

**Universidad Autónoma de Querétaro  
Facultad de Informática  
Ingeniería de Software (SOF18)**

**Administración de Proyectos de Software**

**Maestro:** Alberto Lamadrid Alvarez

**Lunes y Miércoles, de 13:00 - 15:00**

**“Documentación Técnica (Manual Técnico)”**



**Integrantes:**

- García Vargas Michell Alejandro - 259663
  - Flores Espinoza Luis Eduardo - 252589
- Mendieta Robledo Carlos Abraham - 290273
  - Velázquez Campos Leonardo - 250893

**Semestre:** 7°      **Grupo:** 30

**Fecha:** 25 de enero del 2023



## «Documentación Técnica»

# Contenido:

I.	¿Quiénes somos?	2
II.	Propuesta de la aplicación	2
III.	Objetivo	2
IV.	Planeación del proyecto	2
	• Verificación de Riesgos	2
	• Riesgos y soluciones	3
	• Asignación de recursos	4
	• Planificación del tiempo	5
	• Diagrama de GANTT	5
	• División de tareas	5
	• Método FPA (Functional Points Analysis)	6
	• Cálculo de Factores Ajustables (VAF)	8
	• Clasificación de componentes	9
V.	Viabilidad del proyecto	11
VI.	Requerimientos	12
	• Requerimientos del funcionales	12
	• Requerimientos no funcionales	13
	• Funcionalidad y servicios ofrecidos	14
	• Análisis de las tecnologías	14
VII.	Antecedentes	15
VIII.	Software base del sistema y prerequisites	16
IX.	Modelo	16
X.	Resultados	21
	• Diseño	21
	• Código	25
XI.	Instalación de proyecto localmente	76
XII.	Financiación	76

## <¿Quiénes somos?>

En Devs\_lutions Trabajamos juntos para diseñar, crear y desarrollar productos de los que estamos orgullosos para ofrecer un servicio personalizado profesional y confiable.

## <Propuesta de la aplicación>

Una app para proporcionar un servicio de pickup para los pequeños locales que quieran ofrecer otra opción de compra a sus clientes teniendo estos un servicio más rápido y efectivo.

## <Objetivo de la aplicación>

- Dar acceso al menú que se ofrece en el establecimiento.
- Ofrecer otra forma de comprar en el establecimiento.
- Tener un servicio más rápido y efectivo para los clientes.
- Otorgar soluciones sencillas y efectivas a los PYMES (Pequeñas y Medianas Empresas) haciendo uso de tecnologías de la información.

## <Planeación del proyecto>

### Verificación de Riesgos

Riesgo	Posibilidad	Impacto del riesgo	Encargado del riesgo	Prioridad
Ataques externos	Alto	Alta	Michell	Alta
Errores humano	Alto	Alta	Luis	Alta
Problemas de comunicación	Media	Media	Carlos	Media
Retrasos en el cronograma	Baja	Baja	Leonardo	Media
Errores en el servidor	Baja	Alta	Michell	Media
Trabajo imprevisto	Baja	Baja	Carlos	Baja

### Riesgos y soluciones

Riesgo	Solución
<b>Ataques externos</b>	<ol style="list-style-type: none"> <li>1. Verificar actualización de software.</li> <li>2. Ejecutar análisis programados regularmente para verificar posibles brechas en la seguridad.</li> <li>3. Verificar que la red sea segura.</li> </ol>
<b>Errores humanos</b>	<ol style="list-style-type: none"> <li>1. Analizar los posibles errores en el proceso.</li> <li>2. Planificar una solución.</li> <li>3. Agregarla al cronograma.</li> <li>4. Verificar afectaciones al sistema.               <ol style="list-style-type: none"> <li>a. De haber afectaciones, planificar soluciones para cada afectación dada por el error inicial.</li> <li>b. De no haber afectaciones, continuar con el cronograma.</li> </ol> </li> </ol>
<b>Problemas de comunicación</b>	<ol style="list-style-type: none"> <li>1. Reorganizar el equipo.</li> <li>2. Solucionar conflictos internos.</li> <li>3. Consultar si la falta de comunicación afectó el trabajo del equipo.</li> <li>4. Analizar los posibles errores en el proceso.</li> <li>5. Planificar una solución.</li> <li>6. Agregarla al cronograma.</li> <li>7. Verificar afectaciones al sistema.               <ol style="list-style-type: none"> <li>a. De haber afectaciones, planificar soluciones para cada afectación dada por el error inicial.</li> <li>b. De no haber afectaciones, continuar con el cronograma.</li> </ol> </li> </ol>
<b>Retrasos de cronograma</b>	<ol style="list-style-type: none"> <li>1. Reasignar las tareas a los miembros con mayor capacidad en esa actividad.</li> <li>2. Redireccionar actividades futuras a un tiempo adecuado a cumplir.</li> </ol>
<b>Errores en el servidor</b>	<ol style="list-style-type: none"> <li>1. Redireccionar el tráfico del servidor afectado a un servidor secundario.</li> <li>2. Contactar y consultar posible fallo del servidor.</li> </ol>
<b>Trabajo imprevisto</b>	<ol style="list-style-type: none"> <li>1. Analizar la prioridad de cada nueva actividad.</li> <li>2. Conforme a su prioridad agregarlas al cronograma.</li> </ol>

**Estimación de Costos**

Recurso	Emisor	Valor
Licencia de Desarrollador de Apple. Apple Developer Program	Apple Inc	\$99 USD Tarifa anual.
Licencia de Desarrollador de Google	Google LLC	\$25 USD Pago único.
Azure cloud	Microsoft	Costo variable conforme la demanda del servidor.
Comisión por transacción	Entidad bancaria o financiera	3.4% aprox por cada transacción.

**Asignación de Recursos**

Tomando que los recursos son aquellos elementos que se requieren para la implementación de un proyecto y cumplir con sus tareas, se muestra que recursos se asignan a los integrantes de Devs\_lutions para cumplir con el objetivo:

	Labor	Equipo	Materiales
Michelle	Desarrollo, documentación	Computador, internet	Expo, Visual Studio Code
Leonardo	Diseño, documentación	Computador, internet	Lucidchart, Figma
Carlos	Diseño, Desarrollo	Computador, internet	Visual Studio Code
Luis	Desarrollo	Computador, internet	Visual Studio Code



	Actividades Clave	Distribución de Actividades	Líder	Tiempo
1	Viabilidad del Proyecto	Michell	Michell	3 hrs.
1.1	Viabilidad Económica	-	-	1 hr.
1.2	Viabilidad Técnica	-	-	1 hr.
1.3	Viabilidad Tiempo	-	-	1 hr.
2	Planeación	Luis	Luis	6 hrs.
2.1	Verificación de los Riesgos	-	-	1 hr.
2.2	Estimación de Costos	-	-	1 hr.
2.3	Asignación de Recursos	-	-	1 hr.
2.4	Planificación del Tiempo	-	-	2 hrs.
2.5	División de Tareas	-	-	1 hr.
3	Toma de Requerimientos Funcionales	Leonardo	Leonardo	3 hrs.
3.1	Visita al Cliente	-	-	1 hr.
3.2	Análisis de Propuestas	-	-	2 hrs.
4	Toma de Requerimientos No Funcionales	Leonardo	Leonardo	3 hrs.
4.1	Visita al Cliente	-	-	1 hr.
4.2	Análisis de Propuestas	-	-	2 hrs.
5	Análisis	Michell, Luis	Michell	10 hrs.
5.1	Comprensión de los Requerimientos	-	-	2:30 hrs.
5.2	Análisis de las Tecnologías	-	-	2:30 hrs.
5.3	Jerarquización de Requerimientos	-	-	2:30 hrs.
5.4	Revisión de Antecedentes	-	-	2:30 hrs.
6	Diseño	Leonardo, Carlos, Luis	Leonardo	32 hrs.
6.1	Diseño UX de la Interfaz	-	-	6 hrs.
6.2	Diseño (Maquetado/modelado) UI de la Interfaz	-	-	6 hrs.
6.3	Modelación de la Base de Datos	-	-	8 hrs.
6.4	Modelación de la Estructura del Proyecto	-	-	4 hrs.
6.5	Modelación de la Funcionalidad del Proyecto	-	-	8 hrs.
7	Desarrollo (Codificación)	Michell, Luis, Carlos, Leonardo	Michell	110 hrs.

7.1	Versionado	-	-	5 hrs.
7.2	Desarrollo de Interfaces	-	-	30 hrs.
7.3	Desarrollo de la Base de Datos	-	-	15 hrs.
7.4	Desarrollo de la Lógica del Programa	-	-	40 hrs.
7.5	Interconectar la Usabilidad del Programa	-	-	20 hrs.
8	Pruebas	Luis	Luis	40 hrs.
8.1	Testeo de la Aplicación	-	-	15 hrs.
8.2	Detectar Posibles Errores	-	-	10 hrs.
8.3	Validación de Errores	-	-	15 hrs.
9	Desplegar	Michell	Michell	10 hrs.
9.1	Obtención de Licencias de las Tiendas	-	-	3 hrs.
9.2	Empaquetado de la Aplicación	-	-	4 hrs.
9.3	Subir Aplicación	-	-	3 hrs.
10	Documentación	Michell, Leonardo, Carlos	Carlos	240 hrs.
11	Distribución	Michell, Luis, Carlos, Leonardo	Leonardo	10 hrs.
11.1	Dar a Conocer la Aplicación	-	-	10 hrs.
12	Revisión	Leonardo	Leonardo	6 hrs.
12.1	Platica con el Cliente	-	-	2 hrs.
12.2	Recibir Retroalimentación	-	-	4 hrs.
13	Mantenimiento	Carlos	Carlos	240 hrs.
13.1	Revisión de Errores	-	-	Indefinido
13.2	Validación de los Recursos	-	-	Indefinido
13.3	Adaptación a Nuevas Versiones	-	-	Indefinido
			Tiempo total en horas.	473

### **Método FPA (Functional Points Analysis)**

Con la finalidad de medir el tamaño funcional del sistema, se requirió del método de puntos funcionales, que mide el tamaño funcional observando las transacciones (funcionales), y los ficheros de datos (lógicos), que son relevantes al usuario en el negocio. Después de identificar los componentes, se determina el tamaño funcional de la aplicación mediante una suma del tamaño de cada uno de los componentes lógicos.



Atributos		Complejidad del Componente									Total
		Com pon ente	Baj a	Tot al	Co mp one nte	Med ia	Tot al	Comp onente	Alta	Total	
<b>Mediciones Funcional Transaccional</b>	Entrada Externa (EI)	0	3	0	10	4	40	3	6	18	58
	Salida Externa (EO)	6	4	24	0	5	0	2	7	14	38
	Consultas Externas (EQ)	0	3	0	12	4	48	4	6	24	72
<b>Mediciones Funcional de Datos</b>	Archivo Lógico Interno (ILF)	5	7	35	0	10	0	0	15	0	35
	Archivo de Interfaz Externa (EIF)	1	5	5	0	7	0	0	10	0	5
<b>No. Total de Puntos Funcionales sin Ajustar (PFsA)</b>											<b>208</b>

### Calculo de Factor Ajustable (VAF)

Llamado Factor de complejidad técnica, basado en las 14 características generales del sistema que evalúan la funcionalidad general de la aplicación. Cada característica tiene una serie de cuestiones que determina un grado de importancia en el sistema, en función a una escala que va desde cero (sin influencia) a cinco (esencial).

Características Generales del Sistema (GSCs)		Calificación
1	Comunicación de datos.	5
2	Procesamiento de datos distribuido.	4
3	Rendimiento.	4
4	Uso del hardware existente.	2
5	Transacciones.	5
6	Entrada de datos interactiva.	5

7	Eficiencia.	3			
8	Actualizaciones on-line.	2			
9	Complejidad del Procesamiento.	1			
10	Reusabilidad.	3			
11	Facilidad de conversión e instalación.	3			
12	Facilidad de operación.	4			
13	Múltiples instalaciones.	1			
14	Facilidad de mantenimiento.	2			
$VAF = 0.65 + 0.01 \sum_{i=1}^{14} F_i$		<b>VAF =</b>	$\frac{1.0}{9}$	<b>Total =</b>	44

Una vez que se determinó la influencia de cada característica del sistema, se utilizó la fórmula para obtener el valor del VAF. Siendo F, el valor adjudicado a cada característica general del sistema. El VAF es utilizado como factor corrector del conteo de Puntos Funcionales. Se debe tomar en cuenta la influencia en un 35% en la cuenta total de Puntos de Función.

### **Clasificación de componentes:**

Se determina la complejidad de cada componente, siendo subjetivo, se debe tomar lo siguiente:

- **Entradas Externas:**

Para todas las entradas externas que referencian a tres archivos, si consta de cuatro o más tipos de datos distintos, la Entrada externa será considerada como complejidad Alta, si consta de menos de cuatro tipos de datos distintos será considerada complejidad Media, y para cualquier entrada externa que referencia a menos de tres archivos será considerada de complejidad baja.

- **Salidas externas:**

Para todas las salidas externas que referencian a cuatro archivos, si consta de cinco o más tipos de datos distintos, la Entrada externa será considerada como complejidad Alta, si consta de menos de cinco tipos de datos distintos será considerada complejidad Media, y para cualquier entrada externa que referencia a menos de cuatro archivos será considerada de complejidad baja.

- **Consultas externas:**

Como cada consulta externa tiene un componente de entrada y otro de salida, se aplican los criterios de entradas externas a la entrada y los criterios de salida a la salida, en caso de contar con complejidades distintas para cada componente, solo se considera la complejidad alta.

- **Archivos lógicos internos y archivos externos de interfaz:**

Si la mayoría de los archivos tienen un solo registro, si consta de cincuenta o más tipos de datos distintos, la Entrada externa será considerada como complejidad Alta, si consta de menos de cincuenta tipos de datos distintos será considerada complejidad baja.

ILF y EIF				EO y EQ				EI			
No. De tipos de registro	Tipos de Datos Distintos			No. De archivos de referencia	Tipos de Datos Distintos			No. De archivos de referencia	Tipos de Datos Distintos		
	1-19	20-50	+51		1-5	6-19	+20		1-4	5-15	+16
1	Baja	Baja	Media	0-1	Baja	Baja	Media	0-1	Baja	Baja	Media
2-5	Baja	Media	Alta	2-3	Baja	Media	Alta	2-3	Baja	Media	Alta
+6	Media	Alta	Alta	+4	Media	Alta	Alta	+3	Media	Alta	Alta

Obteniendo la suma de los Puntos Funcionales Sin Ajustar; 208, se multiplica por el Factor de Ajuste (FAV); 1.09, con lo cual podremos obtener los Puntos de Función Ajustados; 208:

<b>Puntos Funcionales sin Ajustar (PFsA)</b>	208
<b>Factor de Ajuste (VAF)</b>	1.09
<b>No. Total de Puntos Funcionales Ajustados (PFA)</b>	226.72

**<Viabilidad del proyecto>**

la app es para un servicio de pickup que ayudará al usuario a realizar un pedido y pagarlo para poder pasar a recoger el producto posteriormente de manera más rápida y efectiva que si hubiera tenido que hacer fila en el establecimiento del usuario que implemente el uso de la app. Este proyecto promete tener un alcance para más de un restaurante pudiendo brindarle el mismo servicio a varios restaurantes a la vez. Sin embargo el alcance que se pretende como primera implementación es un solo local que se planea sea la cafetería Aihnoa, el cual se encuentra ubicado en Av Circunvalación 7, Jardines de Querétaro, 76020 Santiago de Querétaro, Qro.

Para realizar esta aplicación tenemos que tener en cuenta varios aspectos siendo uno de estos los recursos que requerimos para realizar esta aplicación siendo estos muy sencillos ya que solamente necesitamos internet, una computadora, un servidor en la nube o servicio del mismo y una licencia para subir la app a la play store o app store. Siendo el más complicado de conseguir el último mencionado, con un costo aproximado de 30 dólares para la play store y una licencia básica de app store con un costo de 99 dólares aproximadamente (dándonos un costo total de desarrollo de 130 dólares).

Como se mencionó anteriormente para realizar esta aplicación no se requieren muchos recursos sin embargo desde el punto de vista económico tenemos que tener en cuenta el caso de hacer la adquisición de una licencia para la appstore o play store, para subir ahí la aplicación, sin embargo, podemos optar por utilizar una página web para la descarga de la app tanto del usuario como del cliente. Una vez vistos estos puntos al ser esta una aplicación y realmente no requerir muchos recursos físicos, para su desarrollo tenemos que tener en cuenta que la mayor parte de los recursos que se requieren serían tiempo de desarrollo horas-hombre y conocimiento por parte del equipo de desarrolladores para llevar de forma exitosa el desarrollo. Haciendo un cálculo aproximado de las horas-hombre que se necesitan tenemos un resultado de 3.31 meses de trabajo tomando en cuenta el siguiente cálculo:

Calcular el Esfuerzo (Meses) por el modelo de Estimación Indicativa o "Ball-Park"				Esfuerzo
$Esfuerzo = \left( \frac{Tamaño\ en\ PF}{150} \right) * Tamaño\ en\ PF$	1.511 46666 7	8.753786 156	13.23	Trabajo
Personas por equipo de Desarrollo (Equipo)	4		3.31	Mes



Actualmente en el mercado de este tipo de aplicaciones existen algunas apps líderes en el área siendo estas UberEats, Rappi y otras. Sin embargo, nosotros a diferencia de estas apps que tienen servicio a domicilio haciendo que el costo de los productos se eleve de forma notable, manejamos la opción de que el usuario recoja él mismo el producto, logrando de igual forma que el usuario no tenga que hacer fila, agilizando el proceso de entrega y siendo una opción más barata en cuanto a comisiones a diferencia de las otras aplicaciones que aparte de cobrar el envío, también tienen comisiones más altas que las nuestras.

### **<Requerimientos>**

Requerimientos del Usuario: Al abrir una cuenta en la plataforma, el sistema debe dar acceso al menú del restaurante, poder añadir productos al carrito, y finalmente comprarlos, el sistema debe ofrecer al usuario la función de elegir el método de pago preferido, electrónico o por contra entrega, el sistema debe ofrecer la opción de escoger en cual sucursal ordenar y recoger su pedido.

- Dar acceso al menú que se ofrece del establecimiento.
- Posibilitar la instalación de la plataforma en la mayoría de dispositivos móviles.
- Sugerir bebidas.
- Mostrar el mapa con la ubicación de los establecimientos.

Requerimientos del Locatario: Al abrir una cuenta en la plataforma, el sistema debe dar acceso a determinadas funciones.

- Agregar menú: permite añadir el menú de su local, así como el nombre, costo, detalles y especificaciones del producto.
- Método de pago: debe especificar los pagos aceptados, efectivo en contra entrega o electrónico, si es este último debe proporcionar al sistema los datos bancarios para sus pagos.
- Ubicación: debe colocar la ubicación de su establecimiento, de contar con diferentes puntos, deberá especificar en cuales es válido el servicio de pickup.

### **Requerimientos Funcionales:**

Requerimientos del Sistema:

1. Mostrar mapa.
2. Mostrar ubicación de las sucursales.
3. Mostrar pedidos.
4. Mostrar precios.
5. Permitir realizar pedidos.
6. Permitir ver comida.
7. Permitir ver bebidas.

8. Sugerir una bebida con la comida.
9. Configurar cuenta.
10. Registrar usuarios.
11. Registrar nombres de usuario.
12. Registrar correos electrónicos.
13. Registrar números telefónicos.
14. Registrar establecimientos.
15. Mostrar formulario de registro.
16. Permitir compra.
17. Permitir agregar al carrito.
18. Mostrar carrito.
19. Mostrar opciones de pago.
20. Mostrar monto a pagar
21. Mostrar ubicación del establecimiento.
22. Confirmar pago del usuario.
23. Realizar pago al establecimiento.
24. Recibir pago del usuario.
25. Cobrar comisión para el equipo.
26. Permitir aceptar o rechazar pedidos.
27. Mostrar confirmación de pedidos.
28. Mostrar confirmación de órdenes.
29. Notificar al usuario de orden lista.

### **Requerimientos no Funcionales:**

- **Disponibilidad:** Capacidad del software de ser accesible y utilizable por el sistema o los usuarios, autorizados cuando esto se requiera. En este sentido, la aplicación debe estar disponible y activa en cualquier momento que un usuario desee ordenar en alguno de los establecimientos abiertos, no puede fallar, debido a que es necesaria para realizar los pedidos.
- **Usabilidad:** Capacidad del software de ser entendido, aprendido, y usado en forma fácil y atractiva. En este sentido, la aplicación debe poder contar con una interfaz clara, la cual tenga inputs y outputs destacables, que si tengan impacto en la experiencia de pedidos del cliente, que no lo dejen estancado sin retroalimentación y permitan un flujo natural a la hora de ordenar.
- **Modificabilidad:** Capacidad del software de aceptar y realizar de manera satisfactoria una modificación, sin desestabilizar al sistema, incluye codificación, diseño y documentación. En este sentido, la aplicación debe permitir agregar nuevos usuarios y establecimientos, sin que los demás dejen de verse o funcionar.
- **Portabilidad:** Capacidad que tiene el software para ser trasladado de un entorno a otro. En este sentido, la aplicación debe ser capaz de ejecutarse en múltiples modelos de celulares y de sistemas operativos (Android y iOS),

Además de ser funcional en múltiples versiones de estos sistemas, para la gran cantidad de usuarios con diferentes dispositivos que existen.

- **Seguridad:** Capacidad del software de no tener altos niveles de riesgo para causar daños a las personas, instituciones, software o entorno. A estos riesgos se les conoce como deficiencias en la funcionalidad. En este sentido, la aplicación al manejar datos sensibles y financieros, debe ser segura, sin permitir que existan brechas de seguridad por donde un agente malicioso desee robar información, o generar un disgusto a algún usuario y/o establecimiento.

### **Funcionalidad y servicios ofrecidos**

Con el desarrollo de esta aplicación se ofrece los siguientes funcionalidades:

- **Locatario:**
  - Abrir una cuenta de usuario como locatario en el sistema.
  - Subir sus productos al sistema, requiriendo los siguientes datos: nombre del artículo, precio del artículo, descripción del artículo.
  - Ofrecer sus productos en una plataforma virtual de “pick-up” (recoger), donde un cliente solicita un pedido de sus productos y el cliente acude a la sucursal para recogerlo.
  - Recibir método de pago electrónico por cada pedido realizado en la plataforma.
- **Cliente:**
  - Abrir una cuenta de usuario como cliente en el sistema.
  - Visualizar, seleccionar y comprar los productos ofrecidos en la plataforma.
  - Cálculo de tiempo estimado para recoger el pedido.
  - Método de pago electrónico.

### **Análisis de las tecnologías**

**Figma:** Editor de gráfico vectorial y herramienta de generación de prototipos basados en la web, diseñar interfaces.

**Lucidchart:** Herramienta de diagrama basado en la web, permite diseñar diagramas de flujo, organigramas, esquemas de sitios web, diseños UML, mapas mentales.

**Azure Cloud:** Plataforma en la nube desarrollada por Microsoft, permite administrar y desarrollar aplicaciones y servicios por medio de centros de datos.

**Expo:** Plataforma que permite desarrollar aplicaciones nativas para Android, iOS y la web, usando JavaScript y React Native.





**MySQL:** MySQL es, como su nombre indica, un sistema de gestión de bases de datos relacionales o SGBD basado en SQL. En la actualidad, este software de código abierto forma parte de Oracle, la empresa que también desarrolló el lenguaje de programación Java. MySQL almacena, gestiona y muestra datos en tablas. Funciona como un sistema cliente-servidor.

**React Native:** React Native es un framework JavaScript para crear aplicaciones reales nativas para iOS y Android, basado en la librería de JavaScript React para la creación de componentes visuales, cambiando el propósito de los mismos para, en lugar de ser ejecutados en navegador, correr directamente sobre las plataformas móviles nativas.

**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.

**JavaScript:** JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D.

### **Antecedentes:**

#### **- Starbucks Coffe App:**

Aplicación de la cadena Starbucks que permite realizar un pedido en el establecimiento más cercano. Limitantes:

- Falta de actualizaciones.
- Falta de solución de errores.
- Prácticas anticompetitivas.
- Falta de servicios al cliente.



#### **- Uber Eats:**

Plataforma digital, que permite realizar un pedido a un restaurante y la posibilidad de recoger el artículo en el establecimiento o entrega a domicilio. Limitantes:

- Falta de servicio al cliente.
- Falta de actualizaciones.
- Altos cobros a los comerciantes.
- Irregularidades con las entregas.





## Software base del sistema y prerequisites

- Dispositivo con Android:

### Componentes y estándares:

#### Diseño:

- Lucidchart.
- Figma.

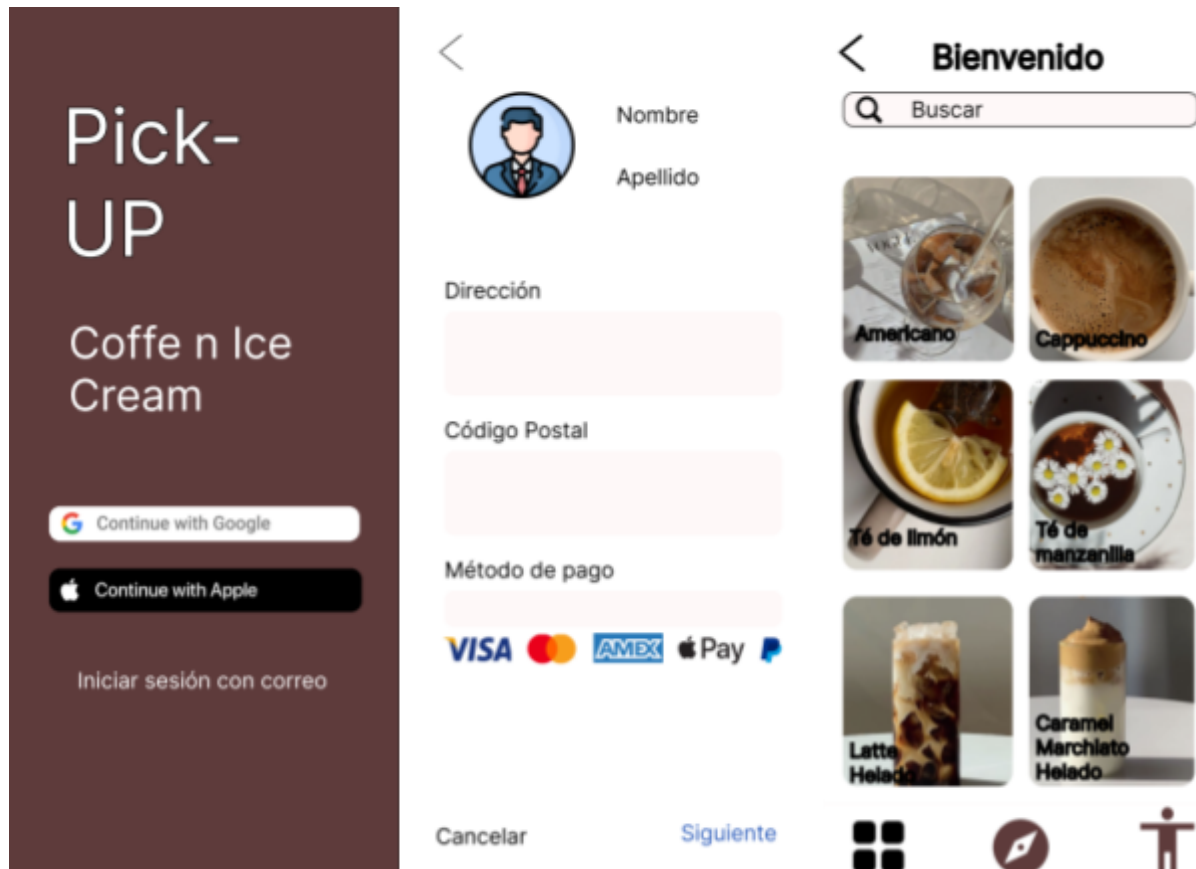
#### Desarrollo:

- Lenguaje de programación: JavaScript.
- Intercambio de datos: Json.
- Expo: Framework de desarrollo para crear experiencias con gestos y gráficos interactivos, usando JavaScript y React Native.

### Modelo:

Los modelos pueden ser encontrados en el siguiente link (Figma):

- [https://www.figma.com/file/3wDzGwW2ZD1XakBCIPbf8F/Devs\\_lutions?node-id=0%3A1&t=Xj8WG7s9AnbBMx1v-1](https://www.figma.com/file/3wDzGwW2ZD1XakBCIPbf8F/Devs_lutions?node-id=0%3A1&t=Xj8WG7s9AnbBMx1v-1)





< Artículo





**Latte Helado** \$45.00

Café espresso mezclado con leche y hielos.

[Agregar al carrito](#)

▼



< Carrito

<b>Latte Helado</b>	\$ 45.00
Café espresso mezclado con leche y hielos.	
<b>Cappuccino</b>	\$ 45.00
Espresso bajo una capa espumosa leche vaporizada.	

Elegir tienda →

Añadir mas productos +

Cancelar

< Carrito

Articulos

<b>Latte Helado</b>	\$ 45.00
Café espresso mezclado con leche y hielos.	
<b>Cappuccino</b>	\$ 45.00
Espresso bajo una capa espumosa leche vaporizada.	

Tienda

**Aihnoa Cafetería**


Av Circunvalación 7, Jardines de Querétaro, 76020 Santiago de Querétaro, Qro.

Total a pagar \$90.00

Cancelar [Pagar](#)

< Métodos de Pago

Apple Pay Tarjeta de Crédito/Debito

 Paypal

Efectivo (Contra Entrega)

Cancelar [Pagar](#)

< Orden # 9568

	\$ 45.00
	\$ 45.00
<b>Metodo de Pago:</b>	<b>Apple Pay</b>
<b>Total:</b>	<b>\$90.00</b>


Ubicación de recogida

**Aihnoa Cafetería**

Av Circunvalación 7, Jardines de Querétaro, 76020 Santiago de Querétaro, Qro.

Tiempo para pedido listo: [Aceptar](#) 10 min.

< Buscar



Ubicaciones

**Aihnoa Cafetería**

Av Circunvalación 7, Jardines de Querétaro, 76020 Santiago de Querétaro, Qro. [Elegir](#)



## < Ubicaciones

### Aihnoa Cafetería

Av Circunvalación 7, Jardines  
de Querétaro, 76020 Santiago  
de Querétaro, Qro.

[Elegir](#)

Hola Maria



Maria  
Ramirez



Cuenta



Método de pago



Terminos y  
Condiciones



Preguntas  
frecuentes

Direcciones

Casa

Trabajo



## Métodos de Pago

Opciones:

 VISA

...7822



...1241

Editar

[Agregar](#)

Añadir Nombre...

Añadir Detalles...

Añadir precio...

Seleccionar tiendas disponibles ▼

Cancelar Añadir

Nuevo (1)	
Willian 12:40	Aceptado
En progreso (2)	
Ana 465CD	10 min
Carlos 123AB	10 min
Finalizados ▼	

<

## Historial de Orden

Ayer

Karen (465CD)	
Completo	\$200

---

Oscar (123AB)	
Completo	\$200

...



<

## Comentarios

Karen (465CD)  
#Ordenes

★★★★★

Buenos cafes y buen precio.

Ultima Orden Total \$

---

Roberto (235CD)  
#Ordenes

★★★★★

Rápido para cuando uno no tiene tiempo.



Diagrama secuencia cliente-realizar orden (Lucidchart):

- [https://lucid.app/lucidchart/c245037b-5e6f-4472-a99c-9a4c7905abe8/edit?viewport\\_loc=-315%2C84%2C3025%2C1456%2CFDQc8L-t-qg&invitationId=inv\\_9545d8b8-7ce7-4b53-a841-1c048aa46d8f](https://lucid.app/lucidchart/c245037b-5e6f-4472-a99c-9a4c7905abe8/edit?viewport_loc=-315%2C84%2C3025%2C1456%2CFDQc8L-t-qg&invitationId=inv_9545d8b8-7ce7-4b53-a841-1c048aa46d8f)

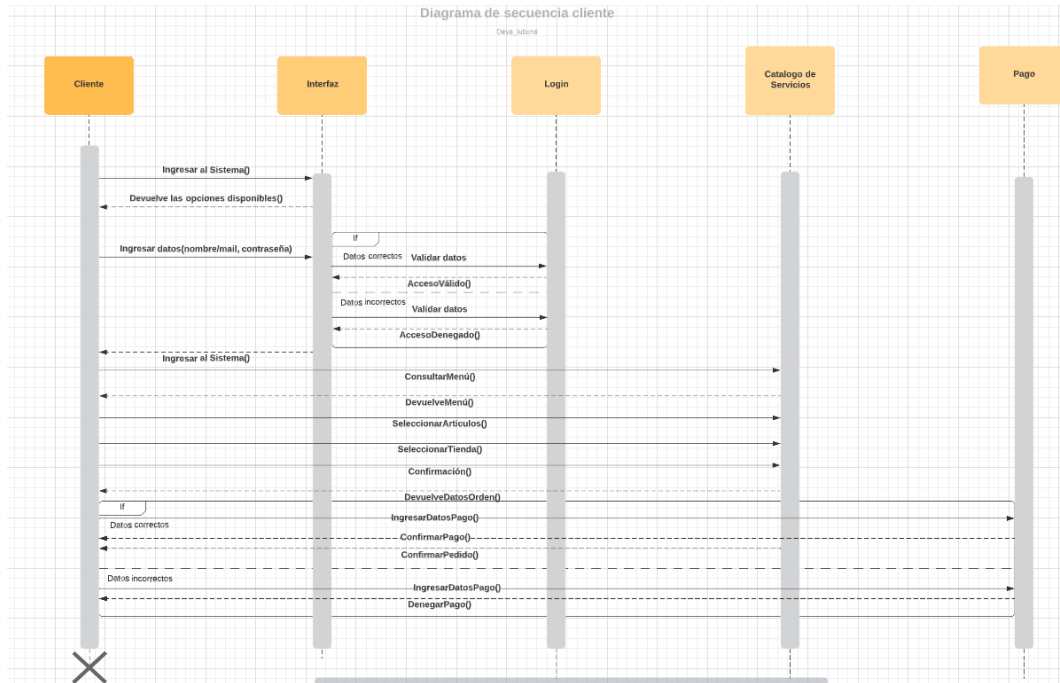
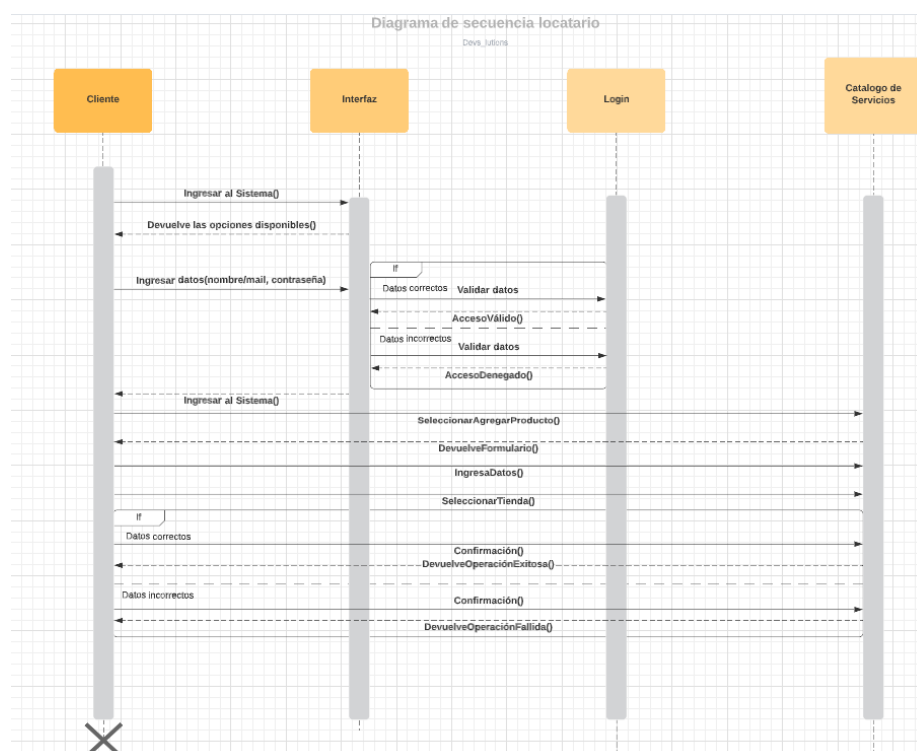


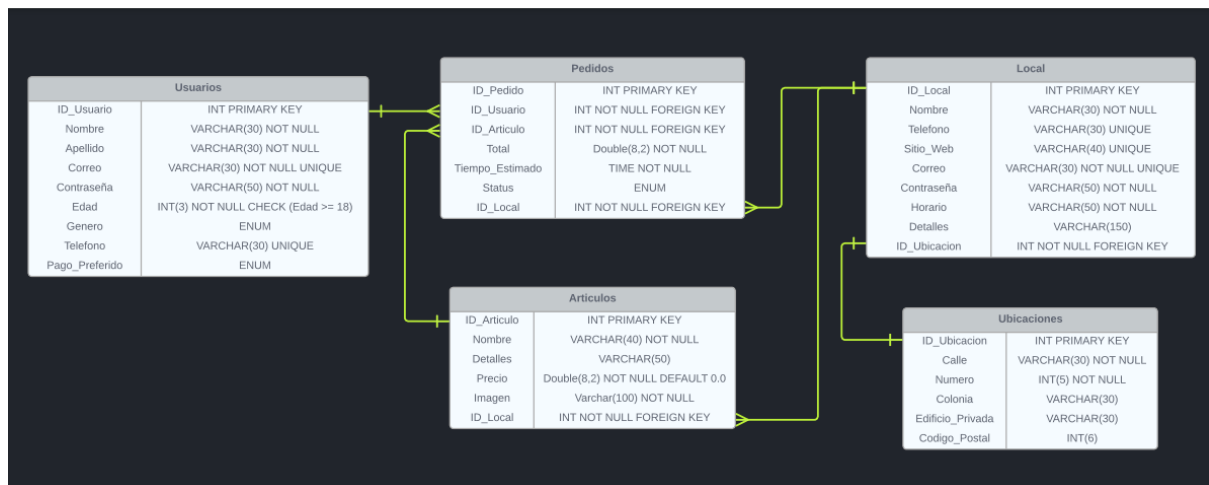
Diagrama de secuencia locatario, subir artículo:

- [https://lucid.app/lucidchart/56ca07b7-6f37-4be0-a76c-1a1eab4e5d12/edit?viewport\\_loc=-397%2C90%2C3150%2C1516%2CFDQc8L-t-qg&invitationId=inv\\_8b9ab87d-a8fa-4f7f-bb88-38393a364f0a](https://lucid.app/lucidchart/56ca07b7-6f37-4be0-a76c-1a1eab4e5d12/edit?viewport_loc=-397%2C90%2C3150%2C1516%2CFDQc8L-t-qg&invitationId=inv_8b9ab87d-a8fa-4f7f-bb88-38393a364f0a)



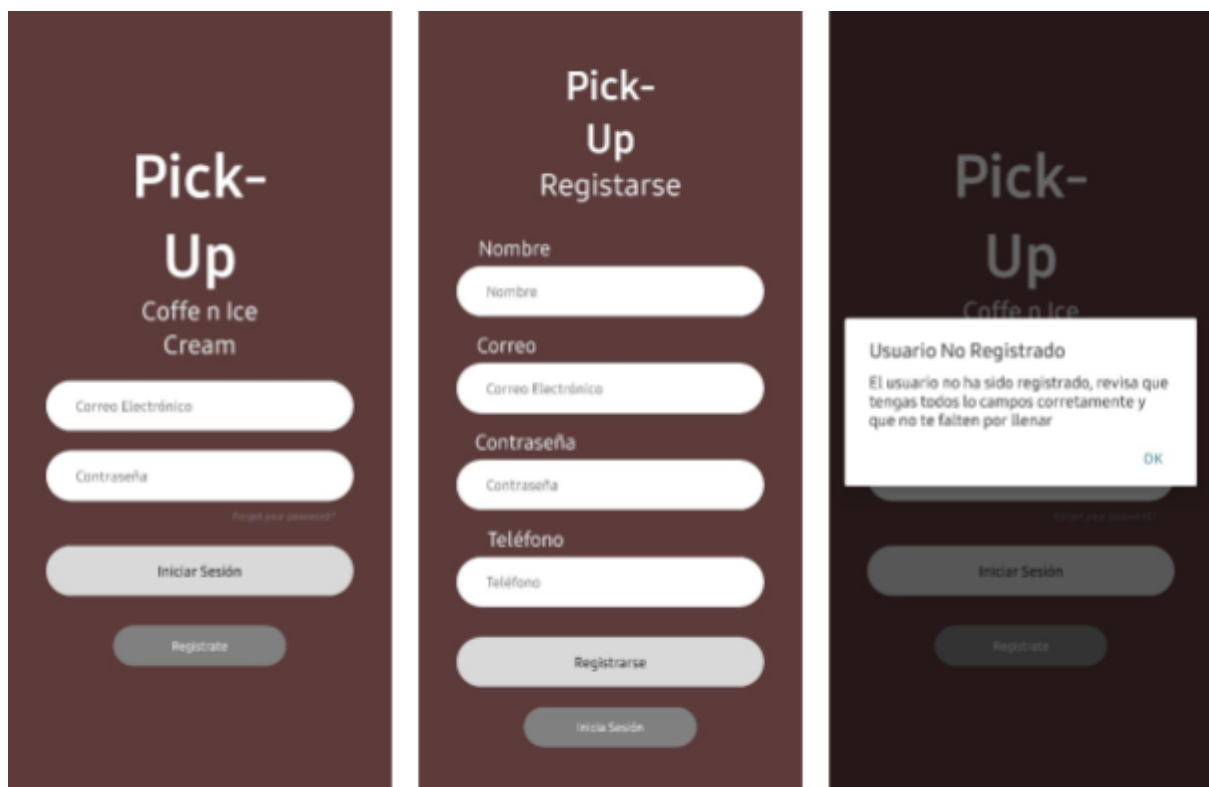


## Diagrama entidad relación de la base de datos:

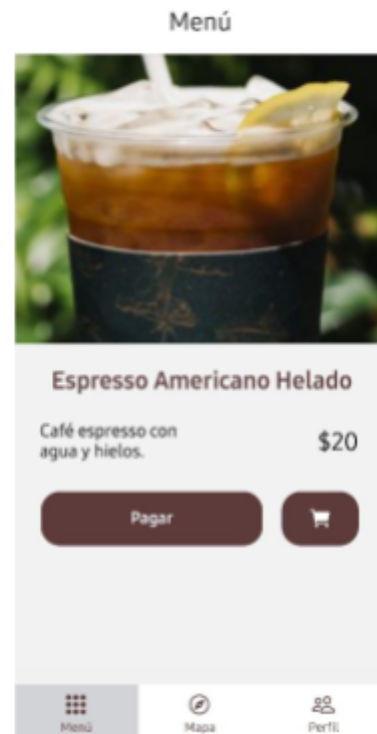
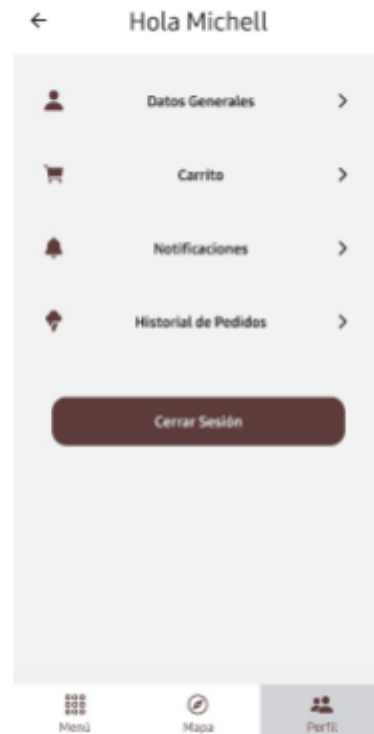
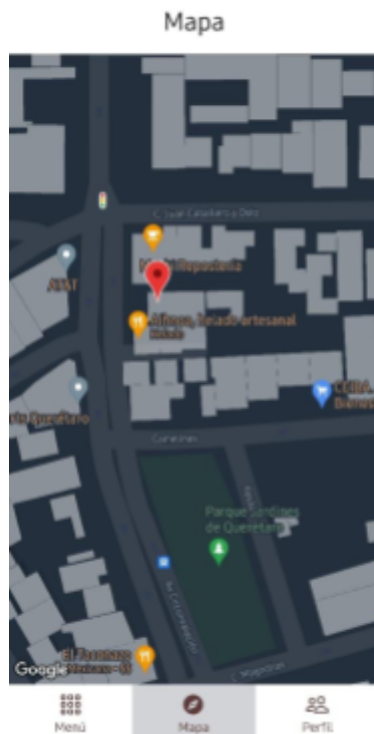
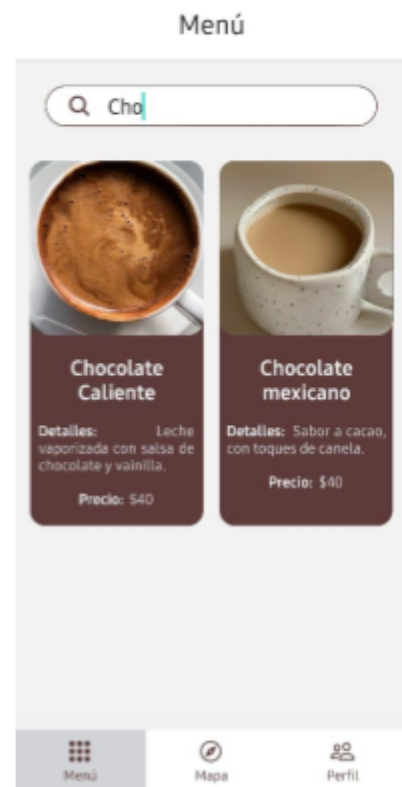
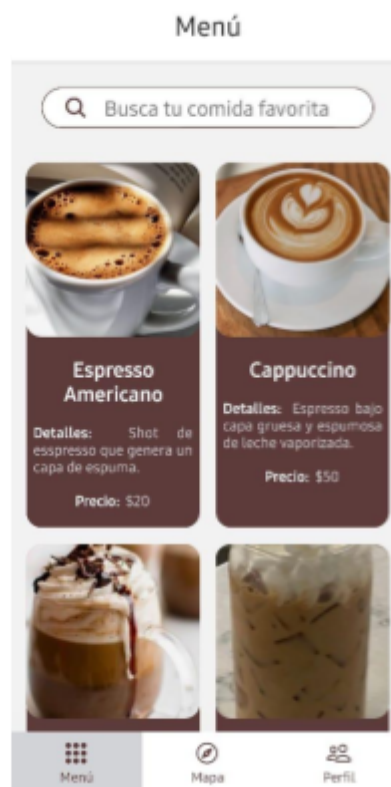
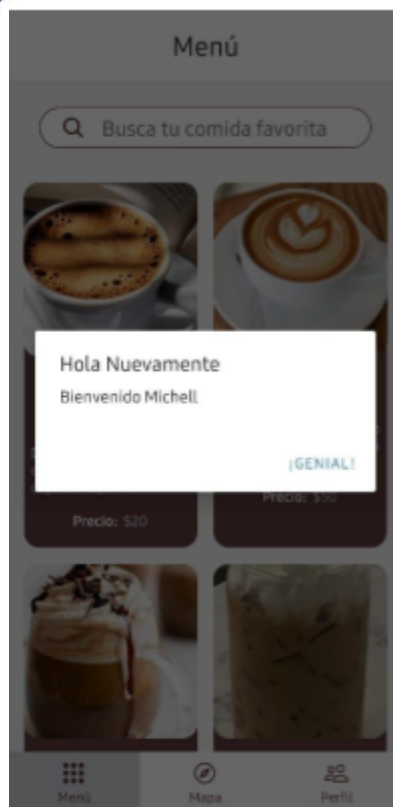


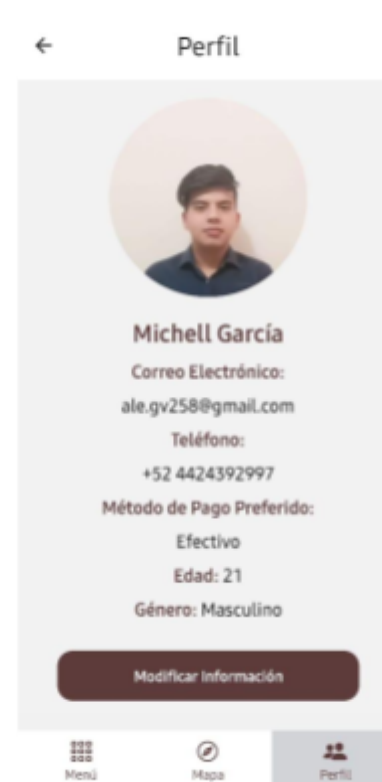
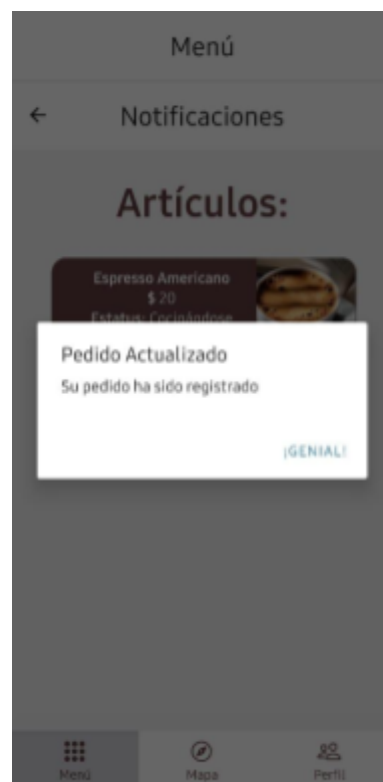
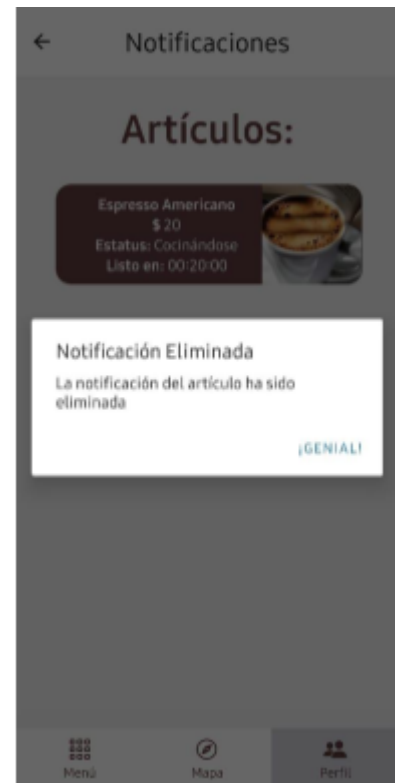
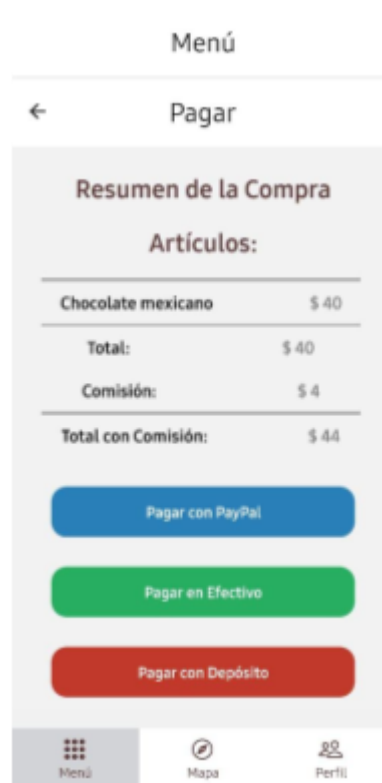
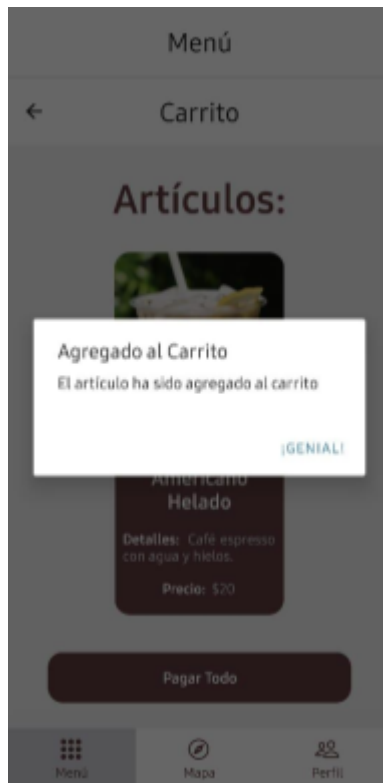
## <Resultados>

### Diseño:

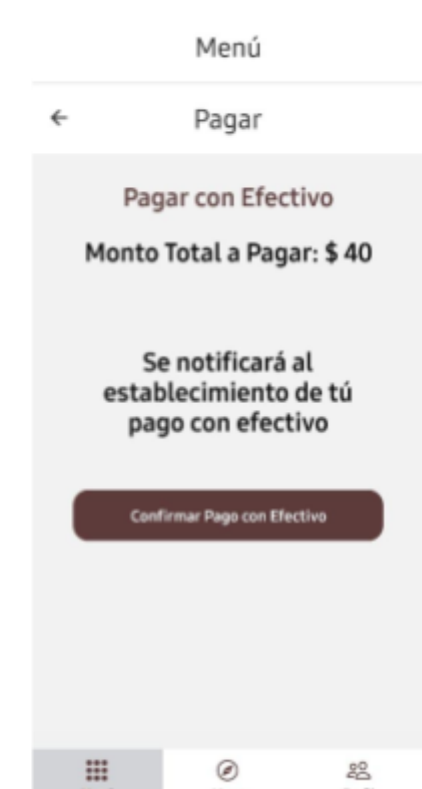
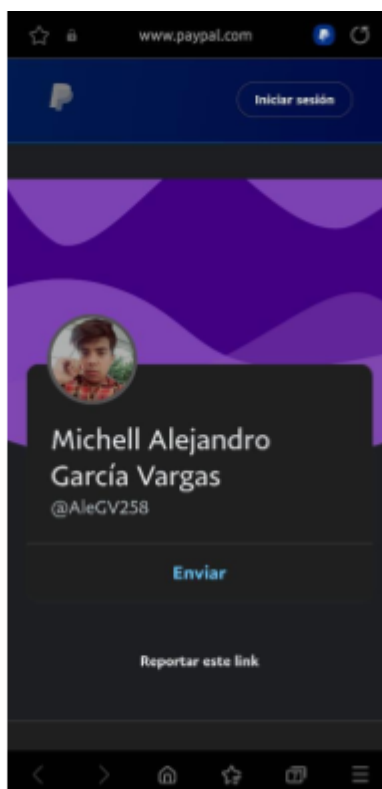
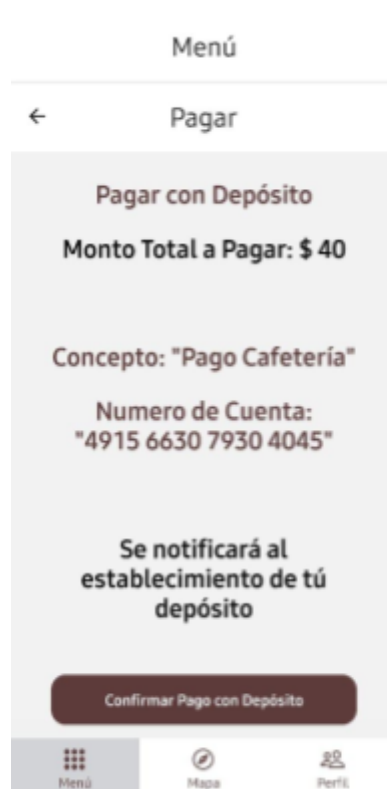
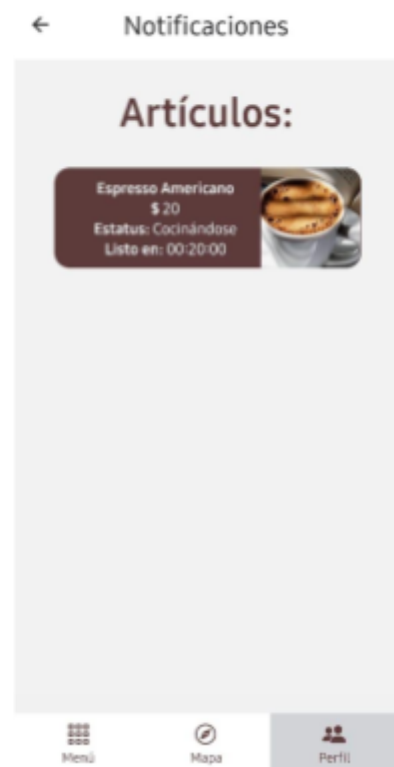
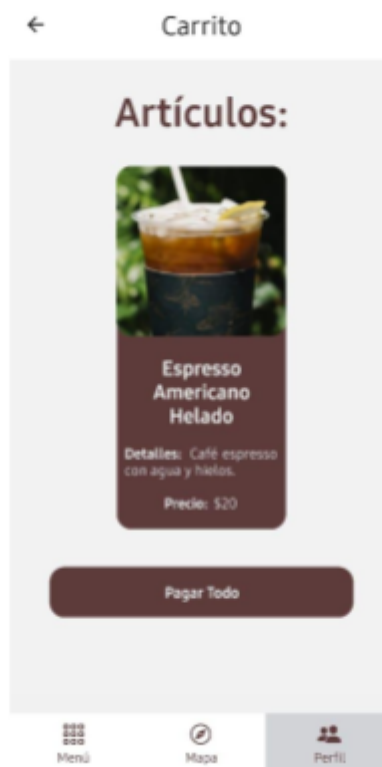














### Código:

El código completo puede ser encontrado en:

- [https://drive.google.com/drive/folders/1qEn6wfAtTYoydEbtbujfqmJCn-s\\_VFz?usp=sharing](https://drive.google.com/drive/folders/1qEn6wfAtTYoydEbtbujfqmJCn-s_VFz?usp=sharing)
- [https://github.com/AleGV258/Devs\\_lutions](https://github.com/AleGV258/Devs_lutions)

Y el código fuente es el siguiente:

- **App.js**

```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, View, SafeAreaView, Platform } from
'react-native';
import LoginNavigation from './src/routes/LoginNavigation';

export default function App() {
  return (
    <SafeAreaView style={styles.area}>
      <StatusBar barStyle="dark-content" backgroundColor="#5E3B3B" />
      <View style={styles.container}>
        <LoginNavigation/>
      </View>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
  },
  area: {
    flex: 1,
    paddingTop: Platform.OS === 'android' ? 25 : 0,
  },
});
```

- **component/Buscador.js**

```
import React, { useState } from 'react';
import { View, TextInput } from 'react-native';
import { Icon } from 'react-native-elements';
import { useTheme } from '@react-navigation/native';
import GlobalStyles from '../routes/GlobalStyles';

const Buscador = ({texto, setTexto}) => {
  const { colors } = useTheme();
  return (
    <View>
      <View style={GlobalStyles.containerBuscador} >
        <Icon name='search' type='ionicon' color='#5E3B3B'
style={GlobalStyles.icono} />
      </View>
    </View>
  );
}
```



```
        <TextInput placeholder='Busca tu comida favorita'
style={GlobalStyles.buscador} value={texto.búsqueda}
onChangeText={newText => setTexto({...texto, búsqueda: newText})}/>
      </View>
    </View>
  );
}

export default Buscador;
```

- component/FoodCards.js

```
import React from 'react';
import { useEffect, useState } from 'react';
import { View, Text, Image, Pressable } from 'react-native';
import GlobalStyles from '../routes/GlobalStyles';
import { useNavigation } from '@react-navigation/native';

const FoodCards = ( textoBuscado ) => {
  try {
    const [data, setData] = useState([]);
    const navigation = useNavigation();

    const fetchData = async () => {
      const resp = await
fetch("https://devs-lutions-api.azurewebsites.net/articulos");
      const data = await resp.json();
      setData(data);
    };

    useEffect(() => {
      fetchData()
    }, [])

    const busquedaData = data.filter( busqueda => {

if ((busqueda.Nombre).toUpperCase().includes(textoBuscado.textoBuscar.bu
squeda.toUpperCase())) {
      return true
    }else{
      return false
    }
  })

    const newData = busquedaData.map( datos => {
      return (
        <Pressable style={GlobalStyles.foodCard}
key={datos.ID_Articulo} onPress={() => navigation.navigate('Artículo',
{articulo: datos.ID_Articulo})}>
          <Image
            source={{uri: datos.Imagen}}
            style={GlobalStyles.foodImage}
          />
          <Text style={GlobalStyles.nombreFood}>{datos.Nombre}</Text>
          <Text style={GlobalStyles.txtFood}><Text
style={GlobalStyles.descPrecFood}>Detalles:
</Text><Text>{datos.Detalles}</Text></Text>

```



```
        <Text style={GlobalStyles.txtFood, {marginBottom:
10,}}><Text style={GlobalStyles.descPrecFood}>Precio:
</Text><Text>${datos.Precio}</Text></Text>
        </Pressable>
    )
  })

  return (
    <View style={GlobalStyles.food}>
      {newData}
    </View>
  );
} catch(error) {
  console.log(error);
}
}

export default FoodCards;
```

- component/Navbar.js

```
import * as React from 'react';
import 'react-native-gesture-handler';
import { useEffect, useState } from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { Icon } from 'react-native-elements';
import { createBottomTabNavigator } from
 '@react-navigation/bottom-tabs';

// Screens
import ProfileNavigation from '../routes/ProfileNavigation';
import ArticleNavigation from '../routes/ArticleNavigation';
//import ProfilePage from '../views/ProfilePage';
import MapPage from '../views/MapPage';

// DEFAULT LIGHT THEME
const LightTheme = {
  dark: false,
  colors: {
    notification: 'rgb(255, 59, 48)',
    primary: 'rgb(0, 122, 255)',
    aplicacion: 'rgb(94, 59, 59)',
    card: 'rgb(255, 255, 255)',
    background: 'rgb(250, 250, 250)',

    opuesto: '#000',
    borde: 'rgb(216, 216, 216)',
    text: 'rgb(28, 28, 30)',
    primero: '#d1d3d6',
    segundo: '#5f666f',
  },
};

// DEFAULT DARK THEME
const DarkTheme = {
  dark: true,
  colors: {
    notification: 'rgb(255, 59, 48)',
    primary: 'rgb(0, 122, 255)',
```



```
aplicacion: 'rgb(94, 59, 59)',
card: '#1b2431',
background: '#323a46',

opuesto: '#fff',
borde: '#f7f1e3',
text: 'rgb(229, 229, 231)',
primero: '#49505a',
segundo: '#bbbec1',
},
};

const Tab = createBottomTabNavigator();

function Navbar({navigation}) {
  const scheme = 'light';
  return (
    <NavigationContainer
      independent={true}
      theme={scheme === 'dark' ? DarkTheme : LightTheme}>
      <Tab.Navigator
        initialRouteName="Menú"
        screenOptions={({ route }) => ({
          tabBarActiveBackgroundColor: scheme === 'dark' ? '#49505a' :
'#d1d3d6',
          tabBarActiveTintColor: '#5E3B3B',
          tabBarIconStyle: { marginTop: 7 },
          tabBarLabelStyle: {
            fontSize: 13,
            color: '#5E3B3B',
            paddingBottom: 7,
          },
        }),
        tabBarStyle: {
          elevation: 100,
          height: 60,
          position: 'absolute',
          bottom: 0,
          left: 0,
          right: 0,
          zIndex: 4,
          borderTopWidth: 0,
        },
        headerShown: true,
        headerTitleAlign: 'center',
        headerBackButtonShown: true,
        headerTitleAllowFontScaling: true,
        unmountOnBlur: true,
        tabBarIcon: ({ focused, color, size }) => {
          let iconName;
          let rn = route.name;
          color = '#5E3B3B';

          if (rn === "Menú") {
            iconName = focused ? 'apps' : 'apps-outline';
          } else if (rn === "Mapa") {
            iconName = focused ? 'compass' : 'compass-outline';
          } else if (rn === "Perfil") {
            iconName = focused ? 'people' : 'people-outline';
          }
        }
      >
    </Tab.Navigator>
  );
}
```



```
    }

    // You can return any component that you like here!
    return <Icon name={iconName} type="ionicon" size={size}
color={color} />;
  },
  )}>
<Tab.Screen
  name="Menú"
  component={ArticleNavigation}
  options={{
    headerStyle: { height: 70, elevation: 3 },
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
  }}
/>
<Tab.Screen
  name="Mapa"
  component={MapPage}
  options={{
    headerStyle: { height: 70, elevation: 3 },
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
  }}
/>
<Tab.Screen
  name="Perfil"
  component={ProfileNavigation}
  options={{
    headerStyle: { height: 70, elevation: 3 },
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
    headerShown: false,
  }}
/>
</Tab.Navigator>
</NavigationContainer>
);
}

export default Navbar;
```

- component/ProfileCards.js

```
import * as React from 'react';
import { useState, useEffect } from 'react';
import { useNavigation } from '@react-navigation/native';
import { Text, View, Pressable, useColorScheme } from 'react-native';
import { Icon } from 'react-native-elements';
import GlobalStyles from '../routes/GlobalStyles';

function ProfileCards({navigation}) {
  const [isEnabled, setIsEnabled] = useState(false);
  const toggleSwitch = () => setIsEnabled((previousState) =>
!previousState);
  const [scheme, setScheme] = useState(useColorScheme());
  return (
    <View>
```



```
<Pressable
  style={GlobalStyles.option}
  android_ripple={{ color: '#bdc3c7' }}
  onPress={() => navigation.navigate('UserPage')}}>
  <Icon name="person" type="ionicon" color="#5E3B3B" />
  <Text style={GlobalStyles.texto}>Datos Generales</Text>
  <Icon
    name="chevron-right"
    type="feather"
    color="#000"
    style={GlobalStyles.iconoFinal}
  />
</Pressable>
<Pressable
  style={GlobalStyles.option}
  android_ripple={{ color: '#bdc3c7' }}
  onPress={() => navigation.navigate('ShoppingCartPage')}}>
  <Icon name="cart" type="ionicon" color="#5E3B3B" />
  <Text style={GlobalStyles.texto}>Carrito</Text>
  <Icon
    name="chevron-right"
    type="feather"
    color="#000"
    style={GlobalStyles.iconoFinal}
  />
</Pressable>
<Pressable
  style={GlobalStyles.option}
  android_ripple={{ color: '#bdc3c7' }}
  onPress={() => navigation.navigate('NotificationPage')}}>
  <Icon name="notifications" type="ionicon" color="#5E3B3B" />
  <Text style={GlobalStyles.texto}>Notificaciones</Text>
  <Icon
    name="chevron-right"
    type="feather"
    color="#000"
    style={GlobalStyles.iconoFinal}
  />
</Pressable>
<Pressable
  style={GlobalStyles.option}
  android_ripple={{ color: '#bdc3c7' }}
  onPress={() => navigation.navigate('OrderHistoryPage')}}>
  <Icon name="ice-cream" type="ionicon" color="#5E3B3B" />
  <Text style={GlobalStyles.texto}>Historial de Pedidos</Text>
  <Icon
    name="chevron-right"
    type="feather"
    color="#000"
    style={GlobalStyles.iconoFinal}
  />
</Pressable>
<Pressable
  style={[GlobalStyles.option, {backgroundColor: '#5E3B3B',
borderRadius: 15, marginHorizontal: 40, marginTop: 50}]}
  android_ripple={{ color: '#bdc3c7' }}
  onPress={() => navigation.navigate('LoginNavigation')}}>
```





```
        <Text style={[GlobalStyles.texto, {textAlign: 'center', width: 240, color: '#fff'}]}>Cerrar Sesión</Text>
      </Pressable>
    </View>
  );
}

export default ProfileCards;
```

- component/ShopCards.js

```
import React from 'react';
import { View, Text } from 'react-native';
import GlobalStyles from '../routes/GlobalStyles';

const ShopCards = ({product}) => {
  return (
    <View style={GlobalStyles.box}>
      <View style={GlobalStyles.boxText}>
        <Text style={GlobalStyles.textTittle}>{product.name}</Text>
        <Text>{product.description}</Text>
      </View>
      <View style={GlobalStyles.boxPrice}>
        <Text>${product.price}</Text>
      </View>
    </View>
  );
}

export default ShopCards;
```

- component/TopBar.js

```
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

class TopBar extends React.Component {
  render() {
    return (
      <View style={styles.container}>
        <Text style={styles.texto}>Pick Up</Text>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    alignSelf: 'stretch',
    height: 60,
    flexDirection: 'row',
    backgroundColor: 'red',
    alignItems: 'center',
    justifyContent: 'center',
  },
  texto: {
    color: 'white',
    fontSize: 26,
  }
});
```





```
    fontFamily: 'sans-serif-medium',  
  },  
});  
  
export default TopBar;
```

- routes/ArticleNavigation.js

```
import React from 'react'  
import { createStackNavigator } from '@react-navigation/stack'  
import { NavigationContainer } from '@react-navigation/native';  
import { useTheme } from '@react-navigation/native';  
  
import DetailsFood from '../views/DetailsFood'  
import HomePage from '../views/HomePage';  
import ShoppingCartPage from '../views/ShoppingCartPage';  
import NotificationPage from '../views/NotificationPage';  
import PayPage from '../views/PayPage';  
import DepositPage from '../views/DepositPage';  
  
const StackArticle = createStackNavigator()  
  
const ArticleNavigation = ({scheme}) => {  
  const { colors } = useTheme();  
  return(  
    <NavigationContainer independent={true}>  
      <StackArticle.Navigator initialRouteName="Menú">  
        <StackArticle.Screen  
          name="Menú"  
          component={HomePage}  
          options={{  
            headerStyle: { backgroundColor: colors.card,  
height: 70 },  
            headerTintColor: colors.text,  
            headerShown: false,  
            headerTitle: 'Página Principal',  
            headerTitleAlign: 'center',  
            headerTitleStyle: { fontSize: 26 },  
          }}  
        />  
        <StackArticle.Screen  
          name="Artículo"  
          component={DetailsFood}  
          options={({ route }) => ({  
            headerStyle: { backgroundColor: colors.card,  
height: 70 },  
            headerTintColor: colors.text,  
            headerShown: false,  
            headerTitle: 'Artículo',  
            headerTitleAlign: 'center',  
            headerTitleStyle: { fontSize: 26 },  
          })}  
        />  
        <StackArticle.Screen  
          name="NotificationPage"  
          component={NotificationPage}  
          options={{
```



```
height: 70 },
      headerStyle: { backgroundColor: colors.card,
        headerTintColor: colors.text,
        headerTitle: 'Notificaciones',
        headerTitleAlign: 'center',
        headerTitleStyle: { fontSize: 26 },
      }
    }
  />
  <StackArticle.Screen
    name="ShoppingCartPage"
    component={ShoppingCartPage}
    options={{
      headerStyle: { backgroundColor: colors.card,
        headerTintColor: colors.text,
        headerTitle: 'Carrito',
        headerTitleAlign: 'center',
        headerTitleStyle: { fontSize: 26 },
      }
    }
  />
  <StackArticle.Screen
    name="PayPage"
    component={PayPage}
    options={{
      headerStyle: { backgroundColor: colors.card,
        headerTintColor: colors.text,
        headerTitle: 'Pagar',
        headerTitleAlign: 'center',
        headerTitleStyle: { fontSize: 26 },
      }
    }
  />
  <StackArticle.Screen
    name="DepositPage"
    component={DepositPage}
    options={{
      headerStyle: { backgroundColor: colors.card,
        headerTintColor: colors.text,
        headerTitle: 'Pagar',
        headerTitleAlign: 'center',
        headerTitleStyle: { fontSize: 26 },
      }
    }
  />
</StackArticle.Navigator>
</NavigationContainer>
)
}

export default ArticleNavigation;
```

- routes/GlobalStyles.js

```
import React, {
  StyleSheet,
  Dimensions
} from 'react-native';
```



```
const { widthWindow, heightWindow } = Dimensions.get('window');

const GlobalStyles = StyleSheet.create({
  // Login
  containerL: {
    flex: 1,
    backgroundColor: '#5E3B3B',
    alignItems: 'center',
    justifyContent: 'center',
  },
  tituloL: {
    fontSize: 60,
    color: '#FFFFFF',
    fontWeight: 'bold',
  },
  subtituloL: {
    fontSize: 25,
    color: '#FFFFFF',
  },
},
  textInputL: {
    padding: 10,
    paddingStart: 30,
    width: '80%',
    height: 50,
    marginTop: 20,
    borderRadius: 30,
    backgroundColor: '#FFFFFF',
  },
  forgotPassL: {
    fontSize: 10,
    color: 'gray',
    marginTop: 8,
    marginBottom: 8,
    marginLeft: '44%',
  },
  textBtnL: {
    fontSize: 14,
  },
  loginBtnL: {
    width: '80%',
    height: 50,
    borderRadius: 25,
    padding: 10,
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: '#D9D9D9',
    marginTop: 15,
  },
  googleBtnL: {
    width: '70%',
    height: 50,
    borderRadius: 25,
    padding: 10,
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: '#FFFFFF',
    marginTop: 20,
  },
});
```



```
},
textBtnAppleL:{
  fontSize: 12,
  color: '#FFFFFF',
},
appleBtnL:{
  width:'45%',
  height: 40,
  borderRadius: 25,
  padding:10,
  alignItems:'center',
  justifyContent:'center',
  backgroundColor: 'blue',
  marginTop:20,
},
textBtnGoogleL:{
  fontSize: 12,
  color: 'white',
},
googlebtnL:{
  width:'45%',
  height: 40,
  borderRadius: 25,
  padding:10,
  alignItems:'center',
  justifyContent:'center',
  backgroundColor: 'grey',
  marginTop:20,
  marginTop:20
},
// SignIn
containerR: {
  flex: 1,
  backgroundColor: '#5E3B3B',
  alignItems: 'center',
  justifyContent: 'center',
},
tituloR: {
  fontSize: 40,
  color:'#FFFFFF' ,
  fontWeight:'bold',
  textAlign: 'center',
},
subtituloR:{
  fontSize: 30,
  color:'#FFFFFF' ,
  marginBottom:30,
},
labelInputR:{
  fontSize: 20,
  color:'#FFFFFF' ,
  marginRight:'50%',
  marginBottom:5,
},
labelInputCorreoR:{
  fontSize: 20,
  color:'#FFFFFF' ,
  marginRight:'54%',
```



```
        marginBottom:5,
    },
    labelInputPassR:{
        fontSize: 20,
        color: '#FFFFFF' ,
        marginRight: '44%',
        marginBottom:5,
    },
    textInputR:{
        padding:10,
        paddingStart:30,
        width: '80%',
        height: 50,
        borderRadius: 30,
        backgroundColor: '#FFFFFF',
        marginBottom:15,
    },
    textInputRUser:{
        padding:10,
        paddingStart:30,
        width: '80%',
        height: 50,
        borderRadius: 30,
        backgroundColor: '#BBBBBB',
        marginBottom:15,
    },
    textBtnR:{
        fontSize: 14,
    },
    signInBtnR:{
        width: '80%',
        height: 50,
        borderRadius: 25,
        padding:10,
        alignItems: 'center',
        justifyContent: 'center',
        backgroundColor: '#D9D9D9',
        marginTop:15,
    },
    textBtnGoogleR:{
        fontSize: 12,
        color: 'white',
    },
    loginbtnR:{
        width: '45%',
        height: 40,
        borderRadius: 25,
        padding:10,
        alignItems: 'center',
        justifyContent: 'center',
        backgroundColor: 'grey',
        marginTop:20,
        marginTop:20
    },
    // Global
    scroller: {
        flex: 1,
        marginBottom: 60
    }
}
```



```
},
container: {
  flex: 1,
  alignItems: 'center',
  justifyContent: 'center',
},
// UserPage
profilePicture: {
  width: 200,
  height: 200,
  borderRadius: 100,
  marginBottom: 20,
  marginTop: 50,
},
name: {
  fontSize: 24,
  fontWeight: 'bold',
  marginBottom: 10,
},
details: {
  fontSize: 18,
  marginBottom: 10,
},
bold:{
  fontWeight: 'bold',
},
boldColor:{
  color: '#5E3B3B',
  fontWeight: 'bold',
},
// ShoppingCartPage
titleArticulo:{
  fontSize: 40,
  textAlign:'center',
  marginTop: 25,
  marginBottom: 20,
  color: '#5E3B3B',
  fontWeight: 'bold',
},
boxCancelar:{
  backgroundColor: '#FFFFFF',
  alignSelf: 'flex-end',
  width: '100%',
  height: 80,
  textAlign: 'center',
  paddingTop: 25,
  position:'absolute',
  top: 420,
},
cancelarArticulo:{
  fontSize: 20,
},
// ShopCardPage
box: {
  width: '100%',
  height: 100,
  marginBottom: 20,
  flexDirection: 'row',
```



```
        marginTop: 10
    },
    boxText: {
        width: '70%',
        height: 100,
        marginBottom: 20,
        marginTop: 10,
        paddingTop: 15,
        marginLeft: 15,
    },
    boxPrice: {
        width: '30%',
        height: 100,
        marginBottom: 20,
        marginTop: 10,
        alignItems: 'center',
        paddingTop: 30
    },
    textTitle:{
        fontWeight: 'bold',
        fontSize: 35,
    },
    // ProfilePage
    // PayMethodPage
    // OrderHistoryPage
    // NotificationPage
    boxNotification: {
        width: '80%',
        height: 100,
        marginHorizontal: 40,
        marginVertical: 10,
        backgroundColor: '#5E3B3B',
        borderRadius: 15,
        color: '#fff',
        flex: 1,
        flexDirection: 'row',
        flexWrap: 'wrap',
        justifyContent: 'center',
    },
    textBox: {
        width: 206,
        height: 100,
        paddingHorizontal: 10,
        justifyContent: 'center',
    },
    notificationTextto: {
        marginLeft: 15,
        fontSize: 15,
        textAlign: 'center',
        color: '#fff',
    },
    pedidoPicture: {
        height: 100,
        width: 100,
        borderBottomRightRadius: 15,
        borderTopRightRadius: 15,
    },
    // MapPage
```



```
// HomePage
food: {
  width: widthWindow,
  flex: 1,
  flexDirection: 'row',
  flexWrap: 'wrap',
  justifyContent: 'center',
},
foodCard: {
  width: '44%',
  backgroundColor: '#5E3B3B',
  alignItems: 'center',
  margin: 8,
  borderRadius: 15,
},
foodImage: {
  width: 170,
  height: 170,
  borderRadius: 15,
},
nombreFood: {
  fontWeight: 'bold',
  color: 'white',
  fontSize: 20,
  padding: 8,
  textAlign: 'center',
  marginTop: 10,
},
txtFood: {
  textAlign: 'justify',
  padding: 7,
  color: '#d1d3d6',
},
descPrecFood: {
  fontWeight: 'bold',
  color: 'white',
},
// TopBar
// ProfileCards
texto: {
  fontSize: 15,
  width: 200,
  fontWeight: '600',
  paddingLeft: 5,
  textAlign: 'center',
},
option: {
  flex: 1,
  flexDirection: 'row',
  justifyContent: 'space-between',
  alignItems: 'center',
  marginTop: 20,
  paddingVertical: 15,
  paddingHorizontal: 30,
},
// Buscador
icono: {
  //padding: 7.7,
```





```
padding: 5.8,
paddingLeft: 20,
backgroundColor: 'white',
borderBottomLeftRadius: 50,
borderTopLeftRadius: 50,
borderColor: '#5E3B3B',
borderTopWidth: 1,
borderBottomWidth: 1,
borderLeftWidth: 1,
marginTop: 0.5,
},
containerBuscador: {
  flex: 1,
  flexDirection: 'row',
  justifyContent: 'center',
  alignItems: 'center',
  paddingBottom: 25,
  paddingTop: 25,
  paddingHorizontal: 30,
},
buscador: {
  flex: 1,
  paddingTop: 5,
  paddingRight: 10,
  paddingBottom: 5,
  paddingLeft: 10,
  backgroundColor: '#fff',
  color: '#000',
  borderColor: '#5E3B3B',
  borderTopWidth: 1,
  borderBottomWidth: 1,
  borderRightWidth: 1,
  borderBottomRightRadius: 50,
  borderTopRightRadius: 50,
  fontSize: 20,
},
// Navbar
// ProfileNavigation
// Article
img: {
  width: 410,
  height: 300,
  marginBottom: 20,
},
textTittle: {
  fontSize: 25,
  marginLeft: 4,
  marginBottom: 30,
},
precio: {
  fontSize: 25,
  marginLeft: 300,
},
detalles: {
  fontSize: 18,
  marginRight: 210,
  marginLeft: 25,
  marginTop: -70,
```



```
},
carrito: {
  fontSize: 18,
  color: 'blue',
  textDecorationLine: 'underline',
  marginLeft: 220,
},
viewPagar: {
  width: 380,
  flex: 1,
  flexDirection: 'row',
  flexWrap: 'wrap',
  justifyContent: 'center',
  marginTop: 10,
  marginBottom: 30,
},
iconoPagar: {
  backgroundColor: '#5E3B3B',
  width: 80,
  marginLeft: 20,
  padding: 15,
  borderRadius: 20,
},
botonPagar: {
  backgroundColor: '#5E3B3B',
  width: 230,
  padding: 15,
  borderRadius: 20,
},
textoBotonPagar: {
  color: '#fff',
  fontWeight: 'bold',
  fontSize: 17,
  textAlign: 'center',
},
// PayPage
tituloPagar: {
  color: '#5E3B3B',
  fontWeight: 'bold',
  fontSize: 25,
  textAlign: 'center',
  marginBottom: 20,
},
viewPagarResumen: {
  width: 380,
  flex: 1,
  flexDirection: 'row',
  flexWrap: 'wrap',
  justifyContent: 'space-around',
  marginTop: 10,
  marginBottom: 10,
},
textoResumenNombre: {
  color: '#000',
  fontWeight: 'bold',
  fontSize: 17,
  textAlign: 'center',
},
},
```



```
textoResumenPrecio: {
  color: 'grey',
  fontWeight: 'bold',
  fontSize: 17,
  textAlign: 'center',
},
linea:{
  width: 320,
  height: 2,
  backgroundColor: 'grey',
  marginLeft: 30,
},
textoPagarNormal: {
  color: '#000',
  fontWeight: 'bold',
  fontSize: 25,
  textAlign: 'center',
  marginBottom: 20,
  paddingHorizontal: 40,
}
});

export default GlobalStyles;
```

- routes/LoginNavigation.js

```
import React from 'react';
import { createStackNavigator } from '@react-navigation/stack';
import { NavigationContainer } from '@react-navigation/native';
import { useTheme } from '@react-navigation/native';

import Login from '../views/Login';
import SignIn from '../views/SignIn';
import Navbar from '../components/Navbar';

const StackLogin = createStackNavigator();

const LoginNavigation = ({scheme}) => {
  const { colors } = useTheme();
  return (
    <NavigationContainer independent={true}>
      <StackLogin.Navigator initialRouteName="Login">
        <StackLogin.Screen
          name="Login"
          component={Login}
          options={{
            headerStyle: { backgroundColor: colors.card, height: 70 },
            headerTintColor: colors.text,
            headerShown: false,
            headerTitle: 'Inicia Sesión',
            headerTitleAlign: 'center',
            headerTitleStyle: { fontSize: 26 },
          }}
        />
        <StackLogin.Screen
          name="SignIn"
          component={SignIn}
          options={{
```



```
        headerStyle: { backgroundColor: colors.card, height: 70 },
        headerTintColor: colors.text,
        headerShown: false,
        headerTitle: 'Registro',
        headerTitleAlign: 'center',
        headerTitleStyle: { fontSize: 26 },
      }}
    />
    <StackLogin.Screen
      name="Navbar"
      component={Navbar}
      options={{
        headerStyle: { backgroundColor: colors.card, height: 70 },
        headerTintColor: colors.text,
        headerShown: false,
        headerTitle: 'Navbar',
        headerTitleAlign: 'center',
        headerTitleStyle: { fontSize: 26 },
      }}
    />
  </StackLogin.Navigator>
</NavigationContainer>
);
};

export default LoginNavigation;
```

- routes/ProfileNavigation.js

```
import React from 'react';
import { createStackNavigator } from '@react-navigation/stack';
import { NavigationContainer } from '@react-navigation/native';
import { useTheme } from '@react-navigation/native';

import ProfilePage from '../views/ProfilePage';
import UserPage from '../views/UserPage';
import ModifyUserPage from '../views/ModifyUserPage';
import ShoppingCartPage from '../views/ShoppingCartPage';
import OrderHistoryPage from '../views/OrderHistoryPage';
import NotificationPage from '../views/NotificationPage';
import LoginNavigation from '../routes/LoginNavigation';
import PayPage from '../views/PayPage';
import DepositPage from '../views/DepositPage';

const StackProfile = createStackNavigator();

const usuario = require('../routes/user.json');

const ProfileNavigation = ({scheme}) => {
  const { colors } = useTheme();
  return (
    <NavigationContainer independent={true}>
      <StackProfile.Navigator initialRouteName="ProfilePage">
        <StackProfile.Screen
          name="ProfilePage"
          component={ProfilePage}
          options={{
            headerStyle: { backgroundColor: colors.card, height: 70 },
```



```
        headerTintColor: colors.text,
        headerTitle: 'Hola ' + usuario[0].Nombre,
        headerTitleAlign: 'center',
        headerTitleStyle: { fontSize: 26 },
    })
  />
<StackProfile.Screen
  name="UserPage"
  component={UserPage}
  options={{
    headerStyle: { backgroundColor: colors.card, height: 70 },
    headerTintColor: colors.text,
    headerTitle: "Perfil",
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
  }}
/>
<StackProfile.Screen
  name="ShoppingCartPage"
  component={ShoppingCartPage}
  options={{
    headerStyle: { backgroundColor: colors.card, height: 70 },
    headerTintColor: colors.text,
    headerTitle: 'Carrito',
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
  }}
/>
<StackProfile.Screen
  name="NotificationPage"
  component={NotificationPage}
  options={{
    headerStyle: { backgroundColor: colors.card, height: 70 },
    headerTintColor: colors.text,
    headerTitle: 'Notificaciones',
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
  }}
/>
<StackProfile.Screen
  name="OrderHistoryPage"
  component={OrderHistoryPage}
  options={{
    headerStyle: { backgroundColor: colors.card, height: 70 },
    headerTintColor: colors.text,
    headerTitle: 'Historial de Ordenes',
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
  }}
/>
<StackProfile.Screen
  name="LoginNavigation"
  component={LoginNavigation}
  options={{
    headerStyle: { backgroundColor: colors.card, height: 70 },
    headerTintColor: colors.text,
    headerTitle: 'Login',
    headerShown: false,
```



```
        headerTitleAlign: 'center',
        headerTitleStyle: { fontSize: 26 },
      })
    }
  }
  </StackProfile.Screen>
  <StackProfile.Screen
    name="PayPage"
    component={PayPage}
    options={{
      headerStyle: { backgroundColor: colors.card, height: 70
    },
    headerTintColor: colors.text,
    headerTitle: 'Pagar',
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
  }}
  </StackProfile.Screen>
  <StackProfile.Screen
    name="DepositPage"
    component={DepositPage}
    options={{
      headerStyle: { backgroundColor: colors.card, height: 70
    },
    headerTintColor: colors.text,
    headerTitle: 'Pagar',
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
  }}
  </StackProfile.Screen>
  <StackProfile.Screen
    name="ModifyUserPage"
    component={ModifyUserPage}
    options={{
      headerStyle: { backgroundColor: colors.card, height: 70
    },
    headerTintColor: colors.text,
    headerTitle: 'Modificar Usuario',
    headerTitleAlign: 'center',
    headerTitleStyle: { fontSize: 26 },
  }}
  </StackProfile.Screen>
</StackProfile.Navigator>
</NavigationContainer>
);
};

export default ProfileNavigation;
```

- views/DepositPage.js

```
import * as React from 'react';
import { ScrollView } from 'react-native';
import { View, Text, Pressable, Alert } from 'react-native';
import { useEffect, useState } from 'react';
import GlobalStyles from '../routes/GlobalStyles';

const usuario = require('../routes/user.json');
```



```
export default function DepositPage({ route, navigation }) {
  try {
    var contador = 0;
    const tipoPago = route.params.tipoPago
    const total = route.params.total
    const articulos = route.params.articulos
    const tipoData = route.params.tipoData
    const [dataPedido, setDataPedido] = useState([]);
    const [dataArticulo, setDataArticulo] = useState([]);
    const fetchDataPedido = async () => {
      const resp = await
fetch("https://devs-lutions-api.azurewebsites.net/pedidos");
      const dataPedido = await resp.json();
      setDataPedido(dataPedido);
    };
    useEffect(() => {
      fetchDataPedido()
    }, [])
    const fetchDataArticulo = async () => {
      const resp = await
fetch("https://devs-lutions-api.azurewebsites.net/articulos");
      const dataArticulo = await resp.json();
      setDataArticulo(dataArticulo);
    };
    useEffect(() => {
      fetchDataArticulo()
    }, [])

    const filtro = usuario.filter( pago => {
      return true
    })

    const insertOrUpdateData = () => {
      articulos.idArticulo.forEach( element => {
        if(tipoData.tipoData == "U"){
          dataPedido.forEach( pedido => {
            if(pedido.ID_Pedido == element){
              var url =
"https://devs-lutions-api.azurewebsites.net/pedidos/update/"+usuario[0]
.Id_Usuario+"/"+pedido.Id_articulo+"/"+pedido.Total+"/00:20:00/Cocinánd
ose/1/"+element
              fetch(url);
              console.log(url)
            }
          })
        }else if(tipoData.tipoData == "I"){
          dataArticulo.forEach( articulo => {
            if(articulo.ID_Articulo == element && contador == 0){
              var url =
"https://devs-lutions-api.azurewebsites.net/pedidos/insert/"+usuario[0]
.Id_Usuario+"/"+articulo.ID_Articulo+"/"+articulo.Precio+"/00:20:00/Coc
inándose/1"
              fetch(url);
              console.log(url)
              contador++;
            }
          })
        }
      })
    }
  }
}
```





```
});
Alert.alert('Pedido Actualizado', 'Su pedido ha sido registrado',
  [
    {text: '¡Genial!'},
  ],
);
};

const newArticleData = filtro.map( pago => {
  if(tipoPago == 'E'){
    return(
      <View key={1}>
        <Text style={[GlobalStyles.tituloPagar, {marginTop:
25}]}>Pagar con Efectivo</Text>
        <Text style={GlobalStyles.textoPagarNormal}>Monto
Total a Pagar: $ {total.total}</Text>
        <Text style={GlobalStyles.textoPagarNormal}></Text>
        <Text style={GlobalStyles.textoPagarNormal}>Se
notificará al establecimiento de tú pago con efectivo</Text>
        <Pressable
          style={[GlobalStyles.option, {backgroundColor:
'#5E3B3B', borderRadius: 15, marginHorizontal: 40, marginTop: 30,
marginBottom: 30,}]}
          android_ripple={{ color: '#bdc3c7' }}
          onPress={() => {insertOrUpdateData();
navigation.navigate('NotificationPage')}}>
          <Text style={[GlobalStyles.texto, {textAlign:
'center', width: 240, color: "#fff"}]}>Confirmar Pago con
Efectivo</Text>
        </Pressable>
      </View>
    )
  }else if(tipoPago == 'D'){
    return(
      <View key={1}>
        <Text style={[GlobalStyles.tituloPagar, {marginTop:
25}]}>Pagar con Depósito</Text>
        <Text style={GlobalStyles.textoPagarNormal}>Monto
Total a Pagar: $ {total.total}</Text>
        <Text style={GlobalStyles.textoPagarNormal}></Text>
        <Text style={GlobalStyles.tituloPagar}>Concepto:
"Pago Cafetería"</Text>
        <Text style={GlobalStyles.tituloPagar}>Numero de
Cuenta:{'\n'}"4915 6630 7930 4045"</Text>
        <Text style={GlobalStyles.textoPagarNormal}></Text>
        <Text style={GlobalStyles.textoPagarNormal}>Se
notificará al establecimiento de tú depósito</Text>
        <Pressable
          style={[GlobalStyles.option, {backgroundColor:
'#5E3B3B', borderRadius: 15, marginHorizontal: 40, marginTop: 30,
marginBottom: 30,}]}
          android_ripple={{ color: '#bdc3c7' }}
          onPress={() => {insertOrUpdateData();
navigation.navigate('NotificationPage')}}>
          <Text style={[GlobalStyles.texto, {textAlign:
'center', width: 240, color: "#fff"}]}>Confirmar Pago con
Depósito</Text>
        </Pressable>
      </View>
    )
  }
});
```



```
        </View>
      )
    }
  })

  return (
    <ScrollView style={GlobalStyles.scroller}>
      {newArticleData}
    </ScrollView>
  );

} catch(error) {
  console.log(error);
}
}
```

- views/DetailsPage.js

```
import * as React from 'react';
import { ScrollView } from 'react-native';
import { View, Text, Image, Pressable, Alert } from 'react-native';
import { Icon } from 'react-native-elements';
import { useEffect, useState } from 'react';
import { useNavigation } from '@react-navigation/native';
import GlobalStyles from '../routes/GlobalStyles';

const usuario = require('../routes/user.json');

export default function DetailsFood({ route }) {
  try {
    const navigation = useNavigation();
    const idArticulo = route.params.articulo
    const [data, setData] = useState([]);
    const fetchData = async () => {
      const resp = await
fetch("https://devs-lutions-api.azurewebsites.net/articulos");
      const data = await resp.json();
      setData(data);
    };
    useEffect(() => {
      fetchData()
    }, [])

    const articleData = data.filter( datos => {
      if(datos.ID_Articulo == idArticulo){
        return true
      }else{
        return false
      }
    })

    const insertData = async () => {
      var url =
"https://devs-lutions-api.azurewebsites.net/pedidos/insert/"+usuario[0]
.Id_Usuario+"/"+articleData[0].ID_Articulo+"/"+articleData[0].Precio+"/
00:30:00/Aceptado/1"
```



```

        await fetch(url);

        Alert.alert('Agregado al Carrito', 'El artículo ha sido agregado al carrito',
            [
                {text: '¡Genial!'},
            ],
        );
    };

    const newArticleData = articleData.map( articulo => {
        return (
            <View style={GlobalStyles.container} key={idArticulo}>
                <Image source={{ uri: articulo.Imagen }}
style={GlobalStyles.img} />
                <Text style={[GlobalStyles.textTittle,
GlobalStyles.boldColor]}>{articulo.Nombre}</Text>
                <Text style={GlobalStyles.precio}>${articulo.Precio}
{'\n'}</Text>

                <Text style={GlobalStyles.detalles}>{articulo.Detalles}
{'\n'}</Text>
                <View style={GlobalStyles.viewPagar}>
                    <Pressable style={GlobalStyles.botonPagar}
android_ripple={{ color: '#bdc3c7' }} onPress={() =>
navigation.navigate('PayPage', {articuloPagar: [articulo.ID_Articulo],
tipoData: "I"})}>
                        <Text style={GlobalStyles.textoBotonPagar}>Pagar</Text>
                    </Pressable>
                    <Pressable style={GlobalStyles.iconoPagar}
android_ripple={{ color: '#bdc3c7' }} onPress={() => {insertData();
navigation.navigate('ShoppingCartPage')}}>
                        <Icon name="cart" type="ionicon" color="#fff" />
                    </Pressable>
                </View>
            </View>
        )
    })

    return (
        <ScrollView style={GlobalStyles.scroller}>
            { newArticleData }
        </ScrollView>
    );

} catch(error) {
    console.log(error);
}
}

```

- views/HomePage.js

```

import * as React from 'react';
import { useState } from 'react';
import { ScrollView } from 'react-native';
import Buscador from '../components/Buscador';
import FoodCards from '../components/FoodCards';
import GlobalStyles from '../routes/GlobalStyles';

```



```
export default function HomeScreen({ navigation }) {
  const [textoBuscado, setTextoBuscado] = useState({ busqueda: '' });
  return (
    <ScrollView style={GlobalStyles.scroller}>
      <Buscador texto={textoBuscado} setTexto={ (newText) =>
        {setTextoBuscado({ ...newText })}}></Buscador>
      <FoodCards textoBuscar={textoBuscado}></FoodCards>
    </ScrollView>
  );
}
```

- views/Login.js

```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { useEffect, useState } from 'react';
import { Text, View, TextInput, TouchableOpacity, Alert } from
'react-native';
import GlobalStyles from '../routes/GlobalStyles';

const usuario = require('../routes/user.json');

export default function Login({ navigation }) {
  const [textoCorreo, setCorreo] = useState({ usuarioCorreo: '' });
  const [textoContrasena, setContrasena] = useState({
    usuarioContrasena: '' });
  const [data, setData] = useState([]);
  const [loginPress, setLogin] = useState({ usuarioLogin: false });
  var contador = 0;

  const fetchData = async () => {
    const resp = await
fetch("https://devs-lutions-api.azurewebsites.net/usuarios");
    const data = await resp.json();
    setData(data);
  };
  useEffect(() => {
    fetchData();
  }, [])

  const userData = data.filter( datos => {
    if((datos.Correo == textoCorreo.usuarioCorreo) && (datos.Contrasena
== textoContrasena.usuarioContrasena)){
      contador++;
      usuario[0].Id_Usuario = datos.ID_Usuario;
      usuario[0].Nombre = datos.Nombre;
      usuario[0].Apellidos = datos.Apellidos;
      usuario[0].Correo = datos.Correo;
      usuario[0].Contrasena = datos.Contrasena;
      usuario[0].Edad = datos.Edad;
      usuario[0].Genero = datos.Genero;
      usuario[0].Telefono = datos.Telefono;
      usuario[0].Pago_Preferido = datos.Pago_Preferido;
      usuario[0].Foto = datos.Foto;
      return true
    }else{
      if(contador == 0){
```



```
        contador++;
        return true
      }else{
        return false
      }
    }
  })

  const alerta = () => {
    Alert.alert('Hola Nuevamente', 'Bienvenido ' + usuario[0].Nombre,
      [
        {text: '¡Genial!', onPress: () => console.log('Si')},
      ],
    );
  };

  const alertaNo = () => {
    Alert.alert('Prueba Nuevamente', 'Correo o Contraseña Incorrectas',
      [
        {text: 'Ok'},
      ],
    );
  };

  var userLogin = userData.map( datos => {
    if((datos.Correo == textoCorreo.usuarioCorreo) && (datos.Contrasena
    == textoContrasena.usuarioContrasena)){
      return (
        <TouchableOpacity style={GlobalStyles.loginBtnL} onPress={() =>
        {alerta(); navigation.navigate('Navbar')}} key={datos.Id_Usuario}>
        <Text style={GlobalStyles.textBtnL}>Iniciar Sesión</Text>
        </TouchableOpacity>
      )
    }else{
      return (
        <TouchableOpacity style={GlobalStyles.loginBtnL}
        key={datos.Id_Usuario} onPress={() => {alertaNo();}}>
        <Text style={GlobalStyles.textBtnL}>Iniciar Sesión</Text>
        </TouchableOpacity>
      )
    }
  })

  return (
    <View style={GlobalStyles.containerL}>
      <Text style={GlobalStyles.tituloL}>Pick-</Text>
      <Text style={GlobalStyles.tituloL}>Up</Text>
      <Text style={GlobalStyles.subtituloL}>Coffe n Ice</Text>
      <Text style={GlobalStyles.subtituloL}>Cream</Text>
      <TextInput style={GlobalStyles.textInputL} placeholder='Correo
      Electrónico' onChangeText={newCorreo => setCorreo({...textoCorreo,
      usuarioCorreo: newCorreo})}/>
      <TextInput style={GlobalStyles.textInputL}
      placeholder='Contraseña' onChangeText={newContrasena =>
      setContrasena({...textoContrasena, usuarioContrasena: newContrasena})}
      />
      <Text style={GlobalStyles.forgotPassL}>Forgot your
      password?</Text>
    </View>
  )
}
```



```
    {userLogin[userLogin.length-1]}
    <TouchableOpacity style={[GlobalStyles.loginbtnR, {marginTop:
30, marginBottom: 5}]} onPress={() => navigation.navigate('SignIn')}>
      <Text style={GlobalStyles.textBtnGoogleR}>Registrate</Text>
    </TouchableOpacity>
    <StatusBar style="auto" />
  </View>
);
}
```

- views/MapPage.js

```
import * as React from 'react';
import { Text, ScrollView, StyleSheet, View, Dimensions } from
'react-native';
import GlobalStyles from '../routes/GlobalStyles';
import MapView, { Marker, PROVIDER_GOOGLE } from 'react-native-maps';

const { widthWindow, heightWindow } = Dimensions.get('window');

export default function MapPage() {
  return (
    <ScrollView style={GlobalStyles.scroller}>
      <MapView
        style={styles.mapStyle}
        initialRegion={{
          latitude: 20.59678557600082,
          longitude: -100.38094955237226,
          latitudeDelta: 0.0122,
          longitudeDelta: 0.0421,
        }}
        customMapStyle={mapStyle}>
        <Marker
          draggable
          coordinate={{
            latitude: 20.59678557600082,
            longitude: -100.38094955237226,
          }}
          onDragEnd={
            (e) => alert(JSON.stringify(e.nativeEvent.coordinate))
          }
          title={'Aihnoa Helado'}
          description={'Helado Artesanal'}
        />
      </MapView>
    </ScrollView>
  );
}

const mapStyle = [
  {elementType: 'geometry', stylers: [{color: '#242f3e'}]},
  {elementType: 'labels.text.fill', stylers: [{color: '#746855'}]},
  {elementType: 'labels.text.stroke', stylers: [{color: '#242f3e'}]},
  {
    featureType: 'administrative.locality',
    elementType: 'labels.text.fill',
    stylers: [{color: '#d59563'}],
  },
],
```





```
{
  featureType: 'poi',
  elementType: 'labels.text.fill',
  stylers: [{color: '#d59563'}],
},
{
  featureType: 'poi.park',
  elementType: 'geometry',
  stylers: [{color: '#263c3f'}],
},
{
  featureType: 'poi.park',
  elementType: 'labels.text.fill',
  stylers: [{color: '#6b9a76'}],
},
{
  featureType: 'road',
  elementType: 'geometry',
  stylers: [{color: '#38414e'}],
},
{
  featureType: 'road',
  elementType: 'geometry.stroke',
  stylers: [{color: '#212a37'}],
},
{
  featureType: 'road',
  elementType: 'labels.text.fill',
  stylers: [{color: '#9ca5b3'}],
},
{
  featureType: 'road.highway',
  elementType: 'geometry',
  stylers: [{color: '#746855'}],
},
{
  featureType: 'road.highway',
  elementType: 'geometry.stroke',
  stylers: [{color: '#1f2835'}],
},
{
  featureType: 'road.highway',
  elementType: 'labels.text.fill',
  stylers: [{color: '#f3d19c'}],
},
{
  featureType: 'transit',
  elementType: 'geometry',
  stylers: [{color: '#2f3948'}],
},
{
  featureType: 'transit.station',
  elementType: 'labels.text.fill',
  stylers: [{color: '#d59563'}],
},
{
  featureType: 'water',
  elementType: 'geometry',
```





```
    stylers: [{color: '#17263c'}],
  },
  {
    featureType: 'water',
    elementType: 'labels.text.fill',
    stylers: [{color: '#515c6d'}],
  },
  {
    featureType: 'water',
    elementType: 'labels.text.stroke',
    stylers: [{color: '#17263c'}],
  },
];
const styles = StyleSheet.create({
  mapStyle: {
    width : widthWindow,
    height : 655,
  },
});
```

- views/ModifyUserPage.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { View, Text, Image, ScrollView, Pressable, TextInput, Alert }
from 'react-native';
import GlobalStyles from '../routes/GlobalStyles';

const usuario = require('../routes/user.json');

export default function ModifyUserPage({ navigation }) {
  try {
    const [textoNombre, setNombre] = useState({ usuarioNombre: '' });
    const [textoApellido, setApellido] = useState({ usuarioApellido: '' });
  });
  const [textoCorreo, setCorreo] = useState({ usuarioCorreo: '' });
  const [textoTelefono, setTelefono] = useState({ usuarioTelefono: '' });
  });
  const [textoContrasena, setContrasena] = useState({
    usuarioContrasena: '' });
  const [textoEdad, setEdad] = useState({ usuarioEdad: '' });
  const [textoGenero, setGenero] = useState({ usuarioGenero: '' });

  const [data, setData] = useState([]);
  const fetchData = async () => {
    const resp = await
fetch("https://devs-lutions-api.azurewebsites.net/usuarios");
    const data = await resp.json();
    setData(data);
  };
  useEffect(() => {
    fetchData()
  }, [])

  const userData = data.filter( datos => {
    if(datos.Correo == usuario[0].Correo){
      return true
    }else{

```



```
        return false
      }
    })

    const newData = userData.map( datos => {
      return (
        <View style={[GlobalStyles.container, {marginTop:40}]}
key={usuario[0].Id_Usuario>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Nombre:</Text></Text>
          <TextInput style={GlobalStyles.textInputRUser}
placeholder='Nombre' onChangeText={newTXT => setNombre({...textoNombre,
usuarioNombre: newTXT})}/>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Apellido:</Text></Text>
          <TextInput style={GlobalStyles.textInputRUser}
placeholder='Apellido' onChangeText={newTXT =>
setApellido({...textoApellido, usuarioApellido: newTXT})}/>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Correo Electrónico:</Text></Text>
          <TextInput style={GlobalStyles.textInputRUser}
placeholder='Correo Electrónico' onChangeText={newTXT =>
setCorreo({...textoCorreo, usuarioCorreo: newTXT})}/>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Teléfono:</Text></Text>
          <TextInput style={GlobalStyles.textInputRUser}
placeholder='Teléfono' onChangeText={newTXT =>
setTelefono({...textoTelefono, usuarioTelefono: newTXT})}/>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Contraseña:</Text></Text>
          <TextInput style={GlobalStyles.textInputRUser}
placeholder='Cotraseña' onChangeText={newTXT =>
setContrasena({...textoContrasena, usuarioContrasena: newTXT})}/>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Edad: </Text></Text>
          <TextInput style={GlobalStyles.textInputRUser}
placeholder='Edad' onChangeText={newTXT => setEdad({...textoEdad,
usuarioEdad: newTXT})}/>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Género: </Text></Text>
          <TextInput style={GlobalStyles.textInputRUser}
placeholder='Femenino / Masculino / Otro' onChangeText={newTXT =>
setGenero({...textoGenero, usuarioGenero: newTXT})}/>
        </View>
      )
    })

    const modifyData = async () => {
      if(textoNombre.usuarioNombre != "" && textoCorreo.usuarioCorreo
!= "" && textoContrasena.usuarioContrasena != "" && textoTelefono != ""
&& textoApellido.usuarioApellido != "" && textoEdad.usuarioEdad != ""
&& textoGenero.usuarioGenero != ""){
        var url =
"https://devs-lutions-api.azurewebsites.net/usuarios/update/"+textoNomb
re.usuarioNombre+"/"+textoApellido.usuarioApellido+"/"+textoCorreo.usua
rioCorreo+"/"+textoContrasena.usuarioContrasena+"/"+textoEdad.usuarioEd
ad+"/"+textoGenero.usuarioGenero+"/"+textoTelefono.usuarioTelefono+"/"+
usuario[0].Pago_Preferido+ "%20/"+usuario[0].Id_Usuario
```



```
    await fetch(url);

    usuario[0].Id_Usuario = usuario[0].Id_Usuario;
    usuario[0].Nombre = textoNombre.usuarioNombre;
    usuario[0].Apellidos = textoApellido.usuarioApellido;
    usuario[0].Correo = textoCorreo.usuarioCorreo;
    usuario[0].Contrasena = textoContrasena.usuarioContrasena;
    usuario[0].Edad = textoEdad.usuarioEdad;
    usuario[0].Genero = textoGenero.usuarioGenero;
    usuario[0].Telefono = textoTelefono.usuarioTelefono;
    usuario[0].Pago_Preferido = usuario[0].Pago_Preferido;
    usuario[0].Foto = usuario[0].Foto;

    Alert.alert('Usuario Modificado', 'El usuario ha sido
modificado correctamente',
    [
        {text: '¡Genial!', onPress: () =>
console.log('Modificado')},
    ],
    );
    }else{
        Alert.alert('Usuario No Modificado', 'El usuario no ha sido
modificado, revisa que tengas todos lo campos corretamente y que no te
falten por llenar',
        [
            {text: 'Ok', onPress: () => console.log('No
Modificado')},
        ],
        );
    }
    };

    const deleteData = async (posicion) => {
        var url =
"https://devs-lutions-api.azurewebsites.net/usuarios/delete/"+usuario[0
].Id_Usuario
        await fetch(url);

        Alert.alert('Usuario Eliminado', 'El usuario ha sido
eliminado',
        [
            {text: '¡Genial!'},
        ],
        );
    };

    return (
        <ScrollView style={GlobalStyles.scroller}>
            { newData }
            <Pressable
                style={[[GlobalStyles.option, {backgroundColor: '#5E3B3B',
borderRadius: 15, marginHorizontal: 40, marginTop: 20, marginBottom:
30}]]}
                android_ripple={{ color: '#bdc3c7' }}
                onPress={() => {modifyData();
navigation.navigate('UserPage')}}>
                <Text style={[[GlobalStyles.texto, {textAlign: 'center',
width: 240, color: "#fff"}]]}>Modificar</Text>
            </Pressable>
        </ScrollView>
    );
}
```



```
        </Pressable>
        <Pressable
            style={[GlobalStyles.option, {backgroundColor: '#5E3B3B',
borderRadius: 15, marginHorizontal: 40, marginTop: 0, marginBottom:
30}]}
            android_ripple={{ color: '#bdc3c7' }}
            onPress={() => {deleteData();
navigation.navigate('LoginNavigation')}}>
            <Text style={[GlobalStyles.texto, {textAlign: 'center',
width: 240, color: "#fff"}]}>Eliminar Perfil</Text>
        </Pressable>
    </ScrollView>
);
} catch(error) {
    console.log(error);
}
}
```

- views/NotificationPage.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { Text, ScrollView, View, Image, Alert, Pressable } from
'react-native';
import GlobalStyles from '../routes/GlobalStyles';

const usuario = require('../routes/user.json');

export default function NotificationPage({ navigation }) {
    try {
        const [dataPedidos, setDataPedidos] = useState([]);
        const [dataArticulos, setDataArticulos] = useState([]);
        const fetchDataPedidos = async () => {
            const respPedidos = await
fetch("https://devs-lutions-api.azurewebsites.net/pedidos");
            const dataPedidos = await respPedidos.json();
            setDataPedidos(dataPedidos);
        };
        useEffect(() => {
            fetchDataPedidos()
        }, [])
        const fetchDataArticulos = async () => {
            const respArticulos = await
fetch("https://devs-lutions-api.azurewebsites.net/articulos");
            const dataArticulos = await respArticulos.json();
            setDataArticulos(dataArticulos);
        };
        useEffect(() => {
            fetchDataArticulos()
        }, [])

        const pedidosData = dataPedidos.filter( pedido => {
            if((pedido.Status == 'Cocinándose') || (pedido.Status ==
'Atrasado')) && (pedido.Id_usuario == usuario[0].Id_Usuario)){
                return true
            }else{
                return false
            }
        })
    }
}
```



```
    })
    const articulosData = dataArticulos.filter( articulo => {
      return true
    })

    const deleteData = async (posicion) => {
      var url =
"https://devs-lutions-api.azurewebsites.net/pedidos/delete/"+pedidosData[posicion].ID_Pedido
      await fetch(url);

      Alert.alert('Notificación Eliminada', 'La notificación del
artículo ha sido eliminada',
        [
          {text: '¡Genial!'},
        ],
        );
    };

    const newData = pedidosData.map( (pedido, index) => {
      return (
        <Pressable style={GlobalStyles.boxNotification}
key={pedido.ID_Pedido} onPress={() => deleteData(index)}
android_ripple={{ color: '#bdc3c7' }}>
          <View style={GlobalStyles.textBox}>
            <Text style={GlobalStyles.notificationTexto}><Text
style={GlobalStyles.bold}>{articulosData[(pedido.Id_articulo)-1].Nombre
}</Text></Text>
            <Text style={GlobalStyles.notificationTexto}><Text
style={GlobalStyles.bold}>$ </Text>{pedido.Total}</Text>
            <Text style={GlobalStyles.notificationTexto}><Text
style={GlobalStyles.bold}>Estatus: </Text>{pedido.Status}</Text>
            <Text style={GlobalStyles.notificationTexto}><Text
style={GlobalStyles.bold}>Listo en:
</Text>{pedido.Tiempo_Estimado}</Text>
          </View>
          <Image
            source={{uri:
articulosData[(pedido.Id_articulo)-1].Imagen}}
            style={GlobalStyles.pedidoPicture}
          />
        </Pressable>
      )
    })

    return (
      <ScrollView style={GlobalStyles.scroller}>
        <View>
          <Text style={GlobalStyles.titleArticulo}>Artículos:</Text>
          { newData }
        </View>
      </ScrollView>
    );
  } catch(error) {
    console.log(error);
  }
}
```



## ● views/OrderHistoryPage.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { Text, ScrollView, View, Image, Alert, Pressable } from
'react-native';
import GlobalStyles from '../routes/GlobalStyles';

const usuario = require('../routes/user.json');

export default function OrderHistoryPage({ navigation }) {
  try {
    const [dataPedidos, setDataPedidos] = useState([]);
    const [dataArticulos, setDataArticulos] = useState([]);
    const fetchDataArticulos = async () => {
      const respArticulos = await
fetch("https://devs-lutions-api.azurewebsites.net/articulos");
      const dataArticulos = await respArticulos.json();
      setDataArticulos(dataArticulos);
    };
    useEffect(() => {
      fetchDataArticulos()
    }, [])
    const fetchDataPedidos = async () => {
      const respPedidos = await
fetch("https://devs-lutions-api.azurewebsites.net/pedidos");
      const dataPedidos = await respPedidos.json();
      setDataPedidos(dataPedidos);
    };
    useEffect(() => {
      fetchDataPedidos()
    }, [])

    const pedidosData = dataPedidos.filter( pedido => {
      if(((pedido.Status == 'Listo') || (pedido.Status == 'Rechazado')
|| (pedido.Status == 'No se pudo completar')) && (pedido.Id_usuario ==
usuario[0].Id_Usuario)){
        return true
      }
    })
    const articulosData = dataArticulos.filter( articulo => {
      return true
    })

    const deleteData = async (posicion) => {
      var url =
"https://devs-lutions-api.azurewebsites.net/pedidos/delete/"+pedidosDat
a[posicion].ID_Pedido
      await fetch(url);

      Alert.alert('Historial Eliminado', 'El historial del artículo ha
sido eliminado',
        [
          {text: '¡Genial!'},
        ],
        );
    };
  };
```





```

    const newData = pedidosData.map( (pedido, index) => {
      return (
        <Pressable style={GlobalStyles.boxNotification}
key={pedido.ID_Pedido} onPress={() => deleteData(index)}
android_ripple={{ color: '#bdc3c7' }}>
          <View style={GlobalStyles.textBox}>
            <Text style={GlobalStyles.notificationText}><Text
style={GlobalStyles.bold}>{articulosData[(pedido.Id_articulo)-1].Nombre
}</Text></Text>
            <Text style={GlobalStyles.notificationText}><Text
style={GlobalStyles.bold}>$ </Text>{pedido.Total}</Text>
          </View>
          <Image
            source={{uri:
articulosData[(pedido.Id_articulo)-1].Imagen}}
            style={GlobalStyles.pedidoPicture}
          />
        </Pressable>
      )
    })

    return (
      <ScrollView style={GlobalStyles.scroller}>
        <View>
          <Text style={GlobalStyles.titleArticulo}>Artículos:</Text>
          { newData }
        </View>
      </ScrollView>
    );
  } catch(error) {
    console.log(error);
  }
}

```

- views/PayMethodPage.js

```

import * as React from 'react';
import { ScrollView } from 'react-native';
import PayMethodCards from '../components/PayMethodCards';
import GlobalStyles from '../routes/GlobalStyles';

export default function PayMethodPage({ navigation }) {
  return (
    <ScrollView style={GlobalStyles.scroller}>
      <PayMethodCards></PayMethodCards>
    </ScrollView>
  );
}

```

- views/PayPage.js

```

import * as React from 'react';
import { ScrollView } from 'react-native';
import { View, Text, Linking, Pressable } from 'react-native';
import { useEffect, useState } from 'react';
import GlobalStyles from '../routes/GlobalStyles';

export default function PayPage({ route, navigation }) {

```





```
try {
  const idArticulo = route.params.articuloPagar
  const tipoData = route.params.tipoData
  var total = 0;
  const [dataPedido, setDataPedido] = useState([]);
  const [dataArticulo, setDataArticulo] = useState([]);
  const fetchDataPedido = async () => {
    const resp = await
fetch("https://devs-lutions-api.azurewebsites.net/pedidos");
    const dataPedido = await resp.json();
    setDataPedido(dataPedido);
  };
  useEffect(() => {
    fetchDataPedido()
  }, [])
  const fetchDataArticulo = async () => {
    const resp = await
fetch("https://devs-lutions-api.azurewebsites.net/articulos");
    const dataArticulo = await resp.json();
    setDataArticulo(dataArticulo);
  };
  useEffect(() => {
    fetchDataArticulo()
  }, [])

  const carrito = dataPedido.filter( datos => {
    return( idArticulo.some( element => {
      if(datos.ID_Pedido == element){
        return true
      }else{
        return false
      }
    })))
  })

  const articulosData = dataArticulo.filter( articulo => {
    return true
  })

  const newArticleData = carrito.map( pedido => {
    if(tipoData == "I"){
      total = total + articulosData[(idArticulo)-1].Precio
      return(
        <View style={GlobalStyles.viewPagarResumen}
key={pedido.ID_Pedido}>
          <Text
style={GlobalStyles.textoResumenNombre}>{articulosData[(idArticulo)-1].
Nombre}</Text>
          <Text style={GlobalStyles.textoResumenPrecio}>${
{articulosData[(idArticulo)-1].Precio}</Text>
        </View>
      )
    }else if(tipoData == "U"){
      total = total + pedido.Total
      return(
        <View style={GlobalStyles.viewPagarResumen}
key={pedido.ID_Pedido}>
```



```

        <Text
style={GlobalStyles.textoResumenNombre}>{articulosData[(pedido.Id_artic
ulo)-1].Nombre}</Text>
        <Text style={GlobalStyles.textoResumenPrecio}>${
{pedido.Total}</Text>
        </View>
    )
    }
    })

    return (
        <ScrollView style={GlobalStyles.scroller}>
            <Text style={GlobalStyles.tituloPagar, {marginTop:
25}}>Resumen de la Compra</Text>
            <Text style={GlobalStyles.tituloPagar}>Artículos:</Text>
            <View style={GlobalStyles.linea}></View>
            {newArticleData}
            <View style={GlobalStyles.linea}></View>
            <View style={GlobalStyles.viewPagarResumen}>
                <Text style={GlobalStyles.textoResumenNombre}>Total:</Text>
                <Text style={GlobalStyles.textoResumenPrecio}>${
{total}</Text>
            </View>
            <View style={GlobalStyles.viewPagarResumen}>
                <Text style={GlobalStyles.textoResumenNombre}>Comisión
(10%):</Text>
                <Text style={GlobalStyles.textoResumenPrecio}>${
{(total/10)}</Text>
            </View>
            <View style={GlobalStyles.linea}></View>
            <View style={GlobalStyles.viewPagarResumen}>
                <Text style={GlobalStyles.textoResumenNombre}>Total con
Comisión:</Text>
                <Text style={GlobalStyles.textoResumenPrecio}>${
{total+(total/10)}</Text>
            </View>
            <Pressable
                style={GlobalStyles.option, {backgroundColor: '#2980b9',
borderRadius: 15, marginHorizontal: 40, marginTop: 30}}
                android_ripple={{ color: '#bdc3c7' }}
                // value={login.usuarioLogin}
                onPress={() =>
Linking.openURL('http://PayPal.Me/AleGV258')}>
                <Text style={GlobalStyles.texto, {textAlign: 'center',
width: 240, color: "#fff"}}>Pagar con PayPal</Text>
            </Pressable>
            <Pressable
                style={GlobalStyles.option, {backgroundColor: '#27ae60',
borderRadius: 15, marginHorizontal: 40, marginTop: 30}}
                android_ripple={{ color: '#bdc3c7' }}
                // value={login.usuarioLogin}
                onPress={() => navigation.navigate('DepositPage',
{tipoPago: "E", total: {total}, articulos: {idArticulo}, tipoData:
{tipoData}})}>
                <Text style={GlobalStyles.texto, {textAlign: 'center',
width: 240, color: "#fff"}}>Pagar en Efectivo</Text>
            </Pressable>
            <Pressable

```



```
        style={[GlobalStyles.option, {backgroundColor: '#c0392b',
borderRadius: 15, marginHorizontal: 40, marginTop: 30, marginBottom:
30,}}]}

        android_ripple={{ color: '#bdc3c7' }}
        // value={login.usuarioLogin}
        onPress={() => navigation.navigate('DepositPage',
{tipoPago: "D", total: {total}, articulos: {idArticulo}, tipoData:
{tipoData}})}>
        <Text style={[GlobalStyles.texto, {textAlign: 'center',
width: 240, color: "#fff"}]>Pagar con Depósito</Text>
        </Pressable>
        </ScrollView>
    );

    }catch(error) {
        console.log(error);
    }
}
```

- views/ProfilePage.js

```
import * as React from 'react';
import { ScrollView } from 'react-native';
import { useTheme } from '@react-navigation/native';
import ProfileCards from '../components/ProfileCards';
import GlobalStyles from '../routes/GlobalStyles';

export default function ProfilePage({ navigation }) {
    const { colors } = useTheme();
    return (
        <ScrollView style={[GlobalStyles.scroller, { backgroundColor:
colors.background }]}>
            <ProfileCards navigation={navigation}></ProfileCards>
        </ScrollView>
    );
}
```

- views/ShoppingCartPage.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { Text, ScrollView, View, Image, Pressable, Alert } from
'react-native';
import GlobalStyles from '../routes/GlobalStyles';

const usuario = require('../routes/user.json');

export default function ShoppingCartPage({ navigation }) {
    try {
        const [dataPedidos, setDataPedidos] = useState([]);
        const [dataArticulos, setDataArticulos] = useState([]);
        var articulos = []
        const fetchDataArticulos = async () => {
            const respArticulos = await
fetch("https://devs-lutions-api.azurewebsites.net/articulos");
            const dataArticulos = await respArticulos.json();
            setDataArticulos(dataArticulos);
        };
    };
```



```
useEffect(() => {
  fetchDataArticulos()
}, [])
const fetchDataPedidos = async () => {
  const respPedidos = await
fetch("https://devs-lutions-api.azurewebsites.net/pedidos");
  const dataPedidos = await respPedidos.json();
  setDataPedidos(dataPedidos);
};
useEffect(() => {
  fetchDataPedidos()
}, [])

const pedidosData = dataPedidos.filter( pedido => {
  if((pedido.Status == 'Aceptado') && (pedido.Id_usuario ==
usuario[0].Id_Usuario)){
    return true
  }
})
const articulosData = dataArticulos.filter( articulo => {
  return true
})

const deleteData = async (posicion) => {
  var url =
"https://devs-lutions-api.azurewebsites.net/pedidos/delete/"+pedidosDat
a[posicion].ID_Pedido
  await fetch(url);

  Alert.alert('Eliminado del Carrito', 'El artículo ha sido sido
eliminado del carrito',
    [
      {text: '¡Genial!', onPress: () => console.log('Si')},
    ],
    );
};

const newData = pedidosData.map( (pedido, index) => {
  articulos.push(pedido.ID_Pedido)
  return (
    <Pressable style={GlobalStyles.food} key={pedido.ID_Pedido}
onPress={() => deleteData(index)} android_ripple={{ color: '#bdc3c7'
}}>
      <View style={GlobalStyles.foodCard} key={pedido.Id_articulo}>
        <Image
          source={{uri:
articulosData[(pedido.Id_articulo)-1].Imagen}}
          style={GlobalStyles.foodImage}
        />
        <Text
style={GlobalStyles.nombreFood}>{articulosData[(pedido.Id_articulo)-1].
Nombre}</Text>
        <Text style={GlobalStyles.txtFood}><Text
style={GlobalStyles.descPrecFood}>Detalles:
</Text><Text>{articulosData[(pedido.Id_articulo)-1].Detalles}</Text></T
ext>
        <Text style={[GlobalStyles.txtFood, {marginBottom:
10,}]}><Text style={GlobalStyles.descPrecFood}>Precio:
```



```

</Text><Text>${articulosData[ (pedido.Id_articulo)-1].Precio}</Text></Text>
    </View>
  </Pressable>
)
})

return (
  <ScrollView style={GlobalStyles.scroller}>
    <View>
      <Text style={GlobalStyles.titleArticulo}>Artículos:</Text>
      { newData }
      <Pressable
        style={[GlobalStyles.option, {backgroundColor: '#5E3B3B',
borderRadius: 15, marginHorizontal: 40, marginTop: 30, marginBottom:
30,}]}
        android_ripple={{ color: '#bdc3c7' }}
        onPress={() => navigation.navigate('PayPage',
{articuloPagar: articulos, tipoData: "U"})}>
        <Text style={[GlobalStyles.texto, {textAlign: 'center',
width: 240, color: "#fff"}]}>Pagar Todo</Text>
      </Pressable>
    </View>
  </ScrollView>
);
} catch (error) {
  console.log(error);
}
}

```

- views/SignIn.js

```

import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { useEffect, useState } from 'react';
import { Text, View, TextInput, TouchableOpacity, Alert } from
'react-native';
import GlobalStyles from '../routes/GlobalStyles';

export default function SignIn({ navigation }) {
  try {
    const [textoNombre, setNombre] = useState({ usuarioNombre: '' });
    const [textoCorreo, setCorreo] = useState({ usuarioCorreo: '' });
    const [textoTelefono, setTelefono] = useState({ usuarioTelefono: ''
});
    const [textoContrasena, setContrasena] = useState({
usuarioContrasena: '' });

    const insertData = async () => {
      if(textoNombre.usuarioNombre !== "" && textoCorreo.usuarioCorreo
!== "" && textoContrasena.usuarioContrasena !== "" && textoTelefono !==
""){
        var url =
"https://devs-lutions-api.azurewebsites.net/usuarios/insert/"+textoNomb
re.usuarioNombre+"/"+%20/"+textoCorreo.usuarioCorreo+"/"+textoContrasena.
usuarioContrasena+"/0/Otro/"+textoTelefono.usuarioTelefono+"/Efectivo/%
20"

```



```
    await fetch(url);
    Alert.alert('Usuario Registrado', 'El usuario ha sido
registrado correctamente',
    [
      {text: '¡Genial!', onPress: () =>
console.log('Registrado')},
    ],
    );
  }else{
    Alert.alert('Usuario No Registrado', 'El usuario no ha sido
registrado, revisa que tengas todos lo campos corretamente y que no te
falten por llenar',
    [
      {text: 'Ok', onPress: () => console.log('No Registrado')},
    ],
    );
  }
}
};

return (
  <View style={GlobalStyles.containerR}>
    <View>
      <Text style={GlobalStyles.tituloR}>Pick-</Text>
      <Text style={GlobalStyles.tituloR}>Up</Text>
      <Text style={GlobalStyles.subtituloR}>Registrarse</Text>
    </View>
    <Text style={GlobalStyles.labelInputR}>Nombre</Text>
    <TextInput style={GlobalStyles.textInputR} placeholder='Nombre'
onChangeText={newTXT => setNombre({...textoNombre, usuarioNombre:
newTXT})}/>
    <Text style={GlobalStyles.labelInputCorreoR}>Correo</Text>
    <TextInput style={GlobalStyles.textInputR} placeholder='Correo
Electrónico' onChangeText={newTXT => setCorreo({...textoCorreo,
usuarioCorreo: newTXT})}/>
    <Text style={GlobalStyles.labelInputPassR}>Contraseña</Text>
    <TextInput style={GlobalStyles.textInputR}
placeholder='Contraseña' onChangeText={newTXT =>
setContrasena({...textoContrasena, usuarioContrasena: newTXT})}/>
    <Text style={GlobalStyles.labelInputPassR}>Teléfono</Text>
    <TextInput style={GlobalStyles.textInputR}
placeholder='Teléfono' onChangeText={newTXT =>
setTelefono({...textoTelefono, usuarioTelefono: newTXT})}/>
    <TouchableOpacity style={GlobalStyles.signInBtnR} onPress={()
=> {insertData(); navigation.navigate('Login')}}>
      <Text style={GlobalStyles.textBtnR}>Registrarse</Text>
    </TouchableOpacity>
    <TouchableOpacity style={GlobalStyles.loginbtnR} onPress={() =>
navigation.navigate('Login')}>
      <Text style={GlobalStyles.textBtnGoogleR}>Inicia
Sesión</Text>
    </TouchableOpacity>
    <StatusBar style="auto" />
  </View>
);
} catch(error) {
  console.log(error);
}
}
```





## ● views/UserPage.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { View, Text, Image, ScrollView, Pressable } from
'react-native';
import GlobalStyles from '../routes/GlobalStyles';
const usuario = require('../routes/user.json');

export default function UserPage({ navigation }) {
  try {
    const [data, setData] = useState([]);
    const fetchData = async () => {
      const resp = await
fetch("https://devs-lutions-api.azurewebsites.net/usuarios");
      const data = await resp.json();
      setData(data);
    };
    useEffect(() => {
      fetchData()
    }, [])

    const userData = data.filter( datos => {
      if(datos.Correo == usuario[0].Correo){
        return true
      }else{
        return false
      }
    })

    const newData = userData.map( datos => {
      if(datos.Foto != "") {
        return (
          <View style={GlobalStyles.container}
key={usuario[0].Id_Usuario}>
            <Image
              source={{uri: datos.Foto}}
              style={GlobalStyles.profilePicture}
            />
            <Text style={GlobalStyles.name}><Text
style={GlobalStyles.boldColor}>{datos.Nombre}
{datos.Apellidos}</Text></Text>
              <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Correo Electrónico:</Text></Text>
                <Text style={GlobalStyles.details}>{datos.Correo}</Text>
                <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Teléfono:</Text></Text>
                  <Text style={GlobalStyles.details}>{datos.Telefono}</Text>
                  <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Método de Pago Preferido:</Text></Text>
                    <Text
style={GlobalStyles.details}>{datos.Pago_Preferido}</Text>
                    <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Edad: </Text>{datos.Edad}</Text>
                    <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Género: </Text>{datos.Genero}</Text>
                  </View>
                )
      }
    })
  }
}
```





```

    }else{
      return (
        <View style={GlobalStyles.container}>
          <Image
            source={{uri:
'https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fwww.androi
diostip.com%2Fimages%2Fusuario-8.png&f=1&nofb=1&ipt=353d46c1b4aa1a81817
7805290bbcc3415e4368a8315c305c3d4222bb031c999&ipt=images'}}
            style={GlobalStyles.profilePicture}
          />
          <Text style={GlobalStyles.name}><Text
style={GlobalStyles.boldColor}>{datos.Nombre}
{datos.Apellidos}</Text></Text>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Correo Electrónico:</Text></Text>
          <Text style={GlobalStyles.details}>{datos.Correo}</Text>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Teléfono:</Text></Text>
          <Text style={GlobalStyles.details}>{datos.Telefono}</Text>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Edad: </Text>{datos.Edad}</Text>
          <Text style={GlobalStyles.details}><Text
style={GlobalStyles.boldColor}>Género: </Text>{datos.Genero}</Text>
        </View>
      )
    }
  })

  return (
    <ScrollView style={GlobalStyles.scroller}>
      { newData }
      <Pressable
        style={[GlobalStyles.option, {backgroundColor: '#5E3B3B',
borderRadius: 15, marginHorizontal: 40, marginTop: 20, marginBottom:
30}]}
        android_ripple={{ color: '#bdc3c7' }}
        onPress={() => navigation.navigate('ModifyUserPage')}>
        <Text style={[GlobalStyles.texto, {textAlign: 'center', width:
240, color: "#fff"}]}>Modificar Información</Text>
      </Pressable>
    </ScrollView>
  );
} catch(error) {
  console.log(error);
}
}

```

- server.js

```

const express = require('express');
const rutas = require('./routes/rutas');
const cors = require('cors');
const app = express();

app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cors());
app.use('/', rutas);

```



```
app.listen(process.env.PORT || 5000, () => { console.log('API
Devs_Lutions'); });

module.exports = app;
```

- routes/rutas.js

```
var express = require('express');
var router = express.Router();
var database = require("../database/conexion.js");

router.get('/', function (req, res) {
  res.send('DEVS_LUTIONS_API - /pedidos - /usuarios -
/locales - /articulos - /ubicaciones');
});

// SELECT USUARIOS
router.get('/usuarios', function (req, res) {
  db = database.conectar();
  db.query("SELECT * FROM usuarios;",
  function (err, rows, fields) {
    if (err) {
      console.log(err)
    } else {
      res.status(200);
      res.json(rows);
    }
  })
  db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// SELECT ARTICULOS
router.get('/articulos', function (req, res) {
  db = database.conectar();
  db.query("SELECT * FROM articulos;",
  function (err, rows, fields) {
    if (err) {
      console.log(err)
    } else {
      res.status(200);
      res.json(rows);
    }
  })
  db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// SELECT LOCALES
router.get('/locales', function (req, res) {
  db = database.conectar();
  db.query("SELECT * FROM locales;",
  function (err, rows, fields) {
    if (err) {
      console.log(err)
    } else {
      res.status(200);
    }
  })
  db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});
```



```
        res.json(rows);
    }
})
db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// SELECT PEDIDOS
router.get('/pedidos', function (req, res) {
    db = database.conectar();
    db.query("SELECT * FROM pedidos;",
    function (err, rows, fields) {
        if (err) {
            console.log(err)
        } else {
            res.status(200);
            res.json(rows);
        }
    })
    db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// SELECT UBICACIONES
router.get('/ubicaciones', function (req, res) {
    db = database.conectar();
    db.query("SELECT * FROM ubicaciones;",
    function (err, rows, fields) {
        if (err) {
            console.log(err)
        } else {
            res.status(200);
            res.json(rows);
        }
    })
    db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// INSERT UBICACIONES
router.get('/ubicaciones/insert/:Calle/:Numero/:Colonia/:Edificio_Priva
da/:Codigo_Postal', function (req, res) {
    var parametros = req.params;
    db = database.conectar();
    db.query("INSERT INTO ubicaciones(Calle, Numero, Colonia,
Edificio_Privada, Codigo_Postal) VALUES(?, ?, ?, ?, ?);",
[parametros.Calle, parametros.Numero, parametros.Colonia,
parametros.Edificio_Privada, parametros.Codigo_Postal], (err, rows,
fields) => {
        if (err) {
            console.log(err)
        } else {
            res.status(200);
            res.json(rows);
        }
    })
    db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});
```



```
});  
  
// INSERT LOCALES  
router.get('/locales/insert/:Nombre/:Telefono/:Sitio_Web/:Correo/:Horario/:Detalles/:Logo/:Id_ubicacion', function (req, res) {  
  var parametros = req.params;  
  db = database.conectar();  
  db.query("INSERT INTO locales(Nombre, Telefono, Sitio_Web, Correo, Horario, Detalles, Logo, Id_ubicacion) VALUES(?, ?, ?, ?, ?, ?, ?, ?);", [parametros.Nombre, parametros.Telefono, parametros.Sitio_Web, parametros.Correo, parametros.Horario, parametros.Detalles, parametros.Logo, parametros.Id_ubicacion], (err, rows, fields) => {  
    if (err) {  
      console.log(err)  
    } else {  
      res.status(200);  
      res.json(rows);  
    }  
  })  
  db.end(function (err) { err ? console.log(err) : console.log('Conexión finalizada'); });  
});  
  
// INSERT PEDIDOS  
router.get('/pedidos/insert/:ID_Usuario/:ID_Articulo/:Total/:Tiempo_Estimado/:Status/:Id_Local', function (req, res) {  
  var parametros = req.params;  
  db = database.conectar();  
  db.query("INSERT INTO pedidos(ID_Usuario, ID_Articulo, Total, Tiempo_Estimado, Status, Id_Local) VALUES(?, ?, ?, ?, ?, ?);", [parametros.ID_Usuario, parametros.ID_Articulo, parametros.Total, parametros.Tiempo_Estimado, parametros.Status, parametros.Id_Local], (err, rows, fields) => {  
    if (err) {  
      console.log(err)  
    } else {  
      res.status(200);  
      res.json(rows);  
    }  
  })  
  db.end(function (err) { err ? console.log(err) : console.log('Conexión finalizada'); });  
});  
  
// INSERT USUARIOS  
router.get('/usuarios/insert/:Nombre/:Apellidos/:Correo/:Contrasena/:Edad/:Genero/:Telefono/:Pago_Preferido/:Foto', function (req, res) {  
  var parametros = req.params;  
  db = database.conectar();  
  db.query("INSERT INTO usuarios(Nombre, Apellidos, Correo, Contraseña, Edad, Genero, Telefono, Pago_Preferido, Foto) VALUES(?, ?, ?, ?, ?, ?, ?, ?);", [parametros.Nombre, parametros.Apellidos, parametros.Correo, parametros.Contrasena, parametros.Edad, parametros.Genero, parametros.Telefono, parametros.Pago_Preferido, parametros.Foto], (err, rows, fields) => {  
    if (err) {  
      console.log(err)  
    } else {
```



```
        res.status(200);
        res.json(rows);
    }
})
db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// INSERT ARTICULOS
router.get('/articulos/insert/:Nombre/:Detalles/:Precio/:Imagen',
function (req, res) {
    var parametros = req.params;
    db = database.conectar();
    db.query("INSERT INTO articulos(Nombre, Detalles, Precio, Imagen)
VALUES(?, ?, ?, ?);", [parametros.Nombre, parametros.Detalles,
parametros.Precio, parametros.Imagen], (err, rows, fields) => {
        if (err) {
            console.log(err)
        } else {
            res.status(200);
            res.json(rows);
        }
    })
    db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// DELETE UBICACIONES
router.get('/ubicaciones/delete/:ID_Ubicacion', function (req, res) {
    var parametros = req.params;
    db = database.conectar();
    db.query("DELETE FROM ubicaciones WHERE ID_Ubicacion = ?;",
[parametros.ID_Ubicacion], (err, rows, fields) => {
        if (err) {
            console.log(err)
        } else {
            res.status(200);
            res.json(rows);
        }
    })
    db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// DELETE LOCALES
router.get('/locales/delete/:ID_Local', function (req, res) {
    var parametros = req.params;
    db = database.conectar();
    db.query("DELETE FROM locales WHERE ID_Local = ?;",
[parametros.ID_Local], (err, rows, fields) => {
        if (err) {
            console.log(err)
        } else {
            res.status(200);
            res.json(rows);
        }
    })
})
```



```
db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// DELETE PEDIDOS
router.get('/pedidos/delete/:Id_Pedido', function (req, res) {
  var parametros = req.params;
  db = database.conectar();
  db.query("DELETE FROM pedidos WHERE Id_Pedido = ?;",
[parametros.Id_Pedido], (err, rows, fields) => {
    if (err) {
      console.log(err)
    } else {
      res.status(200);
      res.json(rows);
    }
  })
  db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// DELETE USUARIOS
router.get('/usuarios/delete/:ID_Usuario', function (req, res) {
  var parametros = req.params;
  db = database.conectar();
  db.query("DELETE FROM usuarios WHERE ID_Usuario = ?;",
[parametros.ID_Usuario], (err, rows, fields) => {
    if (err) {
      console.log(err)
    } else {
      res.status(200);
      res.json(rows);
    }
  })
  db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// DELETE ARTICULOS
router.get('/articulos/delete/:ID_Articulo', function (req, res) {
  var parametros = req.params;
  db = database.conectar();
  db.query("DELETE FROM articulos WHERE ID_Articulo = ?;",
[parametros.ID_Articulo], (err, rows, fields) => {
    if (err) {
      console.log(err)
    } else {
      res.status(200);
      res.json(rows);
    }
  })
  db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// UPDATE UBICACIONES
router.get('/ubicaciones/update/:Calle/:Numero/:Colonia/:Edificio_Priva
da/:Codigo_Postal/:ID_Ubicacion', function (req, res) {
```





```
var parametros = req.params;
db = database.conectar();
db.query("UPDATE ubicaciones SET Calle = ?, Numero = ?, Colonia =
?, Edificio_Privada = ?,Codigo_Postal = ? WHERE ID_Ubicacion = ?;",
[parametros.Calle, parametros.Numero, parametros.Colonia,
parametros.Edificio_Privada, parametros.Codigo_Postal,
parametros.ID_Ubicacion], (err, rows, fields) => {
    if (err) {
        console.log(err)
    } else {
        res.status(200);
        res.json(rows);
    }
})
db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// UPDATE LOCALES
router.get('/locales/update/:Nombre/:Telefono/:Sitio_Web/:Correo/:Horario/:Detalles/:Logo/:Id_ubicacion/:ID_Local', function (req, res) {
    var parametros = req.params;
    db = database.conectar();
    db.query("UPDATE locales SET Nombre = ?, Telefono = ?, Sitio_Web =
?, Correo = ?, Horario = ?, Detalles = ?, Logo = ?, Id_ubicacion = ?
WHERE ID_Local = ?;", [parametros.Nombre, parametros.Telefono,
parametros.Sitio_Web, parametros.Corrreo, parametros.Horario,
parametros.Detalles, parametros.Logo, parametros.Id_ubicacion,
parametros.ID_Local], (err, rows, fields) => {
        if (err) {
            console.log(err)
        } else {
            res.status(200);
            res.json(rows);
        }
    })
    db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// UPDATE PEDIDOS
router.get('/pedidos/update/:ID_Usuario/:ID_Articulo/:Total/:Tiempo_Estimado/:Status/:Id_Local/:Id_Pedido', function (req, res) {
    var parametros = req.params;
    db = database.conectar();
    db.query("UPDATE pedidos SET ID_Usuario = ?, ID_Articulo = ?, Total
= ?, Tiempo_Estimado = ?, Status = ?, Id_Local = ? WHERE Id_Pedido =
?;", [parametros.ID_Usuario, parametros.ID_Articulo, parametros.Total,
parametros.Tiempo_Estimado, parametros.Status, parametros.Id_Local,
parametros.Id_Pedido], (err, rows, fields) => {
        if (err) {
            console.log(err)
        } else {
            res.status(200);
            res.json(rows);
        }
    })
})
```





```
db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// UPDATE USUARIOS
router.get('/usuarios/update/:Nombre/:Apellidos/:Correo/:Contrasena/:Ed
ad/:Genero/:Telefono/:Pago_Preferido/:Foto/:ID_Usuario', function (req,
res) {
  var parametros = req.params;
  db = database.conectar();
  db.query("UPDATE usuarios SET Nombre = ?, Apellidos = ?, Correo =
?, Contraseña = ?, Edad = ?, Genero = ?, Telefono = ?, Pago_Preferido =
?, Foto = ? WHERE ID_Usuario = ?;", [parametros.Nombre,
parametros.Apellidos, parametros.Correo, parametros.Contrasena,
parametros.Edad, parametros.Genero, parametros.Telefono,
parametros.Pago_Preferido, parametros.Foto, parametros.ID_Usuario],
(err, rows, fields) => {
    if (err) {
      console.log(err)
    } else {
      res.status(200);
      res.json(rows);
    }
  })
  db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

// UPDATE ARTICULOS
router.get('/articulos/update/:Nombre/:Detalles/:Precio/:Imagen/:ID_Art
iculo', function (req, res) {
  var parametros = req.params;
  db = database.conectar();
  db.query("UPDATE articulos SET Nombre = ?, Detalles = ?, Precio =
?, Imagen = ? WHERE ID_Articulo = ?;", [parametros.Nombre,
parametros.Detalles, parametros.Precio, parametros.Imagen,
parametros.ID_Articulo], (err, rows, fields) => {
    if (err) {
      console.log(err)
    } else {
      res.status(200);
      res.json(rows);
    }
  })
  db.end(function (err) { err ? console.log(err) :
console.log('Conexión finalizada'); });
});

module.exports = router;
```

- database/conexion.js

```
const mysql = require('mysql');

function conectar() {
  var config =
  {
    host      : 'pick-up.mysql.database.azure.com',
```



```
user      : 'devs_lutions_admin',
password  : 'CmLl-1234',
database  : 'devs_lutions',
port: 3306,
};
const db = new mysql.createConnection(config);
db.connect(
  function (err) {
    if (err) {
      console.log("!!! Cannot connect !!! Error:");
      throw err;
    }
    else {
      console.log("Base de Datos conectada...");
    }
  });
return db;
};

module.exports = {
  "conectar": conectar,
}
```

### <Instalación de proyecto localmente>

1. Tener instalado **Node.js** en la versión 16 (cualquiera de ella, preferentemente la **16.16.0**), si tienes una versión superior desinstalar porque puede generar problemas con expo.
2. Instalar globalmente "**Expo CLI**" con el comando **npm install -g expo-cli** o **npm install --global expo-cli**.
3. Descargar el repositorio y navegar hasta **pick-up\**.
4. Una vez dentro del proyecto ejecutar los comandos **npm i --force**, esto creará la carpeta de **node\_modules**, se va a tardar unos minutos en instalar todo (máximo 10 minutos).
5. Ejecutar el comando **expo start**, esto les debería poner el QR y crear la carpeta **.expo**, quizá, tengan que hacer CTRL+Z o reiniciar la consola 1 vez, pero la segunda vez que pongan el comando ya debería funcionar.
6. Escanear el QR con la app de Expo Go, quizá, la primera vez aparezca una pantalla azul que dice que algo salió mal, reintentar hasta que lo reconozca, o volver a iniciar el entorno virtual.

### <Financiación>

Entidad	Fuente de recursos	Valor
Departamento de desarrollo	Devs_lutions	\$130 USD \$32.5 USD por miembro
Finanzas	Comisiones	10% por cada transacción.