# Patrones Arquitectónicos a Considerar.

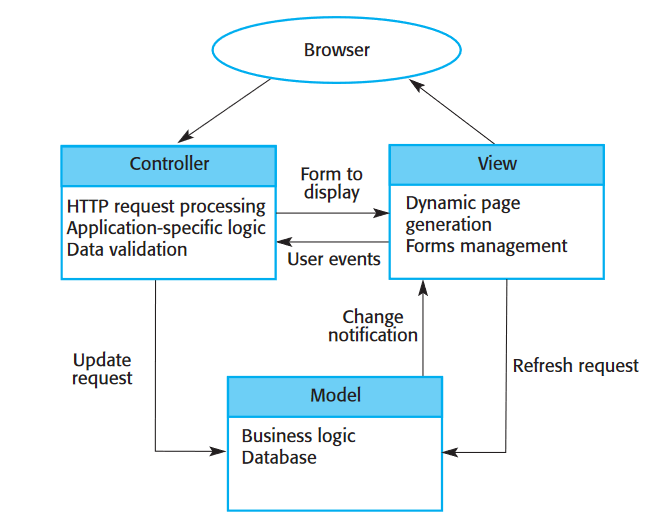
## **MVC.**

El patrón arquitectónico MVC fue el primero en evidenciarse, dado que, la tecnología que utilizamos para este proyecto (RoR) es un framework que se define bajo este paradigma.

En términos generales el Modelo-vista-controlador (MVC) es un patrón de arquitectura de software, que separa la lógica de una aplicación, sus datos, su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

MVC bajo nuestra realidad es el clásico ejemplo de una aplicación WebApp que utiliza este paradigma, más precisamente:

* El Modelo: Maneja todas todo los datos del sistema referido a las Encuestas, Usuarios, Grupos, etc. (RoR)
* El Controlador: Maneja todos los eventos generados a partir de la interacción del usuario con la WebApp. (RoR)
* La Vista: En este caso particular, no usamos RoR, para este proyecto usamos React, es el encargado de proveer las vistas y como los datos son presentados al usuario.



## Cliente / Servidor.

El modelo anterior sirve para generar un contexto sobre el manejo interno de la aplicación. Pero, no hay que obviar el hecho de que nuestra WebApp se integra al cliente como un servicio externo, servicio que será consumido por NeoCoast y sus clientes a través de un navegador. Por eso es necesario considerar el patrón arquitectónico Cliente Servidor.

En una arquitectura cliente/servidor el sistema es presentado como una colección de servicios, con cada servicio siendo potencialmente asociado con un servidor. Los clientes son usuarios de estos servicios y acceden al servidor/es para utilizarlos.

Nuestro sistema, en rasgos generales es presentado al cliente como una colección de servicios de Encuestas que diferentes tipos de usuarios utilizan (por ejemplo, el Administrador crea Encuestas y les programa una agenda, mientras que los usuarios las responden) , que están alojados en la infraestructura provista (Heroku Server).

Se trata entonces de una aplicación multi-cliente que consume un determinado conjunto de servicios concurrentemente. Es necesario estimar la carga de trabajo de cada uno de estos servicios así como la periodicidad para determinar la cantidad de servidores a montar:

* Cuanto es la carga de usuarios.
* Cuanto cuesta en tiempo de cómputo analizar el histórico de las encuestas y el filtrado por parámetros.
* Analizar escenarios de stress (Todos los empleados de neo coast y clientes haciendo encuestas el mismo dia en la misma ventana de tiempo)