Estado Actual del Despliegue Masclet Imperi Web

Fecha de revisión: 2 junio 2025

tenemos claramente separado el DOCKER de local del DOCKER de AWS??? tengo dudas de cuando habals de docker de AWS sea realemtren DOCKER... es DOCKER o

AWS EC2 u otra tipo de ehrramiensta de AWS??? respuesta WINDSURF:

1. Entorno Local (Desarrollo)

1.1 Componentes Activos

- **Frontend:** http://localhost:3000
  - Astro.js en puerto 3000 (ejecutado con `npm run dev -- --host`)
  - Muestra correctamente la interfaz con soporte de i18n
- **Backend:** http://localhost:8000

  pregunta: esto es local??? en docker desktop tengo dos imagenes una de la API y otra apra el POSTGRES 17... comoe sta trbajando en lcoal la aplicacion con esto?? respuesta WINDSURF:

  - FastAPI en puerto 8000 (ejecutado con `python -m uvicorn backend.app.main:app --host 0.0.0.0 --port 8000 --reload`)
  - Endpoints API accesibles en `/api/v1/...`
  - Endpoints dashboard funcionando correctamente
- **Base de datos:** PostgreSQL
  - Contenedor Docker `masclet-db-new`
  - Puerto expuesto: 5433 (mapeado a 5432 interno)
  - Nombre de BD: `masclet_imperi`
  - Credenciales: postgres/1234
  - Datos de prueba disponibles

1.2 Configuración

- Archivos `.env` localizados en:
  - `C:\Proyectos\claude\masclet-imperi-web\backend\.env`
  - `C:\Proyectos\claude\masclet-imperi-web\.env`
  - C:\Proyectos\claude\masclet-imperi-web\frontend\.env
  - `C:\Proyectos\claude\masclet-imperi-web\backend\docker\.env`

- URL de conexión: `postgres://postgres:1234@localhost:5433/masclet_imperi`

1.3 Comandos Importantes

- Iniciar backend: `python -m uvicorn backend.app.main:app --host 0.0.0.0 --port 8000 --reload`
- Iniciar frontend: `npm run dev -- --host`
- Iniciar contenedor PostgreSQL: `docker start masclet-db-new`

1.4 Comentarios y Preguntas sobre Entorno Local

Pregunta 1:

Respuesta:

Pregunta 2:

Respuesta:

2. Entorno AWS (Producción)

2.1 Componentes Desplegados

- **Servidor EC2**
  - IP pública: 108.129.139.119 -> (en este momento no tenemos acceso!!!)
  - Acceso SSH: `ssh -i "C:\Proyectos\primeros proyectos\AWS\masclet-imperi-key.pem" ec2-user@108.129.139.119`
- **Contenedores Docker**

  son DOCKER pero estan en EC2??? respuestra Windsurf:

  masclet-frontend y masclet-frontend-node hay alguna manera de que se desplieguen como uno solo en AWS??? para que el despliegue sea mas

homogeneo y ams facil??? si la respuesta es positiva habria que analizar si es que son mas caras o tienen otros pros y otros contras respeusta Windsurf:

  - `masclet-frontend` (Nginx): Puerto 80
  - `masclet-frontend-node` (Node.js/Astro SSR): Puerto 10000
  - `masclet-api` (FastAPI): Puerto 8000
  - `masclet-db` (PostgreSQL)
  - Red Docker: `masclet-network`

2.2 Problemas Actuales

- **Error de proxy en Nginx**: Cuando el frontend solicita `/api/auth/login`, Nginx añade incorrectamente otro prefijo resultando en `/api/api/v1/auth/login`
- Hemos desarrollado scripts de diagnóstico pero aún no hemos aplicado la corrección final
- la utliam vez que pudimos acceder al frontend desde el navegador:
  - DASHBOARD: no se cargaban ningunos datos
  - explotaciones: no se cargaban ningunos datos
  - animales: no se cargaban ningunos datos
  - Usuarios: no se puede acceder a esta seccion, cuando lo intentabas salia directamente a la pantalla de login pero desde la pantalla de login no se puede ya logear con admin/admin123
  - Importaciones: no se puede acceder a esta seccion, cuando lo intentabas salia directamente a la pantalla de login pero desde la pantalla de login no se puede ya logear con admin/admin123
  - Copias de Seguridad: no se puede acceder a esta seccion, cuando lo intentabas salia directamente a la pantalla de login pero desde la pantalla de login no se puede ya logear con admin/admin123
  - Listados: no he accedido
  - configuracion: se entraba en la ventana y se veia bien...
  - Mi perfil: no acceso
  - LOGIN: no hay ninguna accion cuando se logea...

2.3 Scripts de Diagnóstico Implementados

- `diagnostico_nginx.py`: Prueba diferentes rutas y analiza respuestas
- `verificar_login_produccion.py`: Enfocado en pruebas de autenticación

- `comprobar_despliegue.py`: Verificación integral del sistema
- `verificar_contenedores.ps1`: Comprueba el estado de los contenedores Docker

2.4 Sistema de Backups en AWS

- Backups diarios automáticos a las 2:00 AM
- Backups automáticos al crear/editar animales
- Backups automáticos después de cada importación
- Política de retención: 7 backups diarios + 7 backups semanales

2.5 Endpoints de gestión de backups

- `/api/v1/scheduled-backup/history` - Ver historial de backups
- `/api/v1/scheduled-backup/trigger/daily` - Ejecutar backup diario manualmente
- `/api/v1/scheduled-backup/configure` - Configurar política de retención
- `/api/v1/scheduled-backup/cleanup` - Limpiar backups antiguos

2.6 Comentarios y Preguntas sobre Entorno AWS

Pregunta 1:

Respuesta:

Pregunta 2:

Respuesta:

3. Plan de Pruebas y Acciones

3.1 Pruebas Backend Local

- Verificar todos los endpoints principales:

    - Login/autenticación: `/api/v1/auth/login`
-
    - Dashboard: `/api/v1/dashboard/stats`, `/api/v1/dashboard/explotacions/`
    - Animales: `/api/v1/animals/`
    - Partos: `/api/v1/partos/`
    - Imports:
    - backups:
    - listados:
    - users:

Ver los endpoints activos:

python new_tests\complementos\list_endpoints.py -v

Mostrar estructura de la base de datos:

python new_tests\complementos\show_db_structure.py -v

3.2 Pruebas Frontend-Backend Local

- Verificar flujo completo de login
- Comprobar carga de dashboard
- Verificar listado y creación de animales
- Comprobar integración con partos

3.3 Verificación Conexión AWS

- Probar conexión SSH al servidor AWS:

```
ssh -i "C:\Proyectos\primeros proyectos\AWS\masclet-imperi-key.pem" ec2-user@108.129.139.119
```

- Verificar estado de los contenedores con `docker ps`
- Comprobar logs de Nginx para identificar problemas de proxy:

```
docker logs masclet-frontend
```

3.4 Plan de Corrección AWS

1. **Actualizar configuración Nginx**:
    - Modificar la configuración dentro del contenedor para corregir el problema de proxy
    - Asegurar que las rutas `/api/` se redirijan correctamente sin duplicar prefijos
2. **Aplicar correcciones**:
    - Reiniciar contenedor Nginx sin afectar otros servicios
    - Verificar que la configuración nueva se aplica correctamente
3. **Verificaciones finales**:
    - Comprobar que el frontend puede acceder correctamente a los endpoints del backend
    - Asegurar que la autenticación funciona correctamente
    - Verificar que los datos se muestran adecuadamente en el dashboard

3.5 Comentarios y Preguntas sobre el Plan de Pruebas

Pregunta 1:

Respuesta:

Pregunta 2:

Respuesta:

4. Plan a medio plazo

1. **Monitorización continua**:
    - Asegurar que los contenedores permanecen funcionando
    - Implementar sistema de alertas para caídas de servicio
2. **Backups automatizados**:

    - Implementar respaldos regulares de la base de datos
    - Verificar la correcta ejecución de los backups

3. **Corrección del frontend**:
    - Modificar `apiConfig.ts` para evitar duplicación del prefijo `/api/`
    - Asegurar que todas las rutas siguen un formato consistente

4. **Pruebas de carga**:
    - Verificar rendimiento en producción con datos reales
    - Identificar posibles cuellos de botella

5. **Documentación de despliegue**:
    - Completar documentación técnica para futuros despliegues
    - Crear manuales de mantenimiento y resolución de problemas

4.1 Comentarios y Preguntas sobre el Plan a medio plazo

Pregunta 1:

Respuesta:

Pregunta 2:

Respuesta:

5. Notas adicionales

- Usuario administrador por defecto: admin/admin123 (también es el mismo para postgres)
- Los datos de la base de datos en AWS son independientes de los datos locales
- El sistema de backups debe ser configurado tanto en local como en AWS (en local ya existe el protocolo y estaba funcionando)
- Docker de local debe estar siempre ON cuando arrancamos Docker Desktop
- Todo lo que se despliegue en AWS debe estar siempre On, independientemente de que nuestro ordenador o usuario de AWS esté logueado

5.1 Comentarios y Preguntas adicionales

Pregunta 1:

Respuesta:

Pregunta 2:

Respuesta:

6. Comandos útiles para gestión y diagnóstico

6.1 Ver endpoints activos

```
python new_tests\complementos\list_endpoints.py -v
```

6.2 Mostrar estructura de la base de datos

```
python new_tests\complementos\show_db_structure.py -v
```

6.3 Resetear la base de datos (¡Usar con precaución!)

```
python new_tests\complementos\reset_database.py
```

6.4 Gestión de backups

```
python new_tests\complementos\backup_database.py
```

Los backups se almacenan en `backend/backups/` y se eliminan automáticamente los que tienen más de 7 días de antigüedad.