**Decisiones de arquitectura.**

Nuestra aplicación en un principio fue hecha sobre una arquitectura monolítica específicamente en JAX-RS. Luego de presentar una serie de problemas y de identificar necesidades que se ajustaban de mejor manera a otro tipo de arquitectura, decidimos migrar la aplicación por completo a Play! Además de esto, Play! Puede ayudar de manera gradual y mayor, a características en la implementación de funcionalidades que de alguna manera ayuden al cliente directo de la aplicación. Por otra parte Play! Puede contribuir de manera exponencial en la medida de obtención de mejoras en atributos y escenarios de calidad tales como la Disponibilidad y el Desempeño.

Para la primer entrega nuestra base de datos no corría junto con el servidor de la aplicación, esto aliviaba completamente la competencia por los recursos y ayudaba de manera efectiva a cada uno de los procesos. Al cambiar la arquitectura y manejar la base de datos de esta manera pudimos evidenciar una notable mejora, en ese momento obtuvimos mejores resultados en cuanto a tiempos de respuesta y latencia.

Por otra parte, una de las tácticas implementadas fue la táctica de introducción de concurrencia. Esta táctica fue implementada con el fin de que se procesaran una mayor cantidad de peticiones mediante la creación de hilos. Con esto, el tiempo de bloqueo por petición se puede reducir, si las mismas se procesan en paralelo.

Para la segunda entrega, debido a la cantidad de requerimientos especificados tantos por los clientes directos de la aplicación, así como por la empresa y sus funcionalidades básicas, tuvimos que encontrar una mejora sustancial en la medida de obtención de respuestas y disponibilidad para el cliente. De esta manera, implementamos una herramienta satisfactoria como un “Balanceador de carga”. Este mismo se encarga directamente de un manejo efectivo de los procesos y funcionalidades de la aplicación, direccionándosela a diferentes tipos de recursos.

En cuanto a la modificabilidad de nuestra aplicación, estudiamos propuestas que promovieran el buen uso de tácticas como la alta cohesión y el bajo acoplamiento. El bajo acoplamiento lo logramos reduciendo el número de relaciones y conocimientos entre las clases. Esto reduce el gran impacto de los cambios que vayamos a realizar a futuro sobre la aplicación. En cuanto a la alta cohesión, lo que hicimos como medida de táctica fue la abstracción de los servicios más importantes de la aplicación.

Para esta tercera entrega, tuvimos en consideración la importancia de los distintos patrones de diseño y procedimos a la implementación de algunos y nos percatamos de otros ya presentes en la aplicación. Entre estos se encuentran el patrón singleton aplicado en el archivo de rutas para poder garantizar que solo exista una instancia del archivo de rutas proporcionando un punto de acceso global a dicho archivo; adapter, al transformar las interfaces de los sensores lo suficiente para conectar los datos provenientes de estos con OilCol, por medio de NodeRed.

Además, tuvimos presente las tácticas para ofrecer una buena usabilidad. En la parte de soporte al usuario nos enfocamos en estructurar las interfaces graficas en función de las tareas de usuario logrando hacer la página muy sencilla e intuitiva de usar, para esto lo que hicimos fue hacer un diseño tanto de la página como de los formularios, para esto se tuvo en cuenta no desperdiciar espacio de la pantalla, evitar el esfuerzo físico y la agrupación de la información adecuadamente. Adicionalmente para la parte de la navegación como tal del usuario se trato de hacer la página sumamente sencilla e informándole al usuario donde se encontraba y que funciones puede realizar. Y en cuanto al soporte al desarrollador lo principal fue el uso de hojas de estilo que nos ayudaron con la estética y la navegabilidad de la página.