Árbol de Carpetas (PHP y MySQL):

.

├── .env.example

├── .gitignore

├── composer.json

├── composer.lock

├── index.php

│

├── app

│ ├── config

│ │ └── db.php

│ │

│ ├── controllers

│ │ ├── AsientosController.php

│ │ ├── AuthController.php

│ │ ├── ComprasController.php

│ │ ├── FuncionesController.php

│ │ ├── PagosController.php

│ │ ├── PeliculasController.php

│ │ └── TicketsController.php

│ │

│ ├── helpers

│ │ ├── auth.php

│ │ ├── mailer.php

│ │ └── qr.php

│ │

│ └── models

│ ├── Asiento.php

│ ├── Cine.php

│ ├── Compra.php

│ ├── Funcion.php

│ ├── Pago.php

│ ├── Pelicula.php

│ ├── Sala.php

│ ├── Ticket.php

│ └── Usuario.php

│

├── public

│ ├── asientos.php

│ ├── carrito.php

│ ├── catalogo.php

│ ├── confirmacion.php

│ ├── funcion.php

│ ├── index.php

│ ├── login.php

│ ├── logout.php

│ ├── pago.php

│ ├── perfil.php

│ └── validar.php

│ │

│ ├── admin

│ │ ├── cines.php

│ │ ├── funciones.php

│ │ ├── peliculas.php

│ │ └── salas.php

│ │

│ ├── assets

│ │ └── (CSS, JS, imágenes…)

│ │

│ ├── partials

│ │ ├── footer.php

│ │ └── header.php

│ │

│ └── qr

│ └── (archivos PNG generados por QR)

│

├── sql

│ └── ticketera\_recompensas.sql

│

├── tests

│ └── (scripts de prueba)

│

└── vendor

├── autoload.php

├── chillerlan

│ ├── php-qrcode

│ └── php-settings-container

├── composer

└── phpmailer

Codificación por fases de proyecto Ticketera

**Supuestos Técnicos**

* PHP 8.x + MySQL (PDO), sesiones nativas.
* Composer para librerías: phpmailer/phpmailer y chillerlan/php-qrcode (o phpqrcode clásico).
* Seguridad: consultas preparadas, saneo de salida, validación backend + frontend.
* Rutas públicas en public/\*.php, lógica en app/controllers, app/models, utilidades en app/helpers, conexión en app/config/db.php.

**Fase 0 — Setup & DB**

* Objetivo: Conexión DB y bootstrap del proyecto.
* Entregables:
  + app/config/db.php (PDO singleton).
  + Helper de sesiones y roles en app/helpers/auth.php.
  + Carga de Composer y autoload básico en index.php y public/index.php.
* Validaciones:
  + Conexión a ticketing\_mvp\_final.
  + Semillas mínimas (roles, niveles) ya en SQL.
* Nota: Ajuste menor sugerido en pagos.metodo para incluir ‘TARJETA’ (o usar ‘STRIPE’ como sandbox).

**Fase 1 — Autenticación y Usuarios**

* Objetivo: Registro, login y control de roles.
* Entregables:
  + app/models/Usuario.php (CRUD básico + auth).
  + app/controllers/AuthController.php (registro/login/logout).
  + public/login.php (formulario y manejo de sesión).
  + app/helpers/auth.php (middleware: requireLogin/requireRole).
* Validaciones:
  + password\_hash/verify, sesiones $\_SESSION['usuario\_id'], $\_SESSION['rol'].
  + Rol por defecto: CLIENTE; acceso a validar.php solo AGENTE.
* Perfil (mínimo): endpoint básico para ver puntos e historial (marco para Fase 10).

**Fase 2 — Catálogo de Películas**

* Objetivo: Listado con filtros por ciudad/cine.
* Entregables:
  + app/models/Pelicula.php, app/models/Funcion.php, app/models/Asiento.php (métodos de lectura).
  + app/controllers/PeliculasController.php.
  + public/catalogo.php (lista + filtro ciudad/cine + “Seleccionar función”).
* Validaciones:
  + Muestra funciones por película, con datos de cine/sala.
  + Navega a public/funcion.php?id=....

**Fase 3 — Selección de Función**

* Objetivo: Vista detalle de función.
* Entregables:
  + app/controllers/FuncionesController.php.
  + public/funcion.php (película, cine, sala, fecha/hora, precio, asientos disponibles).
* Validaciones:
  + Calcula asientos disponibles contra reservas\_asientos.
  + Botón “Seleccionar asientos” a public/asientos.php?funcion\_id=....

**Fase 4 — Selección de Asientos**

* Objetivo: Grid simple por sala y selección múltiple.
* Entregables:
  + app/controllers/AsientosController.php.
  + public/asientos.php (render grid, marcar ocupados, seleccionar).
* Validaciones:
  + Guarda selección en $\_SESSION['carrito'] (funcion\_id, asientos).
  + Previene doble selección si otro usuario reservó (validación al confirmar).

**Fase 5 — Carrito y Pre-Compra**

* Objetivo: Resumen y opción de canje de puntos.
* Entregables:
  + app/models/Compra.php.
  + app/controllers/ComprasController.php.
  + Vista embebida en public/pago.php o página intermedia (resumen).
* Validaciones:
  + Muestra película/función/asientos/subtotal.
  + Muestra puntos del usuario; habilita canje si equivalen a ≥ Q100.
  + Al confirmar: crea compras (PENDIENTE) + compra\_detalle (sin pagos aún).

**Fase 6 — Pago (Validaciones de Tarjeta)**

* Objetivo: Formulario y validaciones.
* Entregables:
  + app/models/Pago.php, app/controllers/PagosController.php.
  + public/pago.php (form).
* Validaciones:
  + Número: 16 dígitos numéricos exactos.
  + CVV: 3 dígitos numéricos.
  + Vencimiento: MM/YY (input con máscara + validación backend).
  + Sandbox: inserta en pagos (PENDIENTE → APROBADO automático).
  + Si aprobado: compras.estado = 'PAGADO', aplica canje y calcula acumulación.

**Fase 7 — Tickets + QR**

* Objetivo: Generar ticket por asiento con QR único.
* Entregables:
  + app/models/Ticket.php, app/controllers/TicketsController.php.
  + app/helpers/qr.php (generación QR, ruta de storage public/qr/).
* Validaciones:
  + QR con payload firmado: ticket\_id, hash/HMAC y datos mínimos (opcional: info de canje).
  + Persistencia en tickets con codigo\_qr único por asiento.

**Fase 8 — Confirmación + Email**

* Objetivo: Confirmación y notificación.
* Entregables:
  + app/helpers/mailer.php (PHPMailer).
  + public/confirmacion.php (resumen + links/adjuntos QR).
* Validaciones:
  + Muestra “Compra exitosa” y detalle.
  + Envía email al cliente con QR adjunto o link de descarga.

**Fase 9 — Validación de Tickets (Agente)**

* Objetivo: Escaneo/consulta y marcado de validación.
* Entregables:
  + public/validar.php (solo rol AGENTE).
  + Reutiliza TicketsController.php.
* Validaciones:
  + Recibe ticket\_id o payload del QR.
  + Verifica existencia, compra PAGADA, validado = FALSE.
  + Marca validado = TRUE, set validado\_por, validado\_en.
  + Respuesta clara: ✅ válido / ❌ inválido (mensajes de motivo).

**Fase 10 — Sistema de Puntos**

* Objetivo: Acumulación/Canje con historial.
* Entregables:
  + Métodos en Usuario.php y/o Compra.php para sumar/restar puntos.
  + Modelo Puntos (si separamos) o integrar en Compra.
  + Recalcular nivel según niveles\_fidelizacion.
* Validaciones:
  + Movimientos en puntos\_movimientos (ACUMULACION/CANJE) vinculados a compra.
  + Actualiza usuarios.puntos y usuarios.nivel\_id.

**Fase 11 — Admin (CRUD mínimo)**

* Objetivo: Gestión básica para operar el MVP.
* Entregables:
  + CRUD para peliculas, cines, salas, funciones (páginas simples).
* Validaciones:
  + Solo rol ADMIN; formularios con validación básica.
  + Relaciones integradas (funciones enlazan película/sala).

**Validaciones Clave de Pago**

* Número tarjeta: regex ^[0-9]{16}$ + Luhn opcional.
* CVV: ^[0-9]{3}$.
* Vencimiento: ^(0[1-9]|1[0-2])\\/[0-9]{2}$ y no pasado.
* Manejo de errores UX y validación backend estricta.

**Cierre del Flujo**

* Login → Catálogo → Función → Asientos → Carrito → Pago → Tickets/QR → Confirmación/Email → Validación (Agente) → Puntos actualizados.

**Credenciales Usuarios:**

Admin: [admin@example.com](mailto:admin@example.com) / admin123456

Agente: [agente@example.com](mailto:agente@example.com) / 123456

Cliente: [cliente1@example.com](mailto:cliente1@example.com) / 123456

* Tarjetas de prueba (cualquiera de estas funciona)
  + Número: 4111111111111111 o 4242424242424242 o 5555555555554444 o 5105105105105100
  + CVV: 123
  + Vencimiento: 12/29 (cualquier fecha futura en formato MM/YY)

**Documentación base de datos:**

-- ==========================

-- BASE DE DATOS

-- ==========================

CREATE DATABASE ticketing\_mvp\_final;

USE ticketing\_mvp\_final;

-- ==========================

-- ROLES

-- ==========================

CREATE TABLE roles (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(50) NOT NULL UNIQUE

);

INSERT INTO roles (nombre) VALUES ('CLIENTE'), ('AGENTE'), ('ADMIN');

-- ==========================

-- NIVELES DE FIDELIZACION

-- ==========================

CREATE TABLE niveles\_fidelizacion (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre ENUM('BRONCE','SILVER','GOLD') UNIQUE,

min\_puntos INT NOT NULL

);

INSERT INTO niveles\_fidelizacion (nombre, min\_puntos)

VALUES ('BRONCE', 0), ('SILVER', 500), ('GOLD', 1000);

-- ==========================

-- USUARIOS

-- ==========================

CREATE TABLE usuarios (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(100) NOT NULL,

correo VARCHAR(150) NOT NULL UNIQUE,

contrasena\_hash VARCHAR(255) NOT NULL,

rol\_id INT NOT NULL,

nivel\_id INT NOT NULL,

puntos INT NOT NULL DEFAULT 0,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

ultima\_conexion TIMESTAMP NULL,

FOREIGN KEY (rol\_id) REFERENCES roles(id),

FOREIGN KEY (nivel\_id) REFERENCES niveles\_fidelizacion(id)

);

-- ==========================

-- CINES

-- ==========================

CREATE TABLE cines (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(150) NOT NULL,

ciudad VARCHAR(100) NOT NULL,

direccion VARCHAR(255)

);

-- ==========================

-- SALAS

-- ==========================

CREATE TABLE salas (

id INT AUTO\_INCREMENT PRIMARY KEY,

cine\_id INT NOT NULL,

nombre VARCHAR(50) NOT NULL,

capacidad INT NOT NULL,

FOREIGN KEY (cine\_id) REFERENCES cines(id)

);

-- ==========================

-- ASIENTOS

-- ==========================

CREATE TABLE asientos (

id INT AUTO\_INCREMENT PRIMARY KEY,

sala\_id INT NOT NULL,

fila VARCHAR(5) NOT NULL,

numero INT NOT NULL,

UNIQUE (sala\_id, fila, numero),

FOREIGN KEY (sala\_id) REFERENCES salas(id)

);

-- ==========================

-- PELICULAS

-- ==========================

CREATE TABLE peliculas (

id INT AUTO\_INCREMENT PRIMARY KEY,

titulo VARCHAR(150) NOT NULL,

descripcion TEXT,

duracion INT, -- minutos

clasificacion VARCHAR(10), -- PG, R, etc.

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- ==========================

-- FUNCIONES (pelicula + sala + horario)

-- ==========================

CREATE TABLE funciones (

id INT AUTO\_INCREMENT PRIMARY KEY,

pelicula\_id INT NOT NULL,

sala\_id INT NOT NULL,

fecha\_funcion DATETIME NOT NULL,

precio DECIMAL(10,2) NOT NULL,

FOREIGN KEY (pelicula\_id) REFERENCES peliculas(id),

FOREIGN KEY (sala\_id) REFERENCES salas(id)

);

-- ==========================

-- EVENTOS (no peliculas)

-- ==========================

CREATE TABLE eventos (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(150) NOT NULL,

descripcion TEXT,

ubicacion VARCHAR(150) NOT NULL,

fecha\_evento DATETIME NOT NULL,

precio DECIMAL(10,2) NOT NULL,

capacidad\_total INT NOT NULL,

tickets\_disponibles INT NOT NULL,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- ==========================

-- COMPRAS

-- ==========================

CREATE TABLE compras (

id INT AUTO\_INCREMENT PRIMARY KEY,

usuario\_id INT NOT NULL,

precio\_total DECIMAL(10,2) NOT NULL,

puntos\_usados INT NOT NULL DEFAULT 0,

estado ENUM('PENDIENTE','PAGADO','CANCELADO') DEFAULT 'PENDIENTE',

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (usuario\_id) REFERENCES usuarios(id)

);

-- ==========================

-- DETALLE DE COMPRA

-- ==========================

CREATE TABLE compra\_detalle (

id INT AUTO\_INCREMENT PRIMARY KEY,

compra\_id INT NOT NULL,

tipo ENUM('FUNCION','EVENTO') NOT NULL,

funcion\_id INT NULL,

evento\_id INT NULL,

cantidad INT NOT NULL,

subtotal DECIMAL(10,2) NOT NULL,

FOREIGN KEY (compra\_id) REFERENCES compras(id),

FOREIGN KEY (funcion\_id) REFERENCES funciones(id),

FOREIGN KEY (evento\_id) REFERENCES eventos(id)

);

-- ==========================

-- RESERVA DE ASIENTOS (para funciones)

-- ==========================

CREATE TABLE reservas\_asientos (

id INT AUTO\_INCREMENT PRIMARY KEY,

compra\_detalle\_id INT NOT NULL,

asiento\_id INT NOT NULL,

UNIQUE (asiento\_id, compra\_detalle\_id),

FOREIGN KEY (compra\_detalle\_id) REFERENCES compra\_detalle(id),

FOREIGN KEY (asiento\_id) REFERENCES asientos(id)

);

-- ==========================

-- TICKETS

-- ==========================

CREATE TABLE tickets (

id INT AUTO\_INCREMENT PRIMARY KEY,

compra\_detalle\_id INT NOT NULL,

codigo\_qr VARCHAR(255) NOT NULL UNIQUE,

validado BOOLEAN DEFAULT FALSE,

validado\_por INT NULL,

validado\_en TIMESTAMP NULL,

FOREIGN KEY (compra\_detalle\_id) REFERENCES compra\_detalle(id),

FOREIGN KEY (validado\_por) REFERENCES usuarios(id)

);

-- ==========================

-- PAGOS

-- ==========================

CREATE TABLE pagos (

id INT AUTO\_INCREMENT PRIMARY KEY,

compra\_id INT NOT NULL,

metodo ENUM('PAYPAL','STRIPE') NOT NULL,

referencia VARCHAR(255) NOT NULL,

monto DECIMAL(10,2) NOT NULL,

estado ENUM('APROBADO','FALLIDO','PENDIENTE') DEFAULT 'PENDIENTE',

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (compra\_id) REFERENCES compras(id)

);

-- ==========================

-- MOVIMIENTOS DE PUNTOS

-- ==========================

CREATE TABLE puntos\_movimientos (

id INT AUTO\_INCREMENT PRIMARY KEY,

usuario\_id INT NOT NULL,

compra\_id INT NULL,

tipo ENUM('ACUMULACION','CANJE') NOT NULL,

puntos INT NOT NULL,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (usuario\_id) REFERENCES usuarios(id),

FOREIGN KEY (compra\_id) REFERENCES compras(id)

);