DECISIONES DE DISEÑO TP SISTEMA DE GESTIÓN ENERGÉTICA

ENTREGA 0

Contenido

[Diagramas 3](#_Toc512192019)

[Diagrama de Arquitectura 3](#_Toc512192020)

[Diagrama de Casos de Uso 4](#_Toc512192021)

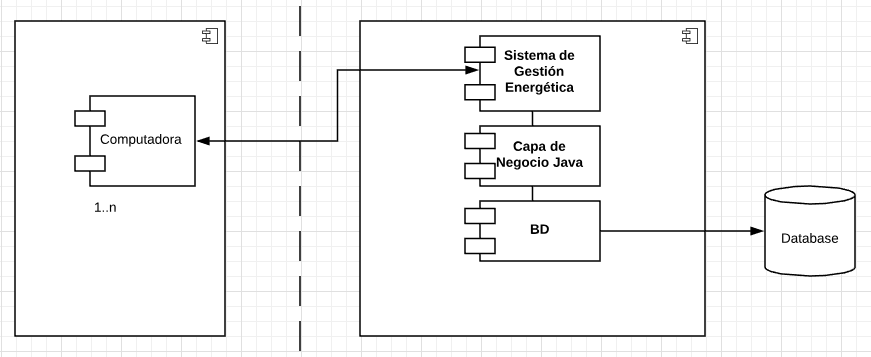
[Diagrama de clases 5](#_Toc512192022)

[Requerimientos no funcionales 7](#_Toc512192023)

[Decisiones de Diseño 8](#_Toc512192024)

# Diagramas

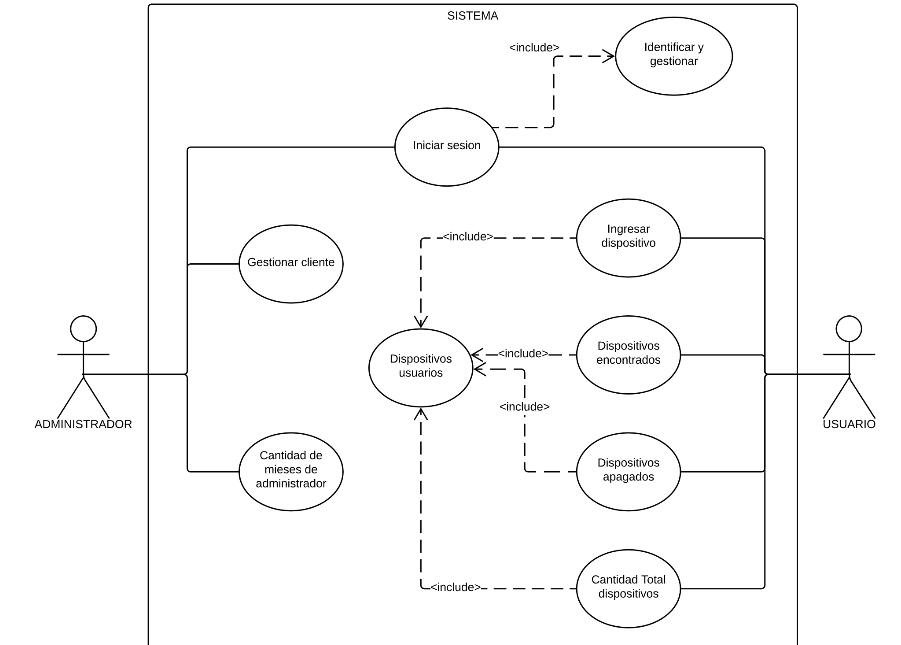
## Diagrama de Arquitectura



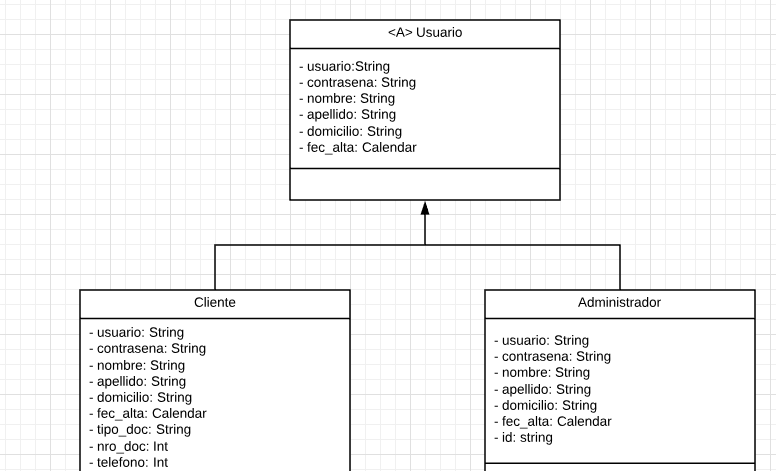
En la arquitectura se detalla que los usuarios podrán acceder al sistema por medio de un dispositivo. Se accederá a una interfaz de presentación. El sistema tendrá una capa de con la lógica de negocio. Se toma en cuenta que en entregas futuras se buscará persistir los datos transaccionales en una base de datos.

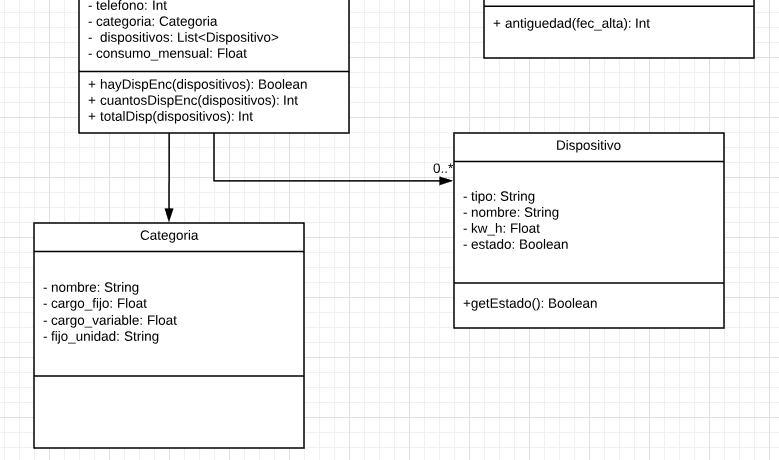
* El usuario del sistema se conectará con el sistema por medio de un dispositivo (celular, ordenador, etc), a una capa de presentación (GUI) donde verá los elementos necesarios para él. Detrás de la capa gráfica, estará el detalle de las reglas de negocio, donde se realizará la operativa del sistema. 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



## 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 |
| 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 |
| 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. | Funcionalidad |
| 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. | Seguridad |

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. | La herencia por ahora se realiza sólo con atributos, y no hay métodos. | Crear las clases sin vincular por herencia, pero implica repetir código. |
| 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 |
| 3 de Abril 2018 | Se decide realizar un método que recorra los dispositivos del cliente y devuelva toda la información necesaria para los demás métodos | Se evita repetición de código en los métodos que necesitan información de los dispositivos del cliente | Se llama a un método que centraliza todo el recorrido de la lista de dispositivos | Que cada método recorra la lista de dispositivos. |
| 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 |
| 1 de Abril | Se decide utilizar la tecnología Java | El lenguaje es amiga y familiar para la mayoría del equipo |  |  |