UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

"CUNOC"

Docente:

Moises Granados

TRABAJO:

MANUAL DE TECNICO:

ESTUDIANTE:
YERRI JONATAN GADIEL BAMACA LOPEZ

202132066

Manual Técnico

Este manual describe la arquitectura, componentes, modelos de datos, y los pasos para instalar y desplegar ambas aplicaciones (frontend Angular y backend Node.js y MySql).

1. Arquitectura y despliegue

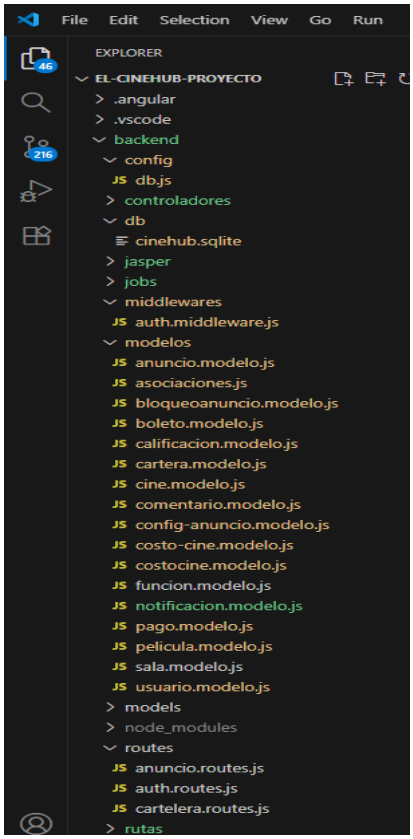
- Frontend: Angular, servido en desarrollo con `ng serve` y compilado con ng build.
- Backend: Node.js + (MySQL). JWT para auth.
- Base de datos: MySQL
- Despliegue recomendado: Backend en un servicio Node (PM2/Docker), Frontend y MySQL administrado.

2. Componentes

- Rutas de backend (`backend/rutas/*.ruta.js`).
- Controladores (`backend/controladores/*.controlador.js`).
- Modelos (`backend/modelos/*.modelo.js`).
- Asociaciones (backend/modelos/asociaciones.js).

- Reportes PDF (backend/jasper/jasper-helper.js, reportes-sistema.controlador.js).

- Frontend (src/app/), rutas en src/app/app.routes.ts.



- Mapeo físico de la DB: manejado por Mysql; ver backend/modelos/*.modelo.js y asociaciones.js.

4. Backend (API)

- Autenticación: POST /api/auth/registro, POST /api/auth/login.

- Admin-Cine: gestión de salas y funciones; bloqueo de anuncios; listar bloqueados.

- Admin-Sistema: configurar costos, moderar anuncios, y reportes.

- Reportes PDF: `/api/reportes-sistema/*` descarga PDF; Dashboard consolidado: `/api/reportes-sistema/dashboard` y `/api/admin-sistema/dashboard`.

Contratos relevantes:

- Registro: body { nombre, correo, contraseña, [edad], [telefono], [tipo] } → 201 + usuario.

- Bloquear anuncio: body { anuncioId, dias } (token admin-cine) → descuenta cartera, crea bloqueo.

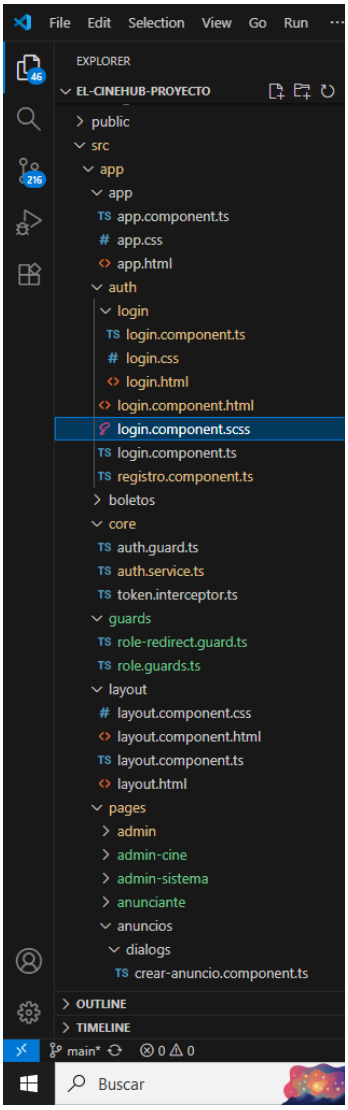
- Dashboard consolidado: JSON con { resumenGanancias, anunciosComprados, gananciasPorAnunciante, salasPopulares, salasMasComentadas }.

5. Frontend (Angular)

- Rutas en src/app/app.routes.ts (incluye /registro, /login).

- Componente RegistroComponent envía payload correcto (correo/contraseña) y muestra snackbars.

- Activos estáticos en `src/assets` mapeados en angular.json.

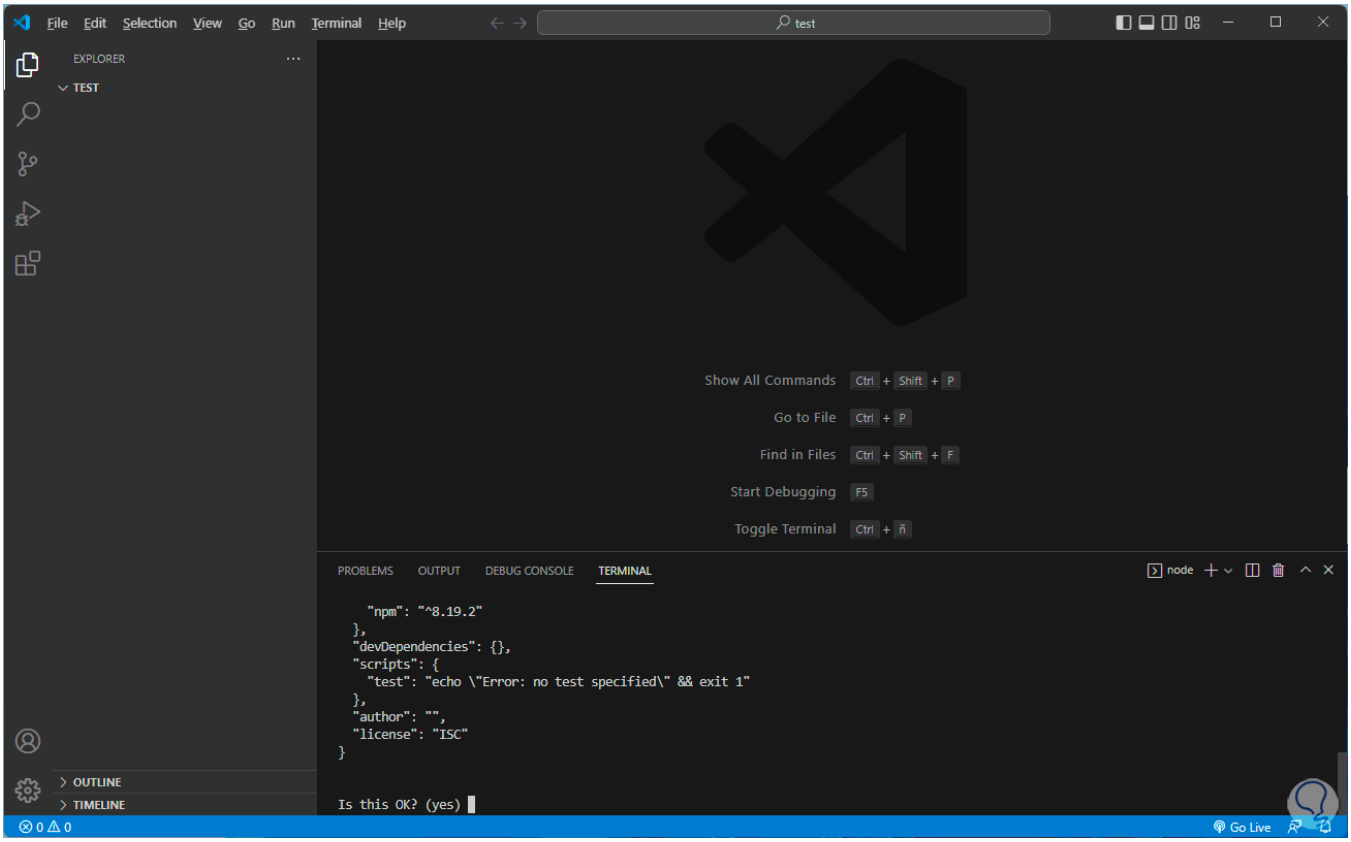


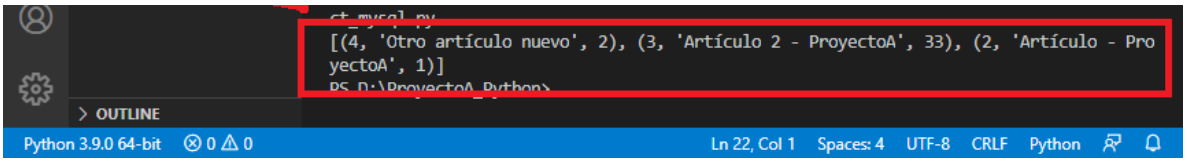
6. Instalación y despliegue

Requisitos: Node 18+, npm, MySQL 8.

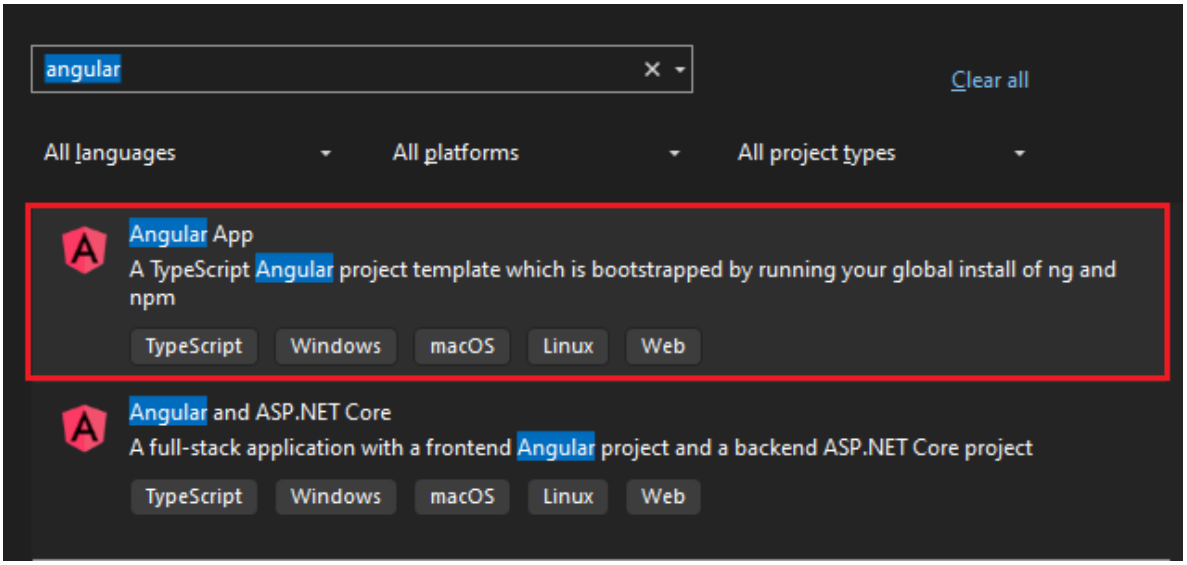
Backend:

1. Configura variables en backend/.env (DB, JWT_SECRET).
2. Instala dependencias: npm install en backend
3. Sincroniza BD: al iniciar node index.js crea/actualiza tablas.
4. Arranca servidor: node index.js

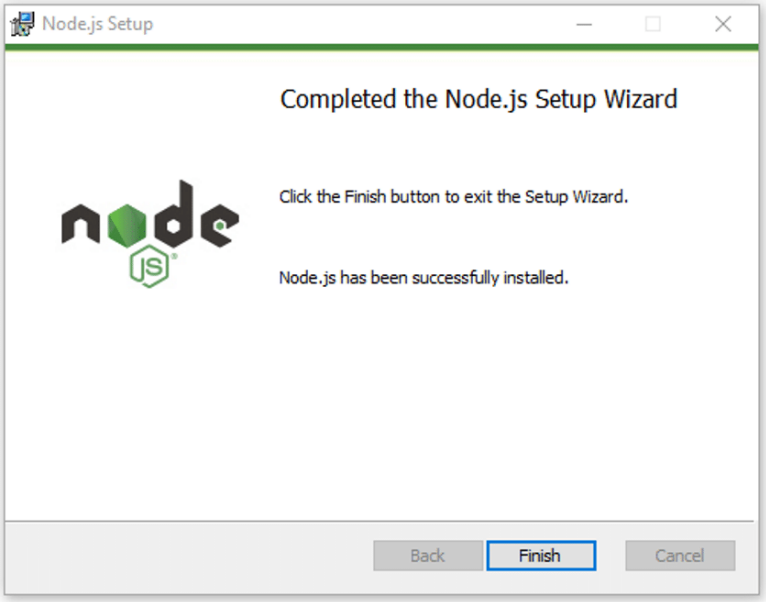




Descargar la extensión de angular:



Instalación de node js:



exitosa se puede encontrar en : https://nodejs.org/en/download

Frontend:

1. En el root del proyecto: npm install

2. Desarrollo: npm start (si usa Angular CLI en el root) o `ng serve --port 4400`.

3. Producción: ng build y servir dist/

referencias a archivos clave:

- backend/controladores/auth.controlador.js

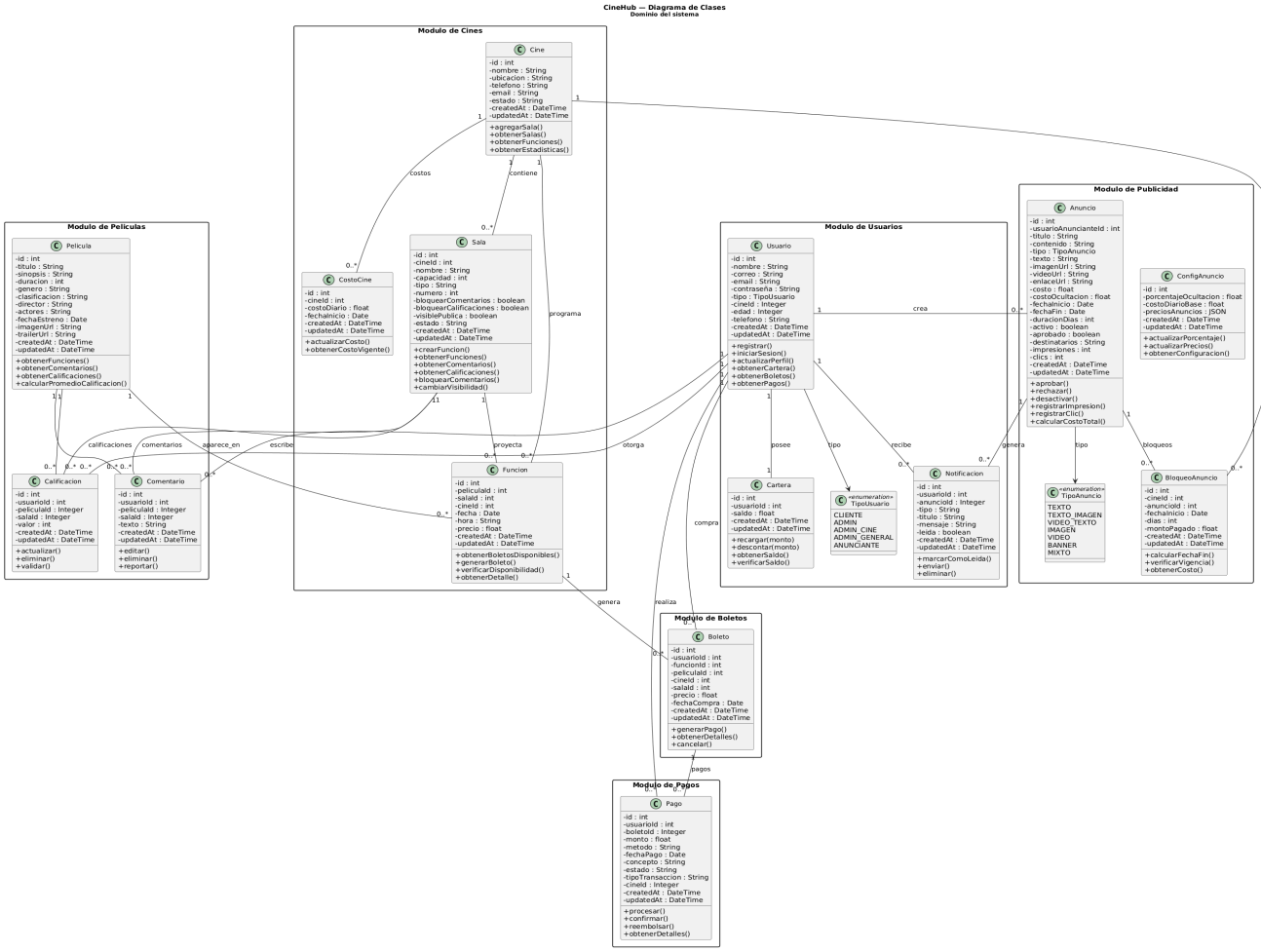
- backend/controladores/admin-cine.controlador.js

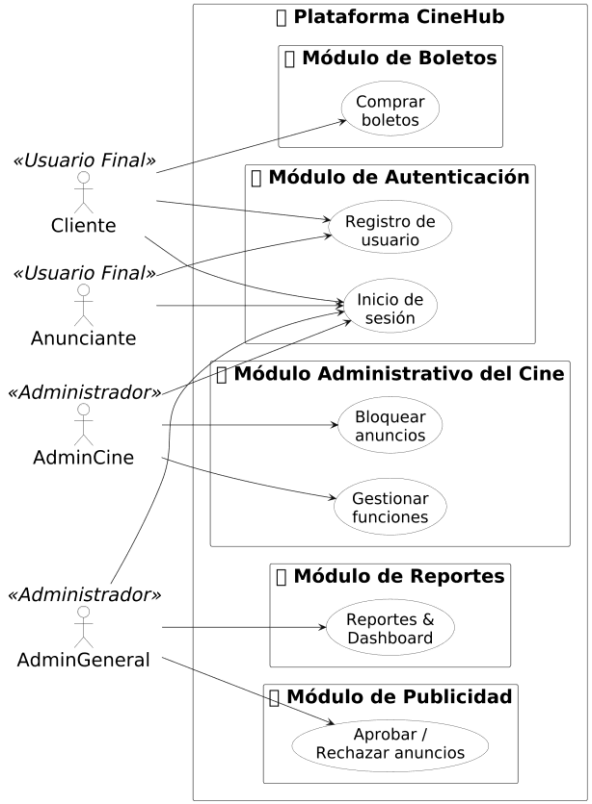
- backend/controladores/reportes-sistema.controlador.js

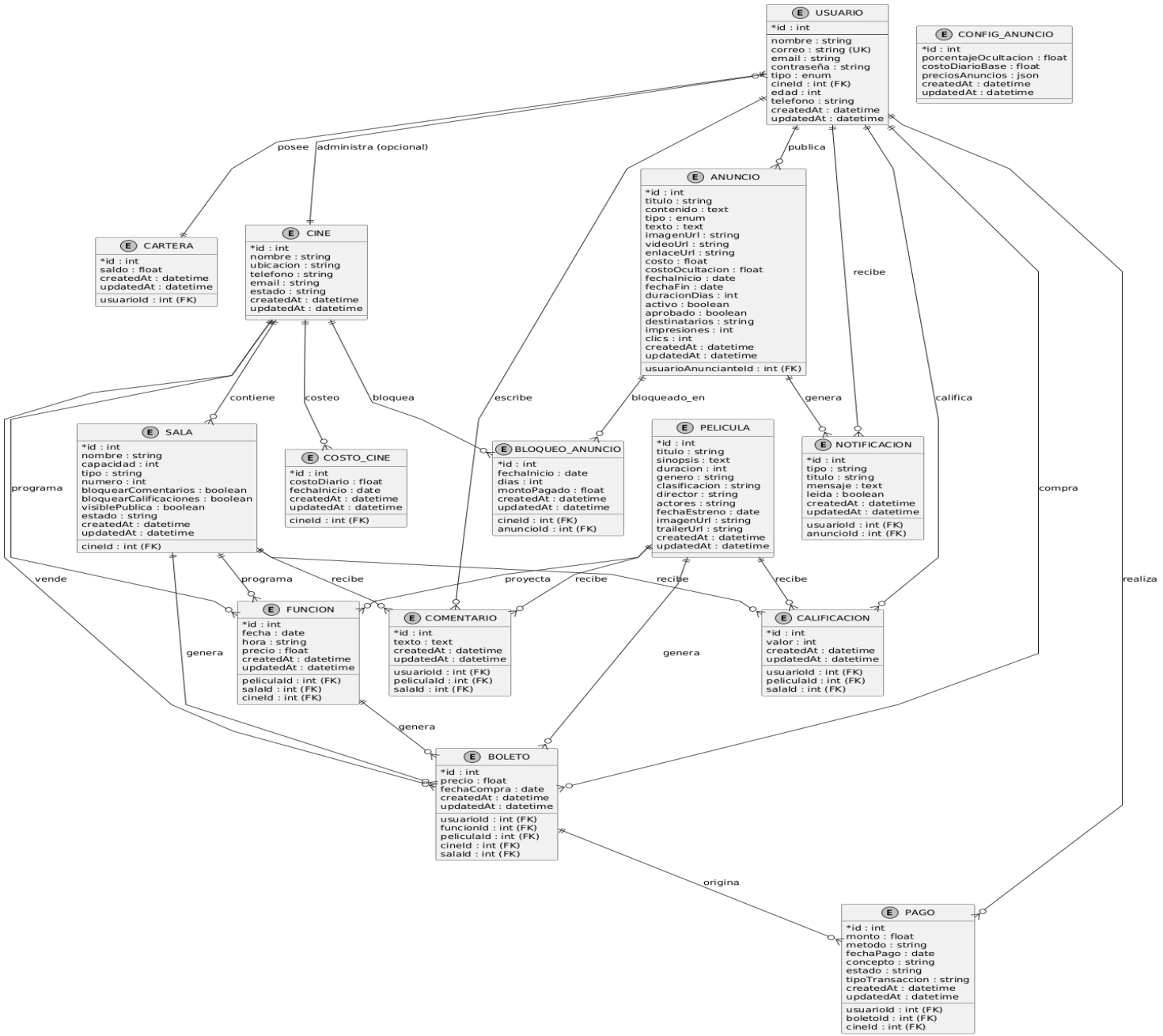
- backend/modelos/asociaciones.js

A continuación área de los diagramas 1.- diagrama de clases

Área de diagramas:




Plataforma CineHub
Módulo de Boletos
Comprar boletos
Módulo de Autenticación
Registro de usuario
Inicio de sesión
Módulo Administrativo del Cine
Bloquear anuncios
Gestionar funciones
Módulo de Reportes
Reportes & Dashboard
Módulo de Publicidad
Aprobar / Rechazar anuncios
«Usuario Final»
Cliente
«Usuario Final»
Anunciante
«Administrador»
AdminCine
«Administrador»
AdminGeneral


USUARIO
*id : int
nombre : string
correo : string (UK)
email : string
contraseña : string
tipo : enum
cineId : int (FK)
edad : int
telefono : string
createdAt : datetime
updatedAt : datetime
CONFIG_ANUNCIO
*id : int
porcentajeOcultacion : float
costoDiarioBase : float
preciosAnuncios : json
createdAt : datetime
updatedAt : datetime
CARTERA
*id : int
saldo : float
createdAt : datetime
updatedAt : datetime
usuarioId : int (FK)
CINE
*id : int
nombre : string
ubicacion : string
telefono : string
email : string
estado : string
createdAt : datetime
updatedAt : datetime
ANUNCIO
*id : int
titulo : string
contenido : text
tipo : enum
texto : text
imagenUrl : string
videoUrl : string
enlaceUrl : string
costo : float
costoOcultacion : float
fechaInicio : date
fechaFin : date
duracionDias : int
activo : boolean
aprobado : boolean
destinatarios : string
impresiones : int
clics : int
createdAt : datetime
updatedAt : datetime
usuarioAnuncianteId : int (FK)
SALA
*id : int
nombre : string
capacidad : int
tipo : string
numero : int
bloquearComentarios : boolean
bloquearCalificaciones : boolean
visiblePublica : boolean
estado : string
createdAt : datetime
updatedAt : datetime
cineId : int (FK)
COSTO_CINE
*id : int
costoDiario : float
fechaInicio : date
createdAt : datetime
updatedAt : datetime
cineId : int (FK)
BLOQUEO_ANUNCIO
*id : int
fechaInicio : date
dias : int
montoPagado : float
createdAt : datetime
updatedAt : datetime
cineId : int (FK)
anuncioId : int (FK)
PELICULA
*id : int
titulo : string
sinopsis : text
duracion : int
genero : string
clasificacion : string
director : string
actores : string
fechaEstreno : date
imagenUrl : string
trailerUrl : string
createdAt : datetime
updatedAt : datetime
NOTIFICACION
*id : int
tipo : string
titulo : string
mensaje : text
leida : boolean
createdAt : datetime
updatedAt : datetime
usuarioId : int (FK)
anuncioId : int (FK)
FUNCION
*id : int
fecha : date
hora : string
precio : float
createdAt : datetime
updatedAt : datetime
peliculaId : int (FK)
salaId : int (FK)
cineId : int (FK)
COMENTARIO
*id : int
texto : text
createdAt : datetime
updatedAt : datetime
usuarioId : int (FK)
peliculaId : int (FK)
salaId : int (FK)
CALIFICACION
*id : int
valor : int
createdAt : datetime
updatedAt : datetime
usuarioId : int (FK)
peliculaId : int (FK)
salaId : int (FK)
BOLETO
*id : int
precio : float
fechaCompra : date
createdAt : datetime
updatedAt : datetime
usuarioId : int (FK)
funcionId : int (FK)
peliculaId : int (FK)
cineId : int (FK)
salaId : int (FK)
PAGO
*id : int
monto : float
metodo : string
fechaPago : date
concepto : string
estado : string
tipoTransaccion : string
createdAt : datetime
updatedAt : datetime
usuarioId : int (FK)
boletoId : int (FK)
cineId : int (FK)
posee
administra (opcional)
publica
recibe
escribe
bloqueado_en
genera
califica
contiene
costeo
bloquea
programa
compra
vende
programa
recibe
proyecta
recibe
recibe
recibe
realiza
genera
genera
genera
origina

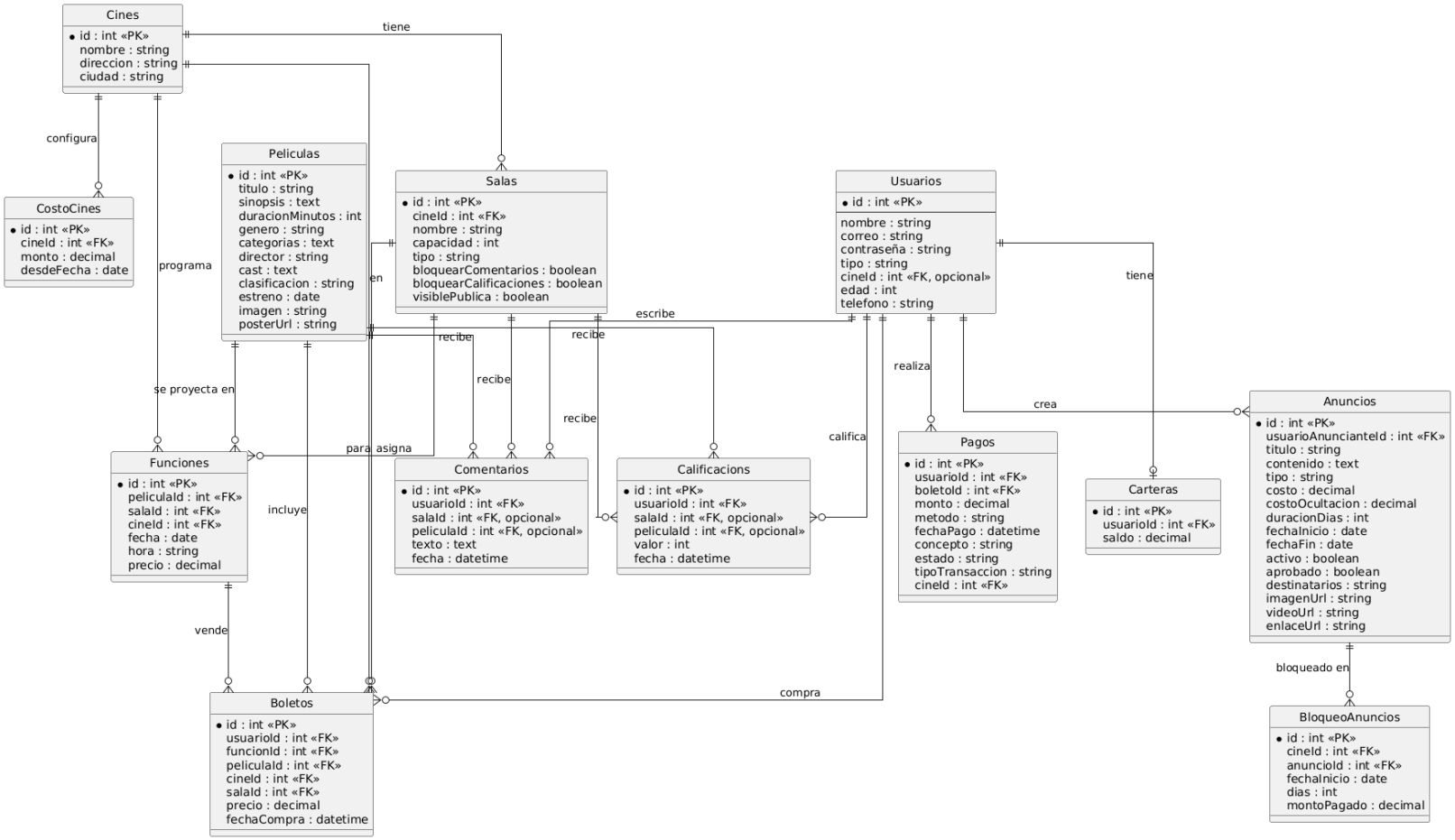


Diagrama de tablas

MAPEO FISICO DE LA DB

Mapeo Físico de la Base de Datos

modelo físico implementado en MySQL para CineHub,. Incluye tipos, claves, índices, constraints, tamaño esperado y notas por tabla.

Convenciones generales

- Motor: InnoDB (transacciones, FK reales)
- Charset/Collation: utf8mb4 / utf8mb4_unicode_ci
- PK: `INT AUTO_INCREMENT`
- Timestamps: `createdAt`, `updatedAt` con `DEFAULT CURRENT_TIMESTAMP`
- Nombres: tablas en plural PascalCase según ORM (Usuarios, Boletos, Funcions, ...)

Usuarios

- Campos: id(PK), nombre, correo(UK), email, contraseña, tipo(ENUM), cineId(FK→Cines.id), edad(CHECK 13–120), telefono, createdAt, updatedAt
- Índices: correo (UK + idx), tipo, cineId

- Consideraciones: `cineId` solo aplica a admin-cine; `correo` es clave de autenticación.

Carteras

- Campos: id(PK), usuarioId(UK, FK→Usuarios.id), saldo(CHECK ≥ 0), createdAt, updatedAt

- Índices: usuarioId (UNIQUE)

- Consideraciones: relación 1:1 con Usuario; crear al registro.

Cines

- Campos: id(PK), nombre, ubicacion, telefono, email, estado, createdAt, updatedAt

- Índices: nombre, estado

- Consideraciones: alto uso en joins con Salas/Funciones/Boletos.

Salas

- Campos: id(PK), cineId(FK→Cines), nombre, capacidad(>0), tipo, numero, bloquearComentarios, bloquearCalificaciones, visiblePublica, estado, createdAt, updatedAt

- Índices: cineId, estado, visiblePublica, UNIQUE(cineId, numero)

- Consideraciones: constraint único para numeración de salas por cine.

Peliculas

- Campos: id(PK), titulo, sinopsis, duracion(>0), genero, clasificacion, director, actores, fechaEstreno, imagenUrl, trailerUrl, createdAt, updatedAt

- Índices: titulo, genero, clasificacion, fechaEstreno

- Consideraciones: campos de texto indexados para filtros comunes.

Funcions (Funciones)

- Campos: id(PK), peliculaId(FK), salaId(FK nullable), cineId(FK), fecha, hora, precio(≥0), createdAt, updatedAt

- Índices: peliculaId, salaId, cineId, fecha, (fecha, hora), UNIQUE(salaId, fecha, hora)

- Consideraciones: evitar solapamiento de funciones en la misma sala/fecha/hora.

Boletos

- Campos: id(PK), usuarioId(FK), funcionId(FK), peliculaId(FK), cineId(FK), salaId(FK nullable), precio(≥0), fechaCompra, createdAt, updatedAt

- Índices: usuarioId, funcionId, peliculaId, cineId, fechaCompra

- Consideraciones: volumen alto; índices críticos para reportes y auditoría.

Pagos

- Campos: id(PK), usuarioId(FK), boletoId(FK nullable), monto(≥0), metodo, fechaPago, concepto, estado, tipoTransaccion, cineId(FK nullable), createdAt, updatedAt

- Índices: usuarioId, boletoId, fechaPago, estado, tipoTransaccion, cineId

- Consideraciones: fuente para ganancias; `estado` y `tipoTransaccion` mejoran filtros.

Anuncios

- Campos: id(PK), usuarioAnunciantеId(FK→Usuarios), titulo, contenido, tipo(ENUM), texto, imagenUrl, videoUrl, enlaceUrl, costo, costoOcultacion, fechaInicio, fechaFin, duracionDias, activo, aprobado, destinatarios(ENUM), impresiones, clics, createdAt, updatedAt

- Índices: usuarioAnunciantеId, tipo, activo, aprobado, (activo, aprobado), (fechaInicio, fechaFin)

- Consideraciones: control por estado, métricas para KPIs.

BloqueoAnuncios

- Campos: id(PK), cineId(FK), anuncioId(FK), fechaInicio, dias(>0), montoPagado(≥0), createdAt, updatedAt

- Índices: cineId, anuncioId, fechaInicio, UNIQUE(cineId, anuncioId)

- Consideraciones: evita bloqueos duplicados por cine/anuncio.

Comentarios

- Campos: id(PK), usuarioId(FK), peliculaId(FK nullable), salaId(FK nullable), texto, createdAt, updatedAt

- Índices: usuarioId, peliculaId, salaId

- Consideraciones: regla CHECK (peliculaId IS NOT NULL OR salaId IS NOT NULL).

Calificacions (Calificaciones)

- Campos: id(PK), usuarioId(FK), peliculaId(FK nullable), salaId(FK nullable), valor(1–5), createdAt, updatedAt

- Índices: usuarioId, peliculaId, salaId, valor, UNIQUE(usuarioId, peliculaId), UNIQUE(usuarioId, salaId)

- Consideraciones: una calificación por usuario y objetivo.

CostoCines

- Campos: id(PK), cineId(FK), costoDiario(≥0), fechaInicio(NULL para legacy), createdAt, updatedAt

- Índices: cineId, fechaInicio

- Consideraciones: compatibilidad con registros antiguos (fechaInicio nullable).

ConfigAnuncios

- Campos: id(PK), porcentajeOcultacion(0–100), costoDiarioBase(≥0), preciosAnuncios(JSON), createdAt, updatedAt

- Índices: ninguno obligatorio (tabla singleton)

- Consideraciones: usar solo el registro más reciente.

Notificacions

- Campos: id(PK), usuarioId(FK), anuncioId(FK nullable), tipo, titulo, mensaje, leida, createdAt, updatedAt

- Índices: usuarioId, anuncioId, tipo, leida, (usuarioId, leida)

- Consideraciones: notificaciones legibles/pendientes rápidas de consultar.

Recomendaciones de mantenimiento y performance

1. Ejecutar ANALYZE/OPTIMIZE TABLE mensual en tablas con alto churn (Pagos, Boletos).

2. Particionar lógicamente reportes por rango de fechas (índices por fecha).

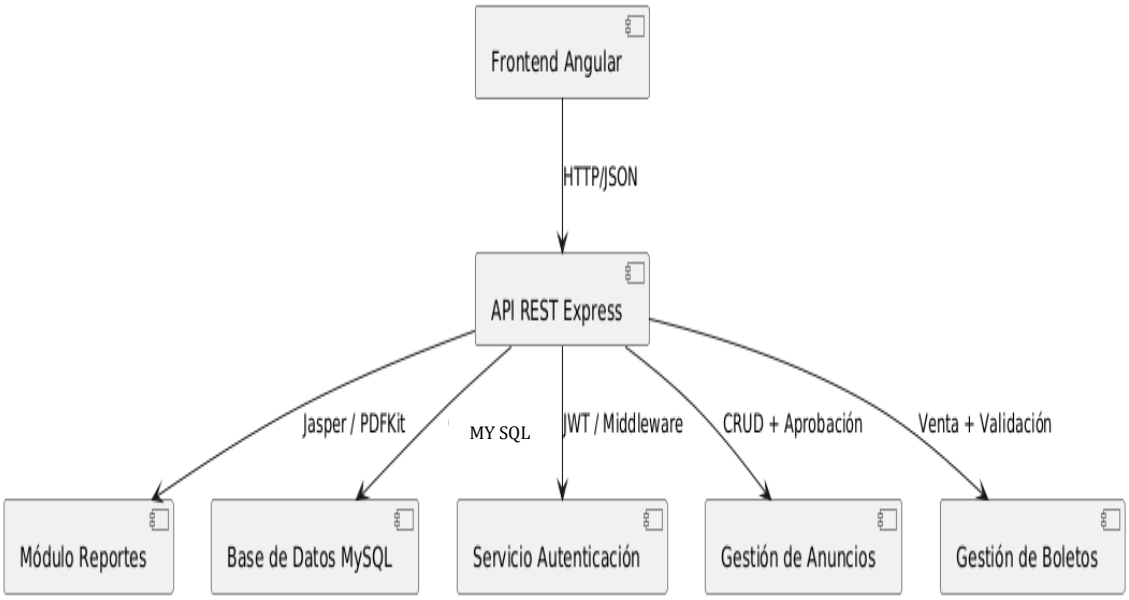
3. Usar transacciones en operaciones compuestas (compra boleto, bloqueo anuncio, registro usuario).

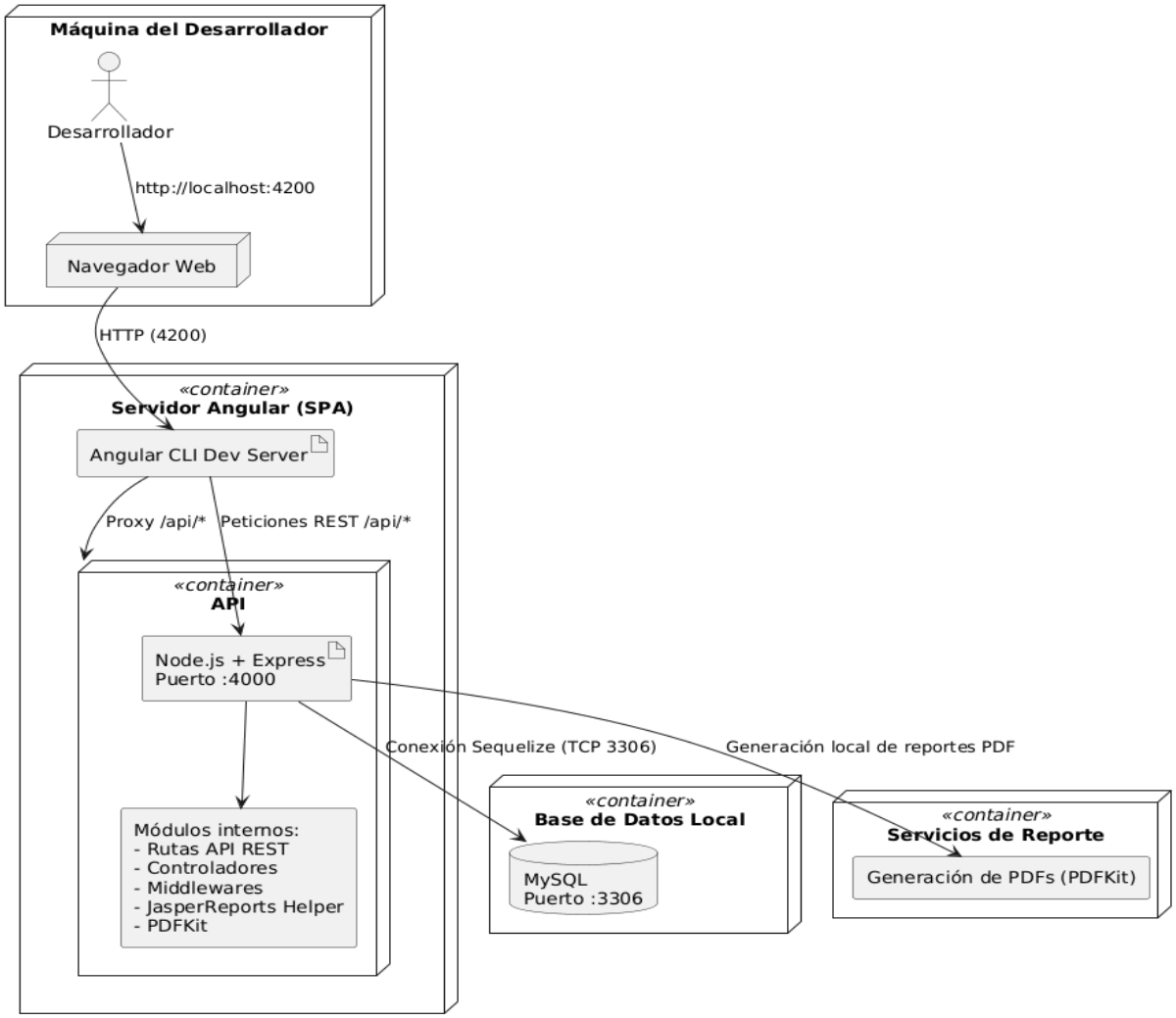
4. Auditoría: createdAt/updatedAt ya incluidos; agregar `deletedAt` si se requiere soft-delete.

5. Backups: completos diarios + binlogs para recuperación puntual.

6. Monitoreo: alertas por crecimiento anómalo de métricas (Pagos/Boletos/Anuncios).

DIAGRAMA DE COMPONENTES




Máquina del Desarrollador
Desarrollador
http://localhost:4200
Navegador Web
HTTP (4200)
«container»
Servidor Angular (SPA)
Angular CLI Dev Server
Proxy /api/*
Peticiones REST /api/*
«container»
API
Node.js + Express
Puerto :4000
Módulos internos:
- Rutas API REST
- Controladores
- Middlewares
- JasperReports Helper
- PDFKit
Conexión Sequelize (TCP 3306)
Generación local de reportes PDF
«container»
Base de Datos Local
MySQL
Puerto :3306
«container»
Servicios de Reporte
Generación de PDFs (PDFKit)

Requisitos del Sistema

Software Requerido

Sistema Operativo:

Windows 10/11

Ubuntu 20.04 LTS o superior

macOS 12 (Monterey) o superior

Node.js:

Versión: 18.x o superior

Descarga: https://nodejs.org/

MySQL:

Versión: 8.0 o superior

Descarga: https://dev.mysql.com/downloads/mysql/

Angular CLI:

Versión: 20.x (se instala automáticamente con el proyecto)

Incluido en dependencias del proyecto

verificar versiones:

Verificar Node.js

node --version

Debe mostrar: v18.x.x o superior

Verificar npm

npm --version

Debe mostrar: 9.x.x o superior

Verificar MySQL

mysql --version

Debe mostrar: mysql Ver 8.0.x o superior