

La Figura muestra el diagrama de clases del aplicativo Guíame, organizado en cuatro grupos principales: interfaz de usuario (UI), gestión de datos, módulo de mapas y navegación y servicios auxiliares.

En la capa de interfaz de usuario se ubican las actividades y fragmentos que el usuario utiliza de forma directa. `BaseActivity` y `BaseGestureActivity` concentran el comportamiento común de las pantallas, incluyendo la gestión de idioma, el ciclo de vida y, en este último caso, la detección de gestos de deslizamiento. A partir de ellas se derivan las actividades principales: `MainActivity`, que funciona como pantalla de inicio y permite acceder mediante gestos de deslizamiento a las secciones de configuración, gestión de destinos y navegación; `ConfigurationActivity`, que agrupa la configuración de mapas (en línea u offline), archivos `MBTiles/GeoJSON`, idioma, tipo de retroalimentación, volumen, altura del usuario y uso de pasos; `DestinationActivity`, que muestra la lista de destinos usando un `RecyclerView` y los adaptadores `DestinoAdapter/DestinationAdapter`; `AddDestination`, encargada de registrar o editar un destino apoyándose en un mapa en línea u offline según la configuración seleccionada; y `NavigationActivity`, que concentra la lógica de navegación paso a paso (selección de destino, cálculo de ruta y emisión de instrucciones por voz y pitidos). El fragmento `DestinationsSheet` se utiliza desde `NavigationActivity` como hoja inferior (bottom sheet) para mostrar y seleccionar destinos durante la navegación.

En la capa de gestión de datos se utiliza el framework `Room` para almacenar de forma local los destinos registrados. La entidad `Destination` representa cada destino, con atributos para el identificador, nombre, latitud, longitud y fecha de creación. La interfaz `DestinationDao` define las operaciones CRUD sobre la tabla de destinos (insertar, actualizar, eliminar, consultar la lista completa u obtener por identificador). La clase `AppDatabase` extiende de `RoomDatabase` y proporciona el acceso centralizado a `DestinationDao` mediante el método estático `get(context)`, implementado como un singleton. Tanto `DestinationActivity` como `DestinationsSheet` utilizan esta base de datos para mostrar y gestionar los destinos.

El módulo de mapas y navegación agrupa las clases relacionadas con la representación cartográfica, el filtrado de la posición y la generación de instrucciones. La interfaz `IMapController` define las operaciones básicas sobre el mapa (adjuntar el mapa a un contenedor, centrar la cámara, dibujar la ruta, actualizar la posición del usuario, limpiar polilíneas). En modo en línea, esta interfaz se implementa mediante un controlador que encapsula `GoogleMap`; en modo offline se implementa a través de un controlador basado en `VectorMapFragment`, que carga la información cartográfica desde archivos locales y se apoya en `LocalTileHttpServer` para servir los mosaicos del mapa.

En cuanto a la navegación, `LocationService` es un servicio en primer plano que solicita actualizaciones periódicas de la ubicación del dispositivo y las envía a `NavigationActivity`. La interfaz `Tracker` y su implementación `PFTracker` permiten filtrar la trayectoria mediante un filtro de partículas, reduciendo el ruido de la señal GPS. La clase `ENU` y el conversor `ENUConverter` se utilizan como sistema de coordenadas local para trabajar con las posiciones, representando la ubicación en coordenadas planas (x, y) alrededor de un punto de referencia. Sobre estos componentes se construye `GpsImuParticleFilter`, que integra la información de GPS e IMU: recibe controles derivados de los sensores inerciales (velocidad y velocidad de giro) y actualiza internamente un `PFTracker`, utilizando `ENUConverter` para operar en el plano local y obtener una estimación más estable de la posición. El manejo de la unidad de medida se complementa con `StepEstimator`, que a partir de la altura del usuario convierte directamente distancias en metros al número de pasos, permitiendo expresar las indicaciones en pasos cuando así se haya configurado.

Para el cálculo de rutas se dispone de dos componentes: `RouteManager`, encargado de solicitar rutas en línea, y `OfflineRouteManager`, que realiza el cálculo sobre los datos de mapa offline. `TurnByTurnEngine` recibe la ruta como una secuencia de puntos y detecta los giros relevantes; a partir de la distancia al siguiente giro y de umbrales predefinidos genera el texto base de las instrucciones (por ejemplo, “En 60 pasos gire a la derecha” o “Ahora gire a la izquierda”). `TurnCueManager` interpreta estas instrucciones y, de acuerdo con las preferencias almacenadas, decide si deben emitirse por voz (mediante síntesis de texto a voz) y/o mediante patrones de pitidos 2D lateralizados. `ImuManager` e `ImuControls` gestionan los datos de los sensores inerciales y permiten que la navegación tenga en cuenta los movimientos del usuario al estimar su orientación y su velocidad.

Por último, en la capa de servicios auxiliares se incluyen componentes de apoyo que no pertenecen directamente a una pantalla concreta pero son utilizados por varias de ellas. `FeedbackManager` se emplea principalmente desde `ConfigurationActivity` para inicializar y probar la realimentación del sistema (por ejemplo, reproducir ejemplos de voz o pitidos) sin necesidad de iniciar una navegación completa. `PermissionHelper` encapsula la comprobación y solicitud de permisos de ubicación, facilitando su reutilización desde `MainActivity` y otras actividades.

Esta estructura en capas permite que la lógica de navegación, el manejo de mapas y la gestión de datos sean reutilizables por distintas pantallas, mientras que las actividades de la capa de interfaz se concentran en la interacción con el usuario y en asegurar la accesibilidad del aplicativo `Guíame`.