**TECNOLÓGICO NACIONAL DE MÉXICO**

**INSTITUTO TECNOLÓGICO DE TLAXIACO**

**CARRERA:** INGENIERÍA EN SISTEMAS COMPUTACIONALES

**DESARROLLO MÓVIL**

**Reporte de Proyecto**

**“CoffApp”**

**DOCENTE:**

Ing. Sánchez Salmorán Eloy

**Integrantes:**

Irving Fernando Reyes Pacheco

Darío Sánchez Linton

Diego Alexis Carlos Cruz

Edwin Ortiz Cruz

**SEMESTRE:** SÉPTIMO **GRUPO:** 7US

AGOSTO-DICIEMBRE 2023

Heroica Ciudad de Tlaxiaco, Oaxaca, a 06 de diciembre del 2023.

**Introducción**

En el dinámico entorno actual, donde la tecnología se entrelaza con la vida cotidiana, el proyecto "CoffApp" cobra vida como una aplicación móvil destinada a simplificar y mejorar la experiencia en el servicio de cafetería escolar. Este proyecto, impulsado por la versatilidad y eficiencia de React Native, se propone revolucionar la forma en que alumnos y docentes interactúan con la cafetería, llevando la comodidad de realizar pedidos a sus dispositivos móviles de manera ágil y eficaz.

Al elegir React Native como el marco de desarrollo, " CoffApp " se beneficia de la capacidad de crear interfaces de usuario atractivas y funcionales que se adaptan a las plataformas iOS y Android. El enfoque central de la aplicación recae en la gestión óptima de las peticiones, desde la incorporación de un catálogo detallado de productos y sus precios, hasta la creación de un diseño intuitivo que garantice una experiencia de usuario satisfactoria.

A lo largo de este informe, exploraremos cómo la elección de React Native ha influido en la arquitectura de la aplicación, permitiendo una integración fluida de datos en tiempo real, esencial para la gestión eficiente de pedidos en una cafetería escolar. Además, se abordará la importancia de ejecutar pruebas exhaustivas para asegurar un rendimiento sin errores, y se describirá el proceso de corrección, depuración y actualización continua del proyecto.

En síntesis, " CoffApp " no solo representa una solución innovadora para la cafetería escolar, sino también un ejemplo práctico de cómo la tecnología móvil, impulsada por React Native, puede mejorar significativamente la eficiencia y la experiencia del usuario en el ámbito de servicios cotidianos.

**Objetivo General**

Desarrollar una Aplicación móvil capaz de gestionar peticiones propias de un servicio de cafetería escolar.

**Objetivos Específicos**

* Incorporar un listado de productos y precios para las peticiones en tiempo real del alumnado y docentes.
* Diseño básico de la App.
* Estructurar la distribución de los componentes.
* Introducir nombres y precios de los productos.
* Anexar lineamientos para uso correcto del servicio
* Ejecutar de pruebas y detección de errores
* Corrección de errores, depuración y actualización del proyecto.

**Justificación**

El desarrollo de la aplicación móvil “CoffApp” surge como respuesta a la necesidad de modernización y racionalización de los servicios de comedor escolar, aprovechando las ventajas tecnológicas para mejorar la experiencia de usuario tanto de estudiantes como de docentes. La elección de React Native como marco de desarrollo se basa en varias razones estratégicas que mejoran la eficiencia y versatilidad de la aplicación.

Primero, React Native le permite crear aplicaciones multiplataforma. Esto significa un importante ahorro de recursos y tiempo al mantener una base de código común en las plataformas iOS y Android. Esto no sólo simplifica el proceso de desarrollo, sino que también garantiza una experiencia de usuario consistente independientemente del dispositivo utilizado.

Además, la capacidad de React Native para generar interfaces de usuario dinámicas y atractivas es esencial para CoffApp. Esta aplicación tiene como objetivo proporcionar una experiencia de usuario intuitiva y facilitar la navegación y los pedidos en tiempo real. La flexibilidad de React Native para crear interfaces receptivas y visualmente atractivas contribuye en gran medida a este objetivo.

La estructura modular y la arquitectura escalable de React Native optimizan la gestión eficiente de los requisitos en las cafeterías escolares. Agregar catálogos de productos, actualizar precios en tiempo real y ejecutar pruebas para detectar y corregir errores se benefician de la eficiencia y confiabilidad de este marco de desarrollo.

**DESARROLLO**

**TYPESCRIPT**

TypeScript es JavaScript con sintaxis para tipos. TypeScript es un lenguaje de programación fuertemente tipado que se basa en JavaScript y le brinda mejores herramientas a cualquier escala. Agrega sintaxis adicional a JavaScript para admitir una integración más estrecha con su editor . Detecte los errores desde el principio en su editor. El código TypeScript se convierte a JavaScript, que se ejecuta en cualquier lugar donde se ejecute JavaScript : en un navegador, en Node.js o Deno y en sus aplicaciones.

TypeScript comprende JavaScript y utiliza la inferencia de tipos para brindarle excelentes herramientas sin código adicional; comprueba si un programa tiene errores antes de su ejecución y lo hace en función de los tipos de valores , lo que lo convierte en un verificador de tipos estáticos .

**Un superconjunto escrito de JavaScript**

Sintaxis

TypeScript es un lenguaje que es un superconjunto de JavaScript: la sintaxis JS es, por lo tanto, TS legal. La sintaxis se refiere a la forma en que escribimos texto para formar un programa.

**VISUAL STUDIO CODE**

Visual Studio Code es un editor de código fuente liviano pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución (como C++, C#, Java, Python, PHP, Go, .NET).

**NODE JS**

Node.js, es un entorno en tiempo de ejecución multiplataforma para la capa del servidor (en el lado del servidor) basado en JavaScript.

Node.js es un entorno controlado por eventos diseñado para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo. Gracias a esta característica, no tienes que preocuparte con el bloqueo de procesos, pues no hay bloqueos.

El diseño de Node.js está inspirado en sistemas como el Event Machine de Ruby o el Twisted de Python. Sin embargo, Node.js presenta un bucle de eventos como una construcción en tiempo de ejecución en lugar de una biblioteca. Este bucle de eventos es invisible para el usuario.

Otra característica especial de Node.js es que está diseñado para simplificar la comunicación. No tiene subprocesos, pero te permite aprovechar múltiples núcleos en su entorno y compartir sockets entre procesos.

Puedes utilizar Node.js para diferentes tipos de aplicaciones. Los siguientes son algunos de los ejemplos:

- Aplicaciones de transmisión de datos (streaming)

- Aplicaciones intensivas de datos en tiempo real

- Aplicaciones vinculadas a E/S

- Aplicaciones basadas en JSON:API

- Aplicaciones de página única

**ANDROID STUDIO**

Android Studio es el entorno de desarrollo integrado (IDE) oficial que se usa en el desarrollo de apps para Android. Basado en el potente editor de código y las herramientas para desarrolladores de IntelliJ IDEA, Android Studio ofrece aún más funciones que mejoran tu productividad cuando compilas apps para Android, como las siguientes:

* Un sistema de compilación flexible basado en Gradle
* Un emulador rápido y cargado de funciones
* Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
* Ediciones en vivo para actualizar elementos componibles en emuladores y dispositivos físicos, en tiempo real
* Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
* Variedad de marcos de trabajo y herramientas de prueba
* Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros
* Compatibilidad con C++ y NDK
* Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine
* En esta página, encontrarás una introducción a las funciones básicas de Android Studio. - Para acceder a un resumen de los cambios más recientes, consulta las notas de la versión de Android Studio.

**Sistema de compilación de Gradle**

Android Studio usa Gradle como base del sistema de compilación, y el complemento de Android para Gradle proporciona capacidades específicas de Android. Este sistema de compilación se ejecuta en una herramienta integrada desde el menú de Android Studio, y lo hace independientemente de la línea de comandos. Puedes usar las funciones del sistema de compilación para lo siguiente:

* Personalizar, configurar y extender el proceso de compilación
* Crear varios APKs para tu app; diferentes funciones usan el mismo proyecto y los mismos módulos
* Volver a utilizar códigos y recursos entre conjuntos de orígenes
* Gracias a la flexibilidad de Gradle, puedes lograrlo sin modificar los archivos fuente de tu app.

Los archivos de compilación de Android Studio se denominan build.gradle.kts si usas Kotlin (recomendado) o build.gradle si usas Groovy. Son archivos de texto sin formato que usan la sintaxis de Kotlin o Groovy para configurar la compilación con elementos que proporciona el complemento de Android para Gradle. Cada proyecto tiene un archivo de compilación de nivel superior para todo el proyecto y archivos de compilación de nivel de módulo independientes para cada módulo. Cuando importas un proyecto existente, Android Studio genera automáticamente los archivos de compilación necesarios.

**METRO BUNDLER**

Metro bundler es la herramienta que empaqueta todos los archivos de React Native en un solo fichero preparado para instalarse en el dispositivo y que escucha los cambios en los archivos para hacer el hot reload en el target que corresponda.

Metro es un paquete de JavaScript que incluye opciones, un archivo de entrada y le devuelve un archivo JavaScript que incluye todos los archivos JavaScript. Cada vez que ejecuta un proyecto nativo de reacción, se realiza una compilación de muchos archivos javascript en un solo archivo. Esta compilación se realiza mediante un paquete que se llama Metro . Metro se inicia junto con el servidor de nodo que se está ejecutando al hacer ' npm start ' y podemos ver BUNDLE escrito en un cuadro verde en la imagen de arriba que muestra el paquete de archivos cada vez que ejecutamos un nuevo proyecto o recargamos un el actual.

El empaquetador Metro realiza algunos trabajos:

La característica más importante de Metro es proporcionar ciclos de recarga de menos de un segundo, ya que agregan y depuran las IU de las aplicaciones muy rápidamente en cuestión de segundos, por lo que no tiene que esperar a que toda la aplicación se reinicie y comience a funcionar. Una función similar está integrada en la biblioteca Flutter, que también es una herramienta multiplataforma para crear aplicaciones móviles.

Otra característica es proporcionar un inicio rápido de la aplicación en el dispositivo/emulador proporcionando una velocidad de agrupación rápida.

Metro combina todo el código JavaScript en un solo archivo y traduce cualquier código Javascript que el dispositivo no entienda (como JSX o alguna sintaxis JavaScript más nueva).

Metro convierte activos (por ejemplo, archivos PNG) en objetos que pueden mostrarse mediante un componente de ‘ Imagen’.

**ANDROID SDK**

El SDK de Android es un conjunto de herramientas y bibliotecas de desarrollo de software que se requieren para desarrollar aplicaciones Android. Cada vez que Google lanza una nueva versión o actualización de Android, también se lanza un nuevo SDK que los desarrolladores deben descargar e instalar. Cabe destacar que, si bien también puedes descargar y utilizar el SDK de Android sin necesidad de usar Android Studio, por lo general deberás trabajar con Android Studio para cualquier desarrollo de Android.

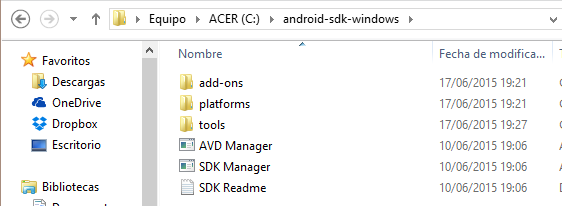
El Android SDK comprende todas las herramientas necesarias para codificar programas desde cero e incluso para ponerlos a prueba. Estas herramientas permiten que el proceso de desarrollo fluya sin problemas, desde el desarrollo y la depuración de programas hasta el empaquetado.

El Android SDK es compatible con Windows, macOS y Linux, de modo que puedes desarrollar en cualquier de estas plataformas.

**VARIABLES DE ENTORNO ANDROID\_HOME**

En las Variables de entorno vamos a especificar lo que ya configuramos, ANDROID\_SDK\_ROOT y PATH. Y cabe mencionar que es muy importante que los valores en las variables coincidan con las rutas configuradas anteriormente en los pasos anteriores, Todo depende de ello.

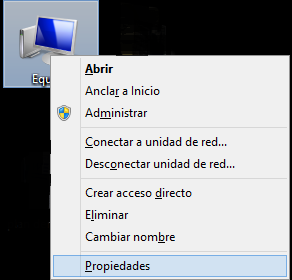
Entramos a la carpeta "android-sdk-windows" y observamos su estructura de carpetas:



**Ahora copiaremos la ruta de esta carpeta y la usaremos para crear la variable de** entorno ANDROID\_HOME:



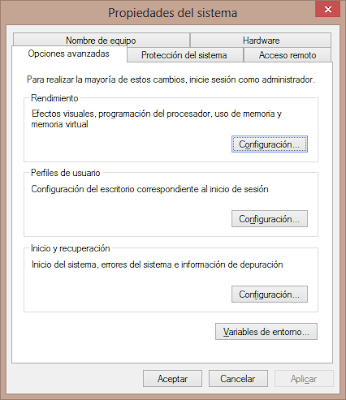
Ahora sobre el icono de "Equipo" hacemos click derecho y del menu que se nos muestre elegimos "Propiedades":



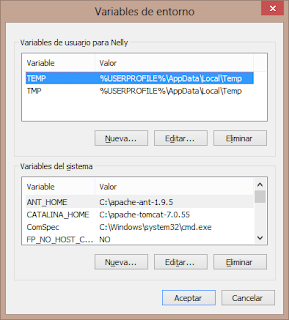
En la siguiente ventana que se nos muestra, del lado izquierdo seleccionamos "Configuración avanzada del sistema":



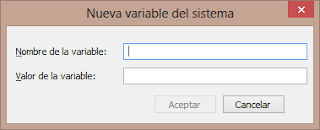
Se nos muestra la ventana "Propiedades del sistema", hacemos click sobre el botón "Variables de entorno ..." :



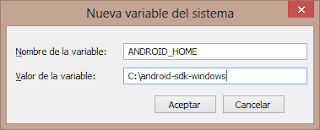
Desde la ventana "Variables de entorno" crearemos la variable de entorno ANDROID\_HOME, para esto hacemos click sobre el botón "Nueva ...":



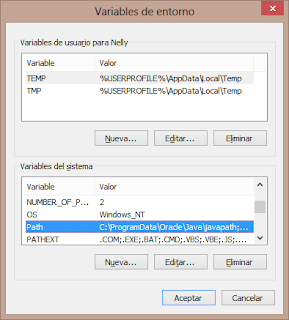
Se nos muestra la ventana "Nueva variable del sistema":



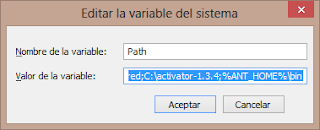
Desde la ventana "Nueva variable del sistema" agregamos la ruta que copiamos del Paso 08 para crear la variable de entorno ANDROID\_HOME, luego hacemos click en el botón "Aceptar" para guardar los cambios.



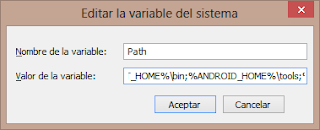
Se nos regresa a la ventana "Variables de entorno", desde aquí buscamos y seleccionamos la variable Path y hacemos click sobre el botón "Editar ...":



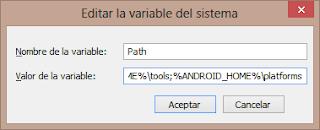
Aparece la ventana "Editar la variable del sistema" con los valores que ya tenia la variable Path:



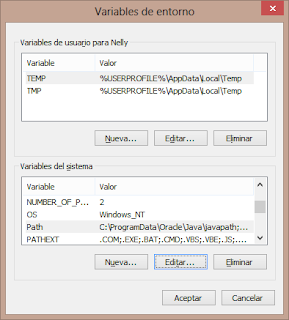
Nos ubicamos en el campo "Valor de la variable" y nos desplazamos hasta el final de lo que contenga dicho campo y le agregamos ";%ANDROID\_HOME%\tools". El carácter ";" se usa para separar valores.



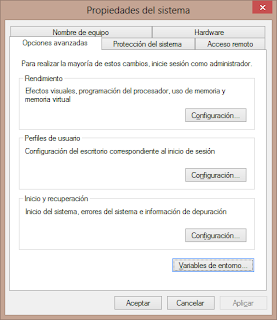
Ahora sobre ese mismo campo agregamos ";%ANDROID\_HOME%\platforms". Finalmente damos click sobre el botón "Aceptar" para guardar los cambios sobre la variable Path.



Se nos regresa a la ventana "Variables de entorno", aquí solo damos click en el botón "Aceptar":



Aparece la ventana "Propiedades del sistema", aquí solo damos click en el botón "Aceptar":



**Descripción de pantallas**

**Login**

La distribución de la pantalla Login consta del título en la parte superior y en orden descendente está el logo, un texto con la palabra bienvenidos y posteriormente, dividido en vistas específicas mediante separadores, están dos cajas de entrada de texto (TextInput) y finalmente un botón Ingresar para acceder al Menú.

La programación del logueo captura una cadena de texto con formato de correo electrónico y una contraseña de tipo cadena de texto, y lo compara con una base de datos establecida en un servidor en la nube proporcionado por la herramienta Firebase, por lo cual es necesaria una conexión a Internet.



**Pantalla Menú**

A continuación, se despliega la pantalla del menú principal conformada por dos bonotes **(TouchableOpacity)** con una imagen integrada para mostrar la lista de productos ya sea de bebidas o comidas con los que cuenta la cafetería. De esta manera al presionar cualquiera de ellos se puede acceder a la pantalla correspondiente para elegir los productos y agregarlos a la orden.

Tanto el botón Ingresar como los botones de menú Comidas y Bebidas ejecutan la función navegación **(Navigation)** heredada de las librerías de **React** mediante la orden **OnPress**.



**Pantalla Comidas**

La pantalla Comidas está gestionada por un navegador de pestañas inferiores **(BottomTabs)** que contiene tres pestañas que muestran una clasificación de comidas cada una. Se añadieron íconos de la fuente **Ionic Open Source** que le dan una vista más atractiva y un manejo más fluido y fácil de entender desde el punto de vista de la experiencia del usuario.

La cabecera de la pantalla del menú comidas tiene propiedades del **Stack Navigation** por lo que con el botón retroceder podemos regresar inmediatamente al Menú Principal independientemente de la pestaña de comidas donde se encuentre el usuario; de esta manera no es necesario regresar a la pestaña de inicio del menú comidas.

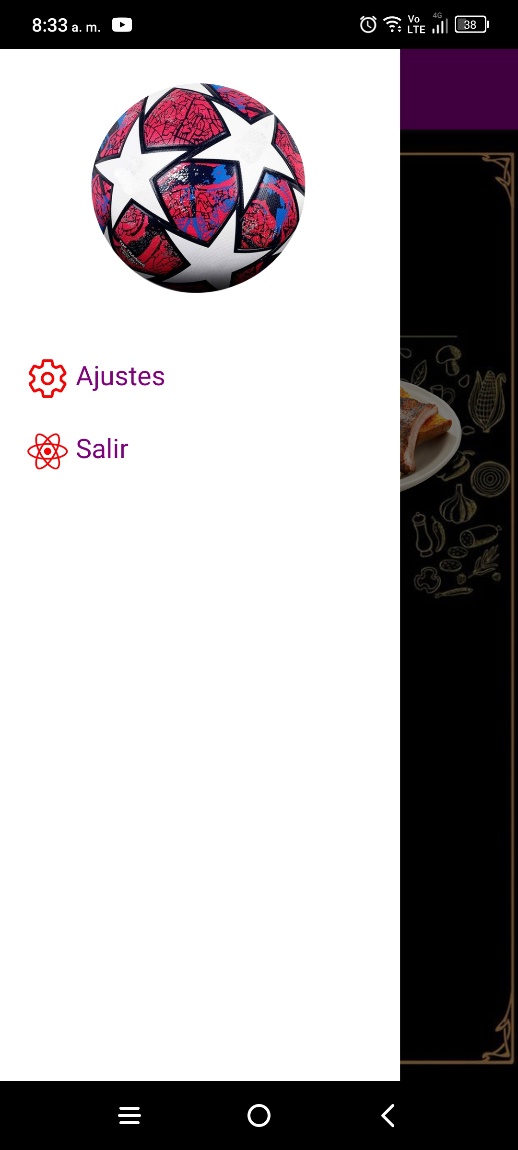
**Pantalla Bebidas**

La pantalla Bebidas está distribuida de una manera similar al menú de la sección de comidas, pero en este caso se implementó un navegador de pestañas superiores **(TopTabs)** incluidas en la navegación que proporciona **React** con la incorporación de íconos para cada pestaña que como ya se mencionó anteriormente ofrece una experiencia de usuario más agradable y precisa.

**Menú Lateral**

Dentro de las posibilidades de navegación se incorporó un menú lateral **(Drawer Navigator)** accesible desde la pantalla de menú principal el cual contiene una sección para realizar algunos ajustes, una seccíon para navegar a la orden realizada y un elemento para salir de la aplicación y cerra la sesión del usuario.



**Estilos Globales**

Para estilizar tanto las pantallas como los componentes y elementos en la aplicación se creó un archivo llamado **EstilosGlobales.tsx** en el cual se define mediante un formato incorporado a la potente herramienta de Typescript cada detalle de estilo para los componentes que así lo requieran y que muestren un diseño agradable a la vista y útil para ubicar con rapidez y certeza las necesidades propias de un menú de cafetería.

**ELEMENTOS ADICIONALES**

**- ImageBackGround**

Cada pantalla cuenta con ina imagen de fondo que junto con los estilos globales se renderizan para dar forma de manera completa a la vista de la aplicación y potencialice la armonía de la presentación de la interfaz de usuario.

**- Button y TouchableOpacity**

Generan botones necesarios para realizar acciones como navegar hacia otras pantallas, ingresar a la aplicación o añadir un producto a la orden.

**- View**

Proporciona el espacio necesario para incorporar elementos que se quieran mostrar en la pantalla, es el componente básico para desplegar elementos visibles en la aplicación.

**- Text**

Forma una cadena de texto que se puede visualizar dentro de una vista, botón o algún elemento que soporte su integración.

**- Navigation Container**

Engloba las pantallas de navegación en pilas que se definen para su eventual acceso.

**- Separator**

Crea un objeto de diseño tipo separador para estilizar la separación entre los elementos View del programa.

**- SafeAreaView**

El propósito de SafeAreaView es presentar contenido dentro de los límites del área segura de un dispositivo.

**INCORPORACIÓN DE UNA API**

Una API, o interfaz de programación de aplicaciones, es un conjunto de reglas definidas que permiten que diferentes aplicaciones se comuniquen entre sí. Actúa como una capa intermedia que procesa las transferencias de datos entre sistemas, permitiendo a las empresas abrir sus datos y funcionalidades de aplicaciones a desarrolladores externos, socios comerciales y departamentos internos dentro de sus empresas.

**Instalación y Configuración para API REST**

Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan como JSON y se sincronizan en tiempo real con cada cliente conectado. Cuando crea aplicaciones multiplataforma con nuestras plataformas Android, Apple y SDK de JavaScript, todos sus clientes comparten una instancia de Realtime Database y reciben automáticamente actualizaciones con los datos más recientes.

Podemos usar cualquier URL de Firebase Realtime Database como punto final REST. Todo lo que tenemos que hacer es agregar .json al final de la URL y enviar una solicitud desde nuestro cliente HTTPS favorito.

- Crear una cuenta

- Crear una base de datos

- Elegir una biblioteca auxiliar

- Estructurar datos para Realtime Database

- Guardar datos

- Recuperar datos

- Ver la base de datos FireBase en consultas.

Para una explicación más detallada es preciso consultar la documentación en loa página oficial de FireBase.

**CONCLUSIÓN**

Al culminar el desarrollo de "CoffApp" con React Native, nos sentimos emocionados al ver cómo nuestro esfuerzo se ha traducido en una herramienta innovadora y práctica para la cafetería escolar. La elección de React Native resultó ser un acierto, ya que nos permitió abordar los desafíos de manera efectiva y ofrecer una experiencia fluida y consistente en dispositivos iOS y Android.

La incorporación del catálogo de productos y precios en tiempo real ha sido un logro gratificante. Ver cómo los usuarios pueden realizar pedidos con solo unos toques en sus teléfonos móviles nos llena de satisfacción, sabiendo que hemos simplificado un proceso que antes podía resultar engorroso.

El diseño de la interfaz de usuario, cuidadosamente desarrollado con las herramientas proporcionadas por React Native, refleja nuestra atención a los detalles y nuestro deseo de proporcionar una experiencia visualmente atractiva. Nos enfrentamos a desafíos, pero la flexibilidad de React Native nos permitió superarlos y crear una aplicación que realmente destaca.

La estructura modular y la arquitectura escalable nos brindaron una base sólida para la gestión de componentes y la incorporación de nuevas funcionalidades. Las pruebas continuas fueron esenciales para identificar y solucionar problemas, y aprendimos mucho durante este proceso, mejorando nuestras habilidades de desarrollo y resolución de problemas.

En última instancia, " CoffApp " no es solo un proyecto, sino una experiencia que ha fortalecido nuestra comprensión del desarrollo de aplicaciones móviles y nos ha enseñado la importancia de adaptarse a las necesidades del usuario. Mirando hacia atrás, estamos orgullosos del resultado y emocionados por el impacto que esta aplicación puede tener en la vida diaria de nuestra comunidad escolar. Este proyecto no solo representa un logro técnico, sino también una lección valiosa sobre la capacidad de la tecnología para mejorar aspectos prácticos de la vida cotidiana.

Principio del formulario

**BIBLIOGRAFÍA**

https://firebase.google.com/docs/database/rest/start?hl=es

<https://reactnavigation.org/docs/getting-started/>

<https://reactnative.dev/docs/>

https://www.ibm.com/mx-es/topics/api

<https://www.typescriptlang.org/>

<https://nodejs.org/docs/latest/api/documentation.html>

<https://developer.android.com/studio?gclid=CjwKCAiAvdCrBhBREiwAX6-6UjZWwPBlktnQc4S_lCkbZJ4NHxdACisfM4l54IlhFjuX0HuOw7HoWRoCE1MQAvD_BwE&gclsrc=aw.ds&hl=es-419>

<https://docs.expo.dev/guides/customizing-metro/>