Challenge Técnico – Frontend (React)

Contexto:
Vas a construir una **web de un reclutador** que consulta una base de desarrolladores, permite **filtrar candidatos** y **enviarles un mensaje** desde la plataforma. El foco principal de la evaluación está en **buenas prácticas** de desarrollo front-end.

> **Referencia visual:** te dejamos un Figma orientativo (link a figma). No es obligatorio replicarlo 1:1; la estética y micro-interacciones quedan a tu criterio siempre que priorices usabilidad y consistencia.

---

1) Objetivos

1. Implementar un **Login** simple (no hace falta backend real).
2. Crear una **vista de listado** que consuma el endpoint de usuarios:
   - `GET https://private-73f5b0-challengefront.apiary-mock.com/userlist`
   - Permitir **filtrar** candidatos (el/los criterios los definís el candidato: por nombre, stack, seniority, etc.).
3. Implementar un **formulario de contacto** para **enviar un mensaje** a un candidato:
   - `POST https://private-73f5b0-challengefront.apiary-mock.com/messages`
   - Debe pedir **email** para enviar una copia de la solicitud de contacto.
   - Manejar correctamente **respuestas 201** y **422** (ver detalles más abajo).
4. Implementar lista de roles permitidos.
   - `GET https://private-73f5b0-challengefront.apiary-mock.com/roleslist`

---

2) API: detalles y contratos

2.1 Envío de mensajes (POST)

```
POST /messages
Content-Type: application/json
Body:
{
  "role": "Frontend",
  "msj": "Mensaje de prueba"
}

```

Respuestas posibles:

- **201 Created**

```
{
  "id": 42,
  "role": "Frontend",
  "msj": "Estoy interesado en participar del challenge de UI, ¿hay linea
  "submitted_at": "2025-10-09T17:00:00.000Z",
  "status": "received"
}

```

- **422 Unprocessable Entity**

```
{
  "error": "invalid_role",
  "message": "El rol debe ser uno de: Frontend, Backend, Fullstack, DBA"
}

```

Requisitos de UI/UX ante estas respuestas

- **201:** mostrar confirmación clara (toast/modal), limpiar el formulario y registrar el envío en un historial local.
- **422:** mostrar error legible, resaltar el/los campos a corregir (p. ej. `role`) y sugerir valores válidos. No perder lo ingresado por el usuario.

---

3) Funcionalidades mínimas

1. **Login**
   - Puede ser simulado (p. ej. usuario: `recruiter@demo.com`, pass: `123456`).
   - Al loguear, persistir sesión (localStorage/SessionStorage) y proteger rutas privadas.
2. **Listado de Candidatos**
   - Consumir `GET /userlist`.
   - **Filtrado**: al menos un criterio (texto libre, por stack, seniority, ubicación, etc.).
   - Ordenamiento y/o paginación.
3. **Enviar Mensaje**
   - Formulario con: **role** (select con {Frontend, Backend, Fullstack, DBA}), **mensaje** (`msj`) y **email** para copia.
   - Validaciones:
     - `role` obligatorio y válido.
     - `msj` obligatorio (mín. 10 caracteres).
     - `email` obligatorio, formato válido.
   - Llamar al `POST /messages`. Manejar 201/422 como se indicó.

---

6) Copia por email

- Incluir un campo **Email** en el formulario de contacto para que el sistema **envíe una copia** de la solicitud.
- Este challenge **no exige** implementar el envío real de email; alcanza con:
  - **Validar el email**,
  - **Incluirlo** en el payload o almacenarlo localmente,
  - **Mostrar** en la UI que “se enviará una copia a {email}”.
- Si querés, podés simular el envío con un **servicio mock** o registrar la “copia” en un **historial local**.

---

7) Flujo de usuario esperado

1. **Login** → redirige a **Listado**.
2. En **Listado**, el reclutador ve candidatos, **filtra** y abre el **formulario de mensaje** (embedded o modal).
3. Completa **role**, **msj** y **email**; envía → manejar 201/422 y feedback visual.

4. (Opcional) Ver **historial de mensajes enviados** en la sesión actual.

---

8) Carpeta de prompts (uso de IA)

- Si usaste IA (para código, copy o UI), **creá una carpeta** `/prompts` con los textos que utilizaste y breves notas de cómo los aplicaste.
- Esta permitido el uso de IA/Agentes; nos ayuda a entender tu proceso.

---

9) Entrega

1. **Repo** (GitHub/GitLab/Bitbucket) con: código, `README.md`, `/prompts`, `.env.example`.
2. **README** con:
   - Requisitos (Node versión), cómo correr (`npm i && npm run dev`), build y test.
   - Decisiones de diseño/arquitectura, librerías usadas y por qué.
   - Cómo probar los flujos (incl. escenarios 201/422).
3. Cualquier otra documentacion que creas necesaria.
4. (Opcional) **URL deploy**.

Éxitos!

Cualquier aclaración o supuesto que tomes, **documentalo en el README**.