Sistema de localización marítima y alerta temprana

José Antonio González López

4 de agosto de 2019

Índice general

Capítulo 1

Introducción

1.1. Interés y Objetivos

Ya sea con una ruta en kayak o prácticando kite surf, cada año, los deportes marítimos cada vez ganan más adeptos, y el abanico de propuestas se multiplican, [17] . Entre los deportes más novedosos que ofrece nuestro litoral almeriense, se encuentra el wakeboard y el paddel surf.

Todos ellos tienen un denominador común, el mar. Un denominador que lejos de presentarse apacible, produjo 115 muertos por ahogamientos en 2016, [14], ya que existen situaciones que pueden suponer un riesgo un navegante, una perdida de consciencia, acompañada de una fuerte corriente pueden desembocar en una situación de peligro, donde la alerta temprana puede suponer la diferencia entre una situación anecdótica y un resultado fatal.

Para ello, en este proyecto, planteamos un sistema de localización marítima y alerta temprana para intentar alertar lo antes posible cuando se produzca un incidente en el mar y que se pueda socorrer con diligencia evitando así accidentes mortales.

El objetivo de este proyecto es implementar una aplicación movil que funcione como cliente, emitiendo la posición GPS del dispositivo en determinadas condiciones, y como monitor para que pueda ser localizada la señal del dispositivo que solicita ayuda. [16]

Realizaremos un estudio sobre como abordar las condiciones necesarias para que el dispositivo empiece a transmitir sus coordenadas, basándonos en los siguientes requisitos:

1. El móvil está en el agua

2. La velocidad de desplazamiento del móvil es inferior a un determinado umbral durante un determinado periodo de tiempo

Cuando se cumplan dichos requisitos, el sistema comenzará a transmitir sus coordenadas y éstas llegarán a un servidor. Dicho servidor debería poder ser monitorizado, al menos, usando un móvil que ejecuta la aplicación radio faro, de manera que permitiera la localización del móvil que está emitiendo la señal de ayuda.

En caso de que se cumplan los requisitos de socorro, la aplicación radio faro comenzará una cuenta atrás de 5 minutos, en caso de no detenerla, el dispositivo móvil llamará automáticamente al número de emergencias 112, que derivará la llamada de socorro al centro más cercano.

Vamos a utilizar el sistema operativo Android [4], ya que es el más utilizado, y por los intereses de usabilidad del proyecto interesa que tenga toda la difusión posible.

Utilizaremos la herramienta de desarrollo Android Studio, proporcionada por Google. [5]

Nos decantamos por utilizar una aplicación en Android nativo, ya que, por los requisitos de utilización de funcionalidades y servicios del dispositivo, como el GPS, y la necesidad de optimizar recursos como la batería, ya que el escenario donde se va a utilizar, principalmente en entornos marinos, no disponen de accesibilidad a la red eléctrica y la utilización de accesorios como baterías externas puede no resultar funcional.

Como ya se menciona en [16] el objetivo es una aplicación móvil que funcione como cliente, emitiendo la posición GPS del dispositivo en determinadas condiciones, y como monitor para que pueda ser localizada la señal del dispositivo que solicita ayuda.

1.2. Fases de la realización del trabajo y su cronograma asociado

El proyecto se realizará siguiendo los siguientes hitos y temporización (suponiéndose una carga diaria de 2 horas/día):

1. **Investigación del estado del arte:** Recopilación de datos relacionados con mareas y vientos, determinar exactamente los parámetros para determinar si una situación es de riesgo o no. 1 mes.

2. **Estudio de viabilidad del proyecto:** Selección de las APIs que mejor se adapten a nuestras necesidades, entre ellas, una API meteorológica y otra de posicionamiento GPS. 1/2 mes.

3. **Diseño técnico de la aplicación:** Búsqueda de soluciones tecnológicas para dar respuesta a los requisitos funcionales.1/2 mes.

4. **Implementación del sistema:** Fase de desarrollo software usando herramientas de software libre. 2 meses.

5. **Fase de pruebas:** Realización de pruebas unitarias como de aceptación para comprobar el correcto funcionamiento en entorno real. Realizar anotaciones de mejoras para implementar en un futuro. 1/2 mes.

6. **Escritura de la memoria del proyecto:** 1 mes.

1.3. Conclusión

Desarrollaremos una aplicación que realice una alerta temprana de un dispositivo que se encuentre a la deriva en el mar. Tiene una funcionalidad de prevención de accidentes marítimos y puede tener una utilidad real en nuestras costas. Para ello necesitaremos llevar el móvil en una bolsa estanca con la aplicación en ejecución. Tendremos la posibilidad de añadir una batería externa al conjunto para ofrecer más horas de utilidad.

Capítulo 2

Peligros en el mar

En este capítulo vamos a describir los peligros a los que se enfrentan aquellos que practican deportes como el kitesurf, windsurf, etc, o que simplemente nadan habitualmente en el mar.
Vamos a dividirlos en tres secciones, viento, oleaje y corrientes. Y de ellas sacaremos conclusiones relevantes para nuestro proyecto.

2.1. Viento

El viento es el movimiento de aire causado por la diferencia de presiones entre las masas de aire. En nuestro caso de estudio debemos tenerlo en consideración ya que es el recurso motriz principal en deportes acuáticos con vela como el kitesurf o el windsurf. Necesitamos tener una escala para poder medir y clasificar los distintos tipos de viento, para ello usaremos la escala Beaufort. [21]

Número de Beaufort	Velocidad del viento (km/h)	Nudos (millas náuticas/h)	Denominación	Aspecto del mar	Efectos en tierra
0	0 a 1	< 1	**Calma**	Despejado	Calma, el humo asciende verticalmente
1	2 a 5	1 a 3	**Ventolina**	Pequeñas olas, pero sin espuma	El humo indica la dirección del viento
2	6 a 11	4 a 6	**Flojito** (Brisa muy débil)	Crestas de apariencia vítrea, sin romper	Se caen las hojas de los árboles, empiezan a moverse los molinos de los campos
3	12 a 19	7 a 10	**Flojo** (Brisa Ligera)	Pequeñas olas, crestas rompientes.	Se agitan las hojas, ondulan las banderas
4	20 a 28	11 a 16	**Bonancible** (Brisa moderada)	Borreguillos numerosos, olas cada vez más largas	Se levanta polvo y papeles, se agitan las copas de los árboles
5	29 a 38	17 a 21	**Fresquito** (Brisa fresca)	Olas medianas y alargadas, borreguillos muy abundantes	Pequeños movimientos de los árboles, superficie de los lagos ondulada
6	39 a 49	22 a 27	**Fresco** (Brisa fuerte)	Comienzan a formarse olas grandes, crestas rompientes, espuma	Se mueven las ramas de los árboles, dificultad para mantener abierto el paraguas
7	50 a 61	28 a 33	**Frescachón** (Viento fuerte)	Mar gruesa, con espuma arrastrada en dirección del viento	Se mueven los árboles grandes, dificultad para caminar contra el viento
8	62 a 74	34 a 40	**Temporal** (Viento duro)	Grandes olas rompientes, franjas de espuma	Se quiebran las copas de los árboles, circulación de personas muy difícil, los vehículos se mueven por sí mismos.
9	75 a 88	41 a 47	**Temporal fuerte** (Muy duro)	Olas muy grandes, rompientes. Visibilidad mermada	Daños en árboles, imposible caminar con normalidad. Se empiezan a dañar las construcciones. Arrastre de vehículos.
10	89 a 102	48 a 55	**Temporal duro** (Temporal)	Olas muy gruesas con crestas empenachadas. Superficie del mar blanca.	Árboles arrancados, daños en la estructura de las construcciones. Daños mayores en objetos a la intemperie.
11	103 a 117	56 a 63	**Temporal muy duro** (Borrasca)	Olas excepcionalmente grandes, mar completamente blanca, visibilidad muy reducida	Destrucción en todas partes, lluvias muy intensas, inundaciones muy altas. Voladura de personas y de otros muchos objetos.
12	+ 118	+64	**Temporal huracanado** (Huracán)	Olas excepcionalmente grandes, mar blanca, visibilidad nula	Voladura de vehículos, árboles, casas, techos y personas. Puede generar un huracán o tifón

Figura 1. Escala Beaufort

Entre una escala 3 y 7 serían los valores recomendados para practicar kitesurf sin riesgos, eso acota la velocidad del viento entre 12 y 61 Km/h, dato que nos será de utilidad en nuestra aplicación para hacernos una idea de los valores que recibiremos de un usuario que practique kitesurf.

Practicar este deporte con vientos superiores, puede suponer para un deportista poco experimentado una situación de riesgo en la que nuestra aplicación puede tener mucha utilidad.

2.2. Oleaje

Las olas marinas son consecuencia de la propagación del movimiento entre dos medios, el aire de la atmósfera y el agua del mar. Los cambios de presión atmosférica provocan oscilaciones en la superficie del líquido. A su vez, la acción del viento que roza la superficie da lugar a lo que se conoce como ondas capilares, cuando su empuje es más leve, u ondas gravitatorias, cuando la fricción sobre la lámina de agua es más intensa.

Podemos clasificar las olas en dos tipos:

1. **Olas de viento:** Olas generadas por el viento. Generalmente, los vientos más fuertes provocan olas más altas. Entran en juego factores como la velocidad e intensidad de la acción eólica, la cantidad de tiempo que el aire mantiene una dirección estable, el área de la superficie del agua afectada y la profundidad. A medida que las olas se acercan a la orilla, avanzan más despacio debido a que hay menos profundidad, mientras que la cresta aumenta su altura. El proceso continúa hasta que la zona levantada se mueve más rápido que la parte subacuática, punto en el que el movimiento se desestabiliza y la ola rompe.

2. **Mar de fondo:** Mar de fondo es el movimiento de las olas (también llamado oleaje o sistema de olas) que se propaga fuera de la zona donde se ha generado, pudiendo llegar a lugares muy alejados. Por tanto este estado del mar no tiene relación con el viento presente, aunque su causa es el viento que se haya originado en otra área distinta. Las olas del mar de fondo se caracterizan por su período regular y sus crestas suaves. La longitud de la onda es muy superior a su altura, presentando crestas redondeadas que no rompen nunca en alta mar.

El oleaje es otro de los factores a tener en cuenta cuando estamos en el mar. Necesitaremos también una forma de clasificar el estado de la mar en función de su oleaje, para ello utilizaremos la escala Douglas. [22]

A partir del grado 6 no es recomendable practicar deportes marinos como el windsurf, y aunque no vamos a tener en cuenta el oleaje para nuestra aplicación, si lo podemos incluir en el apartado de mejoras, añadiendo la suscripción de la aplicación a un servicio de alertas de mal tiempo, donde se verá reflejado el oleaje como uno de los indicadores para desaconsejar la práctica de deportes en el mar.

Grado	Altura de las olas (m)	Descripción	Estado del mar
0	Sin olas	Mar llana o en calma	La superficie del mar está lisa como un espejo.
1	0 a 0,10	Mar rizada	El mar comienza a rizarse por partes.
2	0,10 a 0,5	Marejadilla	Se forman olas cortas pero bien marcadas; comienzan a romper las crestas formando una espuma que no es blanca sino de aspecto vidroso (ovejas).
3	0,5 a 1,25	Marejada	Se forman olas largas con crestas de espuma blanca bien caracterizadas. El mar de viento está bien definido y se distingue fácilmente del mar de fondo que pudiera existir. Al romper las olas producen un murmullo que se desvanece rápidamente.
4	1,25 a 2,5	Fuerte marejada	Se forman olas más largas, con crestas de espuma por todas partes. El mar rompe con un murmullo constante.
5	2,5 a 4	Gruesa	Comienzan a formarse olas altas; las zonas de espuma blanca cubren una gran superficie. Al romper el mar produce un ruido sordo como de arrojar cosas.
6	4 a 6	Muy gruesa	El mar se alborota. La espuma blanca que se forma al romper las crestas comienza a disponerse en bandas en la dirección del viento.
7	6 a 9	Arbolada	Aumentan notablemente la altura y la longitud de las olas y de sus crestas. La espuma se dispone en bandas estrechas en la dirección del viento.
8	9 a 14	Montañosa	Se ven olas altas con largas crestas que caen como cascadas; las grandes superficies cubiertas de espuma se disponen rápidamente en bandas blancas en la dirección del viento, el mar alrededor de ellas adquiere un aspecto blanquecino.
9	Más de 14	Enorme	Las olas se hacen tan altas que a veces los barcos desaparecen de la vista en sus senos. El mar está cubierto de espuma blanca dispuesta en bandas en la dirección del viento y el ruido que se produce es fuerte y ensordecedor. El aire está tan lleno de salpicaduras, que la visibilidad de los objetos distantes se hace imposible.

Figura 2. Escala Douglas

2.3. Corrientes marinas

Una de las corrientes marinas que más impacto puede tener en nuestro baño en el mar, es la corriente de resaca. [20] Una corriente de resaca o corriente de retorno es una fuerte corriente superficial (o casi superficial) de agua, que retrocede desde la costa hacia el mar. Se genera principalmente por el rompimiento irregular de las olas a lo largo de la cresta, llegando bruscamente a la playa con un índice elevado de energía, desvaneciéndose luego sobre el fondo para, posteriormente, regresar hacia el mar por un canal a través de las olas.

Su intensidad depende de la altura de las olas y de las características topográficas de la orilla, siendo además reforzadas por las corrientes de marea, por lo que se hacen más peligrosas en bajamar. Estas corrientes pueden ser visibles o no dependiendo de la intensidad de la corriente y del tipo de sedimento que se encuentra en la playa.

Figura 1. Corriente de resaca

Aunque puede no ser un factor de riesgo en deportes acuaticos de vela como el windsurf o kitesurf, es un elemento a tener en cuenta en otras modalidades sin vela como el surf o el paddle surf. Sobretodo si estamos cansados

después de una sesión de dos horas o más de deporte, es posible que no podamos salir de las corrientes con facilidad incluso quedar a la deriva por una lipotimia. Este es un caso en el que nuestra aplicación puede tener una utilidad real.

Fuente: Federación balear de vela [1]

Capítulo 3

Revisión bibliográfica

Los nuevos tiempos cambian y con ella los instrumentos usados en la navegación sobre todo desde que la electrónica irrumpió de forma intensiva .

Nos tenemos que remontar hasta la aparición de la normativa NMEA,la cual permitía concentrar información dispersa en una sola pantalla procedente de cada uno de los componentes de la electrónica de a bordo (corredera, equipo de viento, sonda, radar, posición),sin importar su fabricante o el modelo

Posteriormente la aparición de otros elementos como el AIS(Automatic Identification System), GPS y plotters de alta definición nos han llevado a recurrir a nuevas herramientas y dispositivos para presentar toda la información de ayuda a la navegación a través de un PC con programas aplicaciones de cartografía escaneada o vectorizada (como OziExplorer, Navionics o Tsunamis), que nos mostraban gracias al estándar NMEA, toda la información que teníamos distribuida en los diversos dispositivos electrónicos instalados a bordo.

Actualmente la mayoría de los navegantes han pasado del papel a las cartas leídas en el ordenador (y ,del compás al ratón, y de la corredera al GPS) , pero aun toca asimilar el siguiente paso :integrar todas las herramientas de ayuda a la navegación en nuestros dispositivos móviles .

Hoy en día los sistemas de geolocalización ayudados de una conexión a internet han revolucionado el transporte , de modo que con un smartphone o tableta podemos tener a nuestro alcance, en un multi-dispositivos a muy bajo coste, toda la información que hace unos años sólo podíamos obtener usando electrónica bastante compleja y costosa de modo que con la evolución y desarrollo de las tabletas Android, las necesidades de electrónica y su utilidad en las embarcaciones ha supuesto un cambio en cuanto a los hábitos de muchos navegantes, ganando cada vez más importancia la tableta como el elemento principal para contener la información que se necesita para la

navegación,

Entre las aplicaciones náuticas del mercado que vamos a ver algunas requieren una conexión a Internet, pero muchas otras ofrecen la opción de trabajar sin cobertura, usando señales de posicionamiento GPS o de los sensores internos (acelerómetro o brújula)

Veamos una muestra de algunas de las aplicaciones mas usadas :

1. **Seguridad**

   La app gratuita de Salvamento Marítimo también es indispensable entre las aplicaciones náuticas.

   a) **Salvamento Marítimo:** Con esta aplicación móvil obtenemos información y orientación sobre cuestiones relacionadas con la seguridad náutica. Por ejemplo, nos ofrece la posibilidad de realizar una serie de controles para ver si estamos cumpliendo con los puntos de seguridad que necesitamos cumplir antes de salir a navegar. Desde consejos y todo tipo de información para navegantes, hasta listados para controlar los puntos esenciales de seguridad antes de la navegación.

   Esta aplicación permite a los navegantes de recreo que Salvamento Marítimo monitorice su travesía y que de la alarma en caso de no cumplir con el plan de navegación establecido.

   El 50 % de las incidencias que atiende Salvamento Marítimo, son de embarcaciones de recreo.

   b) **SafeTrx:** SafeTrx es una aplicación para Smartphone, tanto para dispositivos Android como Apple iOS (iPhone, iPad), que monitoriza los viajes de su embarcación, avisa a los contactos designados por el usuario cuando hay retrasos en el viaje planificado y ofrece una página web para que Salvamento Marítimo pueda consultar con rapidez la derrota realizada por la embarcación y tomar las acciones oportunas.

   Esta aplicación complementa al Sistema Mundial de Socorro y Seguridad Marítima.

2. **Aplicaciones de AIS**

   En los últimos años hemos visto cómo han aparecido algunas aplicaciones que nos ayudan a identificar las embarcaciones. Se trata de aplicaciones que usan el AIS (Automatic Identification System). Gracias a este sistema podremos conocer la identidad, el rumbo, la posición y más datos sobre aquellas embarcaciones que estén situadas en la zona próxima a la que nos encontramos nosotros.

Habitualmente esto se había realizado a través de sistemas de radio VHF, pero actualmente gracias a una conexión a internet se puede reemplazar su cometido con aplicaciones como MarineTrafic,

a) **mAIS marine Traffic Ship Position Reporting:** Esta App basada en Google Maps, nos permiten enviar y visualizar información a tiempo real de la señal AIS. Para ello tan solo necesitamos tener acceso a una conexión de internet y gracias al soporte de Google Maps, pueden visualizarse los barcos en cualquier lugar del mundo. Para emitir la señal es preciso tener una conexión a Internet, esta emisión de señal hay que pararla al llegar a tierra.

b) **Ship Finder – Live vessel tracking:** Con esta App también puede obtenerse la información sobre todas las embarcaciones situadas en la zona seleccionada. Hay una versión gratuita y otra más completa de pago.

3. **Aplicaciones náuticas meteorológicas**

Todos los navegantes están pendientes de la previsión meteorológica. Hoy en día tenemos acceso a datos a tiempo real sobre la evolución de los indicadores básicos en meteorología, así como a los pronósticos. En el universo de las aplicaciones encontramos una gran cantidad de herramientas que nos ayudarán a conocer las condiciones de viento y previsión de lluvia con gran lujo de detalles.

a) **WeatherPRO HD:** es una aplicación realizada por Meteogrup que ofrece un pronóstico a una semana vista, en intervalos de tres horas, en más de dos millones de lugares, con radar en tiempo real e imágenes por satélite.

b) **WinFinderPRO:** para los aficionados a los deportes de vela es una aplicación fundamental. Viento, olas y predicción meteorológica son los principales reclamos de esta app. Su uso es bastante parecido a la WeatherPRO, con la posibilidad de tener nuestra lista de favoritos.

c) **PocketGrib:** Esta app hace posible visualizar todos los datos meteorológicos en una previsión de hasta ocho días, con un elevado nivel de acierto en las previsiones de viento.Basada en su hermano mayor diseñado para PC, uGRib (www.grib.us) permite ver todos los datos meteorológicos con una previsión de hasta ocho días. Seleccionamos directamente en la pantalla la zona sobre la que queremos obtener la previsión, y nos muestra de forma gráfica las

previsiones de viento con una alta precisión y un alto grado de acierto.

d) **WindGuru:** Se trata de una aplicación muy conocida para los navegantes entre las aplicaciones náuticas, que nos muestra de forma gráfica y muy rápida las previsiones de viento y temperatura en la zona que seleccionemos, con un alto grado de fiabilidad.

e) **Eltiempo.es:** La conocida página de internet de Jose Antonio Maldonado tiene su APP, con una visualización panorámica, hora a hora, de vientos, oleajes y temperaturas en la zona de navegación que se seleccione, con unos datos gráficos de gran exactitud.

f) **Rain Alarm:** Mediante esta APP, pueden recibirse los avisos correspondientes mediante alertas, de aquellas lluvias que se acercan, con imágenes animadas sobre la meteorología a nivel mundial.

4. **Aplicaciones náuticas de geolocalización**

a) **Boating, aplicación de Navionics:** es la app náutica de cartografía más vendida en el mundo, y ofrece todas las posibilidades de planificar rutas con facilidad y comodidad, y en compatibilidad con Google Earth y otras aplicaciones de meteorología. Existe una versión HD con más funciones. Se puede usar sin conexión a internet, si previamente a descargado las cartas.

b) **Navionics HD:** Navionics es el software de cartografía más demandado y vendido en todo el mundo. Ha tenido una gran evolución desde su origen, permitiendo utilizar la tableta como un auténtico ploter, con las ventajas de poder moverlo por todo el barco e incluso planificar rutas o ver la información desde cualquier sitio. La versión HD tiene una precisión impresionante y se integra con Google Earth y con ficheros Grib de meteorología, además de información en tiempo real de servicios cercanos a nuestra ubicación (marinas, mecánicos, restaurantes...). Su uso es muy sencillo y no requiere nada más que instalar la App y comenzar a navegar. Puede trabajar sin conexión a internet, cargando previamente las cartas.

c) **Google Earth:** La aplicación del gigante de las búsquedas en Internet no puede faltar. Con toda precisión se puede visualizar la ruta a seguir, así como fotos precisas del lugar. Algunas de las demás App se integran con Google earth. No puede faltar.

d) **Gabenative:** Es otra aplicación náutica que muchos aficionados

a la navegación suelen tener instalada, y es realmente útil para ayudarnos a situar en el medio del mar.

5. **Instrumentación**

   a) **Polaris Navegación GPS:** Convierte un teléfono en un potente sistema de navegación GPS de uso general con Polaris Navigation GPS. Utilizado principalmente como una aplicación de GPS de náutica y rastro, Polaris es un excelente respaldo o reemplazo para su Garmin u otra unidad GPS portátil de navegación por satélite.

   b) **Antigarreo:** Una herramienta para ayudarnos en el fondeo,una de las variables que más influye en la serenidad del barco es el garreo. Con esta aplicación, el navegante recibirá las alertas necesarias para que la embarcación no salga de su espacio de borneo.

   c) **Bearing Pilot:** Se trata de una aplicación para evitar las demoras en la navegación, prever rumbos y eliminar el riesgo de colisiones, cálculos de rumbos de viento, mantenimiento de la dirección, etc. Una nueva herramienta dirigida tanto a navegantes principiantes como a los más profesionales, su extremada sencillez nos permitirá tomar marcaciones de forma rápida, mostrándonos en su interfaz gráfica la dirección de la demora que hemos tomado, el ángulo respecto a nuestro rumbo y también nos indicará hacia que banda debemos caer para poner proa a dicha demora.

   d) **Vaavud:** Este app usa un dispositivo de plástico resistente al agua, polvo y a la arena. Se conecta al jack de sonido para medir la velocidad del viento de una forma precisa en todas las direcciones, promedios, velocidades actuales, etc. Vaavud se compra por internet directamente en la página del fabricante (vaavud.com) por unos 40 Euros. Permite medir de forma muy precisa y fiable la velocidad del viento en cualquier dirección, convirtiendo nuestro dispositivo iPhone o Android en una pequeña estación meteorológica.Además, utilizando la App gratuita de Vaavud, podemos ver las mediciones de la velocidad del viento de otros usuarios registrados en otras ubicaciones.El dispositivo puede medir con precisión la velocidad del viento en un rango de 1 m/s hasta 25 m/s, y ha sido probado y calibrado profesionalmente en un túnel de viento en la Universidad Técnica de Dinamarca.

   e) **NMEA Remote:** Esta App funciona a través de un módem wifi puerto serie que hay que instalar previamente, y actúa como un repetidor de datos de la embarcación.

6. **Otras aplicaciones**

   a) **Marinus Ripa:** Marinus Ripa aporta el Reglamento Internacional de Prevención de Abordajes, una actualización permanente, y toda la información básica de seguridad que un buen navegante debe conocer. Además, se puede obtener también una versión en iBook para iPad en formato libro de alta definición.

   b) **MoonPreview:** Con esta app es posible conocer el ciclo lunar y el estado de la luna la noche que está planificado el fondeo de la embarcación. Ideal para viajes de recreo y contemplación de la cúpula celeste.

   c) **Useful knots:** Para cada necesidad hay una variedad de nudo distinta. Esta App nos enseña y recuerda, paso a paso, la correcta forma de realizar hasta cien nudos diferentes, según sus tipos y utilidades.

   d) **My112:** Esta aplicación permite comunicarte con el Centro 112 de Emergencias, enviando tu posición actual al operador que te está atendiendo, ayudando en tu localización. Además, My112 recibe en tiempo real avisos de emergencias cercanas a tu posición e información actualizada de las mismas en el momento de producirse.

Conclusión

Como podemos ver en el análisis previo de distintas aplicaciones, hay muchas cuya función es el posicionamiento marítimo, la obtención de información meteorológica y que ofrecen seguridad en forma de avisos o llamadas de emergencia.
Hemos realizado una búsqueda pero no hemos logrado encontrar ninguna que unifique los criterios de malas condiciones meteorológicas y alerta temprana de situaciones de riesgo, que es el objetivo que pretendemos conseguir, lo que creemos que podría hacerse un hueco en el saturado mercado de las aplicaciones móviles.

Capítulo 4

Material y métodos

En esta sección describiremos el trabajo realizado.

4.1. Tecnologías empleadas

En esta sección desglosaremos las tecnologías que utilizaremos para realizar el proyecto.

4.1.1. Java

Java es un lenguaje de programación orientado a objetos de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser compilado de nuevo para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados. [24]

Mecanismos básicos

1. **Objetos:** Es una abstracción encapsulada genérica de datos y los procedimientos para manipularlos.

2. **Mensajes:** Los objetos se comunican a través de señales o mensajes, siendo estos mensajes los que hacen que los objetos respondan de diferentes maneras.

3. **Métodos:** Es una acción que determina como debe de actuar un objeto cuando recibe un mensaje.

4. **Clases:** Es la generalización de un tipo específico de objetos, es decir, es el conjunto de características y comportamientos de todos los objetos que componen la clase.

Principios fundamentales

1. **Abstracción:** Es el proceso de representar entidades reales como elementos internos a un programa, la abstracción de los datos es tanto en los atributos, como en los métodos de los objetos.

2. **Encapsulamiento:** Cada objeto está aislado del exterior, esta característica permite verlo como una caja negra, que contiene toda la información relacionada con ese objeto. Este aislamiento protege a los datos asociados a un objeto para que no se puedan modificar por quien no tenga derecho a acceder a ellos. Permite manejar a los objetos como unidades básicas, dejando oculta su estructura interna.

3. **Herencia:** Es el medio para compartir en forma automática los métodos y los datos entre las clases y subclases de los objetos. Los objetos

heredan las propiedades y el comportamiento de todas las clases a las que pertenece.

4. **Polimorfismo:** Esta característica facilita la implementación de varias formas de un mismo método, con lo cual se puede acceder a varios métodos distintos, que tienen el mismo nombre. [24]

Entornos de funcionamiento

El diseño de Java, su robustez de la industria y su fácil portabilidad han hecho de Java uno de los lenguajes con un mayor crecimiento y amplitud de uso en dispositivos ámbitos de la industria de la informática. Los distintos entornos donde podemos ejecutar una aplicación Java son:

1. En dispositivos móviles y sistemas embebidos
2. En el navegador web
3. En sistemas de servidor
4. En aplicaciones de escritorio

4.1.2. Android

Utilizaremos el sistema operativo Android como entorno para implementar nuestra aplicación. Android es un sistema operativo cuyo núcleo está basado en Linux, sistema libre, gratuito y multiplataforma. El sistema operativo fue desarrollado en Octubre de 2003 por Android Inc., y posteriormente comprado por Google en 2005. Aunque principalmente fue diseñado para dispositivos móviles con pantalla táctil como teléfonos inteligentes o tablets, actualmente también se utiliza para dispositivos wearables, televisiones e incluso para automóviles.

Android permite realizar aplicaciones en un lenguaje basado en Java que utiliza como JVM (Java Virtual Machine) ART. Su predecesor es Dalvik.[7]

Esta máquina virtual proporciona una serie de mejoras respecto a la jvm habitual de java:

1. Compilación AOT (Ahead-of-time): Compila utilizando la herramienta dex2oat. Mejora el rendimiento y minimiza el tiempo de instalación de aplicaciones.

2. Mejora el recolector de basura.

3. Mejoras para el desarrollo y la depuración:

   a) Soporta perfiles de pruebas

   b) Añade nuevas características para depurar, particularmente en el monitor de recursos y el recolector de basura

   c) Mejoras en la descripción de los errores, añadiendo detalles como por ejemplo, información relativa a la actividad que intentaba realizar antes de un error de tipo java.lang.NullPointerException

ART nos proporciona un lenguaje de programación con acceso de una forma sencilla a las funcionalidades del teléfono (como GPS, llamadas, agenda, etc.).

Entre las características más importantes de este sistema operativo, es que tiene licencia Open Source, es decir, tanto el código fuente como los archivos binarios pueden ser modificados y redistribuidos libremente, sin tener que pagar al autor original. Esto lo hace muy popular entre fabricantes y desarrolladores, ya que los costes disminuyen mucho a la hora de lanzar un teléfono o realizar una aplicación.

Arquitectura Android

En este apartado vamos a ver una visión global por capas de la arquitectura empleada por Android. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores, tal y como podemos ver en la siguiente figura.

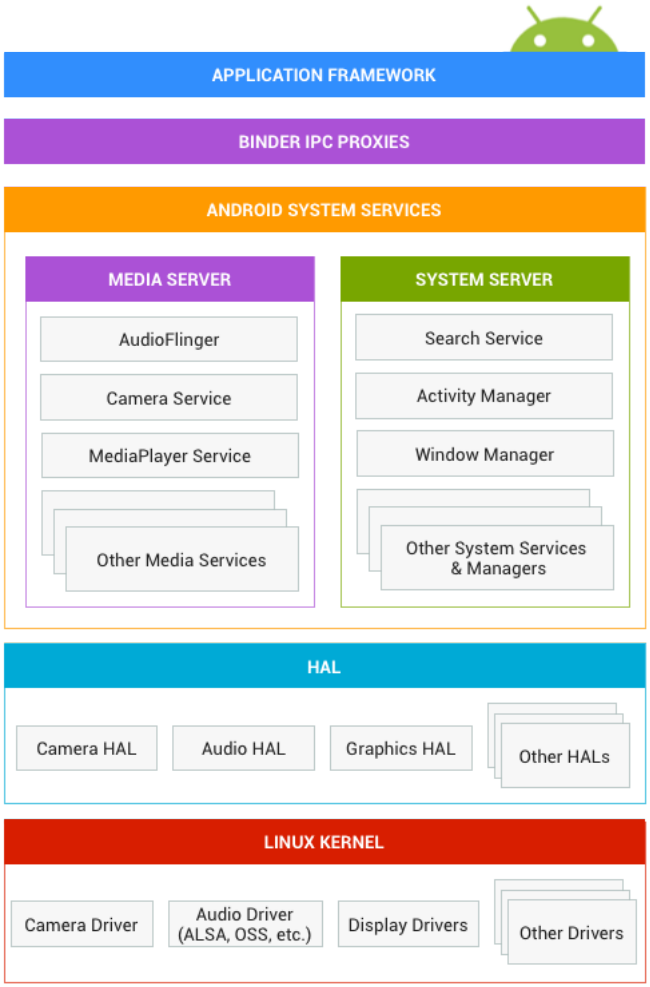

Figura 3. Arquitectura del sistema Android

1. Aplicaciones: Este nivel contiene, tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

2. Framework de Aplicaciones: Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo "framework", representado por este nivel. Entre las API más importantes ubicadas aquí, se pueden encontrar las siguientes:

   a) Activity Manager: Conjunto de API que gestiona el ciclo de vida de las aplicaciones en Android.

   b) Window Manager: Gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.

   c) Content Provider: Permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.

   d) View System: Proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, çheck-boxes", tamaño de ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.

   e) Notification Manager: Mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una llamada entrante, un mensaje recibido, conexión Wifi disponible, ubicación en un punto determinado, etc. Si llevan asociada alguna acción, en Android denominada Intent, (por ejemplo, atender una llamada recibida) ésta se activa mediante un simple clic.

   f) Package Manager: Permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes

   g) Telephone Manager: Incluye todas las API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).

   h) Resource Manager: Permite gestionar los elementos (cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o layouts) que forman parte de la aplicación y que están fuera del código.

   i) Location Manager: Posibilita a las aplicaciones la obtención de información de localización y posicionamiento.

3. XMPP Service: Colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

4. Librerías: La siguiente capa se corresponde con las librerías utilizadas por Android. Estas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android. [7] Entre las librerías más importantes ubicadas aquí, se pueden encontrar las siguientes:

   a) Surface Manager: Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

   b) Media Framework: Proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)

   c) SQLite: Creación y gestión de bases de datos relacionales.

   d) OpenGL/SL y SGL: Representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.

   e) FreeType: Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

   f) WebKit: Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

   g) SSL: Posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.

   h) Libc: Incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.

5. Tiempo de ejecución de Android: Al mismo nivel que las librerías de Android se sitúa el entorno de ejecución. Éste lo constituyen las Core

Libraries, que son librerías con multitud de clases Java y la máquina virtual ART. [7]

6. Núcleo Linux: Android utiliza el núcleo de Linux como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android.

Componentes Android

Existen cinco componentes importantes en las aplicaciones Android

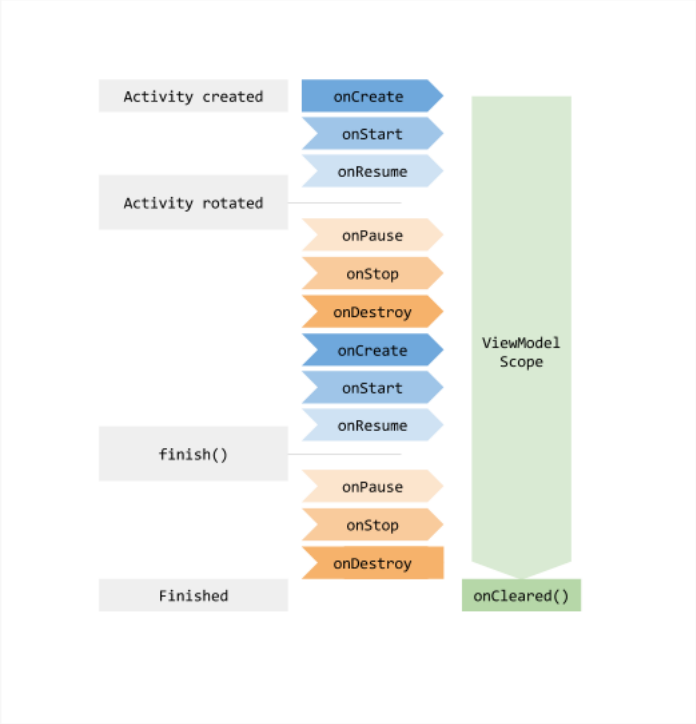

Figura 1. Ciclo de vida de una aplicación Android

1. Activity: Se puede decir que una actividad es cada una de las pantallas que nosotros mostramos en la ejecución de nuestra aplicación. Cada actividad puede encontrarse en diferentes estados:

   a) onCreate: Representa el momento en que se crea la actividad

   b) onStart: Es donde la actividad se muestra en pantalla.

   c) onRestart: Cuando una aplicación estaba parada y la volvemos a activar.

   d) onResume: La actividad va a empezar a responder a la interacción del usuario.

   e) onPause: La actividad va a dejar de responder a la interacción del usuario.

   f) onStop: Cuando la actividad pasa a segundo plano.

   g) onDestroy: Cuando se destruye una actividad y se liberan sus recursos.

   Una actividad consta de dos partes:

   a) La parte lógica: se trata de un archivo java donde se manipula, interactúa y coloca el código de la actividad

   b) La parte gráfica: Se trata de un XML donde se introducen los elementos que formarán la estructura de la pantalla.

2. Services: Son tareas que se ejecutan en segundo plano y no tienen una interfaz que vea el usuario. Se suelen utilizar para realizar operaciones de larga ejecución o para realizar trabajo con procesos remotos. Se encargan de realizar tareas que deben continuar ejecutándose cuando nuestra aplicación no está en primer plano.

3. Content Provider: Se trata de un mecanismo proporcionado por la plataforma Android para permitir compartir información entre aplicaciones. Se permite acceder al sistema de ficheros, bases de datos SQLite, la información se encuentran la lista de contactos, la aplicación de SMS, o el calendario/agenda.

4. Intent: Son objetos que se utilizan para arrancar actividades o servicios, o enviar eventos a múltiples destinatarios. Estos objetos también se pueden utilizar para realizar paso de mensajes entre actividades de la misma aplicación o entre distintas aplicaciones. Broadcast Receiver: Un broadcast es un mensaje que cualquiera aplicación puede recibir. Estos mensajes broadcast pueden ser un reinicio del dispositivo, aviso de batería baja o al conectar el móvil a un cargador. Con este componente se puede configurar el recibir ese broadcast y que nuestra aplicación haga alguna acción determinada en ese momento.

Estructura de una aplicación Android

En este apartado vamos a explicar la estructura de los componentes que tiene una aplicación Android.

1. Src: En esta carpeta se introducen los archivos java que componen nuestra aplicación.

2. Gen: Dentro de esta carpeta hay un archivo llamado "R.java" donde se guardan los identificadores de todos los componentes utilizados en la aplicación.

3. Libs: Se encuentran librerías externas que necesita el proyecto.

4. Res: En este directorio se encuentran todos los recursos de la aplicación.

5. Res/drawable: En esta carpeta se guardan las imágenes que vamos a utilizar en nuestra aplicación.

6. Res/layout: Carpeta donde se guardan los XML de las actividades.

7. Res/values: Carpeta donde se guarda el archivo "strings.xml", el cual almacena las cadenas de texto que utilizaremos en la aplicación. AndroidManifest.xml: Este archivo es el más importante de la aplicación y lo vamos a describir con más detalle en el siguiente apartado. Android Manifest Éste archivo se considera el más importante de la aplicación ya que es donde se declaran todas las actividades de la aplicación, los permisos, versiones del SDK que usamos, etc.

   En la primera línea se indica la versión y el formato del documento:

   La etiqueta principal es <manifest>y tiene varios atributos:

Xmlns : lo coloca Eclipse por defecto.

Package : Hace referencia al nombre del paquete de la aplicación.

VersionCode : Es el número de versión del código de la aplicación

VersionName : También es el número de la versión

   La etiqueta SDK se configura para saber la compatibilidad de las versiones en Android.

   a) minSDKVersion: Indica la versión mínima de SDK para que funcione nuestra aplicación.

   b) targetSdkVersion: Es la versión de la API a la que se dirige principalmente la aplicación.

La etiqueta <uses-permission>se utiliza para dar permisos a la aplicación. En nuestro caso, hemos configurado para que nuestra aplicación pueda acceder a internet, pueda acceder al GPS y pueda escribir en un almacenamiento externo.

Dentro de la etiqueta <application>se introducen todos los elementos que componen la aplicación.

a) allowBackup: Con el valor "true"se le permite al sistema hacer copia de seguridad de la aplicación y del contenido.

b) Icon: Es el icono de la aplicación.

c) Label: Es el nombre de la aplicación.

d) Theme: Es el estilo de la aplicación.

En la etiqueta <activity>es donde se dan de alta todas las actividades que vamos a tener en la aplicación.

a) Name: Contiene el nombre de la clase java que implementa el activity.

b) Label: Es el nombre de la activity.

<Intent-filter>indica lo que la activity tiene permitido hacer.

4.1.3. Sistema de posicionamiento global

Utilizaremos el sistema GPS para localizar el dispositivo móvil en el agua. A continuación describiremos brevemente en que se base este sistema.

El Sistema de Posicionamiento Global (en inglés, GPS; Global Positioning System), y originalmente Navstar GPS, es un sistema que permite determinar en toda la Tierra la posición de un objeto (una persona, un vehículo) con una precisión de hasta centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión. El sistema fue desarrollado, instalado y empleado por el Departamento de Defensa de los Estados Unidos. Para determinar las posiciones en el globo, el sistema GPS se sirve de 24 a 32 satélites y utiliza la trilateración.

El GPS funciona mediante una red de como mínimo 24 satélites en órbita sobre el planeta Tierra, a 20 180 km de altura, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra. Cuando se desea determinar la posición tridimensional, el receptor que se utiliza para ello localiza automáticamente como mínimo cuatro satélites de la red, de los que recibe unas señales indicando la identificación y hora del reloj de cada uno de ellos, además de información sobre la constelación. Con base en estas señales, el aparato sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo, y de tal modo mide la distancia al satélite mediante el método de trilateración inversa, el cual se basa en determinar la distancia de cada satélite al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o coordenadas reales del punto de medición. También se consigue una gran exactitud en el reloj del GPS, similar a la de los relojes atómicos que lleva a bordo cada uno de los satélites.

Funcionamiento

La información que es útil al receptor GPS para determinar su posición se llama efemérides. En este caso cada satélite emite sus propias efemérides, en la que se incluye la salud del satélite,su posición en el espacio, su hora atómica, información doppler, etc.

Mediante la trilateración se determina la posición del receptor:

- Cada satélite indica que el receptor se encuentra en un punto en la superficie de la esfera, con centro en el propio satélite y de radio la distancia total hasta el receptor.

- Obteniendo información de dos satélites queda determinada una circunferencia que resulta cuando se intersecan las dos esferas en algún punto de la cual se encuentra el receptor.

- Teniendo información de un tercer satélite, se elimina el inconveniente de la falta de sincronización entre los relojes de los receptores GPS y los relojes de los satélites. Y es en este momento cuando el receptor GPS puede determinar una posición 3D exacta (latitud, longitud y altitud).

Integración con telefonía móvil

Actualmente dentro del mercado de la telefonía móvil la tendencia es la de integrar, por parte de los fabricantes, la tecnología GPS dentro de sus dispositivos. El uso y masificación del GPS está particularmente extendido en los teléfonos móviles smartphone, lo que ha hecho surgir todo un ecosistema de software para este tipo de dispositivos, así como nuevos modelos de negocios que van desde el uso del terminal móvil para la navegación tradicional punto-a-punto hasta la prestación de los llamados Servicios Basados en la Localización (LBS).

Un buen ejemplo del uso del GPS en la telefonía móvil son las aplicaciones que permiten conocer la posición de amigos cercanos sobre un mapa base. Para ello basta con tener la aplicación respectiva para la plataforma deseada (Android, Bada, IOS, WP, Symbian) y permitir ser localizado por otros.
[1]

[1]Fuente : GPS [23]

4.1.4. JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituído por dos estructuras:

1. Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.

2. Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o sequencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.
[2]

[2]Fuente : [12]

4.1.5. Firebase

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles. Firebase se integra actualmente con otros servicios de Google para poder ofrecer productos a mayor escala para los desarrolladores.
Los servicios ofrecidos por Firebase son los siguientes:

1. **Firebase Analytics:** es una aplicación gratuita que proporciona una visión profunda sobre el uso de la aplicación por parte de los usuarios.

2. **Firebase Cloud Messaging:** Antiguamente conocido como Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) es una plataforma para mensajes y notificaciones para Android, iOS, y aplicaciones web que actualmente puede ser usada de forma gratuita.

3. **Firebase Auth:** es un servicio que puede autenticar los usuarios utilizando únicamente código del lado del cliente. Incluye la autenticación mediante Facebook, GitHub, Twitter y Google. Además, incluye un sistema de administración del usuario por el cual los desarrolladores pueden habilitar la autenticación de usuarios con email y contraseña que se almacenarán en Firebase.

4. **Realtime Database:** Firebase proporciona una base de datos en tiempo real y back-end. El servicio proporciona a los desarrolladores de aplicaciones una API que permite que la información de las aplicaciones sea sincronizada y almacenada en la nube de Firebase. La compañía habilita integración con aplicaciones Android, iOS, JavaScript, Java, Objective-C, Swift y Node.js. La base de datos es también accesible a través de una REST API e integración para varios sistemas de Javascript como AngularJS, React, Ember.js y Backbone.js.La REST API utiliza el protocolo SSE (del inglés Server-Sent Events), el cual es una API para crear conexiones de HTTP para recibir notificaciones push de un servidor.

5. **Firebase Storage:** proporciona cargas y descargas seguras de archivos para aplicaciones Firebase, sin importar la calidad de la red. El desarrollador lo puede utilizar para almacenar imágenes, audio, vídeo, o cualquier otro contenido generado por el usuario. Firebase Storage se basa en el almacenamiento de Google Cloud Storage.

6. **Firebase Firestore:** es un servicio derivado de Google Cloud Platform, adaptado a la plataforma de Firebase. Al igual que Realtime Database, es una base de datos NoSQL, aunque presenta diversas diferencias. Se organiza en forma de documentos agrupados en colecciones,

y en ellos se pueden incluir tanto campos de diversos tipos (cadenas de texto, números, puntos geográficos, referencias a la propia base de datos, arrays, booleanos, marcas de tiempo, e incluso objetos propios) como otras subcolecciones.

Utilizaremos Realtime Database ya que nos ofrece la estabilidad de un producto muy probado y usado y muy baja latencia, por lo que es una excelente opción para la sincronización frecuente de estados.

Bases de datos no relacionales NoSQL

Empezamos definiendo el concepto de base de datos no relacionales NoSQL.

Pese a la no existencia de una definición formal, cuando hablamos de base datos NoSQL nos referimos a una amplia clase de sistemas de gestión de datos (mecanismos para el almacenamiento y recuperación de datos) que difieren, en aspectos importantes, del modelo clásico de relaciones entre entidades (o tablas) existente en los sistemas de gestión bases de datos relacionales, siendo el más destacado el que no usan SQL como lenguaje principal de consulta. Aunque son conocidas desde la década de los 60 del pasado siglo, su auge actual viene determinado por el uso que, de estos sistemas han hecho las principales compañías de internet como Amazon, Google, Twitter y Facebook. Estas compañías tenían que enfrentarse a nuevos desafíos en el tratamiento de los datos motivados por el enorme crecimiento de la Web donde se requería dar respuesta a la necesidad de proporcionar información procesada a partir de grandes volúmenes de datos con unas estructuras horizontales, más o menos, similares y con aplicaciones web que debían dar respuesta a las peticiones de un número elevado e indeterminado de usuarios en el menor tiempo posible. Estas compañías se dieron cuenta de que el rendimiento y sus necesidades de tiempo real eran más importantes que la consistencia de los datos, aspecto este último al que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de proceso.

Las características comunes entre todas las implementaciones de bases de datos NoSQL suelen ser las siguientes:

1. **Consistencia Eventual:** A diferencia de las bases de datos relacionales tradicionales, en la mayoría de sistemas NoSQL, no se implementan mecanismos rígidos de consistencia que garanticen que cualquier cambio llevado a cabo en el sistema distribuido sea visto, al mismo tiempo, por todos los nodos y asegurando, también, la no violación de posibles restricciones de integridad de los datos u otras reglas definidas. En su lugar y para obtener un mayor rendimiento, se ofrece el concepto de çonsistencia eventual", en el que los cambios realizados çon el tiempo"serán propagados a todos los nodos por lo que, una consulta podría no devolver los últimos datos disponibles o proporcionar datos inexactos, problema conocido como lecturas sucias u obsoletas. Asimismo, en algunos sistemas NoSQL se pueden presentar perdidas de datos en escritura. Esto se conoce también como BASE (Basically Available Soft-state Eventual Consistency), en contraposición a ACID (Atomicity, Consistency, Isolation, Durability), su analogía en las bases de datos relacionales.

2. **Flexibilidad en el esquema:** En la mayoría de base de datos NoSQL, los esquemas de datos son dinámicos; es decir, a diferencia de las bases de datos relacionales en las que, la escritura de los datos debe adaptarse a unas estructuras(o tablas, compuestas a su vez por filas y columnas) y tipos de datos pre-definidos, en los sistemas NoSQL, cada registro (o documento, como se les suele llamar en estos casos) puede contener una información con diferente forma cada vez, pudiendo así almacenar sólo los atributos que interesen en cada uno de ellos, facilitando el polimorfismo de datos bajo una misma colección de información. También se pueden almacenar estructuras complejas de datos en un sólo documento, como por ejemplo almacenar la información sobre una publicación de un blog (título, cuerpo de texto, autor, etc) junto a los comentarios y etiquetas vertidos sobre el mismo, todo en un único registro.

3. **Escalabilidad horizontal:** Por escalabilidad horizontal se entiende la posibilidad de incrementar el rendimiento del sistema añadiendo, simplemente, más nodos (servidores) e indicando al sistema cuáles son los nodos disponibles.

4. **Estructura distribuida:** Generalmente los datos se distribuyen, entre los diferentes nodos que componen el sistema. Hay dos estilos de distribución de datos:

    a) **Particionado (ó Sharding):** El particionado distribuye los datos entre múltiples servidores de forma que, cada servidor, actúe como única fuente de un subconjunto de datos. Normalmente, a la hora de realizar esta distribución, se utilizan mecanismos de tablas de hash distribuidas (DHT).

    b) **Réplica:** La réplica copia los datos entre múltiples servidores, de forma que cada bit de datos pueda ser encontrado en múltiples lugares. Esta réplica puede realizarse de dos maneras: Réplica maestro-esclavo en la que un servidor gestiona la escritura de la copia autorizada mientras que los esclavos se sincronizan con este servidor maestro y sólo gestionan las lecturas. Réplica peer-to-peer en la que se permiten escrituras a cualquier nodo y ellos se coordinan entre sí para sincronizar sus copias de los datos

5. **Tolerancia a fallos y Redundancia:** Pese a lo que cualquiera pueda pensar cuando se habla de NoSQL, no todas las tecnologías existentes bajo este paraguas usan el mismo modelo de datos ya que, al ser sistemas altamente especializados, la idoneidad particular de una base de datos NoSQL dependerá del problema a resolver. Así a todo, podemos

agrupar los diferentes modelos de datos usados en sistemas NoSQL en cuatro grandes categorías:

a) **Base de datos de Documentos:** Este tipo de base de datos almacena la información como un documento, usando para habitualmente para ello una estructura simple como JSON, BSON o XML y donde se utiliza una clave única para cada registro. Este tipo de implementación permite, además de realizar búsquedas por clave–valor, realizar consultas más avanzadas sobre el contenido del documento. Son las bases de datos NoSQL más versátiles.

b) **Almacenamiento Clave-Valor:** Son el modelo de base de datos NoSQL más popular, además de ser la más sencilla en cuanto a funcionalidad. En este tipo de sistema, cada elemento está identificado por una clave única, lo que permite la recuperación de la información de forma muy rápida, información que suele almacenarse como un objeto binario. Se caracterizan por ser muy eficientes tanto para las lecturas como para las escrituras.

c) **Bases de datos de grafos:** Usadas para aquellos datos cuyas relaciones se pueden representar adecuadamente mediante un grafo. Los datos se almacenan en estructuras grafo con nodos (entidades), propiedades (información entre entidades) y líneas (conexiones entre las entidades).

d) **Base de datos Columnar (o Columna ancha):** En vez de tablas, en las bases de datos de columna tenemos familias de columnas que, son los contenedores de las filas. A diferencia de los RDBMS, no necesita conocer de antemano todas las columnas, cada fila no tiene por qué tener el mismo número de columnas. Este tipo de bases de datos se adecuan mejor a operaciones analíticas sobre grandes conjuntos de datos.

Pese a todas las opciones proporcionadas por el auge de las bases de datos NoSQL, esto no significa la desaparición de las bases de datos de RDBMS ya que son tecnologías complementarias. Estamos entrando en una era de persistencia políglota, una técnica que utiliza diferentes tecnologías de almacenamiento de datos para manejar las diversas necesidades de almacenamiento de datos. Hemos elegido la implementación de una base de datos NoSQL para nuestra aplicación ya que para nosotros el rendimiento es más importante que preservar el modelo relacional. Además de la posibilidad de escalar de forma sencilla, hemos tenido en cuenta otros criterios:

1. El volumen de lecturas y escrituras

2. Tolerancia por datos inconsistentes en réplicas
3. La naturaleza de las relaciones entre entidades y cómo eso afecta los patrones de consulta
4. Disponibilidad y requisitos de recuperación de desastres
5. La necesidad de flexibilidad en los modelos de datos
6. Requisitos de latencia

Bases de datos no relacionales NoSQL

Nos vamos a decantar por un modelo Almacenamiento Clave-Valor y vamos a almacenar una lista de estructuras JSON.
Este tipo de bases de datos Clave-Valor se utilizan en una amplia gama de aplicaciones, como las siguientes:

1. Almacenamiento en caché de datos de bases de datos relacionales para mejorar el rendimiento
2. Seguimiento de atributos transitorios en una aplicación web, como un carrito de compras
3. Almacenamiento de configuración e información de datos de usuario para aplicaciones móviles
4. Almacenar objetos grandes, como imágenes y archivos de audio

[3]

[3]Fuente: Oracle [15] y NoSQL for mere mortals [18]

Firebase Realtime Database

Firebase Realtime Database es una base de datos alojada en la nube que nos permitirá almacenar y sincronizar datos. Los datos se sincronizan con todos los clientes en tiempo real y se mantienen disponibles cuando la app no tiene conexión.

Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Cuando compilas apps multiplataforma con el SDK de iOS, Android y JavaScript, todos los clientes comparten una instancia de Realtime Database y reciben actualizaciones automáticamente con los datos más recientes.

Las funciones clave que nos proporciona son las siguientes:

1. **Tiempo real:** En lugar de solicitudes típicas de HTTP, Firebase Realtime Database usa la sincronización de datos (cada vez que cambian los datos, los dispositivos conectados reciben esa actualización en milisegundos). Proporciona experiencias colaborativas y envolventes sin pensar en el código de red.

2. **Sin conexión:** Las apps Firebase continúan respondiendo, incluso sin conexión, dado que el SDK de Firebase Realtime Database hace que tus datos persistan en el disco. Cuando se restablece la conexión, el dispositivo cliente recibe los cambios que faltaban y los sincroniza con el estado actual del servidor.

3. **Acceso desde dispositivos cliente:** Se puede acceder a Firebase Realtime Database directamente desde un dispositivo móvil o un navegador web; no se necesita un servidor de aplicaciones. La seguridad y la validación de datos están disponibles a través de las reglas de seguridad de Firebase Realtime Database: reglas basadas en expresiones que se ejecutan cuando se leen o se escriben datos.

4. **Escalamiento en varias bases de datos:** Con Firebase Realtime Database, puedes satisfacer las necesidades de datos de la app a gran escala: podrás dividir la información en diversas instancias de bases de datos dentro del mismo proyecto de Firebase. Usa Firebase Authentication para optimizar el proceso de autenticación en el proyecto. Podrás autenticar a usuarios en varias instancias de la base de datos. Controla el acceso a la información de cada base de datos. Para ello, usa las reglas personalizadas de Firebase Realtime Database en cada una de las instancias de la base de datos.

¿Cómo funciona? Firebase Realtime Database te permite compilar aplicaciones ricas y colaborativas, ya que permite el acceso seguro a la base de datos directamente desde el código del cliente. Los datos persisten de forma local. Además, incluso cuando no hay conexión, se siguen activando los eventos en tiempo real, lo que proporciona una experiencia adaptable al usuario final. Cuando el dispositivo vuelve a conectarse, Realtime Database sincroniza los cambios de los datos locales con las actualizaciones remotas que ocurrieron mientras el cliente estuvo sin conexión, lo que combina los conflictos de forma automática.

Realtime Database proporciona un lenguaje flexible de reglas basadas en expresiones, llamado reglas de seguridad de Firebase Realtime Database, para definir cómo se deberían estructurar los datos y en qué momento se pueden leer o escribir. Integrar Firebase Authentication permite que los programadores definan quién tiene acceso a qué datos y cómo acceden a ellos.

Realtime Database es una base de datos NoSQL y, como tal, tiene diferentes optimizaciones y funcionalidades en comparación con una base de datos relacional. La API de Realtime Database está diseñada para permitir solo operaciones que se puedan ejecutar rápidamente. Eso permite crear una excelente experiencia de tiempo real que puede servir a millones de usuarios sin afectar la capacidad de respuesta. Es importante pensar cómo deben acceder a los datos los usuarios y estructurarlos según corresponda.

4.2. APIs empleadas

Una API (del inglés: Application Programming Interface), es el conjunto de subrutinas, funciones y procedimientos que ofrece cierto ente (en este caso, AEMET) para ser utilizado por un software de un tercero para la obtención de datos, información, documentos, etc.
En esta sección listaremos las APIs que utilizaremos en el proyecto.

4.2.1. OpenWeather Weather API

OpenWeather nos ofrece una potente API con la que obtendremos el pronóstico meteorológico diario. Con una simple llamada como la siguiente:

- `http://api.openweathermap.org/data/2.5/weather?lat=36.7311957&lon=-2.854627&appid=d488064b89a68d822fd0f952df6b7c83`

Podremos obtener la información necesaria acerca de las condiciones meteorológicas. Los parámetros que deberemos rellenar son los siguientes:

1. **lat:** La latitud será la coordenada geográfica proporcionada por el sistema GPS de nuestro dispositivo móvil, nos servirá para identificar la posición. En el ejemplo es *36.7311957*.

2. **lon:** La longitud será la Coordenada geográfica proporcionada por el sistema GPS de nuestro dispositivo móvil, nos servirá para identificar la posición. En el ejemplo es *-2.854627*.

3. **appid:** Nuestra API Key, necesaria para autentificar la llamada, nos la proporcionan al crearnos una cuenta en OpenWeatherMap [19]. En el ejemplo es *d488064b89a68d822fd0f952df6b7c83*.

La respuesta que recibiremos al hacer la llamada del ejemplo, será un mensaje JSON, con la siguiente estructura:

```
{
"coord":
   {
      "lon":-2.85,
      "lat":36.73
   },
"weather":[
   {
      "id":800,
      "main":"Clear",
      "description":"clear sky",
      "icon":"01n"
   }
      ],
"base":"stations",
"main":
   {
```

```
      "temp":285.514,
      "pressure":1013.75,
      "humidity":100,
      "temp_min":285.514,
      "temp_max":285.514
   },
"wind":
   {
      "speed":5.52,
      "deg":311
   },
"clouds":
   {
      "all":0
   },
"dt":1533231000,
"sys":
   {
      "message":0.0025,
      "country":"ES",
      "sunrise":1533187107,
      "sunset":1533237375
   },
"id":2521382,
"name":"Balerma",
"cod":200
   }
}
```


A continuación vamos a describir cada uno de los atributos de la respuesta:

1. **coord:**
   a) lon: Longitud.
   b) lat: Latitud.
2. **weather:** Códigos de la información meteorológica.
   a) id: Identificador del clima.
   b) main: Grupo de condición meteorológica (Lluvia, Nieve, Despejado, etc.)
   c) description: Condición meteorológica dentro de su grupo

   d) icon. Icono del clima.

3. **base:** Párametro interno.
4. **main:**
   a) temp: Temperatura. Unidad por defecto: Kelvin.
   b) pressure: Presión atmosférica.
   c) humidity: Porcentaje de humedad.
   d) temp_min: Temperatura mínima en el momento. Unidad por defecto: Kelvin.
   e) temp_max: Temperatura máxima en el momento. Unidad por defecto: Kelvin.
5. **wind:**
   a) speed: Velocidad del viento. Unidad por defecto: m/s.
   b) deg: Dirección del viento, en grados.
6. **clouds:**
   a) all: Porcentaje de nubosidad.
7. **dt:**Tiempo empleado en calcular los datos, unix, UTC.
8. **sys:**
   a) type: Parámetro interno.
   b) id: Parámetro interno.
   c) message: Parámetro interno.
   d) country: Código del país (GB, JP etc.)
   e) sunrise: Hora del amanecer, unix, UTC.
   f) sunset: Hora del anochecer, unix, UTC.
9. **id:** Identificador de la población.
10. **name:** Nombre de la población.
11. **cod:** Parámetro interno.

Tendremos especialmente en cuenta los parámetros de la velocidad y la dirección del viento, ya que serán los que utilicemos para comprobar si el usuario de la aplicación está a la deriva y realizar la alerta de rescate.
[4]

[4]Fuente: Open Weather [19]

4.2.2. Google Play services location APIs

Utilizaremos las APIs integradas dentro de los Google Play services para obtener la información que necesitamos acerca de la ubicación y el reconocimiento de actividad de nuestro dispositivo. La información que pretendemos obtener es la siguiente:

1. Longitud.
2. Latitud.
3. Velocidad.
4. Dirección y sentido del movimiento.

4.2.3. WorkManager

WorkManager API

Para realizar las llamadas a nuestro backend utilizaremos la API WorkManager, que nos facilitará la programación de tareas asíncronas incluso si la aplicación sale o se reinicia el dispositivo.

Es mejor que utilizar un proceso en segundo plano, ya que mantener aplicaciones en segundo plano consume mucha batería y el problema se acentúa cuando todos esos servicios en segundo plano están escuchando transmisiones frecuentes, por ejemplo para saber si hay conexión a la red.

Las principales características ofrecidas por esta API son:

- Permite agregar restricciones de trabajo como disponibilidad de red.
- Permite programar tareas únicas o tareas asíncronas periódicas.
- Monitorizar y gestionar las tareas programadas.
- Podemos encadenar tareas distintas.
- Asegura la ejecución de tareas, incluso si la aplicación o el dispositivo se reinicia.
- Nos permite utilizar las funciones de ahorro de energía como el modo reposo.

Uso en nuestra aplicación

Encapsularemos nuestra función que comprueba si vamos a la deriva dentro de la clase OneTimeWorkRequest, que nos ejecutará la operación cada cinco minutos.

En el caso que la resolución de la función nos indique que estamos a la deriva, encadenaremos el objeto WorkContinuation para enlazar la función que muestra el aviso, durante cinco minutos de que vamos a llamar al 112.

Una vez terminado este tiempo, si no se ha detenido la ejecución mediante un botón que mostraremos en pantalla, encolaremos la última función, que será la encargada de llamar al 112.

[5]

[5]Fuente:Android WorkManager[10]

Firebase Auth

La mayoría de las apps necesitan identificar a los usuarios. Conocer la identidad de un usuario permite que una app guarde sus datos en la nube de forma segura y proporcione la misma experiencia personalizada en todos los dispositivos del usuario. Firebase Authentication proporciona servicios de backend, SDK fáciles de usar y bibliotecas de IU ya elaboradas para autenticar a los usuarios en tu aplicación. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federados populares, como Google, Facebook y Twitter, y mucho más.

En nuestra aplicación autenticaremos a los usuarios mediante la integración del proveedor de identidad federada de Google.

Firebase Authentication se integra estrechamente con otros servicios de Firebase y aprovecha los estándares de la industria como OAuth 2.0 y OpenID Connect, por lo que se puede integrar fácilmente con tu backend personalizado.

4.3. Herramientas de trabajo

En esta sección, listaremos una serie de herramientas que utilizaremos para desarrollar nuestra aplicación.

4.3.1. Android Studio

Android Studio es el entorno de desarrollo integrado oficial para el desarrollo de aplicaciones para android, esta basado en IntelliJ IDEA [11].

Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android, como las siguientes:

1. Un sistema de compilación basado en Gradle flexible
2. Un emulador rápido con varias funciones
3. Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android
4. Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK
5. Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código
6. Gran cantidad de herramientas y frameworks de prueba
7. Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
8. Compatibilidad con C++ y NDK
9. Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine

Interfaz de usuario

La ventana principal de Android Studio consta de varias áreas lógicas que se identifican en la siguiente figura.

1. La barra de herramientas te permite realizar una gran variedad de acciones, como la ejecución de tu app y el inicio de herramientas de Android.
2. La barra de navegación te ayuda a explorar tu proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana Project.

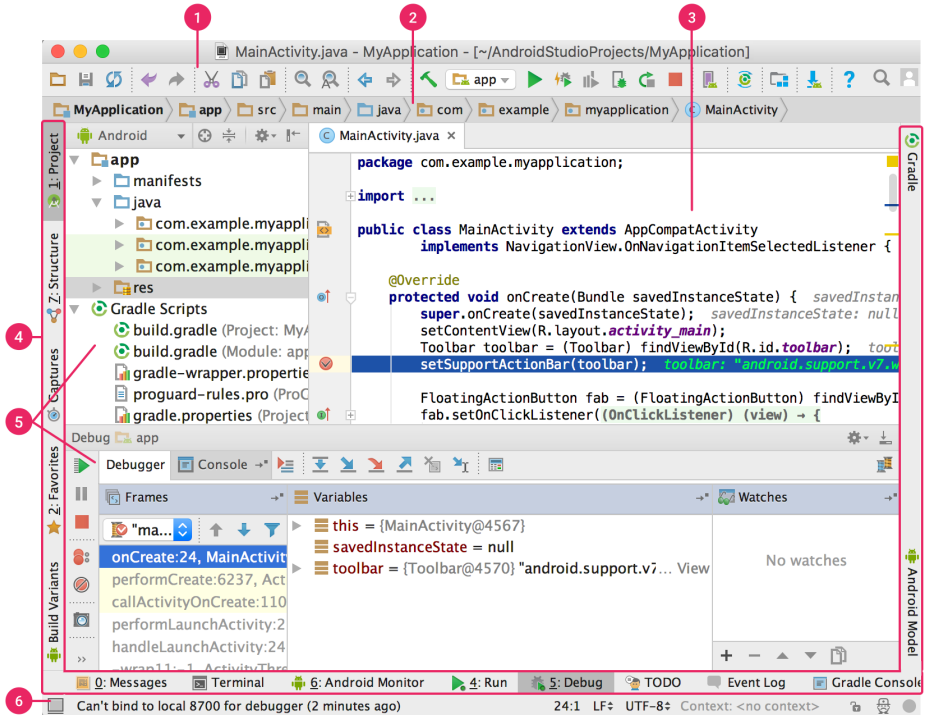

Figura 4. Ventana de Android Studio

3. La ventana del editor es el área donde puedes crear y modificar código. Según el tipo de archivo actual, el editor puede cambiar. Por ejemplo, cuando se visualiza un archivo de diseño, el editor muestra el editor de diseño.

4. La barra de la ventana de herramientas se extiende alrededor de la parte externa de la ventana del IDE y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.

5. Las ventanas de herramientas te permiten acceder a tareas específicas, como la administración de proyectos, las búsquedas, los controles de versión, etc. Puedes expandirlas y contraerlas.

6. En la barra de estado, se muestra el estado de tu proyecto y del IDE en sí, como también cualquier advertencia o mensaje.

6

[6]Fuente: Android Studio[6]

4.4. Métodos

En esta sección, vamos a listar una serie de métodos y buenas prácticas necesarias para este tipo de aplicación

4.4.1. Optimización de la batería del dispositivo

En la mayoría de los casos, optimizar el uso de la batería de por parte de la aplicación es un tema secundario, pero en nuestro caso y debido a la necesidad de mantener el máximo tiempo posible emitiendo al dispositivo. Una hora más de emisión podría llegar a ser vital en el peor de los casos, ayudando incluso a salvar una vida, por lo que debemos mejorar todo lo posible este aspecto. Mejorar el rendimiento de la batería es un tema prioritario.

Para ello vamos a tener en cuenta tres aspectos importantes:

- Hacer una aplicación *Lazy First*.
- Utilizar las distintas características de la plataforma relacionadas con el consumo de batería.
- Utilizar herramientas para encontrar la causa de la perdida de batería.

A continuación vamos a profundizar en cada uno de los aspectos anteriormente citados.

Lazy First

Hacer una aplicación Lazy First significa encontrar formas de reducir y optimizar operaciones que supongan una carga significativa para la carga de la batería. Las conceptos que deben aplicarse son los siguientes:

1. **Reducir:** Reducir el número de operaciones que se realizan, por ejemplo, utilizar una cache de datos descargados, en lugar de volver a descargarlos cada vez que los necesitemos.
2. **Aplazar:** Aplazar operaciones hasta que no supongan un impacto en la duración de la batería, por ejemplo, esperar a tener el cargador conectado para subir información al servidor.
3. **Agrupar:** Agrupar varias operaciones que puedan para que al realizarse de forma simultanea nos ahorre, por ejemplo, sacar el dispositivo de reposo varias veces.

Deberemos utilizar estos principios cada vez que queramos utilizar la CPU, la antena, la pantalla o el GPS del dispositivo.

Características de la plataforma

La plataforma de Android nos ofrece mecanismos para conservar la vida de la batería. De forma que nos ofrece diversas APIs para planificar los procesos, describimos algunas en la sección de APIs [?].

Dejando las APIs a un lado, caben destacar dos funciones de ahorro de energía que presenta Android, ya que estas prolongan la vida de la batería administrando la forma en que las apps se comportan cuando un dispositivo no está conectado a una fuente de energía. Descanso reduce el consumo de batería aplazando la actividad de CPU y de red de las apps en segundo plano cuando el dispositivo no se usa durante períodos de tiempo prolongados. App Standby aplaza la actividad de red en segundo plano de las apps con las cuales el usuario no haya interactuado recientemente.

A continuación vamos a profundizar en ambas funciones.

Modo Descanso: Si un usuario deja un dispositivo desconectado, quieto y con la pantalla apagada durante un período de tiempo determinado, este entra en el modo Descanso. En el modo Descanso, el sistema intenta conservar la carga de la batería restringiendo el acceso por parte de las apps a servicios de uso intenso de red y CPU. También evita que las apps accedan a la red y aplaza sus tareas, sincronizaciones y alarmas estándares.

De forma periódica, el sistema desactiva el modo Descanso durante un tiempo breve para permitir que las apps completen sus actividades aplazadas. Durante este período de mantenimiento, el sistema ejecuta todas las sincronizaciones, tareas y alarmas pendientes, y permite que las apps accedan a la red.

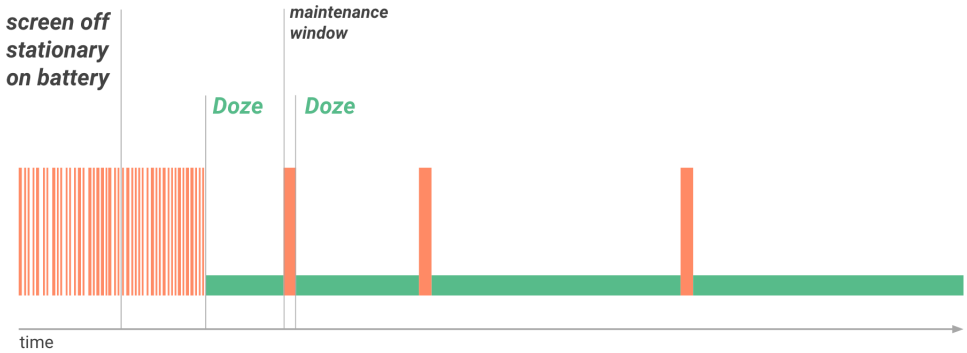

Figura 5: Descanso proporciona un período de mantenimiento recurrente para que las apps usen la red y controlen actividades pendientes.

Al finalizar cada período de mantenimiento, el sistema vuelve a activar el

modo Descanso, suspender el acceso a la red y aplazar tareas, sincronizaciones y alarmas. Con el paso del tiempo, el sistema programa los períodos de mantenimiento cada vez con menos frecuencia, lo cual permite reducir el consumo de batería en casos de inactividad durante más tiempo cuando el dispositivo no está conectado a un cargador.

Si bien el usuario activa el dispositivo moviéndolo, encendiendo la pantalla o conectándolo a un cargador, el sistema desactiva el modo Descanso y todas las apps vuelven a la actividad normal.

Restricciones del modo Descanso: Durante el modo Descanso, se aplican las siguientes restricciones a tus apps:

1. Se suspende el acceso a la red.
2. El sistema ignora los wake locks.
3. Las alarmas estándares AlarmManager (incluidas setExact() y setWindow()) se aplazan hasta el siguiente período de mantenimiento.
    a) Si necesitas programar alarmas que se activen en el modo Descanso, usa setAndAllowWhileIdle() o setExactAndAllowWhileIdle().
    b) Las alarmas programadas con setAlarmClock() se activan con normalidad; el sistema desactiva el modo Descanso antes de que esas alarmas se activen.
4. El sistema no realiza escaneos de Wi-Fi.
5. El sistema no permite que se ejecuten adaptadores de sincronización.
6. El sistema no permite que se ejecute JobScheduler.

Información sobre App Standby El modo App Standby permite que el sistema determine si una app se encuentra inactiva cuando el usuario no la usa activamente. El sistema hace esta determinación cuando el usuario no aplica toques en la app durante un período determinado y cuando no rige ninguna de las siguientes condiciones:

1. El usuario inicia explícitamente la app.
2. La app actualmente tiene un proceso en primer plano (ya sea como actividad o como servicio en primer plano, o en uso por parte de otra actividad u otro servicio en primer plano).

3. La app genera una notificación que los usuarios ven en la pantalla bloqueada o en la bandeja de notificaciones.

Cuando el usuario enchufa el dispositivo en una fuente de energía, el sistema libera las apps del estado de reposo, con lo cual les permite acceder libremente a la red y ejecutar cualquier tarea y sincronización pendientes. Si el dispositivo queda inactivo durante períodos prolongados, las apps inactivas pueden acceder a la red aproximadamente una vez al día.

Firebase Cloud Messaging Para poder utilizar nuestra aplicación en ambos modos de ahorro de batería utilizaremos la solución de mensajería Firebase Cloud Messaging [3], la cual nos permitirá enviar mensajes de forma segura al backend aún estando en ahorro de energía. Nos resultará sencilla su implementación al ya tener Firebase integrado en nuestra aplicación.

Herramientas para analizar el consumo de batería

Ahora vamos a listar un par de herramientas para Android que nos ayudarán a identificar las áreas en las que mejorar el consumo de batería. Utilizaremos estas herramientas para aplicar en estas áreas los principios de Lazy First.

1. **Profile GPU Rendering:** Esta herramienta nos mostrará, en forma de histograma de desplazamiento,una representación visual del tiempo necesario para representar los fotogramas de una ventana de IU en relación con un benchmark de 16 ms por fotograma.

   En las GPU menos potentes, la frecuencia de relleno disponible (la velocidad a la cual la GPU puede rellenar el búfer de fotogramas) puede ser bastante baja. A medida que la aumenta la cantidad de píxeles requeridos para dibujar un fotograma, la GPU puede tardar más en procesar comandos nuevos y solicitar tiempo de espera al resto del sistema hasta ordenarse. La herramienta de generación de perfiles te ayuda a detectar los casos en que el rendimiento de la GPU se ve afectado cuando esta intenta dibujar píxeles o sobrecargado debido a una superposición voluminosa.

2. **Battery Historian:**Una útil herramienta para inspeccionar la relación entre la batería y los eventos de un dispositivo android, estando este apagado. Nos permitirá visualizar eventos de nivel de sistema y aplicación en una línea de tiempo para ver fácilmente varias estadísticas desde que se cargó el dispositivo por completo por última vez, es decir,

podremos seleccionar nuestra aplicación y ver la evolución de la carga de la batería respecto a esta.[8]

[7]

[7]Fuente: android developers [9]

Capítulo 5

Resultados y discusión

El principal objetivo de este proyecto una aplicación móvil radio faro que funcione como cliente, emitiendo la posición GPS del dispositivo en determinadas condiciones, y como monitor para que pueda ser localizada la señal del dispositivo que solicita ayuda.

El resultado de este proyecto es una aplicación que intenta cumplir con los requisitos planteados y creo que lo hemos conseguido, aunque la localización no sea todo lo fiable que nos gustaría hemos conseguido un producto que puede ser de utilidad para ayudar a las personas en apuros y que puede servir en un futuro como base para implementar la arquitectura PEMEA [2] y aplicar la misma a un entorno marítimo, incluso usar el backend de la aplicación para realizar estudios sobre los incidentes reportados para sacar estadísticas y determinar puntos negros de nuestro litoral.

A continuación vamos a comentar en que consiste la arquitectura PEMEA y a explicar el principal punto de discusión del proyecto, que es el cálculo de la velocidad a la que consideramos que un cuerpo se encuentra a la deriva en el mar.

5.1. Arquitectura PEMEA

El proyecto PEMEA (Pan-European Mobile Emergency Apps) es una inciativa de la unión europea para unificar en una aplicación, la conexion con todos los centros de emergencia 112.

El objetivo de este proyecto es proporcionar una arquitectura que permita acceder a los servicios de emergencia en cualquier lugar de Europa y un endpoing para poder dar acceso a información ampliada del usuario, como por ejemplo, a su geolocalización.

El proyecto aún no esta terminado, actualmente esta en fase de pruebas. El inicio de la fase 2 esta planificado a principios de Septiembre 2019.

En la siguiente imagen vemos como funciona esta arquitectura.

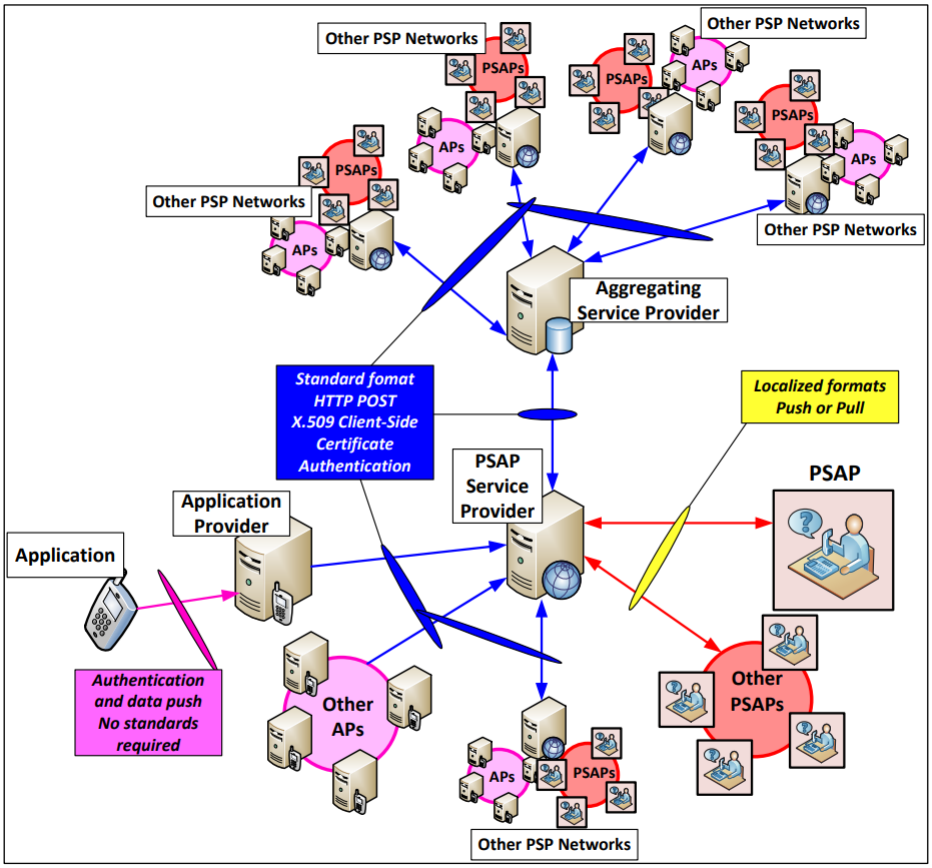

Figura 10. Arquitectura PEMEA completa.

1

[1]Fuente: PEMEA [2]

5.2. Pronosticar la deriva de objetos en el océano

El movimiento de un objeto a la deriva en la superficie del mar es uno de los temas de discusión fundamentales de este proyecto. Este movimiento es el resultado de varias fuerzas que actúan sobre la superficie:

1. Corrientes marinas.
2. Viento.
3. Oleaje.

También influye el centro de gravedad del objeto y su forma.

Tenemos dos cuestiones importantes que definir, la dirección y la velocidad del objeto a la deriva.

5.2.1. Velocidad del objeto

Después de leer varios estudios y artículos, he llegado a la conclusión, que la fórmula para calcular la velocidad de un objeto flotando a la deriva, es directamente proporcional a la velocidad del viento y condicionado por un factor que se calcula empíricamente o con modelos muy complejos de mecánica de fluidos que no tenemos a nuestra disposición, así que vamos a simplificar.

Utilizaremos la siguiente fórmula.

$$x \times \alpha + A$$

Donde:

1. x es la velocidad del viento, en m/s.
2. *alpha* es el coeficiente con valor 0.70.
3. A es la constante de error, que calculamos como un 20 % del valor de x.

[2]

[2]Fuente: Formulation of Leeway-Drift Velocities for Sea-Surface Drifting-Objects Based on a Wind-Wave Flume Experiment [13]

5.2.2. Dirección del objeto

Otro de los puntos más importantes es la dirección de movimiento del objeto a la deriva, comúnmente se piensa que es la misma que la del viento, pero hay que tener en cuenta la componente cruzada del viento principal, a la hora de calcular esta dirección, podemos ver en la siguiente imagen como se compone el vector de movimiento del objeto a la deriva.

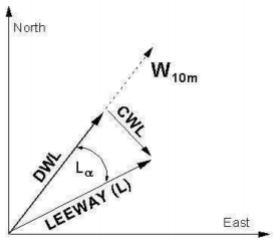

Figura 9. Relacion entre el vector del objeto a la deriva (L), el vector del viento (DWL) y el vector del viento cruzado (CWL). L indica el ángulo de deriva.

Para solventar en nuestro algoritmo el problema del viento cruzado, vamos a añadir un margen de error del treinta por ciento a la hora de calcular la dirección de deriva.
[3]

[3]Fuente: Ocean weather forecasting: An integrated view of oceanography [25]

Capítulo 6

Conclusiones

Como conclusión final, me gustaría añadir que gracias a este proyecto he aprendido mucho sobre las tecnologías implicadas en su desarrollo así como sobre el estado del arte del mismo aprendiendo conocimientos sobre los peligros del día a día en nuestras costas.

Personalmente, es un hito importante para mí ya que supone la finalización de mis estudios en Ingeniería Informática a los que he dedicado mucho tiempo y sacrificio. También he aprendido bastante sobre desarrollo Android y Firebase que no había utilizado antes de empezar con este proyecto y cuyo aprendizaje me ha sido menos costoso gracias a los conocimientos adquiridos durante la carrera.

6.1. Mejoras

Hemos realizado un listado con mejoras que se pueden aplicar, la mayoría relacionadas con el proyecto PEMEA, que se podrían implementar cuando la arquitectura este completamente disponible.

1. Añadir un panel para poder monitorizar dispositivos
2. Añadir avisos maisvos para notificar a quien tenga instalada la aplicación que hay una situación de peligro
3. Geolocalización en las llamadas al 112
4. Añadir alertas meteorológicas para avisar de condiciones peligrosas para la práctica de actividades marítimas.

Bibliografía

[1] Federación Balear de vela. Ciencias aplicadas al kiteboarding. http://www.federacionbalearvela.org.

[2] EENA. Pan-european mobile emergency apps. https://eena.org/apps/.

[3] Firebase. Firebase cloud messaging. https://firebase.google.com/docs/cloud-messaging/.

[4] Google. Android developer. https://developer.android.com/index.html.

[5] Google. Android studio. https://es.wikipedia.org/wiki/Android_Studio.

[6] Google. Android studio. https://developer.android.com/studio/intro/.

[7] Google. Art. https://source.android.com/devices/tech/dalvik/.

[8] Google. Battery historian. https://github.com/google/battery-historian.

[9] Google. Optimización batería. https://developer.android.com/topic/performance/power/.

[10] Google. Schedule tasks with workmanager. https://developer.android.com/topic/libraries/architecture/workmanager.

[11] Jetbrains. Intellij idea. https://www.jetbrains.com/idea/.

[12] JSON. Introducción a json. https://www.json.org/json-es.html.

[13] Atsuhiko ISOBE1 Hirofumi HINATA2 Shinichiro KAKO1 and Shun YOSHIOKA. Formulation of leeway-drift velocities for sea-surface

drifting-objects based on a wind-wave flume experiment. https://www.terrapub.co.jp/onlineproceedings/ec/05/pdf/PR_05239.pdf.

[14] Salvamento Marítimo. Salvamento marítimo. http://www.salvamentomaritimo.es/sm/safeTRX/.

[15] Oracle. ¿que es una base de datos nosql? https://blogs.oracle.com/spain/qu-es-una-base-de-datos-nosql.

[16] Vicente González Ruiz. Desarrollo de un sistema de localización marítima usando radio faros. https://w3.ual.es/~vruiz/Docencia/Proyectos/Ofertados/radio-faro/index.html.

[17] Sport. Incremento deportes acuáticos. http://www.sport.es/es/noticias/deporte-extremo/practica-deportes-aventura-incrementa-3672031.

[18] Dan Sullivan. Nosql for mere mortals. http://almirez.ual.es/record=b1556758~S7*spi.

[19] Open Weather. Open weather map. https://openweathermap.org/.

[20] Wikipedia. Corriente de resaca. https://es.wikipedia.org/wiki/Corriente_de_resaca.

[21] Wikipedia. Escala de beaufort. https://es.wikipedia.org/wiki/Escala_de_Beaufort.

[22] Wikipedia. Escala douglas. https://es.wikipedia.org/wiki/Escala_Douglas.

[23] Wikipedia. Gps. https://es.wikipedia.org/wiki/GPS.

[24] Wikipedia. Java. https://en.wikipedia.org/wiki/Java_(programming_language).

[25] Bruce Hackett Øyvind Breivik and Cecilie Wettre. Ocean weather forecasting: An integrated view of oceanography. https://www.academia.edu/20257040/Forecasting_the_drift_of_objects_and_substances_in_the_ocean.