



*Universidad Tecnológica de Tijuana*

**Materia:**

Desarrollo movil Integral

**Grupo:**

10-B

**Nombre del Alumno:**

Rendon Gurrola Alan Misael

**Docente:**

Ray Brunette Parra Galaviz

**Fecha de Elaboración:**

10/01/2024

**Architecture specification**

## **Especificación de la Arquitectura**

La arquitectura de un sistema define cómo se organizan y conectan los diferentes componentes de la aplicación. En este caso, he optado por una arquitectura basada en tres capas (Three-Tier Architecture), que es adecuada para aplicaciones modernas y permite una separación clara entre frontend, backend y datos. Esta arquitectura es escalable, modular y facilita el mantenimiento del sistema.

### **Arquitectura General: Tres Capas**

#### **Capa de Presentación (Frontend):**

Compuesta por React y React Native, esta capa se encarga de la interacción con el usuario.

#### **Responsabilidades:**

- Renderizar interfaces de usuario dinámicas y responsivas.
- Manejar la lógica del cliente, como la validación de formularios y la navegación.
- Realizar llamadas a las APIs proporcionadas por el backend.

#### **Capa de Aplicación (Backend):**

Implementada con Django, esta capa procesa la lógica de negocio y expone APIs RESTful mediante Django REST Framework (DRF).

#### **Responsabilidades:**

- Manejar solicitudes del cliente y proporcionar respuestas adecuadas.
- Implementar reglas de negocio y validaciones más complejas.
- Orquestar la interacción con la base de datos.

#### **Capa de Datos:**

Gestionada por el ORM (Object-Relational Mapping) de Django, esta capa interactúa con la base de datos subyacente (como PostgreSQL o MySQL).

## **Responsabilidades:**

- Almacenar, recuperar y gestionar datos de forma eficiente.
- Mantener la integridad y consistencia de los datos.

## **Componentes Clave de la Arquitectura**

### **Frontend (React y React Native):**

#### **Patrón de Componentes:**

Divide la interfaz en componentes reutilizables, lo que mejora la modularidad y el mantenimiento.

#### **Estado Centralizado (Redux o Context API):**

Permite manejar estados globales, especialmente para datos compartidos entre múltiples componentes.

#### **Comunicación con el Backend:**

Utiliza fetch o bibliotecas como Axios para realizar solicitudes HTTP hacia las APIs del backend.

### **Backend (Django):**

#### **Django REST Framework (DRF):**

Proporciona una capa de API que actúa como fachada entre el cliente y los datos.

#### **Autenticación y Autorización:**

Usa las herramientas integradas de Django para gestionar sesiones, tokens y permisos de usuario.

#### **Servicios Backend:**

Organización del código en aplicaciones modulares dentro de Django para separar funcionalidades específicas.

## **Base de Datos:**

### **ORM de Django:**

Simplifica la interacción con la base de datos al usar modelos definidos en Python.

### **Modelo Relacional:**

Base de datos relacional como PostgreSQL, ideal para aplicaciones que requieren estructuras de datos bien definidas.

## **Flujo de Trabajo del Sistema**

### **Solicitud desde el Cliente (React/React Native):**

El usuario interactúa con la interfaz de usuario, que envía una solicitud HTTP al backend (por ejemplo, al realizar un inicio de sesión).

### **Procesamiento en el Backend (Django):**

Django recibe la solicitud, la valida y ejecuta la lógica necesaria, como consultar datos desde la base de datos.

### **Respuesta del Backend:**

Django envía una respuesta en formato JSON al cliente con los datos requeridos o un mensaje de error.

### **Renderizado en el Cliente:**

React o React Native procesa la respuesta y actualiza la interfaz de usuario de forma dinámica.

## **Beneficios de esta Arquitectura**

### **Separación de responsabilidades:**

Cada capa tiene un propósito bien definido, lo que facilita el desarrollo, pruebas y mantenimiento.

### **Escalabilidad:**

Permite escalar cada capa de forma independiente según las necesidades del proyecto.

### **Reutilización de código:**

Al usar React y React Native, se puede compartir lógica y componentes entre la versión web y móvil.

### **Flexibilidad:**

La arquitectura basada en APIs facilita la integración con otros servicios o aplicaciones.

### **Seguridad:**

Django ofrece herramientas integradas para proteger contra vulnerabilidades comunes, mientras que la separación cliente-servidor minimiza la exposición de datos sensibles.

## **Conclusión**

La arquitectura basada en tres capas (frontend, backend y datos), implementada con React, React Native y Django, proporciona una solución robusta, flexible y escalable. Esta estructura permite mantener una clara separación entre las responsabilidades, garantizando una experiencia de usuario fluida y un desarrollo eficiente.