CTC
enseñanza técnica profesional

Instituto Tecnológico Colonia CTC

Analista programador
Desarrollo Para Dispositivos Móviles

Obligatorio 1

Profesor: Micaela Rodríguez

Fernando Bonilla

Nelson Álvarez

2025

Índice

Abstract

En el contexto del primer obligatorio de Desarrollo para Dispositivos Móviles, se nos ha asignado el desafío de desarrollar una aplicación para móvil que incentive el reciclaje en jóvenes y adultos mediante un sistema de retos, seguimiento de acciones sustentables y recompensas virtuales.

Módulos

Módulo Usuario

Inicio de sesión: la funcionalidad de inicio de sesión permite a los usuarios acceder al sistema, ya sea registrándose por primera vez o iniciando sesión si ya están registrados. Para crear una cuenta, el usuario debe completar el formulario de registro, el cual incluye las siguientes validaciones:

- **Nombre**: Campo obligatorio.
- **Correo electrónico**: Debe tener el formato nombre@dominio.com.
- **Contraseña**: Debe cumplir con los siguientes requisitos:
    - Mínimo 6 caracteres.
    - Al menos una letra mayúscula.
    - Al menos una letra minúscula.
    - Al menos un número.
- **Imagen** (seleccionada desde la galería mediante ImagePicker desde galería o la cámara)

Si el usuario ya se ha registrado previamente, solo se solicita:

- **Correo electrónico**
- **Contraseña**

Ambos campos son validados antes de permitir el acceso al sistema.

Módulo Configuración del Usuario

Como parte del modulo anterior, el usuario cuenta con la opción de editar sus datos personales luego de haber iniciado sesión, es parte del perfil del usuario ofrece funcionalidades para modificar su nombre, email, contraseña y foto de perfil.

Estructura del componente:

- Se utiliza ‘useState’ para controlar los inputs locales
- Se obtiene el usuario actual desde el contexto ‘userContext’
- Se muestran los datos actuales como valores por defectos en los inputs y en la imagen
- Al presionar guardar
    - Se valida la entrada
    - Se actualiza el contexto {‘setUser’}
    - Se sobrescribe la información en asyncStorage

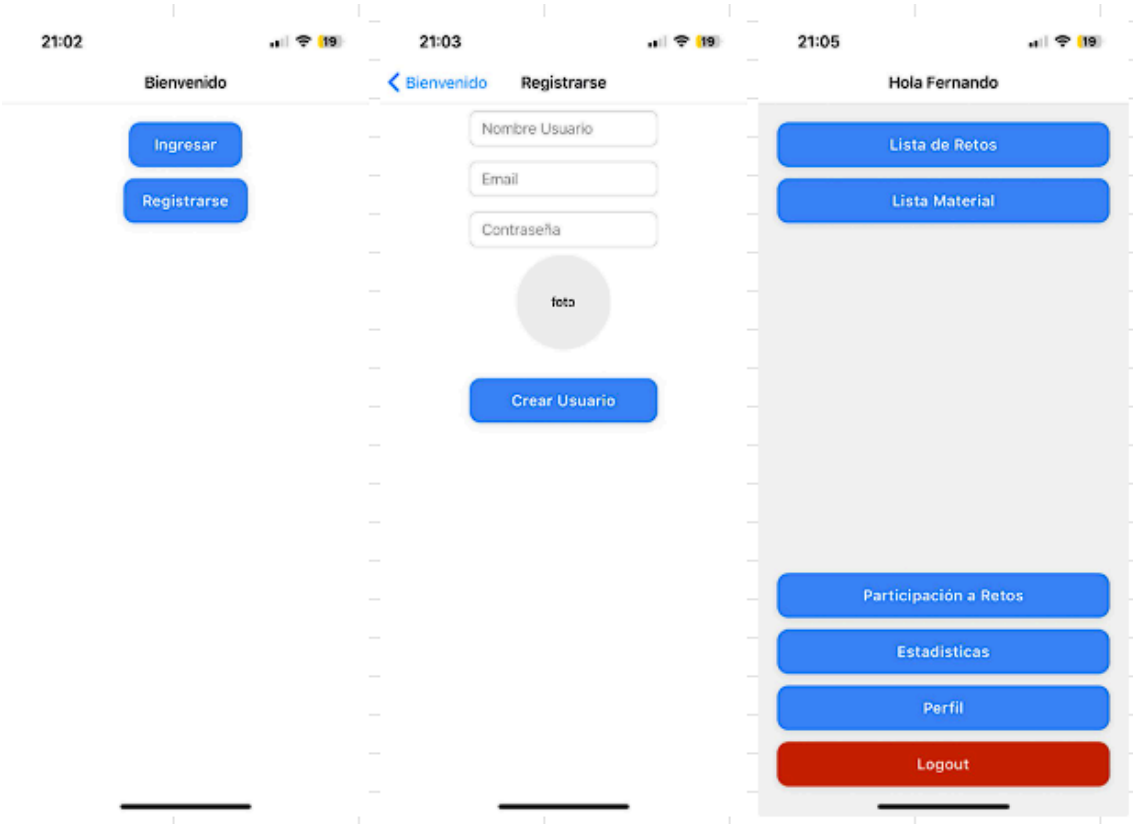
Además, se maneja una persistencia en los datos, es decir, se guarda bajo la nueva clave ‘email’ en asyncStorage. Si el mail fue modificado, el valor anterior también se elimina.

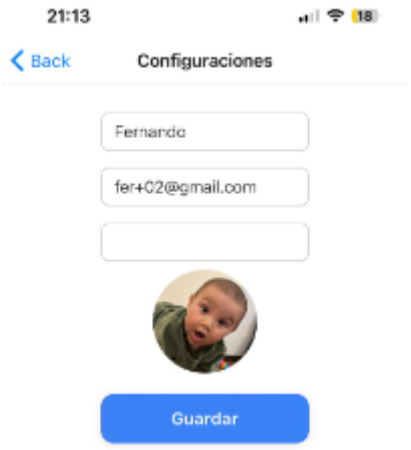
Ejemplo:

```
await AsyncStorage.removeItem(originalEmail);
await AsyncStorage.setItem(newEmail, JSON.stringify(userUpdated));
```

posteriormente, se actualiza el contexto para reflejar los nuevos datos en toda la app

```
setUser(userUpdated);
```


21:02
Bienvenido
Ingresar
Registrarse
21:03
Bienvenido
Registrarse
Nombre Usuario
Email
Contraseña
foto
Crear Usuario
21:05
Hola Fernando
Lista de Retos
Lista Material
Participación a Retos
Estadisticas
Perfil
Logout


21:13
Back
Configuraciones
Fernando
fer+02@gmail.com
Guardar

Módulo Gestión de Reto

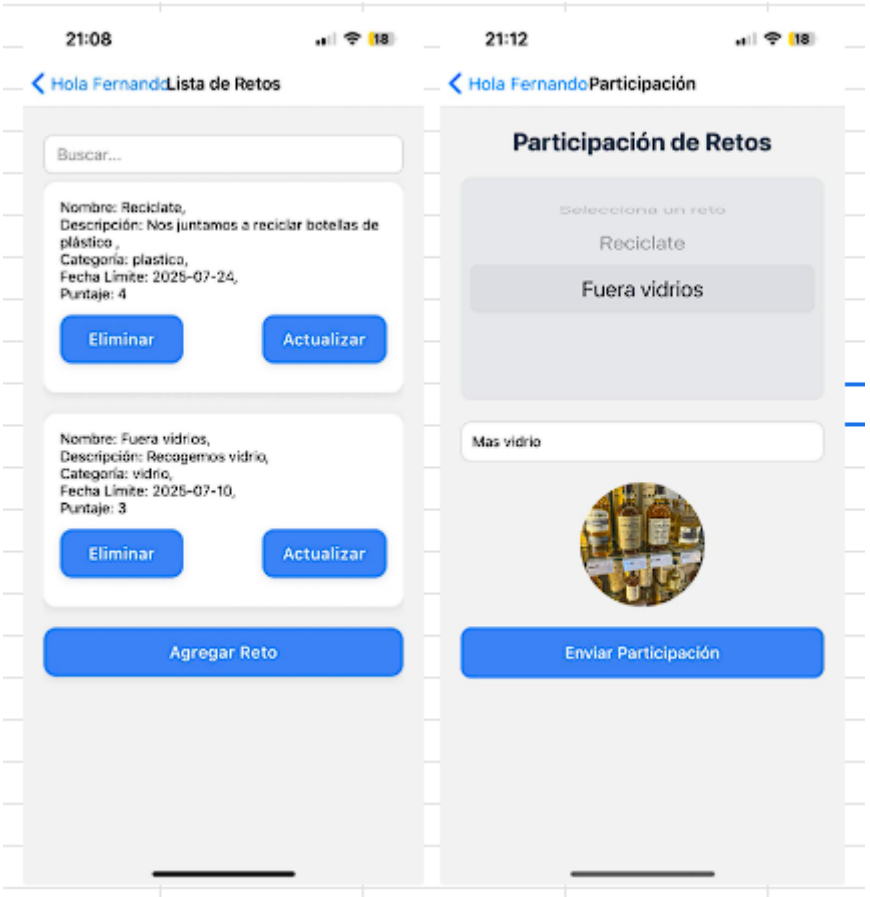
Listado de retos vigentes: Funcionalidad de listado de retos cuya fecha límite (deadline) es igual o posterior a la fecha actual. La información se obtiene desde el almacenamiento local (AsyncStorage) y se filtra automáticamente para no incluir retos vencidos.

Funcionalidades:

- **Carga automática de datos:** Al enfocar la pantalla (useFocusEffect), se cargan y filtran los retos almacenados bajo la clave 'arrayRetos'.
- **Visualización de retos:** Cada reto se muestra utilizando el componente Card, el cual incluye nombre del usuario, descripción, categoría, fecha límite y puntaje.
- **Búsqueda:** Se permite ingresar texto en un campo de búsqueda (a futuro puede conectarse con un filtro real).
- **Gestión individual:**
    - **Eliminar reto:** Se solicita confirmación mediante Alert, y en caso afirmativo, se elimina del listado y del almacenamiento.
    - **Actualizar reto:** Redirige a la pantalla de actualización, enviando el reto actual y su índice como parámetros.
- **Agregar nuevo reto:** Un botón al final de la lista permite navegar a la pantalla de creación de retos.

Validaciones adicionales:

- Las fechas de los retos se comparan con la fecha actual ignorando la hora, para asegurar precisión en la visualización.
- Si no hay retos disponibles, se muestra un mensaje indicando que no hay datos.



Módulo Gestión de Materiales

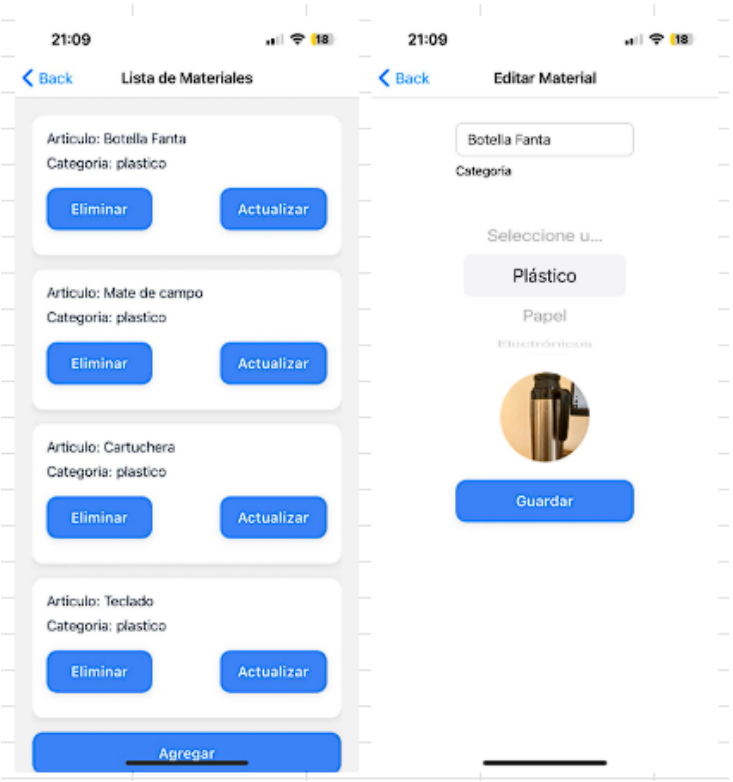
Listado de Materiales: Los materiales previamente guardados en almacenamiento local. El usuario puede consultar la lista, eliminar un material o actualizar su información, así como agregar nuevos materiales.

Funcionalidades:

- **Carga automática de datos:** Al enfocar la pantalla (useFocusEffect), se recupera la lista almacenada bajo la clave 'articulos' en AsyncStorage.
    - Si existen materiales, se cargan y muestran.
    - Si no existen, se notifica al usuario mediante una alerta, y se invita a agregar nuevos materiales.
    - En caso de error durante la carga, se muestra una alerta y se redirige a la pantalla de inicio.
- **Visualización de materiales:** Cada material se presenta en una tarjeta (Card) con su nombre y categoría.
- **Gestión individual de materiales:**
    - **Eliminar material:** Se solicita confirmación antes de eliminar. Una vez confirmado, se actualiza el almacenamiento y se actualiza el listado mostrado.
    - **Actualizar material:** Redirige a la pantalla de edición, enviando el índice y los datos del material seleccionado.
- **Agregar nuevo material:** Botón al final de la pantalla que permite navegar a la pantalla de alta. La misma contiene: Nombre del material, Categoría del material e imágenes(se puede agregar desde galería y desde la cámara).

Consideraciones técnicas:

- Se utiliza KeyboardAvoidingView y ScrollView para asegurar una experiencia de usuario fluida, especialmente en dispositivos móviles con teclado visible.
- El componente Button y Card es reutilizado para todas las acciones posibles sobre cada material.


21:09
Back
Lista de Materiales
Articulo: Botella Fanta
Categoria: plastico
Eliminar
Actualizar
Articulo: Mate de campo
Categoria: plastico
Eliminar
Actualizar
Articulo: Cartuchera
Categoria: plastico
Eliminar
Actualizar
Articulo: Teclado
Categoria: plastico
Eliminar
Actualizar
Agregar
21:09
Back
Editar Material
Botella Fanta
Categoria
Seleccione u...
Plástico
Papel
Guardar

Módulo Gestión de Participación

Participación: permite al usuario participar en un reto vigente, registrando un comentario, una imagen como evidencia y su ubicación actual. Toda la información se almacena localmente en AsyncStorage.

Funcionalidades:

- **Carga de retos vigentes:** Al montar el componente (useEffect), se cargan los retos guardados bajo la clave 'arrayRetos' desde el almacenamiento local, y se filtran por fecha para mostrar únicamente los que aún están vigentes.
- **Selección de reto:** Se utiliza un componente Picker para seleccionar el reto en el que se quiere participar. Solo se listan los retos vigentes.

Campos requeridos para enviar una participación:

1. **Reto seleccionado** (desde el Picker)
2. **Comentario** (campo de texto)
3. **Imagen** (seleccionada desde la galería mediante ImagePicker desde galería o la cámara)
4. **Ubicación** (obtenida al momento del envío con expo-location)

Flujo de envío (handleSubmit):

- Se validan los campos obligatorios.
- Se solicita permiso de ubicación.
- Se obtiene la localización actual del usuario (latitud y longitud).
- Se construye un objeto newParticipation con los siguientes datos:
    - Nombre del reto
    - Comentario
    - Imagen (URI local)
    - Email del usuario
    - Coordenadas de ubicación
    - Fecha actual
    - Puntaje del reto correspondiente
- Se guarda la participación en la clave 'participaciones' de AsyncStorage, agregándola a las existentes si las hubiera.
- Se muestra una confirmación al usuario y se limpian los campos del formulario.

Consideraciones adicionales:

- Se utiliza KeyboardAvoidingView y ScrollView para una correcta experiencia en dispositivos móviles.
- La selección de imagen permite edición básica antes de confirmarla.
- Se muestra una vista previa de la imagen seleccionada.

Módulo Estadísticas

Estadísticas: representa un panel de estadísticas personalizadas para el usuario autenticado. Su propósito principal es visualizar el rendimiento del usuario en las participaciones de los retos, incluyendo:

- Nivel alcanzado
- Total de retos realizados
- Puntaje acumulado

Esto se representa en una gráfica de barras con dicha información.

Funcionalidades principales
Consultas de datos locales:Recupera desde AsyncStorage la información de participaciones guardadas.
Filtrado dinámico: Muestra solo las participaciones asociadas al usuario autenticado.
Cálculo de puntaje: Acumula el puntaje total y clasifica el nivel del usuario en base al mismo.
Visualización gráfica: Usa react-native-chart-kit para mostrar un gráfico de barras con los datos.

Módulo de navegación

Navegación: la navegación está estructurada mediante el uso de React Navigation, una biblioteca estándar en aplicaciones React Native que permite gestionar la transición entre pantallas, el historial de navegación y el flujo de usuario de forma declarativa y escalable.
Estructura
Toda la navegación está contenida en un único archivo central denominado RootStack. Este módulo actúa como **gestor principal del flujo de pantallas**, permitiendo definir las rutas disponibles, los nombres internos de cada vista y las opciones de configuración que se presentan en la interfaz.
La navegación implementada es de tipo **Stack Navigation**, lo que significa que las pantallas se apilan unas sobre otras como páginas, permitiendo al usuario retroceder con gestos o botones nativos.

Objetivos de navegación

- Controlar el flujo entre las distintas funcionalidades de la aplicación.
- Establecer un punto de entrada inicial (Login o Register) y condicionar el acceso al resto de las pantallas según el estado del usuario.
- Permitir la personalización del encabezado, mostrando títulos dinámicos y ocultando botones cuando sea necesario.
- Organizar el sistema en módulos independientes reutilizables y conectables mediante navegación.

Pantallas disponibles y propósito

Cada pantalla representa un "módulo funcional" dentro de la app. Están registradas en el Stack Navigator y permiten navegar entre sí. Las principales incluyen:

- **LoginRegister**: Punto de entrada inicial; permite al usuario elegir entre iniciar sesión o registrarse.
- **Login / Register**: Formulario de acceso o creación de cuenta, con sus respectivas validaciones.
- **Home**: Pantalla principal tras autenticarse. Su encabezado puede mostrar el nombre del usuario logueado.
- **Settings**: Módulo de configuración del usuario; permite editar nombre, correo, contraseña e imagen de perfil.
- **AltaReto / ListRetos / UpdateRetos**: Conforman el módulo de administración de retos.
- **AltaMateriales / ListMateriales / UpdateMateriales**: Gestión de materiales relacionados con retos o actividades.
- **Participate**: Registro de participación del usuario en un reto.
- **UserPanel**: Módulo de visualización de estadísticas o resumen de actividad del usuario.

Contexto de Usuario

El sistema utiliza un **UserContext** para mantener el estado del usuario logueado durante toda la sesión. Esto permite:

- Personalizar contenido (por ejemplo, mostrar "Hola [Nombre]" en el Home).
- Condicionar el acceso a ciertas pantallas a que haya un usuario autenticado.
- Actualizar el perfil desde la pantalla Settings sin romper la navegación.

Comportamiento Dinámico

Algunas pantallas definen comportamientos especiales dentro del Stack:

- **Encabezados personalizados**: se puede cambiar el título mostrado según el usuario u otras condiciones.
- **Control de historial**: en pantallas como Home, se desactiva la posibilidad de volver atrás (evita que un usuario logueado regrese al Login).

Seguridad y Control de Flujo

Aunque el sistema aún no implementa navegación protegida directamente en el Stack, se sientan las bases para ello gracias al UserContext. Esto permitiría, por ejemplo:

- Redirigir automáticamente a LoginRegister si no hay sesión activa.
- Bloquear el acceso manual a rutas sensibles si no hay autenticación.

Consideración de Diseño

- Todas las rutas están definidas en un solo lugar, lo que mejora el mantenimiento y la lectura.
- Las rutas usan nombres internos (name) y títulos visibles (title) separados, permitiendo flexibilidad de cambios en la interfaz sin afectar la lógica.
- Cada pantalla se declara con un componente propio, lo que refuerza la arquitectura basada en módulos independientes.

Módulo Componentes Reutilizables

Componentes: La aplicación cuenta con una serie de **componentes personalizados reutilizables** que encapsulan comportamientos comunes y estilos coherentes. Estos componentes están diseñados para mejorar la modularidad, reducir la duplicación de código y facilitar la escalabilidad de la interfaz.
Los principales componentes definidos son:

1. Button – Botón personalizado
2. Card – Contenedor visual de información
3. ImageComponent – Imagen con estilos dinámicos

Componente Button

Propósito:
Este componente reemplaza el botón nativo para ofrecer un **botón estilizado, reutilizable y personalizable**. Su comportamiento y color pueden definirse desde el exterior.
Props principales:

- title: Texto que se mostrará dentro del botón.
- btnColor: Color de fondo del botón (si no se proporciona, usa un valor por defecto definido en los estilos).
- customPress: Función que se ejecuta al presionar el botón.

Ventajas:

- Centraliza los estilos del botón en un solo lugar (Button.styles).
- Facilita la reutilización en distintas pantallas con diferentes colores o acciones.

Componente Card

Propósito:
Es un contenedor básico para **mostrar datos estructurados**, especialmente usado para representar retos, tareas o entidades con múltiples atributos.
Props esperadas:

- userName: Nombre del creador del contenido.
- description: Descripción del ítem.
- category: Categoría a la que pertenece.
- deadline: Fecha límite asociada.
- score: Valor numérico o puntaje relacionado.

Ventajas:

- Permite representar una unidad informativa de forma clara y ordenada.
- Puede expandirse fácilmente para incluir estilos, íconos o botones de acción si se desea.

Componente ImageComponent

Propósito:
Encapsula la lógica de renderización de una imagen en la interfaz, permitiendo un alto grado de **flexibilidad visual**.
Props principales:

- source: Fuente de la imagen (normalmente un URI).
- style: Permite aplicar estilos adicionales desde el componente padre.

- width, height, borderRadius: Props opcionales que permiten sobrescribir el tamaño y bordes de la imagen.

Ventajas:

- Aporta coherencia visual al manejo de imágenes en la app.
- Permite personalización sin romper el diseño base.
- Especialmente útil cuando se maneja selección de imagen desde cámara o galería (como en Settings).

Relación en el sistema

Estos componentes están pensados para ser **usados en múltiples módulos funcionales**, como:

- Button: utilizado en formularios (Login, Register, Settings, etc.)
- ImageComponent: en selección de perfil o avatares.
- Card: para mostrar retos o materiales dentro de listas.

Su uso asegura una **experiencia de usuario uniforme** y una interfaz visual coherente.

Diagrama de arquitectura de rutas

```
EcoChallenge/
├── src/
│   ├── components/            # Componentes reutilizables
│   │   ├── Button/            # Botón personalizado
│   │   ├── Card/              # Visualización estructurada
│   │   └── Image/             # Imagen con estilo
│   ├── Context/
│   │   └── UserContext.js     # Contexto global de usuario
│   ├── routes/
│   │   └── RootStack.js       # Definición de rutas (pantallas)
│   └── screens/               # Vistas (pantallas) de la app
│       ├── AltaRetos/
│       │   ├── formAlta.js
│       │   ├── listRetos.js
│       │   └── updateRetos.js
│       ├── Materiales/
│       │   ├── AltaMateriales.js
│       │   ├── ListadoMateriales.js
│       │   └── UpdateMateriales.js
│       ├── Login/
│       ├── Register/
│       ├── LoginRegister/
│       ├── Home/
│       ├── Settings/
│       ├── Participate/
│       └── UserPanel/
├── App.js
└── index.js
```