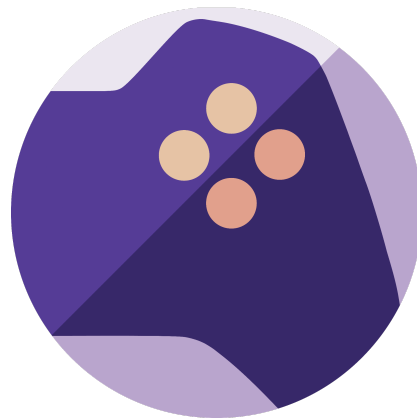


Trabajo Práctico: Implementación de Servidor de Emulación Web con RomM



Materia: Laboratorio de Redes y Sistemas Operativos

Institución: Universidad Nacional de Quilmes (UNQ)

Integrantes:

- Lucas Coronel
- Lourdes de la Cerda
- Hernán Moreyra

Profesor: José Luis Di Biase

Fecha de Entrega: 16 de Julio de 2026

ÍNDICE

1. Descripción del Proyecto	3
2. Relevamiento de características de la máquina utilizada (Host)	3
3. Relevamiento de sistema operativo y software utilizado/necesario	3
4. Instructivo de la instalación	3
Objetivo y tecnología	4
Requisitos Previos	4
Estructura de carpetas	4
Configuración Del Stack Con Docker Compose	4
Volúmenes:	6
Escaneo automático:	6
Caddy Como Proxy Inverso	6
Configuración De Acceso Remoto Con DuckDNS Y Caddy	7
Puesta En Marcha Inicial	8
5. Configuración Inicial:	8
Crear el usuario administrador	8
Roles y usuarios	8
METADATA PROVIDERS: IGDB	9
6. Gestión de Biblioteca:	11
Roms y Bios	11
PLAYSTATION 1	11
MS-DOS	11
ARCADE	11
7. Acceso Remoto y Móvil:	12
Prueba desde móvil	12
Argosy Launcher	12
8. Problemas que surgieron y cómo se resolvieron	22
9. Uso de Inteligencia Artificial	23
10. Conclusión	23

1. Descripción del Proyecto

El proyecto consiste en el despliegue de una infraestructura de red local para alojar **RomM**, un gestor y emulador de videojuegos autoalojado (*self-hosted*). El objetivo principal es proveer una plataforma accesible vía navegador web y app mobile, donde múltiples usuarios puedan gestionar una biblioteca de juegos retro (ROMs) y emularlos en tiempo real sin necesidad de instalar software cliente adicional. Además, se puede administrar usuarios y obtener metadatos de los juegos de la biblioteca desde servicios externos. La arquitectura se basa en microservicios utilizando contenedores, incorporando proxy inverso, control de acceso, y gestión de base de datos relacional.

2. Características de la máquina utilizada (Host)

El servidor principal (Entorno de Desarrollo/Producción) cuenta con las siguientes especificaciones de hardware:

- **Procesador (CPU):** Intel Core i5-11400H
- **Memoria RAM:** 16 GB DDR4
- **Almacenamiento:** SSD NVMe 1TB
- **Adaptador de red principal:** Intel(R) Wi-Fi 6 AX201 160MHz.
- **Tipo de conexión:** Inalámbrica (Wi-Fi 6 / IEEE 802.11ax).

3. Sistema operativo y software utilizado

- **Sistema Operativo Host:** Debian 13
- **Motor de Contenedores:** Docker version 29.6.1, y Docker Compose.
- **Servicios Contenerizados:**
 - **RomM (rommapp/romm):** Aplicación principal (Backend/Frontend).
 - **PostgreSQL (postgres:16):** Motor de base de datos relacional.
 - **Caddy (caddy:latest):** Proxy inverso perimetral.
- **Servicio de DNS Dinámico:** DuckDNS (utilizado para asignar un dominio gratuito a la IP pública del servidor, facilitando el acceso remoto a la plataforma).

4. Instructivo de la instalación

Objetivo y tecnología

Para la implementación se tomó como referencia la [guía oficial de inicio rápido de RomM](#), pero se adaptó la instalación a un entorno propio utilizando **Docker Compose**, **PostgreSQL** como

motor de base de datos y **Caddy** como proxy inverso para el acceso web mediante dominio, con conexión segura (https).

Requisitos Previos

- [Docker Desktop](#) instalado en el host.
- Docker Compose disponible junto con Docker Desktop.
- Conexión a Internet para descargar imágenes, metadatos y dependencias externas.
- Carpeta local destinada a la biblioteca de ROMs y BIOS.
- Cuenta de Twitch con 2FA para habilitar la integración con IGDB.

Estructura de carpetas

Se utilizó una estructura local de biblioteca separando ROMs y BIOS por plataforma.

biblioteca/roms/{platform}/

biblioteca/bios/{platform}/

Esta estructura facilita el escaneo automático de RomM y mantiene separadas las ROMs de los BIOS según la plataforma correspondiente.

Configuración Del Stack Con Docker Compose

En el mismo directorio del proyecto se creó el archivo *docker-compose.yml*, que define los servicios principales del sistema:

```
volumes:
  pg_data:
  romm_resources:
  romm_redis_data:
  caddy_data:
  caddy_config:

services:
  romm:
    image: rommapp/romm:latest
    container_name: romm
    restart: unless-stopped
    environment:
      - ROMM_DB_DRIVER=postgresql
      - DB_HOST=romm-db
      - DB_PORT=5432
      - DB_NAME=romm
      - DB_USER=romm-user
      - DB_PASSWD= ELEGIR UNA CONTRASEÑA QUE COINCIDA CON POSTGRES_PASSWORD
      - ROMM_AUTH_SECRET_KEY= GENERAR UNA CLAVE Y MANTENER EN SECRETO
      - ROMM_BASE_URL= https://labo-room.duckdns.org
```

```
- IGDB_CLIENT_ID= CLIENT_ID
- HASHEOUS_API_ENABLED=true
- IGDB_CLIENT_SECRET= CLIENT_SECRET
- ENABLE_SCHEDULED_RESCAN=true
- SCHEDULED_RESCAN_CRON=*/3 * * * *

volumes:
- romm_resources:/romm/resources
- romm_redis_data:/redis-data
- ./biblioteca:/romm/library
- ./assets:/romm/assets
- ./config:/romm/config

depends_on:
  romm-db:
    condition: service_healthy
    restart: true

caddy:
  image: caddy:2
  container_name: caddy
  restart: unless-stopped
  ports:
    - 80:80
    - 443:443
  volumes:
    - ./Caddyfile:/etc/caddy/Caddyfile:ro
    - caddy_data:/data
    - caddy_config:/config
  depends_on:
    - romm

romm-db:
  image: postgres:16
  container_name: romm-db
  restart: unless-stopped
  environment:
    - POSTGRES_DB=romm
    - POSTGRES_USER=romm-user
    - POSTGRES_PASSWORD= ELEGIR UNA CONTRASEÑA QUE COINCIDA CON DB_PASSWD
  volumes:
    - pg_data:/var/lib/postgresql/data
  healthcheck:
    test: ["CMD-SHELL", "pg_isready -U romm-user -d romm"]
    interval: 10s
    timeout: 5s
    retries: 5
```

- *romm*: aplicación principal
- *romm-db*: base de datos PostgreSQL
- *caddy*: proxy inverso para acceso web seguro, y por un puerto estándar (80/443)

La elección de PostgreSQL permitió separar la persistencia de datos de la aplicación principal, mientras que Caddy se utilizó para exponer el sistema mediante HTTP/HTTPS y facilitar el acceso remoto posterior.

En ROMM_AUTH_SECRET se puede poner un secreto generado por el comando que se ejecuta en terminal: "openssl rand -hex 32". Este comando es una recomendación de seguridad (no es estrictamente necesario escribir el secreto así) para generar una clave de alta entropía de 256 bits. Se usa para firmar criptográficamente los tokens JWT de sesión de los usuarios, garantizando que nadie pueda falsificar una sesión de administrador en la plataforma.

Volúmenes:

Se definieron volúmenes persistentes para conservar la información del sistema aunque los contenedores se reinicien. Estos volúmenes almacenan la base de datos PostgreSQL, los recursos internos de RomM, los datos auxiliares de Redis/Valkey y la configuración persistente de Caddy. Además, se montó la carpeta local *biblioteca* para que RomM pudiera acceder directamente a las ROMs y BIOS almacenadas en el host.

Escaneo automático:

La variable `SCHEDULED_RESCAN_CRON=*/3 * * * *` permite que RomM vuelva a escanear el filesystem automáticamente cada 3 minutos en busca de cambios.

Caddy Como Proxy Inverso

Su función fue recibir el tráfico web externo por los puertos 80 y 443, y reenviarlo internamente hacia el servicio RomM que escucha en el puerto 8080.

Se creó el archivo *Caddyfile*, en la misma carpeta donde está `docker-compose.yml`, con el siguiente contenido:

```
labo-room.duckdns.org {
  encode zstd gzip
  header {
    Strict-Transport-Security "max-age=31536000;"
    X-XSS-Protection "1; mode=block"
    X-Frame-Options "SAMEORIGIN"
    X-Robots-Tag "noindex, nofollow"
    -Server
    -X-Powered-By
  }
  reverse_proxy romm:8080
}
```

- Recibe tráfico por 80/443 y lo redirige a romm:8080
- Permite que el acceso final no dependa del puerto interno del contenedor, sino del dominio configurado para el proyecto.

Configuración De Acceso Remoto Con DuckDNS Y Caddy

Se utilizó [DuckDNS](#) para obtener un dominio gratuito y así evitar depender solamente del acceso local por IP.

- Dominio elegido: labo-room.duckdns.org
- Para que RomM generará correctamente sus enlaces públicos, se definió la `ROMM_BASE_URL` (en `docker-compose.yml`), configurada con la URL pública final <https://labo-room.duckdns.org>.

Para configurar la redirección de puertos 80 y 443 en el router fue necesario acceder al panel de administración del router con las credenciales correspondientes. Se configura como se muestra a continuación:

Puertos

Configuración Puertos

Rellena los siguientes campos y pulsa el botón **Añadir**. Ten en cuenta que para abrir un rango de puertos debes usar el siguiente formato : 5001:5010

Nombre regla de puertos:

Dirección IP:

Protocolo:

Abrir Puerto/Rango Externo (WAN): (ej: 5001:5010)

Abrir Puerto/Rango Interno (LAN): (ej: 5001:5010)

Añadir

Tabla actual de mapeo de puertos

	Nombre	Protocolo	Puerto/Rango Externo	Puerto/Rango Interno	Dirección IP	Activar
✕	VOIP_ATA_0	TCP	8080	80	192.168.1.230	☒
✕	VOIP_ATA_1	TCP	8081	80	192.168.1.231	☒
✕	VOIP_ATA_2	TCP	8082	80	192.168.1.232	☒
✕	VOIP_ATA_3	TCP	8083	80	192.168.1.233	☒
✕	Caddy 443	TCP	443	443	192.168.1.37	☒

Como la IP pública y la IP local pueden cambiar, fue necesario controlar ambos valores durante las pruebas. La IP pública puede verificarse, por ejemplo, mediante <https://api.ipify.org/>, mientras que la IP local puede obtenerse con `ipconfig`. Si la IP local cambia, la redirección de puertos deja de apuntar correctamente al host que ejecuta RomM.

Puesta En Marcha Inicial

Desde una terminal ubicada en el directorio del proyecto se ejecutó: `docker compose up -d`
En la primera ejecución, Docker descarga las imágenes necesarias, crea los volúmenes definidos, inicia PostgreSQL, espera a que la base de datos quede saludable y luego arranca la aplicación RomM.

Durante esta etapa, el acceso inicial puede probarse de manera local utilizando la IP interna del host sobre el puerto de la aplicación, por ejemplo: `http://<IP-local>:8080`

Este acceso corresponde únicamente a la **etapa inicial/local**. Más adelante, una vez configurado Caddy y el dominio, el acceso final pasa a realizarse mediante la URL pública.

5. Configuración Inicial:

Crear el usuario administrador

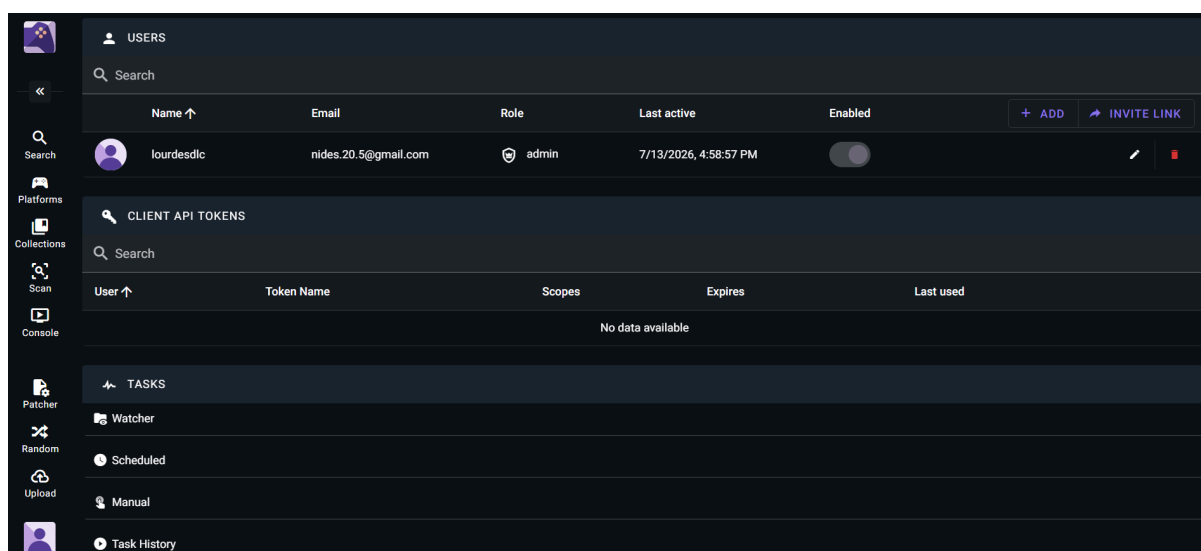
La primera vez que se inicia RomM, el sistema redirige al asistente de configuración inicial. Allí se crea el primer usuario, que automáticamente recibe el rol de administrador.

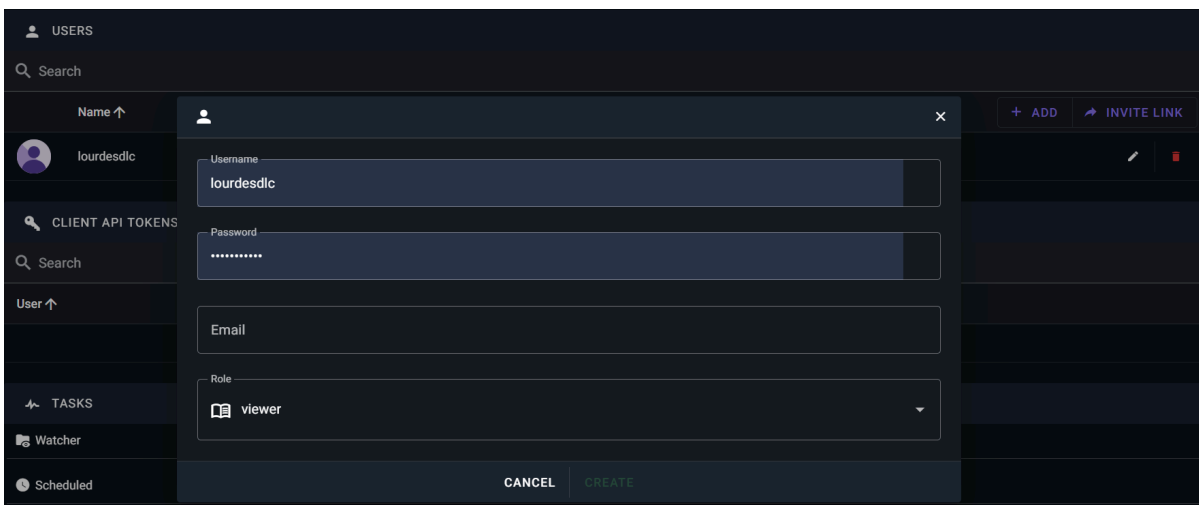
Este usuario administrador es necesario para completar la configuración del sistema, gestionar bibliotecas, asignar roles y habilitar accesos para otros usuarios.

Roles y usuarios

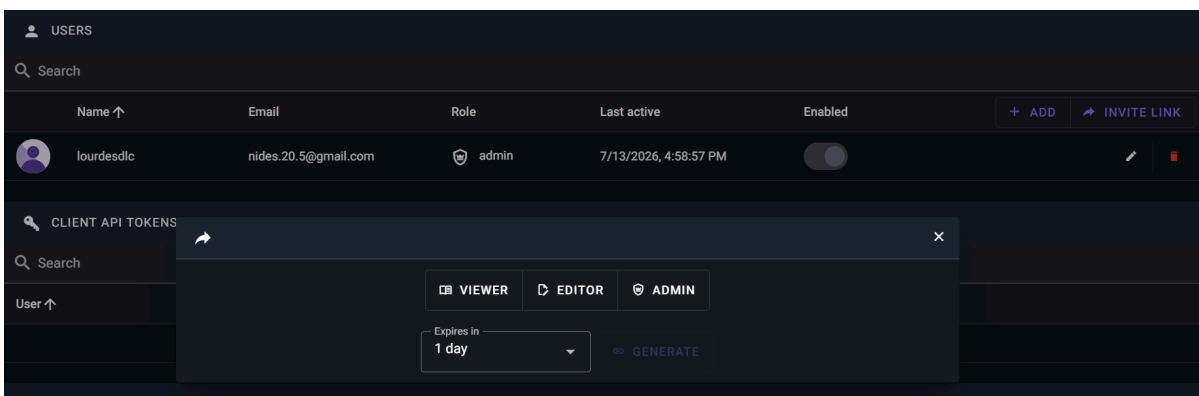
RomM permite agregar usuarios de dos maneras: mediante correo electrónico y mediante enlace de invitación.

En ambos casos, el administrador asigna previamente el rol correspondiente: *viewer*, *editor* o *admin*.





Si el acceso se otorga mediante enlace, además puede definirse un tiempo de validez, entre un mínimo de 6 horas y un máximo de 30 días, configurable usando la variable de entorno `INVITE_TOKEN_EXPIRY_SECONDS`.



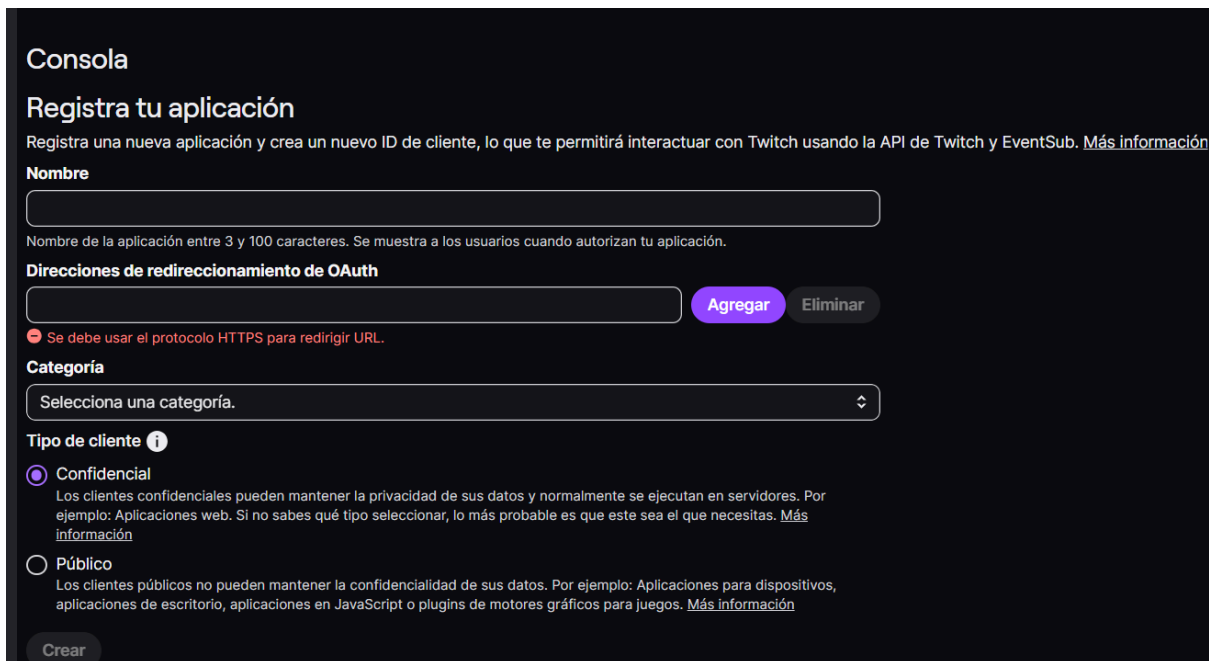
Los roles de los usuarios tienen estas funcionalidades y pueden ser los siguientes:

Rol	Para quién es	Alcances de uso
Administrador	Para vos y para cualquier persona en la que confíes plenamente.	Todos los usos, incluyendo la gestión de usuarios y la ejecución de tareas.
Editor	Miembros de la familia o amigos que ayuden a organizar y gestionar la biblioteca.	Se puede leer todo, editar ROMs/plataformas/colecciones, subir archivos, pero no gestionar usuarios.
Espectador/Viewer	Invitados, cualquier persona que solo quiera jugar y hacer un seguimiento de su propio progreso.	Lee la biblioteca, gestiona tus propios guardados/estados/capturas de pantalla/perfil.

METADATA PROVIDERS: IGDB

Para acceder a la API de IGDB, necesitarás una cuenta de Twitch y un número de teléfono válido para la verificación de dos factores (2FA).

1. Crear o iniciar sesión en [Twitch](#).
2. Tener habilitada la [autenticación de doble factor](#) (2FA).
3. Registrar tu aplicación en el [Portal para desarrolladores de Twitch](#).



Consola

Registra tu aplicación

Registra una nueva aplicación y crea un nuevo ID de cliente, lo que te permitirá interactuar con Twitch usando la API de Twitch y EventSub. [Más información](#)

Nombre

Nombre de la aplicación entre 3 y 100 caracteres. Se muestra a los usuarios cuando autorizan tu aplicación.

Direcciones de redireccionamiento de OAuth

 Agregar Eliminar

Se debe usar el protocolo HTTPS para redirigir URL.

Categoría

Selecciona una categoría.

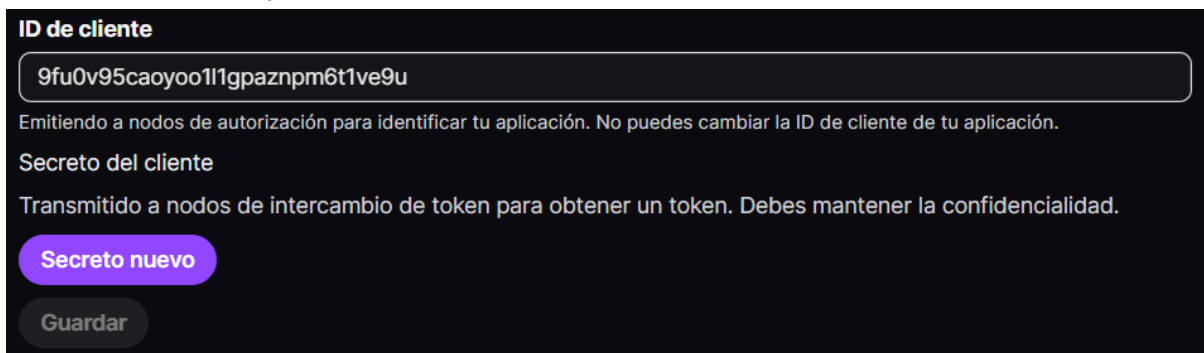
Tipo de cliente

☒ **Confidencial**
Los clientes confidenciales pueden mantener la privacidad de sus datos y normalmente se ejecutan en servidores. Por ejemplo: Aplicaciones web. Si no sabes qué tipo seleccionar, lo más probable es que este sea el que necesitas. [Más información](#)

☐ **Público**
Los clientes públicos no pueden mantener la confidencialidad de sus datos. Por ejemplo: Aplicaciones para dispositivos, aplicaciones de escritorio, aplicaciones en JavaScript o plugins de motores gráficos para juegos. [Más información](#)

Crear

- Nombre: El nombre que elijas debe ser **único**. Si se elige un nombre existente, fallará silenciosamente, sin mensajes de error. Se recomienda algo como *romm-3fca6fd7f94dea4a05d029f654c0c44b*
- URLs de redireccionamiento de OAuth: localhost
- Categoría: Application Integration
- Tipo de cliente: Confidential (Debe estar configurado como *Confidential* para generar secretos de cliente)
- Anotá el ID de cliente y el secreto que aparecen en pantalla y utilízalos para configurar *IGDB_CLIENT_ID* y *IGDB_CLIENT_SECRET*.



ID de cliente

Emitiendo a nodos de autorización para identificar tu aplicación. No puedes cambiar la ID de cliente de tu aplicación.

Secreto del cliente

Transmitido a nodos de intercambio de token para obtener un token. Debes mantener la confidencialidad.

Secreto nuevo

Guardar

6. Gestión de Biblioteca:

Roms y Bios

Las ROMs se almacenan en `/biblioteca/roms/{platform}` y los BIOS en `/biblioteca/bios/{platform}`

PLAYSTATION 1

Para PlayStation 1 se utilizó la BIOS `scph1001.bin`, ubicada en `biblioteca/bios/psx`.

Se descarga y descomprime la rom [Crash Bandicoot](#), en `/biblioteca/roms/psx/Crash Bandicoot`. Asegurarse de descomprimir, ya que Playstation 1 no lee cualquier extensión de archivo.

MS-DOS

Para MS-DOS se utilizó [DOOM2](#).

El juego no arrancaba correctamente porque DOSBox Pure no detectaba automáticamente el ejecutable adecuado dentro del archivo comprimido. Además, la presencia de archivos `DOOMSAV*.DSG` interfería con la selección automática de arranque. Para resolverlo, se eliminaron esos archivos y se agregó un archivo `DOOM2.CONF` con instrucciones de autoarranque para ejecutar directamente `DOOM2.EXE`.

La Rom se ubicó en `/biblioteca/roms/dos`.

Ejemplo de `DOOM2.CONF`

```
[autoexec]
@echo off
Mount C ".."
C:
cls
DOOM2.EXE
exit
```

Por último, se vuelve a comprimir el juego una vez realizado ese ajuste.

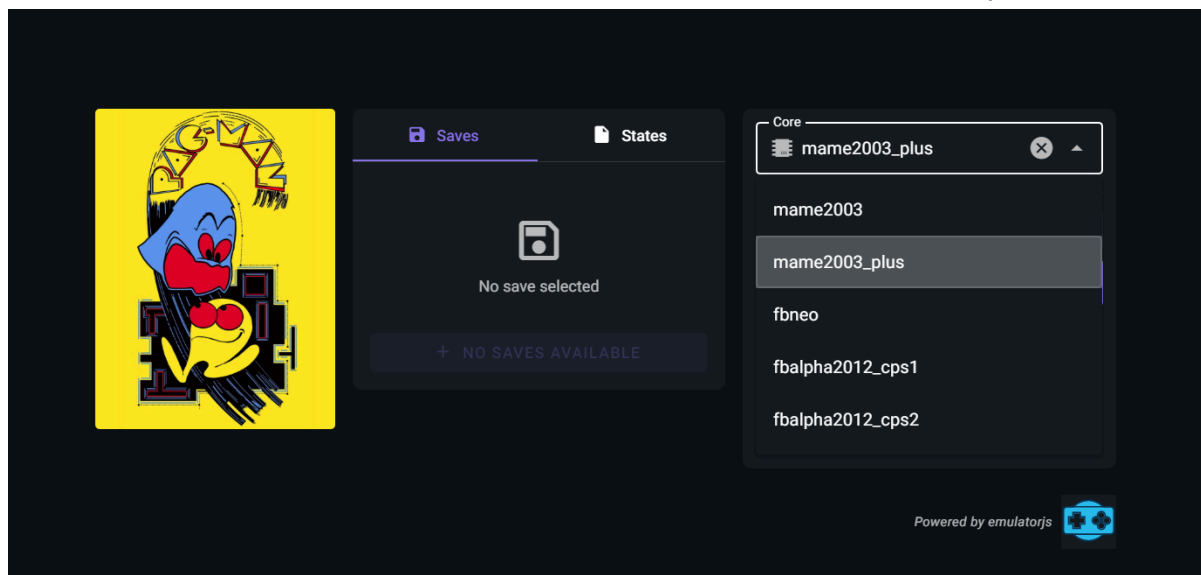
ARCADE

Para Arcade se probaron ROMs como [Pac-Man](#). La rom se guardó en `/biblioteca/roms/arcade/`

Para juegos de la plataforma arcade se recomienda utilizar `mame2003_plus` o `fbneo`, ya que fueron los cores que dieron resultado correcto en las pruebas realizadas y son adecuados para juegos arcade generales.

No se recomienda usar `mame2003` como opción principal, porque puede ser más estricto con la versión exacta del romset. Tampoco se recomienda `fbalpha2012_cps1` ni `fbalpha2012_cps2`

para juegos arcade generales, ya que esos cores están orientados a hardware arcade específico de Capcom CPS-1 y CPS-2.



7. Acceso Remoto y Móvil:

Prueba desde móvil

Se ingresó a la URL pública del sistema <https://labo-room.duckdns.org/> desde la red de datos móviles (4G) de un teléfono celular, comprobando que el acceso externo funciona y que la interfaz se adapta correctamente a un dispositivo táctil.

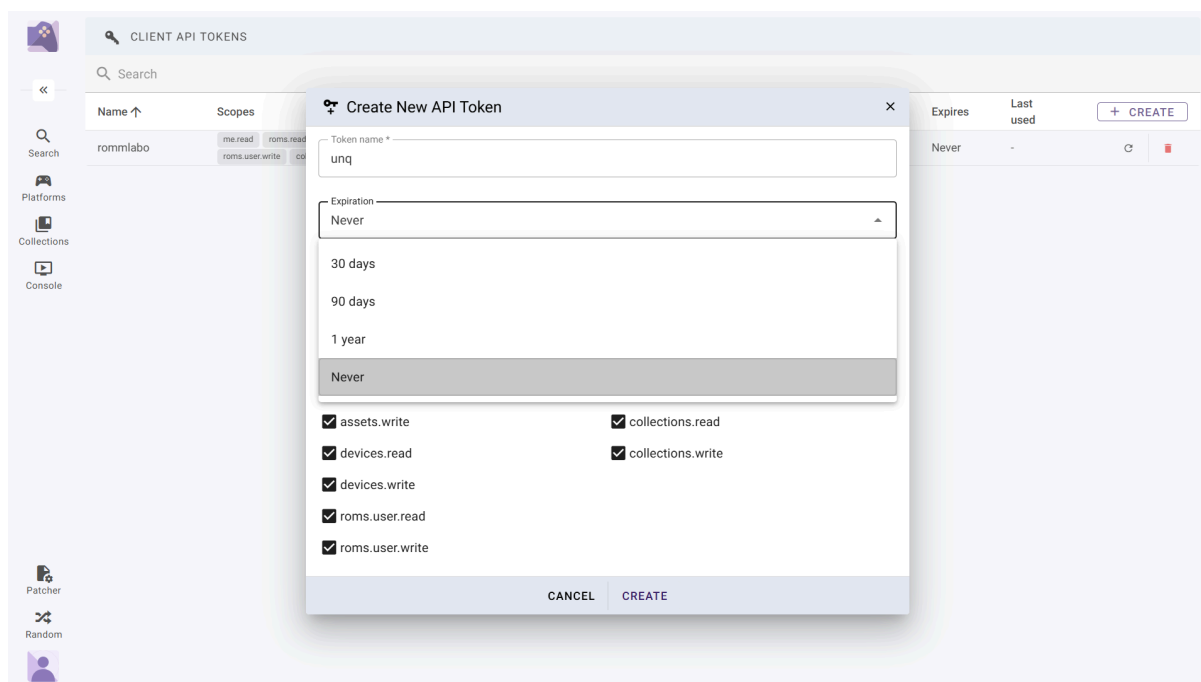
Esta prueba permite verificar tanto el acceso remoto real como el comportamiento general de la interfaz fuera de la red local.

Argosy Launcher

Argosy es un cliente móvil nativo para Android diseñado específicamente para consolas portátiles retro (como las familias Anbernic, Retrodroid Pocket u Odin) y dispositivos móviles, que cuenta con una integración profunda con **RomM**. Permite sincronizar la biblioteca de juegos, descargar archivos bajo demanda, realizar un seguimiento de logros y sincronizar de manera automática las partidas guardadas en la nube.

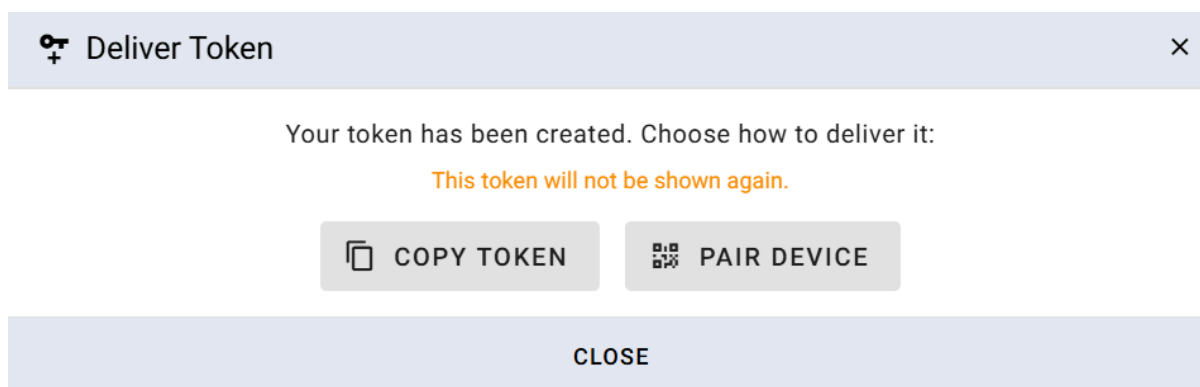
- Paso 1: Generación del Token de Acceso en RomM

Para poder configurar la app mobile se debe generar un token desde la interfaz de RomM. En el cual se detallan los permisos y una fecha de expiración.



- Paso 2: Vinculación del Dispositivo (Pairing)

Luego de haber generado el token debemos vincularlo al dispositivo móvil mediante la opción "Pair device".



La pantalla mostrará un **código de 8 caracteres** y un **código QR** único para el emparejamiento.

Pair Device



EP65-JJKG

58s

BACK

CLOSE

- Paso 3: Configuración Inicial de la App en Android

Desde la app tendremos la siguiente interfaz donde tendremos que introducir la URL del servidor para luego vincularlo mediante el QR o el código de 8 caracteres.

ARGOSY

Welcome! Let's get you set up.

Get Started

Rom Manager Login

Server URL

https://romm.example.com

Enter your RomM server address and we'll
detect how to sign in.

Continue

Back

Enter pairing code

<https://labo-room.duckdns.org/>

Create an API token in the RomM web UI,
then enter the 8-character code.

 -

Connect

Edit URL

Scan QR Code

- Paso 4: Asignación de permisos

La aplicación solicitará accesos del sistema. El más crítico es la **gestión de almacenamiento** de Android, indispensable para que Argosy pueda descargar e iniciar las ROMs y las BIOS directamente en el almacenamiento local del dispositivo.

Permissions

Argosy needs these to manage your library and emulators.



Manage Storage (Required)

Required to browse folders, download games, and sync saves.



Notifications

Shows download and sync progress.



Display Over Other Apps

Enables in-game save detection overlays.



Usage Access

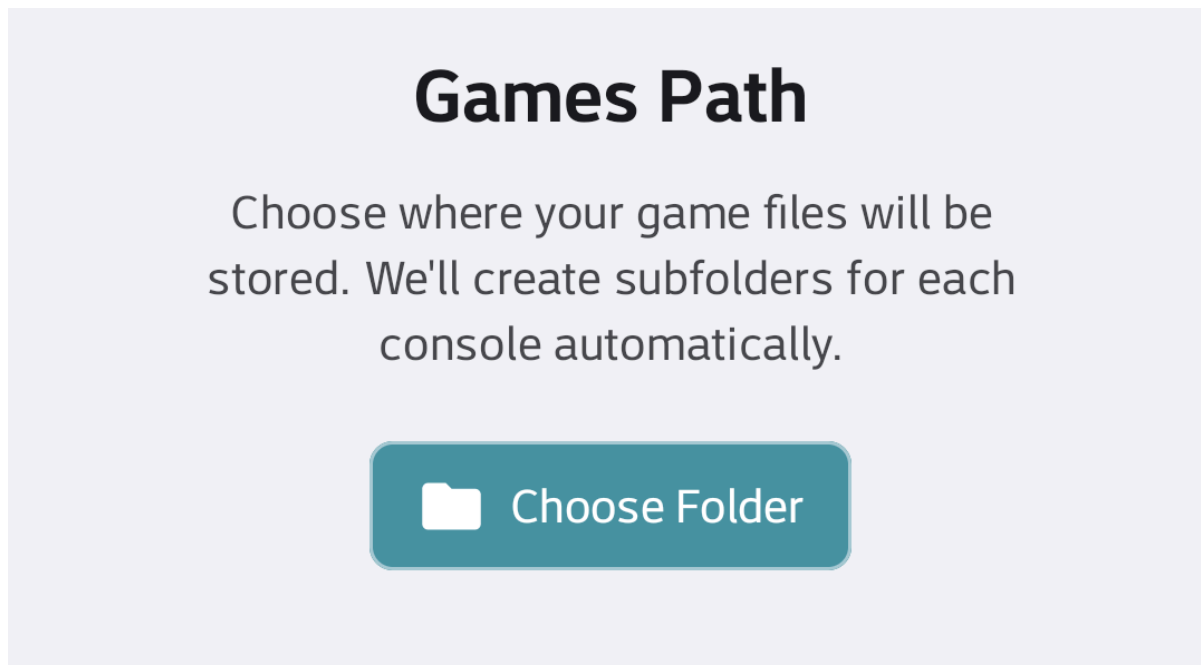
Enables accurate play time tracking and resume.

Continue

You can grant the other permissions later in Settings.

- **Paso 5: Directorios de almacenamiento**

Luego debemos definir la ruta local del dispositivo donde se descargarán y organizarán los archivos de los juegos (ROMs).



De manera opcional, debemos configurar el directorio destinado a almacenar la caché interna del sistema (imágenes de carátulas, metadatos, etc.).

OPTIONAL

Image Cache Location

By default, game artwork is stored in the app's internal cache. If your device has limited storage, you can choose an external location.



Choose External Folder

Use Default (Internal)

Images will be stored in an 'argosy_images' subfolder.

- Paso 6: Sincronización de catálogos

Luego debemos seleccionar las plataformas y juegos que queremos que se sincronicen con el uso de la app.

SELECT PLATFORMS

Choose which platforms to sync

3 of 11 platforms selected

11 platforms   

Nintendo 64 2 games	☒
Game Boy 0 games	☐
Game Boy Advance 0 games	☐
PlayStation 1 games	☒
Ps 0 games	☐
PlayStation Portable 0 games	☐
Arcade 3 games	☒
Mame 0 games	☐
DOS 1 games	☒
Msdos 0 games	☐
Nueva Carpeta 0 games	☐

Select All Continue

- Paso 7: Configuración de Cores de Emulación

Podemos elegir descargar los cores de emulación. De esta forma, se podrá jugar a los juegos de esas plataformas inmediatamente dentro de Argosy, de forma fluida, sin configurar nada y sin tener que salir a aplicaciones externas como RetroArch.



Download Emulator Cores?

3 libretro cores are available for your selected platforms.

These enable built-in emulation without needing separate emulator apps.



Download Cores

Skip (Disable Built-in Emulator)

Skipping turns off the built-in emulator. You can re-enable it and download cores later from Settings.

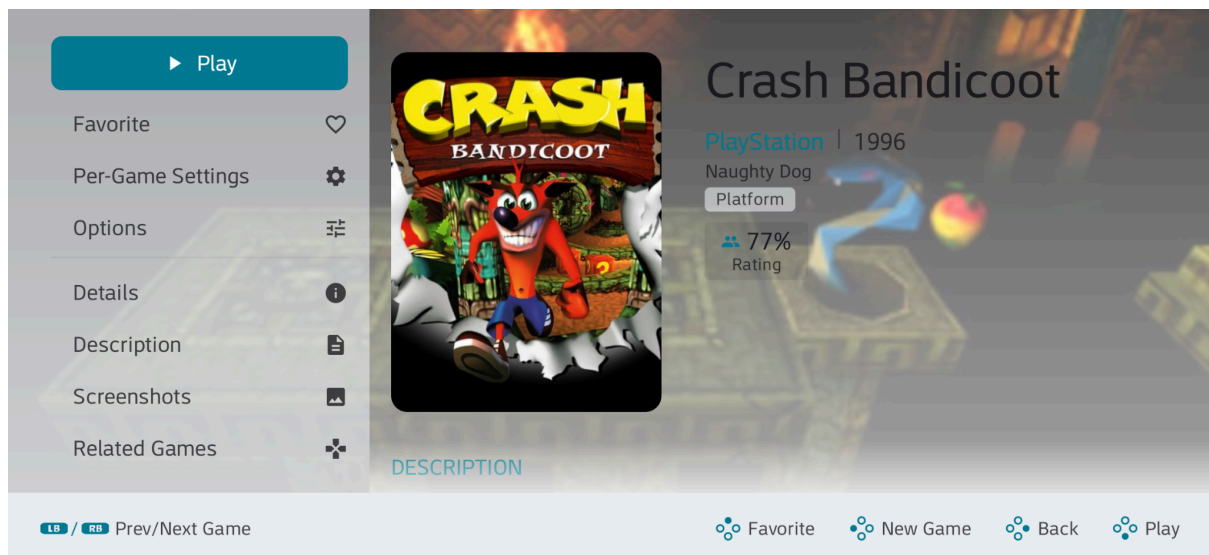
Una vez completados estos parámetros, la interfaz principal presentará el catálogo de videojuegos con sus carátulas y metadatos correspondientes, listos para ser descargados y jugados en tiempo real.



All Set!

7 games across 11 platforms ready to sync
Sync your library from Collection settings to get started.

Start Playing



8. Problemas que surgieron y cómo se resolvieron

Durante la fase de integración y pruebas, surgieron los siguientes inconvenientes técnicos:

1. Fallo en el planificador automático (Cron Error): El contenedor falló al iniciar reportando `ValueError: invalid range specifier: '"*' ('"*)`. Docker Compose estaba interpretando las comillas como literales. **Solución:** Se editaron las

variables de entorno para remover las comillas de la cadena de Cron, dejando el valor puro (`*/3 * * * *`).

2. El cambio de IP pública afectó temporalmente el acceso remoto por DuckDNS. Una vez detectado el inconveniente, se actualizó manualmente la IP pública asociada al dominio.
3. El cambio de IP local obligó a actualizar la redirección de puertos del router para que siguiera apuntando correctamente al host que ejecutaba Docker.
4. En Arcade fue necesario seleccionar cores compatibles como *mame2003_plus* o *fbneo*.
5. En MS-DOS hubo que corregir el arranque de DOOM2 para que DOSBox Pure ejecutará automáticamente el archivo correcto.
6. A algunos integrantes del grupo nos pasó que no podíamos instalar el cliente mobile, por restricciones de Android y Google. Nos decía que Google Play Protect bloqueaba la instalación del .apk. No en todas las versiones de Android sucede esto, por lo que un integrante pudo instalar la app.
7. Juegos de ms-dos no funcionaban sobre HTTP, ya que el emulador de MS-DOS (DOSBox Pure) utiliza una tecnología de WebAssembly que requiere una función de memoria llamada *SharedArrayBuffer*. Por cuestiones de seguridad (para evitar ataques a los procesadores como *Spectre* o *Meltdown*), todos los navegadores modernos bloquean el uso de *SharedArrayBuffer* si la conexión no es un "Contexto Seguro" (HTTPS).

9. Uso de Inteligencia Artificial

- **Modelo utilizado:** Google Gemini.
- **Propósito:** Asistencia técnica para el diseño de la arquitectura de contenedores, redacción inicial del archivo *docker-compose.yml*, y resolución de errores durante el despliegue mediante el análisis de *logs* de Docker. La configuración de Caddy dentro del compose para la configuración del contenedor y los puertos.
- **Prompts utilizados:**
 - Cómo configurar Caddy como proxy reverso para RomM en Docker
Compose redirigiendo el tráfico hacia el puerto romm:8080

10. Conclusión

Con esta configuración se logró una instalación funcional de RomM con acceso local y remoto, metadatos, usuarios, escaneo automático y ejecución de juegos desde el navegador. A través de este proyecto, se consolidaron conocimientos fundamentales sobre orquestación de servicios en Docker, configuración de redes perimetrales y resolución de problemas técnicos complejos en entornos de producción.