PROYECTO

DE

FIN DE GRADO

ÍNDICE

1. JUSTIFICACIÓN 3
2. OBJETIVOS 4
3. METODOLOGÍA DE DESARROLLO 5
4. TECNOLOGÍAS 6
5. PLANIFICACIÓN DEL PROYECTO 7

ESTIMACIÓN INICIAL DEL PROYECTO: 7

COSTES REALES DEL DESARROLLO DEL PROYECTO: 7

6. DESARROLLO Y EJECUCIÓN DEL PROYECTO 9
7. CONCLUSIONES 14
8. LÍNEAS DE INVESTIGACIÓN Y DESARROLLO FUTURAS 15

CONCLUSIONES CON RESPECTO A LO ANTERIOR 16

9. BIBLIOGRAFÍA 17

1. JUSTIFICACIÓN

Este es un proyecto final de grado superior de **Desarrollo de Aplicaciones Multiplataforma** que engloba todas las enseñanzas realizadas a lo largo del curso, es utilizado con el fin de aprender y seguir mejorando en el mundo del desarrollo de aplicaciones, no se hace con objetivos lucrativos, es meramente formativo, pero si se vendiera beneficiaría a las empresas que compraran dicha aplicación.

La aplicación está enfocada a empresas de transporte de paquetería para llevar la gestión de los empleados y de su trabajo, así como las ubicaciones donde ha estado y los clientes a los que ha atendido, para evitar tener que regular cada pedido y gestionar a los empleados mediante papel convencional. Un trabajador empezará su jornada laboral a una hora determinada y con esta aplicación podrá fichar y podrá registrar cada sitio donde ha ido mediante su ubicación GPS y los clientes que ha atendido en ese sitio, para que el jefe pueda llevar el control sobre los empleados y controlar cuantos clientes ha hecho y las horas que ha estado en ese día. De esta manera evitamos tener que declarar mediante un papel cada envío de paquete y cada cliente, todo quedará registrado en la aplicación y el jefe podrá consultarlo cuando quiera.

Es una aplicación que exige un conocimiento técnico de desarrollo elevado y si se buscara su ampliación en el futuro exigiría mucho tiempo de desarrollo. Si se realizara con un equipo de desarrollo especializado saldría rentable la aplicación porque hoy en día muchas empresas exigen que se le haga aplicaciones a medida de este estilo y son capaces de pagar una cantidad elevada de dinero debido a la reducción de los procesos de gestión internos de la misma empresa y el aprovechamiento de tiempo por parte del repartidor.

2. OBJETIVOS

Los objetivos principales son:

- Control del trabajo y clientes de los repartidores.
- Llevar una organización de las tareas de la empresa.
- Tener un medio con el cual el empresario poder interactuar para realizar labores de gestión y tener un control mínimo.
- Facilitar a los repartidores su labor de fichar en la empresa sin necesidad de acudir al centro de transportes.
- Poder reaccionar ante posibles infracciones del repartidor.
- Mantener un control de qué clientes no han recibido su paquete.

Lo que se pretende con el proyecto es cubrir las necesidades de una empresa de transporte, preferiblemente una que lleve paquetería para reducir los procesos de gestión de la empresa como llevar la cuenta de clientes, informe de clientes de empleados, de lugares y ubicaciones donde haya estado. Todo esto sin necesidad de realizar papeleo, todo informatizado lo que ayuda a la empresa a reducir costes y trabajo y aprovechar mejor el tiempo.

El objetivo mas claro sería reducir tiempos y costes tanto en material como en gestión.

Aparte de ayudar mayormente al que esté llevando la empresa haciéndole la vida mas fácil.

3. METODOLOGÍA DE DESARROLLO

Básicamente la metodología usada es en espiral debido a que hemos realizado múltiples entregas. Donde al principio llevaba unas fases de análisis, desarrollo y diseño en las cuales se desarrollan diferentes versiones incrementales, cada una tendrá mas funcionalidades. En este caso hemos realizado dos versiones, una que realicé meses antes y ahora que se pidió mas funcionalidad hemos vuelto a hacer sobre ese prototipo más funcionalidades y se podría seguir usando la misma metodología para hacer un producto que quede bien pulido al ser sus fases incrementales. Pudiendo volver a un paso anterior siempre que sea necesario y no hayamos hecho cosas correctamente o que el cliente nos lo eche atrás y tengamos que volver a una fase anterior.

De todas formas, voy a explicar como lo iba haciendo yo. Iba dibujando en papel cada pantalla y lo que tenía que implementar en esa pantalla, haciendo un análisis exhaustivo de toda la pantalla (incluyendo validaciones, variantes que podrían surgir, que tablas y campos de la base de datos se van a utilizar, que usuarios podrán acceder a esa pantalla, que resultados tiene que mostrar, si está relacionado con otras pantallas, interacción con el usuario), luego pasaba el dibujo de papel a la pantalla de diseño, después ponía nombre a los controles de la pantalla y por último ya me ponía a implementar la lógica de negocio realizando todo lo establecido en el papel del análisis. Finalmente, el paso más importante que he considerado es realizar las pruebas pertinentes porque de ahí han salido muchos errores que hemos tenido que solventar. Simplemente en otro papel aparte he escrito los roles con los que había que probar la funcionalidad y los diferentes casos de prueba que había por ejemplo en la pantalla de login el caso de prueba es validar el usuario y contraseña que sea el mismo que en la base de datos.

He intentado ser ordenado en todo pero a veces se complicaba porque me decían mas funcionalidades para implementar encima de otras y tenía que rediseñar la pantalla y hacer un cambio general de la pantalla y contemplar todos los casos lo cual me retrasaba en tiempo y me complicaba mucho la labor y en algunas pantallas por falta de tiempo he tenido que ir haciendo parches que han hecho que el código no sea tan escalable como lo era al principio de su desarrollo

4. TECNOLOGÍAS

- **Visual Studio 2017**
- Paquetes de compatibilidad Desarrollo Movil ->**Xamarin API .NET** (no confundir con XamarinForms).
- **SDK ANDROID**
- **API Emulator** 21 a la 24 (**minima versión 4.1 y máxima 6.1**)
- **MYSQL-PHPMYADMIN** en servidor en la nube https://mysql8.db4free.net/phpMyAdmin
- **Móvil propio** para realizar pruebas y depuración con **Android 5.1.1.**
- Lenguaje utilizado: **C# en plataforma .NET**, pero es Mono, ya que la estructura de los objetos se basa en ellenguaje de programación de JAVA.
- Diseño realizado en **XAML** orientado a objetos nativos de Android utilizando Drag &Drop y configurando las propiedades.

5. PLANIFICACIÓN DEL PROYECTO

ESTIMACIÓN INICIAL DEL PROYECTO:

Tarea	Horas
Gestión del proyecto	50
Diseño y analisis	240
Construcción	510
Pruebas	100
Despliegue y gestión de base de datos	80
Cierre del proyecto	20
Total	**1000**

La duración ha sido de unos 5 meses que trabajando unas 5 horas al día de lunes a viernes reúnen las 1000 horas que se le ha dedicado a este proyecto por su enorme envergadura.

COSTES REALES DEL DESARROLLO DEL PROYECTO:

Tarea	Dinero
Gestión del proyecto	500 €
Diseño y analisis	1000 €
Construcción	2000 €
Pruebas	500 €
Despliegue y gestión de base de datos	100 €
Cierre del proyecto	20 €
Total	**4120 €**

Este es el presupuesto aproximado de cada una de las fases del proyecto. No es del todo cierto ya que depende quien lo construya y con qué conocimientos técnicos.

El **diagrama de Gantt** lo resumo en lo anterior y **no realizo una especificación de las fechas y tareas exactas** porque ha habido tantas tareas al hacerlo entre el año pasado y este año que sería imposible recopilarlas todas, pero si puedo hacer un listado lo mas parecido al diagrama de Gantt con todas las actividades que he realizado, pero en cuanto a tiempos y lo demás es difícil llegar a una conclusión real.

A continuación doy una referencia de las tareas que se han seguido en todo el proceso, pero hacer un análisis del tiempo y costes es complicado teniendo en cuenta que lo he desarrollado en periodos de tiempo distintos a los habituales y no ha seguido un flujo único, sino que hace unos meses desarrollé una parte que llamo versión 1.0 y esta sería la versión 2.0 que incluye todas las funcionalidades nuevas y se puede

hacer una recopilación de las actividades desarrolladas a lo largo del proceso,pero especificar tiempos y fechas es realmente complicado, en las tablas lo he puesto resumidamente para que se entienda la balanza entre trabajo y tiempo que es lo que se busca con este apartado

Actividades:

- Logueo de usuarios y validaciones en el formulario.
- Pantalla de gestión de usuarios y control por rol de los mismos.
- Pantalla del usuario con sus tareas. Que cargue unos botones en función del rol
- Pantalla de crear empresas.
- Pantalla de crear tarea nueva e implementar biométricas.
- Pantalla de crear un nuevo cliente.
- Pantalla de añadir una nueva dirección y cliente.
- Pantalla de consultas de informes.
- Actividades transversales durante todo el ciclo -> validaciones de campos, ventanas modales,contemplar cualquier caso de prueba, realizar pruebas y depuraciones de código, configuraciones para portabilidad móvil y tareas de investigación de la tecnología usada.

6. DESARROLLO Y EJECUCIÓN DEL PROYECTO

Aquí se explica la fase de desarrollo del proyecto, como se ha desarrollado y como se ha llegado a esos resultados que el usuario ve en pantalla.

Lo primero hay que explicar que antes de esta etapa esta la fase de diseño y análisis del producto final a conseguir una vez tenemos eso nos ponemos a desarrollar la aplicación con la tecnología que hayamos planeado. En este caso usamos **XAMARIN API** que usa objetos **JAVA** nativamente de **Android,**pero no se utiliza.

Voy a hacer un pequeño resumen breve de como he realizado la implementación del código para explicar cómo ha sido la ejecución del proyecto.

Lo primero tendremos que decir que lo hemos desarrollado pantalla por pantalla, primero creamos un .**axml** arrastrando unos controles determinados (**TextView, EditText, Spinner**, etc.) y luego creamos una **Activity.cs** que irá conectada a esa vista **.axml**, en esa **Activity** vamos a localizar los controles mediante una instrucción llamada "**FindViewById**" y con ello poder interactuar entre ellos, un ejemplo es la pantalla de inicio:

```
SetContentView (Resource.Layout.Main);

var tituloApp = FindViewById<TextView>(Resource.Id.txtTituloApp);
var txtLogueate = FindViewById<TextView>(Resource.Id.txtLogueate);
```

```
txtLogueate.Click += delegate
{
    Intent intent = new Intent(this, typeof(ActivityLogueo));
    StartActivity(intent);
};
```

En la primera instrucción se dice qué vista se quiere pintar, en este caso la **Layout** llamada **Main** y luego se localizan los elementos que nos interesan para poder interactuar entre ellos, como por ejemplo al **clickar** sobre el texto nos lleve a una **Activity** que cargará otra vista distinta, se pueden pasar parámetros con:

```
intent.PutExtra("IdUser", idUsuario);
```

Y recoger en la siguiente **Activity** el parámetro enviado de la pantalla anterior con:

```
idUsuario = Convert.ToInt32(Intent.GetStringExtra("IdUser"));
```

Ahora hay métodos que atacan a la base de datos que tenemos en la nube:

Tenemos que recordar que atacamos a una base de datos **MySQL** en la nube y que la aplicación tendrá que acceder a las tablas de esa base de datos y consultar, modificar, insertar y borrar datos de la base de datos. Para ello hemos cargado las referencias para **MySQL** (**MySql.Data.CF y System.Data**) en el proyecto y en cada pantalla hacemos los **métodos** necesarios para **consultar, insertar, modificar o borrar y pintar** esos datos en pantalla, ya sea en forma de mensajes, ventanas o interactuando

con el formulario. Dejo un ejemplo de la **cadena de conexión** que usamos en todos los métodos y la estructura y los objetos que utilizamos:

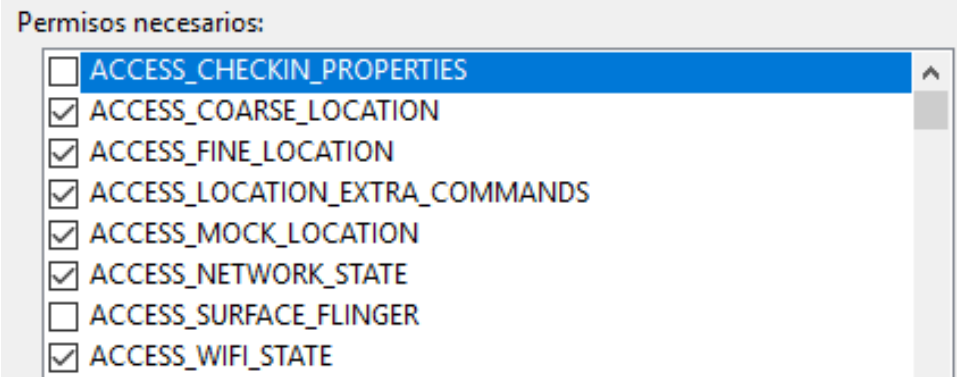
```
con = new MySqlConnection("Server=mysql8.db4free.net;Port=3307;database=agendaismael;User Id=ismalmysql;Password=administrad

if (con.State == ConnectionState.Closed)
{
    con.Open();
    using (MySqlCommand cmd = new MySqlCommand("SELECT Rol FROM Usuario WHERE Id = " + idUsuario, con))
    {

        using (MySqlDataReader reader = cmd.ExecuteReader())
        {
            while (reader.Read())
            {
                rol = reader.GetString(0).ToString();
            }
        }
    }
}
```

Ahora voy a explicar el proceso que he seguido para implementar la lógica de la biométrica que cada vez que entres en una determinada Activity:

La **biométrica usa GPS** a través del **Wi-Fi**, el medio que se necesita para que esta aplicación funcione es un móvil **con acceso a Internet**. La **biométrica** implementada es cuando **creas una nueva tarea o quieres añadir una dirección** a una tarea existente, siempre te registrará donde te encuentres, sea cual sea la parte del mundo, incluyendo un botón que nos llevará a la **API de Google** de nuestro móvil Android diciéndonos donde nos encontramos en el mapa. Todo ello desarrollado bajo unos permisos que le tenemos que establecer a la app **en el fichero de Manifiesto de Android:**



Para implementar la biométrica en una pantalla hay que usar la interfaz "**ILocationListener**" que se heredará en esa clase y **obligará a la implementación de unos métodos** que pondremos a continuación y que son de vital importancia para que nos **localice dentro de la aplicación** y nos **inserte en la base de datos** la **dirección actual, latitud, longitud y distancia topográfica**. A continuación, **muestro una parte del código** para explicarla y poder llegar a un entendimiento técnico de la implementación de biométricas.

```
//IMPLEMENTAMOS LOGICA DE LOCALIZADORES DEL ANDROID (BIOMETRICA)
locationManager = (LocationManager)GetSystemService(LocationService);

Criteria locationServiceCriteria = new Criteria
{
    Accuracy = Accuracy.Coarse,
    PowerRequirement = Power.Medium
};

//encuentra el gps a traves de la posicion del wifi
IList<string> acceptableLocationProviders =
    locationManager.GetProviders(locationServiceCriteria, true);

if (acceptableLocationProviders.Any())
{
    locationProvider = acceptableLocationProviders.First();
}
else
{
    locationProvider = string.Empty;
}

//Si pulsamos el boton de "Ver Mapa" nos accedera a nuestro Google Maps y se vera esa direccion
btnMap.Click += delegate
{
    string url = "geo:" + guardaLatitud + "," + guardaLongitud;
    var geoUri = Android.Net.Uri.Parse(url);
    var mapIntent = new Intent(Intent.ActionView, geoUri);
    StartActivity(mapIntent);
};
```

En esta se aplica a la clase principal y se ve al clickar al botón del mapa se nos redirigirá al **Google Maps** con las coordenadas nuestras que ha guardado en el método **OnLocationChanged** que es el que registra el lugar donde nos encontramos.

También vemos que nos localiza al proveedor de nuestra red y nos establece una prioridad para luego establecérsela a la manera de encontrar WiFi.

Aquí a continuación explico todos los métodos que gracias a la interfaz que hemos implementado en la clase hemos sobrescrito para hacer que todo funcione bien.

```
//Metodos de la interfaz añadidos por nosotros:
protected override void OnResume()
{
    base.OnResume();
    locationManager.RequestLocationUpdates(locationProvider, 0, 0, this);

}

protected override void OnPause()
{
    base.OnPause();
    locationManager.RemoveUpdates(this);
}
```

Estos simplemente inicializan el manejador que va a localizarnos mediante red Wifi o

```
public void OnLocationChanged(Location location)
{
    try
    {
        currentLocation = location;
        if (currentLocation == null)
        {
            Toast.MakeText(ApplicationContext, "Ubicación no encontrada. error al cargar latitud y longitud.", ToastLength.Long).
        }
        else
        {
            guardaLatitud = currentLocation.Latitude.ToString();
            guardaLongitud = currentLocation.Longitude.ToString();
            lati = string.Format("{0:N2}", Math.Truncate(currentLocation.Latitude * 100) / 100);
            longi = string.Format("{0:N2}", Math.Truncate(currentLocation.Longitude * 100) / 100);

            Geocoder geocoder = new Geocoder(this);

            IList<Address> addressList = geocoder.GetFromLocation(currentLocation.Latitude, currentLocation.Longitude, 5);

            Address address = addressList.FirstOrDefault();

            //Coordenadas de Trafalgar (51.50, -0.13):
            double lat1 = currentLocation.Latitude;
            double theta = currentLocation.Longitude - (-0.13);
            double distance = Math.Sin(Math.PI / 180.0 * (lat1))
                              * Math.Sin(Math.PI / 180.0 * (51.50)) +
                              Math.Cos(Math.PI / 180.0 * (lat1)) *
                              Math.Cos(Math.PI / 180.0 * (51.50)) *
                              Math.Cos(Math.PI / 180.0 * (theta));
            distance = Math.Acos(distance);
            distance = distance / Math.PI * 180.0;
            distance = distance * 60 * 1.15;

            distancia = distance * 1.609d; //lo pasa a kilometros

            if (address != null)
            {
                //Al final lo he tenido que poner asi si no, no encontraba la direccion
                string dir = address.GetAddressLine(0) + " " + address.GetAddressLine(1) + " " + address.GetAddressLine(2);
                txtAddress.Text = dir;
                direccion = dir;
            }
```

datos o lo que sea. Estos métodos son importantes ya que sino quedará vacío el manejador y no podremos llamarlo desde el método **OnLocationChanged**

En este método se ve como rellena la **latitud, longitud, dirección y distancia** que son fundamentales para la **geolocalización** actual, estos datos los insertará en base de datos.

Una vez quedan claras las cosas de como se han implementado y que procesos hemos seguido para llevar un orden en **el flujo de desarrollo de la aplicación** solamente puedo añadir que el resto de pantallas son iguales, incluyen métodos que atacan a la base de datos e interacción con los elementos de la interfaz como hemos explicado anteriormente y es seguir el mismo flujo en todas las pantallas, exceptuando que muchas llamadas a la base de datos no son iguales, hay **"selects", "inserts", "updates", "deletes"** y se recogerán de distinta forma y se añadirán al grid de formas distintas. También hay variantes en cuanto a como pintar ciertos controles, como un **spinner** que en función de si está en una pantalla o en otra nos hemos tenido que **montar clases** para manejar estos spinner. Existen muchos **procesos** por los que he seguido para desarrollar cada una de las pantallas, cada una tiene sus variantes y aquí lo he intentado explicar lo más breve posible y resumir de una forma que se vea esa ejecución del proyecto y como se ha desarrollado. Pero al fin y al cabo hay que tener en cuenta que hay mucho mas de lo que he mostrado en esteapartado, pero me parecía hacerlo demasiado extenso.

Debo añadir que en esta fase han entrado **pruebas tanto de caja blanca como de caja negra**, es decir tanto de **código** como a **nivel de usuario**. Por cada pantalla y cada control iba desarrollando casos de pruebas que me permitían probar todo el código y si fallaba algo poder corregirlo antes de pasar a una pantalla nueva. A medida que iba desarrollando también iba probando a nivel de usuario de como se comportaría determinado control y sus resultados, si no estaba contento con el resultado le daba vueltas hasta que saliera, en el caso de muchos **fallos** en código iba **depurando código con mi móvil** y poniendo **puntos de interrupción** donde quisiera probar el código de la aplicación e iba inspeccionando objetos y flujos para ver el origen del problema.

7. CONCLUSIONES

Las conclusiones a las que he llegado me parecen sorprendentes bajo mi punto de vista ya que esta aplicación me ha llevado muchos meses y me ha parecido muy complicado en cuanto a tecnología, investigación y unión de todos los conocimientos del grado.

Considero que un proyecto así de extenso para una sola persona es demasiado trabajo y demasiadas horas de investigación y de pruebas que hay que dedicarle, por suerte el año pasado ya realicé parte de esta aplicación dejándome la puerta abierta para poder ampliarlo a día de hoy pero con el tiempo que se ha dado y que mi vida laboral no me lo permite no podría haberlo acabado de no ser por eso, porque en total he estado unos 5 meses mas o menos para realizar toda la aplicación yo solo, incluyendo la documentación que se pide.

Lo que sí puedo decir que un proyecto así me ha enseñado muchísimo del mundo de desarrollo móvil que totalmente desconocía, porque siempre me he dedicado al desarrollo web y orientar un proyecto al desarrollo móvil me ha enseñado acerca de compilaciones móvil, compatibilidades con diferentes plataformas, como adaptarlo a ciertos dispositivos, que hay detrás de una app que nos venden en Google Play e incluso como llegar a distribuir el producto. He aprendido de configuraciones móviles muchísimo y mucho tema de frontend y backend móvil y como depurar código en un móvil. Aparte me ha hecho recapacitar acerca de que me gusta mas si el desarrollo web o móvil y la verdad me siento más cómodo con la web pero me ha entusiasmado este tema de una manera en la cual no descarto en el futuro dedicarme a hacer alguna aplicación por mi cuenta y poder subirla a Google Play.

He aprendido cosas super fascinantes y me ha encantado trabajar en este proyecto, a pesar de lo duro que ha sido y el escaso nivel de documentación que había en internet.

Me fijé que Xamarin API al ser nativo hay poca documentación a diferencia de XamarinForms que está repleto de documentación por todos los lados. De hecho las biométricas que he implementado solamente había documentación en un video de youtube que me costó encontrar mucho tiempo y tuve que traducir muchas cosas al español para lograr entenderlo todo con claridad y ver la lógica que se escondía detrás. Ha sido una tarea dura pero muy interesante enfrentarme con algo que apenas hay documentación y me tuve que buscar la vida en todo. Me alegro mucho de aprender lo que he aprendido y me lo llevo para en un futuro poder realizar algún proyecto móvil con todo lo que implica.

8. LÍNEAS DE INVESTIGACIÓN Y DESARROLLO FUTURAS

Esta aplicación se ha hecho basándome en un prototipo propio que tenía hecho y lo que he hecho ha sido ir ampliándolo cada vez mas y ha quedado mucho código repetido, muchas llamadas a las bases de datos sin hacer uso de algún **ORM**, los datos han sido extraídos nativamente con **ADO.NET**. Si se quisiese continuar con el desarrollo de este proyecto, habría que reorganizar y reestructurar el código y basarnos en algún tipo de arquitectura con sentido para poder ampliar con facilidad. Mi idea sería utilizar algún **ORM** de **SQLite** e implementar en la solución para tener todos los objetos **POCOs** que vengan de la base de datos. He encontrado uno muy interesante en **GitHub**:

https://github.com/activa/iridium

Sería interesante implementar nuestra base de datos actual de **MySQL** con **SQLite** y ese **ORM** para facilitar el **acceso a datos**, llamadas a métodos propios de **ORM** , etc.

Así podríamos ahorrar memoria, tiempos de respuesta, facilidad de comprender el código y estructurarlo, nos ahorraríamos ir pantalla por pantalla haciendo métodos para acceder a datos específicos (serían unas cuantas llamadas a algún método como *"Select("nombre").From("Tarea")*") que facilitarían la vida del programador. Podríamos hacer librerías (**dlls**) especificas para controladores de **Xamarin**, como por ejemplo el que implemento yo para las **biométricas** (uso de GPS) de una manera "**artesanal**" no sería lo mismo que si incrustara alguna librería planteada para tal fin y nos ahorraríamos muchas líneas de código y ya "lo haría todo por detrás". De todas formas añadir, que todo esto no lo he comprobado al 100% ya que uso **Xamarin API Native**, y lo conveniente para este tipo de proyectos sería utilizar **Xamarin Forms** ya que tiene muchas más ventajas con respecto al anterior en cuanto a código, rendimiento y sobre todo, es multiplataforma, lo único malo es que tienes que controlar más el tema de resoluciones de pantalla porque tienes que adaptarlo a **Android**, **iOS** y algunas modificaciones tienes que hacer, aunque sean mínimas, son costosas, me he fijado que en **Xamarin API Native** venía todo muy bien explicado y venía como "predefinido por detrás" para que se adapte a cualquier android, salvo alguna anomalía.

La facilidad que ofrece el framework de **Xamarin** es muy potente pero si no se sabe aprovechar bien las herramientas que te ofrece .**NET** queda una aplicación costosa de desarrollar, pero ello no implica que sea divertida y te enseñe a desarrollar un proyecto que ni si quiera conoces su tecnología.

CONCLUSIONES CON RESPECTO A LO ANTERIOR

También cuando comencé a desarrollar la idea y los primeros **borradores** y todo el trabajo tan grande que me ha supuesto la aplicación no me paré a pensar en todo esto, empezó con un **hobby** que aproveché para ir agrandándolo cada vez más y se me quedó muy repetitivo. Pero las posibilidades para ampliar este proyecto son enormes y no con mucho coste, pero sí que se necesitaría una persona para ayudar a programar y probar, porque las pruebas son duras, y cada vez le sacas mas y mas fallos y al final tienes que ir "**parcheando**" para que no se rompa lo demás. De todas formas estoy muy orgulloso de como ha quedado el proyecto, salvo la parte de la interfaz gráfica que la verdad podría haberme quedado mejor, pero todo también ha sido porque no he tenido mucho tiempo y la cosa ha sido un poco precipitada. Pero he aprendido muchísimo acerca de la **tecnología móvil** y me ha encantado la parte de **biométricas** aunque haya sido un "*infierno*" programarla, me ha enriquecido mucho ver los **servicios**, **procesos** que usan, el **acceso a una base de datos MySQL** en la nube para una app móvil ha sido todo un reto porque al principio no funcionaba nada de lo que probaba, he aprendido muchísimo **inglés** ya que la documentación de internet estaba el 90% en inglés, apenas encontraba algo en español y me era muy difícil deducir, la parte de biométricas no era fácil y no había nada en español y tuve que investigar mucho, diría que un **60% del tiempo me lo he pasado investigando** y probando "ensayo y error" y el **40% desarrollando**, porque mi dificultad estaba en no conocer la tecnología móvil y cómo funcionaba. Me encantaría el día de mañana retocarla, incluso **comercializar** esta **app** para empresas de este tipo y poder expandir mi aplicación. Ahora mismo sé que encantaría a cualquiera que tuviera un negocio como este pero si se diera más vueltas con todo lo anterior que he mencionado sería un **negocio rentable**.

La verdad me ha enseñado mucho hacer una aplicación **sin conocimientos de la tecnología que se pretendía usa**r y he aprendido muchísimo de cómo desarrollar una aplicación de aquí a un futuro sin tener conocimientos previos de cualquier tecnología que me dieran que no conociera la podría sacar, ya que gracias a hacer este proyecto me sé buscar más la vida en algo que no conozco y me ha encantado desarrollarla. Actualmente yo trabajo en una empresa donde somos equipos multidisciplinares y todos sabemos de todo y en ella hay **android**, **iOS**, .**NET** todos sabemos de todo y con la experiencia que he ganado este ultimo año me ha servido a darme cuenta donde he cojeado y como podría mejorarla en un futuro. La verdad lo tendría muy claro lo que cambiar de aquí a un futuro.

9. BIBLIOGRAFÍA

https://developer.xamarin.com/api/

https://docs.microsoft.com/en-us/xamarin/android/

Ejemplos de **GitHub.**

https://developer.xamarin.com/api/root/MonoAndroid-lib/

StackOverFlow

YouTube con varios videos explicando cómo funcionaba la tecnología poniendo: "Xamarin" habia muchos tutoriales interesantes que seguí para aprender.

Mucho uso de **Google** y especificando mi problema en inglés me iba encontrando diversas explicaciones que me hacían entender mi problema y como solucionarlo.

Como me ha ocupado tanto tiempo no he podido guardarme todos los lugares a los que he consultado cuando tenía una duda pero resumidamente han sido esos sitios los que he estado usando con mucho hincapié. Todo lo que he redactado es mío y mi trabajo no es ninguna copia de ningún proyecto, es hecho y trabajado por mí, si he hecho alguna referencia a alguna página web he puesto su hipervínculo.