Guía de implementación: Sistema para empresa de autobuses nacional

1. Análisis y planificación

1.1 Definición de requisitos

- **Funcionalidades clave:**
  - Gestión de rutas y horarios
  - Reserva y venta de boletos
  - Administración de flota de autobuses
  - Gestión de personal (conductores)
  - Reportes y estadísticas
  - Panel de administración

1.2 Arquitectura del sistema

- **Backend:** Java Spring Boot (API RESTful)
- **Frontend web:** React.js
- **Aplicación móvil:** React Native
- **Base de datos:** PostgreSQL con Hibernate ORM
- **Autenticación:** JWT (JSON Web Tokens)
- **Control de versiones:** Git (GitHub/GitLab)

2. Configuración del entorno de desarrollo

2.1 Herramientas necesarias

- JDK 17+
- Node.js y npm/yarn
- PostgreSQL
- IDE (IntelliJ IDEA, VS Code)
- Git
- Postman (para pruebas de API)

2.2 Estructura del proyecto

Copy

```
proyecto-autobuses/
├── backend/              # Spring Boot API
├── frontend-web/         # Aplicación React
├── mobile-app/           # Aplicación React Native
└── docs/                 # Documentación
```

3. Desarrollo del backend (Spring Boot)

3.1 Configuración inicial

1. Crear proyecto Spring Boot con Spring Initializr
   - Dependencias: Web, JPA, PostgreSQL, Lombok, Security, Validation
2. Configurar `application.properties`:

properties Copy

```properties
# Conexión a PostgreSQL
spring.datasource.url=jdbc:postgresql://localhost:5432/autobuses_db
spring.datasource.username=postgres
spring.datasource.password=password

# Configuración Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

# JWT
jwt.secret=clave_secreta_jwt
jwt.expiration=86400000
```

3.2 Modelo de datos (entidades)

Crear las siguientes entidades:

java Copy

```java
// Usuario.java
@Entity
@Table(name = "usuarios")
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nombre;
    private String email;
    private String password;

    @Enumerated(EnumType.STRING)
    private TipoUsuario tipo; // ADMIN, CLIENTE, CONDUCTOR

    // Getters, setters, constructores
}

// Autobus.java
@Entity
@Table(name = "autobuses")
public class Autobus {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String placa;
    private String modelo;
    private int capacidad;
    private boolean activo;

    // Getters, setters, constructores
}

// Ruta.java, Horario.java, Boleto.java, etc.
```

3.3 Repositorios JPA

java Copy

```java
// UsuarioRepository.java
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {
    Optional<Usuario> findByEmail(String email);
}

// Otros repositorios similares
```

3.4 Servicios

java Copy

```java
// UsuarioService.java
@Service
public class UsuarioService {
    @Autowired
    private UsuarioRepository usuarioRepository;

    @Autowired
    private PasswordEncoder passwordEncoder;

    public Usuario registrarUsuario(Usuario usuario) {
        usuario.setPassword(passwordEncoder.encode(usuario.getPassword()));
        return usuarioRepository.save(usuario);
    }

    // Otros métodos
}
```

3.5 Controladores REST

java Copy

```java
// UsuarioController.java
@RestController
@RequestMapping("/api/usuarios")
public class UsuarioController {
    @Autowired
    private UsuarioService usuarioService;

    @PostMapping("/registro")
    public ResponseEntity<Usuario> registrarUsuario(@Valid @RequestBody Usuario usuari
        return ResponseEntity.ok(usuarioService.registrarUsuario(usuario));
    }

    // Otros endpoints
}
```

3.6 Seguridad y autenticación

1. Configurar Spring Security con JWT
2. Implementar filtros de autenticación
3. Definir roles y permisos

4. Desarrollo del frontend web (React)

4.1 Configuración inicial

1. Crear proyecto React:

bash Copy

```bash
npx create-react-app frontend-web
cd frontend-web
npm install axios react-router-dom formik yup styled-components
```

4.2 Estructura de carpetas

Copy

```
frontend-web/
├── public/
├── src/
    ├── components/     # Componentes reutilizables
    ├── pages/          # Páginas principales
    ├── services/       # Servicios API
    ├── hooks/          # Custom hooks
    ├── context/        # Context API
    ├── utils/          # Utilidades
    ├── assets/         # Imágenes, iconos, etc.
    ├── App.js
    └── index.js
```

4.3 Implementar rutas principales

jsx Copy

```jsx
// App.js
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Login from './pages/Login';
import Dashboard from './pages/Dashboard';
import BusquedaRutas from './pages/BusquedaRutas';
import ReservaPage from './pages/ReservaPage';
import ProtectedRoute from './components/ProtectedRoute';

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/login" element={<Login />} />
        <Route path="/registro" element={<Registro />} />
        <Route path="/buscar" element={<BusquedaRutas />} />
        <Route path="/reservar/:rutaId" element={<ReservaPage />} />
        <Route
          path="/dashboard/*"
          element={
            <ProtectedRoute>
              <Dashboard />
            </ProtectedRoute>
          }
        />
        <Route path="/" element={<Home />} />
      </Routes>
    </BrowserRouter>
  );
}
```

4.4 Servicios API

jsx Copy

```jsx
// api.js
import axios from 'axios';

const API_URL = 'http://localhost:8080/api';

const api = axios.create({
  baseURL: API_URL,
});

// Interceptor para añadir token
api.interceptors.request.use((config) => {
  const token = localStorage.getItem('token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});

export const loginUser = (credentials) => {
  return api.post('/auth/login', credentials);
};

export const getRutas = () => {
  return api.get('/rutas');
};

// Otros servicios
```

5. Desarrollo de aplicación móvil (React Native)

5.1 Configuración inicial

1. Crear proyecto React Native:

bash Copy

```bash
npx react-native init MobileApp
cd MobileApp
npm install @react-navigation/native axios react-native-elements
```

5.2 Estructura similar al frontend web

Copy

```
mobile-app/
├── android/
├── ios/
├── src/
    ├── components/
    ├── screens/
    ├── services/
    ├── hooks/
    ├── context/
    ├── utils/
    ├── assets/
    └── App.js
```

5.3 Navegación

jsx Copy

```jsx
// App.js
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import LoginScreen from './screens/LoginScreen';
import HomeScreen from './screens/HomeScreen';
import BusquedaScreen from './screens/BusquedaScreen';
import ReservaScreen from './screens/ReservaScreen';

const Stack = createStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Login">
        <Stack.Screen name="Login" component={LoginScreen} />
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Busqueda" component={BusquedaScreen} />
        <Stack.Screen name="Reserva" component={ReservaScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default App;
```

6. Base de datos (PostgreSQL con Hibernate)

6.1 Modelo ER

Diseñar el modelo Entidad-Relación con las siguientes tablas principales:

- usuarios
- autobuses
- rutas
- horarios
- boletos
- reservas

6.2 Scripts SQL iniciales

sql Copy

```sql
-- Crear base de datos
CREATE DATABASE autobuses_db;

-- Insertar datos iniciales (ciudades, tipos de usuario, etc.)
INSERT INTO ciudades (nombre) VALUES ('Ciudad de México'), ('Guadalajara'), ('Monterre
```

7. Implementación de CI/CD

7.1 Configurar GitHub Actions o GitLab CI

1. Automatizar pruebas
2. Automatizar despliegue

8. Pruebas

8.1 Pruebas unitarias

java Copy

```java
// Backend
@SpringBootTest
class UsuarioServiceTest {
    @Autowired
    private UsuarioService usuarioService;

    @Test
    void debeRegistrarUsuario() {
        // Implementación
    }
}
```

8.2 Pruebas de integración

- Probar flujos completos (registro, reserva de boletos, etc.)

9. Despliegue

9.1 Opciones de despliegue

- **Backend:** AWS, Azure, Heroku
- **Frontend:** Netlify, Vercel
- **Mobile:** Google Play Store, Apple App Store
- **Base de datos:** AWS RDS, Azure Database

10. Monitoreo y mantenimiento

10.1 Herramientas

- Spring Actuator
- Prometheus/Grafana
- Sentry para errores

Siguiente iteración: Funcionalidades avanzadas

- Notificaciones push
- Pagos en línea
- Mapas en tiempo real
- Programa de lealtad