Contenido

1. Autores del trabajo, planificación y entrega

1.1 Autores

En este apartado se debe indicar el número de grupo y los nombres de los autores, poniendo en primer lugar al coordinador del grupo.

1.2 Planificación

En este apartado se debe incluir un enlace (URL) compartido a la planificación del trabajo utilizando una herramienta online de diagramación Gantt (por ejemplo, GanttPro, versión gratuita).

Hay que tener en cuenta que cada participante del grupo debe tener asignadas tareas que sumen al menos 45 horas. El peso de este trabajo en la calificación total de la asignatura es de un 30%, por tanto requiere de una dedicación de 45 horas del total de 150 horas de la asignatura.

1.3 Entrega

En este apartado debe incluirse un enlace (URL) a un repositorio en GitHub o en BitBucket creado para el trabajo.

En dicho repositorio debe encontrarse, al menos los siguientes archivos en la rama máster:

- Informe del trabajo: con el nombre TG3_final.docx
- Presentación del trabajo: TG3_final.pptx
- Prototipos obtenidos implementando cada una de las tecnologías (deben incluir el código fuente y todos los archivos necesarios para la instalación y uso de cada prototipo):
    - PrototipoTecnologiaA_final.zip (o .rar)
    - PrototipoTecnologiaB_final.zip (o .rar).

Dichos archivos serán los que se tendrán en cuenta para la calificación del trabajo.

2. Requisitos del prototipo a implementar

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

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 este apartado. Si se van a crear dos equipos de trabajo, uno para cada prototipo, el contenido de este apartado es lo que han de compartir ambos equipos como punto de partida.

Cuanto más detallados sean los requisitos, mayor será la precisión en la comparación que se realizará al final del trabajo. Se trata de conseguir dos prototipos con igual funcionalidad, pero utilizando diferentes tecnologías.

Se puede dar libertad a los equipos de desarrollo en cuanto al diseño, pero la funcionalidad debe ser lo más parecida posible. Por ejemplo, no es necesario que los colores utilizados en las pantallas sean exactamente los mismos en ambos prototipos, a no ser que los miembros del grupo lo hayan decidido así, en cuyo caso, esos detalles de colores deben incluirse en el catálogo de requisitos, para que ambos equipos los cumplan.

2.1 Requisitos funcionales

Los requisitos funcionales deben ser los mismos para las dos implementaciones.

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

REQ.	**DESCRIPCIÓN**
RF01	….
RF02	….

2.2 Otros requisitos

Se pueden incluir aquí otros requisitos para el prototipo que no puedan considerarse como funcionales. Por ejemplo, requisitos de datos, de seguridad, de interfaz de usuario, de rendimientos, etc.

Se puede dejar libertad

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

REQ.	**DESCRIPCIÓN**
R01	….
R02	….

3. Criterios de comparación en la implementación

En el trabajo TG2 se definieron criterios de comparación de las dos tecnologías a nivel teórico.

En este trabajo hay que definir criterios para la comparación de la implementación de las tecnologías en la construcción del prototipo de sistema de ejemplo, cuyos requisitos son los establecidos en el apartado 2.

Se trata de criterios del tipo" "horas empleadas en el desarrollo del sistema", "velocidad de funcionamiento del sistema", "recursos necesarios", etc.

3.1 Criterio 1: Nombre del criterio

Por cada criterio hay que indicar el nombre, una breve descripción, y el tipo de valor a asignar al criterio.

Por ejemplo, si se comparan dos herramientas CASE realizar el diseño UML de un mismo sistema, un criterio podría ser:

Nombre del criterio: Tiempo de creación del diagrama de clases del sistema.

Descripción: Horas invertidas en la creación del diagrama de clases utilizando el editor de la herramienta.

Tipo de valor: Numérico (horas).

3.2 Criterio 2: Nombre del criterio

3.N Criterio N: Nombre del criterio

4. Proyecto de implementación de un prototipo del sistema utilizando la tecnología A

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología A, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

4.1 Documentación de diseño

Hay que incluir la descripción del diseño del prototipo, incluyendo diagramas, y el diseño de la interfaz de usuario.

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

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los caso de prueba.

4.4 Documentación de instalación

Descripción suficiente para que una persona que no ha participado en el proyecto pueda instalar el prototipo.

4.5 Manual de usuario

Descripción suficiente para que una persona que no ha participado en el proyecto pueda utilizar toda la funcionalidad que ofrece el prototipo. Que debe coincidir con los requisitos funcionales incluidos en el apartado 2.

5. Proyecto de implementación de un prototipo del sistema utilizando la tecnología B

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología B, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

5.1 Documentación de diseño

Hay que incluir la descripción del diseño del prototipo, incluyendo diagramas, y el diseño de la interfaz de usuario.

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

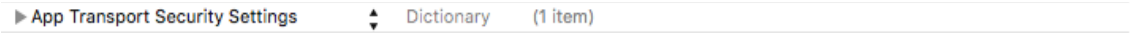
Anexo 1 (*zoom*)

```
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

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los caso de prueba.

5.4 Documentación de instalación

Descripción suficiente para que una persona que no ha participado en el proyecto pueda instalar el prototipo.

5.5 Manual de usuario

Descripción suficiente para que una persona que no ha participado en el proyecto pueda utilizar toda la funcionalidad que ofrece el prototipo. Que debe coincidir con los requisitos funcionales incluidos en el apartado 2.

6. Comparación de las dos implementaciones

Se trata de dar valores a los criterios de comparación definidos en el apartado 3 sobre la implementación de cada uno de los prototipos.

6.1 Evaluación de los criterios en la implementación usando la tecnología A

Debe incluir al menos una tabla con la siguiente estructura.

CRITERIO	**EVALUACIÓN**
Criterio 1	
Criterio 2	
…	
Criterio N	

Y algunos comentarios aclaratorios sobre aquellos criterios cuyo valor indicado en la tabla no sea suficiente para entenderlo.

6.2 Evaluación de los criterios en la implementación usando la tecnología B

7. Comparación de la implementación de las tecnologías

Debe incluir al menos una tabla resumen, en sección de página horizontal, cruzando los criterios y los valores de cada tecnología. Con una columna de comentarios sobre la comparación

CRITERIOS	TECNOLOGÍA A	TECNOLOGÍA B	COMENTARIOS
1			
2			
…			
N			

8. Conclusiones

A partir de la información incluida en el apartado 7 y de la experiencia al realizar el trabajo, el grupo debe estar en condiciones de manifestar su opinión sobre la implementación del sistema utilizando ambas tecnologías, y debe plasmarla en este apartado, indicando las ventajas e inconvenientes más relevantes de utilizar una u otra tecnología para implementar el sistema.

(Hay que cumplir la estructura básica indicada de secciones. Pero si se desea se pueden añadir otras secciones como anexos. Por ejemplo, alguna encuesta de opinión realizada sobre las tecnologías, etc.)