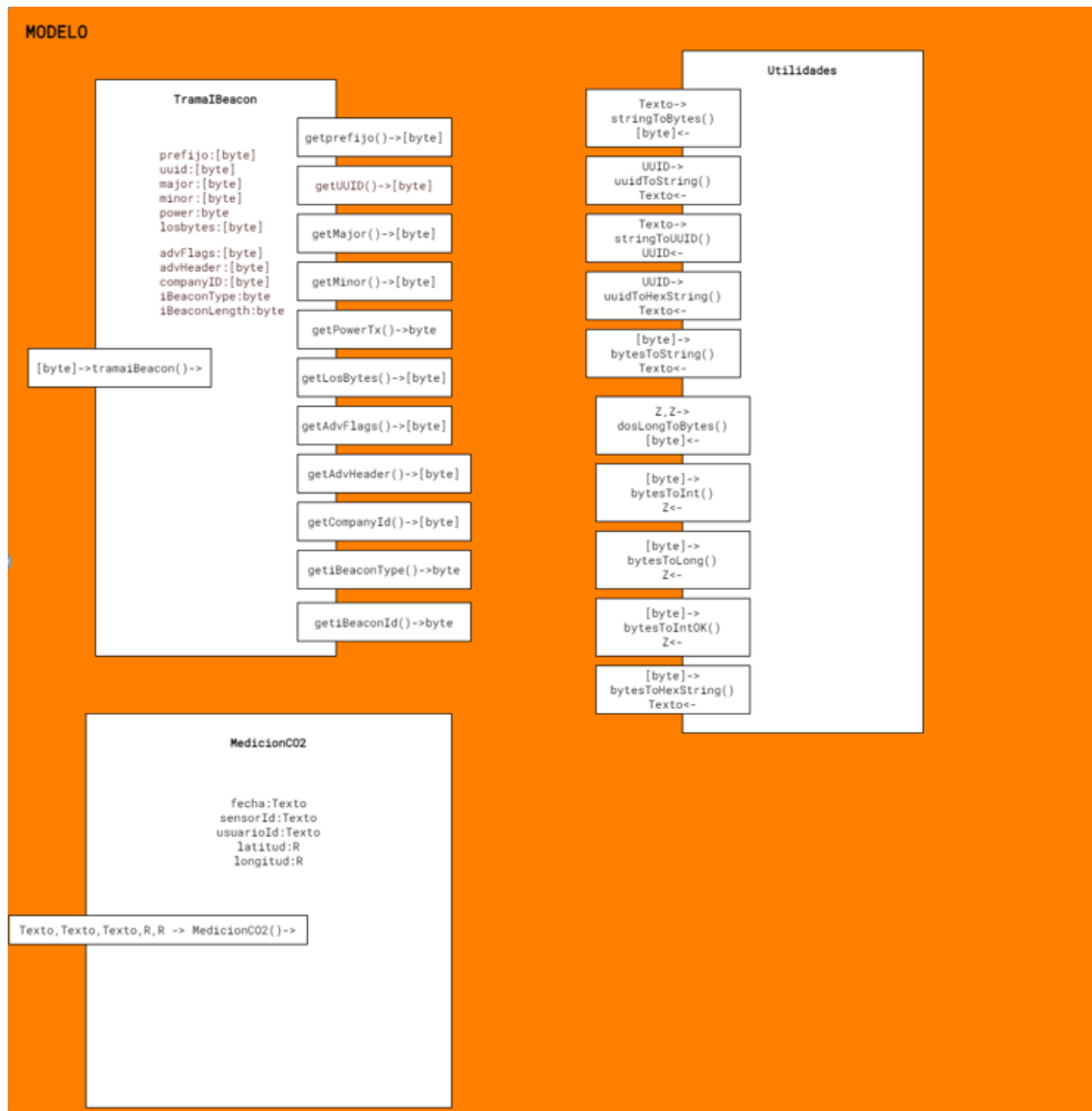


Documento de diseño Android Studio

Jose Julio Peñaranda Jara

A continuación, se muestra el diagrama general del código en Android Studio, así como su comunicación con el servidor.

Empezaremos mostrando el diseño del modelo con una clase para la trama beacon con sus getters y setters, una clase destinada a utilidades y una clase para recoger los datos desde el backend.



O diagrama ilustra a arquitetura de software, dividida em duas camadas principais: MainActivity e Logica.

MainActivity:

- Metodo:**
 - `onStart()`: Inicializa a interface.
 - `buscarDados()`: Busca dados no serviço.
 - `atualizarLista()`: Atualiza a lista de pedidos.
- Componentes:**
 - Servico de Pedidos:** Interface para a lógica de negócios.
 - Pedido:** Classe de dados que representa um pedido.

Logica:

- LogicNegocio:** Classe que gerencia a lógica de negócios.
- PedidoNegocio:** Classe que representa o pedido.

As dependências e chamadas de métodos são as seguintes:

- MainActivity** depende de **Servico de Pedidos** e **Pedido**.
- MainActivity** chama `onStart()` no **Servico de Pedidos**.
- MainActivity** chama `buscarDados()` no **Servico de Pedidos**.
- MainActivity** chama `atualizarLista()` no **Servico de Pedidos**.
- Servico de Pedidos** chama `buscarDados()` no **Pedido**.
- Servico de Pedidos** chama `atualizarLista()` no **Pedido**.
- LogicNegocio** depende de **PedidoNegocio**.
- LogicNegocio** chama `buscarDados()` no **PedidoNegocio**.
- LogicNegocio** chama `atualizarLista()` no **PedidoNegocio**.
- PedidoNegocio** depende de **Pedido**.
- PedidoNegocio** chama `buscarDados()` no **Pedido**.
- PedidoNegocio** chama `atualizarLista()` no **Pedido**.

```

classDiagram
    class Controller {
        start()
        stop()
        run()
    }
    class Model {
        start()
        stop()
        run()
    }
    class Logic {
        start()
        stop()
        run()
    }
    Controller --> Model
    Controller --> Logic
    Model --> Logic
  
```