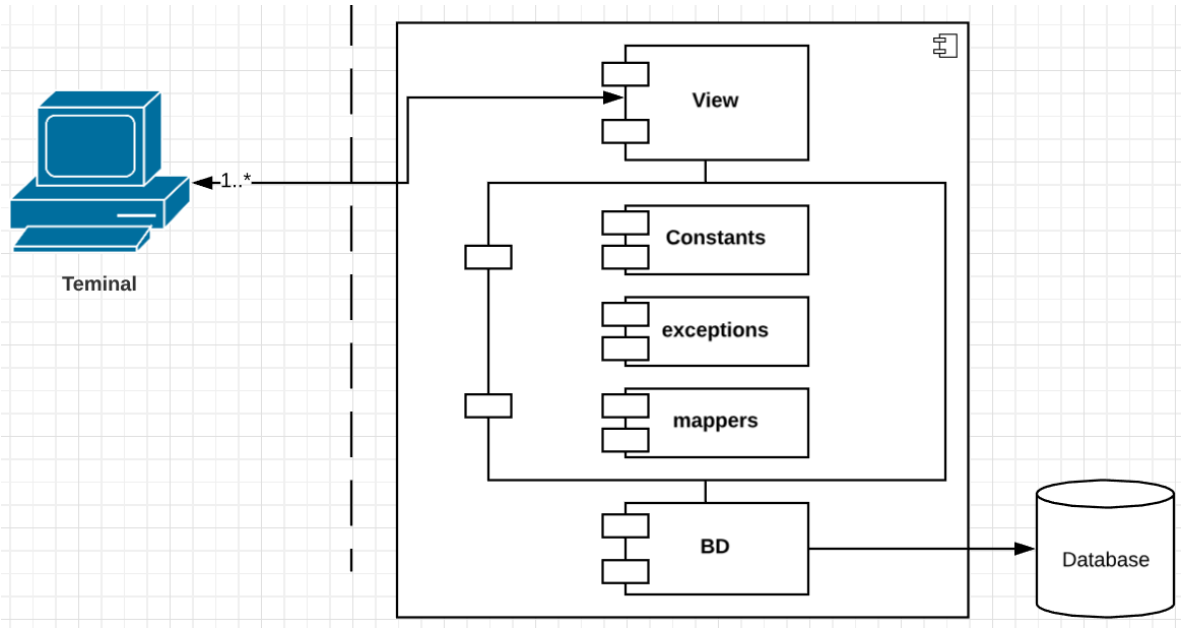
DECISIONES DE DISEÑO TP SISTEMA DE GESTIÓN ENERGÉTICA ENTREGA 0

Contenido

Diagramas

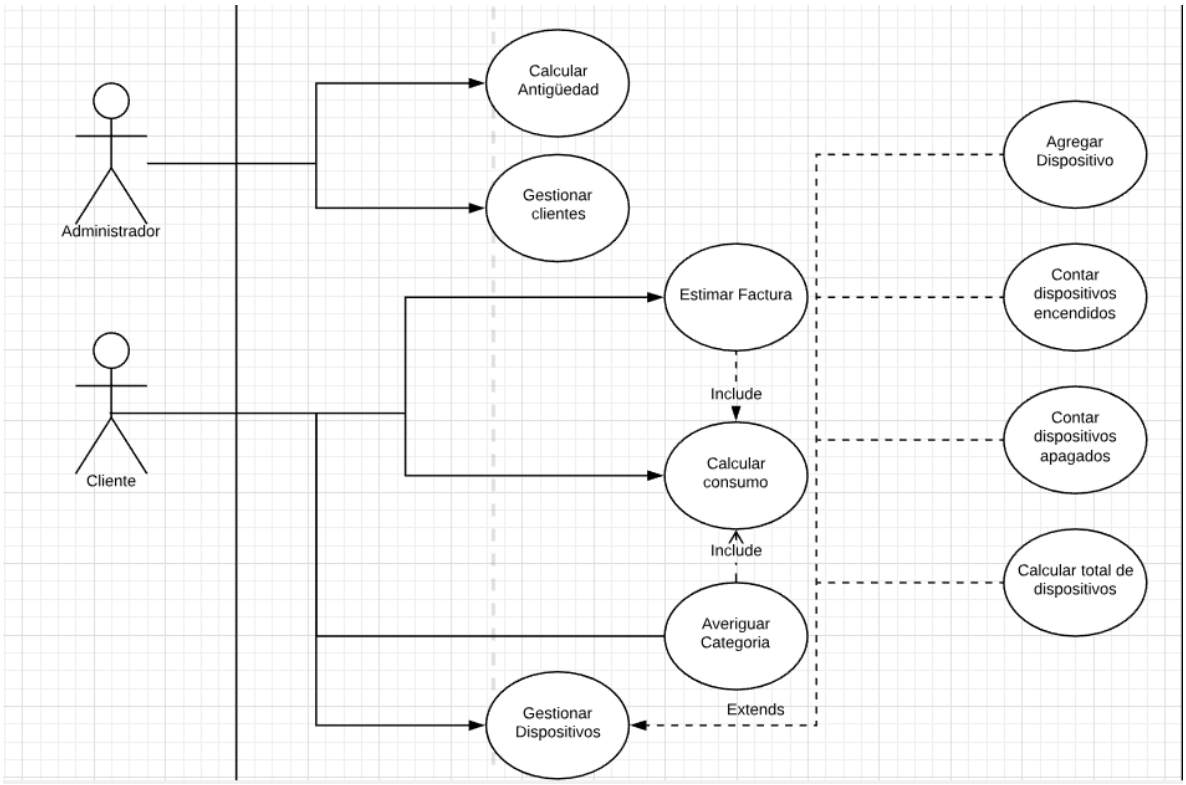
Diagrama de Arquitectura



En la arquitectura se detalla que los usuarios podrán acceder al sistema por medio de un terminal. Se accederá a una interfaz de presentación “View”. El sistema tiene una capa de con la lógica de negocio que incluye Constants, exceptions y mappers. Se toma en cuenta que en entregas futuras se buscará persistir los datos transaccionales en una base de datos.

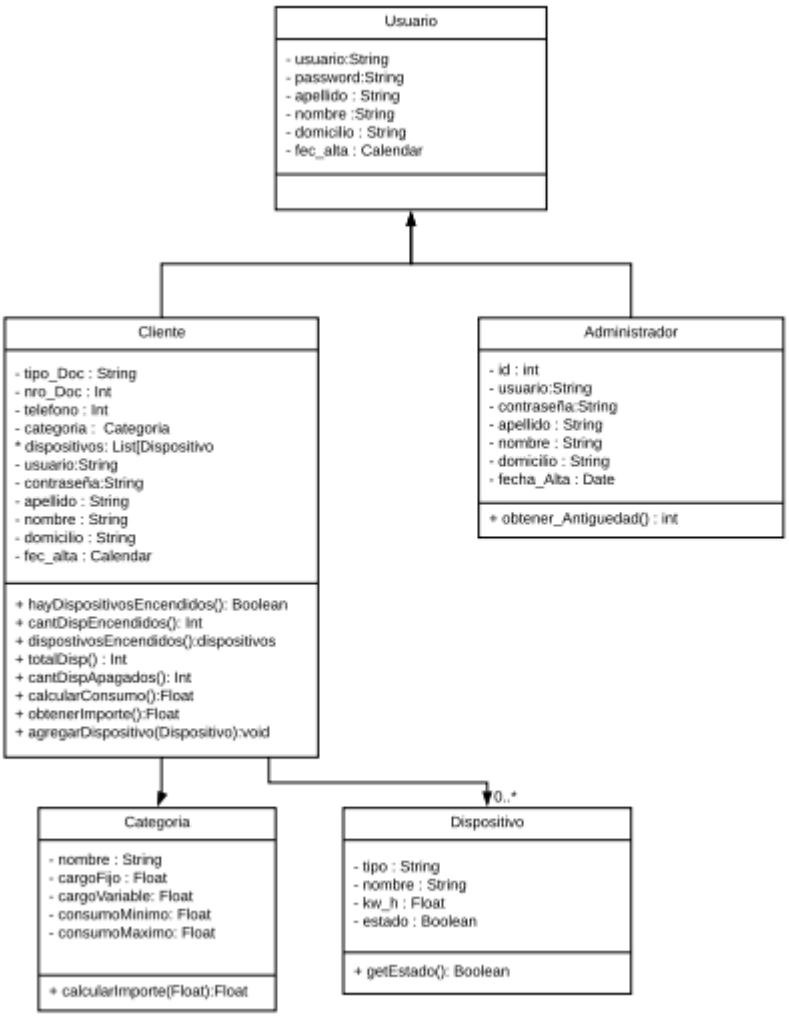
- El usuario del sistema se conectará con el sistema por medio de un dispositivo (celular, ordenador, etc), a una capa de presentación (GUI) donde verá los elementos necesarios para él. Detrás de la capa gráfica, estará el detalle de las reglas de negocio, donde se realizará la operativa del sistema. Luego se buscará que los datos del sistema se guarden en el tiempo, en una base de datos, que puede ser o no relacional.
- Se planea que la interacción entre los dispositivos y el sistema sea mediante un protocolo web.

Diagrama de Casos de Uso



El Caso de Uso “Gestionar Clientes”, no se implementa en esta entrega, se deja el caso de uso para que se entienda el rol del administrador.

Diagrama de clases



En el diagrama se omitieron agregar los métodos de getters y setters de los atributos de cada clase.

Se decidió crear una clase abstracta Usuario, de la cual heredan Cliente y Administrador.

Requerimientos no funcionales

Código	Requerimiento	Descripción	Atributo de calidad afectado	Métricas
R1	El software es tolerante a fallos.	El software es capaz de recuperarse ante fallas. Ofrece las funcionalidades con la capacidad de brindar confiabilidad al usuario en caso de fallas y excepciones.	Confiabilidad	Aceptable: Sistema captura el 100% de las excepciones mostrando mensaje amigable No Aceptable: Sistema no captura alguna excepción
R2	El software realiza todos los requerimientos funcionales	El software tiene las características y funcionalidades que se especifican en los requerimientos funcionales de la entrega 0 y con un 100% de testings unitarios en verde (sin fallas).	Funcionalidad	Aceptable: Sistema cumple con el 100% de lo solicitado en el Doc de la entrega 0 y si hay un porcentaje de incumplimiento debe estar justificado y corroborado por el ayudante No aceptable: AL sistema le falta una funcionalidad que solicita el Doc de la entrega 0
R3	El software es mantenible.	El software puede ser ampliado y adaptado con facilidad. Además sus módulos presentan bajo acoplamiento, lo que permite que sea sencilla su modificación sin afectar a los demás y pueden ser probados.	Flexibilidad, Mantenibilidad	
R4	El sistema es seguro.	Los permisos de acceso al sistema se manejarán con un usuario y contraseña. Los administradores sólo podrán cambiar información de configuración. Las claves deben tener un mínimo de 5 caracteres que incluyan letras y números.	Seguridad	Aceptable: El sistema debe resistir el 80% de los ataques de diccionarios(brute force) , más adelante se realizara un hardening al ingreso de usuarios.
R5	El sistema es escalable	Se espera que el sistema tenga el alcance de clientes	Escalabilidad	Aceptable: el sistema soporta la cantidad

		residenciales, teniendo en cuenta que existen otros tipos.		de clientes residenciales de la ciudad, pero se espera y debería soportar clientes comerciales y fabriles

En la tabla se describieron los requerimientos que consideramos que aplican para la entrega 0.

No se tienen en cuenta los requerimientos relacionados con interfaz gráfica, usabilidad, eficiencia, ya que no están en el alcance de la entrega.

Decisiones de Diseño

Fecha	Decisión	Ventaja	Desventaja	Alternativa
20 de Abril 2018	Se decidió que se creará una clase abstracta Usuario para vincular Cliente y Administrador por herencia, por detectar que tendrán comportamientos similares.	Se evita repetición de código.	Hay acoplamiento entre clases	Crear las clases sin vincular por herencia.
4 de Abril 2018	Se decide utilizar la biblioteca Calendar para el manejo de fechas.	Permite realizar la funcionalidad de diferencia de fechas con un grado de dificultad bajo	Se deben parsear las fechas.	Usar Joda Time
2 de Abril 2018	Se decidio agregar bilbioteca de Jackson para utilizar la clase Object Mapper para parsear los json y crear los objetos en memoria.	Simplifica el proceso de mapeo abstrayendose de la implementación	Al ser automatico el mappeo, no se puede acceder a la implementación y realizar ajustes en este proceso	Hacerlo manual
4 de Abril 2019	Se dividen las tareas entre los integrantes del grupo:			
3 de Abril 2018	Se decide utilizar JUnit para realizar los testings			
2 de Abril 2018	Se decide usar Json para la importación de datos			
1 de Abril	Se decide utilizar la tecnología Java	El lenguaje es amiga y familiar para la mayoría del equipo		