

```

<Grid item xs={12} sm={6} md={3}>
  <MetricCard
    title="Ocupación"
    value={`${data.todayMetrics.occupancy}%`}
    icon={<Hotel />}
    color="primary"
    trend={+2}
  />
</Grid>
<Grid item xs={12} sm={6} md={3}>
  <MetricCard
    title="Rating Promedio"
    value={data.todayMetrics.averageRating.toFixed(1)}
    icon={<Star />}
    color="warning"
    trend={+0.1}
  />
</Grid>
</Grid>

```

```

<Grid container spacing={3}>
  /* Gráfico de check-ins por hora */
  <Grid item xs={12} md={8}>
    <Card>
      <CardContent>
        <Typography variant="h6" sx={{ mb: 2 }}>
          Check-ins por Hora
        </Typography>
        <ResponsiveContainer width="100%" height={300}>
          <LineChart data={data.checkInTrend}>
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis dataKey="time" />
            <YAxis />
            <Tooltip />
            <Line
              type="monotone"
              dataKey="count"
              stroke="#1976d2"
              strokeWidth={2}
            />
          </LineChart>
        </ResponsiveContainer>
      </CardContent>
    </Card>
  </Grid>

```

```
/* Alertas activas */
<Grid item xs={12} md={4}>
  <AlertsList alerts={data.alerts} />
</Grid>

/* Actividad reciente */
<Grid item xs={12}>
  <RecentActivity checkIns={data.recentCheckIns} />
</Grid>
</Grid>
</Box>

);

};
```

```
#### Componente de Gestión de Reservas
```typescript
// src/pages/Bookings/BookingsManagement.tsx
import React, { useState, useEffect } from 'react';
import {
 Box,
 Card,
 CardContent,
 Typography,
 Table,
 TableBody,
 TableCell,
 TableContainer,
 TableHead,
 TableRow,
 Button,
 Chip,
 IconButton,
 Dialog,
 DialogTitle,
 DialogContent,
 DialogActions,
 TextField,
 FormControl,
 InputLabel,
 Select,
 MenuItem,
 Pagination
} from '@mui/material';
import {
 Visibility,
 Edit,
 Phone,
 Email,
 Download
} from '@mui/icons-material';
import { bookingsService } from '../services/bookingsService';

interface Booking {
 id: string;
 reference: string;
 guest: {
 name: string;
 email: string;
 phone: string;
 }
}
```

```
};

property: {
 name: string;
 room?: string;
};

dates: {
 checkIn: string;
 checkOut: string;
};

status: 'confirmed' | 'checked_in' | 'checked_out' | 'cancelled';
totalAmount: number;
}
```

```
interface BookingFilters {
 status: string;
 property: string;
 dateRange: string;
 search: string;
}
```

```
export const BookingsManagement: React.FC = () => {
 const [bookings, setBookings] = useState<Booking[]>([]);
 const [loading, setLoading] = useState(true);
 const [filters, setFilters] = useState<BookingFilters>({
 status: 'all',
 property: 'all',
 dateRange: 'today',
 search: ""
 });
 const [page, setPage] = useState(1);
 const [totalPages, setTotalPages] = useState(1);
 const [selectedBooking, setSelectedBooking] = useState<Booking | null>(null);
 const [detailsOpen, setDetailsOpen] = useState(false);

 useEffect(() => {
 loadBookings();
 }, [filters, page]);

 const loadBookings = async () => {
 setLoading(true);
 try {
 const response = await bookingsService.getBookings({
 ...filters,
 page,
 limit: 20
 });
 setBookings(response.data);
 } catch (error) {
 console.error('Error loading bookings:', error);
 }
 };
}
```

```
 setTotalPages(response.totalPages);
} catch (error) {
 console.error('Error loading bookings:', error);
} finally {
 setLoading(false);
}
};

const getStatusColor = (status: string) => {
switch (status) {
 case 'confirmed': return 'info';
 case 'checked_in': return 'success';
 case 'checked_out': return 'default';
 case 'cancelled': return 'error';
 default: return 'default';
}
};

const getStatusLabel = (status: string) => {
switch (status) {
 case 'confirmed': return 'Confirmada';
 case 'checked_in': return 'Check-in ';
 case 'checked_out': return 'Completada';
 case 'cancelled': return 'Cancelada';
 default: return status;
}
};

const handleViewDetails = (booking: Booking) => {
 setSelectedBooking(booking);
 setDetailsOpen(true);
};

const handleExport = async () => {
try {
 const blob = await bookingsService.exportBookings(filters);
 const url = window.URL.createObjectURL(blob);
 const a = document.createElement('a');
 a.href = url;
 a.download = `reservas_${new Date().toISOString().split('T')[0]}.csv`;
 a.click();
} catch (error) {
 console.error('Error exporting bookings:', error);
}
};

return (

```

```
<Box sx={{ p: 3 }}>
 <Box sx={{ display: 'flex', justifyContent: 'space-between', alignItems: 'center', mb: 3 }}>
 <Typography variant="h4">Gestión de Reservas</Typography>
 <Button
 variant="outlined"
 startIcon={<Download />}
 onClick={handleExport}
 >
 Exportar CSV
 </Button>
 </Box>
```

```
/* Filtros */
<Card sx={{ mb: 3 }}>
 <CardContent>
 <Box sx={{ display: 'flex', gap: 2, flexWrap: 'wrap' }}>
 <TextField
 label="Buscar"
 variant="outlined"
 size="small"
 value={filters.search}
 onChange={(e) => setFilters({ ...filters, search: e.target.value })}
 placeholder="Nombre, email, referencia..."
 sx={{ minWidth: 200 }}
 />
```

```
<FormControl size="small" sx={{ minWidth: 120 }}>
 <InputLabel>Estado</InputLabel>
 <Select
 value={filters.status}
 label="Estado"
 onChange={(e) => setFilters({ ...filters, status: e.target.value })}
 >
 <MenuItem value="all">Todos</MenuItem>
 <MenuItem value="confirmed">Confirmadas</MenuItem>
 <MenuItem value="checked_in">Check-in</MenuItem>
 <MenuItem value="checked_out">Completadas</MenuItem>
 <MenuItem value="cancelled">Canceladas</MenuItem>
 </Select>
</FormControl>
```

```
<FormControl size="small" sx={{ minWidth: 150 }}>
 <InputLabel>Período</InputLabel>
 <Select
 value={filters.dateRange}
 label="Período"
 onChange={(e) => setFilters({ ...filters, dateRange: e.target.value })}
```

```

>
<MenuItem value="today">Hoy</MenuItem>
<MenuItem value="week">Esta semana</MenuItem>
<MenuItem value="month">Este mes</MenuItem>
<MenuItem value="all">Todos</MenuItem>
</Select>
</FormControl>

<Button
 variant="contained"
 onClick={loadBookings}
 disabled={loading}
>
 Filtrar
</Button>
</Box>
</CardContent>
</Card>

/* Tabla de reservas */
<Card>
<CardContent>
<TableContainer>
<Table>
<TableHead>
<TableRow>
<TableCell>Referencia</TableCell>
<TableCell>Huésped</TableCell>
<TableCell>Alojamiento</TableCell>
<TableCell>Fechas</TableCell>
<TableCell>Estado</TableCell>
<TableCell>Importe</TableCell>
<TableCell>Acciones</TableCell>
</TableRow>
</TableHead>
<TableBody>
{bookings.map((booking) => (
 <TableRow key={booking.id}>
 <TableCell>
 <Typography variant="body2" fontWeight="bold">
 {booking.reference}
 </Typography>
 </TableCell>
 <TableCell>
 <Box>
 <Typography variant="body2">{booking.guest.name}</Typography>
 <Typography variant="caption" color="text.secondary">

```

```
{booking.guest.email}
</Typography>

<Typography variant="caption" color="text.secondary">
 {booking.guest.phone}
</Typography>
</Box>
</TableCell>
<TableCell>
<Box>
 <Typography variant="body2">{booking.property.name}</Typography>
 {booking.property.room} && (
 <Typography variant="caption" color="text.secondary">
 {booking.property.room}
 </Typography>
)})
</Box>
</TableCell>
<TableCell>
<Box>
 <Typography variant="body2">
 {new Date(booking.dates.checkIn).toLocaleDateString('es-ES')}
 </Typography>
 <Typography variant="caption" color="text.secondary">
 hasta {new Date(booking.dates.checkOut).toLocaleDateString('es-ES')}
 </Typography>
</Box>
</TableCell>
<TableCell>
<Chip
 label={getStatusLabel(booking.status)}
 color={getStatusColor(booking.status) as any}
 size="small"
/>
</TableCell>
<TableCell>
<Typography variant="body2" fontWeight="bold">
 €{booking.totalAmount.toFixed(2)}
</Typography>
</TableCell>
<TableCell>
<Box sx={{ display: 'flex', gap: 1 }}>
 <IconButton
 size="small"
 onClick={() => handleViewDetails(booking)}
 >
 <Visibility />
 </Box>
</TableCell>
```

```

</IconButton>
<IconButton size="small">
 <Edit />
</IconButton>
<IconButton size="small">
 <Phone />
</IconButton>
<IconButton size="small">
 <Email />
</IconButton>
</Box>
</TableCell>
</TableRow>
)})}
</TableBody>
</Table>
</TableContainer>

<Box sx={{ display: 'flex', justifyContent: 'center', mt: 3 }}>
 <Pagination
 count={totalPages}
 page={page}
 onChange={({ newPage }) => setPage(newPage)}
 color="primary"
 />
</Box>
</CardContent>
</Card>

/* Dialog de detalles */
<Dialog
 open={detailsOpen}
 onClose={() => setDetailsOpen(false)}
 maxWidth="md"
 fullWidth
>
 <DialogTitle>
 Detalles de Reserva - {selectedBooking?.reference}
 </DialogTitle>
 <DialogContent>
 {selectedBooking && (
 <BookingDetails booking={selectedBooking} />
)}
 </DialogContent>
 <DialogActions>
 <Button onClick={() => setDetailsOpen(false)}>Cerrar</Button>
 </DialogActions>

```

```
</Dialog>
</Box>
);
}
```

## 2. CONFIGURACIÓN DE DESPLIEGUE

### Docker Configuration

#### Dockerfile para API

dockerfile

```
Dockerfile
FROM node:18-alpine AS builder

WORKDIR /app

Copiar package files
COPY package*.json .
COPY prisma ./prisma/

Instalar dependencias
RUN npm ci --only=production

Generar Prisma client
RUN npx prisma generate

Copiar código fuente
COPY ..

Build aplicación
RUN npm run build

Imagen de producción
FROM node:18-alpine AS production

WORKDIR /app

Instalar dumb-init para manejo de señales
RUN apk add --no-cache dumb-init

Crear usuario no-root
RUN addgroup -g 1001 -S nodejs
RUN adduser -S nextjs -u 1001

Copiar archivos necesarios
COPY --from=builder --chown=nextjs:nodejs /app/dist ./dist
COPY --from=builder --chown=nextjs:nodejs /app/node_modules ./node_modules
COPY --from=builder --chown=nextjs:nodejs /app/package.json ./package.json
COPY --from=builder --chown=nextjs:nodejs /app/prisma ./prisma

USER nextjs

EXPOSE 3000

ENV NODE_ENV=production
```

```
CMD ["dumb-init", "node", "dist/index.js"]
```

## Docker Compose

yaml

```

docker-compose.yml
version: '3.8'

services:
 # Base de datos PostgreSQL
 postgres:
 image: postgres:15-alpine
 environment:
 POSTGRES_DB: turisgal
 POSTGRES_USER: turisgal_user
 POSTGRES_PASSWORD: ${DB_PASSWORD}
 volumes:
 - postgres_data:/var/lib/postgresql/data
 - ./init.sql:/docker-entrypoint-initdb.d/init.sql
 ports:
 - "5432:5432"
 networks:
 - turisgal-network

 # Redis para caching y colas
 redis:
 image: redis:7-alpine
 command: redis-server --appendonly yes
 volumes:
 - redis_data:/data
 ports:
 - "6379:6379"
 networks:
 - turisgal-network

 # API Backend
 api:
 build:
 context: ./server
 dockerfile: Dockerfile
 environment:
 NODE_ENV: production
 DATABASE_URL: postgres://turisgal_user:${DB_PASSWORD}@postgres:5432/turisgal
 REDIS_URL: redis://redis:6379
 JWT_SECRET: ${JWT_SECRET}
 AWS_ACCESS_KEY_ID: ${AWS_ACCESS_KEY_ID}
 AWS_SECRET_ACCESS_KEY: ${AWS_SECRET_ACCESS_KEY}
 S3_BUCKET: ${S3_BUCKET}
 TWILIO_ACCOUNT_SID: ${TWILIO_ACCOUNT_SID}
 TWILIO_AUTH_TOKEN: ${TWILIO_AUTH_TOKEN}
 SMTP_HOST: ${SMTP_HOST}

```

```
SMTP_USER: ${SMTP_USER}
SMTP_PASS: ${SMTP_PASS}
ports:
- "3000:3000"
depends_on:
- postgres
- redis
networks:
- turisgal-network
volumes:
- ./uploads:/app/uploads
restart: unless-stopped
```

*# Panel Web Administrativo*

```
admin-panel:
build:
context: ./admin-panel
dockerfile: Dockerfile
environment:
REACT_APP_API_URL: http://api:3000
ports:
- "3001:80"
depends_on:
- api
networks:
- turisgal-network
restart: unless-stopped
```

*# Nginx como reverse proxy*

```
nginx:
image: nginx:alpine
ports:
- "80:80"
- "443:443"
volumes:
- ./nginx.conf:/etc/nginx/nginx.conf
- ./ssl:/etc/nginx/ssl
depends_on:
- api
- admin-panel
networks:
- turisgal-network
restart: unless-stopped
```

volumes:

```
postgres_data:
redis_data:
```

`networks:`

`turisgal-network:`

`driver: bridge`

## **Configuración Nginx**

nginx

```
nginx.conf

events {
 worker_connections 1024;
}

http {
 upstream api {
 server api:3000;
 }

 upstream admin {
 server admin-panel:80;
 }

 # Redirects HTTP to HTTPS
 server {
 listen 80;
 server_name turisgal-api.com admin.turisgal.com;
 return 301 https://$server_name$request_uri;
 }

 # API Server
 server {
 listen 443 ssl http2;
 server_name turisgal-api.com;

 ssl_certificate /etc/nginx/ssl/cert.pem;
 ssl_certificate_key /etc/nginx/ssl/key.pem;

 # Security headers
 add_header X-Frame-Options DENY;
 add_header X-Content-Type-Options nosniff;
 add_header X-XSS-Protection "1; mode=block";
 add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;

 # File upload size
 client_max_body_size 10M;

 location / {
 proxy_pass http://api;
 proxy_set_header Host $host;
 proxy_set_header X-Real-IP $remote_addr;
 proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 proxy_set_header X-Forwarded-Proto $scheme;

 # WebSocket support
 }
}
```

```

proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
}

Health check endpoint
location /health {
 proxy_pass http://api/health;
 access_log off;
}
}

Admin Panel
server {
 listen 443 ssl http2;
 server_name admin.turisgal.com;

 ssl_certificate /etc/nginx/ssl/cert.pem;
 ssl_certificate_key /etc/nginx/ssl/key.pem;

 # Security headers
 add_header X-Frame-Options DENY;
 add_header X-Content-Type-Options nosniff;
 add_header X-XSS-Protection "1; mode=block";

 location / {
 proxy_pass http://admin;
 proxy_set_header Host $host;
 proxy_set_header X-Real-IP $remote_addr;
 proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 proxy_set_header X-Forwarded-Proto $scheme;
 }
}

Static assets caching
location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
 proxy_pass http://admin;
 expires 1y;
 add_header Cache-Control "public, immutable";
}
}
}

```

### 3. CONFIGURACIÓN AWS

#### S3 Bucket Policy

json

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "TurisgalAppAccess",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::ACCOUNT-ID:user/turisgal-app"
 },
 "Action": [
 "s3:GetObject",
 "s3:PutObject",
 "s3:DeleteObject"
],
 "Resource": "arn:aws:s3:::turisgal-files/*"
 },
 {
 "Sid": "PublicReadAccess",
 "Effect": "Allow",
 "Principal": "*",
 "Action": "s3:GetObject",
 "Resource": "arn:aws:s3:::turisgal-files/public/*"
 }
]
}
```

## CloudFormation Template

yaml

```
cloudformation-template.yml
AWSTemplateFormatVersion: '2010-09-09'
Description: 'TurisGal App Infrastructure'
```

Parameters:

Environment:

Type: String

Default: production

AllowedValues: [development, staging, production]

Resources:

```
S3 Bucket para archivos
```

FilesBucket:

Type: AWS::S3::Bucket

Properties:

BucketName: !Sub 'turisgal-files-\${Environment}'

VersioningConfiguration:

Status: Enabled

PublicAccessBlockConfiguration:

BlockPublicAcls: false

BlockPublicPolicy: false

IgnorePublicAcls: false

RestrictPublicBuckets: false

CorsConfiguration:

CorsRules:

- AllowedHeaders: ["\*"]

- AllowedMethods: [GET, PUT, POST, DELETE]

- AllowedOrigins: ["\*"]

- MaxAge: 3600

```
CloudFront Distribution
```

CDNDistribution:

Type: AWS::CloudFront::Distribution

Properties:

DistributionConfig:

Enabled: true

Comment: !Sub 'TurisGal CDN - \${Environment}'

DefaultRootObject: index.html

Origins:

- Id: S3Origin

- DomainName: !GetAtt FilesBucket.DomainName

- S3OriginConfig:

- OriginAccessIdentity: !Sub 'origin-access-identity/cloudfront/\${OriginAccessIdentity}'

DefaultCacheBehavior:

- TargetOriginId: S3Origin

- ViewerProtocolPolicy: redirect-to-https

```
 AllowedMethods: [GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE]
 CachedMethods: [GET, HEAD, OPTIONS]
 ForwardedValues:
 QueryString: false
 Cookies:
 Forward: none
```

#### # RDS Instance

```
Database:
 Type: AWS::RDS::DBInstance
 Properties:
 DBInstanceIdentifier: !Sub 'turisgal-db-${Environment}'
 DBInstanceClass: db.t3.micro
 Engine: postgres
 EngineVersion: '15.4'
 AllocatedStorage: 20
 StorageType: gp2
 DBName: turisgal
 MasterUsername: turisgal_user
 MasterUserPassword: !Ref DatabasePassword
 VPCSecurityGroups:
 - !Ref DatabaseSecurityGroup
 BackupRetentionPeriod: 7
 MultiAZ: !If [IsProduction, true, false]
```

#### # ElastiCache Redis

```
RedisCluster:
 Type: AWS::ElastiCache::ReplicationGroup
 Properties:
 ReplicationGroupId: !Sub 'turisgal-redis-${Environment}'
 Description: 'Redis cluster for TurisGal'
 Engine: redis
 CacheNodeType: cache.t3.micro
 NumCacheClusters: 1
 Port: 6379
 SecurityGroupIds:
 - !Ref RedisSecurityGroup
```

#### # ECS Cluster

```
ECSCluster:
 Type: AWS::ECS::Cluster
 Properties:
 ClusterName: !Sub 'turisgal-cluster-${Environment}'
```

#### # Application Load Balancer

```
LoadBalancer:
 Type: AWS::ElasticLoadBalancingV2::LoadBalancer
```

Properties:

Name: !Sub 'turisgal-alb-\${Environment}'

Scheme: internet-facing

Type: application

SecurityGroups:

- !Ref LoadBalancerSecurityGroup

Subnets:

- !Ref PublicSubnet1

- !Ref PublicSubnet2

Conditions:

IsProduction: !Equals [!Ref Environment, production]

Outputs:

DatabaseEndpoint:

Description: 'RDS endpoint'

Value: !GetAtt Database.Endpoint.Address

RedisEndpoint:

Description: 'Redis endpoint'

Value: !GetAtt RedisCluster.RedisEndpoint.Address

LoadBalancerDNS:

Description: 'Load balancer DNS name'

Value: !GetAtt LoadBalancer.DNSName

## 4. SCRIPTS DE DESPLIEGUE

### Deploy Script

bash

```

#!/bin/bash
deploy.sh

set -e

ENVIRONMENT=${1:-staging}
VERSION=${2:-latest}

echo "🚀 Desplegando TurisGal App - Entorno: $ENVIRONMENT"

Verificar dependencias
command -v docker >/dev/null 2>&1 || { echo "Docker no está instalado" >&2; exit 1; }
command -v aws >/dev/null 2>&1 || { echo "AWS CLI no está instalado" >&2; exit 1; }

Variables de entorno
export AWS_REGION=eu-west-1
export ECR_REGISTRY=123456789.dkr.ecr.eu-west-1.amazonaws.com
export IMAGE_TAG=$VERSION

Función para logging
log() {
 echo "[$(date +'%Y-%m-%d %H:%M:%S')] $1"
}

Build y push de imágenes Docker
build_and_push() {
 local service=$1
 local dockerfile_path=$2

 log "Building $service..."

 # Build imagen
 docker build -t turisgal-$service:$VERSION -f $dockerfile_path .

 # Tag para ECR
 docker tag turisgal-$service:$VERSION $ECR_REGISTRY/turisgal-$service:$VERSION

 # Push a ECR
 log "Pushing $service to ECR..."
 docker push $ECR_REGISTRY/turisgal-$service:$VERSION
}

Login a ECR
log "Logging in to ECR..."
aws ecr get-login-password --region $AWS_REGION | docker login --username AWS --password-stdin $ECR_REGISTRY

```

```

Build y push de servicios
build_and_push "api" "server/Dockerfile"
build_and_push "admin" "admin-panel/Dockerfile"

Actualizar stack de CloudFormation
log "Updating CloudFormation stack..."
aws cloudformation deploy \
--template-file infrastructure/cloudformation-template.yml \
--stack-name turisgal-$ENVIRONMENT \
--parameter-overrides Environment=$ENVIRONMENT \
--capabilities CAPABILITY_IAM

Ejecutar migraciones de base de datos
log "Running database migrations..."
docker run --rm \
-e DATABASE_URL=$DATABASE_URL \
turisgal-api:$VERSION \
npx prisma migrate deploy

Actualizar servicios ECS
log "Updating ECS services..."
aws ecs update-service \
--cluster turisgal-cluster-$ENVIRONMENT \
--service turisgal-api-$ENVIRONMENT \
--force-new-deployment

aws ecs update-service \
--cluster turisgal-cluster-$ENVIRONMENT \
--service turisgal-admin-$ENVIRONMENT \
--force-new-deployment

Verificar estado de los servicios
log "Waiting for services to be stable..."
aws ecs wait services-stable \
--cluster turisgal-cluster-$ENVIRONMENT \
--services turisgal-api-$ENVIRONMENT turisgal-admin-$ENVIRONMENT

Health check
log "Performing health check..."
LOAD_BALANCER_DNS=$(aws cloudformation describe-stacks \
--stack-name turisgal-$ENVIRONMENT \
--query 'Stacks[0].Outputs[?OutputKey==`LoadBalancerDNS`].OutputValue' \
--output text)

MAX_ATTEMPTS=30
ATTEMPT=1

```

```

while [$ATTEMPT -le $MAX_ATTEMPTS]; do
 if curl -f -s "https://$LOAD_BALANCER_DNS/health" > /dev/null; then
 log "✓ Health check passed!"
 break
 else
 log "Health check attempt $ATTEMPT/$MAX_ATTEMPTS failed, retrying in 10s..."
 sleep 10
 ATTEMPT=$((ATTEMPT + 1))
 fi
done

if [$ATTEMPT -gt $MAX_ATTEMPTS]; then
 log "✗ Health check failed after $MAX_ATTEMPTS attempts"
 exit 1
fi

log "🎉 Deployment completed successfully!"
log "API URL: https://$LOAD_BALANCER_DNS"
log "Admin Panel: https://admin.$LOAD_BALANCER_DNS"

```

## Rollback Script

```

bash

#!/bin/bash
rollback.sh

set -e

ENVIRONMENT=${1:-staging}
TARGET_VERSION=$2

if [-z "$TARGET_VERSION"]; then
 echo "Usage: $0 <environment> <target_version>"
 exit 1
fi

echo "✅ Rolling back to version: $TARGET_VERSION"

Revertir servicios ECS a versión anterior
aws ecs update-service \
--cluster turisgal-cluster-$ENVIRONMENT \
--service turisgal-api-$ENVIRONMENT \
--task-definition turisgal-api:$TARGET_VERSION \
--force-new-deployment

aws ecs update-service \
--cluster turisgal-cluster-$ENVIRONMENT \
--service turisgal-admin-$ENVIRONMENT \
--task-definition turisgal-admin:$TARGET_VERSION \
--force-new-deployment

echo "✅ Rollback completed to version $TARGET_VERSION"

```

## 5. MONITORING Y OBSERVABILIDAD

### Configuración de Logging

typescript

```

// src/utils/logger.ts
import winston from 'winston';
import { ElasticsearchTransport } from 'winston-elasticsearch';

const esTransport = new ElasticsearchTransport({
 level: 'info',
 clientOpts: {
 node: process.env.ELASTICSEARCH_URL,
 auth: {
 username: process.env.ELASTICSEARCH_USER,
 password: process.env.ELASTICSEARCH_PASS
 }
 },
 index: 'turisgal-logs'
});

export const logger = winston.createLogger({
 level: process.env.LOG_LEVEL || 'info',
 format: winston.format.combine(
 winston.format.timestamp(),
 winston.format.errors({ stack: true }),
 winston.format.json()
),
 defaultMeta: {
 service: 'turisgal-api',
 environment: process.env.NODE_ENV
 },
 transports: [
 new winston.transports.Console({
 format: winston.format.simple()
 }),
 new winston.transports.File({
 filename: 'logs/error.log',
 level: 'error'
 }),
 new winston.transports.File({
 filename: 'logs/combined.log'
 })
],
 esTransport
}, [
 exceptionHandlers: [
 new winston.# Especificaciones Técnicas de Implementación - TurisGal
]
]);

```

## ## 1. STACK TECNOLÓGICO DETALLADO

### Frontend Móvil (React Native)

```

Dependencias Principales

``json
{
 "dependencies": {
 "react": "18.2.0",
 "react-native": "0.72.6",
 "expo": "~49.0.15",
 "@react-navigation/native": "^6.1.9",
 "@react-navigation/stack": "^6.3.20",
 "@react-navigation/bottom-tabs": "^6.5.11",
 "expo-camera": "~13.6.0",
 "expo-barcode-scanner": "~12.6.0",
 "expo-image-picker": "~14.5.2",
 "expo-location": "~16.3.0",
 "expo-notifications": "~0.23.0",
 "@react-native-async-storage/async-storage": "1.19.3",
 "react-native-qrcode-scanner": "^1.5.5",
 "react-native-signature-canvas": "^4.6.1",
 "react-native-image-resizer": "^3.0.4",
 "@reduxjs/toolkit": "^1.9.7",
 "react-redux": "^8.1.3",
 "axios": "^1.6.0",
 "react-native-paper": "^5.11.1",
 "react-native-vector-icons": "^10.0.2",
 "react-hook-form": "^7.47.0",
 "yup": "^1.3.3",
 "react-native-reanimated": "~3.5.4",
 "react-native-gesture-handler": "~2.12.0",
 "expo-secure-store": "~12.5.0",
 "react-native-super-grid": "^4.4.4",
 "react-native-calendars": "^1.1302.0"
 },
 "devDependencies": {
 "@types/react": "~18.2.14",
 "@types/react-native": "~0.72.2",
 "typescript": "^5.1.3",
 "eslint": "^8.51.0",
 "prettier": "^3.0.3",
 "@testing-library/react-native": "^12.3.2",
 "jest": "^29.2.1"
 }
}

```

## Estructura de Carpetas React Native

```
src/
| └── components/ # Componentes reutilizables
| | └── ui/ # Componentes básicos (Button, Input, etc.)
| | └── forms/ # Componentes de formularios
| | └── camera/ # Componentes relacionados con cámara
| └── navigation/ # Componentes de navegación
| └── screens/ # Pantallas principales
| | └── auth/ # Autenticación
| | └── dashboard/ # Dashboard principal
| | └── checkin/ # Proceso de check-in
| | └── checkout/ # Proceso de check-out
| | └── bookings/ # Gestión de reservas
| | └── reviews/ # Sistema de reseñas
| └── profile/ # Perfil de usuario
| └── navigation/ # Configuración de navegación
| └── store/ # Redux store y slices
| └── services/ # Servicios API y utilidades
| └── hooks/ # Custom hooks
| └── utils/ # Funciones utilitarias
| └── constants/ # Constantes y configuración
| └── types/ # TypeScript types
| └── assets/ # Imágenes, iconos, etc.
```

## Configuración Redux Store

typescript

```
// store/index.ts

import { configureStore } from '@reduxjs/toolkit';
import authSlice from './slices/authSlice';
import bookingsSlice from './slices/bookingsSlice';
import checkinSlice from './slices/checkinSlice';
import notificationsSlice from './slices/notificationsSlice';

export const store = configureStore({
 reducer: {
 auth: authSlice,
 bookings: bookingsSlice,
 checkin: checkinSlice,
 notifications: notificationsSlice,
 },
 middleware: (getDefaultMiddleware) =>
 getDefaultMiddleware({
 serializableCheck: {
 ignoredActions: ['persist/PERSIST'],
 },
 }),
});
}

export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

## Slice de Autenticación

typescript

```
// store/slices/authSlice.ts

import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
import { authService } from '../services/authService';

interface AuthState {
 user: User | null;
 token: string | null;
 isLoading: boolean;
 error: string | null;
 isAuthenticated: boolean;
}

export const loginUser = createAsyncThunk(
 'auth/loginUser',
 async (credentials: LoginCredentials, { rejectWithValue }) => {
 try {
 const response = await authService.login(credentials);
 return response.data;
 } catch (error: any) {
 return rejectWithValue(error.response.data.message);
 }
 }
);

const authSlice = createSlice({
 name: 'auth',
 initialState: {
 user: null,
 token: null,
 isLoading: false,
 error: null,
 isAuthenticated: false,
 } as AuthState,
 reducers: {
 clearError: (state) => {
 state.error = null;
 },
 logout: (state) => {
 state.user = null;
 state.token = null;
 state.isAuthenticated = false;
 },
 },
 extraReducers: (builder) => {
 builder
 .addCase(loginUser.pending, (state) => {
```

```

state.isLoading = true;
state.error = null;
})
.addCase(loginUser.fulfilled, (state, action) => {
 state.isLoading = false;
 state.user = action.payload.user;
 state.token = action.payload.token;
 state.isAuthenticated = true;
})
.addCase(loginUser.rejected, (state, action) => {
 state.isLoading = false;
 state.error = action.payload as string;
});
},
);
};

export const { clearError, logout } = authSlice.actions;
export default authSlice.reducer;

```

## Backend API (Node.js + Express)

### Estructura del Proyecto Backend

```

server/
 └── src/
 ├── controllers/ # Controladores de rutas
 ├── middleware/ # Middleware personalizado
 ├── models/ # Modelos de base de datos (Prisma)
 ├── routes/ # Definición de rutas
 ├── services/ # Lógica de negocio
 ├── utils/ # Utilidades y helpers
 ├── config/ # Configuración de la aplicación
 ├── validators/ # Validadores de entrada
 └── types/ # TypeScript types
 └── prisma/ # Esquemas de base de datos
 ├── schema.prisma # Definición del esquema
 ├── migrations/ # Migraciones de BD
 └── seed.ts # Datos de prueba
 └── tests/ # Tests unitarios e integración
 └── uploads/ # Archivos temporales
 └── logs/ # Archivos de log
 └── docs/ # Documentación API

```

### Package.json Backend

json

```
{
 "name": "turisgal-api",
 "version": "1.0.0",
 "dependencies": {
 "express": "^4.18.2",
 "cors": "^2.8.5",
 "helmet": "^7.1.0",
 "morgan": "^1.10.0",
 "compression": "^1.7.4",
 "express-rate-limit": "^7.1.5",
 "express-validator": "^7.0.1",
 "bcryptjs": "^2.4.3",
 "jsonwebtoken": "^9.0.2",
 "prisma": "^5.6.0",
 "@prisma/client": "^5.6.0",
 "multer": "^1.4.5-lts.1",
 "aws-sdk": "^2.1489.0",
 "nodemailer": "^6.9.7",
 "twilio": "^4.19.0",
 "qrcode": "^1.5.3",
 "jimp": "^0.22.10",
 "winston": "^3.11.0",
 "dotenv": "^16.3.1",
 "joi": "^17.11.0",
 "socket.io": "^4.7.4",
 "redis": "^4.6.10",
 "bull": "^4.12.2",
 "sharp": "^0.32.6",
 "uuid": "^9.0.1"
 },
 "devDependencies": {
 "@types/express": "^4.17.21",
 "@types/node": "^20.8.10",
 "@types/bcryptjs": "^2.4.6",
 "@types/jsonwebtoken": "^9.0.5",
 "@types/multer": "^1.4.11",
 "typescript": "^5.2.2",
 "ts-node": "^10.9.1",
 "nodemon": "^3.0.1",
 "jest": "^29.7.0",
 "supertest": "^6.3.3",
 "@types/jest": "^29.5.7"
 }
}
```

## Configuración de Prisma Schema

prisma

```

// prisma/schema.prisma
generator client {
 provider = "prisma-client-js"
}

datasource db {
 provider = "postgresql"
 url = env("DATABASE_URL")
}

model User {
 id String @id @default(uuid())
 email String @unique
 phone String?
 passwordHash String @map("password_hash")
 firstName String @map("first_name")
 lastName String @map("last_name")
 dateOfBirth DateTime? @map("date_of_birth")
 nationality String?
 profileImageUrl String? @map("profile_image_url")
 preferredLanguage String @default("es") @map("preferred_language")
 isVerified Boolean @default(false) @map("is_verified")
 createdAt DateTime @default(now()) @map("created_at")
 updatedAt DateTime @updatedAt @map("updated_at")

 bookings Booking[]
 checkIns CheckIn[]
 reviews Review[]
 notifications Notification[]
 deviceTokens DeviceToken[]

 @@map("users")
}

model PropertyOwner {
 id String @id @default(uuid())
 email String @unique
 passwordHash String @map("password_hash")
 companyName String? @map("company_name")
 contactName String @map("contact_name")
 phone String?
 taxId String? @map("tax_id")
 role String @default("owner")
 permissions Json?
 createdAt DateTime @default(now()) @map("created_at")
}

```

```

properties Property[]
checkIns CheckIn[]
checkOuts CheckOut[]
notifications Notification[]
deviceTokens DeviceToken[]

@@map("property_owners")
}

model Property {
 id String @id @default(uuid())
 ownerId String @map("owner_id")
 name String
 description String?
 propertyType String @map("property_type")
 address Json
 totalRooms Int @default(1) @map("total_rooms")
 maxGuests Int @map("max_guests")
 amenities Json?
 houseRules String? @map("house_rules")
 checkInTime String @default("15:00") @map("check_in_time")
 checkOutTime String @default("11:00") @map("check_out_time")
 qrCodeData String @unique @map("qr_code_data")
 images Json?
 isActive Boolean @default(true) @map("is_active")
 createdAt DateTime @default(now()) @map("created_at")

 owner PropertyOwner @relation(fields: [ownerId], references: [id])
 rooms Room[]
 bookings Booking[]
 checkIns CheckIn[]
 reviews Review[]

 @@map("properties")
}

model Room {
 id String @id @default(uuid())
 propertyId String @map("property_id")
 roomNumber String @map("room_number")
 roomType String? @map("room_type")
 maxGuests Int @default(2) @map("max_guests")
 pricePerNight Decimal @map("price_per_night")
 qrCodeData String @unique @map("qr_code_data")
 isAvailable Boolean @default(true) @map("is_available")
 createdAt DateTime @default(now()) @map("created_at")
}

```

```

property Property @relation(fields: [propertyId], references: [id])
bookings Booking[]

@@unique([propertyId, roomNumber])
@@map("rooms")
}

model Booking {
 id String @id @default(uuid())
 userId String @map("user_id")
 propertyId String @map("property_id")
 roomId String? @map("room_id")
 bookingReference String @unique @map("booking_reference")
 checkInDate DateTime @map("check_in_date")
 checkOutDate DateTime @map("check_out_date")
 guestsCount Int @map("guests_count")
 guestDetails Json? @map("guest_details")
 totalAmount Decimal? @map("total_amount")
 bookingStatus String @default("confirmed") @map("booking_status")
 specialRequests String? @map("special_requests")
 createdAt DateTime @default(now()) @map("created_at")

 user User @relation(fields: [userId], references: [id])
 property Property @relation(fields: [propertyId], references: [id])
 room Room? @relation(fields: [roomId], references: [id])
 checkIns CheckIn[]
 checkOuts CheckOut[]
 reviews Review[]

 @@map("bookings")
}

model CheckIn {
 id String @id @default(uuid())
 bookingId String @map("booking_id")
 userId String @map("user_id")
 propertyId String @map("property_id")
 checkInTimestamp DateTime @default(now()) @map("check_in_timestamp")
 identityDocumentUrl String? @map("identity_document_url")
 selfieUrl String? @map("selfie_url")
 digitalSignature String? @map("digital_signature")
 deviceInfo Json? @map("device_info")
 locationCoordinates String? @map("location_coordinates")
 verificationStatus String @default("pending") @map("verification_status")
 verifiedBy String? @map("verified_by")
 notes String?
}

```

```

booking Booking @relation(fields: [bookingId], references: [id])
user User @relation(fields: [userId], references: [id])
property Property @relation(fields: [propertyId], references: [id])
verifier PropertyOwner? @relation(fields: [verifiedBy], references: [id])
checkOuts CheckOut[]

@@map("check_ins")
}

model CheckOut {
 id String @id @default(uuid())
 bookingId String @map("booking_id")
 checkInId String @map("check_in_id")
 checkOutTimestamp DateTime @default(now()) @map("check_out_timestamp")
 roomConditionPhotos Json? @map("room_condition_photos")
 damagesReported String? @map("damages_reported")
 additionalCharges Decimal @default(0) @map("additional_charges")
 guestSignature String? @map("guest_signature")
 staffNotes String? @map("staff_notes")
 processedBy String? @map("processed_by")

 booking Booking @relation(fields: [bookingId], references: [id])
 checkIn CheckIn @relation(fields: [checkInId], references: [id])
 processor PropertyOwner? @relation(fields: [processedBy], references: [id])

 @@map("check_outs")
}

model Review {
 id String @id @default(uuid())
 bookingId String @map("booking_id")
 userId String @map("user_id")
 propertyId String @map("property_id")
 overallRating Int @map("overall_rating")
 cleanlinessRating Int @map("cleanliness_rating")
 locationRating Int @map("location_rating")
 valueRating Int @map("value_rating")
 serviceRating Int @map("service_rating")
 comment String?
 photos Json?
 isAnonymous Boolean @default(false) @map("is_anonymous")
 responseFromOwner String? @map("response_from_owner")
 responseDate DateTime? @map("response_date")
 createdAt DateTime @default(now()) @map("created_at")

 booking Booking @relation(fields: [bookingId], references: [id])
 user User @relation(fields: [userId], references: [id])
}

```

```

property Property @relation(fields: [propertyId], references: [id])

 @@map("reviews")
}

model Notification {
 id String @id @default(uuid())
 userId String? @map("user_id")
 propertyOwnerId String? @map("property_owner_id")
 notificationType String @map("notification_type")
 title String
 message String
 data Json?
 isRead Boolean @default(false) @map("is_read")
 sentAt DateTime @default(now()) @map("sent_at")

 user User? @relation(fields: [userId], references: [id])
 propertyOwner PropertyOwner? @relation(fields: [propertyOwnerId], references: [id])

 @@map("notifications")
}

model DeviceToken {
 id String @id @default(uuid())
 userId String? @map("user_id")
 propertyOwnerId String? @map("property_owner_id")
 token String @unique
 platform String
 isActive Boolean @default(true) @map("is_active")
 createdAt DateTime @default(now()) @map("created_at")

 user User? @relation(fields: [userId], references: [id])
 propertyOwner PropertyOwner? @relation(fields: [propertyOwnerId], references: [id])

 @@map("device_tokens")
}

```

## Controlador de Check-in

typescript

```
// src/controllers/checkinController.ts

import { Request, Response } from 'express';
import { checkinService } from './services/checkinService';
import { validateQRCode } from './validators/checkinValidator';
import { uploadService } from './services/uploadService';

export class CheckinController {
 async scanQR(req: Request, res: Response) {
 try {
 const { qrData } = req.body;
 const validation = await validateQRCode(qrData);

 if (!validation.isValid) {
 return res.status(400).json({
 success: false,
 message: validation.error
 });
 }

 const booking = await checkinService.getBookingByQR(qrData);

 if (!booking) {
 return res.status(404).json({
 success: false,
 message: 'Reserva no encontrada o código QR inválido'
 });
 }

 // Verificar que el usuario actual puede hacer check-in
 if (booking.userId !== req.user.id) {
 return res.status(403).json({
 success: false,
 message: 'No autorizado para esta reserva'
 });
 }

 // Verificar fechas
 const today = new Date();
 const checkInDate = new Date(booking.checkInDate);

 if (today < checkInDate) {
 return res.status(400).json({
 success: false,
 message: 'Aún no es la fecha de check-in'
 });
 }
 }
 }
}
```

```

res.json({
 success: true,
 data: {
 booking: {
 id: booking.id,
 reference: booking.bookingReference,
 property: booking.property,
 room: booking.room,
 checkInDate: booking.checkInDate,
 checkOutDate: booking.checkOutDate,
 guestsCount: booking.guestsCount
 }
 }
});

} catch (error) {
 console.error('Error en scanQR:', error);
 res.status(500).json({
 success: false,
 message: 'Error interno del servidor'
 });
}

}

async uploadIdentityDocument(req: Request, res: Response) {
 try {
 const { bookingId } = req.params;
 const files = req.files as { [fieldname: string]: Express.Multer.File[] };

 if (!files.document || !files.selfie) {
 return res.status(400).json({
 success: false,
 message: 'Se requieren tanto el documento como la selfie'
 });
 }

 const documentFile = files.document[0];
 const selfieFile = files.selfie[0];

 // Validar archivos
 const validation = await checkinService.validateIdentityFiles(
 documentFile,
 selfieFile
);

 if (!validation.isValid) {

```

```
return res.status(400).json({
 success: false,
 message: validation.error
});
}

// Subir archivos a S3
const documentUrl = await uploadService.uploadFile(
 documentFile,
 'documents',
 req.user.id
);

const selfieUrl = await uploadService.uploadFile(
 selfieFile,
 'selfies',
 req.user.id
);

// Procesar verificación de identidad (OCR + face matching)
const verificationResult = await checkinService.processIdentityVerification(
 documentUrl,
 selfieUrl,
 req.user.id
);

// Crear o actualizar check-in
const checkIn = await checkinService.updateCheckInWithIdentity(
 bookingId,
 {
 identityDocumentUrl: documentUrl,
 selfieUrl: selfieUrl,
 verificationStatus: verificationResult.status,
 deviceInfo: {
 userAgent: req.headers['user-agent'],
 ip: req.ip,
 timestamp: new Date()
 }
 }
);

res.json({
 success: true,
 data: {
 checkInId: checkIn.id,
 verificationStatus: verificationResult.status,
 nextStep: verificationResult.status === 'verified' ? 'terms' : 'manual_review'
 }
});
```

```
 }

 });

} catch (error) {
 console.error('Error en uploadIdentityDocument:', error);
 res.status(500).json({
 success: false,
 message: 'Error procesando documentos'
 });
}

}

async completeCheckIn(req: Request, res: Response) {
 try {
 const { bookingId } = req.params;
 const { digitalSignature, locationCoordinates } = req.body;

 const checkIn = await checkinService.completeCheckIn(bookingId, {
 digitalSignature,
 locationCoordinates,
 userId: req.user.id
 });

 // Actualizar estado de la reserva
 await checkinService.updateBookingStatus(bookingId, 'checked_in');

 // Programar notificaciones futuras
 await checkinService.scheduleCheckOutReminder(bookingId);

 // Enviar notificación al propietario
 await checkinService.notifyPropertyOwner(checkIn.propertyId, {
 type: 'check_in_completed',
 guestName: `${req.user.firstName} ${req.user.lastName}`,
 property: checkIn.property.name
 });

 res.json({
 success: true,
 data: {
 checkIn: {
 id: checkIn.id,
 timestamp: checkIn.checkInTimestamp,
 property: checkIn.property,
 accessInfo: {
 wifi: checkIn.property.wifiCredentials,
 instructions: checkIn.property.accessInstructions
 }
 }
 }
 });
 } catch (error) {
 console.error('Error en completeCheckIn:', error);
 res.status(500).json({
 success: false,
 message: 'Error procesando documentos'
 });
 }
}
```

```
 }
 }
});

} catch (error) {
 console.error('Error en completeCheckIn:', error);
 res.status(500).json({
 success: false,
 message: 'Error completando check-in'
 });
}

}

export const checkinController = new CheckinController();
```

## Servicio de Check-in

typescript

```
// src/services/checkinService.ts

import { PrismaClient } from '@prisma/client';
import { uploadService } from './uploadService';
import { ocrService } from './ocrService';
import { faceMatchingService } from './faceMatchingService';
import { notificationService } from './notificationService';
import { queueService } from './queueService';

const prisma = new PrismaClient();

export class CheckinService {
 async getBookingByQR(qrData: string) {
 return await prisma.booking.findFirst({
 where: {
 OR: [
 { property: { qrCodeData: qrData } },
 { room: { qrCodeData: qrData } }
],
 bookingStatus: 'confirmed'
 },
 include: {
 user: true,
 property: true,
 room: true
 }
 });
 }

 async validateIdentityFiles(documentFile: Express.Multer.File, selfieFile: Express.Multer.File) {
 const maxSize = 10 * 1024 * 1024; // 10MB
 const allowedTypes = ['image/jpeg', 'image/png', 'image/jpg'];

 if (documentFile.size > maxSize || selfieFile.size > maxSize) {
 return { isValid: false, error: 'Los archivos no pueden superar 10MB' };
 }

 if (!allowedTypes.includes(documentFile.mimetype) || !allowedTypes.includes(selfieFile.mimetype)) {
 return { isValid: false, error: 'Solo se permiten archivos JPG, JPEG y PNG' };
 }

 return { isValid: true };
 }

 async processIdentityVerification(documentUrl: string, selfieUrl: string, userId: string) {
 try {
 // Procesar OCR del documento

```

```
const ocrResult = await ocrService.extractDocumentData(documentUrl);

if (!ocrResult.success) {
 return {
 status: 'failed',
 reason: 'No se pudo extraer información del documento'
 };
}

// Comparar rostros
const faceMatchResult = await faceMatchingService.compareImages(
 documentUrl,
 selfieUrl
);

if (faceMatchResult.confidence < 0.8) {
 return {
 status: 'manual_review',
 reason: 'Baja confianza en la comparación facial'
 };
}

// Validar datos extraídos
const user = await prisma.user.findUnique({ where: { id: userId } });
const dataMatch = this.validateExtractedData(ocrResult.data, user);

if (!dataMatch.isValid) {
 return {
 status: 'manual_review',
 reason: dataMatch.reason
 };
}

return {
 status: 'verified',
 extractedData: ocrResult.data,
 confidence: faceMatchResult.confidence
};

} catch (error) {
 console.error('Error en verificación de identidad:', error);
 return {
 status: 'failed',
 reason: 'Error técnico en la verificación'
 };
}
```

```

private validateExtractedData(extractedData: any, user: any) {
 // Comparar nombres (tolerancia a diferencias menores)
 const nameMatch = this.fuzzyMatch(
 `${extractedData.firstName} ${extractedData.lastName}`,
 `${user.firstName} ${user.lastName}`
);

 if (nameMatch < 0.8) {
 return {
 isValid: false,
 reason: 'Los nombres no coinciden suficientemente'
 };
 }

 // Validar fecha de nacimiento si está disponible
 if (extractedData.dateOfBirth && user.dateOfBirth) {
 const docDate = new Date(extractedData.dateOfBirth);
 const userDate = new Date(user.dateOfBirth);

 if (Math.abs(docDate.getTime() - userDate.getTime()) > 24 * 60 * 60 * 1000) {
 return {
 isValid: false,
 reason: 'La fecha de nacimiento no coincide'
 };
 }
 }

 return { isValid: true };
}

private fuzzyMatch(str1: string, str2: string): number {
 // Implementación simple de distancia de Levenshtein normalizada
 const longer = str1.length > str2.length ? str1 : str2;
 const shorter = str1.length > str2.length ? str2 : str1;

 if (longer.length === 0) return 1.0;

 const distance = this.levenshteinDistance(longer, shorter);
 return (longer.length - distance) / longer.length;
}

private levenshteinDistance(str1: string, str2: string): number {
 const matrix = [];

 for (let i = 0; i <= str2.length; i++) {
 matrix[i] = [i];
 }

 for (let j = 1; j < str1.length; j++) {
 matrix[0][j] = j;
 }

 for (let i = 1; i < str2.length; i++) {
 for (let j = 1; j < str1.length; j++) {
 const cost = str1[j] === str2[i] ? 0 : 1;
 matrix[i][j] = Math.min(
 matrix[i - 1][j] + 1, // Insertion
 matrix[i][j - 1] + 1, // Deletion
 matrix[i - 1][j - 1] + cost // Substitution
);
 }
 }

 return matrix[str2.length][str1.length];
}

```

```
}

for (let j = 0; j <= str1.length; j++) {
 matrix[0][j] = j;
}

for (let i = 1; i <= str2.length; i++) {
 for (let j = 1; j <= str1.length; j++) {
 if (str2.charAt(i - 1) === str1.charAt(j - 1)) {
 matrix[i][j] = matrix[i - 1][j - 1];
 } else {
 matrix[i][j] = Math.min(
 matrix[i - 1][j - 1] + 1,
 matrix[i][j - 1] + 1,
 matrix[i - 1][j] + 1
);
 }
 }
}

return matrix[str2.length][str1.length];
}

async updateCheckInWithIdentity(bookingsId: string, data: any) {
 return await prisma.checkIn.upsert({
 where: {
 bookingsId: bookingsId
 },
 update: data,
 create: {
 bookingsId,
 userId: data.userId,
 propertyId: data.propertyId,
 ...data
 },
 include: {
 property: true,
 booking: true
 }
 });
}

async completeCheckIn(bookingsId: string, data: any) {
 const checkIn = await prisma.checkIn.update({
 where: { bookingsId },
 data: {
 digitalSignature: data.digitalSignature,
 status: "COMPLETED"
 }
 });
}
```

```
 locationCoordinates: data.locationCoordinates,
 verificationStatus: 'completed'
 },
 include: {
 property: true,
 booking: true
 }
});

return checkIn;
}

async updateBookingStatus(bookingId: string, status: string) {
 return await prisma.booking.update({
 where: { id: bookingId },
 data: { bookingStatus: status }
 });
}

async scheduleCheckOutReminder(bookingId: string) {
 const booking = await prisma.booking.findUnique({
 where: { id: bookingId }
 });

 if (booking) {
 const reminderTime = new Date(booking.checkOutDate);
 reminderTime.setHours(9, 0, 0, 0); // 9 AM del día de check-out

 await queueService.scheduleNotification({
 userId: booking.userId,
 type: 'check_out_reminder',
 scheduledFor: reminderTime,
 data: { bookingId }
 });
 }
}

async notifyPropertyOwner(propertyId: string, notificationData: any) {
 const property = await prisma.property.findUnique({
 where: { id: propertyId },
 include: { owner: true }
 });

 if (property) {
 await notificationService.sendToPropertyOwner(
 property.owner.id,
 notificationData
);
 }
}
```

```
);
}
}

export const checkinService = new CheckinService();
```

## Panel Web Administrativo (React.js)

### Estructura del Proyecto Frontend Web

```
admin-panel/
├── src/
│ ├── components/ # Componentes reutilizables
│ │ ├── ui/ # Componentes básicos
│ │ ├── charts/ # Gráficos y visualizaciones
│ │ ├── tables/ # Tablas de datos
│ │ └── layout/ # Layout y navegación
│ ├── pages/ # Páginas principales
│ │ ├── Dashboard/ # Dashboard principal
│ │ ├── Bookings/ # Gestión de reservas
│ │ ├── Properties/ # Gestión de propiedades
│ │ ├── Reviews/ # Gestión de reseñas
│ │ └── Analytics/ # Reportes y análisis
│ ├── hooks/ # Custom hooks
│ ├── services/ # Servicios API
│ ├── store/ # Redux store
│ ├── utils/ # Utilidades
│ ├── types/ # TypeScript types
│ └── styles/ # Estilos globales
└── public/
└── package.json
```

### Dashboard Component

typescript

```
// src/pages/Dashboard/Dashboard.tsx
import React, { useEffect, useState } from 'react';
import { Grid, Card,CardContent, Typography, Box } from '@mui/material';
import {
 CheckCircle,
 ExitToApp,
 Hotel,
 Star,
 TrendingUp,
 Warning
} from '@mui/icons-material';
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer } from 'recharts';
import { dashboardService } from '../services/dashboardService';
import { MetricCard } from '../components/ui/MetricCard';
import { AlertsList } from '../components/ui/AlertsList';
import { RecentActivity } from '../components/ui/RecentActivity';

interface DashboardData {
 todayMetrics: {
 checkIns: number;
 checkOuts: number;
 occupancy: number;
 averageRating: number;
 };
 checkInTrend: Array<{ time: string; count: number }>;
 alerts: Array<{
 id: string;
 type: 'warning' | 'error' | 'info';
 message: string;
 timestamp: Date;
 }>;
 recentCheckIns: Array<{
 id: string;
 guestName: string;
 property: string;
 time: string;
 status: 'completed' | 'pending' | 'verifying';
 }>;
}

export const Dashboard: React.FC = () => {
 const [data, setData] = useState<DashboardData | null>(null);
 const [loading, setLoading] = useState(true);

 useEffect(() => {
 loadDashboardData();
 }, []);

 if (loading) {
 return <div>Loading</div>;
 }

 return (
 <div>
 <Card>
 <CardContent>
 <LineChart>
 <Line data={data?.checkInTrend} />
 <XAxis />
 <YAxis />
 <CartesianGrid />
 <Tooltip />
 </LineChart>
 <MetricCard data={data?.todayMetrics} />
 <AlertsList data={data?.alerts} />
 <RecentActivity data={data?.recentCheckIns} />
 </CardContent>
 </Card>
 </div>
);
}

const loadDashboardData = async () => {
 try {
 const response = await fetch('/api/dashboard');
 const data = await response.json();
 setData(data);
 } catch (error) {
 console.error(error);
 }
}
```

```
}, []);
```

```
const loadDashboardData = async () => {
 try {
 const dashboardData = await dashboardService.getDashboardData();
 setData(dashboardData);
 } catch (error) {
 console.error('Error loading dashboard data:', error);
 } finally {
 setLoading(false);
 }
};
```

```
if (loading) {
 return <div>Cargando...</div>;
}

if (!data) {
 return <div>Error cargando datos</div>;
}
```

```
return (
 <Box sx={{ p: 3 }}>
 <Typography variant="h4" sx={{ mb: 3 }}>
 Dashboard - {new Date().toLocaleDateString('es-ES', {
 weekday: 'long',
 year: 'numeric',
 month: 'long',
 day: 'numeric'
 })}
 </Typography>
)
```

```
/* Métricas principales */
<Grid container spacing={3} sx={{ mb: 4 }}>
 <Grid item xs={12} sm={6} md={3}>
 <MetricCard
 title="Check-ins Hoy"
 value={data.todayMetrics.checkIns}
 icon={<CheckCircle />}
 color="success"
 trend={+3}
 />
 </Grid>
 <Grid item xs={12} sm={6} md={3}>
 <MetricCard
 title="Check-outs Hoy"
 value={data.todayMetrics.checkOuts}
```

```
icon={<ExitToApp />}
color="info"
trend={+1}
/>
</Grid>
<Grid item xs={12} sm={6} md={
```