ESCUELA DE INGENIERÍA - Curso 2019/2020

LABORATORIO INTEGRADO DE SOFTWARE

MEMORIA

Agustín Tamayo - 1492214
Arnau Sarabia - 1492826
Jan Tugores - 1494134
Judith Bellido - 1455984
Angel Sacristán Ruiz - 1457243
Martin Susin Palacios - 1492417
Pau Borda - 1458601
Carles Milanés Horno - 1455091

Equipo docente:
Antonio Manuel López Peña
Lluís Gesa Bote
Jose Luis Gomez Zurita

7 de Junio de 2020

Índice

1. Introducción

Las problemáticas sociales son un concepto el cual afecta a todos los individuos por igual. Durante el año 2019, España ha experimentado distintas fluctuaciones en los índices de agresiones callejeras, incendios y accidentes en carretera. Lamentablemente, casuísticas como las agresiones sexuales han incrementado un 18 % desatando un miedo general en las personas.

La inseguridad, juntamente al sentimiento de estar indefenso, son sensaciones fruto de actuaciones no éticas y morales de la sociedad en la que vivimos. Actualmente, nadie puede afirmar con total seguridad que está protegido cuando sale a pasear. Sin embargo, a pesar de que nadie pueda otorgar una garantía absoluta en la seguridad de los individuos, podemos aportar pequeñas herramientas para cruzar el umbral establecido y aumentar el confort de los transeúntes.

Una de los principales motivaciones de los Objetivos Horizonte 2020 es la lucha por un cambio social. Actualmente, uno de sus mayores retos reside en la lucha contra la delincuencia para fomentar la sociedad segura.

Por este motivo surge SHiFT, una aplicación para todo tipo de usuarios que permite realizar llamadas de socorro a los servicios de emergencia independientemente de la situación en la que el afectado este inmerso. Esta, no únicamente permite pedir ayuda, también transmite en directo la geolocalización del socorrido y adquiere información de su entorno. Es decir, tiene la capacidad de reconocer otros usuarios SHiFT cercanos con el fin de pedirles ayuda mediante la transmisión de su ubicación.

Actualmente vivimos en la era de la información, el ser humano está acostumbrándose a realizar sus tareas de forma rápida y sencilla, sin estar ocupado en procedimientos que puedan realizarse automáticamente. La aplicación SHiFT sigue este hilo de desarrollo posibilitando así la activación de alertas mediante la voz. De esta forma, el tiempo involucrado en usar el dispositivo móvil se ve reducido. Gracias a este factor, se incrementa el tiempo de defensa y decrementa el tiempo de socorro, el cual puede resultar clave para asistir a accidentados, por ejemplo.

2. Objetivos

Antes de instanciar las distintas herramientas, metodologías, prerrequisitos y otros factores necesarios para el desarrollo del proyecto en cuestión, resulta esencial entender los objetivos que SHiFT tiene. Mediante este paso, evitamos abrir puertas a confusiones y dejamos constancia de las distintas prioridades existentes.

Así pues, a continuación se reflejan los objetivos de la aplicación ordenados según su nivel de criticidad.

Objetivo	Criticalidad
Ayudar en el proceso de socorro	Alta
Minimizar el tiempo de socorro	Alta
Facilitar la intervención de los servicios de emergencia	Alta
Crear una interfaz con un entorno *User Friendly*	Media
Comunicar emergencias a personas cercanas	Baja
Aportar recursos al peritaje posterior a la emergencia	Baja

Tabla 1: Tabla de objetivos.

Para facilitar la comprensión de los distintos puntos mostrados, se recorrerán las menciones con criticidad alta para otorgar detalles al lector. Las referentes a criticidades media y baja, quedan implícitas en las explicadas:

1. **Ayudar en el proceso de socorro**
   Los servicios de emergencia, gracias a la aplicación SHiFT, recibirán de forma inmediata todas las llamadas de socorro de los usuarios. A pesar de reducir el tiempo requerido para pedir ayuda, se consigue ir un paso más allá mediante la transmisión de la geolocalización del socorrido y el reconocimiento de los dispositivos cercanos a él.

   Gracias a los factores adicionales mencionados, aumentamos el nivel de precisión durante el rescate y obtenemos un reconocimiento lógico de su entorno mediante *beacons*. Así pues, podemos notificar automáticamente a otros usuarios de la aplicación la ayuda que pueden aportar a la situación.

2. **Minimizar el tiempo de socorro**
   Un accidente es la perfecta escenificación para este objetivo. Muchas veces, la vida de un accidentado se debate en cuestión de minutos. Por este motivo, reduciendo el tiempo necesario para realizar la llamada de socorro y aumentando la libertad del ayudante para intervenir en la situación, producimos un incremento en las posibilidades de supervivencia del afectado.

   Gracias a la automatización de procesos de SHiFT y su reconocimiento de voz, el usuario puede activar alertas sin necesidad de ocupar sus manos.

3. **Facilitar la intervención de los servicios de emergencia**
   Toda alerta generada por un usuario en peligro será mostrada en una página privada para los servicios de emergencia. Esta, cuya recarga y sincronización con la base de datos se realizará de forma constante y automática, alertará de una persona en peligro a través de la web citada.

   Una vez notifique al técnico de emergencias, otorgará información crucial para evaluar la situación del usuario en peligro y actuar al respecto:

   - Geolocalización.
   - Edad.
   - Nombre y Apellidos.
   - Número de teléfono.
   - Dónde vive (Ciudad, calle, número de casa o piso).

3. Grado de cumplimiento del contrato

Los propósitos y objetivos planteados en el proyecto se han alcanzado e implementado acorde con la planificación realizada. Para reflejar su cumplimiento, se recorrerá el contrato acordado entre la parte docente y los integrantes del grupo.

Resulta fundamental destacar que se han asumido todas las proposiciones necesarias para poder obtener una calificación de 10. Para su total verificación, pueden verse los resultados en el vídeo de demostración presentado el 25 de Mayo. Sin embargo, a continuación justificaré la afirmación expuesta:

3.1. Propuesta para el 5 (Realizada)

Se han realizado todas las tareas de razonamiento, planteamiento y diseño:

- Diagrama y documentación detallada de casos de uso.
- Diagramas de secuencia.
- Diagrama de clases acorde con el modelo MVVM.
- Diagrama de Entidad-Relación para organizar la base de datos del *backend*.
- Mockups de la aplicación SHiFT.
- Análisis de Requisitos Funcionales y No Funcionales.
- Planificación cronológica y diagrama de Gantt.
- Identificación y definición de objetivos del proyecto.
- Identificación de la razón de ser de SHiFT.

También se han desarrolladas las funcionalidades que el punto del contrato exigía:

- Desarrollar una aplicación simple la cual no aplique ningún sistema de transmisión de geolocalización y detección de voz. Las alarmas se generan mediante la selección de tres botones distintos, eliminando la parte de IA y autonomía prometida.
- La aplicación no aplica ninguna serialización de objetos ya que las alarmas están definidas de forma predeterminada, evitando así su personalización.
- Los usuarios de SHiFT tienen la posibilidad de hacer *Login*, *Register* y modificar sus perfiles sin ningún problema.
- El servidor recibe las alertas y las añade a la base de datos para dejar constancia del problema ocurrido.

3.2. Propuesta para el 7 (Realizada)

En el nivel en cuestión del contrato se exige una mayor complejidad en el desarrollo de la aplicación. Así pues, se han cumplido los siguientes factores acorde con el contrato:

1. Crear la página de emergencias con su respectivo *Login*. Esta, se sincroniza automáticamente y de forma transparente con la base de datos para recibir las alertas más recientes.
2. Mostrar en la página de emergencias información básica sobre el socorrido (Geolocalización, Nombre, Fecha de Nacimiento, Calle donde vive, Ciudad, Código postal, Móvil).
3. Asegurar una buena transmisión de la Geolocalización en directo entre Cliente - Servidor - Emergencias y el posicionamiento del usuario en el mapa.
4. Otorgar al usuario la posibilidad de desactivar la alerta mediante la introducción manual de su contraseña. Esta parte implica una rápida eliminación del *warning* generado y mostrado en la página de Emergencias.
5. *Testing* y refinamiento para asegurar el buen funcionamiento de SHiFT y sus funcionalidades implementadas.

3.3. Propuesta para el 10 (Realizada)

Finalmente, se ha alcanzado la máxima puntuación posible gracias al establecimiento del mayor grado de independencia entre SHiFT y el usuario.

1. Configuración de alertas personalizadas mediante reconocimiento de voz.
2. Serialización de las configuraciones y perfiles establecidos para evitar su volatilidad y decrementar la carga del servidor. Cada vez que la aplicación se abre, se recargan automáticamente las llamadas de socorro establecidas y los datos del perfil de usuario.
3. Activación de alertas en 2 tiempos:
   - Invocación de la aplicación vía voz (Ejemplo: Oye SHiFT).
   - Manifiesto, mediante la vibración del móvil, de la escucha de SHiFT al usuario.
   - Activación de la alerta citando las palabras configuradas.
4. El usuario no necesita realizar ninguna interacción manual con el teléfono para invocar su llamada de socorro. Las situaciones de peligro se gestionan mediante la voz.
5. *Testing* y refinamiento para asegurar el buen funcionamiento de SHiFT y demostrar su utilidad.
6. Implementación de sistema multi-threading para posibilitar la gestión de múltiples usuarios en peligro de forma simultánea.
7. Mejora de la página de Emergencias para diferenciar entre *Warnings* atendidos y no atendidos. Así pues, cada alerta consta de distintas fases y estados.
8. Implementación de tecnología *beacon* para advertir a usuarios SHiFT cercanos del peligro que corre una persona próxima a ellos.

4. Metodología de desarrollo

En este apartado se expone la metodología de trabajo implementada así como la organización y criterios de distribución de tareas y archivos.

4.1. Metodología de trabajo y organización interna

El desarrollo del proyecto se fundamenta sobre una metodología ágil basada en 2 reuniones por semana. La primera realizada los Lunes, cuya finalidad se reduce a la revisión y reflexión de problemáticas intrínsecas surgidas durante el desarrollo de tareas pendientes. La segunda realizada los Miércoles, con intención de planificar y asignar nuevas actividades de diseño e implementación a los distintos miembros del equipo.

Las finalidades de los días de reunión se han escogido en función del tiempo lectivo invertido en el calendario de LIS. Como puede apreciarse, los Lunes son sesiones enfocadas al debate, lluvia de ideas, diseño/rediseño, planteamiento de soluciones, entre otros, debido que el tiempo necesario para llevar a cabo el perfil de la sesión es mucho mayor al disponible los Miércoles. Por contra, el día más breve se rige de tareas más sistemáticas (asignación de tareas y preferencias) para poder evaluar los resultados obtenidos durante el Lunes de reflexión.

Por otro lado, la organización interna se basa en una jerarquía rígida y horizontal. En esta, Ángel Sacristán y Carles Milanés son los responsables de gestionar los equipos de documentación y desarrollo, respectivamente.

Finalmente, el control de versiones, asignación de tareas y control de tiempo invertido se han manejado mediante Bitbucket y Jira.

5. Requisitos de la aplicación

Para desarrollar el proyecto es de suma importancia el reconocimiento de las funcionalidades explícitas que debe de hacer y sus correspondientes restricciones. Por este motivo, se ha realizado una recolección y planteamiento de Requisitos Funcionales y No Funcionales. Cabe destacar que la captura, aceptación y rechazo de estos se ha basado en las características y grado de relación directa con los objetivos planteados. Así pues, con estos se intenta estrechar y definir el camino a seguir para cumplir con todos los propósitos.

A continuación mostraremos los cinco requisitos más característicos del proyecto con un breve resumen para complementarlos:

5.1. REQ-F-1-01

En primer lugar, resulta fundamental poder establecer una comunicación entre el Front-End y Back-end. Sin el requisito expuesto, SHiFT no tendría ningún método para almacenar datos de usuarios ni podría desarrollar funcionalidades de login, registro, enviar alertas a los servicios de emergencias, entre otros.

ID	REQ-F-1-01
Título	Interconexión entre cliente y servidor
Descripción	El sistema SHIFT tiene que establecer una conexión entre cliente y servidor para hacer peticiones
Prioridad	A
Verificación	T
Padres	

Figura 1: Requisito Funcional que describe la interconexión entre cliente y servidor.

5.2. REQ-F-3-01

En segundo lugar, es de suma importancia que la solución pueda correlacionar sonidos con las palabras de activación de alertas previamente configuradas.

ID	REQ-F-3-01
Título	Detección de palabra clave
Descripción	El sistema SHIFT tiene que identificar que el sonido emitido coincide con la palabra clave.
Prioridad	A
Verificación	T
Padres	REQ-F-1-02,REQ-F-1-03a

Figura 2: Requisito Funcional detectar correctamente las palabras clave dictadas por voz.

5.3. REQ-F-3-03

Una de las proezas de SHiFT es su capacidad para enviar la ubicación de un usuario en peligro en tiempo real. Por este motivo consideramos que el requisito mostrado es de suma importancia.

ID	REQ-F-3-03
Título	Detectar geolocalización
Descripción	El sistema SHIFT tiene que permitir enviar la ubicación al detectar la palabra clave.
Prioridad	M
Verificación	T
Padres	REQ-F-1-02,REQ-F-1-03a, REQ-F-3-03

Figura 3: Requisito funcional para transmitir la geolocalización de un usuario cuando activa una alerta.

5.4. REQ-F-3-07

El proyecto desarrollado permite reconocer otros usuarios SHiFT cercanos. Este es el punto más fuerte del sistema ya que permite pedir ayuda a múltiples *endpoints* a la vez que se captura información del escenario.

ID	REQ-F-3-07
Título	Escaneo de dispositivos cercanos SHIFT
Descripción	El sistema SHiFT tiene que ser capaz de detectar dispositivos cercanos de otros usuarios SHiFT.
Prioridad	M
Verificación	T
Padres	

Figura 4: Requisito Funcional para detectar otros usuarios SHiFT cercanos al activar una alerta.

5.5. REQ-NF-1-03

Finalmente, consideramos que el sistema no puede tener una tasa de fallos mayor al 10 % durante el reconocimiento de palabras clave. Resulta esencial garantizar al usuario en peligro que no obtendrá dificultades adicionales para obtener ayuda de forma rápida.

ID	REQ-NF-1-03
Título	Tasa de fallos de la detección de sonidos
Descripción	El sistema SHIFT no puede tener una tasa de fallos en la detección de sonidos superior al 10 %
Prioridad	M
Verificación	T
Padres	

Figura 5: Requisito No Funcional para asegurar una baja tasa de fallos en el reconocimiento de voz.

6. Control de versiones

El control de versiones así como la regulación y compartimiento de documentos o código se ha realizado a través de Bitbucket. Todos los miembros del equipo han tenido acceso de lectura y escritura al repositorio Git para sincronizar las distintas tareas asignadas y su evolución. Cabe destacar que las aplicaciones GitKraken y SourceTree se han usado de forma complementaria a la primera mencionada.

A continuación se explicarán las 3 ramas creadas para llevar a cabo el desarrollo del proyecto.

6.1. Develop

La rama de desarrollo contiene el resultado final de SHiFT. Sobre esta se han ido realizado todos los *Merge* de las distintas funcionalidades programadas e implementadas.

6.2. Feature

Rama creada a partir de *Develop*. Esta constituida por 6 sub-ramas dedicadas a la programación de las funcionalidades que fundamentan la solución presentada. Mediante la división comentada, se permite paralelizar el trabajo sin causar bloqueos en el flujo de avance.

- **BaseAPP:** Estructura de la aplicación y navegación entre pantallas.
- **Login:** Datos relacionados con el usuario y su perfil.
- **Alarm:** Creación de alarmas, editar alarmas y activación de alarmas.
- **Location:** Funcionalidades de localización y enlace con la activación de alarma.
- **BLE:** Búsqueda de dispositivos cercanos y conexión.
- **Test:** Implementación de test sobre la rama Develop.

6.3. Master

Para finalizar, se ha decidido utilizar una rama dedicada íntegramente a la gestión de documentación del proyecto. Cabe destacar que contiene distintos directorios para separar los requisitos, imágenes, diagramas, actas, etc.

7. Test

Es muy importante someter el proyecto a una fase de test y refinamiento. Mediante esta, puede medirse la consistencia y rigidez del modelo creando pruebas y casuísticas que lo lleven a sus límites. Por este motivo, a continuación se exponen distintas metodologías de *testing* utilizadas para verificar el correcto comportamiento de SHiFT:

- **Prueba unitaria (Test Case)**
- **Revisión Técnica Formal (RTF)**
- **Exploratory Testing**

Los puntos comentados nos han permitido detectar y reconocer múltiples fallos o interacciones inadecuadas. Así pues, a medida que evolucionaba SHiFT, el equipo podía solucionar y optimizar el código implicado gracias a la identificación de líneas incorrectas o redundantes.

7.1. Prueba Unitaria

El desarrollo de los Test Case se ha realizado mediante Visual Studio. La superación de las pruebas se acepta o rechaza en función de los "ticks" o cruces reflejados tras la ejecución de códigos.

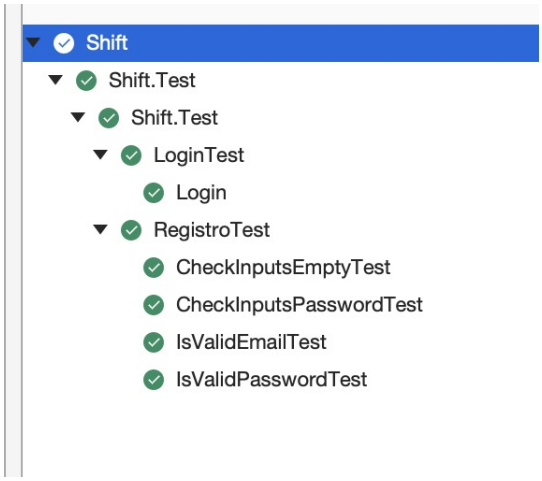

Figura 6: Criterio de validación de los Test Case ejecutados.

7.2. Revisión Técnica Formal (RTF)

El equipo ha realizado un análisis de código en estático, es decir, sin ejecutar su funcionalidad. Mediante este proceso se ha evaluado el formato, estilo de codificación, variables, comentarios, entre otros aspectos. La sección en cuestión, nos ha permitido anotar y comunicar las erratas percibidas a los desarrolladores. Así pues, la comunicación entre equipos ha devenido un factor clave para realizar exitosamente la RTF, sobretodo durante la fase de verificación de cambios y correcciones sobre el código.

7.3. Exploratory Testing

Finalmente, el Exploratory Testing nos ha permitido verificar el correcto comportamiento del Front-End mediante la ejecución de SHiFT en un dispositivo Android. Cara el Back-End, através de ReqBin se han creado peticiones POST con distintas características para observar la reacción del servidor implementado. La superación de pruebas se ha validado según la respuesta del servidor: "*true*" o "*false*".

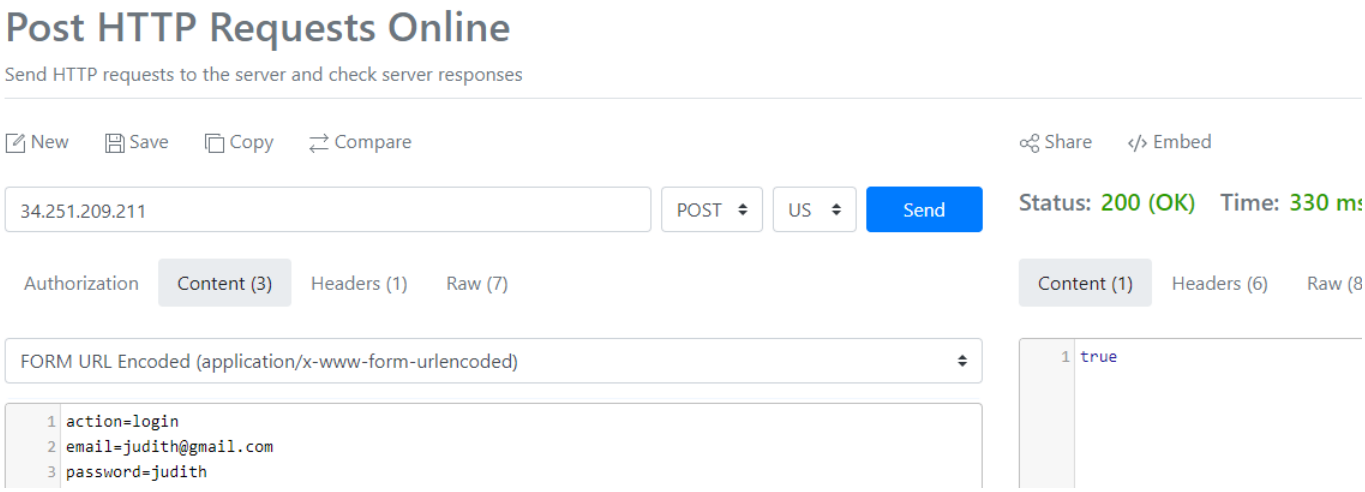

Figura 7: Test superado al intentar hacer Login con un usuario existente.

8. Problemas durante el desarrollo

En este punto, se explican las principales problemáticas experimentadas y la solución implicada.

- La segmentación y estimación de tareas en una franja temporal es la primera problemática encontrada. Esta, surgió durante la fase de planificación y la creación del diagrama de Gantt. La solución implicada consiste en dos puntos completamente independientes: El primero, basado en una semana íntegramente dedicada al estudio de viabilidad del proyecto y la recogida de información para la implementación de sus distintas partes. El segundo, la inclusión de dos semanas de contingencia en el planteamiento de la planificación. Es decir, desarrollar SHiFT un par de semanas antes del *deadline* con la finalidad de tener 14 días para solucionar imprevistos. También cabe destacar que, paralelamente, la resolución de la problemática se complementa con revisiones semanales para verificar el correcto avance del proyecto.

- La idea de SHiFT al principio era muy grande y dispersa. Este hecho causó que el equipo tuviera que premeditar las funcionalidades esenciales que debía tener sin sobrepasar el límite de tiempo establecido. Lamentablemente, a pesar de reflexionar varias veces, el avance del proyecto sobre una base irregular causó que el equipo modificara en múltiples ocasiones los casos de uso. La problemática comentada retrasó el avance y no tiene una solución a comentar, sino que es una conclusión extraída una vez se ha finalizado el proyecto.

- Durante el inicio del proyecto costó encontrar la dinámica de trabajo más adecuada a las personalidades de los integrantes. Este hecho provocó un poco de desinformación y la falta de comunicación entre las distintas partes del equipo. Sin embargo, gracias al modelo flexible que se escogió durante las primeras semanas, los responsables del proyecto pudieron establecer una jerarquía y metodología de trabajo adaptada a las necesidades de SHiFT. Las consecuencias de los hechos comentados supusieron incoherencias temporales entre los diagramas de secuencia, diagramas de clase y diagramas de casos de uso. Sin embargo, se concluye que el sacrificio realizado ha permitido crear un ambiente agradable y adecuado con el objetivo de incitar y motivar a los integrantes a la colaboración.

- La implementación de test en interfaces gráficas programadas con C# supusieron distintos problemas. El principal, y por ende más comprensible, se basaba en la replicación de la interfaz sin la interacción directa con esta. La solución encontrada se fundamenta en el uso de MockObjects, los cuales han permitido al equipo replicar las funcionalidades necesarias para realizar los *Unit Test*.

- Finalmente, el uso de nuevas tecnologías así como la implementación de funcionalidades nunca vistas anteriormente han supuesto problemáticas a gran escala. La más destacable señala las dificultades e incompatibilidades de las herramientas de reconocimiento de voz con la base y entorno del proyecto. Este hecho casi causó la migración de todo el código fuente del Front-End a otra plataforma y lenguaje. Para denotar y contrastar el nivel de criticidad que supuso, se empezó a desarrollar el proyecto de 0 en Android Studio, paralelamente a la rama principal para evitar retrasos no remontables en la planificación. Como puede observarse en el resultado final, la problemática se resolvió gracias a la colaboración y tiempo invertido de todos los integrantes para encontrar herramientas ajustadas a nuestras necesidades.

9. Código destacado

En el siguiente apartado se muestran las capturas más importantes del código del proyecto desarrollado. Cabe destacar que el Front-End está programado en C# haciendo uso de la plataforma Xamarin. El Back-End, en cambio, se halla en una Máquina Virtual alojada en Amazon EC2 y se constituye de distintos lenguajes de programación: JavaScript, HTML y PHP. También utiliza herramientas como Apache y MySql.

9.1. Front-End: Reconocimiento de voz

En el código adjuntado pueden apreciarse las funciones involucradas para permitir la constante escucha de SHiFT y procesar las palabras dictadas vía voz, respectivamente.

En la primera, se accede recursivamente al micrófono del dispositivo para captar las palabras mencionadas por el usuario. En la segunda, en cambio, se procesa la cadena captada con la finalidad de iniciar el proceso necesario. Este, se inicia gracias al intercambio de mensajes entre la función *WordsProcessed* y el *View-Model*.

```
public void StartRecursiveSpeechToText(){
    string rec = global::Android.Content.PM.PackageManager.FeatureMicrophone;
    if (rec == "android.hardware.microphone"){
        try {
            _voiceRec = new STTActivity(_activity);
            _voiceRec.WordsProcessed += WordsProcessed;
            _voiceRec.StartListening();
        }
        catch (ActivityNotFoundException ex){
            String appPackageName = "com.google.android.googlequicksearchbox";
            try{
                Intent intent = new Intent(Intent.ActionView, global::Android.Net.Uri.Parse
                ("market://details?id=" + appPackageName));
                _activity.StartActivityForResult(intent, VOICE);
            }
            catch (ActivityNotFoundException e){
                Intent intent = new Intent(Intent.ActionView, global::Android.Net.Uri.Parse
                ("https://play.google.com/store/apps/details?id=" + appPackageName));
                _activity.StartActivityForResult(intent, VOICE);
            }
        }
    }
    else{
        throw new Exception("No mic found");
    }
}
```

```
public static void WordsProcessed(object sender, string words){
            string textInput = words;
            var act = _activity as MainActivity;
            MessagingCenter.Send<IMessageSender, string>(act, "RSTT", textInput);
}
```

9.2. Front-End: Geolocalización del usuario

Como puede a continuación, mientras el usuario tenga la alerta activa se itera en este segmento de código para captar las coordenadas X e Y del dispositivo móvil. El *delay* entre cada refresco es de 5 segundos, de este modo captamos las coordenadas en tiempo real y las enviamos al Back-End.

```
await Task.Run(async () => {
        while (!_coordinatesTaskCTS.IsCancellationRequested){

            try{

                _coordinatesTaskCTS.Token.ThrowIfCancellationRequested();
                LocationModel lastCoordinate = null;

                if (_coordinates?.Any() ?? false){
                    lastCoordinate = _coordinates?.Last();
                }

                var newCoordinate = await LocationRetriever.GetCurrentLocation();

                if (newCoordinate == null){
                    await Task.Delay(delay);
                    continue;
                }

                LocationChanged?.Invoke(this, new EventArgs());
                BaseSettings.SetSetting<LocationModel>(LastLocationKey, newCoordinate);
                SendLocation(newCoordinate);

            }
            catch (OperationCanceledException){
                ...
```

9.3. Front-End: Conexión con dispositivos cercanos

Finalmente, se muestran las dos funciones implicadas para el reconocimiento de dispositivos cercanos y la conexión hacia ellos. En la primera, se detectan todos los *peers* dentro del rango del usuario SHiFT. En la segunda, en cambio, se establece la conexión con los distintos dispositivos detectados.

```
public void OnPeersAvailable(WifiP2pDeviceList peerList){

    List<WifiP2pDevice> refreshedPeers = peerList.DeviceList.ToList();
    if(!refreshedPeers.Equals(_peers)){
        _peers.Clear();
        _peers.AddRange(refreshedPeers);
    }
    Connect();
    if(_peers.Count == 0){
        return;
    }
}

public void Connect(){

    if(_peers.Any(p => !_peersConnected.Contains(p))){

        var peerToConnect = _peers.First(p => !_peersConnected.Contains(p));
        WifiP2pConfig config = new WifiP2pConfig();
        config.DeviceAddress = peerToConnect.DeviceAddress;
        config.Wps.Setup = WpsInfo.Pbc;
        manager.Connect(channel, config, this);
    }
}
```

9.4. Back-End: Recepción de coordenadas y dispositivos cercanos

Para permitir contrastar el apartado anterior con el tratamiento que realiza el servidor de los datos obtenidos, se adjunta la función *userStoreAlertInfo*. Mediante esta se actualiza la geolocalización y *nearby devices* enviados por un usuario SHiFT.

```
<?php
function userStoreAlertInfo($conn, $data){

        $result = true;
        $email = getCookieInfo();

        try{
            $sql_pk_warning = $conn->prepare("Select pk_warning from `Alert_Call` a
            JOIN `User` u ON a.`pk_user`=u.`pk_user` Where u.`email`='$email'");
            $sql_pk_warning->execute();
            $pk_warning = $sql_pk_warning->setFetchMode(PDO::FETCH_ASSOC);
            $pk_warning = $sql_pk_warning->fetch();

            $sql_warning_exists = $conn->prepare("Select pk_warning from `Alert_Info`
            Where pk_warning='".$pk_warning['pk_warning']."'");
            $sql_warning_exists->execute();
            $exists = $sql_warning_exists->setFetchMode(PDO::FETCH_ASSOC);
            $exists = $sql_warning_exists->fetch();

            if(isset($exists['pk_warning'])){
                   $sql = "UPDATE shift.Alert_Info SET
                   geolocation='".$data['geolocation']."',
                   nearby_devices='".$data['nearby_devices']."'
                   Where pk_warning='".$pk_warning['pk_warning']."';";
            }else{
                   $sql = "INSERT INTO shift.Alert_Info (pk_warning, geolocation,
                   nearby_devices) VALUES('".$pk_warning['pk_warning']."',
                   '".$data['geolocation']."','".$data['nearby_devices']."');";
           }

           $conn->exec($sql);

        }catch(PDOException $e){
                $result = false;
        }

        return $result;
}
?>
```

10. Crystal Clear

Una vez finalizado el proyecto pueden verse distintas propiedades Crystal Clear reflejadas en el *timeline* de desarrollo. Este apartado se ha realizado de forma premeditada y valorando las perspectivas apreciadas por todos los integrantes del equipo. Así pues, a continuación se exponen las observaciones coincidentes:

En primer lugar, a pesar de no dirigirnos a un cliente final, la propiedad de **Entregas Frecuentes** se ha visto reflejada constantemente. Esta, se ha cumplido mediante la actualización iterativa e incremental de documentos acorde con las tareas asignadas. Como se explica en el apartado 4 - Metodología y desarrollo, cada Lunes se realizaba una explicación, valoración y reflexión de los avances entregados por cada miembro del equipo. Los Miércoles en cambio, se asignaban tareas y se definían las prioridades y objetivos primordiales que se querían cumplir con ellas. Así pues, partiendo de la anterior explicación, las propiedades de **Mejora Reflexiva** y **Focus** surgen de forma conjunta y consecuente a la primera. Además, haciendo inciso en las nuevas, la segunda se cumple debida la flexibilidad y adaptación a las necesidades devenidas por replanteamientos e introspecciones de proyecto. En la tercera en cambio, la planificación lineal establecida, la entrada regular de trabajo así como la determinación clara de tareas por individuo, han permitido centrarse absolutamente en el trabajo asignado. Así pues, no han surgido *overheads* por saltos entre tareas o distracciones fuera del contexto de desarrollo.

Respecto la propiedad de **Comunicación Osmótica**, se ha concluido el debate rehusando su cumplimiento. Es cierto que al principio, antes del confinamiento causado por la pandemia COVID-19, se realizaban reuniones de equipo en una misma ubicación física. Sin embargo, esta propiedad va más allá refiriéndose al desarrollo íntegro del proyecto estableciendo comunicaciones cara a cara. Así pues, se cierra este punto como incumplido puesto que se dirige más a entornos profesionales de desarrollo como pueden ser empresas, centros de investigación, incubadoras, entre otros.

La propiedad **Seguridad Personal** ha sido las más abundante durante el avance del proyecto. En todo momento cualquier integrante del equipo ha podido expresar su opinión con total libertad. Además, toda idea ha sido valorada y atendida de forma conjunta.

Finalmente, se ha considerado que la propiedad **Entorno técnico con pruebas automatizadas** se ha cumplido parcialmente mediante el test implementado. En la documentación adjuntada a la entrega, pueden apreciarse códigos para automatizar test de Registro, Login, Modificación de Perfil, etc. Lamentablemente, como puede apreciarse, se cumple "parcialmente" ya que no tenemos los recursos necesarios para realizar una automatización completa. Para ello, requeriríamos de licencias de herramientas cuyo coste no entra dentro de las previsiones del proyecto.

Llegados a este punto y a modo de resumen, las propiedades Crystal Clear cumplidas son las siguientes: **Entregas Frecuentes**, **Mejora Reflexiva**, **Focus**, **Seguridad Personal** y **Entorno técnico con pruebas automatizadas**.