**Arquitectura Técnica Detallada - App Turisgal**

**1. ARQUITECTURA DEL SISTEMA**

**Diagrama de Arquitectura General**

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐

│ App Móvil │ │ Panel Web │ │ API Gateway │

│ (React Native) │────│ (React.js) │────│ (Express.js) │

└─────────────────┘ └─────────────────┘ └─────────────────┘

│ │ │

└───────────────────────┼───────────────────────┘

│

┌────────────────────┼────────────────────┐

│ │ │

┌───────▼──────┐ ┌───────▼──────┐ ┌───────▼──────┐

│ Auth Service │ │ File Storage │ │ Database │

│ (Firebase) │ │ (AWS S3) │ │ (PostgreSQL) │

└──────────────┘ └──────────────┘ └──────────────┘

**Microservicios Propuestos**

* **Auth Service**: Gestión de usuarios y autenticación
* **Booking Service**: Manejo de reservas y check-in/out
* **Property Service**: Gestión de alojamientos
* **Review Service**: Sistema de reseñas y valoraciones
* **Notification Service**: Push notifications y emails
* **File Service**: Gestión de documentos e imágenes

**2. BASE DE DATOS - DISEÑO DETALLADO**

**Esquema PostgreSQL**

sql

*-- Usuarios*

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

email VARCHAR(255) UNIQUE NOT NULL,

phone VARCHAR(20),

password\_hash VARCHAR(255),

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

date\_of\_birth DATE,

nationality VARCHAR(3), *-- ISO country code*

profile\_image\_url TEXT,

preferred\_language VARCHAR(5) DEFAULT 'es',

is\_verified BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

*-- Propietarios/Administradores*

CREATE TABLE property\_owners (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

company\_name VARCHAR(255),

contact\_name VARCHAR(255) NOT NULL,

phone VARCHAR(20),

tax\_id VARCHAR(50), *-- NIF/CIF*

role VARCHAR(20) DEFAULT 'owner', *-- owner, admin, staff*

permissions JSONB, *-- Array de permisos específicos*

created\_at TIMESTAMP DEFAULT NOW()

);

*-- Propiedades/Alojamientos*

CREATE TABLE properties (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

owner\_id UUID REFERENCES property\_owners(id),

name VARCHAR(255) NOT NULL,

description TEXT,

property\_type VARCHAR(50), *-- hotel, apartment, house, etc.*

address JSONB NOT NULL, *-- {street, city, postal\_code, country, coordinates}*

total\_rooms INTEGER DEFAULT 1,

max\_guests INTEGER NOT NULL,

amenities JSONB, *-- Array de servicios*

house\_rules TEXT,

check\_in\_time TIME DEFAULT '15:00',

check\_out\_time TIME DEFAULT '11:00',

qr\_code\_data TEXT UNIQUE, *-- Código QR único*

images JSONB, *-- Array de URLs de imágenes*

is\_active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT NOW()

);

*-- Habitaciones (para hoteles con múltiples habitaciones)*

CREATE TABLE rooms (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

property\_id UUID REFERENCES properties(id),

room\_number VARCHAR(10) NOT NULL,

room\_type VARCHAR(50), *-- single, double, suite, etc.*

max\_guests INTEGER DEFAULT 2,

price\_per\_night DECIMAL(10,2),

qr\_code\_data TEXT UNIQUE,

is\_available BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT NOW(),

UNIQUE(property\_id, room\_number)

);

*-- Reservas*

CREATE TABLE bookings (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id),

property\_id UUID REFERENCES properties(id),

room\_id UUID REFERENCES rooms(id), *-- NULL para propiedades completas*

booking\_reference VARCHAR(20) UNIQUE NOT NULL,

check\_in\_date DATE NOT NULL,

check\_out\_date DATE NOT NULL,

guests\_count INTEGER NOT NULL,

guest\_details JSONB, *-- Información de huéspedes adicionales*

total\_amount DECIMAL(10,2),

booking\_status VARCHAR(20) DEFAULT 'confirmed', *-- confirmed, checked\_in, checked\_out, cancelled*

special\_requests TEXT,

created\_at TIMESTAMP DEFAULT NOW()

);

*-- Check-ins*

CREATE TABLE check\_ins (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

booking\_id UUID REFERENCES bookings(id),

user\_id UUID REFERENCES users(id),

property\_id UUID REFERENCES properties(id),

check\_in\_timestamp TIMESTAMP DEFAULT NOW(),

identity\_document\_url TEXT, *-- URL del documento escaneado*

selfie\_url TEXT, *-- URL de la selfie de verificación*

digital\_signature TEXT, *-- Firma digital*

device\_info JSONB, *-- Información del dispositivo*

location\_coordinates POINT, *-- Coordenadas GPS del check-in*

verification\_status VARCHAR(20) DEFAULT 'pending', *-- pending, verified, failed*

verified\_by UUID REFERENCES property\_owners(id), *-- Staff que verificó*

notes TEXT

);

*-- Check-outs*

CREATE TABLE check\_outs (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

booking\_id UUID REFERENCES bookings(id),

check\_in\_id UUID REFERENCES check\_ins(id),

check\_out\_timestamp TIMESTAMP DEFAULT NOW(),

room\_condition\_photos JSONB, *-- Array de URLs de fotos*

damages\_reported TEXT,

additional\_charges DECIMAL(10,2) DEFAULT 0,

guest\_signature TEXT,

staff\_notes TEXT,

processed\_by UUID REFERENCES property\_owners(id)

);

*-- Reseñas*

CREATE TABLE reviews (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

booking\_id UUID REFERENCES bookings(id),

user\_id UUID REFERENCES users(id),

property\_id UUID REFERENCES properties(id),

overall\_rating INTEGER CHECK (overall\_rating >= 1 AND overall\_rating <= 5),

cleanliness\_rating INTEGER CHECK (cleanliness\_rating >= 1 AND cleanliness\_rating <= 5),

location\_rating INTEGER CHECK (location\_rating >= 1 AND location\_rating <= 5),

value\_rating INTEGER CHECK (value\_rating >= 1 AND value\_rating <= 5),

service\_rating INTEGER CHECK (service\_rating >= 1 AND service\_rating <= 5),

comment TEXT,

photos JSONB, *-- Array de URLs de fotos*

is\_anonymous BOOLEAN DEFAULT FALSE,

response\_from\_owner TEXT,

response\_date TIMESTAMP,

created\_at TIMESTAMP DEFAULT NOW()

);

*-- Notificaciones*

CREATE TABLE notifications (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id),

property\_owner\_id UUID REFERENCES property\_owners(id),

notification\_type VARCHAR(50), *-- check\_in\_reminder, review\_request, etc.*

title VARCHAR(255) NOT NULL,

message TEXT NOT NULL,

data JSONB, *-- Datos adicionales para deep linking*

is\_read BOOLEAN DEFAULT FALSE,

sent\_at TIMESTAMP DEFAULT NOW()

);

*-- Tokens de dispositivos para push notifications*

CREATE TABLE device\_tokens (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id),

property\_owner\_id UUID REFERENCES property\_owners(id),

token TEXT UNIQUE NOT NULL,

platform VARCHAR(10), *-- ios, android*

is\_active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT NOW()

);

*-- Índices para optimización*

CREATE INDEX idx\_bookings\_user\_id ON bookings(user\_id);

CREATE INDEX idx\_bookings\_property\_id ON bookings(property\_id);

CREATE INDEX idx\_bookings\_dates ON bookings(check\_in\_date, check\_out\_date);

CREATE INDEX idx\_bookings\_status ON bookings(booking\_status);

CREATE INDEX idx\_check\_ins\_booking\_id ON check\_ins(booking\_id);

CREATE INDEX idx\_reviews\_property\_id ON reviews(property\_id);

CREATE INDEX idx\_notifications\_user\_id ON notifications(user\_id);

**3. API ENDPOINTS DETALLADOS**

**Autenticación**

javascript

POST /api/auth/register

POST /api/auth/login

POST /api/auth/logout

POST /api/auth/refresh-token

POST /api/auth/forgot-password

POST /api/auth/reset-password

POST /api/auth/verify-phone

POST /api/auth/verify-email

**Usuarios**

javascript

GET /api/users/profile

PUT /api/users/profile

POST /api/users/upload-avatar

GET /api/users/bookings

GET /api/users/reviews

DELETE /api/users/account

**Propiedades**

javascript

GET /api/properties *// Lista pública*

GET /api/properties/:id *// Detalle público*

POST /api/properties *// Solo owners*

PUT /api/properties/:id *// Solo owners*

DELETE /api/properties/:id *// Solo owners*

GET /api/properties/:id/qr-code *// Generar QR*

POST /api/properties/:id/images *// Subir imágenes*

**Reservas**

javascript

GET /api/bookings *// Usuario: sus reservas, Owner: de sus propiedades*

GET /api/bookings/:id

POST /api/bookings *// Crear nueva reserva*

PUT /api/bookings/:id *// Modificar reserva*

DELETE /api/bookings/:id *// Cancelar reserva*

GET /api/bookings/:id/qr-code *// QR para check-in*

**Check-in/Check-out**

javascript

POST /api/checkin/scan-qr *// Validar QR y obtener info de reserva*

POST /api/checkin/verify-identity *// Subir documento y selfie*

POST /api/checkin/complete *// Finalizar check-in*

GET /api/checkin/:bookingId *// Estado del check-in*

POST /api/checkout/start *// Iniciar check-out*

POST /api/checkout/upload-photos *// Fotos del estado*

POST /api/checkout/report-damage *// Reportar daños*

POST /api/checkout/complete *// Finalizar check-out*

**Reseñas**

javascript

GET /api/reviews/property/:id *// Reseñas de una propiedad*

POST /api/reviews *// Crear reseña*

PUT /api/reviews/:id *// Editar reseña*

DELETE /api/reviews/:id *// Eliminar reseña*

POST /api/reviews/:id/response *// Respuesta del propietario*

**Panel Administrativo**

javascript

GET /api/admin/dashboard *// Métricas principales*

GET /api/admin/bookings *// Todas las reservas con filtros*

GET /api/admin/check-ins-today *// Check-ins del día*

GET /api/admin/reviews *// Gestión de reseñas*

GET /api/admin/analytics *// Reportes y análisis*

POST /api/admin/bulk-import *// Importar datos masivamente*

**4. SEGURIDAD Y AUTENTICACIÓN**

**JWT Token Structure**

javascript

{

"header": {

"alg": "HS256",

"typ": "JWT"

},

"payload": {

"userId": "uuid",

"email": "user@example.com",

"role": "guest|owner|admin",

"permissions": ["read:bookings", "write:reviews"],

"iat": 1635724800,

"exp": 1635811200

}

}

**Middleware de Seguridad**

javascript

*// Rate Limiting*

const rateLimit = require('express-rate-limit');

const limiter = rateLimit({

windowMs: 15 \* 60 \* 1000, *// 15 minutes*

max: 100 *// limit each IP to 100 requests per windowMs*

});

*// Validación de entrada*

const { body, validationResult } = require('express-validator');

*// Sanitización*

const mongoSanitize = require('express-mongo-sanitize');

const xss = require('xss-clean');

*// Headers de seguridad*

const helmet = require('helmet');

**Encriptación y Hashing**

javascript

const bcrypt = require('bcryptjs');

const crypto = require('crypto');

*// Hashing de contraseñas*

const hashPassword = async (password) => {

return await bcrypt.hash(password, 12);

};

*// Encriptación de datos sensibles*

const encrypt = (text) => {

const cipher = crypto.createCipher('aes-256-cbc', process.env.ENCRYPTION\_KEY);

let encrypted = cipher.update(text, 'utf8', 'hex');

encrypted += cipher.final('hex');

return encrypted;

};

**5. ALMACENAMIENTO DE ARCHIVOS**

**Configuración AWS S3**

javascript

const AWS = require('aws-sdk');

const s3 = new AWS.S3({

accessKeyId: process.env.AWS\_ACCESS\_KEY\_ID,

secretAccessKey: process.env.AWS\_SECRET\_ACCESS\_KEY,

region: process.env.AWS\_REGION

});

*// Estructura de carpetas*

const uploadFile = async (file, folder, userId) => {

const key = `${folder}/${userId}/${Date.now()}-${file.originalname}`;

const params = {

Bucket: process.env.S3\_BUCKET,

Key: key,

Body: file.buffer,

ContentType: file.mimetype,

ServerSideEncryption: 'AES256'

};

return await s3.upload(params).promise();

};

*// Carpetas organizadas:*

*// /documents/{userId}/{timestamp}-document.jpg*

*// /selfies/{userId}/{timestamp}-selfie.jpg*

*// /properties/{propertyId}/images/{timestamp}-image.jpg*

*// /room-photos/{bookingId}/{timestamp}-photo.jpg*

**Políticas de Retención**

* Documentos de identidad: 6 años (legislación española)
* Selfies de verificación: 1 año
* Fotos de propiedades: Indefinido mientras esté activa
* Fotos de estado de habitaciones: 2 años

**6. NOTIFICACIONES PUSH**

**Configuración Firebase**

javascript

const admin = require('firebase-admin');

admin.initializeApp({

credential: admin.credential.cert({

projectId: process.env.FIREBASE\_PROJECT\_ID,

clientEmail: process.env.FIREBASE\_CLIENT\_EMAIL,

privateKey: process.env.FIREBASE\_PRIVATE\_KEY.replace(/\\n/g, '\n')

})

});

*// Tipos de notificaciones*

const notificationTypes = {

CHECK\_IN\_REMINDER: {

title: 'Recordatorio de Check-in',

body: 'Tu check-in es mañana. ¿Todo listo?',

icon: 'checkin\_icon'

},

CHECK\_OUT\_REMINDER: {

title: 'Hora del Check-out',

body: 'No olvides hacer el check-out antes de las {time}',

icon: 'checkout\_icon'

},

REVIEW\_REQUEST: {

title: '¿Cómo fue tu estancia?',

body: 'Ayuda a otros viajeros con tu reseña',

icon: 'review\_icon'

}

};

**Sistema de Colas para Notificaciones**

javascript

const Queue = require('bull');

const notificationQueue = new Queue('notification processing');

*// Programar notificaciones*

const scheduleNotifications = async (booking) => {

*// Recordatorio 24h antes del check-in*

const checkInReminder = new Date(booking.check\_in\_date);

checkInReminder.setDate(checkInReminder.getDate() - 1);

await notificationQueue.add('send-notification', {

userId: booking.user\_id,

type: 'CHECK\_IN\_REMINDER',

bookingId: booking.id

}, { delay: checkInReminder.getTime() - Date.now() });

*// Recordatorio de check-out el mismo día*

const checkOutReminder = new Date(booking.check\_out\_date);

checkOutReminder.setHours(9, 0, 0, 0); *// 9 AM*

await notificationQueue.add('send-notification', {

userId: booking.user\_id,

type: 'CHECK\_OUT\_REMINDER',

bookingId: booking.id

}, { delay: checkOutReminder.getTime() - Date.now() });

};

**7. MONITOREO Y LOGGING**

**Configuración de Logs**

javascript

const winston = require('winston');

const logger = winston.createLogger({

level: 'info',

format: winston.format.combine(

winston.format.timestamp(),

winston.format.errors({ stack: true }),

winston.format.json()

),

defaultMeta: { service: 'turisgal-api' },

transports: [

new winston.transports.File({ filename: 'error.log', level: 'error' }),

new winston.transports.File({ filename: 'combined.log' }),

new winston.transports.Console({

format: winston.format.simple()

})

]

});

*// Logging de eventos críticos*

const logCriticalEvent = (eventType, data, userId = null) => {

logger.info('Critical Event', {

eventType,

data,

userId,

timestamp: new Date().toISOString(),

requestId: data.requestId

});

};

**Métricas y Monitoring**

javascript

const promClient = require('prom-client');

*// Métricas personalizadas*

const checkinCounter = new promClient.Counter({

name: 'checkins\_total',

help: 'Total number of check-ins',

labelNames: ['property\_id', 'status']

});

const apiDuration = new promClient.Histogram({

name: 'api\_duration\_seconds',

help: 'Duration of API requests in seconds',

labelNames: ['method', 'route', 'status']

});

*// Middleware para medir duración de requests*

const measureDuration = (req, res, next) => {

const start = Date.now();

res.on('finish', () => {

const duration = (Date.now() - start) / 1000;

apiDuration

.labels(req.method, req.route?.path || req.url, res.statusCode)

.observe(duration);

});

next();

};