Go4Me

5. Modelado y patrones

Alejandro Megías Mata,
Alejandro Garau Madrigal,
Jesús Parejo Aliaga,
Raúl Morales Perujo,
Pedro Gallego Vela,
Manuel Veredas Galdeano.

V1.1

Diagrama de clases:

En el diseño del diagrama de clases hemos optado por insertar métodos que consideramos bastante importantes para el funcionamiento del conjunto de clases. Además se han añadido algunos métodos auxiliares para avanzar el trabajo de la implementación de las clases.

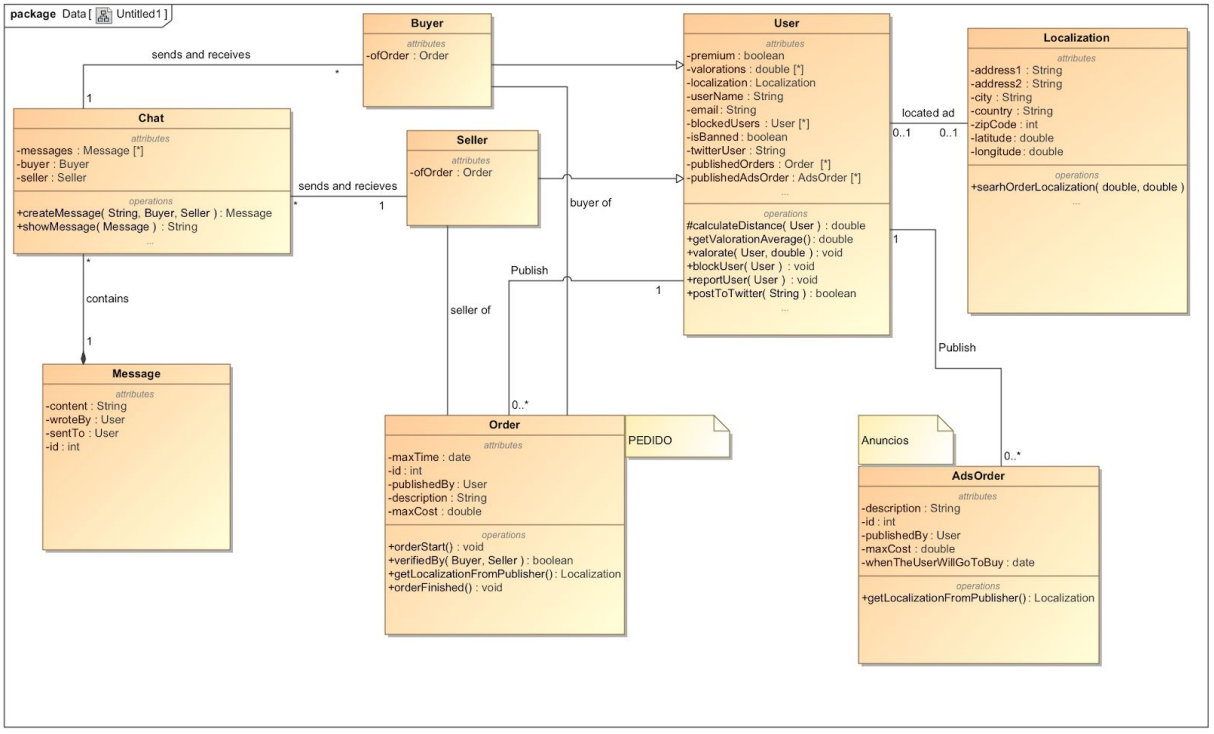

Figura 1. Diagrama de Clases.

Diagrama de secuencia:

La aplicación tendrá un comportamiento similar al diagrama de secuencia, y la interacción entre todas las clases a lo largo del tiempo.

La clase usuario tiene una relación con la clase Localización debido a que continuamente se necesita acceso a esta para filtrar las búsquedas y en el desarrollo del recado, también tiene relación con el chat, el anuncio y el pedido ya que cuando el usuario acepta un anuncio o le aceptan un pedido(depende de cada caso el usuario pasará a ser vendedor o comprador respectivamente) , se abre un chat para negociar las condiciones del mismo con el fin de que ambas partes estén lo más conforme posible.

Por último la clase chat tiene relación con la clase mensaje porque el chat se compone de ellos y según el usuario que lo escriba aparecerá en un lado u otro del chat.

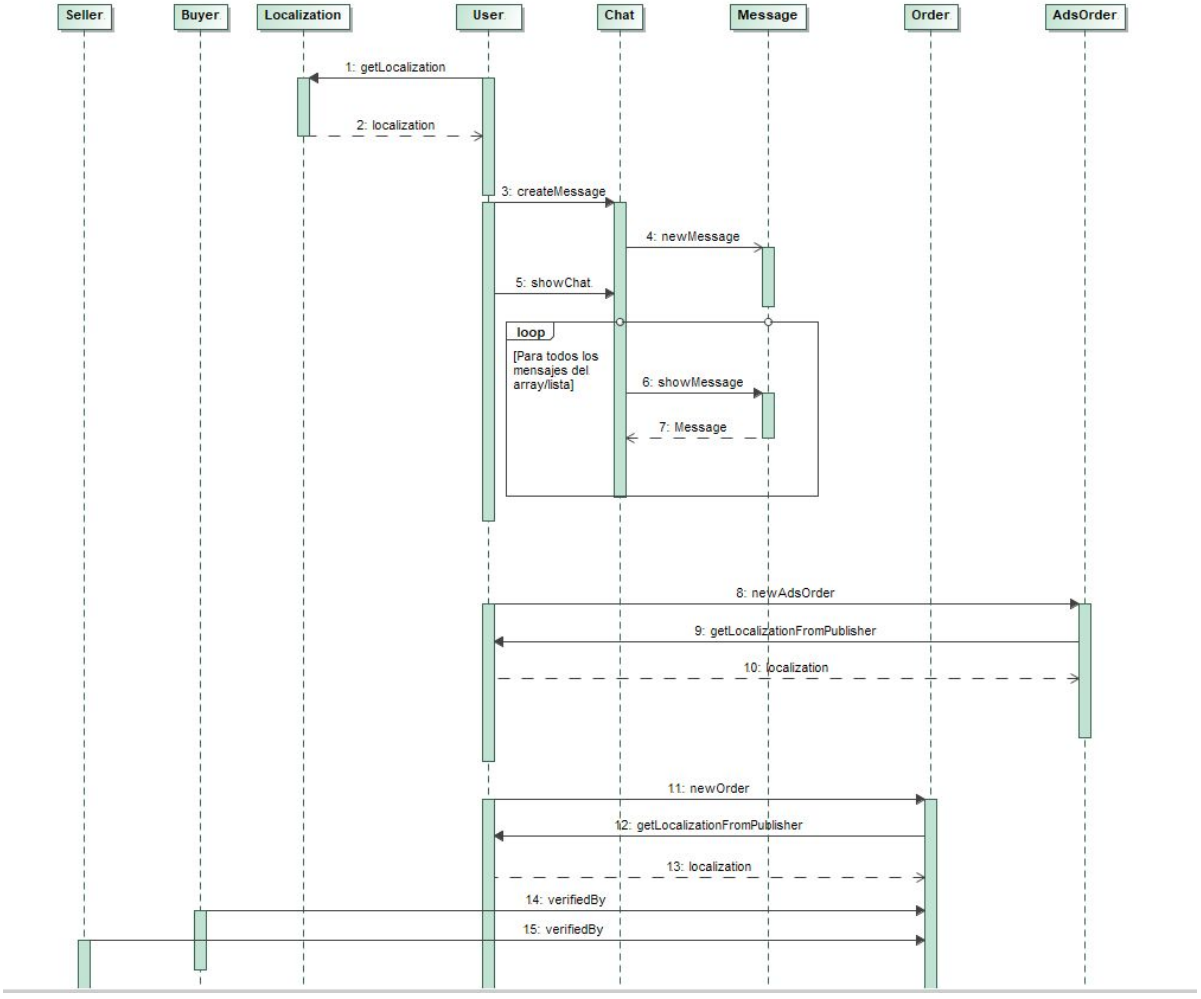

Figura 2. Diagrama de secuencia

Patrones usados:

Para el proyecto de Go4Me, ya que se decidió usar *Spring*, usaremos el patrón de **Modelo Vista-Controlador.**

Algunas de las ventajas que incluye este modelo vista controlador, es la implementación, que se realiza de forma modular; o sus vistas, las cuales muestran la información siempre actualizada. De la misma forma, las modificaciones de estas no afectan al modelo del dominio. Está demostrado que es un patrón de diseño bien elaborado, pues las aplicaciones que lo implementan muestran una extensibilidad y mantenibilidad únicas, comparadas con otras las cuales no lo usan.

Para la vista utilizaremos *Thymeleaf*, que es es un motor de plantillas de Java, en otras palabras, nos proporciona plantillas para crear vistas en HTML. En cuanto al modelo, se usará *Spring,* que nos permitirá realizar el modelo y el controlador de cada una de las clases modeladas. En el caso de que tuviéramos que cambiar la vista de una clase, símplemente se tendría que modificar el HTML que nos proporciona *Thymeleaf*, sin necesidad de modificar ninguna línea de código del *backend.*

A medida que avance el proyecto, se irán añadiendo los patrones de diseño que se utilicen para la implementación del mismo.

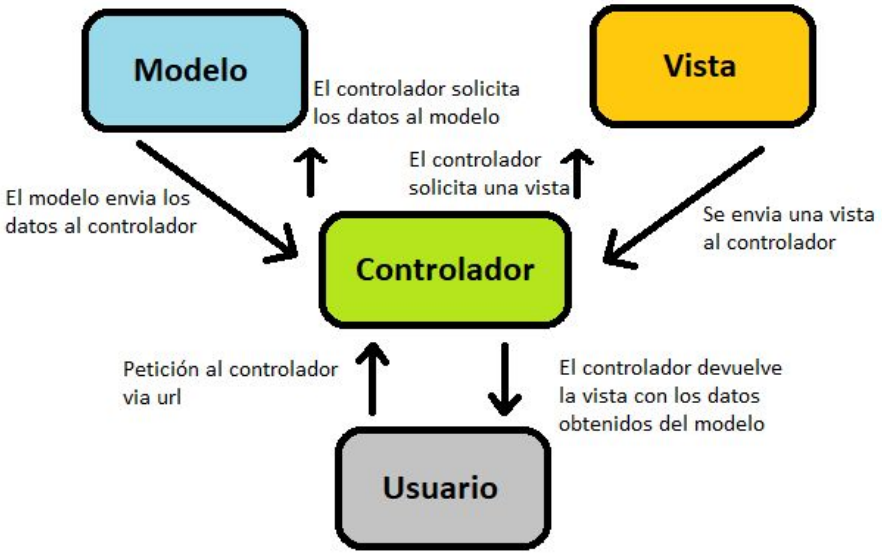

Figura 3. Esquema del Modelo Vista Controlador