**SOFTWARE ARCHITECTURE DOCUMENT**

**(SAD)**

Contents

[**1. Context** 1](#_Toc248926143)

[**2. Functional View** 1](#_Toc248926144)

[**3. Process View** 1](#_Toc248926145)

[**4. Non-functional View** 2](#_Toc248926146)

[**5. Architectural Constraints** 2](#_Toc248926147)

[**6. Architectural Principles** 3](#_Toc248926148)

[**7. Logical View** 3](#_Toc248926149)

[**8. Interface View** 3](#_Toc248926150)

[**9. Design View** 4](#_Toc248926151)

[**10. Infrastructure View** 4](#_Toc248926152)

[**11. Deployment View** 5](#_Toc248926153)

[**12. Operational View** 5](#_Toc248926154)

[**13. Security View** 6](#_Toc248926155)

[**14. Data View** 7](#_Toc248926156)

[**15. Technology Selection** 7](#_Toc248926157)

[**16. Architecture Justification** 7](#_Toc248926158)

**1. Context**

El siguiente documento presenta una visión general de la arquitectura de la aplicación, siendo representado a través de una serie de diferentes vistas arquitectónicas que involucran diferentes aspectos de la Aplicación Web.

Future service es un software que permite agilizar y optimizar el Sistema de pedidos en un restaurante, a la vez que implementa funcionalidades adicionales, que sirven para que tanto los empleados como los administradores, puedan tener una herramienta que agilice sus tareas.

**2. Functional View**

**Las principales funcionalidades de nuestro software son:**

1. Toma y notificación de pedidos.

Historia de usuario 1.1: Como mesero quiero tener un listado de productos para ofrecer a los clientes.

Historia de Usuario 1.2: Como despachador quiero ver todos los pedidos que están en cola.

Historia de usuario 1.3: como despachador quiero tener un mensaje al recibir un nuevo pedido.

Historia de usuario 1.3: como despachador quiero notificar al mesero que el producto ya está listo para entregar.

Historia de usuario 1.4: Como mesero quiero definir una mesa como ocupada al registrar un nuevo pedido.

Historia de usuario 1.5: Como mesero quiero agregar de a un producto al pedido.

Historia de usuario 1.6: Como mesero quiero eliminar un producto del pedido.

2. Agregar y consultar productos del inventario.

Historia de usuario 2.1: Como administrador quiero consultar los productos que hay en el inventario.

Historia de usuario 2.2: Como administrador quiero agregar al inventario nuevos productos.

3. Generar factura de cobro.

Historia de usuario 3.1: Como cajero quiero que se me muestre en pantalla el pedido de cada mesa.

Historia de usuario 3.2: Como cajero quiero generar una factura de cobro.

4. Consultar el reporte de ventas.

Historia de usuario 4.1: Como administrador quiero consultar los productos más vendidos.

Historia de Usuario 4.2: Como administrador quiero consultar el total de ventas.

5. Tener una base de datos de los clientes.

Historia de usuario 5.1: Como mesero quisiera poder agregar un nuevo cliente.

Historia de usuario 5.2: Como cliente quisiera acumular puntos de acuerdo al valor de la compra.

Historia de usuario 5.3: Como cliente quisiera poder redimir mis puntos a cambio de algún producto del menú u obtener descuentos.

Historia de usuario 5.4: Como cajero quiero consultar los puntos de un cliente cuando éste me lo solicita.

6. Manejo de la nómina.

Historia de usuario 6.1: Como administrador quiero crear un nuevo despachador.

Historia de usuario 6.2: Como administrador quiero crear un nuevo mesero.

Historia de Usuario 6.3: Como administrador quiero poder pagarle a la nómina.

Historia de Usuario 6.4: Como administrador quiero ingresar con una clave de acceso.

7. Generar una cola de canciones por reproducir a medida que cada cliente las va solicitando.

**4. Non-functional View**

Describe los requerimientos no funcionales del sistema dentro de las categorías usabilidad, escalabilidad, interoperabilidad y disponibilidad.

**Usabilidad:** El sitio web es simple, diseñada para que los usuarios después de una instrucción sencilla puedan utilizarlo con facilidad.

**Interoperabilidad:** Desde cualquier dispositivo con acceso a internet se puede acceder al sitio ya que está diseñado para visualizar de forma óptima todo el contenido

**Escalabilidad:** El Sistema está construido con un arquitectura de capas permitiendo la modificación y evolución de este sin tener que alterar demasiado el código

**5. Architectural Constraints**

En esta vista se presenta las restricciones normativas a la cuales está sujeto el proceso de desarrollo

**Estandares:** la interfaz de usuario debe estar orientada a la web,debe ser posible visualizar el contenido utilizando cualquiera de los navegadores ,google Chrome ,mozilla Firefox ,entre otros.

**Interfaz web:**

Web services: este sistema posee interoperabilidad ya que el administrador poseera una aplicación móvil en la cual podrá tener acceso a informacion

**Tecnología:**

El desarrollo completo del sistema deberá estar realizado en la plataforma eclipse usando el lenguaje de programación JAVA

El sistema funcionará con la base de datos que el equipo de trabajo diseñe durante el proceso de desarrollo.

El lenguaje de programación utilizada para el desarrollo del software en Java, en la IDE de Eclipse. El servidor de base de datos será MySql.

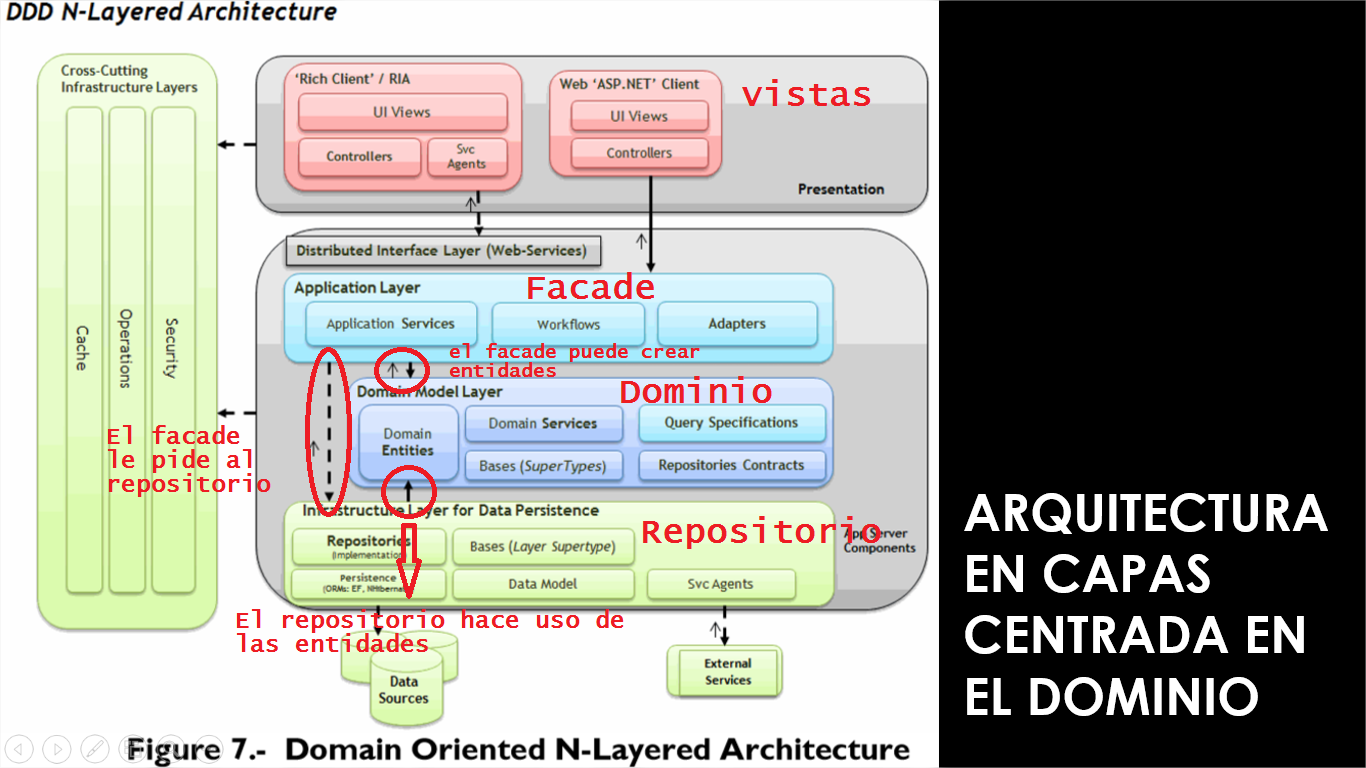
El sistema operativo donde estará corriendo la aplicación será en en Windows server.

Horario: La fecha de entrega final será el viernes 13 de noviembre y será una entrega total de todas las historias de usuario creadas inicialmente.

**6. Architectural Principles**

Desarrollamos nuestro principio de arquitectura en capas, reconocido como “layers”. De esta manera se busca llegar a una solución que se acerque a la visión de nuestro proyecto: La agilización y optimización de tiempos para los establecimientos. El método de capas nos permite maniobrar mucho más fácil los aspectos más importantes y sobresalientes de la aplicación. Así, en cualquier caso, sea editar, crear una solución o solucionar algún problema, se podrá hacer directamente sobre el problema relacionado, en este aspecto, encontrándolo específicamente en la capa que lo contiene.

**7. Logical View**

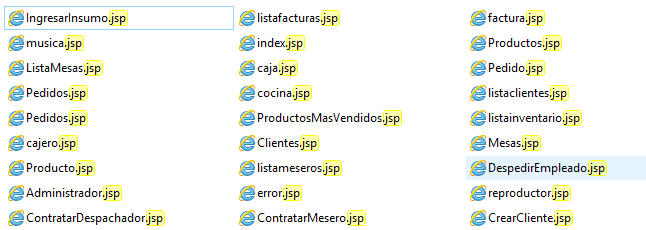
Se usó el modelo MVC más SOA; de esta manera se tienen 4 capas:

1. Vista
2. Servicio
3. Negocio
4. Datos

Como lo podemos observar en las siguientes ilustraciones.

**Capa Vista:**

Acá se encuentran todas la jsp que se encargan de enseñarle al usuario de una forma amigable todas las funciones requeridas por este. Nos podemos encontrar con jsp’s como la de ProductosMasVendidos.jsp que se encarga de enseñarle al cliente la lista de productos más vendidos en una tabla realizada por HTML5 y un Ajax implementado en JavaScript, ubicado en código.js. También podemos encontrar los controlladores encargados de gestionar los requerimientos del cliente.

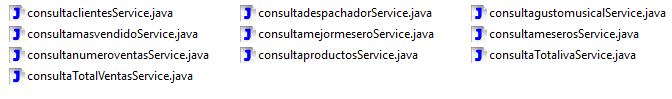


C:\Users\Crisman Carmona\Pictures\proyecto\codigo.PNG

C:\Users\Crisman Carmona\Pictures\proyecto\controllador.PNG

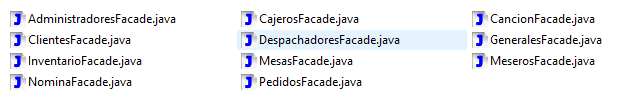
**Capa Servicios:**

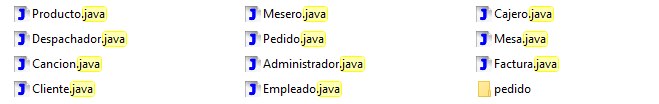
Acá encontramos los servicios que se colocan para el aprovechamiento de las unidades móviles, en este caso para Android. Por ejemplo, consultaclientesService.java, arroja un String con los nombres de todos los clientes, perfecto para el uso del adminitrador desde su Smartphone.



**Capa Negocio:**

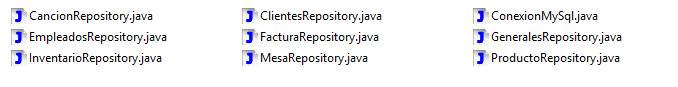
Acá se encuentran los orquestadores del negocio y el negocio en sí. Aquí está la lógica de nuestro negocio y las fórmulas que colocan a correr efectivamente nuestra aplicación y nos acercan a la visión.





**Capa Datos:**

Con esta capa recuperamos los datos necesarios para ‘jugar’ en nuestra aplicación, de esta manera, no habrían datos quedamos en la app y el usuario tendría los datos guardados en su propia base de datos.

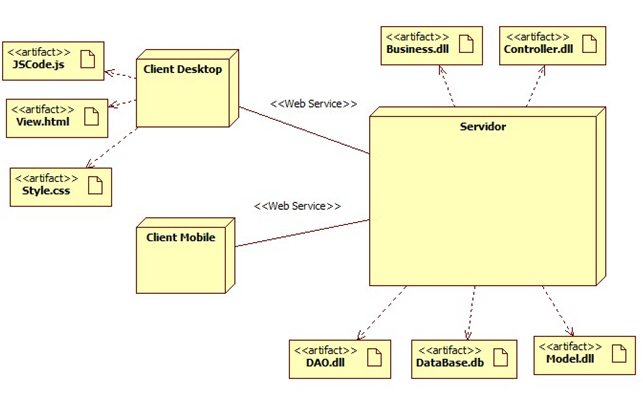


**11. Deployment View**

El software estará instalado en un servidor web, y se podrá acceder a él por medio de una conexión a internet, las peticiones al servidor se harán a través de un navegador tradicional desde las computadoras instaladas en el restaurante o cualquier dispositivo con acceso a internet, de igual modo se podrá acceder a partir de una aplicación móvil instalada en tabletas o teléfonos celulares.

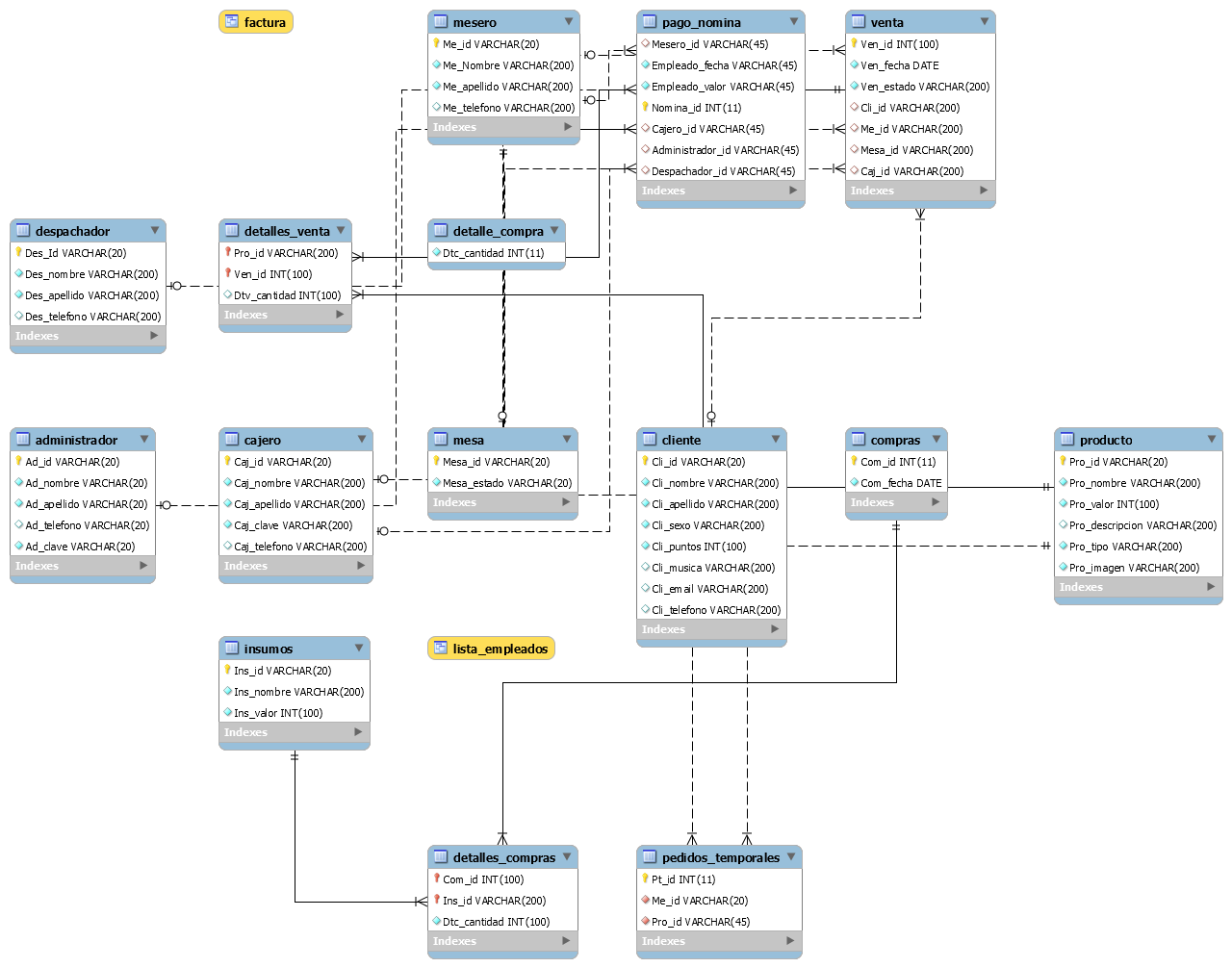
La presentación se hará por medio de una fachada, la cual no representa todo el programa, sólo permitirá la interacción entre los usuarios y el sistema.

Imagen tomada de internet.



**14. Data View**

El siguiente es el modelo de bases de datos generado en Mysql 5.7.



**15. Technology Selection**

Se utilizan diversas tecnologías para mejorar la calidad y la funcionalidad del software, entre ellas están:

**Estilos Css:** Los cuales nos permiten tener hojas con estilos con elementos muy agradables y amables con el usuario, tales como colores, fondos, imágenes, márgenes, bordes, tipos de letra, eventos etc.

**JavaScript:** Éste lenguaje de programación se ejecuta del lado del cliente, es utilizado sobre todo para revisar formularios, mostrar mensajes de alerta y calcular sumatoria de precios.

**Ajax:** Es utilizada para hacer peticiones al servidor, en la cual se solicitan datos, pero no interfiere en el del comportamiento total de la página, nos valemos de ello para la funcionalidad de ciertos botones, listados como productos, facturas, clientes mesas. También es utilizado para enviar datos a los controladores.

**16. Architecture Justification**

Se garantizará que la aplicación tendrá una disponibilidad total, pues estará alojada en la web y se podrá acceder a ella en cualquier instante.

Por la arquitectura diseñada para el desarrollo del software, nos permite inferir que el programa permite un constante crecimiento, cada vez que se requiera un avance para mejorar e implementar nuevas características se puede fácilmente lograr lo que se necesita.

A partir de un diseño pensado en las necesidades reales del cliente, Future service puede ser un software que cualquier persona después de una instrucción mínima pueda utilizar eficazmente el programa.

A través de los servicios rest, se le otorga al programa la facilidad de ser operado desde teléfonos móviles, por una aplicación que se puede instalar en éstos últimos.

**Referencia**

http://www.codingthearchitecture.com/pages/book/software-architecture-document-guidelines.html