

“Pokédex”

Emil Peñaló Colón y Marcos Blanco Durán

Ciencias de la Computación; Facultad de Ciencias e Ingeniería,

Pontificia Universidad Católica Madre y Maestra

ICC 451: Desarrollo Móvil


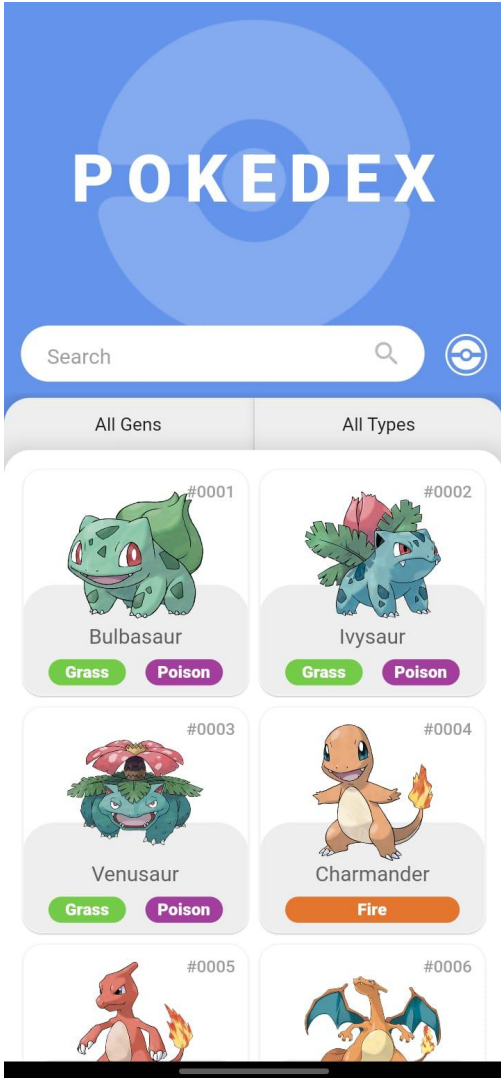
Freddy Peña

Diciembre 6, 2023

Índice

Páginas de la Aplicación.....	3
Uso de la Aplicación.....	5
Diseño.....	8
Tecnologías.....	9
Estructura de Código.....	11
Utilización de la Poké API.....	13
Persistencia.....	15

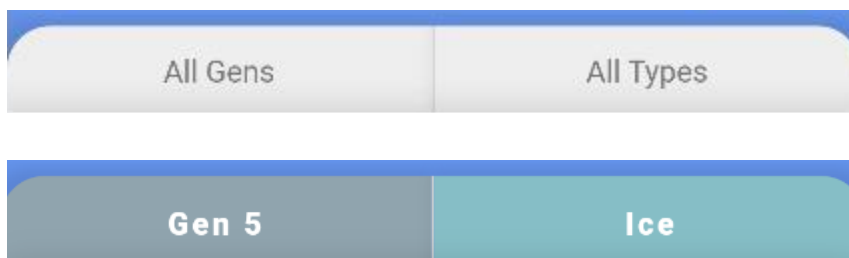
Páginas de la Aplicación

Páginas de la aplicación	
Cargando	Listado General
	
<p>Página inicial donde se cargan los pokemones en la base de datos. También se utiliza en la transición de páginas entre un listado y el detalle del pokémon seleccionado.</p>	<p>El listado general muestra un listado paginado de cartas de pokemones con su nombre, tipo, id e imagen. También, permite ‘capturar’ un pokémon. También se tiene una búsqueda y filtros por generación y tipo que afecta el listado.</p>

Uso de la Aplicación

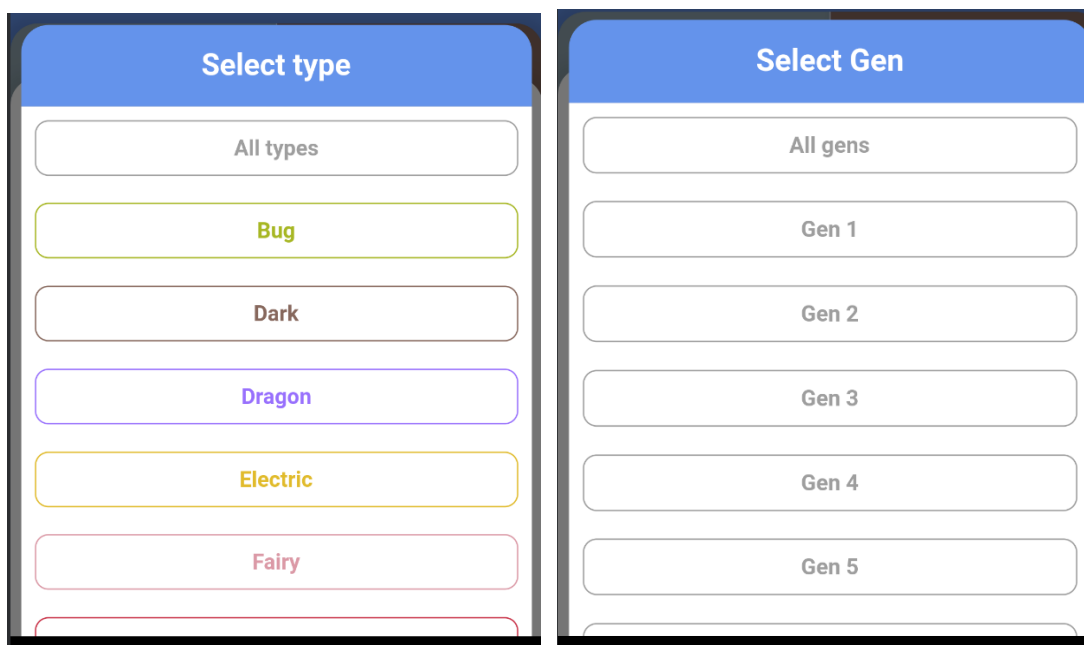
Búsqueda: El usuario puede utilizar el input de búsqueda para buscar un pokemon en base a su nombre o id. Si el input es numérico, se considera id, de lo contrario, se considera nombre. En el caso de la búsqueda por nombre, esta resulta en todo pokémon cuyo nombre que contenga el texto de búsqueda, no necesariamente debe iniciar con el texto. En el caso de búsqueda por id, se busca un pokémon cuyo id es idéntico al indicado por el usuario. No se utiliza un like search para evitar encontrar los pokemones 10, 11, 100, 101, 111, ... al buscar el pokemon con id #1 por ejemplo. Para resetear la búsqueda, se puede buscar un texto vacío. El input de búsqueda sólo afecta el listado presentado en la pantalla en el momento, es decir, que aunque sea el mismo widget, una búsqueda en capturados no afecta el listado general.

Filtros: Adicionalmente de la búsqueda, está el filtro por tanto tipos como por generaciones. Solo se puede seleccionar uno de cada opción pero se pueden combinar entre sí para una búsqueda más específica, esto sin mencionar la búsqueda escrita.



En la figura se muestran los filtros inactivos y activos respectivamente.

Al abrir los filtros se abre un modal que permite seleccionar la generación o tipo deseado utilizando un DraggableScrollableSheet que permite que el usuario utilice un scroll para ver todas sus opciones. Este aún permite dar click fuera de la pantalla para cerrarse.



Capturados: En todo listado de pokemons, un pokemon puede ser capturado utilizando el gesto de double tap. Esto invierte el estado de capturado del pokémon.

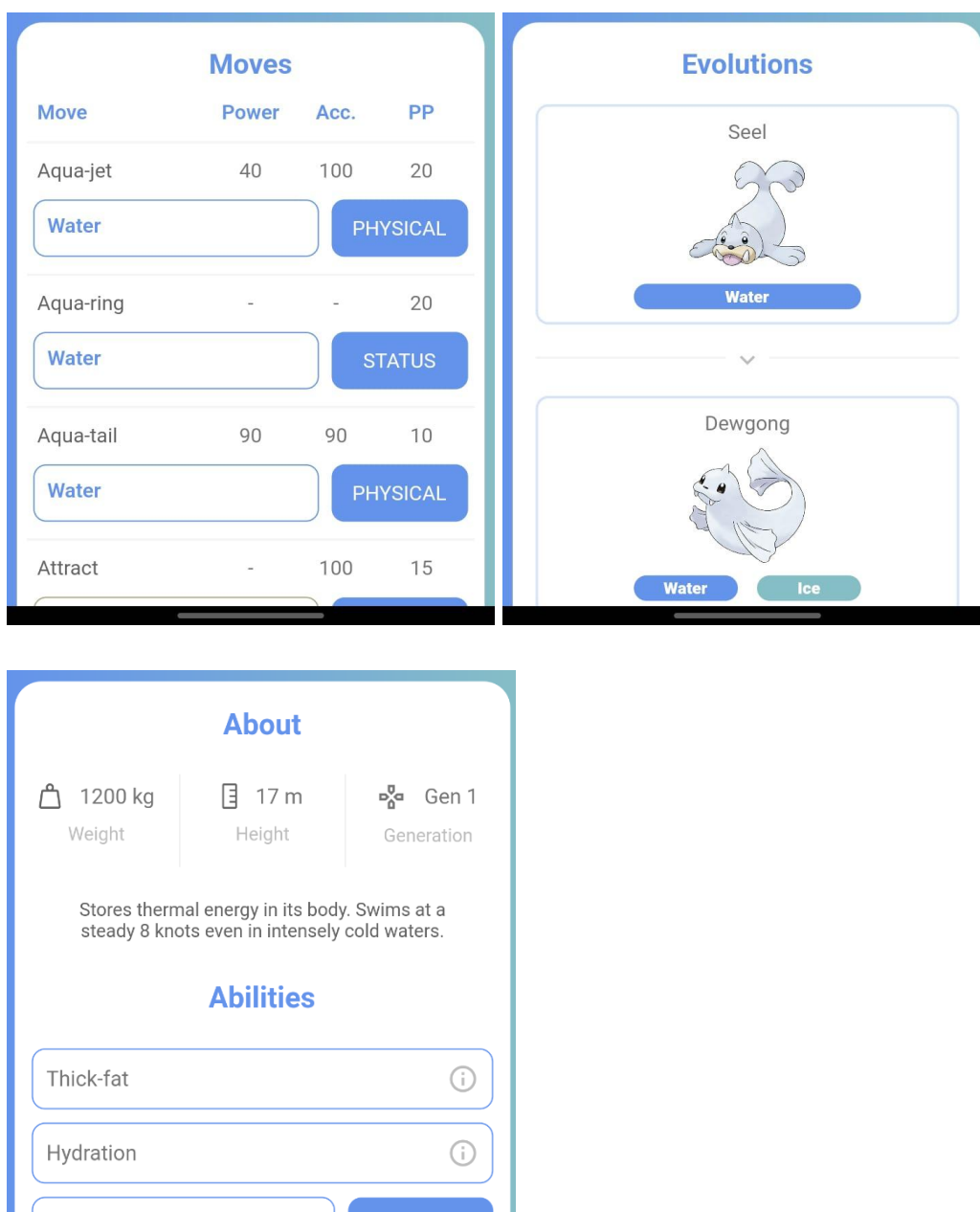
Detalles del Pokémon: Los detalles de un pokémon pueden accederse usando un solo toque a un pokémon en específico dentro de una lista, ya sea general o filtrada de alguna manera.


Navegación Entre Pokemons: Dentro de la página de detalles de un pokemon se tiene la opción de usar los botones de navegación interna (< y >) para ir al pokemon siguiente o previo según su id. No solo aquí, sino incluso en evolutions, existe la posibilidad de poder seleccionar un pokémon de tal cadena.

Información de un Pokémon: Los pokemones tienen un slider dentro de su vista detallada. Esto permite ver la información organizada de manera fácil. El usuario puede hacer slide de izquierda a derecha y viceversa para navegar entre los slides. Los slides totales y el slide actual es presentado al usuario usando los siguientes iconos:

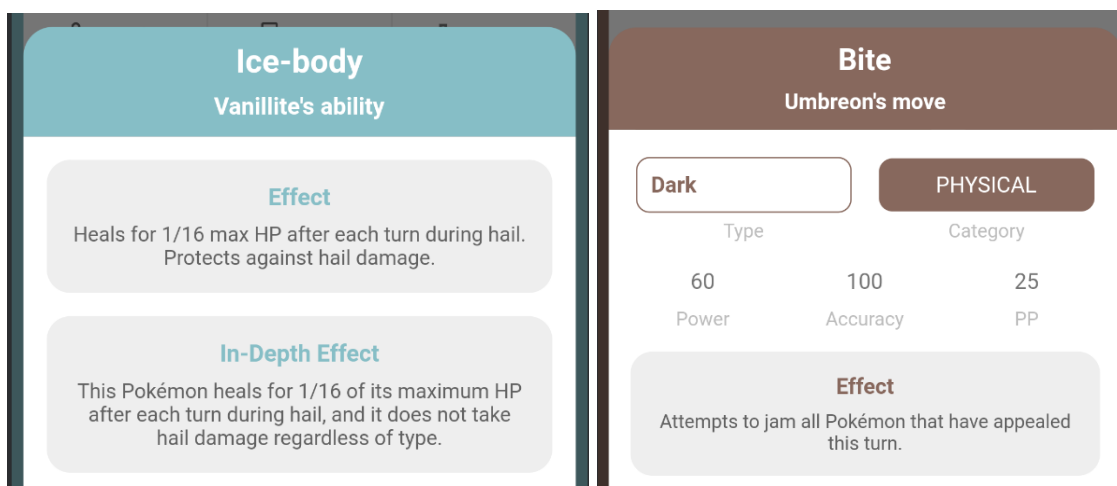


Dentro de estos slides encontramos las secciones de “About”, ”Moves” y “Evolution” mostrando sus respectivas informaciones.




Aquí mismo, se le puede dar click a los contenedores con el siguiente icono  para saber informaciones adicionales del contexto a saber.

Información Adicional de Abilities y Moves: Como explicado anteriormente, al darle click a una habilidad o un movimiento con el icono de información adicional, se abre un modal desde abajo que toma como color principal el del pokémon que se está visualizando.



Diseño


Colores: Usamos un azul para el color principal de la página, ya que era más amigable que el rojo normalmente asociado con un pokedex. Los colores de los tipos de cada pokemon fueron tomados de otros proyectos de diseño públicos de pokédex pero fueron ajustados para su uso en la variedad de situaciones que requiere la app, es decir, su uso como fondo, de color de texto, etc. Por último, los colores de los tipos representan los colores usados en el fondo de la página de detalle como gradiente. También se utiliza el color del tipo principal como color de los encabezados y gráficos presentes en las informaciones del pokemon.

Imágenes: El api proporciona una variedad de imágenes, para la app, se decidió utilizar “official artwork” ya que está disponible para todos los pokemons y según la consideración de los desarrolladores, se ven mejores que las demás opciones. También cabe destacar que existe, para algunos pokemons, la versión “shiny” de tal imagen. Para acceder a ello solo se le tiene que dar click al botón , arriba a la derecha, dentro de detalles.

Cargado Inicial: Dado a la necesidad de utilizar una base de datos, se requiere cargar y/o actualizar los datos a un principio al inicio de la aplicación. Dado a que esto toma sus segundos

dependiendo del internet y el dispositivo del usuario, se implementó una página de cargado para mostrarle al usuario una animación bonita mientras espera. En adición se muestra el progreso del cargado. Cabe destacar que el chequeo principal es en base a la cantidad de pokemons que hay entre la api y la guardada en la base de datos.

Header del Listado Principal: Se utilizó un SliverAppBar para presentar un header grande y bonito al inicio del listado, pero al iniciar un scroll, pues este se esconde para permitir que el usuario vea mas pokemones a la vez y que el header no los estorbe.

Icono de Capturados: El icono de capturados, , representa el estatus de capturado de un pokemon. Este utiliza el color principal de la aplicación y el icono de la pokebola que se reutiliza en varias áreas de la app. Este tiene una animación de entrada y de salida según la interacción con el usuario.

Tecnologías

Paginación: Para una carga suave al momento de mostrar grandes conjuntos de datos de manera incremental a medida que el usuario se desplaza por una lista o vista. Esto es efectuado en todo listado de pokemones en la aplicación.

Gráfico de Estadísticas: Se utiliza *fl_chart* para graficar los stats del pokémon en la página de detalles. Esta es una librería que permite hacer gráficos como el mostrado abajo:

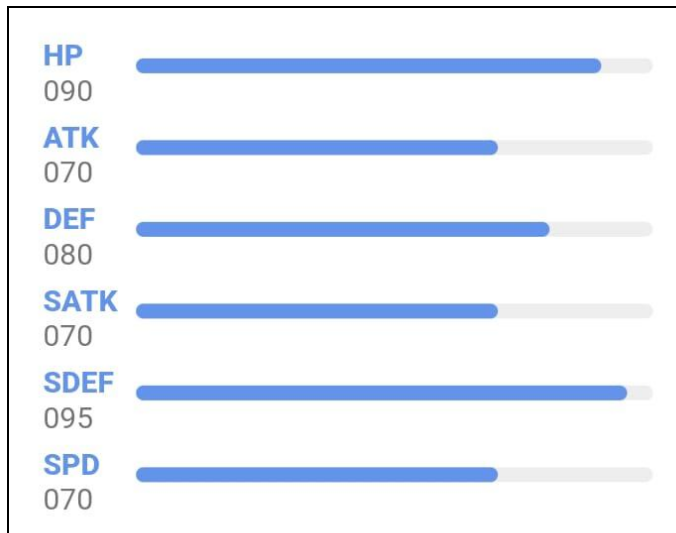


Imagen de Pokemon: Cached Network Image es utilizado para cargar y almacenar en caché imágenes desde la red de manera eficiente. Toma efecto al cargar cualquier imagen de un pokémon, ya sea en el listado general o la vista de detalles.

Slider: Se utilizó *smooth_page_indicator* para poder mostrar los indicadores en la cual el usuario se encuentra viendo en los detalles del pokémon.

Sqflite: Para poder interactuar con bases de datos SQLite.

Path: Para facilitar la manipulación y gestión de rutas de archivos y directorios.

GraphQL: Tiene la capacidad de solicitar sólo los datos que necesitan y nada más. Es utilizada en la carga inicial de pokemons para reducir el tiempo de carga al buscar los datos adicionales de cada pokemon que el rest API no provee de manera directa.

Shared Preferences: Utilizado para guardar datos pequeños o unitarios que no requieren de una base de datos.

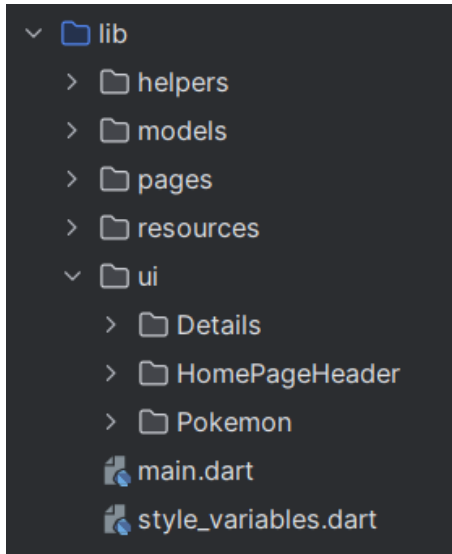
Estructura de Código

Para ver una imagen de la estructura general, ver Figura E.1

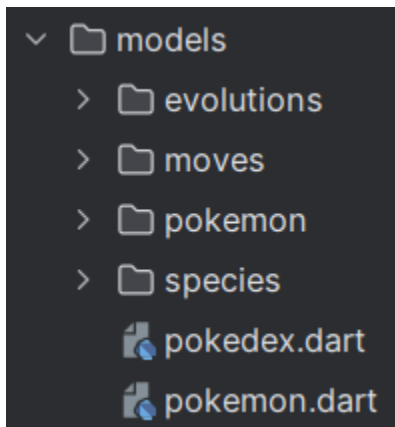
La estructura del código es la siguiente: main.dart inicializa la base de datos y hace el movimiento a la página de inicio (HomePage), donde se encuentra el listado general. Tal página, y las demás, se encuentran en la carpeta de pages. Aquí encuentras todas las páginas principales de la aplicación, así como el loading screen, pokemon list, pokemon details, entre otras. También existe la carpeta resources que almacena los recursos estáticos, como imágenes e iconos externos. Existe una carpeta llamada helpers que tiene la base de datos y otras funciones que ayudan a reutilizar ciertas funciones que se hacen repetitivas en casi todo el código como colocar algún texto en mayúscula, el cargado de imágenes o acceso a la base de datos. Así mismo, está la carpeta UI que contiene los recursos que también son reutilizables en la aplicación como los cards de los pokemons o el app bar. Igualmente están los widgets que, aunque sea una vez que aparecen, al sacarlo aparte, mejoran la organización del código y su legibilidad, como es el caso del gráfico de estadísticas del pokémon.

Evidentemente los modelos DTO están en su respectiva carpeta y organizados ahí dentro a la vez, según su rol de información dentro del pokemon, basándonos en la estructura del api.

Ver Figura E.2 abajo.



(Figura E.1) Visualización de la estructura del código.



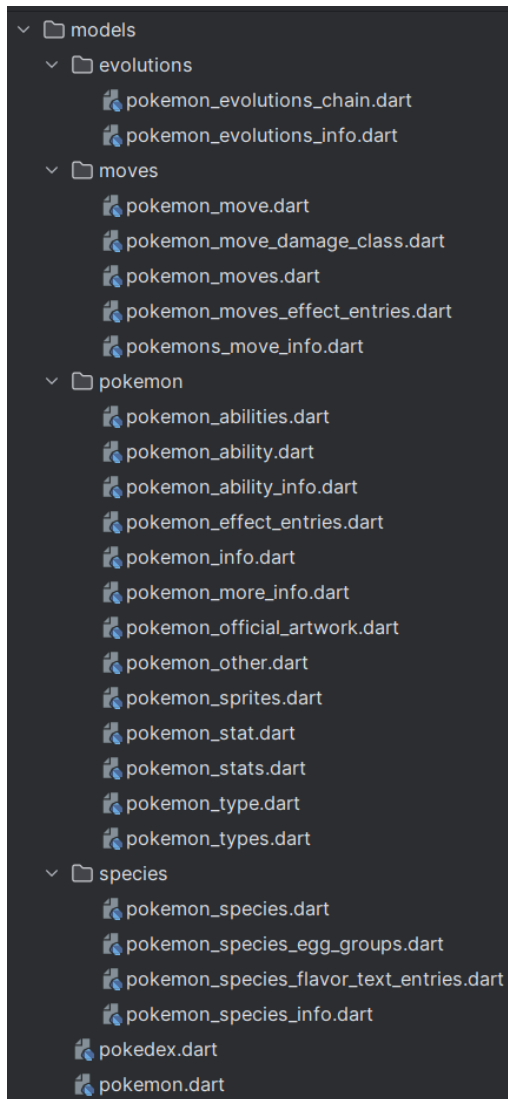
(Figura E.2) Visualización de la estructura de los modelos DTO

Utilización de la Poké API

Se extrajeron los datos necesarios del POKEAPI y se diseñó la clase Pokedex, que contiene un listado de Pokémon con su nombre y URL originales, a los que se les añadió su correspondiente ID y un booleano para indicar si fueron capturados (este último reemplazó el método original de favoritos). Esta fue la estructura principal del sistema, que también se almacena en una base de datos que se actualiza al abrir la aplicación.

Una vez obtenida la URL de un Pokémon específico, se extraen cuatro categorías principales: "pokemon_info," "pokemon_species," "pokemon_moves" y "pokemon_evolution." La primera incluye información como sprites, tipos, especie, altura, peso, estadísticas, habilidades y movimientos. La mayoría de estos datos tienen subclases para completar su información. Es importante mencionar que las dos primeras mencionadas, "sprites" y "types," junto con la información del Pokémon almacenada en la base de datos, se utilizan de manera paginada para la vista del listado de Pokémon, mientras que la información adicional de "pokemon_info" se emplea para la vista de detalles de un Pokémon. En el caso del listado general y la página de detalles, solo se carga del api la información que será mostrada.

En la vista de detalles, se recuperan las otras tres categorías principales, "pokemon_species", "pokemon_moves", "pokemon_evolution" y que también cuentan con subclases para completar su información. En el caso de especies, se obtienen los huevos al que pertenece el pokémon, su hábitat, una breve descripción y la url de su cadena de evolución. Todo esto permitiendo valores nulos ya que hay pokemones que carecen de tales informaciones. En el caso de los movimientos, se sacan los stats y las descripciones de tales. Por último, evolutions se encarga de realizar una llamada recursiva con la información dada por especies.



(Figura API.1) Visualización detallada de los archivos DTO y su estructura.

Persistencia

La primera vez que la aplicación se ejecuta, se crea una base de datos que almacena información, gracias a GraphQL, sobre los Pokémon, incluyendo su ID, nombre, tipos e incluso generación y un indicador de si han sido capturados por el usuario. Esta última variable es esencial para determinar qué Pokémon han sido atrapados.

Como se mencionó previamente, cada vez que se abre la aplicación, se consulta el API para verificar si ha habido algún cambio, como la adición de nuevos Pokémon. El "database_helper" proporciona las operaciones de creación, lectura, actualización y eliminación de registros en la base de datos. La función de eliminación se utiliza dentro del proceso de debugging pero no se ejecuta por la aplicación en su uso regular.

La lista de Pokémon se utiliza no solo para mostrar todos los Pokémon disponibles, sino también es alterado para poder ser usado para buscar. Los queries a la base de datos se utilizan para obtener listados filtrados por texto de búsqueda, id, y por pokemon individual. También por tipos y generación del mismo. Para ello, se utilizan funciones que extienden de un query general a la bd.

Por último, hay un caso particular al buscar el número de generaciones, para evitar una llamada adicional al api y para no guardar un solo dato en una tabla en una base de datos innecesariamente, se utiliza shared_preferences para guardar el número de generaciones. Este valor se encuentra al cargar inicialmente los pokemones a la base de datos según las evoluciones obtenidas de los pokemones. Este valor es guardado en una variables estática en DatabaseHelper para poder acceder a este dato desde otros widgets.