Alquiler de departamentos - Benites

Owner	Reviewer
Anderson Carcamo	

Summary:

Requirements:

Requerimentos Funcionales

F1 — Búsqueda básica y filtros (MVP — obligatorio)

- **Qué hace:** Permite a usuarios buscar inmuebles por texto y filtros: tipo (departamento, casa, local), operación (venta/alquiler), rango de precio, distrito, dormitorios, baños.
- **Por qué importa:** Núcleo de producto; la mayoría de los usuarios llegan por búsqueda.
- **Criterios de aceptación:**
    - Aplicar 3 filtros devuelve resultados coherentes en una sola página.
    - Paginación disponible (limit/offset o cursor).
    - Relevancia por fecha y coincidencia de texto.
- **Consideraciones técnicas (MVP):**
    - Implementación inicial: Postgres con índices compuestos y tsvector (full-text) o un motor simple de búsqueda en DB.
    - Campos indexados: title, description, price, district, type, bedrooms.
    - API ejemplo: GET /api/listings?query=&type=departamento&min_price=&max_price=&district=&page=1&limit=20.

F2 — Listado y detalle de inmueble (MVP)

- **Qué hace:** Listado con miniaturas y vista detalle con galería de imágenes, descripción, ubicación y contacto.
- **Criterios de aceptación:**

    - Página de detalle muestra al menos 5 imágenes, precio, características y botón “Contactar” por whatsapp.
    - Endpoint devuelve 200 y JSON con campos necesarios.
- **Datos mínimos:** id, title, description, price, currency, lat, lon, address, bedrooms, bathrooms, area_m2, images[].

F3 — Publicación y edición de anuncios (MVP básico)

- **Qué hace:** Usuarios o agencias pueden crear anuncios con fotos y metadatos; estado: borrador → publicado.
- **Criterios de aceptación:**
    - Subir hasta 10 imágenes, preview, y publicar en estado “publicado” visible en búsqueda.
    - Validaciones: precio > 0, lat/lon válidos, al menos 1 imagen.
- **Consideraciones prácticas:**
    - Guardar archivos inicialmente en filesystem local /uploads, con nombres seguros; migrable a S3/MinIO luego.
    - Endpoint: POST /api/listings (multipart/form-data).

F4 — Autenticación y gestión de perfiles (MVP)

- **Qué hace:** Registro/login por email + contraseña; roles: user, seller, admin.
- **Criterios de aceptación:**
    - Login devuelve token JWT (access + refresh) y perfil con role.
    - Flujo de recuperación de contraseña por email (token temporal).
- **Consideraciones:** Para MVP puedes enviar emails con SMTP local o servicio gratuito (Mailtrap/SendGrid en dev).

F5 — Favoritos y guardado de búsquedas (MVP)

- **Qué hace:** Usuario guarda inmuebles y búsquedas; lista de favoritos visible.
- **Criterios de aceptación:**
    - POST /api/favorites guarda y GET /api/favorites lista ordenados por fecha.

    - Guardado de búsqueda crea alerta (solo registro ahora; envío por batch en segundo plano).

F6 — Captura de leads / Contacto (MVP)

- **Qué hace:** Formulario "Contactar" en cada anuncio que genera un lead para el anunciante.
- **Criterios de aceptación:**
    - Cada envío crea registro lead con origen (listing id), timestamp, email/phone, mensaje.
    - Panel del anunciante muestra leads (tabular).
- **Consideraciones:** Guardar leads en DB y exponer vía GET /api/agents/:id/leads.

F7 — Panel básico de anunciante (MVP)

- **Qué hace:** Interfaz web para ver/editar anuncios, ver leads y métricas básicas (vistas, contactos).
- **Criterios de aceptación:** Mostrar métricas con frecuencia de actualización a demanda (no real-time obligatorio).

F8 — Mapas y geolocalización (fase 2)

- **Qué hace:** Buscar por radio, dibujar área o cluster en mapa.
- **Por qué esperar:** Requiere integración con API de mapas y optimizaciones geo (PostGIS o Elastic geo queries).
- **Criterio mínimo:** Soportar búsqueda por lat,lon,radius en la API: GET /api/listings?lat=...&lon=...&radius=2000.

F9 — Notificaciones y alertas en tiempo "casi real" (fase 2)

- **Qué hace:** Push/email cuando hay nuevos anuncios que coincidan con búsquedas guardadas.
- **Criterio:** Primer envío dentro de 5-15 minutos en sistema simple por batch.

F10 — Chat / mensajería en tiempo real (futuro – no necesario, pero seria opción más segura que whatsapp)

- **Qué hace:** WebSocket o polling; agendar visitas.
- **Por qué:** Mejora conversión, pero aumenta complejidad operativa. Además que brinda seguridad para el anunciante al no dar información personal.

Requerimentos no funcionales:

NF1 — Arquitectura modular y desplegable localmente (obligatorio MVP)

- **Objetivo:** Que cada componente sea independiente y pueda ejecutarse localmente (docker-compose).
- **Cómo lograrlo:** Microservicios ligeros o servicios monolíticos modularizados (Módulos: auth, listings, media, leads).
- **Criterio de aceptación:** Repositorio con docker-compose up que levanta DB, API, y frontend en local.

NF2 — Rendimiento (MVP targets)

- **Targets iniciales (realistas en local):**
  - Respuesta promedio de búsqueda: p50 < 300 ms (dependiendo de dataset), p95 < 1.5 s.
  - Página de detalle: p95 < 800 ms (sin CDN).
- **Estrategias:** Índices DB, paginación, lazy-loading de imágenes, compresión.

NF3 — Escalabilidad progresiva

- **Requisito:** Arquitectura que permita migrar fácilmente: filesystem → S3, Postgres → particionado, búsquedas en DB → Elasticsearch.
- **Criterio:** Código de almacenamiento desacoplado (Storage Adapter pattern) para intercambiar backend sin reescribir lógica.

NF4 — Disponibilidad y tolerancia a fallos

- **MVP:** manejar caídas en servicios externos (timeouts y retries).
- **Meta futura:** 99.9% uptime.
- **Criterios:** Circuit breaker, retry limitado, timeouts configurables.

NF5 — Seguridad

- **Requisitos concretos:**

    - TLS en todas las comunicaciones (dev: self-signed opcional).
    - Hash de contraseñas con bcrypt/argon2.
    - Validación y sanitización de inputs, almacenamiento seguro de tokens (refresh almacenar hashed).
    - Rate limiting por IP en endpoints sensibles (/login, /api/listings creación).
    - Protección básica OWASP (XSS, CSRF en frontend).
- **Criterios de aceptación:** Escaneo básico con herramienta (OWASP ZAP) sin fallos de alta severidad.

NF6 — Observabilidad y Telemetría (mínimo MVP)

- **Qué incluir desde inicio:**
    - Logs estructurados (JSON) a stdout.
    - Métricas básicas expuestas en /metrics (prometheus format) para el servicio principal.
    - Errores críticos reportados a Sentry (o log local).
- **Criterio:** Alertas en dev cuando error rate > 5% en 10 minutos (puede ser local console alert).

NF7 — Consistencia de datos y backup

- **Requisitos:**
    - Backups diarios de BD (dump) en local dev; migrable a snapshot en remoto.
    - Migrations gestionadas con herramienta (Flyway / Knex / TypeORM migrations).
- **Criterios:** Restauración local de backup en <30 min (procedimiento documentado).
- No es necesario a primera mano, se puede hacer después, ya que solo son copias de seguridad, sin embargo, son importantes en caso se caiga el sistema. Mas adelante si se usa microservicios no será necesario.

NF8 — Control de costos (importante en lanzamientos)

- **Estrategia MVP:** usar recursos locales y servicios gratuitos en dev; evitar costosas soluciones hasta que métricas lo justifiquen, alojar un servidor físico (tiene demora en lanzamiento, es complicado de configurar), usar dominio gratuito.
- **Criterio:** Costos predecibles y medidos; estimación de gasto por transición a cloud documentada.

NF9 — Internacionalización y localización

- **Por qué:** Soporte de S/ y USD, formatos de fecha, idioma (español por defecto).
- **Criterio:** Todas las vistas y API devuelven moneda y locale; i18n-ready.

NF10 — Accesibilidad y UX (mínimo)

- **Requisito:** Frontend responsive mobile-first; cumplimiento básico WCAG AA (normas para que sea considerado accesible) para elementos clave (formularios, contraste).
- **Criterio:** Tests manuales en móvil y auditoría Lighthouse >= 50 accesibilidad (MVP).

Out of Scope:

Keywords:

MVP (Producto Mínimo Viable)
Definición: Versión inicial del producto con las funcionalidades esenciales para validar la idea.
Por qué importa: Permite lanzar rápido y recibir feedback real sin invertir todo el presupuesto.
Ejemplo: solo búsqueda, publicar anuncio y recibir contactos.

MVP local
Definición: MVP que se puede ejecutar en una máquina o servidor local (no en la nube).
Por qué importa: reduce costos iniciales; facilita pruebas y desarrollo.

Frontend
Definición: La parte de la aplicación que ven los usuarios (sitio web o app).
Por qué importa: es la interfaz que determina la experiencia del usuario.

Backend
Definición: La parte que gestiona datos, lógica y comunicación; no es visible para el usuario.
Por qué importa: procesa búsquedas, guarda anuncios y gestiona autenticación.

API (Interfaz de Programación de Aplicaciones)
Definición: Conjunto de rutas/servicios que permiten que el frontend y otros sistemas hablen con el backend.
Ejemplo: GET /api/listings devuelve anuncios.

Endpoint
Definición: Una URL concreta dentro de la API que realiza una acción (obtener, crear, actualizar).
Ejemplo: POST /api/listings crea un anuncio.

REST / GraphQL
Definición: Estilos comunes para diseñar APIs. REST usa rutas y verbos HTTP; GraphQL permite consultas flexibles.
Por qué importa: define cómo integrar otras partes o terceros.

Autenticación / Autorización
Definición: Autenticación = identificar a un usuario (login). Autorización = qué puede hacer ese usuario (roles).
Ejemplo: un anunciante puede publicar; un visitante no.

JWT (JSON Web Token)
Definición: Token seguro que el servidor da al usuario para identificarlo en futuras peticiones.
Por qué importa: permite sesiones sin almacenar estado en el servidor.

OAuth
Definición: Protocolo para "login" con cuentas externas (ej. Google).
Por qué importa: facilita el registro para el usuario.

Base de datos (DB)
Definición: Lugar donde se guardan anuncios, usuarios y leads.
Ejemplo: PostgreSQL es una DB relacional recomendada.

Postgres / PostGIS
Definición: PostgreSQL es la DB; PostGIS añade funciones geográficas (distancias, polígonos).
Por qué importa: útil para búsquedas por zona o radio.

Full-text search / tsvector
Definición: Técnica para buscar texto dentro de títulos y descripciones. tsvector es una opción en Postgres.
Por qué importa: permite búsquedas por palabras en descripciones.

Elasticsearch / OpenSearch
Definición: Motor de búsqueda avanzado optimizado para grandes volúmenes y filtros faceted.
Por qué importa: mejora velocidad y relevancia cuando el volumen crece.

CDN (Content Delivery Network)
Definición: Red de servidores que entrega imágenes y archivos desde ubicaciones cercanas al usuario.
Por qué importa: reduce tiempos de carga de fotos.

S3 / MinIO (Almacenamiento de objetos)
Definición: Lugar para guardar archivos (fotos, videos). S3 es el servicio de AWS; MinIO es alternativa autogestionada.
Por qué importa: facilita escalar almacenamiento sin depender del servidor principal.

Docker / docker-compose
Definición: Herramientas para ejecutar servicios en contenedores; docker-compose levanta varios servicios juntos (DB, API, frontend).
Por qué importa: estandariza el entorno local y facilita pasar al servidor o la nube.

Kubernetes
Definición: Plataforma para orquestar contenedores en producción (escalar, redistribuir).
Por qué importa: importante cuando se necesita alta disponibilidad y escalado automático.

Microservicios vs Monolito
Definición: Microservicios = aplicación dividida en servicios pequeños e independientes. Monolito = todo en una sola aplicación.
Por qué importa: microservicios facilitan escalar por partes; monolito es más simple para empezar.

Adapter (Storage Adapter)
Definición: Capas de código que permiten cambiar componentes (por ejemplo local → S3) sin reescribir el resto.
Por qué importa: facilita migraciones a futuro.

Cache / Redis
Definición: Memoria rápida donde se guardan resultados temporales para acelerar respuestas (ej. búsquedas frecuentes). Redis es una tecnología común.
Por qué importa: reduce latencias y carga en la base de datos.

Load Balancer (Balanceador de carga)
Definición: Distribuye tráfico entre varios servidores para evitar sobrecarga.
Por qué importa: mejora disponibilidad y rendimiento cuando hay muchos usuarios.

Escalado vertical / horizontal
Definición: Vertical = añadir recursos a un servidor (más CPU/RAM). Horizontal = añadir más instancias del servicio.
Por qué importa: estrategia para manejar crecimiento de usuarios.

Latency (latencia) & Throughput
Definición: Latencia = tiempo de respuesta; Throughput = cantidad de peticiones por segundo que el sistema puede manejar.
Por qué importa: métricas clave para la experiencia de usuario.

SLA (Service Level Agreement)
Definición: Objetivo de disponibilidad/tiempo de servicio (por ejemplo 99.9%).
Por qué importa: define expectativas de uptime.

RTO / RPO
Definición: RTO = tiempo objetivo de recuperación tras caída. RPO = cuánto dato se puede perder (p. ej. 1 hora).
Por qué importa: plan de backups y recuperación ante fallos.

CI/CD (Integración Continua / Entrega Continua)
Definición: Automatización de pruebas y despliegues al fusionar código.
Por qué importa: reduce errores y acelera lanzamientos.

Observabilidad (logs, métricas, trazas)
Definición: Conjunto de herramientas para monitorear y entender cómo funciona el sistema en tiempo real.
Por qué importa: detectar y resolver problemas rápido.

WAF / OWASP / Seguridad básica
Definición: WAF = firewall web; OWASP = buenas prácticas de seguridad web.
Por qué importa: protege la app de ataques comunes.

TLS (HTTPS)
Definición: Protocolo que cifra la comunicación entre usuario y servidor.
Por qué importa: seguridad y confianza del usuario.

Backup / Migrations
Definición: Backup = copias de seguridad; Migrations = cambios ordenados en la estructura de la base de datos.
Por qué importa: mantener datos seguros y permitir cambios controlados.

PWA / Responsive
Definición: PWA = aplicación web que se comporta como app móvil; Responsive = diseño que se adapta a pantallas.
Por qué importa: mejora experiencia móvil sin desarrollar app nativa desde el inicio.

Geolocalización / Clustering
Definición: Geolocalización = uso de lat/lon para ubicar anuncios; Clustering = agrupar muchos pins en un mapa para no saturarlo.
Por qué importa: facilita búsquedas por zona y mejora usabilidad.

Paginate (Paginación) / Cursor vs offset
Definición: Técnicas para dividir resultados en páginas. Cursor es más eficiente para grandes volúmenes que offset.
Por qué importa: evita respuestas lentas con muchos anuncios.

Lead / CRM ligero
Definición: Lead = contacto generado por un anuncio. CRM ligero = panel para que anunciante vea leads.
Por qué importa: es la métrica de conversión principal (visitante → contacto).

Push Notifications / FCM / APNs
Definición: Mensajes enviados a la app móvil; FCM = servicio de Google; APNs = Apple.
Por qué importa: retención — avisa a usuarios sobre nuevos anuncios.

WebSocket
Definición: Canal de comunicación en tiempo real entre cliente y servidor.
Por qué importa: útil para chat o notificaciones instantáneas.

Current status:

En planeación.

Technical proposal:

En primera instancia se va a lanzar en un despliegue local, pero modularizado para que pueda estar listo para escalar.

Dev local: docker-compose con:

- Service db (Postgres), api (node/express), frontend (next/react), minio (opcional), redis (opcional).

Configuración: variables mediante. env, storage adapter para archivos que puede apuntar a /uploads o a MinIO.

Migración a producción (pasos conceptuales):

1. Cambiar adapter a S3/MinIO.
2. Añadir CDN.
3. Migrar búsquedas a Elasticsearch si la carga lo requiere.
4. Desplegar servicios en contenedores en k8s o en PaaS (Heroku, Render, Vercel para frontend).

Database design proposal:

Propuesta de entidades para la base de datos. Minimas para el MVP(1ra fase).

1. **users**
    - id, email, password_hash, name, role, phone, created_at, verified
    - Índices: email (unique).
2. **listings**
    - id, user_id, title, description, price, currency, type, operation, bedrooms, bathrooms, area_m2, lat, lon, address, district, status (draft/published/archived), created_at, updated_at
    - Índices: (status, created_at), tsvector(title, description), price, district. Para geo: index lat/lon (PostGIS or GiST if PostGIS used).
3. **images**
    - id, listing_id, url, sort_order, width, height, filesize, created_at.
    - Índice: listing_id.

4. **leads**
    - id, listing_id, from_name, from_email, from_phone, message, created_at, source.
5. **favorites**
    - id, user_id, listing_id, created_at.
    - Índices: user_id.
6. **alerts**
    - id, user_id, query_json, frequency, last_sent_at, active.

APIS minimas para implementación.

- POST /api/auth/register — registro.
- POST /api/auth/login — login → devuelve tokens.
- GET /api/listings — búsqueda y filtros (paginado).
- GET /api/listings/:id — detalle.
- POST /api/listings — crear (multipart).
- PUT /api/listings/:id — editar (propietario).
- POST /api/listings/:id/contact — crear lead.
- POST /api/favorites / GET /api/favorites — gestionar favoritos.
- GET /api/agents/:id/leads — leads para anunciante.
- POST /api/alerts / GET /api/alerts — guardar búsquedas.

Additional information: