

Universidad del Valle de Guatemala
Análisis de Sistemas
Prof. Lynette

Proyecto: Optimización de Restaurante - Corte 3

Rafael Antonio León Pineda - 13361
Pablo Ignacio Arriola Diaz - 131115
Andres Alejandro Oliva Murga - 12149
Pablo José López Aguilar - 14509

II. Resumen

El proyecto a realizar surgió a base de la necesidad de Hotel y Restaurante La Casona para mejorar su atención a los clientes y reducir costos de operación del mismo. En la actualidad el restaurante realiza su funcionamiento de forma muy tradicional, todo se hace a mano y no se tiene ningún inventario cuantificable. Debido a los cambiantes tiempos, un negocio que no se adapta a las mejoras tecnológicas, está condenado a tener dificultades competitivas contra cadenas de restaurantes y otros negocios grandes. Los objetivos de este proyecto son: Encontrar la manera de crear un inventario estándar para el funcionamiento del restaurante, hacer la toma de pedidos más certera y eficaz. Además se busca mejorar la atención al cliente y hacer que el servicio al mismo sea más ameno y ágil.

Modelación del Negocio:

Estaremos trabajando para hotel y restaurante La Casona, una entidad privada que presta servicios de hotelería y restaurante, se especializa en el área de restaurante, ya que es lo que queremos mejorar, el restaurante cuenta con 3 cocineros, y 6 meseros, 20 mesas, los procesos más importantes son la toma de órdenes, la preparación de la orden, el cobro y pago, además del inventario *semanal o mensual*.

Objetivos generales:

- Facilitar al mesero la toma del pedido
- Facilitar al cliente la realización del pedido
- Facilitar al cocinero la preparación del pedido

Objetivos específicos:

- Facilitar el control de inventario

Alcance del proyecto:

- Este proyecto va encaminado a la optimización en el área de meseros junto con la cocina ya que se pretende que el mesero tome las órdenes de una forma más fácil y ordenada, esta orden se desplegará en cocina facilitando al cocinero saber con exactitud la orden sin necesidad que el mesero le entregue la boleta de pedido directamente, esto minimiza el tiempo de pedido y preparación, además que al estar ligado la orden con la cocina, se realiza un sistema de inventariado en base a las órdenes de cada día.
- Se puede hacer notar que dentro de los problemas existentes:
 - a. El mesero tome mal la orden ya que esta es dictada y el mesero debe escribir, esto se elimina ya que nuestro sistema contiene los menús hechos, y si el cliente desea algo extra a su plato o un cambio del mismo, se tendrán check boxes para estos mismos, además de una nota extra por plato dentro del pedido.

[illegible]

Pantalla de carrito de tablet de mesero/tablet en mesa

Menu	Lista del pedido :			HORA
Desayunos	1)	Nombre del platillo	Valor del platillo	+ Cantidad -
Almuerzo	2)	Nombre del platillo	Valor del platillo	+ Cantidad -
Refacciones	3)	Nombre del platillo	Valor del platillo	+ Cantidad -
Cena				
Bebidas				
Carrito				
Log-out	Enviar			Total: Monto total.

Pantalla de tablet de administrador para manejar eventos(reservaciones fiestas)

Menu	Buscar Evento			HORA
Personal				
Inventario				
Calendario				
	1	2	3	
	Descripcion del evento	Descripcion del evento	Descripcion del evento	
	4	5	6	
	Descripcion del evento	Descripcion del evento	Descripcion del evento	
	7	8	9	
	Descripcion del evento	Descripcion del evento	Descripcion del evento	
Log-out	← MES →	← Año →		

Evidencias:



Evidencias: Entrevistas

<https://youtu.be/Daw4LIWz11A>

Observaciones:

Meseros:

- Los Menú necesitan imágenes.
- No se pueden discontinuar los menú en papel si solamente los meseros van a tener tablets.
- “Carrito” no es una palabra apropiada, debería ser menú
- Interfaz fácil para las especificaciones de cada pedido, a modo de agilizar la toma de órdenes y evitar equivocaciones.
- Permitir la edición de pedidos en el carrito.
- Agregar tab para pedidos extras.

Administrador:

- Se debería de poder crear eventos desde la tablet.
- La revisión del personal debería estar en la computadora central también.
- La facturación digital debe ser precisa y debe ser notorio que se está solicitando la cuenta de una mesa. Una X no es suficiente.
- Cada mesero debería tener un control de sus mesas en su tablet también, en caso de que existan pedidos adicionales después del pedido inicial, para poder facturar todo como solo una cuenta.
- El inventario debe ser automático pero permitir ingresos de cosas nuevas, ya que existen gastos que no se pueden justificar automáticamente porque ocurren muy de vez en cuando.

Prototipo 2:

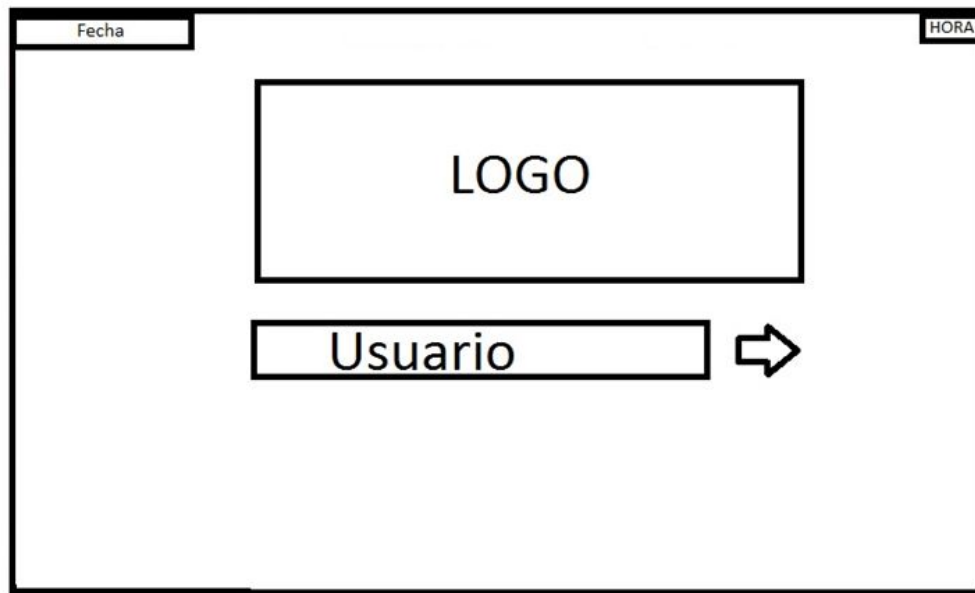
Tablet de administrador con opción para editar eventos.

Menu	Buscar Evento	Nuevo	Eliminar	Editar	HORA
Inventario	1 Descripcion del evento	2 Descripcion del evento	3 Descripcion del evento		
Calendario	4 Descripcion del evento	5 Descripcion del evento	6 Descripcion del evento		
Lista de caja	7 Descripcion del evento	8 Descripcion del evento	9 Descripcion del evento		
	← MES →	← Año →			

Órdenes con imágenes para cada platillo

Nombre del plato [Image Placeholder] Descripcion	Nombre del plato [Image Placeholder] Descripcion	Nombre del plato [Image Placeholder] Descripcion	HORA
Nombre del plato [Image Placeholder] Descripcion	Nombre del plato [Image Placeholder] Descripcion	Nombre del plato [Image Placeholder] Descripcion	

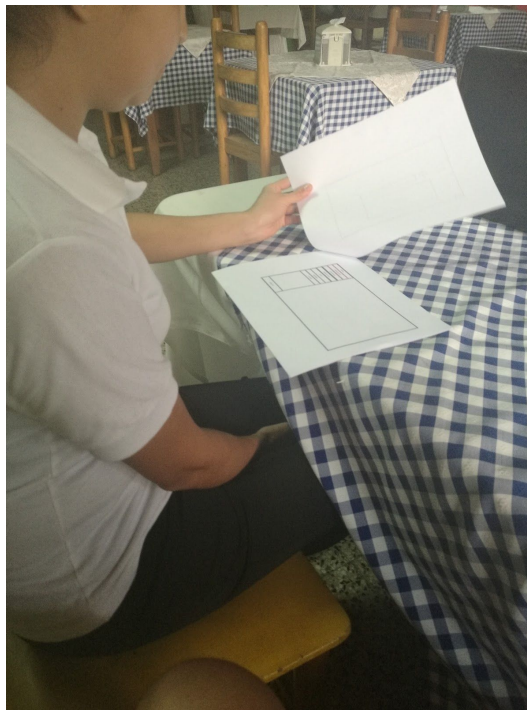
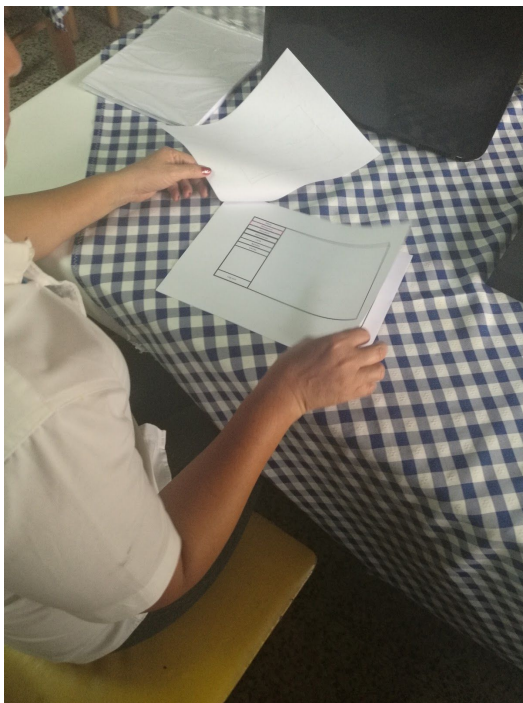
Login para cada usuario



A diagram of a login form within a rectangular frame. At the top left, there is a small box labeled 'Fecha'. At the top right, there is a small box labeled 'HORA'. In the center, there is a large rectangular box labeled 'LOGO'. Below the 'LOGO' box, there is a horizontal box labeled 'Usuario' followed by a right-pointing arrow.

Evidencias:

<https://youtu.be/Daw4LIWz11A>



Observaciones:

Meseros:

- Los meseros deberían de tener contraseña propia también
- Debería de haber un sistema similar al de la cocina para las bebidas en cada tablet, en caso de que a alguien se le olvide, otro mesero más desocupado puede ayudar.
- Los meseros deberían de tener más control sobre su app.

Administradores:

- Debería existir una opción para imprimir una lista de compras en lugar de solo tenerlo en caja y en la tablet.
- Los meseros deberían de tener un sistema simple en donde no se puedan equivocar fácilmente, considerando que muchos de los meseros no tienen educación más allá de estudios básicos.

Cambios a los Requisitos Funcionales:

1. CRUD Inventario (Caja)
2. Calcular Inventario automático (Caja)
3. Mostrar Compras a realizar (Caja, Admin)
4. Manejar Reservas de eventos.(Admin, Caja)
5. Impresión de facturas. (Caja)
6. Tomar pedidos (Mesero)
7. Mostrar menu con imagenes (Mesero)
8. CRUD Calendario (Caja, Admin)
9. Mostrar pedidos en cocina(Caja)
10. Reportar cuándo una mesa quiere cancelar el pedido(mesero y caja)
11. Editar pedidos existentes(Mesero y caja)
12. Anular pedidos (Caja)
13. Mostrar estadísticas del Mes (Caja, Admin)
14. Mostrar cola de bebidas (Mesero)
15. CRUD de personal (Caja, Admin)
16. Mostrar Personal Activo (Admin)

Diagrama de Clases General:

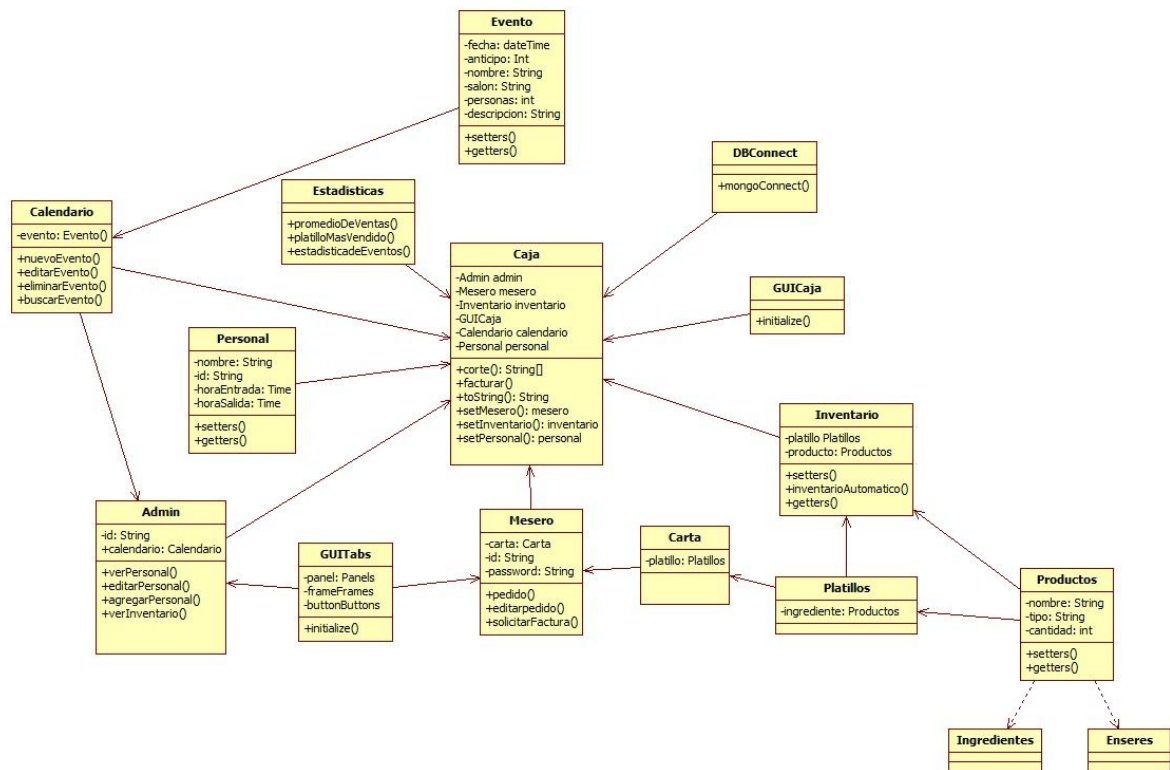
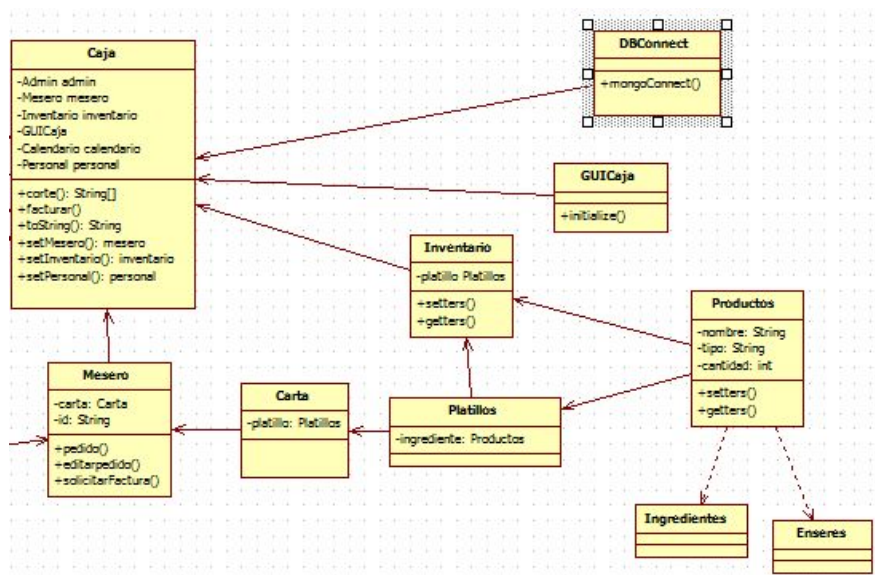
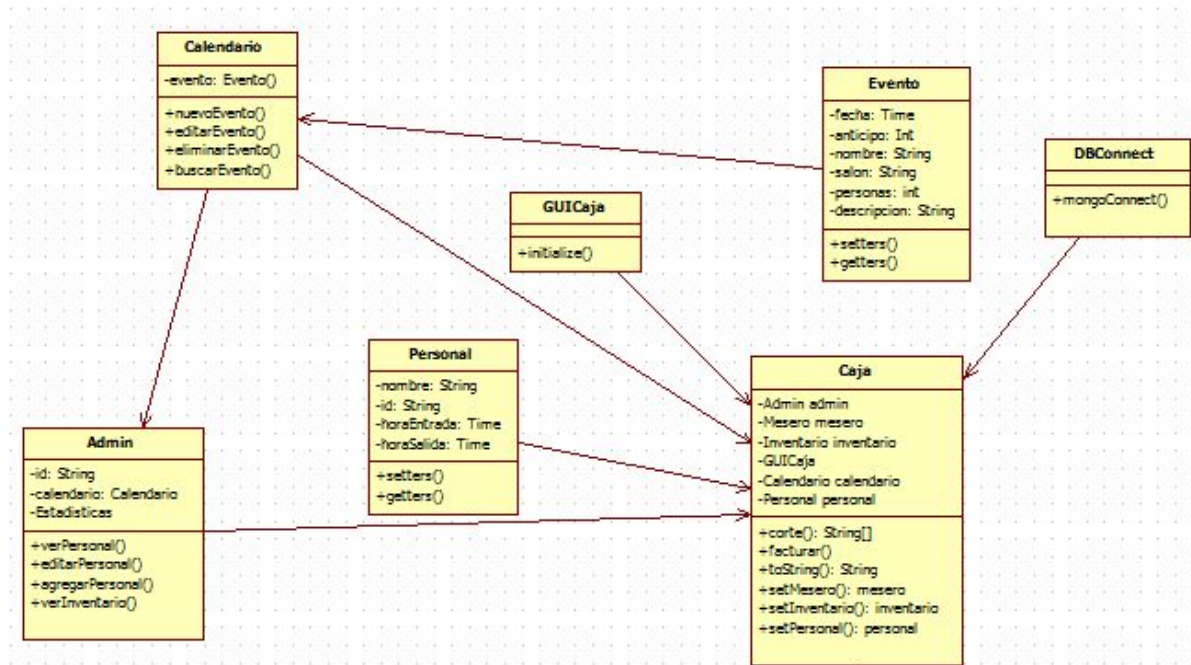


Diagrama de Clases para Casos de Uso Específicos:

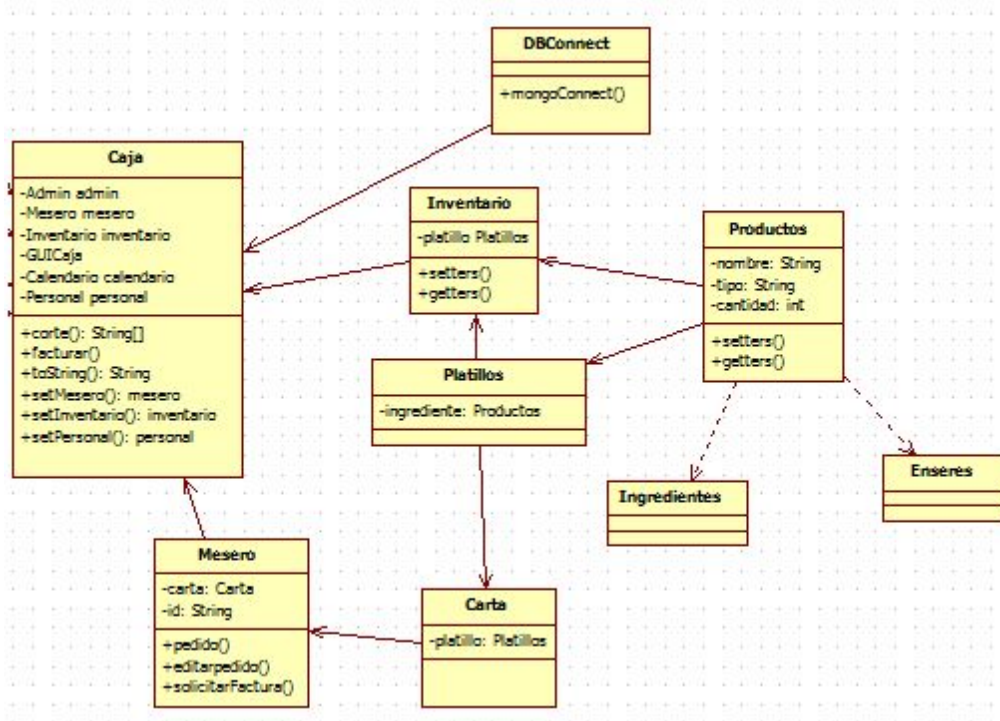
Toma de Ordenes:



Calendario



Inventario:



Descripcion de Clases

Caja:

Clase principal del programa que realiza todos los cálculos del sistema y se conecta con cada una de las tablets y la base de datos.

Atributos:

Admin: (admin) Objeto tipo administrador que maneja los paquetes enviados por la app de administrador.

Mesero: (mesero) Objeto tipo mesero que maneja los paquetes enviados por las tablets de los meseros y de las tablets de las mesas.

Inventario: (inventario) Objeto tipo inventario encargado de manejar el CRUD de inventarios y los cálculos de inventario automatizado

GUICaja: Interfaz gráfica para la interacción del administrador con la clase Caja

Calendario: (calendario) Objeto tipo calendario que maneja el CRUD del calendario de eventos.

Personal: (personal) Objeto tipo personal que maneja el CRUD de personal y los paquetes de Personal enviados por la tablet de admin.

DBConnect: connexion con la base de datos.

Métodos:

Corte: (string): Método que realiza los cálculos del corte diario de caja.

Facturar(): void Método que imprime la factura de la orden.

ReporteInventario(): Devuelve el reporte del inventario

Cocina(): Método que maneja los outputs e inputs de cocina

toString(): (string): Metodo que imprime la información de la caja

Setters y Getters

Calendario

Objeto tipo calendario que maneja el CRUD del calendario de eventos.

Atributos:

Evento: (evento): Objeto con las características generales de reservación de un evento del restaurante.

Métodos:

nuevoEvento: Crea un nuevo evento y lo envia a la base de datos

editarEvento: Cambia un evento existente y lo guarda en la base de datos

eliminarEvento: Elimina uno de los eventos

buscarEvento(): Busca uno de los eventos en la base de datos y mostrar sus datos

Evento

Objeto con las características generales de reservación de un evento del restaurante.

Atributos:

Fecha: (DateTime) DateTime formatted input que determina la fecha y hora del evento

Anticipo: (int) Cantidad del total que pagaron anticipadamente para reservar el evento

Nombre: (String) Nombre de la persona que solicita el evento

Salon: (String) Nombre del salón donde se realizará el evento

Personas:(int) Cantidad de personas que participaran en el evento

Descripción: (String): Descripcion del evento, platillo a utilizar

Métodos:

Setters

Getters

ToString: (String)

Admin

Objeto tipo administrador que maneja los paquetes enviados por la app de administrador.

Atributos:

Id: String: Identificador del administrador

Calendario: Calendario: Objeto tipo calendario con las características editables

Estadísticas: Objeto tipo estadística que permite ver las estadísticas de ventas

Métodos:

verPersonal(): permite observar el personal activo

editarPersonal(): permite editar el personal del restaurante

agregarPersonal(): permite agregar personal a la aplicación

verInventario(): permite ver la lista de compras del inventario

Personal

Objeto que contiene los atributos que definen a cada uno de los trabajadores del restaurante

Atributos:

Nombre: String Identificador del nombre del trabajador

Id: String: Identificador del administrador

horaEntrada: dateTime: Horario de ingreso de cada uno de los trabajadores

horaSalida: dateTime: Horario de salida de los trabajadores

Métodos:

setters()
getters()
toString(): String

Estadísticas:

Método que realiza los cálculos estadísticos del restaurante.

Atributos:

Métodos:

promedioDeVentas(): Muestra el promedio de ventas mensuales
platilloMasVendido(): Muestra el platillo más vendido y las estadísticas del mismo
estadisticadeEventos(): Muestra una estadística de los eventos que se realizaron en el mes

Mesero:

Mesero del restaurante que envía los paquetes de órdenes a la caja

Atributos:

Carta: carta: tiene la información del menú del restaurante con todos los platillos
Id: identificador de cada mesero
Password: password de cada mesero

Métodos:

Pedido(): Realiza el paquete de pedido que envía el paquete de pedido a caja
EditarPedido(): Envía un paquete que reemplaza el pedido en caja
SolicitarFactura(): Envía el paquete que indica en caja que una mesa requiere factura

Carta

Objeto que contiene cada uno de los platillos del menú

Atributos:

-Platillo:Platillos: Contiene cada uno de los platillos del menú

Platillos:

Atributos:

-Ingrediente: Productos muestra los ingredientes que compone cada uno de los platillos

Inventario:

Contiene cada uno de los platillos y productos del inventario del restaurante

Atributos:

Platillo: Platillos: Tiene todos los platillos del restaurante y sus características

Productos: Productos: Tiene todos los productos de la caja

Métodos:

nuevoElemento: Agrega un nuevo elemento al inventario

editarElemento: Cambia un elemento existente de la base

buscarElemento: Busca un elemento en la base de datos

eliminarElemento: elimina un elemento de la base de datos

inventarioAutomatico: Hace los cambios en el inventario automático, comparando los ingredientes con cada uno de los platillos que se solicitaron.

Productos

Atributos:

nombre:(String): Contiene el nombre del producto

tipo: (String): Indica el tipo del producto

cantidad:(int) muestra la cantidad del objeto.

Métodos:

setters()

getters()

Diagrama de paquetes:

Diagrama de paquetes 1:

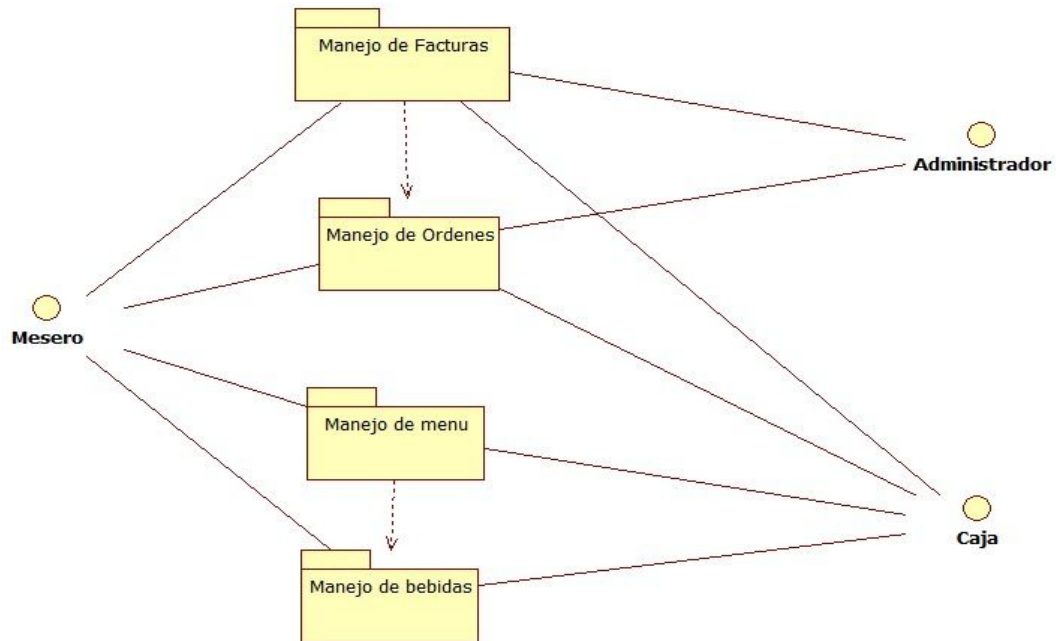
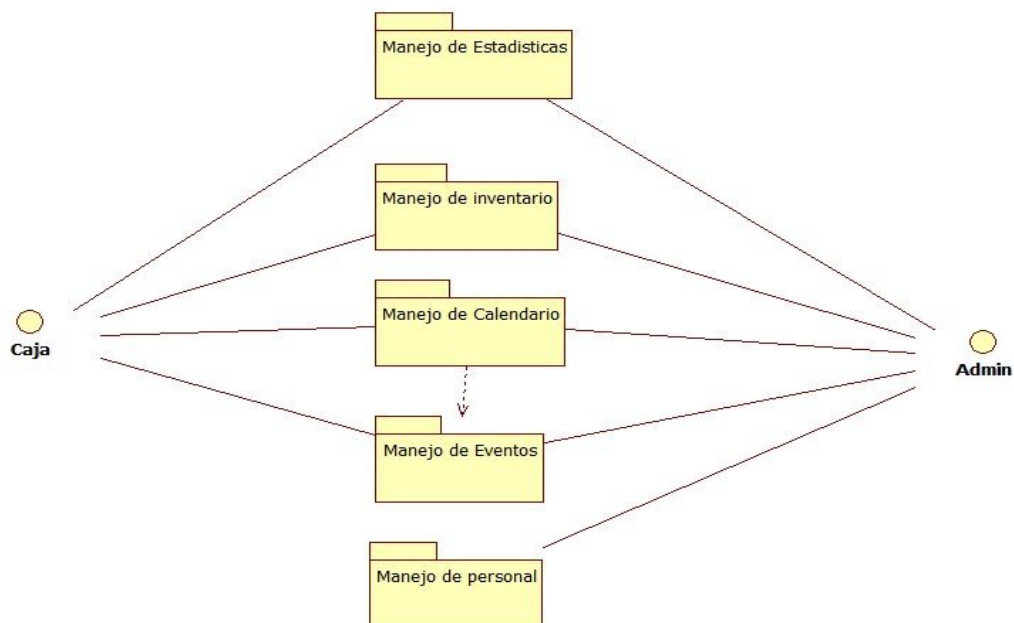


Diagrama de paquetes 2:



Descripción de cada paquete y sus componentes:

Paquetes diagrama 1:

- Manejo de facturas: maneja la impresión de facturas y el pedido de facturas
 - Impresión de facturas: imprime la factura cuando el cliente lo pide
 - Requerimiento de facturas: cuando el cliente se lo pide al mesero
- Manejo de órdenes: contiene todos los procesos del manejo de las órdenes
 - Pedido de orden: cuando el cliente o el mesero hace una orden
 - Modificación del orden: cuando el cliente o el mesero hace una orden
 - Anulación de orden: cuando el mesero o el cliente anulan la orden
- Manejo de menú: contiene todos los procesos que usan el menú
 - Dar el menú al cliente: cuando el mesero da el menú al cliente sin tablet
 - Ver menú: se muestra el menú en la tablet
 - Modificación del menú: modificación por parte de caja
- Manejo de bebidas: todos los procesos que llevan las bebidas
 - Pedir bebida: se pide la bebida desde la tablet
 - Mostrar cola de bebidas: se muestran las bebidas pendientes

Paquetes diagrama 2:

- Manejo de estadísticas: lleva todos los procesos de las estadísticas
 - Promedio: saca el promedio de las ventas en el periodo estimado
 - Platillo más vendido: nos da el platillo más vendido
 - Estadísticas de eventos: dice como fue en promedio en los eventos
- Manejo de inventario: todos los procesos del inventario
 - Productos: nos da los datos de los productos, en comparación con platillo
 - Platillos: nos da los datos de los platillos, en comparación con productos
- Manejo de calendario: tiene todos los eventos presentes, futuros y pasados
 - Ver: muestra todo el calendario
- Manejo de eventos: todos los datos de un evento
 - Crear: el cajero y el admin crean eventos
 - Modificar: el cajero y el admin pueden modificar el evento
 - Eliminar: el cajero y el admin pueden eliminar el evento
- Manejo personal :caja y admin manejan el personal
 - Agregar: pueden agregar un nuevo empleado
 - Despedir: pueden despedir un empleado

Diseño:

Cantidad de usuarios	15			
Tiempo promedio de respuesta	inmediata			
Cantidad promedio usuarios simultáneos.	10			
Escalabilidad				
	Especificación	Cantidad	ubicación	Estimación de Tamaño
	Imagen de cada ítem del menú y su descripción	100	Servidor	Entre 30 y 60 Mb (cada una)
	Aplicación móvil	1	Tablet	Entre 40 a 100 MB
	Gestionador de aplicaciones (aplicación completa)	1	servidor	Entre 1 a 2 GB
	Total	Entre 4.040 a 8.100 GB		
Disponibilidad de la información	100% disponible			
confiabilidad	Alta			
Tolerancia a fallos	Media			
Replicación de información	nula.			

Selección de Tecnologías

Se realizó una investigación de 4 estilos de arquitectura: (Client/Server, Componentes, capas y orientada a objetos).

Arquitectura cliente servidor

Modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor quien le da respuesta. esta idea se puede aplicar a programas que se ejecutan en una sola computadora.

La capacidad de proceso está repartida entre los clientes y los servidores.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un solo programa.

Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo.

El servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La red cliente-servidor es una red de comunicaciones en la cual los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta y que los pone a disposición de los clientes cada vez que estos los soliciten.

Características:

El remitente de una solicitud es conocido como cliente:

- es quien inicia solicitudes o peticiones.
- espera y recibe las respuestas del servidor.
- puede conectarse a varios servidores a la vez.
- interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

El receptor de la solicitud se conoce como servidor:

- al iniciarse esperan a que lleguen las solicitudes de los clientes.
- tras la recepción de la solicitud, la procesan y luego envían la respuesta al cliente.
- Acepta las conexiones de un gran número de clientes.
- La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y sus protocolos.

Patrón de arquitectura basada en Componentes:

Se concentra en la descomposición del diseño en componentes individuales que constan de interfaces de comunicación bien definidas que contienen métodos, eventos y propiedades. Permite un nivel de abstracción mayor al de programación orientada a objetos porque no se enfoca en protocolos de comunicación y estado compartido.

Un componente de software es un paquete de software, servicio web o módulo que encapsula funciones relacionadas.

Todos los procesos del sistema se colocan en componentes de modo que todos los datos y funciones dentro de cada componente están relacionados semánticamente. Se describen los componentes como elementos modulares y cohesivos.

Arquitectura por capas

Es un estilo de arquitectura Cliente/Servidor , cuyo objetivo principal es separar la lógica de negocios con la lógica de diseño. Su mayor ventaja es que la programación se lleva a cabo en diferentes capas, lo que significa que cuando se necesite realizar un cambio en el programa solo se tiene que buscar y modificar en una capa específica y no en un programa con todo mezclado.

El diseño más utilizado en la actualidad consiste de 3 capas (Presentación, negocios y datos).

Capa de presentación: Es la capa que presenta el sistema al usuario, le presenta toda la información necesaria y captura los datos ingresados por el usuario. Esta capa debe ser amigable con el usuario.

Capa de negocios: Es la capa en donde residen los programas a ejecutarse, tiene comunicación tanto con la capa de presentación para recibir todas las solicitudes y para enviar los resultados, y con la capa de datos para solicitar los mismos al momento de ejecutar un programa y actualizarlos al momento de ser necesario.

Capa de datos: Es la capa en donde residen los datos y se encarga de acceder a los mismos.

Estilo de Arquitectura Orientada a Objetos

Es una arquitectura que se basa en la división de responsabilidades de un sistema, en objetos reutilizables y autosuficientes. Cada uno con datos y propiedades específicas para la realización de su responsabilidad. Este estilo mira el sistema no como una lista de

procedimientos sino como un conjunto de objetos cooperativos. Estos objetos se comunican a través de interfaces y métodos que acceden a las propiedades de cada objeto. Los principios claves de éste estilo son :

- Abstracto
- Composición
- Herencia
- Encapsulamiento
- Polimorfismo
- Modulares

Luego de haber realizado esta investigación se llegó a la conclusión que se utilizarán los estilos (Client/Server y Orientado a objetos) debido a que sus características se adecúan más a las necesidades que tenemos.

Cliente/Servidor:

Se piensa utilizar Client/Server Architectural style ya que se piensa tener un único servidor en caja, que sea el encargado de tener la información, de costos e inventarios, y diferentes aplicaciones (programas) para cada usuario, así los usuarios podrán mandar solicitudes o peticiones sin la necesidad de buscar entre los diferentes programas (Menú, Caja y Cocina). El otro beneficio de utilizar este modelo sería ahorrar costos, ya que al tener en caja el servidor los otros dispositivos no tendrían que tener las más altas especificaciones.

Orientado a Objetos:

Se piensa utilizar object oriented para acceder a los servicios de forma independiente dependiendo de qué objeto se dirija al servidor. Esto permitirá reutilizar software en cada una de las tablets y diferenciarlas entre ellas modificando solamente sus atributos.

Al mismo tiempo, manejar un inventario con objetos es más simple ya que se puede designar un objeto base para los platillos y modificar sus atributos para cada plato específico.

Bases de datos

Mongo

Sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto, MongoDB en lugar de guardar los datos en tablas como se hacen la

base de datos relacionales, mongoDB guarda estructura de datos en documentos similares a JSON con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

SQL

Oracle Database:

Sistema de gestión de base de datos tipo objeto-relacional, considerado como uno de los sistemas más completos, destacado en soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma.

Cloudant

Es una base de datos de como servicio (DBaaS) el cual permite centrarse en el desarrollo rápido de aplicaciones en internet y aplicaciones móviles en lugar de preocuparse por la expansión y gestión de la base de datos por su cuenta. El almacén de dato se construye para la escalabilidad y está optimizada para lecturas y escritura de datos simultáneas, maneja una amplia variedad de tipos de datos estructurados y no estructurados entre ellos JSON, textos completos y geoespacial.

Para este proyecto usaremos Mongo, ya que:

- Guarda estructuras de datos en documentos tipo Json con un esquema dinámico.
- Útil para el registro de eventos.
- Fácil implementación en aplicaciones móviles.
- Almacén de datos operacional de una página web
- Soporta la búsqueda por campos, consulta rangos y expresiones regulares.
- Cualquier campo en un documento de MongoDB puede ser indexado.
- puede escalar horizontalmente usando shard.
- Puede ser usado com sistema de archivos.
- Es gratis.