Frontend Challenge - Amalgama

Buenas 👋! Gracias por tomarte el tiempo de completar el desafío de Amalgama.

Te proponemos que apliques las mejores prácticas que conoces y propongas lo que vos crees es la mejor solución de acuerdo a tus conocimientos! Imagina que el siguiente ejercicio se desarrolla en el contexto de un proyecto real, siempre considerando que el mismo debe poder escalar.

Ejercicio 1 - Componentes

Dada la siguiente vista (por simplicidad sólo se brinda la estructura html, pero puede imaginar el diseño que quiera):

```
const ContactsScreen = ({ contacts, cities, states }) => {
  const contactsToDisplay = contacts.map(contact => ({
    id: contact.id,
        avatar_url: contact.avatar_url,
    full_name: `${contact.first_name} ${contact.last_name}`,
        company: contact.company,
    details: truncate(contact.details, 100),
    email: contact.email,
    phone_number: `(${contact.phone.area_code}) ${contact.phone.number}`,
    addresses: contact.addresses.map(address => ({
      line_1: address.line_1,
      line_2: address.line_2,
      zip_code: address.zip_code,
      city: findById(cities, address.city_id),
      state: findById(states, address.state_id),
        }))
    }));

  return (
        <div>
            <nav>
                <ul>
                    <li><a href="/home">Home</a></li>
                    <li><a href="/contacts">My Contacts</a></li>
                </ul>
            </nav>
            <h1>Contacts 👥</h1>
            {contactsToDisplay.map(contact => (
                <div>
                    <div>
                        <img src={contact.avatar_url} />
```

```
                        <h3>{contact.full_name}</h3> - <h4>{contact.company}</h4>
                    </div>
                <p>{contact.details}</p>
                <ul>
                  <li>email: {contact.email}</li>
                          <li>phone: {contact.phone_number}</li>
                  <li>
                              {contact.addresses.length > 1
                                  ? <h4>Addresses:</h4>
                                  : <h4>Address:</h4>}
                    {contact.addresses.map(address => (
                      <ul>
                        <li>{address.line_1}</li>
                        <li>{address.line_2}</li>
                        <li>{address.zip_code}</li>
                        <li>{address.city}</li>
                        <li>{address.state}</li>
                                  </ul>
                              )}
                          </li>
                      </ul>
                  </div>
              ))}
          </div>
      );
};
```

1. Enunciar todos problemas o posibilidades de mejoras para este componente. Mencionar cuáles de los problemas o posibilidades de mejoras enunciados son los más relevantes.
2. Refactorizar el código y adjuntar cómo quedaría la solución luego de la refactorización.
3. Justificar lo realizado en el punto 2 explicando qué mejoras aporta y por qué soluciona lo comentado en el punto 1.
4. Se pide agregar una vista de perfil del contacto (layout similar a como se muestra en la lista), suponiendo que los datos del contacto son `avatar` , `first_name` , `last_name` , `company` , `details`, `email`, `phone_number` y `address` . Adjuntar la solución propuesta.

Ejercicio 2 - Estado

La aplicación está conectada a una API que devuelve las siguientes respuestas:

```
// GET https://api.org/books

{
    response: [
        {
            id: 1,
      title: 'Clean Code',
      author: {
                id: 1,
      name: 'Uncle Bob'
      }
        },
        {
            id: 2,
      title: 'Clean Architecture',
      author: {
```

```
                id: 1,
        name: 'Uncle Bob'
      }
        }
    ]
}
```

```
// GET  https://api.org/users

{
    response: [
        {
            id: 1,
      email: 'chano@amalgama.co',
      nickname: 'Chano',
      favorite_books: [
                {
            id: 1,
            title: 'Clean Code',
            favorited_at: "2024-01-01",
            author: {
                        id: 1,
              name: 'Uncle Bob'
            }
                }
            ]
        },
        {
            id: 2,
      email: 'sebastian@amalgama.co',
      nickname: 'Biche',
      favorite_books: [
                {
            id: 1,
            title: 'Clean Code',
            favorited_at: "2024-06-30",
            author: {
                        id: 1,
              name: 'Uncle Bob'
            }
                },
                {
                    id: 2,
              title: 'Clean Architecture',
              favorited_at: "2024-12-31",
              author: {
                        id: 1,
                name: 'Uncle Bob'
              }
                }
            ]
        }
    ]
}
```

1. Pensar cómo sería la forma más óptima de guardar esta información en el browser. Codear la solución usando alguna estrategia de state management. Adjuntar el código.
2. Adjuntar en formato JSON cómo quedaría el estado, según lo que se planteó en el punto anterior.
3. Explicar las ventajas que tiene la solución propuesta.

Ejercicio 3 - Práctica

Dado el siguiente endpoint:

- https://2v234d7xc7.execute-api.us-east-1.amazonaws.com/default/login [POST]

  ```
  email: user@amalgama.co
  password: password
  ```

  La API devuelve el token de autorización.

  Posibles errores:

  1. `Status Code 401 Unauthorized`.

  NOTA IMPORTANTE: No acepta JSON, hay que enviarle la información en **application/x-www-form-urlencoded** o en **multipart/form-data.**

Se pide:

1. Crear una pantalla de login con email y password.
   a. Handlear el caso de error.
2. Una vez hecho login, se accede a las pantallas privadas de la app.
   a. Si uno no está logueado, sólo puede ver la pantalla de login.
   b. Si uno está logueado e intenta ir a la pantalla de login, debe redirigir a alguna de las pantallas privadas que se considere como la principal.

Aclaraciones

- Podés usar el framework/librería que más te guste (recomendación React).
- El diseño de la UI no es importante. Podés usar una librería de componentes (no es requerido).
- Se evalúa la aplicación de buenas prácticas de programación, pensando siempre en la escalabilidad del proyecto.
- Si tomaste una decisión con la que no estás contento por falta de tiempo, podés explicar cómo lo mejorarías.