V1.0 - MAIG 2025

Actualitzat el 23/05/2025

# MANUAL TÈCNIC

# SWAPHUB

Ghizlane Nouali Nouali
suport@swaphub.cat

# 1. INFORMACIÓ GENERAL

## 1.2 Propòsit del document

Aquest document tècnic té com a objectiu proporcionar una descripció exhaustiva de l'arquitectura, la configuració, el desplegament i el manteniment tècnic de l'aplicació **SwapHub**. Està adreçat a desenvolupadors, administradors de sistemes i personal tècnic que necessiti entendre o intervenir en el funcionament intern del sistema.

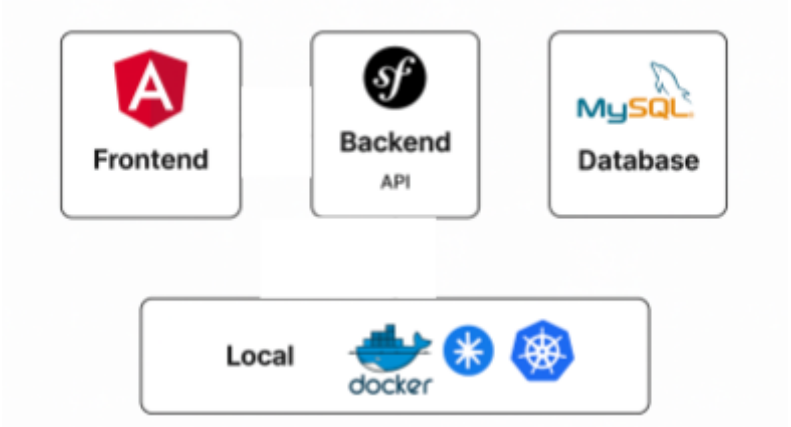
El manual cobreix els components del frontend, backend, base de dades, integracions externes, seguretat i eines de monitorització, així com guies detallades per a la instal·lació i resolució de problemes.

# ÍNDEX

# 2. ARQUITECTURA DEL SISTEMA

## 2.1 Visió General

L'arquitectura de *SwapHub* està basada en una estructura **client-servidor** dividida en tres capes principals: **frontend**, **backend** i **base de dades**. Aquesta arquitectura modular afavoreix la mantenibilitat, l'escalabilitat i la separació de responsabilitats dins del sistema.




**Patrons de disseny utilitzats**

- **MVC (Model-View-Controller)** al backend (Symfony) per organitzar les responsabilitats i separar la lògica de negoci de la interfície.
- **Component-Based Architecture** al frontend (Angular), que permet una modularitat i reutilització del codi.
- **RESTful API** per la comunicació entre frontend i backend.
- **Repository Pattern** per a l'accés a dades des del backend.

**Stack tecnològic implementat**

| Capa | Tecnologia | Descripció |
|---|---|---|
| Frontend | Angular + DaisyUI | Interfície d'usuari moderna i responsive |
| Backend | Symfony (PHP) | Lògica de negoci, gestió d'usuaris i APIs |
| Base de dades | MySQL | Emmagatzematge de dades relacionals |
| Control de versió | Git (GitHub) | Control de versions |
| Altres eines | Composer, NPM, Node.js | Gestors de paquets per backend i frontend |

## 2.2 Components Principals

**Frontend**

- Framework: **Angular**
- Llibreria d'estil: **DaisyUI** sobre TailwindCSS
- Funcionalitats:
    - Visualització de productes
    - Formularis d'intercanvi
    - Gestió de sol·licituds (enviades / rebudes)
    - Autenticació d'usuaris
    - Navegació SPA (Single Page Application)

**Backend**

- Framework: **Symfony (PHP)**
- Responsabilitats:
    - Exposició d'APIs REST
    - Gestió d'usuaris, intercanvis i autenticació
    - Validació de dades i lògica de negoci

**Base de Dades**

- Gestor: **MySQL**
- Model relacional
- Inclou taules per:
    - Usuaris
    - Objectes
    - Sol·licituds d'intercanvi
    - Imatges
    - Notificacions

# 3. ENTORN TÈCNIC

## 3.1 Requeriments de Hardware

Per al correcte funcionament de l'aplicació **SwapHub**, es recomanen les següents característiques mínimes de hardware per al servidor d'allotjament:

- **Capacitat d'emmagatzematge:**
  - Mínim 50 GB per a dades, logs i còpies de seguretat.

- **Memòria RAM:**
  - Recomanada mínim 4 GB, ideal 8 GB per optimitzar el rendiment en processos simultanis.

- **Processador:**
  - CPU amb almenys 2 nuclis, preferiblement 4 o més per a millor resposta sota càrrega.

## 3.2 Requeriments de Software

L'entorn de software necessari per executar i mantenir l'aplicació inclou:

- **Servidor web:**
  - Nginx 1.18 (configuració recomanada per gestionar PHP i les peticions HTTP).
- **Base de dades:**
  - MySQL 8.0 o superior.
- **Framework backend:**
  - Symfony 6.x (PHP 8.1 o superior).
- **Llenguatges de programació:**
  - PHP (backend)
  - TypeScript (frontend Angular)
- **Gestor de paquets i dependències:**
  - Composer (PHP)
  - NPM (Angular)
- **Dependències i llibreries principals:**
  - DaisyUI (estils CSS)
  - Angular CLI
  - Altres paquets PHP i JavaScript gestionats mitjançant Composer i NPM.

# 4. CONFIGURACIÓ DEL SISTEMA

## 4.1 Instal·lació

El procés d'instal·lació de *SwapHub* és el següent:

1. **Clonar el repositori:**
   Obtenir el codi font des del repositori GitHub utilitzant:

```
git clone https://github.com/GhizlaneNouali/SwapHub.git
```

2. **Configuració del backend (Symfony):**
   Instal·lar dependències amb Composer:

```
composer install
```

3. **Configurar les variables d'entorn (fitxer .env):**
   - Connexió a base de dades (DATABASE_URL)

4. **Configuració del frontend (Angular):**
   Entrar a la carpeta del frontend i instal·lar dependències:

```
npm install
```

5. **Base de dades:**
   Crear la base de dades i executar les migracions amb Symfony:

```
php bin/console doctrine:database:create
php bin/console doctrine:migrations:migrate
```

6. **Compilar i executar en mode desenvolupament:**

```
ng serve
```

## 4.2 Desplegament

El procediment de desplegament per a entorn de producció és el següent:

1. **Construir el frontend per producció:**
   Això generarà fitxers optimitzats a la carpeta dist/.

```
ng build --prod
```

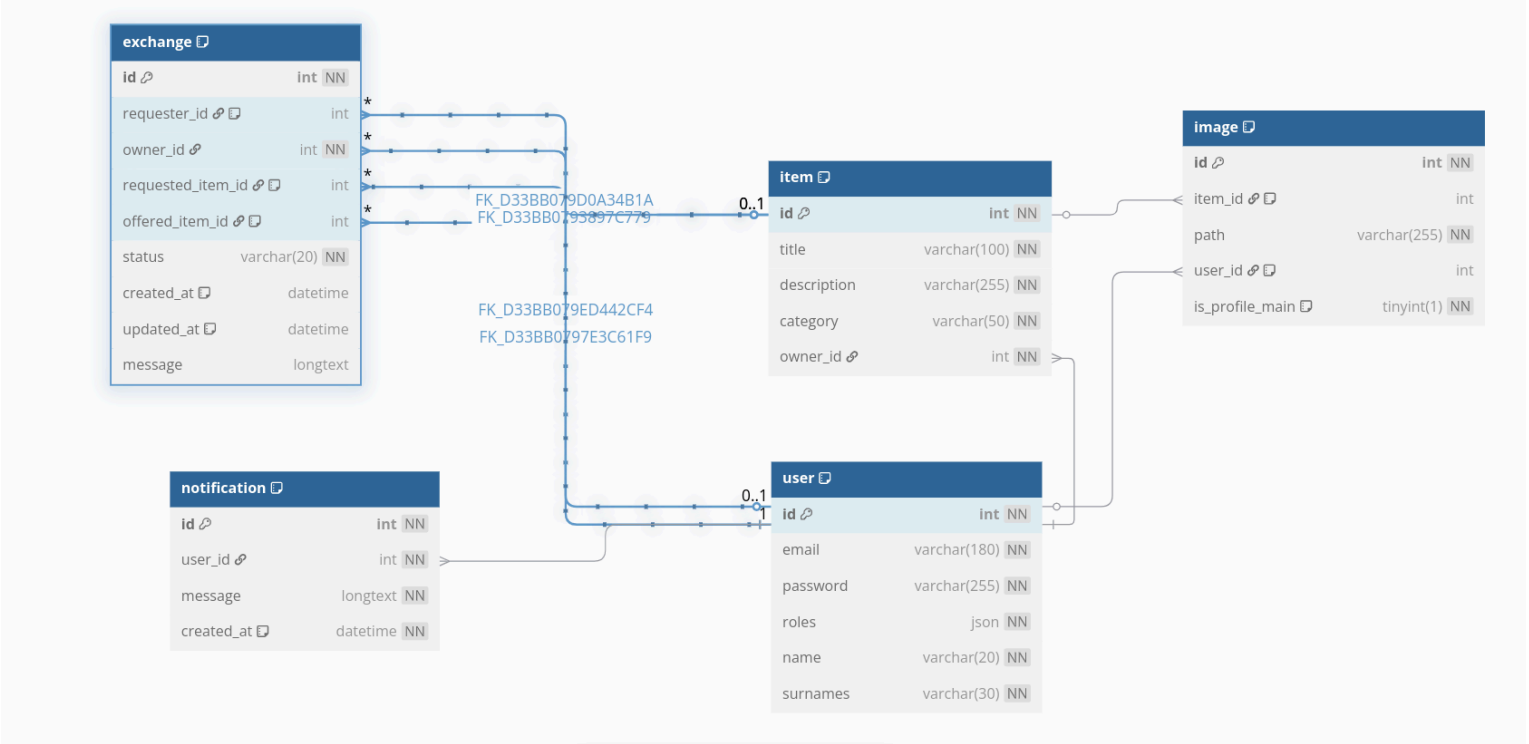
2. **Desplegar el backend:**
   Actualitzar dependències (composer install --no-dev)
   Executar migracions de base de dades si cal.

3. **Copiar fitxers compilats al servidor**

4. **Configuració del servidor web:**
   Ajustar la configuració Nginx per a servir el backend i el frontend, assegurant la correcta redirecció de rutes i la seguretat del servei.

# 5. BASE DE DADES

## 5.1 Model de Dades



# 6. Estructura del Codi

## 6.1 Organització del Projecte

El projecte *SwapHub* està estructurat en dues parts principals: frontend i backend, seguint una arquitectura desacoblada que permet un manteniment més eficient i una escalabilitat més senzilla.

**Estructura de directoris**

**Backend (Symfony):**

/src → Codi font principal (controladors, serveis, entitats)
  /Controller → Controladors que gestionen les peticions HTTP i respostes
  /Repository → Repositoris associats a cada entitat per accedir/consultar la bd
  /Entity → Entitats Doctrine que representen les taules de la base de dades
/config → Configuració de l'aplicació
/migrations → Fitxers de migració de la base de dades
/public → Arxius públics (index.php, recursos estàtics)
/var → Fitxers generats (cache, logs)
/vendor → Dependències gestionades per Composer

**Frontend (Angular amb DaisyUI):**

/src
  /environments → Fitxers de configuració per entorns
  /app → Components, serveis i rutes
    /components → Components reutilitzables i de pàgina
    /guards → Fitxers que controlen l'accés a les rutes segons l'estat de l'usuari
    /models → Definicions de models i interfícies de tipus
    /services → Serveis per a la gestió de dades, autenticació, intercanvis, etc.

**Convencions de nomenclatura**

- Els **components Angular** segueixen el patró nom-funcionalitat.component.ts.
- Les **classes Symfony** es nomenen en PascalCase (NomControlador.php, NomServei.php).
- Les **entitats** es nomenen en singular (Usuari, Objecte).
- S'usa l'idioma anglès per al codi per coherència internacional (UserController, ItemService, etc.).

**Patrons implementats**

- **MVC** (Model-View-Controller) a Symfony per separar les responsabilitats del backend.
- **Routing modular** tant a Angular com a Symfony per facilitar la navegació i manteniment.
- **Observables i serveis compartits** al frontend per la gestió de dades i estat.

## 6.2 Components Principals

**Backend (Symfony)**

- **Controladors (Controllers)**
  Són les rutes del servidor. Cada vegada que un usuari fa una acció (per exemple, enviar una sol·licitud d'intercanvi), el navegador envia una petició, i aquesta arriba a un controlador, que decideix què fer.

  Exemple: ExchangeController gestiona tot el relacionat amb els intercanvis.

- **Entitats (Entities)**
  Representen les taules de la base de dades. Cada entitat és com una plantilla per crear registres, com ara usuaris o objectes.

  Exemple: User, Item, Exchange són entitats que corresponen a les taules del projecte.

- **Repositoris (Repositories)**
  Són les classes encarregades de fer consultes a la base de dades. Per exemple, buscar tots els objectes d'un usuari.

  Exemple: ItemRepository permet recuperar els objectes d'un usuari.

**Frontend (Angular amb DaisyUI)**

- **Components**

  Són les parts visuals de la web: un component per a mostrar objectes, un altre per a la pàgina del perfil, etc.

  Exemple: HomeComponent, ProfileComponent.

- **Serveis (Services)**

  S'encarreguen de parlar amb el backend. Quan vols obtenir els objectes des del servidor, el component demana al servei que ho faci.

  Exemple: ItemService fa peticions a l'API per obtenir els objectes.

- **Guards (Guàrdies)**

  Són mecanismes per controlar si un usuari pot entrar a una pàgina. Per exemple, impedir que un usuari no loguejat accedeixi al seu perfil.

  Exemple: AuthGuard bloqueja l'accés a rutes privades si no estàs autenticat.

# 7. APIS I INTERFÍCIES

## 7.1 APIs Internes

**Endpoints disponibles**

L'aplicació exposa diverses rutes HTTP per permetre la comunicació entre el frontend i el backend. Aquests endpoints són els punts d'accés per a la gestió d'usuaris, objectes, intercanvis i altres funcionalitats clau.

Exemple d'alguns endpoints:

- GET /api/items — Obtenir la llista d'objectes disponibles.
- POST /api/exchanges — Crear una nova sol·licitud d'intercanvi.
- PUT /api/exchanges/{id}/accept — Acceptar un intercanvi.
- DELETE /api/items/{id} — Eliminar un objecte.

**Mètodes HTTP**

Les peticions utilitzen els mètodes habituals: GET, POST, PUT, DELETE per obtenir, crear, modificar o eliminar dades.

**Formats de petició i resposta**

Les dades es transmeten en format JSON per assegurar compatibilitat i facilitat de processament entre client i servidor.

**Autenticació i autorització**

L'accés als endpoints està protegit mitjançant autenticació amb tokens JWT (JSON Web Tokens). Cada petició inclou el token en l'encapçalament per garantir que només usuaris autoritzats puguin executar accions.

## 7.2 APIs Externes

Actualment, l'aplicació no integra cap servei extern a través d'APIs. No obstant això, l'arquitectura està preparada per permetre la incorporació futura de serveis de tercers, com ara plataformes de geolocalització, notificacions o altres funcionalitats addicionals.

# 8. SEGURETAT

## 8.1 Mecanismes de Seguretat

- **Autenticació**
  L'aplicació implementa un sistema d'autenticació basat en tokens JWT (JSON Web Tokens), que permet verificar la identitat dels usuaris de manera segura i eficaç.
- **Autorització**
  Les rutes i funcionalitats estan protegides per controls d'autorització que asseguren que cada usuari només pugui accedir o modificar la informació a la qual està autoritzat.
- **Encriptació**
  Les dades sensibles, com les contrasenyes, es guarden utilitzant algoritmes d'encriptació robustos (bcrypt), evitant emmagatzemar informació en text pla.
- **Gestió de sessions**
  El sistema no manté sessions tradicionals sinó que utilitza tokens amb data de caducitat, millorant la seguretat i la escalabilitat.

# 9. RESOLUCIÓ DE PROBLEMES

## 9.1 Problemes Comuns

A continuació es detallen alguns dels problemes més habituals detectats durant el desenvolupament i ús de l'aplicació, juntament amb les seves possibles solucions:

- **Problemes d'autenticació:**
    - *Descripció:* Usuari no pot iniciar sessió.
    - *Causes possibles:* Contrasenya incorrecta, usuari no registrat.
    - *Solucions:* Comprovar dades d'entrada, restablir contrasenya.
- **Error en l'enviament de sol·licituds d'intercanvi:**
    - *Descripció:* La sol·licitud no arriba a l'usuari destinatari.
    - *Causes possibles:* Problemes de connexió, errors en el backend.
    - *Solucions:* Revisar la connexió, consultar logs per errors.
- **Problemes amb la visualització d'objectes:**
    - *Descripció:* Les imatges no es carreguen correctament.
    - *Causes possibles:* Problemes en l'emmagatzematge d'imatges, rutes incorrectes.
    - *Solucions:* Verificar els enllaços d'imatges i permisos d'accés.

## 9.2 Suport

**Contactes de suport**

Per a consultes tècniques o incidències, es pot contactar amb l'equip de desenvolupament a través de l'adreça de correu electrònic: support@swaphub.cat