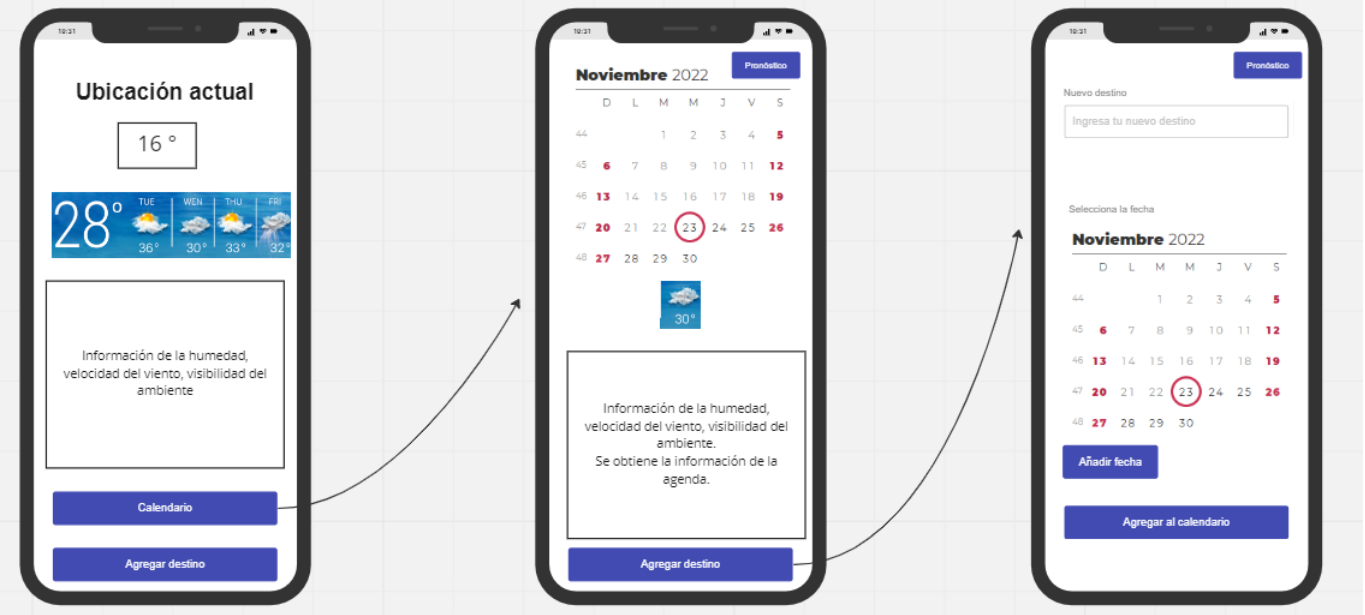
Aplicación Agenda Climática

- Asignatura: Cómputo Móvil
- Grupo: 03
- Semestre: 2023-1
- Fecha de entrega: 6/01/2023
- Profesor: Ing Marduk Pérez de Lara Domínguez
- Equipo 7
- Nombres:
  - Díaz Valenzuela Juan Carlos
  - González González Héctor Emilio
  - Meza Vega Hugo Adrián
  - Ramírez Martínez Humberto
- 3era Evaluación

Índice

1. Wireframes de la app (completos)



2. Explicación del flujo, cómo se recorren las pantallas.

Para ir a la segunda pantalla se presiona el botón de calendario donde se podrá agregar un destino.

Para acceder a la tercera pantalla, se debe presionar el botón agregar destino, en esta tercera pantalla se termina de configurar el evento. Una vez agregado el destino en el calendario, la aplicación regresa a la primera pantalla.

3. Si usan gestos especiales para algún contenido en su navegación o interacción indicarlo.

Gestos especiales no son requeridos. Entiéndase como gestos especiales como pinch to zoom o doble toque.
La app requiere únicamente de toques unitarios sobre los botones para ejecutar la acción indicada con el nombre del botón.
En el apartado del calendario, será necesario utilizar un slide de derecha a izquierda o de izquierda a derecha para seleccionar los meses del año.

4. Explicación de la o las funcionalidades de cada pantalla (el listado por pantalla de las funcionalidades que implementa)

En la pantalla principal contamos con la ubicación actual mostrando la temperatura actual, debajo con el pronóstico de días posteriores, debajo contamos con información de humedad, velocidad del viento, visibilidad del ambiente, debajo unos botones para cambiar a la vista del calendario y otro para agregar destino, las funciones de esta pantalla son obtener la ubicación actual del dispositivo, utilizar la api de pronóstico de clima, obtener el calendario que utilizamos como agenda.
Para la segunda pantalla obtenemos el calendario donde nos despliega la información de acuerdo al día que seleccionemos devolviendonos los datos de la pantalla principal, remarcando las fechas en donde tengamos compromisos. La tercera pantalla la utilizamos para agregar el destino por día deseado, donde también se pueden añadir más fechas para ese destino, en la pantalla 2 y 3 tenemos un botón en la parte superior derecha para regresar a la pantalla principal.

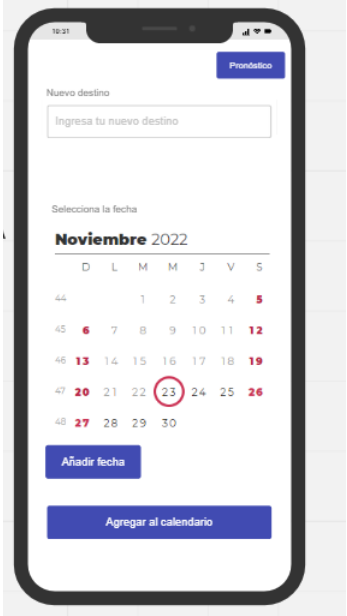
5. Por pantalla el análisis de los datos, tipos de datos y los servicios que necesitan conectar o implementar para el flujo de datos. Detallando si son de consulta, registro, borrado o actualización. También si ya existe el servicio en la nube (la API por ejemplo) detallar las características, costos y requerimientos para conectarse a él (como medios de autenticación, formatos, etc)

Pantalla 1:

Esta pantalla requiere de una estructura para almacenar la información de los JSON que proporciona la API de información del clima.

Además, se requieren Outlets para contener las imágenes e iconos de la interfaz gráfica, como el fondo, icono de clima (nublado, soleado, noche) y botones para las ciudades y acciones.

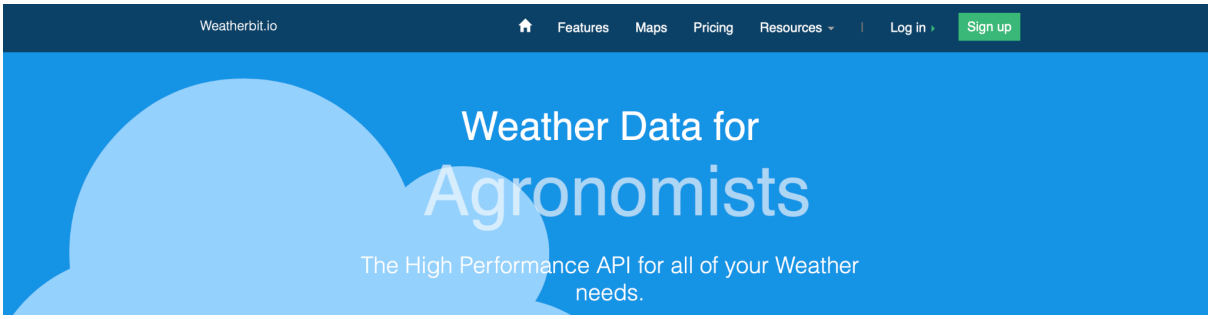
Pantalla 2 y 3:



En este caso se sigue conservando el uso de Outlets para contener los iconos e imágenes necesarias para la interfaz gráfica, utilizando los tipos UIImageView, UILabel, UIButton, por mencionar algunos.

```swift
import UIKit

struct JSONFile: Decodable {
    let data: [Data]
}

struct Data: Decodable {
    let rh: Int
    let pod: String
    let wind_spd: Double
    let vis: Double
    let weather: Weather
    let temp: Double
    let app_temp: Double
}

struct Weather: Decodable {
    let icon: String
    let description: String
}

class ViewController: UIViewController {
    @IBOutlet weak var fondoImage: UIImageView!
    @IBOutlet weak var lugarLabel: UILabel!
    @IBOutlet weak var iconoImage: UIImageView!
    @IBOutlet weak var descLabel: UILabel!
    @IBOutlet weak var tempLabel: UILabel!
    @IBOutlet weak var apTempLabel: UILabel!
    @IBOutlet weak var humedadLabel: UILabel!
    @IBOutlet weak var velVientoLabel: UILabel!
    @IBOutlet weak var visibilidadLabel: UILabel!

    @IBOutlet weak var beijingButton: UIButton!
    @IBOutlet weak var cdmxButton: UIButton!
    @IBOutlet weak var londresButton: UIButton!
    @IBOutlet weak var medellinButton: UIButton!
    @IBOutlet weak var parisButton: UIButton!
    @IBOutlet weak var sydneyButton: UIButton!

    func updateWeather(city: String, country: String, label: String) {
```

Para la recolección de datos del clima, se utilizará una API destinada a esta función. La API es https://www.weatherbit.io/

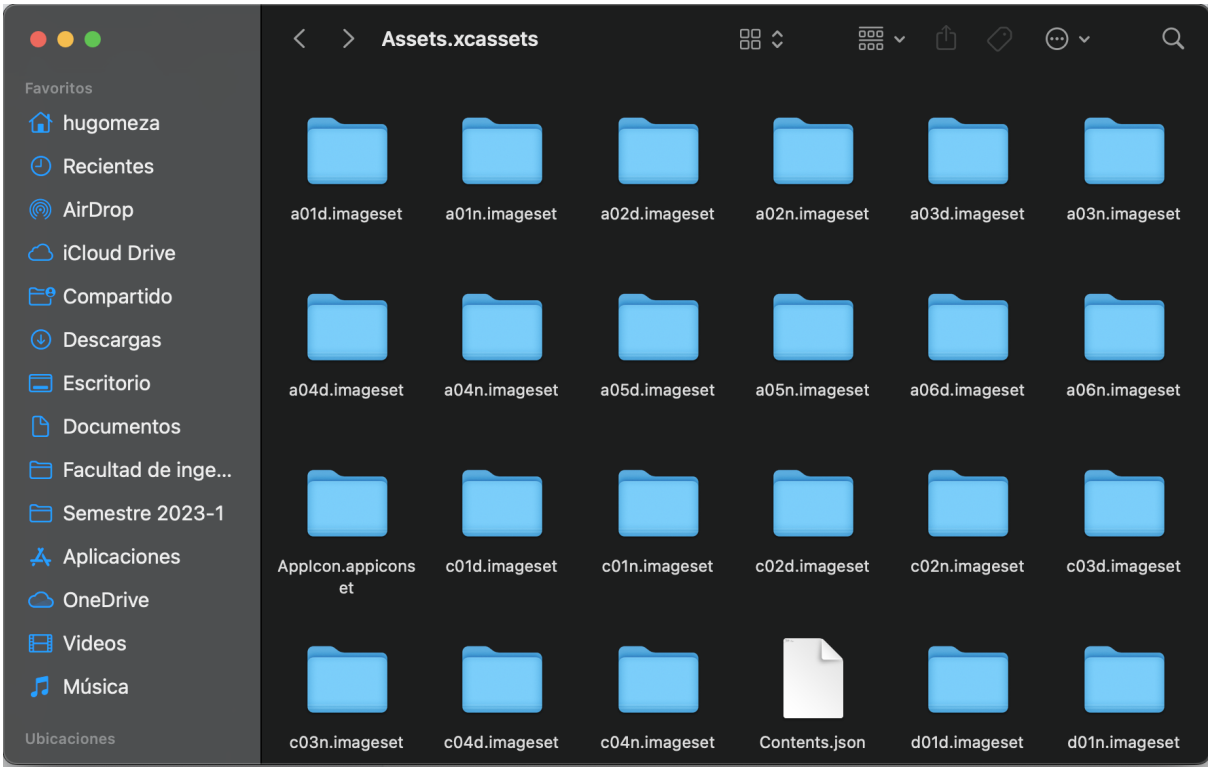


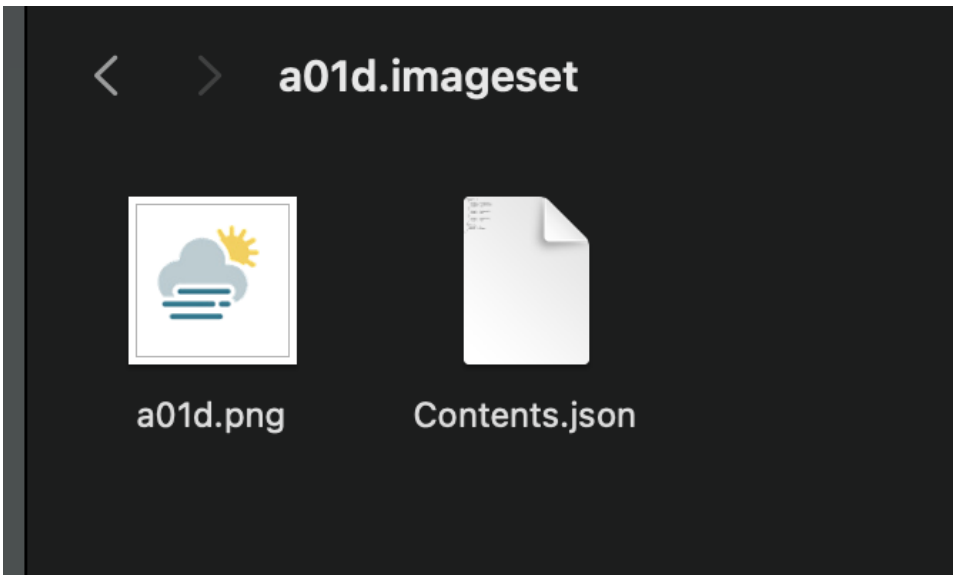
Esta App de uso libre proporciona información del clima de cualquier ciudad en el mundo. Esto lo hace a través de archivos JSON que son entregados a la APP mediante una consulta por medio de una URL especificando la región y ciudad deseada. La comunicación es sencilla y no requiere de ningún tipo de autenticación, puesto que se está manipulando información del clima y se accede a la API mediante la URL concatenando el país y la ciudad.

```
40    func updateWeather(city: String, country: String, label: String) {
41        let url = URL(string:
              "https://api.weatherbit.io/v2
              .0/current?city=\(city)&country=\
              (country)&key=664d2dd1adae46a6a5f816f99a3212aa&lang=es")
42
43        URLSession.shared.dataTask(with: url!) { (data, response, error) in
44            do {
45                let dataDecoded = try JSONDecoder().decode(JSONFile.self, from: data!)
46
47                let rh = dataDecoded.data[0].rh
48                let pod = dataDecoded.data[0].pod
49                let wind_spd = dataDecoded.data[0].wind_spd
50                let vis = dataDecoded.data[0].vis
51                let icon = dataDecoded.data[0].weather.icon
52                let description = dataDecoded.data[0].weather.description
53                let temp = dataDecoded.data[0].temp
54                let app_temp = dataDecoded.data[0].app_temp
55
56                DispatchQueue.main.async {
57                    self.lugarLabel.text = label
58                    self.iconoImage.image = UIImage(named: icon + ".png")
59                    self.descLabel.text = description
60                    self.tempLabel.text = "Temperatura: " + String(temp) + " ºC"
61                    self.apTempLabel.text = "Temp. aparente: " + String(app_temp) + " ºC"
62                    self.humedadLabel.text = "Humedad relativa: " + String(rh) + " %"
63                    self.velVientoLabel.text = "Vel. viento: " + String(wind_spd) + " m/s"
64                    self.visibilidadLabel.text = "Visibilidad: " + String(vis) + " km"
65
66                    if pod == "d" {
67                        self.fondoImage.image = UIImage(named: "day.jpg")
68                        let colorNegro: UIColor = UIColor(displayP3Red: 0, green: 0, blue: 0,
                              alpha: 1)
69                        self.lugarLabel.textColor = colorNegro
70                        self.descLabel.textColor = colorNegro
71                        self.tempLabel.textColor = colorNegro
72                        self.apTempLabel.textColor = colorNegro
                                                                    Line: 41  Col: 149
```

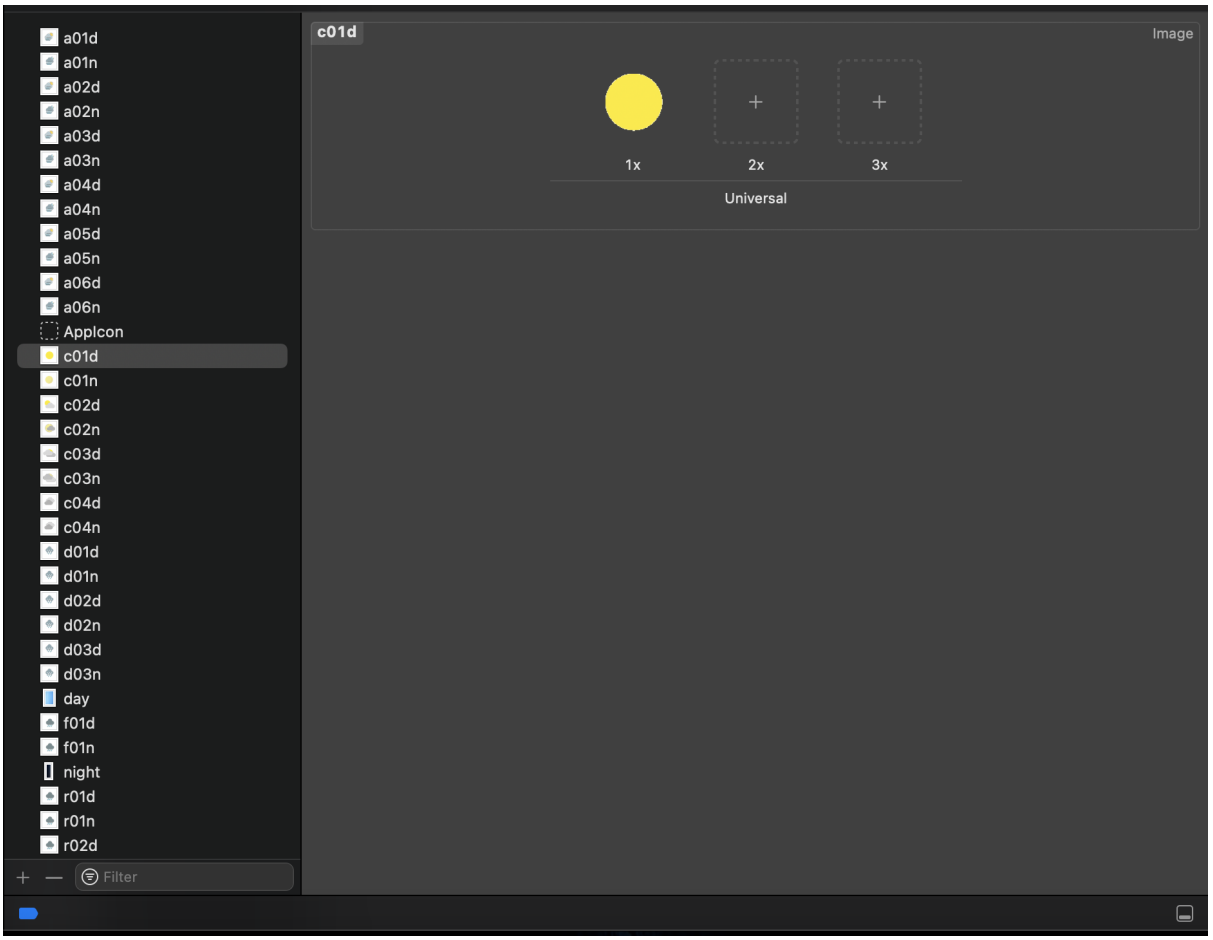
6. Por pantalla si usarán almacenamiento local y que datos serán los que almacene y por qué.

Debido a la poca complejidad de nuestra aplicación, se está optando por utilizar el almacenamiento local para contener todas las vistas, imágenes e iconos necesarios para la aplicación. Esto cambiará de forma dinámica dependiendo de las condiciones climáticas. Dado que no hay tantas variaciones, creemos que no impactará el uso de almacenamiento puesto que serán pocos los recursos a utilizar.





Las imágenes anteriores muestran la estructura de almacenamiento de los recursos gráficos que se utilizan para la interfaz gráfica de nuestra aplicación.



7. Para qué dispositivos está desarrollada, tamaños y orientaciones de pantalla.

Desarrollada para Android e iOS donde será distribuida por las tiendas de Play Store para Android y App Store para iOS, eligiendo estas tiendas ya que representan un gran porcentaje en el mercado de aplicaciones.

Para los tamaños decidimos que se desarrollará con un límite inferior para los smartphones de 5.42 pulgadas en diagonal (considerando al iPhone 13 mini como el smartphone más pequeño) y un límite superior de 7.6 pulgadas en diagonal (considerando el Samsung Galaxy Z Fold3 5G); mientras que para las tablets se considerará solamente un tamaño superior puesto que para él tamaño inferior se puede comenzar a partir del tamaño superior de los smartphones, el cual es de 7.6 pulgadas, hasta llegar a las 14.6 pulgadas que tiene la tablet más grande actualmente (Galaxy Tab S8 Ultra).

Se decidió que la orientación de la app dependerá del dispositivo en el que se esté utilizando, indicando que para cualquier smartphone (dispositivo menor a 7.6 pulgadas en diagonal) la orientación siempre será en vertical, ya que por el tamaño de la pantalla, no sería posible ver los datos correctamente si se pusiera el dispositivo de manera horizontal, mientras que para las tablets sin importar el tamaño de la pantalla será posible cambiar de disposición vertical a horizontal en cualquier momento y lo único que se verá modificado serán los márgenes laterales de la app, ya que la información siempre se encontrará centrada en la pantalla.

Como se tiene la posibilidad de poder interactuar con la app a través de accesorios como lo son Smartwatches, también es necesario considerar

este tamaño, pero al también depender del tipo de accesorio, se decidió crear de manera general sólo la notificación de cualquier aviso o cambio en la app, utilizando de manera básica las notificaciones generales de la app y del dispositivo del usuario.

8. Si utilizarán algún sensor del teléfono, describir cual, cómo se conectarán o usaran ese sensor y para qué.

Será necesario el uso del sensor GPS, ya que es necesario obtener la ubicación exacta del usuario para poder mostrar correctamente el clima y avisar cualquier cambio sin ningún contratiempo.
El dato de la ubicación del usuario será enviado a la API, la cual nos regresará la información del clima, este flujo se decidió así debido que de esta forma se evitará cualquier pérdida, interferencia y habrá más exactitud en la información, ya que la ubicación se actualizará en tiempo real en nuestra app, de lo contrario dependeríamos del funcionamiento de la API para la obtención de la ubicación y podría causar desfases drásticos en nuestra app.
La manera en la que utilizaremos el GPS de los dispositivos será obteniendo la ubicación exacta en tiempo real, esto se pensó así puesto que hay diversas maneras en las que algún usuario puede desplazarse a grandes velocidades (por ejemplo en un tren bala) y por esto será necesario tener siempre actualizada la ubicación.

9. Al menos una demo o maqueta del flujo de 3 pantallas con detalle de vista final (colores, imágenes, elementos finales, datos ejemplo).

En el siguiente link se encuentra un video demostrativo de una versión muy temprana del desarrollo de nuestra aplicación. En este demo se muestra la funcionalidad de la información del clima de varias localidades.

https://drive.google.com/file/d/1yLA7OjVRwOL5jV9sd4o9tk0OoJNrPGwp/view?usp=sharing

10. Detalles sobre el o los lenguajes de programación que usarán y las herramientas para desarrollar.

Lenguajes:
Para la versión de Android, se utilizará el lenguaje de programación Java junto con Python y JS, ya que esto depende mucho del dispositivo y del servicio de lenguaje utilizado para obtener los datos de 'Weather API'.

En el caso del desarrollo de la aplicación de iOS, se utilizará el lenguaje de programación Swift, siendo esto debido a que es el lenguaje principal para desarrollar aplicaciones para estos dispositivos.

Herramientas:

Una de las principales herramientas a utilizar es la API 'Weatherbit.io' debido a que nos provee una interfaz compatible con varios lenguajes de programación que nos permitirá integrar los datos meteorológicos

que necesitamos dentro del sistema de nuestra aplicación de agenda climática.

11. Equipo de trabajo y roles que intervienen para realizarla (la parte de desarrollo de software e implementación).
    - Líder de proyecto (1 persona), debe tener perfil con liderazgo, conocimiento técnico suficiente con respecto al desarrollo de aplicaciones, servicios en la nube.
    - Desarrolladores back end (2 personas), con perfil correspondiente al conocimiento de servicios en la nube para su desarrollo, una persona con conocimiento en desarrollo de aplicaciones iOS y la otra en Android, se irán ayudando en el proyecto ya que en sí es la misma aplicación pero en dos tiendas distintas.
    - Especialista en redes sociales, marketing, ya que buscamos promocionar nuestra aplicación en las redes sociales, principalmente instagram, facebook, necesitamos un perfil creativo que tenga buenas ideas para promocionar nuestra aplicación.
    - Diseñador UI/UX, necesitamos de un perfil hábil y creativo para desarrollar la interfaz de nuestra aplicación.

12. Estimaciones de tiempo de desarrollo y costos (la parte de desarrollo de software, implementación y mantenimiento).
    - Líder de proyecto $45,000 mxn.
    - Desarrolladores back end $35,000 mxn por dos personas.
    - Especialista en redes sociales $20,000 mxn.

- Diseñador UI/UX $35,000 mxn.
- Servicios de nube para bases de datos, hosting de la aplicación, y servicios que puedan ser necesarios después $10,000 mxn a partir del segundo mes.

  Para un total de $180,000 mxn.

  Donde el proyecto será de 6 meses teniendo un costo total de: $1,070,000 mxn

  Donde tenemos un margen teniendo un presupuesto de $200,000 mxn al mes, con $1,200,000 mxn de presupuesto total, en caso de necesitar un colaborador más o un servicio no presupuestado sorpresa así como algún permiso o licencia.