Proyecto 1 Modelado y Programacion

Ricardo Emiliano Apodaca Cardiel

Índice

1. El problema

El problema requiere que se muestre el clima de las ciudades que deseen los sobrecargos y pilotos de manera facil y sencilla.

2. Entendiendo el problema

Para empezar lo que se desea obtener el el clima ciudades, mas concretamente de aeropuetos ya que la aplicación va a dirigida hacia pilotos y sobrecargos, para esto la aplicación tendra una interfaz gráfica para la facilidad de uso al usuario, que en este caso son pilotos y sobrecargos.

Notemso que no es necesario mostrar la ciudad de origen ya que los mismos, pilotos y sobrecargos, ya que se encuentran ahi, o si desean saber el clima exacto pueden solicitarlo como se mostrara en breves y como los usuario estan dentro del mundo de la aviacion es de esperar que conozcan tanto los codigos IATA como ICAO, por lo que solicitaremos estos datos para saber de donde obtener el clima del lugar deseado.

Si es posible encontrar el clima del lugar, esto se discutira mas adelante en sección 5, se le mostara al usuario, al igual que si hubo problemas.

Por lo que de entrada se requerira el codigo IATA o ICAO del aueropuerto que se desee saber el clima, y de salida se le mostrara este al usuario.

3. Arsenal

3.1. Lenguaje y Herramientas

Para este proyecto se decidio que se utilizara Swift para la crecion de este. Ya que nos permitira crear la aplicación para dispotivos iOS, macOs, y iPadOS, para facilitar el uso de la aplicación. Aparte de esto se utilizara el framework de SwiftUI proporcionado por Apple para crear la interfaz grafica, debido a todos los elementos que nos propociona y la facilidad de uso. Y sera desarollada en en la IDE, Xcode, debido a todas las herramientas que propociona para el desarrollo para dentro del enterno de Apple. Tambien debido a que se necesita tranformar los codigos `IATA` a `ICAO` se decidio utilizar Realm basado en MongoDB, el cual ya tiene un Swift SDK. Tambien se genero una base de datos para que sea empaquetada con la applicacion y la transformacion de los codigos sea local. Y cabe recalcar que no es necesario subir la aplicación a la App Store para dar la aplicacion a los empleados, conectando al dispotivo de desarrollo es posible para la App a los dispotivos como una Enterprise App, solo se necesitan dar unos permisos a esta en el dispotivo de uso.

3.2. API

Tambien para la API se decidió por CheckWXAPI ya que usa el codigo ICAO como base para las peticiones, y nos da todo lo que requerimos como el clima y mas datos, que podrian resultar ser utiles. Una API sera necesaria, la expliación en la sección 4.

4. Requistos

La aplicación requerira varias condiciones para su funcinamiento optimo, primero descutiremso los requisitos funcionales, despues los no-funcionales, y por ultimo rapidamente lo que se requerira de parte del usuario.

4.1. Requisitos Funcionales

- Entregar el clima del aeropueto deseado. Como se vio en la sección 2.
- Solicitar al usuario el codigo ICAO o IATA del aeropueto del que se desee saber el clima.
- Hacer la llamada a la API, si es necesaria, mencionada en la sección 3 y procesar la informacion para presentarsela usuario.
- Tener un Cache para almacenar el clima de locaciones solicitadas recientemente para no hacer llamadas inecesarias a la API. Si la informacion tiene una antiguedad t, desecharla y solicitar una nueva a la API.
- Si no es posible obtener la informacion del clima, informale de esto al usuario.
- Debido a que la API solo acepta codigo ICAO, tranformar de IATA al codigo ICAO correspondiente.
- Presentarle la información del clima deseada al usuario en la interfaz gráfica.

4.2. Requisitos No-Funcionales

- Eficiencia y resposibidad por parte de la interfaz gráfica.
- Tolerar que el usuario ingrese un codigo inexiste o incorrecto, y si es posible informarle de su error, como por ejemplo, caracteres extra, o falta, o si es posible si los codigos que esta proporcionando no existen.
- Interfaz sencilla y autoexplicativa.
- Seguridad al guarda la llave de la API del usuario.
- Un manual de usuario integrado en la aplicación.

4.3. Requerimiento por Parte del Usuario

- Una coneccion a internet por parte del usuario para acceder a la API y obtener los climas, ya que son esto no es posible acceder a la API.
- Ingresar correctamente el aeropueto, ya que aunque podemos decirle al usuario el error, necesitamos saber concretamente a cual se refiere.

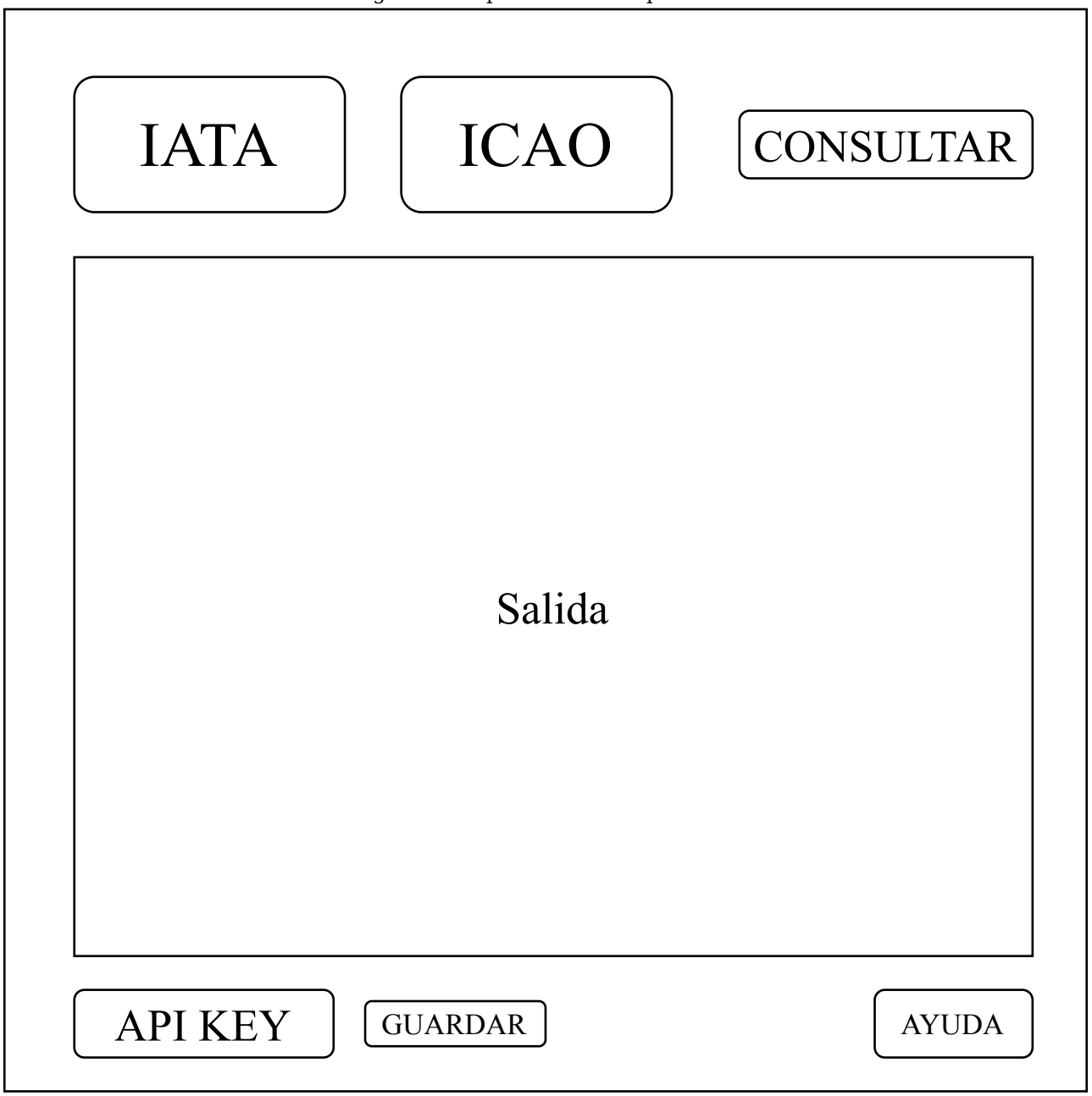
Con esto tenemos todo los requerimientos basicos por parte de la aplicación, aunque aun pueden ser modificados y sirguir nuevos conforme avance el proyecto.

5. Procesos

Ahora se mostraran los procesos y el funcionamiento interno de la aplicación, los metodos, funciones, estructuras y clases.

La manera en que funcioanra la aplicación, com ya se menciono sera con una interfaz gráfica para el usuario, la cual estara comunicandose con el modelo de la aplicación. La disposicion en la figura 5 una idea de la disposicion de la interfaz gráfica.

Figura 1: Disposicion de la aplicación



Las entradas serian los campos de AITA y IACO, mientras que tambien sera necesario que se proporcione una llave para la API, por lo que seria otra entrada, ya que no es seguro incluir en el codigo, por lo que se requeria que el usuario proporcione su llave, tambien se vera la posibilidad de guardar la llave en memoria para la comodidad del usuario.

Ahora pasando al modelo, veremos lo que lo va a conformar. Seguiremos el camino que seguiria los datos de entrada en el programa.

5.1. Algoritmos

Primero recibiremos la informacion recibida a travez de la interfaz, pero hay que tomar en cuanta que como tenemos 2 tipos de entrada, la IATA y la ICAO, por lo que se hara que la interfaz solo reciba 1 al mismo tiempo. Por otro lado recordemos que la API que se presento en la sección 3.2 de Arsenal, funciona con ICAO, no IATA, por lo que deberemos transformar el codigo IATA a ICAO.

Algoritmo 1 Tranforma de IATA a ICAO

Require: Codigo IATA

Primera concidencia de la base de datos del codigo IATA **return** Codigo ICAO de la coincidencia de la base de datos.

Despues de conseguir el codigo ICAO, ya sea directamente del usuario o por medio del algoritmo 1, seguira segura hacer la consulta del clima, para esto se tendra la cache como un diccionario con llaves de tipo `String`, las cuales seran el codigo ICAO, y valores `[Información del Clima]` y `Date` para la cache. Pero para esto definamos primero como guardaremos la informacion. Sera en un arreglo,y las estructuras del arreglo sera de la siguiente manera:

Información del Clima

- Codigo ICAO
- Codigo IATA
- Barometro
- Nubes
- Humedad
- Coordenadas
- Locación
- Nombre
- Temperatura
- Visibilidad
- Viento
- Tiempo de solicitud

Se incluira esto debido a que incluye la informacion basica asi como un poco de avanzada que puede llegar a ser de utilida a pilotos, aprte que la respuesta de la API en METAR, que es un formato para ver el clima actual, se ve de esta manera.

```
  {
"results": 1,
"data": [
  {
    "barometer": {
      "hg": 30.02,
      "hpa": 1017.0,
      "kpa": 101.66,
      "mb": 1016.62
    },
    "clouds": [
      {
        "code": "CLR",
        "text": "Clear skies"
      }
    ],
    "dewpoint": {
```

```
        "celsius": 21,
        "fahrenheit": 70
      },
      "elevation": {
        "feet": 3,
        "meters": 1
      },
      "flight_category": "VFR",
      "humidity": {
        "percent": 79
      },
      "icao": "KPIE",
      "observed": "2021-04-30T23:53Z",
      "raw_text": "KPIE 302353Z AUTO 31009KT 10SM CLR 25/21 A3002 RMK AO2 SLP165 T0250020
          6 10300 20250 55001",
      "station": {
        "geometry": {
          "coordinates": [
            -82.687401,
            27.9102
          ],
          "type": "Point"
        },
        "icao": "KPIE",
        "location": "St Petersburg-Clearwater, FL, USA",
        "name": "St Petersburg Clearwater International Airport",
        "type": "Airport"
      },
      "temperature": {
        "celsius": 25,
        "fahrenheit": 77
      },
      "visibility": {
        "meters": "16,000",
        "meters_float": 16000,
        "miles": "10",
        "miles_float": 10.0
      },
      "wind": {
        "degrees": 310,
        "speed_kph": 17,
        "speed_kts": 9,
        "speed_mph": 10,
        "speed_mps": 5
      }
    }
  ]
}
```


Ahora ya podemos pasar a ver como consultaremos el clima de un lugar. Este estara contenido en una estructura la cual sera la encargada de manejar la informacion dependiendo de lo que se le vaya solicitando la que tenga el diccionario con todos los climas. Tendra nadamas 2 atributos, ambos privados, los cuales seran el diccionario. `InformacionClimas` prevamiente mencionado y la llave de la API.

La llamda de la API tambien sera otra privada funcion dentro de la estructura que controlara toda la información. Por lo que reciberiamos la entrada con lo que despues la procesariamos principalmente con el algoritmo 2 para despues pasar lo obtenido a la interfaz, para que asi sea desplegda al usuario. Tambien hay que recalcar que hay que pasar la información de la peticion hacia API, a la estructura definida previamente para que de esta manera la aplicación entienda los datos, esto se hace de manera

Algoritmo 2 Algorimo para conseguir la informacion de clima de un lugar

Require: Codigo ICAO(ICAO), Llave de API(API_KEY)

```
solicitud_actual ← InformacionClimas[ICAO]
if solicitud_actual no existe o es muy vieja then
    x ← peticioAPI                ▷ PeticionAPI sera otra funcion dentro de la estructura
    if La llamada fue exitosa then

        solicitud_actual ← x
    else
return Llamada a la API fallida
    end if
end if
return solicitud_actual
```

relativamente sencilla con las herramientas que ya nos proporcina Swift como un JSONDecoder.

6. El Futuro

En el futuro es posible que la aplicación necesite mantenimento con futuros problemas que vayan surgiendo, o que surgan algunos bugs en el futuro. Pero por otro lado tambien puede que quede obsoleta debido a varios motivos.

Lo que mas hay que cuidar es la parte de la API, ya que puede cambiar el formato o dejar de existir, pero esto no deberia ser tanto problema asi ya que simplemente modificamos a la API que llamamos, y hacemso los cambios correspondientes, pasaria lo mismo si el cliente desea cambiar de API, o utilizar otra distinta, no seria una gran modificación del codigo fuente, seria simplemente agregar funcionalidad.

Por otro lado tambien podemos agregar mas tipos de entrada, por si desamos tambien recibir coordenadas, recibir la ciudad directamente, o algun otro tipo que deseemos agregar, seria algo similar que con la API, transformar esa entrada con la que ya sabemos como trabajar, por ejemplo ICAO o IATA, sin mencionar que la base de datos ya contiene las ciudades o el nombre del aeropeuerto.

Ahora por otro lado tambien se le pueden hacer mejoras a la interfaz como ocultar el campo de la API cuando ya exista una en memoria. Tambien posiblemente sea necesario agregar opciones de accesibilidad, que el usuario decida que antiguedad puede tener el clima, por ejemplo 10 minutos que es lo que esta actualemente, 20 min, 1 hora, etc. agregar mas idomas o lo que se vaya desando.

7. Cotización

El proyecto esta cotizado en $30,000.00, devido al tiempo invertido y al la especializacion de la implementación,en el que ya se incluye un mes de mantenimiento extra despues de la entrega final. Agregar funcionalidades entraria en un costo extra. Hay que tomar en cuanta que aunque la API es gratis esto es solo para 2000 peticiones diarias, si la empresa desearia utilizar solo 1 para todos su empleados eso entraria en un costo extra por parte de esta.