**SWGEGCSS –Sistema Web de Gestión para la empresa Geo Consult Service S.A.C**

Documento de Arquitectura

Versión 1.2

Revisión Histórica

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 28/03/2019 | 1.0 | Elaboración del Documento | Egg Rosas, Gerhard |
| 24/04/2019 | 1.1 | Modificación del Documento | Egg Rosas, Gerhard |
| 23/06/2019 | 1.2 | Modificación del Documento | Miranda Manrique, Sebastian |

Tabla de Contenidos

1. Introducción 4

1.1 Propósito 4

1.2 Alcance 4

1.3 Definiciones, acrónimos, y abreviaciones 4

1.4 Referencias 5

1.5 Vista General 5

2. Representación de la Arquitectura 5

2.1 Vista de Casos de Uso 5

2.2 Vista Lógica 6

2.3 Vista de Procesos 6

2.4 Vista de Despliegue 6

3. Objetivos y Restricciones de la Arquitectura 7

4. Vista de Casos de Uso 7

4.1 Paquete Finanzas 8

4.2 Paquete Atención al Cliente 8

4.3 Paquete Servicios 9

4.4 Paquete Seguridad 9

4.5 Paquete Gestión 10

5. Vista Lógica 10

5.1 Apreciación Global 10

5.2 Paquetes de Diseño Significativos para la Arquitectura 11

6. Vista de Procesos 11

7. Vista de Despliegue 12

 El Servidor Web: 12

 El Servidor de BD SQL: 12

8. Vista de Implementación 13

8.1 Diagrama de Componentes 13

9. Vista de Datos 22

10. Tamaño y Desempeño 22

11. Calidad 23

**Documento de Arquitectura de Software**

# Introducción

## Propósito

Este documento identifica las decisiones de modelado clave que se necesita tomar durante las primeras fases del desarrollo y proporciona instrucciones a nivel de diseño que ayudarán a elegir entre distintas opciones de proyecto. Asimismo, le ayuda a desarrollar un diseño global mediante la presentación de una arquitectura coherente construida con distintos tipos de componentes que le ayudarán a lograr un correcto modelo y beneficiarse de una plataforma. Puede considerar este documento como una guía básica preliminar de los aspectos más importantes relativos al diseño de la aplicación y la forma en la que será desplegado.

## Alcance

El diseño de una aplicación distribuida implica la toma de decisiones sobre su arquitectura lógica y física, así como la tecnología e infraestructura que se emplearán para implementar su funcionalidad. Para tomar estas decisiones, debe tener un conocimiento claro de los procesos empresariales que realizará la aplicación (sus requisitos funcionales), así como los niveles de escalabilidad, disponibilidad, seguridad y mantenimiento necesarios (sus requisitos no funcionales, funcionales u operativos). El alcance consiste en diseñar una aplicación que:

* Solucione el problema empresarial para el que se diseña.
* Proporcione un alto rendimiento y esté optimizada para operaciones frecuentes entre patrones de implementación.
* Permita la escalabilidad para cumplir las expectativas de la demanda y admita un gran número de actividades y usuarios con el mínimo uso de recursos.
* Se pueda administrar, permitiendo a los operadores implementar, supervisar y resolver los problemas de la aplicación.
* Se pueda mantener. Cada parte de funcionalidad debería tener una ubicación y diseño predecibles teniendo en cuenta distintos tamaños de aplicaciones, equipos con conjuntos de habilidades variadas y requisitos técnicos y cambios empresariales.
* Funcione en los distintos escenarios de aplicaciones y patrones de implementación.

## Definiciones, acrónimos, y abreviaciones

Una definición completa de los conceptos y de la terminología empleada en el documento se encuentra en el glosario de términos descrito en el informe del proyecto.

## Referencias

Como referencia para la elaboración de este documento se ha tomado la información obtenida en la fase de elaboración, y la fase de requerimientos. Además de los documentos de Especificaciones de cada Caso de Uso.

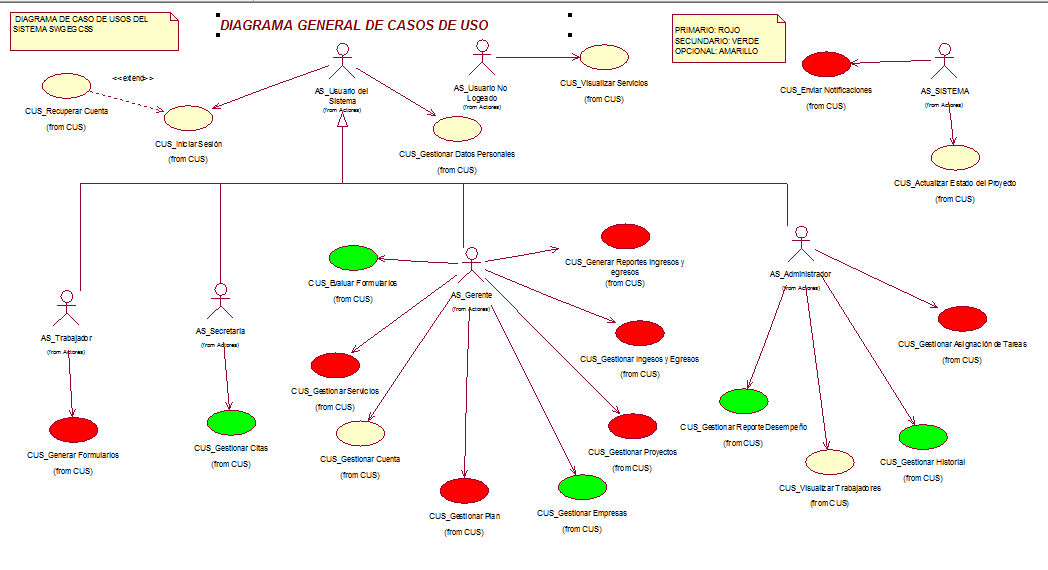
## Vista General

A continuación se presentará la arquitectura del sistema a implementar dividido en dos vistas: Vista de Casos de Uso, Vista Lógica.

# Representación de la Arquitectura

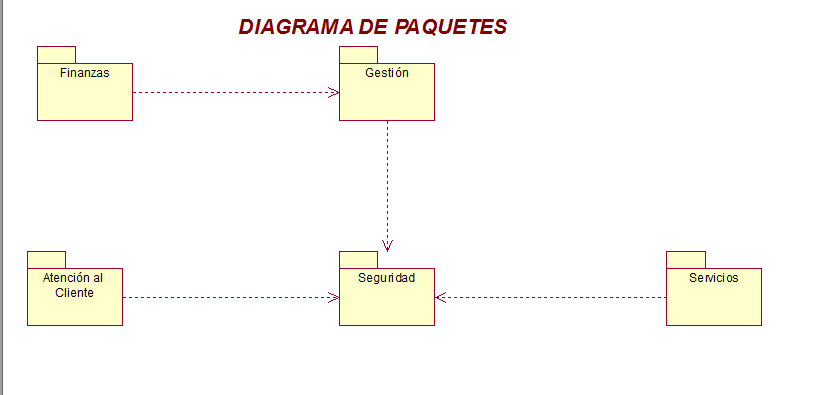
## Vista de Casos de Uso

Esta vista describe los casos de uso más significativos para el desarrollo del Sistema SWGEGCSS , sobre los cuales se han tomado las decisiones de diseño más significativas.



## Vista Lógica

Esta vista describe las partes significativas desde el punto de vista de la arquitectura del modelo de diseño y describe sus responsabilidades, como es su división en paquetes, los subsistemas los cuales están conformados por grupos de clases; el modelo conceptual de la aplicación para comprender así el dominio del problema.

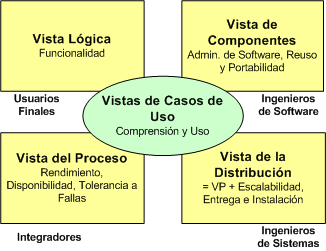


## Vista de Procesos

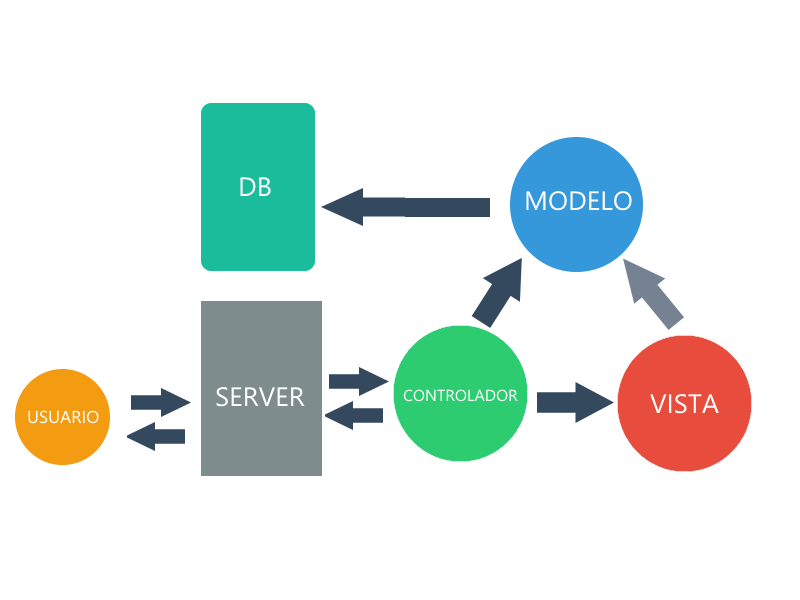
Las rutinas o programas que deberán ejecutarse tanto en el ordenador del cliente, como en los distintos nodos de la aplicación, dando como resultado que el sistema tenga un buen desempeño y funcionamiento.

## Vista de Despliegue

Se muestra la relación de la aplicación a desarrollar con el hardware requerido para el despliegue del sistema.



**Arquitectura MVC**

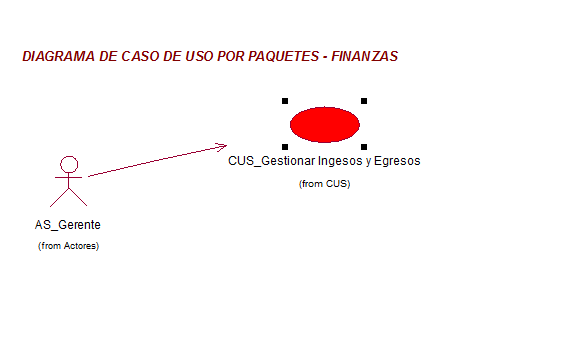


# Objetivos y Restricciones de la Arquitectura

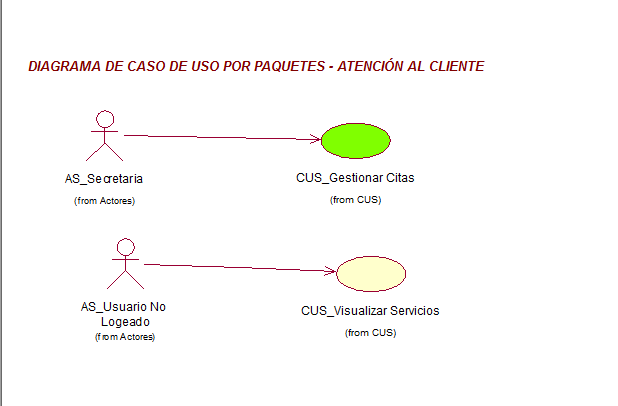
* El producto final debe interactuar con los usuarios en su capa de presentación con interfaces amigables para los usuarios. Las Interfaces están definidas en el documento de diseño.
* El producto final debe asegurar la protección de la data para el acceso a personas no autorizadas. Todo acceso remoto está sujeto a una identificación de usuario.
* El producto final será implementado en ambiente WEB a través de una arquitectura MVC en C#.

# Vista de Casos de Uso

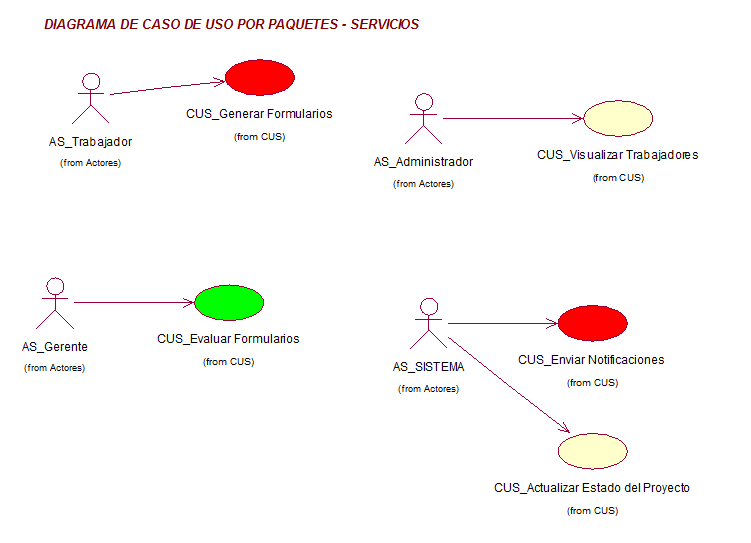
## Paquete Finanzas



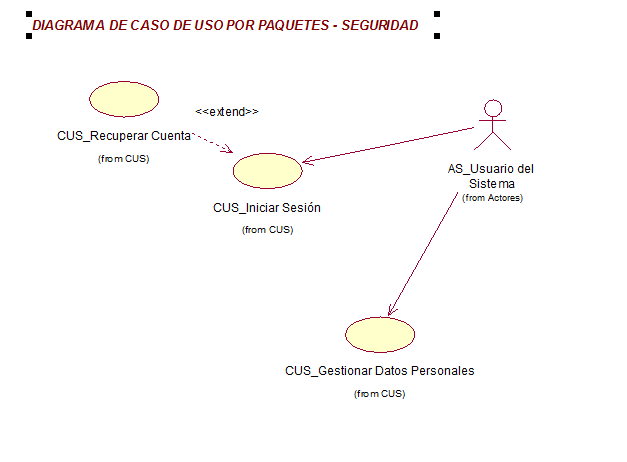
## Paquete de Atención al Cliente



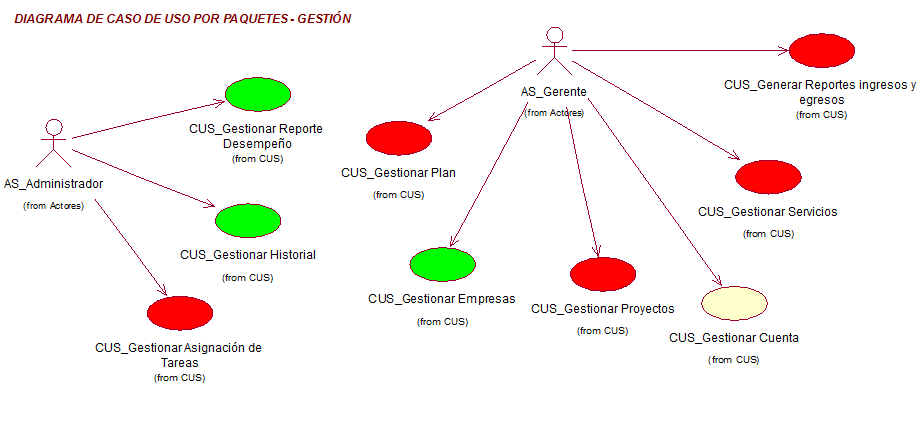
## Paquete Servicios



## Paquete Seguridad



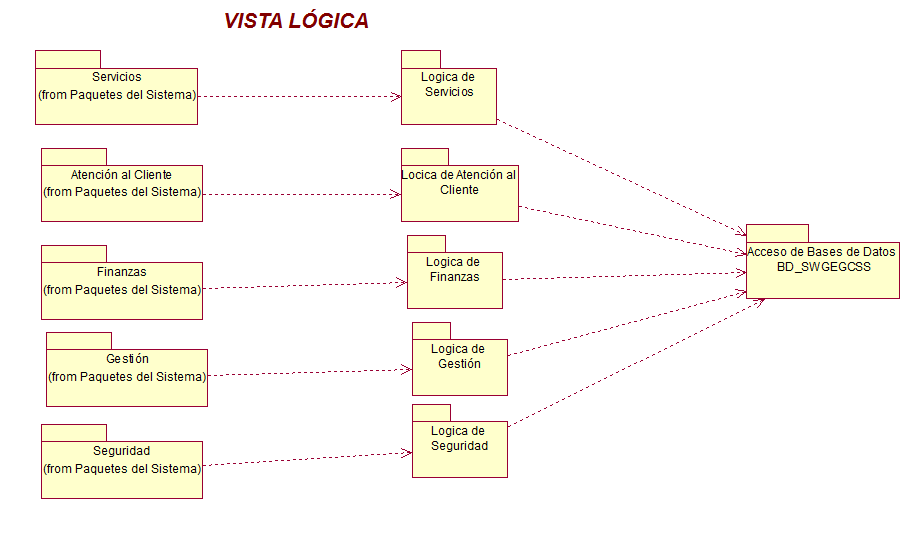
## Paquete de Gestión



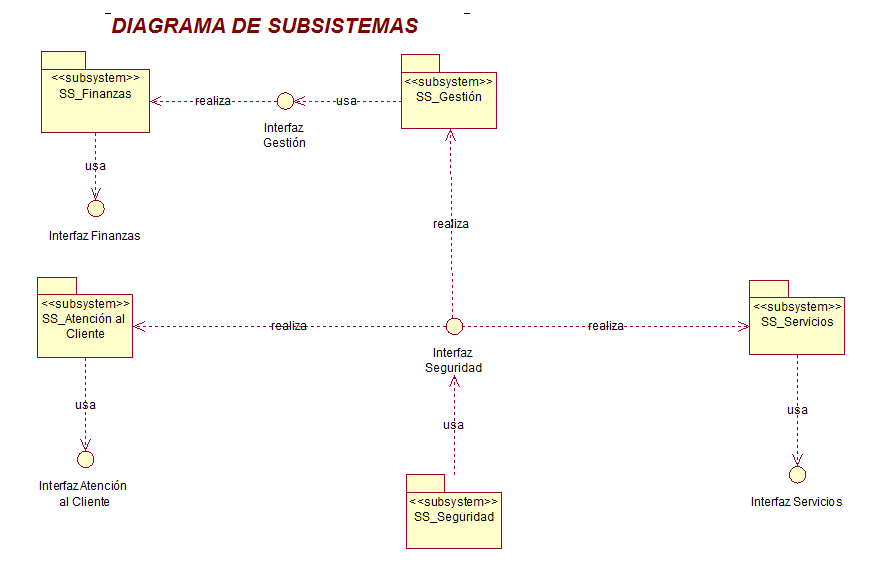
# Vista Lógica

## Apreciación Global

Se han identificado 5 subsistemas, los cuales tienen dependencia entre ellos, y a la vez dependerían de un paquete el cual se referiría a la programación.



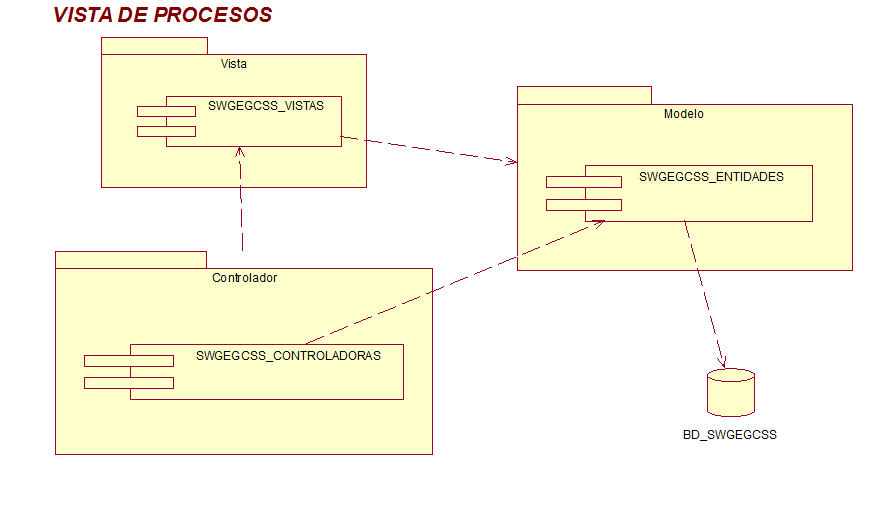
## Paquetes de Diseño Significativos para la Arquitectura



# Vista de Procesos

Esta vista tiene un proceso de alto impacto ya que define todo el desarrollo de la arquitectura de software, así también se dará la especificación de la mayor cantidad de procesos a realizarse.

* Por el lado del servidor de Base de Datos podemos decir que esta se estará ejecutando en SQL Server como proceso.
* Para el servidor de aplicaciones podemos decir que funcionara dentro del ambiente de Desarrollo .NET ISS (Internet Information Server).
* En la posición del cliente se usara el Framework del entorno .NET (Framework Redistributable Package) y en cuanto al browser, de igual manera, dentro de la aplicación.
* Finalmente, en el ámbito de los procesos respecto al servidor de aplicaciones se crearía un proceso cada vez que se realice una llamada a alguna clase.



# Vista de Despliegue

Para concretar físicamente el despliegue del sistema será necesaria la participación de tres elementos en la red:

* **Usuario PC:**

Este elemento accederá a la aplicación, mediante el acceso desde una PC remota.

## El Servidor Web:

El servidor Web será el repositorio de las clases controladoras para la ejecución de las transacciones e interacción entre la Base de Datos, y el cliente.

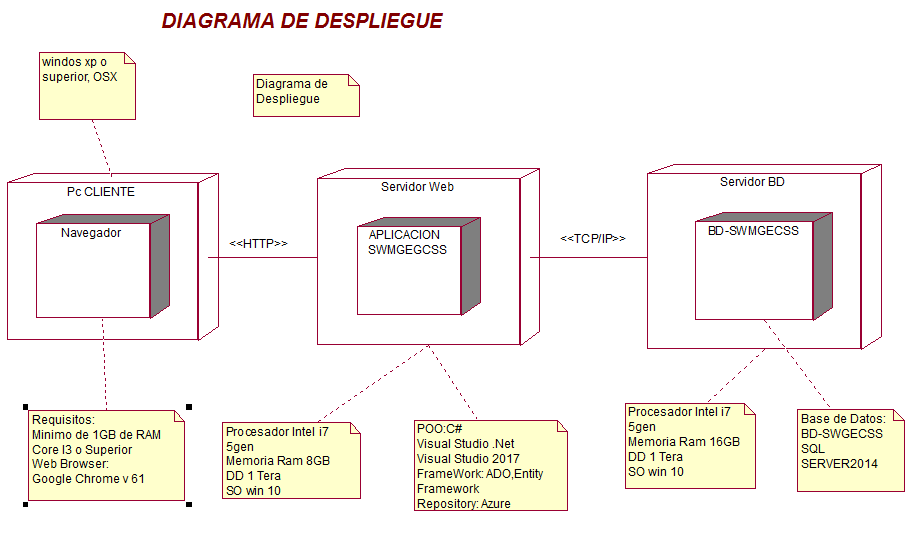
## El Servidor de BD SQL:

En el cual estará alojada la Base de Datos del sistema en SQL.

* **El Aplicación Web:**

Accederá a la aplicación web para realizar respectivas operaciones.

Con estos 4 datos de nuestra vista, podemos obtener lo siguiente, previa muestra de la arquitectura:

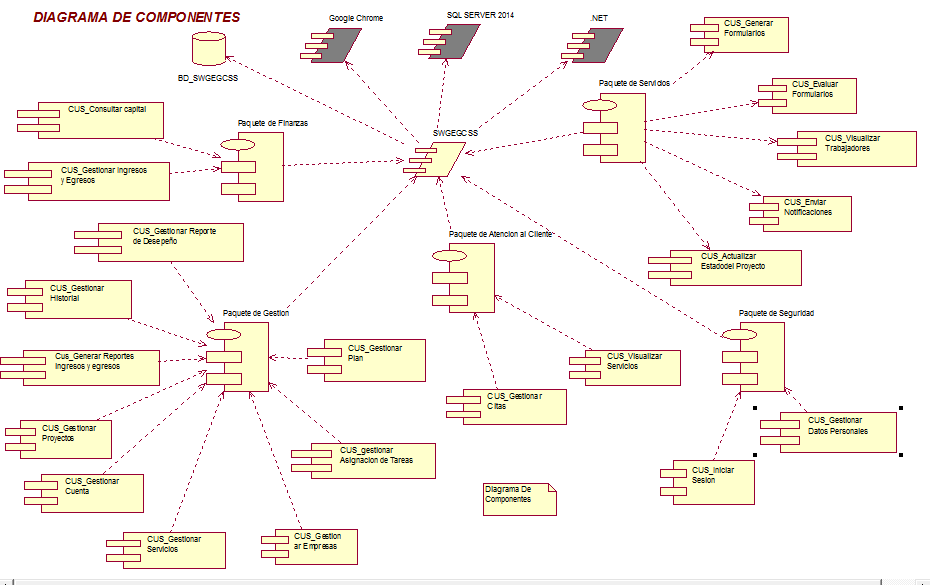


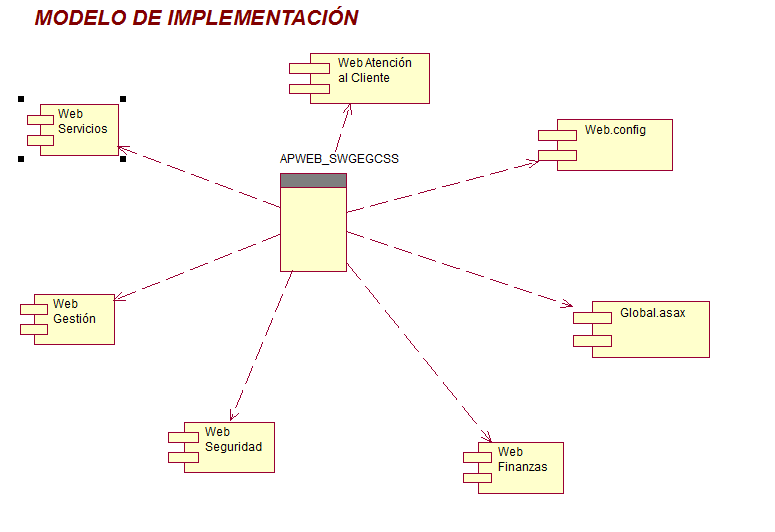
# Vista de Implementación

Esta vista muestra como fluye la información que ingresa el cliente, así como también la información que envía el sistema para él, la cual recorre una serie de procesos de control para una mayor integridad en la información.

Aquí definimos las capas en las que se dividiría el sistema a implementar:

## Diagrama de Componentes

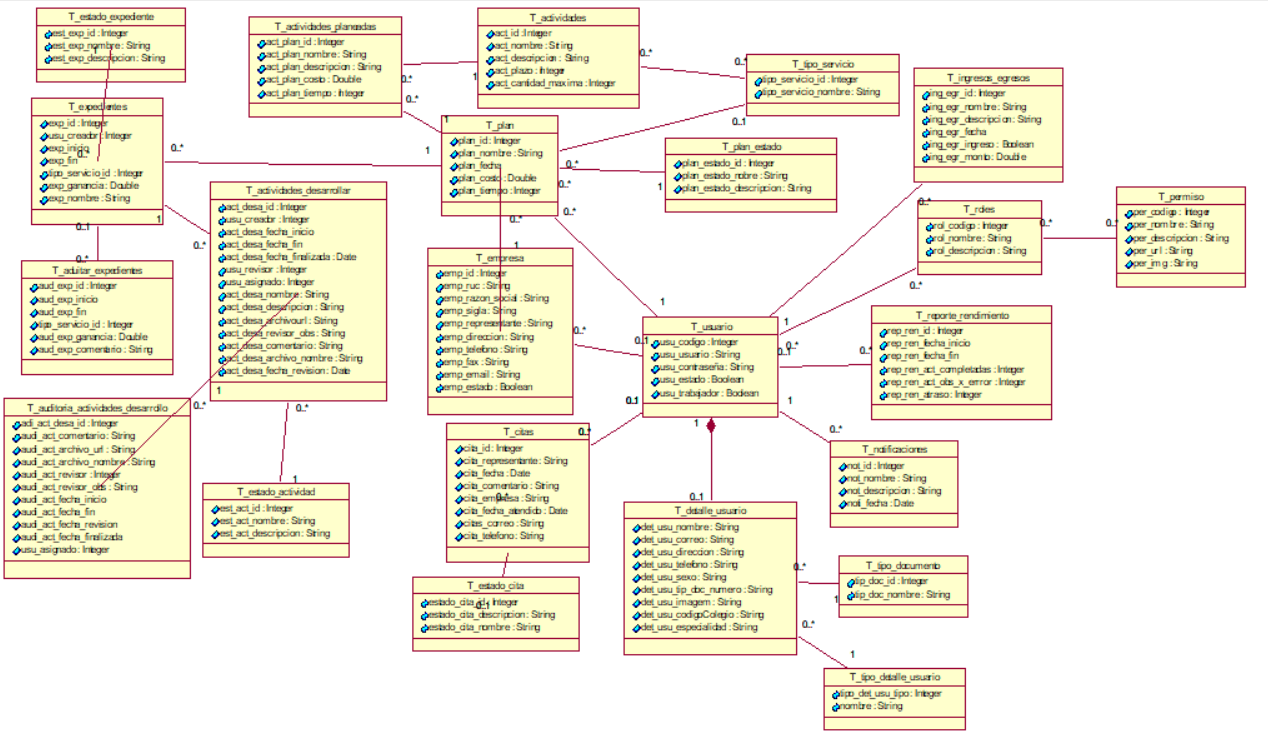




# Vista de Datos

El motor de almacenamiento de información que se utilizara para la gestión de base de datos es en SQL 2014R2. Definitivamente, además de la importancia de la eficiencia y robustez acerca del alojamiento del sistema, también están los aspectos de seguridad y verificación de la consistencia de la data.

**MODELO CONCEPTUAL**



# Tamaño y Desempeño

En lo referente al tamaño del sistema, se puede decir que los componentes que se crearán para llevar a cabo las funcionalidades básicas del Sistema, la base de datos formará a ser el repositorio de la información, que requieren de una buen arquitectura y robustez, para garantizar una buena performance o desempeño, debido a que cada uno de los elementos mencionados debe trabajar de manera integral para poder generar el servicio.

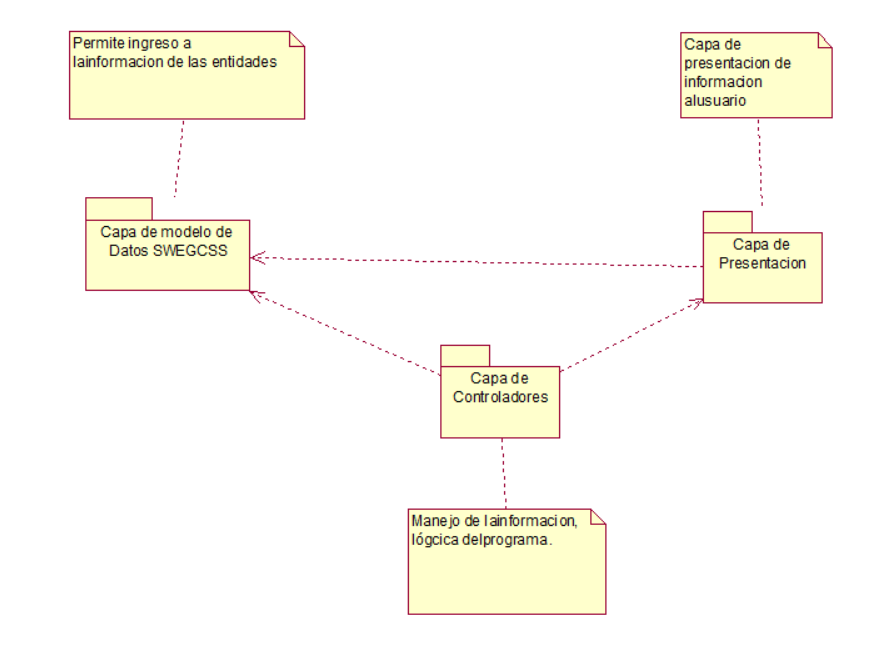
En lo que se refiere a desempeño sería de gran ayuda un elevado ancho de banda y nivel de velocidad para la transferencia de datos, para obtener un alto nivel tiempo de respuesta, y mejor performance de la aplicación.

# Calidad

Definitivamente, además de la importancia de la eficiencia y robustez acerca del alojamiento del sistema, también están los aspectos de seguridad y verificación de la consistencia de la data.

La data también se aloja en la Base de Datos sincronizada con los aspectos funcionales del Sistema SWGEGCSS

Arquitectura



# Riesgos de la Arquitectura

* Para desarrollar una aplicación bajo el patrón de diseño MVC es necesario una mayor dedicación en los tiempos iniciales del desarrollo. Normalmente el patrón exige al programador desarrollar un mayor número de clases que, en otros entornos de desarrollo, no son necesarias. Sin embargo, esta desventaja es muy relativa ya que posteriormente, en la etapa de mantenimiento de la aplicación, una aplicación MVC es mucho más mantenible, extensible y modificable que una aplicación que no lo implementa.
* MVC requiere la existencia de una arquitectura inicial sobre la que se deben construir clases e interfaces para modificar y comunicar los módulos de una aplicación. Esta arquitectura inicial debe incluir, por lo menos, un mecanismo de eventos para poder proporcionar las notificaciones que genera el modelo de aplicación; una clase Modelo, otra clase Vista y una clase Controlador genéricas que realicen todas las tareas de comunicación, notificación y actualización que serán luego transparentes para el desarrollo de la aplicación.
* MVC es un patrón de diseño orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.