**1. Requisitos Funcionales**

Estos son los elementos esenciales que la aplicación debe ofrecer:

**Inicio de sesión**

* Permitir a los usuarios ingresar con:
  + Nombre de usuario y contraseña.
* Validar las credenciales contra la base de datos.
* Mostrar mensajes de error claros si los datos no son correctos.

**Registro de usuarios**

* Recopilar información personal (nombre, apellido, DNI, fecha de nacimiento, email, contraseña).
* Validar datos en el frontend:
  + Contraseñas seguras (mínimo de caracteres, uso de mayúsculas, minúsculas y números).
  + Validar que el email tenga un formato válido.
* Enviar los datos al backend para almacenarlos en la base de datos.

**Pantalla de bienvenida**

* Mostrar el nombre del usuario después de iniciar sesión.
* Botón o enlace para continuar a otras vistas.

**Perfil de usuario**

* Mostrar información del usuario registrada en el sistema (nombre, DNI, edad, fecha de nacimiento).
* Posibilidad de editar la información personal.

**Votación**

* Presentar opciones de candidatos y cargos.
* Validar que el usuario seleccione ambas opciones antes de enviar.
* Almacenar el voto asociado al usuario en la base de datos.
* Bloquear la votación duplicada.

**2. Requisitos No Funcionales**

Estos determinan la calidad de la aplicación.

**Usabilidad**

* Diseño limpio y moderno.
* Navegación intuitiva entre las pantallas.
* Compatible con pantallas de diferentes tamaños.

**Rendimiento**

* Respuesta rápida en todas las interacciones del usuario.
* Backend capaz de manejar varios usuarios simultáneamente.

**Seguridad**

* Contraseñas cifradas antes de almacenarlas en la base de datos.
* Proteger las transacciones con HTTPS.
* Validar datos en el backend para evitar inyecciones de código.

**Compatibilidad**

* Soporte para dispositivos Android e iOS.
* Posibilidad de expandir a una versión web en el futuro.

**3. Requisitos Técnicos**

Estas son las tecnologías y herramientas que podrían ser utilizadas para desarrollar la aplicación.

**Frontend**

* **Frameworks sugeridos**:
  + **Flutter**: Ideal para crear aplicaciones nativas en Android e iOS desde un único código base.
  + **Java**
* **Librerías recomendadas**:
  + Form validation (ejemplo: react-hook-form para React Native).
  + Manejo de estados (ejemplo: Provider en Flutter o Redux en React Native).

**Backend**

* **Lenguaje y framework sugerido**:
  + **Node.js + Express**: Para manejar las solicitudes del usuario y conectarse con la base de datos.
  + **Firebase**: Una opción todo-en-uno si buscas rapidez en el desarrollo.
* **Base de datos**:
  + Relacional: MySQL o PostgreSQL.
  + NoSQL: Firebase Firestore o MongoDB.

**Integración**

* **API REST o GraphQL**:
  + Exponer los servicios para autenticación, registro, votación y edición del perfil.
* **Servicios en la nube**:
  + **Firebase** para autenticación y base de datos en tiempo real.
  + **AWS** o **Google Cloud** para el hosting del backend.

**Seguridad**

* Hash de contraseñas con **bcrypt** o similar.
* Uso de **JWT** (JSON Web Tokens) para mantener la sesión del usuario activa.

**4. Casos de Uso**

**Caso de uso 1: Registro**

* El usuario ingresa los datos requeridos y los valida.
* Si los datos son correctos, se almacena la información en la base de datos.
* El usuario recibe una notificación de registro exitoso.

**Caso de uso 2: Inicio de sesión**

* El usuario ingresa sus credenciales.
* Si son válidas, es redirigido a la pantalla de bienvenida.
* Si no, se muestra un mensaje de error.

**Caso de uso 3: Votación**

* El usuario selecciona un candidato y un cargo.
* El sistema verifica si el usuario ya ha votado.
* Si no ha votado, el voto es registrado.