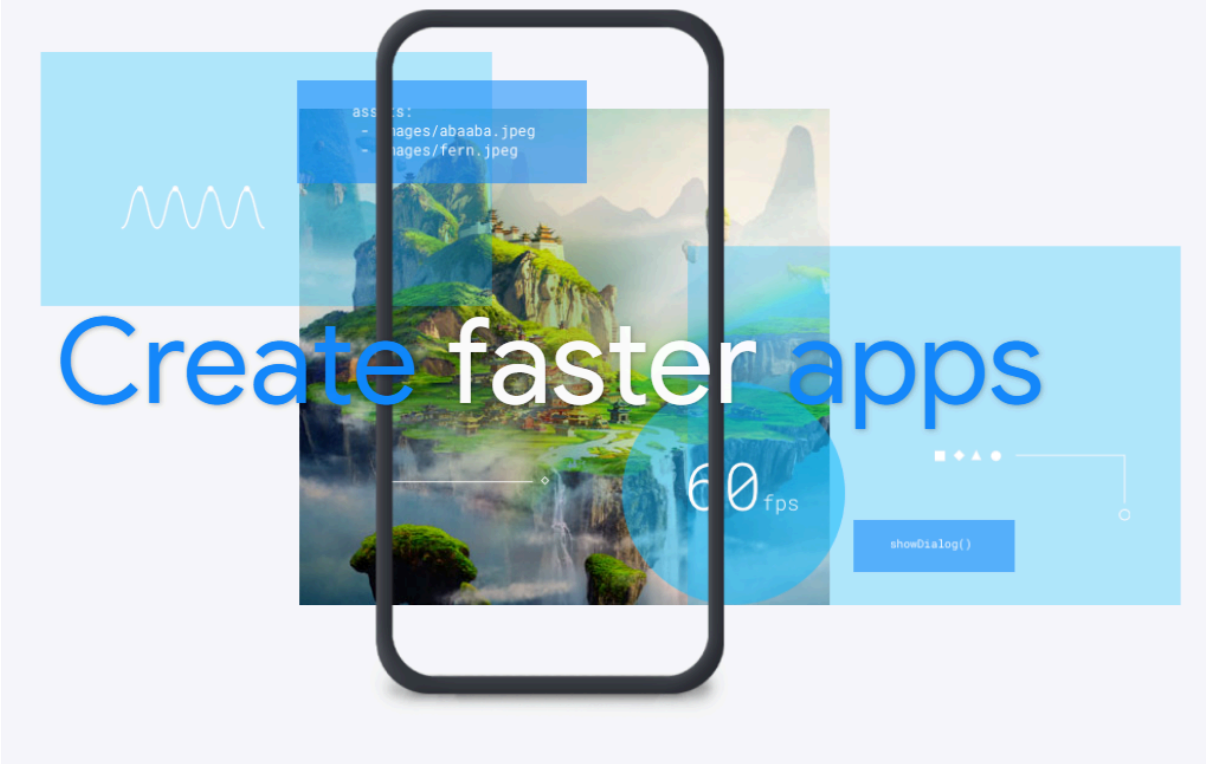
INICIATIVA ALETEO

RETO ENTOMOLOGO



Introducción

Como entomólogo especializado en la clasificación y estudio de insectos, mi trabajo diario implica la recolección y conteo de muestras de diferentes especies de insectos en diferentes hábitats. A menudo, necesito llevar un registro detallado de estas muestras y su ubicación, para poder analizar y entender mejor las tendencias y patrones de distribución de los diferentes insectos.

Solicitud del cliente

Para ayudar en mi trabajo diario, necesito una aplicación móvil que me permita realizar un seguimiento de mis muestras de insectos y contarlas de manera eficiente. La aplicación debe ser fácil de usar y tener varias funciones que me ayuden a organizar y analizar los datos de mis muestras.

A continuación, propongo tres pantallas o vistas que la aplicación debe tener:

1. Pantalla de inicio: Esta pantalla debe permitirme ver una lista de mis muestras de insectos recientes, y también me debe dar la opción de agregar una nueva muestra. Además, debe haber una barra de búsqueda para que pueda buscar rápidamente una muestra específica en función de su ubicación o especie.
2. Pantalla de conteo: Cuando selecciono una muestra específica en la pantalla de inicio, esta pantalla debe permitirme contar los insectos de esa muestra en función de su especie. Debe haber una lista desplegable con las diferentes especies de insectos que he clasificado previamente, y un botón para agregar el número de insectos de cada especie a la muestra. Además, debe haber una sección para tomar notas adicionales sobre la muestra, como la ubicación exacta o las condiciones climáticas.
3. Pantalla de análisis: Esta pantalla debe permitirme analizar los datos de mis muestras de insectos. Debe haber una tabla que muestre el recuento total de cada especie de insecto que he registrado, y también debe haber gráficos y visualizaciones para ayudarme a comprender mejor los patrones de distribución de diferentes especies en diferentes hábitats. También debe haber opciones para filtrar y ordenar los datos de diferentes maneras para obtener información más específica.

En resumen, la aplicación debe tener una interfaz fácil de usar con varias funciones útiles que me permitan llevar un registro detallado de mis muestras de insectos y contarlos de manera eficiente. Debe tener una funcionalidad de búsqueda, capacidad para tomar notas adicionales y analizar los datos.

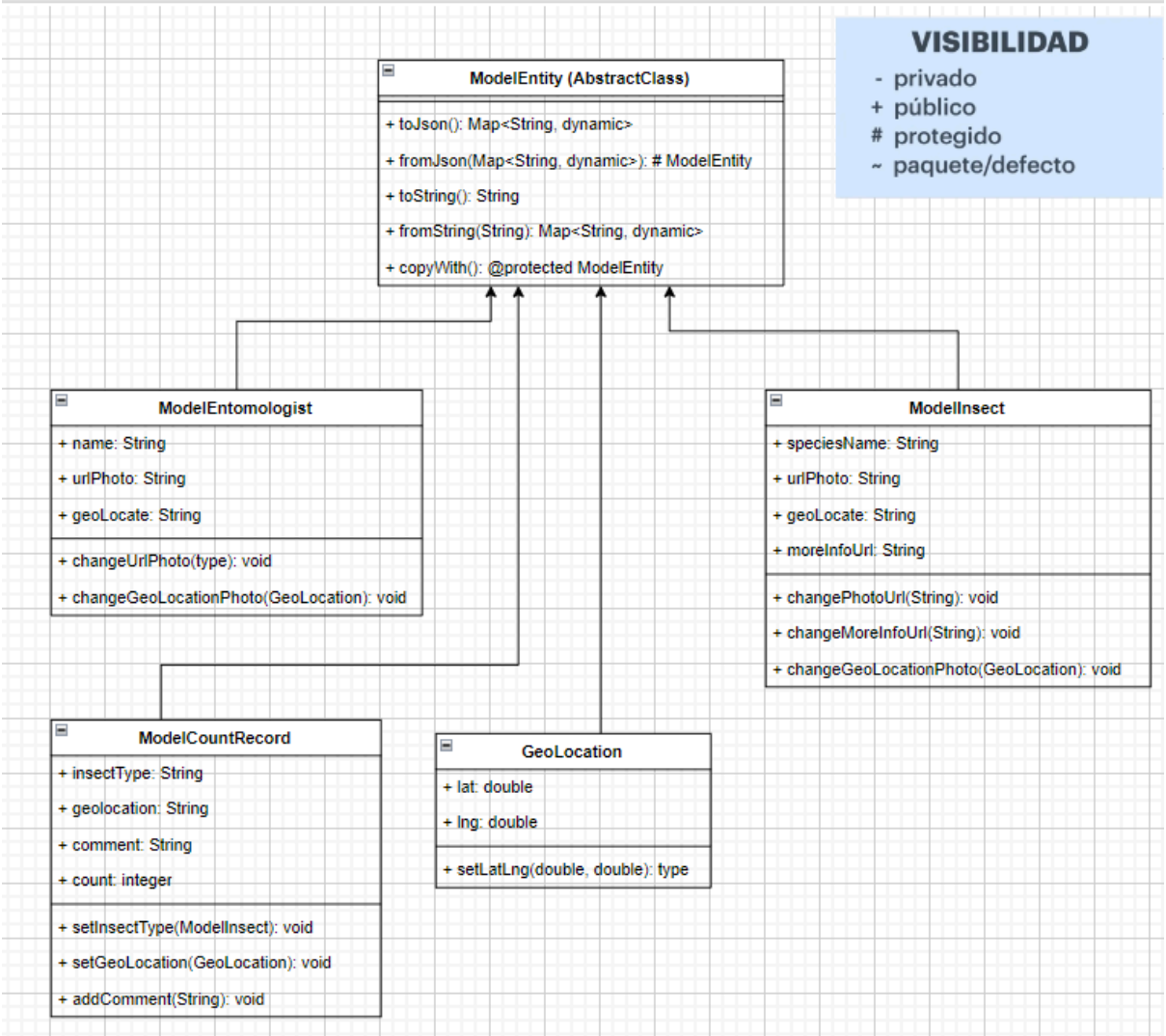
Requerimientos Visuales

De acuerdo a la solicitud del diseñador de UX UI la propuesta a programar es la siguiente : [Figma entomólogo - Propuesta visual]

Requerimientos técnicos

La aplicación debe ajustarse a los siguientes parámetros:

1. Debe respetar el responsive design y en caso de suministrarse un diseño este debe reflejar la solicitud
2. Según el diagrama de clases UML[1] suministrado, la aplicación debe contener mínimo las siguientes clases ([Diagrama de clases reto entomólogo])



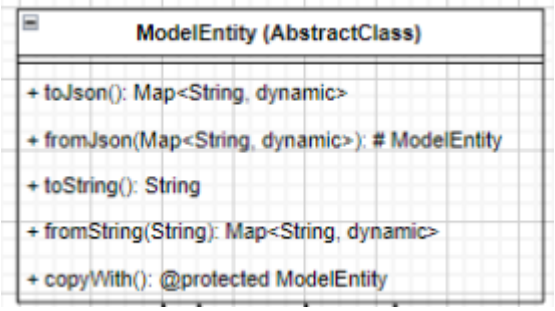
[1] Tutorial de diagrama de clases UML | Lucidchart.

3. MOBILE DREAMERS, de acuerdo al reto descrito realizar el diagrama de casos de uso, apoyarse en el siguiente enlace UML Use Case Diagram Tutorial

Anexos

Una clase abstracta en programación es una clase que no puede ser instanciada directamente, sino que debe ser extendida o heredada por otra clase. En este caso, la clase abstracta "ModelEntity" será una clase base para otras clases que contienen entidades o modelos de datos.

A continuación se presenta un diagrama UML para la clase abstracta "ModelEntity":



1. La función toJson() devuelve un Map<String, dynamic>, que es una representación en formato JSON de los datos almacenados en el objeto. Esta función se utiliza para convertir un objeto en formato de modelo de datos a JSON.
2. La función toString() devuelve una cadena de texto en formato JSON, que se genera a partir del resultado de la función toJson(). Este método es útil para imprimir el objeto de forma legible para los humanos.
3. La función fromJson() toma un Map<String, dynamic> como parámetro y devuelve un objeto del tipo ModelEntity o derivado. Esta función se utiliza para convertir un objeto JSON en un objeto de modelo de datos.
4. La función fromString() recibe una cadena de texto en formato JSON y devuelve un Map<String, dynamic>. Este método decodifica la cadena de texto en formato JSON y devuelve los datos almacenados en ella como un objeto Map<String, dynamic>.
5. La función copyWith() devuelve un objeto de tipo ModelEntity o derivado. Esta función se utiliza para crear una copia del objeto original con algunos de sus parámetros actualizados. Este método es útil para crear objetos de modelo de datos actualizados a partir de objetos existentes.

En resumen, la clase abstracta ModelEntity define una serie de funciones comunes para todas las clases de modelo de datos que se derivan de ella. Estas funciones proporcionan una forma estandarizada de convertir los datos del modelo de datos a JSON y viceversa, así como de crear copias actualizadas de los objetos existentes.