PONTIFICIA UNIVERSIDAD JAVERIANA

Diseño y evaluación de arquitectura de software

Software Architecture Document
SAD

Juan David Cortes Olaya
Jhonatan stiven Gonzales

Proyecto Comedores

Bogotá, 4 de junio de 2020

Contenido

1.Introducción 3
1.1Propósito 3
1.2Alcance 4
1.3 acrónimos: 4
1.5 visión general 4
2.Representación arquitectónica (vistas) 4
3. Escenarios: 5
3.1 Requerimientos funcionales 7
3.1.1 Prepedidos: 7
3.1.2 ordenes 8
3.1.3 recetarios 9
3.1.4 Proveedores 10
3.2 Requerimientos no funcionales 11
3.3 Restricciones: 12
3.4 Priorización 12
3.5 Visualización de sistema 13
4. Casos de uso 15
5. Vista de Desarrollo 17
6. Vista procesos 21
7. Vista Lógica 22
8. Vista de despliegue 22
9. Patrones 24
10.Bibliografia 27

1.Introducción

El presente documento permite identificar aspectos y contenidos del documento de arquitectura de software, las especificaciones planteadas en el presente archivo fueron desarrolladas siguiendo metodología RUP, orientándose hacia el desarrollo de una plataforma de gestión logística de cadena de distribución para los comedores comunitarios del área de Bogotá, de igual manera se realiza el análisis 4+1 planteando las vistas pertinentes para el desarrollo e integración de la arquitectura de forma correcta

1.1Propósito

El presente documento fue elaborado con la finalidad de que se realice una visión y planteamiento general de la arquitectura que se va a realizar en cuanto al sistema de gestión de cadenas de alimentación para comedores comunitarios, para ello se tendrán en cuenta el proceso realizado tanto con requerimientos funcionales y no funcionales y de los mismos atributos de calidad planteados dentro del documento de especificación de requerimientos de software.

A partir de este análisis se desarrollará el modelo de vistas 4+1, permitiendo el desarrollo de cada uno de los modelos de presentación de la solución.

A continuación, se detalla el modelo

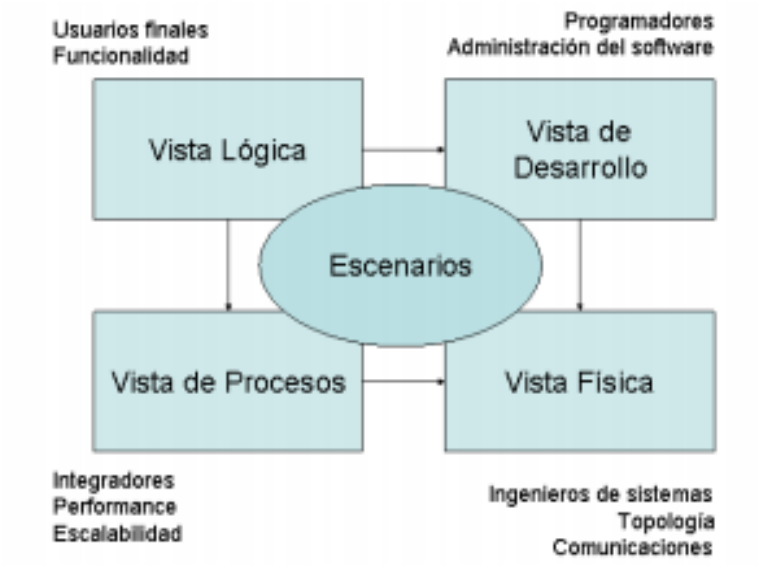


Imagen obtenida de: http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf

Es importante denotar que el modelo lógico será concerniente al diagrama que representa a la arquitectura a modo de componentes de software y no el del diagrama de clases que se suele presentar en cuanto a esta vista

De igual manera el modelo 4+1 al estar orientado a los usuarios finales e interesados, permite a los stakeholders, evidenciar la información que requieran de una manera sencilla y asequible para todas las partes interesadas.

1.2Alcance

Este documento de arquitectura de software ha sido basado en la construcción y desarrollo posible del sistema de distribución de alimentos de comedores, las secciones expuestas aquí fueron extraídas usando como témplate el esqueleto de SAD de SoDA

1.3 acrónimos:

SAD: software architecture design (documento centrado a la explicación y planteamiento de la arquitectura)

RUP: Rational Unified Process

SRS: Software Requirement Specification

1.5 visión general

A continuación, se plantean las secciones en orden de ejecución del presente documento, según la plantilla de software de "software architecture document"

- **2 sección**: Se describen las vistas y se explica porque y como se ejecutaron teniendo en cuenta los atributos de calidad y requerimientos
- **3 sección**: explica y presenta las restricciones que poseerá el sistema
- **4 sección**: Describe los requerimientos arquitectónicamente significativos
- **5 sección**: contendrá el caso de uso central y más importante dentro de la arquitectura
- **6 sección**: explica cómo se desplegará el sistema en los distintos ambientes
- **7 sección**: descripción de capas y subsistemas de la aplicación

2.Representación arquitectónica (vistas)

En esta sección se describirán los aspectos generales de la arquitectura, Utilizando el modelo de vistas 4+1 y la convención de nomenclatura RUP

- Vista Lógica:
  - Dirigido a: Diseñadores e implementadores
  - Factores: Se centra en los requerimientos funcionales, a partir de los cuales se describe de igual manera los casos de uso arquitectónicamente más representativos
  - Resultado: vista lógica

- Vista de procesos:
    - Dirigido a: integradores
    - Factores: Requerimientos no funcionales, se incluyen los aspectos arquitectónicos de sincronización, tiempo de respuesta y seguridad en él envió de solicitudes
    - Resultado: Vista de procesos

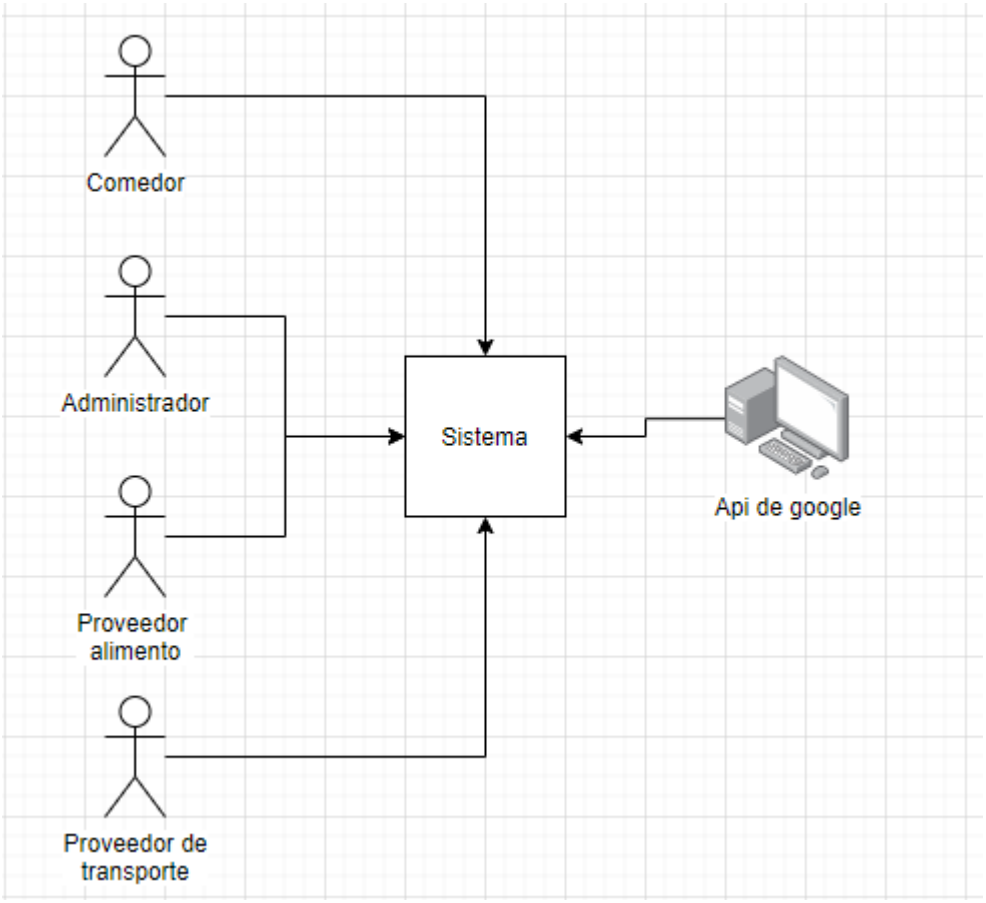
- Vista de despliegue:
    - Dirigido a: Ingenieros de infraestructura, consultores de infraestructura
    - Factores: topología del sistema a implementar, Relaciones entre componentes de hardware y software dependiendo de las necesidades y de los atributos de calidad
    - Resultado: Vista de despliegue

- Vista de implementación:
    - Dirigido a: desarrolladores
    - Factores: Componentes relacionados a software (sistemas y subsistemas)
    - Resultado: Vista de implementación

- Vista de casos de uso:
    - Dirigido a: todos los stakeholders del sistema
    - Factores: evidenciar que funcionalidades a grandes rasgos poseerá y serán implementadas dentro del sistema
    - Resultado: vista de casos de uso

3. Escenarios:

A continuación, se describe el diagrama de escenarios y sus actores



Actor	Descripción
Comedor	Usuario del comedor comunitario, será el encargado de solicitar domicilios y realimentación del algoritmo de efectividad, podrá elegir el proveedor de alimento y el proveedor de transporte, de igual manera podrá ver que disponibilidad de productos se poseen en un momento determinado y podrá crear y modificar nuevas recetas, así como confirmar o rechazar los pedidos
Administrador	Usuario que tendrá acceso a todas las características del sistema, podrá visualizar la disponibilidad actual dentro del sistema y acceder a datos de comedores y proveedores de ambos tipos, de igual manera podrá ejecutar pedidos y administrarlos según sea requerido

Proveedor de alimentos	Usuarios que podrán aceptar o rechazar pedidos dentro de la plataforma según las necesidades de los comedores, de igual manera tendrán actualizada la cantidad de productos disponibles dentro del sistema
Proveedor de transporte	Usuario que podrá aceptar o rechazar las peticiones de los comedores según el tipo de vehículo de igual manera mantendrá actualizada la información de vehículos en sistema con datos de contacto, características y disponibilidad
Api de Google	Sistema externo que permitirá calcular la distancia y el tiempo de recorrido en tiempo real de un comedor a un proveedor de alimento

3.1 Requerimientos funcionales

A partir de la priorización de requerimientos se identificaron 4 categorías principales asociadas a la funcionalidad de estos: preordenes, ordenes, recetarios, y proveedores.

A continuación, se analizan los requerimientos más importantes de cada categoría, si se necesita más información de estos se puede consultar en el anexo de requerimientos SRS:

3.1.1 Prepedidos:

- RF018. El sistema debe permitir al administrador y comedores gestionar los prepedidos con la disponibilidad que tenga el sistema
    - Especificación: el sistema debe permitir al usuario gestionar los prepedidos con los productos teniendo en cuenta la disponibilidad actual del sistema
    - Justificación: este requerimiento hace parte del Core del negocio ya que permite la elaboración de prepedidos según la disponibilidad en un momento determinado permitiendo la formulación futura de pedidos
    - Alcanzable: este requerimiento se podrá cumplir siguiendo la disponibilidad completa de los productos solicitados
    - Medible: se revisará a partir de la cantidad solicitada por el usuario en un momento vs la disponibilidad actual del producto en todos los proveedores alimentarios del sistema
    - Prioridad: alta

- RF 017. El sistema debe permitir al administrador y comedores seleccionar el comedor que desean realizar el prepedido.
    - Especificación: el sistema debe permitir selección del comedor de prepedido, ya que con los datos del mismo se efectuará el proceso del algoritmo de efectividad y permitirá categorizar y ordenar la información de los prepedidos
    - Justificación: este requerimiento hace parte del Core del negocio y permite la visualización categorizada de los pedidos según el comedor que haya solicitado el pedido, permitiendo mayor ordenamiento en los datos
    - Alcanzable: se realizará un request de petición de visualización y generación al servidor que permitirá identifica al prepedido con el Id único del comedor, el cual será almacenado en base de datos
    - Medible: se tendrá en cuenta el numero de prepedidos solicitados por un comedor, dicha medida será verificable en un momento determinado para el administrador del sistema
    - Prioridad: alta

3.1.2 ordenes

- RF 023. El sistema debe sugerir el proveedor con más efectividad para suplir el prepedido del comedor
    - Especificación: el sistema deberá sugerir al usuario el proveedor alimenticio y de transporte más propicio según costo, distancia, fecha de vencimiento y disponibilidad
    - Justificación: este requerimiento plantea el Core del sistema y representa el modulo de sugerencias de proveedores tanto alimenticios como de transporte para el usuario según la ejecución del algoritmo DEA
    - Alcanzable: el sistema recibirá la información del comedor y del proveedor de alimentos a partir del api de Google se calculará la distancia y tiempo de recorrido en tiempo real y con las variables de fecha de vencimiento, costo y disponibilidad ejecutará el algoritmo de eficiencia
    - Medible: calculo en hoja de Excel a cerca de las 4 variables evaluadas de todos los proveedores que tengan disponibilidad del producto, estas serán comparadas y calculadas para evidenciar la mejor eficiencia sugerida
    - Prioridad: Alta
- RF025. El sistema debe permitir a los proveedores alimenticios y proveedores de transporte aceptar o rechazar el pedido.

    - Especificación: el sistema debe permitir en el modulo individual de cada proveedor tanto alimenticio como de transporte el aceptar o rechazar el pedido solicitado por un comedor o por el administrador
    - Justificación: Un proveedor tiene potestad de ofrecer sus servicios o productos o negarlos a los comedores o administradores
    - Alcanzable: se logrará efectuar dicho proceso realizando un proceso de validación en los módulos individuales de cada proveedor donde se evidenciará un resumen del pedido y se tendrá un botón de aceptación o rechazo del mismo
    - Medible: una vez un proveedor haya aceptado o rechazado un prepedido la respuesta será registrada a modo de variable booleana en la base de datos de comedores
    - Prioridad: Alta

- RF026. El sistema debe permitir al administrador volver a sugerir los proveedores para el pedido, si alguno rechaza el pedido sea (transporte o alimenticio).
    - Especificación: el sistema debe permitir al administrador o al comedor el volver a realizar una petición de sugerencia si algún proveedor rechaza el pedido
    - Justificación: Un proveedor tiene potestad de ofrecer sus servicios o productos o negarlos a los comedores o administradores, sin embargo, el comedor o administrador puede volver a pedir una sugerencia ignorando a los proveedores que rechazaron el pedido anteriormente
    - Alcanzable: se logrará cumplir dicho objetivo si una vez el proveedor ha rechazado el pedido, se tiene en cuenta la bandera falsa que genera en base de datos y volviendo a ejecutar el algoritmo de efectividad sin tener en cuenta a los proveedores que tengan dicha bandera en falso o rechazo del mismo
    - Medible: una vez un proveedor haya aceptado o rechazado un prepedido la respuesta será registrada a modo de variable booleana en la base de datos de comedores y posteriormente se procede a ejecutar nuevamente el algoritmo de eficacia, el algoritmo no debe tener en cuenta los proveedores que rechazaron el pedido
    - Prioridad: Alta

3.1.3 recetarios

- RF006. El sistema debe permitir al administrador la visualización de recetas y su descarga en Excel.

    - Especificación: el sistema debe permitir almacenar y visualizar los pedidos de cada comedor a través de la interfaz de usuario
    - Justificación: el requerimiento permite la persistencia de recetas en base de datos y su mantenimiento a través de la interfaz, es indispensable que la misma pueda ser exportada a Excel para manipular sus datos mas fácilmente
    - Alcanzable: el sistema permitirá la consulta de base de datos en cuanto a las recetas actuales por proveedor y podrá exportarse como anexo en un archivo de Excel
    - Medible: el requerimiento podrá ser medido a partir de la revisión de data con respecto a las recetas en base de datos y cuando se genere el Excel de esta
    - Prioridad: Alta
- RF008. El sistema debe permitir al administrador gestionar los ingredientes que se ingresan a la plataforma.
    - Especificación: El sistema debe permitir modificar la receta que se encuentra almacenada en base de datos
    - Justificación: El sistema debe permitir modificar recetas según las necesidades de los comedores
    - Alcanzable: Se empleará la base de datos para almacenar la información pertinente respecto a los ingredientes que conforman una receta
    - Medible: se podrá visualizar los cambios a una receta ingresando directamente a la base de datos
    - Prioridad: Alta

3.1.4 Proveedores

- RF 018. El sistema debe permitir al administrador gestionar el transporte que se ingresan a la plataforma.
    - Especificación: el sistema debe poder ofrecer al administrador la visualización de los vehículos que se encuentren en sistema, desde sus datos de contacto hasta disponibilidad
    - Justificación: el sistema debe permitir monitorear en tiempo real los datos de los vehículos registrados en la plataforma
    - Alcanzable: el sistema ofrecerá información de la data relacionada en la base de datos según el transporte solicitado
    - Medible: una vez se modifique algún valor de los vehículos este debe verse reflejado en la base de datos
    - Prioridad: Alta
- RF 015 El sistema debe permitir al administrador gestionar la disponibilidad de productos que se ingresan a la plataforma.

- Especificación: el sistema debe permitir gestionar la disponibilidad de los productos que se encuentren en la plataforma
- Justificación: el sistema debe tomar la disponibilidad para poder ejecutar el algoritmo de eficacia y ofrecer las mejores opciones a los usuarios
- Alcanzable: toda modificación que se dé a nivel de front con respecto a los productos será actualizada en base de datos de comedores
- Medible: será visible la modificación de productos en base de datos
- Prioridad: Alta

3.2 Requerimientos no funcionales

- RNF003: El sistema debe presentar cada página en menos de 2 segundos
  - Justificación: Por experiencia de usuario se debe mostrar la respuesta a cada acción en tiempos cortos para mantener la atención del usuario centrado y mantener un flujo de la información
  - Criterio de aceptación: Al realizar clic en cualquier página aleatoriamente se debe mostrar contenido de respuesta en menos de 2 segundos. Para casos donde la respuesta por procesos en backend es complejo, está permitido el uso de cortinillas o animaciones de que se está cargando la página.
  - Restricciones: No se pueden mostrar 2 módulos de usuarios a la vez
  - Medible: la presentación de cada pagina se cumple en los límites de tiempo establecidos
  - Prioridad: Alta

- RNF011. El sistema debe cifrar la contraseña y usuario para garantizar seguridad
  - Justificación: Las credenciales de acceso en la autenticación deben ser protegidas para garantizar la no suplantación de los usuarios y acceso no permitido por terceros
  - Criterio de aceptación: La contraseña al ingresar en el formulario de registro o autenticación debe ser ofuscada.
  - Restricciones: ninguna
  - Medible: Ingreso al sistema
  - Prioridad: Alta

- RNF007 El sistema debe expirar el token cuando este supere 24 horas.
  - Justificación: Por seguridad el sistema debe ser la sesión ante la ausencia de usuario registrado, con el fin de evitar suplantación o vulnerar cualquier información sensible por un tercero.
  - Criterio de aceptación: El sistema debe manejar tokens seguros con expiración.
  - Restricciones: Comunicación con sistemas externos. Sincronización entre sesiones.

    - Medible: Usuario conectado exitosamente
    - Prioridad: Alta
- RNF010 El sistema debe garantizar el almacenamiento y disponibilidad de la información
    - Justificación: La información persistida en el sistema es necesaria que tenga respaldo ante cualquier evento de tal manera que se garantice su disponibilidad y manejo de históricos
    - Criterio de Aceptación: la información debe poder ser almacenada correctamente en la base de datos.
    - Restricciones: Capacidad en Disco de almacenamiento y backup de la información
    - Medible: la veracidad de la data debe ser medible en cuanto a cantidad de usuarios y datos ingresados en el front
    - Prioridad: Alta

3.3 Restricciones:

- Diagramas de diseño serán realizado usando UML
- No se requerirá hardware especial por parte del usuario final
- Las integraciones con sistemas externos serán implementadas empleando únicamente API Rest
- Las comunicaciones entre los dispositivos de los usuarios y el servidor que alojara el sistema de gestión de material académico serán realizadas usando http
- El diseño y la implementación serán realizadas empleando paradigma orientado a objetos
- Se emplearán bases de datos estructuradas para el manejo de persistencia
- El algoritmo de eficiencia será utilizado en el sistema usando librerías de .NET

3.4 Priorización

Para la priorización de requerimientos y atributos de calidad se tuvieron las siguientes consideraciones que impactaron la arquitectura:

- Core/ importancia: según el proceso de negocio y el enunciado se debe tener en cuenta que los requerimientos que afecten los procesos a nivel de funcionalidad principal deben ser prioritarios en cuanto a su ejecución e implementación, esto debido a que repercuten directamente en el objetivo final y motivación del sistema
- Esfuerzo/ tiempo de implementación: De acuerdo con la cantidad de esfuerzo se priorizo el requerimiento, debido a que estos iban a necesitar más tiempo de ejecución y repercutirán en los tiempos de elaboración de los demás requerimientos
- Variabilidad: los requerimientos que tengan mayor tasa de variabilidad serán los menos importantes en cuanto a priorización, esto debido a que se puede

continuar el análisis y ejecución simultánea de algunos de estos mientras se elaboran los requerimientos que sean más claros y fijos sobre la ejecución.

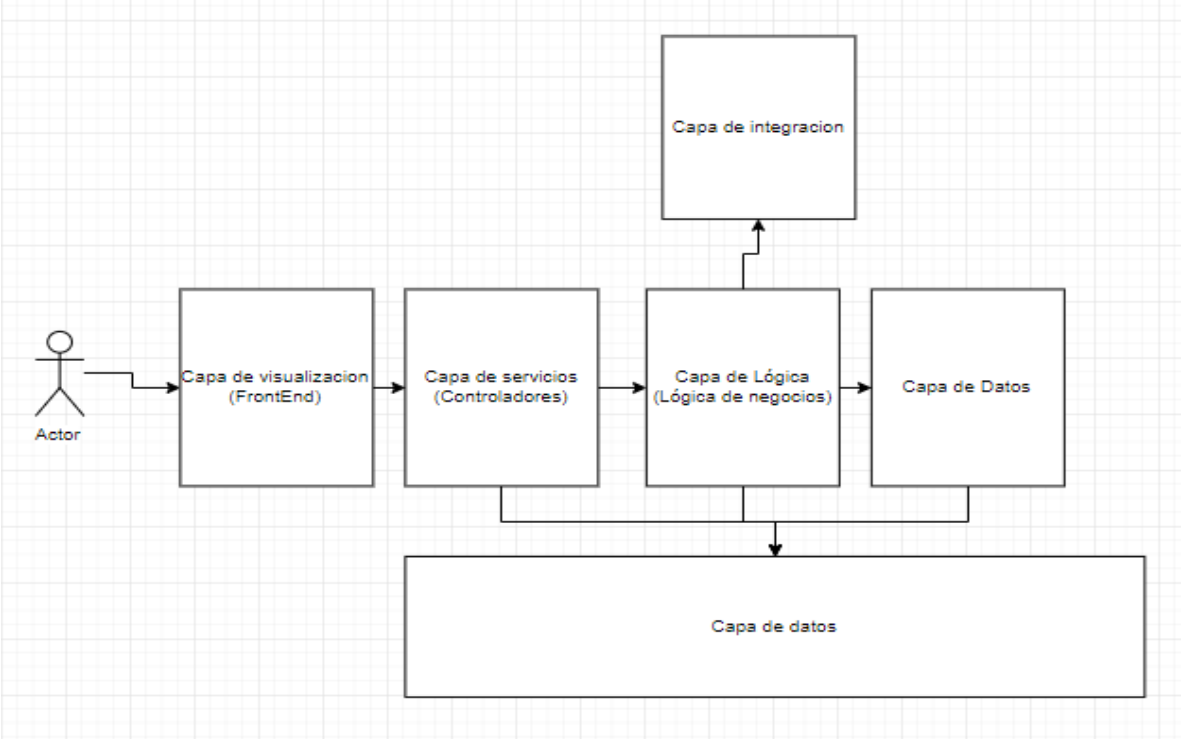
Adicionalmente la priorización tanto de los requerimientos funcionales como no funcionales pueden ser evidenciados en los anexos de “Matriz de requerimientos funcionales” y “Matriz de requerimientos no funcionales”, la trazabilidad de estos será visible en el adjunto “Trazabilidad de requerimientos”

3.5 Visualización de sistema

Siguiendo con el contexto y la problemática planteada en el enunciado, se establece al sistema como una interconexión entre varios niveles de capas de abstracción, dentro de las cuales se identificaron 6 capas, las cuales permitirán la correcta ejecución del sistema de gestión académica a continuación se describen cada uno de ellos en conjunto de sus funcionalidades dentro del modelo de negocio

- **1.Modulo (Presentación):** esta capa será la encargada de alojar la comunicación con el usuario mediante una interfaz gráfica, es a través de esta que el usuario realizará requests a la lógica de negocio teniendo en cuenta las restricciones del sistema y la funcionalidad general del mismo, dentro de sus funcionalidades se incluyen:
    - Presentación modulo comedores
    - Presentación modulo proveedores
    - Presentación modulo administrador
    - Presentación sugerencias del algoritmo de eficiencia
    - Acceso al sistema a través de credenciales
    - Presentación de procesos de la capa lógica
    - Visualización de disponibilidad
- **2. Modulo (lógica de negocio):** esta capa permitirá ejecutar el modelo de negocio según lo planteado en la problemática, permitiendo realizar las funcionalidades base y especializadas del sistema de Comedores, esto será conseguido mediante la integración de requests(provenientes de la capa de presentación) y la capa de datos (Capa de base de datos), dentro de sus funcionalidades se presentan las siguientes:
    - Configuración de requests para la obtención de un response a la capa de visualización
    - Llamados a la data almacenada en la capa de datos para realizar procesos de validación y evaluación
    - Interconexión con la capa de integración para realizar peticiones a clientes externos (sistemas independientes)
    - Cifrado de datos para temas de seguridad y transporte de información
- **3. Modulo (Integración):** Capa que permitirá en conjunto con la de lógica de negocio la integración con servicios externos al sistema, especialmente al sistema de Google maps, añadiendo de esta forma funcionalidades nuevas y complementarias sobre las de base del sistema, sus funcionalidades incluyen

    - Integración en tiempo real con el sistema de georreferenciación de Google para obtener distancia y tiempo de un comedor a un proveedor alimenticio
- **4.Modulo de entidades**: esta capa será transversal a la capa de lógica, la de integración y la de datos y permitirá el modelado de las entidades y clases con sus respectivas relaciones para la resolución de requests enviadas por el usuario desde la capa de visualización, proveerá respuesta a las necesidades del usuario, sus funcionalidades son:
    - Mapeo de entidades dentro del sistema dependiendo de los requests
- **5.Modulo de datos:** Capa designada a la persistencia del sistema, permitirá la comunicación con la base de datos y será indispensable en los mapeos y guardado de entidades para evitar pérdida de información, sus funcionalidades son:
    - Persistencia de los datos del sistema
- **6.Modulo de servicios:** capa que permitirá la recepción de requests desde el front(capa de visualización) y su redireccionamiento a la parte lógica del negocio, teniendo en cuenta aspectos de priorización y de encolamiento, esto teniendo en cuenta las divisiones por servicios que se plantearon durante la sección de requerimientos del mismo documento, dicho lo anterior esta capa también tendrá incidencia sobre los controladores que se empleen para acceder a la lógica mediante la subdivisión de tareas según categoría:
    - Integración de parte front con back a través de controladores
    - Encolamiento de requests



4. Casos de uso

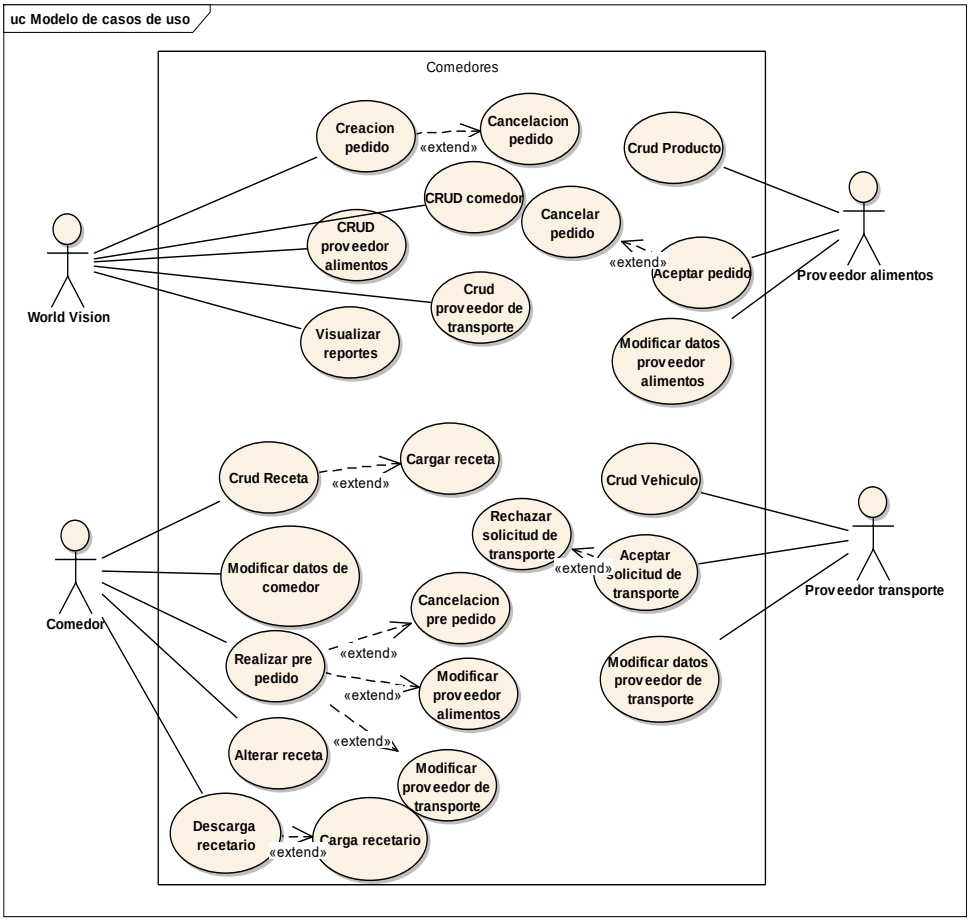


Figura 3 Diagrama de casos de uso

A continuación, se describen cada uno de los casos de uso presentados en la figura 3:

Actor: World Vision

- **Creación Pedido:** World Vision tendrá la opción de crear pedidos teniendo en cuenta los insumos que se encuentren en la tabla de disponibilidad según la cantidad de estos, World Vision podrá realizar dichos pedidos a los comedores comunitarios permitiendo la modificación de productos en cuanto a proveedores alimenticios o transporte en cuanto a proveedores de transporte, de igual manera, podrá anular una orden de pedido si es necesario
- **Cancelación de pedido:** si World Vision no está conforme con el pedido creado con anterioridad podrá eliminarlo y cancelar la orden hacia los proveedores
- **CRUD comedores:** World Vision tiene permisos como administrador del sistema de crear, modificar, actualizar o visualizar comedores almacenados dentro del sistema según la operación lo requiera

- **CRUD Proveedor de alimentos:** World Vision tiene permisos como administrador del sistema de crear, modificar, actualizar o visualizar Proveedores de alimentos almacenados dentro del sistema según la operación lo requiera
- **Visualizar reportes:** World Vision como administrador del sistema podrá ver reportes a nivel de interfaz gráfica que permita visualizar los pedidos realizados, los proveedores alimenticios, proveedores de transporte, productos y comedores y características propias de los mismos

Actor: Comedor

- **CRUD Receta:** los comedores podrán modificar, crear, ver o borrar recetas del sistema según lo requieran
- Carga receta: los comedores podrán cargar recetarios almacenados en la plataforma con anterioridad a modo de plantilla, especificando los productos, cantidades, y procesos necesarios para la elaboración del platillo, esto para agilizar el proceso de petición de prepedidos
- **Modificar datos de comedor**: el comedor podrá modificar sus datos de contacto alojados dentro del sistema según lo requiera
- **Realizar prepedido:** el comedor podrá realizar peticiones de insumos a proveedores alimenticios según lo requiera, el sistema utilizara el algoritmo de optimización sugiriendo los mejores proveedores de alimentos y el mejor transporte para realizar él envió, sin embargo, el usuario puede modificar las sugerencias presentadas por pantalla
- **cancelación de prepedido:** el comedor podrá anular una petición anterior de prepedido anulando la transacción a partir de la interfaz
- **Modificar proveedor de alimentos:** el sistema por naturaleza sugerirá el mejor proveedor de alimentos al usuario a través de la interfaz, sin embargo, el usuario puede cambiarlo durante la creación del prepedido
- **Modificar proveedor de transporte:** el sistema por naturaleza sugerirá el mejor proveedor de transporte al usuario a través de la interfaz sin embargo el usuario puede cambiarlo durante la creación del prepedido
- **Alterar receta:** el comedor podrá alterar la receta que viene por defecto en el sistema permitiendo introducir nuevos insumos a la misma o retirarlos para personalizar el prepedido
- **Descarga de recetario:** los comedores podrán descargar los recetarios que posean a modo de Excel para facilitar la modificación de los mismos o para tenerlos como referencia fuera del sistema
- **Carga de recetario:** el sistema admite la carga de recetas a partir de un documento Excel con la misma estructura de la descarga, esto permite facilitar la modificación de recetas por fuera del sistema

Actor: Proveedor de alimentos

- **Crud producto:** el proveedor podrá modificar, crear, borrar o visualizar los productos que se encuentren en sistema y le pertenezcan
- **Aceptar pedido:** el proveedor podrá decidir aceptar un pedido una vez se haya solicitado ya sea por un comedor o por World Vision

- **Rechazar pedido:** el proveedor podrá decidir rechazar un pedido una vez se haya solicitado ya sea por un comedor o por World Vision
- **Modificar datos proveedor de alimentos:** el proveedor de alimentos podrá modificar sus datos de contacto según necesidad

Actor: Proveedor de transporte

- **CRUD Vehículo:** el proveedor podrá modificar, crear, borrar o visualizar los vehículos que se encuentren en sistema y le pertenezcan
- **Aceptar pedido:** el proveedor podrá decidir aceptar una solicitud de envió una vez se haya pedido ya sea por un comedor o por World Vision
- **Rechazar pedido**: el proveedor podrá decidir rechazar una solicitud de envió una vez se haya pedido ya sea por un comedor o por World Vision
- **Modificar datos proveedor de Transporte:** el proveedor de transporte podrá modificar sus datos de contacto según necesidad

5. Vista de Desarrollo

A continuación, se muestran cada uno de los componentes que permitirán realizar las funcionalidades y ejecutar los servicios expuestos con anterioridad. Se puede apreciar una arquitectura multinivel o N-Tier que permite dividir la lógica por componentes físicos, permitiendo lograr mayor escalabilidad, simplicidad y hacer entendible el Código; de esta forma, se compone de módulos la solución, separando la presentación, de la lógica y de la persistencia, a continuación, se expone cada uno de los niveles de la arquitectura:

- **Nivel Vista:** nivel que permite mostrar gráficamente al usuario las tareas ejecutadas por la plataforma, se compone a partir de componentes que permiten al usuario interactuar con el sistema y realizar peticiones; este nivel se comunica a través de una fachada con el browser del usuario, permitiendo abrir la aplicación independientemente del dispositivo en uso, este nivel corresponde al *frontend* y se conectará al *backend* a partir de controladores dedicados, según el tipo de petición y el objeto que controle el otro extremo, se aprecia de igual manera que existe un subcomponente dentro de la vista que permite utilizar las entidades a modo de espejo con las del *backend*, esto es necesario para la construcción de *wrappers* y, el envío posterior del Json al *backend*.
  De igual manera, se aprecia que la conexión con el *backend* se hará a través de protocolo Http de tipo Rest, esto debido a que es una manera estandarizada de realizar peticiones y permite una Simplicidad mayor a Soap.
  Las funcionalidades y servicios de comedores se expondrán a partir de este componente de arquitectura permitiendo al usuario conectarse al mismo a través de un browser. La visualización de comedores es responsive, permitiendo que sea visualizada en un móvil o en un ordenador.
  De igual manera este componente será el encargado de validar las credenciales del usuario en una pantalla de login, a través de la cual se realizará un request y se obtendrá un Bearer Token del backend, el cual es necesario para ejecutar cualquier otro request de la plataforma una vez se haya logueado el usuario.

- **Nivel Lógica:** será el nivel encargado de ejecutar las tareas del *frontend* y de realizar las tareas de comedores, es en este nivel donde se ubicará el algoritmo de optimización (Simplex) y la normalización de datos de los proveedores para la ejecución del cálculo, de igual manera en este nivel se realizará la creación del ***Bearer token*** que permitirá acceso a los servicios únicamente a los usuarios registrados en plataforma, añadiendo una nueva capa de seguridad. Este nivel también posee una capa transversal a las demás: un sidecar que permitirá el uso de las entidades base a lo largo del *backend*. Adicionalmente el *backend* se conectará con un Api de Google externo empleando Rest para permitir obtener la geolocalización y distancia entre dos puntos (Comedor y Proveedor de alimentos).
  Este nivel permitirá la ejecución de la lógica de negocio contenida en comedores a través de las requests provenientes del Frontend, en este nivel se recibirán los requests a partir de controladores según las entidades de comedores (comedores,proveedores, productos).

- **Nivel de datos**: corresponde al manejo de la base de datos a través de *Entity Framework* permitiendo recopilar información que será utilizada en el *backend*, En este nivel se contendrán las tablas definidas según el negocio de comedores conteniendo data como proveedores, productos, comedores y cada una de sus características correspondientes para la ejecución de procesos a nivel de negocio

El segundo patrón empleado fue el de capas; este fue aplicado en el *backend* y se ubica a lo largo del mismo y permitirá ejecutar las funcionalidades del negocio de manera separada y dividida para aumentar la seguridad y precisión del proceso de ejecución, las capas desarrolladas son representadas a continuación:

- **Capa Controlador de lógica:** contiene a todos los controladores con sus firmas de métodos para ofrecer servicios al exterior del *backend*, se encuentran realizados con un Rest API y asegurados con un JWT (Json Web Tokens) de esta manera únicamente los usuarios que se encuentren registrados en plataforma podrán consumir funcionalidades, todos los controladores tienen habilitado el ***Cross origin*** para que puedan ser consumidos desde cualquier petición externa. Cada entidad dentro del negocio de comedores tiene su propio controlador permitiendo dividir lógica

- **Capa de Lógica:** capa de lógica general, permite la transmisión de datos desde la capa de controladores a la capa de BLL o capa de business logic, la capa de lógica se encarga de redireccionar el contenido de las peticiones a la lógica en capas correspondiente, esto permite centralizar lógica general, es en esta capa donde se ejecutan los procesos de optimización y normalización de los datos provenientes de *frontend*, el proceso de normalización recibirá los parámetros de optimización de la base de datos y ejecutara los procesos necesarios para estandarizar la unidad de medida, a continuación con dichos datos normalizados se procede a ejecutar los procesos de Simplex y DEA para hallar la mejor efectividad en cuanto a proveedor alimenticio y de transporte, las variables a calcular con estos procesos son: costo de producto , costo de envió, distancia y días de vencimiento

- **Capa de negocio o BLL:** capa especializada de lógica de negocio se separa por componentes de entidades permitiendo que la capa asigne al componente correspondiente la ejecución de la petición externa, igualmente permite la conexión con la capa DAL o de acceso a datos separando lógica de persistencia. Esta capa dividirá la lógica general del negocio de comedores permitiendo que cada entidad de negocio posea su propio CRUD y sus funciones especializadas ejecutadas de una manera independiente, por ejemplo: la capa BLL de transporte permitirá buscar el transporte más eficiente según la categoría de alimentos que se requiera transportar minimizando de esta forma costos.

  Es importante destacar que esta capa permitirá a través de la minimización de costo y de distancia y la maximización de fecha de vencimiento y disponibilidad la elección del mejor producto y vehículo, de igual manera cuando sea un pedido "partido" es decir que el proveedor es ideal pero no se posee la cantidad acorde al numero de niños se ejecutara nuevamente el algoritmo de optimización y se elegirá al siguiente proveedor de alimentos más optimo hasta suplir la necesidad en su totalidad.

- **Capa de acceso a datos o DAL:** la capa de acceso a datos permite a través del uso de un *Entity Framework* el acceso al modelo y datos de la base de datos, se empleó dicha tecnología ya que permite modificabilidad en sentido bidireccional, ya sea desde el modelo en *backend* o desde la misma base de datos, permitiendo realizar consultas rápidas y sin necesidad de modificar mucho las *queries*. De igual manera cada entidad permitirá ser accesada por una única clase especializada que si bien usa la misma conexión a base de datos permite efectuar cambios o modificaciones específicas de cada entidad dentro del negocio

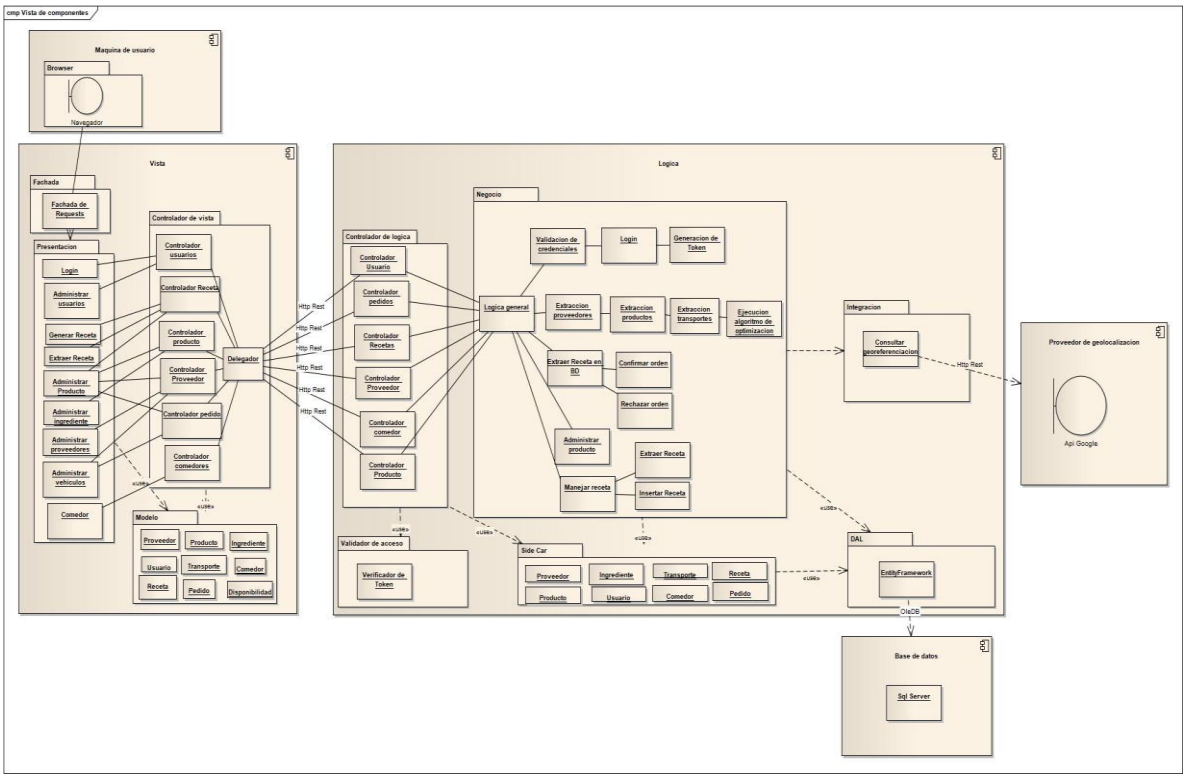
- **Capa de entidades:** capa de datos que es transversal a todas las capas de lógica, esto porque permite consistencia en el transporte de datos entre las mismas. Esta capa permite la simulación de las clases en base de datos a manera de espejo simulando con exactitud los atributos de cada tabla, su funcionalidad es transportar los datos entre las demás capas.Todas las entidades de negocio se encontraran en esta capa permitiendo identificar desde los comedores hasta los proveedores y sus características propias de objeto, Es importante destacar que esta capa está condicionada por el Entity Framework , es decir todo cambio que se realice en base de datos se verá afectado en el modelo .Finalmente en esta capa también se configuran clases tipo Wrapper que permitirán la modificación del Json de Response según el request del front lo requiera

- **Capa de integración:** capa que tendrá la función de integrar la información de la lógica de negocio con la lógica externa del api de Google, se encargará de realizar peticiones de tipo Rest a este sistema remoto y permitirá obtener datos de geolocalización y distancia de un punto A a un punto B, es decir entre un comedor y un proveedor de alimentos, esto con el objetivo de que el algoritmo de eficiencia necesita obtener la distancia en tiempo real de las dos entidades para obtener a mejor opción y que la misma sea entregada al usuario

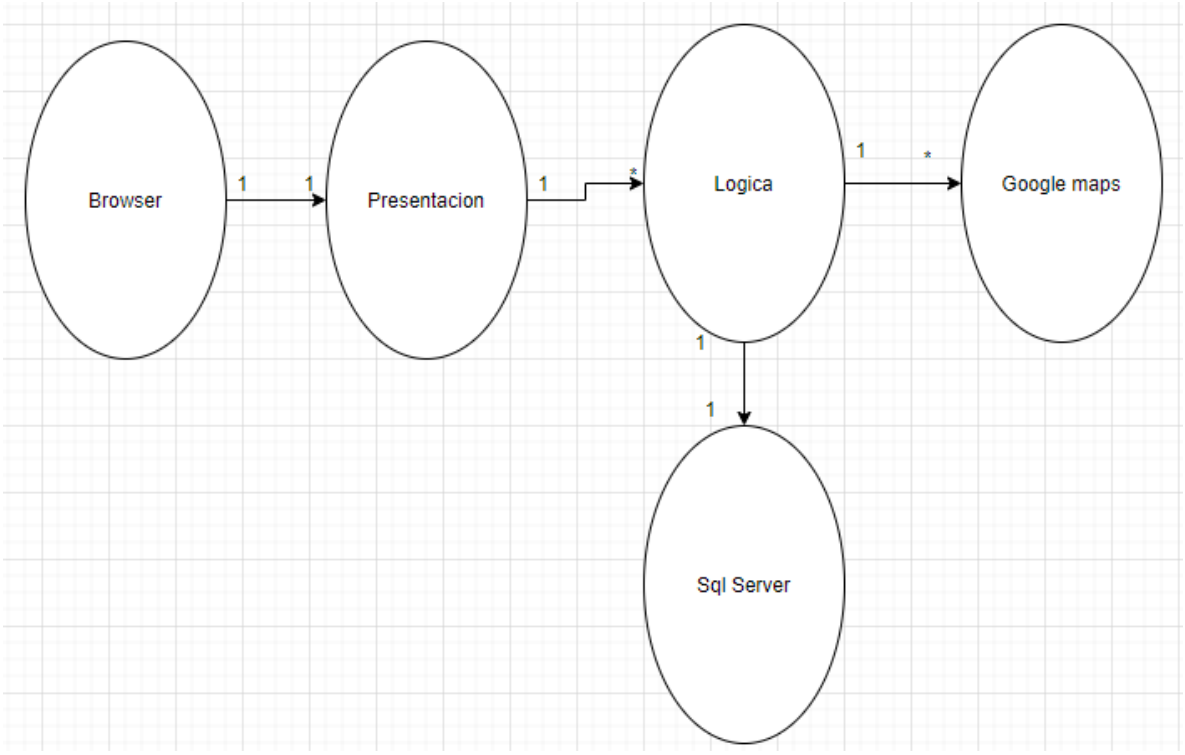
- **Capa de seguridad:** esta capa posicionada al mismo nivel que los controladores, recibirá las credenciales de usuario introducidas en la página de login y verificara que el mismo

si exista en el sistema con las credenciales obtenidas, en caso positivo se le enviará un Bearer Token al front que tendrá vigencia de 24 horas y permitirá acceder a este usuario a las características del sistema de Comedores

Finalmente se plantea la parte de persistencia, este componente permitirá almacenar los datos de las entidades que conforman el sistema y permitirá al nivel de Lógica el acceso a los mismos para la ejecución de funcionalidades y servicios. Este funcionara en conjunto con el Backend y EntityFramework para extraer de manera eficiente los datos de base de datos y utilizarlos en los procesos de lógica de negocio.



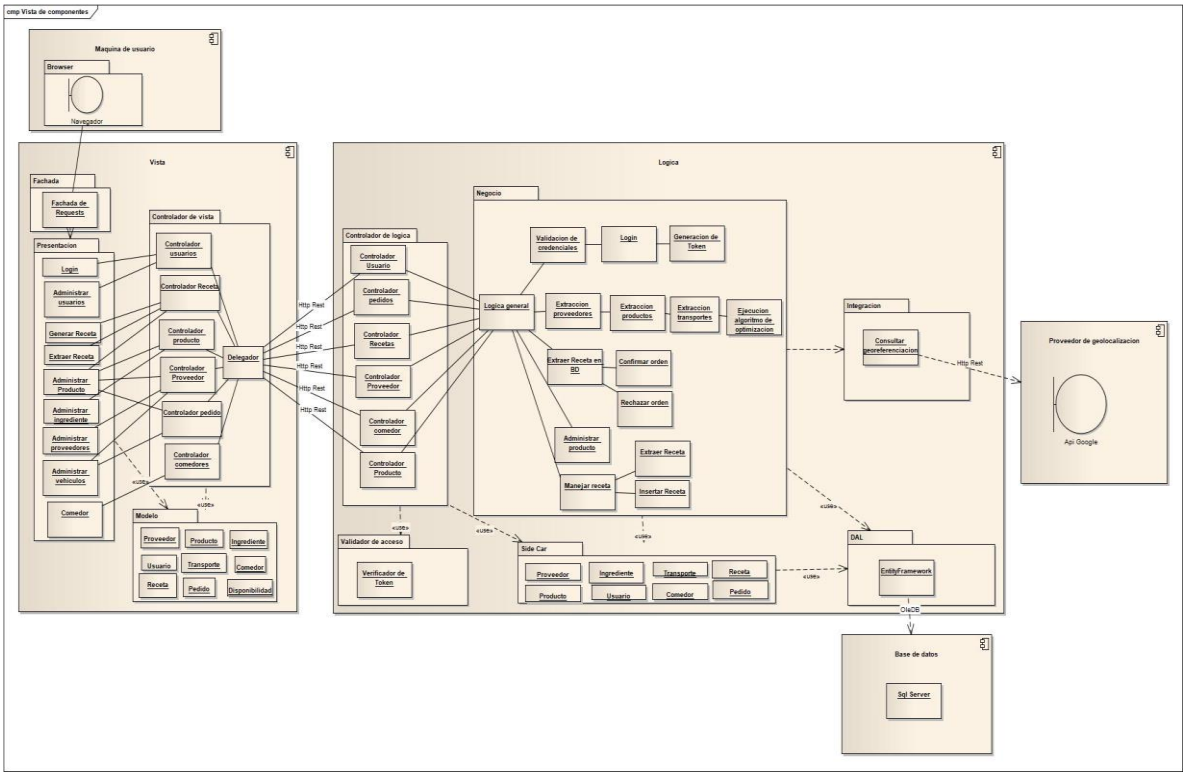
6. Vista procesos



El anterior diagrama describe 5 procesos corriendo, los cuales permitirán la correcta ejecución del proceso de plataforma de gestión académica:

1. Browser: proceso con el cual el usuario final podrá conectarse al sistema, desde el browser el usuario podrá realizar las funcionalidades propias dependiendo de su rol y restricciones
2. Presentacion: proceso que se encontrara corriendo en la máquina del usuario, no tendrá dependencias en cuanto a características de infraestructura, permitiendo ser abierto en el dispositivo de elección de este último, desde este proceso surgirán los request a la parte lógica por parte del usuario, se alojara en un contenedor web
3. Logica: proceso que se encontrará corriendo en la máquina servidor, este proceso podrá ser múltiple ya que se emplea una táctica de servidor espejo para suplir al primer servidor cuando se haga mantenimiento o haya mucha demanda en cuanto a requests, se conformará por dll´s que suplirán las necesidades del usuario final
4. Sistema de pagos: proceso externo al sistema: de tipología lógica, que dará respuesta a las necesidades y solicitudes del proceso de lógica
5. base de datos SQL SERVER: proceso de persistencia que almacenará los datos necesarios por el proceso de lógica de negocio

7. Vista Lógica



El modelo anterior compone la vista de lógica, en la cual se evidencia que está compuesta por componentes a nivel macro especificando únicamente los contenedores y las relaciones entre ellos, se logra observar que se compone de 4 componentes que conllevan a la funcionalidad adecuada del sistema a continuación se describen cada uno y sus funciones:

1. Componente de visualización: componente gráfico permite al usuario funciones de logueo y de lógica de negocio de acuerdo con el enunciado expuesto, desde aquí provienen las requests
2. Componente de lógica: componente que realizará el desarrollo y la adaptación de los datos en persistencia según la normativa del negocio, de igual manera será el encargado de conectarse a sistemas externos mediante integración api, y del manejo de bases de datos para el desarrollo de la persistencia, estará construido a modo de capas
3. Componente de base de datos SQL: componente de persistencia con caracterización estructural, almacenará los datos pertenecientes al core del sistema.
4. Componente de api: Componente externo del sistema que tendrá la lógica para efectuar cálculos sobre distancia y tiempo de desplazamiento de un comedor a un proveedor de alimento

8. Vista de despliegue

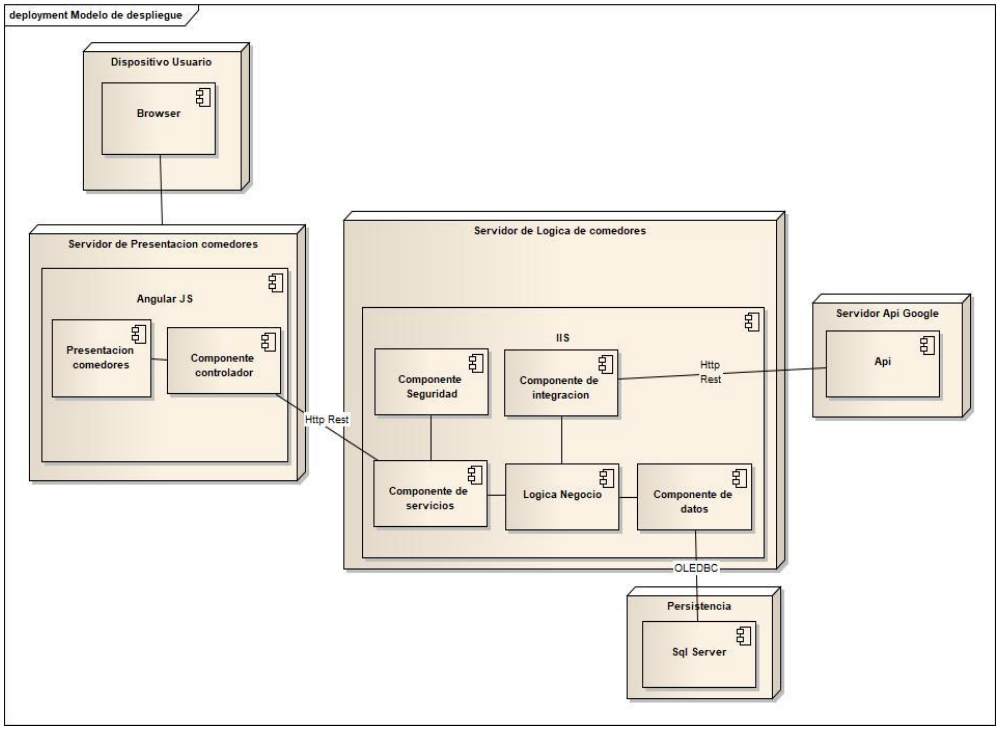


Imagen 7 Arquitectura de despliegue

Se desarrollará el servidor de lógica empleando un servidor de Azure remoto, que a través de configuración, ofrecerá una capa de servicios para el acceso a la funcionalidad interna, esto será realizado mediante un IIS, el servidor permitirá la subdivisión y encapsulamiento de requests según la clasificación de los mismos subdividiendo las tareas y aliviando carga del mismo dicha división vendrá identificada por las entidades encargadas del correcto funcionamiento de la lógica de negocio(comedores, proveedores, productos).

La mayor ventaja de usar un server de Azure es que permiten escalabilidad mediante el uso de múltiples servidores espejo dependiendo de la demanda de requests realizadas por los usuarios, incrementando el número de máquinas según el número de usuarios, esto será eficiente para el manejo de la parte de lógica del sistema y permitirá la asignación dinámica de requests según la carga de los servidores. El servidor contara un sistema operativo de Windows server 2012 ya que permite una configuración sencilla tipo Plug and play que permite la rápida configuración de servicios que hayan sido desarrollados en tecnología Microsoft.

En la parte del frontend será realizada mediante un aplicativo web y será posible ejecutarla en cualquier dispositivo de usuario esto debido a que se empleara framework Angular JS que es altamente adaptativo a dispositivos, las peticiones se realizarán desde el Browser de sus dispositivos, el browser enviara peticiones según los usuarios a este componente y serán ejecutadas mediante peticiones Promises a través de HTTP/REST al servidor lógico remoto de IIS.

Adicionalmente se presenta la conexión con un sistema externo, de geolocalización de Google que permitirá identificar la distancia entre el comedor y el proveedor de alimentos,

Se realizarán Request de tipo Http-Rest para la conexión y respuesta con el api público de este componente.

finalmente, la persistencia será manejada mediante una base relacional Windows server SQL que permitirá la conexión mediante OLEDB al servidor del IIS para obtener la data necesaria para las ejecuciones de procesos de negocio, esta conexión se realizará de manera automática empleando EntityFramework y Linq

9. Patrones

- **MVC(Modelo vista controlador):** se desarrollo una aproximación al patrón de modelo de vista controlador en la medida que se desarrolló un front, para la parte de visualización y presentación de la parte de presentacion al usuario a través de una interfaz gráfica, para el direccionamiento de peticiones según su tipología se crearon controladores que permiten dividir los requests y manejar entidades especiales según la necesidad impuesta desde el front, y finalmente la parte lógica que contendrá todo el modelo del negocio, contendrá una interfaz que expondrá los métodos internos para consulta remota, y realizará las labores lógicas y de persistencia en cuanto a la base de datos, y la integración a proveedores externos mediante consumo de api. a continuación, se enuncian las ventajas y desventajas del uso de este método teniendo en cuenta la arquitectura planteada
    - La implementación se realiza de forma modular, dividiendo aspectos como , el desarrollo de la visualización, la lógica de negocio y la conectividad de lógica a través de la exposición de métodos en los conectores
    - las vistas se vuelven autosuficientes y fácilmente actualizables, esto debido a que la parte lógica debe mantener actualizada la interfaz de usuario
    - cualquier cambio a nivel de lógica queda desacoplado de la vista y de los controladores, permitiendo un desarrollo atómico que permitirá realizar cambios en cuanto a modelo de negocio y será transparente ante el usuario y el programador
    - la información aportada por la vista y la ejecución efectuada sobre ella en el modelo serán independientes, permitiendo gran variabilidad en cuanto a respuestas de requests, como la variación de cursos o material de apoyo
    - Permite un amplio grado de extensibilidad y mantenibilidad dada la distribución de su funcionamiento.

  Los atributos de calidad que impacta este patrón son los siguientes:

    - Reutilización: se permite reutilizar componentes debido a la interfaz que se posee y permite liberar de manera pública funcionalidades propias de la lógica, permitiendo el consumo a partir de un tipo de interfaz que está desacoplada de la lógica, se puede reutilizar el módulo de pagos en este caso para proyectos posteriores ya que la funcionalidad variara poco o nada.

- Mantenibilidad: al estar divididas la parte de presentación y lógica se pueden modificar ambos componentes o revisar de manera separada y eficiente, Esto ligado a una buena documentación permitirá que el código sea sencillo y fácil de comprender
- escalabilidad: el componente de lógica y de visualización están desligados se pueden extender o re implementar nuevas funcionalidades de manera casi-atómica.

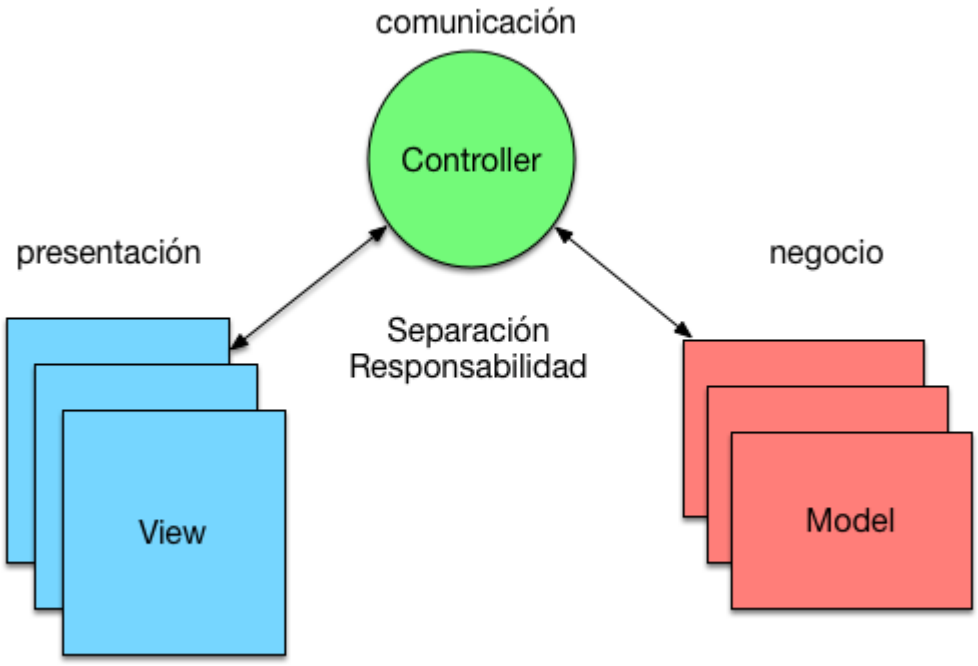


imagen obtenida de https://www.arquitecturajava.com/patron-mvc-arquitectura-cliente-vs-servidor/

- **Capas**: El patrón de capas será implementado en la capa lógica interna de la parte de negocios, se implementará teniendo en cuenta 3 capas principales no visibles en el modelo, una capa de lógica que representa la mayor abstracción de datos, la capa DAL que será la capa de acceso a datos y la capa entities que será la encargada de mapear los datos a clases, de igual manera los casos de uso que se vieron en el diagrama y que extienda de alguna funcionalidad específica serán representadas como capas adicionales al modelo de entidad concerniente, se eligió este patrón debido a los beneficios que ofrece, los cuales serán mostrados a continuación:
    - Se posibilita la estandarización de procesos(capa dedicada a lógica, a data y a base de datos)
    - Se abstrae la funcionalidad específica en cada capa
    - Se garantiza el traspaso de data entre capas debido a su acoplamiento
    - Intercambiabilidad de capas de funcionalidad y parámetros similares

Los atributos de calidad que impacta este patrón son los siguientes:

- Reutilización: se permite reutilizar acciones o funcionalidades de la arquitectura planteada a modo de capas , un ejemplo en este caso sería la implementación de funcionalidad de los pagos donde se conecta directamente con un proveedor, esta parte de la lógica podría ser desplegada en otro ambiente diferente al de la plataforma académica, de igual manera se puede abstraer la función atómica de cada capa que debe ser autosuficiente dependiendo únicamente de las capas inferiores
- Portabilidad, este patrón permite la reutilización de la funcionalidad de todas sus capas en conjunto desplegadas en otro ambiente

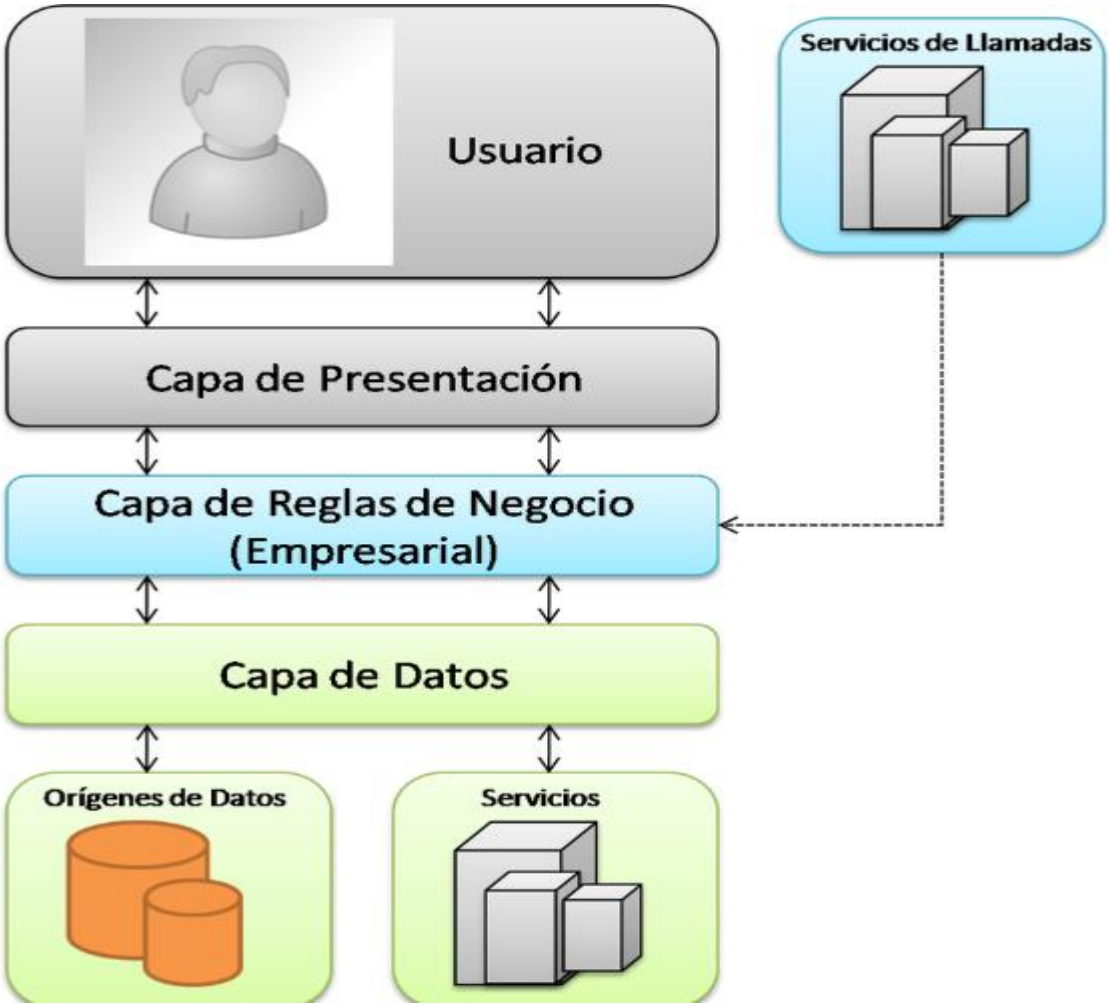


imagen obtenida de: https://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/

- Multinivel:

El modelo de arquitectura manejara patrones de arquitectura de tipo multinivel dividido en 3 niveles, el primero representando la parte de presentación y la parte de lógica que

tendrá la trayectoria de lógica y ejecución en cuanto a normativas del negocio , y la de datos que manejara la parte de persistencia dentro de las bases de datos

10.Bibliografia

1. Philippe Kruchten, Architectural Blueprints—The “4+1” ViewModel of Software Architecture. [En línea]. Available: https://www.researchgate.net/publication/220018231_The_41_View_Model_of_Architecture/download [Último acceso: 28 Abril 2019].

2. D. Alur y Deepak Alur, Core J2EE Patterns: Best Practices and Design Strategies (2nd Edition). Prentice Hall, 2003.
3. [2]
4. C. Nock y Clifton Nock, Data Access Patterns: Database Interactions in Object-Oriented Applications. Addison Wesley, 2003.
5. [3]
6. Simson L. Garfinkel y S. L. Garfinkel, Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable. 2005.
7. [4]
8. F. Marinescu y Floyd Marinescu, EJB Design Patterns: Advanced Patterns, Processes and Idioms. John Wiley & Sons, 2002.
9. [5]
10. D. Alur y Deepak Alur, Head First Design Patterns. O’Reilly Media, 2004.