | Size |
|--|----------|
| # 3_Elfin_Oswaldi_Labator_Terradellas_Instalación entorno de desarrollo.docx | 183.0 KB |
| # 4_Diana_Carolina_Montes_Valencia_Diario de pruebas de stack de desarrollo backend.md | 214.0 KB |
| # 1_Ana_Jincha_Antony_Silveira_Saiz.docx | 242.0 KB |
| # 3_Elfin_Oswaldi_Labator_Terradellas_Paso a paso instalación entorno de desarrollo.md | 341.0 KB |
| # 20240224_FE_QA_Task_87.docx | 425.4 KB |
| # 6_Nelson_Ricardo_Gomez_Hernandez_Evidencias_Instalación contenedores de Fondo Emprender.docx | 609.0 KB |



→ Tarea #89: Preparar sesión de transferencia de conocimiento al equipo

La tarea #89 consistió en preparar y ejecutar una sesión de transferencia de conocimiento al equipo de desarrollo, con el objetivo de socializar la configuración y arquitectura del stack y estandarizar prácticas de trabajo. Para ello se organizó una agenda con bloques definidos: presentación de arquitectura (15 min), demo de instalación en una máquina limpia o VM (20 min), ejercicio práctico guiado (60 min) y espacio de preguntas y respuestas (25 min). Se elaboró un checklist de verificación para cada participante, se entregó documentación previa para lectura, se agendó la sesión con todo el equipo y se grabó la jornada para referencia futura. Finalmente, se recolectaron dudas y problemas surgidos durante el ejercicio práctico, asegurando que el equipo contara con insumos claros para continuar con el desarrollo bajo estándares comunes.

EVIDENCIAS DEL ENTREGABLE

A continuación, se puede observar las pruebas referentes a los entregables:

The screenshot shows a Jira task page for '#89 Preparar sesión de transferencia al equipo'. The task is in the 'CLOSED' state. The description reads: 'This task belongs to #85 Como equipo de desarrollo, queremos conocer la configuración y arquitectura del stack, para poder estandarizar el desarrollo del proyecto'. The taskboard shows a checklist of tasks:

- Entregar y socializar el documento de estándares de desarrollo del proyecto FE
- Preparar agenda: presentación arquitectura (15 min), demo instalación (20 min), ejercicio práctico (60 min), Q&A (15 min)
- Preparar máquina limpia o VM para demo de instalación desde cero
- Preparar checklist de verificación para cada participante
- Agendar sesión con todo el equipo de desarrollo
- Entregar documentación previa para lectura antes de sesión
- Realizar sesión guiada
- Recolectar dudas y problemas encontrados durante ejercicio práctico
- Actualizar documentación con problemas encontrados en sesión

There is one attachment: '20240303_FE_QA_TK_89.docx' (204.4 KB). The comments section shows one comment from Diana Carolina Montoya Valencia on 03 Mar 2024 10:41: 'Se validó y asistió a la sesión de transferencia de conocimiento al equipo de trabajo'.



Trasferencia de conocimiento Stack de desarrollo

Servicio Nacional de Aprendizaje

Unirse 10

viernes, 27 febrero 2026 3:55 p.m. - 5:38 p.m. Más Descargar

Asistencia Participación

11
Asistieron

3:55 p.m. - 5:38 p.m.
Hora de inicio y finalización

1h 43m 2s
Duración de la reunión

1h 22m 53s
Tiempo medio de asistencia

Participantes

| Nombre | Primera unión | Última salida | Duración de L. | Rol | Compromiso |
|--|---------------|---------------|----------------|-------------|---------------------------------|
| Nelson Ricardo Gomez Martinez ngomez@sena.edu.co | 3:57 p.m. | 5:38 p.m. | 1h 40m 50s | Organizador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| Alexander Rodriguez Ararat a.rodriguez@sena.edu.co | 3:55 p.m. | 5:38 p.m. | 1h 42m 59s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| AP Andres Felipe Sanchez Perez afsanchezp@sena.edu.co | 3:55 p.m. | 5:38 p.m. | 1h 39m 1s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| YO Yulian Andres Lopez Osorio yalopez@sena.edu.co | 3:58 p.m. | 5:38 p.m. | 1h 39m 52s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| DC Diego Arnoby Alcibar Castrillon dalcibar@sena.edu.co | 3:59 p.m. | 5:37 p.m. | 1h 38m 19s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| DO Daniel Mendez Ospina dmendez@sena.edu.co | 4:00 p.m. | 5:38 p.m. | 1h 37m 55s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| AL Alex Fernando Pinchao Leiton apinchao@sena.edu.co | 4:00 p.m. | 5:38 p.m. | 1h 37m 35s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| RN read.ai meeting ... (No comprobado) | 4:00 p.m. | 4:01 p.m. | 15s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| Santiago Trujillo Restrepo strujillor@sena.edu.co | 4:02 p.m. | 5:38 p.m. | 1h 35m 13s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| Efraín Oswaldo Lobaton Tierra... elobatont@sena.edu.co | 4:05 p.m. | 5:38 p.m. | 1h 32m 58s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |
| Diana Carolina Montua Valencia dmontua@sena.edu.co | 5:11 p.m. | 5:38 p.m. | 26m 41s | Moderador | 👇 - 🗨️ - 📄 - 👍 - ❤️ - 🙌 - 😊 - 😬 |



→ Tarea #88: Especificar base de datos a utilizar, proceso creación, estructura general de tablas, estrategia de migraciones

La tarea se centró en definir y desplegar la arquitectura de persistencia del proyecto, especificando los motores de base de datos y la estrategia de migraciones. Se implementó un esquema híbrido con PostgreSQL 16 como motor relacional principal y Redis 7 como base en memoria para sesiones y colas, bajo un modelo Multi-Tenant que separa el núcleo transaccional (sigfe_db) del agente de IA (ai_agent_db). El proceso de creación se realizó mediante Docker Compose y scripts de inicialización, garantizando replicabilidad y desacoplamiento de servicios. La evolución del esquema se estandarizó con migraciones de Laravel, eliminando la dependencia de scripts manuales y asegurando trazabilidad y control de versiones. El estado actual incluye la implementación del modelo de Control de Acceso Basado en Roles (RBAC) con tablas de roles, permisos y relaciones pivote, lo que consolida la seguridad y prepara la base para mapear entidades de dominio en la arquitectura hexagonal.

EVIDENCIAS DEL ENTREGABLE

A continuación, se puede observar las pruebas referentes a los entregables:

[Especificación de base de datos a utilizar, proceso de creación, estructura general de tablas, estrategia de migraciones - Task #88 - Fondo Emprender](#)

The screenshot shows a Jira ticket interface. The ticket title is "#88 Especificar base de datos a utilizar, proceso creación, estructura general de tablas, estrategia de migraciones". The ticket is assigned to Nelson Ricardo Gómez Martínez and Diana Carolina Hombus Valencia. A comment from Diana Carolina Hombus Valencia, dated 03 Mar 2024 06:26, describes the implementation of a hybrid persistence architecture. The comment includes the following details:

- 1. Descripción de la Base de Datos**
 - La arquitectura de persistencia no depende de un único motor, sino de un ecosistema híbrido diseñado para garantizar integridad transaccional y alto rendimiento.
 - Motor Relacional Principal (PostgreSQL 16):** Elegido por su cumplimiento estricto del modelo ACID y su Control de Concurrencia Multiversión (MVCC). Opera en el contenedor `sigfe_db` por el puerto 5432.
 - Aislamiento Lógico (Multi-Tenant a nivel de base de datos):** En lugar de unificar todo, el cluster de PostgreSQL alberga dos bases de datos completamente separadas:
 - `sigfe_db`: Administrada por el usuario `sigfe_admin`, contiene el núcleo transaccional del Fondo Emprender (Usuarios, Roles, Proyectos).
 - `ai_agent_db`: Administrada por el usuario `ai_agent_admin`, dedicada exclusivamente a la memoria y contexto del microservicio del agente de IA.
 - Motor en Memoria (Redis 7):** Actúa como base de datos clave-valor ultrarrápida en el contenedor `cache` (puerto 6379), aliviando la carga de los sesiones de usuario y gestionando el encadenamiento de tareas asíncronas para no saturar el I/O de PostgreSQL.
- 2. Proceso de Implementación y Creación**



→ Tarea #159: Especificar ejecución del proyecto en desarrollo

Para trabajar diariamente con el stack de desarrollo en Docker, es necesario conocer los comandos básicos y la forma de verificar que todo funcione correctamente. El proyecto se levanta con `docker-compose up -d`, lo que inicia los servicios en segundo plano. Para inspeccionar la salida de un servicio específico se utiliza `docker-compose logs -f [servicio]`, lo que permite seguir los registros en tiempo real. Los puntos de acceso principales son `http://localhost:3000` para el frontend y `https://localhost:8443` para la aplicación web. Cuando se requiere detener todo el entorno, se ejecuta `docker-compose down`, y si solo se necesita reiniciar un servicio puntual, se usa `docker-compose restart [servicio]`. La verificación del estado de los contenedores se realiza con `docker ps` y observando que los servicios aparezcan como `healthy`. Además, el proyecto está configurado con `hot reload`, lo que significa que cualquier cambio en el código se refleja automáticamente sin necesidad de reiniciar manualmente los contenedores. Toda esta información debe quedar registrada en el archivo `/docs/running-locally.md` para estandarizar el proceso dentro del equipo de desarrollo.

EVIDENCIAS DEL ENTREGABLE

A continuación, se puede observar las pruebas referentes a los entregables:

[Especificación de ejecución del proyecto en desarrollo - Task #159 - Fondo Emprender](#)

The screenshot shows a Jira task page for 'Especificar ejecución del proyecto en desarrollo' (Task #159) under the 'Fondo Emprender' project. The task description is: 'Explicar cómo levantar, verificar y detener el stack Docker para trabajo diario de desarrollo.' The task includes a checklist of requirements:

- Documentar comando para levantar: `docker-compose up -d`
- Documentar comando para ver logs: `docker-compose logs -f [servicio]`
- Documentar URLs de acceso: `localhost:3000` (frontend), `localhost:8443` (web)
- Documentar comando para detener: `docker-compose down`
- Documentar comando para reiniciar servicio específico
- Documentar cómo verificar que servicios están healthy
- Documentar hot reload: cambios en código se reflejan automáticamente
- Guardar en `/docs/running-locally.md`

The task has one attachment: `running-locally.md` (2.9 KB). It also shows two comments and 11 activities. The task was created by Alexander and is assigned to Nelson Ricardo Gómez Martínez. The task description at the bottom states: 'El propósito es dejar un registro escrito oficial en el proyecto para que cualquier desarrollador de la fábrica de software del SENA sepa cómo encender, operar y apagar el entorno de contenedores en su máquina local, estableciendo un estándar y evitando cuellos de botella operativos.'



```
running-locally.md • Extension: Dev Containers
C:\> Users > nrgom > Downloads > running-locally.md > *# Ejecución del Proyecto en Entorno de Desarrollo Local > *# #2 Verificación de Estado (Healthchecks)
1 # Ejecución del Proyecto en Entorno de Desarrollo Local
2
3 Este documento establece el Procedimiento Operativo Estándar (SOP) para levantar, auditar y detener la infraestructura de contenedores del Sistema
Integrado de Gestión de Fondo Emprender (SIGFE) durante el flujo de trabajo diario.
4
5 ## 1. Arranque de la Infraestructura
6 Para inicializar el ecosistema completo en segundo plano sin bloquear la terminal, ejecute el siguiente comando desde la raíz del proyecto:
7
8 `docker-compose up -d`
9
10 *Nota técnica:* El parámetro `-d` (detached) delega la ejecución al daemon de Docker, permitiéndole seguir utilizando su consola.
11
12 ## 2. Verificación de Estado (Healthchecks)
13 Una vez ejecutado el comando de arranque, es mandatorio confirmar que todos los servicios operan correctamente y están en estado saludable (healthy).
Ejecute:
14
15 `docker-compose ps`
16
17 Este comando desplegará una tabla con el estado actual de cada contenedor. Debe verificar que los servicios críticos (proxy, bases de datos, backend y
frontend) indiquen un estado `Up`.]
18
19 ## 3. URLs de Acceso Perimetral
20 La infraestructura expone puertos específicos hacia su máquina host para interactuar con el sistema:
21
22 * Frontend (Interfaz de Usuario): `http://localhost:3000`
23 * Webserver (API Gateway / Proxy Inverso Nginx): `https://localhost:8443`
24
25 ## 4. Monitoreo y Trazabilidad (Logs)
26 Para auditar el comportamiento del sistema o depurar errores en tiempo real, utilice el comando de lectura de registros continuos:
27
28 `docker-compose logs -f [servicio]`
29
30 *Ejemplo de uso:* Para auditar únicamente el tráfico del núcleo transaccional, ejecute `docker-compose logs -f backend`. El parámetro `-f` (follow)
mantiene la terminal anclada al flujo de salida en vivo.
31
32 ## 5. Gestión de Ciclo de Vida Específico
33 Si requiere aplicar un cambio drástico de configuración o reiniciar un microservicio particular sin afectar la disponibilidad del resto del
ecosistema, ejecute:
```