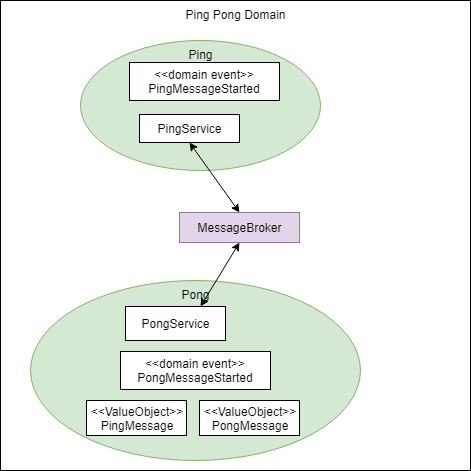
**DDD**

Tendré dos contextos delimitados para separar la recepción de los mensajes desde dos aplicaciones diferentes, ping y pong. Estas dos aplicaciones pueden comunicarse entre ellas bidireccionalmente a través de un OHS, para este caso es un message bróker.

La aplicación ping contará con un servicio de dominio el cual recibirá un mensaje ping\_message, al recibir este mensaje desencadena un evento que notifique que recibió el comando que luego será trasmitido hacia la aplicación pong.

La aplicación pong contará con dos objetos de valor uno que llamaremos PingMessage y el otro PongMessage, los cuales se contendrán en un repositorio con el fin de contar el número de mensajes que entran y el número de mensajes que responde, los mensajes de respuesta se harán por medio de un evento que se lanza luego de dos segundos de espera.

La interacción con esta aplicación se hará con un servicio de domino, este servicio de dominio tendrá dos operaciones, la primera será recibir mensajes ping, llevarlos al repositorio y responder mensajes pong que también serán agregados al repositorio. La segunda es consultar las estadísticas de los mensajes que ha recibido y que ha enviado.

**Arquitectura**

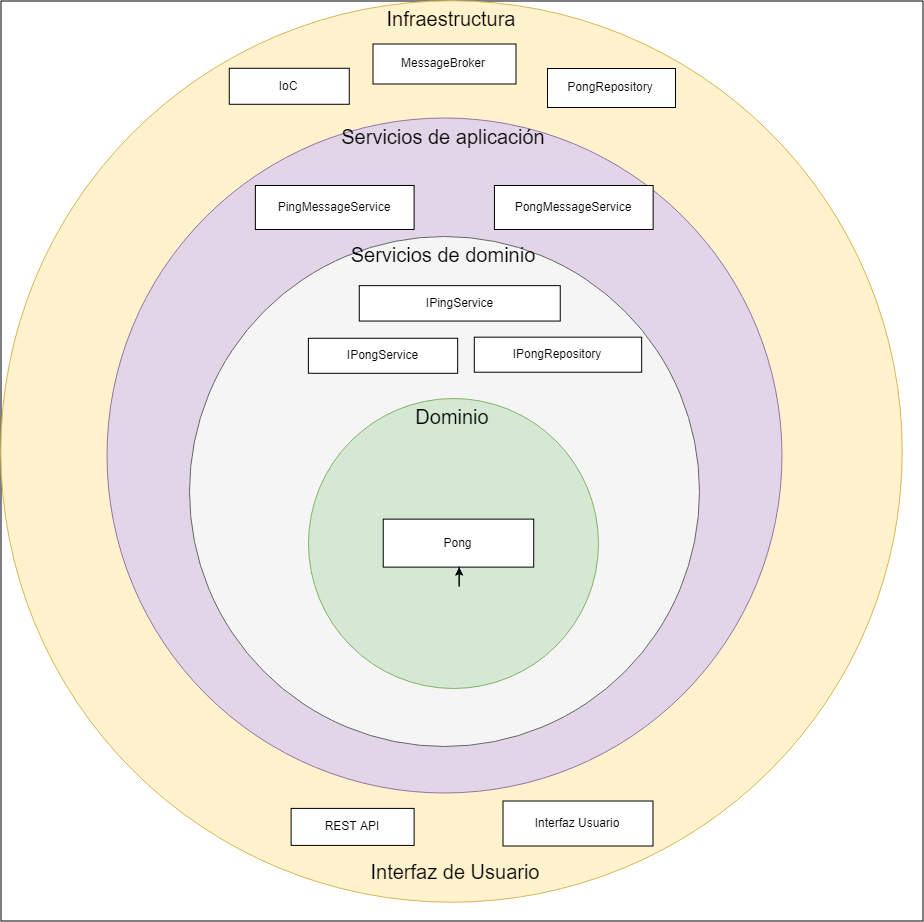
Utilizaré un estilo arquitectónico “cebolla” según su definición esta arquitectura que se enfoca en el dominio y es un buen complemento para trabajar con DDD.

Esta arquitectura controla el acoplamiento ya que su regla general, todo el código puede depender de capas centrales, pero no puede depender de capas superiores. En otras palabras, el acoplamiento se dirige hacia el centro, esta dirección de las dependencias entre capas es la principal diferencia entre una arquitectura de n-capas tradicional y la arquitectura cebolla.

En la arquitectura cebolla aplicamos el principio de inversión de dependencias. Los detalles de la implementación (infraestructura) depende de las abstracciones (dominio).

La siguiente lista describe el propósito de esta arquitectura:

* **Dominio:**  aquí tendré los objetos de valor que fueron definidos en el DDD, con sus servicios de dominio y las interfaces necesarias para hacer la implementación en la capa superior.
* **Servicios de Aplicación:** aquí estará implementada la lógica para coordinar el paso de mensajes entre las aplicaciones y la invocación a los servicios de dominio.
* **Infraestructura:** aquí estará la implementación de las interfaces de los repositorios del dominio, además de la comunicación del message bróker y el contenedor de inversión de control para inyectar dependencias a las capas superiores.
* **Interfaz de usuario:** estará la aplicación web que permitirá la interacción del usuario para enviar mensajes ping, también estarán los REST api que permiten la comunicación por medio del protocolo http.



**Tecnología**

* .Net framework con lenguaje C#
* El message bróker – RabbitMQ