

Proyecto 1 Modelado y Programacion

Ricardo Emiliano Apodaca Cardiel

Índice

1. El problema	2
2. Entendiendo el problema	2
3. Arsenal	2
3.1. Lenguaje y Herramientas	2
3.2. API	2
4. Requisitos	3
4.1. Requisitos Funcionales	3
4.2. Requisitos No-Funcionales	3
4.3. Requerimiento por Parte del Usuario	3
5. Procesos	4
5.1. Algoritmos	5
6. El Futuro	7

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 ahí, 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 desarrollada en en la IDE, Xcode, debido a todas las herramientas que propociona para el desarrollo para dentro del enterno de Apple.

3.2. API

Tambien para la API se decidió por CheckWXAPI ya que usa el codigo ICAO como base para las peticio- nes, 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. Requisitos

La aplicación requiera varias condiciones para su funcionamiento óptimo, primero discutiremos los requisitos funcionales, después los no-funcionales, y por último rápidamente lo que se requiera de parte del usuario.

4.1. Requisitos Funcionales

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

4.2. Requisitos No-Funcionales

- Eficiencia y responsabilidad por parte de la interfaz gráfica.
- Tolerar que el usuario ingrese un código inexistente o incorrecto, y si es posible informarle de su error, como por ejemplo, caracteres extra, o falta, o si es posible si los códigos que está proporcionando no existen.
- Interfaz sencilla y autoexplicativa.
- Seguridad al guardar la llave de la API del usuario.
- Un manual de usuario integrado en la aplicación.

4.3. Requerimiento por Parte del Usuario

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

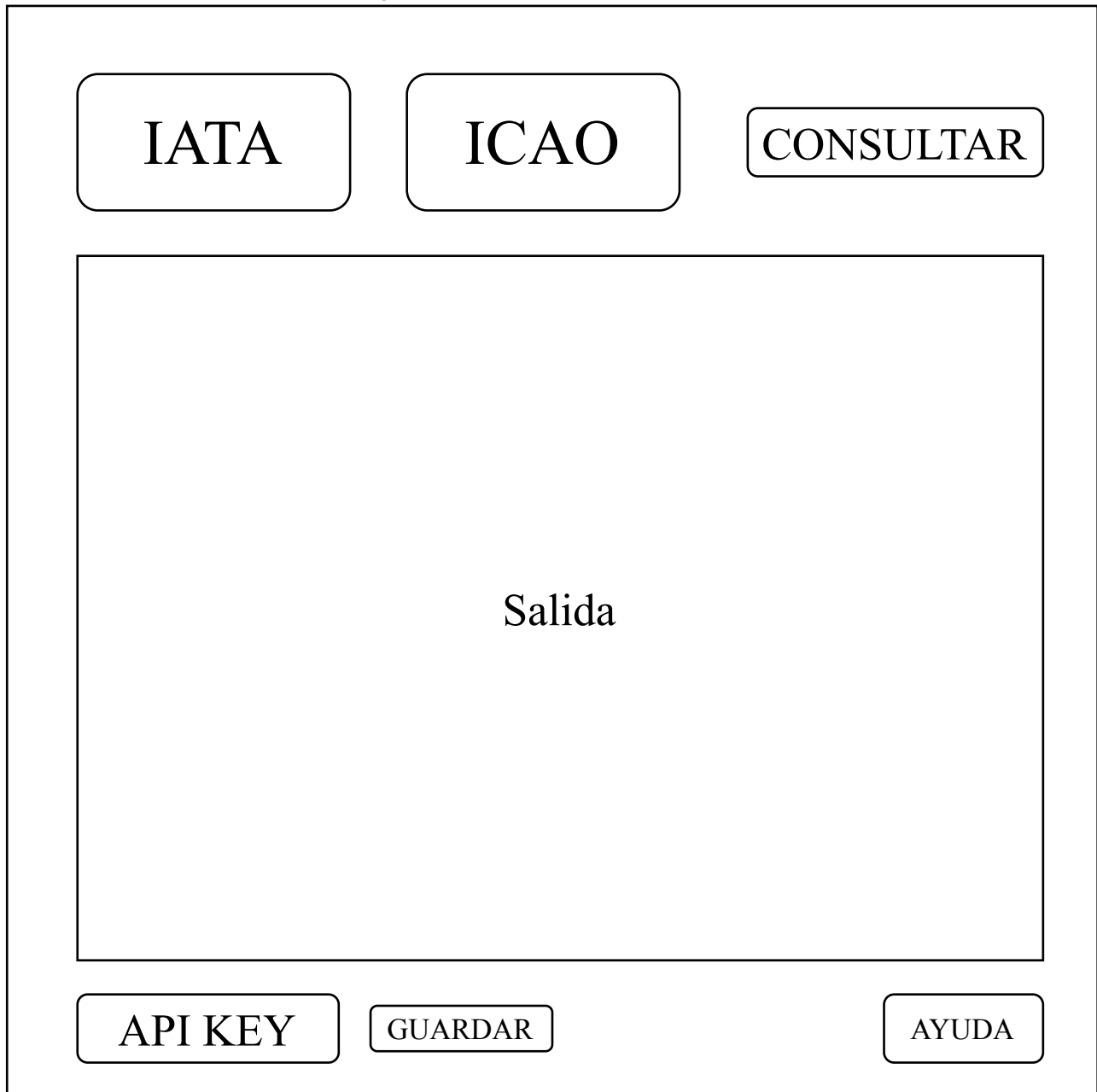
Con esto tenemos todos los requerimientos básicos por parte de la aplicación, aunque aún pueden ser modificados y surgir nuevos conforme avance el proyecto.

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 información recibida a través de la interfaz, pero hay que tomar en cuenta que como tenemos 2 tipos de entrada, la IATA y la ICAO, por lo que se hará que la interfaz solo reciba 1 al mismo tiempo. Por otro lado recordemos que la API que se presentó en la sección 3.2 de Arsenal, funciona con ICAO, no IATA, por lo que deberemos transformar el código IATA a ICAO.

Algoritmo 1 Transforma de IATA a ICAO

Require: Código IATA

Transformar de IATA a ICAO

▷ Aun no está implementado

return Código ICAO

Después de conseguir el código ICAO, ya sea directamente del usuario o por medio del algoritmo 1, seguirá segura hacer la consulta del clima, para esto se tendrá la cache como un diccionario con claves de tipo String, las cuales serán el código ICAO, y valor Información del Clima. Pero para esto definimos primero como guardaremos la información. Será en una estructura la cual guardará los siguientes datos:

Información del Clima

- Código ICAO
- Código IATA
- Barómetro
- Nubes
- Humedad
- Coordenadas
- Locación
- Nombre
- Temperatura
- Visibilidad
- Viento
- Tiempo de solicitud

Se incluya esto debido a que incluye la información básica así como un poco de avanzada que puede llegar a ser de utilidad a pilotos, aparte que la respuesta de la API en METAR, que es un formato para ver el clima actual, se ve de esta manera.

```
1  {
2  "results": 1,
3  "data": [
4    {
5      "barometer": {
6        "hg": 30.02,
7        "hpa": 1017.0,
8        "kpa": 101.66,
9        "mb": 1016.62
10     },
11     "clouds": [
12       {
13         "code": "CLR",
14         "text": "Clear skies"
15       }
16     ],
17     "dewpoint": {
18       "celsius": 21,
```

```

19     "fahrenheit": 70
20 },
21 "elevation": {
22     "feet": 3,
23     "meters": 1
24 },
25 "flight_category": "VFR",
26 "humidity": {
27     "percent": 79
28 },
29 "icao": "KPIE",
30 "observed": "2021-04-30T23:53Z",
31 "raw_text": "KPIE 302353Z AUTO 31009KT 10SM CLR 25/21 A3002 RMK A02 SLP165 T0250020
32     6 10300 20250 55001",
33 "station": {
34     "geometry": {
35         "coordinates": [
36             -82.687401,
37             27.9102
38         ],
39         "type": "Point"
40     },
41     "icao": "KPIE",
42     "location": "St Petersburg-Clearwater, FL, USA",
43     "name": "St Petersburg Clearwater International Airport",
44     "type": "Airport"
45 },
46 "temperature": {
47     "celsius": 25,
48     "fahrenheit": 77
49 },
50 "visibility": {
51     "meters": "16,000",
52     "meters_float": 16000,
53     "miles": "10",
54     "miles_float": 10.0
55 },
56 "wind": {
57     "degrees": 310,
58     "speed_kph": 17,
59     "speed_kts": 9,
60     "speed_mph": 10,
61     "speed_mps": 5
62 }
63 ]
64 }

```

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 llamada de la API tambien sera otra privada funcion dentro de la estructura que controlara toda la información. Por lo que recibieramos 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 desplegada al usuario. Tambien hay que recalcar que hay que pasar la información de la petición hacia API, a la estructura definida previamente para que de esta manera la aplicación entienda los datos, esto se hace de manera relativamente sencilla con las herramientas que ya nos proporcina Swift como un JSONDecoder.

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

Require: Codigo ICAO(ICA0), Llave de API(API_KEY)

solicitud_actual \leftarrow InformacionClimas[ICA0]

if solicitud_actual no existe o es muy vieja **then**

$x \leftarrow$ peticioAPI

 ▷ PeticionAPI sera otra funcion dentro de la estructura

if La llamada fue exitosa **then**

 solicitud_actual \leftarrow x

else

return Llamada a la API fallida

end if

end if

return solicitud_actual

6. El Futuro

En el futuro es posible que la aplicación necesite mantenimiento, 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.