Contenido

1. Autores del trabajo, planificación y entrega

1.1 Autores

Grupo 1 Tarde formado por:

- Patricia Sotodosos
- Eduardo V. Izquierdo
- Roberto Cabrera
- Jesús Melchor
- Eduardo Martín

1.2 Planificación

Planificación del TG3 realizada con Ganttpro:

https://app.ganttpro.com/shared/token/4cf6ce3cf86b5aaa99308ad87b47053c1c8f0137529a32f45786e12fe4f73ea2

1.3 Entrega

Repositorio del TG3 en Github:

https://github.com/Patricia-Sotodosos/TG3

2. Requisitos del prototipo a implementar

El objetivo del proyecto es comparar la implementación de un mismo prototipo de sistema utilizando dos tecnologías diferentes (Android y iOS).

Es importante cumplimentar este apartado antes de empezar a implementar el prototipo de cada tecnología, porque ambos prototipos deben cumplir los requisitos que se establezcan en él. El contenido de este apartado es lo que han de compartir ambos equipos (desarrolladores Android e iOS) como punto de partida.

2.1 Requisitos funcionales

En la siguiente tabla se indicará el catálogo de requisitos funcionales del sistema.

REQ.	DESCRIPCIÓN
RF001	La definición de datos se basará en un archivo en formato XML.
RF002	La aplicación ha de recibir los datos directamente de internet, sin ningún tipo de almacenamiento.
RF003	La aplicación mostrará por pantalla la información del tiempo incluyendo la temperatura.
RF004	La aplicación ha de ser capaz de mostrar un icono en relación a la descripción del tiempo obtenida.
RF005	El sistema ha de poder mostrar el tiempo de una determinada ubicación geográfica.
RF006	La temperatura se mostrará en ºC.
RF007	La aplicación realizará una búsqueda de datos única y seleccionará la hora adecuada en correspondencia con el sistema.
RF008	Existirá un botón que permita la actualización de los datos.
RF009	Cuando la app este sin conexión mostrará valores nulos.

2.2 Otros requisitos

En este apartado se incluyen todos los requisitos no funcionales, de tipo seguridad, interfaz, y otros tipos.

En la siguiente tabla se indicará el catálogo de requisitos no funcionales del sistema.

REQ.	DESCRIPCIÓN
RI001	El sistema ha de soportar un layout minimalista.
RI002	La aplicación ha de tener una interfaz de usuario utilizable por personas con daltonismo.
RI003	La interfaz debe utilizar un tamaño y un tipo de fuente totalmente legibles por los usuarios de la aplicación.
RI004	Los colores de la interfaz no han de ser demasiado intensos ni demasiado claros, con el fin de que sea lo más agradable posible.
RI005	El código de color será *#0088D6* para el fondo y blanco para las letras, botones o iconos.

REQ.	DESCRIPCIÓN
RL001	El sistema ha de desarrollarse en Java (Android) y en Swift (iOS).
RS001	El sistema almacenará adecuadamente los datos de los usuarios con la seguridad necesaria.
RR001	Todas las respuestas del sistema han de producirse en un máximo de 30 segundos.
RR002	Adquirir la información meteorológica no supone ningún coste.
RR003	La aplicación no ha de tardar más de 5 segundos en iniciarse.
RU001	Los usuarios han de aprender a utilizar la aplicación en un minuto.
RU002	La aplicación debe tener una navegación fácil y suave a través de sus pestañas.

3. Criterios de comparación en la implementación

A través del siguiente apartado se trata de establecer un conjunto de criterios que permitan evaluar la implementación de forma íntegra en cada una de las tecnologías en cuestión.

Todos los criterios de las diferentes categorías se valorarán en euros cuando este en el ámbito económico, en horas y minutos en los casos que se trate del tiempo en realizar alguna tarea y en intervalos de 0 a 10 en otro tipo de calificaciones, siendo 0 la peor puntuación y 10 la mejor.

3.1 Criterio 1: Tiempo de la creación de la interfaz de usuario

Se trata de ver el total de horas que se han necesitado para conseguir una interfaz de usuario funcional en el sistema iOS y en el sistema Android.

Tipo de valor (iOS): numérico (horas)

Tipo de valor (Android): numérico (horas)

3.2 Criterio 2: Tiempo de aprendizaje

Horas invertidas en el aprendizaje de los desarrolladores en iOS y Android. Se incluye en este criterio tanto el tiempo de aprendizaje en el entorno de desarrollo como el tiempo invertido en estudiar los lenguajes de programación necesarios.

Tipo de valor (iOS): numérico (horas)

Tipo de valor (Android): numérico (horas)

3.3 Criterio 3: Tiempo de instalación del emulador

Este criterio se corresponde con el total de tiempo invertido en la instalación de los emuladores del sistema correspondiente para evaluar qué sistema ha sido más rápido en este aspecto.

Tipo de valor (iOS): numérico (horas)

Tipo de valor (Android): numérico (horas)

3.4 Criterio 4: Soporte de entorno de desarrollo.

Se trata de ver si los entornos de desarrollo que se han utilizado para el desarrollo de los sistemas poseen un soporte en caso de que el programador necesite consultar alguna duda, o para resolver incidencias

Tipo de valor (iOS): booleano (Si/No)

Tipo de valor (Android): booleano (Si/No)

3.5 Criterio 5: Coste del entorno de desarrollo

Evaluar si se ha necesitado desembolsar en algún momento un capital a cambio de poder acceder al entorno de desarrollo (pago de licencia), o si, por el contrario, el acceso ha sido gratuito desde el primer momento (libre).

Tipo de valor (iOS): económico (euros)

Tipo de valor (Android): económico (euros)

3.6 Criterio 6: Coste del emulador

Al igual que el Criterio 5, evaluar el coste económico que han conllevado las pruebas en los emuladores utilizados en el proceso.

Tipo de valor (iOS): económico (euros)

Tipo de valor (Android): económico (euros)

3.7 Criterio 7: Calidad del sistema

Se trata de evaluar la calidad del sistema creado por los desarrolladores en función de sistemas existentes en el mercado, es decir, realizar una comparación de todos los aspectos de cada aplicación para ver en cuál es mejor cada una.

Tipo de valor (iOS): intervalo [0,10]

Tipo de valor (Android): intervalo [0, 10]

3.8 Criterio 8: Eficiencia del entorno de desarrollo

Se ha de evaluar la eficiencia de los emuladores para ver aspectos como cuál ha sido el más rápido en el arranque o el más rápido en la compilación. Se valorarán también aspectos como los lenguajes disponibles y comodidades para el desarrollador

Tipo de valor (iOS): intervalo [0,10]

Tipo de valor (Android): intervalo [0, 10]

3.9 Criterio 9: Eficiencia del emulador

Al igual que el Criterio 8, se ha de comprobar la eficiencia en el emulador, valorando la rapidez de arranque, la versión del sistema virtual o la posibilidad de pausar la emulación en un estado en concreto

Tipo de valor (iOS): intervalo [0,10]

Tipo de valor (Android): intervalo [0, 10]

3.10 Criterio 10: Lenguajes de programación

Este criterio consiste en analizar los lenguajes de programación que se han necesitado para el desarrollo de las aplicaciones en cada sistema, evaluando tanto el número como la dificultad para realizar la implementación correcta en cada uno.

Tipo de valor (iOS): intervalo [0,10]

Tipo de valor (Android): intervalo [0, 10]

3.11 Criterio 11: Distribución

Este criterio consiste en analizar la forma de distribución de la aplicación. Se medirá por un booleano, siendo No que no se ha conseguido distribuir, y Sí que la distribución ha ido satisfactoria.

Tipo de valor (iOS): booleano (no/si)

Tipo de valor (Android): booleano (no/si)

3.12 Criterio 12: Instalación de la aplicación en un dispositivo

Este criterio consiste en analizar la facilidad de instalar en el dispositivo, ya sea móvil o Tablet. Se analizará mediante un intervalo, siendo 0 muy difícil y 10 muy fácil.

Tipo de valor (iOS): intervalo [0,10]

Tipo de valor (Android): intervalo [0, 10]

3.13 Criterio 13: Tiempo de desarrollo completo

Este criterio consiste en analizar el tiempo completo de desarrollo de la aplicación entera, excluyendo el aprendizaje para poder realizar la implementación. Se medirá en horas.

Tipo de valor (iOS): numérico (horas)

Tipo de valor (Android): numérico (horas)

3.14 Criterio 14: Tiempo de instalación del programa

Este criterio consiste en analizar el tiempo que se ha tardado en descargar el programa de desarrollo y la puesta en marcha del mismo, dado que es tiempo que se pierde antes de empezar a programar.

Tipo de valor (iOS): numérico (minutos)

Tipo de valor (Android): numérico (minutos)

3.15 Criterio 15: Tiempo de inicio del programa

Este criterio consiste en analizar el tiempo que se ha tardado desde que abres el programa hasta que este sincroniza todo para empezar a trabajar.

Tipo de valor (iOS): numérico (segundos)

Tipo de valor (Android): numérico (segundos)

3.16 Criterio 16: Documentación sobre el lenguaje de programación

Este criterio consiste en analizar cómo es de fácil encontrar información sobre el lenguaje de programación sin contar con la documentación oficial.

Tipo de valor (iOS): intervalo [0,10]

Tipo de valor (Android): intervalo [0, 10]

4. Proyecto de implementación de un prototipo del sistema utilizando Android

En este apartado se desarrollará de forma clara y concisa todo lo necesario para explicar el desarrollo desde el principio de la aplicación de Android, así posteriormente, se explicará cómo debe instalarse y su uso.

4.1 Documentación de diseño

El diseño de la aplicación, se ha hecho anterior a la implementación usando la herramienta online Cacoo.com, ya que nos daba la posibilidad de realizarlo de forma gratuita, y podías elegir el modelo que más se adecuaba a tu futuro proyecto. El diseño se ha cumplido a la perfección, y cumple todos los requisitos que le incumbían quedando de la siguiente manera.

Como puede apreciarse en la imagen, tiene un diseño minimalista, es decir sencillo. Dispone de una imagen en la parte centro/inferior de la pantalla, que cambiará dependiendo del tiempo que haga, así podrá ser un sol, un paraguas con lluvia o dos tipos de nubes, una para cuando este nublado y otra para cuando haga sol y nubes. Justo encima de la imagen se encuentra la temperatura en grados centígrados e inmediatamente encima podemos ver el nombre de la ciudad de la que es el tiempo. En nuestro caso siempre será el tiempo de Madrid. Para finalizar en la esquina inferior

derecha encontramos el botón de actualizar, que, al pinchar en este, cambiará la temperatura o la previsión dependiendo el caso, siempre en el intervalo de una hora, como mínimo.

4.2 Documentación de construcción

La construcción del prototipo para la plataforma móvil Android ha supuesto la ejecución de dos fases de desarrollo de forma paralela. Por un lado la implementación del entorno gráfico mediante el cual interactúa el usuario y por otro lado la funcionalidad lógica de la aplicación.

Interfaz de usuario

Android Studio ofrece un completo arsenal de herramientas que posibilitan la creación de una interfaz de usuario en base a pocas iteraciones. Además es posible complementar dichas herramientas con un editor XML a fin de poder acceder a propiedades avanzadas de diseño.

La interfaz se compone por un conjunto de elementos clave que permiten mostrar los datos definidos en el apartado de requisitos, siendo los mismos definidos a continuación.

- **TextView:** Se definen dos elementos *TextView*, los cuales permiten mostrar los datos de la ciudad objetivo así como de la temperatura relacionada al mismo. *Anexo 1.1*
- **ImageView:** Se define un elemento *ImageView*, el cual permite mostrar aquella imagen correspondiente al pronóstico obtenido. *Anexo 1.2*
- **ImageButton:** Se define un elemento *ImageButton*, el cual permite el acceso a la función de recarga de datos. *Anexo 1.3*

Lógica de aplicación

Android Studio es un Entorno de Desarrollo Integrado (IDE) basado en el lenguaje de programación *Java*, lo que define una metodología de desarrollo basada en objetos.

La implementación del prototipo sobre la plataforma móvil Android ha sido ejecutada en torno a tres bloques de desarrollo, las cuales suponen la introducción de nuevas estructuras y métodos requeridos por los bloques consecutivos. Dando lugar a una metodología de desarrollo en cascada.

- **1º Bloque de desarrollo:** Supone la generación de aquellos objetos básicos requeridos para el almacenamiento y gestión de los datos de pronóstico del tiempo recibidos a través de la red.
    - **Forecast:** Objeto en el que se almacenan los datos relacionados con una instancia de predicción. *Anexo 2*
- **2º Bloque de desarrollo:** Supone la generación de objetos que representan estructuras avanzadas para la obtención y procesado de datos.
    - **XMLReader:** Objeto cuyo propósito supone el procesamiento de un archivo XML, extrayendo aquella información precisa y almacenándola en un *ArrayList* de *Forecast*. *Anexo 3*
    - **Best:** Objeto cuyo propósito supone la extracción de aquel objeto *Forecast* que mejor se adapta a la hora obtenida del sistema. *Anexo 4*
- **3º Bloque de desarrollo:** Supone la integración de los dos anteriores bloques a través de una clase principal *Main*, permitiendo mostrar los datos de interés al usuario a través de la interfaz gráfica y siendo necesario para ello la incorporación de dos métodos adicionales esenciales. *Anexo 5*

- **updatePicture(Forecast forecast):** Método gracias al cual se define aquel icono que mejor representa la descripción del pronóstico seleccionado por el sistema. *Anexo 6*
- **update():** Método gracias al cual se realiza la llamada al método *getValues()* responsable de la recuperación de datos de pronóstico a través de internet. *Anexo 5*

Finalmente se han de establecer los permisos requeridos sobre el sistema Android anfitrión, en este caso, únicamente será necesario el permiso sobre internet. Ya que los datos son recogidos a través de redes móviles o WiFi. *Anexo 7*

Anexos

Anexo 1.1

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Null"
    android:id="@+id/ciudad"
    android:textColor="#ffffff"
    android:textSize="50dp"
    android:textAlignment="center"
    android:layout_marginTop="10dp"/>
```

Anexo 1.2

```
<ImageView
    android:layout_width="250dp"
    android:layout_height="250dp"
    android:id="@+id/estado"
    android:layout_above="@+id/temperatura"
    android:layout_centerHorizontal="true"
    android:layout_gravity="center_horizontal" />
```

Anexo 1.3

```
<ImageButton
    android:layout_width="30dp"
    android:layout_height="30dp"
    android:id="@+id/actualizar"
    android:background="@drawable/update"
    android:layout_gravity="right"
    android:layout_marginTop="5dp"
    android:onClick="update" />
```

Anexo 2

```
private String city;
private Date date;
private int temperature;
private String text;
```

Anexo 3 (*zoom*).

```
public class XMLReader {
    public static ArrayList<Forecast> getForecast(URL url){
        ArrayList<Forecast> arrayList = new ArrayList<>();

        try {
            DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
            Document document = documentBuilder.parse(url.openStream());

            //Normalize
            document.getDocumentElement().normalize();

            //City
            Element elementoLocalidad =(Element) document.getElementsByTagName("localidad").item(0);
            String localidad = elementoLocalidad.getElementsByTagName("nombre").item(0).getTextContent();

            //Data
            NodeList listOfHoras = document.getElementsByTagName("hora");
            for (int i = 0; i< listOfHoras.getLength(); i++){
                Element element = (Element) listOfHoras.item(i);

                //Forecast data
                Forecast forecast = new Forecast();
                forecast.setCity(localidad);
                forecast.setDate(new SimpleDateFormat("yyyy-MM-dd/hh:mm").parse(element.getElementsByTagName("fecha").item(0).getTextContent() + "/" + element.getElementsByTagName("hora_datos").item(0).getTextContent()));
                forecast.setTemperature(Integer.parseInt(element.getElementsByTagName("temperatura").item(0).getTextContent()));
                forecast.setText(element.getElementsByTagName("texto").item(0).getTextContent());
                arrayList.add(forecast);

            }

        }catch(Exception e){}
        return arrayList;

    }
}
```

Anexo 4

```
public class Best {
    public static Forecast getBestForecast(ArrayList<Forecast> arrayList){
        Calendar calendar = Calendar.getInstance();
        int i = 0;
        while(i < arrayList.size()){
            if(arrayList.get(i).getDate().getHours() == calendar.get(Calendar.HOUR_OF_DAY)){
                return arrayList.get(i);
            }
            else i++;

        }
        return null;

    }

}
```

Anexo 5

```
private void getValues(){
    TextView city = (TextView)findViewById(R.id.ciudad);
    TextView temperature = (TextView)findViewById(R.id.temperatura);

    try{
        URL url = new URL("http://xml.tutiempo.net/xml/3768.xml");
        this.arrayList = XMLReader.getForecast(url);
        city.setText(Best.getBestForecast(this.arrayList).getCity());
        temperature.setText(""+Best.getBestForecast(this.arrayList).getTemperature()+"ºC");
        this.updatePicture(Best.getBestForecast(this.arrayList));
    }catch(Exception e){};

}
```

Anexo 6

```
private void updatePicture(Forecast forecast){
    ImageView imageView = (ImageView)findViewById(R.id.estado);

    if(forecast.getText().contains("lluvia")) imageView.setImageResource(R.drawable.lluvia);
    else if(forecast.getText().contains("cubierto")) imageView.setImageResource(R.drawable.cubierto);
    else if(forecast.getText().contains("nuboso")) imageView.setImageResource(R.drawable.nuboso);
    else imageView.setImageResource(R.drawable.soleado);

}
```

Anexo 7

```
<uses-permission android:name="android.permission.INTERNET" />
```

4.3 Documentación de pruebas

Prueba	Problema	Solución
La aplicación muestra una determinada imagen en función del tiempo que haga en ese momento para la ciudad de Madrid.	En un principio, al introducir las diferentes imágenes en la aplicación y el código necesario para que la imagen fuera cambiando encontramos el problema de que se superponía la nueva imagen con la que había anteriormente.	El problema venia de que al crear el elemento ImagenView se dejó una de las imágenes que queríamos usar como fija. La solución fue eliminar esa imagen fija y desde el código hacer que si la imagen tiene que cambiar se quite anteriormente la que se había puesto anteriormente.
Uso de los permisos en Android	Nos encontramos con problemas de los permisos al realizar el desarrollo del código.	Se tuvo que modificar el archivo “info” de la aplicación.
Asignación por horas	No se conseguía una correcta asignación por horas.	Al igual que en el punto anterior la solución encontrada fue que había que depurar el código.
Ejecución del Main() en Android.	Problemas al ejecutar el Main() debido a que no era un método asíncrono.	La solución fue añadir dos líneas de código (estas se encuentran en el Main() del prototipo).
Fallo con el emulador de Android Studio	Problemas a la hora de ejecutar la aplicación desde dicho emulador.	Se fue probando la aplicación conectando el teléfono móvil al ordenador con una cable USB.

4.4 Documentación de instalación

Con Android tenemos dos opciones, o bien la instalamos en el dispositivo o con el emulador.

- Para instalar la App en el dispositivo, sólo hace falta descargarse o meter en el dispositivo la APK de la aplicación, que se encuentra en el siguiente enlace https://drive.google.com/file/d/0B2xNdH5lw-ZFTkczMGVjTlE4eFU/view?usp=sharing
Una vez lo tenemos descargado, pinchando en la descarga se instala automáticamente y aparecerá el icono con el nombre en el menú del dispositivo. Al pinchar en el icono, se abrirá directamente la aplicación y podremos hacer uso de ella.

- Por otro lado, si instalamos Android Studio, que es un programa gratuito de desarrollo para Android, abrimos el proyecto y con el emulador correrá la aplicación y podemos hacer las mismas funciones que si la tenemos en el dispositivo móvil o Tablet. El problema de hacerlo de esta forma, es que el emulador en muchos casos da fallos, y por lo general es lento. Por lo que es mejor meter la APK directamente en el dispositivo móvil o Tablet.

4.5 Manual de usuario

Es una aplicación muy sencilla, y tal y como dice uno de los requisitos el usuario es capaz de aprender a utilizarla en menos de un minuto.

Primero de todo debes disponer de conexión a internet, dado que, si esto no es así, saldrán valores “null”, esto se soluciona conectando el dispositivo a la red, bien sea wifi

o mediante datos móviles, y dándole al botón de actualizar que se encuentra en la esquina inferior derecha. En caso que desde un principio se tenga conexión a la red, simplemente es tocar el icono de la aplicación y automáticamente saldrá la ciudad (Madrid) con la temperatura exacta que hace y una imagen que muestra cuál es la predicción, así esta será:

- Un sol, si el tiempo es soleado.
- Un paraguas con gotas de lluvia, si llueve.
- Una nube con un sol en la parte izquierda, si es parcialmente nubloso.
- Tres nubes, si el tiempo es nublado denso.

Si en el mismo momento pulsamos el botón actualizar, no va a pasar visiblemente nada, dado que sólo se actualiza cada hora, dado que es cuando se actualiza el XML.

5. Proyecto de implementación de un prototipo del sistema utilizando iOS

En este apartado se dará toda la información de la aplicación creada para iOS.

5.1 Documentación de diseño

Como en el caso de Android, el diseño de la aplicación, se ha hecho anterior a la implementación usando la herramienta online Cacoo.com, ya que nos daba la posibilidad de realizarlo de forma gratuita, y podías elegir el modelo que más se adecuaba a tu futuro proyecto. El diseño se ha cumplido a la perfección, y cumple todos los requisitos que le incumbían quedando de la siguiente manera.

Como puede apreciarse en la imagen, tiene un diseño minimalista, es decir sencillo. Dispone de una imagen en la parte centro/inferior de la pantalla, que cambiará dependiendo del tiempo que haga, así podrá ser un sol, un paraguas con lluvia o dos tipos de nubes, una para cuando este nublado y otra para cuando haga sol y nubes. Justo encima de la imagen se encuentra la temperatura en grados centígrados e

inmediatamente encima podemos ver el nombre de la ciudad de la que es el tiempo. En nuestro caso siempre será el tiempo de Madrid. Para finalizar en la esquina inferior derecha encontramos el botón de actualizar, que, al pinchar en este, cambiará la temperatura o la previsión dependiendo el caso, siempre en el intervalo de una hora, como mínimo.

5.2 Documentación de construcción

La construcción del prototipo para la plataforma móvil iOS ha implicado ciertas variaciones en torno a la metodología de desarrollo con respecto a la empleada en el desarrollo del prototipo para la tecnología móvil Android.

Estas variaciones se relacionan con la filosofía *Swift*, en la que se recomienda la codificación de funcionalidades básicas y ampliamente relacionadas con la interfaz en una clase *ViewController*.

La ejecución del proceso de implementación se ha realizado siguiendo el modelo de dos fases empleado en el prototipo desarrollado para la plataforma móvil Android. En el que se implementará por separado y en paralelo la interfaz de usuario y la funcionalidad lógica de la aplicación.

Interfaz de usuario

Xcode ofrece un completo arsenal de herramientas las cuales posibilitan la implementación de las distintas interfaces de cualquier aplicación sin importar la complejidad de la misma. Además, permite la definición de relaciones entre las distintas interfaces, así como otros eventos.

La interfaz se compone por un conjunto de elementos clave que permiten mostrar los datos definidos en el apartado de requisitos, siendo los mismos definidos a continuación.

- **Label:** Se definen dos elementos *Label*, los cuales permiten mostrar los datos de la ciudad objetivo, así como de la temperatura relacionada al mismo.
- **ImageView:** Se define un elemento *ImageView*, el cual permite mostrar aquella imagen correspondiente al pronóstico obtenido.
- **Button:** Se define un elemento *Button*, el cual permite el acceso a la función de recarga de datos.

La definición de imágenes predefinidas para algunos elementos de interfaz como es el caso de los botones se ha de realizar en base a la codificación.

Lógica de aplicación

Xcode es un Entorno de Desarrollo Integrado (IDE) basado en los lenguajes de programación *Swift* y *Objetive-c*, lo que define una metodología de desarrollo que gira en torno a la clase *ViewControler* al tratarse de una implementación sencilla.

La implementación del prototipo sobre la plataforma móvil iOS supone la ejecución de tres bloques de desarrollo al igual que sucedía en la implementación del anterior prototipo, implicando nuevamente un desarrollo en cascada.

- **1º Bloque de desarrollo:** Supone la implementación de *NSXMLParser*, responsable de la extracción y almacenamiento de la información requerida del

archivo origen de datos XML. A diferencia del objeto *XMLReader* implementado en *Java*, *NSXMLParser* está integrado en el lenguaje de programación por lo que solo es necesaria la personalización de los métodos que lo componen, adaptándolo a nuestro problema concreto. *Anexo 1*

- **2º Bloque de desarrollo:** Supone la generación de aquellos métodos que permiten definir el pronóstico de tiempo más se adapta según la hora obtenida del sistema. Así como aquel que permitirá definir el icono que mejor representa la descripción del pronóstico seleccionado por el sistema. *Anexo 2*
- **3º Bloque de desarrollo:** Supone la integración de la funcionalidad materializada en los anteriores bloques de desarrollo, permitiendo mostrar los datos de interés al usuario a través de la interfaz gráfica siendo requerido adicionalmente aquel método que permite la actualización de los datos de pronóstico. *Anexo 2*

Finalmente se han de establecer aquellos permisos requeridos para el correcto funcionamiento de la aplicación en el sistema anfitrión, en este caso, únicamente será necesario habilitar el permiso *App Transport Security Settings* el cual permite la capacidad de gestión de aquellos datos recibidos a través de la red. *Anexo 3*

Anexos

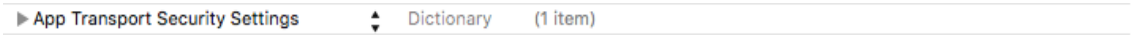
Anexo 1 (*zoom*)

```
func parser(parser: NSXMLParser, didStartElement elementName: String, namespaceURI: String?, qualifiedName qName: String?, attributes attributeDict:
    [String : String])
{
    element = elementName
    if (elementName as NSString).isEqualToString("hora")
    {
        elements = NSMutableDictionary()
        city = NSMutableString()
        date = NSMutableString()
        temperature = NSMutableString()
        text = NSMutableString()

        elements = [:]
        city = ""
        date = ""
        temperature = ""
        text = ""
    }

}

func parser(parser: NSXMLParser, didEndElement elementName: String, namespaceURI: String?, qualifiedName qName: String?)
{
    if (elementName as NSString).isEqualToString("hora") {
        if !date.isEqual(nil) {
            elements.setObject(date, forKey: "hora_datos")
        }
        if !temperature.isEqual(nil) {
            elements.setObject(temperature, forKey: "temperatura")
        }
        if !text.isEqual(nil) {
            elements.setObject(text, forKey: "texto")
        }
        data.addObject(elements)

    }

}

func parser(parser: NSXMLParser, foundCharacters string: String)
{
    if element.isEqualToString("hora_datos") {
        date.appendString(string)
    } else if element.isEqualToString("temperatura") {
        temperature.appendString(string)
    } else if element.isEqualToString("texto") {
        text.appendString(string)
    }

}
```

Anexo 2

```
//Data to interface
func setData()
{
    let currentDateTime = NSDate()
    let userCalendar = NSCalendar.currentCalendar()
    let requestedComponents: NSCalendarUnit = [NSCalendarUnit.Year, NSCalendarUnit.Month, NSCalendarUnit.Day, NSCalendarUnit.Hour,
        NSCalendarUnit.Minute, NSCalendarUnit.Second]
    let dateTimeComponents = userCalendar.components(requestedComponents, fromDate: currentDateTime)

    for i in 0...(data.count-1){
        if(data.objectAtIndex(i).valueForKey("hora_datos")?.integerValue == dateTimeComponents.hour){
            storyboardCity.text = "Madrid"
            storyboardTemperature.text = (data.objectAtIndex(i).valueForKey("temperatura") as? String)! + "ºC"

            if((data.objectAtIndex(i).valueForKey("texto") as? String)!.rangeOfString("lluvia") != nil){
                storyboardText.image = UIImage(named: "images/lluvia.png")
            } else if((data.objectAtIndex(i).valueForKey("texto") as? String)!.rangeOfString("cubierto") != nil){
                storyboardText.image = UIImage(named: "images/cubierto.png")
            } else if((data.objectAtIndex(i).valueForKey("texto") as? String)!.rangeOfString("nuboso") != nil){
                storyboardText.image = UIImage(named: "images/nuboso.png")
            } else{
                storyboardText.image = UIImage(named: "images/soleado.png")
            }
            break

        }

    }

}
```

Anexo 3



5.3 Documentación de pruebas

Prueba	Problema	Solución
Uso de los permisos en iOS.	Nos encontramos con problemas de los permisos al realizar el desarrollo del código.	Se tuvo que modificar el archivo “info” de la aplicación.
XMLPARSE en iOS.	Hubo varios problemas con el XMLPARSE en la aplicación de iOS.	La solución fue que había que depurar el código hasta que se consiguió el correcto funcionamiento.
Asignación por horas.	No se conseguía una correcta asignación por horas.	Al igual que en el punto anterior la solución encontrada fue que había que depurar el código.
Interfaz en iOS.	Colocación inadecuada de los elementos de la interfaz para la aplicación de iOS.	Se expandió cada elemento que queríamos usar en la interfaz de tal forma que ocupase todo el ancho de la pantalla para que posteriormente se centrase.

5.4 Documentación de instalación

Dado que no tenemos licencia de Apple Store, no hay forma posible de instalar la aplicación en el dispositivo, a no ser que este esté pirateado. Existen emuladores que instalas en el ordenador, y puede correr la aplicación de la forma que la haría en el dispositivo móvil o Tablet, el problema vuelve a ser el mismo, solo existe la posibilidad de descargarla desde la Apple Store, aun siendo con el emulador. Por tanto, la única

opción es si se tiene un ordenador MAC, instalarse el programa de desarrollo, que en nuestro caso ha sido xCode, y hacer correr la aplicación desde el emulador.

5.5 Manual de usuario

Dado que la aplicación es exactamente la misma que la de Android, el funcionamiento de la misma es idéntico. Es una aplicación muy sencilla, y tal y como dice uno de los requisitos el usuario es capaz de aprender a utilizarla en menos de un minuto.

Primero de todo debes disponer de conexión a internet, dado que, si esto no es así, saldrán valores “null”, esto se soluciona conectando el dispositivo a la red, bien sea wifi o mediante datos móviles, y dándole al botón de actualizar que se encuentra en la esquina inferior derecha. En caso que desde un principio se tenga conexión a la red, simplemente es tocar el icono de la aplicación y automáticamente saldrá la ciudad (Madrid) con la temperatura exacta que hace y una imagen que muestra cuál es la predicción, así esta será:

- Un sol, si el tiempo es soleado.
- Un paraguas con gotas de lluvia, si llueve.
- Una nube con un sol en la parte izquierda, si es parcialmente nubloso.
- Tres nubes, si el tiempo es nublado denso.

Si en el mismo momento pulsamos el botón actualizar, no va a pasar visiblemente nada, dado que sólo se actualiza cada hora, dado que es cuando se actualiza el XML.

6. Comparación de las dos implementaciones

A continuación, se hará el estudio de los criterios que se han decidido analizar en ambas tecnologías.

6.1 Evaluación de los criterios en la implementación usando Android

CRITERIO	EVALUACIÓN
Criterio 1: Tiempo de creación de la interfaz de usuario	3h y 45 min*
Criterio 2: Tiempo de aprendizaje	14 horas. No teníamos conocimientos previos del desarrollo en esta tecnología.
Criterio 3: Tiempo de instalación del emulador	0 horas. Con la instalación del programa de desarrollo viene integrado el emulador. Si es cierto que el sistema de desarrollo lleva tiempo en instalarse. También hemos utilizado un dispositivo Android que ha hecho de emulador de la aplicación.

CRITERIO	EVALUACIÓN
Criterio 4: Soporte del entorno de desarrollo	Si. Android Silver es un soporte técnico de calidad para dispositivos con sistemas Android
Criterio 5: Coste del entorno de desarrollo	Gratuita. La tecnología para desarrollar una aplicación en Android es gratuita. Si decidiéramos subir la aplicación a la plataforma correspondiente tendría un coste único.
Criterio 6: Coste de emulador	Gratuito. El coste del emulador es gratuito ya que con la descarga del sistema de desarrollo nos proporciona un emulador para la prueba de la aplicación.
Criterio 7: Calidad del sistema	7
Criterio 8: Eficiencia del entorno de desarrollo	4
Criterio 9: Eficiencia del emulador	3 ya que el tiempo de carga del emulador es muy elevado.
Criterio 10: Lenguajes de programación	7 EL lenguaje de programación es más sencillo ya que estábamos más familiarizados y es un lenguaje menos estricto.
Criterio 11: Distribución	Sí, dado que se puede distribuir mediante cualquier repositorio.
Criterio 12: Instalación de la aplicación en un dispositivo	10, es solo descargar el ejecutable en el dispositivo.
Criterio 13: Tiempo completo de desarrollo.	16 horas.
Criterio 14: Tiempo de instalación del programa	15’
Criterio 15: tiempo de inicio del programa	1-2’
Criterio 16: Documentación sobre el lenguaje de programación	8, hay mucha información por lo que se encuentra fácilmente.

6.2 Evaluación de los criterios en la implementación usando iOS

CRITERIO	EVALUACIÓN
Criterio 1: Tiempo de creación de la interfaz de usuario	3h y 30 min*
Criterio 2: Tiempo de aprendizaje	14 horas. No teníamos conocimientos previos del desarrollo en esta tecnología.
Criterio 3: Tiempo de instalación del emulador	0 Horas. Como emulador hemos utilizado un dispositivo iOS para la puesta en marcha de esta tecnología.

CRITERIO	EVALUACIÓN
Criterio 4: Soporte del entorno de desarrollo	Si, iOS dispone de un servicio Web que proporciona soporte técnico de manera genérica.
Criterio 5: Coste del entorno de desarrollo	Gratuita. La tecnología para desarrollar una aplicación en iOS es gratuita. El coste esta al querer publicar dicha tecnología en la plataforma correspondiente y este coste sería anual.
Criterio 6: Coste de emulador	Gratuito. En nuestro caso el emulador ha sido gratuito ya que disponíamos de un dispositivo iOS.
Criterio 7: Calidad del sistema	7
Criterio 8: Eficiencia del entorno de desarrollo	7
Criterio 9: Eficiencia del emulador	6 el tiempo del emulador en el arranque es más rápido que con el de Android.
Criterio 10: Lenguajes de programación	5. El desarrollo de la aplicación con esta tecnología es complejo y no estábamos familiarizados con esta tecnología.
Criterio 11: Distribución	No, dado que hace falta pagar una licencia para poder distribuirla.
Criterio 12: Instalación de la aplicación en un dispositivo	3, dado que si no está en Apple Store no puedes descargártela y ejecutarla.
Criterio 13: Tiempo completo de desarrollo.	23 horas.
Criterio 14: Tiempo de instalación del programa	5-10’
Criterio 15: tiempo de inicio del programa	15-30”
Criterio 16: Documentación sobre el lenguaje de programación	3, dado que es difícil encontrar documentación adicional a la oficial.

* El tiempo de interfaz ha sido menor en iOS ya que hemos comenzado por la interfaz de usuario en la tecnología de Android y hemos intentado adaptarlo con sus modificaciones al sistema de iOS.

7. Comparación de la implementación de las tecnologías

A continuación, se incluye la tabla de comparativa de las dos tecnologías en base a los criterios seleccionados y analizados en apartados anteriores.

CRITERIOS	Android	iOS	COMENTARIOS
Criterio 1: Tiempo de creación de la interfaz de usuario	*3H 45min*	*3H 30 min*	*A pesar de que ambos tiempos de desarrollo son similares, en el sistema de iOS ha sido inferior, y por tanto, preferible*
Criterio 2: Tiempo de aprendizaje	*9H*	*14H*	*Las horas invertidas en aprender lo necesario han sido menores durante el desarrollo de Android.*
Criterio 3: Tiempo de instalación del emulador	*0H*	*0H*	*Al disponer de un dispositivo iOS, no se ha hecho necesario el uso de ningún emulador.*
Criterio 4: Soporte del entorno de desarrollo	*Sí*	*Sí*	*Ambos sistemas disponen de un soporte para desarrolladores, por lo que ninguno es destacable frente al otro en este aspecto.*
Criterio 5: Coste del entorno de desarrollo	*0€*	*0€*	*El entorno de desarrollo es gratuito en ambos sistemas, pero si se tiene en cuenta el coste de una cuenta de desarrollador, Android es preferible en este criterio.*
Criterio 6: Coste de emulador	*0€*	*0€*	*Al igual que el criterio 4, ambos sistemas son igualmente seleccionables en este punto.*
Criterio 7: Calidad del sistema	*7 (sobre 10)*	*7 (sobre 10)*	*Ambos sistemas poseen unos atributos que le hacen tener una calidad de 7 sobre 10, por lo que cada uno posee sus propias ventajas.*

CRITERIOS	Android	iOS	COMENTARIOS
Criterio 8: Eficiencia del entorno de desarrollo	*4 (sobre 10)*	*7 (sobre 10)*	*En cuanto a la eficiencia en el desarrollo, iOS ha demostrado ser más rápido y ofrecer alguna comodidad más.*
Criterio 9: Eficiencia del emulador	*3 (sobre 10)*	*6 (sobre 10)*	*Y en la eficiencia del emulador es igualmente destacable iOS.*
Criterio 10: Lenguajes de programación	*7 (sobre 10)*	*5 (sobre 10)*	*Sin embargo, en cuanto a los lenguajes de programación utilizados durante el desarrollo, los programadores han preferido el entorno de Android puesto que estaban familiarizados con ellos.*
Criterio 11: Distribución	*Sí*	*No*	*Android permite una distribución libre, mientras que iOS no, solo mediante su tienda oficial (Apple Store)*
Criterio 12: Instalación de la aplicación en un dispositivo	*10*	*3*	*Mientras que Android permite instalarla solo descargando el ejecutable, volvemos al caso de que iOS solo deja bajarla de su web, por lo que lo hace más complejo.*
Criterio 13: Tiempo completo de desarrollo.	*16*	*23*	*En Android el tiempo de desarrollo es más bajo, puede haber sido porque es un entorno más amigable, y el lenguaje es java.*
Criterio 14: Tiempo de instalación del programa	*15'*	*5-10'*	*El tiempo depende de la velocidad de internet, pero para poder instalar y descargar Android Studio, tienes que descargarte también los JDK, por lo que es más lento que iOS.*
Criterio 15: tiempo de inicio del programa	*1-2'*	*15-30"*	*Android Studio es mucho más lento en sincronizar los datos para arrancar a programar.*
Criterio 16: Documentación sobre el lenguaje de programación	*8*	*3*	*Dado que Android es un lenguaje más estándar, encontramos mucha documentación a parte de la oficial, mientras que de iOS no se encuentra tan fácilmente.*

8. Conclusiones

En función de lo que ha quedado reflejado en la tabla anterior, Android posee una serie de características que hacen que sea una gran opción en el desarrollo de aplicaciones. Sin embargo, el desarrollo en iOS ha demostrado ser más eficiente y rápido una vez se conocen todos los detalles y se han realizado las pruebas necesarias en el sistema desarrollado. Cabe destacar que Android es más versátil que iOS, dado que en cualquier ordenador puedes instalar Android Studio, mientras que, para poder desarrollar en iOS, es necesario un equipo MAC, además a la hora de la distribución, Android permite una distribución libre, mientras que iOS solo por la tienda oficial, para la cual debes tener una licencia para poder subirla y distribuir dicha app.