

# WeNave

Escola del Treball



Victor de la Rocha  
Yeray Pérez  
Luis Parra

---

# Índice de contenidos

Presentación del proyecto	3
Estructura y organización del grupo	4
Gestión de tiempo y tareas	4
Control de Versiones	6
Software de soporte para la realización del proyecto	7
Funcionamiento general de la aplicación	9
Características de la aplicación	10
Aspecto gráfico de la aplicación.	11
Presentación de colores	12
Presentación de iconos	13
Presentación de fondos de tarjeta	14
Presentación de vistas	16
Presentación de menú	21
Servicios externos	22
Interacción de la aplicación con Mapbox SDK	23
Descarga de datos desde Dark Sky	24
Enlace de los datos con la interfaz gráfica	25
Guardar en BD con Realm	26
Actualizar o eliminar datos con Realm	27
Análisis de código	28
UML	30
Casos de Uso	33
Anexo	36

---

## Presentación del proyecto

WeNave ofrece una solución a todas aquellas personas que tienen que tomar las carreteras en cualquier parte del mundo.

Con WeNave podremos establecer rutas, definiendo un máximo de 25 puntos, obteniendo de esta manera, el clima exacto que hará en el momento de pasar por el punto.

El usuario obtendrá la siguiente información:

- Temperatura
- Humedad relativa
- Lluvia
- Un ícono que indicará el tipo de climatología (Sol, Nubes, etc)
- Un breve resumen de la situación climatológica actual.

Una vez obtenida la ruta, el mismo usuario podrá o bien, tomar la carretera con toda la información que ha obtenido, o bien, guardarla para futuras ocasiones si esa ruta la suele hacer varias veces, ya que la propia aplicación nos dará la opción de guardar la información de la ruta.

WeNave ofrece una manera segura de salir a la carretera teniendo todos los factores climatológicos a favor del usuario.

En las siguientes páginas, se ofrece al detalle el funcionamiento de la aplicación, de manera tanto interna como externa.

## Estructura y organización del grupo

El grupo está formado por tres estudiantes, el número máximo de integrantes que puede tener un grupo. Estos tres integrantes son:

- Victor de la Rocha.
- Yeray Pérez.
- Luis Parra.

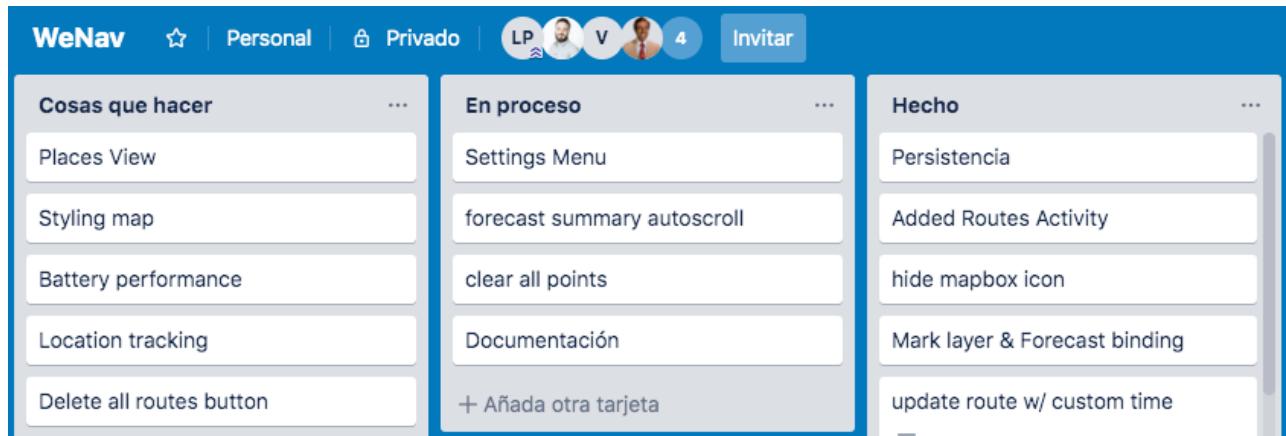
Podemos dividir el proyecto en dos familias, una parte de lógica y una parte de diseño. Teniendo esto claro, Victor desarrolló la mayoría de la parte lógica del programa, Luis se centró en la parte gráfica y Yeray servía como punto de apoyo para los momentos en los que una de las dos partes flaqueaba, aligerando, de esta manera, las tareas de ambas familias.

## Gestión de tiempo y tareas

Para hacer una gestión de tareas hemos utilizado “Trello”. Es una potente herramienta web gratuita donde puedes crear proyectos y para cada proyecto un numero ilimitado de tableros, donde asignar tareas.

Si bien nosotros no hemos distribuido las tareas por “departamento”, si que hemos establecido una serie de tareas al empezar el proyecto.

Una vez establecidos los puntos a implementar en la app, cada uno elegía una actividad y la movía al tablón “En proceso” hasta que la acababa y notificaba al resto del grupo que ya había acabado esta tarea.

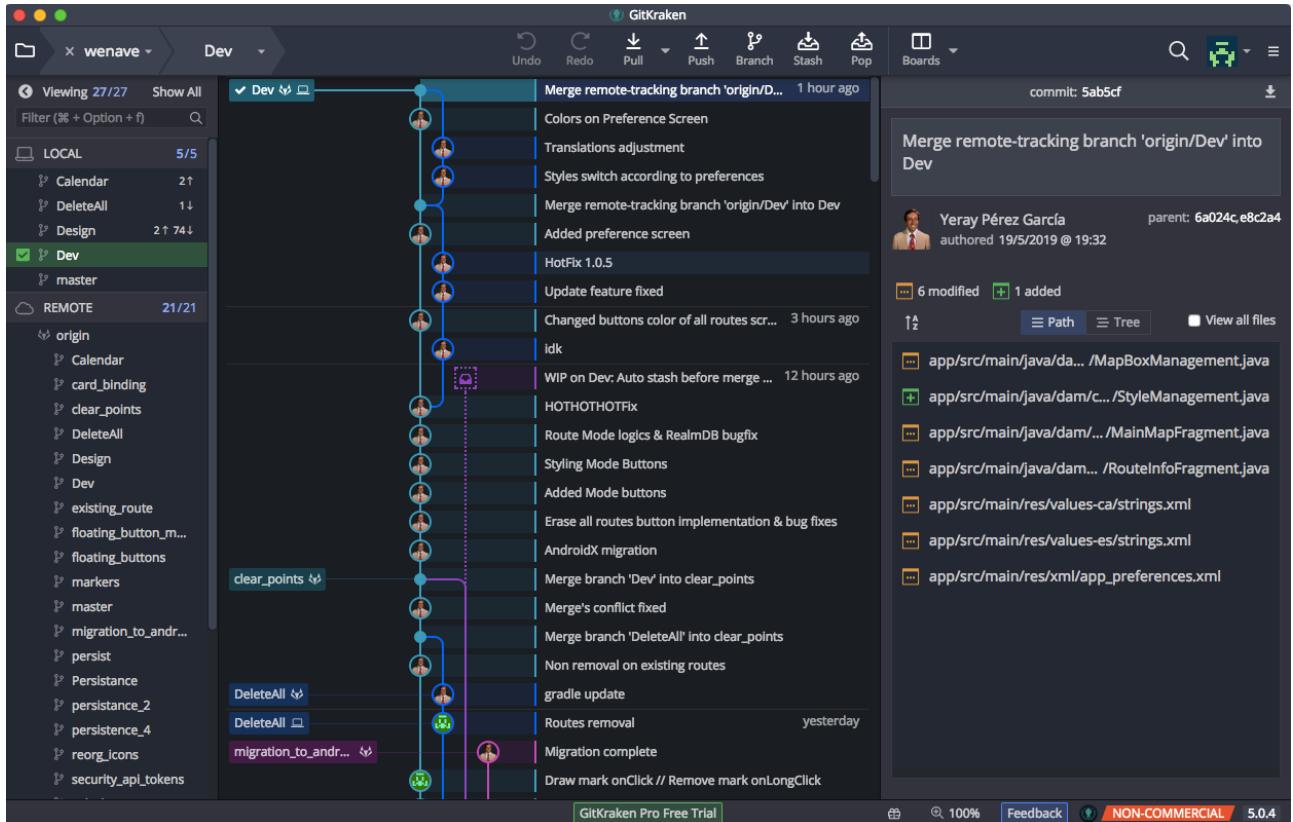


---

Una vez una tarea estaba finalizada, se movía al tablero de “Hecho”.

# Control de Versiones

Como herramienta de control de versiones hemos utilizado GitLab combinado con GitKraken.



Se puede consultar el código fuente en el siguiente repositorio:

<https://gitlab.com/VRoxa/wenave.git>

---

## Software de soporte para la realización del proyecto

Para desarrollar la aplicación WeNave, nos hemos apoyado en varias herramientas.

A continuación se detalla una lista con todo el software usado:

- GitLab
- GitKraken
- PhotoShoP
- Adobe InDesign
- Android Studio

### GitLab

Es el servicio web que hemos usado para tener nuestro proyecto siempre en la última versión.

Este mismo solo lo hemos usado la primera vez que hemos creado el proyecto, únicamente para tener un repositorio en la nube.



### GitKraken

Es una aplicación de escritorio que funciona como sistema de control de versiones.

Enlazamos nuestra cuenta de GitLab desde el primer momento, antes de empezar siquiera a escribir la primera línea de código.



Es una herramienta muy fácil de utilizar si conoces la “metodología Git”, aparte, la interfaz es user-friendly, así que no supone ningún esfuerzo hacer servir esta herramienta.

### PhotoShop

Por motivos obvios, PhotoShop siempre ha sido una herramienta de apoyo muy grande en el mundo de la edición de imágenes, vectores o cualquier tipo de archivo gráfico.



Gracias a su versatilidad y su interfaz intuitiva, hemos podido editar y modificar imágenes que se presentan tanto en la aplicación como en esta documentación.

### InDesign

Simplemente para la elaboración de este manual.



---

## Android Studio

El programa base del proyecto WeNave, el IDE desarrollado por jetBrains y hecho únicamente para el desarrollo de aplicaciones Android, ya sea con lenguaje JAVA como con Kotlin.

Pese a ser un programa “pesado” de funcionar, con los ordenadores proporcionados en clase, no hemos tenido ningún tipo de problema.



A continuación una lista de requisitos mínimos para poder trabajar con Android Studio:

- Escritorio GNOME o KDE sobre ubuntu 14.04
- Sistema de 64-bit capaz de correr aplicaciones de 32-bit.
- Librería GNU C (glibc) 2.19 o superior.
- Mínimo, 3 GB RAM, 8 GB RAM recomendados, + 1 GB para el emulador.
- Mínimo 2 GB de espacio disponible en el disco, 4 GB recomendados (500 MB para el IDE + 1,5 GB para el SDK de Android y la imagen del emulador).

Si bien a la hora de trabajar desde los ordenadores personales, hemos tenido algún problema de rendimiento, no ha sido culpa de Android Studio, si no por el emulador que activa el propio programa.

Este problema lo hemos solucionado usando nuestros propios terminales con Android 28.

---

## Funcionamiento general de la aplicación

El funcionamiento de la aplicación lo podemos dividir en varios apartados.

En un primer apartado podemos comentar el menú en el que nos moveremos, que será un *Navigation Drawer* con el que nos podremos mover por todas las funcionalidades disponibles de la aplicación, como la creación de una ruta, la consulta de las rutas almacenadas o ir a los ajustes de la aplicación.

Por un lado, tenemos la **creación de una ruta**, en el que podremos seleccionar o deseleccionar los puntos por los que queremos que pase nuestra ruta, y estos puntos serán en los que más tarde podremos consultar la información meteorológica. En esta misma pantalla tendremos varios botones, uno para borrar todos los puntos que hayamos seleccionado en el mapa, otro para seleccionar la fecha en la que queremos empezar la ruta, otro con en el que podremos escribir lugares concretos para añadir a nuestros puntos y finalmente otro botón para que la aplicación calcule la ruta y nos muestre el clima de nuestros puntos.

Una vez que le damos al botón de calcular la ruta, nos iremos a otra pantalla en la que se nos mostrará el resumen de la ruta en un mini mapa y justo debajo de este mini mapa una lista de los puntos de la ruta con su clima. Estos puntos pueden ser seleccionados y automáticamente en el mapa habrá un movimiento de cámara que saltará al punto que hemos seleccionado de la lista.

En esta pantalla en la que podremos ver el resumen de nuestra ruta, tendremos un botón con el que le podremos poner un nombre a nuestra ruta y guardarla por si la queremos consultar más tarde. Esta ruta se guardará en una BD local.

Por otra parte, mediante su selección en el *Navigation Drawer*, podemos entrar en la pantalla dónde podemos ver todas las rutas que hemos guardado. Clicando sobre cualquiera de ellas iremos a la pantalla de información de la ruta, la cual será muy similar a la pantalla de resumen de la ruta cuando la creamos, sólo que no podemos poner nuevos puntos en nuestra ruta. Adicionalmente, podemos refrescar la información de la ruta con el tiempo actual.

Por último, tenemos un menú ajustes dónde podremos cambiar el estilo de los mapas de la aplicación y habilitar o deshabilitar la actualización automática de nuestras rutas con la meteorología.

---

## Características de la aplicación

A continuación, enumeramos las características principales de nuestra aplicación:

- Se pueden seleccionar los puntos donde se quiere que pase la ruta, nuestras rutas NO van de un sitio a otro, puede pasar por hasta 25 puntos diferentes.
- Interacción *user-friendly* con el mapa para las diversas opciones con los puntos de la ruta.
- Diseño y estilo unificado en toda la aplicación.
- Datos meteorológicos exactos del punto seleccionado con la API de Dark Sky.
- Utilización de *Recycler Views* para la implementación de las listas que usamos en la aplicación para un mejor rendimiento.
- Utilización de *AsyncTasks* para mejorar el rendimiento multihilo y no dejar colgada la UI de la aplicación mientras se realizan algunas tareas.
- *Tracking* del dispositivo en el mapa para poder crear una ruta mejor.
- Mapas de Mapbox SDK.
- Guardado, eliminado y actualización de rutas en BD.
- Cambio de estilo en los mapas.
- En creación de ruta, podemos elegir una fecha con hora para indicar la fecha a la que queremos empezar la ruta.
- Plugin de *Mapbox* de buscador de sitios para añadir a nuestra ruta.
- Tres modos para crear una ruta, modo normal, modo circuito y modo inicio en ubicación de la persona.
- Ofuscación de las API Keys mediante el NDK que nos brinda Android. Mediante este sistema nuestras API Keys se escriben en C/C++ y se compilan. Por esta razón es más seguro, ya que las librerías de NDK no se pueden descompilar fácilmente , y aunque se hiciese, se tendría que revisar con un editor hexadecimal, y por la naturaleza de las API Key (La sintaxis) es casi imposible de verla con uno de estos editores.

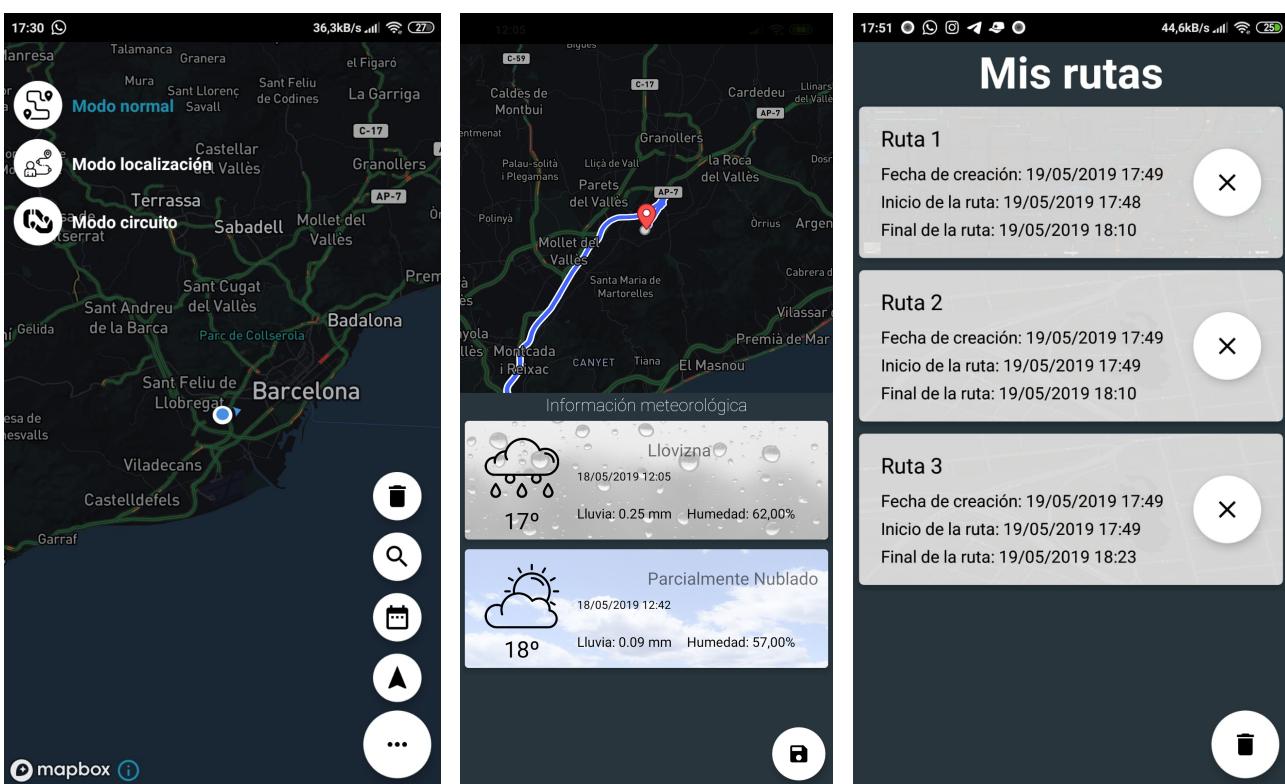
## Aspecto gráfico de la aplicación.

Hemos elegido tres colores para WeNave.

- Blanco
- Negro
- Azul

Definimos así un tono oscuro tanto en el mapa como en las diferentes pantallas que obtiene el usuario, contrastando la información que recibe con unas letras negras sobre un fondo blanco.

Se ve más claro en las siguientes imágenes:



En estas dos capturas de pantalla observamos los colores que hemos usado, un tono azul oscuro para darle profundidad a nuestras pantallas, cartas con un fondo blanco y un texto negro sólido, con la única intención de ofrecer al usuario una consonancia entre colores.

---

## Presentación de colores

En la siguiente página se presentan los colores usados en la app, tres colores en consonancia para no molestar a la vista del usuario cuando utilice WeNave.

Blanco:

Código:  
#ffffff

Muestra:



Negro:

Código:  
#000000

Muestra:



Azul:

Código:  
#FF28353D

Muestra:

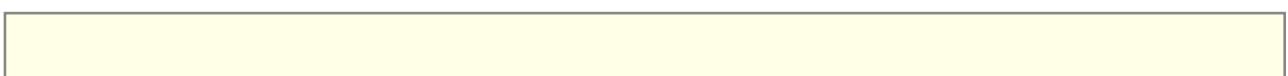


Por otro lado, WeNave usa un cuarto color, para indicar la CardView seleccionada en la pantalla de información de rutas.

Blanco:

Código:  
#FFFFFFE0

Muestra:



## Presentación de iconos

En la siguiente página se presentan los iconos usados en WeNave, divididos en estéticos y funcionales.

Estéticos:

Son los iconos que son utilizados en las CardViews de WeNave.

Su propósito es informar al usuario de una manera gráfica y rápida del tiempo que hará cuando pase por determinados puntos.

A continuación una tabla con cada uno de los iconos y el nombre que toman dentro de WeNave.



Clear\_Day

Aparece cuando el tiempo que hace es despejado durante el día.

Clear\_Night

Aparece cuando el tiempo que hace es despejado de noche

Cloudy

Aparece cuando tenemos nubes en el cielo, independientemente de si es de día o de noche.

Fog

Aparece cuando tenemos una niebla densa en el punto de la ruta. Independiente de día o de noche.

Partly\_Cloudy\_Day

Aparece cuando tenemos el día con sol pero con nubes no abundantes.

Partly\_Cloudy\_Night

Aparece cuando tenemos la noche con nubes poco abundantes.

Rain

Aparece cuando en el punto de la ruta hay precipitaciones.

Sleet

Aparece cuando en el punto de la ruta hay precipitaciones en forma de agua-nieve.

Snow

Aparece cuando en el punto de la ruta hay precipitaciones en forma de nieve.

Wind

Aparece cuando en el punto de la ruta sopla un viento extremadamente fuerte.

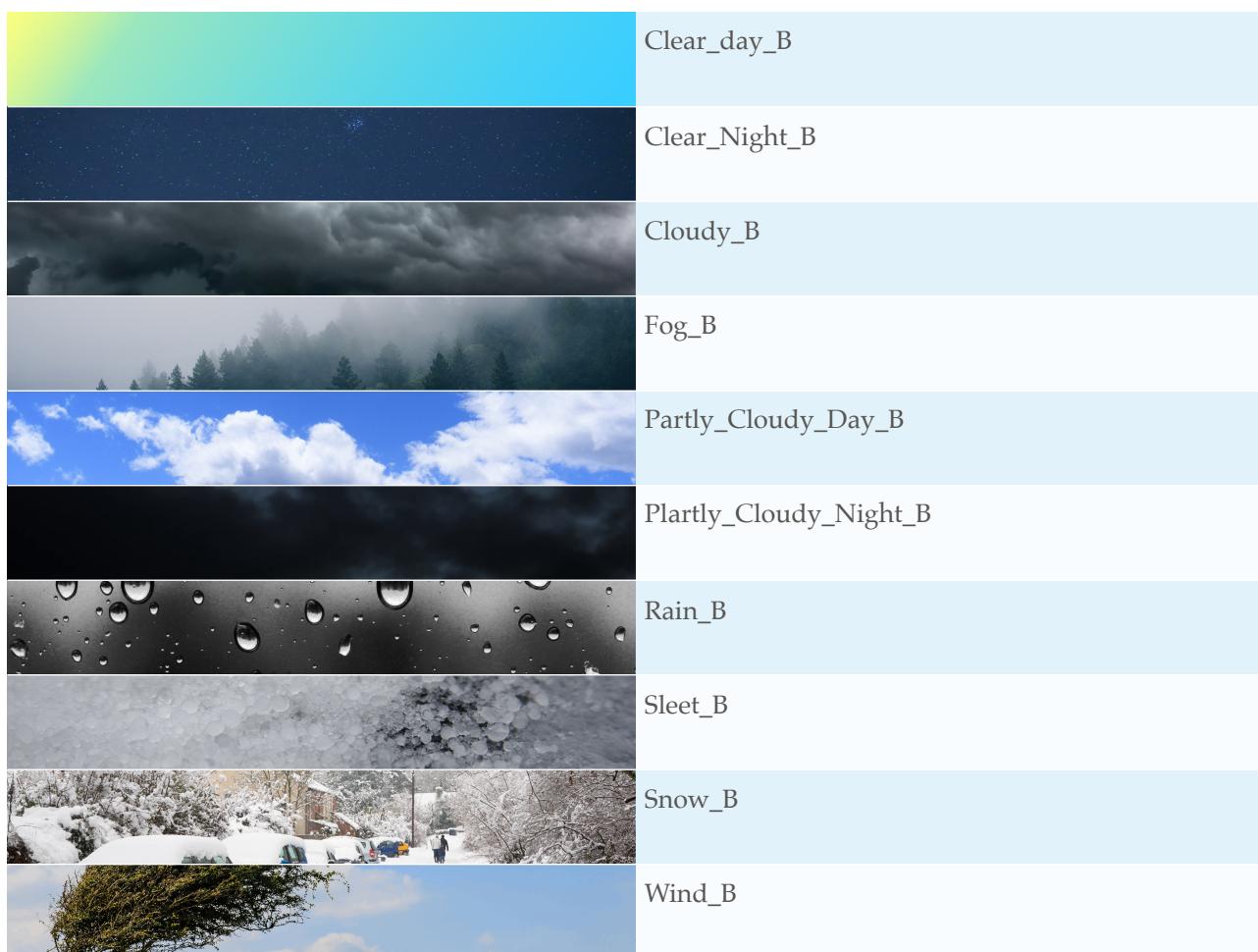
## Presentación de fondos de tarjeta

Cuando especificamos los puntos de ruta obtenemos una serie de CardViews donde WeNave nos muestra toda la información explicada antes. Aún y así, hemos querido hacer una vista un poco mas llena de las tarjetas, no solo texto.

En este apartado veremos los tipos de fondos que hemos usado para las CardViews, tanto para el tiempo como para nuestras rutas guardadas.

Como hemos visto en el apartado anterior, tenemos unos nombres para los iconos. En este apartado se produce una cosa similar, los fondos tienen el mismo nombre pero con un “\_b” al final del nombre. De esta manera cuando especificamos el icono a la CardView, añadimos también este fondo, aplicando un Alpha de 0.2 puntos para facilitar la visión del usuario a la letra que tiene por encima de la imagen.

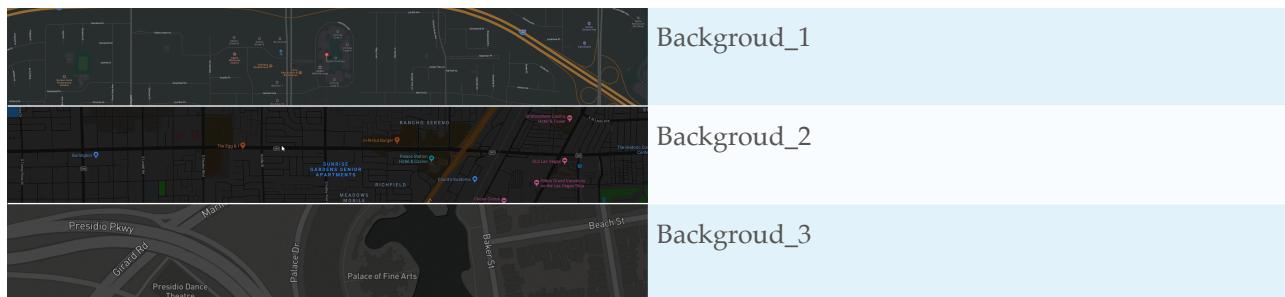
Todo y tener un Alpha especificado, en las imágenes siguientes este Alpha no esta aplicado para que se vea con mas claridad.



---

Por otro lado tenemos los fondos que usa WeNave en las CardViews de las rutas guardadas.

Son un conjunto de 3 imágenes que se establecen automáticamente como fondo de tarjeta de manera aleatoria. Aquí también se aplica un Alpha de 0.2 puntos por el mismo motivo, facilitar al usuario una correcta lectura.



Estas imágenes se asignan de manera aleatoria a cada una de las rutas guardadas.

---

## Presentación de vistas

En WeNave encontramos 4 pantallas funcionales:

- La primera, la actividad principal que se muestra nada mas abrir la aplicación.
- Una segunda, que obtendremos cuando creemos la ruta, que nos mostrará la climatología de cada uno de los puntos establecidos.
- Una tercera actividad donde encontraremos todas nuestras rutas guardadas.
- La última, la pantalla de ajustes, donde podremos decidir el aspecto que tendrá nuestro mapa.

## Actividad Principal, Mapa

Cuando abrimos WeNave por primera vez, la pantalla que nos encontramos es un mapa, con un menú desplegable en forma de botón en la esquina inferior derecha y un seleccionable en la esquina superior izquierda.

La ubicación que obtendremos de primeras, cambiará en función de donde nos encontremos, ya que WeNave determina la ubicación del usuario a tiempo real.

En la siguiente captura se muestra la actividad principal, con el menú inferior desplegado.

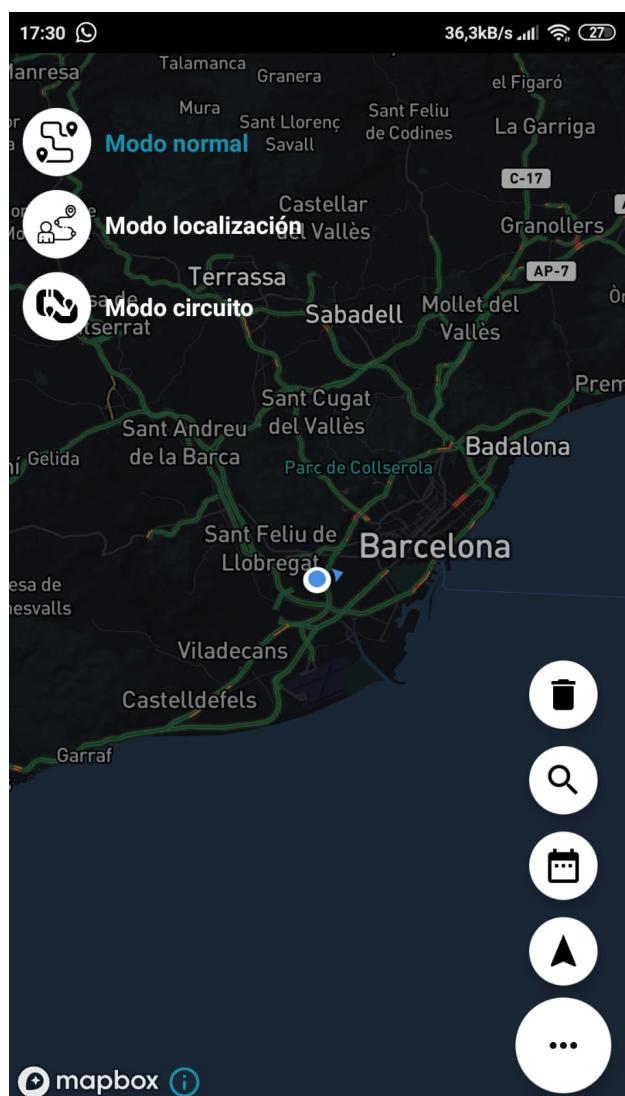
Como se puede observar, obtendremos un mapa con la ubicación y dos menús, uno en la parte inferior izquierda y otro en la esquina superior derecha.

Dentro del menú inferior tenemos 4 opciones (orden de arriba a abajo)

- Papelera: Elimina todos los puntos de la ruta que hemos especificado. Dejando el mapa en una tabula rasa para empezar a planear la ruta otra vez desde el punto0.
- Lupa: Nos permite mover el mapa a la ubicación que nosotros determinemos.
- Calendario: Genera un calendario para elegir el día y la hora exacta en la que queremos salir a hacer esa ruta.
- Navegación: Este botón genera la ruta una vez establecidos, mínimo, dos puntos.

Por otro lado, en el menú superior, tenemos 3 seleccionables:

- Normal: Con este botón seleccionado, la ruta se hará desde el punto A hasta el punto final que hayamos determinado.
- Localización: La ruta empezará desde el punto en el que nos encontremos y el siguiente que tomará será el primer punto que nosotros hayamos establecido.
- Circuito: Con esta opción nuestra ruta empezará y acabará en el primer punto que establecemos.



## Actividad Información de la ruta.

A la siguiente pantalla podemos acceder de dos maneras distintas:

- Especificando una ruta en la pantalla principal.
- Recuperando una ruta que tenemos guardada como usuario.

La actividad que obtendremos de cualquiera de las dos maneras será la misma, lo único que varía es la información que mostramos en ella.

WeNave nos muestra dos elementos principales en pantalla:

En la parte superior tenemos el mapa, con la ruta hecha, mostrando la ruta mas corta desde el punto A al punto B.

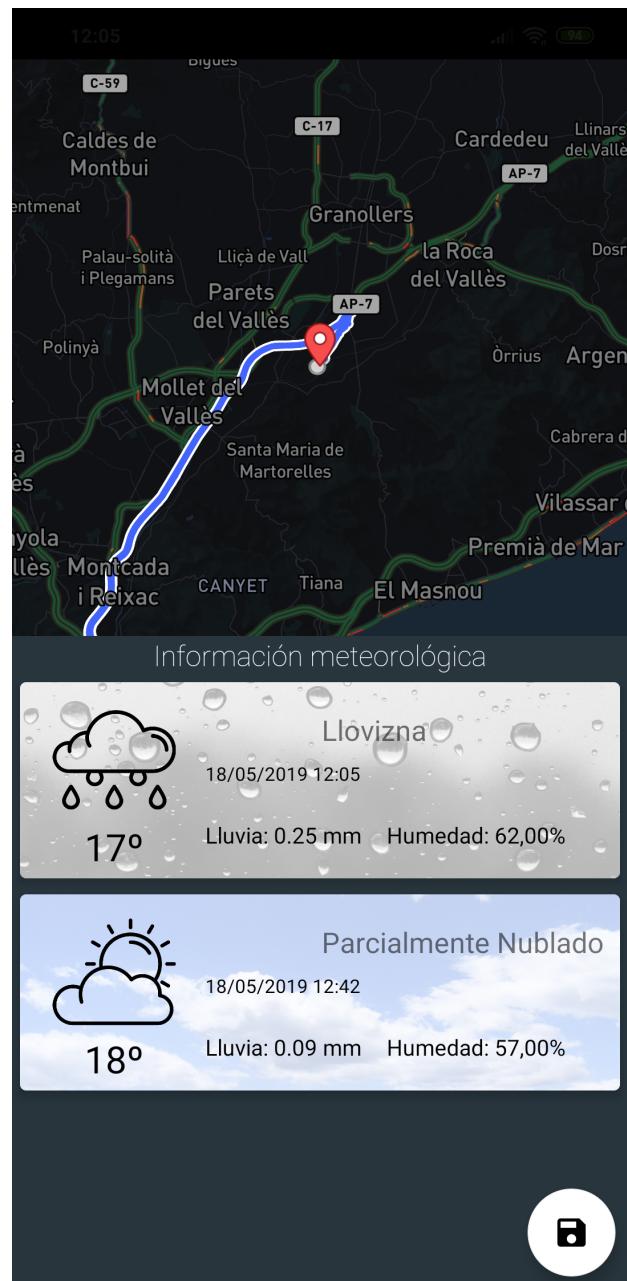
En la parte inferior tenemos la información de cada uno de los puntos.

Se presentan tantas CardViews como puntos tengamos en nuestro mapa, mostrando de cada uno de ellos los siguientes parámetros:

- Breve resumen de la situación meteorológica.
- Hora de pasada por el punto.
- Lluvia.
- Humedad relativa.
- Temperatura.
- Icono del tiempo.

Si hacemos un toque con el dedo en cada uno de los CardViews, este se iluminará y el mapa se moverá situando el centro del centro de la pantalla del mapa en el punto que hemos tocado.

En esta parte inferior podemos observar un icono de Disquete. Lo usaremos cada vez que queramos guardar una ruta en el caso de que la queramos consultar mas tarde o en unos días.



## Actividad Rutas Guardadas

Una vez hemos especificado una ruta, hemos visto los puntos por donde pasamos y recibida la información podemos decidir que hacer con la ruta que hemos hecho.

Volvemos a tener dos opciones:

- Eliminar la ruta, para esto solo tendremos que cerrar la aplicación y no guardaremos ningún tipo de información.
- Por otro lado, si esa ruta la hacemos mas de una vez al día o es nuestra ruta habitual para salir un domingo con el coche o con la moto, podemos elegir guardarla.

Como se muestra en la pantalla anterior, una vez decidimos la ruta, podemos guardarla, y acceder a ella mas tarde.

En la siguiente pantalla observamos el resultado de esta actividad:

Obtendremos todas las rutas guardadas, con información sobre cada una de ellas:

- El nombre que le hemos puesto a la ruta.
- La fecha de creación de la ruta.
- La hora de inicio de la ruta.
- La hora aproximada de fin de la ruta.

También encontramos un botón en forma de X en la parte derecha de cada CardView.

Con este botón podremos borrar nuestras rutas que ya no queramos hacer, así conseguimos tener siempre únicamente las rutas que queremos.

En la esquina inferior derecha, encontramos un botón en forma de papelera, esto nos permite eliminar todas las rutas guardadas.

Para acceder a cualquiera de estas rutas, lo único que tendremos que hacer será hacer clic en la ruta que queramos y automáticamente iremos a la actividad dos: Información de la ruta.



## Actividad Ajustes

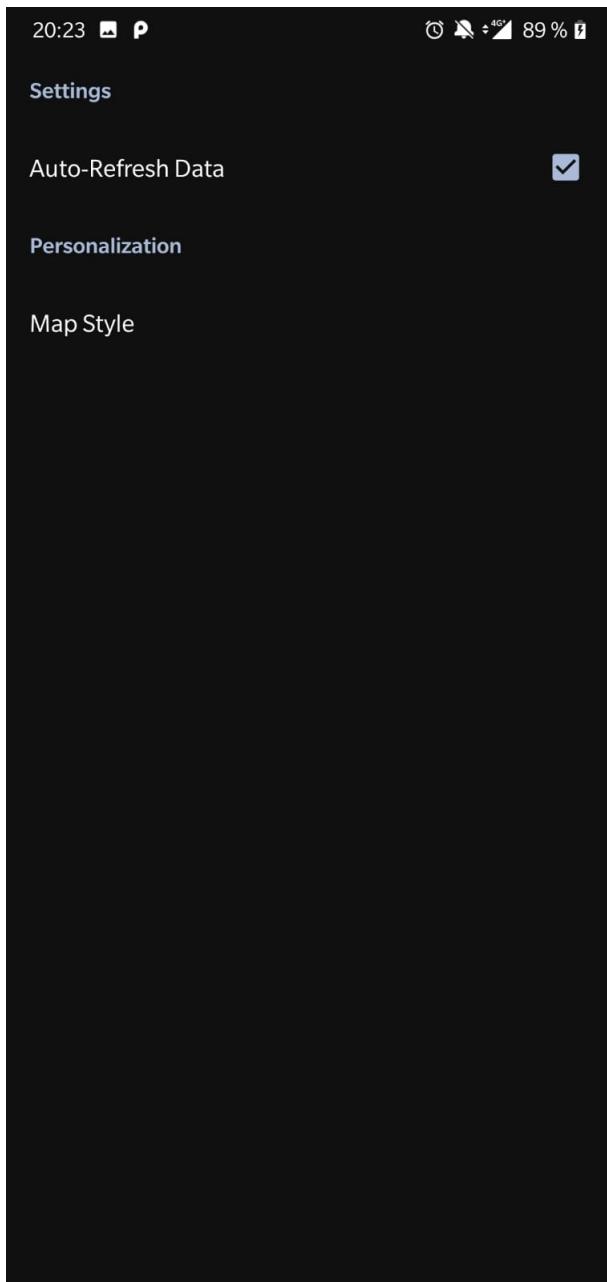
Por último, tenemos la pantalla de ajustes donde podremos seleccionar el aspecto que mostramos en el mapa y la función AutoRefresh, que recarga la información obtenida de cada ruta.

La primera opción, Auto-Refresh Data, es un CheckBox donde el usuario decide si sus rutas actualizan la información cada vez que se abre una de ellas.

Por otro lado, tenemos la personalización de WeNave. Es un desplegable con las siguientes opciones:

- Dark
- Light
- Mapbox\_Streets
- Outdoors
- Satellite
- Satellite\_Streets
- Traffic\_day
- Traffic\_night

Una vez el usuario decide el aspecto que quiere tener la aplicación, solo ha de volver a la pantalla que el quiera y podrá ver sus cambios efectuados.



## Presentación de menú

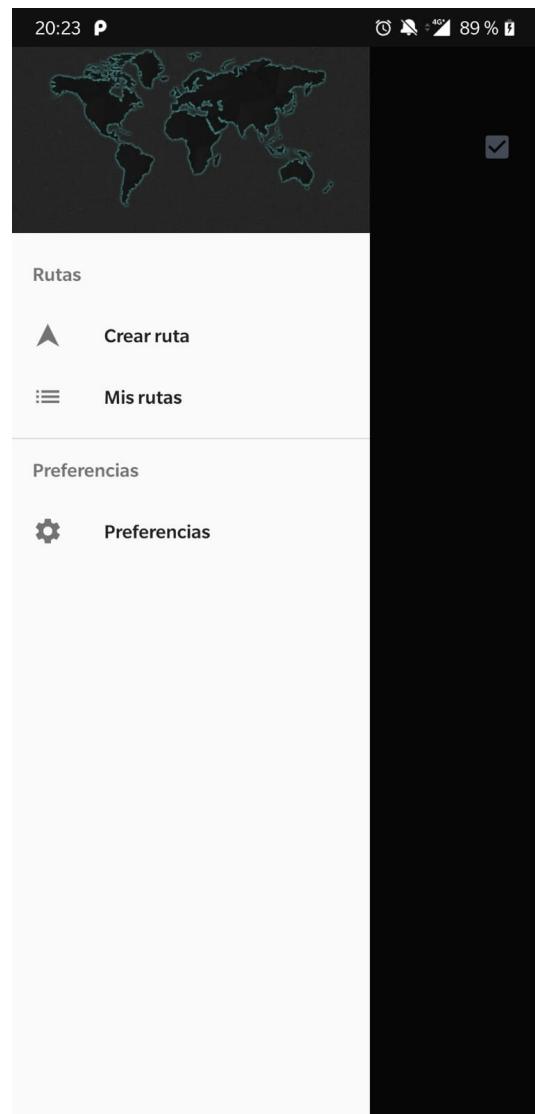
WeNave solo dispone de un menú lateral, donde podremos viajar entre las 3 pantallas:

- Crear rutas.
- Ver rutas
- Ajustes

Para acceder a este menú lo único que tendremos que hacer será desplazar el dedo de izquierda a derecha desde la parte izquierda de la pantalla.

Así descubrimos el siguiente menú:

Este menú esta habilitado en toda la aplicación, por lo que podremos acceder a cualquier funcionalidad estemos en la pantalla que estemos.



---

## Servicios externos

### Mapbox SDK y plugins

Hemos usado este SDK de mapas y más características ya que es la alternativa gratuita dada las nuevas limitaciones que ha puesto *Google* a *Maps SDK*. Además, cuenta con muy buena documentación y bastante comunidad detrás dónde puedes consultar problemas que surjan.

Lo hemos utilizado para cargar los mapas con los que interactuaremos. Nuestros mapas se cargan mediante una llamada a la API de esta compañía. En estos mapas podremos cambiar el estilo a través de la pantalla Ajustes.

Además de *Mapbox SDK*, también usamos *Location* para buscar sitios para agregar a nuestros puntos y *Annotation* para manejarnos con las capas de los puntos en nuestros mapas.

[docs.mapbox.com/android](https://docs.mapbox.com/android)

### Dark Sky

Hemos usado esta API para obtener los datos de climatología de nuestros puntos por dos principales razones: nos permite las primeras 1000 llamadas a la API de manera gratuita y los datos que nos devuelve sobre el clima son más exactos que otras APIs de climatología que hemos consultado.

[darksky.net/dev](https://darksky.net/dev)

### Realm

Hemos usado esta librería para poder gestionar nuestra base de datos. Nuestra aplicación guarda cada ruta, cada *forecast* de esta ruta y cada *point* en la base de datos.

Hemos usado esta librería dada su facilidad para almacenar objetos complejos que por ejemplo definamos con una clase, ya que con *Room* por ejemplo el almacenamiento con objetos anidados era muy complejo, ya que un objeto solo podía tener tipos primitivos dentro para que fuese relativamente fácil almacenarlo.

También es verdad que con *Realm* el acceso a los objetos no era fácil, ya que solo se podía acceder a objetos si el hilo donde se ejecutaba el código era el mismo hilo que había creado estos objetos. Para acceder a objetos que no se habían creado en el mismo hilo, se tenían que hacer consultas a la BD directamente.

[realm.io/products/realm-database](https://realm.io/products/realm-database)

---

## Interacción de la aplicación con Mapbox SDK

Nuestra aplicación carga los mapas mediante el SDK de *Mapbox*. Para esta tarea hemos creado una clase *MapBoxManagement* que se encarga de gestionar los procesos de *MapBox*.

Esto se hace mediante nuestra vista *MapView*, en la que le pedimos que nos devuelva un mapa de forma asíncrona mediante nuestra *API key* que nos proporciona *Mapbox*.

Una vez tenemos nuestro mapa, hemos de integrarle un estilo. Para esta tarea además hemos implementado en nuestra aplicación un selector de estilos en la pantalla ajustes. Cuando se integra el estilo también se añaden los recursos pertinentes, como la imagen que vamos a querer utilizar cuando añadamos puntos en el mapa.

Dicho esto hay que añadir que en nuestra clase *MapBoxManagement* tenemos dos métodos los cuáles nos cargarán nuestros mapas dependiendo en la pantalla que estemos, lo que nos ofrecerá más o menos funcionalidades dependiendo en que pantalla estemos.

El método *SetUpMapbox* nos devolverá el mapa y además podremos añadir y quitar marcas en nuestros puntos preferidos, ya que este mapa será el mapa de la vista “Creación de ruta”.

El método *SetUpMapBoxRoute* nos devolverá el mapa con los puntos que hemos marcado en la pantalla de “Creación de ruta” pero sin opciones adicionales como añadir nuevos puntos, ya que esta funcionalidad no estará disponible en la vista del resumen de la ruta y los *Forecast*; además llamaremos al *Adapter* para mostrar nuestros puntos en el *RecyclerView*.

---

## Descarga de datos desde Dark Sky

El funcionamiento es sencillo, cuando queramos la climatología de un punto de nuestra ruta le mandamos un *HttpRequest* para que nos devuelva una serie de datos en formato JSON de los cuales seleccionamos los que nos interesan con un *parser* que hemos implementado nosotros. Una vez tenemos estos datos los pasamos por un *Adapter* también implementado por nosotros para vincularlos a la interfaz gráfica.

Para ser lo más descriptivos posible, vamos a explicar que petición hacemos a la API. A continuación se muestra la URL a dónde vamos a hacer la petición:

`https://api.darksky.net/forecast/token/lat,lon,mil?exclude=[hourly,daily]&units=auto&lang=language`

`https://api.darksky.net/forecast`

Esta es la dirección web donde le vamos a hacer la petición a la API de un *forecast*.

`token`

El token que nos ha dado el servicio de Dark Sky.

`lat,lon`

Latitud y longitud del punto del que queremos obtener el *forecast*.

`mil`

Fecha de la que queremos extraer el clima del punto en segundos.

`exclude=[hourly,daily]`

Excluimos la predicción de la fecha que le hemos pasado por horas y la diaria. Sólo queremos la de ese momento exacto.

`units=auto`

Unidades de temperatura y más datos automática. Se basa en la localización geográfica.

`lang=language`

Lenguaje en el que queremos que nos devuelva el *summary* -> resumen del tiempo.

Al acabar de hacer la llamada a la API, ésta nos devuelve una JSON con el que comenzamos el proceso de parsear para obtener los datos que nos interesan. Estos datos son el *summary*, resumen del tiempo, *icon*, nombre del ícono asociado a la condición climatológica, *date* que nos devuelve la fecha del tiempo, *temperatura* para la temperatura, *humidity* para la humedad y *precipitation* para saber las precipitaciones.

Con estos datos ya podemos crear un objeto *Forecast* que utilizaremos más adelante.

---

## Enlace de los datos con la interfaz gráfica

Una vez tenemos descargados los datos y creado el objeto *RoutePoint* el siguiente paso es pasarlo con nuestra clase *Adapter* a la interfaz gráfica.

Para esto tenemos un *RecyclerView* en el que vamos a cargar objetos *CardView* con nuestros *RoutePoints*. Nuestro *Adapter* va a vincular los datos con las *views* dentro del *Cardview*. Con esto conseguimos que se vayan generando vistas *Cardviews* dinámicamente dependiendo de los objetos *RoutePoints* que le pasemos al *Adapter*. Al *Adapter* habrá que pasarle una lista con todos los *RoutePoints* que queramos mostrar.

Con el uso de *RecyclerView* respecto al *ListView* tenemos grandes ventajas, como:

Con *RecyclerView* podemos scrolllear vertical y horizontalmente, mientras que con *ListView* sólo se puede verticalmente.

*RecyclerView* tiene bastantes más opciones de personalización en las animaciones que *ListView*.

Con *RecyclerView* se puede decorar los elementos dinámicamente de manera más fácil.

Respecto a los contras, podemos decir que es un poco más complejo que una *ListView*, el código es más difícil de entender que el de un *ListView* y se tarda más tiempo en implementarlo.

---

## Guardar en BD con Realm

Para guardar objetos en *Realm* basta con llamar al método de esta librería *createObject* pasándole como argumento el tipo de clase que es, como por ejemplo *Route.class*. Una vez hecho esto *Realm* automáticamente asigna al objeto un *id* ya que en la clase hemos anotado *id* como *PrimaryKey*.

Una vez creado el objeto tenemos que asignarles los valores que queramos, pero es obligatorio hacerlo dentro de una transacción, ya que si no se hace un *beginTransaction* y un *commitTransaction* los valores no se quedarán guardados en el objeto.

Si nos abstraemos al funcionamiento general de una base de datos, vemos que es la forma segura de hacer una operación en un BD, ya que si el guardado de datos no se hace dentro de una transacción puede haber problemas como que se guarden unos valores y otros no o hayan datos corruptos.

Haciendo las operaciones con transacciones nos ahorraremos estos problemas además de que con *Realm* es obligatorio el uso de éstas, ya que sin transacciones no te deja guardar los datos.

---

## Actualizar o eliminar datos con Realm

*Realm* se utiliza en nuestra aplicación cuando queremos actualizar el tiempo de una ruta, ya que habrá que rescatar los *Forecast* y actualizar sus datos, o en eliminar una *Route* o varias, también usaremos la opción de eliminar.

Es importante saber, cómo en el caso anterior de guardar los objetos, que todas estas operaciones se tienen que efectuar mediante una transacción ya que *Realm* pone estos requerimientos.

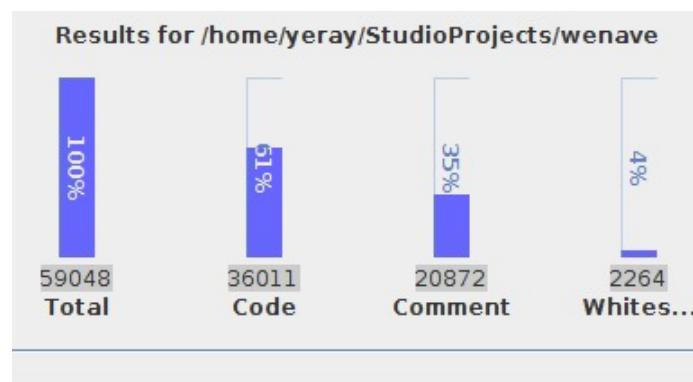
También es importante apuntar que si no estamos en el hilo que creó el objeto, habremos de rescatar el objeto con una *query* para realizar las operaciones oportunas.

# Análisis de código

## CodeAnalyzer

Hemos usado CodeAnalyzer para calcular métricas de nuestro proyecto tales como:

- Archivos totales
- Lineas Totales
- Media del tamaño de linea
- Lineas de código
- Lineas de comentarios
- Lineas en blanco



Metric	Value
Total Files	120
Total Lines	59048
Avg Line Length	55
Code Lines	36011
Comment Lines	20872
Whitespace Lines	2264
Code/(Comment+Whitespace) Ratio	1.56
Code/Comment Ratio	1.73
Code/Whitespace Ratio	15.91
Code/Total Lines Ratio	0.61
Code Lines Per File	300
Comment Lines Per File	173
Whitespace Lines Per File	18

Los resultados de este test se adjuntan en un archivo comprimido con este mismo proyecto.

---

## FindBugs

Esta herramienta nos ayuda a depurar el código, a parte de ayudarnos a ver donde estamos usando nombres inapropiados para una clase o donde estamos definiendo variables que no usamos nunca.

Herramienta indispensable para la elaboración de proyectos de larga duración.

## FindBugs Report

### Project Information

Project: weNave

FindBugs version: 3.0.1

Code analyzed:

- /home/yeray/StudioProjects/wenave/app/build/intermediates

### Metrics

0 lines of code analyzed, in 0 classes, in 42 packages.

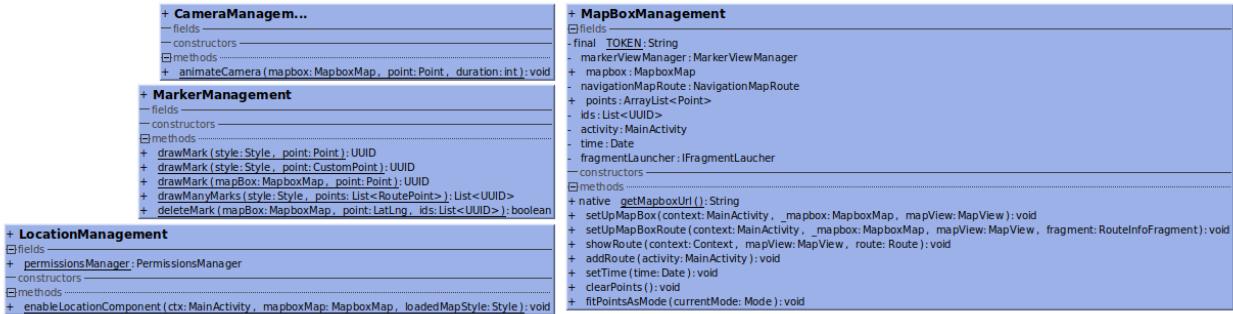
Metric	Total	Density*
High Priority Warnings	1	0.00
Medium Priority Warnings	376	0.00
Low Priority Warnings	728	0.00
<b>Total Warnings</b>	<b>1105</b>	<b>0.00</b>

(\* Defects per Thousand lines of non-commenting source statements)

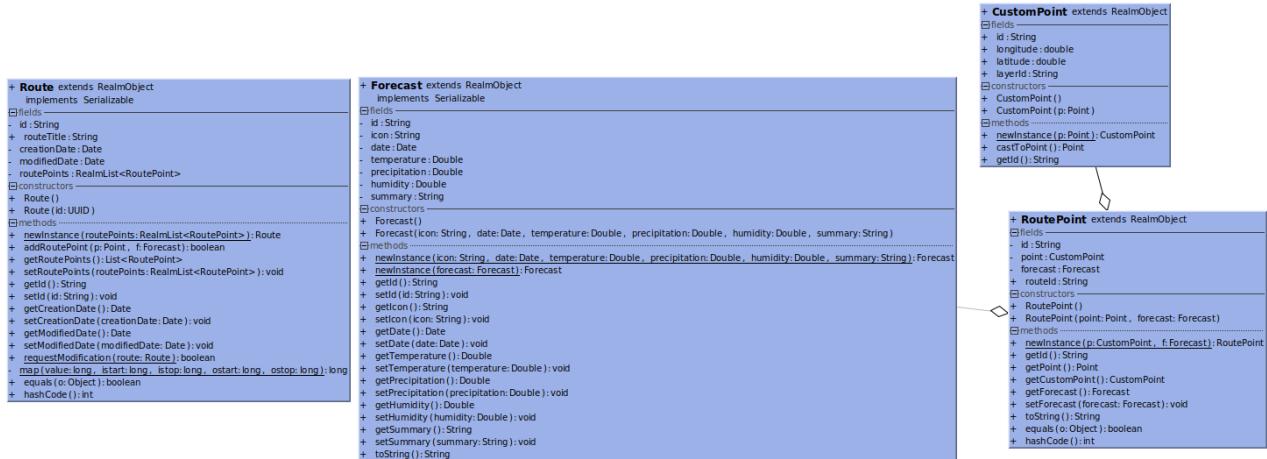
Los resultados de este test se adjuntan en un archivo comprimido con este mismo proyecto.

# UML

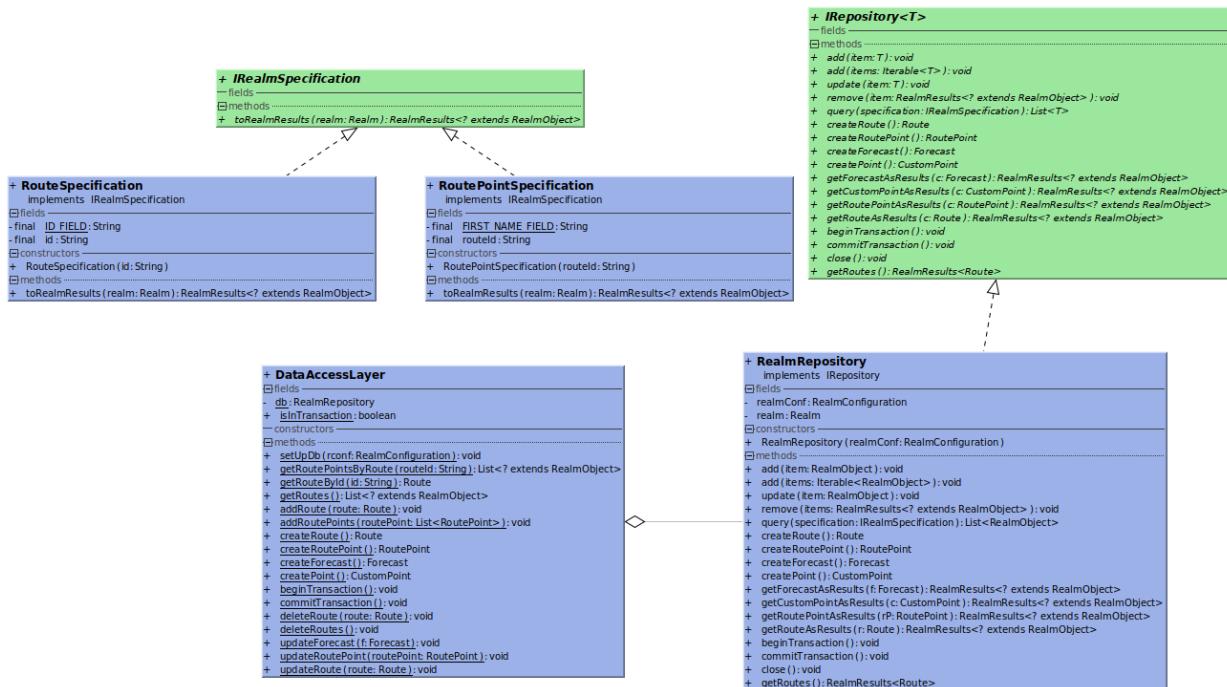
## Management:



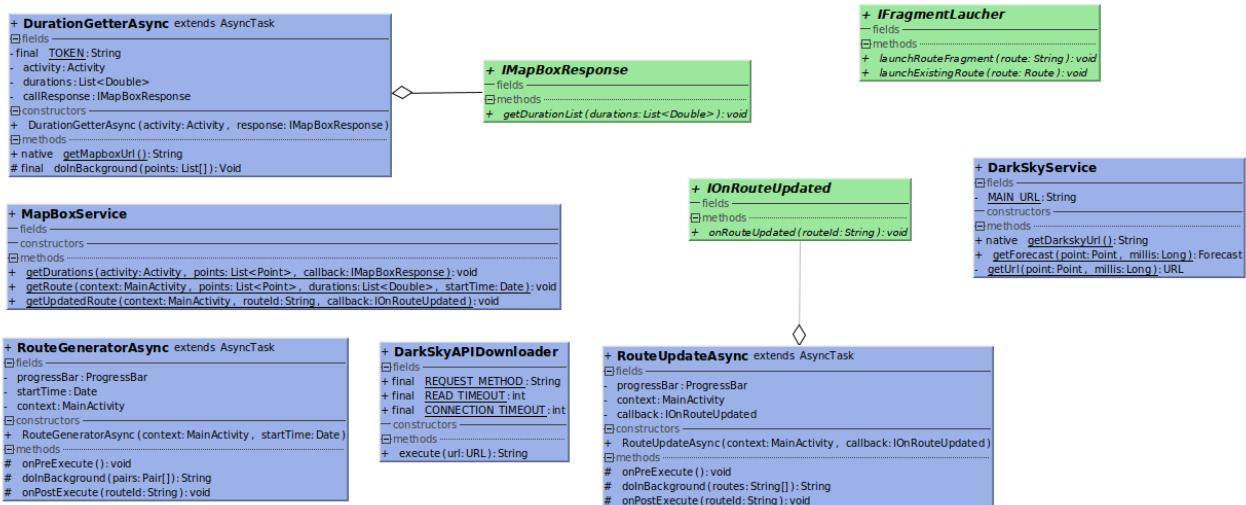
## Models



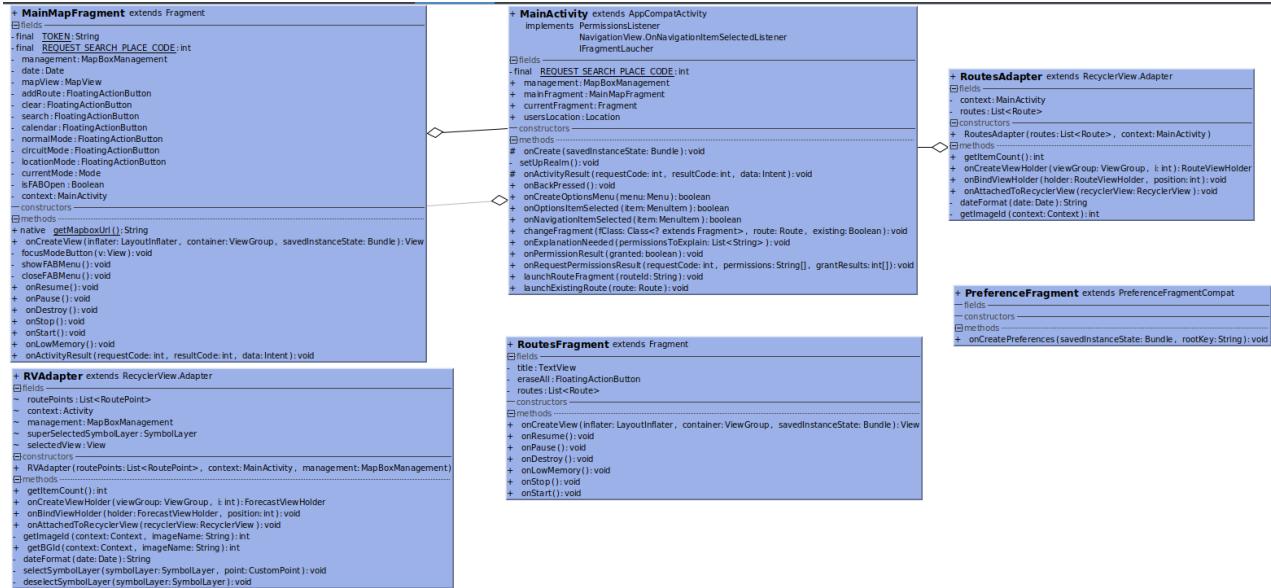
## Persistence



## Services



## Views



---

## Casos de Uso

Los siguientes casos de uso han de interpretar por orden de puntuación siendo 1 el primer paso, 2 el segundo, etc.

Caso de uso, definir ruta:

1	Definir 2 o mas puntos
2	Abrir menú
3	Seleccionar la opción de ruta

Caso de uso, guardar ruta:

1	Definir dos o más puntos
2	Abrir menú
3	Seleccionar la opción de ruta
4	Seleccionar el botón de guardar,
5	Especificar nombre de la ruta
6	Pulsar guardar.

Caso de uso, definir dia de ruta:

1	Abrir menú
2	Seleccionar calendario
3	Definir fechas
5	Seleccionar aceptar.

---

### Caso de uso, eliminar punto de ruta:

1	Seleccionar uno o mas puntos
2	Abrir menú
3	Seleccionar la papelera

1	Seleccionar uno o más puntos
2	Seleccionar un punto con el dedo
3	Mantener pulsado el punto 1 seg. aprox

### Caso de uso, seleccionar lugar:

1	Abrir menú.
2	Seleccionar la lupa
3	Definir un lugar
4	Seleccionar aceptar

### Caso de uso, ver rutas guardadas:

1	Abrir menú lateral.
2	Seleccionar "Mis rutas".

### Caso de uso, eliminar una ruta:

1	Abrir menú lateral.
2	Seleccionar "Mis rutas".
3	Seleccionar el botón en forma de X de la ruta que queramos borrar

### Caso de uso, eliminar todas las rutas:

1	Abrir menú lateral
2	Seleccionar "Mis rutas"
3	Seleccionar la papelera en la esquina inferior derecha

---

Caso de uso, ver detalles de ruta guardada:

1	Abrir menú lateral.
2	Seleccionar “Mis rutas”.
3	Seleccionar una ruta.

Caso de uso, actualizar la información de una ruta guardada:

1	Abrir menú lateral.
2	Seleccionar “Mis rutas”
3	Seleccionar una ruta.
4	Seleccionar el botón situado en la esquina inferior derecha con forma de flecha.

Caso de uso, habilitar el modo actualización automática:

1	Abrir menú lateral
2	Seleccionar “Ajustes”
3	Habilitar la opción de actualizado automático

Caso de uso, cambiar el estilo de mapa:

1	Abrir menú lateral.
2	Seleccionar “Ajustes”.
3	Seleccionar tipo de mapa.
4	Seleccionar opción del desplegable.

---

## Anexo

Fuentes de iconos:

<https://www.flaticon.com/>

Imágenes de fondo de pantalla:

Google bajo licencia no comercial