Especificaciones Técnicas de Implementación - TurisGal

1. STACK TECNOLÓGICO DETALLADO

Frontend Móvil (React Native)

Dependencias Principales

json

```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

```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/               # Esqu
```