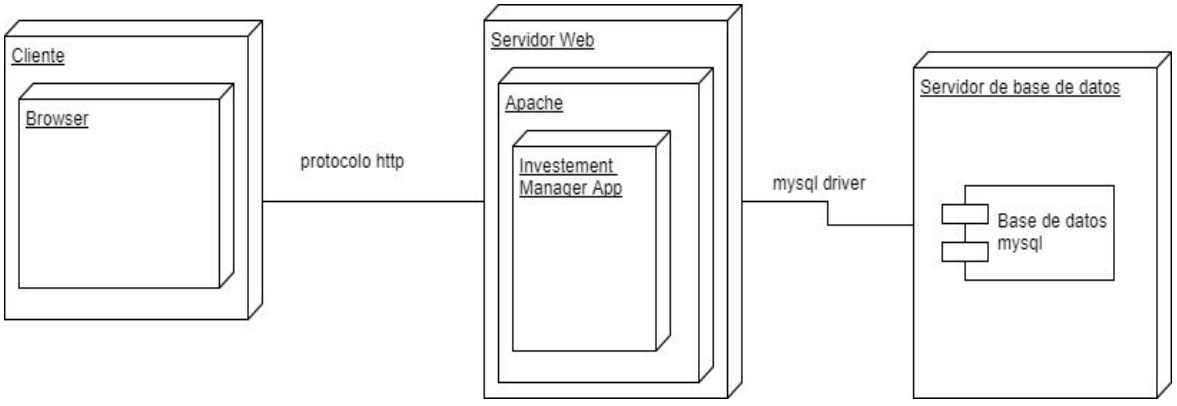
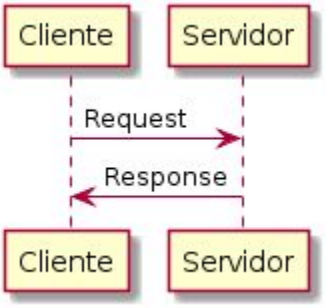
# Entrega 5

La aplicación Investment Manager requiere de un servidor en donde pueda ser alocada para su acceso de forma pública a través de otros dispositivos (Computadora Personal, Smartphone). La misma se ejecutará mediante el software Apache.
Al mismo tiempo requiere de un servidor  para poder alocar la base de datos correspondiente a la aplicación y que se utiliza para persistir los datos de la misma.

A continuación presentamos un diagrama de alto nivel mostrando las diferentes partes implicadas:

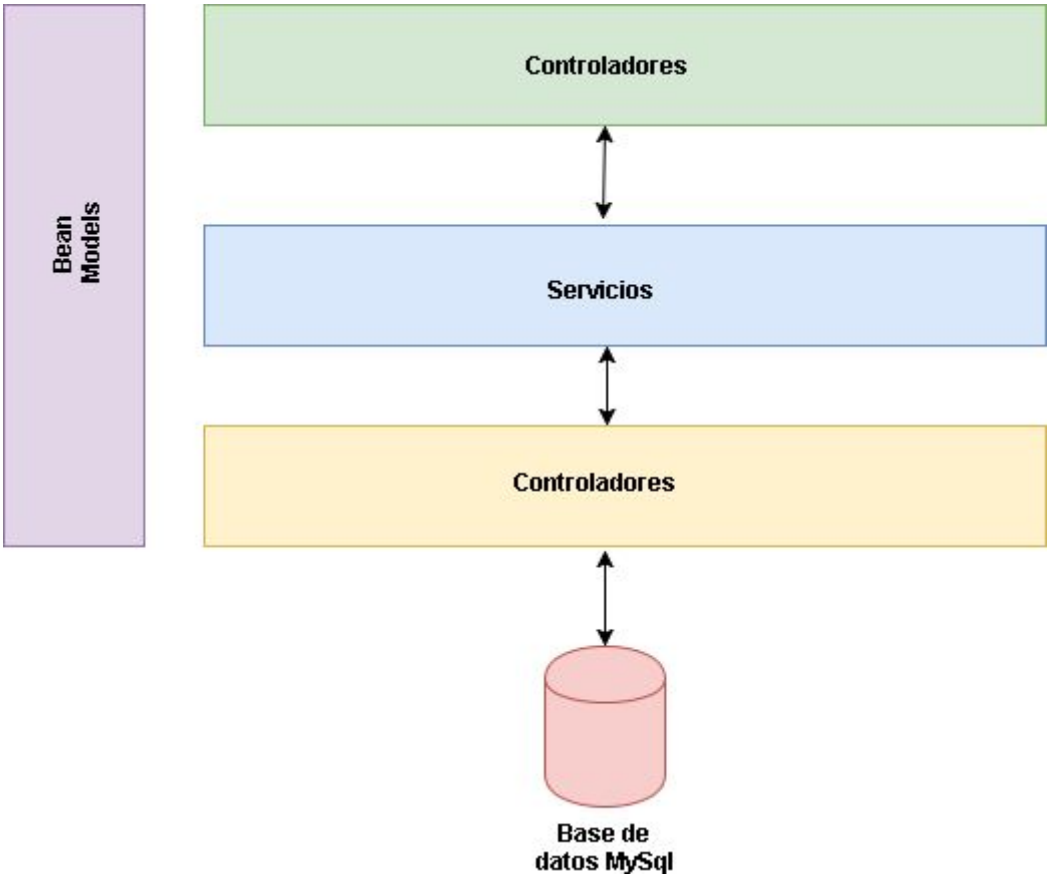



Cada usuario podrá acceder a la aplicación mediante el uso de un navegador web. De acuerdo a esta premisa el protocolo utilizado para poder establecer una comunicación entre cliente/servidor será HTTPs.
Este protocolo de comunicación permite las transferencias de información en la World Wide Web. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de sesión, y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.



**Componentes lógicos**

La aplicación está estructurada en capas cuyo propósito es otorgar una correcta división de responsabilidades y aislamiento entre los distintos componentes:




Controladores:
En esta capa se definen los componentes cuyo propósito es brindar respuesta antes las distintas solicitudes realizadas por los clientes. Los controladores se encargan de obtener la información necesaria para responder a una solicitud y de definir cuál será la vista encargada de mostrar la misma. A su vez funcionan como un punto de entrada para las solicitudes que persisten información en la aplicación.

Servicios:
Los servicios abstraen cualquier lógica de negocio que pudiera llegar a existir en la aplicación.
Actúan como un intermediario entre los controladores y los repositorios

Repositorios:
Esta capa es la responsable de gestionar la persistencia y consulta de datos en la base.

**Base de datos**

El motor de base de datos relacional utilizado es MySQL. El mismo provee  funciones para el almacenamiento y protección de los datos de la aplicación. A su vez permite que varios usuarios puedan acceder a la base de datos de forma concurrente.

**Indicar además qué acciones son necesarias para escalar horizontalmente a la nueva arquitectura.**

Para poder determinar las oportunidad que tiene un sistema para ser escalable horizontalmente primero tenemos que tener en claro el significado de este concepto.

Escalabilidad: Es la propiedad de un sistema, que indica su habilidad para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

Por lo tanto la escalabilidad es una propiedad que deseablemente puede tener el sistema, por lo tanto podríamos decir que es un atributo propio de un sistema de calidad. Al mismo tiempo es una cualidad que debe definirse para poder saber cuan creciente será el volumen de carga de un sistema y cuán relevante es que este sea escalable.

La escalabilidad está fuertemente ligada a los siguientes conceptos:

- Sistema de calidad (es un atributo propio de una buena arquitectura)
- Buen diseño (muchos sistemas son re-diseñados por poca escalabilidad)
- Diseños arquitectónicos específicos:
    **a)** Arquitectura de capas (proporción buen rendimiento)
    **b)** Arquitectura Peer - to - Peer
    **c)** Arquitectura orientada a servicios (utilizando mensajería asincrónica)
    **d)** Arquitectura de microservicios (utilizando mensajería asincrónica)

Por último tenemos que saber que la escalabilidad horizontal está fuertemente ligada al hardware, se basa en aumentar el número de elementos que realizan una determinada tarea. Por ejemplo en una arquitectura Cliente-Servidor deberíamos agregar el número de servidores y conectarlos entre sí para trabajar como un todo.

Ventajas:

- El crecimiento es prácticamente infinito, podríamos agregar cuantos componentes sean necesarios
- Es posible combinarse con el escalamiento vertical.
- Soporta la alta disponibilidad
- Si un componente falla (en caso de tener una correcta infraestructura), los demás sigue trabajando.
- Soporta el balanceo de cargas.
-

Desventajas:

- Requiere de mucho mantenimiento
- Es difícil de configurar
- Requiere de grandes cambios en las aplicaciones (si no fue diseñado así en un principio)
- Requiere de una infraestructura más grande.

Para concluir podemos observar que lograr la correcta escalabilidad de un sistema no es una tarea fácil, depende de muchos factores, de todas formas a continuación resumimos cuáles creemos que son las acciones claves a realizar para poder garantizar que nuestro sistema tenga la escalabilidad necesaria:

1) **Determinar el grado de escalamiento:** Es importante poder determinar desde un principio cuan creciente será el volumen de carga de un sistema. En base a esto definiremos cuándo podrá soportar la aplicación, dado que las mismas no podrán crecen indefinidamente.

2) **Determinar una correcta estrategia:** Es importante que al momento de diseñar nuestro sistema tengamos en cuenta el grado de escalamiento y adoptemos una estrategia para contener y mitigar el posible crecimiento de nuestra aplicación. Un mal diseño será difícilmente escalable.

3) **Determinar una arquitectura adecuada:** Es importante seleccionar de manera adecuada un diseño arquitectónico específico que nos permita escalar nuestra solución. Hay diseños arquitectónicos que dificultan la adición de nuevos hardwares o simplemente no están preparados para un volumen tan alto de carga.

4) **Determinar algoritmos dinámicos:** Es importante al momento de desarrollar la solución tener en mente el posible crecimiento de la aplicación. Específicamente para una escalabilidad horizontal necesitamos poseer un software que nos permita la integración de hardware y su conexión para el trabajo en conjunto.

5) **Determinar una infraestructura correcta:** A medida que una aplicación crece se le va integrando más hardware y es muy común que se pierda la estructura principal con la cual fue pensada (eso concluye en un funcionamiento no deseado de los componentes). Es importante tener en mente cuál es la infraestructura deseada y adecuada para que todos los componentes de la misma trabajen de forma adecuada y pueda aprovecharse el mayor rendimiento de cada una.

6) **Realizar un correcto seguimiento y mantenimiento:** A medida que un sistema crece es necesario realizar un seguimiento de los algoritmos para asegurarnos de que estos estén trabajando de forma óptima, correcta y no hayan sido afectados. Además es necesario realizar el correcto mantenimiento de todos los componentes para que se encuentre en su funcionamiento óptimo (en caso de ser posible). Infraestructuras y arquitecturas más grandes requieren de mayor esfuerzo en seguimiento de software y mantenimiento de hardware. El no realizar esta última tarea adecuadamente puede resultar en un funcionamiento peor del obtenido en un principio.