PROYECTO:

Apartamentos Turísticos CyL

Realizado por:

EQUIPO: YAR

- ***Yeiner Freyler Bonett Rodríguez***
- ***Alex González Suárez***
- ***Ángel Rolando Gutiérrez Hurtado***

Desarrollo de Aplicaciones Web (DAW)

Valladolid, a 5 de Febrero de 2026

Índice

1. Introducción

El **Proyecto Intermodular** del segundo curso del **Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Web** tiene como finalidad integrar de manera práctica los conocimientos adquiridos durante el ciclo mediante el desarrollo de una aplicación web **completa, funcional y documentada.**

Este documento recoge el **análisis, diseño, implementación, planificación y evaluación** de "YAR", una plataforma web orientada a la **consulta y reserva de apartamentos turísticos de Castilla y León**, construida a partir de **datos abiertos oficiales de la Junta de Castilla y León.**

La aplicación surge para dar respuesta a la necesidad de **centralizar la oferta de alojamiento turístico de la región**, facilitando a los usuarios una experiencia de búsqueda y reserva **intuitiva y eficiente**. El proyecto aprovecha la **API REST de datos abiertos** del gobierno autonómico, que proporciona información actualizada sobre el registro oficial de apartamentos turísticos.

El desarrollo del proyecto ha requerido la aplicación conjunta de tecnologías de **backend (PHP 8.x, MySQL), frontend (HTML5, CSS3, JavaScript ES6+)**, visualización de datos (Leaflet.js para mapas interactivos), gestión de **APIs REST**, control de versiones con Git, y despliegue web, siguiendo buenas prácticas de desarrollo y principios de **diseño de software limpio y mantenible**.

El resultado final es una aplicación **modular, escalable y funcional** que cumple los requisitos técnicos exigidos, integrando conceptos de **programación orientada a objetos, patrón DAO (Data Access Object), arquitectura por capas, seguridad web** (validaciones, protección contra inyección SQL, hash de contraseñas con bcrypt), y **diseño responsive.**

2. Análisis

2.1Dataset utilizado

Nombre del data set: Registro de Turismo de Castilla y León – Apartamentos Turísticos.

Fuente: API REST de Datos Abiertos de la Junta de Castilla y León.

URL: https://datosabiertos.jcyl.es/web/es/api-de-datos/registro-turismo-castilla-leon-apartamentos-turisticos.

Formato: JSON mediante API REST.

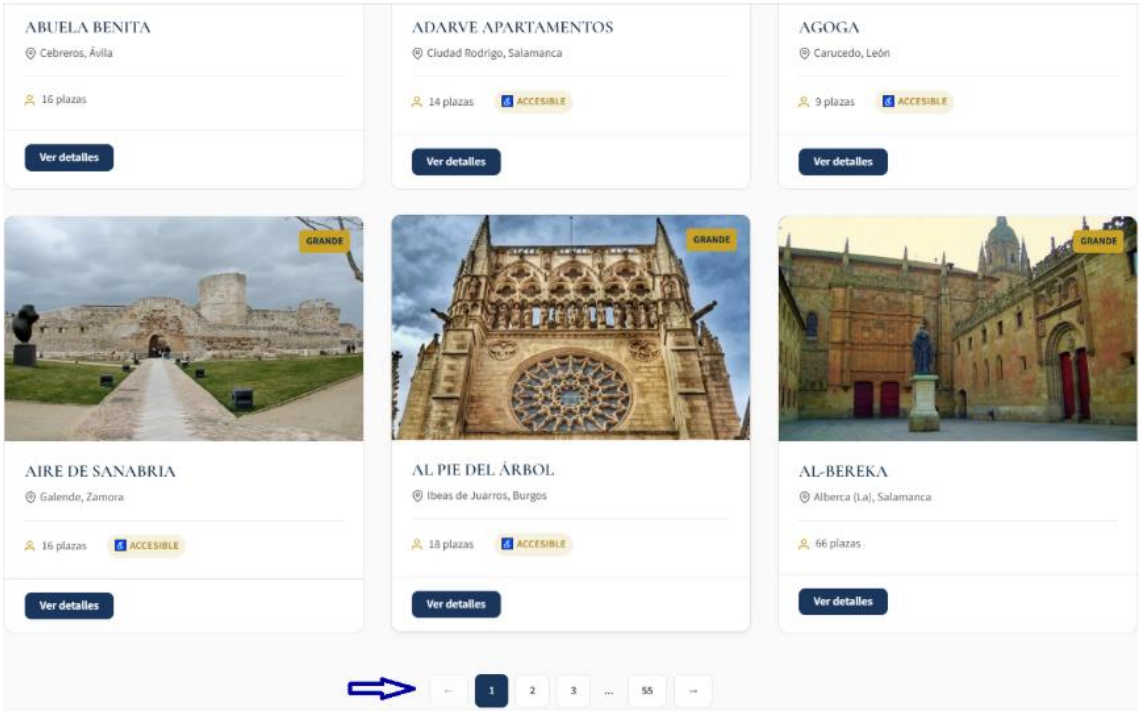
Campos principales utilizados del dataset:

- **n_registro**: Identificador único del establecimiento en el registro oficial.
- **nombre:** Denominación oficial del apartamento turístico.
- **provincia:** Provincia de ubicación (9 provincias de Castilla y León).
- **municipio:** Municipio de ubicación.
- **localidad:** Localidad específica dentro del municipio.
- **dirección:** Dirección postal completa.
- **Código postal:** Código postal del establecimiento.
- **telefono_1/2/3:** Números de contacto del establecimiento.
- **email:** Dirección de correo electrónico.
- **web:** Página web del establecimiento.
- **plazas:** Capacidad máxima de ocupantes.
- **categoría:** Clasificación oficial del apartamento.
- **accesible:** Indicador de adaptación para personas con movilidad reducida.
- **calidad**: Distintivo de calidad turística.
- **gps_latitud, gps_longitud:** Coordenadas geográficas (WGS84) para visuali zación en mapa.
- **especialidades:** Características especiales del alojamiento.

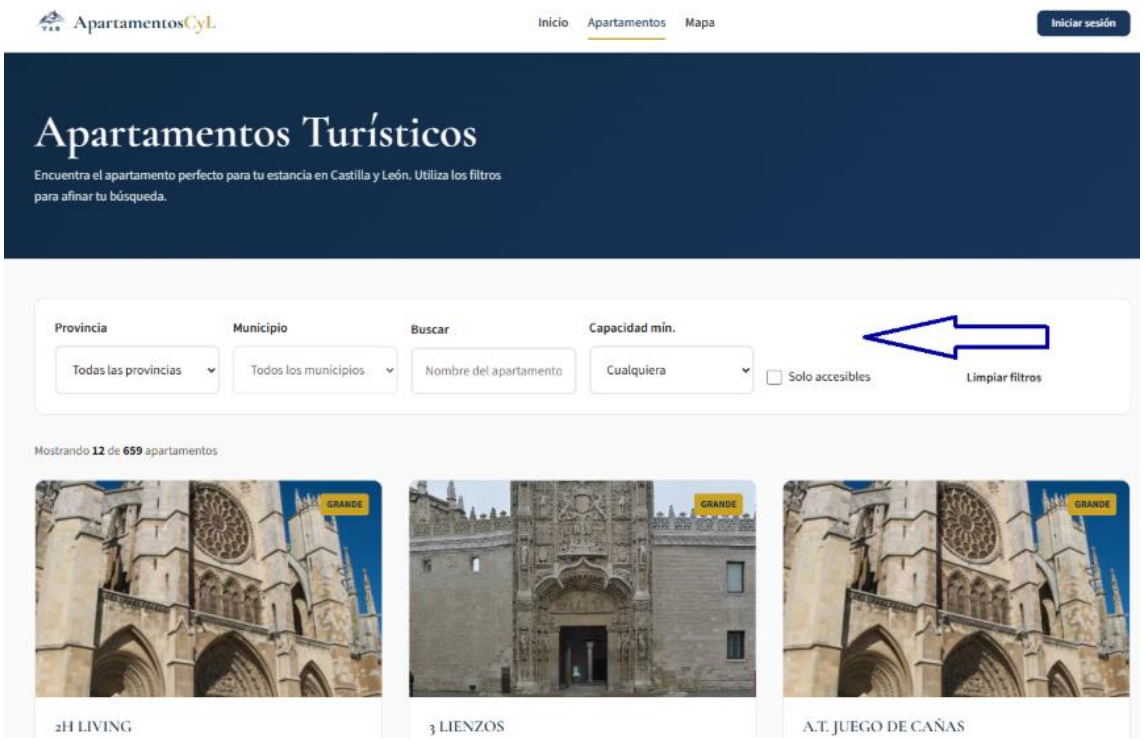
2.2 Requisitos funcionales

2.2.1 Funcionalidades para usuario no autentificado (Visitante)

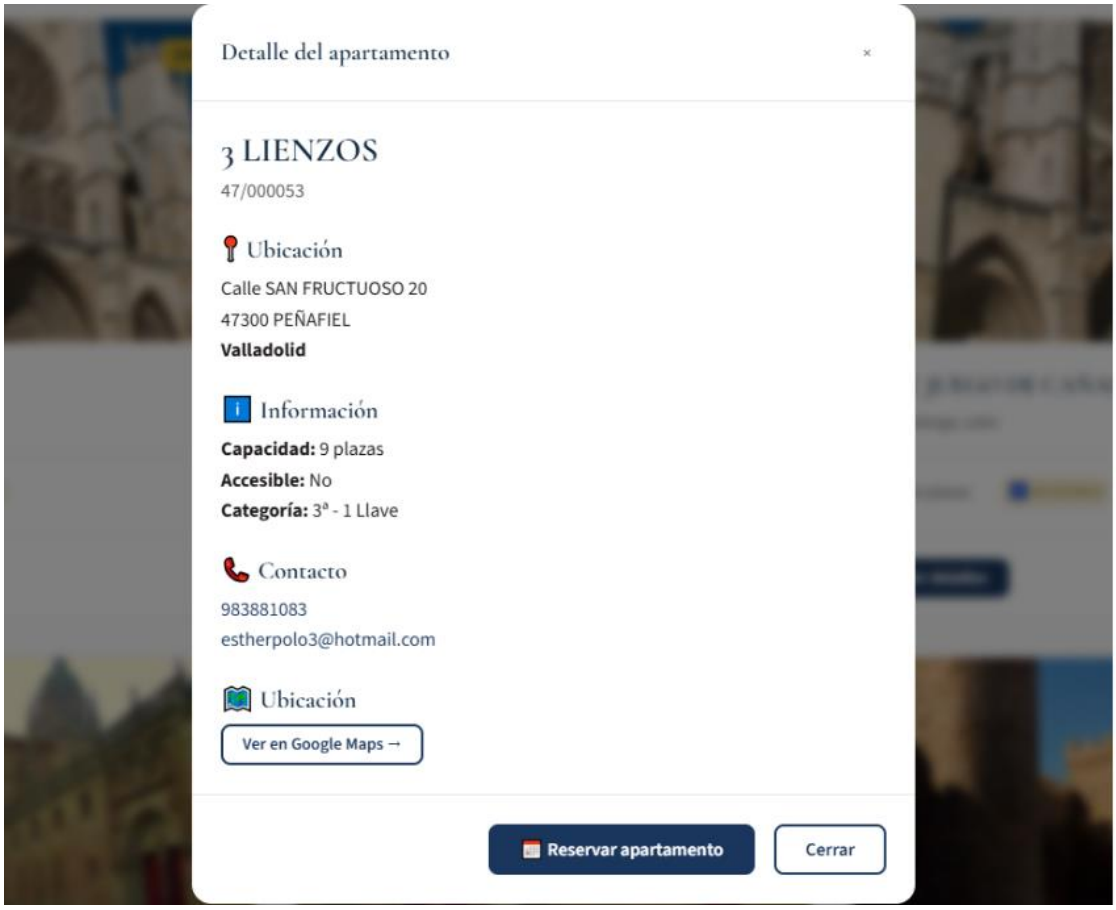
- Visión del catálogo completo de apartamentos con sistema de paginación.



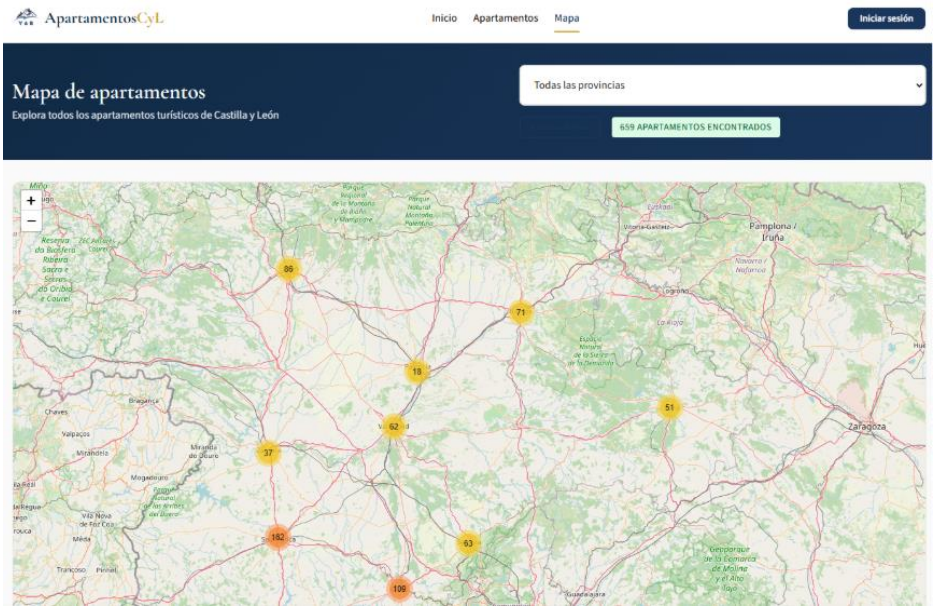
- Sistema de filtrado multicriterio: provincia, municipio, número de plazas, categoría y accesibilidad.



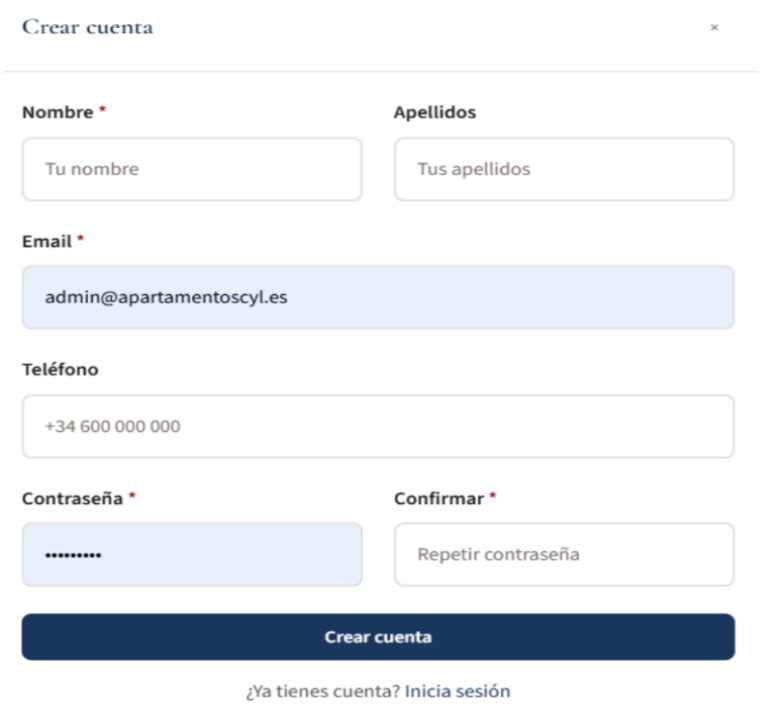
- Visualización detallada de la ficha completa de cada apartamento.



- Mapa interactivo con geolocalización usando Leaflet.js.



- Sistema de registro de nuevos usuarios con validación en dos capas.



- Funcionalidad de inicio de sesión.

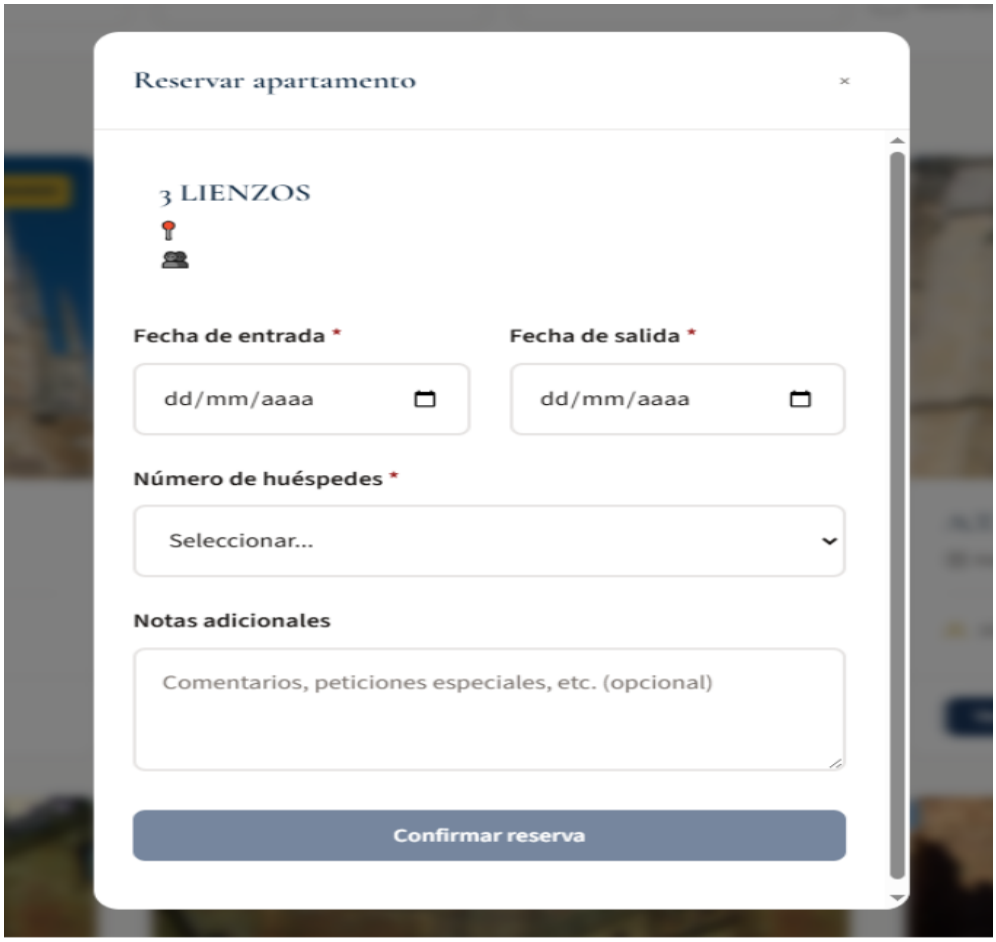
- Acceso a gráfica de estadísticas.

- Navegación responsive adaptada a dispositivos móviles, tablets y ordenado res.

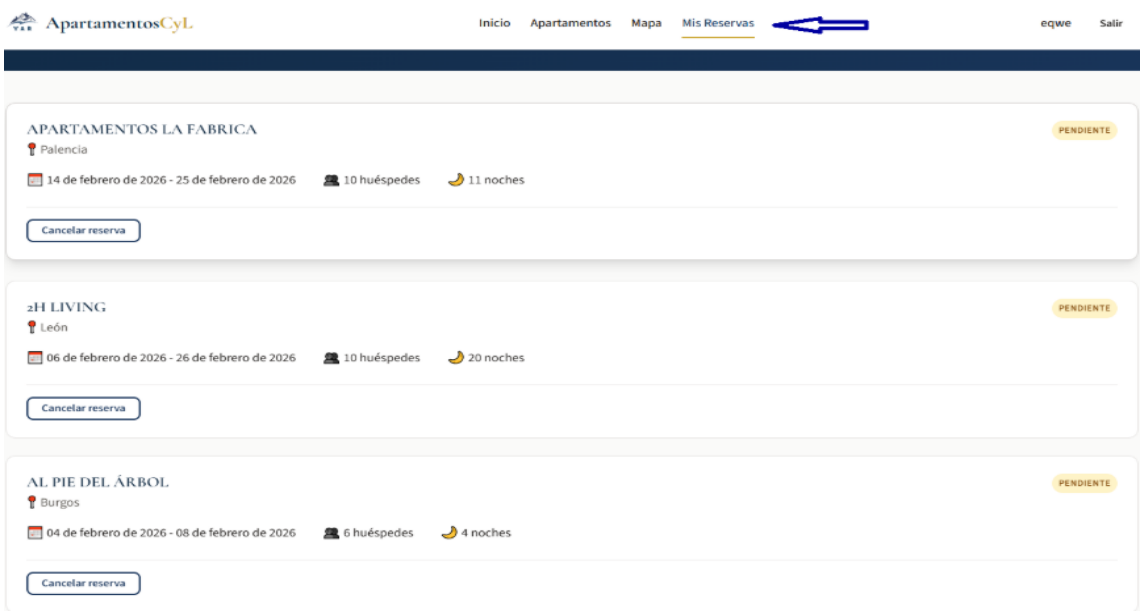
2.2.2 Funcionalidades para usuario autentificado (Cliente)

Todas las funcionalidades del visitante, más:

- Realización de reservas con selección de fechas y validación de disponibili dad.

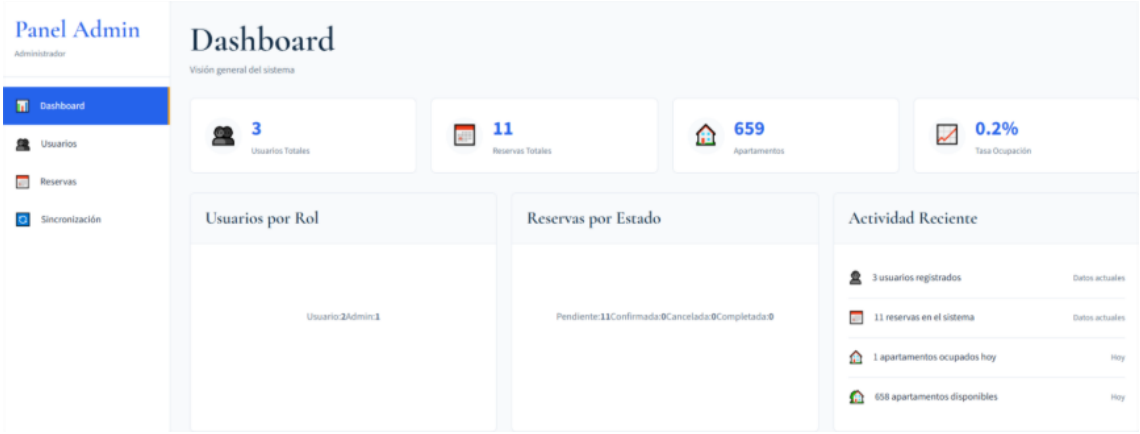


- Panel personalizado 'Mis Reservas'.

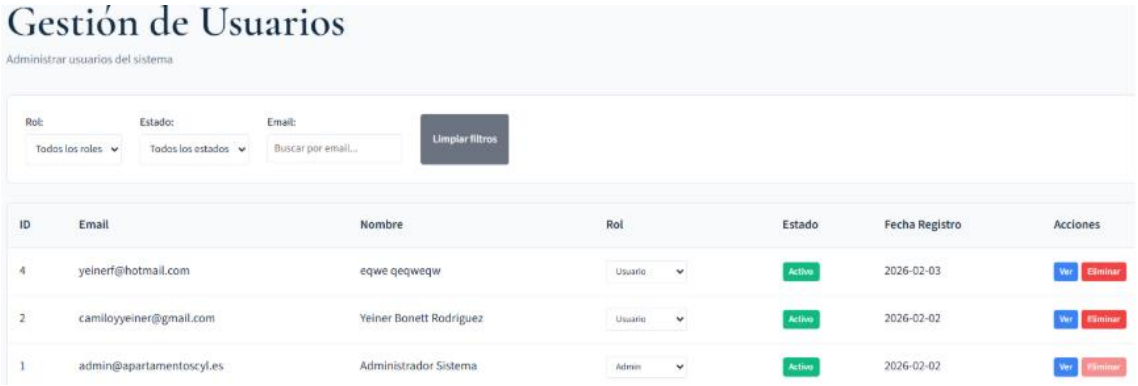


2.2.3 Funcionalidades para administrador

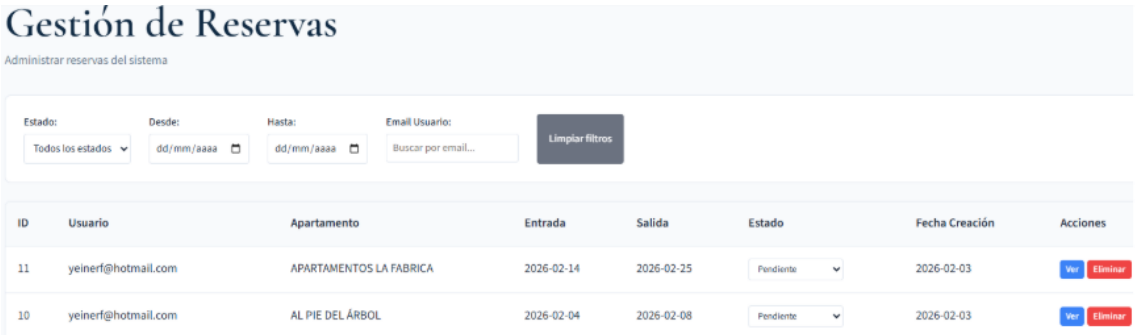
- Todas las funcionalidades del cliente, más:
- Panel de control con dashboard de métricas del sistema



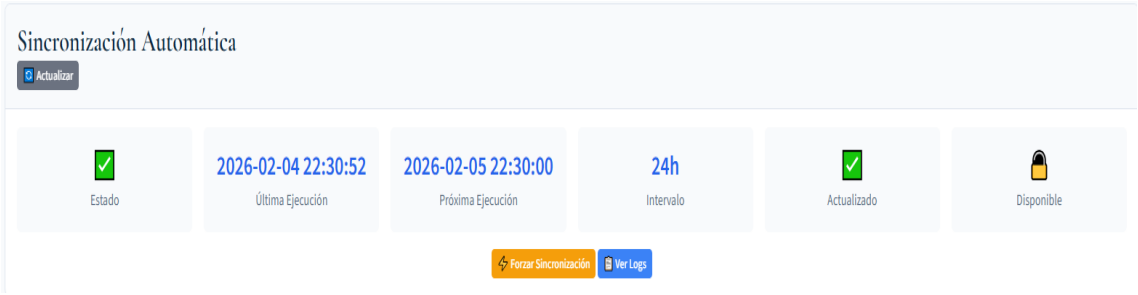
- Gestión completa CRUD de usuarios registrados



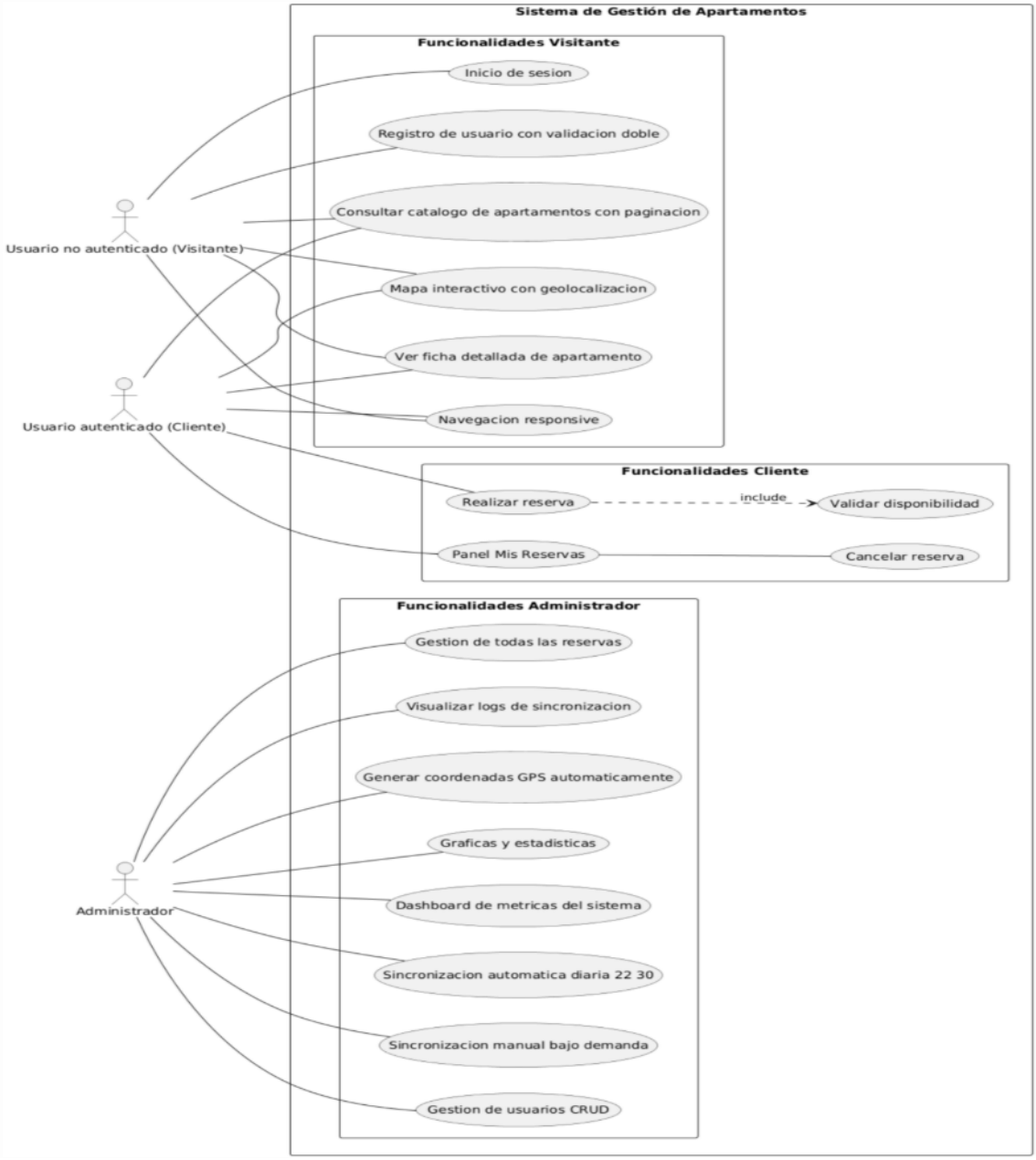
- Supervisión y gestión de todas las reservas del sistema



- Visualización de logs
  de sincronización y estadísticas, generación automática de coordenadas G
  PS
  para apartamentos sin geolocalización y creación de gráficas con estadístic
  as como la distribución por provincia y las tasas de ocupación.



2.3. Diagramas de casos de uso



La descripción completa de nuestro diagrama de uso consistiría en:

Usuario no autenticado (Visitante):

- **Inicio de sesión**: Accede al sistema con credenciales válidas.
- **Registro de usuario con validación doble**: Crea una cuenta mediante un proceso seguro que incluye confirmación por correo electrónico o SMS.
- **Consultar catálogo de apartamentos con paginación**: Visualiza listados organizados por páginas, con filtros básicos.
- **Mapa interactivo con geolocalización**: Explora ubicaciones de apartamentos en un mapa, con posibilidad de usar su ubicación actual.

- **Ver ficha detallada de apartamento**: Accede a información completa de un apartamento (descripción, fotos, precios, servicios, etc.).

Usuario autenticado (Cliente):

- **Navegación responsive**: Accede desde cualquier dispositivo con interfaz adaptada.
- **Realizar reserva**:
    - **Validar disponibilidad**: Verifica fechas y estado del apartamento antes de confirmar.
- **Panel Mis Reservas**: Consulta, modifica o revisa el historial de sus reservas activas y pasadas.
- **Cancelar reserva**: Puede cancelar una reserva dentro del plazo permitido, sujeto a políticas del sistema.

Administrador:

- **Gestión de todas las reservas**: Visualiza, modifica, cancela o confirma reservas de todos los usuarios.
- **Visualizar logs de sincronización**: Revisa registros de las sincronizaciones automáticas y manuales.
- **Generar coordenadas GPS automáticamente**: Asigna o actualiza coordenadas a apartamentos sin ubicación definida.
- **Gráficas y estadísticas**: Accede a informes visuales sobre ocupación, ingresos, reservas, etc.
- **Dashboard de métricas del sistema**: Monitoriza el rendimiento y uso del sistema en tiempo real.
- **Sincronización automática diaria 22:30**: Actualización programada de datos (por ejemplo, con plataformas externas).
- **Sincronización manual bajo demanda**: Ejecuta procesos de sincronización cuando sea necesario.
- **Gestión de usuarios CRUD**: Crear, leer, actualizar y eliminar cuentas de usuarios (clientes y posibles administradores).

Relaciones y extensiones de casos de uso:

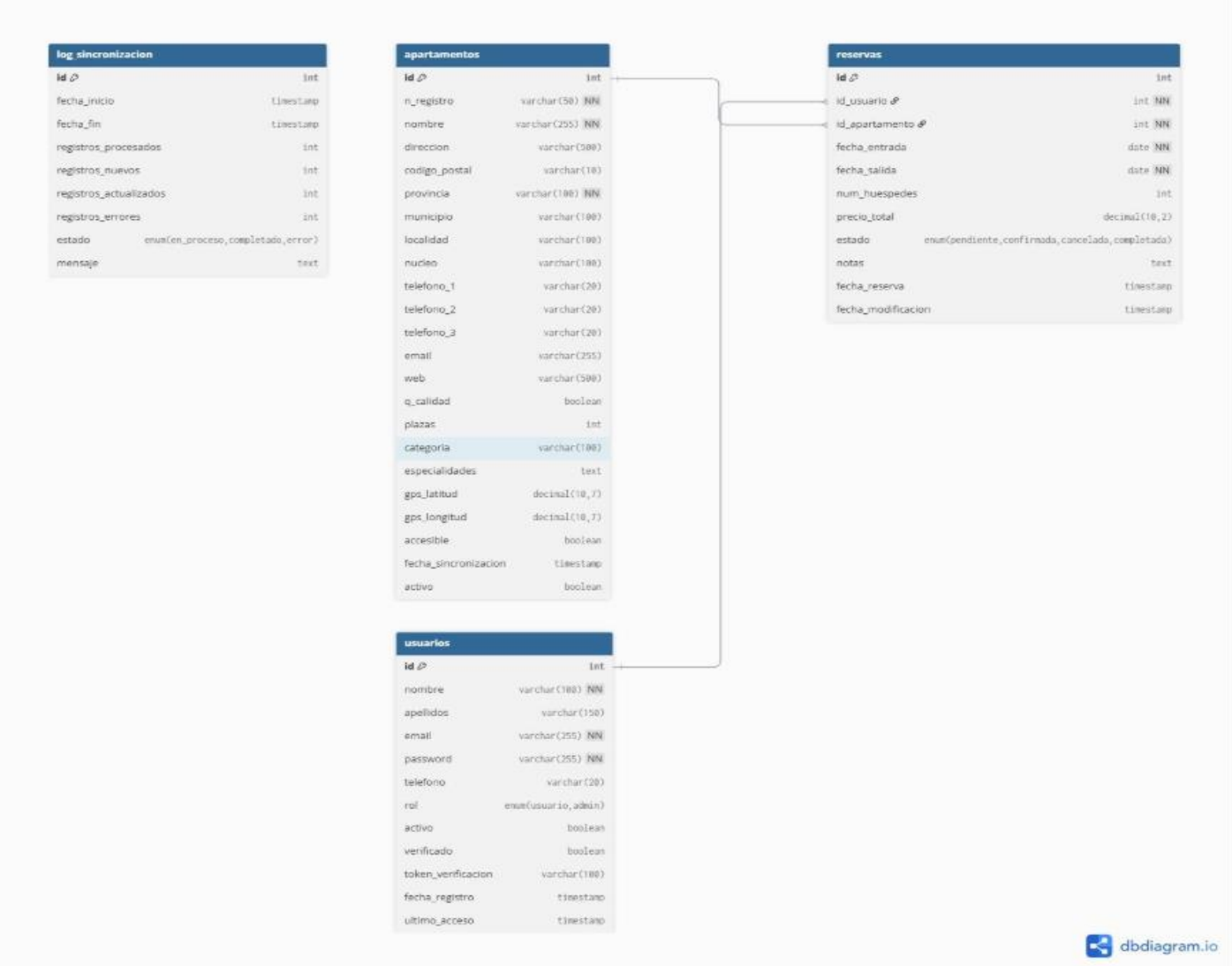
- **“Realizar reserva” incluye “Validar disponibilidad”** como subproceso obligatorio.

- **“Registro de usuario”** puede extenderse con **“Verificación en dos pasos”**.
- **“Consultar catálogo”** puede generalizarse para ambos tipos de usuarios autenticados y no autenticados.
- **“Sincronización automática”** y **“Sincronización manual”** podrían ser especializaciones de un caso de uso general **“Sincronizar datos”**.

3. Diseño

3.1 Modelo de base de datos

El modelo de datos se diseñó siguiendo los principios de normalización hasta la tercera forma normal (3FN)
para evitar redundancias y garantizar la integridad referencial.



La base de datos 'apartamentos_cyl' está implementada en MySQL 8.0 con motor InnoDB, charset utf8mb4 y collation utf8mb4_unicode_ci
para soporte completo de caracteres internacionales.

Entidades principales

1. Tabla apartamentos:

- Almacena información de apartamentos de la API pública
- Campos clave: id (PK), n_registro (UNIQUE), nombre, provincia, municipio
- Contacto: direccion, codigo_postal, telefono_1/2/3, email, web
- Características: plazas, categoria, especialidades, accesible, q_calidad
- Geolocalización: gps_latitud, gps_longitud (DECIMAL 10,7)
- Índices optimizados en provincia, municipio, coordenadas GPS, plazas

2. Tabla usuarios:

- Gestiona usuarios registrados del sistema
- Campos: id (PK), email (UNIQUE), nombre, apellidos, telefono
- Seguridad: password (VARCHAR 255 con hash bcrypt cost 12)
- Control: rol (ENUM 'usuario', 'admin'), activo, verificado
- Auditoría: fecha_registro, ultimo_acceso

3. Tabla reservas:

- Relaciona usuarios con apartamentos
- Campos: id (PK), id_usuario (FK), id_apartamento (FK)
- Fechas: fecha_entrada, fecha_salida (DATE NOT NULL)
- Detalles: num_huespedes, precio_total, notas
- Estado: ENUM ('pendiente', 'confirmada', 'cancelada', 'completada')
- Claves foráneas con ON DELETE CASCADE

4. Tabla log_sincronizacion:

- Registra cada sincronización con API externa
- Campos: fecha_inicio, fecha_fin, registros_procesados
- Estado: ENUM ('en_proceso', 'completado', 'error')

Relaciones entre entidades

En el modelo de datos se establecen las siguientes relaciones principales:

- **USUARIOS (1) — (N) RESERVAS**
  Un usuario puede realizar **cero, una o varias reservas** a lo largo del tiempo.
  Sin embargo, **cada reserva pertenece obligatoriamente a un único usuario**, ya que no puede existir una reserva sin un usuario asociado.
- **APARTAMENTOS (1) — (N) RESERVAS**
  Un apartamento puede tener **cero, una o varias reservas**.
  No obstante, **cada reserva está asociada obligatoriamente a un único apartamento**, ya que toda reserva debe corresponder a un alojamiento concreto.

Por tanto, desde el punto de vista de cardinalidad y opcionalidad, las relaciones quedan definidas como:

- **USUARIOS (0..N) — (1) RESERVAS**
- **APARTAMENTOS (0..N) — (1) RESERVAS**

Esto significa que tanto usuarios como apartamentos pueden existir en el sistema sin tener reservas, pero **no puede existir una reserva sin estar vinculada a un usuario y a un apartamento**.

Estas relaciones se implementan en la base de datos mediante claves foráneas en la tabla **reservas** (id_usuario e id_apartamento), garantizando la integridad referencial del sistema.

Además, existen restricciones de negocio que no se reflejan directamente en el modelo entidad-relación, como por ejemplo la no superposición de reservas en un mismo apartamento, el control de fechas válidas o el límite de huéspedes según las plazas disponibles, las cuales se gestionan a nivel de lógica de aplicación o mediante validaciones adicionales en la base de datos.

3.2. Diseño de la interfaz

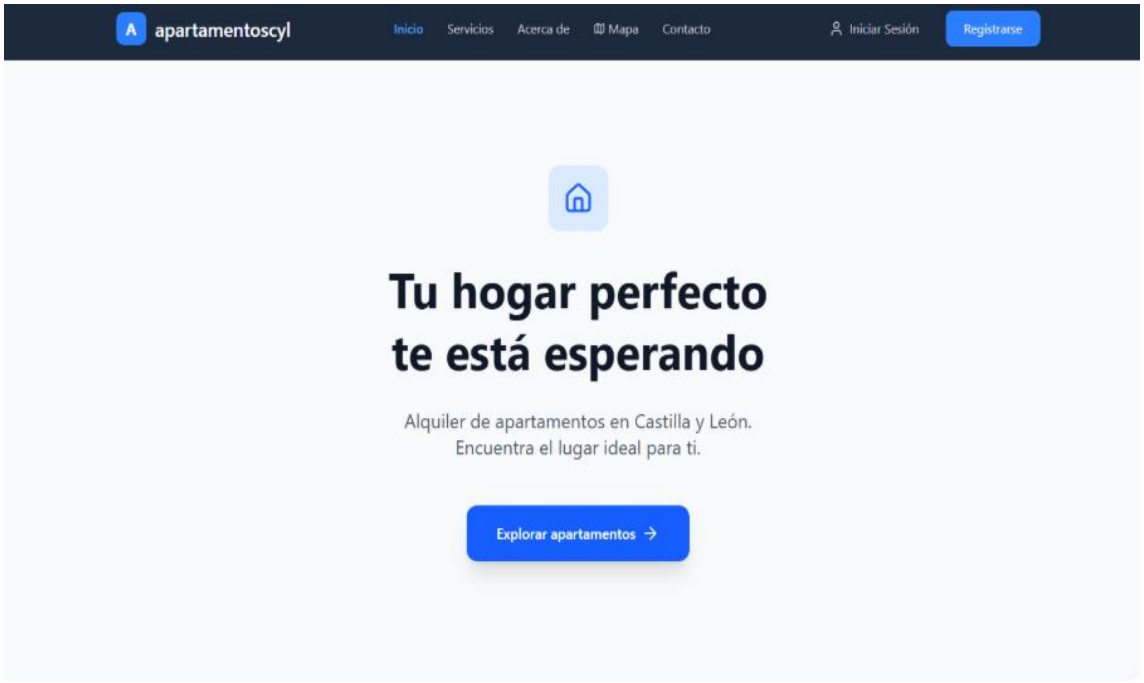
El diseño se planteó bajo un enfoque responsive y mobile first, garantizando experiencia óptima en dispositivos móviles, tablets y ordenadores. Además, se usa una paleta de colores oportuna que encaja con la temática de la página web.

PRIMER PROTOTIPO

Para nuestro primer prototipo de index, pensamos en algo simple y orientativo, un *header* con las funcionalidades iniciales que eran las más básicas:

- Inicio.
- Mapa.
- Inicio sesión.
- Registrarse.

En este primer prototipo no habíamos pensado todavía como ubicar el footer, por lo que lo incluimos en el header, de una manera orientativa.



RESULTADO FINAL

Tras seguir nuestro primer prototipo, en nuestra página final, optamos por cambiar la paleta de colores (serán descritas en el siguiente apartado) por una paleta que tenga una mayor sintonía y que de una sensación más casual. Además, creamos un footer, que cumple con los requisitos de una página estándar.

Paleta de colores

- **Color primario** (#102940): Azul profundo - encabezados y botones principa les.
- **Color acento** (#BF9926):
  Dorado heráldico - elementos interactivos y llamadas a la acción.
- **Color de fondo** (#F2F2F2): Blanco cálido - fondo principal.
- **Texto principal** (#1c1917): Negro cálido - contenido principal.
- **Colores de estado**: Verde (éxito), rojo (error), amarillo (advertencia), azul (información).

Tipografía

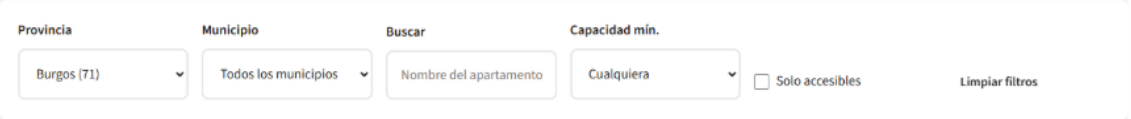
- **Títulos**: 'Cormorant Garamond' (serif elegante que evoca patrimonio histórico).
- **Cuerpo**: 'Source Sans 3' (sans-serif moderna y legible).
- **Código**: 'JetBrains Mono' (monoespaciada).
- **Escala responsiva**: Los títulos se adaptan fluidamente entre dispositivos usando clamp().

Estructura del layout

Cabecera

- Navegación fija (**sticky**) que permanece visible al hacer scroll.
- **Móvil**: Menú hamburguesa con overlay desplegable, altura 64px.
- **Desktop**: Navegación horizontal inline, altura 72px.
- Incluye logo, menú de navegación y estado de sesión.

Sistema de Filtros



- **Móvil**: Filtros apilados verticalmente.
- **Desktop**: Grid de 6 columnas con filtros en línea.
- Componentes: provincia, municipio, búsqueda por nombre, capacidad, accesibilidad.
- Botón de limpiar filtros y contador de resultados.

Listado de Apartamentos

- Grid responsivo adaptativo:
  - Móvil: 1 columna.
  - Tablet: 2 columnas.

- Desktop: 3 columnas.
- Pantallas grandes: hasta 4 columnas.

- Tarjetas con imagen, información básica y botón de acción.
- Efecto hover con elevación (solo en desktop).
- Paginación para grandes volúmenes de datos.

Mapa Interactivo (Leaflet.js)

- Altura adaptativa (calc(100vh - header - 120px), mínimo 500px).
- Marcadores personalizados por provincia con colores distintivos.
- Clustering automático para múltiples apartamentos cercanos.
- Popups interactivos al hacer clic: imagen, nombre, ubicación y botón de acción.
- Leyenda con contadores por provincia .

Footer

- Fondo azul oscuro (#102940) con texto blanco.
- **Desktop**: 3-4 columnas (información, enlaces, contacto).
- **Móvil**: Columnas apiladas.
- Se incluyen enlaces a inicio, apartamentos y provincias.

4. Desarrollo

4.1. Stack tecnológico

La elección del stack tecnológico se fundamentó en la búsqueda de un equilibrio entre robustez, mantenibilidad y eficiencia en el desarrollo. Se optó por tecnologías consolidadas que garantizan estabilidad a largo plazo y amplio soporte comunitario.

Backend

Lenguaje y Runtime

- **PHP 8.x**: Lenguaje principal del backend, aprovechando características modernas como:
  - Tipado estricto (strict types) para mayor seguridad y prevención de errores
  - Named arguments para mejor legibilidad del código
  - Enumeraciones nativas (Enums) para estados y constantes
  - Mejoras de rendimiento significativas respecto a versiones anteriores
  - JIT (Just InTime) compilation para operaciones intensivas

Base de Datos

- **MySQL 8.0**: Sistema gestor de bases de datos relacional elegido por:
  - Alto rendimiento en consultas complejas con múltiples JOINs
  - Soporte nativo para JSON y datos geoespaciales
  - Transacciones ACID que garantizan integridad de datos
  - Optimizaciones para índices y consultas geográficas (crucial para el mapa)
  - Amplia documentación y comunidad activa
- **Capa de Acceso a Datos**
- **PDO (PHP Data Objects)**: Abstracción de base de datos que proporciona:
  - Prepared statements para prevención de inyección SQL
  - Manejo consistente de errores mediante excepciones
  - Compatibilidad con múltiples gestores de BD (portabilidad)
  - Binding de parámetros tipados
- **Arquitectura y Patrones**

- **Patron DAO (Data Access Object)**:
  - Separación clara entre lógica de negocio y persistencia de datos
  - Clases específicas: ApartamentoDAO, ReservaDAO, UsuarioDAO
  - Facilita testing unitario y mantenimiento
  - Reutilización de consultas comunes

Frontend

HTML5: Marcado semántico moderno

- Uso de elementos apropiados (<header>, <nav>, <main>, <section>, <article>, <footer>)
- Atributos de accesibilidad (ARIA labels, roles)
- Formularios con validación nativa (required, pattern, type)
- Optimización para SEO y lectores de pantalla

CSS3: Hojas de estilo avanzadas

- **Variables CSS (Custom Properties)**: Sistema de diseño centralizado y consistente
- **Flexbox**: Layouts unidimensionales flexibles (navegación, cards)
- **CSS Grid**: Layouts bidimensionales complejos (grid de apartamentos, dashboard)
- **Media Queries**: Diseño responsive con breakpoints estratégicos
- **Animaciones y transiciones**: Microinteracciones suaves (transform, opacity, filter)
- **Pseudo-elementos**: Efectos visuales sin JavaScript adicional

JavaScript ES6+: Programación moderna sin frameworks

- **Vanilla JavaScript**: Sin dependencias de frameworks pesados para mayor control y rendimiento
- **Módulos ES6**: Organización del código en archivos independientes (import/export)
- **Async/Await**: Manejo elegante de asincronía en llamadas API
- **Clases**: Programación orientada a objetos para componentes reutilizables

- **Template literals**: Generación dinámica de HTML legible
- **Destructuring y spread operator**: Código más conciso
- **Arrow functions**: Sintaxis moderna y lexical scoping

APIs y Comunicación

- **Fetch API**:
- Reemplazo moderno de XMLHttpRequest
- Basado en promesas nativas
- Soporte para async/await
- Manejo de respuestas JSON simplificado
- Interceptores para manejo centralizado de errores

Librerías Especializadas

- **Leaflet.js 1.9+**: Biblioteca líder de mapas interactivos open source
- Alternativa ligera a Google Maps (sin costes de API)
- Tiles de OpenStreetMap de alta calidad
- Marcadores personalizables por provincia
- Clustering automático para optimización de rendimiento
- Popups interactivos con contenido HTML
- Eventos de clic, zoom y movimiento
- Plugins: MarkerCluster para agrupación inteligente
- Responsive y compatible con dispositivos táctiles

Herramientas

Entorno de Desarrollo Integrado (IDE)

- **Visual Studio Code**: Editor principal con extensiones específicas:
- PHP Intelephense: Autocompletado y análisis estático de PHP
- ESLint: Linting de JavaScript para mantener estándares
- Prettier: Formateo automático de código

- Live Server: Servidor de desarrollo con recarga automática
- GitLens: Visualización avanzada de historial Git
- MySQL: Cliente de base de datos integrado

Control de Versiones

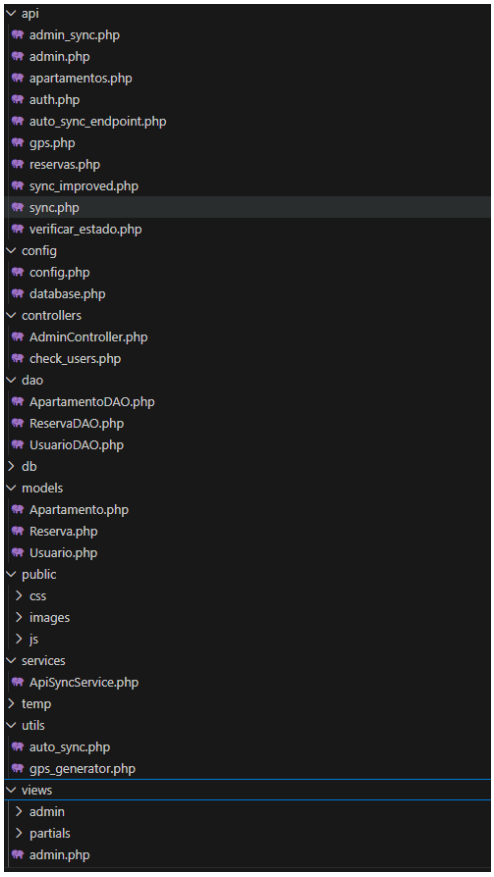
- **Git**: Sistema de control de versiones distribuido
- Commits semánticos siguiendo convención: feat:, fix:, refactor:, docs:
- Branching strategy: main (producción), develop (desarrollo), feature bra nches
- Historial completo de cambios y capacidad de rollback
- Colaboración mediante plataforma remota (GitHub/GitLab)

Entorno de Servidor Local

- **XAMPP/WAMP/Laragon**: Stack de desarrollo local
- **Apache 2.4**: Servidor web con mod_rewrite para URLs limpias
- **PHP 8.x**: Intérprete con configuraciones de desarrollo (display_errors)
- **MySQL 8.0**: Servidor de base de datos local
- **phpMyAdmin**: Interfaz gráfica para administración de BD
- Configuración de virtual hosts para simular entorno de producción
- **Testing y Debugging**
- **Chrome DevTools**: Herramientas de desarrollo del navegador
- Inspector de elementos y estilos CSS en vivo
- Debugger JavaScript con breakpoints
- Network tab para análisis de peticiones HTTP
- Performance profiling para optimización
- Responsive design mode para pruebas móviles
- Console para logging y ejecución de código
- **Firefox Developer Edition**:
- Herramientas especializadas para CSS Grid y Flexbox

- Mejor soporte para inspección de accesibilidad
- Simulación de condiciones de red limitadas
- **Postman**: Plataforma de testing de APIs
- Pruebas de endpoints REST (GET, POST, PUT, DELETE)
- Colecciones organizadas por módulos (apartamentos, reservas, auth)
- Variables de entorno (desarrollo, producción)
- Simulación de autenticación y sesiones
- Generación automática de documentación de API
- Testing automatizado con scripts pre-request y post-request

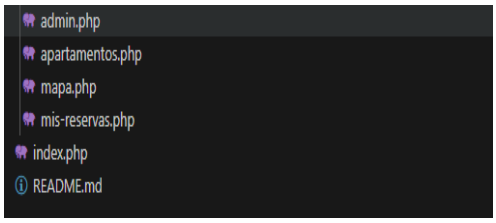
4.2. Estructura de carpetas



Esta organización sigue un **patrón de arquitectura MVC (Modelo-Vista-Controlador)** con algunas adaptaciones y separación clara de responsabilidades. Aquí te detallo la importancia de cada carpeta:

Importancia general de esta estructura:

- **Separación de preocupaciones**: Cada capa tiene una responsabilidad clara.
- **Mantenibilidad**: Cambios en una capa no afectan directamente a las demás.
- **Escalabilidad**: Fácil agregar nuevas funcionalidades.
- **Testeabilidad**: Capas independientes facilitan pruebas unitarias.
- **Seguridad**: Separación lógica entre lógica de negocio, datos y presentación.



Descripción detallada por carpeta:

api/ - Endpoints API REST

- **Importancia**: Expone funcionalidades del sistema para clientes externos (apps móviles, integraciones).
- **Contiene**: Controladores específicos para API, manejo de JSON, autenticación por tokens.
- **Cumple con**: Comunicación cliente-servidor desacoplada.

config/ - Configuración

- **Importancia**: Centraliza configuración del sistema (BD, APIs externas, constantes).

- **Contiene**: Archivos .php con arrays de configuración, parámetros de entorno.
- **Cumple con**: Facilidad de cambiar entornos (desarrollo/producción).

controllers/ - Controladores MVC

- **Importancia**: Reciben peticiones HTTP, coordinan lógica entre modelos y vistas.
- **Contiene**: Clases que manejan rutas específicas (ReservaController.php, UsuarioController.php).
- **Cumple con**: Patrón MVC (capa intermedia).

dao/ - Data Access Objects

- **Importancia**: Abstracción de acceso a datos, separa lógica SQL del resto.
- **Contiene**: Clases con métodos CRUD específicos para cada tabla.
- **Cumple con**: Patrón DAO, facilita cambiar el sistema de BD.

db/ - Scripts de base de datos

- **Importancia**: Control de versiones del esquema de BD.
- **Contiene**: SQL de creación de tablas, datos iniciales, migraciones.
- **Cumple con**: Reproducibilidad del entorno de BD.

models/ - Modelos de dominio

- **Importancia**: Representan entidades del negocio con sus reglas.
- **Contiene**: Clases como Apartamento.php, Reserva.php, Usuario.php.
- **Cumple con**: Patrón MVC (capa de negocio).

public/ - Assets públicos

- **Importancia**: Punto de entrada seguro para el servidor web.
- **Contiene**: CSS, JavaScript, imágenes, fuentes, index.php.
- **Cumple con**: Seguridad (solo esta carpeta es accesible desde web).

services/ - Servicios de negocio

- **Importancia**: Lógica compleja que involucra múltiples modelos.
- **Contiene**: Clases como ReservaService.php, SincronizacionService.php.
- **Cumple con**: Single Responsibility Principle.

temp/ - Archivos temporales

- **Importancia**: Almacena archivos efímeros (caché, logs temporales, uploads).
- **Cumple con**: Evita saturar otras carpetas con datos temporales.

utils/ - Utilidades

- **Importancia**: Funciones helper reutilizables.
- **Contiene**: Validadores, formateadores, helpers de fechas, email, etc.
- **Cumple con**: DRY (Don't Repeat Yourself).

views/ - Vistas y componentes

- **Importancia**: Presentación HTML al usuario.
- **Contiene**: Plantillas .php o .html, componentes reutilizables.
- **Cumple con**: Patrón MVC (capa de presentación).

Archivos raíz:

.htaccess - Configuración Apache

- **Importancia**: Rewrite rules para URLs amigables, seguridad.
- **Cumple con**: Enrutamiento moderno y protección básica.

index.php - Punto de entrada

- **Importancia**: Front controller único.
- **Cumple con**: Patrón Front Controller, todas las peticiones pasan por aquí.

README.md - Documentación

- **Importancia**: Guía para desarrolladores.
- **Cumple con**: Onboarding y mantenimiento del proyecto.

4.3. Descripción del funcionamiento

1. El usuario accede a index.php que verifica sincronización automática.
2. Sistema genera GPS automáticamente si faltan coordenadas.
3. Carga interfaz con estadísticas en tiempo real vía AJAX.
4. Usuarios pueden filtrar apartamentos dinámicamente sin recargar página.
5. Visualización en mapa interactivo con clustering de marcadores.
6. Sistema de autenticación con sesiones PHP y bcrypt.
7. Usuarios autenticados pueden realizar y gestionar reservas.
8. Administradores acceden a panel completo con dashboard de métricas.
9. Sincronización automática cada hora con API externa.
10. Validaciones en cliente y servidor en todos los formularios.

Medidas de seguridad implementadas

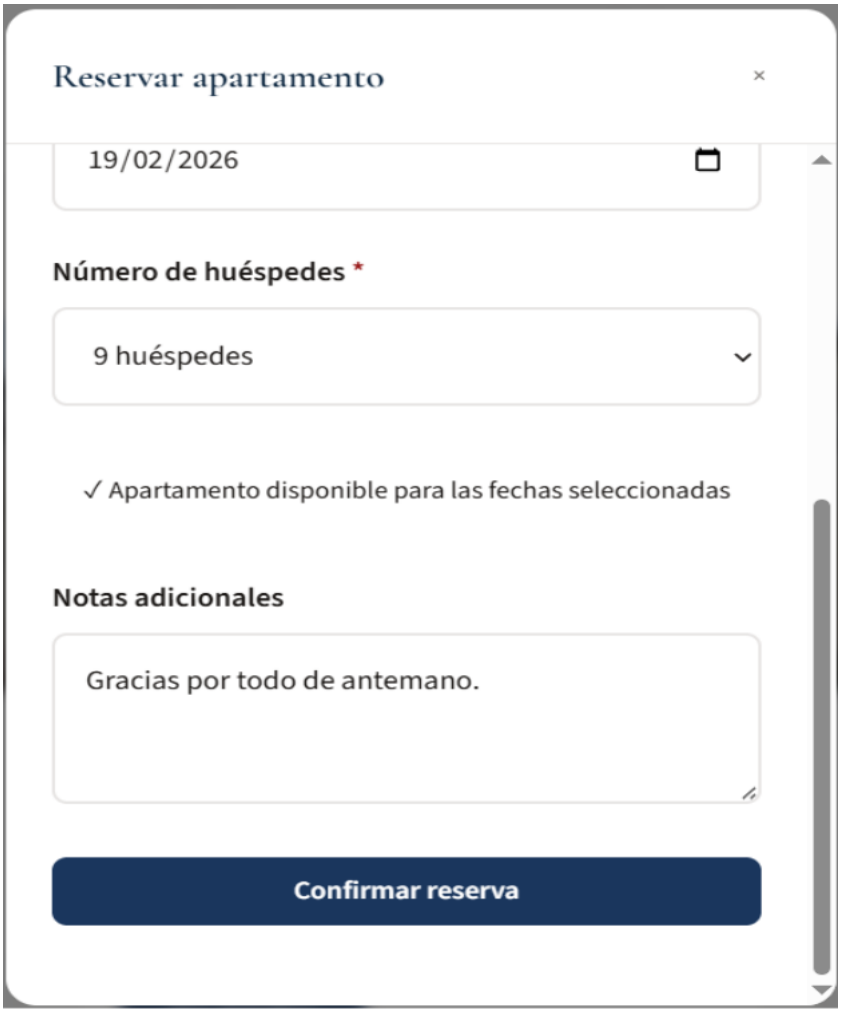
- Contraseñas hasheadas con bcrypt cost 12
- Sesiones con httponly, secure y samesite=lax
- PDO con prepared statements para prevenir SQL Injection
- Escape de HTML con htmlspecialchars() para prevenir XSS
- Tokens CSRF en formularios críticos
- Validación doble (cliente JavaScript, servidor PHP)
- Rate limiting en intentos de login
- Headers de seguridad: X-Content-Type-Options, X-Frame-Options, X-XSS-Protection

5. Pruebas

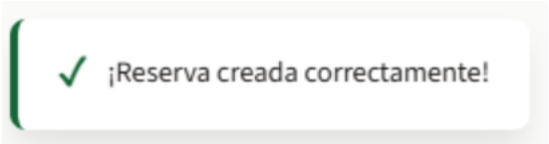
5.1. Pruebas funcionales unitarias

Se implementaron tests unitarios con Jest para módulos críticos:

- **Módulo de reservas:** validación de modales, disponibilidad, creación, can celación.

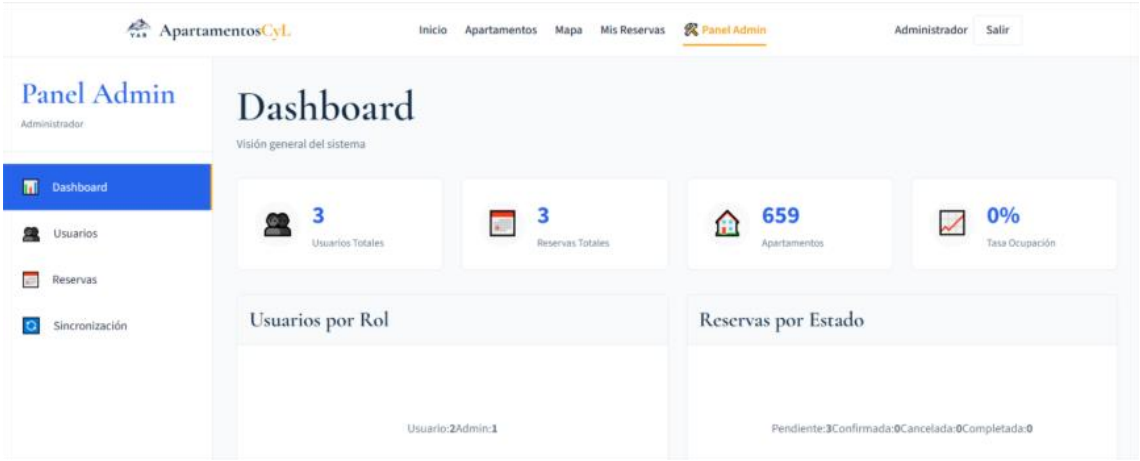


Se puede apreciar el mensaje de disponibilidad del apartamento.



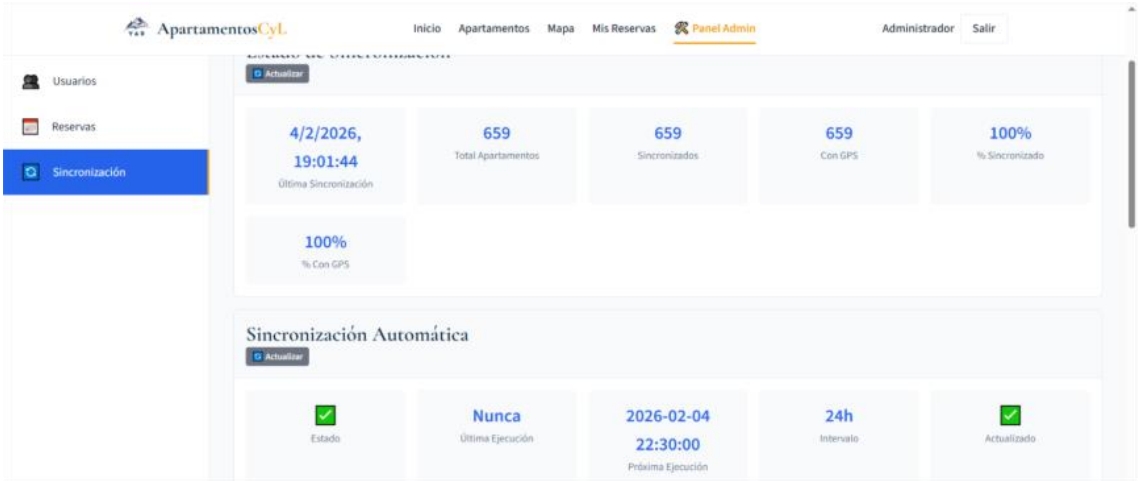
Este mensaje aparece cuando se hace una reserva.

- Módulo de administración: estadísticas, gestión de usuarios, sincronizació n.



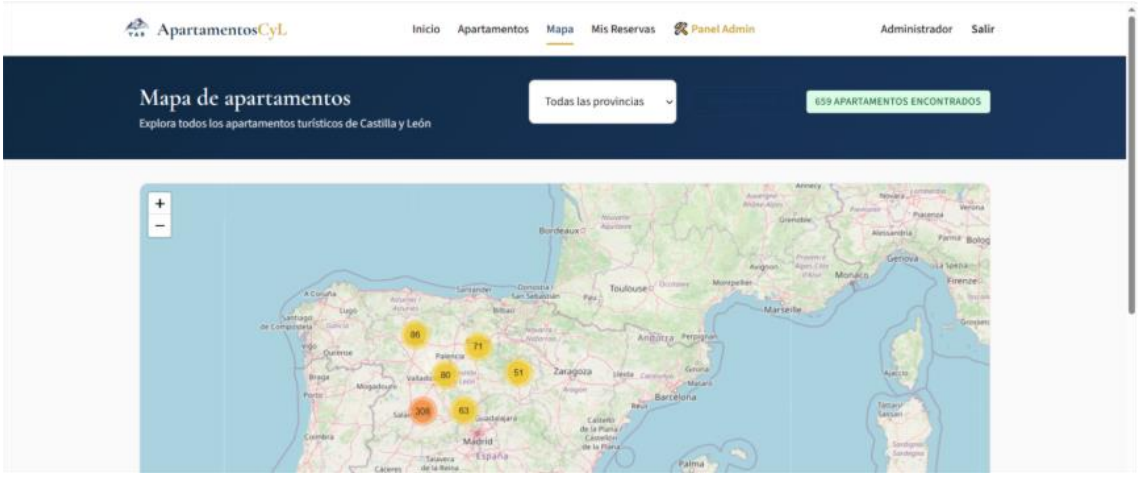
En el dashboard están todo tipo de estadísticas.

Se administra a los usuarios desde este apartado.

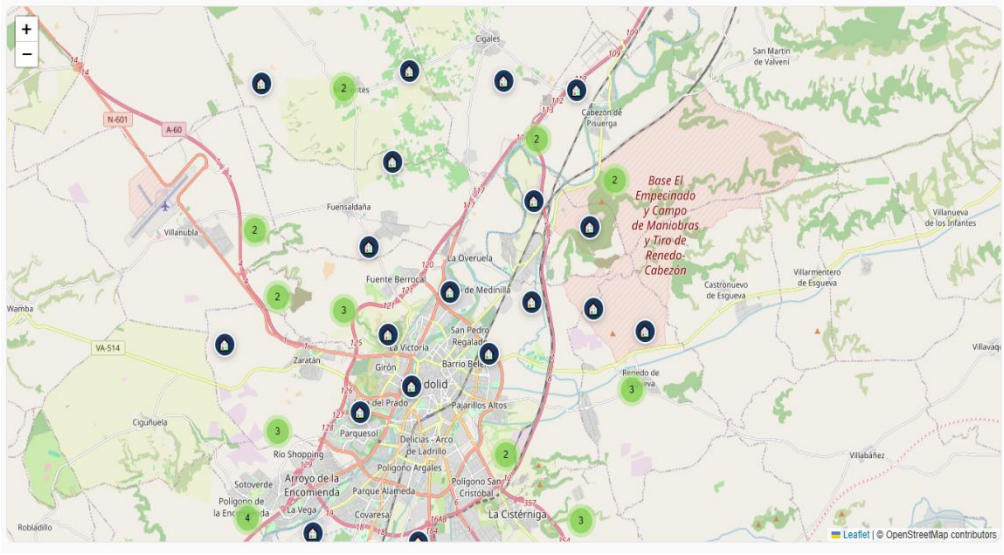


La sincronización se hace automática con intervalos de 24h y también puede ser forzada.

- Módulo de mapa: inicialización Leaflet, carga de marcadores, clustering

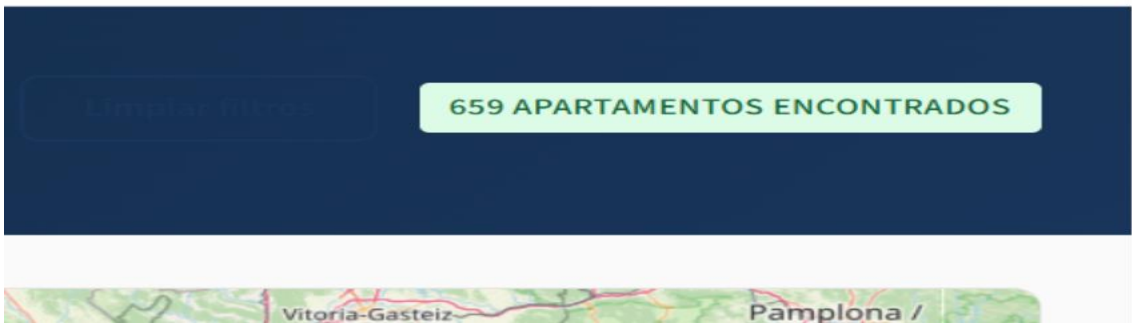


Los apartamentos se agrupan principalmente por provincias.

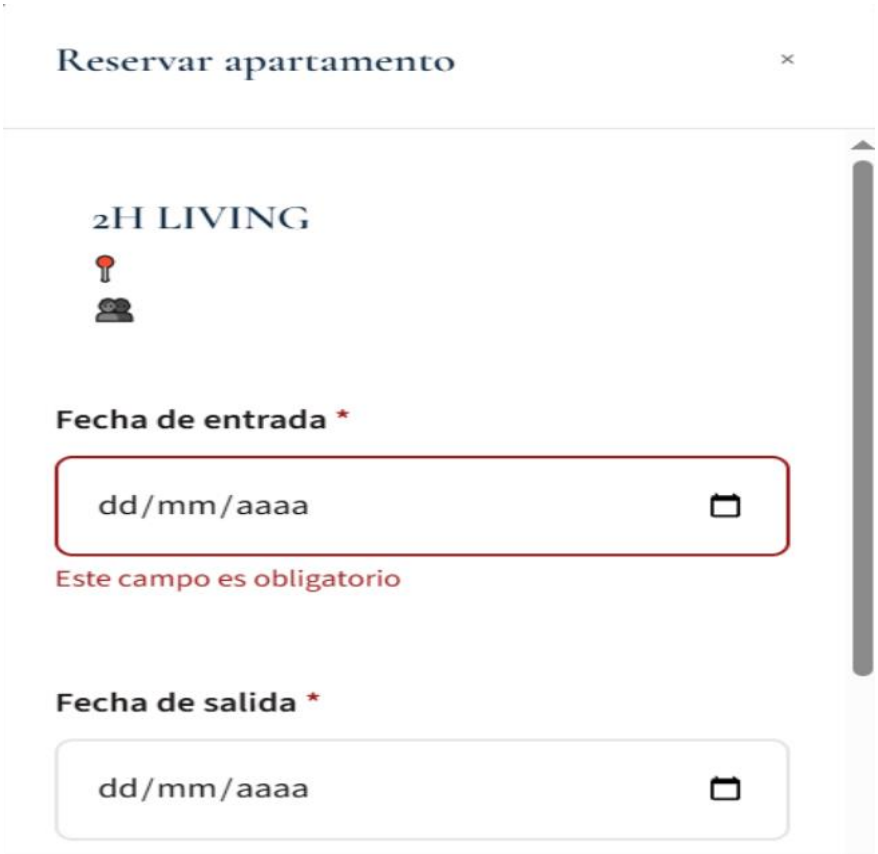


A medida que se hace zoom en el mapa se dividen los apartamentos hasta que aparecen individualmente.

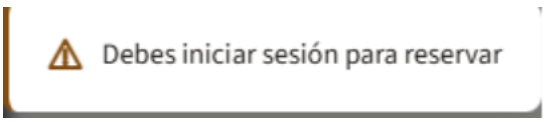
- Manejo de errores: respuestas de red, timeouts, retry, mensajes al usuario



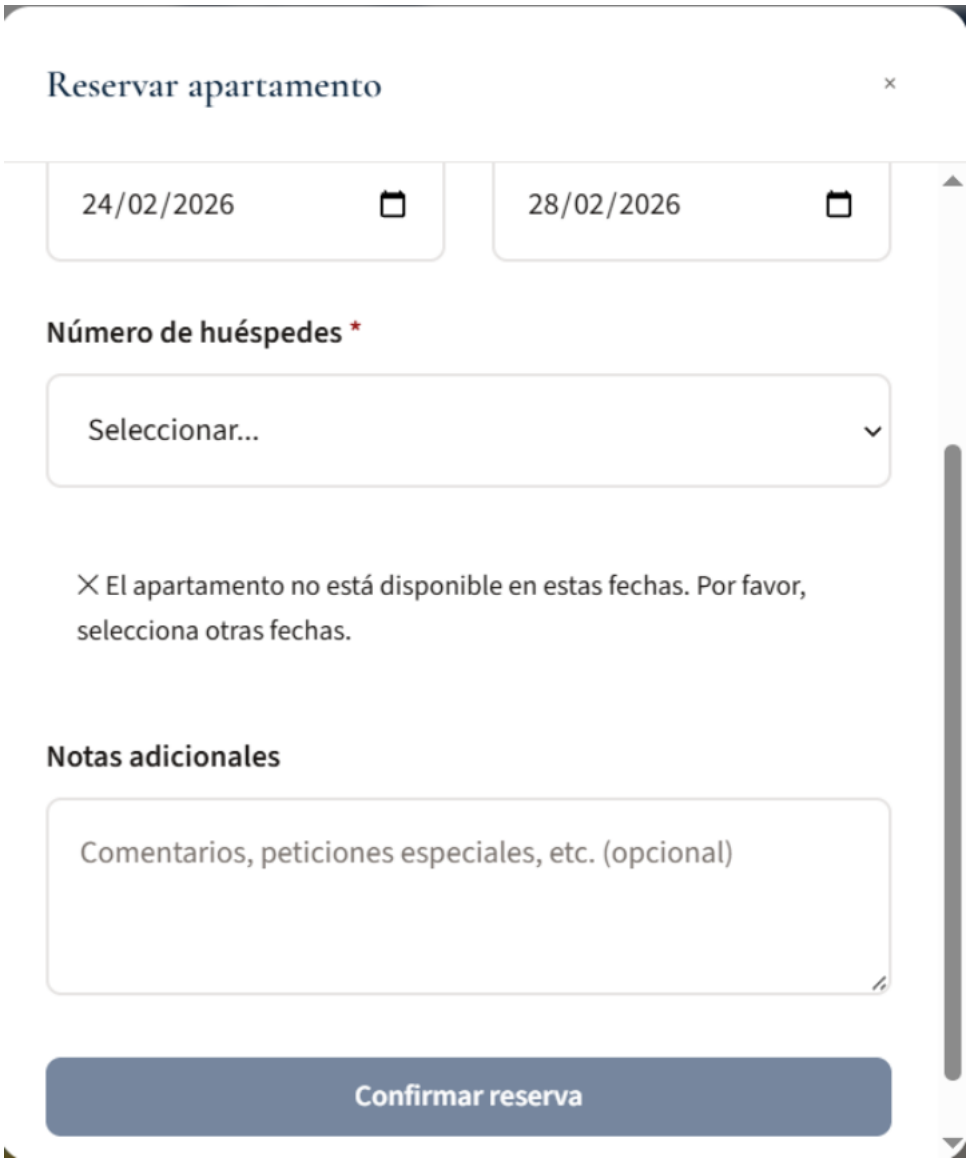
Hay varios mensaje en dashboard que informan al administrador del estado y funcionalidad de los apartados.



También, se hacen varias validaciones a la hora de hacer una reserva.



Aparece si se intenta hacer una reserva sin haberse registrado previamente.



Otro ejemplo de manejo de errores es la imposibilidad de reservar en fechas que ya están reservadas (en el mismo apartamento).

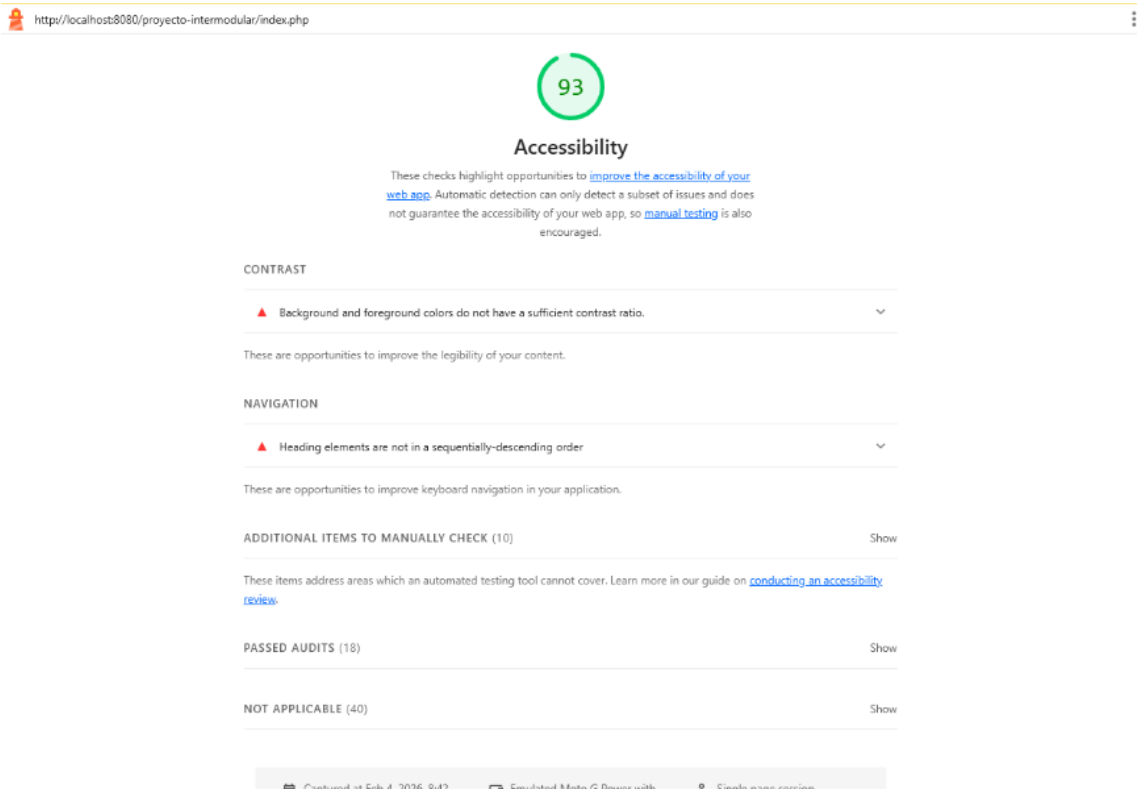
5.2. Pruebas de usabilidad

Se realizó una prueba de usabilidad con
5 usuarios de diferentes perfiles para evaluar la facilidad de uso.

Tarea asignada: "Buscar un apartamento en Valladolid para 4 personas
y visualizar su ubicación en el mapa"

Resultados:
- Tiempo medio de completación: 3 minutos 45 segundos
- Tasa de éxito: 100%
- Satisfacción media: 4.4/5
- Comentarios positivos: filtros intuitivos, mapa útil, proceso de reserva claro

5.3. Análisis de accesibilidad con Lighthouse



Puntuación y evaluación general

- El análisis automatizado con Lighthouse para el proyecto local arroja una puntuación de 93/100 en accesibilidad, lo que indica una base técnica bien construida, pero con dos deficiencias específicas que impiden la excelencia. Estos errores no son meramente cosméticos, sino que afectan directamente la usabilidad para personas con discapacidad visual o que navegan mediante teclado.

Problemas críticos identificados

- Contraste insuficiente entre colores de fondo y texto. Lighthouse detecta al menos una combinación de colores que no alcanza el ratio mínima de 4.5:1 exigido por el nivel AA de WCAG. Este fallo, correspondiente al criterio 1.4.3, dificulta o imposibilita la lectura para usuarios con baja visión o ciertas deficiencias de percepción del color.
- Jerarquía de encabezados no secuencial. Los elementos de título (<h1>, <h2>, etc.) no siguen un orden descendente lógico (por ejemplo, un <h3> aparece después de un <h1> sin un <h2> intermedio). Este error (WCAG 1.3.1 y 2.4.10) desorienta a los usuarios de lectores de pantalla, que dependen de esta estructura para comprender la organización del contenido y navegar por él.

Elementos para verificación manual

- Además, Lighthouse señala 10 ítems que requieren revisión manual, un número considerable. Esta lista suele incluir comprobaciones que una herramienta automatizada no puede realizar, como verificar que el foco del teclado no quede atrapado, que el contenido fuera de pantalla esté correctamente oculto para lectores de pantalla o que los controles personalizados anuncien correctamente su estado. La presencia de esta cantidad de puntos pendientes de revisión humana sugiere que, a pesar de la alta puntuación automática, la accesibilidad real del sitio depende de un examen más profundo.

Conclusión del análisis automatizado

- En resumen, el proyecto obtiene una calificación alta en una evaluación automática, lo que es positivo. Sin embargo, los errores de contraste y estructura de encabezados son fallos fundamentales que deben corregirse como prioridad. La extensa lista de verificaciones manuales pendientes actúa como una advertencia: la accesibilidad no termina con una auditoría automática; requiere una evaluación continua y consciente para garantizar que todos los usuarios puedan interactuar con el contenido en igualdad de condiciones.

5.4. Pruebas de rendimiento

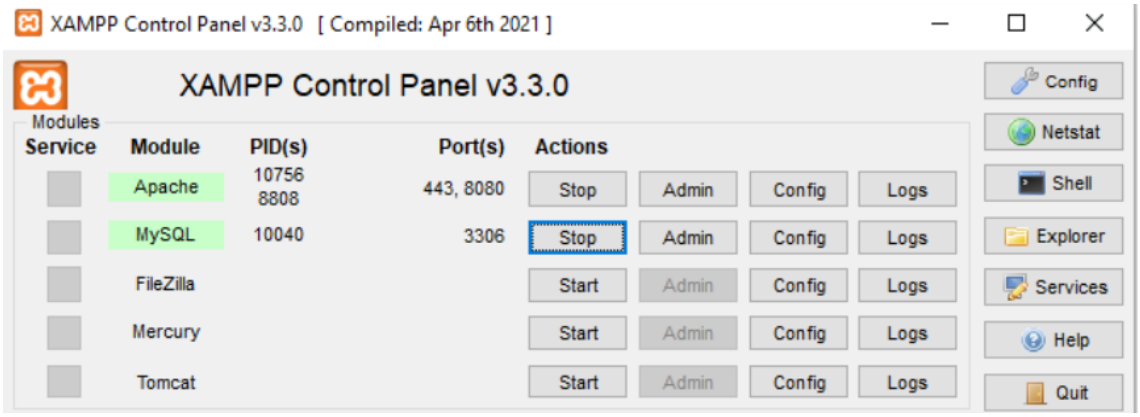
- Tiempo de carga inicial: < 2 segundos en 4G
- Respuesta API: < 300ms promedio
- Carga mapa con 500+ marcadores: < 3 segundos
- Puntuación Lighthouse: Performance 85/100, Accessibility 93/100

6. Despliegue

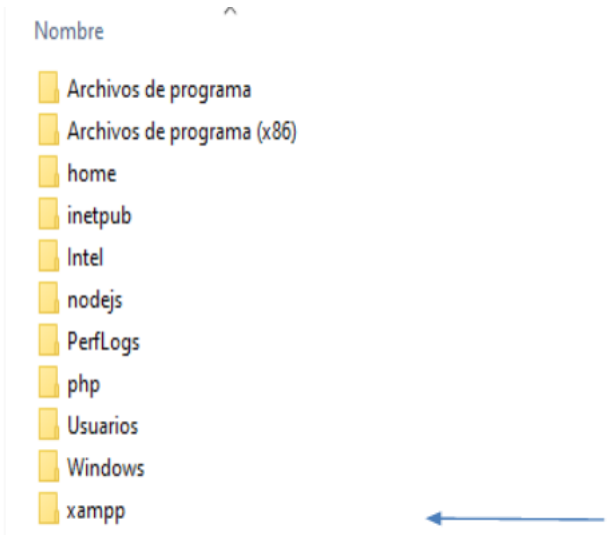
6.1. Instrucciones de instalación local

1. **Instalar** un entorno con **Apache + PHP + MySQL** (por ejemplo, XAMPP).

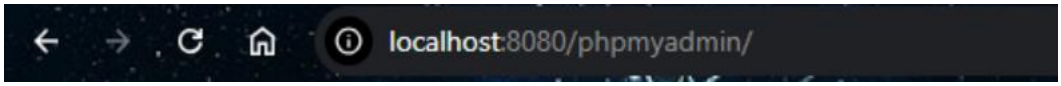
2. **Abrir** el **panel** de control de XAMPP e **iniciar** los **servicios** (APACHE,mYSQL) :



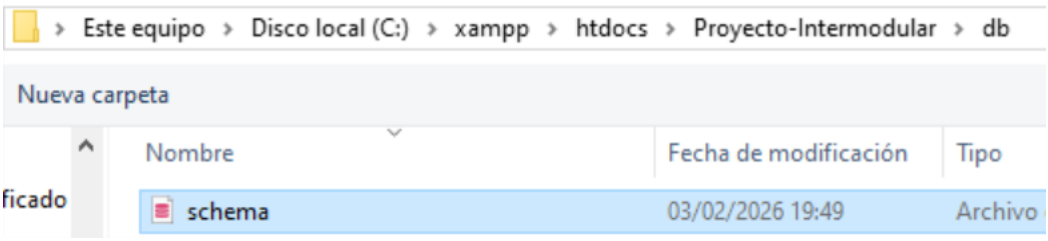
3. **Copiar** la carpeta del proyecto en **xampp/htdocs**.



4. **Accedemos** a **phpMyAdmin** con la URL mostrada en la imagen.



5. **Importamos** la base de datos. Esta base la encontraremos en la **carpeta db** de nuestro proyecto.



En phpMyAdmin accedemos a SQL, copiamos y pegamos la base de datos de nuestro proyecto. Le damos a continuar, y si todo va bien phpMyAdmin nos informara de ello. A la izquierda nos aparecerá nuestra base de datos creada, en nuestro caso “apartamentos_cyl”.

6. El último paso sería acceder desde nuestro navegador (preferiblemente Google Chrome) con la siguiente URL: http://localhost:8080/proyecto-intermodular/.

6.2. URLs del proyecto

Repositorio Git: https://github.com/alexclclclclc-droid/Proyecto-Intermodular

URL LocalHost: http://localhost:8080/proyecto-intermodular/

7. Sostenibilidad y Green Coding

7.1. Sistema de caché inteligente

El sistema de caché reduce drásticamente las peticiones a la API externa:

Funcionamiento:
- Almacena timestamp de última sincronización en temp/last_sync.txt
- Verifica si ha pasado el intervalo (24 horas por defecto) antes de sincronizar
- Usa datos en caché local (MySQL) si son válidos
- Solo sincroniza cuando es necesario, en segundo plano

7.2. Impacto medioambiental estimado

- Sin caché, cada carga de la página implicaría una llamada a la API externa.
- Con el sistema de caché:
  - Se reduce el número de peticiones HTTP.
  - Se disminuye el consumo de red y de procesamiento del servidor externo.
  - Esto supone un ahorro energético significativo y mejora el rendimiento gene ral de la aplicación.

7.3. Otras optimizaciones

- Separación modular de código (reutilización de header/footer)
- Carga selectiva de scripts (Leaflet.js solo en página de mapa)
- Paginación del lado del servidor (20 resultados por página)
- Índices de base de datos optimizados
- Vistas precomputadas para agregaciones complejas
- Compresión gzip/brotli en producción
- Minificación de CSS y JavaScript

8. Conclusiones

8.1. Autoevaluación

8.1.1 Objetivos alcanzados

El **Proyecto Intermodular YAR** ha cumplido satisfactoriamente todos los **requisitos técnicos y funcionales** establecidos. Se logró crear una aplicación web completa con **arquitectura MVC**, **sistema de autenticación robusto** con cifrado de contraseñas, **base de datos relacional normalizada** con sistema de caché, **frontend responsive mobile-first** con Fetch API, integración con **API REST externa** y mapa interactivo con **Leaflet.js**. Se implementaron **validaciones cliente-servidor** y medidas de seguridad contra **vulnerabilidades OWASP Top 10** (SQL injection, XSS, CSRF). El uso de **Git para control de versiones** facilitó la colaboración y mantenimiento del código. El proyecto consolidó **competencias técnicas** (pensamiento arquitectónico, integración de sistemas, optimización de rendimiento) y **habilidades transversales** (gestión del tiempo, auto-aprendizaje, perseverancia).

8.1.2 Dificultades y soluciones

Control de versiones con Git: Dificultades iniciales con commits, branches, merges y autenticación SSH. Solucionado implementando Git Flow y reuniones diarias de coordinación.

Coordenadas GPS incompletas: La API no proporcionaba coordenadas para todos los apartamentos. Desarrollado sistema automático de geocodificación a partir de direcciones postales.

Sistema de sincronización automática: Problemas de procesos bloqueantes, bloqueos de base de datos y gestión de fallos. Implementado sistema en segundo plano con caché inteligente, transacciones y cola de procesamiento.

Inconsistencias en estadísticas: Datos descoordinados en diferentes vistas. Solucionado con sistema unificado de invalidación de caché y precálculo de estadísticas agregadas.

Problemas con filtros: Usuarios atrapados sin resultados. Rediseñada la lógica con sistema de cascada y botón "Limpiar filtros".

Rutas relativas: Fallos en diferentes entornos. Definida constante BASE_URL que se ajusta automáticamente.

Diseño responsive: Elementos desbordados en dispositivos reales. Implementadas media queries granulares, diseño fluido, menú hamburguesa y Flexbox/CSS Grid.

8.2. Líneas futuras

Existen múltiples áreas de mejora para convertir la aplicación en un **producto comercialmente viable**:

Sistema de valoraciones y reseñas verificadas: **Verificación de reservas reales**, **puntuación por categorías** y moderación de contenido para generar confianza.

Galería de imágenes profesional: Upload con **compresión automática**, **generación de miniaturas** y almacenamiento mediante **AWS S3 o Cloudinary**.

Pasarela de pagos integrada: Integrar **Stripe, PayPal o Redsys** para procesar reservas, reembolsos y cumplir con **normativas PCI-DSS**.

Sistema de notificaciones automatizadas: Email transaccional con confirmaciones, recordatorios y newsletters mediante **SendGrid, Mailgun o Amazon SES**.

Favoritos con sincronización multiplataforma: **Sincronización automática entre dispositivos** mediante **WebSockets o polling** y notificaciones de cambios de precio.

Internacionalización (i18n): **Soporte multiidioma** (español, inglés, francés, alemán) con detección automática y adaptación de formatos.

Panel de gestión para propietarios: **Portal self-service** con calendario de disponibilidad, gestión de precios y **sistema de verificación**.

Motor de ofertas y promociones: Sistema flexible de descuentos con **códigos promocionales** y **campañas dirigidas**.

Infraestructura de hosting profesional: Migrar a **hosting de producción escalable** para comercialización.

Reflexión final

El desarrollo del Proyecto Intermodular ha supuesto la **aplicación práctica e integrada** de todos los conocimientos del Ciclo Formativo, simulando un **escenario real de desarrollo profesional**. Se han alcanzado los **objetivos RA1–RA4** y los aprendizajes más relevantes incluyen la **gestión de integración de datos externos** de APIs públicas, **mecanismos de sincronización** robustos y solución de problemas mediante **geocodificación**. El proyecto ha consolidado **competencias técnicas** (Git, MVC, APIs REST, seguridad) y **habilidades transversales** (planificación, documentación), representando una **aproximación realista al entorno profesional del desarrollo web**.

9. Bibliografía

Junta de Castilla y León. (2025). Portal de
Datos Abiertos. Recuperado de https://datosabiertos.jcyl.es/web/es/datos-abiertos-castilla-leon.html

WebAIM. (2026). WAVE
Web Accessibility Evaluation Tool. Recuperado de https://wave.webaim.org/

PHP Group. (2025). PHP Manual
- Version 8.x. Recuperado de https://www.php.net/manual/es/

Oracle Corporation. (2025). MySQL 8.0 Reference
Manual. Recuperado de https://dev.mysql.com/doc/

Mozilla Developer Network. (2025). JavaScript
Guide. Recuperado de https://developer.mozilla.org/es/

Leaflet. (2025). Leaflet Documentation. Recuperado de

https://leafletjs.com/reference.html

W3C. (2025). Web Content Accessibility Guidelines (WCAG)
2.1. Recuperado de https://www.w3.org/WAI/WCAG21/

The Shift Project. (2024).
Lean ICT: Pour une sobriété numérique. Recuperado de https://theshiftproject.org/

Apache Software Foundation. (2025). Apache HTTP
Server Documentation. Recuperado de https://httpd.apache.org/docs/