***Tabla de decisiones de diseño***

| **Fecha** | **Decisión** | **Ventaja** | **Desventaja** | **Alternativa** |
| --- | --- | --- | --- | --- |
| 22/08/2018 | Se creo una clase zona y otra transformador las cuales una posee una colección de trasformadores y otra que posee una colección de los clientes que están conectados a él, | Se desacopla la obtención del consumo ya que al preguntarle a la zona su consumo esta le pregunta a su lista de transformadores su consumo, los cuales le pregunta a su lista de clientes su consumo, el cual le pregunta a sus dispositivos el consumo, entonces si en un futuro se desea modificar la forma de consumo solo hay que tocar el método de consumo de los dispositivos |  |  |
| 22/08/2018 | se utilizo la api de Google geocoding la cual pasándole una dirección obtenemos su latitud y longitud | No debemos saber la latitud y longitud y un cliente, ya que es un dato que no se sabe fácilmente al momento de instanciar un nuevo cliente, entonces delegamos en la api la obtención de los datos, y posteriormente seteado de dichos atributos cuando se importan desde el json |  |  |
| 22/08/2018 | Al momento de importar los clientes desde el json se implementó un método que le asigna el transformador más cercano | Al poseer las coordenadas del posicionamiento de los trasformadores y de los clientes el repositorio automáticamente cuando importa los clientes se fija cual es el transformador mas cercano calculando la distancia con todos los transformadores se fija cual es el más cercano y asigna al transformador dicho cliente, esto tiene la ventaja de con solo cargar desde los clientes el domicilio automáticamente le asigna el transformador, y no me tengo que preocupar por asignarle a mano ningún transformador |  |  |